OMAP730
Technical Reference Manual

Literature Number: SWPU063B
September 2003

OMAP™
TEXAS INSTRUMENTS TECHNOLOGY

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Preface

Read This First

About This Manual

This technical reference manual provides technical information on the OMAP730 multimedia processor.

How to Use This Manual

This document contains the following chapters:

Chapter 1: Introduction to the OMAP730 System

This chapter introduces the setup, components, and features of the OMAP730 multimedia processor and provides a high-level view of the device architecture.

Chapter 2: MPU Subsystem

This chapter presents a description of the OMAP730 MPU subsystem

Chapter 3: GSM Subsystem

This chapter discusses the OMAP730 GSM-S device.

Chapter 4: Traffic Controller Interface

This chapter discusses the traffic controller interface (TCIF) module.

Chapter 5: Clock Generation and Reset Module

This chapter discusses the clock generation and reset module (CLKM), which is part of the microprocessor unit (MPU) subsystem in the OMAP730 hardware engine platform.

Chapter 6: MPU-S Interrupt Handler

This chapter discusses the MPU interrupt handler of the OMAP730 hardware engine.

Chapter 7: MPU-S Direct-Memory Access

This chapter describes the direct memory access (DMA) controller for the OMAP730 hardware engine.

Chapter 8: LCD Controller

This chapter discusses the LCD controller module for the OMAP730 hardware engine.

Chapter 9: UART Modem/IrDA

This chapter provides programmers with a functional presentation of the universal asynchronous receiver/transmitter (UART) infrared data association (IrDA) module. It includes a register description and a module configuration example. It also shows the basic UART module pins.

Chapter 10: Universal Serial Bus

This chapter describes the universal serial bus (USB) host on the OMAP730 multimedia processor.

Chapter 11: Multimedia Card (MMC/SD/SDIO) Interface

This chapter describes the multimedia card (MMC) interface of the OMAP730 multimedia processor.

Chapter 12: McBSPs

This chapter describes the three multichannel buffered serial ports (McBSPs) available on the OMAP730 device.

Chapter 13: NAND Flash

This chapter describes the NAND flash memory interface of the OMAP730 multimedia processor.

Chapter 14: MPU-S I^2C Serial Interface

This chapter describes the I^2C serial interface of the OMAP730 multimedia processor.

Chapter 15: Enhanced Audio Controller (EAC)

This chapter describes the universal serial bus (USB) host on the OMAP730 multimedia processor.

Chapter 16: Multichannel Serial Interface (MCSI)

This chapter describes the multichannel serial interface for the OMAP730 multimedia processor.

Chapter 17: Power and Control Clock

This chapter describes the power and control clock (PCC) module, outlines its architecture, and provides hardware and software information to designers.

Chapter 18: VLYNQ Serial Communications Interface

This chapter discusses the VLYNQ serial communications interface.

Chapter 19: Security Features

This chapter describes the security features of the OMAP730 multimedia processor.

Chapter 20: Smart Card Controller

This chapter discusses the OMAP730 SmartCard controller (SMC).

Chapter 21: Dual-Mode Timer

This chapter discusses the dual-mode timer of the OMAP730 multimedia processor.

Chapter 22: Camera Interface

This chapter describes two camera interfaces implemented in the OMAP730 multimedia processor.

Chapter 23: Real-Time Clock

This chapter discusses the real-time clock of the OMAP730 multimedia processor.

Chapter 24: Memory Mapping

This chapter describes the shared memory and memory mapping for the MPU-S and the GSM-S.

Chapter 25: Interrupt Mapping

This chapter describes MPU-S and GSM-S interrupt mapping.

Chapter 26: DMA Requests

This chapter discusses the DMA interrupt requests.

Appendix A: MPU-S Registers

This appendix lists the registers of the OMAP730 multimedia processor.

Appendix B: GSM Subsystem Registers

This appendix describes the inputs/outputs of the OMAP730 multimedia processor.

Appendix C: Pin Descriptions

This appendix describes the error code correction (ECC) algorithm for the OMAP730 multimedia processor.

Appendix D: Packaging

This appendix provides the OMAP730 GZG packaging configuration.

Appendix E: Peripherals Revision Number

This appendix defines acronyms and important terms used in this manual.

Notational Conventions

This document uses the following conventions.

- Program listings, program examples, and interactive displays are shown in a `special typeface` similar to a typewriter's. Examples use a **`bold version`** of the special typeface for emphasis; interactive displays use a **`bold version`** of the special typeface to distinguish commands that you enter from items that the system displays (such as prompts, command output, error messages, etc.).

  Here is a sample program listing:

```
0011  0005  0001          .field    1, 2
0012  0005  0003          .field    3, 4
0013  0005  0006          .field    6, 3
0014  0006                .even
```

  Here is an example of a system prompt and a command that you might enter:

```
C:  csr –a /user/ti/simuboard/utilities
```

- In syntax descriptions, the instruction, command, or directive is in a **bold typeface** font and parameters are in an *italic typeface*. Portions of a syntax that are in **bold** should be entered as shown; portions of a syntax that are in *italics* describe the type of information that should be entered. Here is an example of a directive syntax:

  .asect "*section name***",** *address*

  .asect is the directive. This directive has two parameters, indicated by *section name* and *address*. When you use .asect, the first parameter must be an actual section name, enclosed in double quotes; the second parameter must be an address.

- Square brackets (**[** and **]**) identify an optional parameter. If you use an optional parameter, you specify the information within the brackets; you don't enter the brackets themselves. Here's an example of an instruction that has an optional parameter:

  LALK *16–bit constant [, shift]*

  The LALK instruction has two parameters. The first parameter, *16-bit constant*, is required. The second parameter, *shift*, is optional. As this syntax shows, if you use the optional second parameter, you must precede it with a comma.

  Square brackets are also used as part of the pathname specification for VMS pathnames; in this case, the brackets are actually part of the pathname (they are not optional).

- Braces ({ and }) indicate a list. The symbol **|** (read as *or*) separates items within the list. Here's an example of a list:

```
{ * | *+ | *– }
```

  This provides three choices: `*`, `*+`, or `*–`.

Unless the list is enclosed in square brackets, you must choose one item from the list.

- Some directives can have a varying number of parameters. For example, the .byte directive can have up to 100 parameters. The syntax for this directive is:

  .byte *value$_1$ [, ... , value$_n$]*

  This syntax shows that .byte must have at least one value parameter, but you have the option of supplying additional value parameters, separated by commas.

Information About Cautions and Warnings

This book may contain cautions and warnings.

This is an example of a caution statement.

A caution statement describes a situation that could potentially damage your software or equipment.

This is an example of a warning statement.

A warning statement describes a situation that could potentially cause harm to you.

The information in a caution or a warning is provided for your protection. Please read each caution and warning carefully.

FCC Warning

This equipment is intended for use in a laboratory test environment only. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

Trademarks

OMAP, TMS320C54x, TMS320C55x, C54x, and C55x are trademarks of Texas Instruments Incorporated.

1-Wire is a registered trademark of Dallas Semiconductor.

All other trademarks are the property of their respective owners.

If You Need Assistance. . .

If you want to. . .	Do this. . .
Request more information about Texas Instruments Digital Signal Processing (DSP) products	Call the CRC† hotline: **(800) 336–5236** Or write to: Texas Instruments Incorporated Market Communications Manager, MS 736 P.O. Box 1443 Houston, Texas 77251–1443
Order Texas Instruments documentation	Call the CRC† hotline: **(800) 336–5236**
Ask questions about product operation or report suspected problems	Call the DSP hotline: **(713) 274–2320**
Report mistakes in this document or any other TI documentation	Fill out and return the reader response card at the end of this book, or send your comments to: Texas Instruments Incorporated Technical Publications Manager, MS 702 P.O. Box 1443 Houston, Texas 77251–1443

† Texas Instruments Customer Response Center

Contents

Figures

Tables

Examples

Chapter 1

Introduction to the OMAP730 System

This chapter describes the OMAP730 system architecture, presents the main system features, and discusses system security.

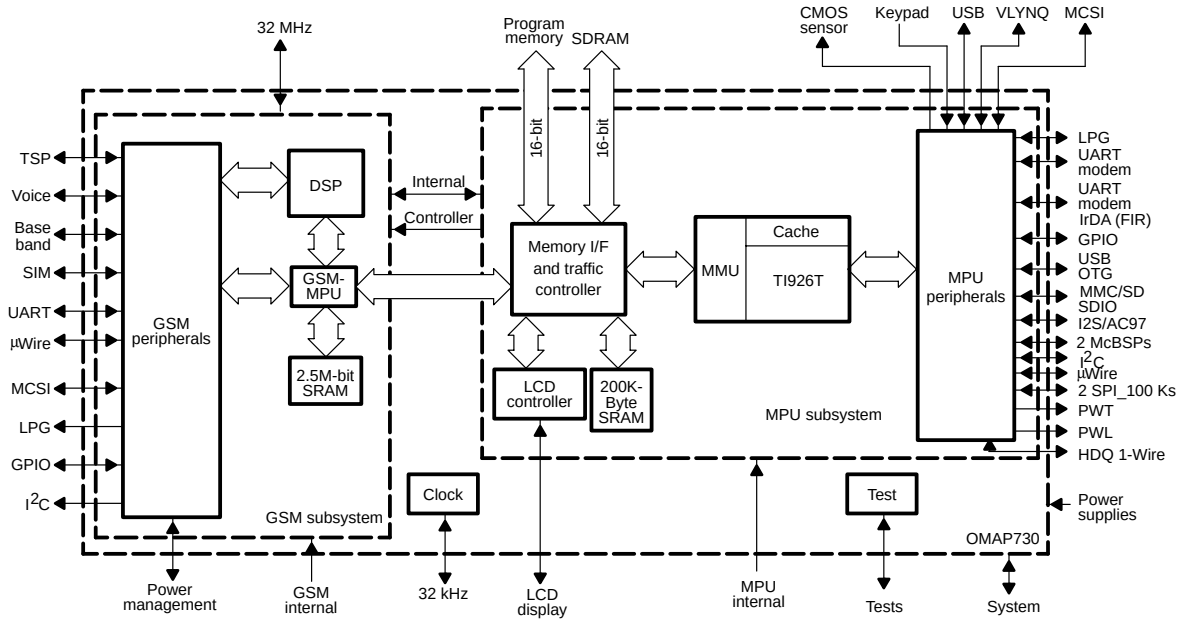
Topic **Page**

1.1 Device Description

The OMAP730 is the Texas Instruments solution for wireless pocket information devices such as wireless PDA, smart phones, Java-enabled web phones, and other wireless handsets that combine voice and data. The OMAP730 is designed to run with Windows CE, EPOC, Palm, Linux, and other operating systems.

The OMAP730 consists of two main subsystems that share external memories with the help of the memory and traffic controller. The GSM subsystem (GSM-S) handles the complete GSM protocol stack and signal processing. The MPU subsystem (MPU-S) runs the OS-controlling application tasks and all non-GSM peripherals.

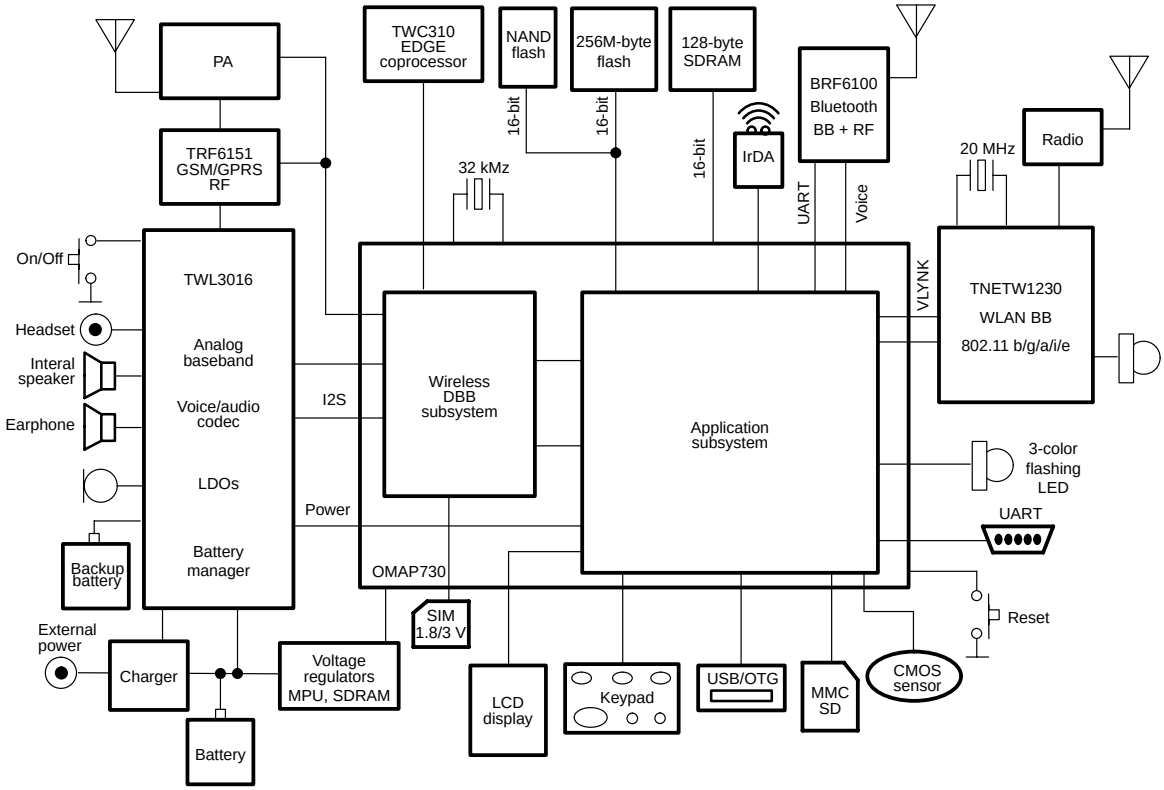
Figure 1–1. OMAP730 Block Diagram Overview



The GSM-S external peripherals include the RF module, the TWL3016 or ABB analog-baseband/audio/D-D/A and battery manager chip, the touch screen, a serial link, and a SIM card interface. The MPU-S controls peripherals such as display, keyboard/keypad, IrDA, and other serial links. Other peripherals such as flash card or GPS can also be connected to the system.

The OMAP730 contains a set of secure modules, including ROM, a single port SRAM, and eFuse cells. These components enable the system to support secure applications.

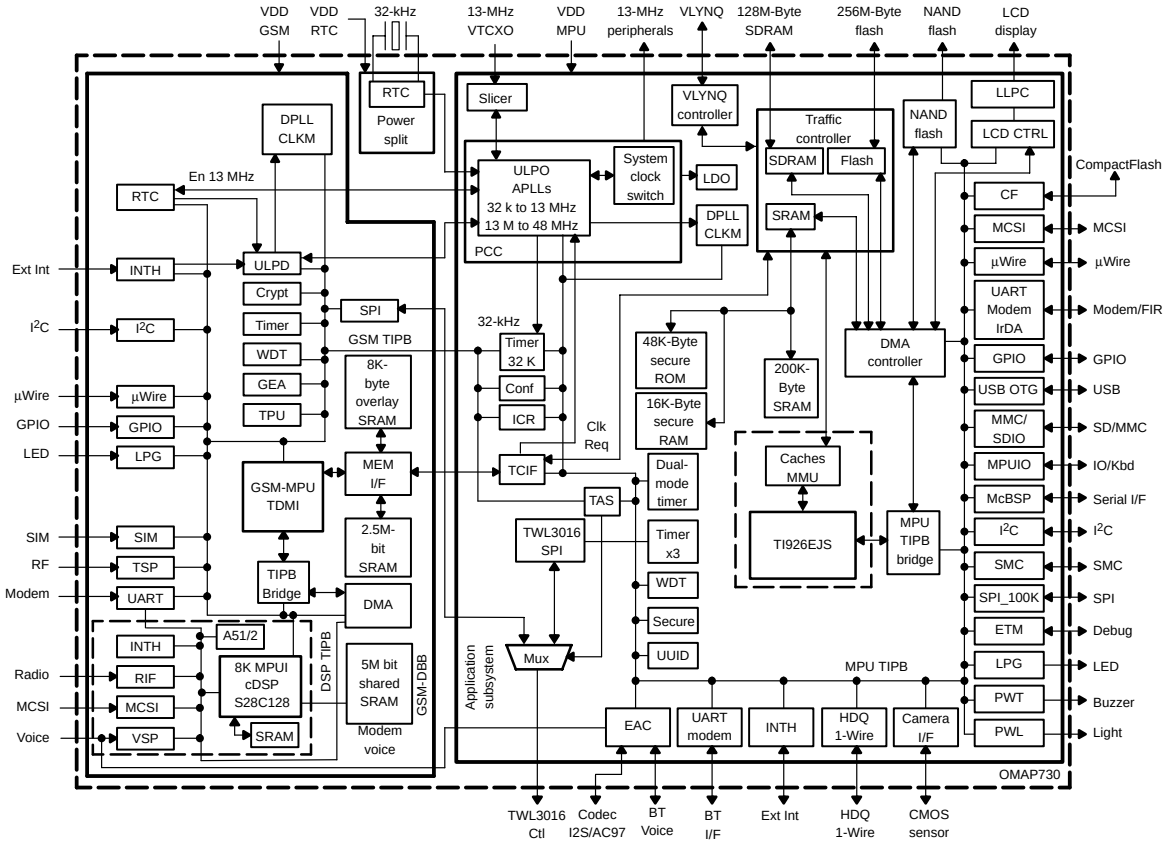
Figure 1–2. OMAP730 Wireless-PDA Application Overview



1.1.1 OMAP730 Architecture

The OMAP730 is based on the OMAP 3.2 MPU core and the TBB2100 GSM baseband subsystem (GSM-MPU + TMS320C54x™ DSP).

Figure 1–3. OMAP730 Block Diagram



1.1.1.1 GSM Subsystem Description (GSM-S)

The GSM subsystem implements the digital baseband processes of a GSM/GPRS mobile phone. This subsystem combines a DSP subchip (TMS320C54x DSP CPU) with its program and data memories, a microcontroller core with emulation facilities (GSM-MPUDMIE), internal 8K bytes of overlay boot SRAM memory, up to 2.5M bits of SRAM memory, .5M bits of SRAM memory sharable between DSP and GSM-MPU, a clock squarer cell, and several compiled single-port or two-port RAM and CMOS gates.

This subsystem is used in the management of the GSM/GPRS baseband processes through the GSM layer 1, 2, and 3 protocols, as described in the ETSI standard with specific attention to power consumption in both GSM dedicated and idle modes and GPRS (class 12) capability.

The GSM subsystem has capability for future enhancement such as EDGE coprocessor connectivity.

The GSM subsystem fully supports the GSM full-level test approval (FTA) for both full-rate, enhanced full-rate, and half-rate speech coding. It implements

all features for structural test of the logic (full scan, BIST, PMT, JTAG boundary scan).

1.1.1.2 MPU Subsystem Description (MPU-S)

The MPU-S performs all personal communication system tasks such as call manager, email/fax reader/composer, internet access, personal digital assistant (PDA) and personal information management (PIM). The MPU-S also controls the GSM subsystem.

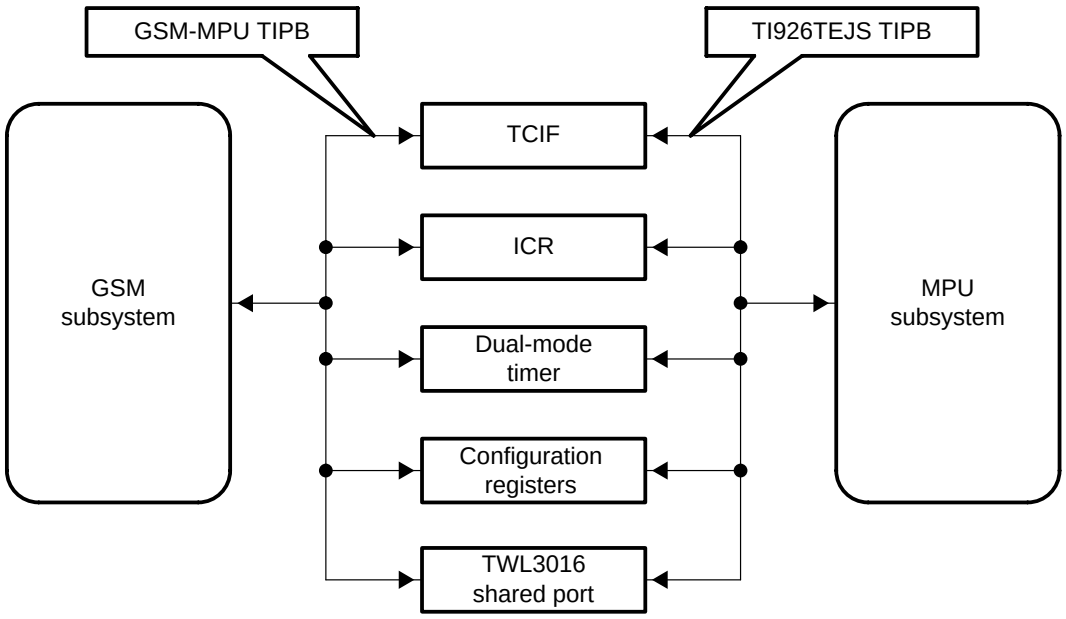
The MPU-S comprises an OMAP3.2 MPU subsystem plus some MPU peripherals.

1.1.1.3 OMAP730 System Peripherals

In addition to the GSM and the MPU subsystems, some modules are used for GSM-S/MPU-S interworking management:

- ❏ Timer running on 32-kHz clock (Timer32K)
- ❏ Traffic controller interface (TCIF)
- ❏ Intersystem control register (ICR)
- ❏ Configuration registers
- ❏ TWL3016 shared port

Figure 1–4. OMAP730 Intersystem Peripherals



1.1.1.4 Other OMAP730 Processor Peripherals

- JTAG interface module (JTAG)
- Test and debug blocks
- eFuse modules
- Boot ROM and security

1.2 Features

This section describes the features of the subsystem modules.

1.2.1 GSM-MPU Module

The following GSM-MPU modules are used on GSM applications:

- ❑ GSM-MPUTDMIE megamodule
  - ■ GSM-MPUTDMI CPU core (32/16-bit RISC processor)
  - ■ GSM-MPU ICECrusher™ for emulation purposes
- ❑ GSM-MPU memory interface for embedded SRAM and external MPU subsystem shared flash and SDRAM access management (through TCIF)
- ❑ 2.5M-bit static RAM with write buffer
- ❑ .5M-bit static RAM sharable with DSP with mutual exclusive access
- ❑ 64K-bit static RAM with external memory overlay for internal boot
- ❑ Memory protection unit (MPU)
- ❑ Debug unit (DU)
- ❑ Die-ID cell read access
- ❑ TIPB bridge
- ❑ DMA controller (4 channels, 2 ports)
- ❑ 4K-byte TMS320C54x DSP-GSM-MPU shared memory (MPUI)
- ❑ GSM-MPU interrupt handler (INTH)
- ❑ Watchdog timer (WDT)
- ❑ Two generic timers
- ❑ GSM real-time sequencer (TPU)
- ❑ GSM real-time serial port (TSP)
- ❑ SIM interface (SIM)
- ❑ GSM-MPU serial port interface (SPI)
- ❑ UART 16C750 modem with hardware flow control (DCD, CTS/RTS) and autobaud capability shared with DSP
- ❑ Real-time clock (RTC)
- ❑ GSM ultralow-power-down controller (ULPD)
- ❑ Clock generator and control with DPLL (CLKM)
- ❑ Programmable controller for three-color LED pulse generation (LPG)

- ❑ Master I^2C serial interface
- ❑ Microwire interface
- ❑ GPRS encryption coprocessor (GEA1&2)

1.2.2 DSP Subchip

The GSM subsystem has the following ASIC DSP features:

- ❑ TMS320C54x DSP module (S28C128)
  - ■ TMS320C54x DSP core
  - ■ 28K words of embedded RAM
  - ■ 128K words of ROM
  - ■ .5M-bit static RAM sharable with MPU with mutual exclusive access
  - ■ MPUI (8K words, part of the 28K-word RAM)
  - ■ SPI
  - ■ Timer
- ❑ TIPB bridge
- ❑ Radio interface (RIF)
- ❑ Multichannel serial interface (MCSI)
- ❑ A51/A52 ciphering (crypt)
- ❑ UART 16C750 modem with hardware flow control (DCD, CTS/RTS) and autobaud capability shared with MPU
- ❑ DMA controller (four channels)
- ❑ TMS320C54x DSP interrupt handler (INTH)

1.2.3 MPU Module

The MPU-S comprises an OMAP3.2 MPU subsystem plus MPU peripherals.

The OMAP3.2 subsystem contains:

- ❑ ARM926EJ MPU:
  - ■ ARM926EJ megacell including:
    - ▪ ARM926EJS core, running at 200 MHz maximum frequency
    - ▪ MMU with translation lookaside buffer (TLBx)
    - ▪ L1 16K-byte, four-way set-associative instruction cache
    - ▪ L1 8K-byte, four-way set-associative data cache with write buffer
  - ■ MPU level 1 interrupt handler
  - ■ Coprocessor15 (CP15) and protection module

- 17-word write buffer (WB)
- System bus interface

- Memory and traffic controller (TC)
  - 16-bit data width memory interface for 128M bytes of addressable SDRAM
  - 16-bit data width memory interface for 256M bytes of addressable flash/RAM/ROM
- Color LCD controller: 2/4/8/16 and pseudo 18 bits/pixel
- MPU interrupt handler: 32 lines (INTH)
- MPU TIPB bridge: 32-bit
- Three 32-bit timers
- Watchdog timer
- Clock generator with DPLLs and power management
- 200K bytes of SRAM with frame buffer DMA channel
- System DMA controller
- Embedded trace macrocell module, ETM version 2.a in 13-bit mode configuration or in 17-bit demultiplexed mode configuration

The MPU-S subsystem contains:

- OMAP3.2 data-processing core
- Power and clock control (PCC) with the following main functions:
  - Performs the transitions between the power modes (awake, big sleep, deep sleep)
  - Handles idle/wake-up handshake of MPU-S and DBB
  - Monitors wake-up events
  - Controls system clock input sources (VCTXO, 32-kHz oscillator)
  - Performs calibration of 32-kHz oscillator (gauging)
  - Manages the clocks and resets distributed to MPU-S, DBB, and to some peripherals
  - Manages security resets and violations
  - Handles power-up sequence
  - Manages the 32-kHz oscillator-to-VCTXO, and VCTXO-to-32-kHz oscillator clock switch
  - Handles embedded LDO with bypass possibility.
  - Implements full PMT wrapper for all analog cells embedded inside it.

- Two UART modems—SIR/MIR/FIR IrDA with the following main features:
  - Selectable UART/IrDA modes
  - Dual 64-entry FIFOs for received and transmitted data payload
  - Programmable and selectable transmit and receive FIFO trigger levels for DMA and interrupt generation
  - Programmable sleep mode
  - Complete status-reporting capabilities in both normal and sleep modes
  - Frequency prescaler values from 0 to 16383 to generate the appropriate baud rates
  - Single 48-MHz clock reference for baud setting
  - Two DMA requests and one interrupt request to system
  - UART/modem functions:
    - Baud rate from 300 bits/s up to 3.6864M bits/s
    - Autobaud between 1200 bits/s and 115.2K bits/s
    - Software/hardware flow control:
      - Programmable Xon/Xoff characters
      - Programmable auto-$\overline{\text{RTS}}$ and auto-$\overline{\text{CTS}}$
    - Programmable serial interface characteristics:
      - 5-, 6-, 7- or 8-bit characters
      - Even, odd, or no parity bit generation and detection
      - 1-, 1.5-, or 2-stop bit generation
    - False-start bit detection
    - Line-break generation and detection
    - Fully-prioritized interrupt system controls
    - Internal test and loopback capabilities
    - Modem control functions ($\overline{\text{CTS}}$, $\overline{\text{RTS}}$, $\overline{\text{DSR}}$, $\overline{\text{DTR}}$, $\overline{\text{RI}}$, and $\overline{\text{DCD}}$)
  - IrDA functions:
    - Slow infrared (SIR: 115.2K baud), medium infrared (MIR: 0.576M baud), and fast infrared (FIR: 4M baud) operations (very fast infrared (VFIR) not supported).
    - The width of the pulse can be either 1.6 μs or 3/16th of a single-bit time.
    - Framing error, cyclic redundancy check (CRC) error, illegal symbol (FIR), abort pattern (SIR, MIR) detection.
    - Eight-entry status FIFO (with selectable trigger levels) available to monitor frame length and frame errors.

- General-purpose I/Os (GPIO) with interrupt support: Maskable interrupt generation on high-to-low or low-to-high transition of pins configured as input.

Note:

The peripheral GPIO modules have ARMPER_CK as their input clock. These GPIOs are asynchronous, and the modules do not perform input line debouncing. These GPIOs, then, can wake up the system on an input level change when ARMPER_CK is off.

- USB OTG controller with 48-MHz APLL. The On-The-Go (OTG) supplement allows a USB peripheral to have the following enhancements:
  - Limited host capability to communicate with other selected USB peripherals
  - A small USB connector to fit the mobile form factor
  - Low-power features to preserve battery life
- Multichannel buffered serial port (McBSP)
  - Based on multichannel buffered serial port TI standard
  - Simultaneous RX and TX DMA support
  - Each processor master of its TX clock and TX data
  - Supports bit rates up to 5M bits/sec
  - RX data overrun interrupt
  - Functionally identical to C5510 McBSP
- Enhanced audio controller (EAC): Supports I2S, AC97, and SPDIF codecs. The enhanced audio controller (EAC) provides an application with stand-alone audio control without any CPU (DSP or MPU) support. It allows connecting a modem and/or Bluetooth subsystem to an AC97, a PCM, or an I2S codec while the MPU subsystem (PDA, WinCE MCU, etc.) is in power-down mode. It also suppresses the need for two codecs (one for the modem and one for the MPU subsystem).

  The EAC also provides the ability to record/play PCM (wave) files with various sample frequencies and bit-length formats without any CPU processing. It also allows using the same microphone input line for the modem 8-kHz sampling frequency or for the high-quality voice recording.

  A high-quality audio file also can be mixed with the voice input channel, down-sampled, and sent to the modem/Bluetooth uplink path (the wave file plays during a phone call). The EAC also provides the CPU the ability to get the data samples coming from the microphone path at the codec sampling frequency, to process the signal, and to send it back to the modem uplink path. The audio signal coming from the modem/Bluetooth downlink path can also be processed by the CPU at 8 kHz or at the codec sampling frequency before being sent to the codec and loudspeakers.

  Modem/Bluetooth inputs and/or output voice data samples can be automatically stored in the CPU memory through the DMA write channel (record the phone call).

The EAC also provides the ability to loop back the Bluetooth audio data samples to the modem uplink path.

- Multimedia memory card and secure digital I/O host controller (MMC/SD and 4-bit SDIO), supports the following combination of external devices:
  - One or more MMC memory cards sharing the same bus plus up to four devices with 8-bit SPI protocol interface (serial flash memories, etc.)
  - One single SD memory card or SDIO card plus up to four devices with 8-bit SPI protocol interface

  The main features of the multimedia memory card and secure digital I/O host controller are:
  - Full compliance with MMC command/response sets as defined in MMC standard specification v.2.2 [1] (SPI mode extension not supported)
  - Full compliance with SD command/response sets as defined in SD Physical Layer specification v.1.0 [2] (SPI mode extension not supported)
  - Flexible architecture allowing support for new command structure
  - Separate SPI interface with four C/S. Provides supports for up to four serial devices such as serial flash
  - Built-in 64-byte FIFO for buffered read or write
  - 16-bit-wide access bus to maximize bus throughput
  - Low-power design
  - Wide interrupt capability
  - Programmable clock generation
  - Two DMA channels
- CompactFlash interface: CompactFlash is a very small removable mass storage device. It provides complete PCMCIA-ATA functionality.
- NAND flash controller: The NAND-type flash memory chip contributes to the rapid write and erase capabilities since data is rewritten in small increments, leading to improved performance through continuous data recording and other benefits.
- Fast I^2C master/slave controller: External components attached to the I^2C bus can serially transmit/receive up to 8-bit data to/from the local host (LH) device through the two-wire I^2C interface (400K bits/s). This I^2C peripheral supports any slave or master I^2C-compatible device.
- SMC (SmartCard) controller with direct I/O interface: The key to secure network access, it provides secure authentication of the user to the network, enabling the delivery of personalized services as well as providing data storage for address books. Extending this concept, the SmartCard (SMC) application connects any 3G terminal to any 3G service.
- µWire controller: This serial synchronous interface can drive four serial external components (EEPROM or LCD with µWIRE standard). For the ex-

ternal devices, this interface is compatible with the µWire standard and is seen as the master.

- Two chip-selects: Each has a configurable level and can accept a ready signal from an external device.
- The serial clock is derived from the reference 13-MHz clock: The serial clock period is derived from the reference 13-MHz clock and can be configured as

$T_{SCLK} = CK\_FREQ \times Csi\_FRQ \times T_{13M} = [2/4/7/10] \times [2/4/8] \times T_{13M.}$

The serial-clock polarity can be selected. Auto-CS toggle-transmit and DMA-TX modes are supported.

❑ HDQ and 1-Wire master controller: The HDQ/1-Wire battery monitoring serial interface module implements the hardware protocol of the master function of the Benchmarq HDQ and the Dallas Semiconductor 1-Wire® protocol. The module works off a command structure that is programmed into transmit-command registers. The received data is in the receive data register. The firmware is responsible for doing the correct sequencing in the command registers. The module only implements the hardware interface layer of the protocols.

❑ Camera interface: Defined to support DSC function (preview + picture). Video acquisition/encoding is not considered in the OMAP730 product definition.

There are two ways to support a camera on OMAP730:

1) Using an external component that performs sensor acquisition plus compression (JPEG, for instance) connected to OMAP730 using a standard serial link such as I^2C, µWire, USB, or SPI (all with DMA support).

2) Using a CMOS sensor connected directly to OMAP730 using a parallel interface (OMAP1510-like) as shown in Table 1–1.

Table 1–1. Preview Mode Versus Picture Mode

Preview Mode Resolution	Picture Mode Resolution
Up to QVGA (320, 240) Frame target rate: more than 15 fps	Up to VGA (640 x 480)

An 8-bit CMOS camera sensor with scaling capability is supported on OMAP730 parallel-camera IF function. The sensor must support SRGB data output format (YUV cannot be used for preview).

The following scheme is used:

- SRGB → RGB translation for preview: Performed by MPU
- Image scaling: Performed by the CMOS sensor
- Image windowing: Performed by DMA or MPU (on-the-fly during SRGB → RGB conversion)

Since part of the preview function is performed by software using the MPU (SRGB → RGB conversion), the CPU cost is estimated to be less than 50 MHz for a QVGA size picture preview at 25 fps (worst case).

- Pulse-width tone modulator (PWT): This module generates a modulated frequency signal for the external buzzer. The frequency is programmable between 349 Hz and 5276 Hz with 12 half-tone frequencies per octave. The volume is also programmable.
- Pulse-width light modulator (PWL): This module allows control of the backlight of the LCD and the keypad by employing a 4096-bit random sequence. This voltage-level-control technique decreases the spectral power at the modulator harmonic frequencies. The block uses a switchable 32-kHz clock that is independent of UPS.
- Serial port interface (SPI_100K): The serial port interface is a bidirectional three-line interface dedicated to the transfer of data to and from external devices offering a three-line serial interface.
- LCD controller: The LCD controller operates only in single-panel mode (dual-panel mode is not supported in this version). The panel size is programmable and can have any width (line length) from 16 to 1024 pixels in 16-pixel increments. The number of lines is set by programming the total number of pixels in the LCD. The total video-frame size is programmable up to 1024x1024.

  The main features of the LCD controller are:

  - Encoded pixel data stored in external memory in a frame buffer in 1-, 2-, 4-, 8-, 12- or 16-bit increments and loaded into the LCD DMA 64-entry FIFO (16 bits per entry).
  - Programmable pixel display modes
  - Programmable display size
  - 16 grayscale levels
  - Palette allowing full logical-to-physical address mapping
  - Programmable pixel rate
  - Support for four types of displays: Passive and active color and passive and active monochrome
  - A total of 3375 possible colors available in passive STN mode, allowing the display of any 16, 256, or 3375 colors in each frame, as well as 15 grayscale levels for monochrome screens
  - Support for any screen size up to 1024x1024 (assuming enough bandwidth is available)
  - Frame, line, and pixel clocks
  - ac-bias drive signal
  - 4-, 8-, 12-, 16-, and pseudo 18-bit-per-pixel display modes
  - Patented dithering algorithm
- LCD low-power controller (LLPC): The LLPC is a module between OMAP LCD controller and the external pins. It allows stopping of some signals, such as pixel clock and data lines for a period of time, thus reducing power dissipation on these lines and on the LCD panel itself.

- VLYNQ interface: This serial communications interface enables the extension of an internal CBA bus segment to one or more external physical devices. The VLYNQ accomplishes this function by serializing bus transactions in one device, transferring the serialized transaction between devices via a VLYNQ port, and deserializing the transaction in the external device.
- VLYNQ ID = 0x000E for OMAP730

Main features:

- Low pin count (as few as three signals)
- No 3-state signals
  - All signals are dedicated and driven by one single device
  - Provides significant reduction in I/O timing analysis complexity
  - Required to support high speed PHYs
- Scalable performance/support for different PHY technologies:
  - TTL @ 150 MHz and 1, 2, 4, or 8 bits for TX and RX data
  - LVDS @ 622 MHz and 1, 2, 4, or 8 bits for TX and RX data
  - Gandalf @ 3.5 GHz and one bit for TX and RX data
  - Linear increase in performance with any given PHY as the data-port width is increased
- Simple packet-based transfer protocol for memory mapped access:
  - Write-request/data packet
  - Read-request packet
  - Read-response data packet
  - Interrupt-request packet
- Symmetric operation:
  - TX pins on first device connect to RX pins on second device and vice versa.
  - Request packets, response packets, and flow control information are all multiplexed and sent across the same physical pins.
  - Supports both host/peripheral and peer-to-peer communication models
  - Able to emulate all currently-used peripheral interface mechanisms
- Simple block-code packet formatting (8b/10b)
- Supports in-band flow control:
  - No extra pins needed
  - Allows the receiver to momentarily throttle back the transmitter when overflow is about to occur

- Uses special built-in code capability of block code to seamlessly interleave flow control information with user data
- Allows system designers to balance cost of data buffering versus performance

- Supports multiple outstanding transactions
- Automatic-packet formatting optimizations
- Internal loopback mode

❑ Dual-mode timer

This programmable-interval 32-bit timer is required to generate a periodic interrupt, also called system clock tick, to OS. This is used to keep track of the current time and control the operation of device drivers.

The dual-mode timer main features are:

- Counter timer with compare and capture modes
- Autoreload mode
- Start-stop mode
- Programmable divider clock source
- 16- to 32-bit addressing
- On-the-fly read/write registers
- Interrupts generated on overflow, compare, and capture
- Interrupt enable
- Wake-up enable
- Write posted mode
- Dedicated input trigger for capture mode and dedicated output trigger/ PWM signal
- OCP-interface compatible

❑ SHA1/MD5 accelerator: The SHA1/MD5 security module provides hardware-accelerated hash functions. It can run either the SHA-1 algorithm in compliance with FIPS 180-1 standard or the MD5 message-digest algorithm developed by Rivest in 1991. Up to 2^{20}–1 bytes (1M byte) of data can be hashed in a single operation to produce a 160-bit signature in the case of SHA-1. or 128-bit signature in the case of MD5.

❑ DES/3DES: The DES/3DES module provides hardware-accelerated data encryption/decryption functions. It can run either the single DES algorithm or the triple DES algorithm in compliance with FIPS 46-3 standard. It supports ECB (electronic codebook) and CBC (cipher block chaining) modes of operation. It does not support the CFB (cipher feedback) or the OFB (output feedback) modes of operation in hardware.

❑ Random number generator (RNG): The RNG module provides a true, nondeterministic noise source for the purpose of generating keys, initiali-

zation vectors (IVs), and other random-number requirements. It is designed for FIPS 140-1 compliance, facilitating system certification to this security standard. It also includes built-in self-test (BIST) logic that allows testing the randomness of the module output and its compliance with FIPS 140-1 standard. An ANSI X9.17 Annex C postprocessor is available to meet the NIST requirements of FIPS 140-1.

1.2.4 Shared Module

In addition to the GSM and the MPU subsystems, some modules are used for GSM-S/MPU-S interworking management:

- Timer running on 32-kHz clock (Timer32K): This programmable interval 24-bit timer is required to generate a periodic interrupt, also called system clock tick, to OS. This is used to keep track of the current time and control the operation of device drivers.
- Traffic controller interface (TCIF): The TCIF module allows the GSM-S memory interface to access OMAP730 external memory through the MPU-S traffic controller.
- Intersystem control register (ICR): The intersystem control register module is a symmetrical interface between the GSM and MPU subsystems which allows them to exchange synchronization flags. It also allows defining configuration values used by other modules (the advantage of defining them in this module is to make then accessible to both MPU and GSM subsystems).
- Configuration registers
- TWL3016 shared port

1.3 Architecture

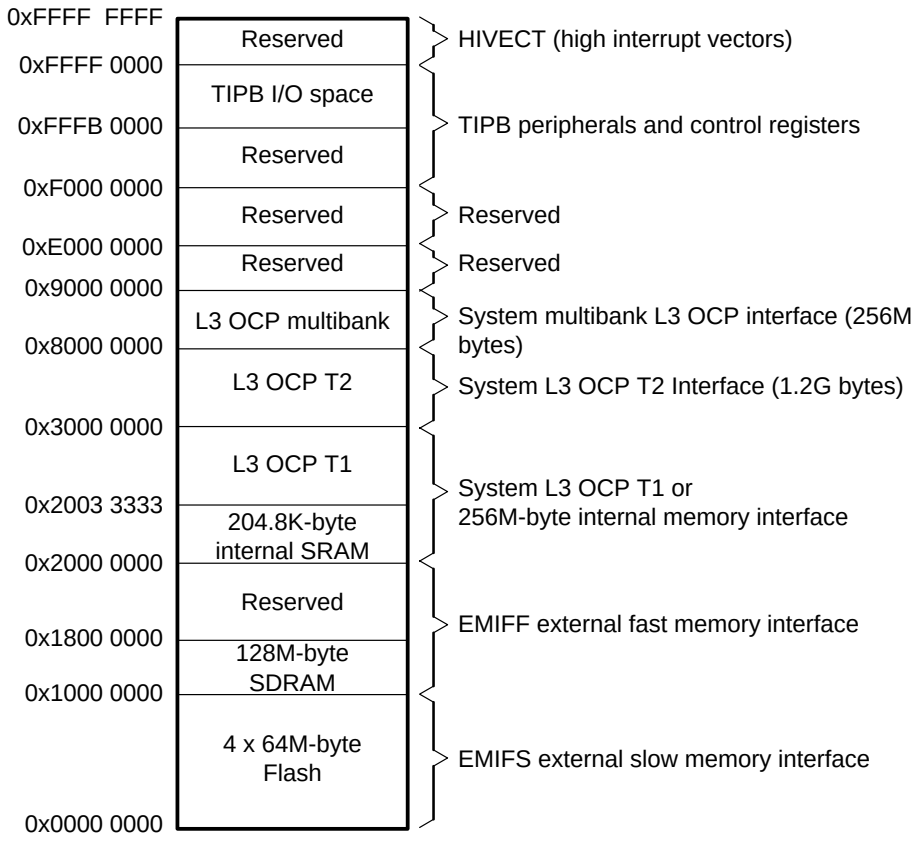
The OMAP730 device includes the MPU subsystem, the GSM subsystem, a memory and traffic controller, general-purpose peripherals, dedicated multimedia application (MMA) peripherals, and multiple interfaces. The MPU and GSM share access to the 128M bytes of fast memory space, 256M bytes of slow memory space, and 640 bytes of dual-port RAM in ICR.

1.4 Memory Maps

1.4.1 Memory

The GSM-S and the MPU-S share memory via the traffic controller (TC).

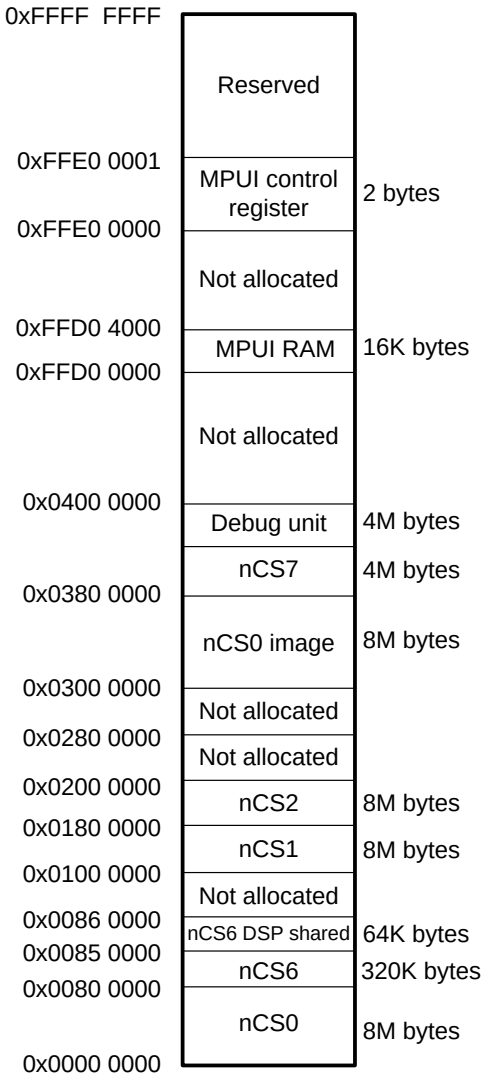
Figure 1–5. MPU Memory Map



1.4.2 GSM-S MPU Memory Map

The MPU memory space is shared between the external memory interface and the TIPB bus. The memory interface provides six chip-select signals. All internal peripherals are mapped on MPU memory space with a range of 32K bytes.

Figure 1–6. MPU Memory Map



The 8K bytes of internal RAM (0380:0000h to 0380:1000h) can overlay the first 8K-byte region 0000:0000h – 0000:1000h of the GSM-MPU address space. In this case, the first 8K bytes of external memory are not accessible to the GSM-MPU. This overlay is controlled by the GSM-MPU using a register of GSM-MPU memory interface.

1.4.3 GSM-S DSP Memory Space

The GSM-S DSP memory space consists of the following types of memory:

- DARAM: Dual-access data RAM. It is always mapped in data space and can be overlaid in program space using the OVLY bit.
- MPUIRAM: Dual-access data RAM. It is always mapped in data space and can be overlaid in program space using the OVLY bit. The MPU host processor can also access this memory via the MPUI interface module. It behaves as a communication memory between the TMS320C54x DSP CPU and the MPU host processor.
- PROM: Program ROM, always in program space
- DROM: Data ROM, always in data space
- PDROM: Program or data ROM. This ROM is always mapped in program space and can also be mapped in data space by setting the DROM control bit.
- Shared PDRAM: Program/data RAM mapped on both the data space and the program space of the DSP XIO interface

The memory mapping for this S28C128 configuration is:

- 28K words of data memory (RAM-based) mapped in both data space 0 and 1.
  - 2K words of dual-access memory (DARAM)
  - 8K words of dual-access memory (MPUI DARAM) shared between the DSP and the GSP-MPU/DMA
  - 18K words of dual-access memory (DARAM)
- 128K words of program memory (ROM-based)
  - 100K words of program memory (PROM) mapped in program space 0
  - 20K words of data memory (DROM) mapped in data space 1
  - 8K words of mixed program/data memory (PDROM) mapped in both program space 0 and data space 1

Table 1–2 shows the DSP memory mapping.

Table 1–2. DSP Memory Mapping

<table>
<tr><th></th><th>Data</th><th>Prog0</th><th>Prog1</th><th>Prog2</th><th>Prog3</th><th>Prog4</th><th>Prog5</th><th>Prog6</th></tr>
<tr><td>0000</td><td colspan="8">DARAM overlay over the program area—2K</td></tr>
<tr><td>0800
1000
1800
2000</td><td colspan="8">MPUI overlay over the program area—8K</td></tr>
<tr><td>2800
3000
3800
4000
4800
5000
5800
6000
6800</td><td colspan="8">DARAM overlay over the program area—18K</td></tr>
<tr><td>7000
7800</td><td></td><td rowspan="3">PROM
28K</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>8000
8800
9000
9800
A000
A800
B000
B800
C000
C800
D000
D800
E000
E800
F000
F800</td><td rowspan="3">PDRAM
32K
Drom=0</td><td></td><td rowspan="3">PROM
32K</td><td rowspan="3">PROM
32K</td><td>PROM
8K</td><td rowspan="3">PDRAM
32K</td><td rowspan="3"></td><td rowspan="3"></td></tr>
<tr><td>DROM
20K
Drom=1</td><td></td></tr>
<tr><td colspan="2">PDROM
8K</td><td></td></tr>
</table>

Note: Hatched areas represent memory extension on XIO space

1.5 Security

1.5.1 Main Characteristics

The secure mode is intended to allow the execution of trusted code only, which is the code stored in the secure boundary (secure ROM and RAM or encrypted and signed code in flash ROM–OS and native applications are not trusted).

The secure mode, which creates a virtual security processor, allows the use of the full capacity of the OMAP platform:

- Instruction/data caches enabled
- MMU enabled
- Interruption unmasked

The secure mode offers advanced crypto operations by combining protected keys (secure eFuse) and hardware crypto processor.

Due to the restrictions of the secure mode, the OMAP730 is configured in two ways:

- Production devices:
  - Full security available, no JTAG, debug, or emulation in secure mode. Secure mode violations generate OMAP resets. SDRAM GSM memory-protection violations generate GSM resets.
  - Boot is only allowed in internal boot ROM.
- Emulation devices:
  - Security is partially bypassed, JTAG, debug, and emulation are available. Secure mode and SDRAM GSM memory protection generates only debug requests.
  - Boot on external flash is allowed.

1.5.2 Security Architecture

Figure 1–7 through Figure 1–9 illustrate the security architecture.

Figure 1–7. OMAP730 Security Architecture

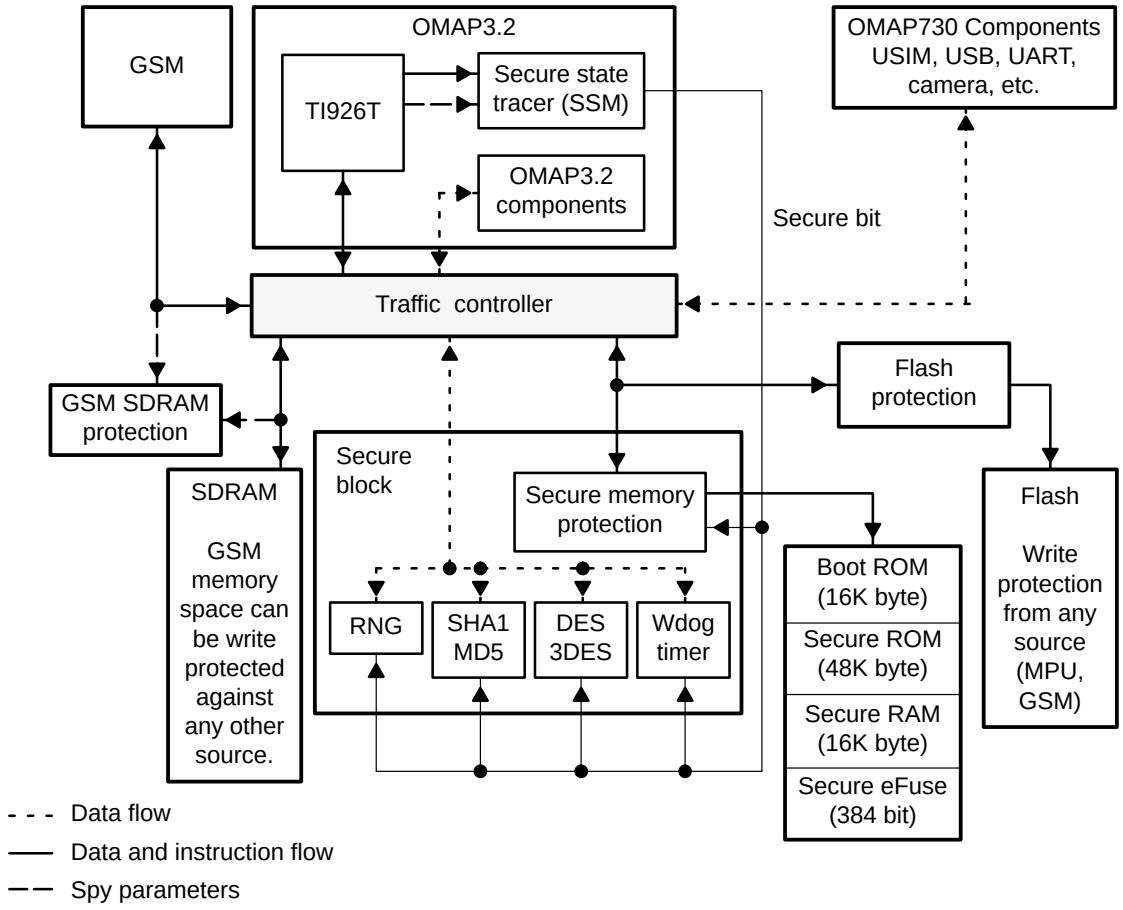


Figure 1–8. OMAP730 Dynamic Security Architecture

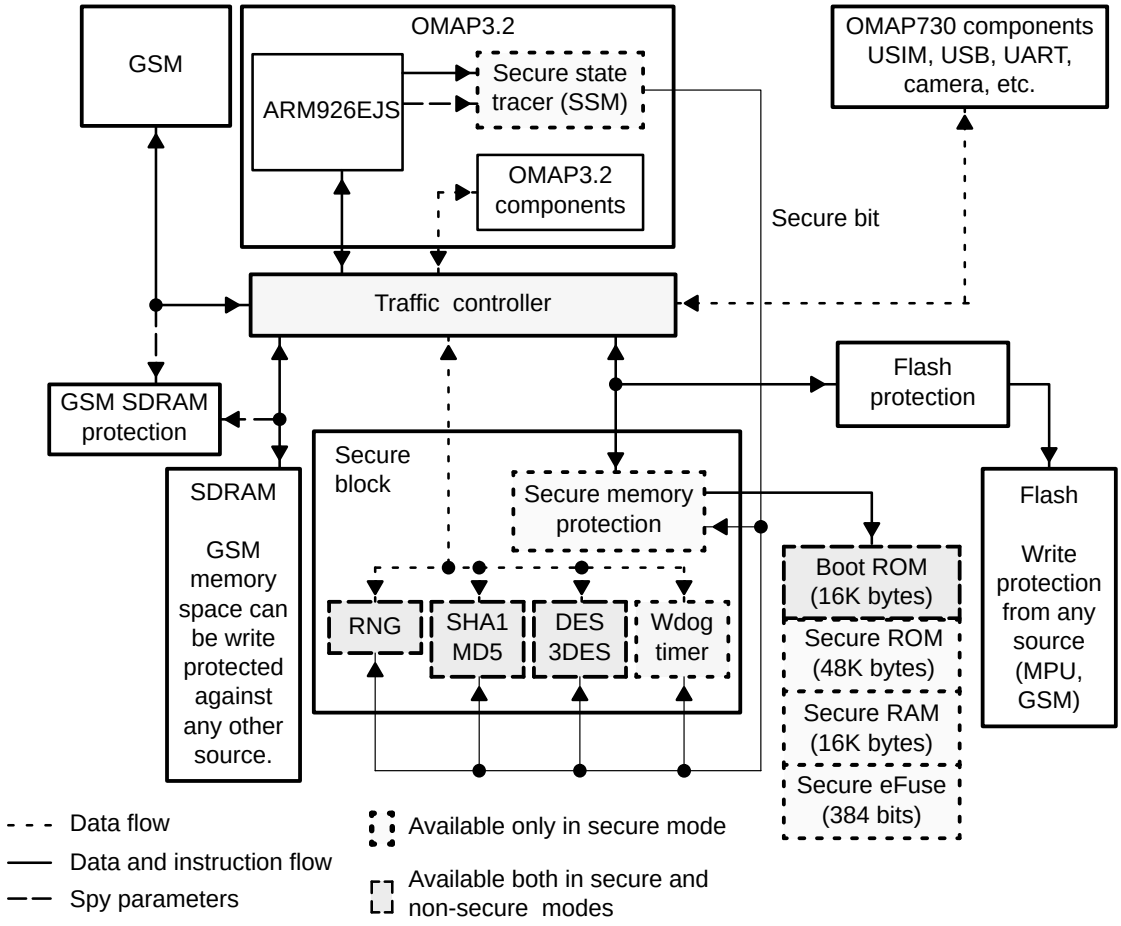
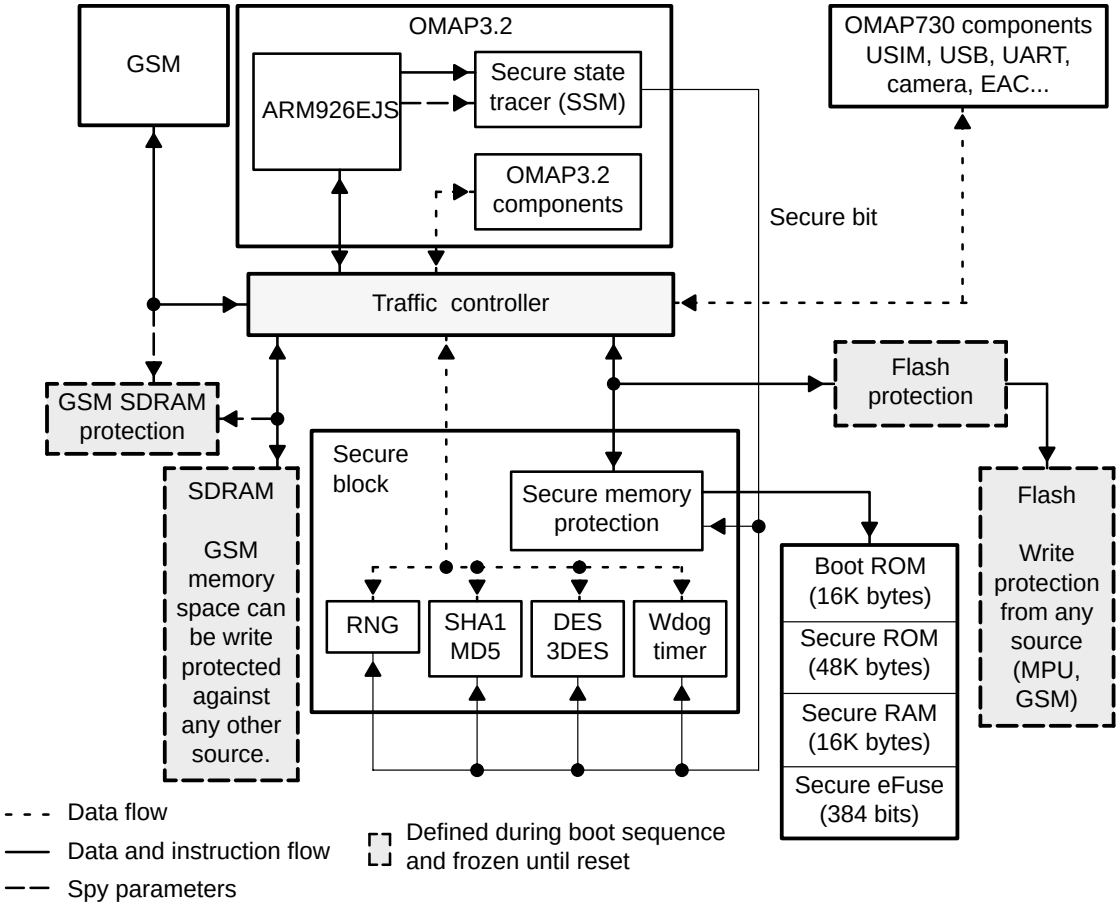


Figure 1–9. OMAP730 Static Security Architecture



1.5.3 Hardware Security

The hardware security in the OMAP730 consists of:

- Hardware secure state tracer, which creates a *third level of privilege* for the ARM926EJS to operate as a separate *virtual security processor* while executing security operations. This is the OMAP secure mode.
- Dedicated secure ROM (48K bytes), RAM (16K bytes), and secure eFuse (384 bits) accessible only in the secure mode.
- Hardware-optimized crypto processors:
  - True random number generator (RNG)
  - Hasher SHA1 and MD5
  - Symmetric encryption DES and 3DES
- Dedicated watchdog timer accessible only in secure mode
- ARM926EJS (OMAP) and GSM-MPU JTAG/debug/emulation protection
- Dedicated NOR/NAND flash write protection against ARM926EJS/GSM access
- Dedicated SDRAM write protection for the GSM against OMAP added components such as USB, UART, etc. access

1.5.4 Boot ROM and Security

1.5.4.1 OMAP730 Boot

Boot Device Configuration

For OMAP730 emulation devices, the MPU core can boot either from the flash or the boot ROM. For OMAP730 production devices, the MPU core boots only from the boot ROM. Booting up is accomplished with an orderly combination of hardware and software control sequences.

1.5.4.2 Security Layer

The OMAP gigacell includes special-purpose security hardware that is used to activate a secure mode. The secure mode can be viewed as a third privilege level on the MPU. It is used to create an environment for protecting sensitive information from access by unreliable software. The secure mode is set by the assertion of a dedicated signal (secure bit) that propagates across the OMAP730 and creates a boundary between resources that trusted software might access and resources available to any software.

The resources controlled by the secure mode are:

- ❏ Part of the ROM (48K bytes of secure ROM)
- ❏ 16K bytes of SRAM (secure RAM)
- ❏ 128-bit and 256-bit eFuses
- ❏ Security watchdog timer
- ❏ RNG
- ❏ SHA1/MD5
- ❏ DES/3DES
- ❏ Security control register

When the secure bit is set, access to the secure resources is unlocked only for the MPU. The system DMA is not granted access to secure resources.

Note:

A secure procedure can grant DMA access to SHA1/MD5 in secure or non-secure mode.

On production devices, the debug and emulation capabilities are disabled for security reasons. However, emulation and debug can be enabled using a secure procedure.

1.5.5 eFuses

- Product_id: 64-bit package level programmable, used to set security parameters and production/emulation mode
- Secure eFuse:
  - KEK: 128-bit probe, level programmable; the random encryption key generated at TI
  - Public_Key: 128-bit package-level programmable, client key field A
  - Private_Key: 128-bit package-level programmable, client key field B
- Die_id: 64-bit probe-level programmable, unique number per device

1.5.6 SHA1/MD5 Accelerator

The SHA1/MD5 security module provides hardware-accelerated hash functions. It can run either the SHA-1 algorithm in compliance with FIPS 180-1 standard or the MD5 message-digest algorithm developed by Rivest in 1991. Up to 2^{20}–1 bytes (1M byte) of data can be hashed in a single operation to produce a 160-bit signature in the case of SHA-1 and a 128-bit signature in the case of MD5.

Note:

It takes 80 steps per 512-bit block of data to process the SHA-1 algorithm.

It takes 64 steps per 512-bit block of data to process the MD5 algorithm.

Each step takes one clock cycle.

Blocks are processed sequentially. This means that to start processing a new block, the accelerator must wait for the end of the previous 80 operation steps (for SHA-1, 64 for MD5) of the previous block.

The SHA1/MD5 can be interfaced either to a host or to a DMA.

1.5.7 DES/3DES

The DES/3DES module provides hardware accelerated data encryption/decryption functions. It can run either the single DES algorithm or the triple DES algorithm in compliance with FIPS 46-3 standard. It supports ECB (electronic codebook) and CBC (cipher block chaining) modes of operation. It does not support the CFB (cipher feedback) and the OFB (output feedback) modes of operation in hardware.

Note:

The DES/3DES module includes:

8-byte input and output buffers

56-bit key size + 8-bit error detection per key (up to 3keys)

16 (DES) round cycles per 8-byte data block

48 (TDES) round cycles per 8-byte data block

Write and read DMA channels

Write and read MPU

No IRQs

1.5.8 Random Number Generator (RNG)

The RNG module provides a true, nondeterministic noise source for the purpose of generating keys, initialization vectors (IVs), and other random number requirements. It is designed for FIPS 140-1 compliance, facilitating system certification to this security standard. It also includes built-in self-test (BIST) logic, which allows testing the randomness of the module output and its compliance with FIPS 140-1 standard. An ANSI X9.17, Annex C postprocessor is available to meet the NIST requirements of FIPS 140-1.

The RNG module is made of a hardware-based nondeterministic random number generator core and a wrapper, which provides a bus interface, the clock, reset, and test features.

Note:

It takes 160 RNG clock cycles to generate a new key.

The computation of a new key starts after each host-read access to the key output register.

1.5.9 Secure Watchdog Timer

There are two instances of the watchdog module in OMAP730. One is clocked at 32 kHz and the other at a different clock frequency (secure watchdog). One is used for security purposes and the other for unsecure applications.

The watchdog module is a 32-bit general-purpose counter (same programming model as the 32-bit general-purpose timer) with watchdog capability (generates a reset). When it is used as secure watchdog, it is accessible only in secure mode.

When OMAP730 secure configuration is used, the secure watchdog is only reset on power up and then it starts counting. It is not sensitive to other reset sources.

Chapter 2

MPU Subsystem

This chapter presents a description of the OMAP730 MPU subsystem.

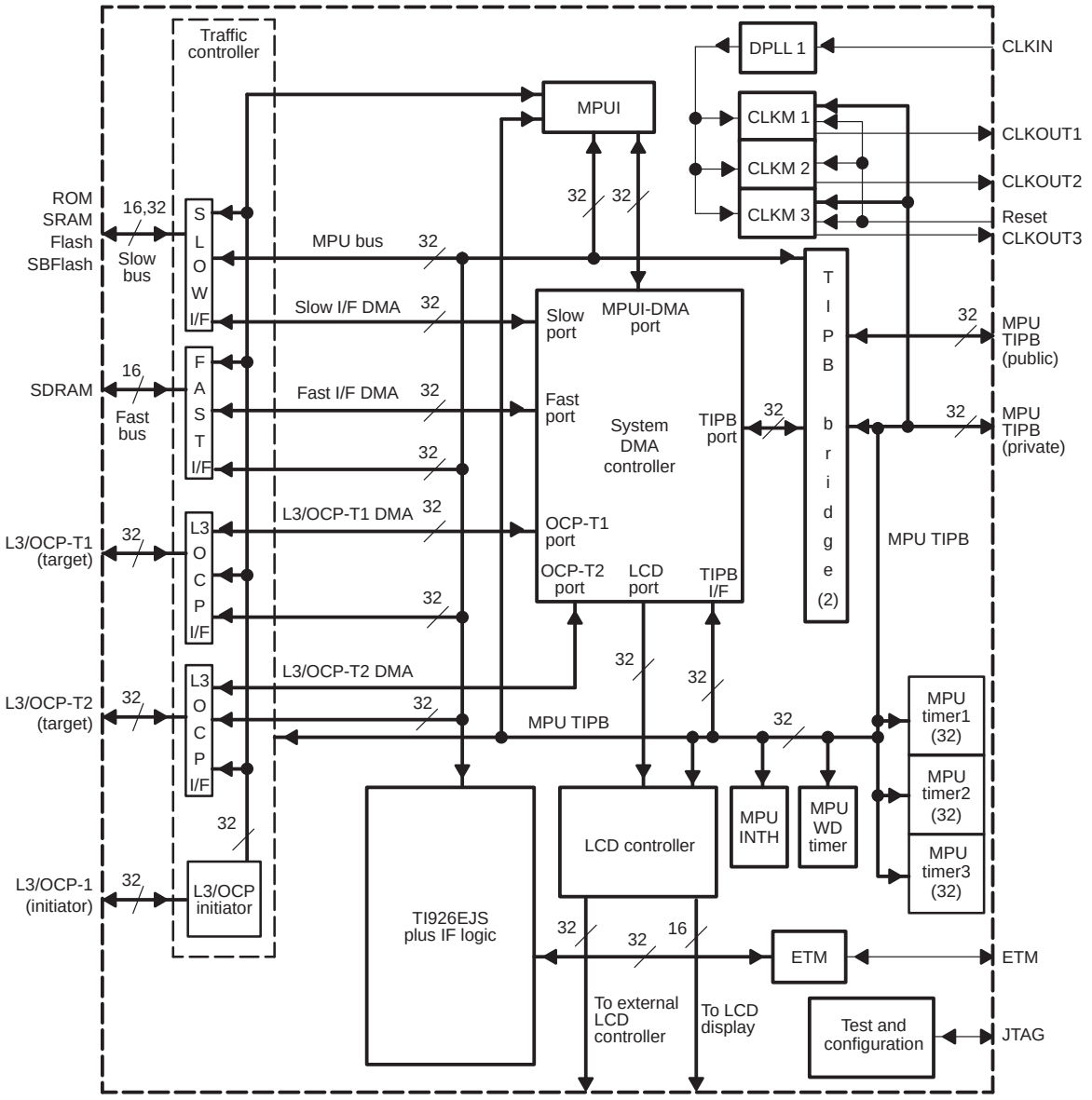
Topic **Page**

2.1 OMAP730 Platform Description

The MPU subsystem is based on the ARM926EJS CPU and includes:

- Memory management units (MMU)
- Instruction cache
- Data cache
- L3 system interface ports (compliant with Open Core Protocol)
- DMA controller (multichannel)
- Traffic controller that handles data flow among the:
  - MPU
  - System DMA
  - L3 OCP
  - External memory
  - OCP T1
  - OCP T2
  - Other peripherals
- Three 32-bit timers
- One 16-bit watchdog timer
- Interrupt handler
- One DPLL and three clock managers
- TI peripheral bus (TIPB) bridges
- Liquid crystal display (LCD) controller
- ETM9 module in medium configuration

Figure 2–1. OMAP730 Platform Layout



The MPU-S subsystem contains:

- OMAP 3.2 data processing core
- Power and clock control (PCC)
- Two UART modems—SIR/MIR/FIR IrDA
- General-purpose I/Os (GPIO)
- USB OTG controller with 48-MHz APLL
- Multichannel buffered serial port (McBSP)
- Enhanced audio controller (EAC)
- Multimedia memory card and secure digital I/O host controller (MMC/SD and 4-bit SDIO)
- CompactFlash interface
- NAND flash controller
- Fast I^2C master/slave controller
- SmartCard (SMC) controller with direct I/O interface
- µWire controller
- HDQ and 1-Wire master controller
- Camera interface
- Pulse-width tone modulator (PWT)
- Pulse-width light modulator (PWL)
- Serial port interface (SPI_100K)
- LCD controller
- LCD low-power controller (LLPC)
- VLYNQ interface
- Dual-mode timer
- SHA-1/MD5 accelerator
- DES/3DES
- Random number generator (RNG)

2.2 Boot Sequences for OMAP730 ES1.0 Device Revision

Note:

This boot ROM specification is valid only for OMAP730 ES1 device revision.

A new mechanism common to all other OMAP devices is used for the production device revision. For information on other device revisions, see Section 2.3, *Boot Sequences for OMAP730 Revision ES1.1 (or higher)*.

This section details the functionality and use of the OMAP730 public boot code, referred to as the *boot code* in this document.

This embedded application is executed after reset of the device. On ES1 samples of OMAP730, this ROM_CODE can be bypassed by enabling the overlay mode (set pin ARM_BOOT_MPULPG2 to high level). In this case, the device boots from external memory mapped on CS3.

The OMAP730 ROM code software is an embedded application that allows booting from the following external devices:

- ❏ UART, by a download into the internal memories of the application
- ❏ NOR flash mapped on CS1, CS2, or CS3
- ❏ NAND flash mapped either on CS2 or CS3

The OMAP730 ROM code also contains some production tests not described in this document.

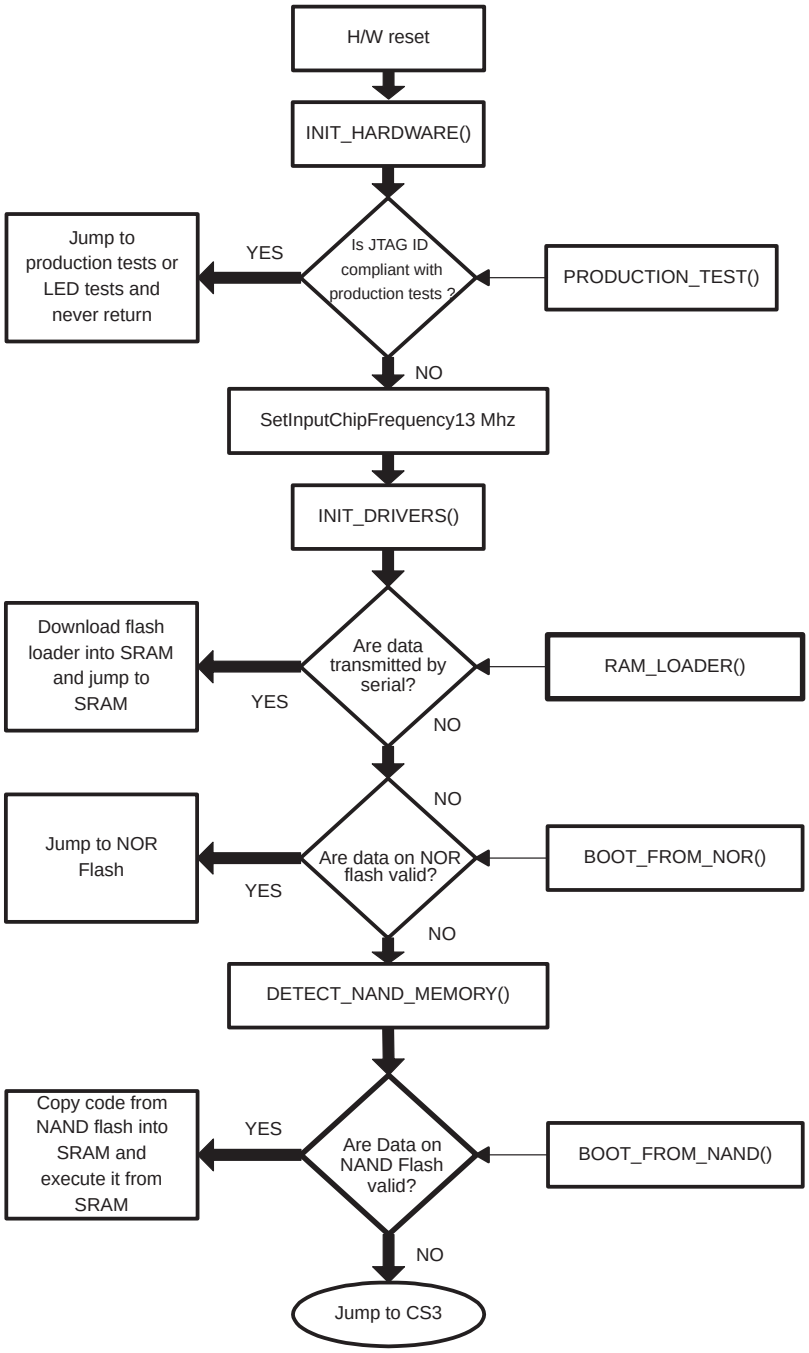
The ROM code does not provide any security or authentication drivers to check the validity of flash or execute security algorithms. It only provides access to security hardware accelerators (SHA-1/MD5, DES3DES, and RNG) in nonsecure mode.

2.2.1 ROM Code Overview

Figure 2–2 is a flow diagram of the ROM code execution. The execution is split into several parts:

- ❏ Hardware initialization, such as enabling clocks, releasing peripheral reset, and setting up the internal frequency to 13 MHz
- ❏ 30-ms wait for application download into internal memory from MPU UART1. If download is successful, execute the downloaded application.
- ❏ Detection of NOR flash mapped on CS3, CS2, OR CS1 by reading the firmware certificate. If successful, execute the code from flash.
- ❏ Detection of NAND flash mapped on CS2 or CS3 by reading manufacturer and device identifiers. If successful, check for the presence of a firmware certificate in flash. If it is found, the image is copied from NAND flash to internal RAM and execution is transferred to this image.

Figure 2–2. ROM Code Execution



2.2.2 ROM Code Description

This section presents a description of the ROM code.

2.2.2.1 Hardware Initialization

The execution of the ROM code starts with hardware and software initializations:

- ❏ Initialization of supervisor stack pointer

- Initialization of clocks, memory interface, TC, secure mode:
  - Initialization of OMAP CLKM/TC/DPLL/PCC:
    - Clock mode is unchanged: Use default mode → fully synchronous mode.
    - DPLL mode is unchanged: Use default mode → DPLL in bypass mode.
    - Enable the following clocks: XORPCLK, PERCLK, and TIMER CLK.
    - Release peripherals reset.
  - Memory interface configuration:
    - Disable write buffer of private TIPB bridge Strobe1 to have safe accesses to memory interface registers.
    - CS1 and CS2 are configured in asynchronous mode (default mode), 16-bit mode, flash clock divided by 1, RDWST = F, WRWST = F, PGWST/WELEN = F.
    - CS3 is configured as CS1.
    - CS0 is defined in TI ACE ROM mode as 32 bit (default mode).
  - No SDRAM configuration
  - No interrupt configuration (ARM926EJS and interrupt handler) and no exception management
- ARM926EJS instruction cache disable
- ARM926EJS MMU and data cache disable
- Watchdog timer disable
- Secure mode: The PERSEUS_CONF register (SECCTRL) allows accesses to the following peripherals in nonsecure and secure modes:
  - SHA-1/MD5
  - DES3DES
  - RNG
- Call to production test if a special value is found in the JTAG user register
- Set internal frequency to 13 MHz: This is done by gauging of the input clock (13 MHz or 26 MHz) by timer 1 on eight periods of the 32-kHz timer in order to determine input chip frequency. In case of 26-MHz input frequency, the Pcc divider is enabled so as to have 13 MHz on the chip. In case of 13-MHz frequency, the Pcc divider is disabled.
- Global and static areas (allow use of C-code)
- MPU UART1 setup
  - Enable UART clock (48 MHz): The DIS_UART1_DPLL_REQ bit of SOFT_DISABLE_REQ_REG must be set to 0 (default value), and bit

UART1_DPLL_REQ of PERSEUS_PCC_CONF_REG must be set to 1.

- Configure MPU UART 1 to 19200 bits per second, 8 bits, no parity.
- Initialize pin multiplexing for MPU-UART1 on top level pins MPU UART1 Tx and MPU UART1 Rx.

2.2.2.2 Boot From UART

The flash programmer procedure downloads an application into internal memory on the device using MPU UART modem 1. Top level pins MPU UART1 TX and MPU UART1 RX are used for the download. The flash programmer download is called when synchronization on UART1 has been detected before a 30-ms time-out. In case of a time-out, the software tries a boot from NOR flash.

The application size must be less than 180K bytes.

The OMAPHostLoader application is used on a PC. The default format used on the serial port is 19200 bits per second, 8 bits, no parity.

Download Protocol

The RAM loader contained in boot ROM code is considered as a slave application, that is, the RAM loader can only receive commands and send associated responses in order to inform the remote application. The following table shows the set of commands used for synchronization between the PC and the device during download.

Table 2–1. Download Synchronization Commands

PC Side		Device Side	
Description	**Command**	**Description**	**Response**
Signaling request	<i	Signaling response	>i[X][Y]
Parameter request	<p[X][Y][Z]	Parameter ACK response	>p
		Parameter NACK response	>P[X]
Write request	<w[X][Y][Z]	Write ACK response	>w
		Write NACK response	>W[X]
Abort request	<a		
Certificate request	<c	Certificate ACK response	>c[X]

Application downloading is done in several steps:

1) First, the synchronization between the PC and the device is done by a sequence of signaling requests/responses.
2) The host application sends the parameter sequence if needed.
3) The host downloads the application certificate.
4) The host downloads the application by performing several sequences of write requests/responses.
5) At the end of the download, the device jumps to the starting address defined in the certificate.

The details of the protocol are described in Table 2–2.

Table 2–2. Download Protocol

Format	Parameters	Definition
>i[X][Y]	X: ROM code version	Defined on 16 bits: 0000
	Y: Reserved	Defined on 128 bits: 0000000000000000
<p[X][Y][Z]	X: UART baud rate to use during download	0x00: 812500 bps 0x01: 406250 bps 0x02: 203125 bps 0x03: 115200 bps 0x04: 57600 bps 0x05: 38400 bps 0x06: 28800 bps 0x07: 19200 bps Defined on 32 bits
	Y: UART time-out configuration updated according to the MPU clock configuration. The time-out avoids putting the RAM loader in a deadlocked state if a complete command is not received.	Value 0x00000000 disables the time-out. The command decoder waits for the entire command to be received.
	Z: Flash programmer certificate	
>p	None	None
>P[X]	X: Error code	Defined on 8 bits 0x01: Incorrect baud rate 0x02: Incorrect certificate 0x03: Incorrect code address
<w[X][Y][Z]	X: Number of bytes in the block	Defined on 32 bits, hexadecimal value. Must be a modulus of 64 bytes, except for the last command.
	Y: Address where the block must be written Z: Block	Defined on 32 bits, hexadecimal value. Each data defined on 8 bits
>W[X]	X: Status of the error	Defined on 8 bits 0x01: Address error 0x02: Bad block size 0x03: First block address does not correspond to code address specified in certificate 0x04: Firmware signature error 0x05: Received code size does not correspond to code size specified in certificate

Table 2–2. Download Protocol (Continued)

Format	Parameters	Definition
<a	None	None
<c	None	None
<a	None	None
<c[X]	X: Firmware certificate	To be filled

Manufacturer Certificate Description

The manufacturer certificate is used by the firmware to certify and check the integrity of the program before execution. Table 2–3 presents a detailed description of the manufacturer certificate.

Table 2–3. Manufacturer Certificate Description

Name	Size in Bits	Size in Bytes	Description
CERT_SIZE	16	2	Certificate size in bytes. Forced to 1332 bytes.
RESERVED	8	1	Reserved
RESERVED	8	1	Reserved
CODE_ADDR (1)	32	4	Address of the code to check
CODE_SIZE (2	32	4	Code size in bytes
CODE_START_ADDR	32	4	Address of firmware entry point
RESERVED	2112	264	Reserved
RESERVED	2112	264	Reserved
RESERVED	2048	256	Reserved
RESERVED	2048	256	Reserved
RESERVED	160	20	Reserved
RESERVED	2048	256	Reserved

Notes: 1) Code address must fit in the range 0x2000:000: 0x2002:D000.
2) Code size must be less than 180K bytes (0x2D000).

RAM Loader State Machine (Device Side)

Table 2–4 presents the state machine of the the RAM loader program.

Table 2–4. RAM Loader State Machine

STATE	EVENT	ACTION	NEXT STATE
Signalling	SIGNALLING_ REQUEST	Send(SIGNALLING_ RESPONSE)	Signalling
	PARAMETER_ REQUEST &&(Bad Baud rate || Bad Certificate)	Send(PARAMETER_ NACK_RESPONSE)	Signalling
	PARAMETER_ REQUEST && (Correct baud rate && correct certificate)	Prepare response Send(PARAMETER_ACK_ RESPONSE) Set new baud rate	AWAIT_COMMAND_ PHASE
	CERTIFICATE_ REQUEST	Prepare response Send(CERTIFICATE_ACK_ RESPONSE)	Signalling
	ABORT_REQUEST	None	Signalling
AWAIT_ COMMAND_ PHASE	PARAMETER_ RQUEST && (bad baud rate || bad certificate)	Send(PARAMETER_ NACK_RESPONSE). Set default baud rate.	Signalling
	PARAMETER_ REQUEST && correct baud rate && correct certificate	Prepare response Send(PARAMETER_ACK_ RESPONSE). Set new baud rate.	COMMAND_PHASE
	WRITE_REQUEST && Bad arguments	Prepare response. Send(WRITE_NACK_ RESPONSE). Set default baud rate.	Signalling
	WRITE_REQUEST && Correct arguments	Copy block in internal RAM. Send(WRITE_ACK_ RESPONSE)	COMMAND_PHASE
	ABORT_REQUEST	Set default baud rate	SIGNALLING
	SIGNALLING_ REQUEST	Send (SIGNALLING_ RESPONSE)	AWAIT_COMMAND_ PHASE
COMMAND_ PHASE	WRITE_REQUEST && Bad arguments	Send (PARAMETER_ NACK_RESPONSE) . Set default baud rate.	Signalling
	WRITE_REQUEST && Correct arguments	Copy block in internal RAM. Increment code size counter. Send(WRITE_ACK_ RESPONSE)	COMMAND_PHASE

Table 2–4. RAM Loader State Machine (Continued)

STATE	EVENT	ACTION	NEXT STATE
	WRITE_REQUEST && Correct arguments && Code size reached && Code signature OK	Copy block in internal RAM. Send(WRITE_ACK_ RESPONSE) Set default baud rate. Execute the flash programmer.	Signalling
	WRITE_REQUEST && Correct arguments && Code size reached && Code signature Not OK	Prepare response. Send(PARAMETER_ NACK_RESPONSE) Set default baud rate.	Signalling
	SIGNALLING_ REQUEST	Send(SIGNALLING_ RESPONSE)	Signalling

2.2.2.3 Booting From NOR Flash

This boot operates only on nonmultiplexed 16-bit NOR flash. This boot mode is compliant with a boot on the first area of Disk-On chip.

This boot begins with detection of NOR flash. This operation is done by reading the firmware certificate. The firmware certificate consists in an array of 992 bytes located at address 0x20 of the flash.

Byte Address	Value	Description
0x20	0x809795A3	Magic number
0x24		Code size in bytes
0x28	-	Reserved for future use
…	-	Reserved for future use
0x3FC	-	Reserved for future use

Validation of the code present in flash is done by checking the value of the *magic number*. If data are valid, software jumps directly to address 0 of th NOR flash. The CODE_SIZE field is not used in this case. Software at address 0 of the NOR flash is assumed to be in little endian format and corresponds to a valid instruction from the 32-bit MPU instruction set.

The detection occurs sequentially on each chip select (CS3, CS2, CS1). If no devices are detected, the device tries a boot from NAND flash.

Note:

EMIFS configuration is kept in asynchronous mode regardless of the NOR flash detected.

2.2.2.4 Booting From NAND Flash

The detection of NAND memories is done by reading the manufacturer and device codes on CS3 and then on CS2.

This boot mode only detects CE don't care Samsung NAND flash (8 bits and 16 bits/1G bits, 2G bits, and 256M bits) defined as follows:

- K9F1G08Q0M
- K9F1G08U0M
- K9F1G16Q0M
- K9F1G16U0M
- K9F2G08Q0M
- K9F2G08U0M
- K9F2G16Q0M
- K9F2G16U0M
- K9F5608Q0B
- K9F5616Q0B

Note:

Reading the manufacturer and device codes generates a write to CS3 or CS2. This write can affect other devices connected to EMIFS (ASIC, and so on). Therefore, if a NAND flash is detected on CS3, software does not scan CS2.

The validity of the software in NAND flash is controlled in the same way as in NOR flash: by checking the magic number of the firmware certificate. If it matches, the code contained in flash is copied into internal SRAM. The amount of data to be transferred is defined in field CODE_SIZE of the firmware certificate. After the copy, execution starts from internal memories.

Limitations

Access is managed only to the first block of the Samsung NAND flash. This access is done without performing any error corrections because the first block of the NAND flash is ensured to be a valid block. Other blocks are not accessed by software.

Therefore, the size of the code downloaded from the NAND flash to internal SRAM must be less than 128K bytes for 1G-bit- and 2G-bit NAND flash, and 16K bytes for 256M-bit NAND flash.

2.3 Boot Sequences for OMAP730 Revision ES1.1 (or Higher)

2.3.1 Boot ROM Execution

This section covers the essential mechanisms of the on-chip ROM, code, and booting scenarios that can be handled. This also includes information on how a programming host preloads the internal RAM with a flash loader in order to program flash memories connected to the device, which is also referred to as *flashing*.

2.3.1.1 Authentication

All code that is external to the on-chip ROM must be authenticated for security reasons. A failed authentication causes the chip to enter the reset state. Authentication of code residing in external flash or code that is downloaded is handled by routines in the secure part of the on-chip ROM.

The signing tools provided by TI are responsible for generating (on the programming host PC before download) the final image containing relevant security information to be programmed onto the OMAP73x platform.

2.3.1.2 Reset Considerations

At the rising edge of the OMAP73x reset pin NRESPWRON (power-on reset), the three mode input pins RESET_MODE, ARM_BOOT_MLPG2, and MUX_MODE_MLPG1 are sampled and their values are stored in the CONF_STATUS register:

- ARM_BOOT_MLPG2: Use of internal ChipSelect0 or external ChipSelect3
- MUX_MODE_MLPG1: EMIFS multiplexed or not (for reset mode 0 only)

In addition, the factory setting of the silicon device type (value originating from eFuse and configuration registers affects how the two mode pins force the ARM to select the method for fetching the first instruction.

A production-type device always starts from on-chip public ROM. For an emulation-type device to do so, one must ensure that the mode pin ARM_BOOT_MLPG2 = 0. An emulation-type device has the option to boot from chip-select 3 if ARM_BOOT_MLPG2 = 1 (see Table 2–5). This provision is for running nonsecure test code and is not further discussed here.

Table 2–5. EMIF Configuration After Reset

Device_type Set in eFuse	ARM_BOOT at Reset	Boot Selected	EMIFS Boot Protocol Address/Data
Normal 00 (Production device)	X	Internal CS 0	Nonmultiplexed (FLASH_CFG_[22] = 0)
			Multiplexed (FLASH_CFG_[22] = 1)
Emulator 10 (Debugging Device)	0		Nonmultiplexed (FLASH_CFG_[22] = 0)
			Multiplexed (FLASH_CFG_[22] = 1)
	1	External CS 3	Nonmultiplexed (FLASH_CFG_[22] = 0)
			Multiplexed (FLASH_CFG_[22] = 1)

Refer to the following CONF_STATUS register bits:

- ❏ [5:4] eFuse generated device type
- ❏ [3] Sampled value of MUX_MODE_MLPG1 pin
- ❏ [1] Sampled value of ARM_BOOT_MLPG2 pin

The secure state machine (SSM) address comparators also remain active for emulation devices booting from EMIFS CS3. This allows the user flash code to enter secure environment (SE) through a far branch at 0x00003C18 that jumps to the secure entry point function now offset to the location 0x0C003C18.

In case SE is not meant to be used, it is the user's responsibility to avoid accidental violation of the secure entry point address 0x00003C18 that is monitored by the SSM by reserving the addresse range 0x00003C00–0x00003C1F and not using addresses in this range. This range corresponds to the 32-byte (8 words) cache line that upon prefetch triggers an SSM secure entry decision, because it always includes a fetch from 0x00003C18.

2.3.1.3 Public ROM Code

The public ROM code determines boot sequence, queries interfaces for initiating flash programming, and provides generic housekeeping, such as:

- ❏ Disabling of maskable interrupts
- ❏ Clock setting and DPLL setting
- ❏ Configuration of ULPD registers to configure the APLL in order to provide 48-MHz clocks to the UART
- ❏ Configuration of MPU internal registers
- ❏ Configuration of EMIFS registers
- ❏ The configuration of the NAND flash controller
- ❏ The building of MPU exception vectors table

and, if flashing is required,

- The configuration of UART1 (UART modem) and 3 (UART modem-IrDA)
- The configuration of USB (only in reset mode 0)
- The setting of multiplexing depending on reset mode

The ROM boot code detects whether the device must be flashed or not. This is determined by querying the supported interfaces used for download. If flashing is not called for, the boot code looks for external code to run.

The different ways of booting are:

- Normal boot from a NOR or a NAND flash without flashing
- Flashing via UART or USB, and then booting from NOR or NAND

The boot code uses secure mode to:

- Check the signature of keys and header information
- Check the signature of the application

2.3.1.4 Initialization

The boot code execution starts by checking for flashing requests; else, the code identifies the X-LOADER to be launched. The X-LOADER is a user-defined pre-OS bootstrap code residing at the beginning of the external flash. Its purpose is to give the user the option to perform one of the following actions:

1) Branching to the normal OS-bootstrap entry point, that is, normal boot
2) Branching to an alternate production boot entry point of the OS environment
3) Branching to an additional downloader (ADL) for upgrading parts or all of the OS-environment image in flash

2.3.2 ROM Operating Modes

The boot code has five different operating modes determined by a programming host's boot command or by its absence:

- Normal boot: Searches for the TOC items pointing out the subimages X-LOADER and KEYS in flashes and passes execution to X-LOADER if X-LOADER is present and all verifications passed. If the X-LOADER is not present, software loops until the watchdog reset.
- ADL Boot: Same as normal boot except that the boot code passes a parameter to the X-LOADER to use ADL mode instead of normal mode. The X-LOADER implementation is requested to start an ADL application from flash or internal RAM.
- Flash download boot: Downloads an image containing a subimage containing a flash loader called 2ND, and another subimage called KEYS via

a supported interface to the internal RAM, starting from 0x20000400. After the download has been completed, KEYS are imported to secure RAM and verified, and after that the 2ND flash loader is authenticated and executed.

- Production boot: Same as normal boot except that the boot code passes a parameter to the X-LOADER to use production mode instead of normal mode. The X-LOADER implementation can use this information to start an OS with a specific mode (for example, fast start up). The ASIC is alive only for a couple of minutes because the secure watchdog update is disabled.

- AT-speed: Is a special case and detection of it is based on the content of the user JTAG register. If the LSB in that register contains 0xA5, the boot code branches to production test code at CS3 0x0C000000 in thumb mode. The address of the user JTAG register is 0xFFFE5800.

Start-Up

Figure 2–3 shows the two main boot-code processes.

Figure 2–3. Main Boot Code Processes

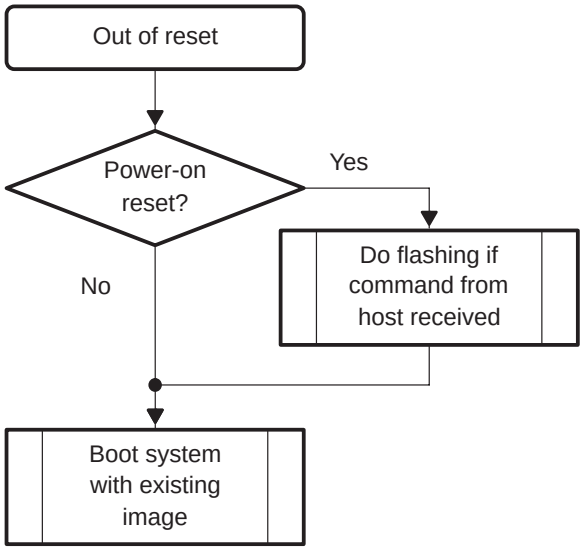
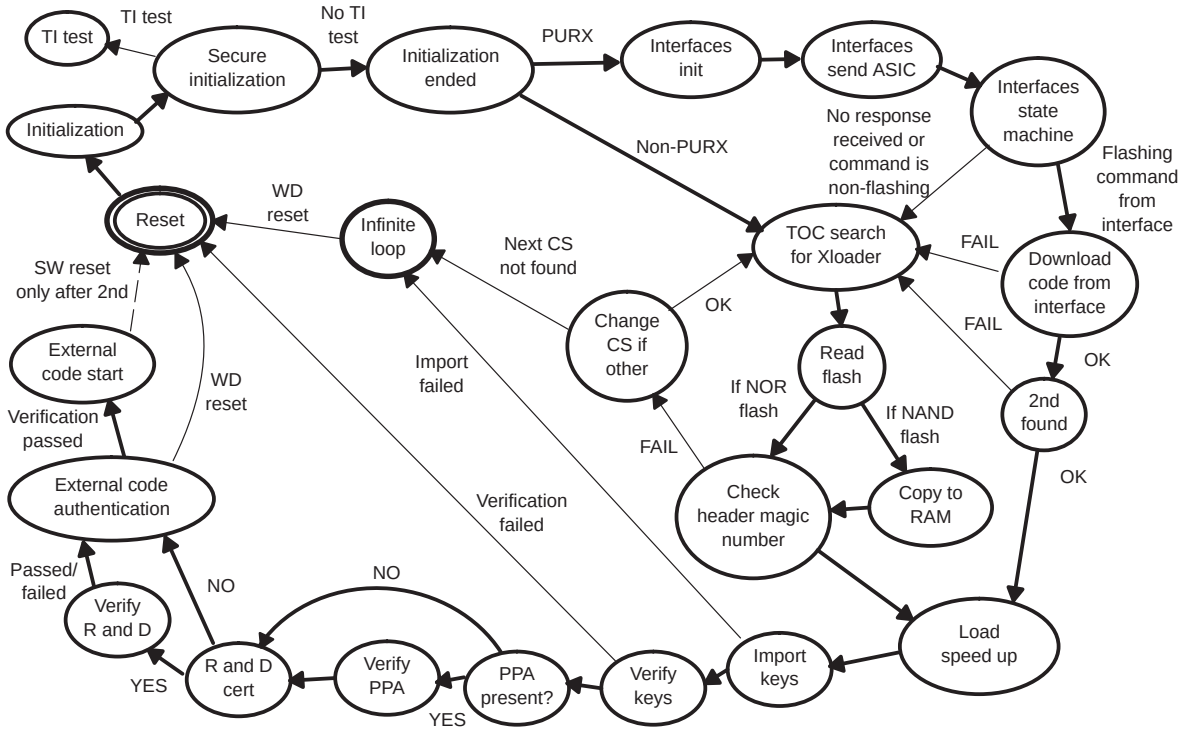


Figure 2–4 shows the major states of the boot code. The boot code contains only one real state machine, which is implemented to handle interface communication.

The **light-grey** states in the figure are security ROM states.

Figure 2–4 is not a formal state machine diagram: states that do not have dependencies are always executed once, and the execution order of those states is clockwise starting from the reset state. PURX means power-on reset.

Figure 2–4. States Of The Boot Code



2.3.3 Flashing

Flashing is possible only after power-on reset. The boot code detects the desired interface for flashing by initializing each interface (UART1, UART3, or USB) and sending a 69-byte ASIC ID to each interface (58-byte only for UART—see Table 2–6). When the ASIC ID has been sent, the boot code polls for 200 ms to receive a boot command from the programming host in order to download a flash loader. The interface must be one of the following:

- ❏ UART1
- ❏ UART3
- ❏ USB

The received 4-byte boot command either indicates flash download programming or ADL booting and flashing sequence.

If no command is received, the boot code starts normal boot without flashing. The downloaded flash loader (defined by the user) is a secondary software for writing an image to flash memory. The flash loader is also an image including a header, a flash loader application, and some additional information. The flash loader is downloaded to the internal RAM. ROM code verifies its integrity by checking signatures in secure mode. The device is reset if integrity check fails and executes the loader if passed. The flash loader then downloads and writes the larger OS-image to the flash type it supports.

2.3.3.1 Interface Interaction

The interface interaction is performed only after a power-on reset to avoid uncontrolled flashing requests. In case of warm reset, normal boot is done.

UART connection speed is 115.2K bits-per-second and does not use hardware or software flow control.

After power-on reset, the boot code sends out an ASIC ID via UART1, UART3, and the USB to give security and version-related information. The ASIC ID can be used for ASIC detection. A programming host can send a command for normal boot to avoid the 200 ms delay during boot that is described below.

Hence, there are two possibilities to execute in normal boot mode after power-on reset:

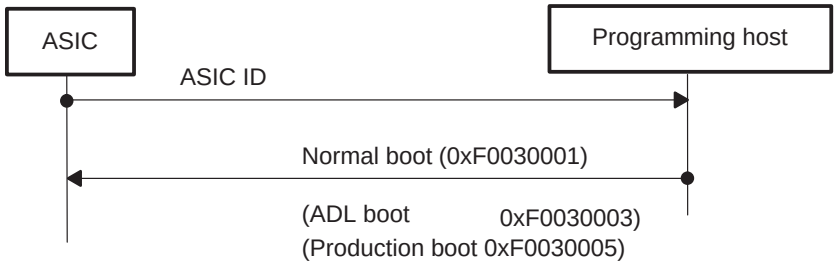
1) The fastest way is to send a normal boot command to the boot code.

2) Wait 200 ms for a time-out when no command was sent.

Normal, production, and ADL boot commands are described in Figure 2–5.

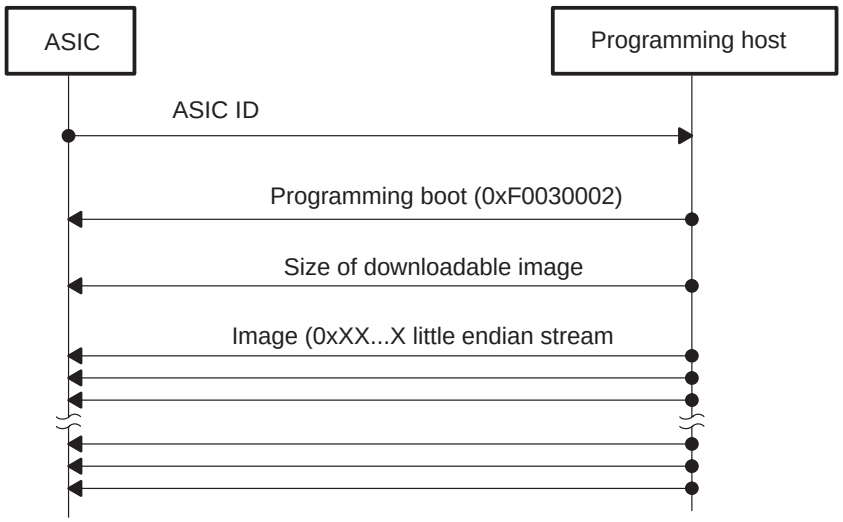
Commands must be sent as a byte stream in little endian format.

Figure 2–5. Normal Boot



Programming sequence starts as in normal boot, but continues as in Figure 2–6.

Figure 2–6. Programming Sequence



The time-out to receive all mentioned data is set to one minute. Normal boot is performed after the time-out.

When the programming boot command is received, the boot ROM code copies data directly to internal RAM address 0x20000400.

The downloaded image must contain a table of contents (TOC) pointing to the included subimages called 2ND and KEYS. The primary protected application and the R&D certificate are not mandatory, but those are imported if they are included with the image.

The boot code performs a speed-up based on the header of 2ND subimage, and then authenticates and executes 2ND (performed by secure ROM).

The 2ND subimage is responsible for the actual flashing sequence. Before it is executed, a magic number found in the subimage header must match a predefined value.

2.3.3.2 ASIC ID Contents

The ASIC ID (see Table 2–6) contains sub-blocks (Table 2–7 through Table 2–11) from several different entities. The checksum sub-block is sent via the USB, but not via the UARTs (because the size of UART FIFO buffers is 64 bytes and the full ASIC-ID is 69 bytes).

Table 2–6. ASIC ID Contents

Register	Description
Items	1 byte Number of sub-blocks (UART = 4, and USB = 5)
ID sub-block	7 bytes
Secure mode sub-block	4 bytes
Public ID sub-block	23 bytes
Root key hash sub-block	23 bytes
Checksum sub-block	11 bytes

Table 2–7. ID Sub-Block

Field	Value	Description
ID	0x01	Sub-block ID
Length	0x05	Size of sub-block
Fixed	0x01	Identifies OMAP73x sub-block
OMAP_ID	0x161007	3-byte long fixed OMAP73x ID
Version	xx	From 0xFFFED404 most significant byte (32-bit access)

Table 2–8. Secure Mode Sub-Block

Field	Value	Description
ID	0x13	Sub-block ID
Length	0x02	Size of sub block
Fixed	0x01	Identifies OMAP73x sub-block
Data	0xX…X	Read from structure that is filled during the secure ROM

Table 2–9. Public ID Sub-Block

Field	Value	Description
ID	0x12	Sub-block ID
Length	0x15	Size of sub-block
Fixed	0x01	Identifies OMAP73x sub-block
Data	0xX…X	The 20-byte long public ID is generated by secure ROM.

Table 2–10. Root Key Hash Sub-Block

Field	Value	Description
ID	0x14	Sub-block ID
Length	0x15	Size of sub-block
Fixed	0x01	Identifies OMAP73x sub-block
Data	0xX…X	The 20-byte long public ID is generated by secure ROM.

Table 2–11. Chcecksum Sub-Block

Field	Value	Description
ID	0x15	Sub-block ID
Length	0x09	Size of sub-block
Fixed	0x01	Identifies OMAP73x sub-block
Public Csum	0xX…X	4-byte long CRC of the public ROM
Secure Csum	0xX…X	4-byte long CRC of the secure ROM

2.3.4 Booting

2.3.4.1 Booting From External Flash

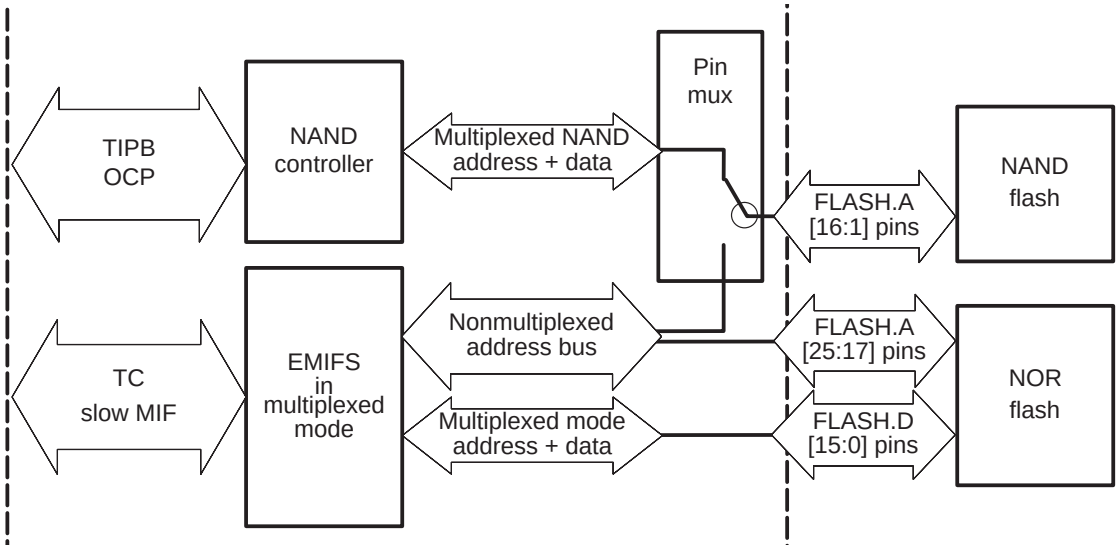
The detection algorithm for external boot flashes searches for a 64-bit hardware code identifying the flash manufacturer’s device type. There are different ID codes for NAND and NOR devices. The readout of this ID requires writing to key positions in the flash that are potentially destructive for devices other than flashes connected to the EMIFS. For that reason, the NOR detect in reset mode 0 is done by scanning for the X-LOADER instead. After the external boot

flash has been found and security and hardware configuration operations have been performed, the X-LOADER execution can be started. The X-LOADER is the pre-OS bootstrap code that can call for additional user-defined boot loading (ADL) or just invoke the OS image directly.

The NAND flash controller only supports an 8-bit data bus and requires the EMIFS to be in multiplexed mode. The external EMIFS pins FADD[1..16] are internally pin multiplexed to the NAND controller bus. The rest of the external EMIFS pins FADD[25..17] and FDATA[0..15] can still be used to access multiplexed NOR flashes in this case. See Figure 2–7.

There is additional software support added for reset mode 0 to boot from NAND flashes on an 8-bit or 16-bit data bus via EMIFS in nonmultiplexed mode.

Figure 2–7. Sharing of EMIFS External Pins



2.3.4.2 Supported Flashes

For normal, production and ADL boot, the boot code searches for an X-LOADER in external flash memories. The ADL case is special because the X-LOADER is passed an ADL command upon invocation so that its own additional boot loader can finalize the earlier initiated flash programming sequence.

The ROM boot code supports start-up from both NAND and NOR flash.

Booting procedure:

1) Initial boot start in on-chip ROM
2) Start the find flash procedure

Case reset mode 0:

- Case MUX_MODE_MLPG1 = 0:
  - Set EMIFS to nonmultiplexed mode

- Scan EMIFS CS3 for nonmultiplexed NOR flash
- Scan EMIFS CS3 with software driver for NAND flash (implemented in ES1.1)

❑ Case MUX_MODE_MLPG1 = 1:

- Set EMIFS to multiplexed mode
- Scan EMIFS CS3 for multiplexed NOR flash
- Change pin multiplexing and scan NAND flash controller CS0-3 for NAND flash

Booting From NOR Flash

If the X-LOADER is found in a NOR flash, security and hardware configuration is performed and execution can start directly from NOR.

Booting From NAND Flash

A mechanism to transfer code from the NAND flash to the 16K-byte internal RAM is implemented.

NAND flashes are divided into blocks and pages, where one block includes several pages. The boot code has a mechanism for avoiding corrupted blocks when copying the image to the internal RAM. In the case where the boot loader needs to be downloaded in the secure RAM, the ROM code downloads it to the internal RAM. Then execution of this code is in charge of downloading the next phase into secure RAM.

2.3.4.3 Table of Contents (TOC)

The boot code searches for a TOC in both NOR and NAND flashes.

The TOC includes information on contents of the programmed image. In this document an image means an entity of programmable data, and the subimage is one piece of that image (such as OS, certificate, and so on). There is one TOC item per subimage, and each TOC item contains fields described in Table 2–12.

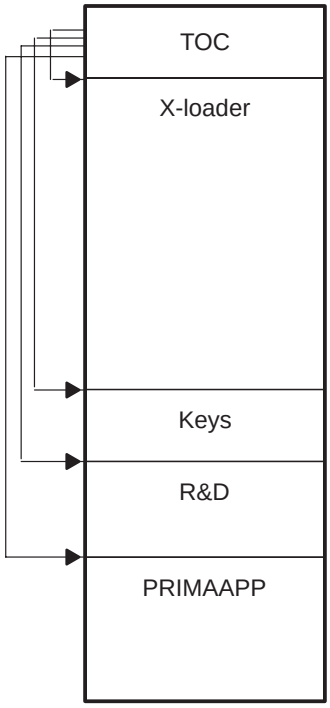
Table 2–12.A TOC Item Structure

Field	Size (in bytes)	Description
Start	4	Start address of subimage
Size	4	Size of subimage
Flags	4	Not used in boot code
Align	4	Not used in boot code
Spare	4	Reserved for future use
File name	12	12-character long name of subimage

The TOC contains as many TOC items as subimages. The size field refers to the size of the subimage. The boot code does not use the flags, align, and

spare fields. The filename is the name of a subimage, and it must end with a NULL character (\0). If fewerthan sixteen (16) TOC items used are, a terminating TOC item is filled by 0xFF.

Figure 2–8. TOC Items



The TOC items are located one after the other. The maximum number of TOC items is 16. The maximum TOC size is 512B. This limit is based on the size of a NAND page.

The boot code detects the TOC by its first item, where the filename must be X-LOADER. The X-LOADER is a general loader started by the boot code from flash. The X-LOADER implementation takes care of the start-up of OS or ADL. The X-LOADER must be authenticated in one part because the ROM does not support multipart authentication handling.

After the X-LOADER file is found, the KEYS TOC item must be found. KEYS contains all the keys required for authentication and must be imported to the secure RAM.

If present, the boot code imports an R&D certificate (filename R&D) and a primary protected application (filename PRIMAAPP); however, these are not mandatory subimages.

The KEYS, R&D, and PRIMAAPP subimages each contain their own specific header/structure.

Speed-up is done based on customer-specific OMAP peripheral register patch values in the header of the X-LOADER subimage.

Before the authentication and execution of the subimage, the magic number of the header must match a predefined value (0x809795A3).

When transferring an image via SSI, UART, or USB, the boot sequence is almost the same regardless of what is written to the TOC. The only difference is that the filename X-LOADER instead is 2ND.

The filenames used by the boot code are listed in Table 2–13 and the magic numbers are listed in Table 2–14.

Table 2–13. File Names

File Name	Description
X-LOADER	Code in flash started by the boot code
2ND	Code downloaded to the internal RAM and started from there by the boot code
KEYS	Contains all keys required to authenticate the software
R&D	Contains R&D information
PRIMAAPP	Reserved for future use

Table 2–14. Magic Numbers

Magic Number	Description
0x809795A3	Magic of the common header
0x2595D4EB	Magic of the secure header
0x16793A22	Magic of the speed-up

2.3.4.4 Speed-Up Data Structure

The speed-up data structure supports user-defined images to alter the contents of any memory-mapped location in OMAP73x before the security mode is entered. This is intended to allow users to change the peripherals that affect execution speed, such as DPLL and EMIF configuration, thus improving boot time.

Sequence of ADL Boot

The boot ROM code does not start the ADL software; it is started by the X-LOADER. The X-LOADER must have the knowledge to detect ADL requests either from a function parameter BOOT_MODE passed over by the boot ROM code and/or from a permission word in the customer specific area of internal RAM.

The boot mode can be:

- ❏ Normal boot
- ❏ Production boot
- ❏ ADL boot

One example of ADL installation and boot:

- ❏ OS is up and running.
- ❏ User connects UART and downloads an image file to SDRAM with an ADL install application, but also containing an ADL flash image.

- ❏ User runs the ADL installer application that programs the flash with the enclosed ADL flash image.
- ❏ User accepts ADL programming, then the ADL installer adds an ADL TOC item in the flash TOC and sets a permission word in the customer part of internal RAM.
- ❏ The user performs a power-on (cold) reset, and the UART receives the ADL boot command from the programming host.
- ❏ The ROM boot code passes an ADL boot mode parameter to the X-LOADER, that in its turn checks the permission word, authenticates and starts to execute the ADL sub-image in flash.
- ❏ ADL downloads a new OS image via UART and programs it to flash.
- ❏ User/software restarts the device (SW/PURX reset).

Note:

For devices where the user cannot perform a power-on reset without removing the battery, see below scenarios using software reset only.

More About ADL

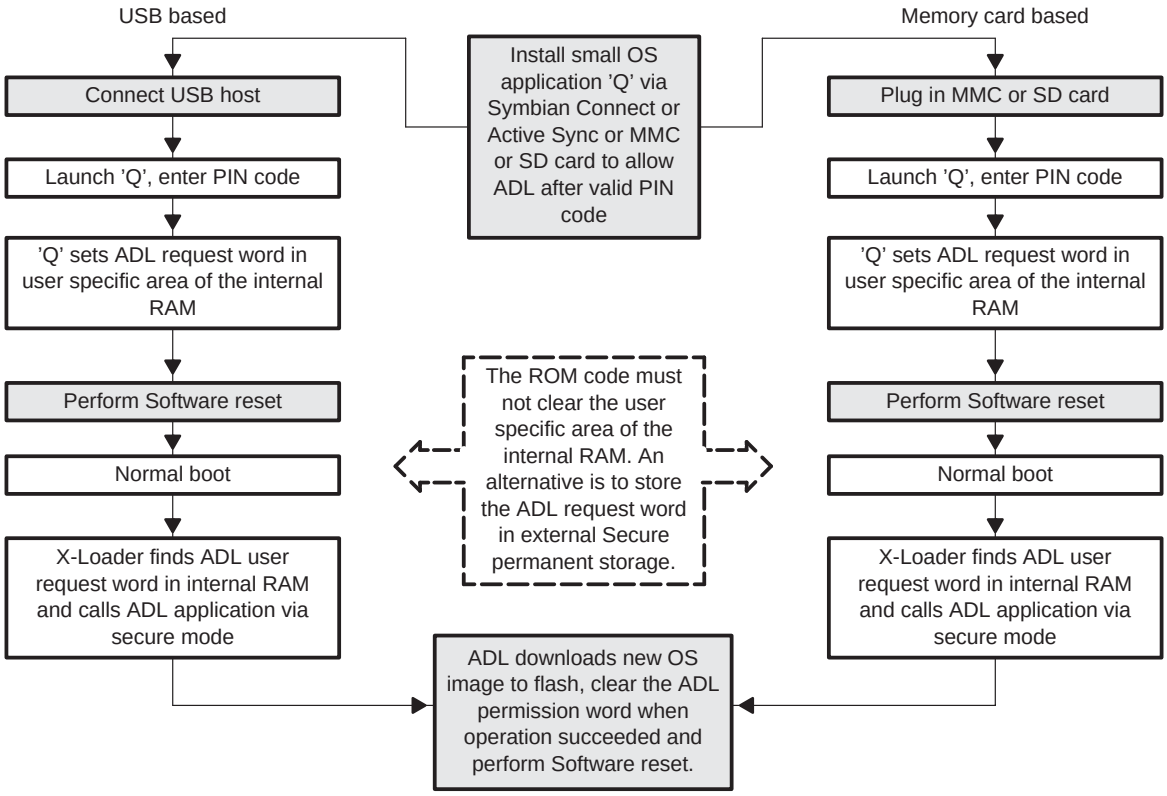
An ADL is implemented to enable other interfaces, not supported by the boot code, to be used for flashing e.g. MMC or IRDA. The ADL application may be used for partial flashing on field or in service, when only part of a flash memory is to be updated.

The ADL is an independent application, which may be activated

- ❏ By the serial interface ADL booting command after a power-on reset
- ❏ From the OS by writing a valid ADL status to the internal RAM and restarting the device with a software reset.

Figure 2–9 show scenarios that assume that an ADL subimage already is programmed in flash and that the starting point is a running OS environment.

Figure 2–9. ADL Examples without ADL Boot Command



These scenarios are hypothetical since, in the end, it is the responsibility of the OEM to implement ADL mechanisms to fit their needs.

We have not discussed the use of the secure environment hardware accelerators for a higher degree of safety during ADL, but it is worth mentioning that those hardware accelerators can be set accessible for non-secure environment code, though this is not the default setting.

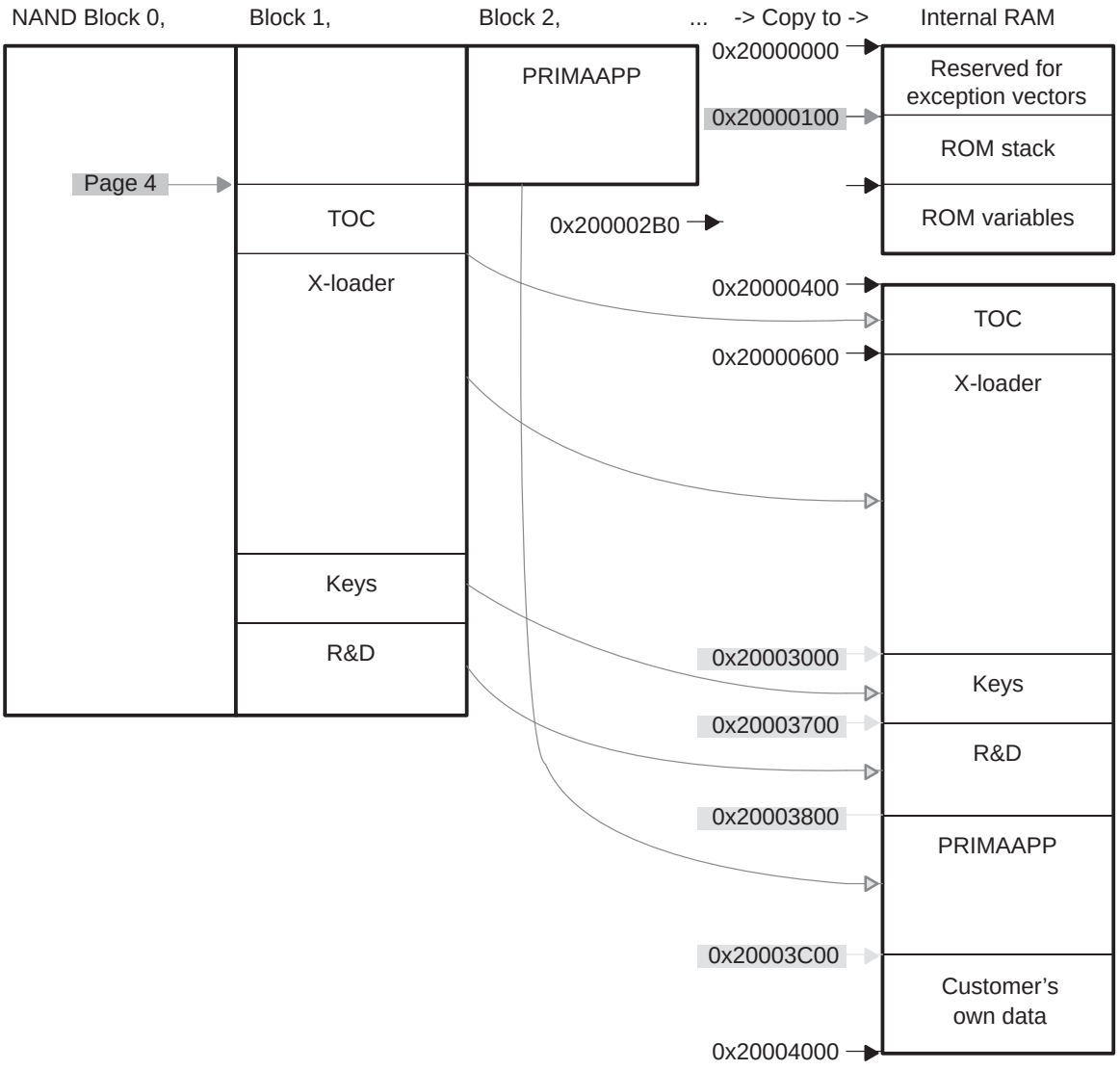
2.3.4.5 NAND TOC Example

The boot code reserves (four blocks ×16KB when pages/block is 32 or eight blocks × 8KB when pages/block is 16) 64KB from the beginning of the NAND. Blocks are reserved only from flash that is used for booting. The reserved block is called the *NAND boot sector*.

The boot sector must contain at least one TOC. The TOC and contents of subimages are copied to the internal RAM to address 0x20000400. This means that the maximum size of TOC + subimages is less than the available free space in the internal RAM (0x20004000 - 0x20000600 ≈ 14.8 KB). Data from the NAND is copied out to the RAM in 512-byte pieces. Customer data area must start at 512B boundary to avoid overwriting it when copying the NAND boot sector.

The ROM searches page by page for the TOC from the NAND until end of the NAND boot sector. This allows use of redundant copying of TOC and subimages in the boot sector (in practice up to four copies).

Figure 2–10. NAND TOC Example



2.3.4.6 NOR TOC Example

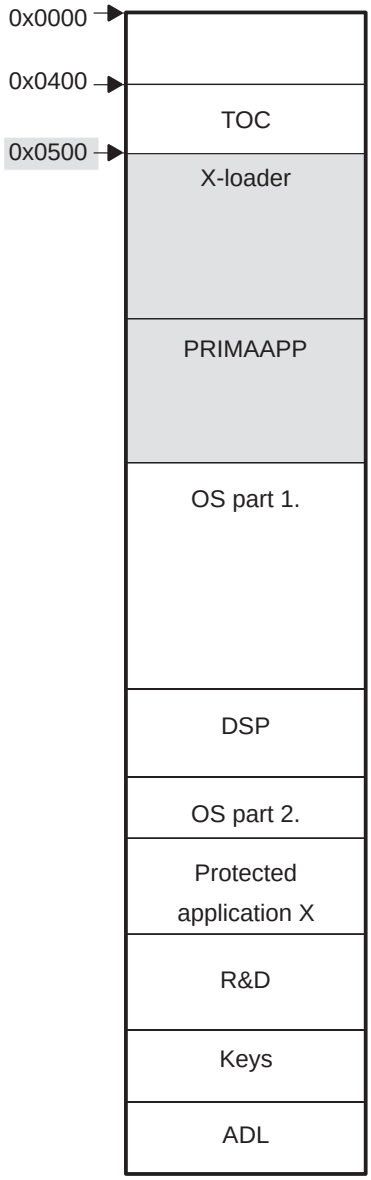
Subimages located in NOR flash are executed directly from the flash (XIP). Therefore, the X-LOADER size is not limited.

The TOC item X-LOADER is checked from offset 0x0100, and if the filename does not match, the next CS (NOR/NAND) is checked. NOR chip-selects are checked from 3 to 1.

The NOR TOC can contain other than boot supported items, because boot does not take care of the other items.

Items that are needed/supported by the boot code (see Table 2–13) must be located inside of first 16 TOC items, and any of the boot code supported items must not be filled by 0xFF.

Figure 2–11. NOR TOC Example



2.3.4.7 Security Images

Security images that are transferred to the secure RAM by the boot code are listed below according to the order of transfer.

1) Key certificate (named Keys\0). This must be presented if TOC is searched by the boot code.

2) R&D certificate (named RD Certifi\0). This is not mandatory to have, but if it is presented at TOC during a boot, the boot code transfers it to the secure RAM.

3) Primary protected applications (named PriProtAppl\0). This is not mandatory, but if it is present at TOC during a boot, the boot code transfers it to the secure RAM.

2.3.4.8 Implementation Details

Memory Usage

The following table describes used memory during the boot sequence. The memory layout limits usage of the internal RAM, so that the first 0x400 bytes are reserved to the boot code and the rest to the TOC + subimages. The TOC + subimages must not fill the rest of the memory; the end of the RAM can be used for vendor specific needs. The TOC + subimages space is used only in case the start-up is done from the NAND. When the start-up is done from the NAND, the KEYS subimage must always be at the same place in internal RAM.

Table 2–15. Memory Usage Of The Boot Code

Address Range	Description
0x00000000	ROM exception vectors
0x0000003C	ROM code section containing all the functional code
0x00003BF8	4-byte long checksum of the whole boot ROM
0x00003BFC	4-byte long ROM versioning information
0x00003C00	Security entry functionality is implemented to this specific range of the ROM.
0x00003C78	Constant and initial data of ROM
0x00003F60	Jump table
0x00004080	End of ROM
0x00004084	Security entry function at secure ROM (only address to the entry)
0x20000000	RAM exception vectors
0x200000D0	Reserved for stack (stack size 480B)
0x200002B0	ROM code variables
0x200003D8	Reserved for tracing
0x200003E0	Reserved for application_data variable
0x20000400	Reserved space for TOC + items. Maximum size is 15360B.
0x20004000	End of the internal RAM

2.3.4.9 Configuration

Clock

Table 2–16 shows the register values that are needed to enable the internal RAM clock in order to use C-code stack. The registers are written using an *OR* operation on the existing register contents.

Table 2–16. OMAP Clock Initialization Register Values

Register	Reset Mode 0 Values	Reset Mode 1 Values
CLKRST_ARM_IDLECT1	0x00000806	0x00000806
CLKRST_ARM_IDLECT3	0x00000011	0x00000011
CLKRST_ARM_RSTCT2	0x00000001	0x00000001

EMIFS

The EMIFS is configured to Table 2–17 values after a reset.

Table 2–17. EMIFS Configuration Values

Register	Reset Mode 0 Values	Reset Mode 1 Values
EMIFS_CS_CFG_ 1 to 3	0xFF80FFF3 0xFFC0FFF3 See Note 1.	0xFFC0FFF3 See Note 2.
EMIFS_ADV_CS_CFG_ 1 to 3	0x00000002	0x00000002

Notes: 1) EMIFS CS configuration depends on the state of the GPIO 1 (8th bit of the CONF_GPIO_1_STAT_R). The pin must be set to low when muxed NOR is connected and to high state for non-muxed NOR.

2) Reset mode 1 does not support non-muxed NOR at boot.

PIN Multiplexing

PIN multiplexing is not configured at reset mode 1, but for reset mode 0 changes must be made.

PIN configuration is required either to enable external access for the NAND controller and provide access to multiplexed NOR flash or to set up the EMIFS to use only nonmultiplexed NOR flashes.

PIN configuration is also needed to use UART3 signals for flashing.

MMU, Data, and Instruction Cache

The MMU is disabled after every reset. The configuration values of the MMU and caches (CP15 register 1) are detailed in Table 2–18.

Table 2–18. MMU Configuration Register (CP15)

Bit Position	Value	Acronym	Description
[0]	0	M	MMU disabled
[1]	0	A	Alignment check disabled
[2]	0	C	Data cache disabled
[3]	1	-	Should be one (SBO)
[4]	1	-	(SBO)
[5]	1	-	(SBO)

Table 2–18. MMU Configuration Register (CP15) (Continued)

Bit Position	Value	Acronym	Description
[6]	1	-	(SBO)
[7]	0	B	Endian mode is set to little endian.
[8]	1	S	Protection bit is ignored because MMU is not ON.
[9]	1	R	Protection bit is ignored because MMU is not ON.
[10]	0	-	Should be zero (SBZ)
[11]	0	-	(SBZ)
[12]	1	I	Instruction cache is enabled.
[13]	0	V	Exceptions are handled at beginning of 0x00000000 address.
[14]	0	RR	Normal cache replacement strategy (not used)
[15]	1	L4	MPU to thumb mode by BX command support enabled
[16:31]	X	-	Use existing values.

Exceptions (Interrupts)

Reset and IRQ are the only exceptions that are implemented in the boot code. Other exceptions during boot code execution generate a global software reset.

Other exception vector contents are fetched from the internal RAM. Other than for reset and IRQ, the stacks are not initialized by the boot code. The IRQ exception uses previous mode stack so that it continues stack usage from where it was at previous mode.

Table 2–19 details which internal RAM addresses have jump addresses to each interrupt service routine. ISRs are called in MPU mode.

Table 2–19. RAM Exception Jump Table

Address	Exception
0x20000000	Undefined instruction
0x20000004	SWI
0x20000008	Prefetch abort
0x2000000C	Data abort
0x20000010	Reserved
0x20000014	IRQ
0x20000018	FIQ

The reserved space for the exception jump table is 0xF8 bytes, so there is the possibility to add a small FIQ handler starting from the FIQ address.

Reset Reason Reading

Reset reason registers ULPD_RESET_STATUS and ARM_SYSST are read and cleared by the boot code. Only the ARM_SYSST (OMAP3.2) register is used to detect the reset reason.

Both registers are stored to the application_data->reset_reason variable on the memory. The reset reasons variable is a table that contains two 32-bit indexes: index 0 contains the value of the RESET_STATUS register of the PCC module (external reset) and index 1 contains the value of the ARM_SYSST register (OMAP3.2 reset) of the CLKRST module.

2.3.4.10 Speed-Up

The speed-up data structure in the image common header allows the ROM code to import and deploy user-defined values to alter the contents of any memory-mapped location in the OMAP device before the security mode is entered. This is meant to allow users to change the peripherals that affect execution speed, such as DPLL and EMIF configuration, and thereby improve boot time.

Speed-up is not done if the reset reason was a software reset . The boot code initially checks a field called MAGIC. If MAGIC is not valid then speed-up is not done either.

The field called MASK is the bit mask for the configurations that are required. The mask values are listed in Table 2–20.

Table 2–20. Speed-Up Mask Content

Mask Bit/0x	Function
00000001	Write DATA[00] [1] to address of DATA[00] [0]
00000002	Write DATA[01] [1] to address of DATA[01] [0]
00000004	Write DATA[02] [1] to address of DATA[02] [0]
00000008	Write DATA[03] [1] to address of DATA[03] [0]
00000010	Write DATA[04] [1] to address of DATA[04] [0]
00000020	Write DATA[05] [1] to address of DATA[05] [0]
00000040	Write DATA[06] [1] to address of DATA[06] [0]
00000080	Write DATA[07] [1] to address of DATA[07] [0]
00000100	Write DATA[08] [1] to address of DATA[08] [0]
00000200	Write DATA[09] [1] to address of DATA[09] [0]
00000400	Write DATA[10] [1] to address of DATA[10] [0]
00000800	Write DATA[11] [1] to address of DATA[11] [0]
00001000	Write DATA[12] [1] to address of DATA[12] [0]
00002000	Write DATA[13] [1] to address of DATA[13] [0]
00004000	Write DATA[14] [1] to address of DATA[14] [0]
00008000	Write DATA[15] [1] to address of DATA[15] [0]
00010000	Write DATA[16] [1] to address of DATA[16] [0]
00020000	Write DATA[17] [1] to address of DATA[17] [0]
00040000	Write DATA[18] [1] to address of DATA[18] [0]
00080000	Write DATA[19] [1] to address of DATA[19] [0]
00100000	Write DATA[20] [1] to address of DATA[20] [0]
00200000	Write DATA[21] [1] to address of DATA[21] [0]
00400000	Write DATA[22] [1] to address of DATA[22] [0]
00800000	Write DATA[23] [1] to address of DATA[23] [0]
01000000	Write DATA[24] [1] to address of DATA[24] [0]
02000000	Write DATA[25] [1] to address of DATA[25] [0]
04000000	Write DATA[26] [1] to address of DATA[26] [0]
08000000	Write DATA[27] [1] to address of DATA[27] [0]
10000000	Poll register DATA[28] [0], rules in poll[0]
20000000	Poll register DATA[29] [0], rules in poll[1]
40000000	Poll register DATA[30] [0], rules in poll[2]
80000000	Poll register DATA[31] [0], rules in poll[3]

The DATA[0..27][0] field contains a register address, and DATA[0..27][1] contains the data to be written to that address.

The poll field contains rules on how to poll registers; address and content are stated at DATA[28][x]–DATA[31][x]. The register is read as long as defined by the poll field in Table 2–21.

Table 2–21. Speed-Up Poll Register Content

Value	Meaning
0x01	Register must contain value that is stated at poll.
0x02	Register bit: Position stated at poll must be set to one.
0x03	Register bit: Position stated at poll must be set to zero.

The speed-up structure is as follows:

```
typedef struct {
  dword magic;
  union {
    dword spare[67];
    OMAP730_SPEEDUP_STR omap730;
  } data;
} SPEEDUP_STR;
typedef struct {
  dword mask;
  dword data[32][2];
  byte  poll[4];
  byte  spare[4];
} OMAP730_SPEEDUP_STR;
```

2.3.4.11 Tracing

The boot code trace implementation is simple and small. The purpose of tracing is to store the major states of the boot code to a defined memory address.

The trace functionality is as follows:

1) Trace output address is read from the address.
2) Trace byte is written to a defined address. By default the address is set to +4B, which can be changed to, for example, UART1 (0xFFFB0000) or UART3 (FFFB9800).

Definition of the trace bytes is outside the scope of this document.

2.3.5 Secure Environment at Boot Time

OMAP73x devices provide a security mechanism in internal boot ROM, which at boot time requires the successful verification of external code prior to its execution. The external code can be executed at two times:

- During the flashing process
- During boot from external flash

The same verification process applies in both situations. In the following sections, for the sake of simplicity, this verification is referred to as the *external code verification mechanism*.

The OMAP73x emulation devices can boot either from internal boot ROM or external memory. In the latter case, the whole secure architecture is inaccessible; hence, authentication and integrity checks are not performed. When emulation devices boot from the internal boot ROM, the same security scheme applies as for normal devices.

2.3.5.1 Signed Image

Images are all signed with respect to the structure formats. The public key header and the key signature, generated with the root private key, are both encapsulated in the SEC_ROM_PUBLIC_KEY_HEADER structure.

Common header data and header signature, generated with the OEM1 private key, are both encapsulated in the COMMON_HEADER structure. The common header data also contains the hash (digest field) of the code to be executed.

2.3.5.2 External Code Verification Mechanism

The external code verification (authentication and integrity check) is done in three sequential steps in secure ROM:

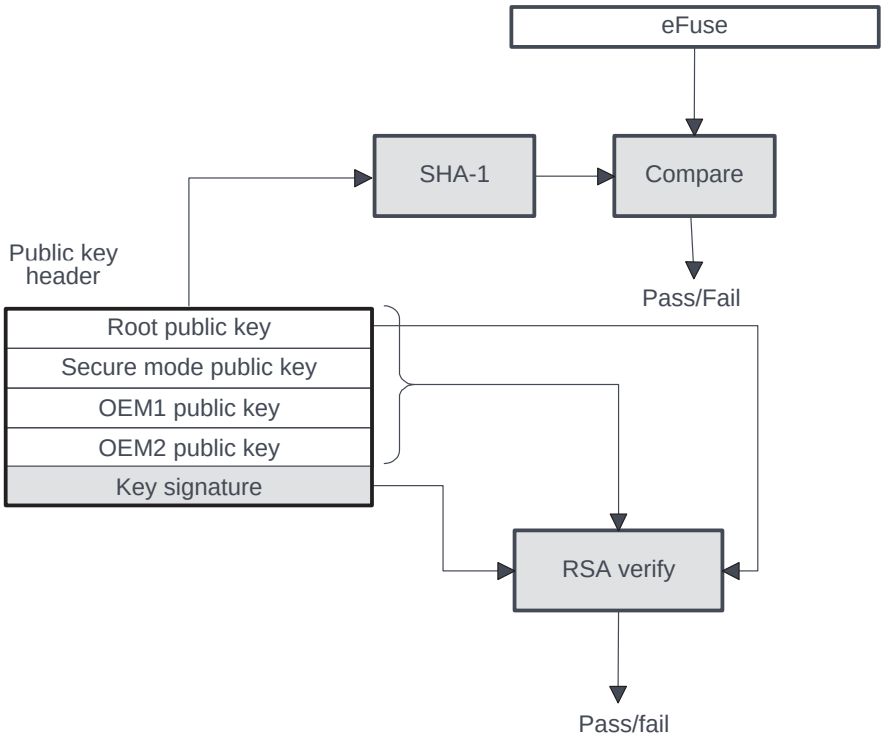
1) Keys verification
2) Header verification
3) Image verification

If one of the three steps is not successfully passed, a security violation results and the boot ROM automatically exits the booting or flashing procedure by reset.

Keys Verification

Key verification is done in two steps. First, the root public key present in the public key header is hashed and the digest is compared to the eFuse content (B-Key). Then, if both values match, RSA verification is done on the entire public key header.

Figure 2–12. Keys Verification



Header Signature Authentication

Header signature authentication is done by executing RSA verification on the common header using OEM1 public key.

Figure 2–13. Header Verification

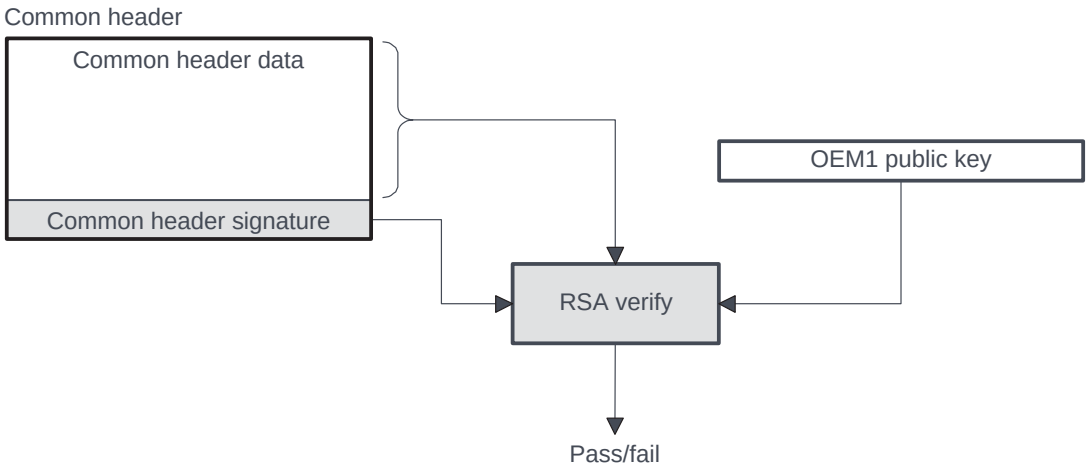
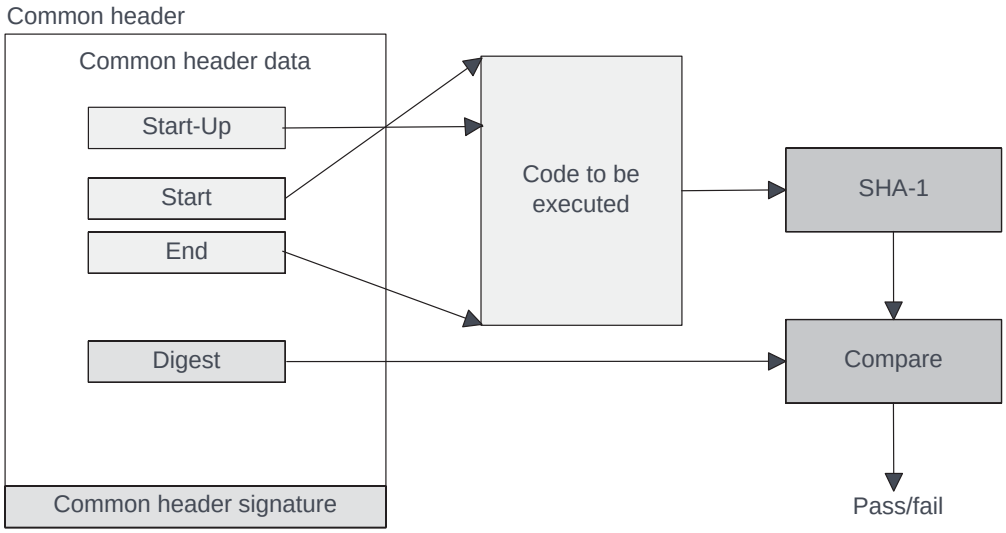


Image Authentication

The application hash information present in the common header includes pointers to the start and end addresses of the image.

Authentication is done by comparing the hash of the code section, delimited by start and end pointers, with the digest (hash) present in the common header. After successful authentication, execution is started at the address pointed out by the start-up value. There is flexibility in executing hash only on a subsection of the code with start and end values, but the start-up address must be within the authenticated code.

Figure 2–14. Image Authentication



Execution

After successful authentication, the external code is executed. For flashing and booting from a NAND flash, the flash loader and the bootstrap are downloaded within the internal RAM and executed from there. For booting from a NOR flash, code is directly executed from the NOR flash.

2.3.5.3 Header Of Executable Image

This section describes the header content each of the executable subimages requires. The header provides speed-up data, authentication information, and entry address to executable code.

First, the header is authenticated based on header signature (HEADER_SIGNATURE). Part of the spare area from the data item (COMMON_HEADER_DATA） can be left out of the authentication. The size of the authentication is specified by signed_header_size variable.

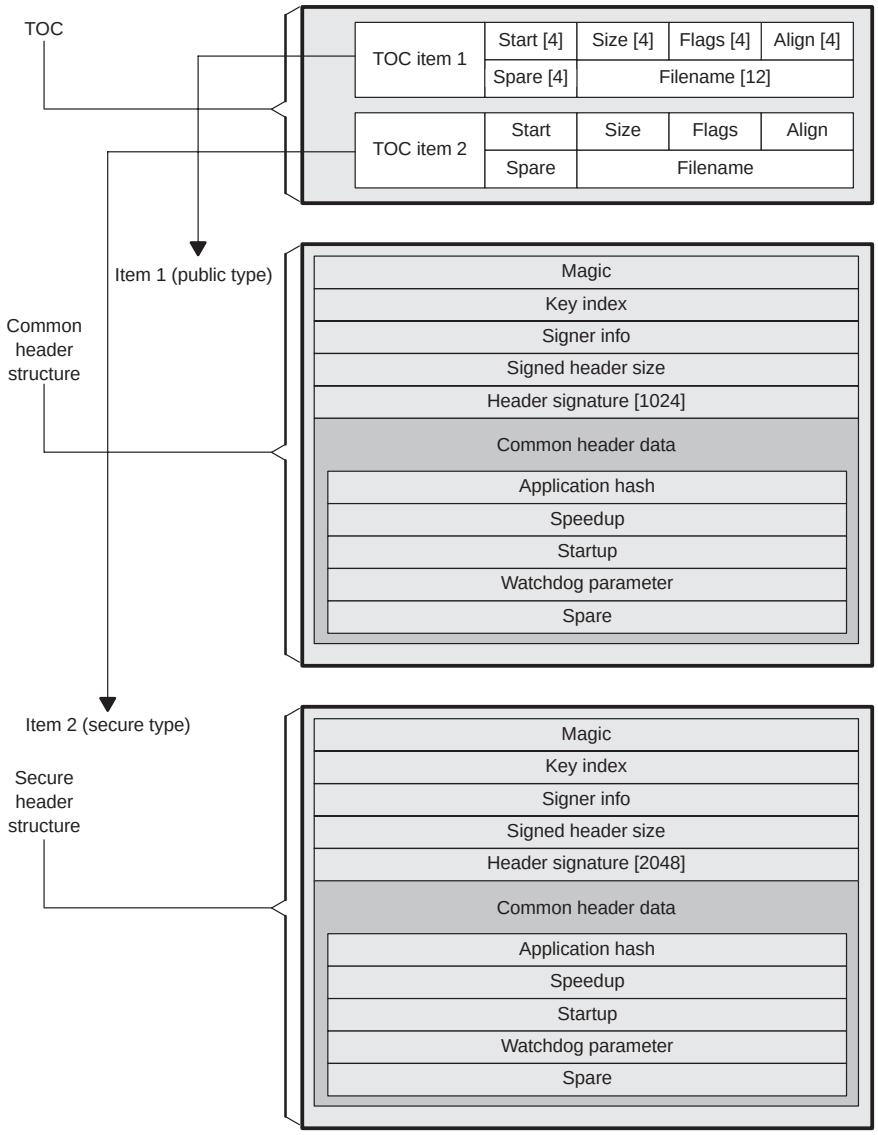
Executable subimages that are started by the boot code can be X-LOADER or 2ND. Other subimages are started by X-LOADER.

The magic number of the header is described in Figure 2–15.

The last item in the header is called data. The item is described in Section 2.3.5.4.

Figure 2–15 shows the image header structure. Example 2–1 details the header structure.

Figure 2–15. Image Header Structure



Example 2–1. COMMON_HEADER

```
typedef struct {

dword magic;

dword key_index;

dword signed_header_size;

SIGNER_INFO signer;

COMMON_HEADER_DATA data;

Byte header_signature[HEADER_SIGNATURE_LENGTH];

} COMMON_HEADER;
```

2.3.5.4 Data

The data item contains authentication information (fields APPLICATION_ HASH and SIGNED_HEADER_SIZE), speed-up data, entry point to an application, and security watchdog update information; the last item (spare1) is reserved for user data.

Example 2–2. COMMON_HEADER_DATA

```
typedef struct {

SEC_ROM_HASH_ARRAY_STR application_hash;

SPEEDUP_STR speedup;

dword startup /* void (*startup)(dword, dword, dword,
dword);*/

dword watchdog_param;

dword spare1[128];

} COMMON_HEADER_DATA;
```

Security on Header

The APPLICATION_HASH contains information on the start and end addresses of the authenticated image. The hash of authenticated area is located also in the structure.

A second structure SEC_ROM_HASH_ARRAY_INFO contains information about whether the subimage is one executable or whether is it split into several parts. Subimages that the boot code uses must be authenticated in one part. One reason to split the image into several parts could be, for example, when the executable image does not fit in one flash.

```
typedef union sec_rom_hash_array_str {

SEC_ROM_HASH_AREA hash;

SEC_ROM_HASH_ARRAY_INFO info;
```

```
} SEC_ROM_HASH_ARRAY_STR;
typedef struct
{
dword start;
dword end;
byte  hash[20];
} SEC_ROM_HASH_AREA;
typedef struct sec_rom_hash_array_info {
dword zero0;
dword zero1;
HASH_INFO_TYPE   type;      /*   END_OF_LIST,   FLASH_SIZE,

                          NEXT_LIST */
union {
    dword size;
    union sec_rom_hash_array_str * next_list;
} attribute;
} SEC_ROM_HASH_ARRAY_INFO;
```

Single Physical Memory Configuration

In case of a subimage that contains several hash areas (but supports only a single physical memory configuration), the array of SEC_ROM_HAS_ARRAY_STR unions consists of one or more SEC_ROM_HAS_AREA structures.

After the last SEC_ROM_HASH_AREA, there is a SEC_ROM_HASH_ARRAY_INFO structure with the field type having the value END_OF_LIST.

Several Physical Memory Configurations

When one subimage supports several physical memory configurations, the authentication data that is available in the image must be calculated for each configuration separately.

One hardware configuration array starts with one or more SEC_ROM_HASH_ARRAY_INFO structures with the field type having the value FLASH_SIZE.

The field SIZE of the union attribute tells the physical size of the first flash device for this configuration.

After the FLASH_SIZE information structures, one or more SEC_ROM_HASH_AREA structures follow.

After the last SEC_ROM_HASH_AREA structure comes either another hardware configuration (filed type with value FLASH_SIZE) or on end-of-list indicator (field type with value END_OF_LIST).

Table 2–22. Device Overview

Device	Detection Sequence			NAND Supported by ROM Software Driver on EMIFS
	Non Muxed	**Muxed**	**NFC**	
OMAP730 ES1.0	NOR: CS3 | CS2 | CS1	NOR: CS3 | CS2 | CS1	NAND: CS0 | CS1 | CS2 | CS3	Not available
OMAP730 ES1.1	NOR: CS3	NOR: CS3 NAND: CS		CE don't care – 8/16 bits – 1.8 V/3.0 V Page program: (2K+ 64)byte/(1K+32)word The first block is ensured to be a valid block and does not require error code correction (minimum size 15KB).
				CE don't care – 8/16 bits – 1.8 V/3.0 V Page program: (2K+ 64)byte/(1K+32)word (512+16)byte/(256+8)word The first block is ensured to be a valid block and does not require error code correction (minimum size 15KB).

The boot ROM code includes a NAND_EMIFS driver.

In order to detect NOR and NAND memories connected on EMIFS and NAND flash controller, the following methods are used:

- ❑ NOR bootstrap research process: *(Not using NOR READ_ID anymore)*

  To avoid destructive writes in cases where devices other than NOR flashes might be connected to EMIFS CS3, the so-called bootstrap research for NOR flash is done. This means checking whether there is a first TOC item starting at address 0x400 with a filename field containing the subimage filename X-LOADER. NOR flashes are in read array mode at power up.

- ❑ NAND READ_ID_EMIFS process:

1) Write value 0x90 at address 0x0002.
2) Write value 0x00 at address 0x0004.
3) Read value at address: 0x0000.

- ❏ NAND READ_ID_NFC process:

1) Send Read_ID command to NFC.
2) Send address 0x00 to the NFC.
3) Read value from the NFC (Device_ID).

NAND Flash Supported on EMIFS/NAND Flash Controller

On EMIFS:

- ❏ CE don't care – 8/16 bits – (1.8 V, 3.0 V)
- ❏ Page program: 8-bit device: (2K + 64) byte
  16-bit device: (1K + 32) word
- ❏ The first block is ensured to be a valid block and does not require error code correction (minimum size 15K bytes)

Corresponding flashes:

NAND Samsung - CE don't care - 8/16 bits - 1G & 2G - TSOP:

(K9F1G08U(Q)0M - K9F1G16U(Q)0M -K9K2G08U(Q)0M - K9K2G16U(Q)0M)

On NAND flash controller: DI = Device ID
8 bits

Page program: 528 bytes

32M bytes	(1.8 V)	DI=E3h
64M bytes	(1.8 V, 3.0 V)	DI=39h/E6h
128M bytes	(1.8 V, 3.0 V)	DI=33h/73h
256M bytes	(1.8 V, 3.0 V)	DI=35h/75h
512M bytes	(1.8 V, 3.0 V)	DI=36h/76h
1G bytes	(1.8 V, 3.0 V)	DI=78h/79h

Detection Sequence

RESET_MODE_0: (EMIFS in nonmultiplexed mode)

- ❏ NOR connected to EMIFS CS3 using NOR bootstrap research
- ❏ NAND connected to EMIFS CS3 using NAND READ_ID_EMIFS

RESET_MODE_0: (EMIFS in multiplexed mode)

- ❏ NOR connected to EMIFS CS3 using NOR bootstrap research
- ❏ NAND connected to NAND flash controller CS0-3 using NAND READ_ID_NFC

RESET_MODE_1: (EMIFS in multiplexed mode)

- NOR connected to EMIFS CS3-CS2-CS1 using NOR bootstrap research
- NAND connected to NAND flash controller CS0-3 using NAND READ_ID_NFC

Restrictions Concerning Image on NAND Connected to EMIFS

Bootloader must be placed in the first block of the NAND flash; valid TOC must be mapped at the beginning of a page, in the first:

- 512-byte boundary for NAND flash, 8 bits
- 256-word boundary for NAND flash, 16 bits

Booting Process

The booting process contains two parts:

- First: Flash detection (NOR and NAND detection)
- Second: Bootstrap research on detected flash

RESET_MODE_0:

- Bootstrap NOR research on EMIFS CS3

If no bootstrap found on NOR:

- NAND detection on EMIFS CS3 (if EMIFS in non-muxed mode)

or

- NAND detection on NFC CS0-3 (if EMIFS in muxed mode)
- Bootstrap research on NAND

RESET_MODE_1:

- NAND detection on NFC CS0-3
- Bootstrap NOR research on EMIFS CS3-CS2-CS1
- Bootstrap research on NAND (if no bootstrap found on NOR)

USB Bootloading

The USB boot driver is implemented in ES1.1 but is not fully functional.

USB bootloading functionality is implemented in reset mode 0 only. The following USB port configuration and USB configuration time chapters apply to reset mode 0 only.

- USB port configuration

USB bootloading functionality is used in port0/internal transceiver configuration.

- ❏ USB configuration time

200-ms time for polling download interface is not sufficient in case of USB usage as download interface because of the additional requirement for the USB host enumeration/configuration time [see the USB1.1 specification, Chapter 9].

An additional “wait 500 ms for enumeration/configuration” phase is added by the USB driver in case of USB host attachment detection to enable USB enumeration/configuration to complete.

To avoid the 500-ms additional boot time (USB configuration time) in cases when USB is not to be used for bootloading, the VDSSHV2 pin (ball AA2) must not be powered.

The USB driver, after detecting USB host attachment (base on VBUS/ VDSSHV2 voltage), waits for enumeration/configuration phases to be completed within 500 ms (see the pseudocode below).

```
PSEUDOCODE:

IF (attached)

   wait 500ms for enumeration/configuration completion

                  OR download interface polling response
received

              by either SSI or UART download interface

ENDIF
```

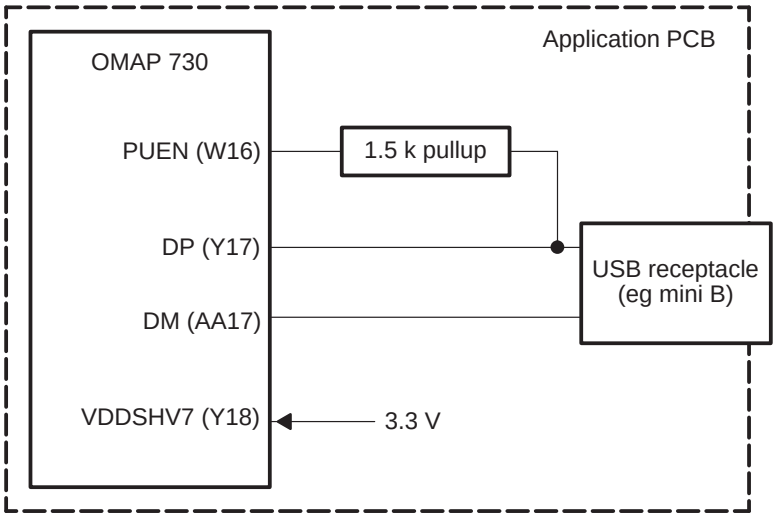
There is no impact if UART is used as download media. In the case when download interface polling response is received (by UART), the wait for enumeration/configuration phase is abandoned.

2.3.6 USB Boot Configurations

Power-on Reset Conditions: USB PORT0

- ❏ The built-in transceiver must receive a 3.3-V supply on VDDSHV7.
- ❏ The USB pullup must either be connected by default or controlled by the active-high USB_PUEN pin (shown in Figure 2–16—serial resistors and transient suppressors on USB lines not shown).
- ❏ The USB built-in transceiver I/O lines must be connected to a USB connector receptacle (accepting B or mini-B plugs)

Figure 2–16. Configuration Schematics

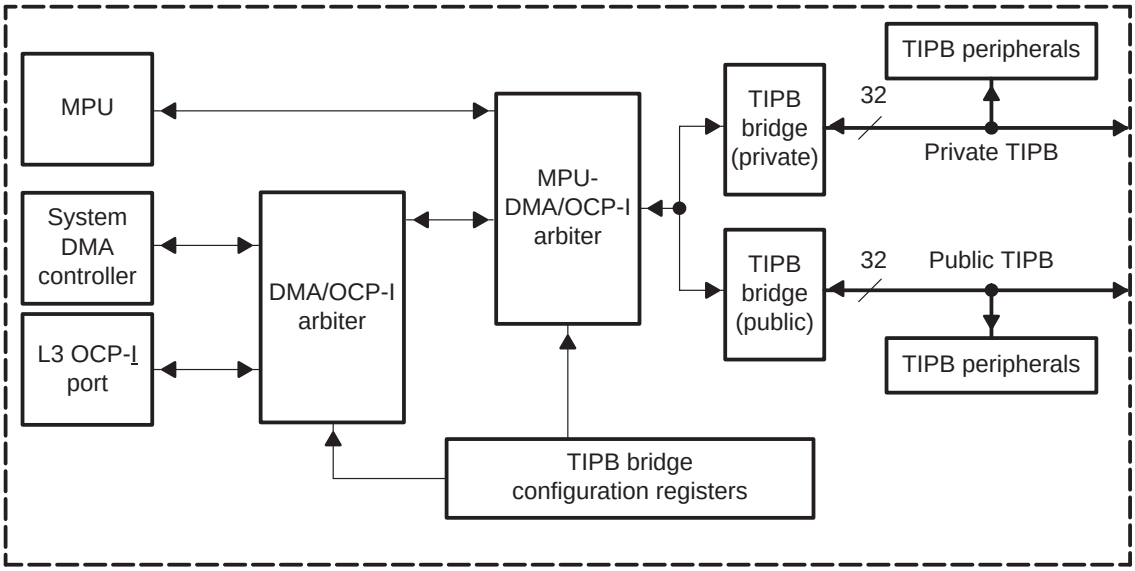


2.4 TIPB Bridge

The TIPB bridge allows the TIPB and all peripherals connected to it to be shared with three hosts: the MPU, the OCP initiator (OCP-I), and the system DMA controller. The TIPB controls accesses to avoid conflicts between the hosts, and it allows the MPU to configure the protocol parameters of the TIPB.

The TIPB bridge module is shown in Figure 2–17.

Figure 2–17. OMAP 3.2 Platform TI Peripheral Bridge



2.4.1 Functionality

This section describes the functionality of the TIPB bridge.

2.4.1.1 Bus Allocation

The TIPB is shared among the MPU memory interface, the OCP initiator (OCP-I), and the DMA controller. Two levels of bus allocation are used to resolve conflicts and prioritize accesses among the three requestors.

The first level of the two-level arbitration process selects between the DMA and OCP-I for control of the TIPB. Fixed- or round-robin priority schemes can be programmed in the FIXNROUND_PRIORITY bit field of the TIPB allocation control register, RHEA_BUS_ALLOC. When fixed priority is selected, the EXTNINT_PRIORITY bit field determines whether DMA or OCP-I has fixed priority.

The second level of the two-level arbitration process selects between the MPU and the DMA/OCP-I. The value programmed in the RHEA_PRIORITY bits of RHEA_BUS_ALLOC defines the priority. If the value is 0, the MPU memory interface has priority over the DMA/OCP-I. If the value equals *n* (where *n* is from 1 to 7), the DMA/OCP-I has priority over the MPU and it can perform *n* accesses before giving the priority back to the MPU memory interface.

2.4.1.2 Access Permissions

Access permissions to the MPU public and private TIPBs vary with initiator and target. The MPU has unconditional access permissions to the MPU public and private TIPBs.

The system DMA has unconditional access permission to the MPU public TIPB and conditional, software-controlled access permission to the MPU private TIPB. System DMA permission for the MPU private TIPB is programmed in ACCESS_CNTL.

The OCP-I has conditional, software-controlled access to both the MPU public and private TIPBs. See Section 2.5, *Memory Interface Traffic Controller,* for details on programming OCP-I access permissions.

2.4.1.3 TIPB Strobes and Access Factor

The TIPB strobes are an integral part of the TIPB bridge. The TIPB strobes are (active-low) signals output from the TIPB bridge module that drive the peripheral interfaces of the MPU TIPB. The TIPB bridge uses two strobes (strobe 0 and strobe 1), and each strobe is fixed to control access to distinct ranges of the MPU TIPB peripherals address space. In addition, address spaces assigned to strobe 0 and strobe 1 are further segmented into areas of MPU public TIPB and MPU private TIPB memory space. See Chapter 24, *Memory Mapping*, for details on strobe address space assignments.

The TIPB bridge is required to communicate with peripherals of varying speeds. To allow slow peripherals to answer, it is possible to lengthen the strobe 0 and strobe 1 periods using ACCESS_FACTOR0 and ACCESS_FACTOR1 in RHEA_CNTL. By programming ACCESS_FACTOR0 and ACCESS_FACTOR1 to *n*, the respective strobe stretches its access over 2**n* TIPB bridge clock cycles (*n* cycles the strobe is inactive high; *n* cycles the strobe is active low).

The TIPB bridge clock referenced here is the input clock reference to the TIPB bridge module generated from the clock and reset-management module. The clock rate for the TIPB bridge is fixed to the same value as that of the OMAP 3.2 traffic controller module. For details, see Chapter 5, *Clock Generation and Reset Management*.

2.4.1.4 MPU Posted Write

The MPU can perform a posted write. When a posted write is enabled inside ARM_RHEA_CNTL, data sent by the MPU is buffered in the TIPB bridge module and the MPU can continue accessing other locations. The bridge handles the access to the TIPB peripheral so that the MPU is not stalled during the access.

If the MPU/system DMA/OCP-I performs another TIPB operation when there is a posted write, this operation must wait until the posted write is complete. If the system DMA or OCP-I performs a read operation to the same address as the posted write, the posted write data is not forwarded to the DMA or OCP-I. Posted write is not supported for DMA and OCP-I accesses.

Using ARM_RHEA_CNTL, you can enable posted write independently for strobe 0 or strobe 1 address spaces. This provides some flexibility over which MPU TIPB peripherals are configured for posted write. See Chapter 24, *Memory Mapping*, for details on strobe address space assignments.

2.4.1.5 Time-Out

A TIPB access time-out limits the maximum time a peripheral can stall the processor. When starting an access on the TIPB, the time-out counter is loaded with the value programmed in the TIMEOUT bits of RHEA_CNTL. If the current access is not finished when the counter reaches 0, the cycle is aborted and an abort is generated to the MPU/system DMA/OCP-I.

The time-out value is calculated by

$t_{time\text{-}out} = (1/f_{bridge\_clk}) \times (RHEA\_CNTL.TIMEOUT + 1)$

where f_{bridge_clk} is the traffic controller clock frequency setting.

The time-out can be used in conjunction with the posted write. The counter begins counting down when the posted write transaction has been scheduled, and this count continues against the posted transaction even if another transaction to the TIPB bridge occurs.

The time-out can be disabled using the TIMEOUT_EN bit in ENH_RHEA_CNTL.

2.4.1.6 Debug

Debug registers are saved on the occurrence of an MPU_TIPB abort. Abort can be caused by a time-out or by a size mismatch between the access word width and the word width of the addressed peripheral.

The access address, data, and error flags are saved to DEBUG_ADDRESS, DEBUG_DATA_LSB, DEBUG_DATA_MSB, and DEBUG_CTRL_SIGNALS.

2.4.2 Registers

All TIPB bridge configuration and debug registers are 16-bit registers. Write accesses to all TIPB registers can be performed only in MPU supervisor mode. Read accesses can be performed in MPU supervisor or user modes.

Table 2–23 lists the TIPB registers. Table 2–24 through Table 2–32 describe the register bits.

Table 2–23. TIPB Registers

Register	Description	Offset (Byte)
RHEA_CNTL	TIPB control	0x00
RHEA_BUS_ALLOC	TIPB allocation control	0x04
ARM_RHEA_CNTL	MPU TIPB allocation control	0x08
ENH_RHEA_CNTL	Enhanced MPU TIPB allocation control	0x0C
DEBUG_ADDRESS	Debug address	0x10
DEBUG_DATA_LSB	Debug data LSB	0x14
DEBUG_DATA_MSB	Debug data MSB	0x18
DEBUG_CTRL_SIGNALS	Debug control signals	0x1C
ACCESS_CNTL	Access control	0x20

Table 2–24. TIPB Control Register (RHEA_CNTL)

Bit	Name	Function	Reset Value
15:8	TIMEOUT	TIPB bus access time-out. When starting an access on TIPB bus, the time-out counter is loaded with this value. If the current access is not finished when the counter reaches 0, the cycle is aborted and abort indications are given to the peripheral and the MPU, DMA, or OCP-I. Maximum value for TIMEOUT is 255.	0xFF

Table 2–24. TIPB Control Register (RHEA_CNTL) (Continued)

Bit	Name	Function	Reset Value
7:4	ACCESS_FACTOR1	Clock period multiplication factor for TIPB strobe 1. Allows access to slow peripherals by lengthening TIPB strobe 1 period by a multiple of the internal TIPB bridge clock period. The TIPB bridge clock period is the same as the OMAP traffic controller clock period. 0000: Same as 0001 0001: Strobe period = TIPB bridge clock period 0010: Strobe period = TIPB bridge clock period x 4 0011: Strobe period = TIPB bridge clock period x 6 0100: Strobe period = TIPB bridge clock period x 8 0101: Strobe period = TIPB bridge clock period x 10 0110: Strobe period = TIPB bridge clock period x 12 0111: Strobe period = TIPB bridge clock period x 14 1000: Strobe period = TIPB bridge clock period x 16 1001: Strobe period = TIPB bridge clock period x 18 1010: Strobe period = TIPB bridge clock period x 20 1011: Strobe period = TIPB bridge clock period x 22 1100: Strobe period = TIPB bridge clock period x 24 1101: Strobe period = TIPB bridge clock period x 26 1110: Strobe period = TIPB bridge clock period x 28 1111: Strobe period = TIPB bridge clock period x 30	0x1

Table 2–24. TIPB Control Register (RHEA_CNTL) (Continued)

Bit	Name	Function	Reset Value
3:0	ACCESS_FACTOR0	Clock period multiplication factor for TIPB strobe 0. Allows access to slow peripherals by lengthening TIPB strobe 0 period by a multiple of the internal TIPB bridge clock period. The TIPB bridge clock period is the same as the OMAP traffic controller clock period. 0000: Same as 0001. 0001: Strobe period = TIPB bridge clock period 0010: Strobe period = TIPB bridge clock period x 4 0011: Strobe period = TIPB bridge clock period x 6 0100: Strobe period = TIPB bridge clock period x 8 0101: Strobe period = TIPB bridge clock period x 10 0110: Strobe period = TIPB bridge clock period x 12 0111: Strobe period = TIPB bridge clock period x 14 1000: Strobe period = TIPB bridge clock period x 16 1001: Strobe period = TIPB bridge clock period x 18 1010: Strobe period = TIPB bridge clock period x 20 1011: Strobe period = TIPB bridge clock period x 22 1100: Strobe period = TIPB bridge clock period x 24 1101: Strobe period = TIPB bridge clock period x 26 1110: Strobe period = TIPB bridge clock period x 28 1111: Strobe period = TIPB bridge clock period x 30	0x1

Table 2–25. TIPB Allocation Control Register (RHEA_BUS_ALLOC)

Offset: 0x04

Bit	Name	Function	Reset Value
15:6	Reserved		
5	EXTNINT_PRIORITY	Priority between DMA and OCP-I when fixed priority scheme is selected. 0: DMA has priority. 1: OCP-I has priority.	0
4	FIXNROUND_PRIORITY	Type of priority scheme used in DMA and OCP-I arbitration: 0: Round-robin scheme used 1: Fixed-priority scheme used	0
3	PRIORITY_ENABLE	0: TIPB bus allocation is done using the RHEA_PRIORITY bits. 1: MPU has the same priority as the DMA/OCP-I transfers regarding TIPB bus allocation when it is in exception mode (IRQ and FIQ).	1
2:0	RHEA_PRIORITY	Defines TIPB priority between MPU and DMA/OCP-I. 000: MPU has priority over the DMA/OCP-I. 001 through 111: DMA/OCP-I has priority over the MPU and can perform the programmed number of accesses before the MPU can access the bus.	001

Table 2–26. MPU TIPB Control Register (ARM_RHEA_CNTL)

Offset: 0x08

Bit	Name	Function	Reset Value
15:2	Reserved		
1	W_BUF_EN_1	0: Posted write buffer is bypassed. 1: Posted write buffer is enabled for MPU public and private TIPB peripherals having address space assigned to TIPB strobe 1.	0
0	W_BUF_EN_0	0: Posted write buffer is bypassed. 1: Posted write buffer is enabled for MPU public and private TIPB peripherals that have address space assigned to TIPB strobe 0.	0

Table 2–27. Enhanced TIPB Control Register (ENH_RHEA_CNTL)

Offset: 0x0C

Bit	Name	Function	Reset Value
15:4	Reserved		
3	MASK_ABORT	0: An abort signal is sent to the MPU whenever an MPU to_TIPB access is aborted. 1: The abort signal is not sent. MASK_ABORT does not mask/unmask DMA or OCP-I aborts.	1
2	Not Used	Not used in OMAP 3.2 (always 1). Legacy HIGH_FREQ mode from OMAP 3.0/3.1	1
1	MASK_IT	0: An interrupt is sent to the MPU whenever a TIPB write access (from MPU/DMA/OCP-I) is aborted or any TIPB access has a size mismatch. 1: The interrupt is masked.	1
0	TIMEOUT_EN	0: Do not enable the TIMEOUT feature. 1: Enable the TIMEOUT feature.	1

Table 2–28. Debug Address Register (DEBUG_ADDRESS)

Offset: 0x10

Bit	Name	Function	Reset Value
15:0	ADDRESS_DBG	Address from MPU memory interface; saved when an abort or access size mismatch occurs.	0xFFFF

Table 2–29. Debug Data LSB Register (DEBUG_DATA_LSB)

Offset: 0x14

Bit	Name	Function	Reset Value
15:0	DATA_DBG_LOW	Bits 15 to 0 of data bus from MPU. The value of the MPU data input is saved when a read access has a size mismatch, and the MPU data output bus is saved when a write access is aborted or has a size mismatch. If a read access is aborted, the value of this register is irrelevant.	0xFFFF

Table 2–30. Debug Data MSB Register (DEBUG_DATA_MSB)

Offset: 0x18

Bit	Name	Function	Reset Value
15:0	DATA_DBG_HIGH	Bits 31 to 16 of data bus from MPU. The value of the MPU data input bus is saved when a read access has a size mismatch, and the MPU data output bus is saved when a write access is aborted or has a size mismatch. If a read access is aborted, the value of this register is irrelevant.	0xFFFF

Table 2–31. Debug Control Signals Register (DEBUG_CTRL_SIGNALS)

Offset: 0x1C

Bit	Name	Function	Reset Value
15:11	Reserved		
10:9	HOST_ID	Host-ID that caused the abort 00: MPU 01: DMA 10: OCP-I 11: Invalid	00
8	BURST_ACC	Indicates single or burst access on the TIPB; saved when abort or access size mismatch occurs. 0: Single access 1: Burst access	0
7:6	DBG_PERHMAS(1:0)	Peripheral memory access size on TIPB; saved when abort or access size mismatch occurs. 00: 8 bits 01: 16 bits 1x: 32 bits	11
5:4	DBG_MAS(1:0)	Memory access size on TIPB; saved when abort or access size mismatch occurs. 00: 8 bits 01: 16 bits 1x: 32 bits	11
3	DBG_NSUPV	Indicates supervisor mode status of MPU; saved when abort or access size mismatch occurs. 0: Processor in supervisor mode 1: Processor not in supervisor mode	1
2	DBG_RNW	Indicates read or write transaction on the TIPB; saved when abort or access size mismatch occurs. 0: Write transaction 1: Read transaction	1
1	WR_SIZE_FLAG	Flag set to 1 when there is a mismatch between memory access size and peripheral memory access size. When read, the bit is reset to 0.	0
0	ABORT_FLAG	Flag set to 1 when TIPB access is aborted. When read, the bit is reset to 0.	0

Table 2–32. Access Control Register (ACCESS_CNTL)

Offset: 0x20

Bit	Name	Function	Reset Value
15:1	Reserved		
3	DPS_EN	0: Dynamic power-saving mode is disabled. 1: Dynamic power-saving mode is enabled. When DPS is enabled, the bridge clock is turned.	0
2	MASK_OCPI_NABORT	1: The abort for DMA access is masked before sending back to the DMA. 0: The abort for DMA access is sent back to the DMA. 1: The abort for OCPI access is masked before sending back to the OCPI.	0
1	MASK_DMA_NABORT	1: The abort for DMA access is masked before sending back to the DMA. 0: The abort for DMA access is sent back to the DMA.	0
0	DMA_ENABLE	1: DMA can access peripherals on the TIPB bridge. 0: DMA cannot access peripherals on the TIPB bridge.	1

2.5 Memory Interface Traffic Controller

This section discusses the OMAP730 memory interface traffic controller and its handling of system initiators and targets.

2.5.1 Description

The OMAP730 traffic controller (TC) is the central interconnect that manages all accesses between the following:

- OMAP internal initiators and target resources
- OMAP external initiators and target resources

The TC can have its own clock domain or be synchronous to the MPU clock domain. See Chapter 5, *Clock Generation and Reset Management,* for more details on clock domains. Typical performance in C035 is 100-MHz clock frequency. Accurate information can be found in the IC device documentation.

System initiators are:

- MPU. The MPU is connected to the TC using the MPU bus. The MPU can access memory devices or other type of targets connected to OCP-T1, OCP-T2, EMIFF, and EMIFS. Selection of the destination target is based on address decoding within the TC.
- Modem. The modem is connected to the TC via the DSP bus. The modem can access memory devices or other types of targets connected to OCP-T1, OCP-T2, EMIFF, and EMIFS. Selection of the destination target is based on address decoding within the TC.
- System DMA. The system DMA is connected to the TC with four dedicated ports: one for OCP-T1, one for OCP-T2, one for EMIFF, and one for EMIFS. It can access memory devices or other types of targets connected to any of these interfaces. Selection of the destination port is based on system DMA programming.
- USB VLYNQ initiator. An initiator can be connected to the TC via the OMAP OCP-I port. The external master can access memory devices or other types of targets connected to OCP-T1, OCP-T2, EMIFF, and EMIFS.

System targets are:

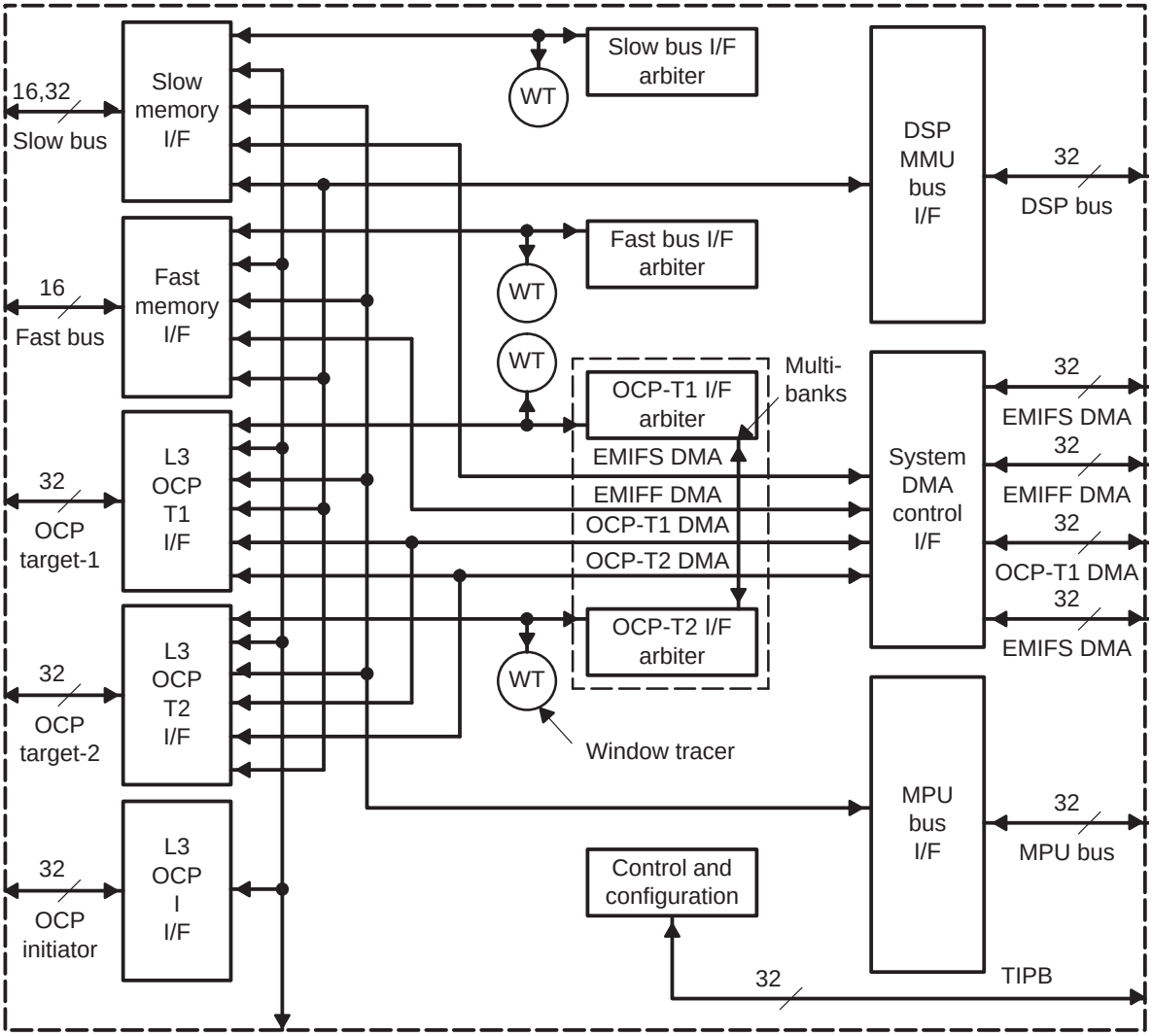
- EMIFS memory interface. This memory controller supports most common memory interface protocols through a flexible programming and timing control that also allows support of any type of internal or external IC target module.
  - External memory devices like asynchronous ROM, RAM, flash, or synchronous burst flash (16-bit).
  - Internal TI ACE ROM and RAM memories.
  - Internal and external asynchronous target modules
- EMIFF memory interface. This memory controller supports most common SDRAM interface protocols through a flexible programming and timing control.

- External synchronous standard single-data rate (SDR SDRAM) and low-power SDR SDRAM.

- OCP-T port interface. OMAP includes two ports on which OCP-compliant slaves can be connected. These two target ports are called OCP-T1 and OCP-T2. Attached target modules can be either internal or external memory subsystems or any type of target peripheral.

Figure 2–18 shows the traffic controller functional block diagram:

Figure 2–18. Traffic Controller Functional Block Diagram



2.5.2 OCP-T1/OCP-T2 Description

The OMAP 3.2 hardware provides two identical general-purpose ports for the connection of custom peripherals or memory subsystems. These ports can be accessed by all the initiators mentioned in the previous sections.

Each port arbitrates between all initiator requests and allows atomic transactions for the MPU, for the modem, and for the external initiator to implement

semaphore-based synchronization schemes between software tasks (running either on the same CPU or on two different CPUs).

Once granted, the selected request is converted from an OMAP internal protocol into an OCP-compliant transaction. As seen at the OMAP boundary, the OCP-Tx is an OCP master.

OCP-T1 and OCP-T2 have their own addressing spaces within the OMAP memory map (256M bytes accessible by OCP-T1 and 1.2G bytes accessible by OCP-T2) plus an additional common address range called the multibank address space (256M bytes). The multibank address space is used in some devices for connecting a dual-port OCP slave, such as a memory controller, that can provide concurrent data flow to/from memory banks.

A request that targets the multibank address space is analyzed and can be routed either to OCP-T1 or OCP-T2 by the traffic controller. This mechanism is transparent to the programmer.

OCP-T1 and OCP-T2 both implement the arbitration schemes described in Section 2.5.10, *Priority Algorithms*—see Figure 2–19.

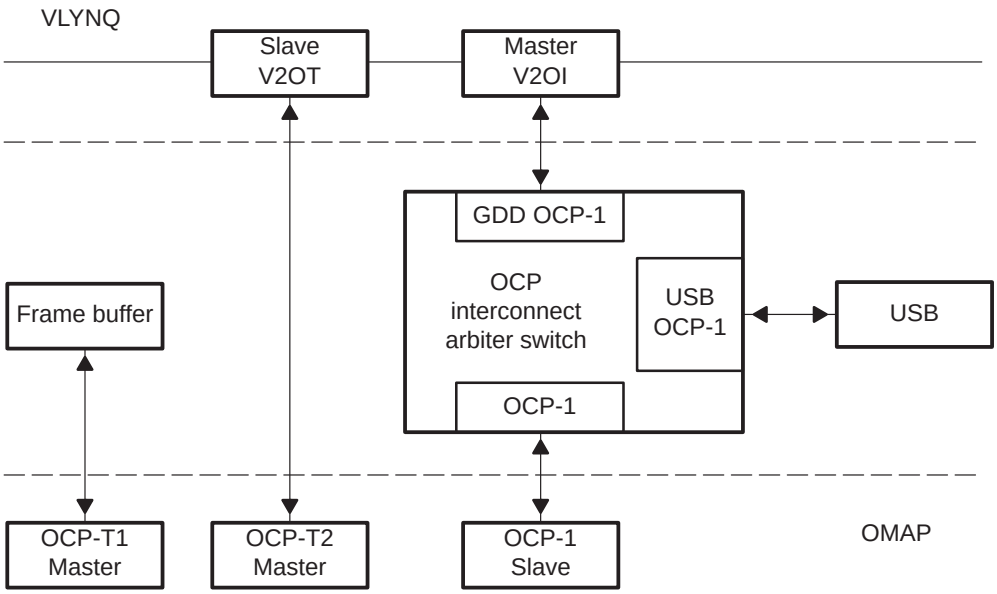
Bus error transactions that generate abort signals on the external OCP interconnect can be tracked within the OCP-T1/T2 controllers.

Two types of errors can be detected:

- ❏ Bus error returned by the OCP interface
- ❏ Time-out error. A request has been sent to the interface but has not been completed in the time-out delay. The timer is programmable from 0 to 255 TC clock cycles.

The abort interrupt handler can read the error reporting registers, which indicate the address that caused the error, the initiator at the origin of the request, and the type of error (see the abort address register and the abort type register, in Section 2.5.9, *OCPI Registers*).

Figure 2–19. OCP-T1 and OCP-T2



2.5.3 EMIFS Programming

Note:

The OMAP730 EMIFS does not support 32-bit wide devices.

The synchronous/asynchronous external memory interface slow (EMIFS) supports most common memory interface protocols through a flexible programming and timing signals control.

- The EMIFS can control up to four devices without any external adding logic through the four independent chip-selects (nCS0-nCS3) and through dedicated memory interface control signals.
- The EMIFS supports common external memory control signals: OE, WE, ADV, BE[0-1], BE[2-3], BAA, READY, Device CLK, RST, and WP.
- Each chip-select controls an address range of 64M bytes with dedicated configuration registers to fulfill the protocol and the timing constraints compliant with the external device associated with it. Each chip-select configuration register supports dynamic configuration.
- At boot time or at run time, CS0 and CS3 address mapping can be swapped.
  - CS0 from 0000:0000 to 03FF:FFFF
  - CS1 from 0400:0000 to 07FF:FFFF
  - CS2 from 0800:0000 to 0BFF:FFFF
  - CS3 from 0C00:0000 to 0FFF:FFFF
- The EMIFS supports 16-bit external device width. Based on the CS configuration, the interface adjusts the access size (splitting word32) according to the external device attached to the CS. 8-bit device width is not supported without external adding logic. See the IC device documentation for the external device width supported by the IC (16-bit width only).
- The EMIFS can control multiplexed address and data memory devices without external adding logic based on the CS configuration. The multiplexing scheme is supported for synchronous and asynchronous access mode. Both multiplexed and nonmultiplexed devices can be supported with the same IC on different chip-selects (embedded IC nonmultiplexed memories and external multiplexed devices).
- The EMIFS behavior conforms to the little endian protocol.
- The EMIFS supports 8- and 16-bit asynchronous and synchronous reads and 8- and 16-bit asynchronous writes.
- The EMIFS boot configuration is controlled by dedicated pins that are sampled at IC reset time. This provides flexible boot CS and configuration selection.
- The EMIFS is a multimaster memory interface. It supports flexible and programmable arbitration protocol (LRU priority ordering or dynamic time-based priority ordering).

- The EMIFS includes a programmable time-out timer to prevent system hanging with nonresponding devices. Automatic access completion with interrupt and status logging are issued on time out events.
- The EMIFS supports dynamic local idle mode control. The EMIFS also supports IC deep power-down mode request synchronization.

See the device IC documentation for timing and maximum frequency operations.

2.5.3.1 General Description

EMIFS Synchronous and Asynchronous Modes

The operation mode of the EMIFS for a given chip-select region is selected by the RDMODE bit field of the CS configuration registers. Operations supported are:

- Mode 0: Asynchronous read. Used for any asynchronous memory, including flash devices.
- Mode 1-2-3: Asynchronous page mode read with control of 4 (mode1), 8 (mode 2), and 16 (mode 3) words (device width) per page. These modes are mainly used for page-mode flash devices.
- Mode 4-5: Synchronous burst read (with burst advance control for mode 4). These modes are mainly used for synchronous-burst flash devices.
- Mode 7: Synchronous pipelined burst read. This mode is mainly used for TI embedded IC ROM and RAM memories.

  For all of these modes, write accesses are performed according to asynchronous write protocol.

Single and burst access address alignment:

- The OMAP master only issues Word16 and Word32 aligned access (word address must be aligned on word size address boundary).

EMIFS Memory Timing Control

- In both asynchronous and synchronous modes, all EMIFS to memory control signals are controlled with an EMIFS internal REF_CLK. Depending on the CS configuration, this internal clock can be available outside through the FCLK EMIFS output pin.
  - The REF_CLK is divided from TC_CK (traffic controller root clock) by a programmable value contained in the FCLKDIV bit field of the CS configuration register. This provides easy accommodation of the timing constraints of slow devices, even with high system clock rate.
- In asynchronous mode 0-1-2-3, the REF_CLK is not activated outside EMIFS. The FLASH_CLK_O is kept low.
- In synchronous mode 4-5-7, REF_CLK is available through FCLK. This is also the case during asynchronous write access. The FCLK clock is

connected to the external synchronous device input clock (synchronous flash or ASIC). The REF_CLK frequency must be set to comply with the attached device timing constraints in combination with the other EMIFS timing controls.

- In synchronous mode 4-5, a retiming mode enables read data to be latched by a delayed REF_CLK.
  - The REF_CLK delay is obtained through the IC I/O feedback of FCLK to offer optimum data and clock alignment. Pipelined access offers relaxed timing constraints.
  - The retiming mode is enabled through the RT bit field in the CS configuration register.
- Memory control signal (CS, OE, WE, ADV, and ADDRESS) setup and hold timing can be controlled by programmable internal delay generation.
  - OE valid/invalid timing from/to CS and address valid/invalid is programmable through the OESETUP and OEHOLD bit fields in the advanced CS configuration register.
  - WE valid timing from the CS, address, and data valid is programmable through the WRWST bit field in the CS configuration register. WE hold time is fixed to one REF_CLK.
  - ADV valid pulse time is programmable through the ADVHOLD bit field in the advance CS configuration register.
    - In synchronous read mode, the ADV setup to REF_CLK rising edge is controlled by the ADVHOLD bit field. The ADV hold time from the REF_CLK rising edge is fixed to one TC_CK.
  - Address setup and hold time with ADV control:
    - In synchronous mode, the address setup time from the REF_CLK rising edge (ADV valid) is controlled by the ADVHOLD bit field.
    - In multiplexed synchronous mode, the address hold time from the REF_CLK rising edge is fixed to one REF_CLK from ADV invalid time.
    - In asynchronous mode, the address setup time to ADV invalid is controlled by the ADVHOLD bit field.
    - In asynchronous mode, the address hold time from ADV invalid is fixed to one REF_CLK.
- Read and write access time is controlled by programmable internal wait state generation (non-full-handshaking mode). An external ready input pin can also be used in combination with internal wait state (full-handshaking mode).
  - In non-full-handshaking mode, the RDWST and PGWST (mode 1-2-3 only) bit fields in the CS configuration register are used to control internal read wait-state generation.
  - In non-full-handshaking, the WELEN bit field in CS configuration register is used to control internal write wait-state generation.

- In full-handshaking mode, the ready input pin is monitored by the EMIFS to control read and write access time. Access is completed when both the internal wait state has expired and the ready pin is asserted by the external device. The ready pin assertion/deassertion timing constraint depends on synchronous or asynchronous access mode.
- Modes 0-4-5 are by default in full-handshaking mode. Full-handshaking support on a particular CS can be disabled for these modes through the dynamic wait-state register.

  Modes 1-2-3-7 always follow the non-full-handshaking protocol and the ready pin is never monitored in these modes, even if the full-handshaking bit field in the dynamic wait-state register is cleared.

❑ To prevent data bus contention when slow devices are attached to the IC, the BTWST bit field in the CS configuration register is used to control the TC_CK cycle idle time between specific access sequences.

❑ The BTMODE field in the advance CS configuration register extends the previous mode. The BTWST bit field controls the CS negation time between successive accesses to the same CS.

❑ To prevent data bus floating when no access is requested at the interface (idle sequence), the EMIFS keep the data bus driven with the previous written data or read data (bus keeping feature). In case of a read to idle sequence, the delay time from read to write-back is at least one TC_CK cycle or is controlled by BTWST (TC_CK) cycles.

See the IC device documentation for timing information.

2.5.3.2 EMIFS CS0 and CS3 Decoding Control

CS0 and CS3 address decoding (address in the TC memory mapping) can be swapped through the BM bit field in the EMIFS global control register. When the BM bit field is set, CS3 is activated in the 0000:0000-03FF:FFFF range and CS0 is activated in the 0C00:0000-0FFF:FFFF range. After reset, the BM bit value is controlled by a dedicated input pin sampled at reset release time. The IC boot is executed from CS0 or from CS3 attached memories. During usual execution, BM can be changed dynamically but obvious software precautions are required to prevent a system crash.

2.5.3.3 EMIFS CS0 Access Control

On certain IC devices, access to CS0-attached devices is restricted to the MPU host only. This access restriction is controlled by internal hardware to enable secure mode execution. In case of an access violation, the EMIFS prevents access from reaching the destination, and EMIFS issues an abort sequence. See the IC functional specification for details regarding secure mode execution.

2.5.3.4 EMIFS Miscellaneous Memory Signal Control

Common flash memory supports the write protection (WP) input pin to prevent an erroneous write-access sequence from corrupting their content. EMIFS interface includes the WP output pin and provides full software control of it.

Common flash memory supports the Reset input pin to allow the device state machine to be properly reset on power up and to minimize power consumption in idle mode. EMIFS interface includes RSTPWR output pin and provides software and hardware control of it.

2.5.3.5 EMIFS Configuration

EMIFS control and configuration can be done dynamically. EMIFS ensures that a new CS configuration takes effect only when no access is requested (CS idle). To prevent inconsistency and critical behavior, the EMIFS configuration must be done by MPU software while other masters are inactive.

2.5.3.6 EMIFS Abort Control

In case of an access restriction violation or an access time-out error, the EMIFS can handle an interrupt line and abort status register to allow abort events system monitoring.

2.5.3.7 Detail Description

EMIFS Data Bus Lane Control

- Read access
  - During read access, the attached device must drive the bus lane according to the BE (byte-enable signal) activation to drive valid data on the relevant bus lanes. Devices that do not support BE during read access must drive the complete bus lane: D0-D15 for 16-bit device width.
  - BEs are activated according to the access size, the address accessed, the device width, and the little endian protocol. See Table 2–33 for the bus lane activation description.
- Write access
  - During write access, the EMIFS drives the complete bus lane: D0-D15 for 16-bit device width.
  - Valid data is posted on the relevant bus lanes according to the byte enable (BE) activation. Activation is done according to the access size, the address accessed, the device width, and the little endian protocol. See Table 2–33 for the byte-enable activation description.

Table 2–33. 16-Bit Interface Width IC

Access Size	D0-D7	D8-D15	BE{0-1]	HOST A[0]
Byte access	Valid	Invalid	01	0
Byte Access	Invalid	Valid	10	1
Word16 access	Valid	Valid	00	0

Note:

An external 8-bit width (byte addressable) device can be connected on a 6-bit width interface if external logic is implemented to drive the complete IC data bus lanes. The proper bus lane to be driven or to be looked at is extracted from the BE activation.

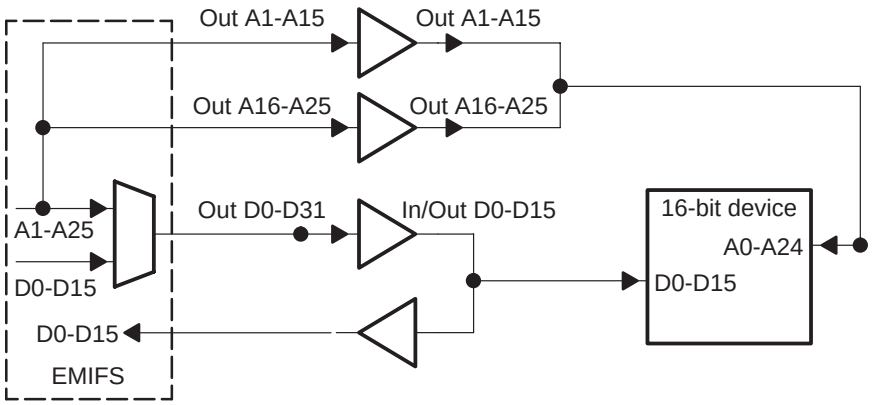
An external 8-bit width device can be attached without external logic on D[0-7] if accesses are limited to bytes with Word16 address aligned.

Common flash devices do not support the byte enable signal. Write access size must fit with the device interface width or invalid data is written. It is the responsibility of software to use proper access width.

2.5.3.8 EMIFS Address Mapping Data Control in Nonmultiplexed Mode

An external 16-bit width is connected to the IC as shown in Figure 2–20. The device size selection is done through the BW bit field of the CS configuration register. See the IC device documentation for the external device width supported by the IC (16-bit width only).

Figure 2–20. EMIFS Address Mapping (Nonmultiplexed)



The external device address bus is shifted relative to the host address bus according to the device width. See Table 2–34 for address mapping.

Table 2–34. Host and External Address Mapping Rule in Nonmultiplexed Mode

Device Address	Host Address in Case of 16-Bit Device	
A0	A1	
A1	A2	
A2	A3	
A3	A4	
….	….	
A23	A24	
A24	A25	64M-byte limit for 16-bit device width

2.5.3.9 EMIFS Address Mapping and Data Control in Multiplexed Mode

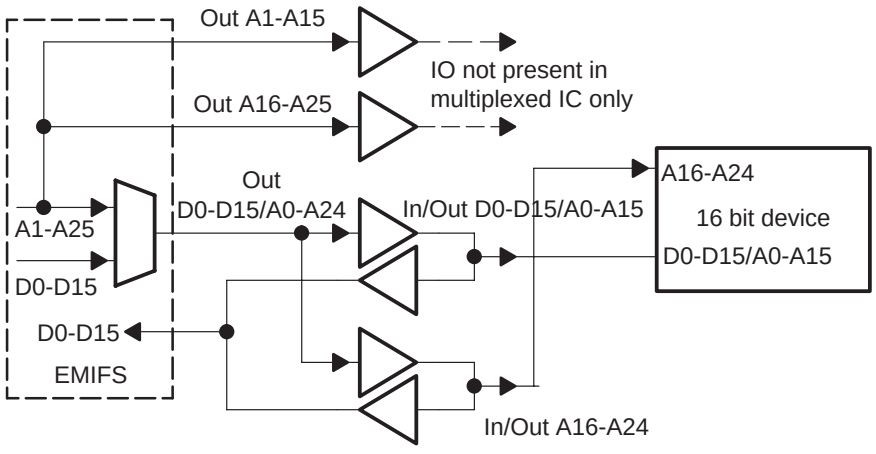
In order to minimize the number of IC pins for the external memory connection, the EMIFS can support multiplexed address and data memory devices without external adding logic.

Selection of the multiplexed mode is done through the MAD bit in the CS configuration register. Multiplexed mode is available only in the following modes:

- ❏ Mode 0 asynchronous read and write
- ❏ Mode 4 and mode 5 synchronous burst read

16-bit multiplexed device widths are connected to the IC, as shown in Figure 2–21. Device size selection is done through the BW fit field of the CS configuration register. See the IC device documentation for the external device width supported by the IC (16-bit width only).

Figure 2–21. EMIFS Address Mapping (Multiplexed)

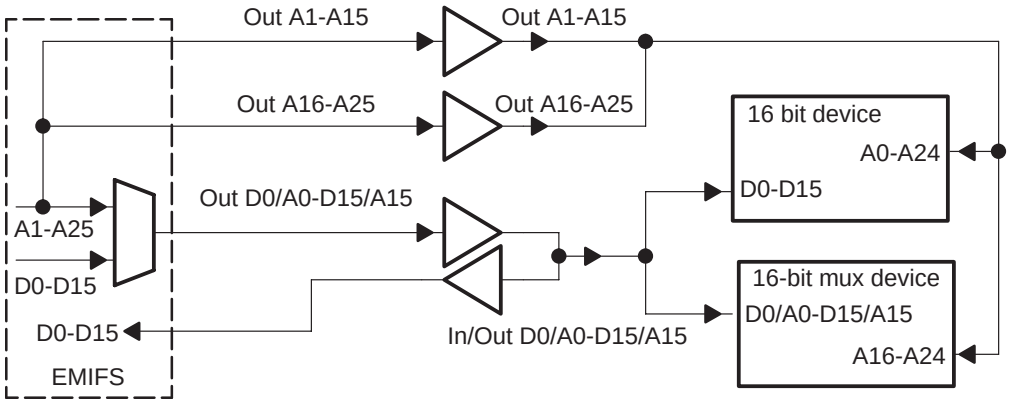


The address and data bus are multiplexed on the same signal bus as shown in Table 2–35.

Table 2–35. Host and External Address Mapping Rule in Multiplexed Mode

Host and External Address Mapping Rule for 16-Bit Device		
Device Add/Data	**Host Address**	**Host Data**
A0/D0	A1	D0
A1/D1	A2	D1
A2/D2	A3	D2
A3/D3	A4	D3
. . .	. . .	. . .
A14/D14	A15	D14
A15/D15	A16	D15
A23	A24	
A24	A25	

Figure 2–22. EMIFS Address Mapping (Both Multiplexed)



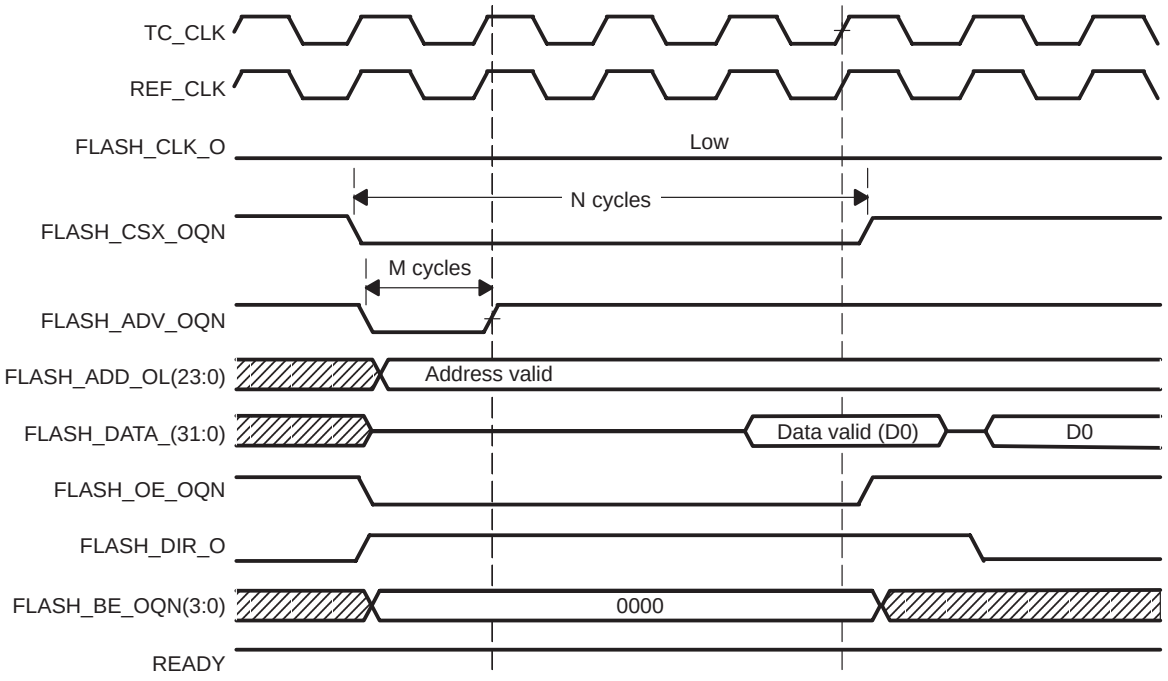
2.5.3.10 Mode 0—Asynchronous Read Operation

Basic Programming Model

The asynchronous read mode 0 is selected by setting RDMODE = 0 in the corresponding EMIFS chip-select configuration register.

Figure 2–23 shows a typical timing diagram.

Figure 2–23. Asynchronous 32-Bit Read Operation on a 32-Bit-Wide Device. RDWST=2 FCLKDIV=0 OESETUP = 0 OEHOLD = 0 ADVHOLD = 0. Data write-back on the bus after read completion.



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

The REF_CLK is divided from TC_CK (traffic controller root clock) by a programmable value contained in FCLKDIV bit field of the CS configuration register:

FCLKDIV	REF_CLK
00	TC_CK:1
01	TC_CK:2
10	TC_CK:4
11	TC_CK:6

The CS pulse width depends on the RDWST bit field of the CS configuration register. The CS pulse width equals:

- (RDWST + 2) REF_CLK (N cycles in Figure 2–24)
- CS minimum pulse width is two REF_CLK.

The ADV pulse width depends on the ADVHOLD bit field of the CS configuration register. The ADV pulse width equals:

- (ADVHOLD + 1) REF_CLK (M cycles in Figure 2–24)
- The ADV minimum pulse width is one REF_CLK.

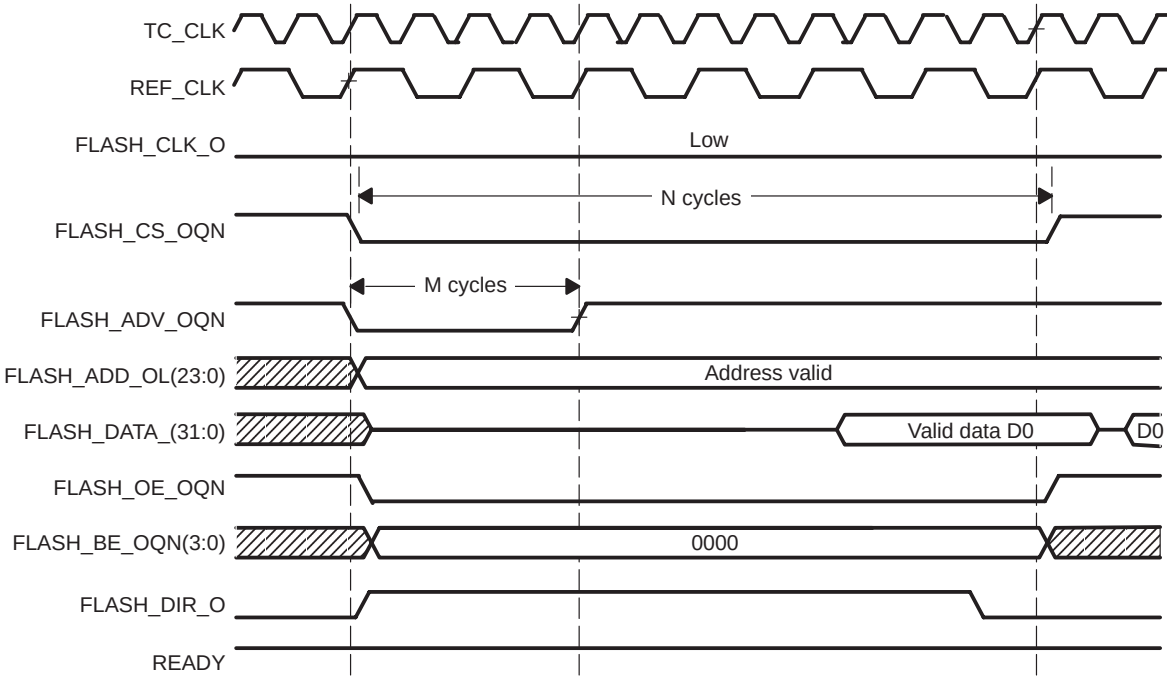
Address drive time follows CS activation (no setup time ensured). Address setup time to the ADV rising edge is controlled by ADVHOLD. Address hold time from ADV rising edge is controlled by CS pulse width.

Read data is latched on the same TC_CK rising edge that deactivates the OE signal.

After a read completion, if no other access (RD, WR) is pending, the data bus is driven with the previous read value. The bus turnaround time (OE going high to direction going out) is a minimum of one TC_CK cycle and can be extended through BTWST (see Section 1.2.9).

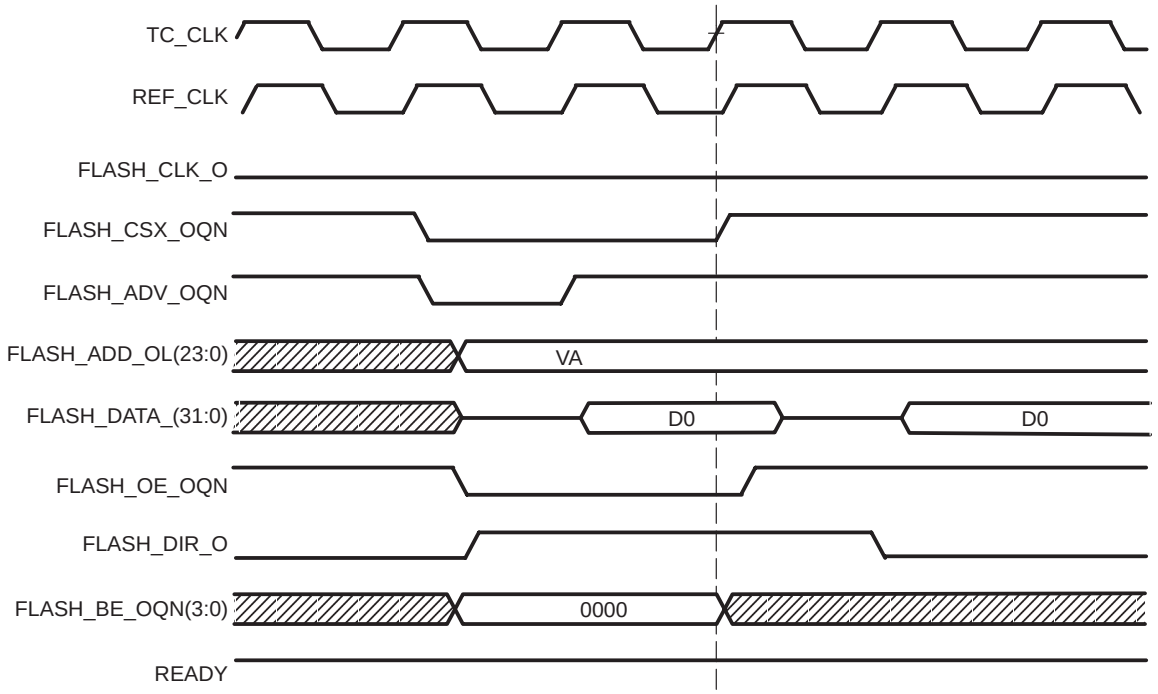
In asynchronous mode, REF_CLK is not provided outside the EMIFS and FLASH_CLK_O is kept low.

Figure 2–24. Asynchronous 32-Bit Read Operation on a 32-Bit-Wide Device. RDWST=4 FCLKDIV=1 OESETUP=0 OEHOLD=0 ADVHOLD=1. Data write-back on the bus after read completion.



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

Figure 2–25. Asynchronous 32-Bit Read Operation on a 32-Bit-Wide Device. RDWST=0 FCLKDIV=0 OESETUP=0 OEHOLD=0 ADVHOLD=0. Data write-back on the bus after read completion.



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

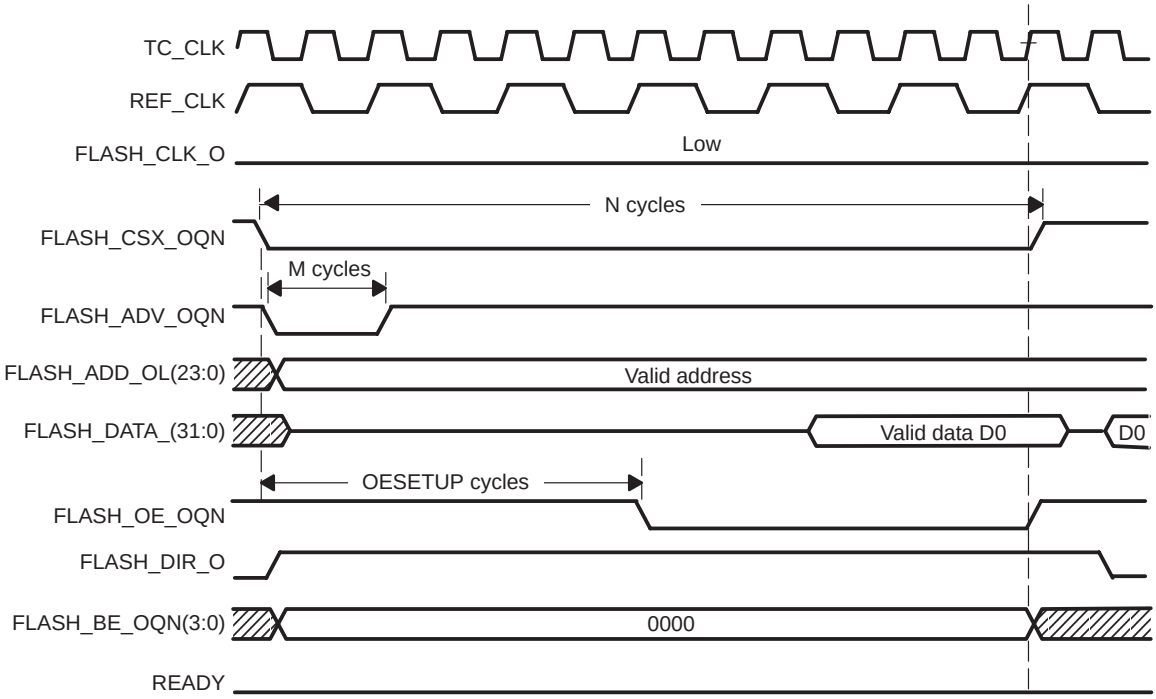
Advanced OE Control

OE activation delay time from CS and address valid is programmable through the OESETUP bit field in the advanced CS configuration register. Activation delay timing is equal to (OESETUP) REF_CLK (see the OESETUP cycles in Figure 2–26).

OE deactivation advance time to CS and address invalid is programmable through the OEHOLD bit field in the advanced CS configuration register. Deactivation advance timing is equal to(OEHOLD) REF_CLK (see the OEHOLD cycles in Figure 2–26).

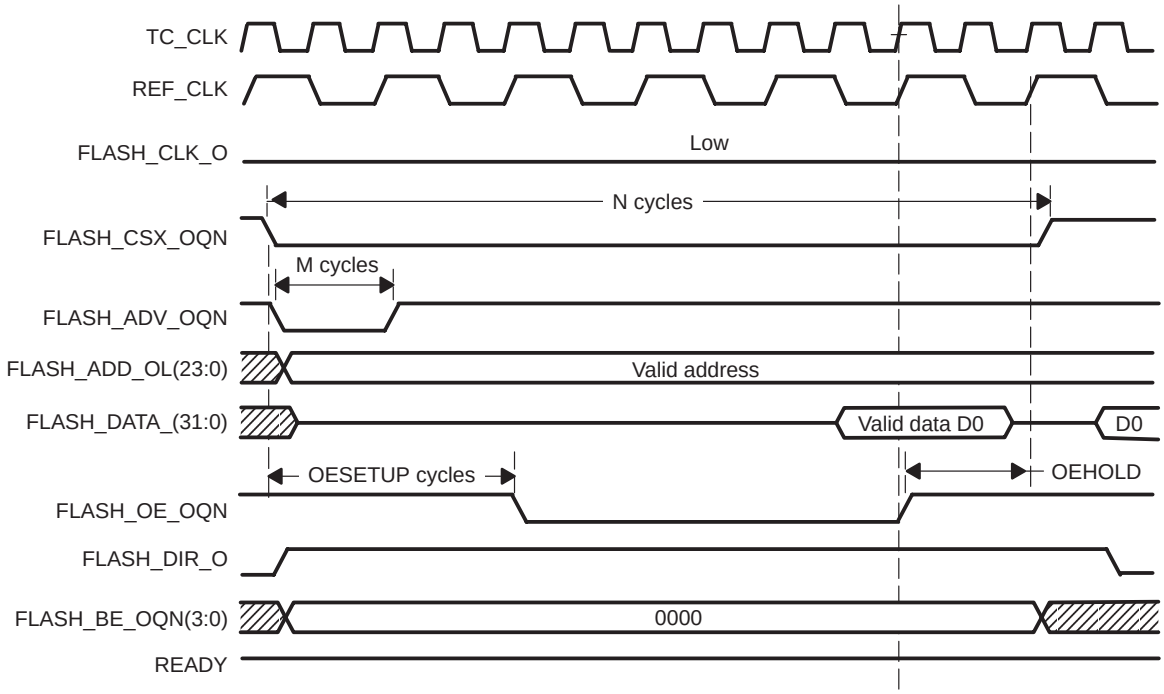
Because CS minimum pulse width is two REF_CLK, the OE delay and advance timing value must be set so that (OESETUP + OEHOLD) is less than or equal to RDWST. Noncompliant programming results in bad access completion.

Figure 2–26. Asynchronous 32-Bit Read Operation on 32-Bit-Wide Device. RDWST=4 FCLKDIV=1 OESETUP=3 OEHOLD=0 ADVHOLD=0. Data write-back on the bus after read completion.



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

Figure 2–27. Asynchronous 32-Bit Read Operation on a 32-Bit-Wide Device. RDWST=4 FCLKDIV=1 OESETUP = 2 OEHOLD = 1 ADVHOLD = 0. Data write-back on the bus after read completion.



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

Read Access Size Adaptation and CS Pulse Width High Control

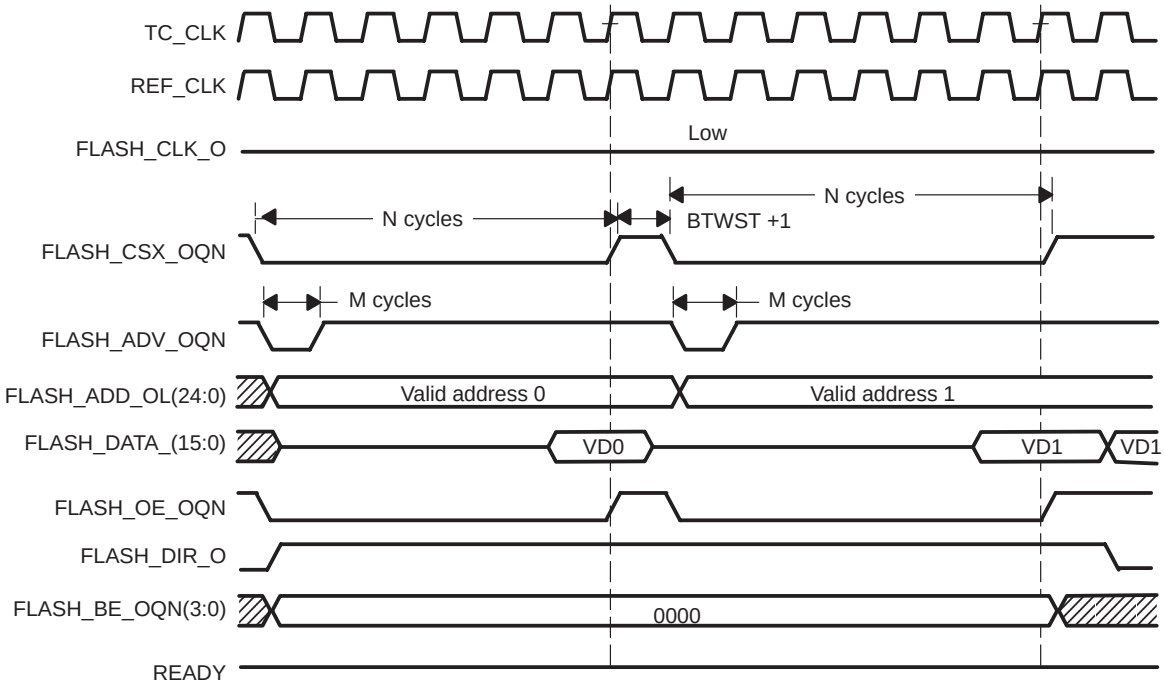
In read mode 0, the EMIFS splits the Word32 access into two Word16 accesses in case of 16-bit device width; 4 x Word32 burst read are split into eight successive Word16 accesses. The split process follows the little endian protocol (Word32 LSB part at lower Word16 address).

In read mode 0, EMIFS divides 4 x Word32 burst read into four successive Word32 accesses in case of 32-bit device width.

During split read accesses and during burst read accesses, the CS signal is deactivated for at least one TC_CK between two successive accesses. CS pulse-width high time can be extended by the BTWST field in the CS configuration register.

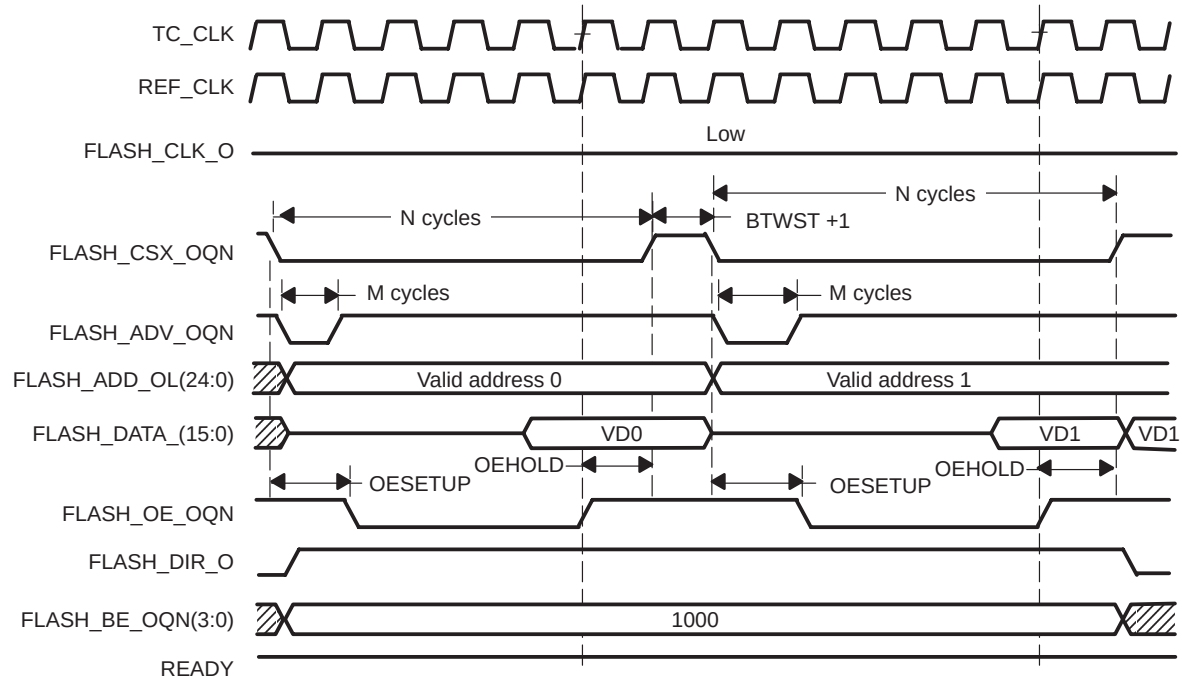
CS pulse width high = (BTWST +1) TC_CK

Figure 2–28. Asynchronous 32-Bit Read Operation on a 16-Bit Width Device. RDWST=4 FCLKDIV=0 OESETUP = 0 OEHOLD = 0 ADVHOLD = 0 BTWST=0 BTMODE=0. Data write-back on the bus after read completion.



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

Figure 2–29. Asynchronous 32-Bit Read Operation on a 16-Bit Width Device. RDWST=4 FCLKDIV=0 OESETUP = 1 OEHOLD = 1 ADVHOLD = 0 BTWST=0 BTMODE=0. Data write-back on the bus after read completion.



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

Full-Handshaking and READY Pin Usage in Asynchronous Read Mode

In full-handshaking mode, the READY input pin is monitored by the EMIFS to control read access time. The access is completed when the internal wait state RDWST expires, and the READY pin is asserted by the external device.

ADV pulse width time and OE assertion time are not dependent on the READY pin state.

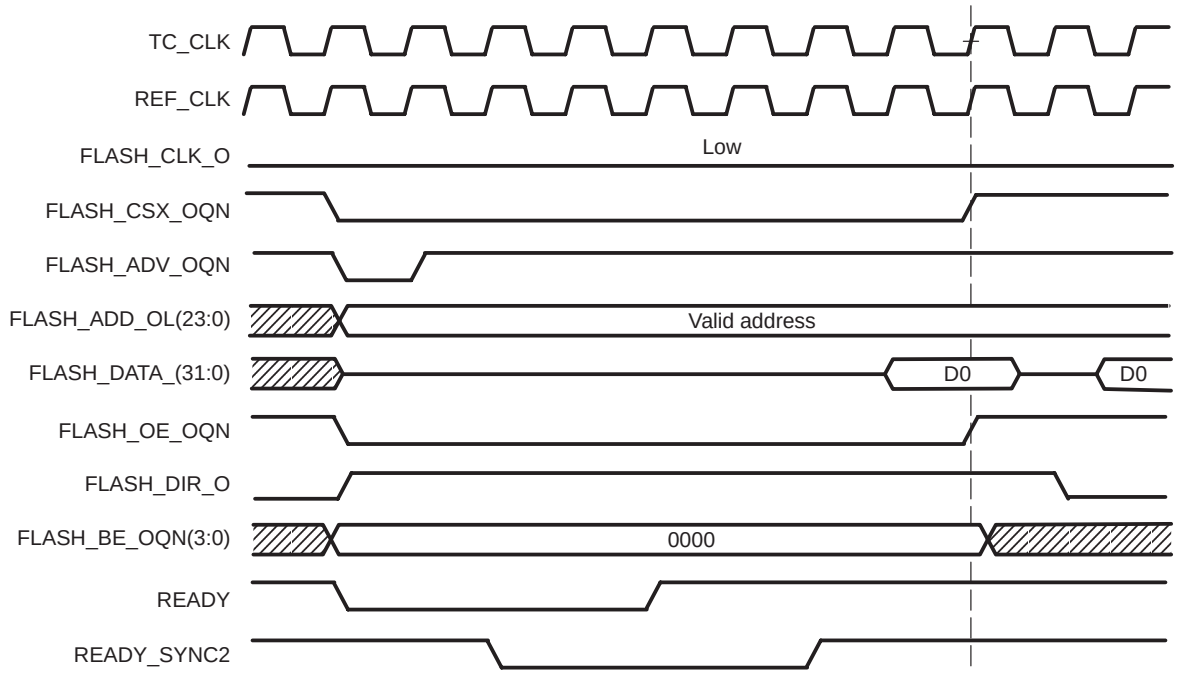
When the READY pin is used to extend the access time, the access completion is not controlled by internal delay generation. CS, OE, and address are deactivated when the READY pin is detected high. No OEHOLD time can be controlled in this case, and the bit field must be equal to zero.

Because READY is an asynchronous signal, a nonready device must drive READY low enough time ahead of the minimum access time completion. Depending on the READY assertion low delay from CS active and depending on the REF_CLK frequency, a minimum RDWST value can be needed for the READY pin state to be monitored correctly by the EMIFS.

As an example, a minimum of RDWST = 2 is needed for a nonready device that drives READY low with 0 time delay from CS low and for a CS configuration FCLKDIV = 0.

See the IC device documentation for timing information regarding the READY assertion timing constraint.

Figure 2–30. Asynchronous 32-Bit Read Operation With Ready. RDWST=2 FCLKDIV=0 OESETUP = 0 OEHOLD = 0 ADVHOLD = 0. Data write-back on the bus after read completion.



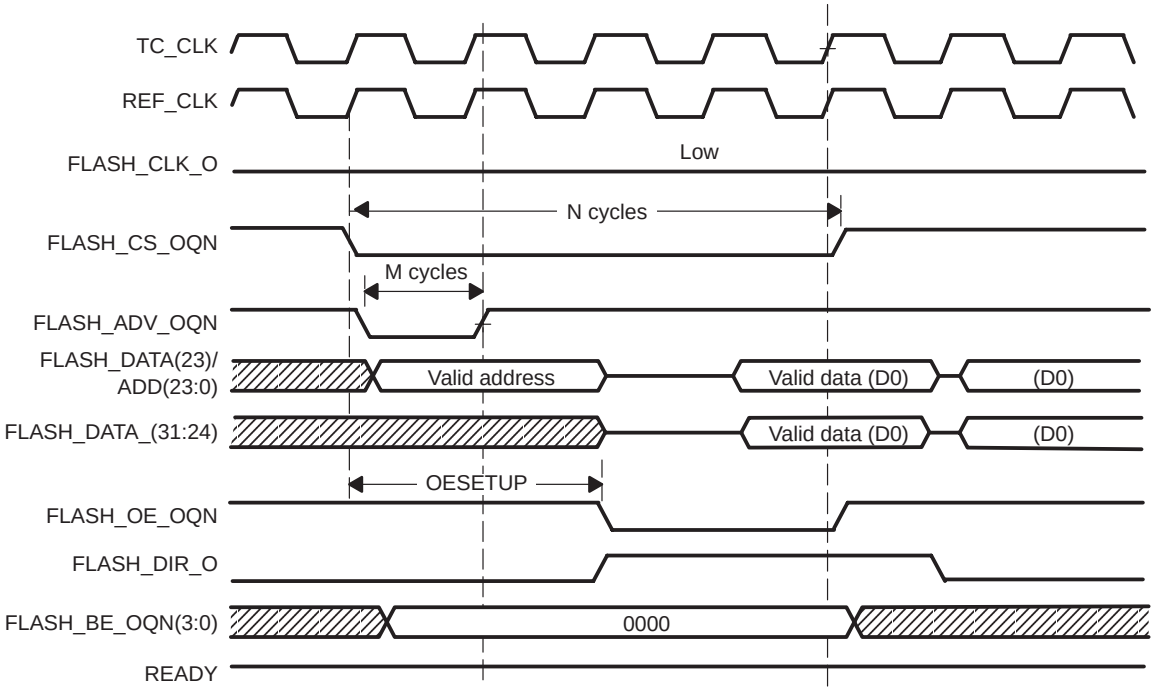
Note: The OMAP730 EMIFS does not support 32-bit wide devices.

Asynchronous Read With Multiplexed Address and Data Memory

The EMIFS can support multiplexed address and data memory devices without external adding logic. Multiplexed mode is enabled when the MAD bit field in the CS configuration register is set to 1.

Figure 2–31 shows an asynchronous read operation with multiplexed address/data bus.

Figure 2–31. Asynchronous 32-Bit Read Operation With Multiplexed Address/Data Bus Memory. RDWST=2 FCLKDIV=0 OESETUP=2 OEHOLD=0 ADVHOLD=0. Data write-back on the bus after read completion.



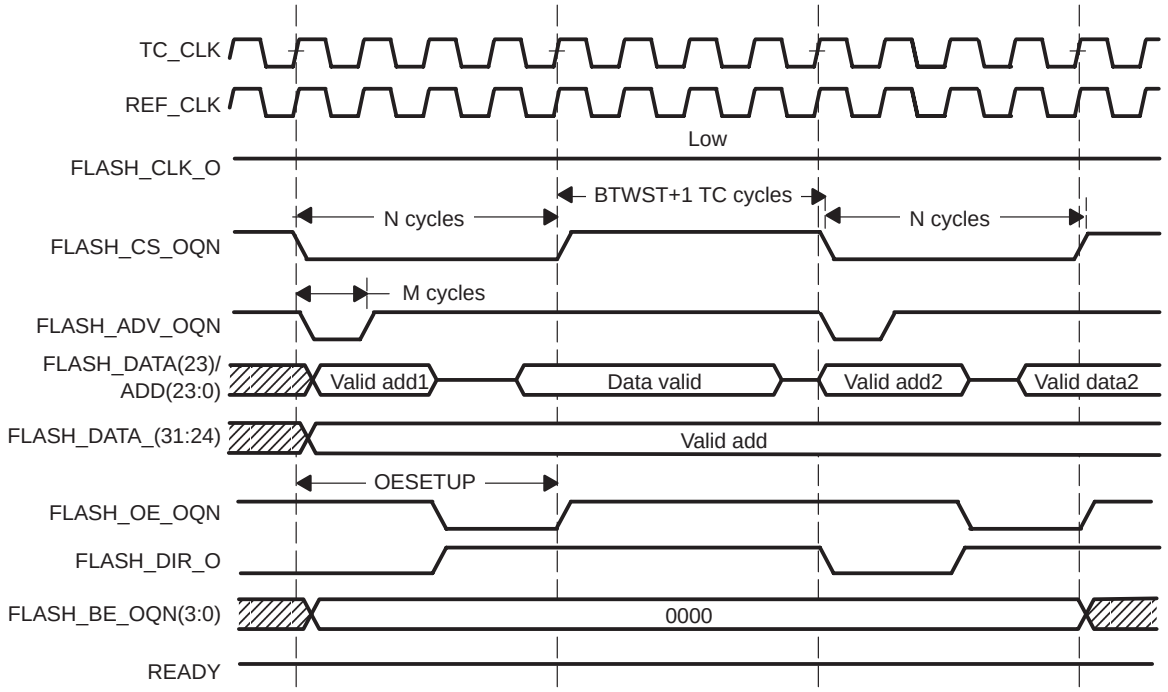
Note: The OMAP730 EMIFS does not support 32-bit wide devices.

Address drive time follows CS activation (no setup time ensured). Address set-up time to the ADV rising edge is controlled by ADVHOLD. Address hold from the ADV rising edge is a minimum of one REF_CLK (delay for direction to change from out to in). FCLKDIV and OESETUP must be properly programmed to prevent bus contention and to ensure that the address hold time device requirement is respected.

During split read accesses and during burst read accesses, the CS signal is deactivated for at least one TC_CK between two successive accesses. CS pulse width high time can be extended by the BTWST field in the CS configuration register.

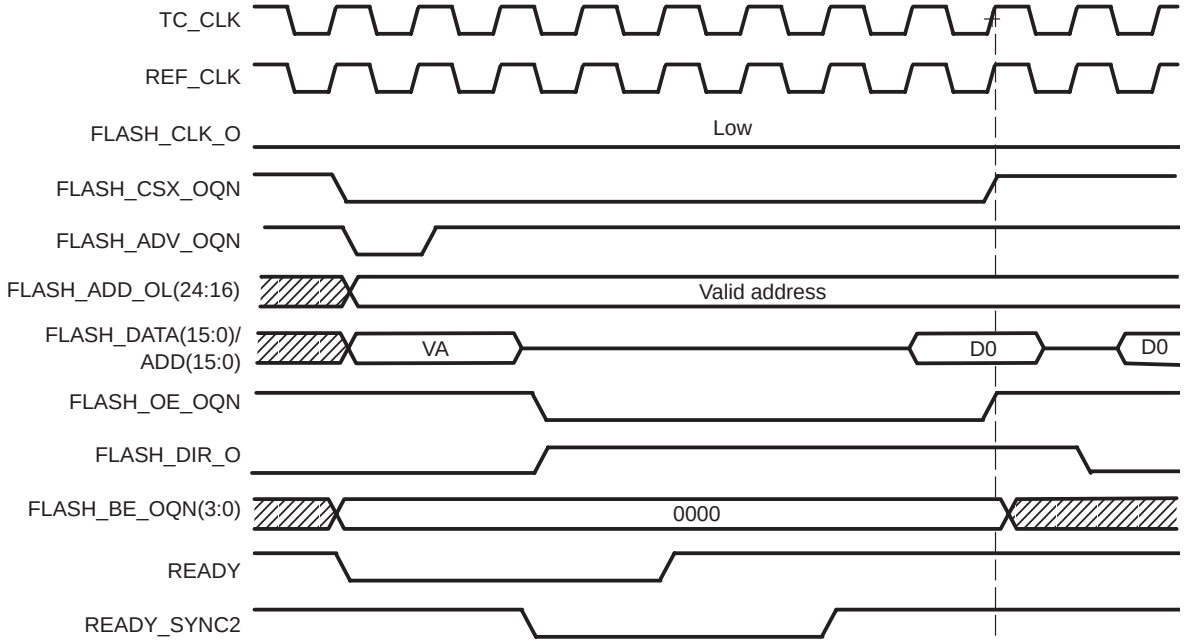
CS pulse width high = (BTWST +1) TC_CK

Figure 2–32. Asynchronous 32-Bit Read Operation on a 16-Bit Multiplexed Address and Data Memory. RDWST=2 FCLKDIV=0 OESETUP=2 OEHOLD = 0 ADVHOLD = 0



The full-handshaking scheme is also valid in multiplexing mode. Figure 2–33 shows an asynchronous Word16 read operation with a 16-bit multiplexed memory control by an external ready pin.

Figure 2–33. Asynchronous 16-Bit Read Operation with Ready on 16-Bit Multiplexed Address and Data Memory. RDWST=2 FCLKDIV=0 OESETUP=2 OEHOLD = 0 ADVHOLD = 0 BTWST = 0, BTMODE = 0

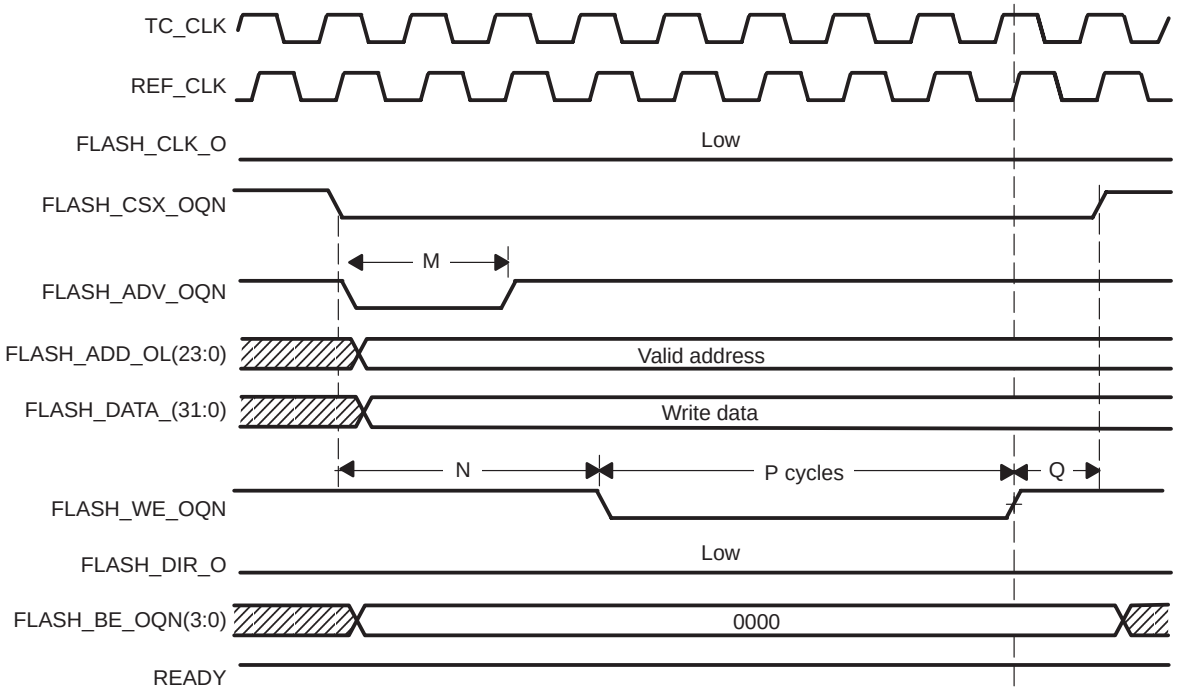


2.5.3.11 Asynchronous Write Operation

Nonmultiplexed Asynchronous Write Operation

The asynchronous write is the only write protocol supported by the EMIFS. The asynchronous write access protocol is used in all EMIFS modes, whatever the CS configuration register RDMODE value is.

Figure 2–34. Asynchronous 32-Bit Write Operation on a 32-Bit-Wide Device (WRWST=2, WELEN=4 FCLKDIV=00 and ADVHOLD=1)



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

The REF_CLK is divided from TC_CK by a programmable value contained in the FCLKDIV bit field of the CS configuration register.

The CS and address setup time from WE low is controlled by the programmable WRWST bit field of the CS configuration register:

(WRWST + 1) REF_CLK (N cycles in Figure 2–34)

WRWST minimum pulse width is 1 REF_CLK.

The ADV pulse width depends on the ADVHOLD bit field of the CS configuration register. ADV pulse width equals:

(ADVHOLD + 1) REF_CLK (M cycles in Figure 2–34)

ADV minimum pulse width is 1 REF_CLK.

The WE pulse width depends on the WELEN bit field of the CS configuration register. WE pulse width equals:

(WELEN + 1) REF_CLK (P cycles in Figure 2–34)

WE minimum pulse width is 1 REF_CLK.

The CS address and data hold time setup from WE high is fixed to one REF_CLK (Q cycle in Figure 2–34).

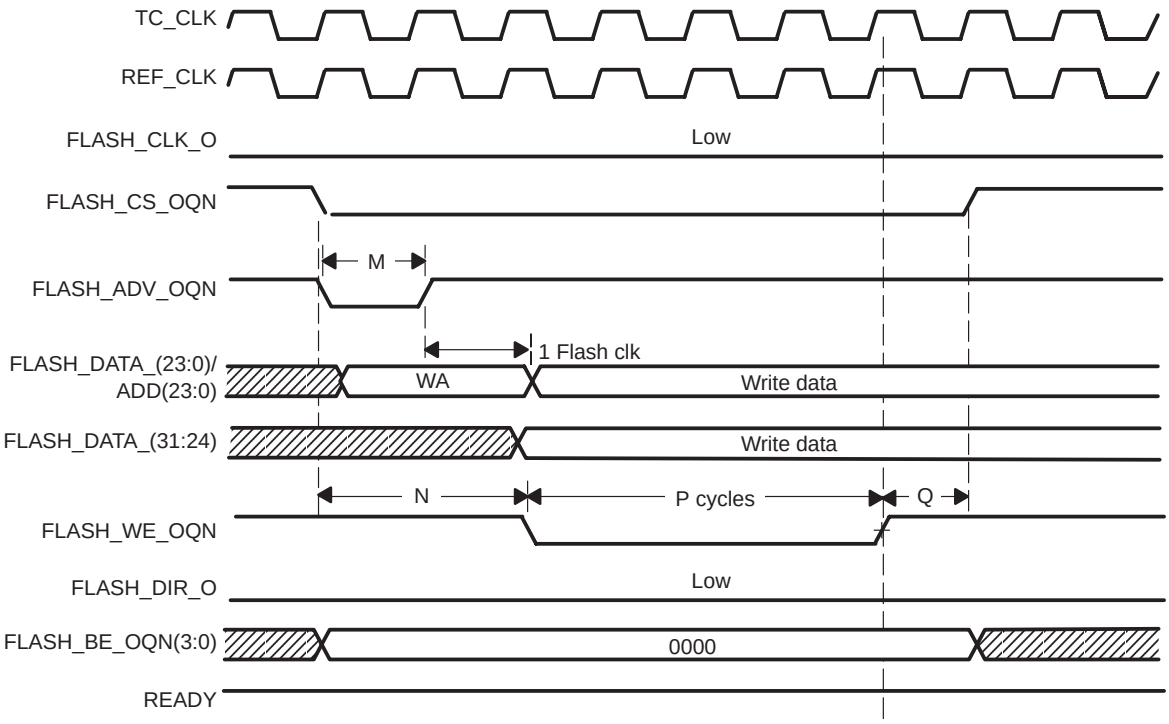
In asynchronous mode 0-1-2-3, REF_CLK is not provided outside the EMIFS and FCLK is kept low. In synchronous mode 4-5, REF_CLK is provided outside

the EMIFS through the FCLK (see modes 4, 5). In synchronous mode 7, REF_CLK is inverted and provided outside the EMIFS through the FCLK (see mode 7).

Multiplexed Asynchronous Write Operation

Figure 2–35 shows a timing diagram with multiplexed address/data bus.

Figure 2–35. Asynchronous 32-Bit Write Operation on a Multiplexed Address/32-Bit Data Bus (WRWST=1, WELEN=3, FCLKDIV=00 and ADVHOLD=0)



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

Multiplexed mode is enabled when the MAD bit field in the CS configuration register is set.

Address drive time follows CS activation (no setup time ensured). Address set-up time to the ADV rising edge is controlled by ADVHOLD. Address invalid from the ADV rising edge is a minimum of one REF_CLK cycle.

The CS and address setup time from WE low is controlled by the programmable WRWST bit field of the CS configuration register:

(WRWST + 1) REF_CLK (N cycles in Figure 2–35)

WRWST minimum pulse width is 1 REF_CLK.

The ADV pulse width depends on the ADVHOLD bit field of the CS configuration register. ADV pulse width equals:

(ADVHOLD + 1) REF_CLK (Mcycles in Figure 3-17)

ADV min pulse width is 1 REF_CLK.

The WE pulse width depends on the WELEN bit field of the CS configuration register. WE pulse width equals:

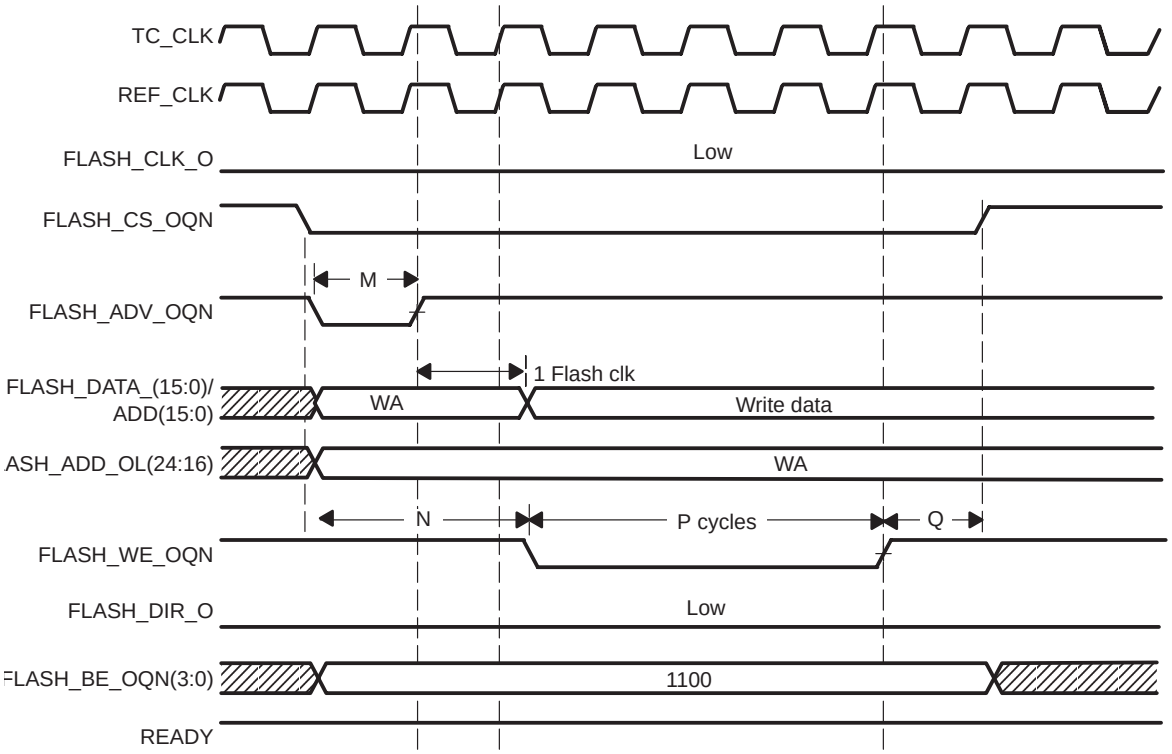
(WELEN + 1) REF_CLK (P cycles in Figure 2–35)

WE minimum pulse width is 1 REF_CLK.

The CS and data hold time setup from WE high is fixed to one REF_CLK (Q cycle in Figure 2–35).

In asynchronous mode 0-1-2-3, REF_CLK is not provided outside the EMIFS and FCLK is kept low. In synchronous mode 4-5, REF_CLK is provided outside the EMIFS through the FCLK. In synchronous mode 7, REF_CLK is inverted and provided outside the EMIFS through the FCLK (see mode 7).

Figure 2–36. Asynchronous 16-Bit Write Operation on a Multiplexed Address/16-Bit Data Bus (WRWST = 1, WELEN = 3, FCLKDIV = 00 and ADVHOLD = 0)



Full-Handshaking and READY Pin Usage in Asynchronous Write Mode

In full-handshaking mode, the READY input pin is monitored by the EMIFS to control read access time. The access is completed when both internal wait state WELEN expires and the READY pin is asserted by the external device. Full-handshaking is the default mode in both multiplexed and nonmultiplexed modes.

ADV pulse width time and WE assertion time are not dependent on the READY pin state.

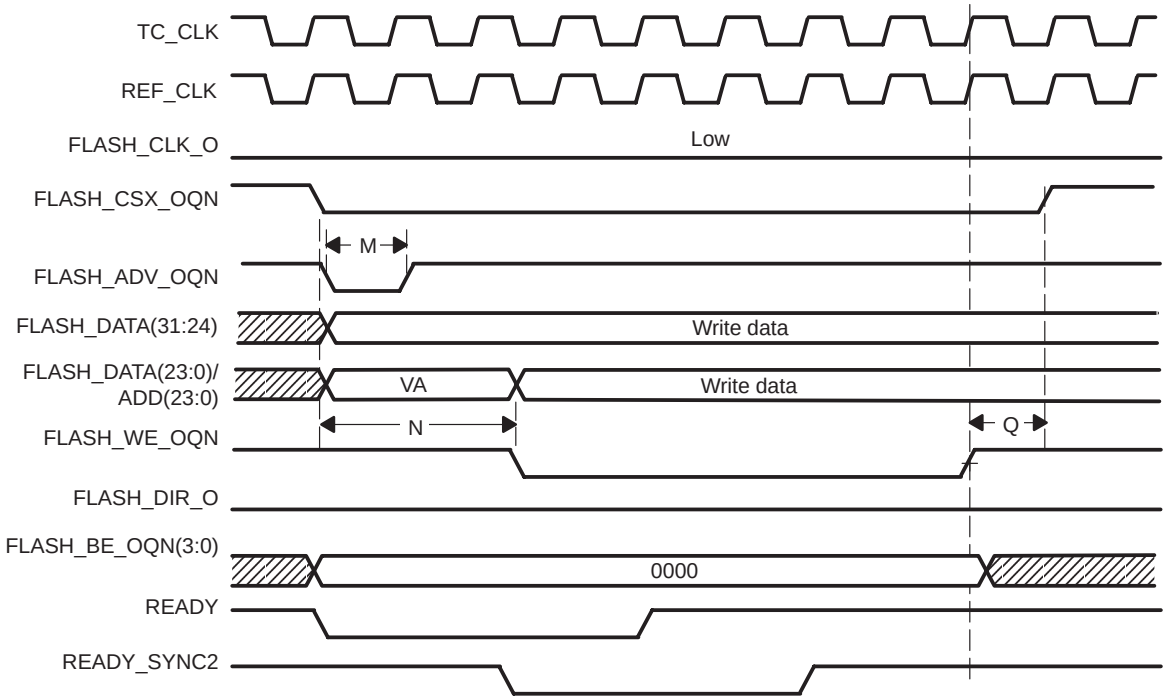
When the READY pin is used to extend the access time, the access completion is not controlled by internal delay generation. The CS and data hold time setup from WE high is still fixed to one REF_CLK (Q cycle in previous figures).

Because READY is an asynchronous signal, a nonready device must drive READY low enough time ahead of the minimum access time completion. Depending on the READY assertion low delay from CS active and depending on the REF_CLK frequency, a minimum WRWST value can be needed for the READY pin state to be correctly monitored by the EMIFS.

As an example, a minimum of WRWST=1 is needed for a nonready device that drives READY low with 0 time delay from CS low, for a CS configuration FCLKDIV=0.

See the IC device documentation for timing information regarding the READY assertion timing constraint.

Figure 2–37. Asynchronous 32-Bit Write Operation on 32-Bit Multiplexed Address and Data Memory With Ready (WELEN = 2, WRWST = 0, FCLKDIV = 0)



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

Write Access Size Adaptation and CS Pulse Width High Control

During write access, the EMIFS splits the Word32 access into two Word16 accesses in case of 16-bit device width. 4 x Word32 burst write are split into 8 successive Word16 accesses. The split process follows the little endian protocol (Word32 LSB part at lower Word16 address).

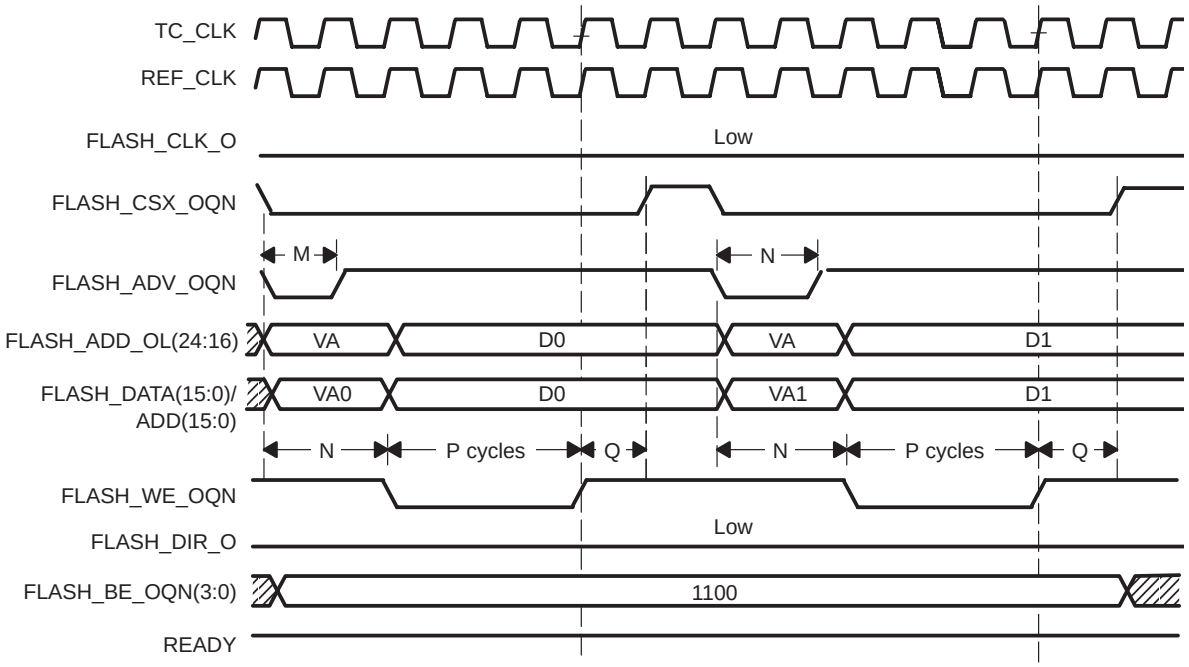
During write access, the EMIFS divides 4 x Word32 burst write into four successive Word32 write accesses in case of 32-bit device width.

During split write accesses and during burst write accesses, the CS signal is not deactivated unless BTMODE in the advance CS configuration register is set. When BTMODE is set the CS pulse width high time can be controlled by the BTWST field in the CS configuration register (see the bus turnaround and CS negation time control).

CS pulse width high= (BTWST +1) TC_CK

This equation is applicable to both multiplexed and nonmultiplexed access modes.

Figure 2–38. Asynchronous 32-Bit Write Operation on 16-Bit Multiplexed Address and Data Memory (WELEN = 2, WRWST =1, FCLKDIV =0, BTWST = 0, and BTMODE = 1)



2.5.3.12 Mode 1-2-3- Asynchronous Page Mode Read Operation

The asynchronous page mode read 1-2-3 is selected by setting the RDMODE bit field in the corresponding EMIFS chip-select configuration register.

- RDMODE = 1 selects the 4-words per page mode.
- RDMODE = 2 selects the 8-words per page mode.
- RDMODE = 3 selects the 16-words per page mode.

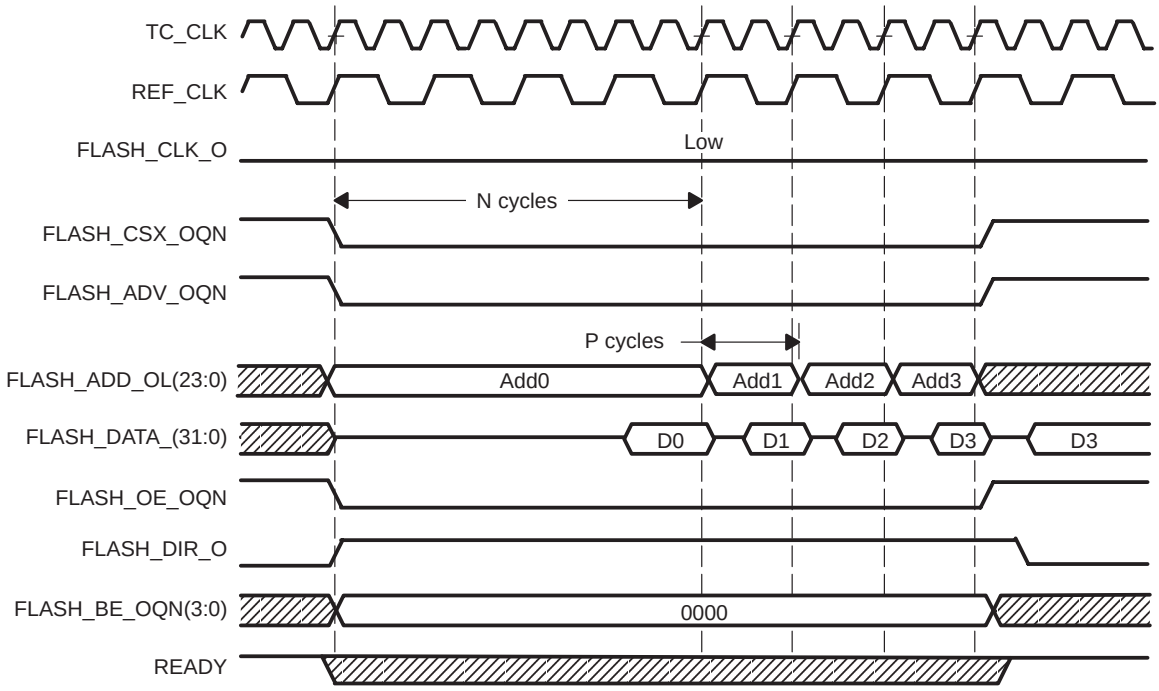
This mode provides single access or fast consecutive accesses in a page. Optimized access time for 2×Word16 read (Word32 read in 16-bit width device) and for burst read (4×Word32 or 8×Word16 in case of 16-bit width device). During consecutive accesses the EMIFS increments the address after each word read completion.

The word length of the access is equal to the memory data bus width and is defined by the BW field of the CS configuration register.

The delay for the first word in the page is controlled by RDWST bit field in the CS configuration register (initial wait state). Depending on the device page length and word size (device width), the EMIFS can control device page crossing during a burst request (4×word32) and inserts an initial wait state delay on purpose.

The delay between successive words in the page is controlled by the PGWST bit field in the CS configuration register (in page wait state).

Figure 2–39. Asynchronous Page Mode 4x32-Bit Read Operation on 32-Bit-Wide Device (RDWST=2, PGWST=0 and FCLKDIV =1, RDMODE=2). Data write-back on the bus after read completion.



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

The REF_CLK is divided from TC_CK by a programmable value contained in the FCLKDIV bit field of the CS configuration register.

The initial wait state depends on the RDWST bit field of the CS configuration register. Delay equals:

❑ (RDWST + 2) REF_CLK (N cycles in Figure 2–35)

The in page wait state depends on:

- The PGWST/WELEN[15:12] bit field if PGWSTEN = 0 in the CS configuration register
- The PGWST[30:27] bit field if PGWSTEN =1 in the CS configuration register

Delay equals (PGWST + 1) REF_CLK (P cycles in Figure 2–33).

The ADV is kept low for the entire access.

Address drive time follows CS activation (no setup time ensured).

Delay time (OESETUP) and advanced time (OEHOLD) are disabled (OESETUP and OEHOLD bit fields are don't care).

Address and data multiplexed scheme is not supported in modes 1-2-3.

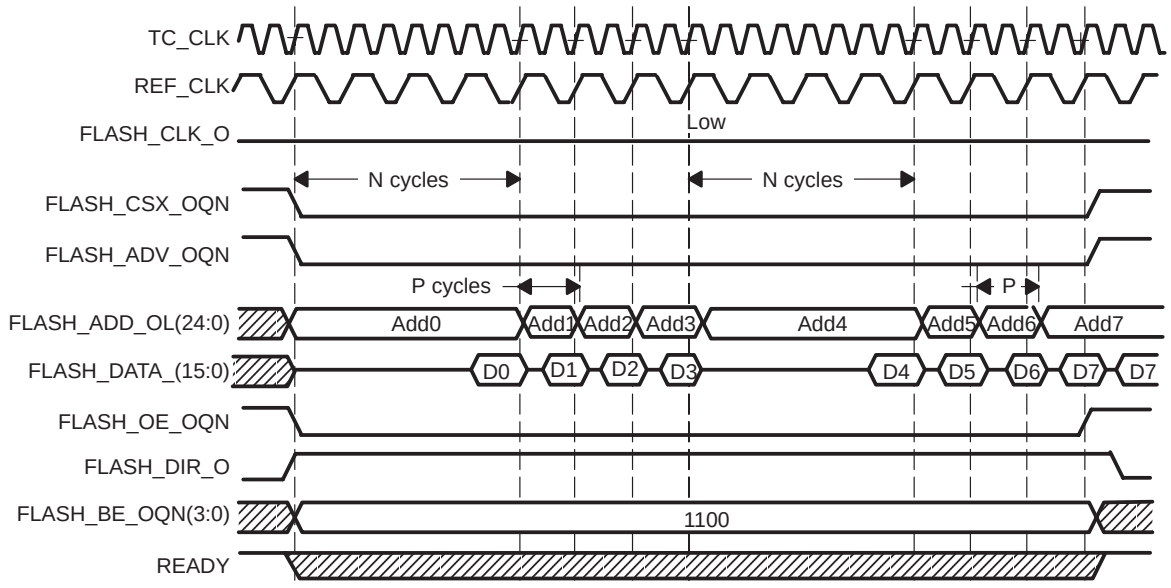
Read data are latched on the TC_CK rising edge corresponding to the wait state delay completion (initial and in page wait state).

One TC_CK cycle after access completion (CS high), the data bus is driven with the previous read value (see Figure 2–35 for direction activation and data copy timing).

In asynchronous mode, REF_CLK is not provided outside the EMIFS and FLASH_CLK_O is kept low.

Page mode always follows the non-full-handshaking protocol and the READY pin is never monitored, no matter what the value of the full-handshaking bit field in the dynamic wait state register.

Figure 2–40. Asynchronous Page Mode 8x16-Bit Read With Page Crossing Operation on 16-Bit Width Device (RDWST=2, PGWST=0 FCLKDIV=1, RDMODE=1). Data write-back on the bus after read completion.



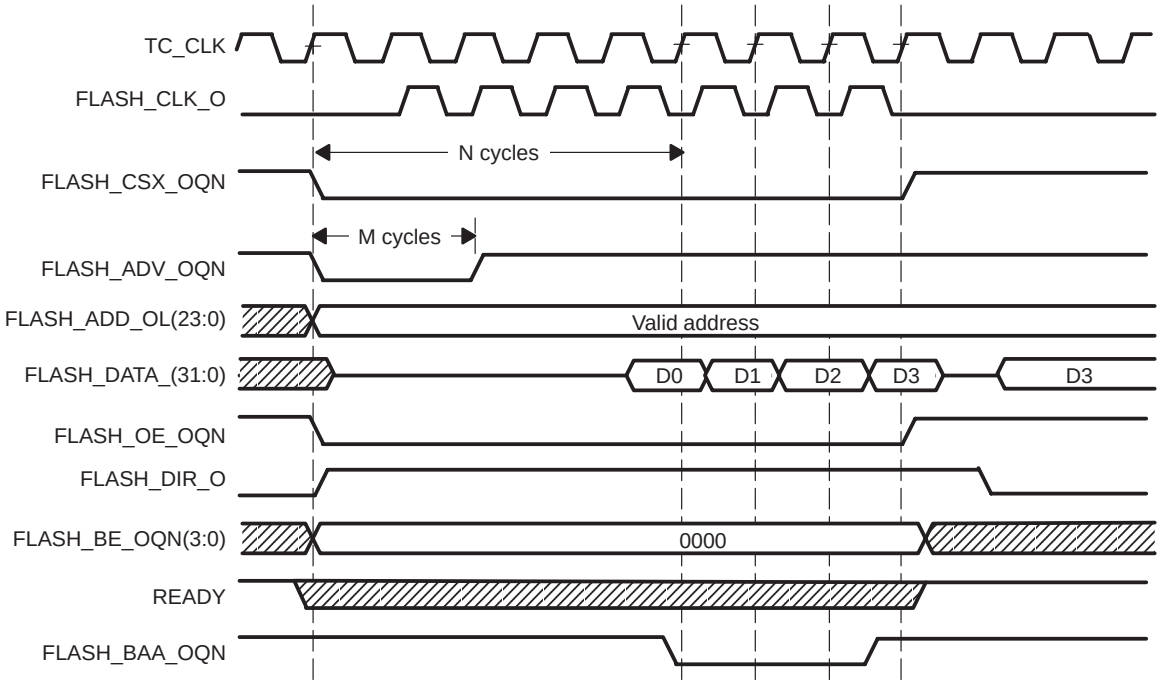
2.5.3.13 Mode 4 and Mode 5 Synchronous Burst Read Operation Mode

Synchronous Read in Nonmultiplexed Address and Data Memory

The synchronous burst read modes 4 and 5 are selected by setting the RDMODE bit field in the corresponding EMIFS chip-select configuration register (RDMODE = 4 or 5).

This mode only supports synchronous read accesses (single or consecutive). Flash devices usually require synchronous setup and enable mode to be done after power up. RDMODE must be changed to mode 4 or mode 5 only after flash device setup.

Figure 2–41. Mode 4 Synchronous Burst 4x32-Bit Read Operation on 32-Bit-Wide Device (RDWST=3, FCLKDIV =0, ADVHOLD=0, RDMODE=4). Data write-back on the bus after read completion.



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

The REF_CLK is divided from TC_CK by a programmable value contained in FCLKDIV bit field of the CS configuration register.

CS, ADV and address are driven one REF_CLK cycle before the first FLASH_CLK_O rising edge is provided externally. This ensures CS, ADV, and address valid setup time to device clock rising edge to be met.

In case this one REF_CLK cycle advance is not enough to meet the setup time requirement, the ADV pulse width can be extended by ADVHOLD. The real access time start from CS & ADV & address setup time to device clock rising edge valid.

The ADV pulse width depends on ADVHOLD bit field of the CS configuration register. ADV pulse width equals (ADVHOLD + 1) REF_CLK + 1 TC_CK (M cycles in Figure 2–37).

Modes 4-5 are by default in full-handshaking mode. The READY input pin is monitored by the EMIFS to control read access time. The READY pin must be asserted synchronously to FLASH_CLK_O.

The first access is completed when both internal RDWST wait state expired and when READY pin is asserted by the external device.

The internal initial wait state depends on the RDWST bit field of the CS configuration register. The RWDST value must include the extra nonactive output REF_CLK cycle used for CS, ADV, address setup time. Delay equals (RDWST + 2) REF_CLK (N cycles in Figure 2–37).

Read data are latched on each TC_CK rising edge corresponding to a REF_CLK rising edge when the ready pin has been sampled high on the previous REF_CLK rising edge.

The following in-burst access wait state depends only on the READY pin state (RDWST expired).

In mode 4, the BAA control signal is asserted low on the first date sampling REF_CLK rising edge and is maintained low during the full-burst access. BAA is kept high in mode 5 (no burst advance control is this mode).

OE activation delay time from the CS and address valid is programmable through the OESETUP bit field in the advanced CS configuration register. Activation delay timing equals (OESETUP) REF_CLK.

Advanced time (OEHOLD) control is disabled (OEHOLD bit field is don’t care).

One TC_CK cycle after access completion (CS high) the data bus is driven with the previous read value (see Figure 2–37 for direction activation and data copy timing).

Figure 2–42. Mode 5 Synchronous Burst 8x16-Bit Read Operation on 16-Bit Width Device (RDWST=3, FCLKDIV =0, ADVHOLD=0, RDMODE=5). Data write-back on the bus after read completion.

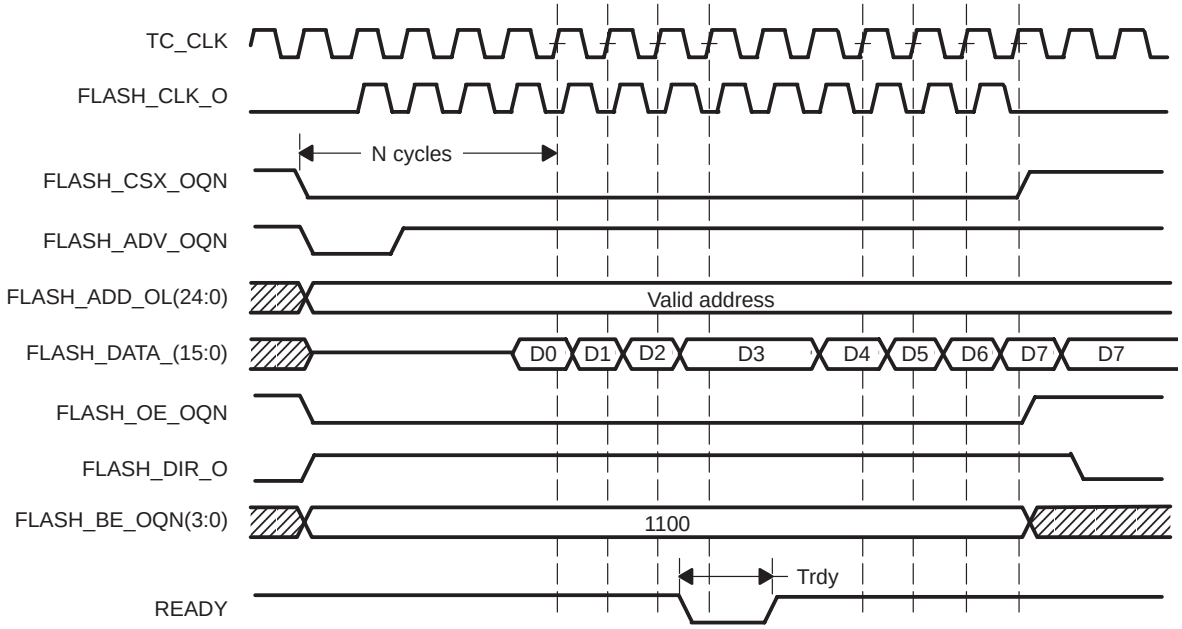
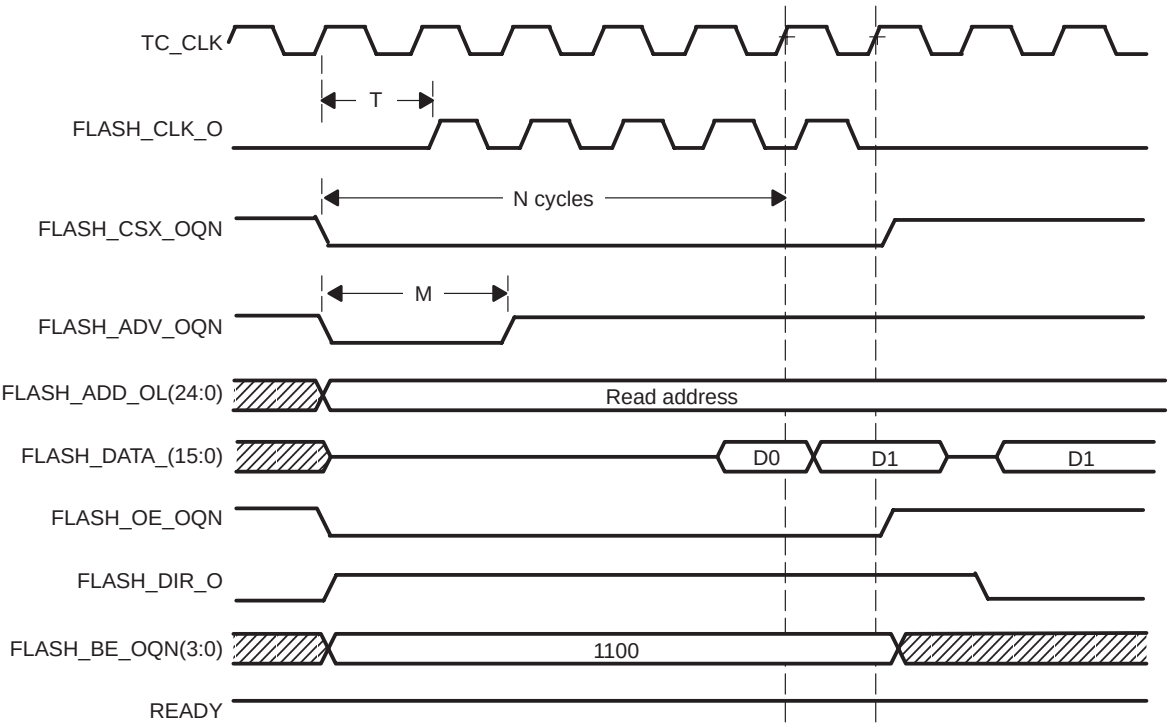


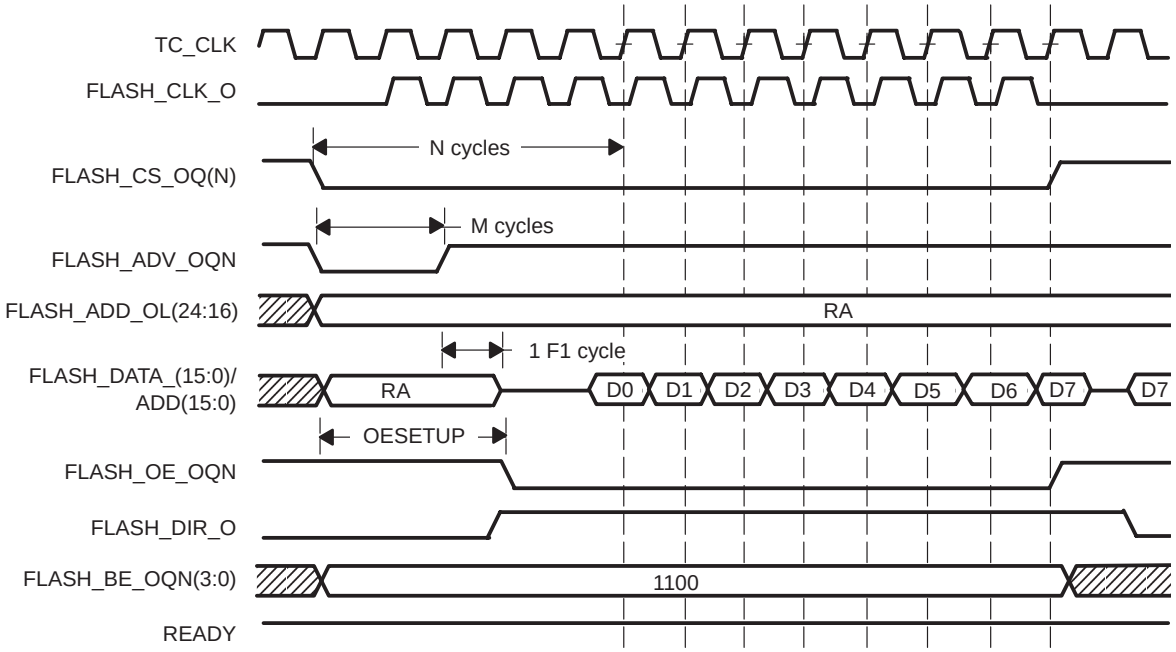
Figure 2–43. Mode 5 Synchronous Burst 2x16-Bit Read Operation on 16-Bit Width Device (RDWST=3, FCLKDIV =0, ADVHOLD=0, RDMODE=5). Data write-back on the bus after read completion



Synchronous Read in Multiplexed Address and Data Memory

Multiplexed mode is enabled when the MAD bit field in the CS configuration register is set to 1.

Figure 2–44. Mode 5 Synchronous Burst 8x16-Bit Read Operation on Multiplexed Address/Data 16-Bit Width Device (RDWST=2, FCLKDIV =0, ADVHOLD=0, OESETUP = 3, RDMODE=5). Data write-back on the bus after read completion.



CS, ADV, and address are driven one REF_CLK cycle before the first FLASH_CLK_O rising edge is provided externally. This ensures CS, ADV, and address valid setup time to the device clock rising edge to be met.

In case this one REF_CLK cycle advance is not enough to meet the setup time requirement, the ADV pulse width can be extended by ADVHOLD. The real access time starts from CS, ADV, and address setup time to the device clock rising edge valid.

Address hold time from ADV rising edge is ensured to be a minimum of one REF_CLK (delay for direction to change from out to in).

FCLKDIV and OESETUP (REF_CLK) must be properly programmed to prevent bus contention and to ensure that the address hold-time device requirement is respected.

Delay time OEHOLD is disabled.

The ADV pulse width depends on the ADVHOLD bit field of the CS configuration register. ADV pulse width equals (ADVHOLD + 1) REF_CLK + 1 TC_CK (M cycles in Figure 2–44).

Modes 4-5 are by default in full-handshaking mode. The READY input pin is monitored by the EMIFS to control read access time. The READY pin must be asserted synchronously to REF_CLK.

The first access is completed when both internal RDWST wait states expire and the READY pin is asserted by the external device.

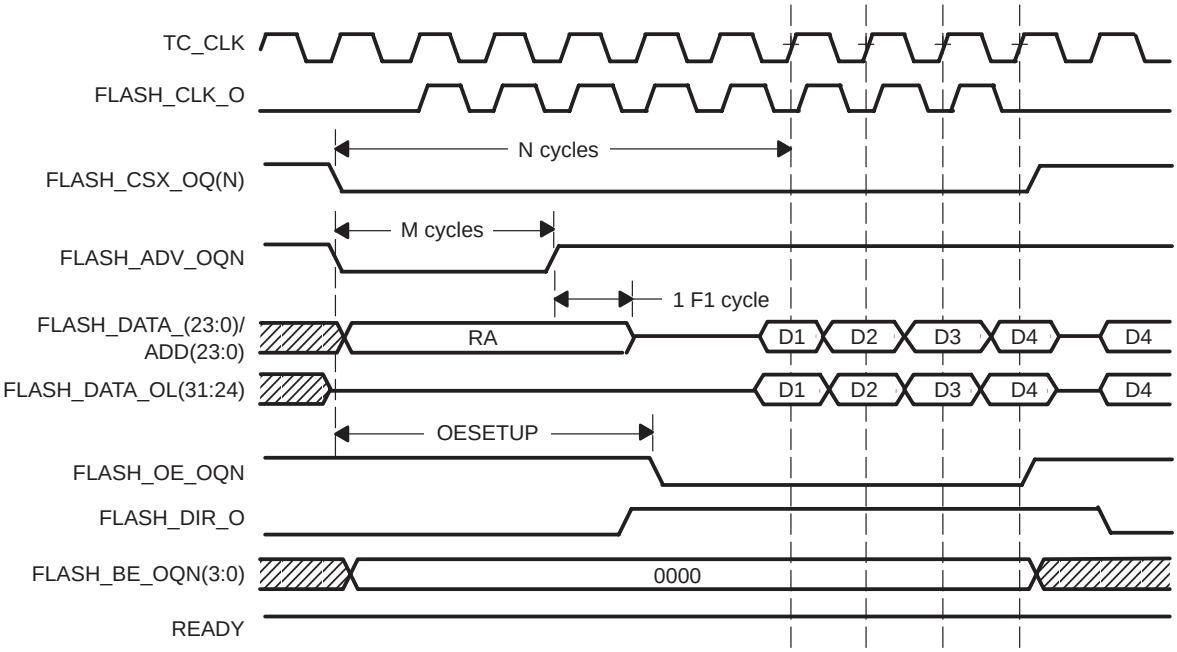
The internal initial wait state depends on the RDWST bit field of the CS configuration register. The RWDST value must include the extra nonactive output REF_CLK cycle used for CS, ADV, and address setup time. The delay equals (RDWST + 2) REF_CLK (N cycles in Figure 2–44).

Read data is latched on each TC_CK rising edge corresponding to a REF_CLK rising edge when the READY pin has been sampled high on the previous REF_CLK rising edge.

The following in-burst access wait state depends only on the READY pin state (RDWST expired).

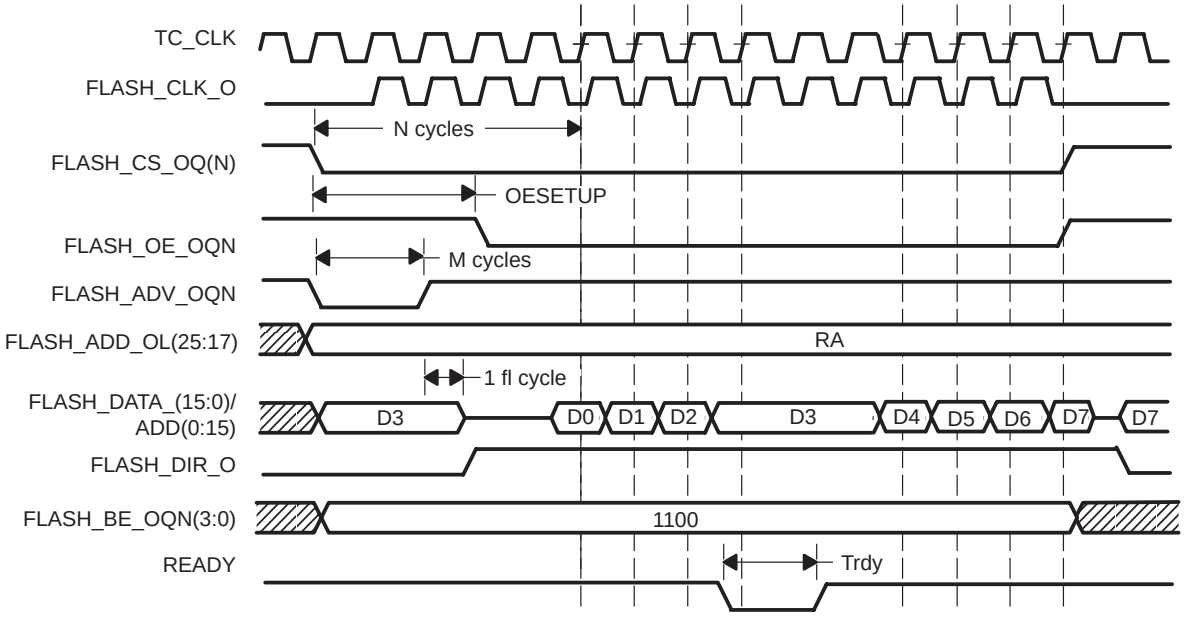
One TC_CK cycle after access completion (CS high), the data bus is driven with the previous read value (see Figure 2–44 for direction activation and data copy timing).

Figure 2–45. Mode 5 Synchronous Burst 4x32-Bit Read Operation on Multiplexed Address/Data 32-Bit-Wide Device (RDWST=4, FCLKDIV =0, ADVHOLD=1, OESETUP = 4, RDMODE=5). Data write-back on the bus after read completion.



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

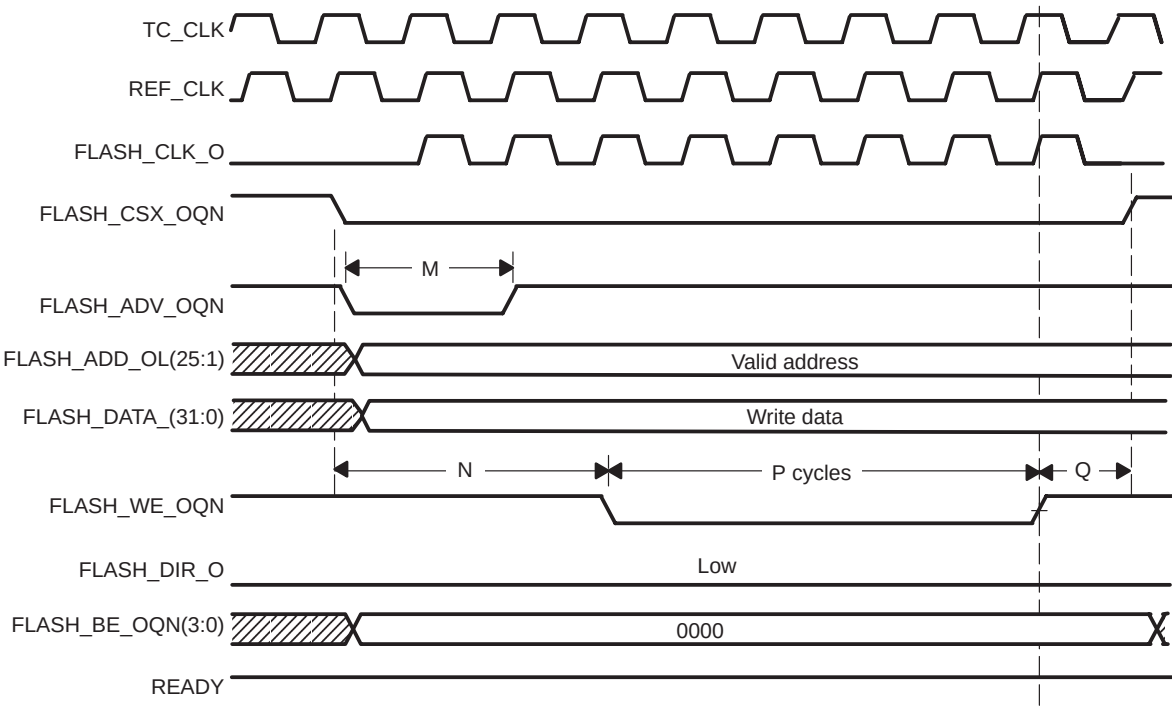
Figure 2–46. Mode 5 Synchronous Burst 8x16-Bit Read Operation on Multiplexed Address/Data 16-Bit Width Device (RDWST=3, FCLKDIV =0, ADVHOLD=0, OESETUP = 3, RDMODE=5). Data write-back on the bus after read completion.



Write Access in Modes 4 and 5

Figure 2–47 shows FLASH_CLK_O activation details during a write access in mode 5 (nonmultiplexed). The behavior is the same as for the multiplexed address and data protocol.

Figure 2–47. Asynchronous 32-Bit Write Operation on a 32-Bit-Wide Device (RDMODE = 5, WRWST=2, WELEN=4 FCLKDIV=00 and ADVHOLD=1)



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

2.5.3.14 Read Retimed Protocol

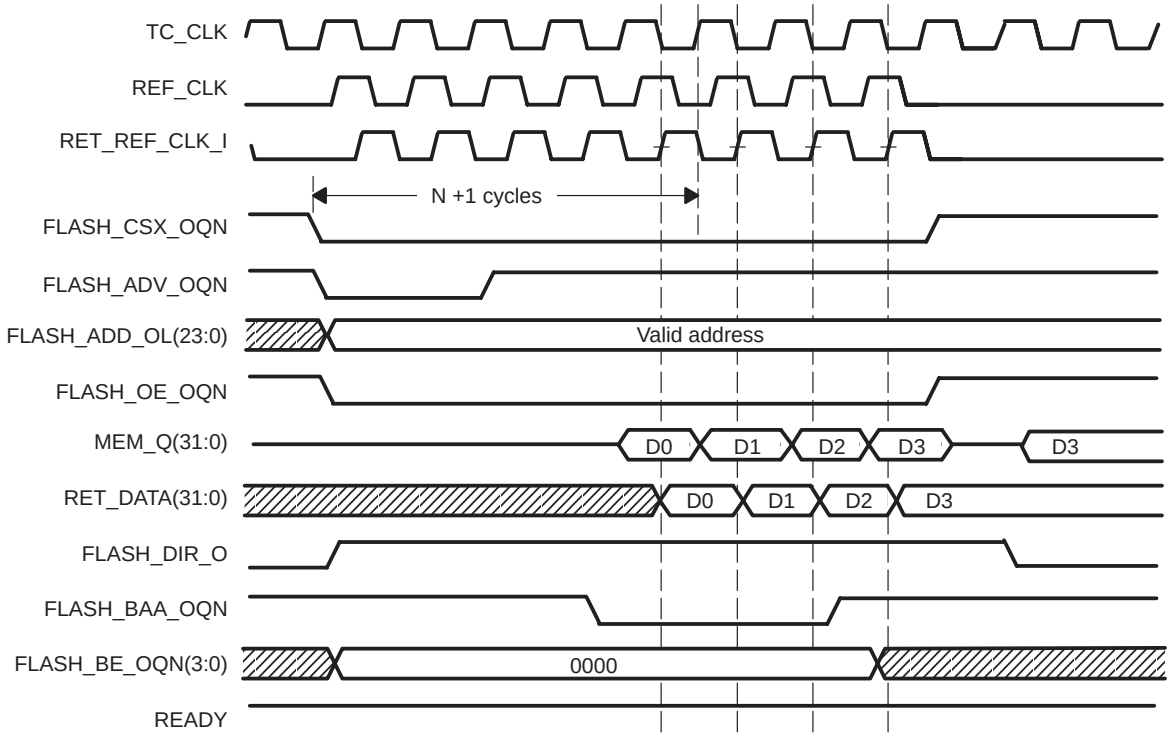
Because of IC input/output and board delays, the theoretical external memory maximum frequency may not be usable for the REF_CLK value without the re-timing function. In synchronous mode 4-5, the retiming mode allows read data to be latched by a delayed RET_REF_CLK obtained through the IC input/output feedback of FLASH_CLK_O. This offers optimum data and sampling clock alignment.

Retiming mode enables a pipelined read access. Compared to non-retimed access, the first access takes one extra REF_CLK cycle and the following accesses in a burst take one REF_CLK cycle each.

The retiming mode is enabled through the RT bit field in the CS configuration register. Retiming mode is effective only in synchronous modes 4-5-7 and has no effect on write accesses.

See the IC device documentation for information about the retiming mode.

Figure 2–48. Mode 4 Synchronous Burst 4x32-Bit Read Operation on 32-Bit-Wide Device With Retiming on (RDWST=2, FCLKDIV =0, ADVHOLD=0, RDMODE=4). Data write-back on the bus after read completion.



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

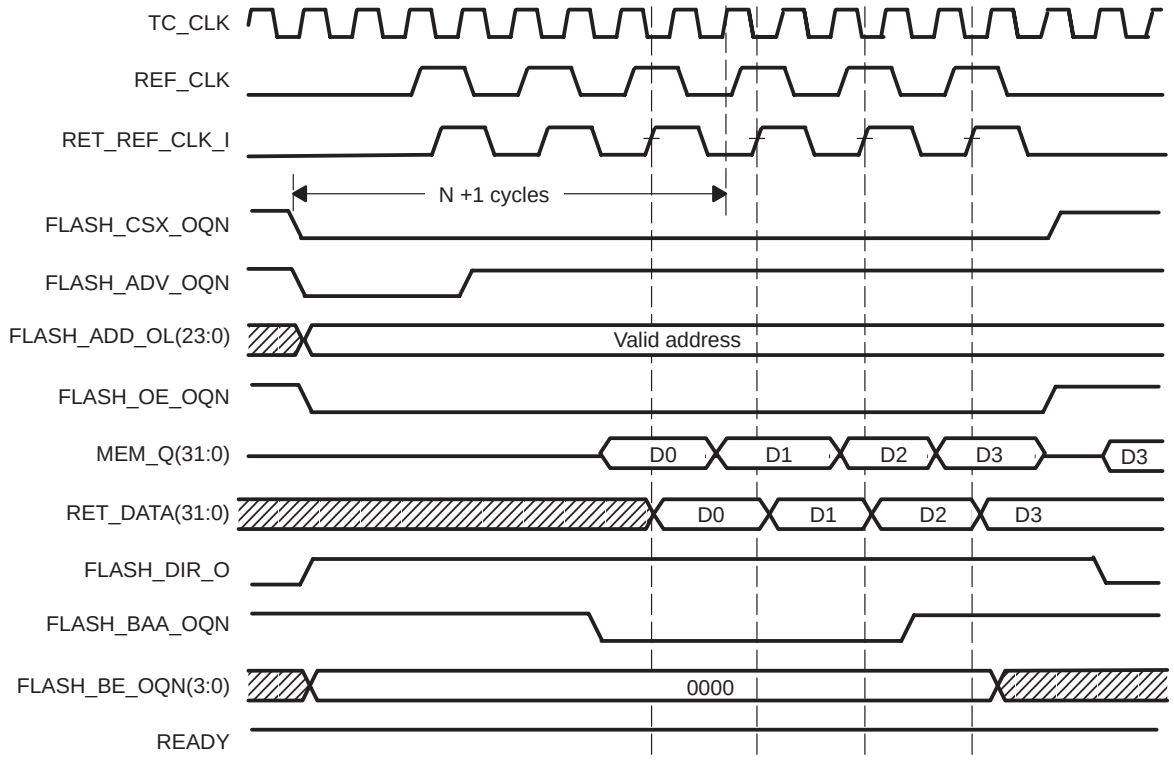
As in non-retimed mode, CS, ADV, Address, BE, OE, and BAA are driven with respect to REF_CLK.

The modes 4 and 5 programming model and protocol behavior remain the same as in non-retimed mode.

In retiming mode, the RDWST is still referenced to REF_CLK. The retiming relaxed timing (extra delay for data valid) is included in the IC timing parameters.

The ready input pin is also retimed with the RET_REF_CLK. The retiming relaxed timing (extra delay for ready valid) is included in the IC timing parameters.

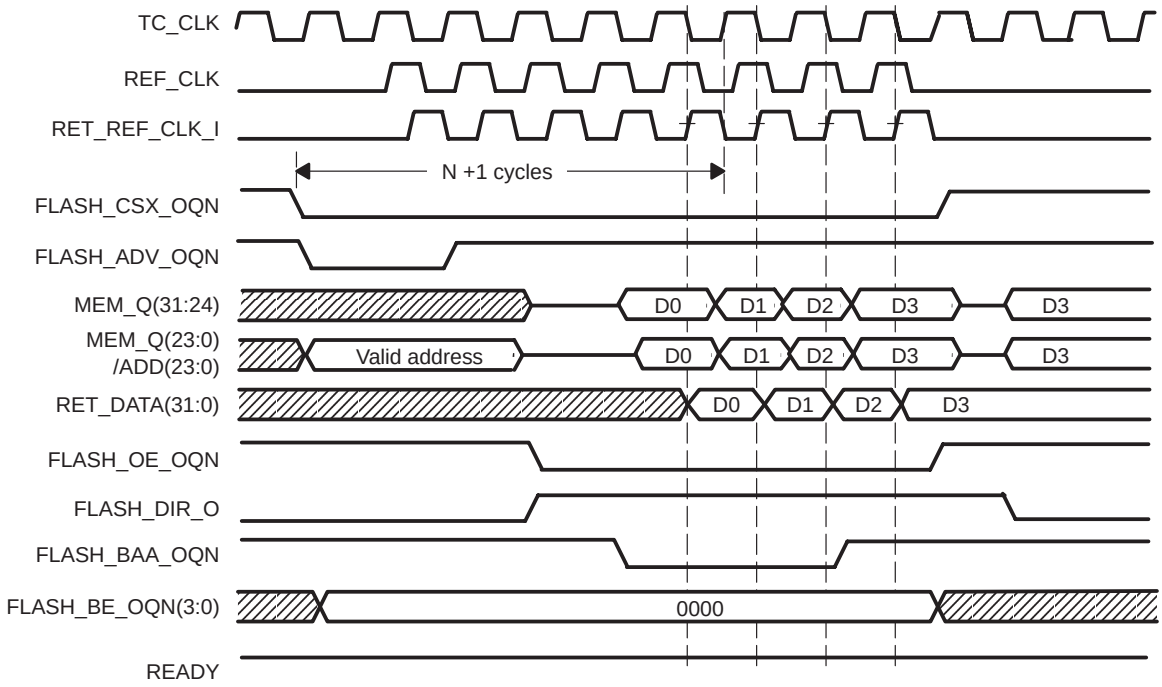
Figure 2–49. Mode 4 Synchronous Burst 4x32-Bit Read Operation on 32-Bit-Wide Device With Retiming on (RDWST=1, FCLKDIV =1, ADVHOLD=0, RDMODE=4)



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

Retiming mode is also available in the multiplexing address and data mode.

Figure 2–50. Mode 4 Synchronous Burst 4x32-Bit Read Operation on Multiplexed Address/Data 32-Bit-Wide Device With Retiming on (RDWST=3, FCLKDIV =0, ADVHOLD=0, OESETUP = 3, RMODE=4). Data write-back on the bus after read completion.



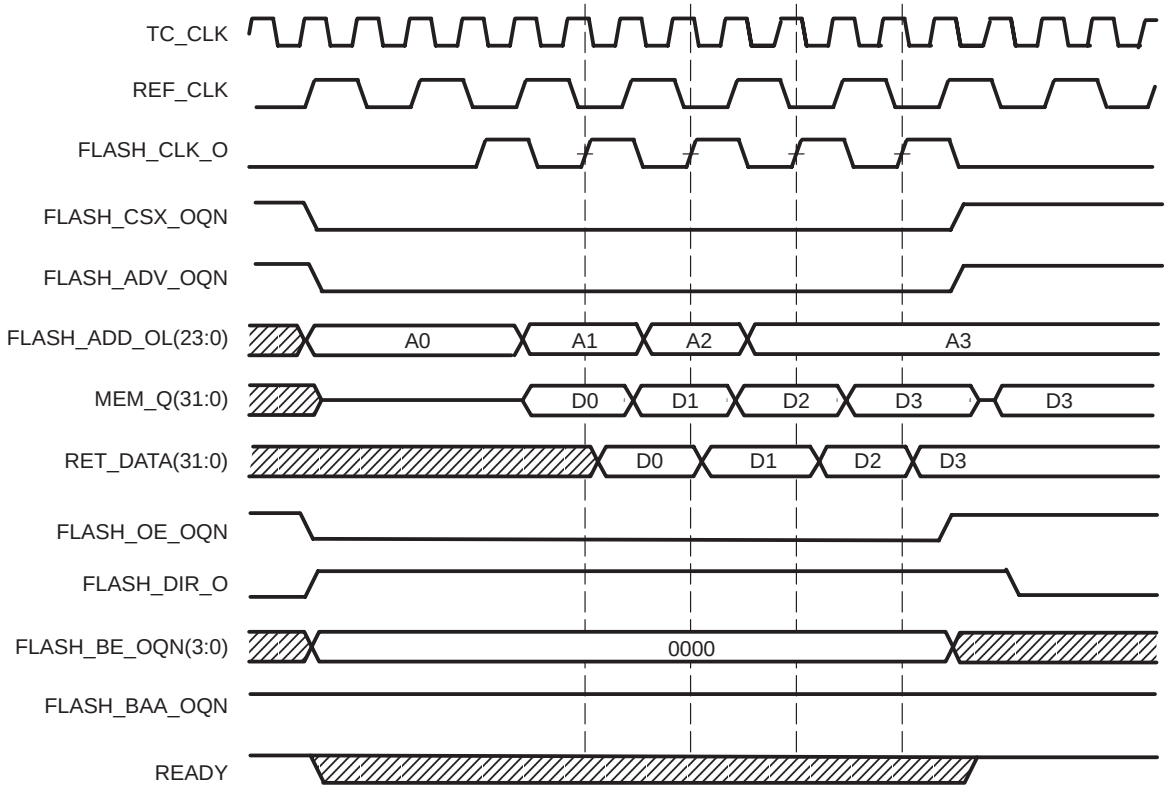
Note: The OMAP730 EMIFS does not support 32-bit wide devices.

2.5.3.15 Mode 7—Synchronous Burst Read Operation Mode

The synchronous burst read mode 7 is selected by setting the RDMODE bit field in the corresponding EMIFS chip-select configuration register (RDMODE = 7)

This mode only supports synchronous read accesses (single or consecutive). It is primarily used to interface IC embedded synchronous ROM and RAM (call TI ACE memory).

Figure 2–51. Mode 7 Synchronous Burst 4x32-Bit Read Operation on 32-Bit-Wide Device (RDMODE = 7, FCLKDIV =1). Data write-back on the bus after read completion.



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

The REF_CLK is divided from TC_CK by a programmable value contained in FCLKDIV bit field of the CS configuration register. REF_CLK is inverted and provided as external FLASH_CLK_O.

The retimed mode must be enabled in mode 7. After reset, the RT bit is set for CS0 and CS3 if mode 7 is selected during the reset boot configuration process.

Mode 7 enables a pipelined read access. Within the burst the addresses are provided one half FLASH_CLK_O cycle before the FLASH_CLK_O rising edge. Use the FLASH_CLK_O rising edge to latch addresses into the embedded ROM/RAM. The ROM/RAM must provide data with enough setup so data can be latched by the EMIFS on the next FLASH_CLK_O rising edge.

REF_CLK frequency must be set so the memory access time can be met in less than one cycle. There is no internal or external wait state control during read access in mode 7. The RDWST field is not active and the ready input pin is not monitored in this mode (non-full-handshaking mode only).

CS, ADV, and OE are driven low for the entire access. ADVHOLD, OESETUP, and OEHOLD time control are disabled in this mode.

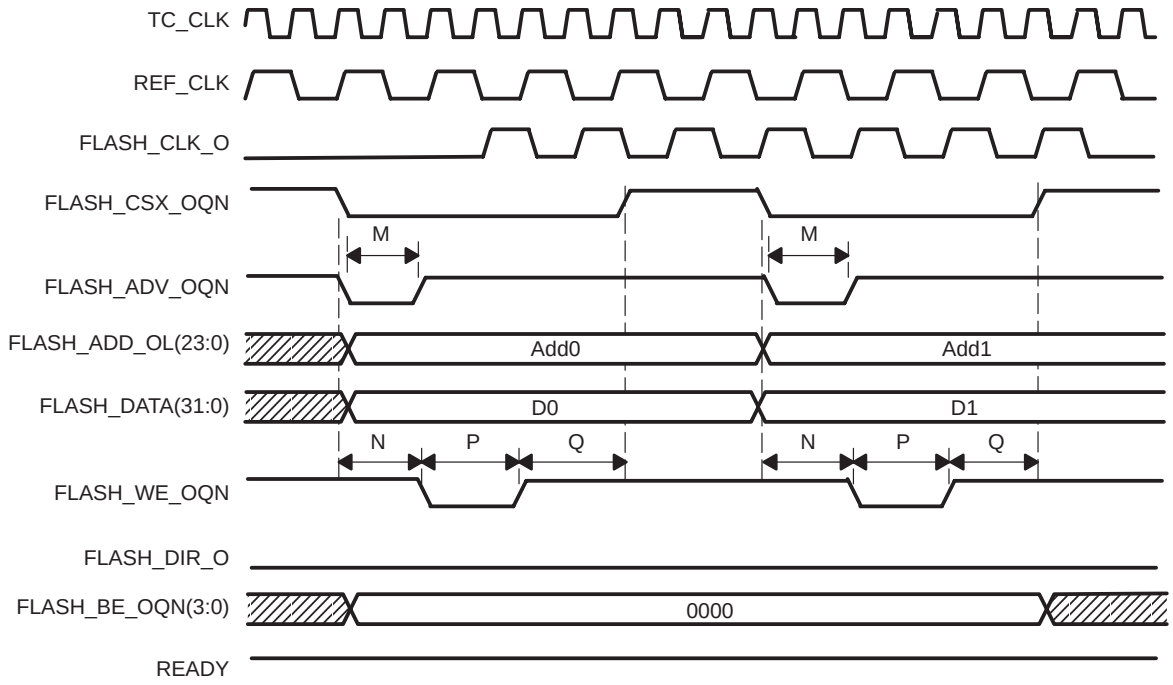
Address and data multiplexed protocol is not supported in mode 7 (the MAD bit field is not considered).

One TC_CK cycle after access completion (CS high) the data bus is driven with the previous read value.

Write Access in Mode 7

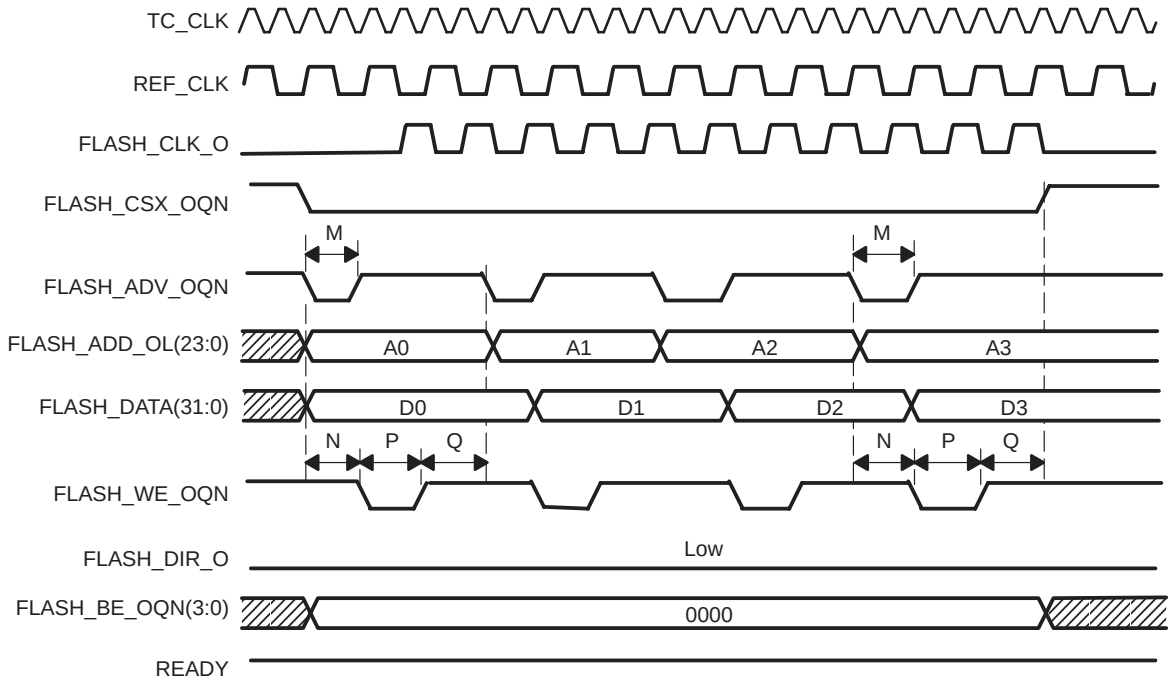
Figure 2–52 shows FLASH_CLK_O activation details during a write access in mode 7. Figure 2–52 shows two successive write accesses with minimum access time.

Figure 2–52. Mode 7 Asynchronous Two Successive 32-Bit Write Operations on a 32-Bit-Wide Device (WRWST=0, WELEN=0 FCLKDIV=1 and ADVHOLD=0)



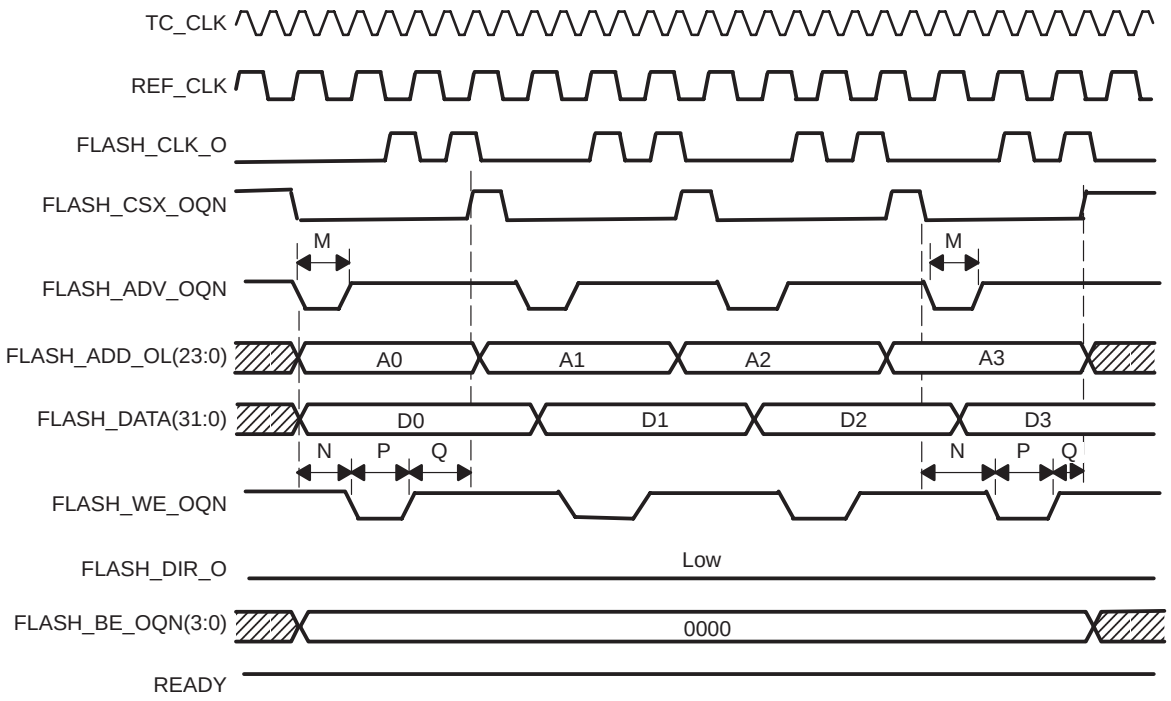
Note: The OMAP730 EMIFS does not support 32-bit wide devices.

Figure 2–53. Mode 7 Asynchronous 32-Bit Burst Write Operations on a 32-Bit-Wide Device (WRWST=0, WELEN=0 FCLKDIV=1 and ADVHOLD=0, BTMODE = 0)



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

Figure 2–54. Mode 7 Asynchronous 32-Bit Burst Write Operations on a 32-Bit-Wide Device (WRWST=0, WELEN=0 FCLKDIV=1 and ADVHOLD=0, BTMODE = 1 and BTWST = 0)



Note: The OMAP730 EMIFS does not support 32-bit wide devices.

2.5.3.16 Bus Turnaround and CS Negation Time Control

When slow devices are attached to the IC, it can be necessary to control the next data bus activation time after a read access to this slow device. This data bus activation time can be either an EMIFS write access or an EMIFS read access to a device attached to a different CS. The minimum idle time before the next CS activation is two TC_CK cycles for independent access (no split or burst accesses). It can be extended to more than two TC_CK cycles by setting the proper value in the BTWST bit field of the CS configuration register attached to the slow device.

- CS pulse width high = (BTWST +1) TC_CK.
- BTWST must be at least 2 to have more than two-TC_CK-cycles idle time

In case of successive read accesses because of EMIFS access size adaptation, or in case of burst read access in mode 0, the CS negation time between the successive read accesses is at least one TC_CK cycle. It can also be extended by the BTWST field.

After a read completion, if no other access (RD, WR) is pending, the data bus is driven with the previous read value. The bus turnaround time (OE going high to direction going out) is a minimum of one TC_CK cycle and can be extended through BTWST.

Table 2–36 shows the bus turnaround cycles inserted for various transitions with EMIFS when BTMODE=0.

Table 2–36. Idle Time Between Different Bus Access Transitions (BTMODE = 0)

Access(n)	Access(n+1)	Chip-Select	Idle Time	Length(BTWST)
RD(csx)	RD(csx)	Same	Inserted	CSX
RD(csx)	WR(csx)	Same	Inserted	CSX
WR(csx)	RD(csx)	Same	Not inserted	-
WR(csx)	WR(csx)	Same	Not inserted	-
RD(csx)	RD(csy) x != y	Different	Inserted	CSX
RD(csx)	WR(csy) x != y	Different	Inserted	CSX
WR(csx)	RD(csy)x != y	Different	Not inserted	-
WR(csx)	WR(csy)x !=y	Different	Not inserted	-

Figure 2–55. Wait States During a Read to Read Operation (BTWST (CSX) = 2 and BTWST (CSY) = 1, BTMODE=0)

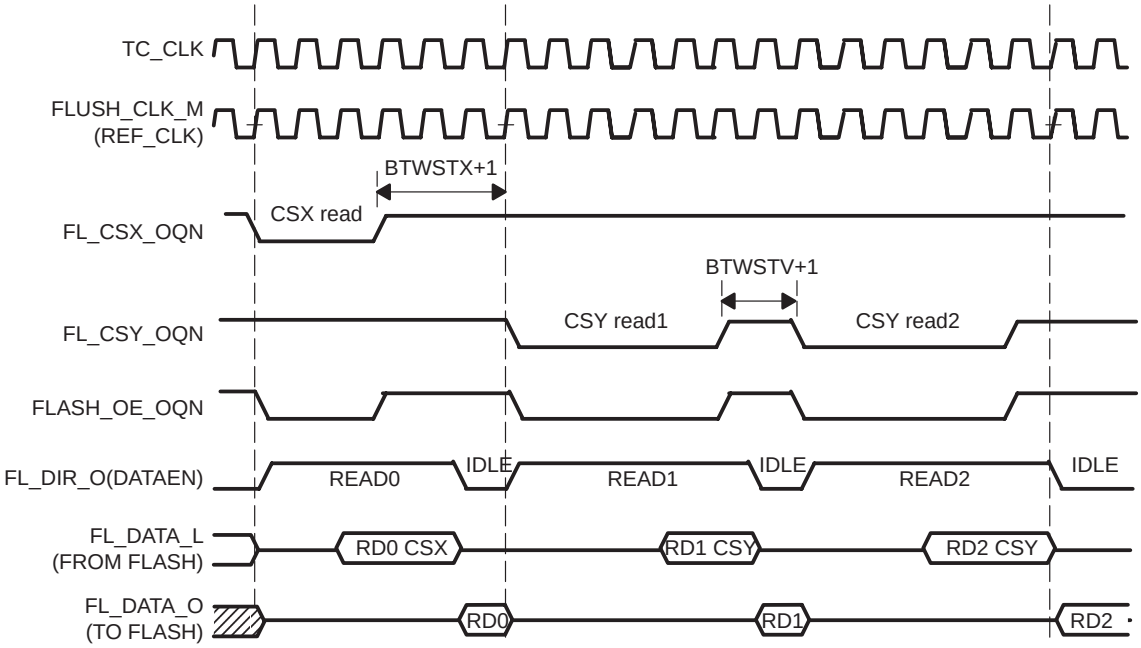
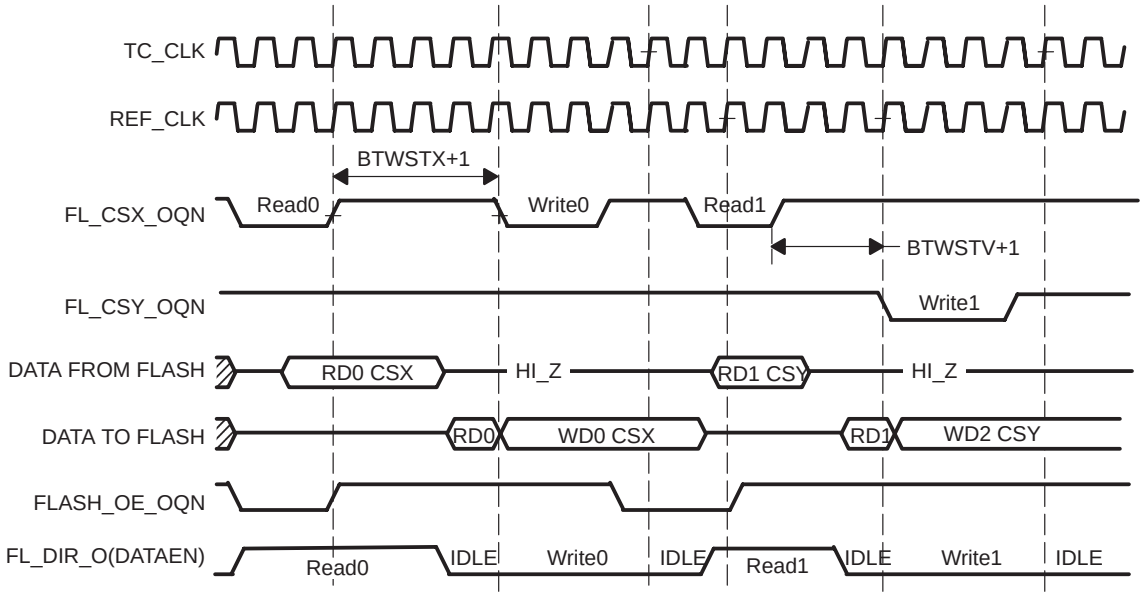


Figure 2–56. Wait States During a Read to Write Transition (BTWST(CSX)= 3 BTWST (CSY) = 2, BTMODE=0)



The BTMODE field in the advance CS configuration register extends the previous mode. When BTMODE = 1, the BTWST bit field controls the CS negation time between any type of successive accesses to the same CS.

In case of successive write accesses because of EMIFS access size adaptation or in case of write burst read access, there is no CS negation time between successive write accesses unless BTMODE is set. When BTMODE is set, the

CS negation time between the successive write accesses is at least one TC_CK cycle, and it can be extended by the BTWST field in the CS configuration (BTMODE set or clear).

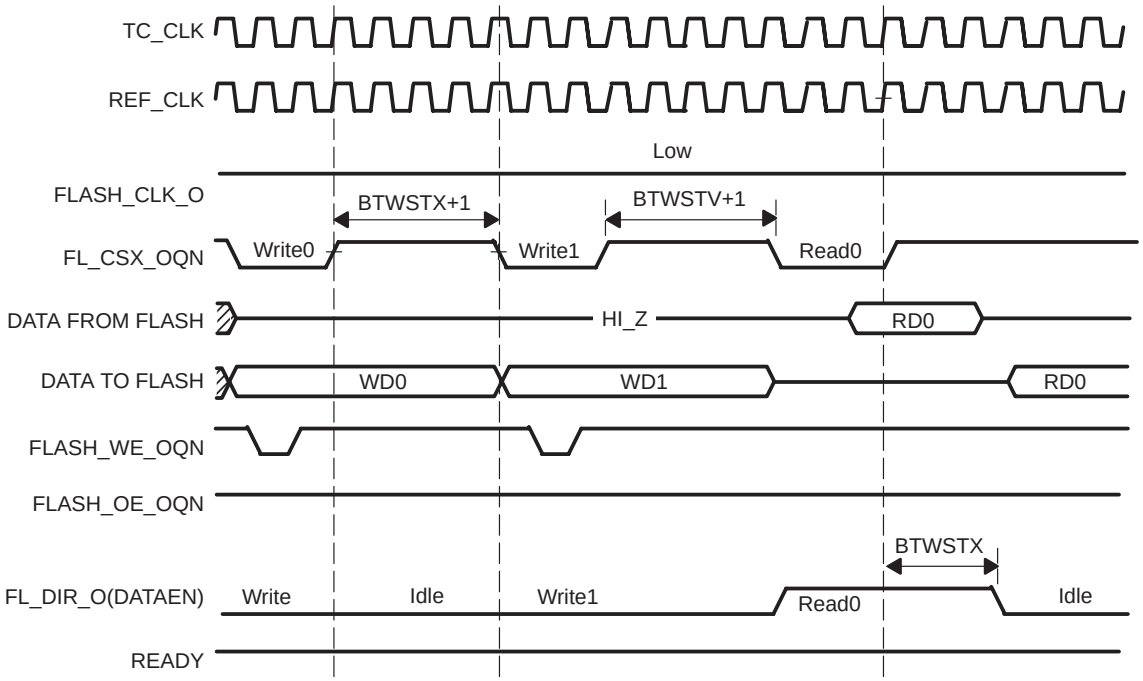
CS pulse width high= (BTWST +1) TC_CK

Table 2–37 shows the idle cycles inserted for various transitions with EMIFS when BTMODE = 1.

Table 2–37. Idle Time Between Different Bus Access Transitions (BTMODE = 1)

Access (n)	Access (n+1)	Chip-Select	Idle Time	Length (BTWST)
RD(csx)	RD(csx)	Same	Inserted	CSX
RD(csx)	WR(csx)	Same	Inserted	CSX
WR(csx)	RD(csx)	Same	Inserted	CSX
WR(csx)	WR(csx)	Same	Inserted	CSX
RD(csx)	RD(csy) x != y	Different	Inserted	CSX
RD(csx)	WR(csy) x != y	Different	Inserted	CSX
WR(csx)	RD(csy)x != y	Different	Not inserted	-
WR(csx)	WR(csy)x !=y	Different	Not inserted	-

Figure 2–57. Wait States During a Write-to-Write and Write-to-Read Transition to Same Chip-Select (BTWST CSX = 3 BTMODE = 1)



2.5.3.17 External Device Reset Control

Common flash memory supports the reset input pin to allow the device state machine to be reset properly on power up and also to minimize power consumption in idle mode. The EMIFS interface includes the RSTPWR output signal that is controlled by the OMAP clock and reset module. The RSTPWR output pin is activated during OMAP warm and cold reset. The RSTPWR output pin is also activated when the TC enters the idle state, if the RESPWREN bit field in the clock and the reset ARM_EWPCT configuration register is set.

ARM_EWPCT bit field in the clock and reset configuration register is used to specify the minimum time between the RESPWR deasserted and the TC going out from idle state (see OMAP TRM chapter 4).

2.5.3.18 Dynamic Autoidle and System Idle Synchronization

EMIFS supports autoidle mode (clock gating) to dynamically reduce power consumption when no requests are pending and no accesses are ongoing. The dynamic autoidle mode is enabled by setting to 1 the PWD_EN bit field of the EMIFS configuration register.

Upon clock and reset idle request, the EMIFS can send the idle request acknowledge when all ongoing transactions are completed. This allows the clock and reset module to cut the EMIFS source clock properly. The idle request acknowledge process is enabled by setting to 1 the PDE bit field of the EMIFS configuration register.

2.5.3.19 Abort Management

The EMIFS can issue an interrupt in two cases of abort scenario:

- Restricted access mode violation on CS0
  - CS0_ARM_ONLY input pin enables the restricted access mode to prevent any initiators other than MPU to access the CS0 space.
  - When an access to the CS0 space is initiated by the DMA, modem, or OCP-I, the EMIFS completes the access by giving back a 0 value as the read value or by not writing the given value to the final destination.
  - The EMIF raises an interrupt, sets the abort flag bit, sets the restricted access error status bit, and updates the host ID bit field in the abort type register. The EMIFS updates the abort address register with the requested address that is causing the access violation.
  - The abort flag bit is cleared when the abort type register is read.
- Time-out issue during any access in case of full-handshaking mode
  - This feature can be enabled with the TIMEOUT_EN bit field and with an 8-bit time-out programmable value (expressed in REF_CLK cycles) in the EMIFS abort time-out register.
  - On the beginning of an access, the time-out counter starts counting down. If the counter reaches 0 before the device responds, the EMIF raises an interrupt, sets the abort flag bit, sets the time out error status

bit, and updates the host ID bit field in the abort type register. The EMIFS updates the abort address register with the requested address that is causing the access time out.

- The abort flag bit is cleared when the abort type register is read.
- At reset, TIMEOUT_EN is set to 1 and the time-out value is set to 255 REF_CLK cycles.

2.5.3.20 EMIFS Boot Mode

Table 2–38 shows the EMIFS boot-mode input pins.

Table 2–38. EMIFS Boot-Mode Input Pins

OMAP710 Input Pins			
DEVICE_TYPE	**ARM_BOOT_MODE_ MLPG2**	**MUX_MODE_MLPG1**	**Comments**
00: Production 01: Bad 11: Test	X	X	Boot from CS0 in synchronous 32-bit mode, nonmultiplexed mode
10:Emulation	0	X	Boot from CS0 in synchronous 32-bit mode, nonmultiplexed mode
10: Emulation	1	0	Boot from CS3 in asynchronous 16-bit mode, nonmultiplexed mode
10: Emulation	1	1	Boot from CS3 in asynchronous 16-bit mode, multiplexed mode

Table 2–39. CS1 and CS2 Configuration Register Reset Value

Bit	Field	Description	Reset Value
31	PGWSTEN		0
30:27	PGWST		0
26:23	BTWST		0
22	MAD		0
21	FL		0
20	Reserved		1
19	Reserved		0
18:16	RDMODE		0
15:12	PGWST/WELEN		1
11:8	WRWST		1
7:4	RDWST		1
3	Reserved		1
2	RT		1
1:0	FCLKDIV		1

2.5.4 EMIFF Programming

This section describes the EMIFF programming.

Note:

The OMAP730 does not support DDR.

2.5.4.1 Main Features

The OMAP EMIFF is an SDRAM controller that manages all accesses by the various initiators of an OMAP-based system.

It can support one 16-bit device or two 8-bit devices. The external interface data bus width is always 16 bits.

The following devices are supported:

- Standard single-data-rate SDRAM
- Low-power single-data-rate SDRAM
- Mobile double-data-rate SDRAM

In terms of capacity and organization for the memory components that can be attached, the EMIFF can handle:

- 1G-bit, 512M-bit, 256M-bit, 128M-bit, 64M-bit, and 16M-bit devices
- Two-bank or four-bank devices for 16M-bit or 64M-bit devices; four-bank devices-only for any other capacity
- x8 (two devices) or x16 (single device) data bus configuration, except for the 512M-bit and 1M-bit device. The EMIFF only supports two x8-512M-bit devices and a single x16-1M-bit device (64 megabytes, maximum memory configuration)

The SDRAM_TYPE field of the EMIFF interface SDRAM configuration register must be used to specify the physical configuration of the devices.

The SDRAM type selection is the first action required from the software driver, using the SDRAM_TYPE field of the EMIF SDRAM operation register. The types supported are regular SDR SDRAM, low-power SDRAM, regular DDR SDRAM (JEDEC DDR1), and mobile DDR SDRAM.

The SDRAM controller supports:

- The self-refresh mode (idle), autorefresh, and other operating modes (HPHB, LPLB, and POM0).
- MRS command and extended MRS command for DDR SDRAM and low-power SDRAM, sent via the SDRAM request manager
- For SDR SDRAM, all burst sizes, between 1 and 32 consecutive accesses
- For DDR SDRAM, only bursts of 8
- Two pipe-lined levels of request from the SDRAM request manager to enable page interleave timing and reduce overhead cycles by the burst interruption mechanism.

2.5.4.2 Initialization Sequence

Before it can be accessed for writing or reading after power on, or when exiting deep power-down mode, an SDRAM device must be initialized with various protocol parameters (burst size, CAS idle time, write burst, and so on). This is known in SDRAM literature as the *initialization sequence*.

Because SDR SDRAM and DDR SDRAM require different initialization sequences, the sequence is manually software driven, using the EMIFF SDRAM manual command register (writing a command opcode into that register generates that command on the SDRAM interface).

The SDRAM Operations register must only be programmed after the a precharge all command has been issued.

For SDR SDRAM devices, the initialization of the SDRAM bank is accomplished via a preprogrammed series of writes to the command register and the MRS register. Two possibilities can generate the initialization sequence:

- Writing to the EMIFF SDRAM MRS register (legacy) in the SDRAM controller sets up the device to interface with OMAP. The following SDRAM command sequence is automatically generated:

1) Precharge all
2) Load mode register
3) Autorefresh

The EMIFF SDRAM MRS register (legacy) is provided for compatibility with existing code and must not be used by new software drivers.

- The new recommended procedure is:

1) Wait for 200 µs after power up and memory clock is running and stable. The software is responsible for this initial wait. During this wait, NOP commands are the default commands automatically sent to the memory.
2) Apply a PRECHARGE ALL command. This is accomplished by writing to the command register.
3) Apply two AUTOREFRESH commands. This is accomplished by two consecutive writes to the command register.
4) Apply an MRS LOAD command. This is accomplished by writing to the EMIFF SDRAM MRS register (new) (uses the new address, not the legacy address).

This procedure is also suitable for the mobile DDR device, because this kind of device does not include delay-locked loop technology (DLL).

For standard DDR-SDRAM devices (JEDEC DDR1, DLL included), the typical initialization sequence is:

1) At power up or when exiting deep power-down mode, maintain CKE at a low state.
2) The clock must be maintained stable for a minimum of 200 µs. The software is responsible for this initial wait.

3) During this minimum 200 µs after clock stable, NOP command is presented by default to the memory device (it is not necessary to generate NOP commands using the command register).

4) Set CKE high using the command register.

5) Precharge all banks. This is accomplished by writing to the command register.

6) Issue an EMRS LOAD command to enable DLL. This is accomplished by writing the appropriate value to the EMRS1 register.

7) Wait 200 clock cycles to enable DLL lock.

8) Precharge all banks. This is accomplished by writing to the command register.

9) Issue two (or more if required by the device used) AUTOREFRESH commands. This is accomplished by writing twice to the command register.

10) Issue the MRS LOAD command (initialize DLL for normal operation; other parameters are unchanged). This is accomplished by writing to the EMIFF SDRAM MRS register (new) register (new address).

2.5.4.3 Memory Mode Registers

The following restrictions apply to the MRS register programming:

- ❏ Burst length must be programmed to full page for SDR devices.
- ❏ For standard DDR devices, burst length must be programmed to 8 and the CAS idle time must be set to 2.
- ❏ For mobile DDR devices, burst length must be programmed to 8 and the CAS idle time must be set to 3.

All SDRAM internal mode registers are mirrored in the EMIFF controller. Standard MRS is provided. EMRS0 is provided for the DDR EMRS register. EMRS1 is provided for low-power SDRAM new features (such as partial array self-refresh, or temperature compensated self-refresh). EMRS2 is provided for future use.

For each register write access, a 12-bit data value is loaded in the memory device to support future new option bits in the existing registers. Always write 0 in all reserved bits.

When reading to these registers, the data in the mirrored register is returned.

2.5.4.4 EMIFF SDRAM Configuration

The EMIFF SDRAM configuration register must then be programmed to define the other parameters of the interface:

- ❏ Power-down strategy. When the PWD bit is set to 1, the power-down state is automatically entered between memory accesses. When there is no ac-

tive transaction on the interface, CKE goes low. Any new access request awakens the device before making the access.

- ❏ SDRAM autorefresh control. To optimize SDRAM bandwidth usage for data transfer, it is preferable to generate a sequence of autorefresh requests rather than a single autorefresh request. A sequence of four or eight successive autorefreshes can be programmed.

The autorefresh burst request is generated when a 16-bit refresh timer (see the SDRAM configuration register) reaches the following user-defined values, according to selected SDRAM frequency (via SDRAM configuration register):

- ❏ Memory size and configuration
- ❏ Entering self-refresh (SLRF bit). Self-refresh can also be controlled using the command register; the two methods are equivalent. The self-refresh mode is automatically exited when the EMIFF receives an access request from OMAP initiators.

You can set the refresh counter value corresponding to system frequency. The following formula can be used for the refresh counter value:

Counter value = (((64 ms (refresh rate) / Number of row) / t_f) * (burst refresh size)) - 400 cycles (margin due to a possible transfer currently starting when the refresh request occurs).

Where t_f = 1/system frequency [ns] (system freq = TC_CK freq)

Example: 100-MHz system clock, 4096 rows and 8-burst refresh

t_f = 10 ns

Counter value = (((64.10^6 ns/4096)/10) * 8) - 400

Counter value = 12100 cycles (0x2F44)

2.5.4.5 SDRAM Interface ac Parameters

The 2-bit SDRAM frequency (SDF) field in the EMIFF configuration register allows for selection of a set of ac timings parameters for the interface, depending on the type and speed grade of the memory component used. These parameters are commonly used in the SDRAM literature. Because the naming can change from one memory manufacturer to another, Table 2–40 provides the functional description of each individual parameter, as named in this specification document.

Table 2–40. ac Parameters Description

tRC	Active to active command period
tRAS	Active to precharge command period
tRP	Precharge command period
tRCD	Active to write/read delay
tRRD	Active bank a to active bank b command

Four ac configurations are provided for standard SDR and DDR1 memory components and another set of four configurations is provided for the mobile DDR memory. The selection between these two sets is done automatically, based on the memory type programmed in the EMIFF operation register.

All parameters are individually specified in the number of clock cycles, whereas in the memory data sheets, absolute values are used for timing parameters. For a given working frequency, any ac parameter of the selected configuration, once converted into an absolute timing value, must be greater than the equivalent parameter specified by the memory manufacturer.

2.5.4.6 Page-Closing Strategy

The SDRAM operation register that selects between SDR and DDR also allows selection of a page-closing strategy. Three variants are available:

- High-power/high-bandwidth mode (HPHB)

  In this mode of operation, *the active page is always left open at the end of access*. The EMIFF controller keeps track of the open pages. It then determines if the current access is to an opened page (access proceeds without latency) or to a closed page (the currently active page within that internal bank is closed, the necessary page is activated, and the access proceeds after a necessary latency). This mode of operation is useful in situations where power consumption is not critical, but the bandwidth achieved is.

- Low-power/low-bandwidth mode (LPLB)

  In this mode of operation, *the page being accessed is always closed at the end of the current access* (the read and write exit with an autoprecharge command). For every access, the page being accessed must be activated at the beginning of the access, which causes extra cycles of latency and lowers the memory bandwidth. Though this may not be an efficient or optimal operating mode, this scheme is useful in situations where minimizing power consumption is of primary concern. The power consumption is minimized because only the currently accessed page is open, and all pages are closed during idle cycles.

- Programmable operating mode (POM0)

  In this mode of operation, *a time-out register is associated with each open page*. Because the maximum number of simultaneous open pages is four (one per bank), four time-out registers are provided. Upon an access to an open page, the time-out register is set to the user-programmed value and starts to count down. If the counter reaches zero, the controller then closes the inactive page. This mode of operation has better bandwidth performance than the low-power/low-bandwidth operating mode and does not consume power as heavily as the high-bandwidth/high-power mode.

As mentioned previously, the command register is used during the initialization sequence, but also controls the deep power mode (enter/exit commands) if supported by the memory device. Enter-self-refresh and exit-self-refresh commands are also available.

Arbitration in EMIFF is performed according to the priority algorithms described in Section 2.5.10, *Priority Algorithms.*

2.5.5 OCP-I Programming

The OMAP 3.2 OCP external initiator port is an interface intended to connect an external master device to the OMAP 3.2 platform (typically an external CPU or a peripheral with DMA-mastering capabilities). The OMAP 3.2 core appears as a slave, and its entire memory map, including TIPB peripherals, is accessible. The interconnect bus is compliant with the OCP specification.

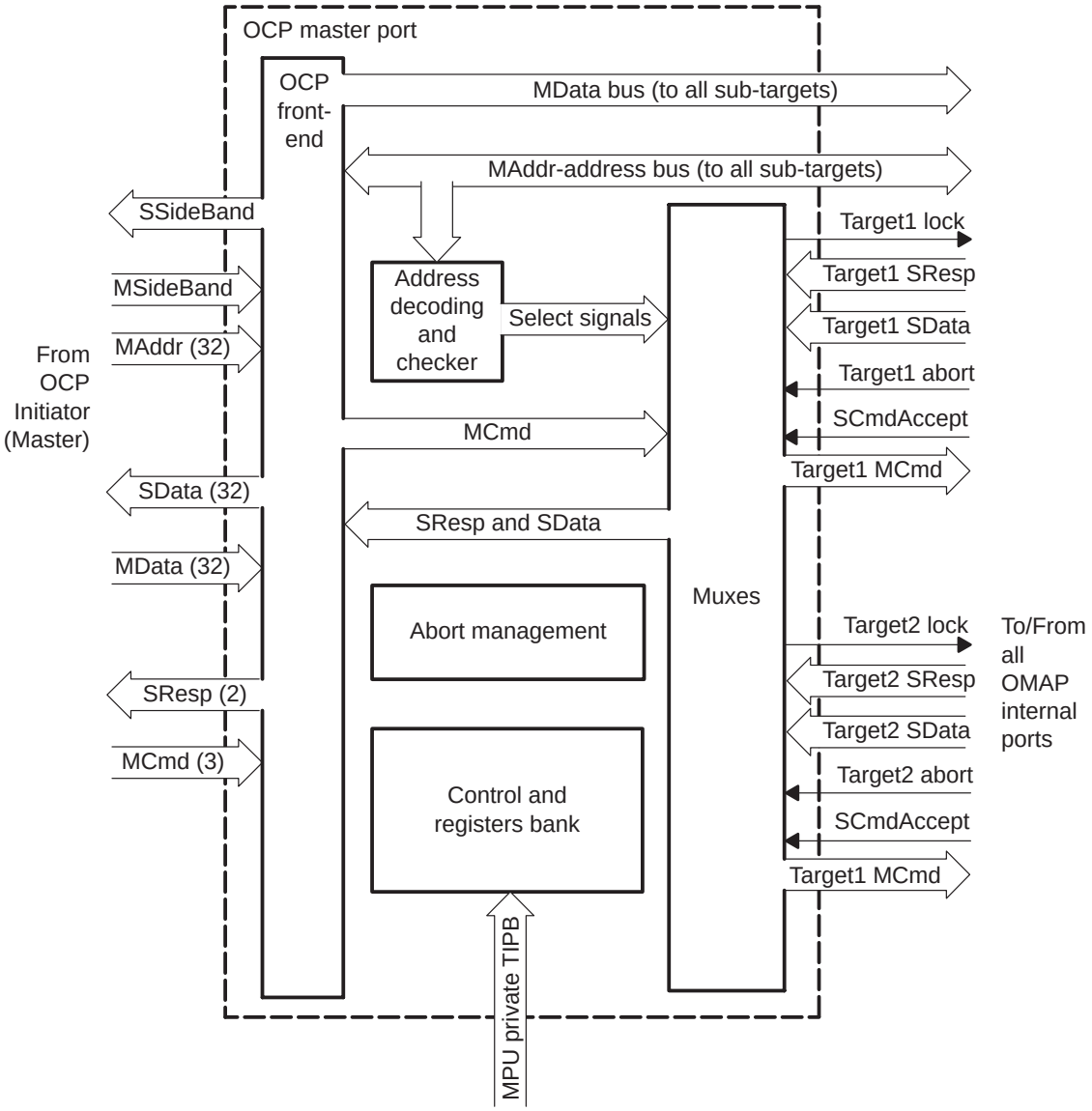
The OCP initiator is capable of single transfers in 8, 16, and 32 bits and burst mode transfers in 32 bits.

OCP-I provides access to the following OMAP targets:

- EMIFS (external slow memories)
- EMIFF (external fast memories)
- OCPT1
- OCPT2
- Multibank OCPT1/2
- MPUI (not part of the TC; see Chapter 8, *MPUI*)
- MPU private TIPB bridge (not part of the TC; see Chapter 10, *TIPB Bridge*)
- MPU public TIPB bridge (not part of the TC; see Chapter 10, *TIPB Bridge*)

Figure 2–58 shows the OCP-I block diagram.

Figure 2–58. OCP-I Block Diagram



Though OCP-I has visibility to all OMAP resources, a protection register allows validation of the access permission on a per-target basis. Any access to an unauthorized resource generates an abort.

The MPU controls the OCP-I port through its private TIPB; if permission is given to the external initiator to access the private TIPB. The external master also accesses the OCP-I configuration registers.

If OS protection is needed, it is recommended that any device integrating the OMAP 3.2 core provide an MMU or a translation table controlled by the OMAP MPU. Then it is the responsibility of the system software to ensure that undefined or unauthorized locations are not accessed.

An internal address checker continuously compares the OCP address bus and the OMAP 3.2 memory map. An abort signal is provided if the transaction is not correct.

OCPI contains other status and control registers described in Section 2.5.9, *OCP-I Registers*.

2.5.5.1 Address and Command Fault Registers

When a non-OMAP-defined address is detected or a bus error occurs from the subtargets on a write access, an abort is sent to the initiator to indicate that the current access cannot be completed. The access to the OMAP subtarget is terminated and the address and command buses are stored in the fault registers.

2.5.5.2 Abort Type Register

The type of the abort event is reported in a status register.

The interrupt handler can clear this register and the interrupt request by reading any value to this register.

2.5.5.3 Interrupt Registers

It is possible for the OMAP MPU to trigger two interrupt lines to the external OCP initiator(s) by using the interrupt registers (SINTERRUPT0 and SINTERRUPT1). Edge-sensitive or level-sensitive can be used.

2.5.6 Traffic Controller Registers

TC registers are distributed into the traffic controller submodules.

Note:

Before accessing the TC registers, the TIPB bridge write buffer must be activated.

Table 2–41 lists the 32-bit traffic controller registers. Table 2–42 through Table 2–49 describe the register bits.

Table 2–41. OCP-T1/OCP-T2 Registers

Name	Description	R/W	Address (0xFFFE:)
OCPT1_PRIOR	OCP-T1 LRU priority	R/W	CC00
OCPT1_PTOR1	OCP-T1 dynamic priority time-out 1	R/W	CCA0
OCPT1_PTOR2	OCP-T1 dynamic priority time-out 2	R/W	CCA4
OCPT1_PTOR3	OCP-T1 dynamic priority time-out 3	R/W	CCA8
OCPT1_ATOR	OCP-T1 abort time-out	R/W	CCAC
OCPT1_ADDR	OCP-T1 abort address	R	CCB0
OCPT1_ATYPER	OCP-T1 abort type	R	CCB4
OCPT2_PRIOR	OCP-T2 LRU priority	R/W	CCD0
OCPT2_PTOR1	OCP-T2 dynamic priority time-out 1	R/W	CCD4

Table 2–41. OCP-T1/OCP-T2 Registers (Continued)

Name	Description	R/W	Address (0xFFFE:)
OCPT2_PTOR2	OCP-T2 dynamic priority time-out 2	R/W	CCD8
OCPT2_PTOR3	OCP-T2 dynamic priority time-out 3	R/W	CCDC
OCPT2_ATOR	OCP-T2 abort time-out	R/W	CCE0
OCPT2_ADDR	OCP-T2 abort address	R	CCE4
OCPT2_ATYPER	OCP-T2 abort type	R	CCE8
OCPT_CONFIG_REG	Configuration	R/W	CCB8

Table 2–42. OCP Priority Registers (OCPT1_PRIOR and OCPT2_PRIOR)

Bits	Field	Description	Reset Value
31:16		Reserved. To ensure software compatibility, reserved bit must be written to 0 and read value is considered undefined.	0x0000
15:12	OCP_PRIORITY	Number of consecutive accesses allowed for OCP-I	0b0000
11:8	DMA_PRIORITY	Number of consecutive accesses allowed for DMA	0b0000
7		Reserved	
6:4	DSP_PRIORITY	Number of consecutive accesses allowed for modem	0b000
3		Reserved	
2:0	ARM_PRIORITY	Number of consecutive accesses allowed for MPU	0b000

The OCP priority registers (OCPTx_PRIOR) in Table 2–42 allow the target to give consecutive accesses to a host when the host is granted the OCP target.

The MPU and DSP can have from 0 to 7 consecutive accesses. The DMA and the OCP initiator can have 0 to 15 consecutive accesses, depending on the content of corresponding bits in their priority registers.

The priority time-out registers in Table 2–43 through Table 2–45 store the number of TC_CK clock cycles before modem, DMA, LCD, or OCPI requests are made high priority in the dynamic priority scheme for the OCP target.

Table 2–43.OCP Priority Time-Out Registers 1 (OCPT1_PTOR1, (OCPT2_PTOR1)

Bits	Field	Description	Reset Value
31:8		Reserved. To ensure software compatibility, reserved bit must be written to 0 and read value is considered undefined.	0x00 0000
7:0	DMA	Number of TC_CK cycles that DMA must wait in low-priority queue before going to high-priority queue	0x00

Table 2–44. OCP Priority Time-Out Registers 2 (OCPT1_PTOR2, (OCPT2_PTOR2)

Bits	Field	Description	Reset Value
31:24		Reserved. To ensure software compatibility, reserved bit must be written to 0 and read value should be considered undefined.	0x00
23:16	DSP	Number of TC_CK cycles that the modem must wait in low-priority queue before going to high-priority queue	0x00
15:8		Reserved. To ensure software compatibility, reserved bit must be written to 0 and read value is considered undefined.	0x00
7:0	LCD	Number of TC_CK cycles that LCD must wait in low-priority queue before going to high-priority queue	0x00

Table 2–45. OCP Priority Time-Out Registers 3 (OCPT1_PTOR3, (OCPT2_PTOR3)

Bits	Field	Description	Reset Value
31:8		Reserved. To ensure software compatibility, reserved bit must be written to 0 and read value is considered undefined.	0x00 0000
7:0	OCP-I	Number of TC_CK cycles that OCP-I must wait in low-priority queue before going to high-priority queue	0x00

Table 2–46. OCP Abort Time-Out Registers (OCPT1_ATOR, OCPT2_ATOR)

Bits	Field	Description	Reset Value
31:9		Reserved. To ensure software compatibility, reserved bit must be written to 0 and read value is considered undefined.	0x000000
8	TIMEOUT_EN	Enable time-out bit 1: Enable 0: Disable	1
7:0	TIMEOUT	Number of counted-down clock cycles before sending out abort signal if there is no response from the slave	0xFF

The abort time-out registers store the number of clock cycles the OCP target counts down before activating the abort signal because the peripheral did not return an SRESP signal.

Table 2–47. OCP Abort Address Registers (OCPT1_ADDR, OCPT2_ADDR)

Bits	Field	Description	Reset Value
31:0	Address	Address of access that caused abort	0x0000 0000

Table 2–48. OCP Abort Type Register (OCPT1_ATYPER, OCPT2_ATYPER)

Bits	Field	Description	Reset Value
31:5	Reserved	Reserved. To ensure software compatibility, reserved bit must be written to 0 and read value is considered undefined.	0x0000000
4	TIMEOUT_ERR	Abort generated by time-out register 0: Abort 1: No abort	0
3	BUS_ERR	Abort generated by error comes from external peripherals 0: Abort 1: No abort	0
2:1	HOST_ID	Host ID of request that caused memory fault 00: MPU 01: Modem 10: DMA 11: OCPI	0b00
0	ABORT_FLAG	0: No abort 1: Abort	0

Table 2–49. Configuration Register (CONFIG_REG)

Bits	Field	Description	R/W	Reset Value
31:2	Reserved	Reserved, must be all 0s	R/W	0x00000000
1	PIPELN_RD_EN	0: Pipeline read operation disabled 1: Pipeline read operation enabled	R	1
0	AUTO_GATED_CLK	0: Autogating clock disabled 1: Autogating clock feature enabled to save power	R	0

The configuration register contains the PIPELN_RD bit to enable or disable the pipeline read operation in OCPTarget.

2.5.7 EMIFS Registers

Table 2–50 lists the 32-bit EMIFS registers. Table 2–51 through Table 2–63 describe the register bits.

Table 2–50. EMIFS Registers

Name	Description	R/W	Offset
EMIFS_LRUREG	EMIFS LRU priority	R/W	0x04
EMIFS_CONFIG	EMIFS configuration	R/W	0x0C
FLASH_CFG_0	EMIFS chip-select configuration nCS0	R/W	0x10
FLASH_CFG_1	EMIFS chip-select configuration nCS1	R/W	0x14
FLASH_CFG_2	EMIFS chip-select configuration nCS2	R/W	0x18
FLASH_CFG_3	EMIFS chip-select configuration nCS3	R/W	0x1C
FL_CFG_DYN_WAIT	EMIFS dynamic wait states	R/W	0x40
EMIFS_TIMEOUT1_REG	EMIFS dynamic priority time-out 1	R/W	0x28
EMIFS_TIMEOUT2_REG	EMIFS dynamic priority time-out 2	R/W	0x2C
EMIFS_TIMEOUT3_REG	EMIFS dynamic priority time-out 3	R/W	0x30
EMIFS_ABORT_ADDR	EMIFS abort address	R	0x44
EMIFS_ABORT_TYPE	EMIFS abort type	R	0x48
EMIFS_ABORT_TOUT	EMIFS abort time-out	R/W	0x4C
FLASH_ACFG_0_1	Advanced EMIFS chip-select configuration nCS0	R/W	0x50
FLASH_ACFG_1_1	Advanced EMIFS chip-select configuration nCS1	R/W	0x54
FLASH_ACFG_2_1	Advanced EMIFS chip-select configuration nCS2	R/W	0x58
FLASH_ACFG_3_1	Advanced EMIFS chip-select configuration nCS3	R/W	0x5C

Note: The EMIFS chip-select configuration register reset values depend on the input boot pin state at IC reset release time. For more details, see Section 2.5.3.20, *EMIFS Boot Mode*.

Table 2–51. EMIFS LRU Priority Register (EMIFS_LRUREG)

Bit	Name	Function	Reset Value
31:16	Reserved	Reserved. To ensure software compatibility, the reserved bit must be written to 0, and the read value is considered undefined.	0x0000
15:12	OCPI	OCPI consecutive access	0x0
11:8	DMA	DMA consecutive access	0x0
6:4	DSP	Modem consecutive access	0x0
2:0	MPU	MPU consecutive access	0x0

The EMIFS LRU priority register allows the EMIFS to give consecutive accesses to a master when EMIFS is configured for least recently used (LRU) priority arbitration.

The MPU and the modem can have from 0 to 7 consecutive accesses, whereas the DMA and the OCP-I can have 0 to 15 consecutive accesses, according to the content of the corresponding bits. In case of burst accesses, a four 32-bit burst access is considered as one access.

Table 2–52. EMIFS Configuration Register (EMIFS_CONFIG)

Bit	Field	Description	R/W	Reset Value
31:5	Reserved	Reserved. To ensure software compatibility, reserved bit must be written to 0 and read value is considered undefined.	R/W	0x000 000
4	FR	READY signal: copy of the READY input pin sample by TC_CK (2 TC_CK cycles delay from input pin to register update). 0: READY pin is low. 1: READY pin is high.	R	0x0
3	PDE	System power-down acknowledge 0: Acknowledge disabled 1: Acknowledge enabled	R/W	0x0
2	PWD_EN	Dynamic auto idle 0: Disable 1: Enable	R/W	0x0

Table 2–52. EMIFS Configuration Register (EMIFS_CONFIG) (Continued)

Bit	Field	Description	R/W	Reset Value
1	BM	MPU boot mode. This bit enables CS0 and CS3 address decoding swapping. 0: CS0 [0000:0000 - 03FF:FFFF] CS3 [0C00:0000 - 0FFF:FFFF] 1: CS0 [0C00:0000 - 0FFF:FFFF] CS3 [0000:0000 - 03FF:FFFF] BM reset value is equal to the ARM_BOOT_MODE input pin state sampled at IC reset release time.	R/W	0x0
0	WP	Write protect output pin control 0: WP output pin is set low. 1: WP output pin is set high.	R/W	0x0

Table 2–53. EMIFS CS Configuration Registers (FLASH_CFG_0...FLASH_CFG_3)

Bit	Field	Description	R/W	Reset Value
31	PGWSTEN	0: PGWST is specified by 15:12. 1: PGWST is specified by 30:27.	R/W	0
30:27	PGWST	Controls the wait states cycle number between accesses in a page for asynchronous page mode. This field takes effect only if PGWSTEN bit field is set to 1.	R/W	0
26:23	BTWST	Controls the IDLE cycle number for bus turn-around and CS high-pulse-width timing	R/W	0
22	MAD	Enables the multiplexed address and data bus protocol 0: Nonmultiplexed protocol 1: Multiplexed protocol For CS0 and CS3 configuration register MAD reset value is equal to the MUX_DEVICE input pin state sampled at IC reset release time.	R/W	0
21	FL	0: Address in not incremented for the second write with 32-bit writes to 16-bit memories. 1: Address is incremented for the second 16-bit write with 32-bit writes to 16-bit memories.	R/W	0
20	BW	Must be written to 0 for FLASH_CFG_0 and FLASH_CFG_3 Must be written to 1 for FLASH_CFG_1 and FLASH_CFG_2	R/W	1
19	Reserved	Reserved for RDMODE mode expansion; read value is considered undefined.	R	0

Table 2–53. EMIFS CS Configuration Registers (FLASH_CFG_0...FLASH_CFG_3) (Continued)

Bit	Field	Description	R/W	Reset Value
18:16	RDMODE	Read mode select (see table below and notes 2, 3, 4, 5). For CS0 and CS3 configuration register, RDMODE reset value is controlled by the BOOT_TI_ACE_ROM_RAM input pin state sampled at IC reset release time. RDMODE = 0 if BOOT_TI_ACE_ROM_RAM = 0 RDMODE= 7 if BOOT_TI_ACE_ROM_RAM = 1	R/W	0
15:12	PGWST/WELEN	Controls the wait states cycle number between accesses in a page for asynchronous page mode. Controls the WE pulse length during a write access. This bit field is used as both PGWST/WELEN when PGWSTEN bit field is set to 0. This field specifies only WE-LEN when the PGWSTEN bit field is set to 1.	R/W	1
11:8	WRWST	Controls the wait states cycle number for write operation	R/W	1
7:4	RDWST	Controls the wait states cycle number for asynchronous read operation and the initial idle time for asynchronous read page mode and synchronous read mode	R/W	1
3	Reserved	Reserved. Writing to this bit has no effect. Reading it returns undefined value.	R/W	1
2	RT	Enable the read retimed protocol 0: Non-retimed protocol 1: Retimed protocol For CS0 and CS3 configuration register RT reset value is equal to 1 if the BOOT_TI_ACE_ROM_RAM input pin state is sampled to 1 at IC reset release time.	R/W	0
1:0	FCLKDIV	Controls the TC_CK divider. REF_CLK. 00: REF_CLK = TC_CK divide by 1 01: REF_CLK = TC_CK divide by 2 10: REF_CLK = TC_CK divide by 4 11: REF_CLK = TC_CK divide by 6 For CS0 and CS3 configuration register FCLKDIV reset value equals to 00 if the BOOT_TI_ACE_ROM_RAM input pin state is sampled to 1 at reset release time.	R/W	1

The EMIFS chip-select configuration registers set the protocol to be used with the device connected to the corresponding chip-select (nCS0-nCS3).

Reset values for EMIFS chip-select configuration registers CS0, CS1, CS2, and CS3 depend on boot input pins BOOT_ROM_SIZE, MUX_DEVICE, and BOOT_TI_ACE_ROM_RAM. See Section 3.3.20, *EMIFS Boot Mode*.

Table 2–54 lists the EMIFS CS RDMODE field definition.

Table 2–54. EMIFS Chip-Select Configuration Register RDMODE Field Definition

RDMODE	Memory
000	Mode 0: Asynchronous read
001	Mode 1: Page mode ROM read—4 words per page
010	Mode 2: Page mode ROM read—8 words per page
011	Mode 3: Page mode ROM read—16 words per page
100	Mode 4: Synchronous burst read mode
101	Mode 5: Synchronous burst read mode
110	Reserved for future extension
111	Mode 7: Synchronous burst read mode

The time-out registers in Table 2–55 through Table 2–57 control the number of TC clock cycles before DMA, DSP, or OCPI requests are made high priority in the dynamic priority scheme used inside the traffic controller.

Table 2–55. EMIFS Time-Out Register 1 (EMIFS_TIMEOUT1_REG)

Bit	Field	Description	Reset Value
31:8	Reserved	Reserved	
7:0	DMA	Store the number of clock cycles before DMA requests are made high priority in dynamic priority arbitration.	0x0

Table 2–56. EMIFS Time-Out Register 2 (EMIFS_TIMEOUT2_REG)

Bit	Field	Description	Reset Value
31:8	Reserved	Reserved	
7:0	DSP	Store the number of clock cycles before DSP requests are made high priority in dynamic priority arbitration.	0x0

Table 2–57. EMIFS Time-Out Register 3 (EMIFS_TIMEOUT3_REG)

Bit	Field	Description	Reset Value
31:8	Reserved	Reserved	
7:0	OCPI	Store the number of clock cycles before OCPI requests are made high priority in dynamic priority arbitration.	0x0

Table 2–58. EMIFS Dynamic Wait States Control Register (FL_CFG_DYN_WAIT

Bit	Field	Description	Reset Value
31:8	Reserved	Reserved	
7	HANDSHAKE_ENABLE	Handshake enables for CS3 0: Full-handshaking 1: Non-full-handshaking	0x0

Table 2–58. EMIFS Dynamic Wait States Control Register (FL_CFG_DYN_WAIT (Continued)

Bit	Field	Description	Reset Value
6	HANDSHAKE_ENABLE	Handshake enables for CS2 0: Full-handshaking 1: Non-full-handshaking	0x0
5	HANDSHAKE_ENABLE	Handshake enables for CS1 0: Full-handshaking 1: Non-full-handshaking	0x0
4	HANDSHAKE_ENABLE	Handshake enables for CS0 0: Full-handshaking 1: Non-full-handshaking	0x0
3	WAIT_STATE_ENABLES	Dynamic wait states enables for CS3 0: Dynamic wait states mode disabled 1: Dynamic wait states mode enabled	0x0
2	WAIT_STATE_ENABLES	Dynamic wait states enables for CS2 0: Dynamic wait states mode disabled 1: Dynamic wait states mode enabled	0x0
1	WAIT_STATE_ENABLES	Dynamic wait states enables for CS1 0: Dynamic wait states mode disabled 1: Dynamic wait states mode enabled	0x0
0	WAIT_STATE_ENABLES	Dynamic wait states enables for CS0 0: Dynamic wait states mode disabled 1: Dynamic wait states mode enabled	0x0

The EMIFS dynamic wait states control register controls whether EMIFS has dynamic wait states by using the ready signal.

Table 2–59.EMIFS Abort Address Register (EMIFS_ABORT_ADDR)

Bit	Name	Function	Reset Value
31:0	ABORT_ADDR	Address of the abort access	0x0

Table 2–60. EMIFS Abort Type Register (EMIFS_ABORT_TYPE)

Bit	Field	Description	Reset Value
31:5	Reserved	Reserved. To ensure software compatibility, reserved bit must be write to 0 and read value is considered undefined.	0x0
4	TIMEOUT_ERR	Time out status bit set to 1 if abort was caused by a time-out error.	0x0
3	SECURE_ERR	Restricted access status bit is set to 1 if abort was caused by a restricted access error.	0x0
2:1	HOST_ID	Specify the source of the aborted transaction: 00: MPU 01: Modem 10: DMA 11: OCP-I	0x0
0	ABORT	Abort status bit. Reading the abort type register reset the abort status bit. 0: No abort 1: Abort	0x0

Table 2–61. EMIFS Abort Timeout Register (EMIFS_ABORT_TOUT)

Bit	Name	Function	Reset Value
8	TIMEOUT_EN	Time-out enable 0: Disable 1: Enable	0x1
7:0	TIMEOUT_VALUE	Time-out value	0xFF

Table 2–62. EMIFS Timeout Registers (EMIFS_TIMEOUT1_REG...EMIFS_TIMEOUT3_REG))

Bit	Field	Description	Reset Value
31:9	Reserved	Reserved. To ensure software compatibility, reserved bit must be written to 0 and read value is considered undefined.	0x0
8	TIMEOUT_EN	Enable the time-out timer 0: Timer is disabled. 1: Timer is enabled.	0x0
7:0	TIMEOUT	Time out counter value in REF_CLK clock cycles	0x0

Table 2–63. Advanced EMIFS CS Configuration Registers (FLASH_ACFG_0_I...FLASH_ACFG_3_1)

Bit	Field	Description	Reset Value
31:10		Reserved. To ensure software compatibility, reserved bit must be written to 0 and read value is considered undefined.	0x0
9	BTMODE	Enables extended BTWST usage 0: Bus turn around control and RD to RD/WR same CS pulse width high control 1: Bus turn around control and RD/WR to RD/WR same CS pulse width high control	0x0
8	ADVHOLD	Controls the ADV pulse width low	0x0
7:4	OEHOLD	Controls the number of cycles from OE high to CS high	0x0
3:0	OESETUP	Controls the number of cycles inserted from CS low to OE low. When mux device is set to 1, the reset value of OESETUP is 0x2.	0x0

2.5.8 EMIFF Registers

Table 2–64 lists the 32-bit EMIFF registers. Table 2–65 through Table 2–80 describe the register bits.

Table 2–64. EMIFF Registers

Name	Description	R/W	Offset
EMIFF_PRIORITY_REG	EMIFF priority	R/W	0x08
EMIFF_SDRAM_CONFIG	EMIFF SDRAM configuration	R/W	0x20
EMIFF_MRS	EMIFF SDRAM MRS register (legacy)	R/W	0x24
EMIFF_TIMEOUT1	EMIFF time-out1	R/W	0x8C
EMIFF_TIMEOUT2	EMIFF time-out2	R/W	0x90
EMIFF_TIMEOUT3	EMIFF time-out3	R/W	0x94
EMIFF_CONFIG_REG2	EMIFF configuration 2	R/W	0x3C
EMIFF_MRS_NEW	EMIFF SDRAM MRS (new)	R/W	0x70
Reserved	Reserved	R/W	0x74
EMIFF_EMRS1	EMIFF SDRAM EMRS1	R/W	0x78
EMIFF_EMRS2	EMIFF SDRAM EMRS2	R/W	0xC8
SDRAM_OPERATION_REG	EMIFF SDRAM operation	R/W	0x80
SDRAM_MANUAL_CMD_REG	EMIFF SDRAM manual command	R/W	0x84
EMIFF_ABORT_ADDRESS	EMIFF abort address	R	0x98
EMIFF_ABORT_TYPE	EMIFF abort type	R	0x9C
Reserved	Reserved	R/W	0xC0
Reserved	Reserved	R	0xC4
Reserved	Reserved	R/W	0xCC

Table 2–64. EMIFF Registers (Continued)

Name	Description	R/W	Offset
Reserved	Reserved	R	CCBC
Reserved	Reserved	R/W	CC64
Reserved	Reserved	R	CC68

Table 2–65.EMIFF Priority Register (EMIFF_PRIORITY_REG)

Bit	Field	Description	R/W	Reset Value
31:16	Reserved	Reserved	R	0x0
15:12	L3 OCP	L3 OCP consecutive access	R/W	0x0
11:8	DMA	DMA consecutive access	R/W	0x0
7		Reserved	R	0x0
6:4	DSP	Modem consecutive access	R/W	0x0
3		Reserved	R	0x0
2:0	MPU	MPU consecutive access	R/W	0x0

The EMIFF priority register allows the EMIFF to give consecutive accesses to a master when the master has been granted the EMIFF interface. The MPU and modem can have from 0 to 7 consecutive accesses and the DMA and the Level3 OCP initiator can have 0 to 15 consecutive accesses, according to the content of the corresponding bits (see Table 2–67 and Table 2–68).

Table 2–66. EMIFF SDRAM Configuration Register (EMIFF_SDRAM_CONFIG)

Bit	Field	Description	R/W	Reset Value
31:30	Reserved	Must be 00	R	0x0
29:28	LG SDRAM Type	Used to define the larger SDRAM memories (>256MB). See Table 2–67.	R/W	0x0
27	CLK	Disable SDRAM clock	R/W	0x0
		0: Enable the clock to the external SDRAM bank		
		1: Disable the clock to the external SDRAM bank		
26	PWD	Power-down enable. Put the SDRAM device into power-down mode. The CKE signal to the device is held high only for an active transaction.	R/W	0x0
25:24	SDRAM frequency	SDRAM frequency range. To control the idle time of SDRAM regarding to the clock organization:	R/W	0x0
		00: SDF0 (reset value)		
		01: SDF1		
		10: SDF2		
		11: SDF3		

Table 2–66. EMIFF SDRAM Configuration Register (EMIFF_SDRAM_CONFIG) (Continued)

Bit	Field	Description	R/W	Reset Value
23:8	ARCV	Autorefresh counter register value This value is calculated using the formula: Value = (((Refresh Interval/Clock period) % Number of rows) × Burst refresh size) - 400	R/W	0x6188
7:4	SDRAM type	Set the SDRAM internal organization	R/W	0x0
3:2	ARE	Autorefresh enable. When autorefresh enable is set, the EMIF generates a REFR request, depending on the autorefresh counter and the burst refresh counter. If refresh enable is not set, the refresh must be done as a RAS_ONLY refresh under CPU control. 00: Autorefresh disable 01: Autorefresh enable (one command every 14.7 µs) 10: Autorefresh by burst of 4 commands 11: Autorefresh by burst of 8 commands	R/W	0x0
1	Reserved	Must be 0	R	0x01
0	SLRF	Self-refresh, when set, puts the SDRAM in self-refresh mode if an access is made to the SDRAM after the SDRAM automatically comes out of self-refresh.	R/W	0x0

Table 2–67.EMIFF SDRAM Register Memory/Data Bus Size

Register Value [29:28, 7:4]	Memory Size (M bits)	Data Bus Size	Number of Banks
00 0000	16	8	2
00 0001		8	4
00 0010		16	2
00 0011		16	4
00 0100	64	8	2
00 0101		8	4
00 0110		16	2
00 0111		16	4
00 1001	128	8	4
00 1011		16	4
00 1101	256	8	4
00 1111		16	4
01 0000	512	8	4
10 0000	1024	16	4

Table 2–68. Frequency Range (SDRAM)

ac Parameters	SDF0 (Cycles)	SDF1 (Cycles)	SDF2 (Cycles)	SDF3 (Cycles)
tRC	9	5	3	2
tRAS	5	3	2	2
tRP	3	2	2	2
tRCD	3	2	2	2
tRRD	2	2	2	2

Table 2–69. EMIFF SDRAM MRS Register—Legacy (EMIFF_MRS)

Bit	Field	Description	Reset Value
31:10	Reserved	Must be all 0	0x00000
9	WBST	Write burst must be 0 (burst write same as burst read).	0x0
8:7	Reserved	Must be 00	0x0
6:4	CASL	CAS idle time: 001: Reserved 010: CAS idle time = 2 011: CAS idle time = 3 (default) Must be set to 2 for DDR.	0x3
3	S/I	Serial = 0/Interleave = 1 (must be serial)	0x0
2:0	PGBL	Page burst length. Must be set to full page burst (111) for SDRAM and burst of 8 (011) for DDR SDRAM.	0x7

The time-out registers in Table 2–70 through Table 2–72 store the number of clock cycles before modem, DMA, or Level 3 OCP initiator requests are made high priority in the dynamic priority scheme for the EMIFF SDRAM interface.

Table 2–70. EMIFF Time-Out 1 Register (EMIFF_TIMEOUT1)

Bit	Field	Description	R/W	Reset Value
31:8	Reserved		R	0x000000
7:0	DMA	DMA time-out counter value	R/W	0x00

Table 2–71. EMIFF Time-Out 2 Register (EMIFF_TIMEOUT2)

Bit	Field	Description	R/W	Reset Value
31:24	Reserved		R	0x00
23:16	DSP	DSP time-out counter value	R/W	0x00
15:8	Reserved		R	0x00
7:0	LCD	LCD time-out counter value	R/W	0x00

Table 2–72. EMIFF Time-Out 3 Register (EMIFF_TIMEOUT3)

Bit	Field	Description	R/W	Reset Value
31:8	Reserved		R	0x000000
7:0	OCPI	Level3 OCP Initiator	R/W	0x00

Table 2–73. EMIFF Configuration Register 2 (EMIFF_CONFIG_2REG)

Bit	Field	Description	R/W	Reset Value
31:3	Reserved	Must be all 0s	R	0x00000000
2	SD_AUTO_CLK	Allows controller to suspend its internal clocks when idle. The clocks are automatically reenabled when there is an autorefresh or host request. 1: Enable 2: Disable (reset) This bit must be set in conjunction with the CLK bit in the SDRAM configuration register in order to turn off the clock to the external SDRAM device.	R/W	0x0
1	RFRSH_RESET	Place the SDRAM into SELF_REFRESH when in reset (active high).	R/W	0x1
0	RFRSH_STDBY	Place the SDRAM into SELF_REFRESH when in standby mode (active high).	R/W	0x1

Table 2–74. EMIFF SDRAM MRS—New Register (EMIFF_MRS_NEW)

Bit	Field	Description	R/W	Reset Value
31:10	Reserved	Must be all 0s	R	0x000000
9	WBST	Write burst must be 0 (burst write same as burst read)	R/W	0x0
8:7	Reserved	Must be 00	R	0x0
6:4	CASL	CAS idle time: 001: Reserved 010: CAS idle time = 2 011: CAS idle time = 3 (default) Must be set to 2 for DDR.	R/W	0x3
3	S/I	Serial = 0/Interleave=1 (must be serial)	R/W	0x0
2:0	PGBL	Page burst length. Must be set to full page burst (111) for SDRAM and burst of 8 (011) for DDR SDRAM.	R/W	0x7

There is only one physical MRS register. Using that address, no automatic initialization sequence is generated; only a LOAD MODE register command isssued.

A CPU write to EMIFF_MRS_NEW generates a LOAD MODE register command, with BA1,BA0 = 0,0.

Table 2–75. EMIFF Low Power SDRAM EMRS1 Register (EMIFF_EMRS1)

Bit	Field	Description	R/W	Reset Value
31:5	Reserved	Must be all 0s	R	0x0
4:3	TCSR	Temperature-compensated self-refresh	R/W	0x00
		00: 70°C maximum temperature		
		01: 45°C maximum temperature		
		10: 15°C maximum temperature		
		11: 85°C maximum temperature		
2:0	PASR	Partial-array self-refresh	R/W	0x0
		000: All banks		
		001: 1/2 array (BA1=0)		
		010: 1/4 array (BA1=BA0=0)		
		011: Reserved		
		100: Reserved		
		101: 1/8 array (BA1=BA0=0, RA11=0)		
		110: 1/16 array (BA1=BA0=0, RA11=RA10=0)		
		111: Reserved		

Note: Bit designation is given for a 128M-bit device.

EMIFF_EMRS1 is used for SDRAM memories dedicated for low-power wireless applications. The description is given here with reference to standard devices, but must be checked with the specification of the device actually used in a given application.

A CPU write to this register generates a LOAD MODE register command, with BA1=1 and BA0 = 0. Twelve bits can be loaded.

Table 2–76. EMIFF SDRAM EMRS2 Register (EMIFF_EMRS2)

Bit	Field	Description	Reset Value
31:0	Reserved	Must be all 0s	

EMIFF_EMRS2 is provided to anticipate its use in future designs by memory manufacturers and must not be used by current applications.

A CPU write to this register generates a LOAD MODE register command, with BA1, BA0 = 1,1. Twelve bits can be loaded.

Table 2–77. EMIFF SDRAM Operation Register (SDRAM_OPERATION_REG)

Bit	Field	Description	Reset Value
31:25	Time-out_B4	Time-out value for Bank4, in TC clock cycles	0x0
24:18	Time-out_B3	Time-out value for Bank3	0x0
17:11	Time-out_B2	Time-out value for Bank2	0x0
10:4	Time-out_B1	Time-out value for Bank1	0x0
3:2	Operation mode	00: Low-power/low-bandwidth mode (LPLB mode) 01: High-power/high-bandwidth mode (HPHB mode) 10: Programmable operating mode 0 (POM0 mode) 11: Reserved. Must not be used.	0x1
1:0	SDRAM type	The type of SDRAM: 00: Regular SDR SDRAM 01: Regular DDR SDRAM 10: Low-power SDR SDRAM 11: Mobile DDR SDRAM	0x0

Table 2–78. EMIFF SDRAM Manual Command Register (SDRAM_MANUAL_CMD_REG)

Bit	Field	Description	R/W	Reset Value
31:4	Reserved	Must be 0000:000	R	0x0
3:0	SDRAM	Manual command 0000: NOP command 0001: Precharge command 0010: Autorefresh command 0011: Enter deep sleep command 0100: Exit deep sleep command 0111: Set CKE signal high 1000: Set CKE signal low 1xxx: Reserved. Not for use.	R/W	0x0

Table 2–79. EMIFF Abort Address Register (EMIFF_ABORT_ADDRESS)

Bit	Field	Description	Reset Value
31:0	ABORT_ADDRESS	Address of transaction aborted	0x0

Table 2–80. EMIFF Abort Type Register (EMIFF_ABORT_TYPE)

Bit	Field	Description	Reset Value
31:3	Reserved	Must be all 0s	0x0
2:1	HostID	ID of host whose transaction aborted 00: MPU 01: Modem 10: DMA 11: OCP-I	0x0
0	ABORT FLAG	Set when an abort occurs, reset when this register is read.	0x0

2.5.9 OCPI Registers

Table 2–81 lists the 32-bit OCPI registers. Table 2–82 through Table 2–89 describe the register bits.

Table 2–81. OCP Registers

Name	Description	R/W	Offset
ADDRFAULT	OCP address fault	R	0x00
MCMDFAULT	OCP master command fault	R	0x04
SINTERRUPT0	OCP SINTERRUPT0	R/W	0x08
ABORTTYPE	OCP type of abort	R/W	0x10
SINTERRUPT1	OCP SINTERRUPT1	R/W	0x14
PROTECT	OCP protection	R/W	0x18
SECURE_MODE	OCP-I secure mode	R/W	0x0C

Table 2–82. OCP Address Fault Register (ADDRFAULT)

Bit	Name	Function	Reset Value
31:0	ADDRESS	Address accessed by master that causes abort or error	0x0

Table 2–83. OCP Master Command Register (MCMDFAULT)

Bit	Name	Function	Reset Value
31:3	Reserved	Reserved. Must be 0.	0x0
2:0	MCMD	Command that causes an abort or error	0x0

Table 2–84. Master Command Register Supported Commands

MCMD	Transaction Type
000	Idle
001	Write
010	Read
011	ReadEX
OTHER	Not supported/not interpreted

The master command supported commands are listed in this register. For non-supported MCMD encoding, OCPI translates it to IDLE.

The READEX command is for atomic transfer: Read-modify-write. This command is internally decoded to generate a lock signal for the OMAP3 traffic controller and is replaced by a simple read with a lock signal to the subtargets. The READEX command *must* be followed by a write that matches the address of the READEX as specified in the Sonic OCP Specification. A READEX command followed by a read results in unpredictable behavior.

Table 2–85. Generate Interrupts Register 0 (SINTERRUPT0)

Bit	Name	Function	R/W	Reset Value
31:2	Reserved	Reserved. Must be all 0s.		0x0
1	Level Interrupt	Level-sensitive interrupt	R/W	0x1
		SINTERRUPT_n signal has the same value as this bit	H/W Auto-cleared	
		0: Interrupt		
		1: No interrupt		
0	Edge Interrupt	Edge-triggered interrupt	R/W	0x1
		SINTERRUPT_n signal is asserted low for one L3_OCPI_CK cycle when this bit is set to 0.	H/W Auto-cleared	
		0: Interrupt		
		1: No interrupt		
		A read from this register always returns 1s, because it is autocleared.		

Table 2–86. Generate Interrupts Register 1 (SINTERRUPT1)

Bit	Name	Function	Reset Value
31:2	Reserved	Reserved. Must be all 0s.	0x0
1	Level Interrupt	Level-sensitive interrupt	0x1
		Sinterrupt_n signal has the same value as this bit	
		0: Interrupt	
		1: No interrupt	
0	Edge Interrupt	Edge-triggered interrupt	0x1
		Sinterrupt_n signal is asserted low for one L3_ocpi_ck cycle when this bit is set to 0.	
		0: Interrupt	
		1: No interrupt	
		A read from this register always returns 1s, since it is auto-cleared.	

Table 2–87. Protection Register (PROTECT)

Bit	Name	Function	Reset Value
31:8	Reserved	Reserved. Must be 0.	0x0
7	API	Access to MPUI is prohibited.	0x0
6	RHEA_PUB	Access to MPU public TIPB is prohibited.	0x0
5	RHEA_PRIV	Access to MPU private TIPB is prohibited.	0x0
4	OCPMult	Access to OCPT multibank is prohibited.	0x0
3	OCPT2	Access to OCPT2 is prohibited.	0x0
2	OCPT1	Access to OCPT1 is prohibited.	0x0
1	EMIFF	Access to EMIFF is prohibited.	0x0
0	EMIFS	Access to EMIFS is prohibited.	0x0

Table 2–88. Secure Mode Register (SECURE_MODE)

Bit	Name	Function	Reset Value
31:7	Reserved	Reserved. Must be 0.	0x0
6	API	In secure mode	0x1
		0: Access is allowed.	
		1: Access to API is prohibited from initiator.	
5	RHEA_PRIV	In secure mode	0x1
		0: Access is allowed.	
		1: Access to MPU TIPB public is prohibited from initiator. TIPB private bus is not accessible regardless of the setting of this bit.	
4	OCPMult	In secure mode	0x1
		0: Access is allowed.	
		1: Access to OCP multibank is prohibited from initiator.	
3	OCPT2	In secure mode	0x1
		0: Access is allowed.	
		1: Access to OCPT2 is prohibited from initiator.	
2	OCPT1	In secure mode	0x1
		0: Access is allowed.	
		1: Access to OCPT1 is prohibited from initiator.	
1	EMIFF	In secure mode	0x1
		0: Access is allowed.	
		1: Access to EMIFF is prohibited from initiator.	
0	EMIFS	In secure mode	0x1
		0: Access is allowed to CS1-CS3.	
		1: Access to EMIFS CS1-CS3 is prohibited from initiator; CS0 is not accessible regardless of the setting of this bit.	

When not in secure mode, these secure mode register values are ignored. When in secure mode, EMIFS cs0 and TIPB private access is not allowed, regardless of the register values. Every other target secure mode is determined by the register value.

Table 2–89. Type of Abort Register (ABORTTYPE)

Bit	Name	Function	R/W	Reset Value
31:4	Reserved	Reserved. Must be 0.		0x0
3	Burst Error	Burst access to the MPUI or TIPB was requested.		0x0
2	PROTECT	Address hit a protected area		0x0
1	TRGABORT	Abort coming from the accessed target		0x0
0	ADDDEC	Address decoding error (initiator sent unknown address)		0x0

The abort type register is cleared after being read. In other words, reading the register once returns whatever bits correspond to the abort type. On the next read, the register returns all 0s. Also a write to this register has no effect. This register is cleared on read.

2.5.10 Priority Algorithms

The traffic controller provides a choice of two priority algorithms for simultaneous requests. Arbitration is performed in each TC target port (OCP-T1, OCP-T2, EMIFF, and EMIFS).

Selection of the arbitration scheme is common to all TC ports. Depending on the OMAP device, it can be either hardwired to one of the two algorithms or programmable (outside of OMAP).

- Enhanced round-robin with LRU

  In accordance with the standard round-robin scheme, the requestor having the highest priority becomes the lowest-priority requestor after it has been granted access.

  The first enhancement is that when it gets the highest priority, a requestor can keep it for a programmable number of consecutive accesses.

  The number of consecutive accesses is individually programmable for the MPU, modem, the system DMA, and OCP-I. See OCPT1_PRIORITY, OCPT2_PRIORITY, EMIF_SLOW_PRIORITY, and EMIF_FAST_PRIORITY registers.

  In parallel with the round-robin scheme, the second enhancement is that the least recently granted requestor always has the highest priority. That means if a requestor has missed its slot because it had no request pending at that time, it keeps the highest priority for the subsequent arbitration cycles.

- Dynamic priority order

  Most of the time, the priority order is fixed: highest priority is MPU, then modem, OCP-I, and lowest priority is DMA. A programmable time-out is attached to any host except the MPU to ensure that it is not blocked indefinitely by higher priority hosts.

  When a low-priority requestor gains the arbitration (that is, modem, OCP-I, DMA), the associated time-out counter is loaded with the programmed value and starts decrementing.

If a counter reaches 0, the associated requestor gets the highest priority for its next request (that may be already pending). If several requestors are in this situation, the priority order is: DMA LCD, modem, OCP-I, DMA other than LCD.

2.5.11 Endianism Register

Table 2–90. Endianism Register (FFFE:CC34)

Register	Offset
CONTROL_REGISTER	0x00

Table 2–91. CONTROL_REGISTER

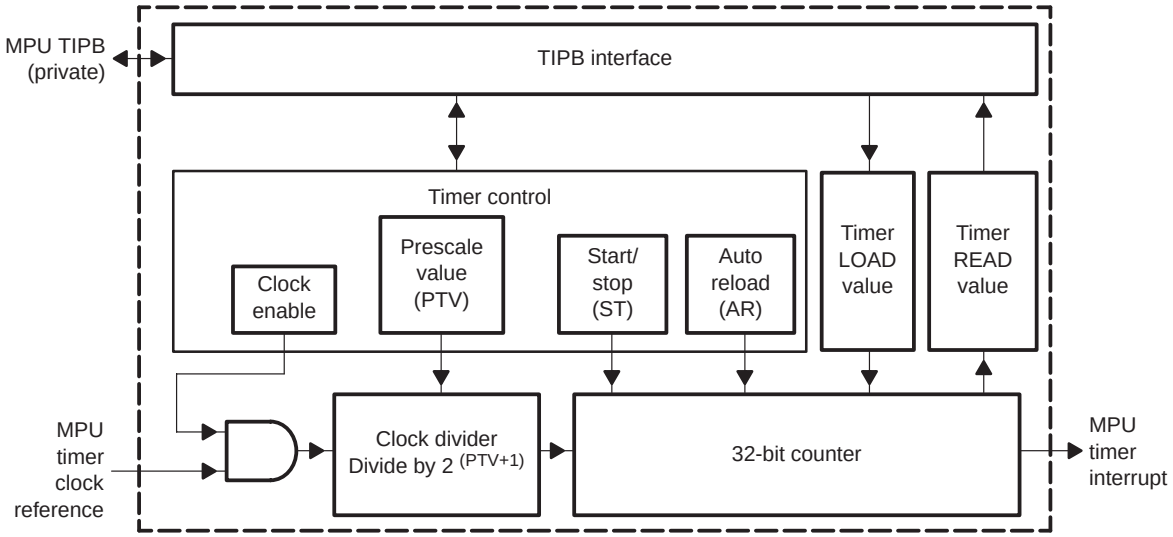
Bit	Name	Function	R/W	Reset Value
31:1	RESERVED	Reading this bit gives an undefined value and writing to it has no effect	R/W	0xU
1	SWAP	Controls the byte or word swap for DSP endianism block 1: Word swap 0: Byte swap	R/W	0x0
0	EN	Enable the DSP endianism module	R/W	0x0

2.6 Operating System Timers

There are three timers for the OS and general-purpose housekeeping functions inside the OMAP730 hardware engine.

The timer is configured either in autoreload or in one-shot mode with on-the-fly read capability. The timer generates an interrupt to the MPU when the value is equal to zero.

Figure 2–59. Timer Block Diagram



2.6.1 Functionality

2.6.1.1 MPU OS Timer Start and Stop

You can start an OS timer by setting the ST bit of the MPU_CNTL_TIMER register. At start, the timers are loaded with the values programmed to the MPU_LOAD_TIMER register. You can stop the timer by resetting the ST bit. When stopped, the timer value is frozen at the current value.

2.6.1.2 MPU OS Timer Autoreload

If the autoreload (AR) bit is set in MPU_CNTL_TIMER, a new value (from the load register) is loaded into the timer when it passes through zero.

If the AR bit is reset, the timer decrements from the loaded value to zero and then stops.

2.6.1.3 Reading OS Timer Values

You can read timer values from the MPU read timer registers either on the fly or after the timer is stopped.

- ❏ MPU_READ_TIMER is a 32-bit register and can be read directly.

2.6.1.4 MPU OS Timer Input Clocks

This section describes clocking for the MPU OS timer.

Configuration of the Input Clock for MPU 32-Bit OS Timers

The input clock reference for the MPU timer modules is controlled from the clock generation and reset module. Programming the ARM_TIMXO bit of the MPU prescaler selection register ARM_CKCTL in the clock module selects between two possible clock sources.

- ❏ Programming the ARM_TIMXO bit to 0 selects the main OMAP3.2 input reference clock as the clock reference for the MPU timers.
- ❏ Programming the ARM_TIMXO bit to 1 selects the output from the DPLL1 module (the clock source for the MPU clock domain) as the source for the MPU 32-bit OS timers.

The default is referenced from DPLL1. When operating from DPLL1, it is recommended to stop the MPU OS timer before programming a change to the DPLL1 frequency divisor. See Chapter 5, *Clock Generation and Reset Management,* for details of ARM_CKCTL and on programming the DPLL1 module.

Input Clock Enable

Input reference clocks for the MPU OS timer module are ANDed with a clock enable signal to gate the clock from timer internal logic. The clock is enabled to the timers when the CLOCK_ENABLE bit is set in the MPU_CNTL_TIMER.

2.6.2 Timer Interrupts

An interrupt occurs when the corresponding timer decrements to zero.

The MPU OS timer interrupt period is defined by:

1) The programmed value for the MPU OS timer input clock reference (see Section 2.6.1.4, *MPU OS Timer Input Clocks).*
2) The value of the prescaler bit field, PTV in MPU_CNTL_TIMER. Table 2–92 lists valid entries for PTV.
3) The value of load register MPU_LOAD_TIMER

Table 2–92. Valid Prescaler Values for OS Timer Configuration

PTV Bit Field in MPU OS Timer Control Register	Input Clock Divisor
000	2
001	4
010	8
011	16
100	32
101	64
110	128
111	256

The following equation is used to determine the timer interrupt periods for the MPU OS timer:

$$T_{MPU\_interrupt} = T_{MPU\_ref\_clk} \times (< MPU\_LOAD\_TIMER > + 1) \times 2^{(PTV+1)}$$

where $T_{MPU\ ref\ clk}$ is the clock period of the MPU OS timer input clock described in Section 2.6.1.4.

Based on this equation, examples for various MPU OS timer interrupt periods are calculated in Table 2–93 for a hypothetical MPU OS timer input clock reference frequency of 100 MHz ($T_{MPU\ ref\ clk}$ = 10 ns).

Table 2–93. Example MPU OS Timer Interrupt Periods

MPU_LOAD_TIMER	PTV = 000	PTV = 111
0x0 (granularity)	20 ns	2.56 µs
0xFFFFFFFF(max period)	85.9 s	10995 s (3 hr 3’ 15”)

2.6.3 Timer Programming

To activate the MPU OS timer, perform the following sequence:

1) Program the load register.

   The content of the load register is loaded into the timer read register when the ST bit is set.

2) Write to MPU_CNTL_TIMER to configure and start the timer.

   - Set the PTV value as the dividing factor for the timer clock.
   - Set the autoreload feature with the AR bit.
   - Set the ST bit to start the timer.

3) You can stop the timer at any time by resetting the ST bit (bit 0) of the MPU_CNTL_TIMER.

Note:

Only the ST or CLOCK_ENABLE bits of the MPU_CNTL_TIMER can be written while the timer is running. Undefined results occur if the prescaler (PTV) or autoreload (AR) bits in the control register or timer load registers are written while the timer is running.

2.6.4 Registers

This section describes the MPU operating system timer registers.

Note:

The register descriptions and base address offsets given here apply identically to each of the three MPU OS timers. See Section 1.4, *Memory Maps*, to determine the base address for each timer.

2.6.4.1 MPU 32-Bit OS Timers

Accesses to the MPU 32-bit OS timer configuration registers are controlled by the MPU private TIPB.

Table 2–94 lists the MPU OS timer registers. Table 2–95 through Table 2–97 provide register bit descriptions.

Table 2–94. MPU OS Timer Registers

Name	Description	R/W	Offset (Byte)
MPU_CNTL_TIMER	MPU control timer	R/W	0x00
MPU_LOAD_TIMER	MPU load timer	W	0x04
MPU_READ_TIMER	MPU read timer	R	0x08

Table 2–95. MPU Control Timer Register (MPU_CNTL_TIMER)

Bit	Name	Function	Reset Value
31:7	Reserved		
6	FREE	Specifies what action the timer takes upon receiving a suspend indication from the TIPB bridge. (For example, suspend is indicated if processor execution has been suspended at an emulation breakpoint.) 0: Stop counting on suspend indication (even when ST=1). 1: Suspend indication has no effect on the count.	0x0
5	CLOCK_ENABLE	Enable input reference clock to the MPU OS timer module 0: Disable 1: Enable	0x0

Table 2–95. MPU Control Timer Register (MPU_CNTL_TIMER) (Continued)

Bit	Name	Function	Reset Value
4:2	PTV	Prescale timer input reference clock 000: Divide input reference clock by 2 001: Divide input reference clock by 4 010: Divide input reference clock by 8 011: Divide input reference clock by 16 100: Divide input reference clock by 32 101: Divide input reference clock by 64 110: Divide input reference clock by 128 111: Divide input reference clock by 256	000
1	AR	0: One-shot mode 1: Autoreload mode	0x0
0	ST	0: Stop timer value decrement 1: Start timer value decrement If one-shot mode is selected (AR=0), this bit is automatically reset by internal logic when the timer value is equal to 0.	0x0

Table 2–96. MPU Load Timer Register (MPU_LOAD_TIMER)

Bit	Name	Function	Reset Value
31:0	MPU_LOAD_TIMER	This value is loaded when the timer passes through 0 or when it starts.	Undefined

Table 2–97. MPU Read Timer Register (MPU_READ_TIMER)

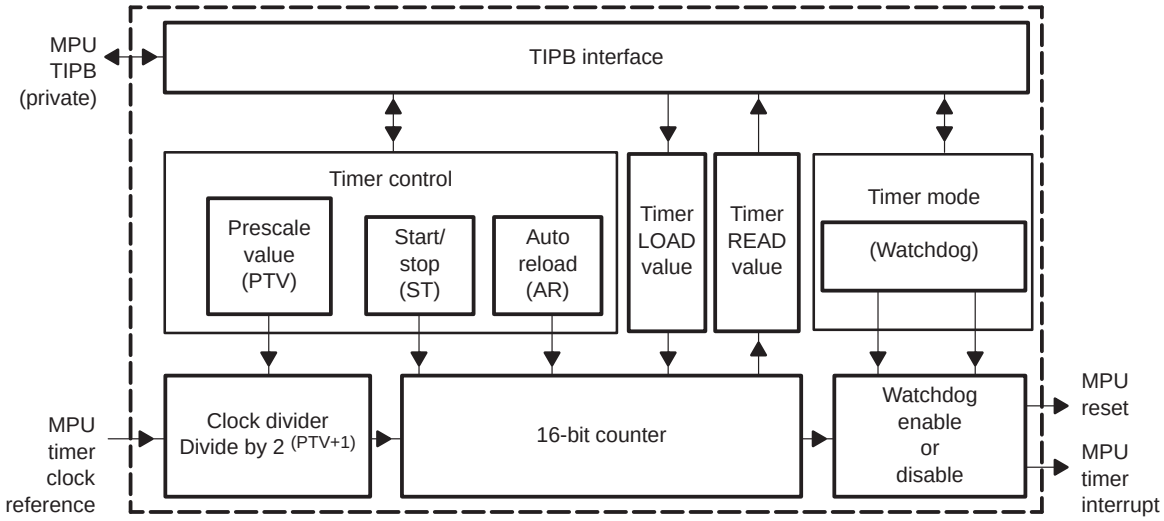
Bit	Name	Function	Reset Value
31:0	MPU_READ_TIMER	Value of the timer	Undefined

2.7 Watchdog Timers

In the OMAP 3.2 hardware engine, one timer is dedicated to watchdog supervision over the MPU. The MPU watchdog timer is based on 16-bit downcounters, and is programmable to function as a watchdog or a general-purpose timer. In general-purpose mode, the timer has features similar to those described in Section 2.6, *Operating System Timers.*

The MPU watchdog timer is accessed through the 32-bit wide MPU private TIPB. Figure 2–60 shows the DSP and MPU watchdog timers.

Figure 2–60. MPU Watchdog Timer



2.7.1 Functionality

This section describes the timer input clock source and the watchdog and the general-purpose operating modes of the MPU watchdog timer module.

2.7.1.1 Input Clocks for the MPU Watchdog Timer Module

The clock generation and reset module provides the input clock reference for the MPU watchdog timer. The timer input reference is derived from the main OMAP 3.2 input reference clock divided by a factor of 14. A prescaler in the MPU watchdog timer module (described below) further divides the timer input reference for use by the timer counter. For more detail on watchdog timer input reference clocks, see Chapter 5, *Clock Generation and Reset Management.*

2.7.1.2 Timers in Watchdog Mode

The MPU watchdog timer is power-up-enabled with watchdog mode as default. A program must write periodically to the MPU_LOAD_TIMER before the counter underflows. A write is considered valid only if the new loaded value is different from the previous one.

Note: Read/Write Transactions

Read/write transactions must be performed as 32-bit-wide accesses to the MPU watchdog timer registers. Incorrect access width results in a size mismatch error on the TIPB bridge. Such an error generates an interrupt to the MPU and causes TIPB bridge context to be saved in the TIPB bridge debug registers. For more detail on TIPB bridge debug registers, see Section 2.4, *TIPB Bridge.*

Underflow of the MPU watchdog timer generates a reset to the MPU core. The ARM_WDRST bit is set in ARM_SYSST of the clock and reset generation module to indicate that a watchdog timer underflow generated a reset to the MPU. For more details on ARM_SYSST, see Chapter 5, *Clock Generation and Reset Management.*

The watchdog underflow period is calculated as follows:

$$t_{underflow} = t_{ref\_clk} \times (< \text{MPU\_LOAD\_TIMER} > + 1) \times 2^{(PTV+1)}$$

where T_{ref_clk} is the period of the input clock reference to the watchdog timer module and PTV is fixed to 7 (PTV bits are 111) in hardware when operating in watchdog mode.

Values for MPU_LOAD_TIMER can range from 0 to 0xFFFF. As described in Section 2.7.1.1, if the main OMAP 3.2 input reference is 13 MHz, then the input reference to the watchdog timer modules is (13 MHz/14) or about 928 kHz, which corresponds to t_{ref_clk} = 1.077 µs. In this case the time to counter underflow (and reset) is 275 µs < T < 18 s.

The MPU watchdog timer is enabled at power up with the maximum value of 0xFFFF loaded into MPU_LOAD_TIMER. This gives the user a time of ($256 \times (0xFFFF+1) \times t_{ref\_clk}$) to switch to the general-purpose timer mode or to write a new value (different from 0xFFFF) into MPU_LOAD_TIMER.

2.7.1.3 Timers in General-Purpose Mode

The watchdog function can be disabled (if desired) and the timer can be used as a general-purpose timer.

Note: Idle Modes

The watchdog timer must be disabled before the MPU idle mode can be activated. For more detail on idle modes, see Chapter 5, *Clock Generation and Reset Management.*

To disable the watchdog function, write a predefined sequence (0xF5 followed by 0xA0) in the WATCHDOG_DIS bit field of MPU_TIMER_MODE. Receiving 0xF5 initializes a sequence decoder. Once in this state, if the next write is different from the 0xA0, the watchdog timer resets the MPU system. Note that if the timer is already configured to general-purpose mode, an inadvertent write of 0xF5 to WATCHDOG_DIS must be avoided, because it also causes reset.

To use the watchdog timer in general-purpose timer mode, perform the following sequence:

1) Disable the watchdog function.

   Write the predefined data, 0xF5 then 0xA0, to the WATCHDOG_DIS bit field of MPU_TIMER_MODE. These are two consecutive writes to the mode register.

2) Write the timer start value to MPU_LOAD_TIMER.

   The load registers are 16 bits; therefore, the value range is between 0x0000 and 0xFFFF. This value defines the number of timer-divided clock periods used by the downcounter before the counter underflows.

3) Program the timer control parameters by writing to MPU_CNTL_TIMER.

   Program the FREE bit field to 0 to enable the counter content freeze feature. This feature freezes the count if the timer receives a suspend indication from the TIPB bridge. For example, a suspend indication can be received if execution is suspended for debug purposes.

Note: Register Programming Constraints

In general-purpose mode, only the ST and FREE bits of MPU_CNTL_TIMER can be written while the timer is running. Undefined results occur if the PTV or AR bits in the control register or timer load registers are written while the timer is running. These restrictions do not apply in watchdog timer mode.

   Program the AR bit field to select between one-shot or autoreload when the downcounter underflows. In autoreload mode, the value programmed for MPU_LOAD_TIMER is loaded when the timer passes through zero.

   Program the PTV bit field to select the prescaler value. This fixes the ratio between the timer input reference clock period and the downcounter clock period. Table 2–98 lists valid entries for PTV.

   Set the ST bit to 1 to start the timer.

Table 2–98. Valid Prescaler Values in General-Purpose Timer Mode

PTV Bit Field	Input Clock Divisor
000	2
001	4
010	8
011	16
100	32
101	64
110	128
111	256

In general-purpose timer mode, the MPU watchdog timer issues an interrupt when the corresponding downcounter decrements to zero. Interrupt period is defined by the following equation:

$$t_{Interrupt} = t_{ref\_clk} \times (< MPU\_LOAD\_TIMER > + 1) \times 2^{(PTV+1)}$$

where t_{ref_clk} is the period of the input clock reference to the watchdog timer modules.

It is possible to return to the default mode (watchdog timer) by setting the WATCHDOG bit field of MPU_TIMER_MODE to active. In this case, the value loaded into MPU_LOAD_TIMER is set to the maximum value (0xFFFF) as on power up.

When switching from general-purpose mode to watchdog timer mode, you must wait for three timer clock periods before writing a new value into the load registers. This restriction does not exist when switching from watchdog mode to general-purpose timer mode.

2.7.2 Registers

This section describes the MPU watchdog timer registers. To determine base addresses for the respective timers, see Section 1.4, *Memory Maps.*

2.7.2.1 MPU Watchdog Timer

Read and write transactions with the MPU watchdog timer registers must be performed as 32-bit-wide accesses. Incorrect access width results in a size mismatch error on the TIPB bridge. Such an error generates an interrupt to the MPU and causes TIPB bridge context to be saved in the TIPB bridge debug registers. See Section 2.4, *TIPB Bridge,* for TIPB bridge register description.

Table 2–99 lists the MPU watchdog timer registers. The MPU_LOAD_TIMER and the MPU_READ_TIMER share the same offset value, such as when a value is written to the MPU_LOAD_TIMER and it is copied to the MPU_READ_TIMER at the same offset of 0x04, allowing the loaded value to be read from this register on the fly. Table 2–100 through Table 2–103 descrive the register bits.

Table 2–99. MPU Watchdog Timer Registers

Name	Description	R/W	Offset (Byte)
MPU_CNTL_TIMER	MPU control timer	R/W	0x00
MPU_LOAD_TIMER	MPU load timer	W	0x04
MPU_READ_TIMER	MPU read timer	R	0x04
MPU_TIMER_MODE	MPU read timer	R/W	0x08

Table 2–100. MPU Control Timer Register (MPU_CNTL_TIMER)

Bit	Name	Function	R/W	Reset Value
31:12	Reserved			
11:9	PTV	Prescale timer input reference clock. Programmable only in general-purpose timer mode. Fixed to divide by 256 in watchdog mode. Reset value for PTV reflects this fixed value since watchdog mode is the default at power up. 000: Divide input reference clock by 2 001: Divide input reference clock by 4 010: Divide input reference clock by 8 011: Divide input reference clock by 16 100: Divide input reference clock by 32 101: Divide input reference clock by 64 110: Divide input reference clock by 128 111: Divide input reference clock by 256	R/W	111
8	AR	Applicable only in general-purpose timer mode 0: One-shot timer 1: Autoreload timer	R/W	1
7	ST	0: Stop timer 1: Start timer When the one-shot mode is selected (AR=0), this bit is automatically reset by internal logic when the timer is equal to 0. The ST bit is only applicable in general-purpose timer mode. For watchdog mode, ST has no effect.	R/W	0
6:2	Reserved			
1	FREE	This bit specifies what action the timer takes upon receiving a suspend indication from the TI peripheral bus bridge. For example, suspend can be indicated if processor execution has been suspended at an emulation breakpoint. 0: Stop counting on suspend indication (even when ST=1). 1: A suspend indication has no effect on the count.	R/W	1
0	Reserved			

Table 2–101. MPU Load Timer Register (MPU_LOAD_TIMER)

Bit	Name	Function	Reset Value
31:16	Reserved		
15:0	LOAD_TIMER	General-purpose timer mode: This value is loaded when the timer passes through 0 or when it starts. Watchdog timer mode: Reload the timer with this value.	0xFFFF

Table 2–102. MPU Read Timer Register (MPU_READ_TIMER)

Bit	Name	Function	Reset Value
31:16	Reserved		
15:0	VALUE_TIMER	Value of timer	0xFFFF

Table 2–103. MPU Timer Mode (MPU_TIMER_MODE)

Bit	Name	Function	R/W	Reset Value
31:16	Reserved			
15	WATCHDOG	Write access 0: No effect 1: Switch the timer mode back to watchdog Read access: Status of timer mode: 0: Timer is used as a general-purpose counter. 1: Timer is used as a watchdog timer.	R/W	1
14:8	Reserved			
7:0	WATCHDOG_DIS	Write access only Writing a predefined sequence (0xF5 followed by 0xA0) in this field disables watchdog functionality. After having received 0xF5, if the second write access is different from 0xA0, MPU core is reset.	W	Undefined

2.8 CompactFlash Controller

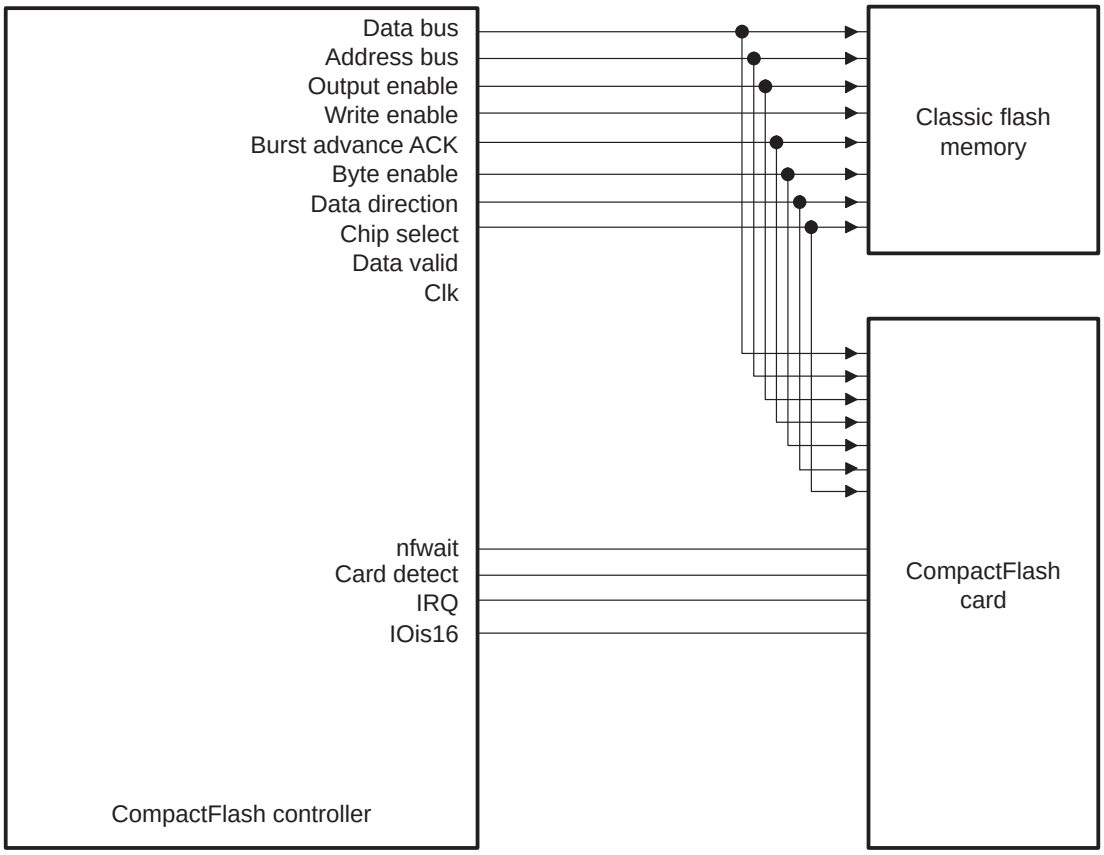
The CompactFlash controller (CFC) interfaces a CompactFlash and a classical memory interface. Control signals from memory interface are processed through the CFC to drive a CompactFlash card, and control signals from CompactFlash are processed to perform a data transfer to the memory interface.

2.8.1 Connection

The CFC interfaces the slow-memory interface block and the memories. When CompactFlash access is required, the CFC generates CompactFlash signals and processes the CompactFlash protocol.

Figure 2–61 shows the CompactFlash controller.

Figure 2–61. ConpactFlash Controller



2.8.2 Signal Connections

Table 2–104 lists the CFC signal connections.

Table 2–104. CFC Signal Connections

Name	I/O	CompactFlash Name	CompactFlash Description
FADD(10-1)	OUT	A10-A1	Address bus
FADD(14)	OUT	A0	Address bus
FADD(13)	OUT	REG	Attribute memory select
FADD(12)	OUT	IOWR	I/O data write
FADD(11)	OUT	IORD	I/O data read
NFWAIT	IN	WAIT	Wait
NFWE	OUT	WE	Strobe
NFBE(1)	OUT	CE2	Card enable
NFBE(0)	OUT	CE1	Card enable
Fdata	I/O	D15-D0	Data bus
NFOE	OUT	OE	Output enable
CF_nIOIS16	IN	WP IOIS16	Write protect 8/16bits selection
CF_RESET	OUT	RESET	Reset
CF_nCD1	IN	CD1	Card detect
CF_nCD2	IN	CD2	Card detect
CF_INTREQ	IN	RDY/BSY IREQ	Ready for new data

The I/O values are signals from OMAP710.

2.8.3 Memory Access Mode Selection

The CompactFlash card supports the following access modes:

- ❑ Common memory
- ❑ Attribute memory
- ❑ I/O
- ❑ True IDE

The controller manages the access to the different modes of the Compact-Flash (access protocol in these modes follows the CompactFlash standard).

Table 2–105 lists the memory mapping used to select the access mode.

Table 2–105. CFC Memory Mapping

Space Name	Start Address and CS Address	Stop Address	Size
CF memory space	0000:0000	0000:07FF	2K bytes
CF attribute space	0000:0800	0000:0FFF	2K bytes
CF I/O space	0000:1000	0000:17FF	2K bytes
CF true IDE	0000:1800	0000:1FFF	2K bytes

The CFC does not support the true IDE mode yet. Whatever memory space is reserved for true IDE accesses, this mode must not be used.

2.8.4 Interface Registers

The CFC contains a control register to reset the CompactFlash card, a status register, and a configuration register.

These registers are accessed using the following definitions:

- MPU TIPB address domain: strobe line
- MPU address: start address (Hex) = refer to TI925T specification
- Bit width: 16 bits
- Address of one register: start address plus offset address

Table 2–106 lists the 32-bit CompactFlash controller registers. Table 2–107 through describe the registsr bits.

Table 2–106. CompactFlash Controller Registers

Name	Description	Size	Offset
CF_STATUS_REG	CFC status Written by the CFC Read by TIPB bus	32 bits	0x00
CF_CFG_REG_I	CFC configuration 1 Written by the CFC Read by TIPB bus	32 bits	0x02
CF_CONTROL_REG	CFC control Written by the CFC Read by TIPB bus	32 bits	0x04

If a CompactFlash access is attempted while the CompactFlash card is not correctly connected, the last access direction is written in status register to inform the system that last transfer has not been completed.

Table 2–107. CFC Status Register (CF_STATUS_REG)

Bit	Name	Function	Reset Value
15:3	Reserved	Reserved	0x1FFF
2:1	LAST_ACCESS	Bad read access (active low) When a CompactFlash read access is performed when the CompactFlash card is not correctly connected (Card Detect is high), the CFC asserts this bit low. Bad write access (active low) When a CompactFlash write access is performed when the CompactFlash card is not correctly connected (Card Detect is high), the CFC asserts this bit low.	0x1
0	CARD_DETECT	CompactFlash correctly connected When CompactFlash is correctly connected, the card detect is low and this bit is asserted low.	Undefined

When a CompactFlash is connected or disconnected, a status register is written and an interrupt request is generated.

After this interruption, the status register must be accessed through the TIPB bus following an interrupt subroutine. The register contains data about the CompactFlash connection (connected or not) and current data transfer (read access, write access, no access).

Table 2–108. CFC Configuration Register 1 (CF_CFG_REG_1)

Bit	Name	Function	Reset Value
15:4	Reserved	Reserved	0xFFF
3:0	CS_CONFIGURATION	Chip-select connected on chip-select 3 (active low). Any transfer to CS_3 is a CompactFlash access. Chip-select connected on chip-select 2 (active low). Any transfer to CS_2 is a CompactFlash access. Chip-select connected on chip-select 1 (active low). Any transfer to CS_1 is a CompactFlash access. Chip-select connected on chip-select 0 (active low) Any transfer to CS_0 is a CompactFlash access.	0x1

The CFC configuration register indicates whether or not and on which chip-select CompactFlash is connected.

This register is run through a CFC TIPB interface and must be configured using the TIPB bus.

The CompactFlash card is reset from the CFC by a CF_RESET read in the CFC control register, which can only be written through the TIPB bus.

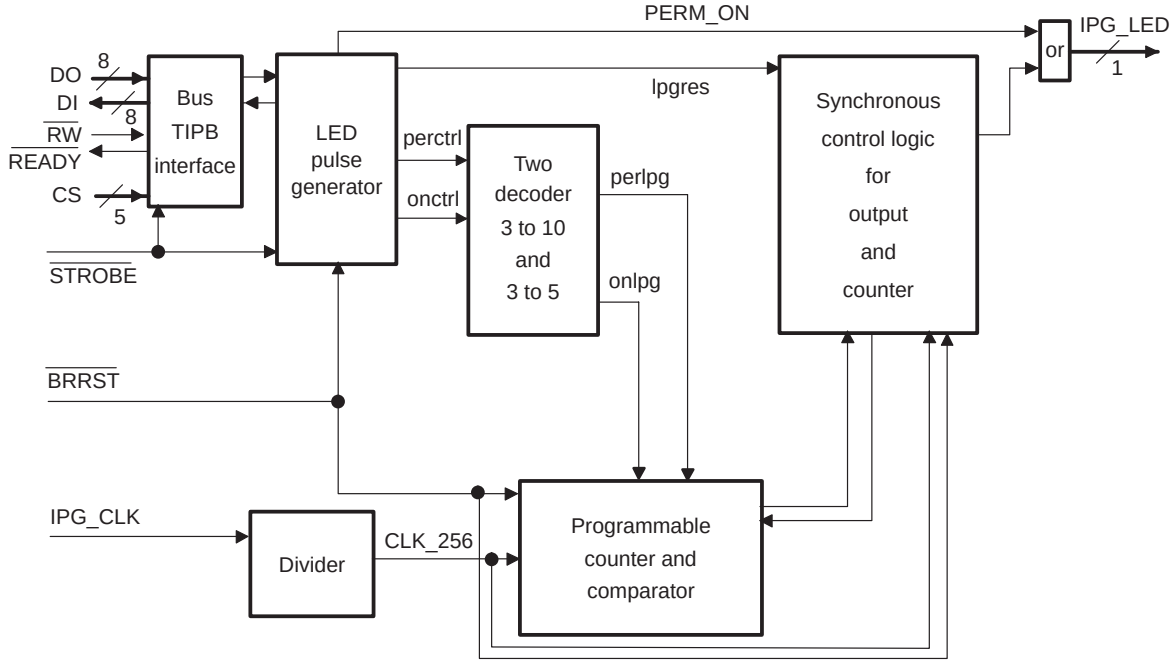
Table 2–109. CFC Control Register (CF_CONTROL_REG)

Bit	Name	Function	Reset Value
15:1		Reserved	0x0000
0	CF_CARD_RESET	CompactFlash card reset (active low)	0x0

2.9 LED Pulse Generator

The LED pulse generator (LPG) module controls an indication LED (see Figure 2–62). The blinking period is programmable between 152 ms and 4s, and the LED can be switched on permanently.

Figure 2–62. LED Pulse Generator Block Diagram



2.9.1 Features

The LPG has the following features:

- ❏ Divider generating a 256-Hz frequency clock
- ❏ TIPB control interface
- ❏ Two 8-bit registers to control the whole LPG block
- ❏ Decoder for three blink-frequency control bits (LPG2-0)
- ❏ Decoder for three pulse-width control bits (LPG5-3)
- ❏ Programmable counter with integrated comparison for the PWM
- ❏ Synchronous control logic for the output and the counter
- ❏ Multiplexer to generate a faster clock for testing

Table 2–110 lists the LPG functional I/O signals.

Table 2–110. LPG Functional I/O Signals

Name	Description	Type	Size	Active Level	Reset Level
nRESET	Asynchronous general reset	IN	1	0	--
LPG_CLK	LPG functional clock, 32-kHz frequency	IN	1	--	--
LPG_LED	Control LED signal	OUT	1	1	0

2.9.2 LPG Design

LCR bit 6 = 0 resets the whole PWM circuit (but not the control register) and switches off the LED. It is possible to switch on the LED independently from the PWM circuit with bit 7 of the LCR (1 = permanent light). The reset PWRON is active-low and resets the whole LPG (with the control register) and the output LPG_LED to zero asynchronously.

2.9.3 LPG Power Management

The LPG input clock comes from the 32-kHz ULPD clock, because it must work even when the OMAP730 system is in deep sleep mode. The internal clock of the LPG runs with 256 Hz. For this reason the power consumption of this block can be neglected. Nevertheless, switch the LPG_CLK off if LPG is not used.

2.9.4 LPG Registers

Both receive and transmit registers are mapped in the MPU address space.

Two instances of LPG are mapped in the OMAP730 device:

- First LPG: LPG_1 address is FFFB:A800.
- Second LPG: LPG_2 address is FFFB:A801.

Table 2–111 lists the LPG receive and transmit registers. Table 2–112 and Table 2–115 describe the register bits.

Table 2–111. LED Pulse Generator Receive and Transmit Registers

Register	Description	Size	Offset (hex)
LCR	LPG control	8 bits	0x00
PMR	Power management	8 bits	0x04

Table 2–112. LPG Control Register (LCR)

Bit	Name	Function	Reset Value
7	PERM_ON	Set high to force permanent light on. Asynchronous writing and reading.	0
6	LPGRES	LPG counter reset, active low. Asynchronous writing and reading.	0
5:3	ONCTRL	Time LED is on parameter. Asynchronous writing and reading.	000
2:0	PERCTRL	LED blink frequency. Asynchronous writing and reading.	000

Table 2–113. LED Blinking Period

LCR bit 2	LCR bit 1	LCR bit 0	Period of LED	Number of Clock Cycles
0	0	0	125 ms	32
0	0	1	250 ms	64
0	1	0	500 ms	128
0	1	1	1 s	256
1	0	0	1.5 s	384
1	0	1	2 s	512
1	1	0	2.5 s	640
1	1	1	3 s	768

With the LCR bits 2-0, the blinking period of the LED is determined.

Table 2–114. LED On-Time

LCR bit 5	LCR bit 4	LCR bit 3	Time LED On	Number of Clock Cycles
0	0	0	3.889 ms	1
0	0	1	7.789 ms	2
0	1	0	15.59 ms	4
0	1	1	31.39 ms	8
1	0	0	46.59 ms	12
1	0	1	62.59 ms	16
1	1	0	78.39 ms	20
1	1	1	93.59 ms	24

With the LCR bits 5-3 the on-time of the LED is determined.

Table 2–115. Power Management Register (PMR)

Bit	Name	Function	Reset Value
0	CLK_EN	Functional clock enable: 0: Clock disabled 1: Clock enabled Asynchronous writing and reading	0

2.10 MPU Serial Port Interface

The serial port interface (SPI) is a bidirectional 3-line interface dedicated to the transfer of data to and from external devices that offer a 3-line serial interface (see Figure 2–63).

The SPI is compatible with the Philips UMA1018M, the Fujitsu MB15F02, and the Siemens PMB2306T synthesizers, which are devices dedicated to mobile phone applications.

This SPI is based on a looped shift-register that allows both transmit (PISO) and receive (SIPO) modes.

The SPI is fully controlled by the TIPB bus (data write, data read, and activation of serialization operations).

The only difference between SPI_100 kHz and classic SPI is that the SPI_100 kHz has more values for prescale clock divisor (PTV) (1, 2, 4, 8, 16, 32, 64, and 128).

Figure 2–63. SPI Block Diagram

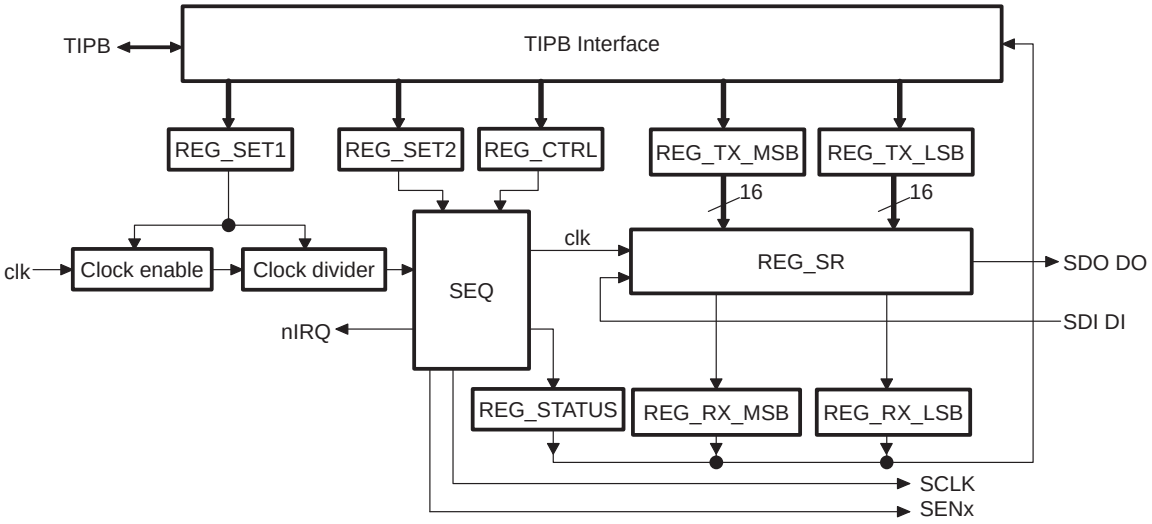


Figure 2–64. SPI System Block Diagram

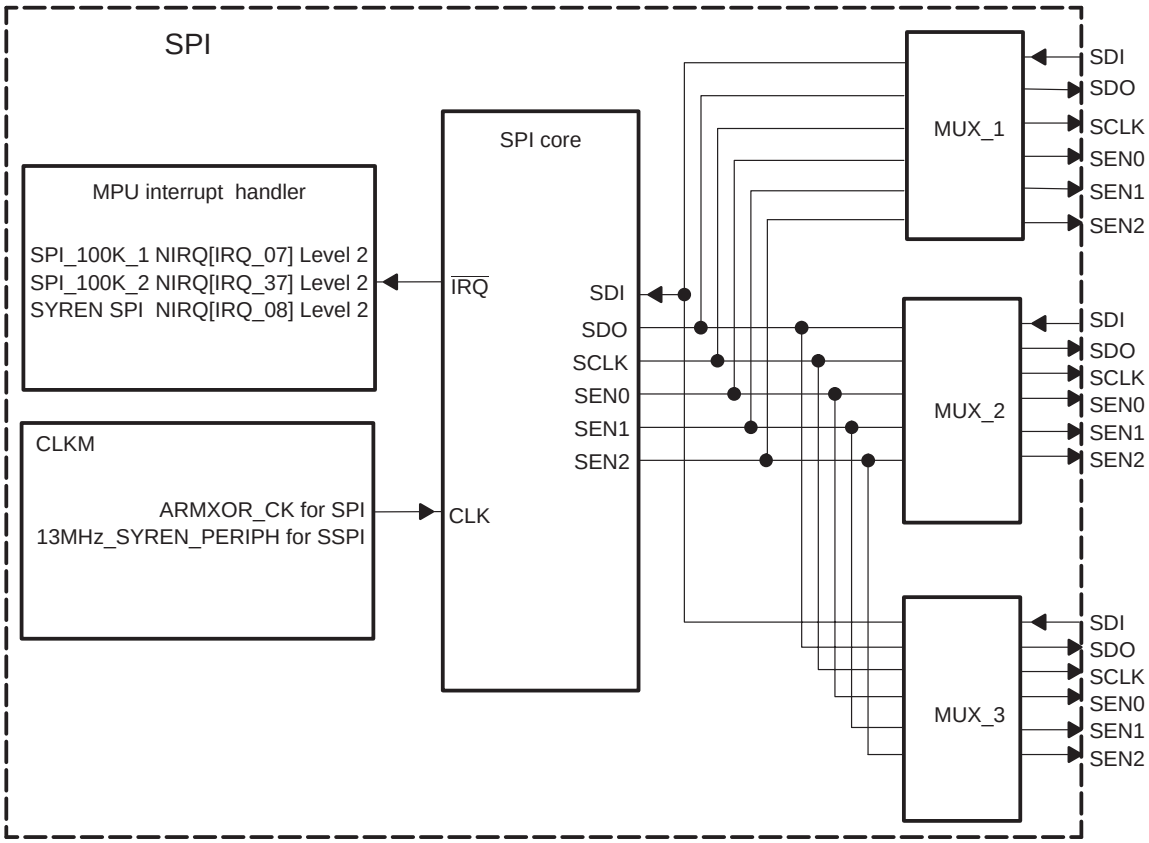


Table 2–116. Pin Multiplexing Configuration

	MUX1	MUX2	MUX3
SPI_1	D_SPI1_SCLK=PERSEUS2_IO_CONF_8[19:17]=000 D_SPI1_SDO=PERSEUS2_IO_CONF_8[23: 21]=000 D_SPI1_SDI=PERSEUS2_IO_CONF_8[27:25]=000 D_SPI1_SEN0=PERSEUS2_IO_CONF_8[31:29]=000 D_SPI1_SEN1=PERSEUS2_IO_CONF_9[3:1]=000 D_SPI1_SEN2=PERSEUS2_IO_CONF_9[7:5]=000	D_SDMC=PERSEUS2_IO_CONF_2[11:9]=001 D_SDMC_DAT2=PERSEUS2_IO_CONF_2[15:13]=001 D_SDMC_DAT3=PERSEUS2_IO_CONF_2[19:17]=001	D_UART_TX_RX=PERSEUS2_IO_CONF_3[23: 21]=100 D_UART_RTS_CTS=PERSEUS2_IO_CONF_3[27: 25]=100

Table 2–116. Pin Multiplexing Configuration (Continued)

	MUX1	MUX2	MUX3
SPI_2	D_LCD_PLX_15_12=PERSEUS3_IO_CONF_3[31:29]=010	D_EMIF_FADD16=PERSEUS2_IO_CONF_6[23:21]=100	
	D_LCD_PLX_11=PERSEUS2_IO_CONF_4[7:5]=010	D_EMIF_FADD17=PERSEUS2_IO_CONF_6[19:17]=100	
	D_LCD_PLX_10=PERSEUS2_IO_CONF_4[3:1]=010	D_EMIF_FADD18=PERSEUS2_IO_CONF_6[15:13]=100	
		D_EMIF_FADD19=PERSEUS2_IO_CONF_6[11:9]=100	
		D_EMIF_FADD20=PERSEUS2_IO_CONF_6[7:5]=100	
		D_KB9=PERSEUS2_IO_CONF_13[27:25]=100	
		D_KB8=PERSEUS2_IO_CONF_13[23:21]=100	
		D_KB7=PERSEUS2_IO_CONF_13[19:17]=100	
		D_KB6=PERSEUS2_IO_CONF_13[15:13]=100	
		D_KB5=PERSEUS2_IO_CONF_13[11:9]=100	
		D_KB4=PERSEUS2_IO_CONF_13[7:5]=100	
Syren_SPI	D_SYREN_SPI=PERSEUS2_IO_CONF_2[3:1]=000		

Table 2–117. Output Pin Multiplexing

		MUX1	MUX2	MUX3
SPI1	SCLK	MPU_SPI1_SCLK	SDMC_CLK	MPU_UART_TX1
	SDO	MPU_SPI1_SDO	SDMC_CMD	MPU_UART_RX1
	SDI	MPU_SPI1_SDI	SDMC_DAT_0	MPU_UART_CTS1
	SEN0	MPU_SPI1_SEN0	SDMC_DAT_1	MPU_UART_RTS1
	SEN1	MPU_SPI1_SEN1	SDMC_DAT_2	
	SEN2	MPU_SPI1_SEN2	SDMC_DAT_3	
SPI2	SCLK	LCD_PIXEL_15	FADD_20/ KBC_4	
	SDO	LCD_PIXEL_14	FADD_19/ KBC_2	
	SDI	LCD_PIXEL_13	FADD_18/ KBC_3	
	SEN0	LCD_PIXEL_12	FADD_17/ KBC_4	
	SEN1	LCD_PIXEL_11	FADD_16/ KBC_0	

Table 2–117. Output Pin Multiplexing (Continued)

		MUX1	MUX2	MUX3
	SEN2	LCD_PIXEL_10	KBC_1	
Syren_SPI	SCLK			
	SDI	MCUDI		
	SDO	MCUDO		
	SEN0	MCUEN		
	SEN1			
	SEN2			

2.10.1 SPI Registers

The SPI has input and output registers that load data to serialize (TRANSMIT) or to read parallelized (RECEIVE) data. The characteristics of the registers are as follows:

- ❏ Start address in the peripheral range (hex): 0000
- ❏ Bit width: 16 bits
- ❏ Address of one register: start address plus offset address

Table 2–118 lists the 16-bit MPU-S SPI registers. through describe the register bits.

Table 2–118. MPU-S Serial Port Interface Registers

Name	Description	Offset
REG_SET1	SPI setup 1	0x00
REG_SET2	SPI setup 2	0x02
REG_CTRL	SPI control	0x04
REG_STATUS	SPI status	0x06
REG_TX_LSB	Transmit LSB	0x08
REG_TX_MSB	Transmit MSB	0x0A
REG_RX_LSB	Receive LSB	0x0C
REG_RX_MSB	Receive MSB	0x0E

Table 2–119. SPI Setup 1 Register (REG_SET1)

Bit	Name	Function	Reset Value
15:6	Reserved		0x000
5	MSK1	Enable interrupt for read/write cycle 0: Interrupt active 1: Interrupt disable	0x1
4	MSK0	Enable interrupt for write cycle 0: Interrupt active 1: Interrupt disable	0x1
3:1	PTV	Prescale clock divisor 000: 1 001: 2 010: 4 011: 8 100: 16 101: 32 110: 64 111: 128	0x0
0	EN_CLK See Note.	Clock enable 0: Clock is shut off. 1: Clock is running.	0x0

Note: EN_CLK must be enabled before setting other registers.

The SPI 1 setup register configures the serial port.

Table 2–120. Setup SPI 2 Register (REG_SET2)

Bit	Name	Function	Reset Value
15	Reserved		0x0
14:10	L	Format of enable signals nTSPEN 0: Level trigger 1: Edge trigger	0x00
9:5	P	Format of enable signals nTSPEN 0: Negative level 1: Positive level	0x00
4:0	C	Active edge of the clock for each device 0: Falling 1: Rising	0x00

The SPI 2 setup register configures the serial port.

Table 2–121. Control SPI (REG_CTRL)

Bit	Name	Function	Reset Value
15:10	Reserved		0x00
9:7	AD	Mask for SENx selection 000: SEN0 001: SEN1 010: SEN2 Other: Reserved	0x0
6:2	NB	Word size (from 1 to 32), transmission of NB + 1 bits 00000: One-bit transmit 11111: 32-bit transmit	0x00
1	WR	Write process activation	0x0
0	RD	Read and write process activation (toggle at 1)	0x0

The control register activates the serial port and starts the operation of the interface as soon as one of its two bits is set (see Table 2–121). The register defines:

- WRITE activation of the serial port (transmit only)
  1) READ activation of the serial port (simultaneously receive and transmit)
  a) Number of bits to transfer (1 to 32)
  b) External device address

Table 2–122. Status Register (REG_STATUS)

Bit	Name	Function	R/W	Reset Value
15:2	Reserved		R/W	0x0000
1	WE	Write end 1: The serialization is finished.	R	0x0
0	RD	Read end 1: Transmit register is updated.	R	0x0

The status register provides SPI operational information (see Table 2–122). To read the status register or to write to setup register 2, the internal clock must be running (REG_SET1(0) = 1).

Table 2–123. Transmit Registers (REG_TX_LSB/MSB)

Bit	Name	Function	Reset Value
15:0	DATA_TX	Data to transmit	0x0000

The LSB and MSB transmit registers transmit word data and are accessible by the TIPB bus in read or write.

The choice of implementation implies that, whatever its size, the word must be aligned on the MSB side.

Table 2–124. Receive Registers (REG_RX_ MSB/LSB)

Bit	Name	Function	Reset Value
15:0	DATA_RX	Receive data	Undefined

The LSB and MSB receive registers access data from the TIPB bus.

2.10.2 Protocol

The serial port is configured via the setup registers.

After loading at least one of the transmit registers, the serialization and parallelization processes are initiated by setting either the RD bit or the WR bit of the corresponding control register to 1.

A read process is always simultaneous with a write process, because the internal shift register is based on a loop (first-in last-out principle). The concurrent write process can be a dummy process if no data is to be transmitted.

The external transfer of a word starts as soon as one of the transmit clocks is generated. The transmitted data word is shifted out on the rising or falling edge of the transmission clock (CLKX), and the received data word is shifted in on the falling or rising edge of CLKX (complementary edge).

Depending on the value of the P and L parameters, loading the word in the external device is validated on either:

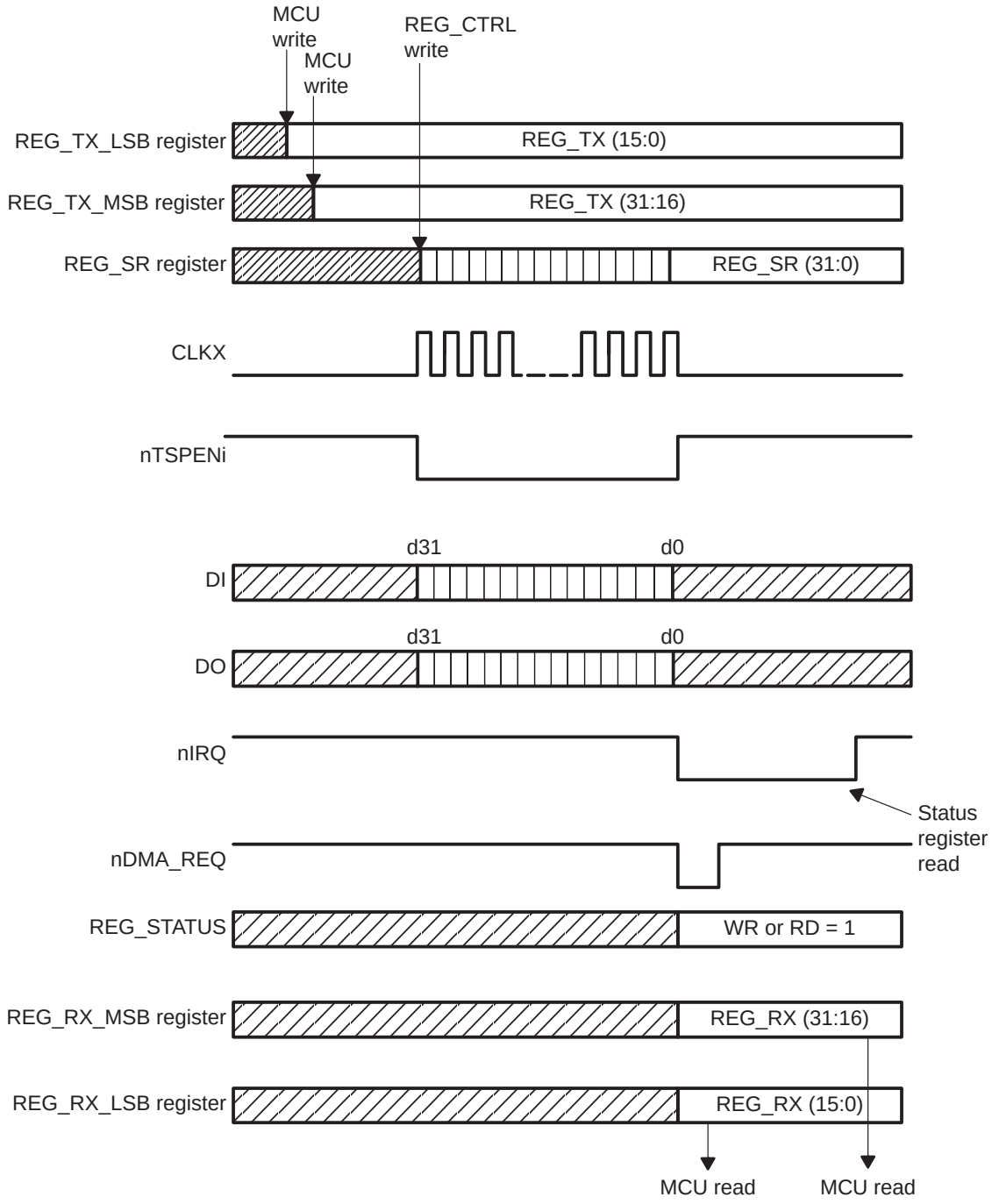
- ❏ Deactivation of the enable signal (nTSPENi) (rising or falling edge)
- ❏ High level/low level of the enable signal (nTSPENi) (load enable of the receiver latch)

An interrupt can be generated (depending of the setup register) at the end of the write or read/write cycle; then the RD bit or the WR bit of the corresponding control register must be cleared.

The IRQ request is reset when the MCU reads the status register.

Figure 2–65 shows the protocol chronograms.

Figure 2–65. Protocol Chronograms



2.10.2.1 Transmit Protocol

The word serialization protocol is as follows:

1) Initialization:

   a) MCU writes (via TIPB bus) in set up register 1 (REG_SET1)
   b) MCU writes in set up register 2 (REG_SET2)

2) Loading of the data word:

   a) MCU writes upper byte (REG_TX_MSB)
   b) MCU writes lower byte [optional] (REG_TX_LSB)

   Start of the serialization:

   c) MCU sets WR bit in control register (REG_CTRL)

3) On WR setting:

   a) Write in REG_SR <- REG_TX_M/L
   b) Activate serialization

   On internal CLK rising edge:

   c) Select external device serial port i (nTSPENi = 0)

   On internal CLK rising (C = 1) or falling (C = 0) edge

4) Data serialization:

   a) Activate CLK

      i) Loop 1 from 0 to n (DO=REG_SR(n-i))
      ii) End loop

   b) Deactivate CLK

5) Deselect external device serial port i (nTSPENj = 1 if (P = 0))

   a) IT generation

2.10.2.2 Receive Protocol

The word parallelization protocol is as follows:

1) Loading of the dummy data word:

   a) MCU writes upper word (REG_TX_MSB)
   b) MCU writes lower word (REG_TX_LSB)

   Start of the concurrent parallelization/serialization:

   c) MCU set RD bit in control register (REG_CTRL)

2) On RD setting:

   a) Then REG_SR <- REG_TX_M/L
   b) Activate parallelization/serialization

3) On internal CLK rising edge:

   a) Select external device serial port i (nTSPENi = 0 if (P = 0))

      On internal CLK rising (C = 1) or falling (C = 0) edge

4) Data serialization:

   a) Activate CLK

      i) Loop I from 0 to n (DO=REG_SR(n-I), REG_SR(I)=DI)
      ii) End loop

   b) Deactivate CLK

5) Unselect external device serial port i (nTSPENi =1 if (P = 0))

   IT generated

6) Reading of received data word by the MCU:

   a) MCU reads MSB [optional] (REG_RX_MSB)
   b) MCU reads LSB (REG_RX_LSB)

The read process requires two data transfers, because the external device must receive the address of the data to be read before it can transmit the data.

Consecutive read processes or read and write processes can be piped and you can save the dummy transfer, which can be replaced by an effective transfer.

2.10.3 Transmission Mode Chronograms

The serial interface is active as soon as the transmit clock is activated.

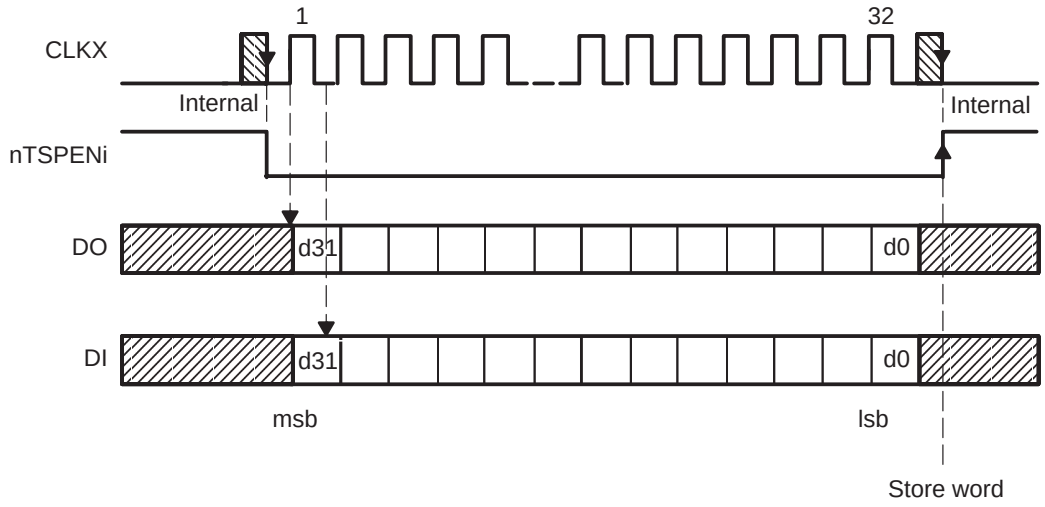
The transmitted data word is shifted out on the rising or falling edge of the transmission clock (CLK), and the received data word is shifted in on the falling or rising edge of CLK (complementary edge).

On the deactivation of the enable signal:

- ❏ The transmitted word is stored in the external device.
- ❏ The received word is stored in the two receive registers of the serial port (REG_RX_MSB and REG_RX_LSB) if the read mode has been selected (bit 0 of register REG_CTRL).

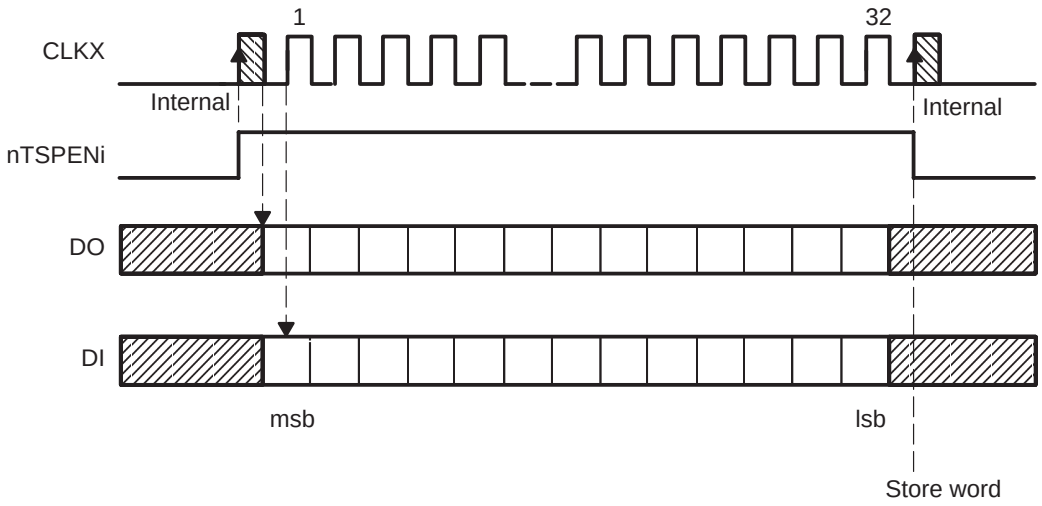
Figure 2–66 through Figure 2–68 show the transmission mode chronograms.

Figure 2–66. Transmission Mode Chronogram 1



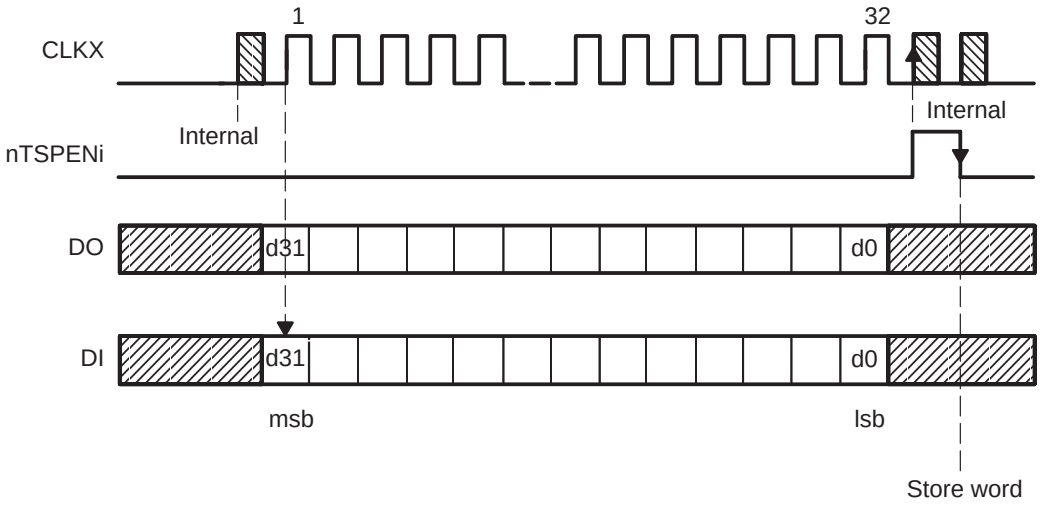
Case C=0 : DO on rising edge, DI on falling edge
P=0; L=0

Figure 2–67. Transmission Mode Chronogram 2



Case C=0 : DO on falling edge, DI on rising edge
P=1; L=0

Figure 2–68. Transmission Mode Chronogram 3



Case C=0 : DO on falling edge, DI on rising edge
P=1; L=1

2.10.4 SPI Inputs/Outputs

Table 2–125 lists the SPI inputs/outputs.

Table 2–125. SPI Inputs/Outputs

Name	Function	Direction
	Clock	
SCLK	Bit clock	Out
	Data	
SDO	Serial data output	Out
SDI	Serial data input	In
	Device selection	
SEN0	Enable device 0	Out
SEN1	Enable device 1	Out
SEN2	Enable device 2	Out

2.10.5 Timing Characterization

Table 2–126 lists the SPI timing characterizations.

Table 2–126. SPI Timing Characterization

Condition	V_{DD}	T_j	Process
Worst case delay (industrial)	2.3 V	125°C	Max
Best case delay (industrial)	2.7 V	-40°C	Min

2.10.6 MPU TIPB Bus

Figure 2–69 shows the TIPB peripheral parameters. Table 2–127 shows the TIPB peripheral timings.

Figure 2–69. TIPB Peripheral Parameters

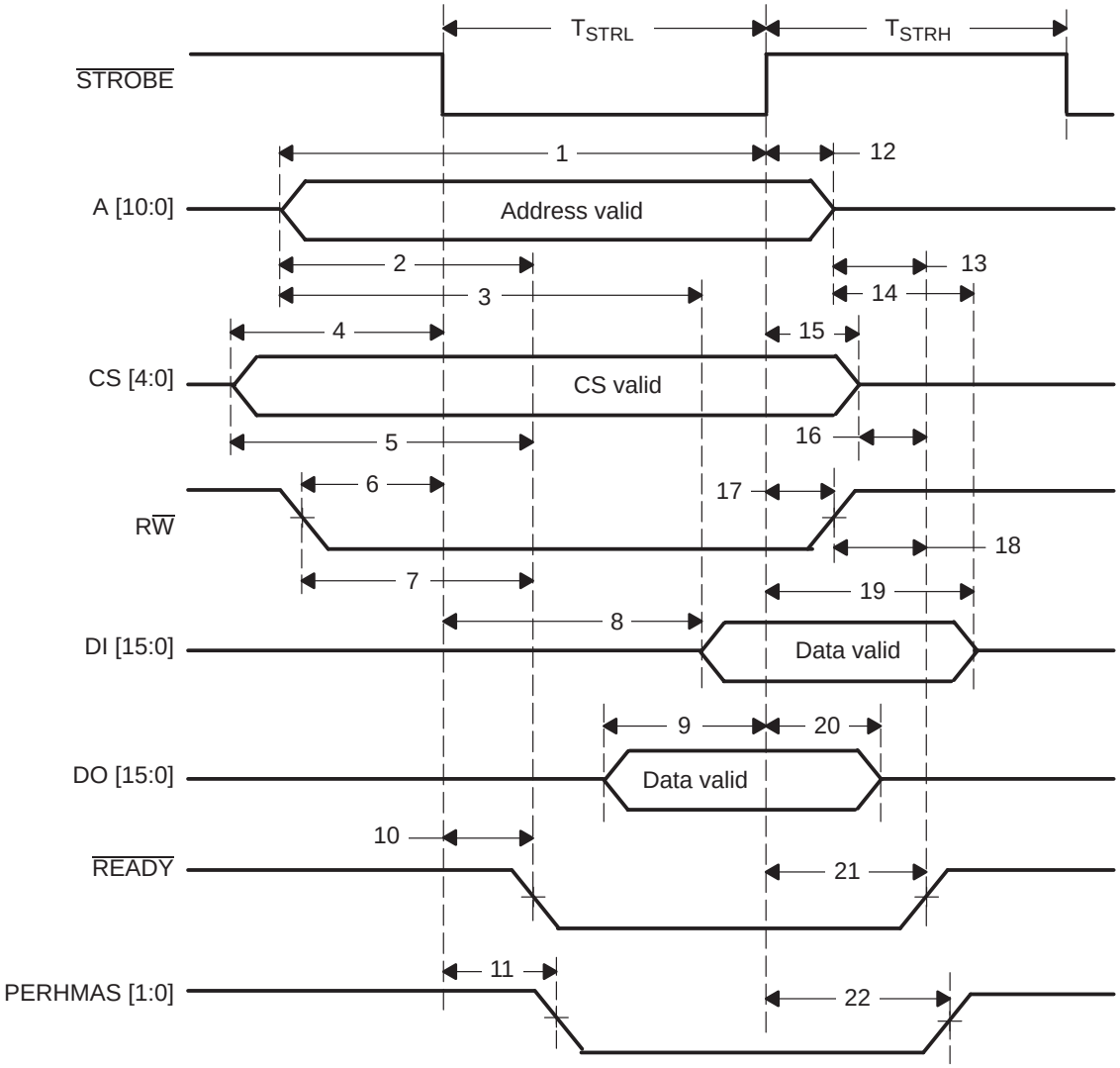


Table 2–127. TIPB Peripheral Timings

Ref.	Definition	Timing (ns)	
		Minimum	**Maximum**
1	$Ts_{A->nSTROBE}$		1.95
2	$Ts_{A->nREADY}$		3.39
3	$Ts_{A->DI}$		3.76
4	$Ts_{CS->nSTROBE}$		0.78
5	$Ts_{CS->nREADY}$		3.15
6	$Ts_{RnW->nSTROBE}$		1.18
7	$Ts_{RnW->nREADY}$		2.59
8	$Ts_{nSTROBE->DI}$		3.08
9	$Ts_{nDO->nSTROBE}$		0.05
10	$Ts_{nSTROBE->nREADY}$		2.02
11	$Ts_{nSTROBE->PERHMAS}$		2.02
12	$Th_{nSTROBE->A}$	0.01	
13	$Th_{A->nREADY}$	0.89	
14	$Th_{A->DI}$	0.97	
15	$Th_{nSTROBE->CS}$	0.27	
16	$Th_{CS->nREADY}$	0.92	
17	$Th_{nSTROBE->RnW}$	0.67	
18	$Th_{RnW->nREADY}$	0.78	
19	$Th_{nSTROBE->DI}$	1.77	
20	$Th_{nSTROBE->DO}$	0.55	
21	$Th_{nSTROBE->nREADY}$	1.10	
22	$Th_{nSTROBE->PERHMAS}$	1.10	

2.10.7 Serial Interface

Figure 2–70 shows the serial interface parameters. Table 2–128 shows the serial interface timings.

Figure 2–70. Serial Interface Parameters

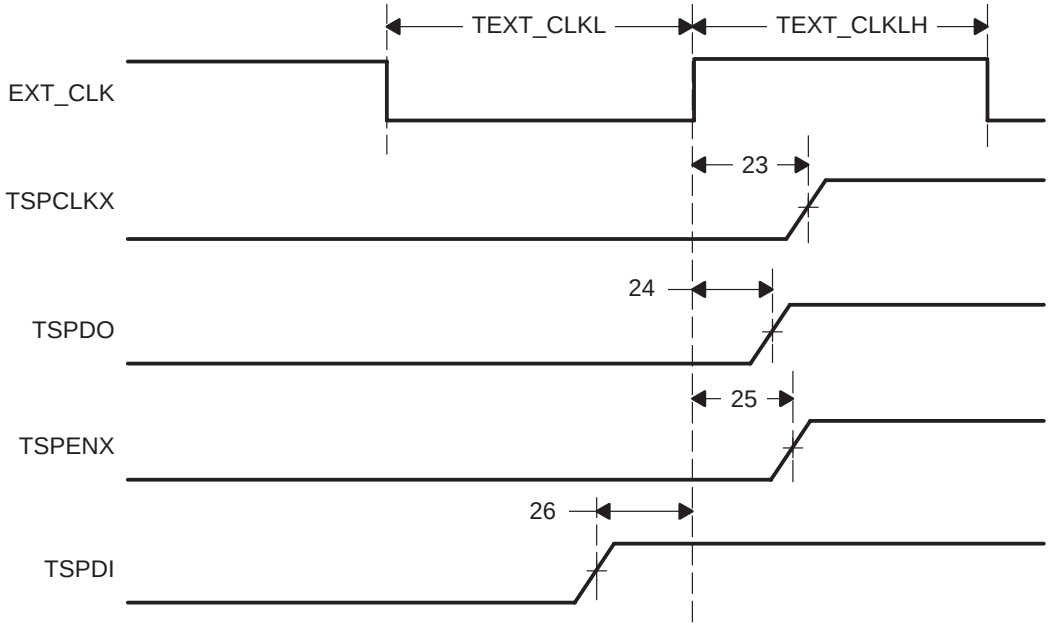


Table 2–128. Serial Interface Timings

Ref.	Definition	Timing (ns)	
		Minimum	**Maximum**
23	$Th_{ext\_clk->tspclkx}$	1.50	
24	$Th_{ext\_clk->tsdo}$	1.47	
25	$Th_{ext\_clk->tspenX}$	1.45	
26	$Ts_{ext\_clk->tspdi}$		2.29

2.11 MPU General-Purpose Input/Output

The MPU-S contains six general-purpose input/output (GPIO) controller modules.

The MPU asynchronous GPIO module does not perform input line debouncing; it does not need a 13-MHz clock. It can wake up the system when the 13-MHz clock is off on an input line level change.

Table 2–129. GPIO Modules

GPIO	MODULE	IO	GPIO	MODULE	IO	GPIO	MODULE	IO
GPIO_0	1	0	GPIO_59	2	27	GPIO_118	4	22
GPIO_1	1	1	GPIO_60	2	28	GPIO_119	4	23
GPIO_2	1	2	GPIO_61	2	29	GPIO_120	4	24
GPIO_3	1	3	GPIO_62	2	30	GPIO_121	4	25
GPIO_4	1	4	GPIO_63	2	31	GPIO_122	4	26
GPIO_5	1	5	GPIO_64	2	0	GPIO_123	4	27
GPIO_6	1	6	GPIO_65	3	1	GPIO_124	4	28
GPIO_7	1	7	GPIO_66	3	2	GPIO_125	4	29
GPIO_8	1	8	GPIO_67	3	3	GPIO_126	4	30
GPIO_9	1	9	GPIO_68	3	4	GPIO_127	4	31
GPIO_10	1	10	GPIO_69	3	5	GPIO_128	4	0
GPIO_11	1	11	GPIO_70	3	6	GPIO_129	5	1
GPIO_12	1	12	GPIO_71	3	7	GPIO_130	5	2
GPIO_13	1	13	GPIO_72	3	8	GPIO_131	5	3
GPIO_14	1	14	GPIO_73	3	9	GPIO_132	5	4
GPIO_15	1	15	GPIO_74	3	10	GPIO_133	5	5
GPIO_16	1	16	GPIO_75	3	11	GPIO_134	5	6
GPIO_17	1	17	GPIO_76	3	12	GPIO_135	5	7
GPIO_18	1	18	GPIO_77	3	13	GPIO_136	5	8
GPIO_19	1	19	GPIO_78	3	14	GPIO_137	5	9
GPIO_20	1	20	GPIO_79	3	15	GPIO_138	5	10
GPIO_21	1	21	GPIO_80	3	16	GPIO_139	5	11
GPIO_22	1	22	GPIO_81	3	17	GPIO_140	5	12
GPIO_23	1	23	GPIO_82	3	18	GPIO_141	5	13
GPIO_24	1	24	GPIO_83	3	19	GPIO_142	5	14
GPIO_25	1	25	GPIO_84	3	20	GPIO_143	5	15
GPIO_26	1	26	GPIO_85	3	21	GPIO_144	5	16
GPIO_27	1	27	GPIO_86	3	22	GPIO_145	5	17
GPIO_28	1	28	GPIO_87	3	23	GPIO_146	5	18

Table 2–129. GPIO Modules (Continued)

GPIO	MODULE	IO	GPIO	MODULE	IO	GPIO	MODULE	IO
GPIO_29	1	29	GPIO_88	3	24	GPIO_147	5	19
GPIO_30	1	30	GPIO_89	3	25	GPIO_148	5	20
GPIO_31	1	31	GPIO_90	3	26	GPIO_149	5	21
GPIO_32	1	0	GPIO_91	3	27	GPIO_150	5	22
GPIO_33	2	1	GPIO_92	3	28	GPIO_151	5	23
GPIO_34	2	2	GPIO_93	3	29	GPIO_152	5	24
GPIO_35	2	3	GPIO_94	3	30	GPIO_153	5	25
GPIO_36	2	4	GPIO_95	3	31	GPIO_154	5	26
GPIO_37	2	5	GPIO_96	3	0	GPIO_155	5	27
GPIO_38	2	6	GPIO_97	4	1	GPIO_156	5	28
GPIO_39	2	7	GPIO_98	4	2	GPIO_157	5	29
GPIO_40	2	8	GPIO_99	4	3	GPIO_158	5	30
GPIO_41	2	9	GPIO_100	4	4	GPIO_159	5	31
GPIO_42	2	10	GPIO_101	4	5	GPIO_160	6	0
GPIO_43	2	11	GPIO_102	4	6	GPIO_161	6	1
GPIO_44	2	12	GPIO_103	4	7	GPIO_162	6	2
GPIO_45	2	13	GPIO_104	4	8	GPIO_163	6	3
GPIO_46	2	14	GPIO_105	4	9	GPIO_164	6	4
GPIO_47	2	15	GPIO_106	4	10	GPIO_165	6	5
GPIO_48	2	16	GPIO_107	4	11	GPIO_166	6	6
GPIO_49	2	17	GPIO_108	4	12	GPIO_167	6	7
GPIO_50	2	18	GPIO_109	4	13	GPIO_168	6	8
GPIO_51	2	19	GPIO_110	4	14	GPIN_1	6	9
GPIO_52	2	20	GPIO_111	4	15	GPIN_2	6	10
GPIO_53	2	21	GPIO_112	4	16	GPIN_3	6	11
GPIO_54	2	22	GPIO_113	4	16	GPIN_4	6	12
GPIO_55	2	23	GPIO_114	4	18	GPIN_5	6	13
GPIO_56	2	24	GPIO_115	4	19			
GPIO_57	2	25	GPIO_116	4	20			
GPIO_58	2	26	GPIO_117	4	21			

2.11.1 GPIO Registers

The data input register (offset: 0x00) registers the data read from the GPIO input pins. The input data is captured synchronously and clocked in on the rising edge of ARM_GPIO_CK. The data input register is a read-only register. The GPIO input data is captured into this register three clock cycles after the GPIO

input pin(s) change. This is because the DFF is used for synchronization and debouncing to remove any input glitches. Bits not configured as input are undefined during read back.

The data output register (offset: 0x04) writes data to the GPIO output pins. Data is written to the data output register on the rising edge of TIPB clock.

The direction control register (offset: 0x08) configures the GPIO pins for either input or output. At reset, all of the GPIO pins are configured as inputs. Data is written to the direction control register on the rising edge of TIPB clock.

The interrupt control register (offset: 0x0C) defines when an interrupt request occurs. The interrupt can either be generated from a high-to-low transition (function 0) or from a low to high transition (function 1). Data is written to the interrupt control register on the rising edge of TIPB clock.

The interrupt mask register (offset: 0x10) masks certain input pins from generating an interrupt request. Data is written into the interrupt mask register on the rising edge of TIPB clock.

The interrupt status register (offset: 0x14) determines which input pin requested an interrupt. Bit 0 corresponds to GPIO_PIN 0 and so forth. If the value is a 1, then that pin is requesting the interrupt. The processor services the interrupt and resets the appropriate bit in the status register. A 1 must be written to the appropriate bit in order to reset it, but an interrupt cannot be generated by writing a 1 to the interrupt status register. Writing a 0 to a bit in the status register does not change its value. The *interrupt status register* uses the rising edge of ARM_GPIO_CK to capture data.

There are six sets of parallel registers, and each set is associated with one of the following modules:

- The MPUIO32_1 registers (FFFB:C000) allow configuration of GPIO_[0:31].
- The MPUIO32_2 registers (FFFB:C800) allow configuration of GPIO_[32:63].
- The MPUIO32_3 registers (FFFB:D000) allow configuration of GPIO_[64:95].
- The MPUIO32_4 registers (FFFB:D800) allow configuration of GPIO_[96:127].
- The MPUIO32_5 registers (FFFB:E000) allow configuration of GPIO_[128:159].
- The MPUIO32_6 registers (FFFB:E800) allow configuration of GPIO_[160:170] and GPIN_(1:5).

Table 2–130 lists the 32-bit GPIO registers, and Table 2–131 through Table 2–136 describe the register bits.

Table 2–130. GPIO Registers

Name	Description	R/W	Offset
DATA_INPUT	Data input	R	0x00
DATA_OUTPUT	Data output	R/W	0x04
DIRECTION_CONTROL	Direction control	R/W	0x08
INTERRUPT_CONTROL	Interrupt control	R/W	0x0C
INTERRUPT_MASK	Interrupt mask	R/W	0x10
INTERRUPT_STATUS	Interrupt status	R/W	0x14

Note: n = 0 for MPUIO32_1
n = 1 for MPUIO32_2
n = 2 for MPUIO32_3
n = 3 for MPUIO32_4
n = 4 for MPUIO32_5
n = 5 for MPUIO32_6

Table 2–131. Data Input Register (DATA_INPUT)

Bit	Name	Function	Reset Value
31:0	RECEIVE_DATA	Register data read from the GPIO input pins Input data captured synchronously and clocked in on rising edge of ARM_GPIO_CK	0x0000 0000

The data input register controls the data read from the GPIO input.

Table 2–132. Data Output Register (DATA_OUTPUT)

Bit	Name	Function	Reset Value
31:0	DATA_TO_TRANSMIT	Write data to the GPIO output pins Data written to data output register on rising edge of TIPB clock	0x0000 0000

The data output register controls the data to be transmitted to the GPIO output pins.

Table 2–133. Direction Control Register (DIRECTION_CONTROL)

Bit	Name	Function	Reset Value
31:0	DIRECTION_CONTROL	0: Output 1: Input	0xFFFF FFFF

The direction control register configures the direction of the GPIO pins.

Table 2–134. Interrupt Control Register (INTERRUPT_CONTROL)

Bit	Name	Function	Reset Value
31:0	INTERRUPT_EDGE_ CONTROL	0: High-to-low transition 1: Low-to-high transition	0xFFFF FFFF

The interrupt control register configures the GPIO interrupt trigger.

Table 2–135. Interrupt Mask Register (INTERRUPT_MASK)

Bit	Name	Function	Reset Value
31:0	INTERRUPT_MASK	0: Unmasked 1: Masked	0xFFFF FFFF

The interrupt mask register masks out specific input pins from the interrupt generation logic.

Table 2–136. Interrupt Status Register (INTERRUPT_STATUS)

Bit	Name	Function	Reset Value
31:0	INTERRUPT_STATUS	Read 0: No interrupt request 1: Interrupt request, used to acknowledge the IRQ by writing a 1 in the appropriate bit	0x0000 0000

The interrupt status register provides current interrupt status of each GPIO pin.

2.11.2 Interrupt Logic Block

The interrupt logic generates the interrupt request to the CPU when an input pin transitions between logic states.

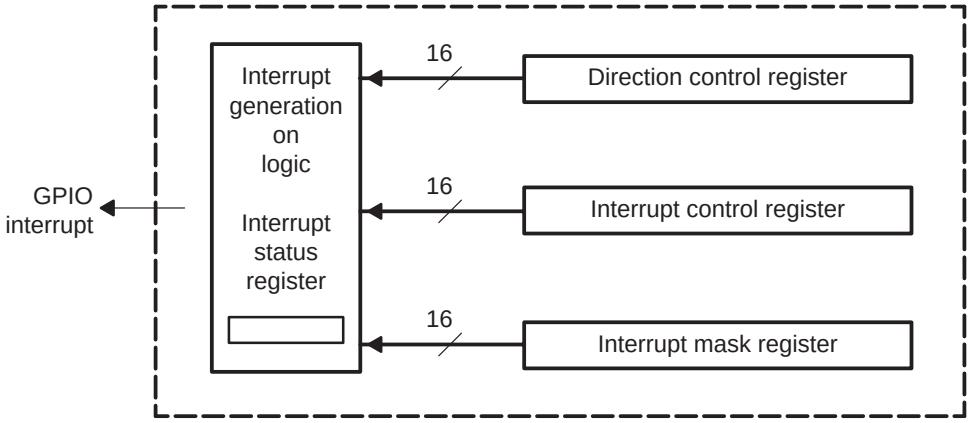
For an interrupt to occur, the GPIO pin must be configured as an input pin via the direction control register, and the interrupt mask register bit must be set to unmasked. Then the interrupt logic checks for the appropriate transitions on the input pins.

You can define whether an interrupt occurs when there is a high-to-low or low-to-high transition by writing to the interrupt control register.

When the appropriate transition on the input pin occurs, then the interrupt signal GPIO_INT is generated. The interrupt status register is automatically updated with the interrupt status of the input pins. The CPU can read this register to discover which input pin needs servicing. After servicing the interrupt, the CPU resets the status bit (see Table 2–136, *Interrupt Status Register*).

Figure 2–71 shows the registers used to generate an interrupt.

Figure 2–71. Registers Used to Generate Interrupt

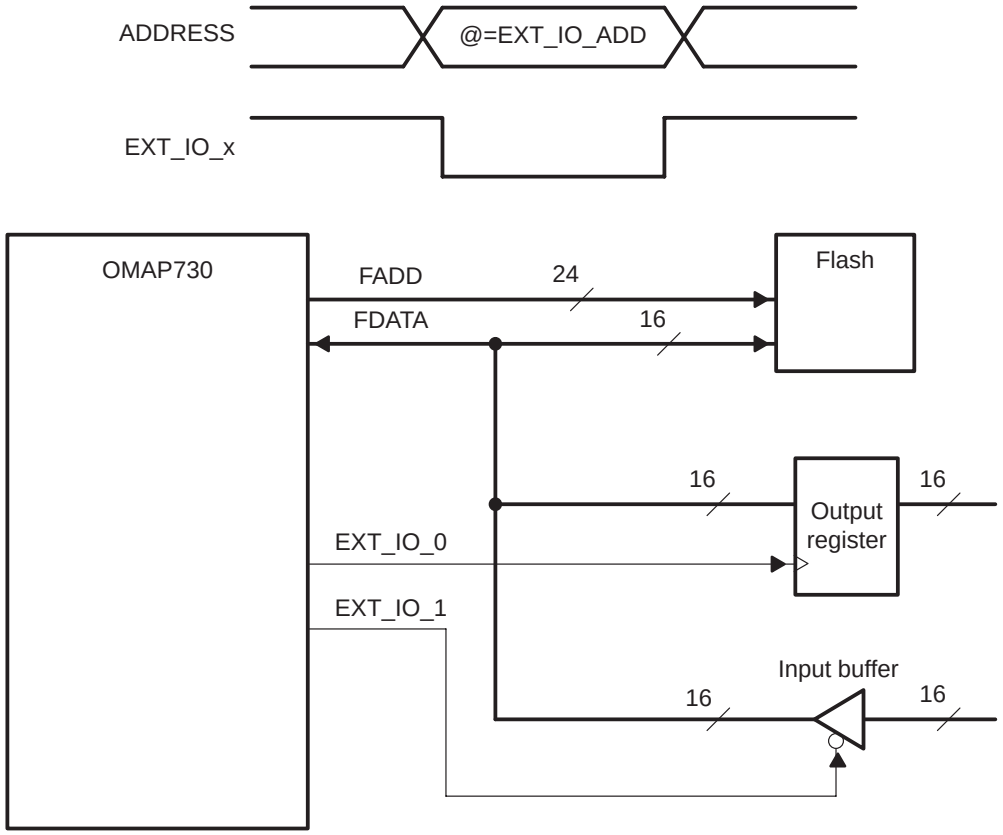


2.11.3 External GPIO Expansion

In addition to the internal GPIO controllers, another mechanism extends the number of GPIOs using external devices. It is based on the use of EXT_IO[3:0] signals to control external register and 3-state drivers.

Figure 2–72 shows a 16-bit register controlled by EXT_IO_0 to build a 16-bit general-purpose output function, and where EXT_IO_1 is used to control a 16-bit general-purpose input function via a 16-bit 3-state drive.

Figure 2–72. Extended GPIOs



EXT_IO_[0:3] control the input buffer or the output register.

When the EXTernal_IO mode is selected, four addresses (0xFFFFF8, 0xFFFFFA, 0xFFFFFC, and 0xFFFFFE) of one of the flash CS memory spaces (either CS2 or CS3) are reserved to access the external GPIO function. The CS selection is done using EXT_IO_CS_SEL in PERSEUS2_MODE1 register.

When an access is done to the EXT_IO addresses (0xFFFFF8, 0xFFFFFA, 0xFFFFFC, or 0xFFFFFE), the corresponding EXT_IO_[0:3] signal can be used for one of the following functions:

- If it is configured to be used as a control output register (EXT_IO_CTRL(x) in PERSEUS2_MODE1 register = 0), it must be used as a write-enable signal.
- If it is configured to be used as a control input buffer (EXT_IO_CTRL(x) in PERSEUS2_MODE1 register = 1), it must be used as a read-enable signal.

PERSEUS2_MODE1 in PERSEUS2_CONF registers must be used to configure the EXTERNAL IO modes, as shown in Table 2–137.

Table 2–137. External I/O Pins

Multimode Pins			Reset		Pad Location	Pad Name
EXT_IO_3	IN/OUT	MPU extended generic IO signals	I	Z	W5 or D3	MPU_I2C_SCK or KBC_1 ‡
EXT_IO_2			I	Z	V6 or C2	MPU_I2C_SDA or KBC_0 ‡
EXT_IO_1			I	Z	J7 or N7	KBR_1 or ARM_BOOT_MLPG2_MLPG2 ‡
EXT_IO_0			I	Z	E2 or N2	KBR_0 or MUX_MODE_MLPG1 ‡

‡ Pin used in multimode function

2.11.4 Extended GPIO Register

Table 2–138 lists the external I/O register. Table 2–139 describes the register bits.

Table 2–138. External I/O Register

Register	Description	Address
PERSEUS2_MODE1	OMAP mode configuration	0x10

Table 2–139. Mode 1 Register (PERSEUS2_MODE1)

Bit	Name	Function	Reset Value
22:19	EXT_IO_CTRL	0: Control output register 1: Control input buffer EXT_IO_CTRL(5) => EXT_IO_3 EXT_IO_CTRL(4) => EXT_IO_2 EXT_IO_CTRL(3) => EXT_IO_1 EXT_IO_CTRL(2) => EXT_IO_0	0x0
18	EXT_IO_CS_SEL	0: CS2 memory space selected for EXT_IO extension 1: CS3 memory space selected for EXT_IO extension	0x0
17	EXT_IO_EN	1: Enable CS3 (or CS2), disable when Add = FFFFF8 - FFFFFE and Ext_IO enabled 0: Disable EXT_IO function Accesses to the following CS space offset are interpreted as EXT_IO_X command: EXT_IO_3: Add 0xFFFFFE EXT_IO_2: Add 0xFFFFFC EXT_IO_1: Add 0xFFFFFA EXT_IO_0: Add 0xFFFFF8	0x0

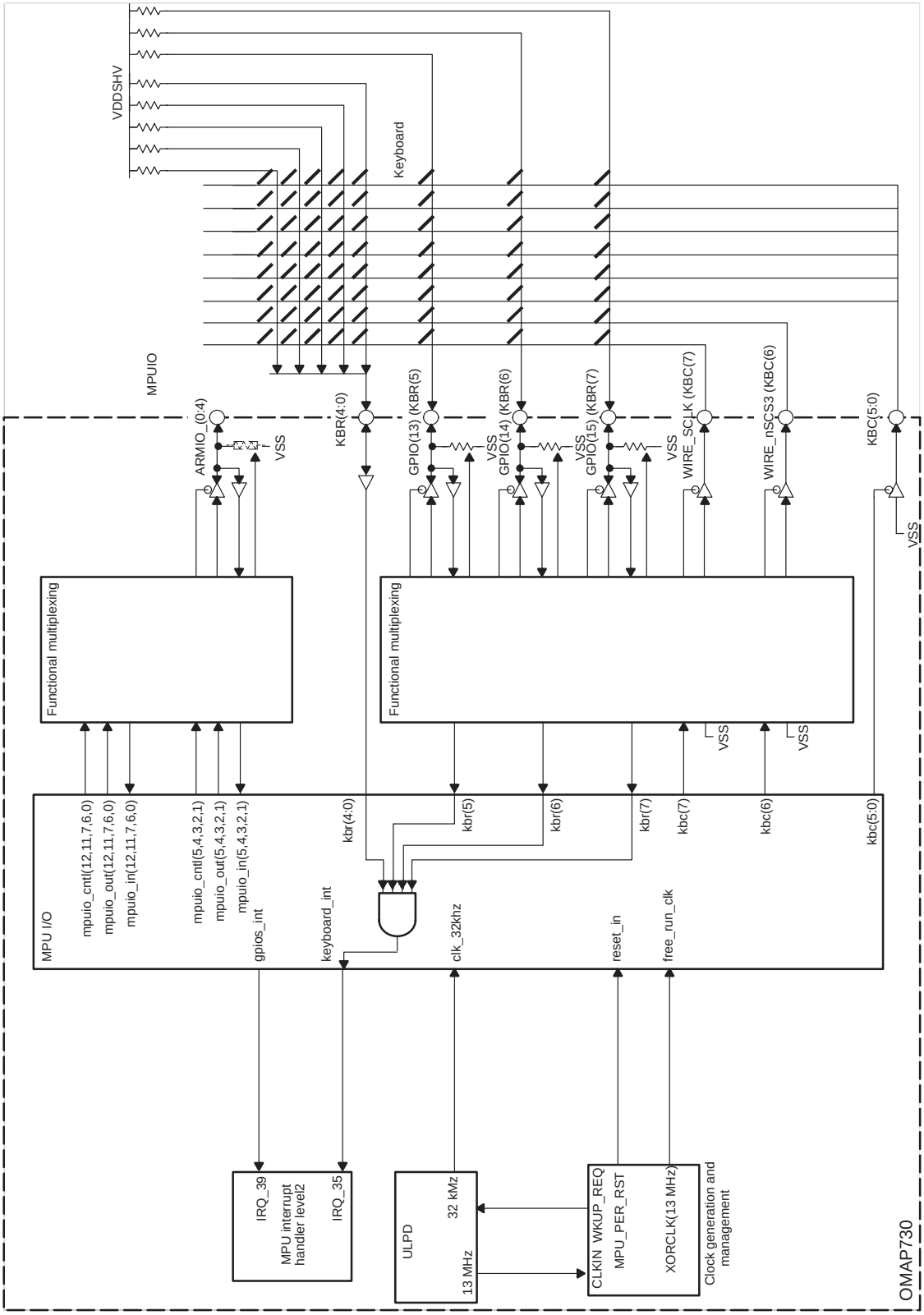
2.12 MPU I/O

The MPUIO module enables direct I/O communications between the MPU (through the public TIPB) and external devices (see the Figure 2–73).

Two types of I/Os can be used:

- Specific I/Os dedicated for 8x8 keyboard connection:
  - Eight inputs (KB R[7:0]) for row lines
  - Eight outputs (KB.C[7:0]) for column lines
- General-purpose I/Os:
  - Five MPU I/O signals (5, 4, 3, 2, and 1) are available in the default OMAP730 multiplexing.
  - Five additional MPU I/O signals (12, 11, 7, 6, and 0) can be used by configuring the OMAP730 multiplexing. For more detail, see Section Appendix C.3, *I/O Functional Multiplexing*.

Figure 2–73. MPU I/O Environment



2.12.1 MPU I/O Interrupts

The MPU I/O module generates two interrupts:

- The keyboard interrupt (KEYBOARD_INT), used to detect a key press, connected to the MPU interrupt handler level2, line1 (edge-sensitive)
- The MPUIO interrupt (GPIOS_INT), used to detect an edge on one MPUIO input, connected to the MPU interrupt handler level2, line5 (level-sensitive).

2.12.2 MPU I/O Clocks and Reset

The MPU I/O module has two clocks:

- The 32-kHz system clock (CLK_32KHZ), which comes through the ULPD from either the OMAP730 32-kHz oscillator or the OMAP730 CLK32K_IN CMOS input. For more details, see Chapter 5, *Clock Generation and System Reset Management*.
- The 13-MHz clock (FREE_RUN_CLK), used to resynchronize the GPIO_INT register read. Comes from the MPU peripheral fixed clock (XORCLK). This clock is free-running when OMAP730 is awake.

The MPU TIPB reset (MPU_PER_RESET) resets the MPU I/O module.

2.12.3 Keyboard Interface

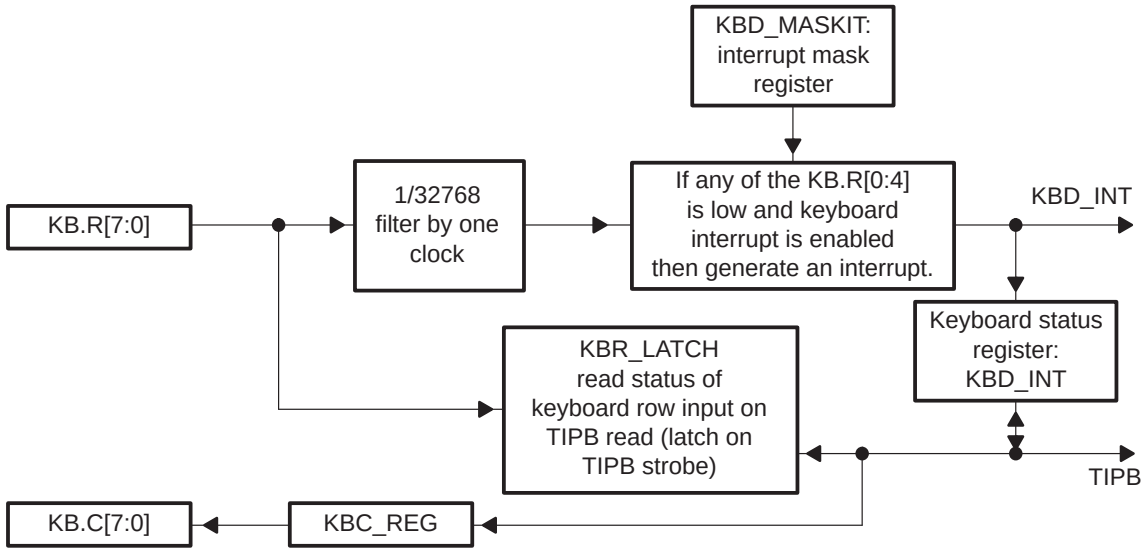
To allow button press detection:

- All the row lines (KB.R) must have an external pullup.
- All the column lines (KB.C) drive a low level (idle state of Table 2–140).

The keyboard interrupt (KEYBOARD_INT) to the MPU is an AND of the eight row lines filtering during one 32-kHz clock period (CLK_32KHZ).

As soon as any key of the keyboard matrix is pressed, the corresponding row and column lines are shorted together and a low level is driven on the corresponding row line, generating a keyboard interrupt (see Figure 2–74).

Figure 2–74. Keyboard Process Block Diagram



Once the keyboard interrupt is received, the MPU scans the column lines in the sequence described in the Table 2–140 in order to detect the key that has been pressed.

Table 2–140. Keyboard Scanning Sequence

	Idle	Keyboard Scanning									Idle
KB.C[0]	0	1	0	1	1	1	1	1	1	1	0
KB.C[2]	0	1	1	1	0	1	1	1	1	1	0
KB.C[3]	0	1	1	1	1	0	1	1	1	1	0
KB.C[4]	0	1	1	1	1	1	0	1	1	1	0
KB.C[5]	0	1	1	1	1	1	1	0	1	1	0
KB.C[6]	0	1	1	1	1	1	1	1	0	1	0
KB.C[7]	0	1	1	1	1	1	1	1	1	0	0
KB.C[1]	0	1	1	0	1	1	1	1	1	1	0

For each step of the sequence, the MPU:

- ❏ Writes the specific value (with a low level on one column line) in the KBC_REG register
- ❏ Reads the value of the KBR_LATCH register and thus detects if one key of the concerned column line (this one which drives a low level) is pressed

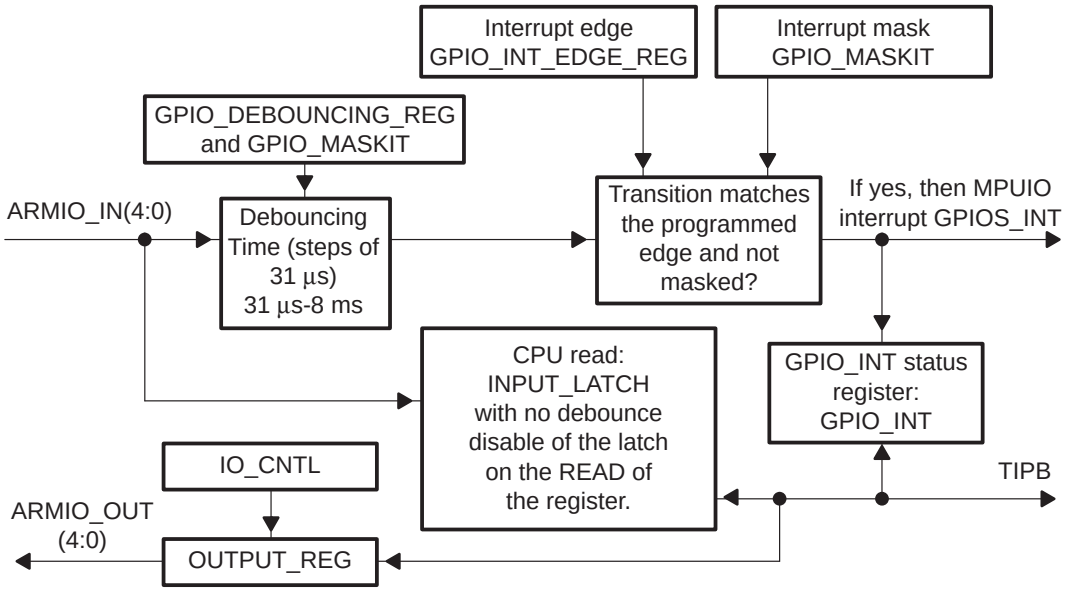
At the end of the scanning sequence, the MPU is able to establish which keys have been pressed.

2.12.4 MPUIO Interface

This interface has the following characteristics (see Figure 2–75):

- ❑ Every I/O can be configured individually either in input or in output mode.
- ❑ Interrupt generation (GPIOS_INT) on edge detection (rising or falling) after debouncing preprocessing
- ❑ Edge detection can be used to latch all the MPUIOs (event capture mode).
- ❑ MPUIO interface works with the 32-kHz-system clock and consequently can be used to wake up the OMAP730 device by generating the MPUIO interrupt.

Figure 2–75. MPUIO Process



2.12.5 MPUIO Interrupt Reset

The MPUIO interrupt (GPIOS_INT) is generated when one event occurs on one MPU I/O input (see Figure 2–76).

The edge detection and the interrupt generation are done synchronously with the 32-kHz system clock (CLK_32KHZ).

These events (and consequently the GPIO_INT interrupt) are reset on one MPUIO interrupt register (GPIO_INT) read.

Only the bits that are active after masking are reset.

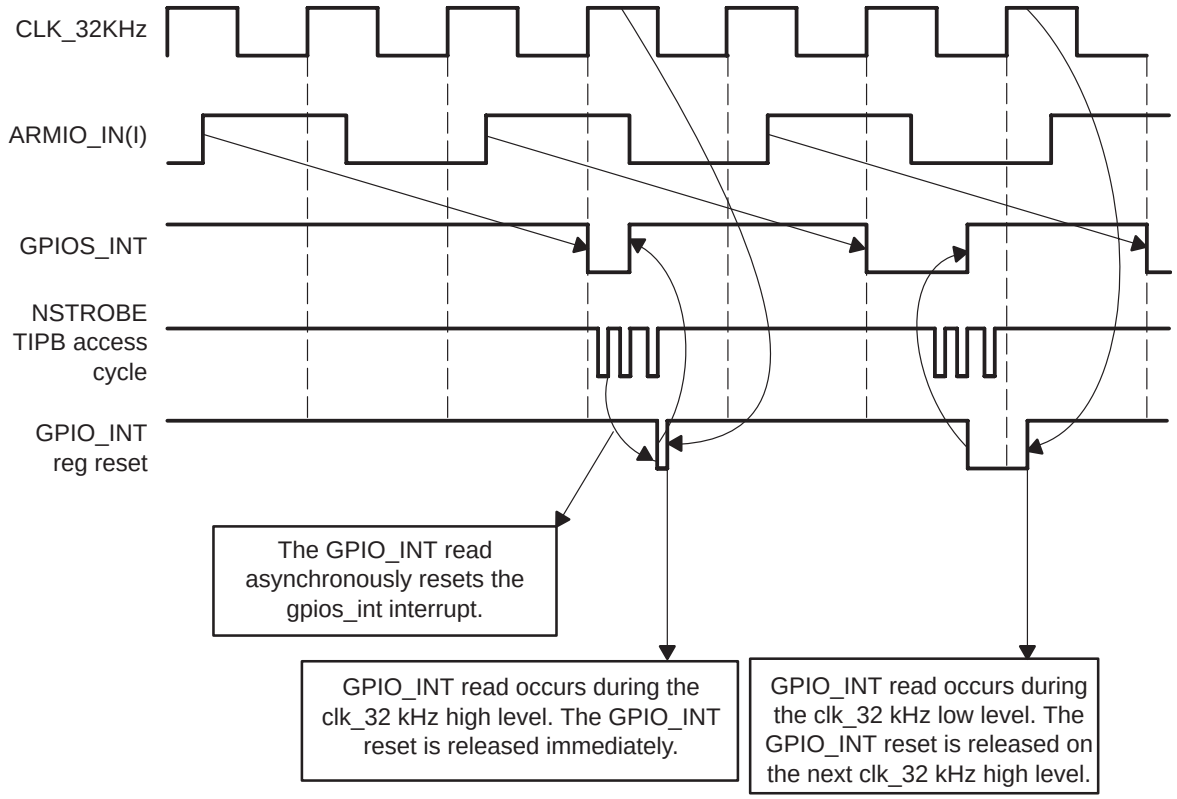
The GPIO_INT reset is synchronously asserted and synchronously released with the 32-kHz system clock. The GPIO_INT register read and the 32-kHz-system clock are resynchronized with the MPU TIPB fixed peripheral clock (12-MHz clock) FREE_RUN_CLOCK.

When the GPIO_INT read occurs:

- ❏ During a high level of the system clock, the release of the reset is done immediately.
- ❏ During a low level of the system clock, the reset is done on the next high level of the system clock.

Even the worst case (reset release on the next 32-kHz cycle) supports the maximum speed of the MPU I/O module (one edge can be detected every two 32-kHz cycles with a debouncing 0).

Figure 2–76. GPIO_INT Register Read Timing



2.12.6 MPUIO Interrupt Masking

The MPUIO interrupt mask register (GPIO_MASKIT) can mask the edge detection on the MPU I/O inputs.

This mask is applied asynchronously on each detected edge after debouncing. If all the edges detected are masked, then GPIOS_INT is masked.

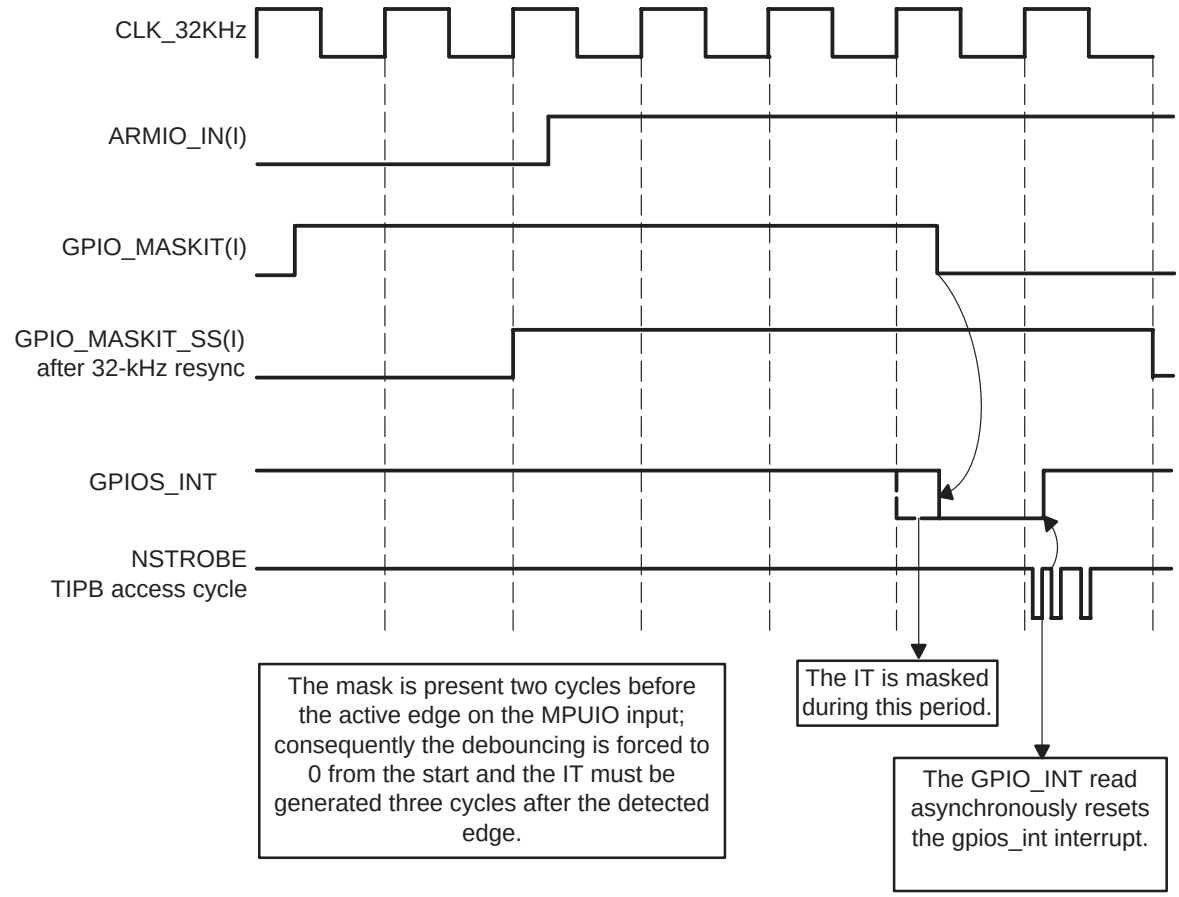
Masking one MPU I/O input forces its corresponding debouncing value to 0, which ensures that GPIOS_INT is generated three cycles after the corresponding detected edge.

To ensure that this force is active from the start of the edge detection, the mask must be present two cycles before the detected edge. In this case, the interrupt

is generated three cycles after the detected edge (see the corresponding timing in Figure 2–77). Otherwise, the mask can be activated during the debouncing period; the debouncing is then forced dynamically to 0 and the interrupt is generated five cycles after the mask presence. You must decide whether or not to mask these interrupts by maintaining or releasing the mask activation.

When one detected edge is masked, the event is not reset when a GPIO_INT register read occurs. Thus, when the mask becomes inactive, the corresponding detected edge generates the GPIOS_INT and this interrupt is reset on the next MPUIO interrupt register (GPIO_INT) read, as shown in Figure 2–77.

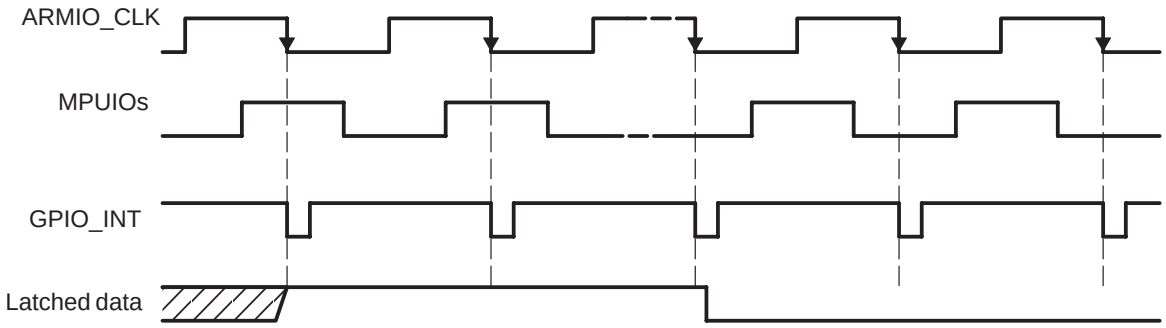
Figure 2–77. MPU/O Input Masking Timing



2.12.7 Event Capture Module

The MPUIO event capture mode allows latching the input value present on the MPUIO ports each time a rising or a falling edge occurs on a selected MPUIO port, here called ARMIO_CLK. If not masked, the ARMIO_CLK-selected edge generates an interrupt to the processor, as shown in Figure 2–78.

Figure 2–78. ARMIO_CLK Timing



ARMIO_CLK can be generated from an external physical module. Consequently, it may be necessary to insert a debouncing delay on this signal. The debouncing time is programmable in the MPUIO debouncing register (GPIO_DEBOUNCING_REG) in steps of 31 μs.

The MPUIO event mode register (GPIO_EVENT_MODE_REG) enables or disables the MPUIO event mode. It also selects the external pin used as the ARMIO_CLK. The MPUIO interrupt edge register (GPIO_INT_EDGE) selects the ARMIO_CLK falling or rising edge to generate the GPIO_INT interrupt.

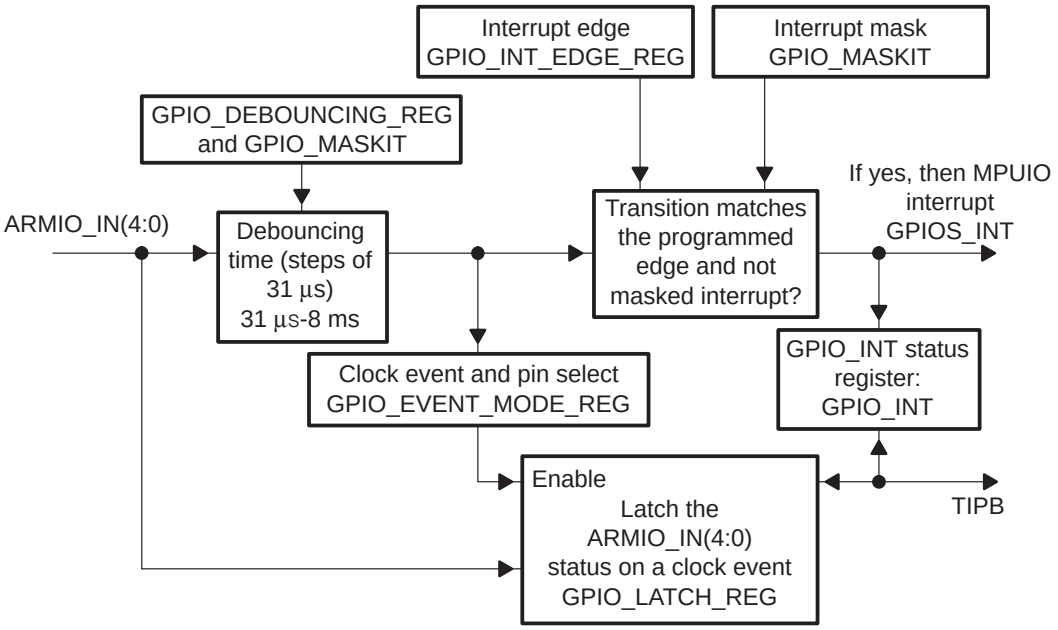
When the GPIOS_INT interrupt (active low) is generated, the GPIO_INT register must be read by the MPU to define any active MPUIO signal interrupts.

The MPUIO interrupt mask register (GPIO_MASKIT) masks MPUIO interrupts individually.

On the ARMIO_CLK programmed edge, after the debouncing delay, the internal ARMIO_IN bus is latched in the MPUIO latch register (GPIO_LATCH_REG). Its value can be read after the detection of the interrupt, even if the external value has changed.

The event capture process is shown in Figure 2–79.

Figure 2–79. Event Capture Process



2.12.8 MPU I/O Registers

Table 2–141 lists the 16-bit MPU I/O registers. Table 2–142 through Table 2–154 describe the individual registers. Start address in the MPU I/O range (hex): FFFB:5000

Table 2–141. MPU Input/Output Registers

Register	Description	R/W	Offset
INPUT_LATCH	General-purpose input	R	0x00
OUTPUT_REG	Output	R/W	0x04
IO_CNTL	Input/Output control	R/W	0x08
KBR_LATCH	Keyboard row inputs	R	0x10
KBC_REG	Keyboard column outputs	R/W	0x14
GPIO_EVENT_MODE_REG	GPIO event mode	R/W	0x18
GPIO_INT_EDGE_REG	GPIO interrupt edge	R/W	0x1C
KBD_INT	Keyboard interrupt	R	0x20
GPIO_INT	GPIO interrupt	R	0x24
KBD_MASKIT	Keyboard mask interrupt	R/W	0x28
GPIO_MASKIT	GPIO mask interrupt	R/W	0x2C
GPIO_DEBOUNCING_REG	GPIO debouncing	R/W	0x30
GPIO_LATCH_REG	GPIO latch	R	0x34

Table 2–142. General-Purpose Input Register (INPUT_LATCH)

Bit	Name	Function	Reset Value
15:0	INPUT_LATCH	General-purpose inputs	Reflects input pins

Table 2–143. Output Register (OUTPUT_REG)

Bit	Name	Function	Reset Value
15:0	OUTPUT_REG	General-purpose outputs	Undefined

Table 2–144. Input/Output Control Register (IO_CNTL)

Bit	Name	Function	Reset Value
15:0	IO_CNTL	In/out control for general-purpose I/O 0: I/O is configured as output. 1: I/O is configured as input.	All bits at 1

Table 2–145. Keyboard Row Inputs Register (KBR_LATCH)

Bit	Name	Function	Reset Value
15:7	Reserved		
4:0	KBR_LATCH	Keyboard row inputs	Reflects input pins

Table 2–146. Keyboard Column Outputs Register (KBC_REG)

Bit	Name	Function	Reset Value
15:8	Reserved		
7:0	KBC_REG	Keyboard columns outputs	0

Table 2–147. GPIO Event Mode Register (GPIO_EVENT_MODE_REG)

Bit	Name	Function	Reset Value
15:5	Reserved		
4:1	PIN_SELECT	Select ARMIO_IN[4:0] pin to be the ARMIO_CLK event 0000: Pin 0 0001: Pin 1 0010: Pin 2 0011: Pin 3 0100: Pin4 Others: Reserved	0000
0	SET_GPIO_EVENT_MODE	0: GPIO event mode disable 1: GPIO event mode enable	0

Table 2–148. GPIO Interrupt Edge Register (GPIO_INT_EDGE_REG)

Bit	Name	Function	Reset Value
15:5	Reserved		0
4:0	EDGE_SELECT[4:0]	Set interrupt on falling/rising edge 0: Falling edge 1: Rising edge	0

Table 2–149. Keyboard Interrupt Register (KBD _INT)

Bit	Name	Function	Reset Value
15:1	Reserved		
0	KBD_INT	Keyboard interrupt (active high)	0

Note: KBD_INT is a status bit only (duplication of the level of the corresponding interrupt signal).

Table 2–150. GPIO Interrupt Register (GPIO_INT)

Bit	Name	Function	Reset Value
15:5	Reserved		0
4:0	GPIO_INT	GPIO interrupts (active high)	0

Note: GPIO_INT is reset on read access to the GPIO_INT register. The value read is the value after mask application.

Even in emulation mode, the GPIO interrupts are reset by a read in the GPIO interrupt register (GPIO_INT).

Table 2–151. Keyboard Mask Interrupt Register (KBD_ MASKIT)

Bit	Name	Function	Reset Value
15:1	Reserved		
0	KBD_MASKIT	Mask is active at level 1, inactive at level 0.	00

Table 2–152. GPIO Mask Interrupt Register (GPIO_MASKIT)

Bit	Name	Function	Reset Value
15:5	Reserved		0
4:0	GPIO_MASKIT	Mask is active at level 1, inactive at level 0.	00

Table 2–153. GPIO Debouncing Register (GPIO_DEBOUNCING_REG)

Bit	Name	Function	Reset Value
15:9	Reserved		
8:0	GPIO_DEBOUNCING_ REG	000000000: 0 μσ to 31 μs debouncing time 100000010: 7,97 ms to 8 ms debouncing time Programming step is 31 μs.	0000

Note: Because GPIO_CLK is an asynchronous signal, loading GPIO_DEBOUNCING_REG with 01 hex minimum value is recommended to ensure that you have a 31-μs minimum debouncing time. If the value is 00 hex, the interrupt may be generated immediately when an edge is met.

Table 2–154. GPIO Latch Register (GPIO_LATCH_REG)

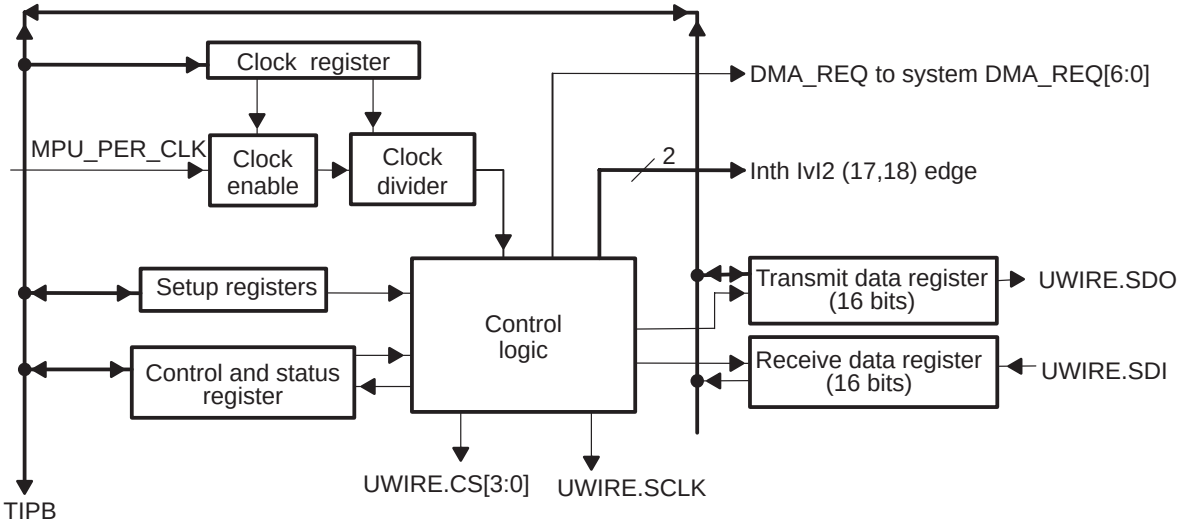
Bit	Name	Function	Reset Value
15:0	GPIO_LATCH_REG	After debouncing time, the ARMIO_IN bus is latched in this.	00

2.13 MicroWire Interface (µWire)

This serial synchronous interface can drive four serial external components. For the external devices, this interface is compatible with the µWire standard and is seen as the master (see Figure 2–80).

A transmit DMA mode is available.

Figure 2–80. Block Diagram



2.13.1 MicroWire Registers

Start address in the peripheral range (hex): FFFB:3000

Table 2–155 lists the 16-bit MicroWire registers. Table 2–156 through Table 2–163 describe the individual registers.

Table 2–155. MicroWire Registers

Register	Description	R/W	Offset
TDR	Transmit data	W	0x00
RDR	Receive data	R	0x00
CSR	Control and status	R/W	0x02
SR1	Setup 1	R/W	0x04
SR2	Setup 2	R/W	0x06
SR3	Setup 3	R/W	0x08
SR4	Setup 4	R/W	0x0A
SR5	Setup 5	R/W	0x0C

Table 2–156. Transmit Data Register (TDR)

Bit	Name	Function	Reset Value
15:0	TD	Data to transmit	Undefined

Note: MSB (bit 15) is the first transmitted bit.

Whatever the size of the data to be transmitted, the word must be aligned on the most significant bit (MSB) side.

Table 2–157. Receive Data Register (RDR)

Offset address (hex): 0x00

Bit	Name	Function	Reset Value
15:0	RD	Received data	Undefined

Note: LSB (bit 0) is the last received bit.

Whatever the size of the data received, the word is aligned on the least significant bit (LSB) side.

Table 2–158. Control and Status Register (CSR)

Bit	Name	Function	Reset Value
15	RDRB	RDRB bit at 1 indicates that the receive (RDR) is full. When the controller reads the content of the RDR, this bit is cleared. This bit is read only.	0
14	CSRB	CSRB bit at 0 indicates that the control and status (CSR) is ready to receive new data. This bit is set to 1 after writing either the CS_CMD or the START bits in the control and status register. CSRB is reset after the corresponding action has been done. This bit is read only.	0
13	START	1: Start a write and/or a read process. This bit is automatically reset by internal logic when a write or a read process is activated. Send NB_BITS_WR bits (contained in TDR) to the serial output DO. If NB_BITS_WR is equal to zero, then the write process is not started. Receive NB_BITS_RD bits from the serial input DI and store them in RDR.	0
12	CS_CMD	1: Set the chip-select of the selected device to its active level.	0
11:10	INDEX	Index of the external device 00: CS0 01: CS1 10: CS2 11: CS3	Undefined
9:5	NB_BITS_WR	Number of bits to transmit	Undefined
4:0	NB_BITS_RD	Number of bits to receive	Undefined

Table 2–159. Setup Register 1 (SR1)

Bit	Name	Function	Reset Value
15:12	Reserved	Reserved	Undefined
11	CS1_CHK	Idem CS0_CHK. Used when the CS1 is selected.	Undefined
10:9	CS1_FRQ	Defines the frequency of the serial clock SCLK when CS1 is selected 00: F_INT/2 01: F_INT/4 10: F_INT/8 11: Undefined	Undefined
8	CS1CS_LVL	Defines the active level of the CS1 chip select	0
7	CS1_EDGE_WR	Idem CS0_EDGE_WR when CS1 is selected	Undefined
6	CS1_EDGE_RD	Idem CS0_EDGE_RD when CS1 is selected	Undefined
5	CS0_CHK	Before activating a write process, checks if external device is ready. 0: No check is done and the write process is immediately executed. 1: If DI signal is low, the interface considers the external component busy; if DI is high, the interface considers that the first external component is ready and starts the write process. Used when CS0 is selected.	Undefined
4:3	CS0_FRQ	Defines the frequency of the serial clock SCLK when CS0 is selected (F_INT is the frequency of the internal clock). 00: F_INT/2 01: F_INT/4 10: F_INT/8 11: Undefined	Undefined
2	CS0CS_LVL	Defines the active level of the chip-select by CS0	0
1	CS0_EDGE_WR	When CS0 is selected, defines the active edge of the serial clock SCLK used to write data to the serial input D0. (Output data is generated on this edge) 0: Falling (serial clock not inverted) 0: Rising (when serial clock inverted) 1: Rising (serial clock not inverted) 1: Falling (when serial clock inverted)	Undefined
0	CS0_EDGE_RD	When CS0 is selected, defines the active edge of the serial clock SCLK used to read data from the serial input DI. (Input data is strobed on this edge.) 0: Falling (serial clock not inverted) 0: Rising (when serial clock inverted) 1: Rising (serial clock not inverted) 1: Falling (when serial clock inverted)	Undefined

Note: Content of this register must not be changed when a read or write process is running.

SR1 sets up the serial interface for the first and the second external components.

Table 2–160. Setup Register 2 (SR2)

Bit	Name	Function	Reset Value
15:12	Reserved	Reserved	Undefined
11	CS3_CHK	Same as CS0_CHK. Used when CS3 is selected.	Undefined
10:9	CS3_FRQ	Defines the frequency of the serial clock SCLK when CS3 is selected 00: F_INT/2 01: F_INT/4 10: F_INT/8 11: Undefined	Undefined
8	CS3CS_LVL	Defines the active level of the CS3 chip-select	0
7	CS3_EDGE_WR	Same as CS0_EDGE_WR when CS3 is selected	Undefined
6	CS3_EDGE_RD	Same as CS0_EDGE_RD when CS3 is selected	Undefined
5	CS2_CHK	Idem CS0_CHK when CS2 is selected	Undefined
4:3	CS2_FRQ	Defines the frequency of the serial clock SCLK when CS2 is selected 00: F_INT/2 01: F_INT/4 10: F_INT/8 11: Undefined	Undefined
2	CS2CS_LVL	Defines the active level of the CS2 chip-select	0
1	CS2_EDGE_WR	Idem CS0_EDGE_WR when CS2 is selected	Undefined
0	CS2_EDGE_RD	Idem CS0_EDGE_RD when CS2 is selected	Undefined

Note: Content of this register must not be changed when a read or write process is running.

SR2 sets up the serial interface for the first and the second external components.

Table 2–161. Setup Register 3 (SR3)

Bit	Name	Function	Reset Value
15:3	Reserved	Reserved	Undefined
2:1	CK_FREQ	Defines the frequency of the internal clock, F_INT, when CLK_EN = 1. All the internal logic is controlled by F_INT (F is the frequency of the external input clock). 00: F/2 01: F/4 10: F/7 11: F/10	00
0	CLK_EN	Switch off the clock if 0. Switch on the clock if 1.	0

Note: Content of this register must not be changed when a read or write process is running.

SR3 sets up the serial interface for the internal clock.

Table 2–162. Setup Register 4 (SR4) (Read/Write)

Bit	Name	Function	Reset Value
15:1	Reserved	Reserved	Undefined
0	CLK_IN	Serial clock is not inverted if 0.	0
		Serial clock is inverted if 1.	

Note: Content of this register must not be changed when a read or write process is running.

SR4 sets up the serial clock polarity.

Table 2–163. Setup Register 5 (SR5) (Read/Write)

Bit	Name	Function	Reset Value
15:4	Reserved	Reserved	Undefined
3	CS_TOGGLE_TX_EN	CS_TOGGLE_TX_EN is possible only in autotransmit mode.	0
		When in autotransmit mode with CS_TOGGLE_TX_EN inactive, the CS does not go to its active level automatically. Control the CS with the CS CMD bit of the control and status register (CSR) in the software.	
		CS_toggle transmit mode is disabled if 0.	
		CS_toggle transmit mode is enabled if 1.	
2	AUTO_TX_EN	In autotransmit mode, the CS_CMD and START bits of the control and status register (CSR) are not used. A hardware state machine detects a TXD write and automatically sets the programmed CS to its active value, then starts the transmission.	0
		The CS-CMD and the START bits in the control and status register (CSR) are not updated during autotransmit.	
		Autotransmit mode is disabled if 0.	
		Autotransmit mode is enabled if 1.	
1	IT_EN	In IT mode, an interrupt is generated each time a word has been transferred or a received. This interrupt is a low-level interrupt. A status register (IST) allows the CPU to know which interrupt (receive or/and transmit) occurred.	0
		IT mode is disabled if 0.	
		IT mode is enabled if 1.	
0	DMA_TX_EN	DMA transmit mode is disabled if 0.	0
		DMA transmit mode is enabled if 1.	

Note: Content of this register must not be changed when a read or write process is running.

Set up the DMA, IT, AUTO_TX, and CS_TOGGLE modes in SR5.

In DMA mode, a DMA request is initiated each time a transmission slot is available.

The maximum word size in DMA mode is 16 bits.

Note:

You cannot use another CS in normal or DMA mode when a DMA mode is active on one specific CS.

Note:

To use the µWire in DMA transmit mode, DMA_EN and AUTO_TX_EN must be enabled, and IT_EN is best disabled. The AUTO_TX_EN can be active when DMA_EN is disabled.

2.13.2 Protocol Description

The serial port must be configured in order to use the setup registers.

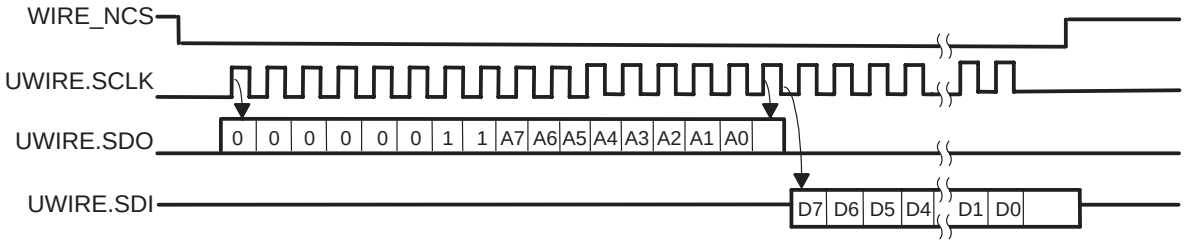
This interface can only drive one device at a given time. Therefore, the chip-select of the selected device must be set to its active level before starting any read or write process.

After the loading of the transmit data register (TDR), a write process is activated by setting the START bit to 1 and by writing a value different from zero to the NB_BITS_WR field.

A read process is always simultaneous with a write process, which means that at every serial clock (SCLK) cycle data is read. After having finished a write process (if necessary), a number (defined by NB_BITS_RD) of SCLK cycles is generated to allow storage of data from the serial input DI.

The transmitted data word is shifted out on the rising or falling edge of the serial clock (according to the value of the _$\overline{\text{EDGE_WR}}$ bits of the setup registers). The received data word is shifted in on the falling or rising edge of the serial clock (according to the value of the _$\overline{\text{EDGE_RD}}$ bits of the setup registers). When _$\overline{\text{EDGE_WR}}$ and _$\overline{\text{EDGE_RD}}$ bits have the same value, it is assumed that the device behavior is the one shown in Figure 2–81. Otherwise, the required behavior of the external device is shown in Figure 2–82.

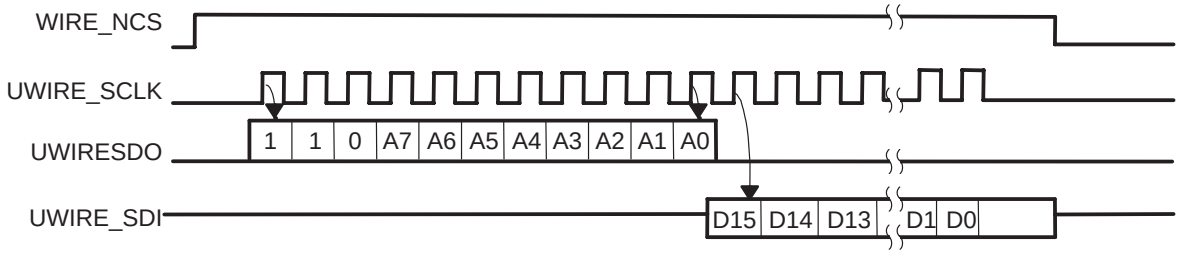
Figure 2–81. Behavior of a X25C02 EEPROM Read Cycle



On the DO line, data is generated from the µWire interface on SLCK falling edge and read by the EEPROM interface on SCLK rising edge.

On the DI line, data is generated from the EEPROM interface on SCLK falling edge and read by the µWire interface on SCLK falling edge.

Figure 2–82. Behavior of a XL93LC66 EEPROM Read Cycle



On the DO line, data is generated from the µWire interface on SLCK falling edge and read by the EEPROM interface on SCLK rising edge.

On the DI line, data is generated from the EEPROM interface on SCLK rising edge and read by the µWire interface on SCLK rising edge.

2.13.3 Example of Protocol Using a Serial EEPROM (XL93LC66)

Set up the interface by writing the following values in setup 1 register (SR1):

- CS_EDGE_RD = 1
- CS_EDGE_WR = 0
- CSCS_LVL = 1
- CS_FRQ = 00
- CS_CHK = 1

In this example, only two cycles (read and write) are described.

2.13.3.1 Read Cycle

1) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 0
   - NB_BITS_WR: 0
   - INDEX: 00
   - CS_CMD: 1
   - START: 0
2) Load the transmit data register (TDR) with:
   - 1 1 0 A7 A6 A5 A4 A3 A2 A1 A0 x x x x x x: *Don't care*
   - A7 ... A0: Address of the selected memory register
3) Wait for the CSRB bit of CSR to be reset.
4) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 16 (decimal)
   - NB_BITS_WR: 11 (decimal)
   - INDEX: 00
   - CS_CMD: 1
   - START: 1
5) Wait until CSRB = 0 and RDRB = 1 (status bits of CSR).
6) Read the content of receive data register (RDR).

7) To continue reading data external component, the EEPROM, go to 8. Else go to 9.

8) Set the following fields of the control and status register (CSR):

- NB_BITS_RD: 16 (decimal)
- NB_BITS_WR: 0 (decimal)
- INDEX: 00
- CS_CMD: 1
- START: 1
- Go to Step 5.

9) Set the following fields of the control and status register (CSR):

- INDEX: 00
- CS_CMD: 0
- START: 0

2.13.3.2 Write Cycle

1) Set the following fields of the control and status register (CSR):

- NB_BITS_RD: 0
- NB_BITS_WR: 0
- INDEX: 00
- CS_CMD: 1
- START: 0

2) Load the transmit data register (TDR) with:

- 1 0 1 A7 A6 A5 A4 A3 A2 A1 A0 x x x x x: *Don't care*
- A7 ... A0: Address of the selected memory register

3) Wait for the CSRB bit of the control and status register (CSR) to be reset.

4) Set the following fields of the control and status register (CSR):

- NB_BITS_RD: 0
- NB_BITS_WR: 11 (decimal)
- INDEX: 00
- CS_CMD: 1
- START: 1

5) Wait for the CSRB bit of the control and status register (CSR) to be reset.

6) Load the transmit data register (TDR) with:

- D15 D14 ... D0
- D15 ... D0: Data

7) Set the following fields of the control and status register (CSR):

- NB_BITS_RD: 0
- NB_BITS_WR: 16 (decimal)
- INDEX: 00
- CS_CMD 1
- START: 1

8) Wait for the CSRB bit of CSR to be reset.

9) Set the following fields of the control and status register (CSR):

- INDEX: 00
- CS_CMD: 0
- START: 0

2.13.4 Example of Protocol Using an LCD Controller (COP472-3)

Set up the interface by writing in setup 1 register (SR1) the following value:

- CS_EDGE_RD = 1
- CS_EDGE_WR = 0
- CSCS_LVL = 0
- CS_FRQ = 10
- CS_CHK = 0

In this example, a loading sequence to drive a four-digit display is described.

2.13.4.1 Loading Sequence

1) Set the following fields of the control and status register (CSR):

- NB_BITS_RD: 0
- NB_BITS_WR: 0
- INDEX: 01
- CS_CMD: 1
- START: 0

2) Wait for the CSRB bit of the control and status register (CSR) to be reset.

3) Load the transmit data register (TDR) with:

- D7d1...D0d1 D7d2...D0d2 D7d1...D0d1: Data for digit 1
- D7d2...D0d2: Data for digit 2

4) Set the following fields of the control and status register (CSR):

- NB_BITS_RD: 0
- NB_BITS_WR: 16 (decimal)
- INDEX: 01
- CS_CMD: 1
- START: 1

5) Wait for the CSRB bit of the control and status register (CSR) to be reset.

6) Load the transmit data register (TDR) with:

- D7d3...D0d3 D7d4...D0d4 D7d3...D0d3: Data for digit 3
- D7d4...D0d4: Data for digit 4

7) Set the following fields of the control and status register (CSR):

- NB_BITS_RD: 0
- NB_BITS_WR: 16 (decimal)
- INDEX: 01
- CS_CMD: 1
- START: 1

8) Wait for the CSRB bit of the control and status register (CSR) to be reset.

9) Load the transmit data register (TDR) with:

   - D7...D0 x x x x x x x x: *Don't care*
   - D7...D0: Data for special segment and control function

10) Set the following fields of the control and status register (CSR):

   - NB_BITS_RD: 0
   - NB_BITS_WR: 8 (decimal)
   - INDEX: 01
   - CS_CMD: 1
   - START: 1

11) Wait for CSRB to go low, which indicates the CSR is ready to receive new data. It is advised that you read the bit before and after every write access to CSR to check the status.

12) Set the following fields of the control and status register (CSR):

   - INDEX: 01
   - CS_CMD: 0
   - START: 0

2.13.5 Example of Protocol Using Autotransmit Mode

The autotransmit mode is controlled by the setup 5 register (SR5). The following example configures µWire for a read access on CS0 with serial clock out inverted, CS autotoggle enabled, DMA request disabled, and interrupt enabled:

1) SR5 = DMA_TX_EN: 0

   IT_EN: 1

   AUTO_TX_EN: 1

   CS_TOGGLE_TX_EN: 1

2) SR1 = CS0_EDGE_RD: 0

   CS0_EDGE_WR: 1

   CS0CS_LVL: 0

   CS0_FREQ: 00

   CS0_CHK: 1

Note:

Data out is latched on falling edge of the serial clock. Data in is sampled on rising edge.

3) SR3 = CLK_EN: 1

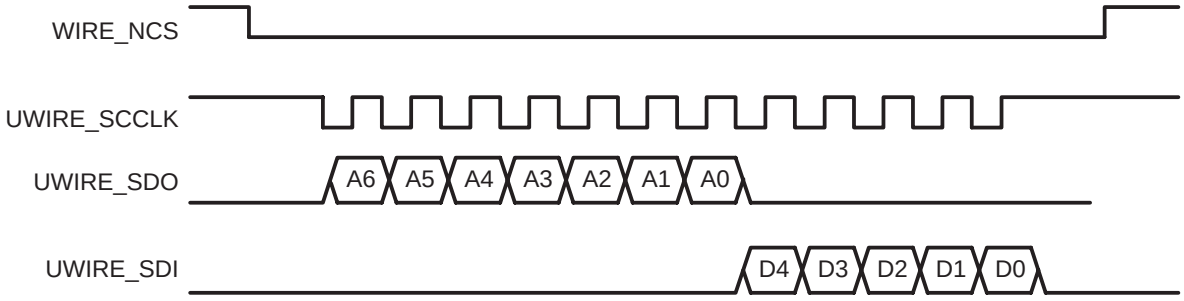
   CK_FREQ: 00 (must wait for 1 external clock + 1 F_INT cycle before any other register access)

4) SR4 = CLK_IN: 1
5) Set the following fields of the control and status register (CSR):
   - NB_BITS_RD: 5
   - NB_BITS_WR: 7
   - INDEX: 00
   - CS_CMD: 0
   - START: 0
6) Wait for the CSRB = 0 of the control and status register (CSR).
7) Load the transmit data register (TDR) with:
   - A6 A5 A4 A3 A2 A1 A0 x x x x x x x x x: *Don't care*
   - A6 ... A0: Address of the selected memory register

   Transfer is automatically started.
8) Wait until CSRB = 0 and RDRB = 1 (status bits of CSR).
9) Read the content of receive data register (RDR).
10) To continue reading data external component, go to 5 else go to 11.
11) Release auto transmit mode: SR5 = AUTO_TX_EN: 0.
12) END

The corresponding behavior of the serial interface is described in Figure 2–83.

Figure 2–83. Read Cycle in Autotransmit Mode



2.13.6 Example of Autotransmit Mode With DMA Support

The autotransmit mode and DMA mode are controlled by the setup 5 register (SR5). The following example configures μWire for a 16-bit write access on CS1 with serial clock out not inverted, CS auto toggle enabled, DMA request enabled, and interrupt disabled:

1) Set up and enable the DMA channel.
2) Program the configuration registers SR1, SR3, and SR4.
3) Check CSRB status to ensure that the peripheral is ready to receive (low).
4) Program the control and status register (CSR) as follows:
   - NB_BITS_RD = 0
   - NB_BITS_WR = 16
   - INDEX = 00
   - CS_CMD: = 1
   - START = 0

5) Write to the setup register (SR5) to configure and initiate the transfer:

   - DMA_TX_EN = 1
   - IT_EN = 0
   - AUTO_TX_EN = 1
   - CS_TOGGLE_TX_EN = 1 (Note that in AUTO TX mode, setting the DMA_TX_EN bit to 1 starts the transfer)

6) When the DMA transfer is complete, check the status of CSRB to find whether μWire has finished the serial data transfer.

7) Write to the setup register (SR5) to disable DMA and AUTO TX mode:

   - DMA_TX_EN = 0
   - IT_EN = 0
   - AUTO_TX_EN = 0
   - CS_TOGGLE_TX_EN = 0

Using Autostart and Autotoggle CS Mode

You must wait for a minimum of 2 x F_INT clock cycles after the end of transfer (transition 1 to 0 detected on CSRB) before setting the SR3 register to turn off the internal clock.

2.13.7 Limitations of μWIRE Interface

According to the configuration, Table 2–164 shows the limitations of μWIRE interface.

Table 2–164. Limitations of μWIRE Interface

	CSx_EDGE_RD = 0 CSx_EDGE_WR - 0	CSx_EDGE_RD = 1 CSx_EDGE_WR = 1	CSx_EDGE_RD = 0 CSx_EDGE_WR = 1	CSx_EDGE_RD = 1 CSx_EDGE_WR = 0
SC not inverted	No limitation	No limitation	Bit 0 of the receive data register (shift right one bit) need to be masked. **NOTE:** This configuration is not available with 1 bits data to receive.	Do not use this configuration. Use one of the alternate configurations available for this module.
SCLK inverted	No limitation	No limitation	Bit 0 of the receive data register (shift right one bit) need to be masked. **NOTE:** This configuration is not available with 16 bits data to receive.	Do not use this configuration. Use one of the alternate configurations available for this module.

2.14 HDQ and 1-Wire Protocols

This module implements the hardware protocol of the master function of the HDQ™ and 1-Wire™ protocols.

This module works off a command structure that is programmed into transmit command registers. The received data is in the receive data register. The firmware is responsible for performing correct sequencing in the command registers. The module only implements the hardware interface layer of the protocols.

The HDQ and the 1-Wire modes are selectable in software, and this must be done before any transmit and receive from the module is performed. The mode is assumed static during operation of the device. From a timing perspective, both the 1-Wire and the HDQ protocols use HDQ timing.

2.14.1 Functional Description

The module is intended to work with both the HDQ and the 1-Wire protocols. The protocols use a single wire to communicate between the master and the slave. The protocols employ a return-to-1 mechanism, where after any command the line is pulled up to a high. This necessitates an external pullup.

An open-drain configuration is used on the wire. The DX is connected to the GZ pin of an external 3-state output driver, and the input pin is connected to a zero level.

A control bit selects whether the HDQ or the 1-Wire protocol is to be used. Although this bit can be modified at any point, it is recommended that the bit be only modified as part of boot up configuration. For the design, the bit is assumed static. By default, the configuration complies with the HDQ spec.

2.14.1.1 Receive and Transmit Operation

The receive and the transmit operations are performed with respect to the timing that is specified in the HDQ protocol. This is done to keep the hardware interface section compatible between the two devices. In essence the 1-Wire mode runs at slower speeds than the capabilities of the mode. The differences between the protocol at the hardware layer are described in the following subsections.

HDQ Mode (Default)

In HDQ mode, the firmware does not require the host to create an initialization pulse to the slave. However, the slave can be reset using an initialization pulse (also referred to as a break pulse). The pulse is created by setting the appropriate bit in the control and status register. The slave does not respond with a presence pulse, as it does in the 1-Wire protocol.

In a typical write to the slave, two bytes of data are sent to the slave. This is the command/address byte followed by the data that must be written. In a typical read, one command/address byte is sent to the slave, and the slave returns a byte of data.

The master implementation is a byte engine. The sending of the ID, command/address, and data is the responsibility of the firmware. The master engine provides for only one data TX register only.

HDQ is a return-to-1 protocol. This means that after a data byte (either command/address + write data for writes, or just command/address for reads) is sent to the slave, the host pulls the line high. This is accomplished in the OMAP730 device by setting the line to high (with an external pullup). The slave pulls the line low to initiate a transaction. This is the case when a read happens and the slave must send the read data back to the host.

If the host initiates a read and data is not received in a specified interval (the slave does not pull the line low within this time), a time-out status bit is set. This indicates that a read was not successfully completed. On successful completion, the time-out bit is cleared. The bit remains set or cleared until the next transaction by the host.

An interrupt condition indicates either a TX complete, RX complete, or time-out condition. The read of the interrupt status register clears all the interrupt conditions. Only one interrupt signal is sent to the microcontroller and only an overall mask bit exists for the enabling and disabling of the interrupt. Each of the interrupt conditions cannot be individually masked.

The following sequence must be performed by the programmer for the reads and writes to the slave:

Write operation:

1) Write the command or data value to the TX write register.

2) Write 0 to the R/W bit of the control and status register to indicate a write.

3) Write 1 to the go bit of the control and status register to start the actual transmit. This step and the above step can be done at the same time.

    a) The hardware sends the byte from the TX data register.

    b) The time-out bit always is cleared in a write, because the hardware has no acknowledge mechanism from the slave.

    c) The completion of the operation sets the TX complete flag in the interrupt status register. If interrupts are masked, no interrupt is generated. The interrupt status register is always cleared at the beginning of any read or write operation.

    d) At the end of the write, the go bit is cleared.

4) Software must read the interrupt status register to clear the interrupt.

5) Repeat for each successive byte.

Read operation:

1) Write the command value to the TX write register.

2) Write 0 to R/W bit, 1 to the go bit, and wait for TX complete interrupt.

3) Write 1 to the R/W bit of the control and status register to indicate a read.

4) Write 1 to the go bit of the control and status register to start the actual read. This step and the above step can be done at the same time.

   a) The hardware detects a low-going edge of the line (created by the slave) and receives 8 bits of data in the RX receive buffer register. The first bit that is received from the slave is the LSB and the last bit is the MSB of the byte. The master performs this step as soon as the slave sends the data irrespective of the state of the go bit. However, an RX complete interrupt is generated only when the go bit is written by the software.

   b) If a time-out occurs, a time-out bit is set in the control and status register.

   c) The completion of the operation sets the RX complete flag in the interrupt status register. If interrupts are masked, no interrupt is generated. The interrupt status register is always cleared at the beginning of either a read or a write operation.

   d) At the end of the read, the go bit is cleared. It is also cleared if a time-out is detected.

5) Software must read the interrupt status register, to determine if RX was complete or whether there was a time-out.

6) Software does a read of the RX buffer register to retrieve the read data from slave.

7) Repeat for each successive byte.

In HDQ16 mode, the address/command is only written once to the slave. However, after the first byte is received, if an RX complete interrupt is received, the software must initiate the read of the second byte by writing the go bit of the control and status register. The first byte that was received is shadowed and provided to the software, while the hardware is fetching the second byte of data.

1-Wire Mode (SDQ)

This section highlights the primary differences between the HDQ and the 1-Wire protocols.

In the 1-Wire mode, the firmware must send an initialization pulse to the multiple slaves that can be connected on the interface. If any slave is present, the slave responds with a presence pulse.

The initialization pulse is sent by setting the appropriate bit in the control and status register. A presence detect is indicated in the appropriate bit of the register. If no presence is received, then a time-out bit is set in the status register. The initialization bit is cleared at the end of the initialization pulse. Also, the presence detect and the time-out bits are cleared at the end of the initialization pulse, if a presence detect is received. The time-out bit has no other signifi-

cance in this mode; that is, unlike in HDQ mode, it is always cleared during a read operation.

1-Wire mode is a bit-by-bit protocol for a read. Unlike HDQ, which sends eight bits of data on a read, the slave must be clocked by the host in 1-Wire protocol for each bit. At the end of the command/address byte, the line is pulled high and the host creates a low-going edge to initiate a bit read from the slave. The host then pulls the line high, and the slave either pulls the line low to indicate a 0 or does not drive the line to indicate a 1. The host repeats the operation for the next bit that need to be read.

The first bit that is received is the LSB and the last bit is the MSB in the RX data register.

An interrupt condition indicates either a TX complete, RX complete, or time-out condition. The read of the interrupt status register clears all the interrupt conditions. Only one interrupt signal is sent to the microcontroller and only an overall mask bit exists for the enabling and disabling of the interrupt. Each of the interrupt conditions cannot be individually masked.

The following sequence must be performed by the programmer for the reads and writes to the slave:

Write operation:

1) Write the ID, command, or data value to the TX write register.
2) Write 0 to the R/W bit of the control and status register to indicate a write.
3) Write 1 to the go bit of the control and status register to start the actual transmit. This step and the above step can be done at the same time.
    a) The hardware sends the one byte of the TX write data register.
    b) The time-out bit is always cleared in a write.
    c) The completion of the operation sets the TX complete flag in the interrupt status register. If interrupts are masked, no interrupt is generated. The interrupt status register is always cleared at the beginning of any read or write operation.
    d) At the end of the write the go bit is cleared.
4) If interrupt is enabled, software must read the interrupt status register to clear the interrupt.
5) Repeat for each successive byte.

Read operation:

1) Write the ID value to the TX write register.
2) Write 0 to R/W bit and 1 to the go bit and wait for TX complete interrupt.
3) Write the command value to the TX write register.
4) Write 0 to R/W bit and 1 to the GO bit and wait for TX complete interrupt.

5) Write 1 to the R/W bit of the control and status register to indicate a read.
6) Write 1 to the go bit of the control and status register to start the actual read. This step and the above step can be done at the same time.
   a) The hardware creates a low-going edge of the line (created by the slave), and clocks 8 bits of data into the RX receive buffer register. The first bit that is received from the slave is the LSB and the last bit is the MSB of the byte.
   b) The time-out bit is always cleared in a read.
   c) The completion of the operation sets the RX complete flag in the interrupt status register. If interrupts are masked, no interrupt is generated. The interrupt status register is always cleared at the beginning of any read or write operation.
   d) At the end of the read, the go bit is cleared. It is also cleared if a time-out is detected.
7) If interrupt is enabled, software must read the interrupt status register to determine if RX was completed or whether there was a time-out.
8) Software does a read of the RX buffer register to retrieve the read data from slave.
9) Repeat for each successive byte.

1-Wire Bit Mode Operation

A single-bit mode can be entered by writing to the appropriate bit in the control and status register. In this mode, only one bit of data is received each time from the slave. After the bit is received, an RX complete interrupt is generated. Bit 0 of the receive buffer is updated each time a bit is received.

The mode has no effect in HDQ mode, as HDQ does not support single-bit protocol.

2.14.1.2 Timing Diagrams

Figure 2–84 shows the timing diagram for the read, reset, and write. In the HDQ, the reset pulse only contains the initialization and not the presence pulse. The timing required for the various signals are specified in the *BQ2023* spec.

The master works at the timing of the HDQ interface, which encompasses the HDQ and the 1-Wire timing. Therefore, in 1-Wire mode, the master runs slower than the full performance capability of the protocol.

Figure 2–84. Read Timing Diagram

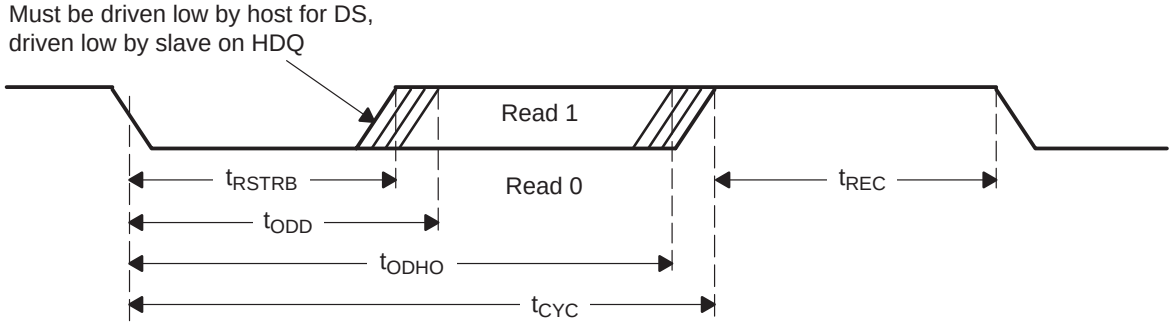


Figure 2–85. Reset Timing Diagram

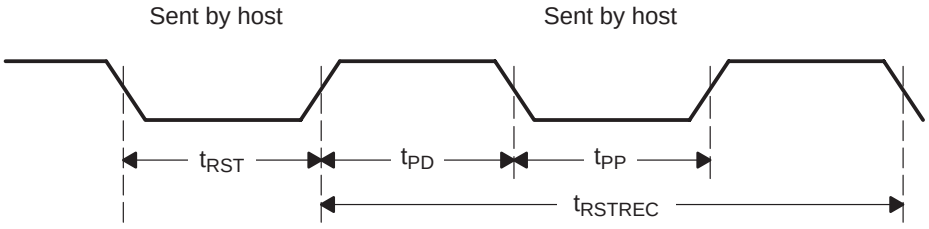
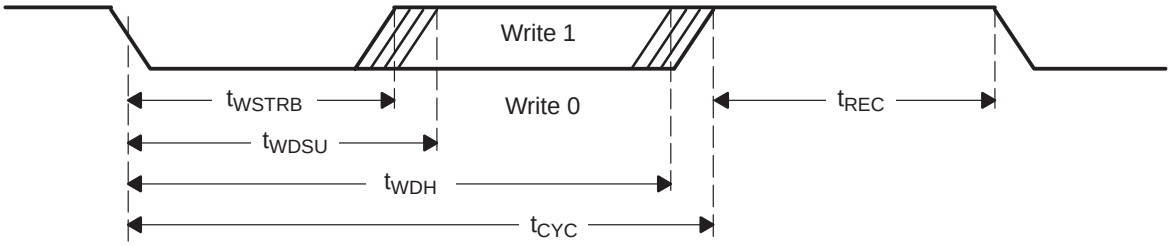
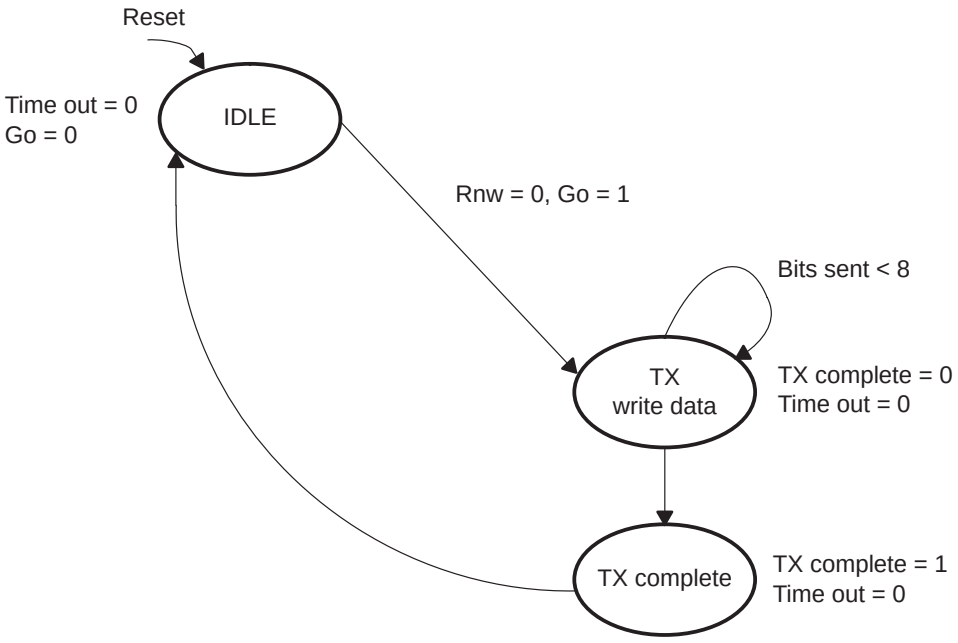


Figure 2–86. Write Timing Diagram



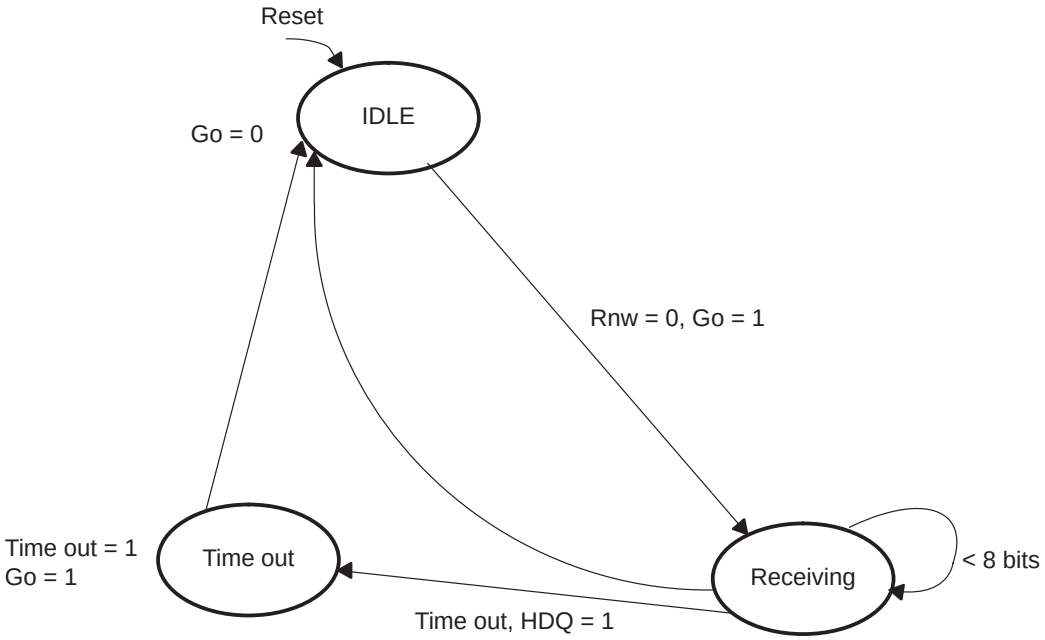
2.14.1.3 Write State Diagram

Figure 2–87. Write State Machine #1



2.14.1.4 Read State Diagram

Figure 2–88. Read State Machine #1



2.14.1.5 Status Flags

The status flags are provided in the status register, which contains status flags from the transmitter, the receiver, and the presence detect logic.

The presence condition detected status flag is contained in the status register. This is valid only in 1-Wire mode. It is cleared when the host sends an initialization pulse and then is set to 1 if a pulse is received; otherwise it stays cleared at 0.

2.14.1.6 Interrupts

The following interrupt status is provided by the module:

- Transmitter complete

  A write of one byte was completed. Successful or unsuccessful completion is not indicated, because there is no acknowledge from the slave in either HDQ or 1-Wire mode. Cleared at beginning of write command.

- Read complete

  Indicates successful completion of a byte read in both modes. Cleared at beginning of read command.

- Presence detect/time-out
  - In 1-Wire mode, it indicates that it is now valid to check the presence detect received bit. Cleared at beginning of initialization sequence.
  - In HDQ mode, it indicates that after a read command was issued by the host, the slave did not pull the line low within specified time. In HDQ mode, bit is cleared at beginning of read command.

Only one interrupt is generated to the microcontroller, based on any of the above interrupt status conditions. A read to the interrupt status register clears all the status bits that have been set.

The interrupt can be masked by setting the appropriate bit in the control and status register.

A read of the interrupt status register clears the interrupt. If there is a pending interrupt the interrupt line stays low and no low-high-low transition is created. The interrupt therefore must be handled as a level interrupt (where a low-going edge is not needed) in an upstream interrupt handler (or processor).

2.14.2 Power-Down Mode

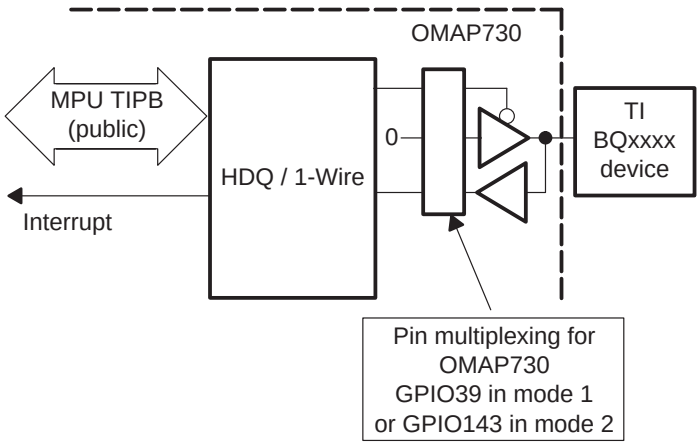
Writing to the appropriate bit in the control and status register shuts the clock to the state machine. The state machines are reset when the clock is disabled, and if any transaction is being performed it is aborted into the reset state. The register values are not affected by disabling the clock. No register access must be performed to the module registers after the software puts the module in power-down mode (by setting bit 5 of the control and status register to 0) other than a write to the power-down bit to take it out of power-down mode.

2.14.3 HDQ and 1-Wire Battery Monitoring Serial Interface

The HDQ and 1-Wire battery monitoring serial interface module implements the hardware protocol of the master function of the Benchmarq HDQ and the Dallas Semiconductor 1-Wire protocol. The module works off a command structure that is programmed into transmit command registers. The received data is in the received data register. The firmware is responsible for doing the correct sequencing in the command registers. The module only implements the hardware interface layer of the protocols.

The HDQ and the 1-Wire mode are selectable in software, which must be done before any transmit and receive from the module is performed. The mode is assumed static during operation of the device.

Figure 2–89. HDQ and 1-Wire Overview



2.14.4 Software Interface

The mapping of registers to the TI peripheral bus (TIPB) address signals is shown in Table 2–165 and Table 2–166. The OMAP730 memory map identifies the 2K space associated with the peripheral. All addresses here are offsets written to the 2K space.

No synchronization is provided by the hardware between the register clock domain and the state machine domain. This means that during a read the hardware has the capability to modify the receive buffer and it is also possible that any access to the transmit write data register corrupts the data that is being sent if a TX is being performed.

However, these hazards can be avoided in software by observing the following limitations:

- A read is not performed from the interrupt status register or receive buffer register unless the processor has been interrupted by the peripheral.
- After the release of the go bit in the control and status register, no access to the TX write data buffer or the control and status registers is performed until the processor has been interrupted by the peripheral.
- Polling of the interrupt status register is not allowed by software to determine if an interrupt was generated.
- No register access can be done to the module registers after the software puts the module in power-down mode (by setting bit 5 of the control and status register to 0), except to reenable the clock.

Table 2–165. Memory Map Summary

Address	Name	Type
0x00	TX write data	R/W
0x04	RX receive buffer	R
0x08	Control and status	R/W
0x0C	Interrupt status, read to clear	R/W
Other	Writes ignored; reads return 0	Reserved

Table 2–166. Registers Accessible From TIPB

Bit	Name	Description	Access Type	Reset Value
31:24	TX write data	Reserved. Read aliased to bits 7:0, writes ignored.	Read/Write at 8h00	0x00000000
23:16		Reserved. Read aliased to bits 7:0, writes ignored.		
15:8		Reserved. Read aliased to bits 7:0, writes ignored.		
7:0		Write data (Used in both HDQ and 1-Wire modes)		

Table 2–166. Registers Accessible From TIPB (Continued)

Bit	Name	Description	Access Type	Reset Value
31:24	RX buffer register	Reserved. Read aliased to bits 7:0, writes ignored.	Read only	Unknown (read only when data is ready)
23:16		Reserved. Read aliased to bits 7:0, writes ignored.		
15:8		Reserved. Read aliased to bits 7:0, writes ignored.		
7:0		Next received character		
31:24	Interrupt status register	Bit is set to 1 if cause of interrupt. Read clears all interrupts that have been set. Reserved. Read aliased to bits 7:0, writes ignored.	Read only/ read to clear	
23:16		Reserved. Read aliased to bits 7:0, writes ignored.		
15:8		Reserved. Read aliased to bits 7:0, writes ignored.		
7:3		Reserved. Read 0s, writes ignored.		0
2		TX completed		0
1		Read complete		0
0		Presence detect/time-out: In 1-Wire mode this is due to presence detect, and in HDQ mode this is due to time-out on read.		0

Table 2–166. Registers Accessible From TIPB (Continued)

Bit	Name	Description	Access Type	Reset Value
31:24	Control and status register	Reserved.Read aliased to bits 7:0, writes ignored.		
23:16		Reserved. Read aliased to bits 7:0, writes ignored.		
15:8		Reserved. Read aliased to bits 7:0, writes ignored.		
7		Single-bit mode for 1-Wire	R/W	0
6		Interrupt mask (1= enable, 0 =disable interrupts)	R/W	0
5		Power-down mode (1= enable clocks, 0 = disable clocks)	R/W	0
4		Go bit Write 1 to send the appropriate commands. Bit returns to 0 after the command is complete.	R/W	0
3		Presence detect received, 1-Wire mode only. 0: Not detected 1: Detected	R	0
2		Write 1 to send initialization pulse. Bit returns to 0 after pulse is sent.	R/W	0
1		R/W bit (determines if next command is read or write) 0: Write 1: Read	R/W	0
0		Mode 0: HDQ 1: 1-Wire	R/W	0

2.14.5 HDQ/1Wire Registers

Table 2–167. HDQ/1-Wire Registers (FFFB:7000)

Register	Offset	Register	Offset
TX_DATA	0x00	RX_RECEIVE_BUFFER_REG	0x04
CNTL_STATUS_REG	0x08	INTERRUPT_STATUS	0x0C

Table 2–168. TX_DATA

Bit	Name	Function	R/W	Reset Value
31:8	RESERVED	Reserved	R	0x0000
7:0	WRITE_DATA	Byte of space to be used as Write Data by both of the 1-Wire/HDQ modes	R/W	0x00

Table 2–169. RX_RECEIVE_BUFFER_REG

Bit	Name	Function	R/W	Reset Value
31:8	RESERVED	Read zeros.	R	0x0
7:0	RECEIVED_CHARACTER	The data byte that is received is stored in the lower 8 bits of the register.	R	0x00

Hardware detects a low-going edge of the line (created by the slave) and receives 8 bits of the data in the receive buffer register. The register is read-only.

Table 2–170. CNTL_STATUS_REG

Bit	Name	Function	R/W	Reset Value
31:8	RESERVED	Reserved	R	0x0000
7	SINGLE_BIT	Single-bit mode for 1-Wire	R	0x0
6	INTERRUPT_MASK	Interrupt mask (1= enable; 0= disable interrupts)	R/W	0x0
5	POWER_MODE	1: Enable clocks 0: Disable clocks	R/W	0x0
4	GO_BIT	Write 1 to send the appropriate commands. Bit returns 0 after the command is complete.	R/W	0x0
3	PRESENCE_DETECT	Presence detect received; 1-Wire mode only. 1: Detected 0: Not detected	R	0x0
2	INIT_PULSE	Write 1 to send initialization pulse. Bit returns to 0 after the pulse is sent.	R/W	0x0

Table 2–170. CNTL_STATUS_REG (Continued)

Bit	Name	Function	R/W	Reset Value
1	READ_WRITE_BIT	Determines if the next command is read or write. 1: Read 0: Write	R/W	0x0
0	MODE_BIT	Mode is selected using this bit, either the 1Wire or the HDQ. 0: HDQ mode 1: 1–Wire mode	R/W	0x0

Bits present can be used to turn ON various operations such as interrupt masking, initiation of the read/write transfer using the Go bit, presence detect, initialization pulse, R/W bit, and mode of operation.

Table 2–171. INTERRUPT_STATUS

Bit	Name	Function	R/W	Reset Value
31:3	RESERVED	Reserved	R	0x0000
2	TX_COMPLETE	Indicates the TX is complete Is set to 1 to indicate the interrupt	R	0x0
1	READ_COMPLETE	Indicates the receive operation is complete Bit is set to 1 if cause of interrupt.	R	0x0
0	PRESENCE_TIMEOUT	In 1-Wire mode, this is due to the presence detect; in HDQ mode, this is due to time-out on read. Bit is set to 1 if cause of interrupt.	R	0x0

Bit is set to 1 if cause of interrupt. Read of the register clears all interrupts that have been set.

2.15 Pulse-Width Tone Modulator

This pulse-width tone (PWT) module generates a modulated frequency signal for the external buzzer. The frequency is programmable between 322 Hz and 4868 Hz with 12 half-tone frequencies per octave. The volume level is also programmable. All frequencies are generated from the 13-MHz PWT_CLK clock.

2.15.1 Overview

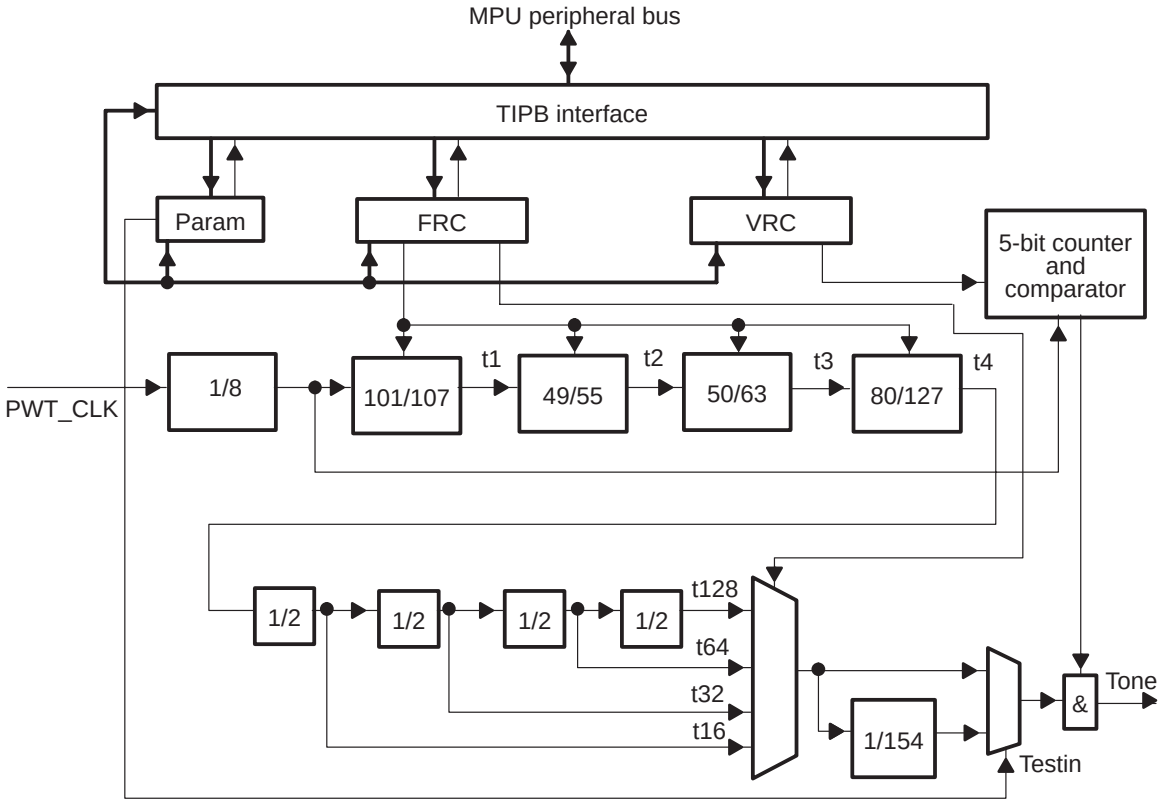
The PWT module creates the output tone signal for a buzzer. The frequency and volume of this signal are programmable.

2.15.2 PWT Features

The PWT module has the following features (see Figure 2–90):

- ❑ Divider generating a 1500-kHz frequency clock
- ❑ TIPB control interface
- ❑ Four dividers with $^{101}/_{107}$, $^{49}/_{55}$, $^{50}/_{63,}$ and $^{80}/_{127}$ to generate each note characteristics
- ❑ Four 1/2 dividers and a mux to select the octave
- ❑ 6-bit register to control tone frequency
- ❑ 6-bit register to control tone volume
- ❑ 2-bit register for test and CLK_EN
- ❑ 5-bit counter and comparator for creating volume pulse
- ❑ Divide by 154 to obtain the correct final frequency.

Figure 2–90. PWT Block Diagram



2.15.3 PWT Registers

Start address (hex): FFFB6000

Table 2–172 lists the 8-bit PWT registers. Table 2–173 through Table 2–175 describe the individual register bits.

Table 2–172. PWT Registers

Register	Description	Offset
FRC	PWT frequency control	0x00
VRC	PWT volume control	0x04
GCR	PWT general control	0x08

Table 2–173. PWT Frequency Control Register (FRC)

Bit	Name	Function	Reset Value
7:6	Reserved	Reserved	0000
5:2	FRQ	Frequency selection (12 frequencies) resynchronized writing, asynchronous reading	0000
1:0	OCT	Octave selection resynchronized writing, asynchronous reading	00

Table 2–174. PWT Volume Control Register (VRC)

Bit	Name	Function	Reset Value
7:2	Reserved	Reserved	000000
1	VOL	Volume selection Resynchronized writing, asynchronous reading	000000
0	ONOFF	Switch ON/OFF tone (on: 1, off: 0). Resynchronized writing, asynchronous reading	0

Table 2–175. PWT General Control Register (GCR)

Bit	Name	Function	Reset Value
7:2	Reserved	Reserved	0
1	TESTIN	Divider 1/154 switched ON/OFF (on: 0, off: 1). Asynchronous writing and reading	0
0	CLK_EN	PWT clock enable (clock disabled: 0, clock enabled: 1). Asynchronous writing and reading	0

2.15.4 PWT Programming

2.15.4.1 Buzzer Frequency

To obtain the required frequencies, the PWT clock is divided in a special way. Four frequency dividers with $^{101}/_{107}$, $^{49}/_{55}$, $^{50}/_{63,}$ and $^{80}/_{127}$ coefficients are connected in a series and can be enabled with the four-frequency selection bits (FRQ) in the frequency register. If a divider is not enabled, the clock passes through the divider without any change so different frequencies can be produced. After this a multiplexer can choose from this clock divided by 2, 4, 8, or 16. The frequency is always halved (this unit is called an octave). This gives the PWT a range of four octaves.

The clock behind the multiplexer is divided by 154 to get the required frequencies on the TONE output. The 12 frequencies in an octave can be programmed with bits 5 to 2 of the frequency control register (FRC), and the octave can be programmed with bits 1 and 0 of the FRC. Forty-eight different frequencies can be programmed and are subdivided into four octaves with twelve frequencies. The four frequency dividers with complex coefficients $^{101}/_{107}$, $^{49}/_{55}$, $^{50}/_{63,}$ and $^{80}/_{127}$ work with the fade-out principle. To get the divider $^{101}/_{107}$ from a periodic pulse, six pulses every 107 pulses are fade out. Over a long length of time the resulting frequency is $^{101}/_{107}$. The resulting signal has two different periods which differ by one period of the original signal. Because of this difference, the resulting signal has jitter. To minimize this jitter, the divider works with high frequencies resulting in short periods producing low jitter (see Table 2–176).

Table 2–176. Buzzer Frequencies

FRC Bits 5-2 1-0	Buzzer Frequency	FRC Bits 5-2 1-0	Buzzer Frequency
0000 00	4868 Hz e5	0000 10	1217 Hz e3
0001 00	4595 Hz dis5	0001 10	1149 Hz dis3
0010 00	4337 Hz d5	0010 10	1084 Hz d3
0011 00	4093 Hz cis5	0011 10	1023 Hz cis3
0100 00	3864 Hz c5	0100 10	966 Hz c3
0101 00	3647 Hz h4	0101 10	912 Hz h2
0110 00	3442 Hz ais4	0110 10	860 Hz ais2
0111 00	3249 Hz a4	0111 10	812 Hz a2
1000 00	3066 Hz gis4	1000 10	767 Hz gis2
1001 00	2894 Hz g4	1001 10	723 Hz g2
1010 00	2732 Hz fis4	1010 10	683 Hz fis2
1011 00	2579 Hz f4	1011 10	644 Hz f2
0000 01	2434 Hz e4	0000 11	608 Hz e2
0001 01	2297 Hz dis4	0001 11	574 Hz dis2
0010 01	2168 Hz d4	0010 11	541 Hz d2
0011 01	2046 Hz cis4	0011 11	511 Hz cis2
0100 01	1932 Hz c4	0100 11	483 Hz c2
0101 01	1824 Hz h3	0101 11	456 Hz h1
0110 01	1721 Hz ais3	0110 11	430 Hz ais1
0111 01	1624 Hz a3	0111 11	406 Hz a1
1000 01	1533 Hz gis3	1000 11	383 Hz gis1
1001 01	1447 Hz g3	1001 11	361 Hz g1
1010 01	1366 Hz fis3	1010 11	341 Hz fis1
1011 01	1289 Hz f3	1011 11	322 Hz f1
		11XX XX	Not allowed

2.15.4.2 Buzzer Volume

The buzzer volume can be programmed (see Table 2–177) with bits 6 to 1 in volume control register VRC. The higher the 6-bit value is, the louder is the buzzer/loudspeaker. To perform this programming, a 6-bit binary counter is clocked with the PWT clock and rolls over to 0h after reaching its terminal value (3 Fh). The counter value is compared with the 6-bit value programmed in VRC. If the count value is less than or equal to VRC, the output has the value H, else L:

- Y = VOL value: 0 = < y < 64
- PWT_CLK = 13 MHz
- Output signal H period = (y+1) × PWT_CLK
- Output signal L period = (63-y) × PWT_CLK

Table 2–177. Buzzer Volume

VRC Bits 6-1 ; 0	Buzzer/Loudspeaker
111111 1	Loud
000000 1	Quiet
xxxxx 0	Off

2.16 Pseudonoise Pulse-Width Light Modulator

This pulse-width light (PWL) module provides control of the LCD backlighting and keypad by employing a 4096-bit random-sequence generator. This voltage-level control technique decreases the spectral power at the modulator harmonic frequencies. This module uses a 32-kHz clock from ULPD.

2.16.1 PWL Functional Description

The PWL module is composed of a pseudorandom 8-bit data generator and a programmable threshold comparator (see Figure 2–91).

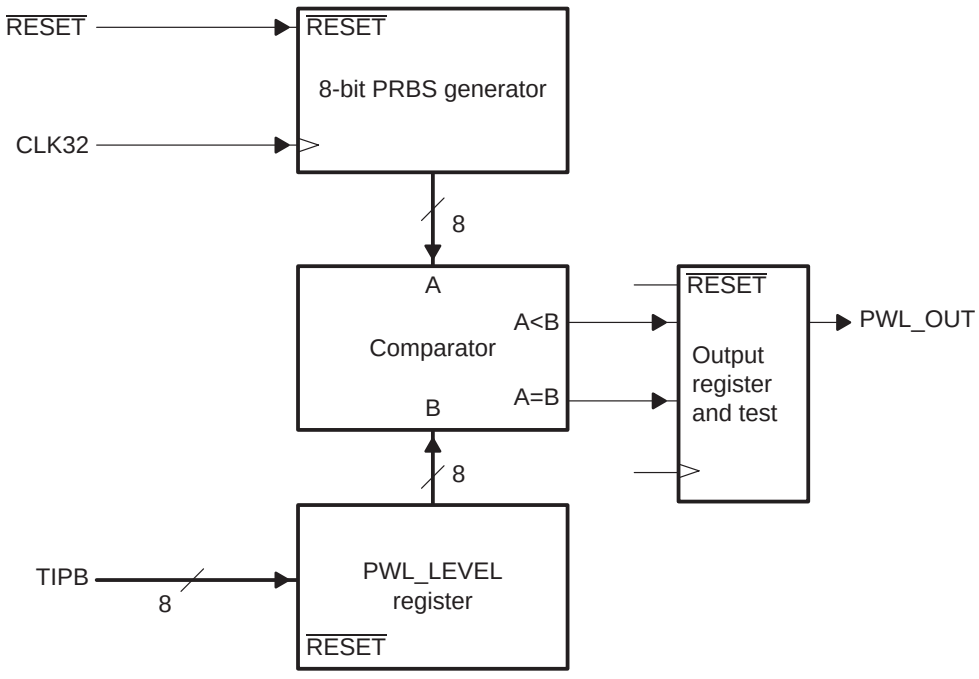
The pseudorandom 8-bit data generator is built using an LFSR. It generates a white normal-law random value between 1 and 255. The LFSR polynomial generator is $P(x) = x^7 + x^3 + x^2 + x + 1$.

The comparator generates:

- 0 if the random value is greater than or equal to the programmable threshold
- 1 if the random value is less than the programmable threshold

Assuming the random sequence is normal, it generates a sequence whose mean value is proportional to the comparator threshold.

Figure 2–91. PWL Block Diagram



2.16.2 PWL Registers

The PWL is connected to the host with a TIPB. The PWL control is done with two 8-bit registers. All TIPB accesses are asynchronous to the 32-kHz clock, meaning there is no TIPB wait-state insertion.

Table 2–178 lists the 8-bit PWL registers. Table 2–179 through Table 2–180 describe the individual register bits.

Start address (hex): FFFB:5800

Table 2–178. PWL Registers

Name	Description	Offset
PWL_LEVEL	PWL-level	0x00
CLK_ENABLE	Clock enable	0x04

Table 2–179. PWL Level Register (PWL_LEVEL)

Bit	Name	Function	Reset Value
7:0	PWL_LEVEL	Defines the mean value of the PWL output signal. 0 leads to a continuous 0 output. 255 to an almost continuous 1 output: 255/256 cycles in high level.	0

Table 2–180. PWL Control Register (PWL_CTRL)

Bit	Name	Function	Reset Value
7:1	–	Reserved	
0	CLK_ENABLE	Internal clock is enabled when 1.	0

2.17 32-kHz Timer

The MPU subsystem operating system (OS) requires interrupts at regular time intervals for OS scheduling purpose. OS time intervals can be from 1 ms to 30 ms. These time intervals can be generated using the three 32-bit OS/general-purpose TI925T timers, which use CLKIN or DPLL1; however, they can not be used when the system clock is not operating. Therefore, a 32-kHz clock-based timer is needed to provide the required OS timing interval. The clock period of 32 kHz is 30.60 µs. 32 kHz refers to 32678, not 32000.

2.17.1 Operating System Scalable Clock-Tick Interrupt Function

A programmable interval timer is required to generate a periodic interrupt, also called system clock tick, to the OS. This is used to keep track of the current time to control the operation of device drivers.

For example, Microsoft Windows CE OS scheduling requires the following:

- ❏ The periodic interrupt occurs every 1-25 ms.
- ❏ The timer is expected to run in all modes except when suspended.

32-kHz timer is a 24-bit down-counter that generates CPU interrupts for the TI925T processor. The following capabilities are available:

- ❏ Reset the timer
- ❏ Read current value of the timer
- ❏ Start and stop the timer
- ❏ Generate interrupt as the timer down-counts to zero
- ❏ Autorestart the timer after it counts to zero
- ❏ On the fly register read and write
- ❏ Disable the interrupt by writing a 1 to the interrupt bit in the control register

Timer	Corresponding Level 2 Interrupt
32-kHz timer	IRQ_22

The tick value register (TVR) contains the desired value for the timer to count down. The tick counter register (TCR) is loaded with this value, then starts to count down to zero and generates a negative edge sensitive interrupt (low-level pulse duration = 15 µs) to the interrupt handler. Once the interrupt is back to the high level, the counter (TCR) is reloaded from its register (TVR) and then starts to count down again.

2.17.1.1 Overriding Normal Counting

Normal operation can be overridden by using two bits in the timer control register (TCR):

- The timer reload bit (TRB) causes the counter to be reloaded on the next clk32-kHz cycle (whether or not the timer is counting).
- The timer start stop bit (TSS) causes the counter to be stopped on the next clk32-kHz cycle. When the timer is stopped, the content of the counter is frozen.

2.17.1.2 Loading/Autorestart of the Timer

Loading the counter in the timer can be done in two ways:

- Write a 1 to the TRB bit in the timer control register (TCR).
- Wait until the counter reaches zero and is reloaded from its register if the autorestart bit (ARL) in the timer control register (TCR) is set to 1. If not, then the timer is stopped.

2.17.1.3 Timer Interrupt Period

The timer interrupt period is defined by the value loaded into the tick value register (TVR).

The timer interrupt rate is as follows:

IRQ rate =(TVR+1) / 32768

Table 2–181. Timer Interrupt Period

TVR Value	Interrupt Period
0x000000	30.5 µs
0x00028F	19.9 ms
0xFFFFFF (Value at reset)	512 s (8 mn 32 s)

2.17.2 32-kHz Timer Registers

Base address for 32-kHz timer: FFFB:9000

Table 2–182 lists the 32-bit 32-kHz timer registers. Table 2–184 through Table 2–186 describe the individual registers.

Table 2–182. 32-kHz Timer Registers

Name	Description	R/W	Offset
CR	Timer control	R/W	0x08
TVR	Tick value	R/W	0x00
TCR	Tick counter	R	0x04

2.17.2.1 Synchronization Issues

Synchronization of reads and writes to the 32-kHz clock is done in different ways for each register. This leads to slight restrictions concerning register access (see Table 2–183).

Table 2–183. Read/Write Synchronization

Register Name	Read	Write
CR	Can be read anytime. The value read is the last value written.	Two consecutive writes must be separated by at least 1 CLK32 period (31 µs). If this is not the case, the value written is not ensured.
TCR	Reads are resynchronized on ARMXOR_CK clock to prevent peripheral bus from timing out. Can be read anytime, providing ARMXOR_CK is running. If not, the value is not ensured.	Writing to this has no effect.
TVR	Can be read anytime. The value read is the last value written.	Two consecutive writes must be separated by at least 1 CLK32 period (31 µs). If this is not the case the value written is not ensured.

Table 2–184. Timer Control Register (CR)

Bit	Name	Function	Reset Value
31:4	Reserved		
3		Autoreload/start	1
	ARL	1: Sets the timer to autorestart mode	
		0: One-shot mode. When the counter reaches zero, an interrupt is generated and the timer is stopped.	
2	IT_ENA	Interrupt enable	0
		0: Interrupt disabled	
		1: Interrupt enabled	
1	TRB	Timer reload bit	0
		TRB = 1 reloads the counter. Once the counter is reloaded, TRB is set to 0.	
0	TSS	Timer start/stop	0
		0: Stop timer	
		1: Start timer	
		If one-shot mode is selected (ARL = 0), this bit is automatically reset by internal logic when timer is equal to 0.	

Table 2–185. Tick Value Register (TVR)

Bit	Name	Function	Reset Value
31:24	Reserved		
23:0	TICK_VALUE_REG	This value is loaded when timer passes through 0 or when it starts.	0xFFFFFF

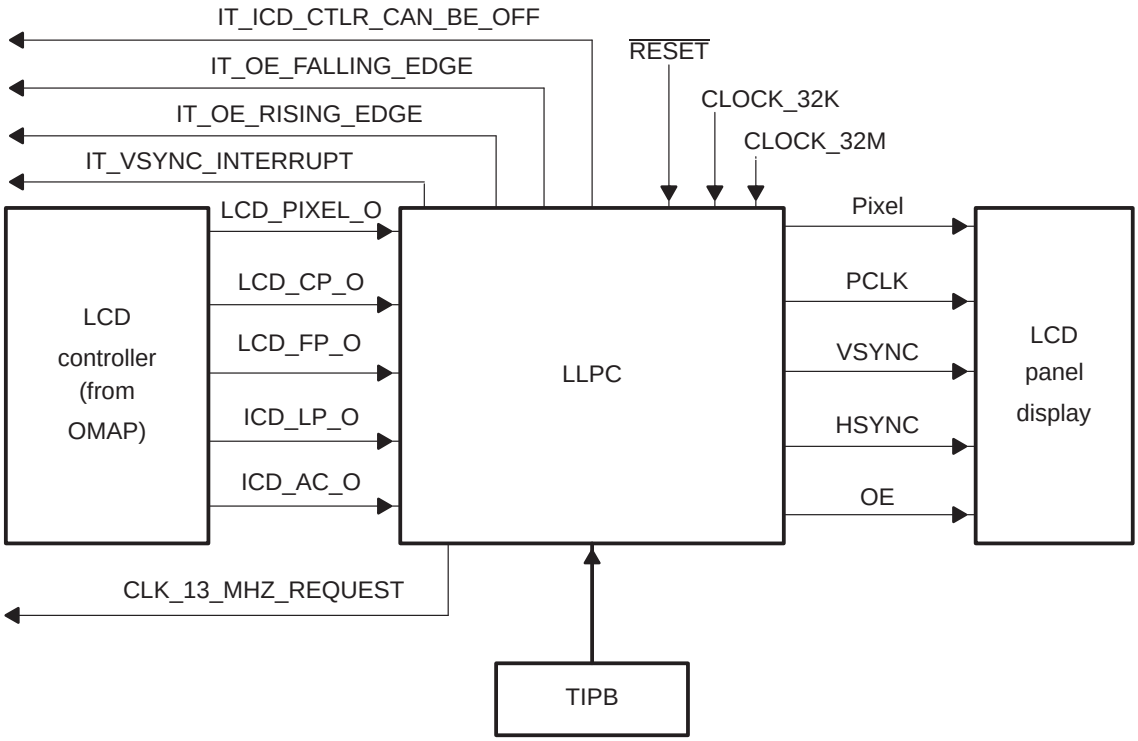
Table 2–186. Tick Counter Register (TCR)

Bit	Name	Function	Reset Value
31:24	Reserved		
23:0	TICK_COUNTER_REG	Value of timer	0xFFFFFF

2.18 LCD Low-Power Controller

This section describes the architecture, functionality, and configuration required by the LCD low-power controller (LLPC) to perform TFT panel control. The LLPC is a module located between the OMAP LCD controller and the external pins. It allows stopping of some signals, such as the pixel clock and data lines for a certain period of time in order to reduce power dissipation on these lines and on the LCD panel itself.

Figure 2–92. LCD Low-Power Controller Block Diagram



Notes:
1) The LLPC registers are configured (written and read) by the TIPB interface as usual.
2) The CLOCK_32K and CLOCK_13M are used in mode 3 to recreate HSYNC and VSYNC outputs signals.
3) IT_LCD_CTLR_CAN_BE_OFF indicates to the ULPD (or PCC) that clocks (except CLOCK_32K or CLOCK_13M) can be shut off in mode 3 (consequently, inputs signals are not generated).
4) IT_VSYNC_INTERRUPT indicates to the CPU that a programmable number of VSYNCs has occurred. It is an automatic wake-up signal. This interrupt can be disabled.
5) IT_OE_FALLING/RISING_EDGE indicates to the CPU that a falling/rising edge of OE input signal occurs. This interrupt can be disabled.
6) CLK_13MHZ_REQUEST indicates to the PCC that LLPC only needs the 13-MHz clock for mode 3. If this signal is low, only the 32-kHz clock is available for LLPC.

2.18.1 General Description

Table 2–187. Pinning Details

Functional Pins			
Pin Name	**Bits**	**Polarity**	**Function**
NRESET	1	Low	Asynchronous reset signal
CLOCK_32K	1	-	32-kHz clock for mode 3

Table 2–187. Pinning Details (Continued)

Functional Pins			
Pin Name	**Bits**	**Polarity**	**Function**
CLOCK_13M	1	-	13-MHz clock for mode 3
LCD_PIXEL_O	16	-	Pixel input
LCD_CP_O	1	-	Pixel clock input
LCD_FP_O	1	-	VSYNC input
LCD_LP_O	1	-	HSYNC input
LCD_AC_O	1	-	Output enable input
IT_LCD_CTLR_CAN_BE_OFF	1	Low	Interrupt signal to switch OMAP off
IT_VSYNC_INTERRUPT	1	Low	Interrupt signal to WAKE_UP OMAP
IT_OE_FALLING_EDGE	1	Low	Interrupt signal when OE FALLING_EDGE
IT_OE_RISING_EDGE	1	Low	Interrupt signal when OE RISING_EDGE
CLK_13MHZ_REQUEST	1	High	Request to PCC when 13-MHz clock needed
PIXEL	16	-	Pixel output
PCLK	1	-	Pixel clock output
VSYNC	1	-	VSYNC output
HSYNC	1	-	HSYNC output
OE	1	-	Output enable output
TIPB Ports			
TIPB Inputs			
RHEA_NRESET	1	Low	TIPB reset
RHEA_NSTROBE	1	Low	TIPB strobe
RHEA_CS	5	Low	TIPB chip select
RHEA_ADD	11	-	TIPB address
RHEA_RNW	2	-	TIPB read/not write
RHEA_DO	16	-	TIPB bus data input
RHEA_CS_VALUE	5	-	TIPB chip select value reference
TIPB Outputs			
RHEA_NREADY	1	Low	TIPB not ready
RHEA_PERHMAS	2	-	TIPB peripheral access size
RHEA_DI	16	-	TIPB data bus output
RHEA_NOE	1	Low	TIPB peripheral is driving the bus

Table 2–187. Pinning Details (Continued)

Functional Pins			
Pin Name	**Bits**	**Polarity**	**Function**
Scan Ports			
SCAN_CLK	1	-	Scan clock
SCAN_MODE	1	High	Scan mode select
SCAN_IN	1	-	Scan chain input
SCAN_OUT	1	-	Scan chain output
SCAN_ENABLE	1	High	Scan enable

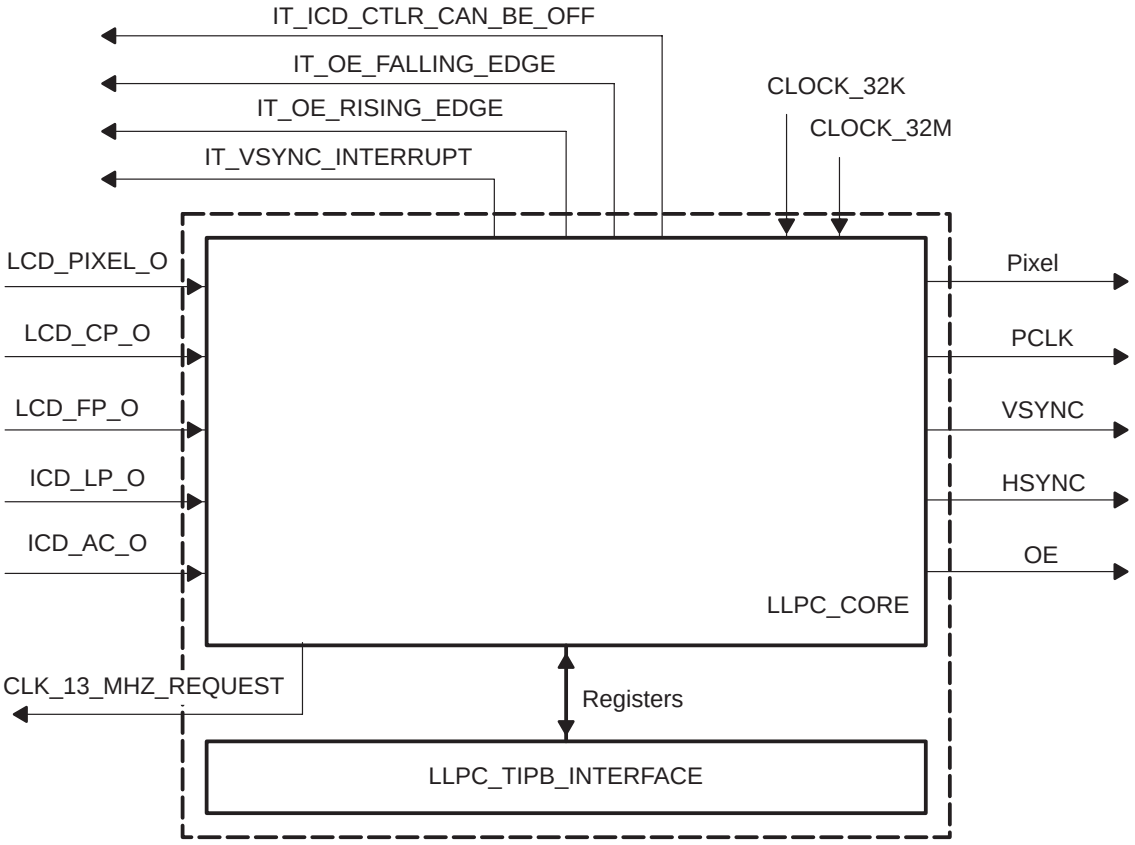
2.18.2 Block Diagram

The LLPC is composed of two blocks: the LLPC_RHEA_INTERFACE and the LLPC_CORE.

The LLPC_RHEA_INTERFACE controls TIPB access and permits writing and reading in LLPC registers.

The LLPC_CORE is the LLPC module core. Its behavior depends on the selected mode (0,1, 2, or 3). It has to control input signals and either bypass them or regenerate the outputs when there are no more input signals.

Figure 2–93. LLPC Block Diagram



2.18.3 LLPC Functional Description

Four different modes define how the LLPC behaves. The MODE field of the suspend register selects the mode as shown in Table 2–188.

Table 2–188. LLPC Functional Modes

MODE	Description
0	LLPC is in bypass mode and all signals are active. This is the default mode after reset. When this mode is selected while in modes 1 to 3, all suspended signals remain at the inactive state until the end of the current frame. All signals are activated at the beginning of the next frame. LLPC registers can only be modified in this mode.
1	All signals selected by the suspend register are suspended for VSOFFP of VSYNC periods. After that, all signals are activated during VSONP complete frames. This sequence is repeated until mode 0 is set.
2	All signals selected by the suspend register are suspended indefinitely until mode 0 is set.
3	All signals are suspended forever. Depending on the suspend register, the VSYNC and HSYNC signals are continuously generated from the 32-kHz clock (or 13-MHz clock). All the LCD clocks can then be shut off and the OMAP device put in low-power mode. Before switching to mode 0, the LCDC must be fully configured with all signals running.

When any of the LLPC_MODE 1 to 3 are selected, all signals remain active until the end of the frame. At the end of the frame (VSYNC), all signals selected are set into inactive level (programmable low or high) for the specified number of VSYNC periods. The following figures show the behavior in the different modes. In these examples, the inactive level is low for all signals. Depending on the LCD panel and LCDC configuration used, the inactive signal level can also be high level. The LVLC register can select this level individually for each signal. Modes 1 to 3 are turned off by selecting mode 0. All signals are activated at the beginning of the next VSYNC pulse.

Figure 2–94. Mode 0 Signals Handling Example

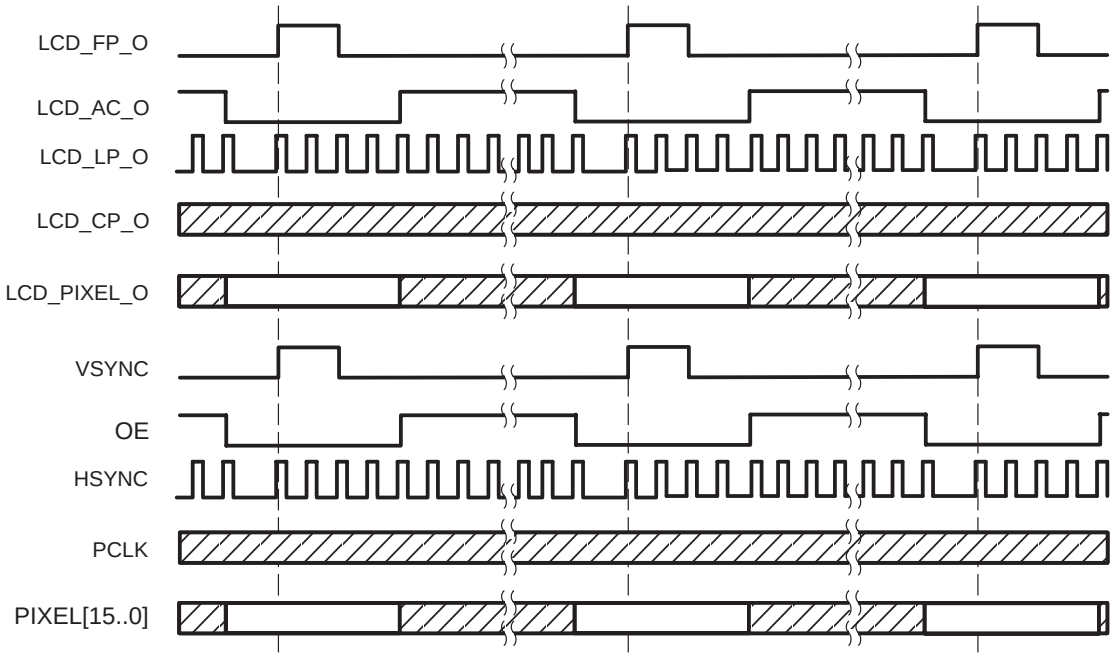


Figure 2–95. Mode 1 Example With OE, PCLK, and PIXEL Signals Suspended

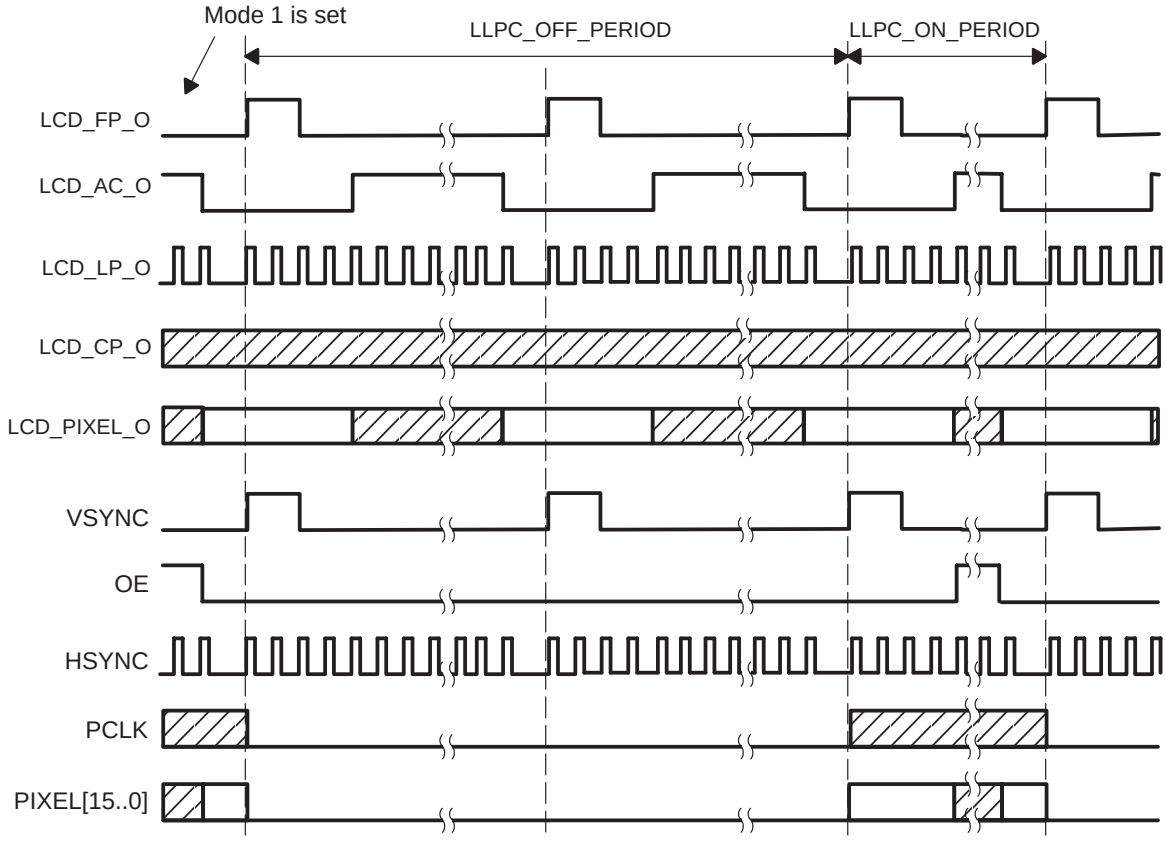


Figure 2–96. Mode 1 Example With HSYNC, OE, PCLK, and PIXEL Signals Suspended

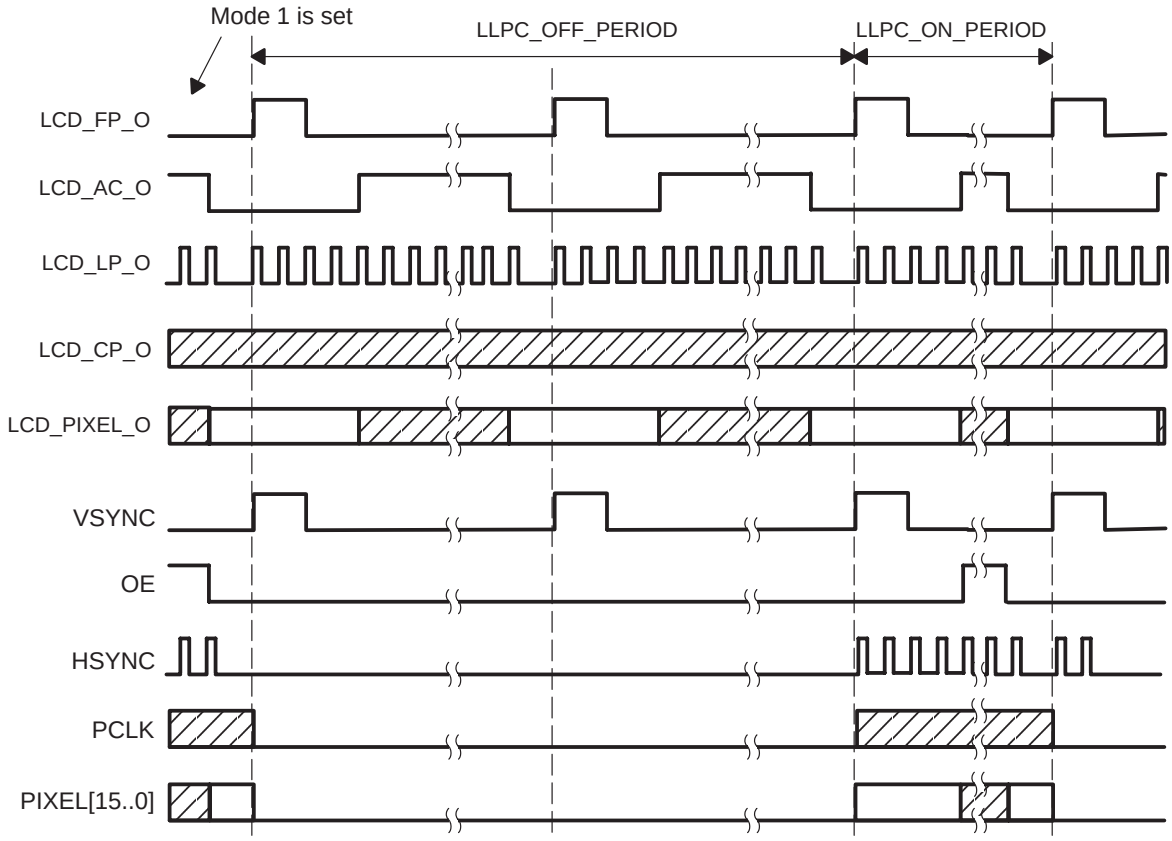


Figure 2–97. Mode 2 Signals Handling Example

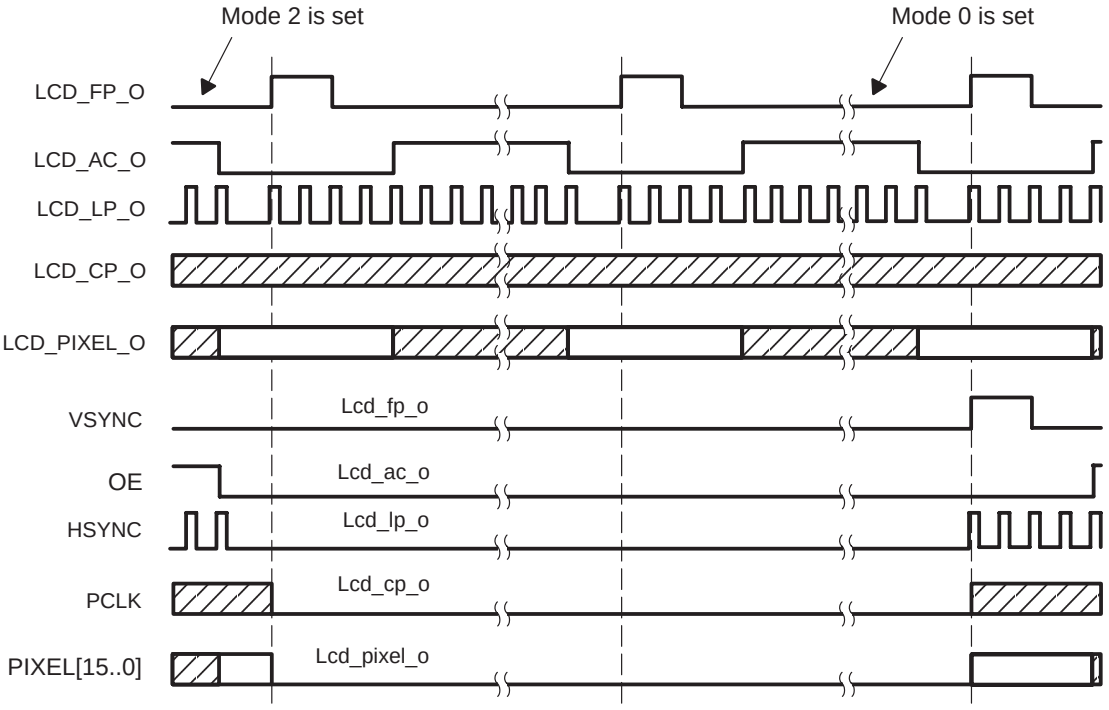
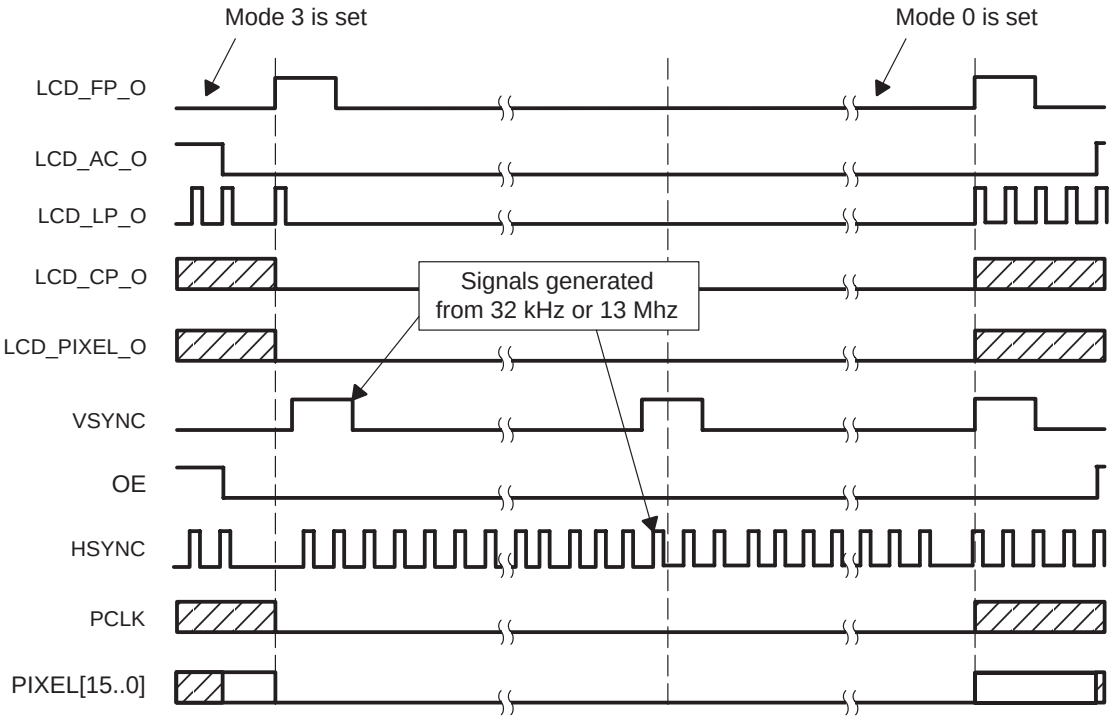


Figure 2–98. Mode 3 Signals Handling Example



2.18.4 LLPC Registers

2.18.4.1 LLPC Registers Overview

The LLPC has 12 internal registers, which can only be acceded by the TIPB interface. All but the SIGS register are 16-bit wide and have R/W access permissions. The SIGS register is also 16 bits wide, but only has read access permission. No register is resynchronized; they are all read/written on strobe (except the SIGS register, which is only read on strobe).

Note:

Address of one register: Start address + offset address

In Table 2–189 memory mapping is used to select the register access.

Table 2–189. LLPC Registers Memory Mapping

Register Name	Register Name in VHDL Code (Reference for Designers)	TIPB Address (offset Hex)	Reset Value (Hex)
VSOFFP	LLPC_OFF_PERIOD_REG	0000	FFFF
VSONP	LLPC_ON_PERIOD_REG	0002	FFFF
SUSPEND	LLPC_MODE_SUSPEND_SIG_REG	0004	0000
HSCNT	HSW_PPL_REG	0006	FFFF
VSCNT	VSW_LPP_REG	0008	07FF
LVLC	LEVEL_ACTIVITY_CONTROL_REG	000A	0000

Table 2–189. LLPC Registers Memory Mapping (Continued)

Register Name	Register Name in VHDL Code (Reference for Designers)	TIPB Address (offset Hex)	Reset Value (Hex)
HFBP	HFP_HBP_REG	000C	FFFF
VFBP	VFP_VBP_REG	000E	0101
VSIT	VSYNC_INTERRUPT_REG	0010	00FF
SIGS	VSYNC_OE_STATUS	0012	0000
CLKM	CLK_MODE_REG	0014	0000
OEEVT	OE_EVENT_REG	0016	0000

2.18.5 LLPC Registers Content

The HFBP and HSCNT registers must be written in a first time, but a waiting period of at least one 32-kHz period (or 13-MHz, according low-power clock mode) is required before changing modes to ensure that data in the registers are stable and latched correctly.

Registers VFBP and VSCNT must be written in a first time, but a minimum waiting period of HSYNC is required before changing modes to ensure that data inside the registers are stable and latched correctly.

Registers VSOFFP and VSONP must be written in a first time, but a minimum waiting period of VSYNC is required before changing modes to ensure that data inside are stable and latched correctly.

Table 2–190. VSYNC OFF Period Register (VSOFFP)

Bits	Field	Description	Reset Value
15:8	UNUSED	-	0x1
7:0	CNT	Number of VSYNC periods to fix OFF_PERIOD	0xFF

The VSYNC OFF period register is used in mode 1 to load a count-down register. Its value corresponds to a number of VSYNC periods to wait for OFF_PERIOD and can be changed only in mode 0.

Table 2–191. VSYNC ON Period Register (VSONP)

Bits	Field	Description	Reset Value
15:8	UNUSED	-	0x1
7:0	CNT	Number of VSYNC period to fix ON_PERIOD	0xFF

The VSYNC on register is used in mode 1 to load a count-down register. Its value corresponds to a number of VSYNC periods to wait for ON_PERIOD and can be changed only in mode 0.

Table 2–192. Suspend Register (SUSPEND)

Bits	Field	Description	Reset Value
15:7	UNUSED	-	0
6:5	MODE	Mode selected (changes at the 2nd next VSYNC rising edge)	Normal
4	PIXEL	The PIXEL output is suspended: 0: Disable 1: Enable	0
3	PCLK	The PCLK output is suspended: 0: Disable 1: Enable	0
2	OE	The OE output is suspended: 0: Disable 1: Enable	0
1	HSYNC	The HSYNC output is suspended: 0: Disable 1: Enable	0
0	VSYNC	The VSYNC output is suspended: 0:Disable 1: Enable	0

The suspend register is used in select mode to select signal outputs to suspend in modes 1, 2, or 3.

Note:

Signals suspended in outputs can be suspended at high- or low level according to the LEVEL_ACTIVITY_CONTROL register. The VSYNC output signal cannot be suspended in mode 3.

Table 2–193. HSYNC Counters Register (HSCNT)

Bits	Field	Description	Reset Value
15:10	HSW	Horizontal synchronous pulse width (min = 1)	FF
9:0	PPL	Pixels per line (min = 1)	FF

HSCNT is used in mode 3 for HSYNC signal generation (see timings in the LCD controller specification).

- HSW is used to load a count-down register. Its value corresponds to a number of CLOCK_32K or CLOCK_13M periods to determine HSYNC pulse width and can be changed only in mode 0.
- PPL is used to load a count-down register. Its value corresponds to a number of CLOCK_32K or CLOCK_13M periods to represent PPL timing (used in LCD controller). Its value can be changed only in mode 0.

Table 2–194. VSYNC Counters Register (VSCNT)

Bits	Field	Description	Access	Reset Value
15:10	VSW	Vertical synchronous pulse width (min = 1)	R/W	07
9:0	LPP	Lines per panel (min = 1)	R/W	FF

VSCNT is used in mode 3 for VSYNC signal generation (see timings in the LCD controller specification).

- VSW is used to load a count-down register. Its value corresponds to a number of HSYNC periods to determine VSYNC pulse width and can be changed only in mode 0.
- LPP is used to load a count-down register. Its value corresponds to a number of HSYNC periods to represent LPP timing (used in LCD controller). Its value can be changed only in mode 0.

Table 2–195. Level Activity Control Register (LVLC)

Bits	Field	Description	Reset Value
15:6	UNUSED	–	0
5	HVSACT	HSYNC and VSYNC active level	0
4	PIXELSPD	PIXEL suspended level	0
3	PCLKSPD	PCLK suspended level	0
2	OESPD	OE suspended level	0
1	HSYNCSPD	HSYNC suspended level	0
0	VSYNCSPD	VSYNC suspended level	0

The level activity control register selects the output level of suspended signals. If bit[4:0] is 0, these signals are suspended at low level in mode 1, 2, or 3; else, high level. This register is also used to inform the LLPC whether VSYNC and HSYNC input signals are active high or low. Hence, if bit[5] is 0, HSYNC and VSYNC input signals are active high; else, they are active low. This register can only be changed in mode 0.

Table 2–196. Horizontal Front/Back Porch Register (HFBP)

Bits	Field	Description	Reset Value
15:8	HFP	Horizontal front porch (min = 1)	FF
7:0	HBP	Horizontal back porch (min = 1)	FF

HFBP is used in mode 3 for HSYNC signal generation (see timings in the LCD controller specification).

- HFP is used to load a count-down register. Its value corresponds to a number of CLOCK_32K or CLOCK_13M periods to represent HFP timing (used in LCD controller). Its value can be changed only in mode 0.
- HBP is used to load a count-down register. Its value corresponds to a number of CLOCK_32K or CLOCK_13M periods to represent HBP timing (used in LCD controller). Its value can be changed only in mode 0.

Table 2–197. Vertical Front/Back Porch Register (VFBP)

Bits	Field	Description	Reset Value
15:8	VFP	Vertical front porch	01
7:0	VBP	Vertical back porch	01

The vertical front/back porch register is used in mode 3 for VSYNC signal generation (see timings in the LCD controller specification).

- VFP is used to load a count-down register. Its value corresponds to a number of HSYNC periods that represent VFP timing (used in LCD controller). Its value can be changed only in mode 0.
- VBP is used to load a count-down register. Its value corresponds to a number of HSYNC periods that represent VBP timing (used in LCD controller). Its value can be changed only in mode 0.

Table 2–198. VSYNC Interrupt Register (VSIT)

Bits	Field	Description	Reset Value
15:10	UNUSED	-	0
9	MODE	Interrupt mode: 0: Repetitive 1: One-shot	0
8	ENAB	VSYNC_INTERRUPT enabling: OFF(0): Disable ON (1): Enable	0
7:0	CNT	Number of VSYNC periods to wait until an interrupt occurs	FF

VSIT is used in all modes and can be changed only in mode 0. Bits[7:0] correspond to the number of VSYNC periods that must be counted until LLPC launches an interrupt on IT_VSYNC_INTERRUPT port. Bit[9] configures the mode of interruption. One-shot mode permits launching of only one interrupt (LLPC reset VSYNC_INTERRUPT signal after IT). Consequently, the CPU must reenable this register to have another interrupt. Repetitive mode is used to launch an interrupt the same number of VSYNC periods until the CPU disables this register (this mode is used to wake up the CPU regularly).

Note:

The count-down register must be set before enabling the register in a second access. See Section 2.18.8, *Software Procedures*.

Table 2–199. Signals Status Register (SIGS)

Bits	Field	Description	Reset Value
15:2	UNUSED	-	0
1	VSYNC	VSYNC status	0
0	OE	OE status	0

This read-only SIGS register indicates the CPU values of VSYNC and OE output signals in all modes.

Note:

As register NFWAIT is clocked on 32-kHz or 13-MHz clock (according to CLOCK_MODE selected), the status of these bits is not realistic. Hence, there is a 32-kHz or13-MHz period of errors between the real value and the value stored in this register.

2.18.5.1 Clock Mode Register (CLKM)

Bits	Field	Description	Reset Value
15:3	UNUSED	–	0
2	HS_VS_DELAYED	HSYNC and VSYNC outputs must be delayed? 0: HS and VS are not 1/2 cycle delayed to meet requirements of such panels. 1: HS and VS are 1/2 cycle delayed to meet requirements of such panels.	0
1	PCLK_INVERTED	Pixeclock input inverted? (information bit) 0: Input PCLK not inverted Pixels are latched on PCLK rising-edge 1: Input PCLK inverted Pixels are latched on PCLK falling-edge	0
0	CLK13M_REQ	Clock source selection: 0: LLPC 32-kHz input clock 1: PCC request for 13-MHz clock	0

The clock mode register is used in mode 3 and can be changed only in mode 0. It informs the LLPC which clock (32-kHz or 13-MHz) to use to generate HSYNC and VSYNC output signals. Consequently, when bit[0] is set, the signal CLK_13_MHZ request from LLPC to PCC is activated. It is also used to delay the PCLK VSYNC and HSYNC signals ½ cycle in output (according to the polarity of PCLK). When bit[2] = 1, HSYNC and VSYNC are delayed to meet the requirements of such panel. Bit[1] corresponds to the edge where pixels are latched inside the LCD controller.

Note:

Depending on the clock mode selected by the CPU, all timings (HSW, VSW, LPP, PPL, VBP, VFP, HFP, and HBP) must be recalculated and corresponding registers must be rewritten.

This register must be set before all other registers and must wait two periods of 32-kHz or 13-MHz clock to prevent glitch generation.

2.18.5.2 OE Event Register (OEEVT)

Bits	Field	Description	Reset Value
15:2	UNUSED	–	0
1	FALL	Enable of falling edge interrupt: 0: Disable 1: Enable	0
0	RISE	Enable of rising edge IT: 0: Disable 1: Enable	0

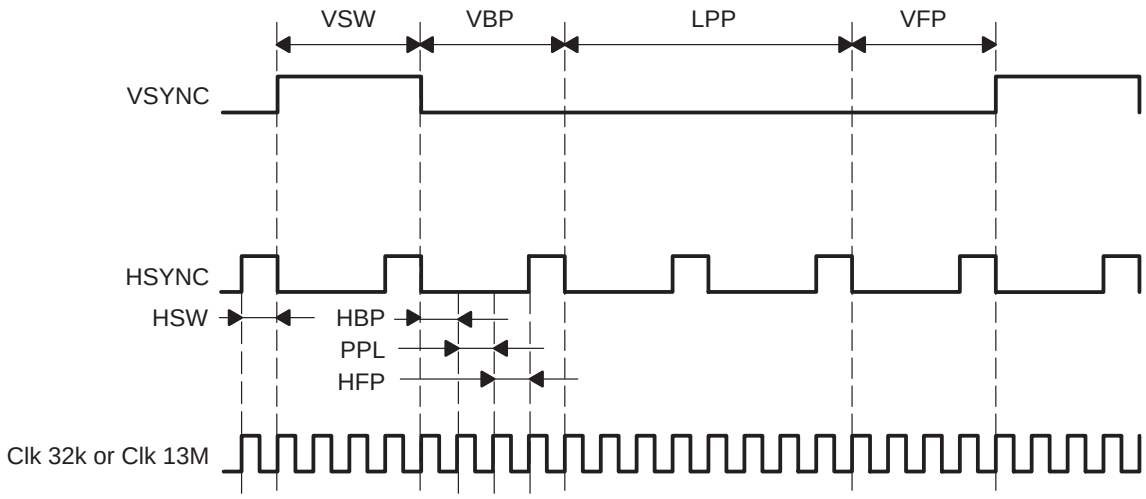
The OE event register is used in mode 0, 1, and 2 and can be changed only in mode 0. It is used to enable or disable an interrupt on the rising or falling edge of an OE input signal.

> **WARNING**
>
> To properly detect a rising or falling edge on OE, the OE pulse width must be at least two 32-kHz (or 13-MHz, according to low-power clock) periods.

2.18.6 Required Timing

To correctly configure timing registers such as HSW_PPL, VSW_LPP, HFP_HBP, and VFP_HBP, the width of VSYNC and HSYNC pulses must be recalculated according to the only clock used in mode 3. All timing of VSYNC and HSYNC frames generated by the LCD controller must be noted and values of the registers must be consequently filled.

Figure 2–99. Example of VSYNC and HSYNC Output Signals Generated by LLPC in Mode 3



The values of the registers in the example illustrated in Figure 2–99 are:

HSW = 1 (one clock period of 32-kHz or 13-MHz clock)
HBP = 1 (one clock period of 32-kHz or 13-MHz clock)
PPL = 1 (one clock period of 32-kHz or 13-MHz clock)
HFP = 1 (one clock period of 32-kHz or 13-MHz clock)
VSW = 1 (one HSYNC period)
VSW = 1 (one HSYNC period)
VSW = 2 (two HSYNC periods)
VSW = 1 (one HSYNC period)

2.18.7 Interruptions and Requests

The LLPC module has four interruptions and one request.

All interruptions are destined to CPU (OMAP), active on falling-edge, and automatically reset by LLPC itself (In fact, LLPC generates a pulse). They are queued in the interrupt handler.

Only three interruptions (IT_VSYNC_interrupt, IT_OE_FALLING_EDGE, and IT_OE_RISING_EDGE) can be disabled individually. IT_LCD_CTLR_CAN_BE_OFF is always enabled. CLK_13_MHZ_REQUEST is destined to PCC. If this bit is low, LLPC must generate HSYNC and VSYNC signals with 32-kHz clock in mode 3. If this bit is high, PCC must provide a 13-MHz clock to LLPC during each low level of this bit request.

Note:

This bit is directly updated with bit[o] of CLKM register.

2.18.8 Software Procedures

- The CLKM register must be the first register to be configured after reset. Next, the suspended signals and HSYNC and VSYNC active level must be defined in the suspend register. Then, other timing registers can be configured. Finally, select MODE in the suspend register.
- VSYNC_INTERRUPT registers must be written in two accesses: the first one to select the interrupt mode and to define the value of the count-down register and, after a period of time (VSW+VBP+LPP+VFP), the second one to enable this interruption.

Note:

The (VSW+VBP+LPP+VFP) time is necessary because of the VSYNC_INTERRUPT generation mechanism. Moreover, these registers are updated two more VSYNC periods later (because of resynchronization).

- Do not forget that when a mode is selected by writing to the suspend register, this mode is effectively taken into account the second VSYNC period after (because of the resynchronization process).

- The VSYNC output signal cannot be suspended in mode 3.
- No register, except for SUSPEND, can be modified in modes 1, 2, or 3.
- The generation of the VSYNC and HSYNC inputs signals from the LCD controller must be stopped when IT_LCD_CTLR_CAN_BE_OFF occurs.
- For CPU (OMAP) to make LLPC work with 13-MHz clock in mode 3 (by writing to the CLKM register), it must wait for this 13-MHz clock (provided by a PLL) to stabilize before changing modes from 0 to 3.

2.19 DSP Memory Management Unit (DSP MMU)

The DSP memory management unit (MMU) supports memory mapping for the DSP port. On OMAP730, the GSM subsystem and the TCIF are connected to the DSP port. The platform system software can relocate regions of the GSM subsystem logical address space via a page table and the DSP MMU.

The DSP MMU contains a 32-entry translation lookaside buffer (TLB) that holds translations and permissions for current pages. This TLB is often managed statically by the MPU OS. The MMU also includes hardware table walking logic, as in the MPU, to autonomously traverse the page table on a TLB miss. In this case, the TLB can be seen as a cache of recently used page table entries.

The format of the page table and TLB entries of the DSP match those of the MPU.

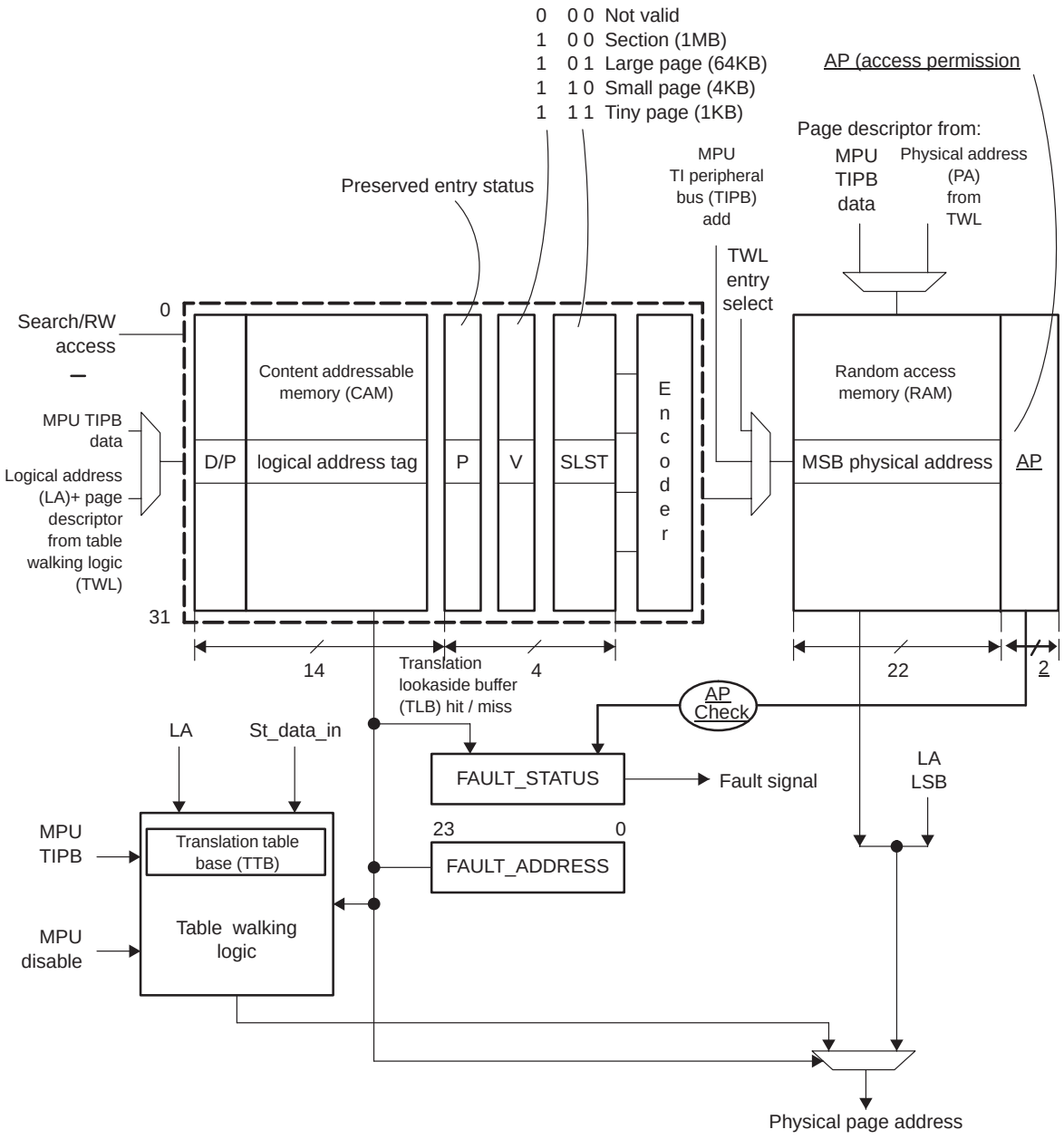
If the DSP MMU requires software intervention, the MPU is responsible for servicing the event; DSP MMU errors are signaled to the MPU with a dedicated interrupt. The DSP MMU is programmed by the MPU.

The main functions of the MMU are:

- Translating the DSP internal (logical) addresses into OMAP (physical) addresses
- Preventing the DSP software from making invalid accesses to system memory

Figure 2–100 shows the relation between the major blocks of the MMU. Subsequent sections describe the functions of each of the blocks.

Figure 2–100. DSP MMU Architecture



2.20 Address Translation

The address translation from logical address to physical address is made using the translation lookaside buffer (TLB) contained in the MMU.

The MPU software typically loads the TLB entries of the MMU before enabling the MMU (MMU_EN bit in CNTL_REG).

When the MMU is disabled, no translation is done. The host addresses pass through untranslated, and no permission checks or table walking are performed.

The TLB contains two embedded memories:

- CAM

  Each entry contains the logical address tag, the preserved bits, valid bits, and page size.

- RAM

  Each entry contains the upper part of the associated physical address and the access protection field.

The MPU can access the CAM and the RAM when hardware table walking logic is disabled.

2.20.1 Translation Process

This section includes a brief introduction to MMU behavior. For more details, see the MPU subsystem documentation.

The following page sizes are supported:

- Section: 1M byte
- Large page: 64K bytes
- Small page: 4K bytes
- Tiny page: 1K byte

The page size and the upper bits of the input (logical) address are used to index the TLB CAM. Assuming there is a match (hit) in the CAM, the upper bits of the output (physical) address are read from the associated TLB RAM entry. These bits are then concatenated with the lower bits of the logical address. The boundary of translated and untranslated bits is a function of the page size.

Figure 2–101 shows an example of how the physical address is built with the logical address.

Figure 2–101. Address Translation Process for a Section

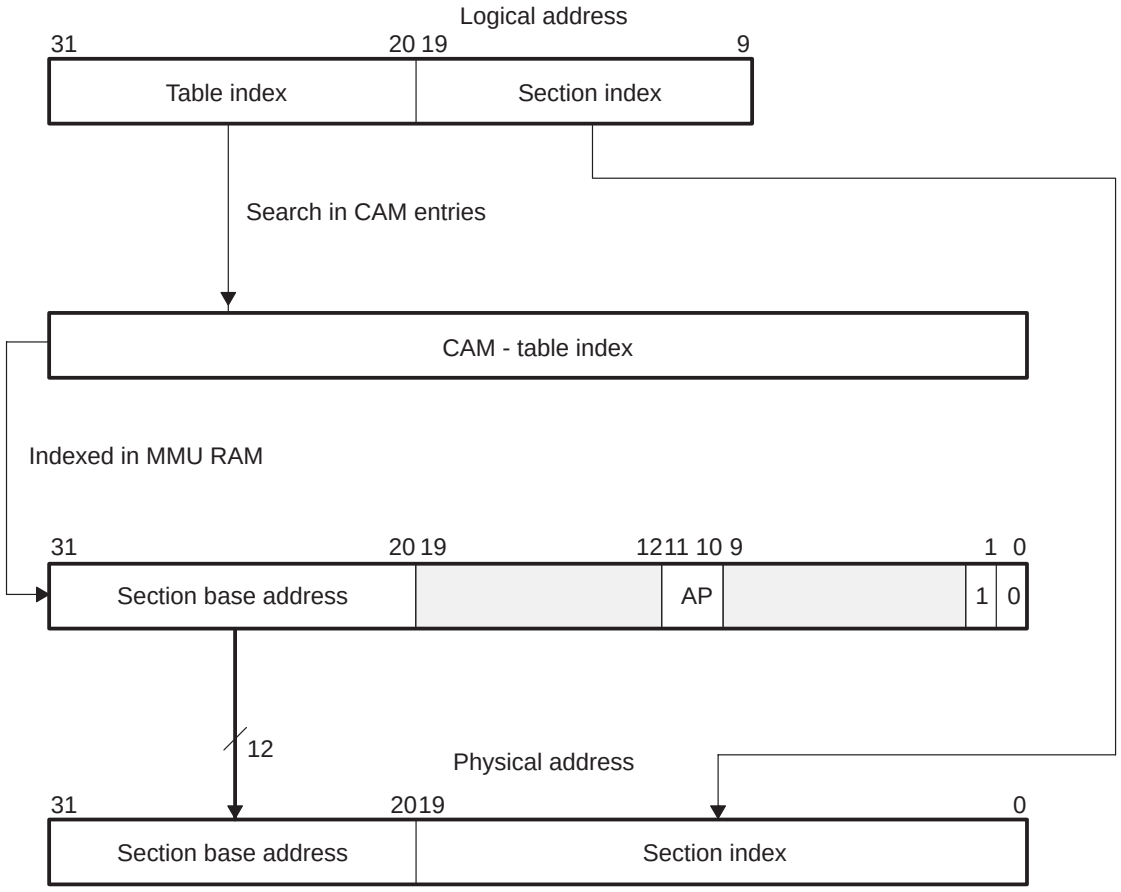
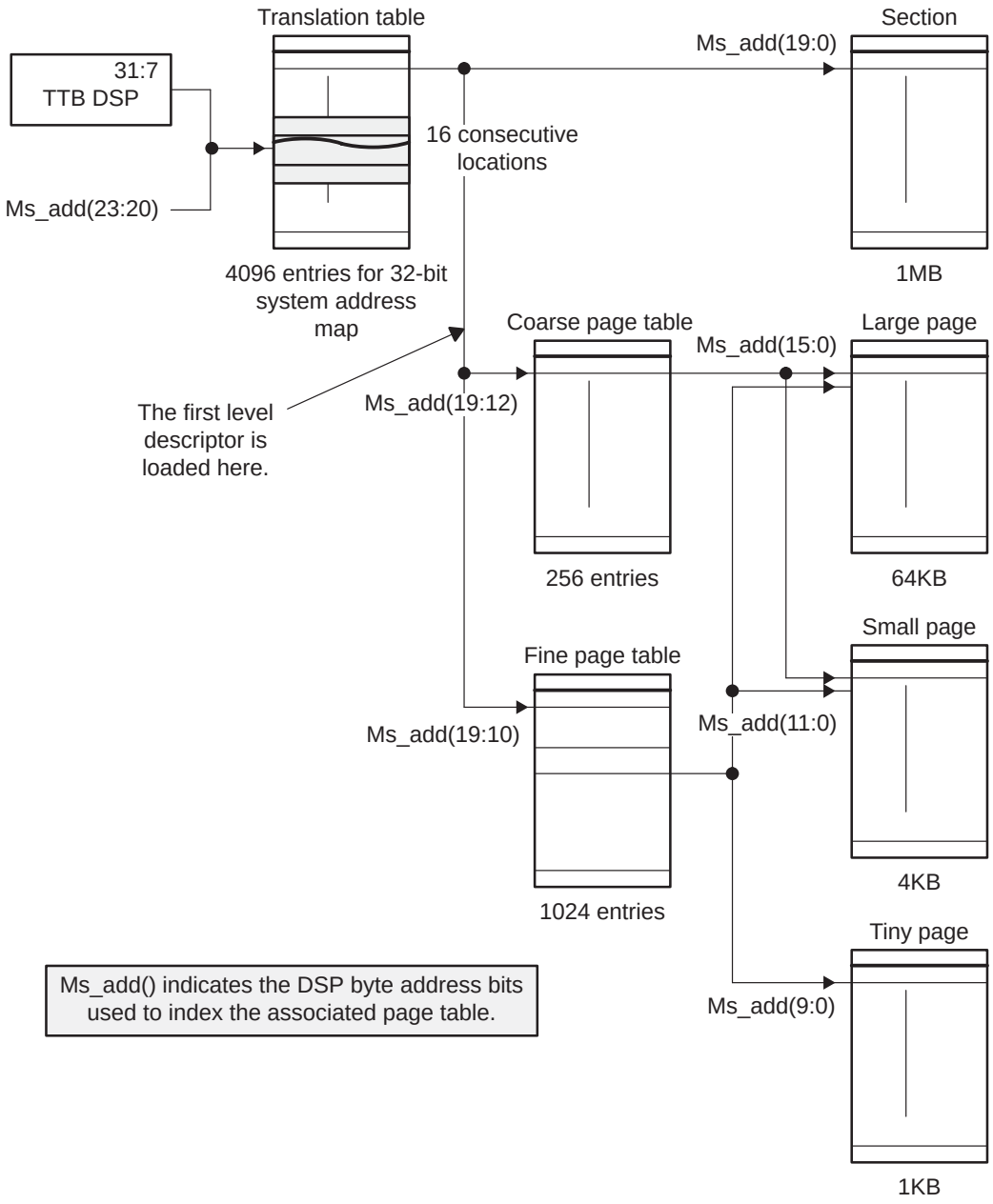


Figure 2–102 shows the translation table hierarchy. The physical address is built with the logical address.

As with the MPU, the page table may be hierarchical, as shown in Figure 2–102.

Figure 2–102. Translation Table Hierarchy



Note: The MMU passes the lower bits of the DSP address unchanged.

2.20.2 Page Table Format

Table 2–200. Level One Descriptor

31:20	19:12	11:10	9:2	1	0	
				0	0	Fault
Coarse page table base address			R	0	1	Coarse page
Section base address		AP		1	0	Section
Fine page table base address		R		1	1	Fine page

Notes: 1) AP = Protection bits for the page
2) R indicates reserved bits to be written as 0 and read value to be ignored.

Table 2–201. Level Two Descriptor

31:16	15:12	11:10	9:6	5:4	3:2	1	0	
						0	0	Fault
Large-page table base address	R			AP	R	0	1	Large page
Small-page table base address		R		AP	R	1	0	Small page
Tiny-page table base address			R	AP	R	1	1	Tiny page

Notes: 1) AP = Protection bits for the page
2) R indicates reserved bits to be written as 0 and read value to be ignored.

Table 2–202. Page Protection Field

AP	Access	Description
0x	No access	Any access causes a permission fault.
10	Read only	Write causes a permission fault.
11	Full access	Read and write access allowed.

2.20.3 Coarse Page Tables

Coarse page tables have 256 entries and each entry describes 4K bytes. These entries provide a base address for either small or large pages. Large page descriptors must be repeated in 16 consecutive entries.

Figure 2–103 shows the translation for a section.

Figure 2–103. Translation for a Section

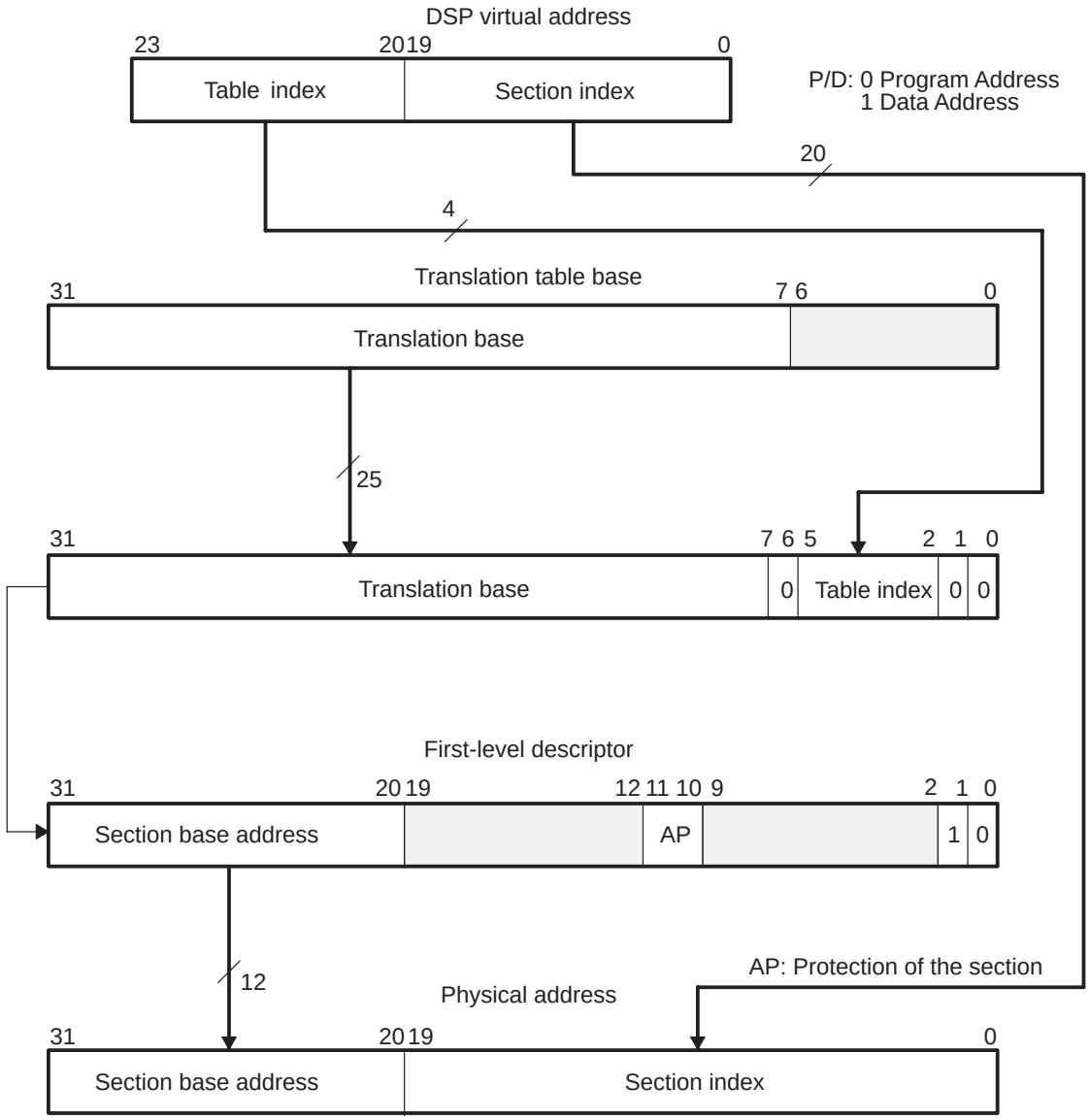


Figure 2–104 and Figure 2–105 show how the physical address is built as a function of the page size and hierarchy.

Figure 2–104. Translation for a Large Page Included in a Coarse Page

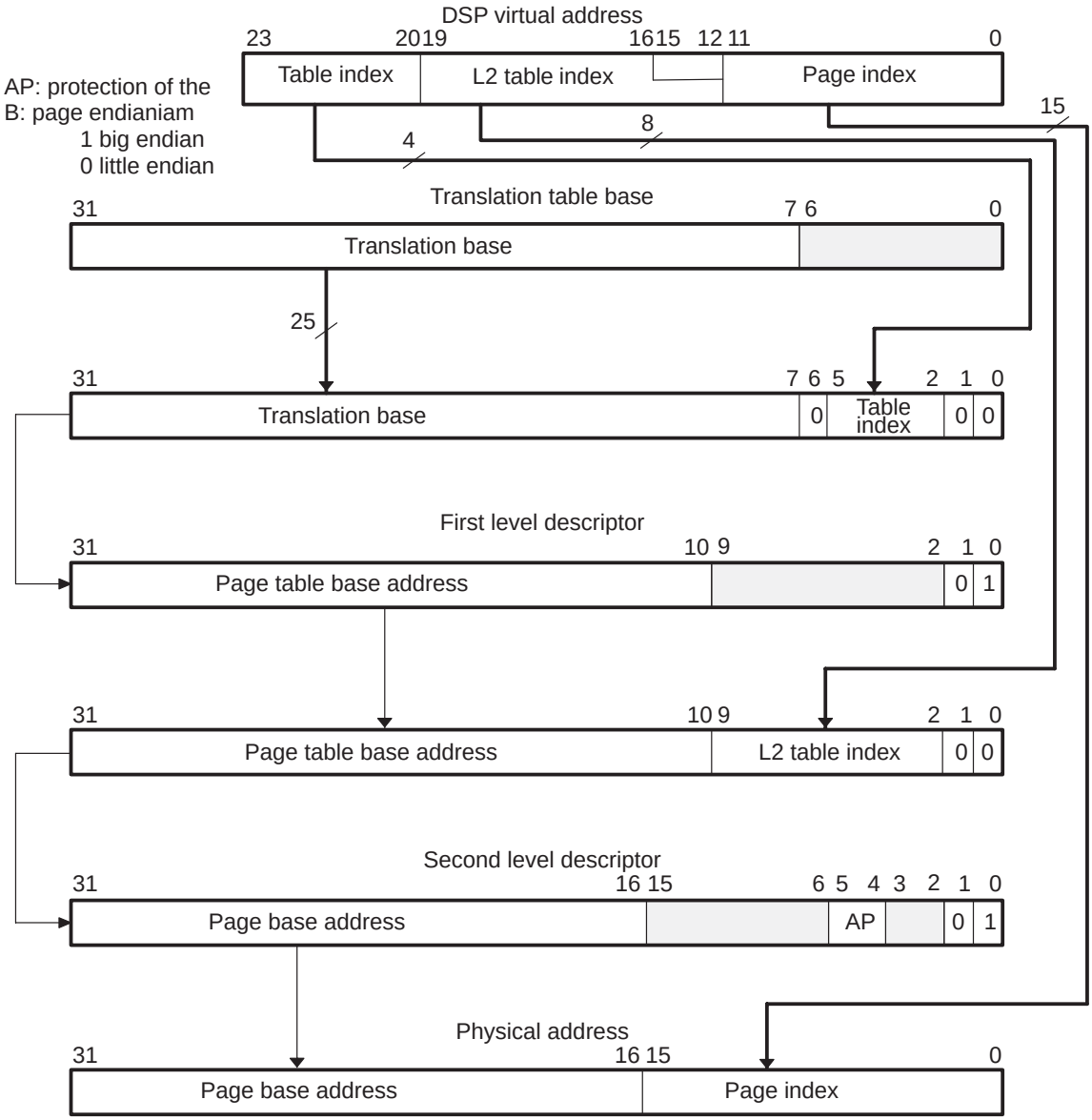
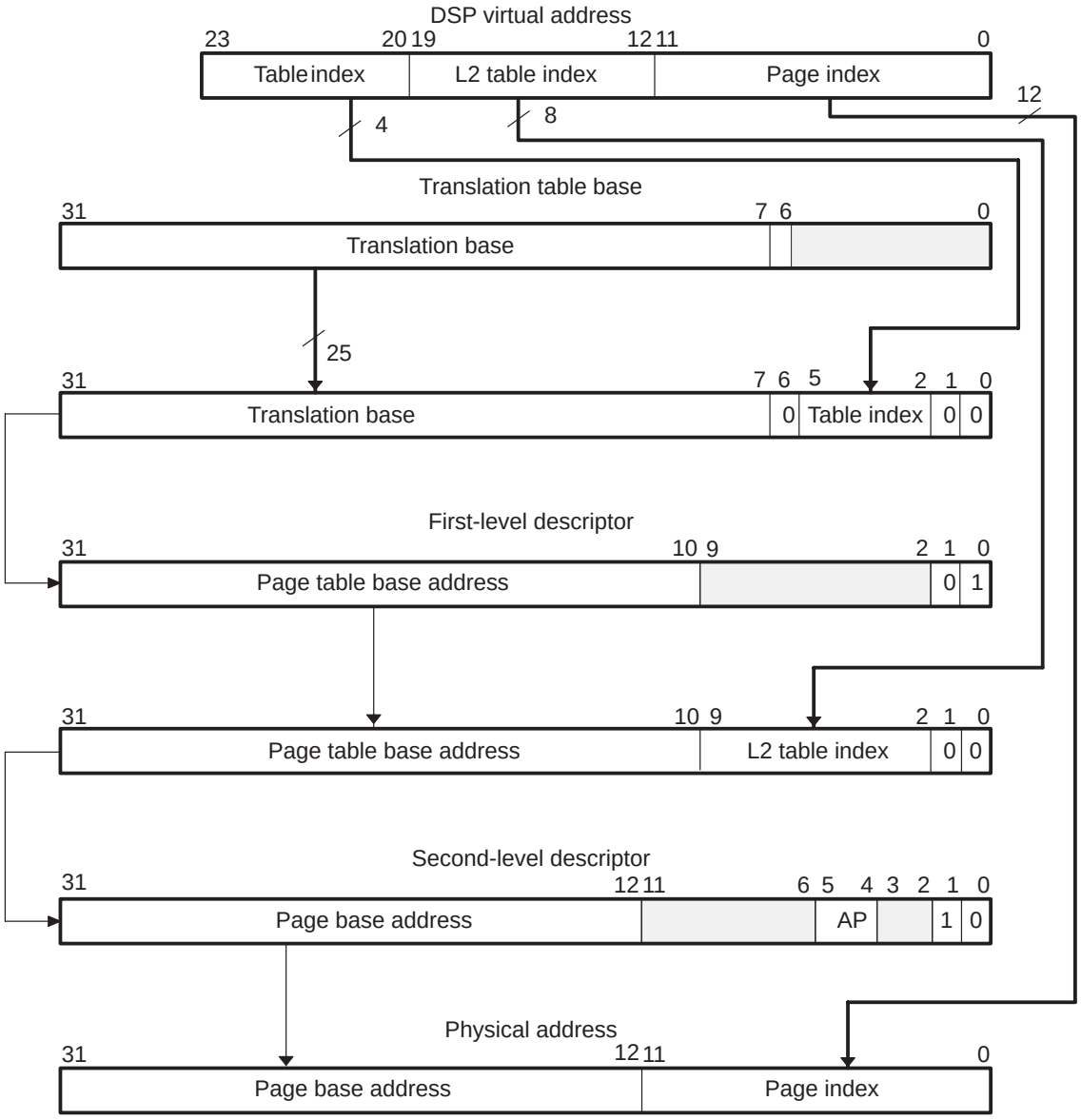


Figure 2–105. Translation for a Small Page Included in a Coarse Page



2.20.4 Fine Page Tables

Fine page tables have 1024 entries, and each entry describes 1K byte. These entries provide a base address for tiny, small, or large pages. Small-page descriptors must be repeated in four consecutive entries. Large-page descriptors must be repeated in 64 consecutive entries.

Figure 2–106, Figure 2–107, and Figure 2–108 show how the physical address is built as a function of the page size and hierarchy.

Figure 2–106. Translation for a Large Page Included in a Fine Page

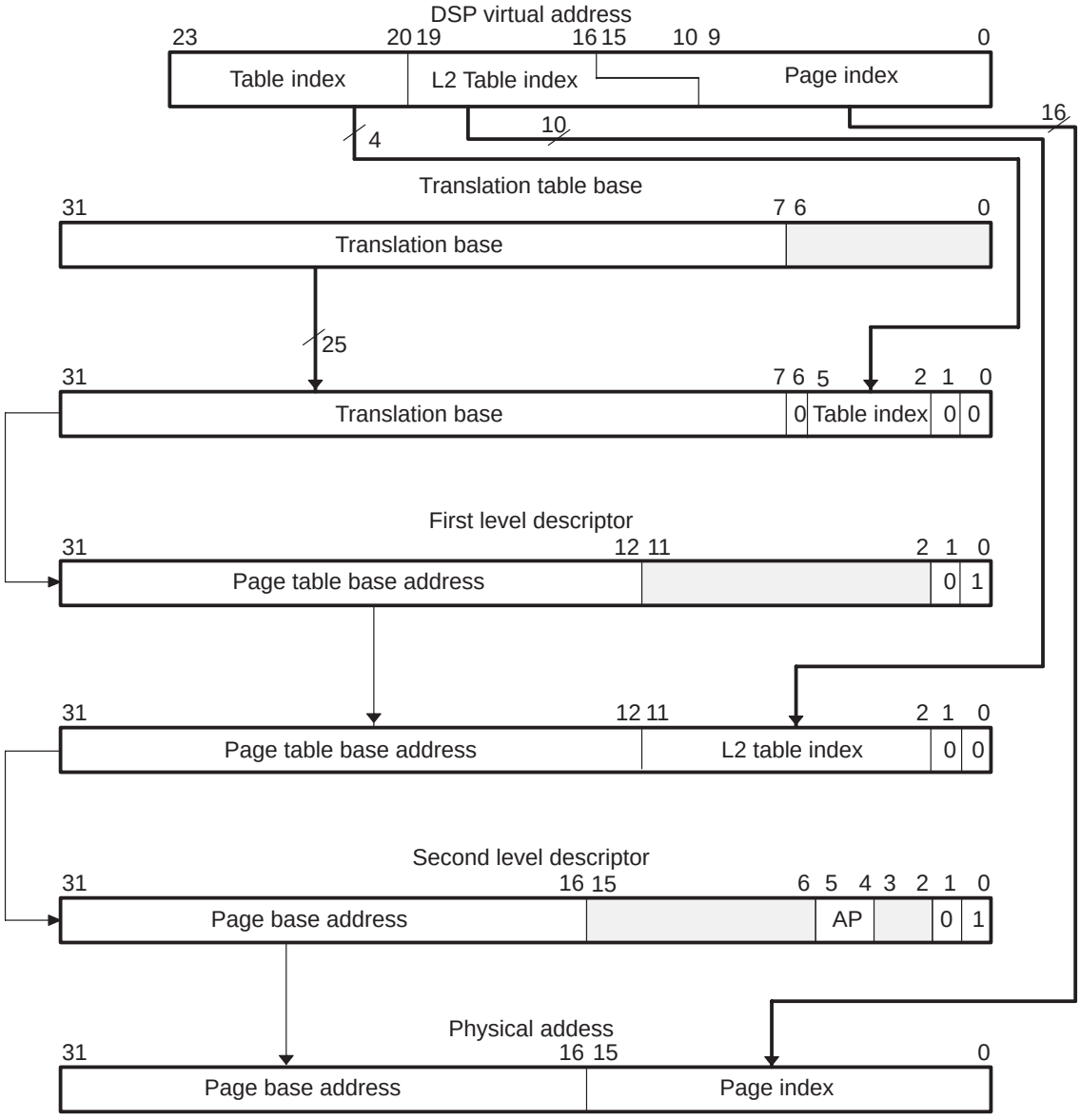


Figure 2–107. Translation for a Small Page Included in a Fine Page

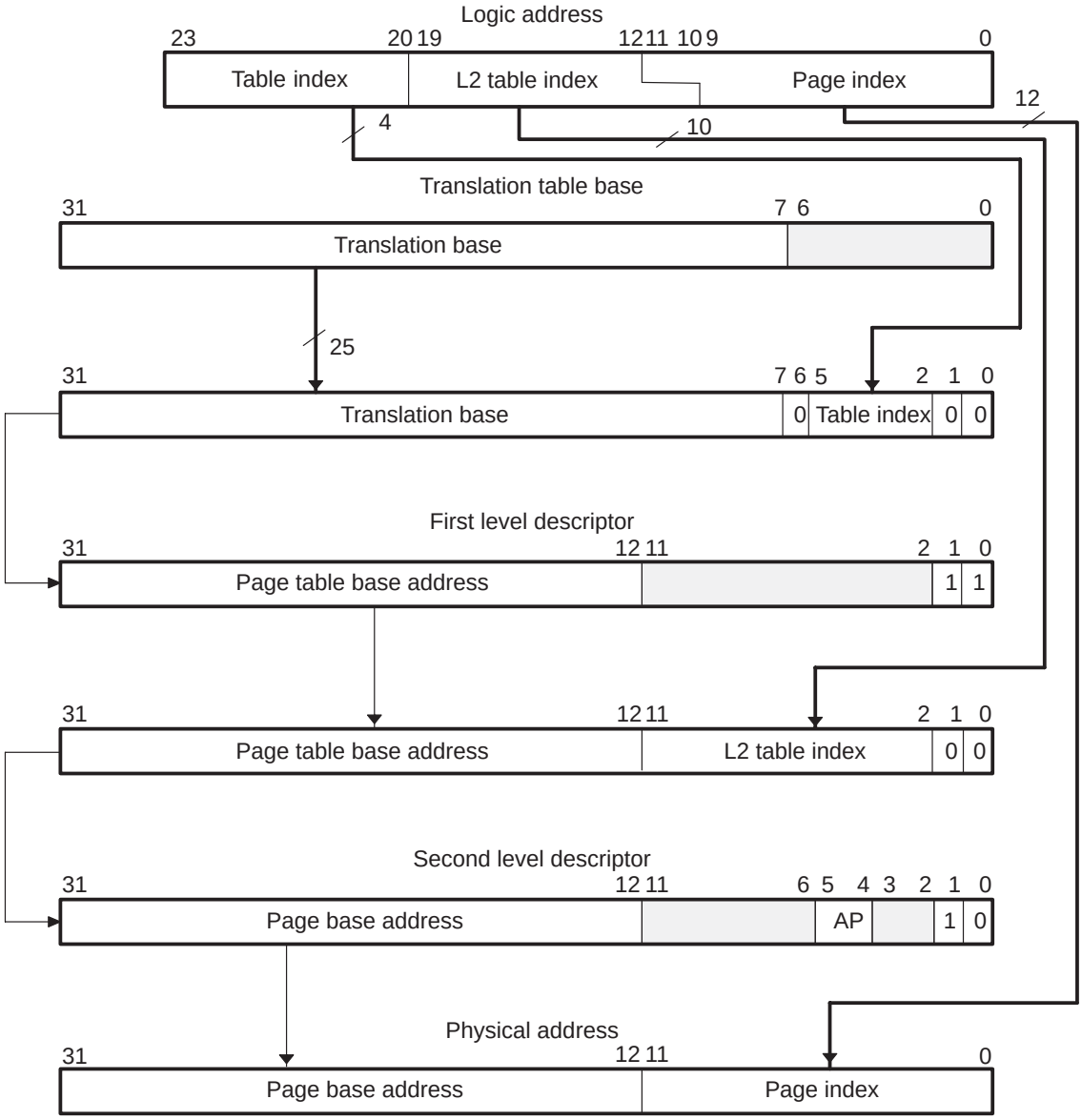
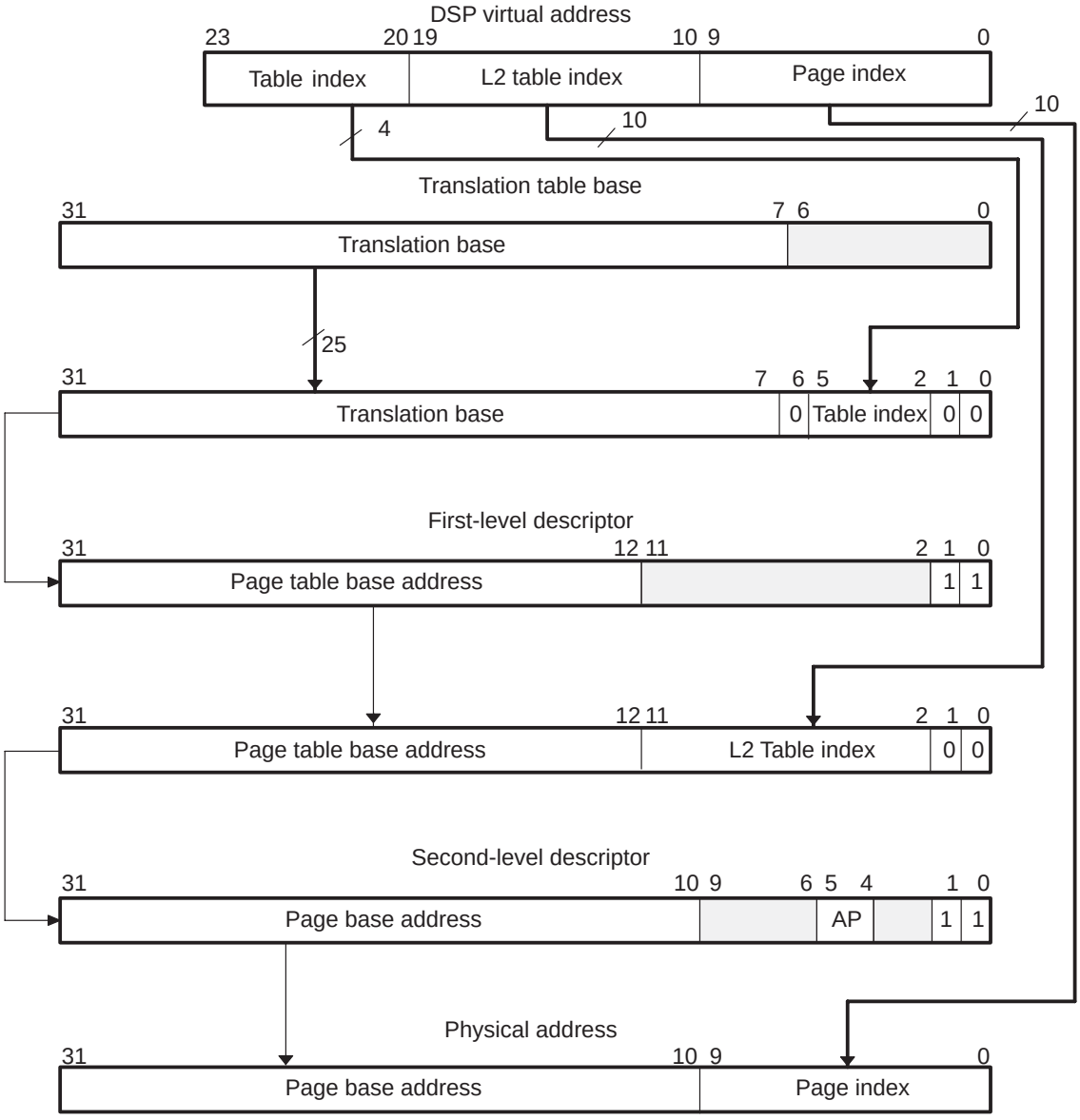


Figure 2–108. Translation for a Tiny Page Included in a Fine Page

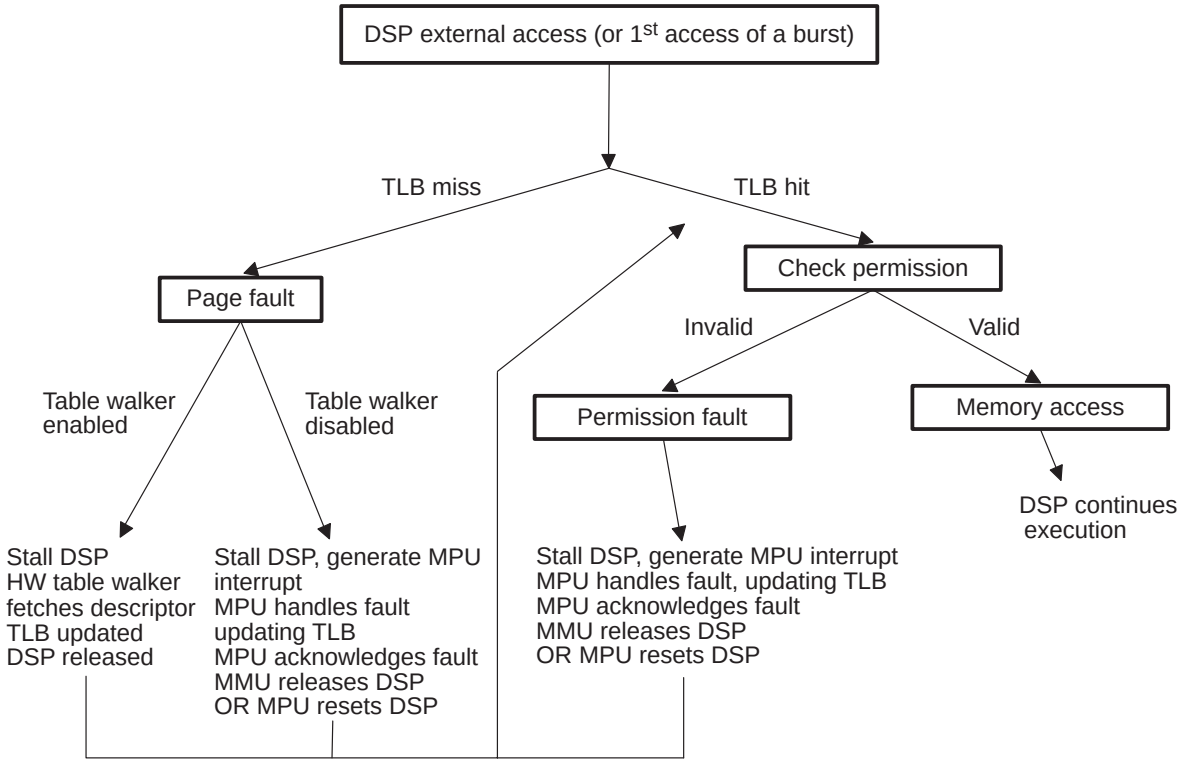


2.21 Functionality

2.21.1 Translation Summary

Figure 2–109 shows the possible results of a DSP memory request.

Figure 2–109. DSP Memory Request Results Example



2.21.2 Lock Mechanism and the CURRENT_VICTIM Counter

Any TLB entry can be locked. The maximum number that can be locked is 31 entries. The lock mechanism prevents an entry of the TLB from being replaced by another entry when a TLB miss occurs. If the BASE_VALUE field of LOCK_REG is > 0, TLB entries from BASE_VALUE - 1 down to 0 are locked.

Frequently used memory areas or memory areas used by applications with critical real-time constraints must be loaded into these locations and the BASE_VALUE counter must be set accordingly.

The CURRENT_VICTIM counter of the LOCK_REG register specifies the location of the entry that is loaded or replaced.

A TLB miss fault does not occur if the hardware table is enabled. Not all TLB entries can be locked (a maximum of 31 entries can be locked).

2.21.3 Fault Handling

The following types of faults can occur:

- TLB miss with table walker disabled

  TLB entries can be locked from [BASE_VALUE-1] down to 0.

  No translation is found for the logical address required. If the hardware table walker is disabled, a fault is generated.

- Translation fault

  No translation is found for the logical address required (TLB miss). The hardware table walker is enabled but no page table entry exists for the requested address.

- Access permission fault

  Read or write access and access permission (AP) set to no access, or write access and AP set to read only.

When a fault occurs, an interrupt is generated to the MPU. The interrupt service routine (ISR) is then responsible for fault recovery. For example, for a TLB miss, the ISR routine might load the missing entry from the page table. The DSP is stalled by the MMU while the fault is handled.

The ISR can determine the cause of the abort interrupt by reading the F_ST_REG register. The logical address that caused the fault can be determined by reading the FAULT_AD_H_REG and FAULT_AD_L_REG registers.

Following the fault and completion of the fault handling code, the ISR releases the MMU by writing to the interrupt acknowledge register (IT_ACK_REG). The MMU then continues servicing the DSP request. The ISR can also terminate DSP operation by resetting the DSP and the MMU.

2.21.4 Initializing Locked TLB Entries

Follow this procedure to load a translation lookaside buffer manually.

1) Load the logical address, the preserved bit, and the section/page format (SLST field) to the CAM_H_REG and CAM_L_REG registers.

2) Load the physical address and the access permission bits to the RAM_H_REG and RAM_L_REG registers.

3) Update the CURRENT_VICTIM field (in the LOCK_REG register), which specifies the location of the entry to be loaded.

4) Set the LD_TLB_ITEM bit of the LD_TLB_REG register to load these values.

2.21.5 Table Walking Logic

The entire TLB can be flushed at once by setting the GLOBAL_FLUSH bit in the GFLUSH_REG register. TLB entries with a preserved bit set to 1 (bit P of the CAM_L_REG register) are not flushed.

Regardless of the preserved bit setting, a specific TLB entry can be flushed by setting the FLUSH_ENTRY bit in the FLUSH_ENTRY_REG register. The specific entry to be flushed is specified by the logical address in the CAM_H_REG and CAM_L_REG registers.

The 32 entries of the TLB may not be sufficient to store all the necessary translations for all the memory space to be accessed. In this situation, a DSP access generates a TLB miss when a logical address with no matching translation is presented to the MMU. If the hardware table walker is disabled, this miss generates an interrupt to the MPU while the DSP is held in a stalled state. The MPU system software can update the TLB with the required translation and release the DSP. A more efficient option is to enable the table walker, as described in this section.

When the hardware table walker is enabled, it automatically fills the TLB when misses occur. When a memory access is made and there is no TLB entry for the logical address, the table walker loads the missing entry from the page table stored in system (OMAP) memory (the MMU autonomously performs the memory access and no intervention of the processor is required).

The OMAP memory area must be initialized with the level 1 and level 2 section and page descriptors. The translation table base (TTB) address, in the TTB_H_REG and TTB_L_REG registers, must also be initialized with the (physical) address of this OMAP memory area before enabling the MMU.

When a TLB miss occurs, a level 1 descriptor is read based on the logical address and the value of the TTB register. The read value provides information to the MMU about the page (page size, protection, and upper bits of the physical address). If the page size read is not a section but a coarse page or fine page, a level 2 descriptor is loaded. The level 1 descriptor address field and the middle part of the logical address contain the address where this level 2 descriptor is read. Once the value of this second level descriptor is read, the MMU finally builds the TLB entry.

The TLB entry is written into one of the 32 TLB lines (CAM/RAM), replacing a randomly chosen existing entry.

2.21.6 Boot

After reset, the TLB is empty, the MMU is disabled, and the DSP is held in reset.

The fields of LOCK_REG must be initialized from the MPU before enabling the MMU. TLB entries can then be initialized as needed, starting from entry zero and incrementing CURRENT_VICTIM in LOCK_REG as each entry is written. The BASE_VALUE field in LOCK_REG can be set to lock the initialized TLB entries if the table walker is to be enabled.

If required, the TTB is then programmed and the table walker enabled.

The MMU then can be released from reset and enabled (a single write to CNTL_REG) and the DSP can be released from reset.

If some TLB entries are not initialized (and the CURRENT_VICTIM counter has not reached 31), the table walker fills the TLB based on misses until CURRENT_VICTIM = 31. The random replacement algorithm is then activated for subsequent misses. When disabled, the MMU performs no translation and builds the 32-bit OMAP address, setting bits 31-24 to 0 and bits 23-0 to the DSP byte address.

2.22 DSP MMU Registers

All DSP MMU registers are 16 bits wide and are accessed at the given offset from the base address of the MMU in the OMAP physical address space:

- Bits marked as unused must be written as zero and return an unpredictable value when read.
- The value returned from a read of a bit marked as not readable is unpredictable.
- Writes to bits marked as not writeable are ignored.

Table 2–203 lists the DSP MMU registers. Table 2–204 through Table 2–223 describe the register bits.

Table 2–203. DSP MMU Registers

Name	Description	R/W	Offset
WALKING_ST_REG	Status	R	004
CONTROL_REG	Control	R/W	008
FAULT_AD_H_REG	MSB fault address	R	00C
FAULT_AD_L_REG	LSB fault address	R	010
FAULT_ST_REG	Fault status	R	014
IT_ACK_REG	Interrupt acknowledge	W	018
TTB_H_REG	MSB TTB	R/W	01C
TTB_L_REG	LSB TTB	R/W	020
LOCK_REG	Lock counter	R/W	024
LD_TLB_REG	Load entry in TLB	R/W	028
CAM_H_REG	MSB of CAM entry	R/W	02C
CAM_L_REG	LSB of CAM entry	R/W	030
RAM_H_REG	MSB of RAM entry	R/W	034
RAM_L_REG	LSB of RAM entry	R/W	038
GFLUSH_REG	Global flush	R/W	03C
FLUSH_ENTRY_REG	Flush 1	R/W	040
READ_CAM_H_REG	MSB read CAM	R	044
READ_CAM_L_REG	LSB read CAM	R	048
READ_RAM_H_REG	MSB read RAM	R	04C
READ_RAM_L_REG	LSB read RAM	R	050

Table 2–204. Status Register (WALKING_ST_REG)

Bit	Name	Function	Reset Value
15:2	Reserved		-
1	WALK_WORKING	When 1, the table walking logic is handling a page miss.	0
0	Reserved		

Table 2–205. Control Register (CNTL_REG)

Bit	Name	Function	Reset Value
15:3	Reserved		
2	WTL_EN	Enables the table walking logic When active, writes to LD_TLB_REG, TTB_H_REG, TTB_L_REG, and LOCK_REG have undefined effect. Active high	0
1	MMU_EN	Enables the MMU When the MMU is disabled, the logical addresses are re-mapped into CS0 from 0000:0000 to 00FF:FFFF. Active high	0
0	RESET_SW	Resets the module This bit must be set manually before activating the MMU. Active low	0

Table 2–206. MSB Fault Address Register (FAULT_AD_H_REG)

Bit	Name	Function	Reset Value
15:8	Reserved		-
7:0	FAULT_ADDRESS_MSB	MSB of logical address of the access that generated a permission fault	0x00

Table 2–207. LSB Fault Address Register (FAULT_AD_L_REG)

Bit	Name	Function	Reset Value
15:0	FAULT_ADDRESS_LSB	LSB of logical address of the access that generated a permission fault	0x0000

Table 2–208. Fault Status Register (FAULT_ST_REG)

Bit	Name	Function	Reset Value
15:4	Reserved		-
3	PREFETCH_ERR	This error occurs during a prefetch. Active high	0
2	PERM_FAULT	Permission fault Active high	0
1	TLB_MISS	TLB miss (when WTB is disabled) Active high	0
0	TRANS_FAULT	Translation fault (invalid descriptor) Active high	0

The fault status registers contain the logical address from the DSP that generated the interruption.

Table 2–209. Interrupt Acknowledge Register (IT_ACK_REG)

Bit	Name	Function	Reset Value
15:1	Reserved		-
0	IT_ACK	Writing 1 to this register acknowledges the interrupt.	0

Table 2–210. MSB TTB Register (TTB_H_REG)

Bit	Name	Function	Reset Value
15:0	TTB_H_REG	MSB of TTB	0x0000

Table 2–211. LSB TTB Register (TTB_L_REG)

Bit	Name	Function	Reset Value
15:7	TTB_L_REG	LSB of TTB	0x000
6:0	Reserved		-

Table 2–212. Lock Counter Register (LOCK_REG)

Bit	Name	Function	Reset Value
15	Reserved		0
14:10	BASE_VALUE	Locked entries base value. 0 ≤ BASE_VALUE	0x00
9	Reserved		0
8:4	CURRENT_VICTIM	Current entry pointed by the hardware table walking logic Base value ≤ 31	0x00
3:0	Reserved		0x0

Table 2–213. Load Entry in TLB Register (LD_TLB_REG)

Bit	Name	Function	R/W	Reset Value
15:1	Reserved			-
0	LD_TLB_ITEM	Load data in TLB Toggle bit Loaded when 1 is written Always 0 when read. Automatically reset.	R/W	0

Table 2–214. MSB of CAM Entry Register (CAM_H_REG)

Bit	Name	Function	R/W	Reset Value
15: 3	Reserved			-
2:0	VA_tag_l1_H	Table index level 1 MSB	R/W	0x0

Table 2–215. LSB of CAM Entry Register (CAM_L_REG)

Bit	Name	Function	R/W	Reset Value
15:14	VA_TAG_L1_L	Table index level 1 LSB, which means bits (21:20) of logical address	R/W	0x0
13:4	VA_TAG_L2	Table index level 2: Tiny page: Bits 13:4 Small page: Bits 13:6, rest of bits discarded Large page: Bits 13:10, rest of bits discarded Reserved bits must be written as zero; read value is unpredictable.	R/W	0x000
3	P	Preserved bit: 0: CAM entry not preserved 1: CAM entry preserved	R/W	0
2	V	Valid bit: 0: CAM entry not valid 1: CAM entry valid	R	0
1:0	SLST	SLST: When 00 section (1MB) 01: Large pages (64KB) 10: Small page (4KB) 11: Tiny page (1KB)	R/W	0x0

Table 2–216. MSB of RAM Entry Register (RAM_H_REG)

Bit	Name	Function	Reset Value
15:0	RAM_MSB	MSB physical address	0x0000

Table 2–217. LSB of RAM Entry Register (RAM_L_REG)

Offset Address: 038

Bit	Name	Function	Reset Value
15:10	RAM_LSB	LSB physical address	0x00
9:8	AP	Access permission bits	0x0
7:0	Reserved		-

Table 2–218. Global Flush Register (GFLUSH_REG)

Bit	Name	Function	Reset Value
15:1	Reserved		-
0	GLOBAL_FLUSH	Flush all the nonprotected TLB entries when 1 is written.	0

The global flush register flushes the nonprotected TLB entries. Flushing the entire TLB does not change the base value and the victim counter of the lock counter register.

Table 2–219. Flush One Register (FLUSH_ENTRY_REG)

Bit	Name	Function	Reset Value
15:1	Reserved		-
0	FLUSH_ENTRY	Flush the TLB entry pointed by the logical address in CAM_H_REG and CAM_L_REG registers (even if this entry is set protected). Active high	0

Table 2–220. MSB Read CAM Register (READ_CAM_H_REG)

Bit	Name	Function	Reset Value
15:3	Reserved		-
2:0	VA_TAG_L1_H	Table index level 1 MSB	0x0

Table 2–221. LSB Read CAM Register (READ_CAM_L_REG)

Bit	Name	Function	Reset Value
15:14	VA_TAG_L1_L	Table index level 1 LSB, which is bits (21:20) of logical address	0x0
13:4	VA_TAG_L2	Table index level 2: Tiny page: Bits 13:4 Small page: Bits 13:6, rest of bits discarded Large page: Bits 13:10, rest of bits discarded Reserved bits must be written as zero; read value is unpredictable.	0x000
3	P	Preserved bit: 0: CAM entry not preserved 1: CAM entry preserved	0
2	V	Valid bit: 0: CAM entry not valid 1: CAM entry valid	0
1:0	SLST	SLST: 00: Section (1MB) 01: Large pages (64KB) 10: Small page (4KB) 11: Tiny page (1KB)	0x0

Table 2–222. MSB Read RAM Register (READ_RAM_H_REG)

Bit	Name	Function	Reset Value
15:0	RAM_MSB	MSB physical address	0 x0000

Table 2–223. LSB Read RAM Register (READ_RAM_L_REG)

Bit	Name	Function	Reset Value
15:8	RAM_LSB	LSB physical address	0x00
7:0	Reserved		-

2.23 Additional MPU Components

The MPU subsystem also contains the following components:

- Clock generation and system reset management (CLKRST). For more details see Chapter 5, *Clock Generation and Reset Management*.
- Interrupt handler (INTH). For more details see Chapter 6, *Interrupt Handler*.
- System DMA controller (DMA)

  The system direct memory access (DMA) controller transfers data between points in the memory space without intervention by the MPU. The DMA allows movements of data to and from internal memory, external memory, and peripherals to occur in the background of MPU operation. It is designed to off-load the block data transfer function from the MPU processor. For more details see Chapter 7, *System DMA Controller*.
- LCD controller (LCDC)

  The OMAP730 device includes an LCD controller that interfaces with most industry-standard LCD displays. The LCD controller operates only in single-panel mode (dual-panel mode is not supported in this version). The panel size is programmable and can be any width (line length) from 16 to 1024 pixels in 16-pixel increments. The number of lines is set by programming the total number of pixels in the LCD. The total video frame size is programmable up to 1024x1024.

  For more details see chapter 8, *LCD Controller*.
- MPU universal asynchronous receiver/transmitter (UART)

  The main features of the OMAP730 UART are:

  - Selectable UART/IrDA modes
  - Dual 64-entry FIFOs for received and transmitted data payload
  - Programmable and selectable transmit and receive FIFO levels for DMA and interrupt generation
  - Programmable sleep mode
  - Complete status reporting capabilities in both normal and sleep modes
  - Frequency prescaler values from 0 to 16383 to generate the appropriate baud rates
  - Single 48-MHz clock reference for baud setting
  - Two DMA requests, one interrupt request to system
  - UART/modem functions:
    - Baud rate from 300 bits/s up to 3.6864M bits/s
    - Autobaud between 1200 bits/s and 115.2K bits/s
    - Software/hardware flow control: Programmable Xon/Xoff characters and programmable auto-$\overline{\text{RTS}}$ and auto-$\overline{\text{CTS}}$

- Programmable serial interface characteristics: 5-, 6-, 7-, or 8-bit characters; even, odd, or no parity bit generation and detection; 1, 1.5, or 2 stop bit generation
- False start bit detection: Line break generation and detection; fully prioritized interrupt system controls; internal test and loopback capabilities; modem control functions ($\overline{CTS}$, $\overline{RTS}$, $\overline{DSR}$, $\overline{DTR}$, $\overline{RI}$, and $\overline{DCD}$).
- IrDA functions: Slow infrared (SIR: 115.2K baud), medium infrared (MIR: 0.576M baud), and fast infrared (FIR: 4M baud) operations (very fast infrared (VFIR) not supported). The width of the pulse can be either 1.6 µs or 3/16th of a single bit time. Framing error, cyclic redundancy check (CRC) error, illegal symbol (FIR), abort pattern (SIR, MIR) detection. Eight entry-status FIFO (with selectable trigger levels) available to monitor frame length and frame errors.

For more details see Chapter 9, *Universal Asynchronous Receivers Transmitters*.

❑ Universal serial bus (USB)

The OMAP730 USB OTG supports the implementation of a full-speed device fully compliant with the USB 2.0 standard and the implementation of a low-speed and full-speed host fully compliant with USB 2.0.

It provides an interface between a local host (LH) such as an MPU and the USB analog transceiver. It prepares or handles USB transactions with minimal LH intervention.

For more details see Chapter 10, *Universal Serial Bus On-The-Go*.

❑ Multimedia memory card and secure digital IO host controller (MMC/SD and 4-bit SDIO)

The main features of the OMAP730 MMC/SD/SDIO are:

- Full compliance with MMC command/response sets as defined in the MMC Standard Specification v.2.2 [1] (SPI mode extension not supported).
- Full compliance with SD command/response sets as defined in the SD Physical Layer Specification v.1.0 [2] (SPI mode extension not supported).
- Flexible architecture allowing support for new command structure
- Separate SPI interface with 4 C/S. Provides supports for up to four serial devices, such as serial flash.
- Built-in 64-byte FIFO for buffered read or write
- 16-bit wide access bus to maximize bus throughput
- Low-power design
- Wide interrupt capability

- Programmable clock generation
- Two DMA channels

For more details see Chapter 11, *Multimedia Memory Card*.

❑ Multichannel buffered serial port (McBSP)

The main features of the OMAP730 McBSP are:

- Based on TI standard multichannel buffered serial port
- Simultaneous RX and TX DMA support
- Each processor is master of its TX clock and TX data.
- Supports bit rates up to 5M bits/sec
- Has RX data overrun interrupts
- Same functionally as C5510 McBSP

For more details see Chapter 12, *Multichannel Buffered Serial Port*.

❑ NAND flash controller

The NAND-type flash memory chip contributes to rapid write and erase capabilities because data is rewritten in small increments, which leads to improved performance through continuous data recording and to other benefits.

For more detail, see Chapter 13, *NAND Flash*.

❑ Fast interintegrated circuit master/slave controller (I^2C)

External components attached to the I^2C bus can serially transmit/receive up to 8-bit data to/from the local host (LH) device through the two-wire I^2C interface (400K bits/s). This I^2C peripheral supports any slave or master I^2C-compatible device.

For more details see Chapter 14, *Interintegrated Circuit*.

❑ Enhanced audio controller (EAC):

Supports I2S, AC97, and SPDIF codecs. The enhanced audio controller (EAC) provides to an application a stand-alone audio control without any CPU (DSP or MCU) support. It allows connecting a modem and/or Bluetooth subsystem to an AC97, a PCM, or an I2S codec, while the MPU subsystem (PDA, WinCE MCU…) is in power-down mode. It also suppresses the need for two codecs (one for the modem and one for the MPU subsystems).

The EAC also provides the ability to record/play PCM (wave) files with various sample frequencies and bit-length formats without any CPU processing. It also allows using the same microphone input line for the modem's 8-kHz sampling frequency or for high-quality voice recording.

A high-quality audio file can also be mixed with the voice input channel, down-sampled, and sent to the modem/Bluetooth uplink path (wave file play during phone call). The EAC also provides the CPU the ability to get the data samples coming from the microphone path at the codec sampling frequency to process the signal and to send it back to the modem uplink path. The audio signal coming from the modem/Bluetooth downlink path

can be also processed by the CPU at 8 kHz or at the codec sampling frequency before being sent to the codec and loudspeakers.

Modem/Bluetooth inputs and/or output voice data samples can be automatically stored in the CPU memory through the DMA write channel (record the phone call).

The EAC also provides the ability to loopback the Bluetooth audio data samples to the modem uplink path.

For more details see Chapter 15, *Enhanced Audio Controller*.

- ❑ Multichannel serial interface (MCSI)

  The multichannel serial interface (MCSI) has multichannel transmission capabilities. The MCSI expands the parallel interface of an ARM926EJS to connect to external devices such as codecs, DSPs, and system simulators.

  The OMAP730 MCSI device provides full duplex transmission and master or slave clock control. All transmission parameters are configurable to cover the maximum number of operating conditions:

  - Master or slave clock control (transmission clock and frame synchronization pulse)
  - Programmable transmission clock frequency
  - Single-channel or multichannel (x16) frame structure
  - Programmable word length: 3 to 16 bits
  - Full-duplex transmission
  - Programmable frame configuration:
    - Continuous or burst transmission
    - Normal or alternate framing
    - Normal or inverted frame polarity
    - Short or long frame pulse
    - Programmable oversize frame length
    - Programmable frame length
  - Programmable interrupt occurrence time (TX and RX)
  - Error detection with interrupt generation on wrong frame length
  - DMA support for both TX and RX data transfers

  For more details see Chapter 16, *Multichannel Serial Interface*.

- Power and clock control (PCC)

  The main features of the OMAP730 PCC are:

  - Performs the transitions between the power modes (awake, big sleep, deep sleep)
  - Handles idle/wake-up handshake of MPU-S and DBB
  - Monitors wake-up events
  - Controls system clock input sources (VCTXO, 32-kHz oscillator)
  - Performs calibration of 32-kHz oscillator (gauging)
  - Manages the clocks and resets distributed to MPU-S, DBB, and to some peripherals
  - Manages the security resets and violations
  - Handles the power-up sequence
  - Manages the 32-kHz oscillator to VCTXO and VCTXO to 32-kHz oscillator clock switch
  - Handles embedded LDO with bypass possibility
  - Implements full PMT wrapper for all the analog cells embedded in it

  For more details see Chapter 17, *Power and Clock Control*.

- VLYNQ serial communications interface:

  This serial communications interface enables the extension of an internal CBA bus segment to one or more external physical devices. The VLYNQ accomplishes this function by serializing bus transactions in one device, transferring the serialized transaction between devices via a VLYNQ port, and deserializing the transaction in the external device.

  VLYNQ ID = 0x000E for OMAP730

  The main features of the OMAP730 VLYNQ are:

- Low pin count (as few as three signals)
- No 3-state signals
  - All signals are dedicated and driven by only one device.
  - Provides significant reduction in I/O timing analysis complexity
  - Allows support of high speed PHYs
- Scalable performance/support for different PHY technologies
  - TTL @ 150 MHz and 1, 2, 4 or 8 bits for TX and RX data
  - LVDS @ 622 MHz and 1, 2, 4 or 8 bits for TX and RX data
  - Gandalf @ 3.5 GHz and one bit for TX and RX data
  - Performance with any given PHY increases linearly as the data port width increases.

- Simple packet-based transfer protocol for memory-mapped access
  - Write request/data packet
  - Read request packet
  - Read response data packet
  - Interrupt request packet
- Symmetric operation
  - TX pins on first device connect to RX pins on second device and vice versa.
  - Request packets, response packets, and flow control information are all multiplexed and sent across the same physical pins.
  - Supports both host/peripheral and peer-to-peer communication models
  - Able to emulate all currently-used peripheral interface mechanisms
- Simple block code packet formatting (8b/10b)
- Supports in-band flow control
  - No extra pins required
  - Allows receiver to momentarily throttle back transmitter when overflow is about to occur
  - Uses built-in special code capability of block code to seamlessly interleave flow control information with user data
  - Allows system designers to balance cost of data buffering versus performance
- Supports multiple outstanding transactions
- Automatic packet formatting optimizations
- Internal loopback mode
- SHA-1/MD5 accelerator:

  The SHA-1/MD5 security module provides hardware-accelerated hash functions. It can run either the SHA-1 algorithm in compliance with FIPS 180-1 standard or the MD5 message-digest algorithm developed by Rivest in 1991. Up to 2^{20}-1 bytes (1M byte) of data can be hashed in a single operation to produce a 160-bit signature in the case of SHA-1 and a 128-bit signature in the case of MD5.

- DES/3DES:

  The DES/3DES module provides hardware accelerated data encryption/decryption functions. It can run either the single DES algorithm or the triple DES algorithm in compliance with the FIPS 46-3 standard. It supports ECB (electronic code book) and CBC (cipher block chaining) modes of operation. It does not support the CFB (cipher feedback) and the OFB (output feedback) modes of operation in hardware.

- Random number generator (RNG):

  The RNG module provides a true, nondeterministic noise source for generating keys, initialization vectors (IVs), and other random number requirements. It is designed for FIPS 140-1 compliance, facilitating system certification to this security standard. It also includes built-in self-test (BIST) logic that allows testing the randomness of the module output and its compliance with the FIPS 140-1 standard. An ANSI X9.17 annex C postprocessor is available to meet the NIST requirements of FIPS 140-1.

Chapter 3

GSM Subsystem

This chapter discusses the OMAP730 GSM-S device.

Topic **Page**

Topic **Page**

3.1 Introduction

The OMAP730 GSM subsystem (GSM-S) implements the digital baseband processes of a GSM/GPRS mobile phone. This device combines a TMS320C54x™ (C54x™) DSP with its program and data memories, a micro-controller core with emulation facilities (GSM-MPU), 8K bytes of internal boot overlay RAM memory, 2.5M bits of SRAM memory, .5M bits of SRAM memory shareable between DSP and GSM-MPU, a clock squarer cell, and several compiled single-port or dual-port RAM and CMOS gates.

The GSM is designed for future enhancements such as EDGE coprocessor connectivity.

This device can be applied in the management of the GSM/GPRS baseband processes through GSM layers 1, 2, and 3 protocols as described in the ETSI standard, with specific attention to power consumption in both GSM dedicated and idle modes, and GPRS (class 12) capability.

This device fully supports the GSM full-level test approval (FTA) for full-rate, enhanced full-rate, and half-rate speech coding. It also implements all features for the structural test of the logic (full scan, BIST, PMT, and JTAG boundary scan).

3.1.1 Features

The OMAP730 GSM-S architecture is based on two processor cores: GSM-MPU and C54x DSP, using the generic TI peripheral bus (TIPB) standard to interface with their associated application peripherals.

The OMAP730 GSM-S has the following features:

- GSM-MPU
  - ARM7TDMI CPU core (32/16-bit RISC processor)
  - MPU ICECrusher for emulation purposes
- C54x DSP
  - C54x DSP core with 28K words of RAM and 128K words of ROM
  - MPUI (8K words, part of the 28K of RAM)
  - SPI
  - Timer
- Clock squarer analog cell
- GSM-MPU peripherals
  - General-purpose peripherals:
    - GSM-MPU memory interface for external RAM, flash, or ROM
    - TIPB bridge
    - 2.5M-bit static RAM with write buffer
    - .5M-bit static RAM shareable with DSP with mutual exclusive access

- Memory protection unit (MPU)
- Debug unit (DU)
- 64K-bit static RAM with external memory overlay for internal boot
- Die-ID cell (48 bits + 5 spares)

■ Application peripherals:

- GSM-MPU general-purpose I/O
- µWire interface
- Three timers (generic1, generic 2, and watchdog)
- UART 16C750 interface (UART modem) shared with DSP:
  —Software flow control
  —Hardware flow protocol (DCD, CTS/RTS)
- SIM interface
- GSM-MPU interrupt handler (INTH)
- GSM real-time sequencer (TPU)
- GSM real-time serial port (TSP)
- DMA controller (four channels, two ports)
- Real-time clock (RTC)
- GSM ultralow-power device (ULPD)
- Clock generator and control with digital phase-locked loop (CLKM)
- Programmable controller for LED pulse generation (LPG)
- Master I^2C serial interface
- GPRS encryption algorithm modules 1 and 2

❑ ASIC DSP peripheral

■ General-purpose peripherals:

- TIPB bridge

■ Application peripherals:

- Radio interface (RIF)
- Multichannel serial interface (MCSI)
- A51/A52 ciphering (CRYPT)
- UART 16C750 interface (UART modem) shared with GSM-MPU:
  —Software-flow control
  —Hardware-flow protocol (DCD, CTS/RTS)
  —Autobauding function, local echo
- DMA controller (4 channels)
- DSP interrupt handler (INTH)

❑ Other ASIC peripherals

■ JTAG TAP controller

3.2 GSM-MPU Core

The GSM-MPU is a 32-bit RISC microcontroller core. This microprocessor works with 32-bit or 16-bit instructions and on 32-, 16- or 8-bit data.

Pipelining is employed so that all parts of the system processing and memory can operate continuously. Typically, while one instruction is being executed, its successor is being decoded and a third instruction is being fetched from memory. In the OMAP730 GSM-S device family, the GSM-MPU is intended to work in little endian mode only.

The CPU is associated with an emulation module called ICECrusher, which provides the following debug functions:

- Single-processor and multiprocessor debugging
- High-level language and assembly debugging (run, halt, step, etc.)
- Real-time (CPU continuously running) or nonreal-time (CPU stopped) debug options
- Combined 32- and 16-bit modes for MPU processor
- Endianism transparency
- Unlimited breakpoints via opcode replacement (software breakpoint)
- Two hardware breakpoints (one configurable as software breakpoint) with maskable cycle type, address, and data compare
- Two external breakpoint events
- Internal events generate external triggers
- Benchmarking/profiling capability

3.3 DSP Subchip

The DSP subchip is a digital signal processor core compliant with the TMS320C54x family. The CPU core is associated with an MPU port interface (MPUI), an interrupt handler, a parallel interface XIO, a timer, 28K words of RAM including 8K words of MPUI shared memory, 128K words of ROM, a serial port, and a JTAG interface.

The input clock frequency is delivered by an external DPLL and the functional cycle frequency is in the 0 to 91 MHz range.

3.4 CLKM

3.4.1 Functional Description

This module has three main functions:

1) Generation and control of clock distribution to the MPU core, the DSP core, and DSP internal and external peripherals

2) Control of reset signals of MPU core, TMS320C54x core, and peripherals

3) Control of the deep-power mode of external flash

3.4.1.1 Clock Management

The clocks generated are derived from either the CLKIN or VTCXO clocks.

The memory interface has the same clock as the MPU core.

The interrupt handler, GSM-MPU-to-TIPB bridge, and DMA controller clocks are identical to the MPU core clock, except in sleep mode.

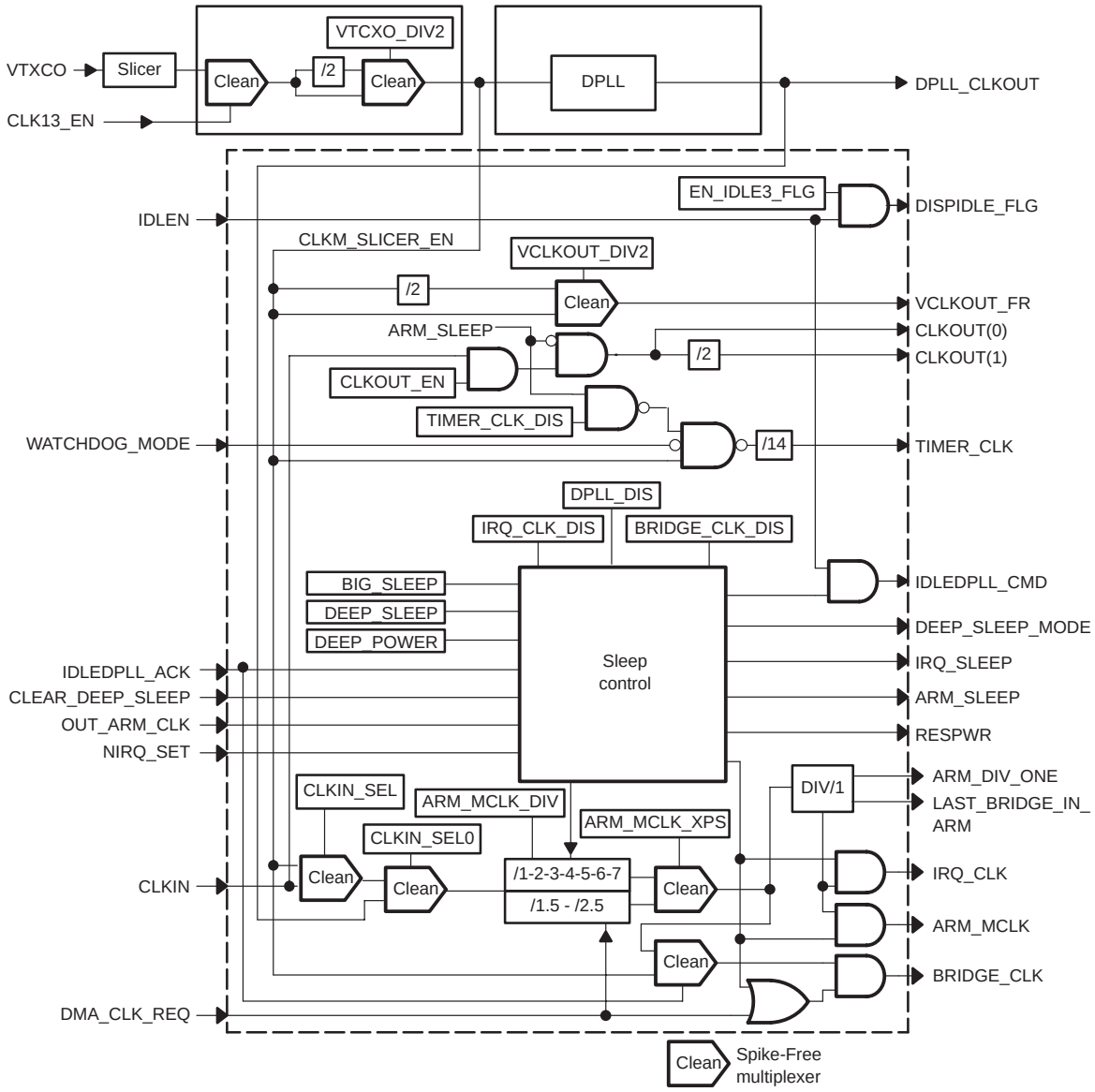
External peripherals are connected to the VCLKOUT_FR (free-running VTCXO). This clock is never shut off.

The MCSI peripheral may need a faster clock than VTCXO. Three different clocks can be selected for this peripheral: VTCXO, bridge clock, or bridge clock divided by two. A dedicated register controls this clock (programmable free-running clock).

VTCXO is shut off in deep-power mode. It is therefore necessary to filter the slicer output in order to avoid spikes during the VTCXO power up. This is done and controlled by external pin VTCXO_EN (VTCXO enable/active at level one).

A programmable (enable/disable) divide-by-two block is inserted between the slicer and the DPLL input clock. This allows keeping the same DPLL reference clock without having to change the DPLL programming model for applications running either with a 13-MHz or a 26-MHz VTCXO clock.

Figure 3–1. Clock Schematic



3.4.2 CLKM Registers

These registers are directly controlled by the internal TIPB. The clock generator is selected when CS[2:0] is equal to 5. The address of one register is equal to the start address plus the address offset. Table 3–1 lists the CLKM registers. Table 3–2 through Table 3–6 describe the register bits.

Table 3–1. CLKM Registers(FFFF:FD00)

Name	Description	Address	Size	Type	Reset Value
CNTL_ARM_CLK	MPU clock control	FFFF:FD00	16 bits	R/W	0001 1000 0001
CNTL_CLK	Clock control	FFFF:FD02	16 bits	R/W	0001 0001
CNTL_RST	Reset control	FFFF:FD04	16 bits	R/W	011?
CNTL_CLK_DSP	Controls division factor between DPLL_CLKOUT and DSP_CLK	FFFF:FD0A	16 bits	R/W	00
CNTL_CLK_ PROG_FREE	Controls free-running clock dedicated to UART modem and MCSI2	FFFF:FD0E	16 bits	R/W	00

NOTE: In addition to these control registers, the clock configuration is set through the DPLL module, which includes its own TIPB control register. The DPLL module is selected when CS[4:0] is equal to 19 (decimal).

Table 3–2. MPU Clock Control Register (CNTL_ARM_CLK)

Bit	Name	Function	Reset Value†
12	DEEP_SLEEP	Deep-sleep mode state. The ARM_MCLK clock goes to wake state if IRQ_SET = 0 and DEEP_SLEEP = 0. Reset by the CLEAR_DEEP_SLEEP input signal.	1
11:8	DEEP_POWER	t_w delay (RESPWR high to ARM_CLK restart)	0000
7		Reserved	1

† When ARM_NRESET is active

Table 3–2. MPU Clock Control Register (CNTL_ARM_CLK) (Continued)

Bit	Name	Function	Reset Value†
6:4	ARM_MCLK_DIV	Defines the division factor applied to the clock source (CLKIN, VTCXO, or DPLL_CLKOUT) to generate the MPU system clock.	000
		ARM_MCLK_XP5	
		6:4 / **0** / **1**	
		000: /1 /1.5 001: /1 /1.5 010: /2 /2.5 011: /3 /2.5 100: /4 /1.5 101: /5 /1.5 110: /6 /2.5 111: /7 /2.5	
3	ARM_MCLK_XP5	Enables the 1.5 or 2.5 division factor (2/3 duty cycle: _|‾|_|‾|_|‾) for generating the MPU system clock (from either CLKIN, VTCXO, or DPLL_CLKOUT) The division factor is 1, 2, 3, 4, 5, 6, or 7, as defined by the ARM_MCLK_DIV(6:4) vector (see next bit field). The /1.5 or /2.5 ratio is enabled and defined by bit 5 of the ARM_MCLK_DIV: ARM_MCLK_DIV(5): 0 → Division factor is 1.5. ARM_MCLK_DIV(5): 1 → Division factor is 2.5.	0
2	CLKIN_SEL	Clock-in selection: 0: VTCXO source (typically 13 MHz) 1: CLKIN input (external up to 40 MHz)	0
1	CLKIN_SEL0	Source clock for GSM-MPU system: 0: DPLL output clock 1: VTCXO or CLKIN (see CLKIN_SEL bit)	0
0	BIG_SLEEP	GSM-MPU master clock (MCLK): 0: Disable 1: Enable The MPU can set this bit to 0 in order to cut off its clock. Only IRQ_SET can set it back to 1.	1

† When ARM_NRESET is active

Table 3–3. Clock Control Register (CNTL_CLK)

Bit	Name	Function	Reset Value
7	VTCXO_DIV2	VTCXO divider: 0: VTCXO is divided by 1. 1: VTCXO is divided by 2. This bit is used to divide the VTCXO input clock by 2 before it is supplied to the DPLL, the peripheral clock generator, and the MPU clock (when selected). This bit is typically used to switch from a 13-MHz to a 26-Mhz VTCXO application without changing the DPLL programming model.	0
6	VCLKOUT_DIV2	VCLKOUT_FR divider: 0: VCLKOUT_FR is divided by 1. 1: VCLKOUT_FR is divided by 2. This bit is used to divide by 2 the system reference clock that supplies the DPLL (VTCXO control above) to generate the clocking signal for the peripherals.	0
5	EN_IDLE3_FLG	DSP idle flag control (to MPUI) 0: SAM/HOM wait-state is unchanged. 1: SAM/HOM wait-state is forced to HOM when DSP is in IDLE3.	0
4	CLKOUT_EN	CLKOUT clock control: 0: Disable 1: Enable CLKOUT[2:0] output clocks	1
3	DPLL_DIS	DPLL control: 0: DPLL is not stopped and continues to provide an output clock (according to its control content) when both DSP and MPU systems are in IDLE3 and SLEEP modes, respectively. 1: DPLL is set to IDLE mode when the DSP and MPU systems are in IDLE3 and sleep modes, respectively. DPLL restarts automatically (according to its control content) as soon as one processor wakes up. See DPLL specification for a detailed description of the DPLL idle/wake-up sequences.	0
2	TIMER_CLK_DIS	Timer clock control: 1: TIMER_CLK is disabled or enabled according to the sleep command (ARM_MCLK_EN bit). 0: TIMER_CLK is always running at ARM_CLK frequency and is never cut off.	0

Table 3–3. Clock Control Register (CNTL_CLK) (Continued)

Bit	Name	Function	Reset Value
1	BRIDGE_CLK_DIS	Bridge clock control: 1: BRIDGE_CLK is disabled or enabled according to the sleep command (ARM_MCLK_EN bit). 0: BRIDGE_CLK is always running.	0
0	IRQ_CLK_DIS	IRQ clock control: 1: IRQ_CLK is disabled or enabled according to the sleep command (ARM_MCLK_EN bit). 0: IRQ_CLK is always running at ARM_CLK frequency and is never cut off.	1

Table 3–4. Reset Control Register (CNTL_RST)

Bit	Name	Function	Reset Value
3	WATCHDOG_RESET	Watchdog reset: 0: None 1: Occurred	0
2	EXT_RESET	External peripherals reset command: 0: nReset out is high 1: nReset out is low	1
1	DSP_RESET	DSP reset command: 0: None 1: Reset DSP	1
0	RESERVED	Reserved	

Table 3–5. Clock Control Register (CNTL_CLK_DSP)

Bit	Name	Function	Reset Value
1:0	CNTL_CLK_DSP	Defines the DSP clock division: 00: DPLL clock 01: DPLL clock/1.5 10: DPLL clock/2 11: DPLL clock/3	00

Table 3–6. Free-Running Clock Control Register (CNTL_CLK_PROG_FREE_RUNNING)

Bit	Name	Function	Reset Value
1:0	CNTL_CLK_PROG_ FREE_RUNNING	Defines the free-running clock: 00: VTCXO 01: GSM-MPU_SOURCE_CLK 10: GSM-MPU_SOURCE_CLK /2 11: No clock	00

Note:

When the GSM-MPU_SOURCE_CLK is cut off, the peripherals connected to PROG_FREE_RUNNING_CLK (MCSI) do not receive a clock if the register value is different from 00. If these peripherals need a clock, this register must be set to 00 before entering a sleep mode.

3.5 Real-Time Clock (RTC)

The RTC block is an embedded real-time-clock module which is directly accessible from the TIPB interface. Its basic functions are:

- Time information (seconds/minutes/hours) coded in BCD
- Calendar information (day/month/year/day of the week) directly in BCD code up to the year 2099
- Alarm function with interrupt generation based on a periodical (second/minute/hour/day) or a precise time event in the century (second accuracy)
- 30-s time range correction
- 32-kHz oscillator frequency gauging

3.5.1 RTC Registers

Start address (hex): FFFE:1800. The address of one register is equal to the start address plus the address offset. Table 3–7 lists the RTC registers. Table 3–8 through Table 3–26 list the individual register bits.

Table 3–7. RTC Registers (FFFE:1800)

Name	Description	Address	Size	Type	Reset Value
SECONDS_REG	Seconds	FFFE:1800	8 bits	R/W	0000 0000
MINUTES_REG	Minutes	FFFE:1801	8 bits	R/W	0000 0000
HOURS_REG	Hours	FFFE:1802	8 bits	R/W	0000 0000
DAYS_REG	Days	FFFE:1803	8 bits	R/W	0000 1111
MONTHS_REG	Months	FFFE:1804	8 bits	R/W	0000 1111
YEARS_REG	Years	FFFE:1805	8 bits	R/W	0000 0000
WEEKS_REG	Weeks	FFFE:1806	4 bits	R/W	0000
RESERVED	Reserved	FFFE:1807	–		–
ALARM_SECONDS_REG	Alarm seconds	FFFE:1808	8 bits	R/W	0000 0000
ALARM_MINUTES_REG	Alarm minutes	FFFE:1809	8 bits	R/W	0000 0000
ALARM_HOURS_REG	Alarm hours	FFFE:180A	8 bits	R/W	0000 0000
ALARM_DAYS_REG	Alarm days	FFFE:180B	8 bits	R/W	0000 1111
ALARM_MONTHS_REG	Alarm months	FFFE:180C	8 bits	R/W	0000 1111
ALARM_YEARS_REG	Alarm years	FFFE:180D	8 bits	R/W	0000 0000
RESERVED	Reserved	FFFE:180E	–		–
RESERVED	Reserved	FFFE:180F	–		–
RTC_CTRL_REG	Real-time clock control	FFFE:1810	7 bits	R/W	000 0000

Table 3–7. RTC Registers (FFFE:1800)(Continued)

Name	Description	Address	Size	Type	Reset Value
RTC_STATUS_REG	RTC status	FFFE:1811	7 bits	R/W	100 0000
RTC_INT_REG	RTC control	FFFE:1812	4 bits	R/W	0000
RTC_COMP_LSB_REG	RTC compensation LSB	FFFE:1813	8 bits	R/W	0000 0000
RTC_COMP_MSB_REG	RTC compensation MSB	FFFE:1814	8 bits	R/W	0000 0000
RTC_RES_PROG_REG	RTC current supply programming	FFFE:1815	6 bits	R/W	10 0111

Table 3–8. TC Seconds Register (SECONDS_REG)

Bit	Name	Function	Reset Value
7:4	SEC1	2nd digit of seconds: Range is 0 to 5.	0x0
3:0	SEC0	1st digit of seconds: Range is 0 to 9.	0x0

Table 3–9. TC Minutes Register (MINUTES_REG)

Bit	Name	Function	Reset Value
7:4	MIN1	2nd digit of minutes: Range is 0 to 5.	0x0
3:0	MIN0	1st digit of minutes: Range is 0 to 9.	0x0

Table 3–10. TC Hours Register (HOURS_REG)

Bit	Name	Function	Reset Value
7	PM_nAM	Only used in PM_AM mode (otherwise 0) 0:AM 1: PM	0x0
6:4	HOUR1	2nd digit of hours: Range is 0 to 2.	0x0
3:0	HOUR0	1st digit of hours: Range is 0 to 9.	0x0

Table 3–11. TC Days Register (DAYS_REG)

Bit	Name	Function	Reset Value
7:4	DAY1	2nd digit of days: Range is 0 to 3.	0x0
3:0	DAY0	1st digit of days: Range is 0 to 9.	0xF

Table 3–12. TC Months Register (MONTHS_REG)

Bit	Name	Function	Reset Value
7:4	MONTH1	2nd digit of months: Range is 0 to 1.	0x0
3:0	MONTH0	1st digit of months: Range is 0 to 9.	0xF

NOTE: Usual notation is taken for month value: 01 = January, 02 = February ...12 = December

Table 3–13. TC Years Register (YEARS_REG)

Bit	Name	Function	Reset Value
7:4	YEAR1	2nd digit of years: Range is 0 to 9.	0x0
3:0	YEAR0	1st digit of years: Range is 0 to 9.	0x0

Table 3–14. TC Weeks Register (WEEKS_REG)

Bit	Name	Function	Reset Value
3:0	WEEK	1st digit of days in a week: Range is 0 to 6.	0x0

Table 3–15. TC Alarm Seconds Register (ALARM_SECONDS_REG)

Bit	Name	Function	Reset Value
7:4	ALARM_SEC1	2nd digit of seconds: Range is 0 to 5.	0x0
3:0	ALARM_SEC0	1st digit of seconds: Range is 0 to 9.	0x0

Table 3–16. TC Alarm Minutes Register (ALARM_MINUTES_REG)

Bit	Name	Function	Reset Value
7:4	ALARM_MIN1	2nd digit of minutes: Range is 0 to 5.	0x0
3:0	ALARM_MIN0	1st digit of minutes: Range is 0 to 9.	0x0

Table 3–17. TC Alarm Hours Register (ALARM_HOURS_REG)

Bit	Name	Function	Reset Value
7	ALARM_PM_nAM	Only used in PM_AM mode (otherwise 0) 0: AM 1: PM	0x0
6:4	ALARM_HOUR1	2nd digit of hours: Range is 0 to 2.	0x0
3:0	ALARM_HOUR0	1st digit of hours: Range is 0 to 9.	0x0

Table 3–18. TC Alarm Days Register (ALARM_DAYS_REG)

Bit	Name	Function	Reset Value
7:4	ALARM_DAY1	2nd digit for days: Range is from 0 to 3.	0x0
3:0	ALARM_DAY0	1st digit for days: Range is from 0 to 9.	0xF

Table 3–19. TC Alarm Months Register (ALARM_MONTHS_REG)

Bit	Name	Function	Reset Value
7:4	ALARM_MONTH1	2nd digit of months: Range is from 0 to 1.	0x0
3:0	ALARM_MONTH0	1st digit of months: Range is from 0 to 9.	0xF

Table 3–20. TC Alarm Years Register (ALARM_YEARS_REG)

Bit	Name	Function	Reset Value
7:4	ALARM_YEAR1	2nd digit of years: Range is from 0 to 9.	0x0
3:0	ALARM_YEAR0	1st digit of years: Range is from 0 to 9.	0x0

Table 3–21. Real-Time Clock Control Register (RTC_CTRL_REG)

Bit	Name	Function	Reset Value
6	nDELTA_OMEGA	Analog baseband type: 0: ABB circuit is TWL3014. 1: ABB circuit is TWL3012.	0x0
5	SET_32_COUNTER	32K counter: 0: No action 1: Set the 32-kHz counter with COMP_REG value (see note)	0x0
4	TEST_MODE	Functional/test mode: 0: Functional mode 1: Test mode (autocompensation is enabled when the 32-kHz counter reaches its end)	0x0
3	MODE_12_24	Hour mode: 0: 24 hours mode (see note) 1: 12 hours mode (PM-AM mode)	0x0
2	AUTO_COMP	Autocompensation: 0: Disabled 1: Enabled	0x0

Table 3–21. Real-Time Clock Control Register (RTC_CTRL_REG) (Continued)

Bit	Name	Function	Reset Value
1	ROUND_30S	Update: 0: No update 1: When a one is written, the time is rounded to the closest minute (see note).	0x0
0	STOP_RTC	RTC control: 0: Freeze 1: Run	0x0

NOTE: SET_32_counter must only be used when the RTC is frozen. ROUND_30S bit is a toggle bit; GSM-MPU can only write 1, and RTC clears it. If the GSM-MPU sets the ROUND_30S bit and then reads it, the GSM-MPU reads 1 until the round to the closet minute is performed at the next second. MODE_12_24: Switching between the two modes can be done at any time without disturbing the RTC; read or write is always performed with the current mode.

Table 3–22.Real-Time Clock Interrupt Register (RTC_INT_REG)

Bit	Name	Function	Reset Value
3	IT_ALARM	Enables one interrupt when the alarm value is reached (TC ALARMs) by the TCs	0x0
2	IT_TIMER	Periodic interrupt: 0: Disabled 1: Enabled	0x0
1:0	EVERY	Periodic interrupt frequency: 0: Second 1: Minute 2: Hour 3: Day	0x0

NOTE: The GSM-MPU must respect the BUSY period to prevent spurious interrupt.

Table 3–23. Real-Time Clock Status Register (RTC_STATUS_REG)

Bit	Name	Function	Access	Reset Value
7	POWER_UP	Reset status: 0: None 1: Occurred	R/W	1
6	ALARM	Alarm interrupt: 0: None 1: Has been generated	R/W	0
5	1D_EVENT	One day has occurred	R	0x0
4	1H_EVENT	One hour has occurred	R	0x0
3	1M_EVENT	One minute has occurred	R	0x0

Table 3–23. Real-Time Clock Status Register (RTC_STATUS_REG) (Continued)

Bit	Name	Function	Access	Reset Value
2	1S_EVENT	One second has occurred	R	0x0
1	RUN	RTC status: 0: Frozen 1: Running	R	0x0
0	BUSY	Event update: 0: Updating event in more than 15 µs 1: Updating event	R	0x0

NOTE: The alarm interrupt keeps its low level until the GSM-MPU writes a 1 to the ALARM bit of the RTC_STATUS_REG register.
The timer interrupt is a low-level pulse (15-µs duration).
The RUN bit shows the real state of the RTC because the STOP_RTC signal was resynchronized on 32-kHz clock: the action of this bit is delayed.
POWER_UP is set by a reset and cleared by writing a 1 to this bit.

Table 3–24. Real-Time Clock Compensation MSB Register (RTC_COMP_MSB_REG)

Bit	Name	Function	Reset Value
7:0	RTC_COMP_MSB	Number of periods to be added to the counter every hour	0x00

Table 3–25. Real-Time Clock Compensation LSB Register (RTC_COMP_LSB_REG)

Bit	Name	Function	Reset Value
7:0	RTC_COMP_LSB	Number of periods to be added to the counter every hour	0x00

NOTE: This register must be written in twos-complement format. This means:
- To add one 32-kHz oscillator period every hour, the GSM-MPU must write FFFF into RTC_COMP_MSB_REG and RTC_COMP_LSB_REG.
- To remove one 32-kHz oscillator period every hour, the GSM-MPUmust write 0001 into RTC_COMP_MSB_REG and RTC_COMP_LSB_REG.
- The 7FFF value is forbidden.

Table 3–26. Real-Time Clock Current Supply Programming Register (RTC_RES_PROG_REG)

Bit	Description	Reset C05
7:6	Reserved	–
5:3	Control analog switches from R2 to FB	0x4
2:0	Control analog switches from R1 to R3	0x7

RTC_RES_PROG_REG is used to set a current supply by switching resistors. This current supply is used to start and keep quartz oscillations. Table 3–27 describes the individual register bits.

Table 3–27. Current/Resistance Value for OMAP730 GSM-S C05

Crystal Register Value (Hex)	Programmed Resistor Value (kΩ)	Oscillator Current
2F	0	
27	57	Default
2E	80	
26	137	
2C	160	
24	217	
2A	240	
22	297	
28	320	
20	377	

3.6 Configuration Registers

The address of one register is equal to the start address plus the address offset. Table 3–28 presents the configuration registers. Table 3–29 through Table 3–38 describe the individual register bits.

Table 3–28. Configuration Registers(FFFE:F000)

Name	Description	Address	Size	Type	Reset Value
PERSEUS2_GSM_DEV_ID 0	Device ID code	FFFE:F000	16 bits	R	0xB396
PERSEUS2_GSM_DEV_ID 1	Device version code	FFFE:F002	4 bits	R	0x2
DSP_CONF_REG	DSP configuration	FFFE:F004	11 bits	R/W	0x000
DIE_ID_CODE	Die ID code	FFFF:F010	64 bits	R	0x0128
MCU_ID_CODE	GSM-MPU version code	FFFF:FE00	4 bits	R	0x3
CDSP_ID_CODE	cDSP ID code	FFFF:FE02	16 bits	R	0x0128
ARM_CONF_REG	Extended GSM-MPU configuration	FFFE:F006	8 bits	R/W	0x00
ASIC_CONF_REG	ASIC Configuration	FFFE:F008	16 bits	R/W	0x0000
IO_CONF_REG	I/O selection	FFFE:F00A	12 bits	R/W	0x000
GSM_ MPU_SW_TRACE	GSM-MPU software trace	FFFE:F00E	6 bits	R/W	0x00

Table 3–29. Device ID Code Register (PERSEUS2_GSM_DEV_ID0)

Bit	Name	Function	Reset Value (Version A)	Reset Value (Version B)
15:0	PART	Device ID code part	0xB2AC	0xB396

Table 3–30. Device Version Code Register (PERSEUS2_GSM_DEV_ID1)

Bit	Name	Function	Reset Value (Version A)	Reset Value (Version B)
15:4	Reserved	–		
3:0	VERSION	Device code version	0x0	0x2

Table 3–31. DSP Configuration Register (DSP_CONF_REG)

Bit	Name	Function	Reset Value
15:11		Reserved, fixed to 0	0x00
10		0: IrDA transceiver shut down mode (SD_IRDA) 1: DSP clock out (CLKOUT_DSP)	0x0
9		0: Flash deep low power (FDP) 1: Interrupt acknowledge (nIACK)	0x0
8		0: Chip-select 3 (nCS3) 1: MCSI TXINT (DSP-INT4n)	0x0
7		0: µWire data in (SDI) 1: TPU frame interrupt (INT8n)	0x0
6		0: µWire serial clock (SCLK) 1: RIF transmit interrupt (INT1n)	0x0
5		0: µWire data out (SDO) 1: DMA interrupt (INT10n)	0x0
4		0: CTS_MODEM 1: External flag (XF)	0x0
3		0: RTS_MODEM 1: Timer output (TOUT)	0x0
2		0: Enable LMM power 1: DSP strobe signal (DSP_IOSTRBN)	0x0
1		0: Functional mode (TXIR_IRDA, nFOE, nSCS1, RXIR_IRDA, nFWE) 1: DSP address bus (DSP_A(4..0))	0x0
0		0: Keyboard interface (KBR(4..0), KBC(4..2)) 1: XDI_O(7..0) DSP data bus	0x0

Note: When configured in DSP mode, signal CTS_MODEM is internally forced to high level in order to prevent any unwanted signal level change detection in UART module when externally observing XF.

Table 3–32. Die Identification Code Register (DIE_ID_CODE)

Bit	Name	Function	Reset Value
63:47	Reserved		Unique by die
46:32	Die-ID MSB		Unique by die
31:16	Die-ID MID		Unique by die
15:0	Die-ID LSB		Unique by die

Note: The die ID value is all 1s with no laser fuses blown.

The die identification code register is a 64-bit fuse-based register.

Table 3–33. GSM-MPU Version Code Register (MCU_ID_CODE)

Bit	Name	Function	Reset Value (Version A)	Reset Value (Version B)
15:4	Reserved	Reserved		
3:0	GSM_MPU_ID	GSM-MPU ID code	0x3	0x3

Note: Only the first four bits are used in this register; the remaining bits are always cleared to 0.

Table 3–34. cDSP ID Code Register(CDSP_ID_CODE)

Bit	Name	Function	Reset Value (Version A)	Reset Value (Version B)
15:0	CDSP_ID	CDSP ID code (F7...)	0xB396	0x0128

Table 3–35. Extended GSM-MPU Configuration Register(ARM_CONF_REG)

Bit	Name	Function	Reset Value
15:8		Reserved	0x00
7		0: TSPACT4 1: nRDYMEM ready signal for external slow device	0x0
6		0: I/O(2) 1: IRQ4	0x0
5		0: TSPACT6 1: Internal memory chip-select nCS6	0x0
4		0: Serial clock (BCLKR) 1: ARMCLK	0x0
3		0: CS4 1: ADD(22)	0x0
2		0: Keyboard interface (KBC(1)) 1: GSM-MPU normal interrupt (NIRQ)	0x0
1		0: Keyboard interface (KBC(0)) 1: GSM-MPU fast interrupt (NFIQ)	0x0
0		Reserved	0x0

Note: To maintain the RIF receive path when configured in GSM-MPU extended mode, signal BCLKR must be internally generated from the RIF transmit clock by selecting either the digital loopback mode or the clock loop mode in RIF register SPCR_REG.

Table 3–36. ASIC Configuration Register (ASIC_CONF_REG)

Bit	Name	Function	Reset Value
15		0: TSPACT5 1: DPLL clock output	0x0
14		0: SPI configuration clock RX (falling edge) 1: SPI configuration clock RX (rising edge)	0x0
13		0: RIF configuration clock RX (falling edge) 1: RIF configuration clock RX (rising edge)	0x0
12		0: µWire data in (SDI) 1: Data of I^2C (SDA)	0x0
11		0: TSPEN3 1: µWire chip-select (nSCS2)	0x0
10		0: I/O(3) 1: SIM RnW signal (SIM_RnW)	0x0
9		0: TSPACT7 1: Clock of SPI (CLKX_SPI)	0x0
8		0: ADD(21) 1: Clock of UART IRDA (CLK_16X_IRDA)	0x0
7		0: nSCS0 1: Clock of I^2C (SCL)	0x0
6		0: DSR_MODEM 1: LPG	0x0
5		Buzzer output 0: BU 1: PWT	0x0
4		Light output 0: LT 1: PWL	0x0
3		Reserved Must be fixed to 0	0x0
2		0: Reset of external peripheral (CLK13M_OUT) 1: Start bit detection (START_BIT)	0x0

Table 3–36. ASIC Configuration Register (ASIC_CONF_REG) (Continued)

Bit	Name	Function	Reset Value
1		0: I/O(1) 1: TPU idle mode (TPU_IDLE)	0x0
0		0: I/O(0) 1: TPU wait mode (TPU_WAIT)	0x0

Table 3–37. I/O Selection Register (IO_CONF_REG)

Bit	Name	Function	Reset Value
15:12		Reserved	0x0
11		0: nBLE 1: I/O(15)	0x0
10		0: nBHE 1: I/O(14)	0x0
9		0: GSM-MPUEN(2) 1: I/O(13)	0x0
8		0: MCSI_FSYNCH 1: I/O(12)	0x0
7		0: MCSI_CLK 1: I/O(11)	0x0
6		0: MCSI_RXD 1: I/O(10)	0x0
5		0: MCSI_TXD 1: I/O(9)	0x0
4		0: GSM-MPUEN(1) 1: I/O(8)	0x0
3		0: nRESET_OUT 1: I/O(7)	0x0
2		0: BCLKX 1: I/O(6)	0x0
1		0: SIM_PWCTRL 1: I/O(5)	0x0
0		0: TSPDI 1: I/O(4)	0x0

Table 3–38. GSM-MPU Software Trace Register (GSM_MPU_SW_TRACE)

Bit	Name	Function	Reset Value
15:6		Reserved	0x000
5		0: TSPACT8 1: nMREQ	0x0
4		0: RFEN 1: nOPC	0x0
3		0: TSPACT10 1: nWAIT	0x0
2		0: TSPACT11 1: MCLK	0x0
1		0: TSPACT9 1: MAS(1)	0x0
0		0: SIM_CD 1: MAS(0)	0x0

3.7 GSM-MPU Peripherals

3.7.1 Memory Interface

The GSM-MPU memory interface handles:

- External GSM-MPU memory access management
  - GSM-MPU read and write access size adaptation to the memory width (from 8 to 32 bits)
  - GSM-MPU access duration management (wait state insertion or using nREADY input) to enable the connection of slow memory devices
  - Memory control signal generation (chip-selects, write strobe generation, etc.) with up to four chip-select signals, each corresponding to an address range of 8M bytes
- GSM-MPU to MPUI memory access management
  - GSM-MPU access size adaptation for MPUI read and write access. A 32-bit MPUI read transaction is transformed into two 16-bit read accesses. A 32-bit MPUI write access is transformed into two 16-bit write transactions.
  - Address signal-timing adaptation is compliant with MPUI interface requirements.
- GSM-MPU to TIPB bridge access management
  - GSM-MPU access size adaptation for TIPB access. The access size adaptation is performed according to the TIPB peripheral minimal transaction size. If the TIPB can not handle byte transactions, then no byte write transaction can be performed.
- GSM-MPU nWAIT and access control flag generation (byte-latch, etc.)

3.7.2 Memory Interface Registers

Table 3–39 shows the memory interface register mapping. Table 3–40 through Table 3–48 describe the individual register bits.

Table 3–39. Memory Interface Register Mapping

Register Name	Address	Access	Reset Value
nCS0 memory range	FFFF:FB00	12 bits R/W	xxxx 1110 0011 1111
nCS1 memory range	FFFF:FB02	12 bits R/W	xxxx 1110 0011 1111
nCS2 memory range	FFFF:FB04	12 bits R/W	xxxx 1110 0011 1111
nCS3 memory range	FFFF:FB06	12 bits R/W	xxxx 1110 0011 1111
nCS7 memory range	FFFF:FB08	12 bits R/W	xxxx 0011 0000 0000
CS4 memory range	FFFF:FB0A	12 bits R/W	xxxx 1110 0011 1111
nCS6 memory range	FFFF:FB0C	12 bits R/W	xxxx 0011 0000 0000

Table 3–39. Memory Interface Register Mapping (Continued)

Register Name	Address	Access	Reset Value
API-TIPB control register	FFFF:FB0E	7 bits R/W	xxxx xxxx x000 0000
Extra control register	FFFF:FB10	9 bits R/W	xxxx x?10 xx00 0000

Table 3–40 describes the individual bits of the nCS0 memory range register.

Table 3–40. nCS0 Memory Range Register

Bit	Name	Function	Reset (Hex)
11:9	DC	Dummy cycles that must be inserted while bank switching from nCS0 to a faster memory bank	0x7
8	–	Reserved	0x0
7	WE	0: Abort when write operation on an nCS0 1: Write enable	0x0
6:5	DVS†	Device data size:‡ 00: 8 bits, 01: 16 bits, 10: 32 bits	0x1
4:0	WS	Number of wait states for nCS0 memory access	0x1F

† Read-only. These values are sampled at system reset on CS5 for BIGEND and nCS4 and nCS3 for DVS.
‡ CS0 device data size. This field is read-only. Its value is sampled at system reset on ROMSIZE pins.

Table 3–41 describes the individual bits of the nCS1 memory range register.

Table 3–41. nCS1 Memory Range Register

Bit	Name	Function	Reset (Hex)
11:9	DC	Dummy cycles that must be inserted while bank switching from nCS1 to a faster memory bank	0x7
8	–	Reserved	0x0
7	WE	0: Abort when write operation on an nCS1 1: Write enable	1
6:5	DVS	Device data size:† 00: 8 bits, 01: 16 bits, 10: 32 bits	1
4:0	WS	Number of wait states for nCS1 memory access	0x1F

† CS0 device data size. This field is read-only. Its value is sampled at system reset on ROMSIZE pins.

Table 3–42 describes the individual bits of the nCS2 memory range register.

Table 3–42. nCS2 Memory Range Register

Bit	Name	Function	Reset (Hex)
11:9	DC	Dummy cycles that must be inserted while bank switching from nCS2 to a faster memory bank	0x7
8	–	Reserved	0x0

Table 3–42. nCS2 Memory Range Register (Continued)

Bit	Name	Function	Reset (Hex)
7	WE	0: Abort when write operation on an nCS2 1: Write enable	0x1
6:5	DVS	Device data size:† 00: 8 bits, 01: 16 bits, 10: 32 bits	0x1
4:0	WS	Number of wait states for nCS2 memory access	0x1F

† CS0 device data size. This field is read-only. Its value is sampled at system reset on ROMSIZE pins.

Table 3–43 describes the individual bits of the nCS3 memory range register.

Table 3–43. nCS3 Memory Range Register

Bit	Name	Function	Reset (Hex)
11:9	DC	Dummy cycles that must be inserted while bank switching from nCS3 to a faster memory bank	0x7
8	–	Reserved	0x0
7	WE	0: Abort when write operation on an nCS3 1: Write enable	0x1
6:5	DVS	Device data size:† 00: 8 bits, 01: 16 bits, 10: 32 bits	0x1
4:0	WS	Number of wait states for nCS3 memory access	0x1F

† CS0 device data size. This field is read-only. Its value is sampled at system reset on ROMSIZE pins.

Table 3–44 describes the individual bits of the CS4 memory range register.

Table 3–44. CS4 Memory Range Register

Bit	Name	Function	Reset (Hex)
11:9	DC	Dummy cycles that must be inserted while bank switching from CS4 to a faster memory bank	0x7
8	–	Reserved	0x0
7	WE	0: Abort when write operation on a CS4 1: Write enable	0x1
6:5	DVS	Device data size:† 00: 8 bits, 01: 16 bits, 10: 32 bits	0x1
4:0	WS	Number of wait states for CS4 memory access	0x1F

† CS0 device data size. This field is read-only. Its value is sampled at system reset on ROMSIZE pins.

Table 3–45 describes the individual bits of the nCS6 internal memory range register.

Table 3–45. nCS6 Internal Memory Range Register

Bit	Name	Function	Reset (Hex)
11:9	DC	Dummy cycles that must be inserted while bank switching from nCS6 to a faster memory bank	0x0
8	–	Reserved	0x0
7	WE	0: Abort when write operation on an nCS6 1: Write enable	0x1
6:5	DVS	Device data size:† 00: 8 bits, 01: 16 bits, 10: 32 bits	0x2
4:0	WS	Number of wait states for nCS6 memory access	0x0

† CS0 device data size. This field is read-only. Its value is sampled at system reset on ROMSIZE pins.

Table 3–46 describes the individual bits of the nCS7 internal memory range register.

Table 3–46. nCS7 Internal Memory Range Register

Bit	Name	Function	Reset (Hex)
11:9	DC	Dummy cycles that must be inserted while bank switching from nCS7 to a faster memory bank	0x0
8	–	Reserved	0x0
7	WE	0: Abort when write operation on an nCS7 1: Write enable	0x1
6:5	DVS	Device data size:† 00: 8 bits, 01: 16 bits, 10: 32 bits	0x2
4:0	WS	Number of wait states for nCS7 memory access	0x0

† CS0 device data size. This field is read-only. Its value is sampled at system reset on ROMSIZE pins.

Table 3–47 describes the individual bits of the API-TIPB control register.

Table 3–47. API-TIPB Control Register CTRL

Bit	Name	Function	Reset
15:7	–	Reserved	0x0
6	DEBUG	Debug/visibility enable: 0: ADD frozen when no external access performed 1: MPU address can be observed on the external memory ADD bus.	0x0
5	ADAPTAPI	API access size adaptation: 1: 32-bit access transformed into two 16-bit API transactions	0x0
4	–	Reserved	0x0

Table 3–47. API-TIPB Control Register CTRL (Continued)

Bit	Name	Function	Reset
3	ADAPT1	TIPB strobe 1 access size adaptation: 0: MPU access size using the target device size 1: MPU TIPB access can be transformed into several TIPB transactions according to the MPU access size and the TIPB target data width.	0x0
2	–	Reserved	0x0
1	ADAPT0	TIPB strobe 0 access size adaptation: 0: MPU access size using the target device size 1: MPU TIPB access can be transformed into several TIPB transactions according to the MPU access size and the TIPB target data width.	0x0
0	–	Reserved	0x0

Table 3–48 describes the individual bits of the extracontrol register.

Table 3–48. Extracontrol Register

Bit	Name	Function	Reset
15:12	Reserved		–
11	DISABLE_DU	Debug unit control: 0: The debug unit is enabled and records debug data if the processor is not running the abort mode. 1: The debug unit is disabled; it can be accessed as general-purpose RAM.	0x0
10	nIBOOT_PIN	Value on the nIBOOT pin (read only)	–
9:8	IBOOT_SRC	Boot memory selection: x0: Defined by pin nIBOOT state (low = internal) 11: External memory boot 01: Internal memory boot	0x2
7:6	Reserved		–
5	CTRL_WS4	0: Accessing CS4 with programmed wait state 1: When accessing CS4, the number of waits is set to 127. Access ends when NREADY_MEM goes low. If NREADY_MEM stays at 1 after 127 wait states, an abort is generated.	0x0
4	CTRL_WS3	0: Accessing nCS3 with programmed wait state 1: When accessing nCS3, the number of waits is set to 127. Access ends when NREADY_MEM goes low. If NREADY_MEM stays at 1 after 127 wait states, an abort is generated.	0x0

Table 3–48. Extracontrol Register (Continued)

Bit	Name	Function	Reset
3	CTRL_WS2	0: Accessing nCS2 with programmed wait state	0x0
		1: When accessing nCS2, the number of waits is set to 127. Access ends when NREADY_MEM goes low. If NREADY_MEM stays at 1 after 127 wait states, an abort is generated.	
2	CTRL_WS1	0: Accessing nCS1 with programmed wait state	0x0
		1: When accessing nCS1, the number of waits is set to 127.Access ends when NREADY_MEM goes low. If NREADY_MEM stays at 1 after 127 wait states, an abort is generated.	
1	CTRL_WS0	0: Accessing nCS0 with programmed wait state	0x0
		1: When accessing nCS0, the number of waits is set to 127. Access ends when NREADY_MEM goes low. If NREADY_MEM stays at 1 after 127 wait states, an abort is generated.	
0	1WSR	Write strobe control:	0x0
		0: RnW signal stays low for half a clock cycle during 0-WS write access.	
		1: The 0-WS write accesses to the MPU memory are transformed into one WS write access to increase the write access duration.	

3.7.3 Internal Static RAM

Four megabits of static RAM (SRAM) are embedded on the die and mapped on nCS6 chip-select of the memory interface. These memories can be read or written either in 8-, 16-, or 32-bit format. The access cycle is ensured with 0 wait-state for any cycle frequency up to 39 MHz.

3.7.4 Internal Boot Memory

A 64K-bit ROM is embedded and mapped on nCS7 chip-select of the memory interface. This memory can be read either in 8-, 16-, or 32-bit format. The access cycle is ensured with 0-wait state for any cycle frequency up to 39 MHz.

Depending of the state of the nIBOOT signal, this memory is mapped on nCS0 or nCS7 to allow the CPU to boot (on reset) on external or internal memory.

3.7.5 Die ID Cell

The die ID cell is a 64-bit register composed of fuse cells programmed by laser during the fabrication process. Each die has a unique ID.

3.7.6 Interrupt Handler (INTH)

The interrupt handler provides 21 prioritized and maskable interrupts to the GSM-MPU core. It receives interrupts from both internal modules and external chip environment. Each incoming interrupt is configured as a low-level-sensitive or falling-edge-sensitive interrupt and can be individually masked using dedicated configuration registers.

An interrupt-level register is associated with each incoming interrupt to define a priority for the corresponding interrupt. If several interrupts have the same priority level, they are sent in a predefined order.

Each interrupt can be routed to one of the two input interrupts of the GSM-MPU core: fast-interrupt request (FIQ) or low-priority-interrupt request (IRQ).

3.7.7 General-Purpose I/O (GPIO)

There are fifteen I/O pins provided. They can be configured in read or write mode by internal registers.

Ten special I/O pins are dedicated to keyboard connection: five output (columns) and five input (rows). The five input pins are connected on one interruption for wake-up mode.

3.7.8 Microwire Interfaces (μWire)

The μWire interface can drive external devices such as serial EEPROMs or LCDs using two chip-selects that comply with the μWire standard.

The serial clock period is derived from the reference 13-MHz clock and can be calculated as:

$$T_{SCLK} = CK\_FREQ \times Csi\_FRQ \times T_{13M} = [2/4/7/10] \times [2/4/8] \times T_{13M}$$

Note:

The use of the μWire interface is exclusive to the use of the I^2C interface (same I/O pins).

3.7.9 Timers

The timer device implements three 16-bit timers configurable either in autoreload or in one-shot mode. These timers generate interrupts to the GSM-MPU when equal to zero.

The first timer is configured by default as a watchdog for the GSM-MPU. Programmers who do not want to have this function must write a specific sequence into a dedicated register in order to configure it as a general-purpose timer. The two other timers are general-purpose timers.

3.7.10 Universal Asynchronous Receiver/Transmitter 16C750 (UART Modem)

This UART interface is compatible with 16C750-compliant devices. The UART can be connected through a standard wired interface. The UART is primarily intended to be linked with the modem of an external PC for concurrent debugging purposes.

The module integrates two 64-word (9 and 11 bits) receive and transmit FIFOs with programmable trigger levels. The baud rate is internally generated from a programmable divisor. Transmission parity can be even, odd, or none, and the number of stop bits can be 1, 1.5, or 2.

The receiver can detect break, idle, or framing errors, FIFO overflow, and parity errors. The transmitter can detect FIFO underflow.

All modem operations can be controlled via a software interface.

3.7.11 Subscriber Identity Module (SIM) Interface

The subscriber identity module interface is fully compliant with the GSM 11.11 and ISO/IEC 7816-3 standards. Its external interface is 3-V only. 5-V adaptation is based on external level shifters.

The SIM interface supports cold and warm reset procedures and disabling of the clock (static SIM cards).

3.7.12 Serial Port Interface (SPI)

The SPI is a full-duplex serial port configurable from 1 to 32 bits. It provides three enable signals programmable either as positive or negative edge or level sensitive. The serial clock period is derived from the reference 13-MHz clock and can be calculated as:

$T_{GSM\text{-}MPUCLK} = PTV \times T_{13M} = [1/2/4/8/16] \times T_{13M}$

3.7.13 Time Processing Unit (TPU)

The TPU is a real-time sequencer dedicated to the monitoring of GSM base-band processing. Working from an event table referring to a GSM TDMA time base, the TPU activates tasks to control DSP peripherals with respect to time constraints related to the GSM sequencing.

To store the real-time microinstructions of the sequencer, the TPU includes a two-port RAM of 1024 16-bit words with dual page addressing capability. The GSM-MPU can access the full RAM in write mode only when the TPU is running.

3.7.14 Time Serial Port (TSP)

The TSP is a peripheral of the TPU which includes both a serial port (32-bit) and a parallel interface. The serial port can be programmed by the TPU with a time accuracy of a quarter of a GSM bit. The serial port is unidirectional (transmit only) when used with the TWL3012, but can be configured as bidirectional to comply with VEGA3. The serial port provides four enable signals programmable either as positive or negative edge or level sensitive.

The parallel interface allows control of 13 external individual outputs and one internal signal with a time accuracy of a quarter of a GSM bit. These parallel output signals are mainly used to control the RF activity. The single internal signal is looped on one interrupt line of the DSP in order to offer the possibility of sequencing the DSP task scheduling on a quarter a of GSM bit time accuracy event. The serial clock frequency is fixed at 6.5 MHz.

3.7.15 Direct Memory Access (DMA) Controller

The DMA controller manages data transfers between the TIPB peripherals and the DSP MPUI memory. The main features are:

- ❏ Four independent DMA channels
- ❏ Configuration of channels from the GSM-MPU only (master, direction)
- ❏ Control of each DMA channel by either the DSP or the GSM-MPU host
- ❏ Use of the GSM-MPU TIPB for all DMA transactions
- ❏ DMA transaction widths can be either 8- or 16-bit data
- ❏ Unknown transfer-length support and double-page destination buffers

3.7.16 Clock Management (CLKM)

This module controls the clock activity for the DSP, GSM-MPU, and TIPB peripherals. It includes one DPLL and one configuration register for DSP and GSM-MPU clock-frequency programming. CLKM also manages the reset of all modules connected either to the GSM-MPU TIPB or the DSP TIPB, and the scheduling of the deep power of the external flash memory.

The DPLL is programmable in multiplication mode with the following values:

$$F_{out} = F_{in} \times \frac{m}{d}$$

m = 1 to 32 (step 1)
d = 1, 2, 3, or 4

It can also be programmed in division mode with the following values:

$$F_{out} = \frac{F_{in}}{k}$$

k = 1, 2, or 4

3.7.17 Light Pulse Generator (LPG)

This peripheral produces the signal for the blinking LED. The blink period and duration are programmable.

3.7.18 I²C Master Serial Interface (I²C)

The I²C is a half-duplex serial port using two lines (data and clock) for data transmission with software-addressable external devices. The interface is compliant with the I²C Philips standard.

The main features of the interface are:

- ❏ Single master only
- ❏ Standard (100-kHz) and fast (400-kHz) transmission modes
- ❏ Supports both burst write, single read, and combined read modes
- ❏ 16-word transmit burst buffer
- ❏ 3-bit programmable spike filtering logic
- ❏ Error handling capability during I²C bus access

Note:

The use of I²C interface is exclusive to the use of µWire interface.

3.7.19 Memory Protection Unit (MPU)

Within a memory space, the MPU allows definition of memory subregions, each having a separate read/write protection attribute; this permits partitioning

of the memory space into program instruction, system data, user data, and stack.

The application program configures the MPU, which interfaces with the processor via the TIPB. The address bus directly issued by the processor is monitored, providing a real-time position of the memory region accessed. When a protection breach attempt occurs, the memory control signals are affected, not selecting the memory, and the fault condition is indicated to the processor.

The MPU allows control of:

- ❑ Up to four programmable protected regions within a memory space of 512K bytes
- ❑ A maximum protected size of 128K bytes for each region (512K bytes for four regions)
- ❑ A minimum granularity of eight bytes
- ❑ A privileged-code memory region

For each region, the memory mapped control register defines:

- ❑ Protected memory region base address
- ❑ Starting/ending addresses within the protected memory region
- ❑ Protection mode (nonuser R/W, user read-only, ROM, privileged-region write, see protection mode definition) applied to the memory subregion bounded by the starting/ending addresses
- ❑ Out-of-protection (upper-bound) indication enabled/disabled

The MPU generates:

- ❑ An illegal-access signal if the application program attempts an unauthorized access to a memory region (that is, user-write access to a region programmed for user read-only accesses)
- ❑ An out-of-protection signal when the application program attempts to read or write to a location within a memory region (defined by the base address) and above the upper limit (end address) of the protected subregion (helpful for stack overflow/underflow monitoring)
- ❑ An MPU fault signal which is the ORed combination of both illegal access and out-of-protection signals
- ❑ A fault indication (illegal access and/or out-of-protection) flagged into the MPU status register which is available to the processor for fault analysis

3.7.20 Debug Unit (DU)

The debug unit is a hardware resource intended to provide additional support to a software-abort handler. The DU provides a 64-stage deep-history table of

the last memory accesses prior to entering the abort mode, thus permitting analysis of previous bus transactions.

The DU is an autonomous function that does not need to be configured; hence, it does not interface to a control bus such as TIPB. The DU is directly connected to the processor buses (address and control) from where it collects the data and to the memory interface system where the saved history table is read.

The debug unit offers the following features:

- ❏ Sixty-four 32-bit words deep FIFO register file
- ❏ 26-bit processor address and eight processor-control signals recorded (nM[1:0], MAS[1:0], nEXEC, nOPC, nMREQ, and nRW)
- ❏ Continuous storage for every processor fetch (either instruction or data)
- ❏ Data record automatically frozen upon switching to abort mode
- ❏ Memory-like read access during abort operating mode
- ❏ Enable/disable control from input module pin (typically connected to a chip configuration-register bit)
- ❏ General-purpose RAM when the debug function is disabled

3.7.21 GPRS Encryption Algorithm (GEA1-2)

In GPRS mode, the data confidentiality is performed by a ciphering function (GPRS encryption algorithm). The ciphering is executed within the logical link control (LLC) upper layer. GEA mode 1 or 2 may be selected, depending on the option negotiated with the network.

The purpose of the LLC is to convey information between the mobile station and the serving GPRS support node. The procedures used are modeled upon the HDLC concepts. The LLC supports both acknowledge and unacknowledged modes and implements an FCS according to the mode used.

The hardware block provided supports computation of the FCS according to the LLC frame to send/receive as well as to cipher/decipher the GEA modes 1 and 2. These procedures are described in parts 01.61 and 04.64 of the European Technical Standards Institute (ETSI) GSM recommendations and detailed in GSM MoU documents.

3.7.22 Internal RAM Write Buffer (WRB)

The write buffer increases the speed of memory accesses via the write pipe mechanism (to attain a complete CPU cycle time for read access).

3.7.23 Real-Time Clock (RTC)

The RTC block is an embedded real-time clock module which is directly accessible from the TIPB interface. Its basic functions are:

- ❏ Time information (seconds/minutes/hours) coded in BCD

- Calendar information (day/month/year/day of the week) directly in BCD code up to the year 2099
- Alarm function with interrupt generation based on a periodical (second/ minute/hour/day) or a precise time event in the century (second accuracy)
- 30-s time range correction
- 32-kHz oscillator-frequency gauging

3.7.24 Ultralow-Power-Down Controller (ULPD)

The ULPD block is used to manage the deep-sleep mode. It allows stopping the accurate 13-MHz VTCXO during the discontinuous reception phases in GSM idle mode in order to lower the power consumption. Moreover, the ULPD sequencer is used to monitor the device activity in terminal-off mode, thus deriving the original GSM-mode concept to the whole terminal activity.

The main functions of the ULPD block are:

- Gauging of the 32-kHz quartz-based oscillator
- Maintenance of GSM time during deep-sleep mode with minimum time accuracy to allow a burst demodulation at wake-up
- Programmable timer to exit deep-sleep mode
- Delivery of the 13-MHz master to the CLKM module
- Switching between 13-MHz and 32-kHz clocks
- Generation of chip functional reset

3.8 Peripherals Definition

3.8.1 GPRS Encryption Algorithm (GEA 1 and 2)

In GPRS mode, the data confidentiality is performed by a ciphering function (GPRS encryption algorithm). The ciphering is executed within the LLC upper layer. GEA mode 1 or 2 may be selected according to the option negotiated with the network.

The purpose of the LLC is to exchange information between the mobile station and the serving GPRS support node. The procedures used are modeled upon the HDLC concepts. The LLC supports both acknowledge and unacknowledged modes and implements an FCS according to the mode used.

The hardware block provided supports the computation of the FCS according to the LLC frame to send/received, as well as the ciphering/deciphering of GEA modes 1 and 2. These procedures are described in GSM recommendations 01.61 and 04.64, and detailed in GSM MoU documents.

3.8.1.1 LLC Overview

The LLC is considered a sublayer of layer 2 in the ISO 7-layer model. The purpose of the LLC is to convey information between layer-3 entities in the MS and SGSN. The frame formats defined for the LLC are based upon those defined for LAPD and RLP, and the LLC procedures are modeled upon the concepts of HDLC.

The LLC supports variable-length information frames. These information frames are furnished by layer-3 protocols. The logical link control layer service access points (SAPs) are the points at which the LLC layer provides services to the layer-3 protocols. In addition to the SNDC protocol, the LLC provides services to the GPRS mobility management (GMM) protocol, and to the SMS protocol.

The LLC supports both acknowledged and unacknowledged data transfers.

3.8.1.2 Unacknowledged Operation

With this type of operation, layer-3 information is transmitted in numbered unconfirmed-information (UI) frames. The UI-frames are not acknowledged at the LLC layer. Neither error recovery nor reordering mechanisms are defined, but transmission and format errors are detected. Duplicate UI-frames are discarded.

Two modes of unacknowledged operation are defined:

- ❏ Protected mode, in which the FCS field protects the frame header and information field
- ❏ Unprotected mode, in which the FCS field protects the frame header and only the first N202 octets of the information field

3.8.1.3 Acknowledged Operation

With this type of operation, layer-3 information is sequentially transmitted in numbered information (I) frames. The I-frames are acknowledged at the LLC

layer. Error recovery and reordering procedures based on retransmission of unacknowledged I-frames are specified. Several I-frames may be unacknowledged at the same time. In case of errors that can not be corrected by the logical link control layer, a report to GPRS mobility management must be made.

3.8.1.4 Frame Format

Each LLC frame consists of a header, an information field, and an FCS.

- ❑ The header size is a minimum of 2 bytes and a maximum of 37 bytes. It is split into an address field (1 byte) and a control field (up to 36 bytes).
- ❑ The information field has a variable length with a maximum size of N201† bytes (maximum size is negotiated).
- ❑ The FCS has a fixed size of three bytes.

LLC FRAME:	H	INFORMATION FIELD	FCS

The identification of the type of frame and processing applied to an LLC frame are based upon the frame header. It is used by the GSM-MPU in order to know the FCS and ciphering configurations and to split the header and the information fields.

† N201 size depends on the SAPI chosen. The maximum size is 1520 bytes and is defined in [2].

FCS

The FCS is a 24-bit cyclic redundancy check code used to detect bit errors in frame headers and information fields. The FCS field contains the value of a CRC calculation that is performed over the entire contents of the header and information field (except for UI-frames in unprotected mode).

For UI-frames transmitted in unprotected mode, the FCS field contains the value of a CRC calculation performed over the frame header and the first N202 bytes of the information field only. The information used to calculate the CRC is referred to as the dividend. The first bit of the dividend is the highest-order term in the calculation. The CRC calculation must be performed before ciphering at the transmitting side and deciphering at the receiving side.

Note:

The CRC is the ones-complement of the sum (modulo 2) of:

—The remainder of x^k ($x^{23} + x^{22} + x^{21} + \ldots + x^2 + x + 1$) divided (modulo 2) by the generator polynomial, where k is the number of bits of the dividend

—The remainder of the division (modulo 2) by the generator polynomial of the product of x 24 by the dividend.

The CRC-24 generator polynomial is:

$$G(x) = X^{24} + X^{23} + X^{21} + X^{20} + X^{19} + X^{17} + X^{16} + X^{15} + X^{13} + X^8 + X^7 + X^5 + X^4 + X^2 + X$$

The result of the CRC calculation is placed within the FCS field defined in Section 3.7.19, with the highest-order terms transmitted first.

Ciphering

The LLC provides user data confidentiality by means of a ciphering function. This ciphering is applied on the information field and on the FCS, but not on the frame header. There are currently two ciphering algorithms defined by the ETSI. Both of them are supported by the GEA module.

These two algorithms require the following input variables:

- Kc
  The ciphering key (Kc) is generated in the GPRS authentication and key management procedure. The length of the key is 64 bits.

- Input
  This is the LLC frame-dependent input parameter (32 bits) for the ciphering algorithm. It is also called the message key.

- Direction
  This defines the direction (1 bit) of the data transmission (uplink/downlink). Set to 0 if the direction of the LLC frame transmission is from the MS to the SGSN. Set to 1 if the direction of the LLC frame transmission is from the SGSN to the MS.

Kc is received by the mobile from the GMM and is valid for all the communication, whereas the mobile regularly computes the ciphering input.

First and second GEA algorithms use the same input variables; they only differ in their internal processing.

UI Frames

There are two modes of FCS computation of the UI-frames.

- The first mode is the classical one where FCS is applied on the header plus all the information bits (protected mode)

- The second mode consists of applying the FCS only to the header and the N202 information bits (unprotected mode). If the length of the information field is less than N202 octets, then the FCS should cover the complete information field. This solution protects only the LLC and the SNDCP headers but not the information bits.

In order to improve data transfers when unprotected mode and no ciphering are selected (second mode), the GSM-MPU has two possibilities:

- The GSM-MPU writes the LLC-PDU header and the following N202 information bytes to the input buffer. The size of the LLC-PDU given to the module is equal to LLC-PDU header + N202 size. The module computes the FCS only on these bytes and returns its result after the last N202 byte. This method is optimal for CPU load used on data transfer.

- The GSM-MPU writes the LLC-PDU header and the full LLC-PDU information field to the input buffer. The LLC-PDU size given to the module is

equal to LLC-PDU header + LLC-PDU information field. However, the GSM-MPU specifies the use of the nonprotected mode on the module. Therefore, the module computes its FCS only on LLC-PDU header+N202 (see the following note) bytes. This method is optimal whether or not the GSM-MPU is using the same data flow for protected mode. In both cases, the same buffer formats are used and first byte shifting is still applied when specified.

Note:

The maximum number of octets in the layer-3 unit data PDU header (N202) is an LLC layer parameter. The maximum value for N202 must be 5 for LLC version number 0.

GEA Processing

The GSM-MPU transfers, through the TIPB, part or the entire LLC frame to the module memory. The module can then start its processing. The GSM-MPU can write part of an uplink frame when this is processed, write a downlink frame, and then write the end uplink frame. To enable this when the processing of a frame is not finished, some registers are saved (that is, X, Y, Z registers needed for the encryption algorithm and the number of bytes processed) in order to restart processing from where it was stopped.

3.8.1.5 Memory Management

All external access to the memory inside the GEA module is performed through the TIPB. From the outside the memory is seen as a register. The address increment is managed by the GEA module.

Management is performed using three address pointers. The write pointer is incremented each time a word is written to the data register. Once all the words are written and the START bit is activated, the data is processed by the GEA module and the processed pointer is incremented. When all the data are processed, the working signal goes down and a maskable interrupt is sent. The data can then be read by the GSM-MPU and the read pointer is incremented each time a word is read. It is not mandatory to read the data. The GSM-MPU can start writing again and the read pointer takes the *processed value*. The encryption module directly manages these pointers.

Access to the memory can take place in 8- or 16-bit format through two different registers: DATA16 and DATA8. Up to 1600 bytes can be written to the memory; therefore, a whole frame can be processed in one shot.

Note:

Only 1599 bytes can be written if the OS bit is set.

3.8.2 GEA Registers

The address of one register is equal to the start address plus the address offset. Table 3–49 presents the GEA registers. Table 3–50 through Table 3–71 describe the individual register bits.

Table 3–49. GEA Registers

Name	Description	Address	Size	Type	Reset Value
CNTL_REG	GEA module control	FFFF:C000	6 bits	R/W	???? ???? ??00 0011
STATUS_REG	FCS code status	FFFF:C002	2 bits	R	???? ???? ???? ??00
STATUS_IRQ_REG	Interrupt status	FFFF:C004	1 bit	R	???? ???? ???? ???0
CONF_UL_REG1	Memory configuration 1 for data uplink	FFFF:C006	8 bits	R/W	???? ???? 0000 0?00
CONF_UL_REG2	Memory configuration 2 for data uplink	FFFF:C008	16 bits	R/W	0000 0000 0000 0000
CONF_UL_REG3	Memory configuration 3 for data uplink	FFFF:C00A	16 bits	R/W	0000 0000 0000 0000
CONF_UL_REG4	Memory configuration 4 for data uplink	FFFF:C00C	16 bits	R/W	0000 0000 0000 0000
CONF_UL_REG5	Memory configuration 5 for data uplink	FFFF:C00E	16 bits	R/W	0000 0000 0000 0000
CONF_DL_REG1	Memory configuration 1 for data downlink	FFFF:C010	8 bits	R/W	???? ???? 0000 0000
CONF_DL_REG2	Memory configuration 2 for data downlink	FFFF:C012	16 bits	R/W	0000 0000 0000 0000
CONF_DL_REG3	Memory configuration 3 for data downlink	FFFF:C014	16 bits	R/W	0000 0000 0000 0000
CONF_DL_REG4	Memory configuration 4 for data downlink	FFFF:C016	16 bits	R/W	0000 0000 0000 0000
CONF_DL_REG5	Memory configuration 5 for data downlink	FFFF:C018	16 bits	R/W	0000 0000 0000 0000
KC_REG1	Ciphering key 1	FFFF:C01A	16 bits	R/W	0000 0000 0000 0000
KC_REG2	Ciphering key 2	FFFF:C01C	16 bits	R/W	0000 0000 0000 0000
KC_REG3	Ciphering key 3	FFFF:C01E	16 bits	R/W	0000 0000 0000 0000
KC_REG4	Ciphering key 4	FFFF:C020	16 bits	R/W	0000 0000 0000 0000
FCS_UL_REG1	FCS uplink 1	FFFF:C022	16 bits	R/W	0000 0000 0000 0000
FCS_UL_REG2	FCS uplink 2	FFFF:C024	16 bits	R/W	0000 0000 0000 0000

Table 3–49. GEA Registers (Continued)

Name	Description	Address	Size	Type	Reset Value
FCS_DL_REG1	FCS downlink 1	FFFF:C026	16 bits	R/W	0000 0000 0000 0000
FCS_DL_REG2	FCS downlink 2	FFFF:C028	16 bits	R/W	0000 0000 0000 0000
DATA16_REG	Data 16	FFFF:C030	16 bits	R/W	???? ???? ???? ????
DATA8_REG	Data 8	FFFF:C032	16 bits	R/W	???? ???? ???? ????

Table 3–50. Control Register (CNTL_REG)

Bit	Name	Function	Reset
15:6	Reserved	–	–
5	IT_EN	Enable interrupt: 0: No interrupt 1: Interrupt generated when processing completed	0x0
4	UL_DL	Select processing: 0: Uplink (ciphering + FCS) 1: Downlink (deciphering + FCS check)	0x0
3	CLOCK_EN	Enable the internal clock 0: Clock stopped 1: Clock enabled	0x0
2	START	Starts the module 0: The uplink/downlink state machine that is started is selected with the UL_DL bit. This bit is cleared by internal logic.	0x0

Notes:
1) NRESET_UL and NRESET_DL can be used to reset the module at any time. Setting them to 0 resets the module. NRESET_UL and NRESET_DL are resynchronized; hence, they are fully taken into account when the clock is enabled.
2) The START bit is used to start the module. All the control bits/words and buffers of the module must have been filled before. When this bit is set, the WORKING bit of STATUS_REG is set and the START bit is automatically reset.
3) The CLOCK_EN bit is used to enable the internal clock of the module. The clock must have been enabled before starting to write to the other control registers and to the data buffer.
4) The UL_DL bit is used to select the part of the module to be enabled. The GEA module can be considered split into two internal modules, that is., a module for ciphering + FCS and a module for deciphering + FCS checking. This bit is different from the D bit of the CONF_xx_REG1.
5) The IT_EN bit allows disabling the ciphering interrupt. If disabled, the GSM-MPU must poll the WORKING bit of the STATUS_REG and wait for a return to 0. The interrupt can also be masked by INTH.

Table 3–50. Control Register (CNTL_REG) (Continued)

Bit	Name	Function	Reset
1	NRESET_DL	Resets downlink module (deciphering + FCS) 0: Abort current processing and reset internal variables. This bit is set by internal logic.	0x1
0	NRESET_UL	Resets uplink module (ciphering + FCS) 0: Abort current processing and reset internal variables. This bit is set by internal logic.	0x1

Notes:
1) NRESET_UL and NRESET_DL can be used to reset the module at any time. Setting them to 0 resets the module. NRESET_UL and NRESET_DL are resynchronized; hence, they are fully taken into account when the clock is enabled.
2) The START bit is used to start the module. All the control bits/words and buffers of the module must have been filled before. When this bit is set, the WORKING bit of STATUS_REG is set and the START bit is automatically reset.
3) The CLOCK_EN bit is used to enable the internal clock of the module. The clock must have been enabled before starting to write to the other control registers and to the data buffer.
4) The UL_DL bit is used to select the part of the module to be enabled. The GEA module can be considered split into two internal modules, that is., a module for ciphering + FCS and a module for deciphering + FCS checking. This bit is different from the D bit of the CONF_xx_REG1.
5) The IT_EN bit allows disabling the ciphering interrupt. If disabled, the GSM-MPU must poll the WORKING bit of the STATUS_REG and wait for a return to 0. The interrupt can also be masked by INTH.

Table 3–51. Status Register (STATUS_REG)

Bit	Name	Function	Reset
15:2	Reserved		–
1	FCS_STATUS	FCS status (downlink only): 0: FCS good 1: FCS error	0x0
0	WORKING	Module activity: 0: Not working 1: Working	0x0

Notes:
1) The WORKING bit indicates that the module is processing. This bit is automatically enabled when the START bit of CNTL_REG has been set.
2) The FCS_STATUS bit is valid only in downlink and on the last frame of the processed LLC_PDU.

Table 3–52. Interrupt Status Register (STATUS_IRQ_REG)

Bit	Name	Function	Reset
15:2	Reserved		–
0	LLC_IT	Processing status: 0: Ongoing 1: Finished (until read) When read, the interrupt is an acknowledge	0x0

Notes: 1) The LLC_IT bit is used only when the IT_EN bit of CNTL_REG register has been enabled. When an GSM-MPU interrupt occurs, this register is checked by the GSM-MPU in order to know if the interrupt comes from the GEA module. It is automatically acknowledged when read.

2) The interrupt is mapped in the GSM-MPU as defined in the top level specification.

Table 3–53. Uplink Configuration Register 1 (CONF_UL_REG1)

Bit	Name	Function	Reset
15:8	Reserved	–	–
7	AS	GEA algorithm selection: 0: First GEA algorithm [4] 1: Second GEA algorithm [5]	0x0
6	D	Direction bit: See *Ciphering* in Section 3.8.1.4 for direction bit values.	0x0
5	F	FCS computation: 0: No FCS computed 1: FCS computed	0x0
4	PM	Protection mode: 0: FCS computed on frame header + N202 bytes 1: FCS computed on frame header + info field	0x0
3	E	Encryption: 0: No encryption 1: Encryption	0x0
2	Reserved	–	–

Notes: 1) IS bit: Input buffer shift. When set to 1, byte 0 is ignored. First byte sent to ciphering/FCS computation is byte 1.

2) OS bit: Output buffer shift. When set to 1, first byte into the output buffer is set to 0x00 and the output data are shifted by one byte.

3) E bit: Encryption mode bit. When set to 1, the information and FCS fields of the frame should be encrypted (ciphered). When set to 0, no encryption is applied on the frame.

4) PM bit: Protection mode bit. When set to 1, the FCS is calculated using both the frame header and information fields (LLC-PDU size). When set to 0, the FCS is calculated using only the frame header and the first N202 octets of the information field.

5) F bit: FCS computation bit. When set to 1, the module must compute the FCS. When set to 0, the FCS is included into the LLC-PDU data and the module must not compute it. The PM bit and N202 field are ignored when F bit is set to 0.

6) D bit: Direction bit. This bit is used as an input to the ciphering algorithm. Its value depends on which part of the network the user is located (MS or SGSN). It does not have the same use as UL_DL bit of CNTL_REG.

Table 3–53. Uplink Configuration Register 1 (CONF_UL_REG1) (Continued)

Bit	Name	Function	Reset
1	OS	Output buffer shift	0x0
		0: No action	
		1: Output frame is shifted by one byte.	
0	IS	Input buffer shift	0x0
		0: No action	
		1: Byte 0 of the LLC frame is ignored.	

Notes:
1) IS bit: Input buffer shift. When set to 1, byte 0 is ignored. First byte sent to ciphering/FCS computation is byte 1.
2) OS bit: Output buffer shift. When set to 1, first byte into the output buffer is set to 0x00 and the output data are shifted by one byte.
3) E bit: Encryption mode bit. When set to 1, the information and FCS fields of the frame should be encrypted (ciphered). When set to 0, no encryption is applied on the frame.
4) PM bit: Protection mode bit. When set to 1, the FCS is calculated using both the frame header and information fields (LLC-PDU size). When set to 0, the FCS is calculated using only the frame header and the first N202 octets of the information field.
5) F bit: FCS computation bit. When set to 1, the module must compute the FCS. When set to 0, the FCS is included into the LLC-PDU data and the module must not compute it. The PM bit and N202 field are ignored when F bit is set to 0.
6) D bit: Direction bit. This bit is used as an input to the ciphering algorithm. Its value depends on which part of the network the user is located (MS or SGSN). It does not have the same use as UL_DL bit of CNTL_REG.

Table 3–54. Uplink Configuration Register 2 (CONF_UL_REG2)

Bit	Name	Function	Reset
15:0	PDU_SIZE	LLC-PDU size in bytes	0x0000

Notes:
1) LLC-PDU size is the number of bytes of the LLC-PDU contained in the buffer (IS bit has no influence on this size).
2) If the FCS computation is disabled (F bit set to 0), then this size takes into account the LLC header, the LLC information field, and the FCS (full LLC-PDU size).
3) If the FCS computation is enabled (F bit set to 1), then this size takes into account the LLC header and the LLC information field, but not the FCS (full LLC-PDU size-3).

Table 3–55. Uplink Configuration Register 3 (CONF_UL_REG3)

Bit	Name	Function	Reset
15:8	HEADER_SIZE	LLC-PDU header size	0x00
7:0	N202	N202 variable size	0x00
		Used only when PM bit is set to 1	

Notes:
1) Header size is the number of bytes in the LLC frame header. This indicates where the information field is starting for ciphering or for unprotected mode in order to know, together with N202, the size of the data to be protected. It is not mandatory to fill it if no ciphering and protected mode are used.
2) N202 indicates the number of bytes of the information field that must be used to calculate the FCS when the PM bit is set to 0 (UI-frames only). When the length of the information field is less than N202 bytes, the FCS should cover the complete information field. When the PM bit is set to 1, the module should ignore this field. N202 is defined in [2].

Table 3–56. Uplink Configuration Register 4 (CONF_UL_REG4)

Bit	Name	Function	Reset
15:0	CIPH_IN_LSB	LSB part of the message key	0x0000

Table 3–57. Uplink Configuration Register 5 (CONF_UL_REG5)

Bit	Name	Function	Reset
15:0	CIPH_IN_MSB	MSB part of the message key	0x0000

3.8.2.1 CONF_UL_REG(4:5) Bit Mapping

Table 3–58 presents the the uplink registers (4:5) bit mapping.

Table 3–58. Uplink Configuration Register(4:5) Bit Mapping

CONF_REG4	b15	…													…	b0
CONF_REG5	b31	…													…	b16

NOTE: These two registers are used for CIPH_IN control word. CONF_UL _REG4 contains the LSB and CONF_UL _REG5 contains the MSB of this register.

Example: If CIPH_IN = 0x12345678, CONF_UL_REG4 = 0x5678 and CONF_UL_REG5 = 0x1234

3.8.3 Downlink Configuration Registers: CONF_DL_REG(1:5)

The downlink registers configure the memory for data downlink. Table 3–59 through Table 3–63 describe the individual register bits.

Table 3–59. Downlink Configuration Register 1 (CONF_DL_REG1)

Bit	Name	Function	Reset
15:8	Reserved	–	–
7	AS	GEA algorithm selection: 0: First GEA algorithm [4] 1: Second GEA algorithm [5]	0x0
6	D	Direction bit. See *Ciphering* in Section 3.8.1.4 for direction bit values.	0x0
5	F	FCS computation: 0: No FCS computed 1: FCS computed	0x0
4	PM	Protection mode: 0: FCS computed on frame header + N202 bytes 1: FCS computed on frame header + info field	0x0

Notes:
1) IS bit: Input buffer shift. When set to 1, byte 0 is ignored. First byte sent to ciphering/FCS computation is byte 1.
2) OS bit: Output buffer shift. When set to 1, first byte into the output buffer is set to 0x00 and the output data are shifted by one byte.
3) E bit: Encryption mode bit. When set to 1, the information and FCS fields of the frame should be encrypted (ciphered). When set to 0, no encryption is applied on the frame.
4) PM bit: Protection mode bit. When set to 1, the FCS is calculated using both the frame header and the information fields (LLC-PDU size). When set to 0, the FCS is calculated using only the frame header and the first N202 octets of the information field.
5) F bit: FCS computation bit. When set to 1, the module must compute the FCS. When set to 0, the FCS is included into the LLC-PDU data and the module must not compute it. The PM bit and N202 field are ignored when F bit is set to 0.
6) D bit: Direction bit. This bit is used as an input to the ciphering algorithm. Its value depends on which part of the network the user is located (MS or SGSN). It does not have the same use as UL_DL bit of CNTL_REG.

Table 3–59. Downlink Configuration Register 1 (CONF_DL_REG1) (Continued)

Bit	Name	Function	Reset
3	E	Encryption: 0: No encryption 1: Encryption	0x0
2	Reserved	–	–
1	OS	Output buffer shift: 0: No action 1: Output frame is shifted by one byte.	0x0
0	IS	Input buffer shift: 0: No action 1: Byte 0 of the LLC frame is ignored.	0x0

Notes:
1) IS bit: Input buffer shift. When set to 1, byte 0 is ignored. First byte sent to ciphering/FCS computation is byte 1.
2) OS bit: Output buffer shift. When set to 1, first byte into the output buffer is set to 0x00 and the output data are shifted by one byte.
3) E bit: Encryption mode bit. When set to 1, the information and FCS fields of the frame should be encrypted (ciphered). When set to 0, no encryption is applied on the frame.
4) PM bit: Protection mode bit. When set to 1, the FCS is calculated using both the frame header and the information fields (LLC-PDU size). When set to 0, the FCS is calculated using only the frame header and the first N202 octets of the information field.
5) F bit: FCS computation bit. When set to 1, the module must compute the FCS. When set to 0, the FCS is included into the LLC-PDU data and the module must not compute it. The PM bit and N202 field are ignored when F bit is set to 0.
6) D bit: Direction bit. This bit is used as an input to the ciphering algorithm. Its value depends on which part of the network the user is located (MS or SGSN). It does not have the same use as UL_DL bit of CNTL_REG.

Table 3–60. Downlink Configuration Register 2 (CONF_DL_REG2)

Bit	Name	Function	Reset
15:0	PDU_SIZE	LLC-PDU size in bytes with FCS	0x0000

Note: LLC-PDU size is the number of bytes of the LLC-PDU contained in the buffer (the IS bit has no influence on this size). This size takes into account the LLC header, the LLC information field, and the LLC FCS. It is used to know exactly when the end of the frame is reached. When the end of frame has been reached, all the internal variables must be reset to get ready to process a new frame.

The PDU size is not known in advance of downlink because it is only known on the last RLC/MAC block received. This is why the GSM-MPU has to set it only for the processing of the last part of the frame; otherwise it must set it to 0. The module reads it every time the start bit is set; if its value is different from 0, then this is the last part of the LLC-PDU. The PDU size must have been written before all the PDU bytes have been processed.

The GEA module never modifies CONF_DL_REG2. The GSM-MPU must set/reset it with the right values.

Table 3–61. Downlink Configuration Register 3 (CONF_DL_REG3)

Bit	Name	Function	Reset
15:8	HEADER_SIZE	LLC-PDU header size	0x00
7:0	N202	N202 variable size It is used only when PM bit is set to 1	0x00

Notes: 1) Header size is the number of bytes in the LLC frame header. It is used to know where the information field starts for ciphering or for unprotected mode in order to know, together with N202, the size of the data to be protected. It is not mandatory to fill it if no ciphering and protected mode are used.

2) N202 indicates the number of bytes of the information field that must be used to calculate the FCS when the PM bit is set to 0 (UI-frames only). When the length of the information field is less than N202 bytes, the FCS should cover the complete information field. When the PM bit is set to 1, the module should ignore this field. N202 is defined in [2].

Table 3–62. Downlink Configuration Register 4 (CONF_DL_REG4)

Bit	Name	Function	Reset
15:0	CIPH_IN_LSB	LSB part of the Ciph_in (message key)	0x0000

Table 3–63. Downlink Configuration Register 5 (CONF_DL_REG5)

Bit	Name	Function	Reset
15:0	CIPH_IN_MSB	MSB part of the Ciph_in (message key)	0x0000

3.8.3.1 CONF_DL_REG(4:5) Bit Mapping

The CONF_DL_REG(4:5) bit mapping is shown in Table 3–64.

Table 3–64. CONF_DL_REG(4:5) Bit Mapping

CONF_REG4	b15	…													…	b0
CONF_REG5	b31	…													…	b16

NOTE: These two registers are used for CIPH_IN control word. CONF_UL _REG4 contains the LSB and CONF_UL _REG5 contains the MSB of this register.

Example: If CIPH_IN = 0x12345678, then CONF_UL_REG4 = 0x5678 and CONF_UL_REG5 = 0x1234

3.8.4 Ciphering Key Registers: KC_REG(1:4)

The 64-bit ciphering key Kc is split into four 16-bit registers with the order shown in Table 3–65:

Table 3–65. KC_REG(4:5) Bit Mapping

	15															0	RESET
KC_REG1	b15	…													…	b0	0x0000
KC_REG2	b31	…													…	b16	0x0000
KC_REG3	b47	…													…	b32	0x0000
KC_REG4	b63	…													…	b48	0x0000

This variable is set by the GSM-MPU and is taken into account by the module at the start of processing of each LLC-PDU. This means that the a_kc_gprs

variable can be modified as soon as one byte of the LLC-PDU has been processed. This new value is used only on the next LLC-PDU.

3.8.5 FCS Uplink Registers: FCS_UL_REG(1:2)

This is the 24-bit FCS computed by the LLC module on the uplink LLC-PDU. It is split into two 16-bit registers. The highest-order terms of the FCS are transmitted first, so they are located in the LSB of the register. These registers contain the plain or the ciphered FCS, depending on the E bit of CONF_UL_REG1. The FCS_UL_REG(1:2) bit mapping is shown in Table 3–66.

Table 3–66. FCS_UL_REG(1:2) Bit Mapping

	15															0
FCS_UL_REG1	b8	…													…	b23
FCS_UL_REG2	0	0	0	0	0	0	0	0	b0	…					…	b7

3.8.6 FCS Downlink Registers: FCS_DL_REG(1:2)

This is the 24-bit FCS computed by the LLC module on the downlink LLC-PDU. It is split on two 16-bit registers. The highest-order terms of the FCS are received first, so the computed FCS is displayed in the same order. These registers always contain the plain FCS. The FCS_DL_REG(1:2) bit mapping is shown in Table 3–67.

Table 3–67. FCS_DL_REG(1:2) Bit Mapping

	15															0
FCS_DL_REG1	b8	…													…	b23
FCS_DL_REG2	0	0	0	0	0	0	0	0	b0	…					…	b7

NOTE: The FCS must be computed on the LLC header and information field but not on the received FCS.

3.8.7 Data Register

This register is used to send/receive data to/from the GEA module. Data are redirected to an internal buffer of the GEA module (there is only one internal buffer that can be used for uplink or downlink). This internal buffer can contain up to 1600 bytes (see Section 3.8.1.5, *Memory Management* for details).

The size of the data given to the module are automatically determined by an internal counter that counts the number of write accesses to the register before the START bit is enabled.

These data are in 16-bit word format with the order shown in Table 3–68, with bit 0 processed first.

Table 3–68. Data Register Bit Mapping

15	8	7	0
DATA_REG	DATA_MSB	DATA_LSB	

Table 3–69. Data16 Register (DATA16_REG)

Bit	Name	Function	Reset
15:8	DATA_LSB	LSB part of the 16-bit word	U
7:0	DATA_MSB	MSB part of the 16-bit word	U

NOTE: To efficiently manage this interface, the GSM-MPU is considered to be in little endian only. For example, data 0x1234 located in RAM is copied as 0x1234 into the data register (no swap) with DATA LSB equal to 0x34 and DATA MSB equal to 0x12.

DATA16_REG contains the MSB and LSB portions of the data word. Table 3–69 describes the individual register bits.

3.8.8 Frame Bit Order

An LLC frame contains the header, information field, and checksum (FCS) (see Section 2.6.5, *Frame Format*.) In uplink, the FCS may or may not be given to the module, depending on the F bit. In downlink, the FCS must always be given to the module.

This frame is split into 16-bit words that are sent to the GEA module for processing. These bits are processed in the order b0...b15b16...b32..., with b0 as the first bit of the header of the LLC frame (see previous schematic). However, FCS is given with the highest-order terms first. The LLC frame bit mapping is shown in Table 3–70.

Table 3–70. LLC Frame Bit Mapping

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
b47														…	b32
…															…
…															…
bf8	…													…	bf23
0	0	0	0	0	0	0	0	bf0	…					…	bf7

NOTE: Boldface numbers are FCS bits.

3.8.9 Frame Splitting

The LLC frames have a dynamic size (two LLC frames do not necessarily have the same size). These frames are split into RLC/MAC frames for transmission, which have a fixed size that depends on the protection chosen. The LLC frames are transmitted continuously; that is, no padding exists between two frames, and therefore an RLC/MAC frame can contain parts of more than one LLC frame.

Most of the time, the GEA processing (especially in downlink) is performed as soon as new data are received, so the processing is performed on an RLC/MAC frame basis (every 20 ms).

Uplink and downlink LLC frames are multiplexed, that is, the GPRS can start ciphering a part of an uplink LLC frame and then start to process a part of a received LLC frame, and so on. Therefore, the GEA module keeps all the static variables used for uplink or downlink in memory.

Note:

If ciphering is disabled but not FCS, the GSM-MPU must write the frame into the buffer, but it is not mandatory to read them back (no modification has been performed on these data).

Table 3–71 describes the DATA8 register bits.

Table 3–71. Data8 Register (DATA8_REG)

Bit	Name	Function	Reset
7:0	DATA	8 bits of data	U

NOTE: If an 8-bit write access is followed by a 16-bit write access, the data is continuous in the memory. Example: For an 8-bit write of 0x11, followed by a 16-bit write of 0x3322, the data is stored in memory as follows:

MSB Byte	LSB Byte
22	11
	33

3.8.10 GEA Programming Schedule

The GSM-MPU must program the module in the following order:

1) Set CLOCK bit from CNTL_REG to 1. Then wait for the return to 0. The clock is on.

2) Set RESET_UL and RESET_DL bits from CNTL_REG to 0. Then wait for the return to 1. The uplink and downlink GEA are reset.

3.8.10.1 GEA—In Uplink

1) Fill configuration registers CONF_UL_REG1-3.

2) Fill ciphering registers KC_REG1-4 (not mandatory to set them at every frame, can be performed once at the beginning) and CONF_UL_REG4-5.

3) Fill DATA_REG n times with the data to be processed (if 6 words, n = 6).

4) Set, in one shot, CNTL_REG START bit to 1, UL_DL to 0, and IT_EN to 1.

5) Wait for interrupt (Note: If IT_EN is set to 0, no interrupt occurs. The GSM-MPU must poll the WORKING bit of STATUS_REG. This bit is automatically set to 0 when the module finishes its processing.)

6) When interrupt occurs, read LLC_IT bit from STATUS_irq_REG in order to check that the GEA module has the IT.

7) If yes, read n times the processed data from DATA_REG.

8) If this is the last part of the LLC-PDU, read LLC_UL_REG1-2 registers that contain the FCS result. The clock can be disabled here or if another frame must be processed, the module is ready to restart from Step 3.

9) If this is not the last part of the LLC-PDU, go to Step 5.

3.8.10.2 GEA—In Downlink

1) Fill configuration registers CONF_DL_REG1 and 3.

2) Fill ciphering registers KC_REG1-4 (not mandatory to set it at every frame, can be performed once at the beginning) and CONF_DL_REG4-5.

3) If this is the last part of the LLC_PDU, CONF_DL_REG2 is filled with the downlink PDU size.

4) Fill DATA_REG n times with the data to be processed (if 6 words, n = 6).

5) Set, in one shot, CNTL_REG START bit to 1, UL_DL to 0, and IT_EN to 1.

6) Wait for interrupt (Note: If IT_EN set to 0, no interrupt occurs. The GSM-MPU must poll the WORKING bit of STATUS_REG. This bit is automatically set to 0 when the module finishes its processing.)

7) When interrupt occurs, read LLC_IT bit from STATUS_irq_REG in order to check that the GEA module has generated the IT.

8) If yes, read n times the processed data from DATA_REG.

9) If this is the last part of the LLC-PDU, then read LLC_UL_REG1-2 registers that contain the FCS result. The clock can be disabled here or, if another frame must be processed, the module is ready to restart from Step 3.

10) If this is not the last part of the LLC-PDU, go to Step 5.

3.9 DSP XIO to TIPB Registers

These are two 16-bit control registers: transfer rate and bridge control. Both are mapped directly into DSP I/O space. The transfer rate register is mapped at I/O space address 0xF800, and the bridge control register is mapped at I/O space address 0xF801.

Table 3–72 contains the XIO to TIPB registers. Table 3–73 and Table 3–74 describe the individual register bits.

Table 3–72. DSP TIPB Registers(XIO:F800)

Name	Description	Address	Size	Type	HWD Reset Value
TRANSFER_RATE_REG	Transfer rate	XIO:F800	16 bits	R/W	0110 0110 0110 0110
BRIDGE_CTRL_REG	Bridge control	XIO:F801	10 bits	R/W	???? ?001 0111 1111

Table 3–73. Transfer Rate Register (TRANSFER_RATE_REG)

Bit	Name	Function	Reset
15:12	TRANS_RATE_3	Duration, in half period DSP cycles, of nXSTROBE[3] The transfer rate is governed by programming the duration of nXSTROBE[3]. The programmable range is from full speed (0), which is the duration of half a DSP clock cycle, down to slowest speed (5)—duration of 8 DSP clock cycles.	0x6
11:8	TRANS_RATE_2	Duration, in half period DSP cycles, of nXSTROBE[2] The transfer rate is governed by programming the duration of nXSTROBE[2]. The programmable range is from full speed (0), which is the duration of half a DSP clock cycle, down to slowest speed (5)—duration of 8 DSP clock cycles.	0x6
7:4	TRANS_RATE_1	Duration, in half period DSP cycles, of nXSTROBE[1] The transfer rate is governed by programming the duration of nXSTROBE[1]. The programmable range is from full speed (0), which is the duration of half a DSP clock cycle, down to slowest speed (5)—duration of 8 DSP clock cycles.	0x6
3:0	TRANS_RATE_0	Duration, in half period DSP cycles, of nXSTROBE[0] The transfer rate is governed by programming the duration of nXSTROBE[0]. The programmable range is from full speed (0), which is the duration of half a DSP clock cycle, down to slowest speed (5)—duration of 8 DSP clock cycles.	0x6

Programmers must define two registers to allow correct DSP accesses to external peripherals connected to the TIPB:

- ❏ SWWSR internal register of the DSP to set I/O waits states
- ❏ TRANSFER_RATE_REG XIO register to fit DSP access to TIPB

The program sequence must be SWWSR first, and then TRANSFER_ RATE_REG.

When the number of wait states in the SWWSR register is equal to 0 or 1, the DSP peripheral accesses are not monitored anymore by READY signal.

The TRANS_RATE_x value defines the duration of the associated nXSTROBE_x. The strobe length depends on the working frequency of the TIPB. The following relations must be always satisfied:

$$\text{Strobe\_Length} \geq \frac{1}{2} \times \text{RHEA\_CLK\_Period}$$

$$\text{Strobe\_Length} = \frac{1}{2} \text{ LEAD\_CLKout} \times (\text{TRANS\_RATE} + 1)$$

$$\text{TRANS\_RATE} \geq \text{Number\_of\_WS\_in\_SWWSR}$$

The DSP can write to or read from all the bits of this register.

The transfer rate register controls the IO transfer rate.

Table 3–74. Bridge Control Register (BRIDGE_CTRL_REG)

Bit	Name	Function	Reset
15:10	–	Reserved	–
9	NSUPV	TIPB supervisor flag. This flag is used to control the access to privileged peripherals. If NSUPV is set to 0, access can be performed on privilegeds. If NSUPV is set to 1, access is forbidden.	0x0
8	TIMEOUT_ ENABLE	When set to logic 1, enable the TIPB time-out watchdog to send an NMI interrupt when the maximum TIPB access time is reached.	0x1
7:0	TIMEOUT	TIPB access time-out Limits the maximum time a peripheral can stall the processor by counting nXSTROBE cycles. When starting a transaction on the TIPB, the time-out counter is at a count of zero. If the current cycle is not finished when the counter reaches a count of time-out + 1, the transaction is aborted (by sending nXABORT to the peripheral and a NMI interrupt if TIMEOUT_ENABLE is set).	0x7F

All bits of the bridge control register can be read and written by the DSP.

3.10 MPUI Register

In order to optimize MPUI access from the host side, the DSP software must provide the following information to the GSM-MPU-to-TIPB bridge (in charge of managing access to MPUI memory) via register API_CONF.

Table 3–75 describes the DSP MPUI configuration register. Table 3–76 describes the individual register bits.

Table 3–75. DSP MPUI Configuration Register

Name	Description	Address	Size	Type	HW Reset Value
API_CONF	DSP MPUI configuration control	XIO:F900	3 bits	R/W	???? ???? ???? ?010

Table 3–76. MPUI Configuration Register (API_CONF)

Bit	Name	Function	HW Reset
2	BRIDGE_CLK_EN	0: GSM-MPU-TIPB bridge and DMA controller clock runs according to the GSM-MPU sleep mode and the DMA channel activity. 1: Forces GSM-MPU-TIPB bridge and DMA controller clock to run regardless of the GSM-MPU sleep mode.	0x0
1	API_HOM	0: MPUI is configured in SAM mode. 1: MPUI is configured in HOM mode.	0x1
0	–	Reserved. Must be 0.	0x0

The DSP software must use API_CONF during the GSM-MPU sleep mode when a DMA channel is controlled by the DSP. The control and status registers of each DMA controller channel (DMA1_CTRL and DMA2_CTRL) are only writable and readable when the DMA controller clock runs. This means that each time the DSP software accesses these registers, it must first set the BRIDGE_CLK_EN bit to 1, access these registers, and then set back BRIDGE_CLK_EN bit to 0 in order to conserve power.

3.11 MPUIC Control Register—GSM-MPU Reads From MPUIC

Table 3–77 contains the MPUIC control register. Table 3–78 describes the register bits.

Table 3–77. MPUIC Control Register (GSM-MPU Reads) Mapping

Name	Description	Address	Size	Type	HW Reset Value
MPUIC	MPUIC control	FFE0:0000	2 bits	R	0000 0000 0000 0010

The MPUIC control register contains the status and control bits for the MPUI. The MPUIC is not a memory-mapped register, but is accessed by the DSP through the bank-switching control register (BSCR). The three bits of the MPUIC are:

- SMODE: Enables the SAM or HOM (sent from DSP)
- HINT: Interrupt sent by the DSP to GSM-MPU
- DSPINT: Interrupt sent by GSM-MPU to DSP

The GSM-MPU accesses the MPUIC through the memory interface at address FFE0:0000. The SMODE bit is read on data bus input on location (1) and the HINT is read on location (3). HINT and SMODE information is synchronized on GSM-MPU access before being sent on the data bus. All other bits are tied to 0.

Table 3–78. MPUIC Control Register—GSM-MPU Reads

Bit	Name	Function	GSM-MPU	DSP	Reset
15:4	–	These bits are read as 0.	R	R	0x000
3	HINT	Host processor interrupt bit: 0: No effect 1: The GSM-MPU acknowledges and clears the interrupt. This bit enables/disables an interrupt written by the DSP to the GSM-MPU. At reset, the HINT bit is cleared. After sending the interrupt, the DSP must put the HINT bit into the inactive state (HINT=0) so that the GSM-MPU receives an interrupt pulse. Pulse duration is controlled by DSP software.	R	W	0x0
2	–	This bit is read as 0.	R	R	0x0

Table 3–78. MPUIC Control Register—GSM-MPU Reads (Continued)

Bit	Name	Function	GSM-MPU	DSP	Reset
1	SMODE	Shared access mode (SAM): 0: MPUI is configured in SAM mode. 1: MPUI is configured in HOM mode. This bit enables/disables the shared-access mode (SAM). During reset, the SMODE bit is set (HOM mode is automatically selected). After reset, the SMODE bit is cleared (SAM mode is automatically selected).	R	W	0x1
0	–	This bit is read as 0.	R	R	0x0

MPUIC Control Register—GSM-MPU Writes to MPUIC

The DSPINT is a write-only bit. Only the GSM-MPU can write to this bit. The GSM-MPU writes using the address FFE0:000 mapped in the memory interface. The other bits are not modified by the GSM-MPU. Table 3–79 contains the MPUIC control register mapping for GSM-MPU writes to MPUIC. Table 3–80 describes the individual register bits.

Table 3–79. MPUIC Control Register (GSM-MPU Writes) Mapping

Name	Description	Address	Size	Type	HW Reset Value
MPUIC	MPUIC control—GSM-MPU writes from MPUIC	FFE0:0000	1 bit	W	???? ???? ???? ?0??

Table 3–80. MPUIC Control Register—GSM-MPU Writes

Bit	Name	Function	GSM-MPU	DSP	Reset
15:3	–	These bits are not modified	R	R	–
2	DSPINT	DSP interrupt bit: 0: No interrupt is generated. 1: The GSM-MPU writes a 1 to generate a DSP interrupt (INT9). This bit enables/disables an interrupt from the GSM-MPU to the DSP. The DSPINT bit is written by the GSM-MPU; a DSP write has no effect on the DSPINT bit. The write operation is locally latched and resynchronized on DSP clocks. The DSPINT is automatically reset in MPUI logic. This interrupt can be used to wake up the DSP from an idle state or to generate an interrupt.	W	R	0x0
1:0	–	These bits are not modified.	R	R	–

3.11.1 DSP Accesses From/To MPUIC

The DSP cannot directly read from or write to the MPUIC, because the MPUIC is not a memory-mapped register. However, the SMODE and HINT bits can be read and modified by reading/modifying the SMODE and HINT bits in the BSCR. When the DSP reads from or writes to the BSCR, it is equivalent to reading from or writing to the corresponding bit in the MPUIC. The comparison between BSCR and MPUIC is as follows:

BSCR

15–12	11	10–4	3	2	1	0
BNKCMP	PS-DS	Reserved	HINT	SMODE	APIBN	EXIO

MPUIC

15–4	3	2	1	0
	HINT	DSPINT	SMODE	

3.12 DMA Mapping

The DMA controller manages access to the DSP MPUI 6K-word shared memory:

- GSM-MPU GSM-MPU
- Radio interface (RIF)
- MODEM UART
- IrDA UART

The RIF-RX and RIF-TX each have a dedicated channel. The following combinations are possible for UART:

- The UART-MODEM has two channels:
  - Channel #2: TX
  - Channel #3:RX
- The UART-IrDA has two channels:
  - Channel #2: RX
  - Channel #3: TX
- The UART-MODEM and UART-IrDA have one channel each:
  - Channel #2: UART-MODEM
  - Channel #3: UART-IrDA

The RX and/or TX are independently selectable for each module.

- The UART-MODEM has one channel:
  - Channel #2: RX or TX
- The UART-IrDA has one channel:
  - Channel #3: RX or TX
- The UARTs have no DMA.

After reset, all modules have their DMA functions disabled. Table 3–81 describes the DMA channel allocation.

Table 3–81. DMA Channel Allocation

DMA Request	Channels			
	0	1	2	3
RIF_DMA_REQ_X	✓			
RIF_DMA_REQ_R		✓		
nDMA_REQ_ARM(0) modem			✓	
nDMA_REQ_ARM(1) modem				✓
nDMA_REQ_ARM(0) IrDA				✓
nDMA_REQ_ARM(1) IrDA			✓	

NOTE: Only one peripheral at a time should be allocated to one DMA channel. The potential conflicts between concurrent DMA requests from several modules must be solved at system level with only one peripheral configured in DMA mode.

3.13 DMA Registers

The DMA registers are connected to a TIPB. The DMA controller configuration register is connected to the GSM-MPU TIPB. The DMA transfer configuration registers can be accessed by the GSM-MPU or the DSP TIPB, depending on the value of the DMA_ALLOC register.

Table 3–82 contains the DMA registers. Table 3–83 through Table 3–108 describe the individual register bits.

Table 3–82. DMA Registers (FFFF:FC00 – XIO:FC00)

Name	Description	GSM-MPU Address/DSP Address	Size	Type	Reset Value
CONTROLLER_ CONF	Configuration control	FFFF:FC00/ XIO:FC00	6 bits	R/W	11 111?
ALLOC_CONFIG	Allocation configura-tion	FFFF:FC02 XIO:FC02	4 bits	R/W	1111
DMA1_RAD	DMA1 configuration	FFFF:FC10 XIO:FC10	16 bits	R/W	0000 0000 0000 0000
DMA1_RDPTH	DMA1 TIPB buffer depth	FFFF:FC12 XIO:FC12	11 bits	R/W	000 0000 0000
DMA1_AAD	DMA1 MPUI address	FFFF:FC14 XIO:FC14	12 bits	R/W	0000 0000 0000
DMA1_ALGTH	DMA1 MPUI length	FFFF:FC16 XIO:FC16	12 bits	R/W	0000 0000 0000
DMA1_CTRL	DMA1 control	FFFF:FC18 XIO:FC18	13 bits	R/W	0 0100 1?10 0010
DMA1_CUR_ OFFSET_API	DMA1 MPUI current offset control	FFFF:FC1A XIO:FC1A	12 bits	R	0000 0000 0000
DMA2_RAD	DMA2 configuration	FFFF:FC20 XIO:FC20	16 bits	R/W	0000 0000 0000 0000
DMA2_RDPTH	DMA2 TIPB buffer depth	FFFF:FC22 XIO:FC22	11 bits	R/W	000 0000 0000
DMA2_AAD	DMA2 MPUI address	FFFF:FC24 XIO:FC24	12 bits	R/W	0000 0000 0000
DMA2_ALGTH	DMA2 MPUI length	FFFF:FC26 XIO:FC26	12 bits	R/W	0000 0000 0000
DMA2_CTRL	DMA2 control	FFFF:FC28 XIO:FC28	13 bits	R/W	0 0100 1?10 0010
DMA2_CUR_ OFFSET_API	DMA2 MPUI current offset control	FFFF:FC2A XIO:FC2A	12 bits	R	0000 0000 0000
DMA3_RAD	DMA3 configuration	FFFF:FC30 XIO:FC30	16 bits	R/W	0000 0000 0000 0000

Table 3–82. DMA Registers (FFFF:FC00 – XIO:FC00) (Continued)

Name	Description	GSM-MPU Address/DSP Address	Size	Type	Reset Value
DMA3_RDPTH	DMA3 TIPB buffer depth	FFFF:FC32 XIO:FC32	11 bits	R/W	000 0000 0000
DMA3_AAD	DMA3 MPUI address	FFFF:FC34 XIO:FC34	12 bits	R/W	0000 0000 0000
DMA3_ALGTH	DMA3 MPUI length	FFFF:FC36 XIO:FC36	12 bits	R/W	0000 0000 0000
DMA3_CTRL	DMA3 control	FFFF:FC38 XIO:FC38	13 bits	R/W	0 0100 1?10 0010
DMA3_CUR_ OFFSET_API	DMA3 MPUI current offset control	FFFF:FC3A XIO:FC3A	12 bits	R	0000 0000 0000
DMA4_RAD	DMA3 configuration	FFFF:FC40 XIO:FC40	16 bits	R/W	0000 0000 0000 0000
DMA4_RDPTH	DMA3 TIPB buffer depth	FFFF:FC42 XIO:FC42	11 bits	R/W	000 0000 0000
DMA4_AAD	DMA3 MPUI address	FFFF:FC44 XIO:FC44	12 bits	R/W	0000 0000 0000
DMA4_ALGTH	DMA3 MPUI length	FFFF:FC46 XIO:FC46	12 bits	R/W	0000 0000 0000
DMA4_CTRL	DMA3 control	FFFF:FC48 XIO:FC48	13 bits	R/W	0 0100 1?10 0010
DMA4_CUR_ OFFSET_API	DMA3 MPUI current offset control	FFFF:FC4A XIO:FC4A	12 bits	R	0000 0000 0000

3.13.1 GSM-MPU Registers

The DMA registers are connected to a TIPB. The DMA controller configuration register is connected to the GSM-MPU TIPB. The DMA transfer configuration registers can be accessed by the GSM-MPU or the DSP TIPB, depending on the value of the DMA_ALLOC register.

Table 3–83. Configuration Control Register (CONTROLLER_CONF)

Bit	Name	Function	Reset
15:6	Reserved	–	–
5	PRIORITY_ ENABLE	0: The TIPB allocation is performed using the DMA_BURST factor 1: The GSM-MPU has the same priority as the DMA transfers regarding TIPB allocation when it is in exception mode (IRQ and FIQ).	0x1

Table 3–83. Configuration Control Register (CONTROLLER_CONF) (Continued)

Bit	Name	Function	Reset
4:2	DMA_BURST	Defines the length of the DMA transfer burst. The GSM-MPU can not access the TIPB during the DMA burst.	0x1
1:0	Reserved	–	–

The configuration control register can only be accessed if the GSM-MPU is in supervisor mode. The bits can be read and written by the GSM-MPU. This register can only be accessed by the GSM-MPU, not by the DSP.

Table 3–84. Allocation Configuration Register (ALLOC_CONFIG)

Bit	Name	Function	Reset
15:4	Reserved	–	–
3	DMA_ALLOC_4	DMA channel 4 control: 0: DMA channel 4 is controlled by DSP. 1: DMA channel 4 is controlled by GSM-MPU.	0x1
2	DMA_ALLOC_3	DMA channel 3 control: 0: DMA channel 3 is controlled by DSP. 1: DMA channel 3 is controlled by GSM-MPU.	0x1
1	DMA_ALLOC_2	DMA channel 2 control: 0: DMA channel 2 is controlled by DSP. 1: DMA channel 2 is controlled by GSM-MPU.	0x1
0	DMA_ALLOC_1	DMA channel 1 control: 0: DMA channel 1 is controlled by DSP. 1: DMA channel 1 is controlled by GSM-MPU.	0x1

The allocation configuration register can only be accessed if the GSM-MPU is in supervisor mode. The bits can be read and written by the GSM-MPU. This register can only be accessed by the GSM-MPU, not by the DSP.

Table 3–85 through Table 3–90 define the channel 1 transfer parameters. The DMA transfer configuration registers are connected either to the GSM-MPU TIPB or the DSP TIPB, depending on the corresponding DMA_ALLOC flag.

Table 3–85. DMA Channel 1 Configuration Register (DMA1_RAD)

Bit	Name	Function	Reset
15:11	RHEA_CS	Peripheral chip-select	0x00
10:0	RHEA_START	TIPB buffer start address	0x000

Table 3–86. DMA1 TIPB Buffer Depth Register (DMA1_RDPTH)

Bit	Name	Function	Reset
15:11	Reserved		–
10:0	RHEA_DEPTH	TIPB buffer depth in bytes	0x000

Table 3–87. DMA1 MPUI Address Register (DMA1_ADD)

Bit	Name	Function	Reset
15:12	Reserved		–
11:0	API_START	Start address of the reception buffer in the MPUI memory The address is always expressed in bytes	0x000

Table 3–88. DMA1 MPUI Length Register (DMA1_ALGTH)

Bit	Name	Function	Reset
15:12	Reserved		–
11:0	API_LENGTH	MPUI page length in bytes	0x00

Table 3–89. DMA1 Control Register (DMA1_CTRL)†

Bit	Name	Function	Access	Reset
15:13	–	Reserved	–	–
12:11	PRIORITY	Number of additional readings on the bus: 00: Zero more reads, one reading 01: One more read, two successive readings 10: Two more reads, three successive readings 11: Three more reads, four successive readings	R/W	0x0
10	DIRECTION	1: Transactions are performed on TIPB → MPUI. 0: Transactions are performed on MPUI → TIPB.	R/W	0x1
9	RHEA_ERROR	0: Cleared after being read 1: DMA TIPB access is aborted, an IRQ is generated.	R	0x0
8	IRQ_STATE	0: Cleared after being read 1: An IRQ is requested by DMA channel 1.	R	0x0
7	IRQ_MODE	0: Interrupt is requested at the end of TIPB buffer transfer or if there is an nEND_DMA. 1: An interrupt must be launched at the end of MPUI page transfer	R/W	0x1
6	DMA_START	0: No effect 1: (Write) initiates the DMA transfer. Reading of this bit always returns zero.	R/W	–
5	MAS	0: DMA transactions are performed on 8-bit MPUI and TIPB data. 1: DMA transactions are performed on 16-bit MPUI and TIPB data.	R/W	0x1

† DMA1_CTRL is only writable and readable when the DMA controller clock runs. It means that each time the DSP software want to access these registers, if GSM-MPU is in sleep mode, it must set the BRIDGE_CLK_EN bit of DSP MPUI configuration register to 1.

Table 3–89. DMA1 Control Register (DMA1_CTRL)† (Continued)

Bit	Name	Function	Access	Reset
4	CUR-RENT_PAGE	0: MPUI access is performed on the first MPUI page. 1: MPUI access is performed on the 2nd MPUI page. This bit is automatically updated during transfer.	R/W	0x0
3	FIFO_MODE	1: RHEA_START address is used for all TIPB accesses. The peripheral can stop the transfer only by nEND_DMA signal. RHEA_DEPTH is not used in this mode.	R/W	0x0
2	ONE_SHOT	0: No effect 1: ENABLE flag is cleared at the end of the MPUI page transfer.	R/W	0x0
1	IDLE	0: DMA transfer is running. 1: DMA channel is idled.	R	0x1
0	ENABLE	0: DMA channel 1 is disabled. 1: DMA channel 1 is enabled. When DMA channel is disabled, hard or soft requests are ignored.	R/W	0x0

† DMA1_CTRL is only writable and readable when the DMA controller clock runs. It means that each time the DSP software want to access these registers, if GSM-MPU is in sleep mode, it must set the BRIDGE_CLK_EN bit of DSP MPUI configuration register to 1.

Table 3–90. DMA1 MPUI Current Offset Control Register (DMA1_CUR_OFFSET_API)†

Bit	Name	Function	Reset
15:12	Reserved		–
11:0	API_OFFSET	This field indicates MPUI offset of the next MPUI memory reading or writing.	0x000

† This register can be read only if the DMA controller is not running.

Table 3–91 through Table 3–96 define the channel 2 transfer parameters. The DMA transfer configuration registers are connected either to the GSM-MPU TIPB or the DSP TIPB, depending on the corresponding DMA_ALLOC flag.

Table 3–91. DMA2 TIPB Address Register (DMA2_RAD)

Bit	Name	Function	Reset
15:11	RHEA_CS	Peripheral chip-select	0x00
10:0	RHEA_START	TIPB buffer start address	0x000

Table 3–92. DMA2 TIPB Depth Register (DMA2_RDPTH)

Bit	Name	Function	Reset
15:11	Reserved		–
10:0	RHEA_DEPTH	TIPB buffer depth in bytes	0x000

Table 3–93. DMA2 MPUI Address Register (DMA2_AAD)

Bit	Name	Function	Reset
15:12	Reserved		–
11:0	API_START	Start address of the reception buffer in the MPUI memory. The address is always expressed in bytes	0x000

Table 3–94. DMA2 MPUI Length Register (DMA2_ALGTH)

Bit	Name	Function	Reset
15:12	Reserved		–
11:0	API_LENGTH	MPUI page length in bytes	0x000

Table 3–95. DMA2 Control Register (DMA2_CTRL)†

Bit	Name	Function	Access	Reset
15:13	–	Reserved	–	–
12:11	PRIORITY	Number of additional readings on the bus: 00: Zero more reads, one reading 01: One more read, two successive readings 10: Two more reads, three successive readings 11: Three more reads, four successive readings	R/W	0x0
10	DIRECTION	1: Transactions are performed on TIPB → MPUI. 0: Transactions are performed on MPUI → TIPB.	R/W	0x1
9	RHEA_ERROR	0: Cleared after being read 1: DMA TIPB access is aborted, an IRQ is generated.	R	0x0
8	IRQ_STATE	0: Cleared after being read 1: An IRQ is requested by DMA channel 2.	R	0x0
7	IRQ_MODE	0: Interrupt is requested at the end of TIPB buffer transfer or if there is an nEND_DMA. 1: An interrupt must be launched at the end of MPUI page transfer.	R/W	0x1
6	DMA_START	0: No effect 1: (Write) initiates the DMA transfer. Reading of this bit always returns zero.	R/W	–
5	MAS	0: DMA transactions are performed on 8-bit MPUI and TIPB data. 1: DMA transactions are performed on 16-bit MPUI and TIPB data	R/W	0x1

† DMA2_CTRL is only writable and readable when the DMA controller clock runs. Therefore, if GSM-MPU is in sleep mode, each time the DSP software wants to access these registers, it must set the BRIDGE_CLK_EN bit of DSP MPUI configuration register to 1.

Table 3–95. DMA2 Control Register (DMA2_CTRL)† (Continued)

Bit	Name	Function	Access	Reset
4	CURRENT_ PAGE	0: MPUI access is performed on the first MPUI page. 1: MPUI access is performed on the 2nd MPUI page. This bit is automatically updated during transfer.	R/W	0x0
3	FIFO_MODE	1: RHEA_START address is used for all TIPB accesses and the peripheral can stop the transfer only by nEND_DMA signal. RHEA_DEPTH is not used in this mode.	R/W	0x0
2	ONE_SHOT	0: No effect 1: ENABLE flag is cleared at the end of the MPUI page transfer.	R/W	0x0
1	IDLE	0: DMA transfer is running. 1: DMA channel is idled.	R	0x1
0	ENABLE	0: DMA channel 2 is disabled. 1: DMA channel 2 is enabled. When channel is disabled, hard or soft requests are ignored.	R/W	0x0

† DMA2_CTRL is only writable and readable when the DMA controller clock runs. Therefore, if GSM-MPU is in sleep mode, each time the DSP software wants to access these registers, it must set the BRIDGE_CLK_EN bit of DSP MPUI configuration register to 1.

Table 3–96. DMA2 MPUI Current Offset Register (DMA2_CUR_OFFSET_API)†

Bit	Name	Function	Reset
15:12	Reserved		–
11:0	API_OFFSET	This indicates offset MPUI of the next reading or writing MPUI memory.	0x000

† This register must be read only if the DMA controller is not running.

Table 3–97 through Table 3–97 define the channel 3 transfer parameters.The DMA transfer configuration registers are connected either to the GSM-MPU TIPB or to the DSP TIPB, depending on the corresponding DMA_ALLOC flag.

Table 3–97. DMA3 TIPB Address Register (DMA3_RAD)

Bit	Name	Function	Reset
15:11	RHEA_CS	Peripheral chip-select	0x00
10:0	RHEA_START	TIPB buffer start address	0x000

Table 3–98. DMA3 TIPB Buffer Depth Register (DMA3_RDPTH)

Bit	Name	Function	Reset
15:11	Reserved		–
10:0	RHEA_Depth	TIPB buffer depth in bytes	0x000

Table 3–99.DMA3 MPUI Address Register (DMA3_AAD)

Bit	Name	Function	Reset
15:12	Reserved		0x0
11:0	API_START	Start address of the reception buffer in the MPUI memory. The address is always expressed bytes.	–

Table 3–100. DMA3 MPUI Length Register (DMA3_ALGTH)

Bit	Name	Function	Reset
15:12	Reserved		–
11:0	API_OFFSET	This indicates MPUI offset of the next reading or writing MPUI memory.	0x000

Table 3–101. DMA3 Control Register (DMA3_CTRL)

Bit	Name	Function	Access	Reset
15:13	–	Reserved		–
12:11	PRIORITY	Number of additional readings on the bus: 00: Zero more reads, one reading 01: One more read, two successive readings 10: Two more reads, three successive readings 11: Three more reads, four successive readings	R/W	0x0
10	DIRECTION	1: Transactions are performed on TIPB → MPUI. 0: Transactions are performed on MPUI → TIPB.	R/W	0x1
9	RHEA_ERROR	0: Cleared after being read 1: DMA TIPB access is aborted, an IRQ is generated.	R	0x0
8	IRQ_STATE	0: Cleared after being read 1: An IRQ requested by DMA channel 3	R	0x0
7	IRQ_MODE	0: Interrupt is requested at the end of TIPB buffer transfer or if there is an nEND_DMA. 1: An interrupt must be launched at the end of MPUI page transfer.	R/W	0x1
6	DMA_START	0: No effect 1: (Write) initiates the DMA transfer. Reading of this bit always returns zero.	R/W	–
5	MAS	0: DMA transactions are performed on 8-bit MPUI and TIPB data. 1: DMA transactions are performed on 16-bit MPUI and TIPB data.	R/W	0x1
4	CURRENT_PAGE	0: MPUI access is performed in the first MPUI page. 1: MPUI access is performed in the 2nd MPUI page This bit is automatically updated during transfer.	R/W	0x0
3	FIFO_MODE	1: RHEA_START address is used for all TIPB access and the peripheral can stop the transfer only by nEND_DMA signal. RHEA_DEPTH is not used in this mode.	R/W	0x0
2	ONE_SHOT	0: No effect 1: ENABLE flag is cleared at the end of the MPUI page transfer.	R/W	0x0

Table 3–101. DMA3 Control Register (DMA3_CTRL) (Continued)

Bit	Name	Function	Access	Reset
1	IDLE	0: DMA transfer is running. 1: DMA channel is idled.	R	0x1
0	ENABLE	0: DMA channel 3 is disabled. 1: DMA channel 3 is enabled. When channel is disabled, hard or soft requests are ignored.	R/W	0x0

Table 3–102. DMA3 MPUI Current Offset Register (DMA3_CUR_OFFSET_API)

Bit	Name	Function	Reset
15:12	Reserved		–
11:0	API_OFFSET	This indicates MPUI offset of the next MPUI memory reading or writing.	0x000

† This register can be read only if the DMA controller is not running.

Table 3–103 through Table 3–108 define the channel 4 transfer parameters.The DMA transfer configuration registers are connected either to the GSM-MPU TIPB or to the DSP TIPB, depending on the corresponding DMA_ALLOC flag.

Table 3–103. DMA4 TIPB Address Register (DMA4_RAD)

Bit	Name	Function	Reset
15:11	RHEA_CS	Peripheral chip-select	0x00
10:0	RHEA_START	TIPB buffer start address	0x000

Table 3–104. DMA4 TIPB Depth Value Register (DMA4_RDPTH)

Bit	Name	Function	Reset
15:11	Reserved		–
10:0	RHEA_DEPTH	TIPB buffer depth in bytes	0x000

Table 3–105. DMA4 MPUI Address Register (DMA4_AAD)

Bit	Name	Function	Reset
15:12	Reserved		–
11:0	API_START	Start address of the reception buffer in the MPUI memory. The address is always expressed in bytes.	0x000

Table 3–106. DMA4 MPUI Length Register (DMA4_ALGTH)

Bit	Name	Function	Reset
15:12	Reserved		–
11:0	API_LENGTH	MPUI page length in bytes	0x000

Table 3–107. DMA4 Control Register (DMA4_CTRL)

Bit	Name	Function	Access	Reset
15:13	–	Reserved		–
12:11	PRIORITY	Number of additional readings on the bus: 00: Zero more reads, one reading 01: One more read, two successive readings 10: Two more reads, three successive readings 11: Three more reads, four successive readings	R/W	0x0
10	DIRECTION	1: Transactions are performed on TIPB → MPUI. 0: Transactions are performed on MPUI → TIPB.	R/W	0x1
9	RHEA_ERROR	0: Cleared after being read 1: DMA TIPB access is aborted, an IRQ is also generated.	R	0x0
8	IRQ_STATE	0: Cleared after being read 1: An IRQ requested by DMA channel 4.	R	0x0
7	IRQ_MODE	0: Interrupt is requested at the end of TIPB buffer transfer or if there is an nEND_DMA. 1: An interrupt must be launched at the end of MPUI page transfer.	R/W	0x1
6	DMA_START	0: No effect 1: (Write) initiates the DMA transfer. Reading of this bit always returns zero.	R/W	–
5	MAS	0: DMA transactions are performed on 8-bit MPUI and TIPB data. 1: DMA transactions are performed on 16-bit MPUI and TIPB data.	R/W	0x1
4	CUR-RENT_PAGE	0: MPUI access is performed in the first MPUI page. 1: MPUI access is performed in the 2nd MPUI page. This bit is automatically updated during transfer.	R/W	0x0
3	FIFO_MODE	1: RHEA_START address is used for all TIPB accesses. The peripheral can stop the transfer only by nEND_DMA signal. RHEA_DEPTH is not used in this mode.	R/W	0x0
2	ONE_SHOT	0: No effect 1: ENABLE flag is cleared at the end of the MPUI page transfer.	R/W	0x0

Table 3–107. DMA4 Control Register (DMA4_CTRL) (Continued)

Bit	Name	Function	Access	Reset
1	IDLE	0: DMA transfer is running. 1: DMA channel is idled.	R	0x1
0	ENABLE	0: DMA channel 4 is disabled. 1: DMA channel 4 is enabled. When channel is disabled, hard or soft requests are ignored.	R/W	0x0

Table 3–108. DMA4 MPUI Current Offset Register (DMA4_CUR_OFFSET_API)

Bit	Name	Function	Reset
15:12	Reserved		–
11:0	API_OFFSET	This indicates offset MPUI of the next reading or writing MPUI memory.	0x000

Note: This register can be read only when the DMA controller is stopped.

3.14 Radio Interface Registers

Table 3–109 contains the RIF registers. Table 3–110 through Table 3–113 describe the individual register bits.

Table 3–109. RIF Registers(FFFF:7000 – XIO:0000)

Name	Description	GSM-MPU/ DSP Addresses	Size	Type	Reset Value
DXR	Transmit data, accessible via the TIPB DSP interface.	FFFF:7000 XIO:0000	16 bits	R/W	Undefined
DRR	Receive data, accessible via the TIPB DSP and GSM-MPU interfaces.	FFFF:7002 XIO:0001	16 bits	R	Undefined
SPCX	Transmit control	NA XIO:0002	15 bits		000 0101 1001 1110
SPCR	Receive control	NA XIO:0003	15 bits		011 1100 1010 0010

Table 3–110. Transmit Data Register (DXR)

Bit	Name	Function	Access	Reset
15:0	DXR	Data to transmit	R/W	Undefined

Table 3–111. Receive Data Register (DRR)

Bit	Name	Function	Access	Reset
15:0	DRR	Received data DRR is updated when RSR is ready to be read and RRDY = 1. DRR cannot be written via the TIPB interface.	R	Undefined

Table 3–112. Control Register (SPCX)

Bit	Name	Function	Access	Reset
14:12	DIV_CLK	Frequency of transmission clock (CLK13m) 000: f13, 001: f13/2 010: f13/4, 011: f13/8 100: f13/16; others not used	R/W	0x0
11:10	THRESHOLD	Threshold level for FIFO-almost-empty flag The maximum value is 2.	R/W	0x1
9	FIFO_FULL	1: The transmit FIFO is full.	R	0x0
8	FIFO_EMPTY	0: The transmit FIFO is not empty.	R	0x1
7	ALMOST_ EMPTY	1: Transmit FIFO is almost empty.	R	0x1
6	NCK13_EN	Clock enable for the reference clock (RIF_CLK13)	R/W	0x0

Table 3–112. Control Register (SPCX) (Continued)

Bit	Name	Function	Access	Reset
5	NCLK_EN	Clock enable for the internal transmission clock	R/W	0x0
4	TXM	FSX configuration: 0: Input 1: Output	R/W	0x1
3	CLKX_AUTO	CLKX_AUTO is reset to 0 upon device reset. 0: RIF_CLK_OUT is equal to the RIF_CLKX input. 1: Allows the RIF to shut off RIF_CLKX_OUT when MCM = 1 and there is no word to transmit. Note: RIF_CLKX_OUT is stuck at one when not active. RIF_CLKX_OUT provides one falling edge before the rising edge of FSX. RIF_CLKX_OUT provides one falling edge and one rising edge after the LSB shift.	R/W	0x1
2	XRDY	Reserved	R	0x1
1	XRST	The transmit reset (XRST) resets the transmitter. If SPCX is modified to reconfigure the transmit serial port, a total of two write operations must be performed into the SPCX. The first must write a zero to XRST and the second must write the desired configuration and a one to XRST. Note: if XRST = 0, internal clocks of the serial port are not shut off as in the TMS320C54x™ DSP RIF. Clocks shut-off must be performed externally of the RIF.	W	0x1
0	MCM	Clock source for RIF_CLKX 0: RIF_CLKX is taken from the CLKX pin. 1: RIF_CLKX is the internal ASIC clock.	R/W	0x0

Table 3–113. Control Register (SPCR)

Bit	Name	Function	Access	Reset
14	CLKLB	Clock loopback mode bit: 1: Receive and transmit clocks are generated from the same internal source.	R/W	0x0
13	RDMA_MASK	1: RIF receive DMA request masked	R/W	0x1
12	XDMA_MASK	1: RIF transmit DMA request masked	R/W	0x1

Note: The RSRFULL bit is not reset by a DRR read event. This feature differs from the TMS320C54x™ DSP RIF functional specification. This allows keeping trace of a reception problem. The RV_WAIT state of the receive state machine matches with the TMS320C54x™ DSP RSRFULL bit. The advantage of this RIF RSRFULL definition is to avoid a short *hidden* receiving problem. Therefore, RSRFULL bit is not a control bit for the receive state machine; the RV_WAIT state is used instead. RSRFULL is reset by hardware and RRST is reset.

Table 3–113. Control Register (SPCR) (Continued)

Bit	Name	Function	Access	Reset
11	RINT_MASK	1: RIF receive interrupt masked	R/W	0x1
10	XINT_MASK	1: RIF transmit interrupt masked	R/W	0x1
9:7	THRESHOLD	Threshold of receive FIFO-not-empty flag The maximum value is 4.	R/W	0x1
6	FIFO_FULL	1: Receive FIFO is full.	R	0x0
5	FIFO_EMPTY	0: Receive FIFO is empty.	R	0x1
4	AL-MOST_FULL	1: Receive FIFO is not empty.	R	0x0
3	RSRFULL (See Note)	Receive shift full: 0: Upon device reset and SPI receiver reset (RRST) 1: A word has been shifted in the RSR and the current DRR value has not been read yet (RRDY=1). Receiver halts and waits for DRR to be read; the data in RSR is preserved but any data sent via DR is lost.	R	0x0
2	RRDY	Reserved	R	0x0
1	RRST	Receiver reset: If the SPCR is modified to reconfigure the receiving serial port, a total of two writes must be made into SPCR. The first writes a zero to RRST; the second writes the desired configu-ration and a one to RRST. Note: If RRST = 0, internal clocks of the serial port are not shut off as in the TMS320C54x™ DSP RIF. Clocks shut off must be performed externally of the RIF. Writing a zero to RRST clears the RSRFULL and RRDY bits.	R/W	0x1
0	DLB	Digital loopback mode: 0: Normal mode 1: DR and FSR are connected in the RIF to DX and FSX, respectively. The transmit clock is internally looped back to the receive clock.	R/W	0x0

Note: The RSRFULL bit is not reset by a DRR read event. This feature differs from the TMS320C54x™ DSP RIF functional specification. This allows keeping trace of a reception problem. The RV_WAIT state of the receive state machine matches with the TMS320C54x™ DSP RSRFULL bit. The advantage of this RIF RSRFULL definition is to avoid a short *hidden* receiving problem. Therefore, RSRFULL bit is not a control bit for the receive state machine; the RV_WAIT state is used instead. RSRFULL is reset by hardware and RRST is reset.

3.14.1 Shift Data Registers (XSR and RSR)

XSR and RSR are not directly accessible. XSR is driven by CLKX. RSR is1 driven by CLKR. The data size is always 16 bits.

3.15 Cipher Registers

Table 3–114 presents the cipher registers. Table 3–115 through Table 3–124 describe the individual register bits.

Table 3–114. Cipher Registers (XIO:2800)

Name	Description	Address Offset	Access	Type	Reset Value
CNTL_REG	Ciphering control	2800 (00)	6 bits	R/W	00 0000
STATUS_IRQ_REG	Interrupt status: Monitors completion of ciphering	2801 (01)	1 bit	R	0
STATUS_WORK_ REG	Ciphering status monitoring	2802 (02)	1 bit	R	0
KC_REG_1	KC key value	2803 (03)	16 bits	R/W	0000 0000 0000 0000
KC_REG_2	KC key value	2804 (04)	16 bits	R/W	0000 0000 0000 0000
KC_REG_3	KC key value	2805 (05)	16 bits	R/W	0000 0000 0000 0000
KC_REG_4	KC key value	2806 (06)	16 bits	R/W	0000 0000 0000 0000
COUNT_REG_1	Lower 11 bits of count value	2807 (07)	11 bits	R/W	000 0000 0000
COUNT_REG_2	Upper 11 bits of count value	2808 (08)	11 bits	R/W	000 0000 0000
DECI_REG_1	Contains lower 16 bits of BLOCK1	2809 (09)	16 bits	R	0000 0000 0000 0000
DECI_REG_2	Contains 16 bits of BLOCK1	280A (0A)	16 bits	R	0000 0000 0000 0000
DECI_REG_3	Contains 16 bits of BLOCK1	280B (0B)	16 bits	R	0000 0000 0000 0000
DECI_REG_4	Contains 16 bits of BLOCK1	280C (0C)	16 bits	R	0000 0000 0000 0000
DECI_REG_5	Contains 16 bits of BLOCK1	280D (0D)	16 bits	R	0000 0000 0000 0000
DECI_REG_6	Contains 16 bits of BLOCK1	280E (0E)	16 bits	R	0000 0000 0000 0000
DECI_REG_7	Contains 16 bits of BLOCK1	280F (0F)	16 bits	R	0000 0000 0000 0000
DECI_REG_8	Contains upper 16 bits of BLOCK1	2810 (10)	2 bits	R	00

Table 3–114. Cipher Registers (XIO:2800)(Continued)

Name	Description	Address Offset	Access	Type	Reset Value
ENCI_REG_1	Contains lower 16 bits of BLOCK2	2811 (11)	16 bits	R	0000 0000 0000 0000
ENCI_REG_2	Contains 16 bits of BLOCK2	2812 (12)	16 bits	R	0000 0000 0000 0000
ENCI_REG_3	Contains 16 bits of BLOCK2	2813 (13)	16 bits	R	0000 0000 0000 0000
ENCI_REG_4	Contains 16 bits of BLOCK2	2814 (14)	16 bits	R	0000 0000 0000 0000
ENCI_REG_5	Contains 16 bits of BLOCK2	2815 (15)	16 bits	R	0000 0000 0000 0000
ENCI_REG_6	Contains 16 bits of BLOCK2	2816 (16)	16 bits	R	0000 0000 0000 0000
ENCI_REG_7	Contains 16 bits of BLOCK2	2817 (17)	16 bits	R	0000 0000 0000 0000
ENCI_REG_8	Contains upper 16 bits of BLOCK2	2818 (18)	2 bits	R	00

Table 3–115. Control Register (CNTL_REG)

Bit	Name	Function	HW Reset
15:6	Reserved		Undefined
5	CIPHER_ONLY	0: Both data decipher and encipher are performed. 1: Only data decipher is performed.	0x0
4	CLK_EN	Internal clock 0: Disabled 1: Enabled	0x0
3:2	MODE	Determines which algorithm is used: 00: No algorithm, outputs are forced to zero, inputs do not matter. 01: Algorithm A51 10: Algorithm A52 11: Forbidden	0x0

Table 3–115. Control Register (CNTL_REG) (Continued)

Bit	Name	Function	HW Reset
1	RESET_SW	0: Module reset	0x0
		1: No effect	
0	START	Start ciphering:	0x0
		0: No effect	
		1: Starts the process (rising edge only)	
		Toggle bit (always returns 0 when read)	

Table 3–116. Interrupt Status Register (STATUS_IRQ_REG)

Bit	Name	Function	HW Reset
15:2	Reserved		Undefined
0	IT_FIN	1: Ciphering is complete. Remains 1 until read.	0x0

Note: It_fin is reset when on read access.

Table 3–117. Working Status Register (STATUS_WORK_REG)

Bit	Name	Function	HW Reset
15:1	Reserved		Undefined
0	WORKING	1: Ciphering ongoing	0x0

Table 3–118. KC Registers (KC_REG)

Bit	Name	Function	HW RESET
15:0	KC_REG	Contains 16 bits of KC	0x0000

Table 3–119. KC Key Values

	b_{15}															B_0
1st word XIO:2803 Kc_REG1	6	5	4	3	2	1	0	0	0	0	0	0	0	0	0	0
2nd word XIO:2804 Kc_REG2	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7
3rd word XIO:2805 Kc_REG3	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23
4th word XIO:2806 Kc_REG4	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39

Note: Bit 1 is the LSB and bit 54 is the MSB of KC.

The key is written into the KC registers as described in the KC key values table.

Table 3–120. Count Registers (COUNT_REG_#)

Bit	Name	Function	HW Reset
15:11	Reserved		–
10:0	COUNT_REG#	Contain 11 bits of frame number COUNT as specified in Table 3–121.	0x000

Note: Bit 1 is LSB of COUNT, bit 22 is the MSB.

Table 3–121. Frame Number Count Values

15	b_{15}															b_0
1st word XIO:2807 COUNT_REG1	0	0	0	0	0	11	10	9	8	7	6	5	4	3	2	1
2nd word XIO:2808 COUNT_REG2	0	0	0	0	0	22	21	20	19	18	17	16	15	14	13	12

Table 3–122. Decipher Data Registers (DECI_REG_#)

Bit	Name	Function	Reset
15:0	DECI_REG#	Contain the 114 bits of BLOCK1. The least-significant bit of BLOCK 1 is in DECI_REG_1[15] and the most-significant bit is in DECI_REG_8[14]. Table 3–123 contains the BLOCK1 bits.	0x0000

Table 3–123. Block1 Bits Location

Nb	Name	Address	b_{15}															b_0
1	DECI_REG_1	2809	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	DECI_REG_2	280A	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
3	DECI_REG_3	280B	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
4	DECI_REG_4	280C	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
5	DECI_REG_5	280D	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
6	DECI_REG_6	280E	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
7	DECI_REG_7	280F	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
8	DECI_REG_8	2810	113	114	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3–124. Encipher Data Registers (ENCI_REG_#)

Bit	Name	Function	Reset
15:0	ENCI_REG#	These contain the 114 bits of BLOCK2. The least-significant bit of BLOCK2 is ENCI_REG_1[15] and the most-significant bit is ENCI_REG_8[14].	0x0000

Table 3–125. Block2 Bits Location

Nb	Name	Address	b_{15}															b_0
1	ENCI_REG_1	2811	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	ENCI_REG_2	2812	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
3	ENCI_REG_3	2813	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
4	ENCI_REG_4	2814	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
5	ENCI_REG_5	2815	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
6	ENCI_REG_6	2816	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
7	ENCI_REG_7	2817	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
8	ENCI_REG_8	2818	113	114	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3.16 MCSI Registers

The MCSI is on the chip-select number 1. Table 3–126 lists the MCSI registers. Table 3–127 through Table 3–135 describe the register bits.

Table 3–126. MCSI Registers(XIO:0800)

Name	Description	Address	Access	Type	Reset Value
RX15	Receive data 15	083F (3F)	16 bits	R	???? ???? ???? ????
RX14	Receive data 14	083E (3E)	16 bits	R	???? ???? ???? ????
RX13	Receive data 13	083D (3D)	16 bits	R	???? ???? ???? ????
RX12	Receive data 12	083C (3C)	16 bits	R	???? ???? ???? ????
RX11	Receive data 11	083B (3B)	16 bits	R	???? ???? ???? ????
RX10	Receive data 10	083A (3A)	16 bits	R	???? ???? ???? ????
RX9	Receive data 9	0839 (39)	16 bits	R	???? ???? ???? ????
RX8	Receive data 8	0838 (38)	16 bits	R	???? ???? ???? ????
RX7	Receive data 7	0837 (37)	16 bits	R	???? ???? ???? ????
RX6	Receive data 6	0836 (36)	16 bits	R	???? ???? ???? ????
RX5	Receive data 5	0835 (35)	16 bits	R	???? ???? ???? ????
RX4	Receive data 4	0834 (34)	16 bits	R	???? ???? ???? ????
RX3	Receive data 3	0833 (33)	16 bits	R	???? ???? ???? ????
RX2	Receive data 2	0832 (32)	16 bits	R	???? ???? ???? ????
RX1	Receive data 1	0831 (31)	16 bits	R	???? ???? ???? ????
RX0	Receive data 0	0830 (30)	16 bits	R	???? ???? ???? ????
TX15	Transmit data 15	082F (2F)	16 bits	R	???? ???? ???? ????

Table 3–126. MCSI Registers(XIO:0800) (Continued)

Name	Description	Address	Access	Type	Reset Value
TX14	Transmit data 14	082E (2E)	16 bits	R	???? ???? ???? ????
TX13	Transmit data 13	082D (2D)	16 bits	R	???? ???? ???? ????
TX12	Transmit data 12	082C (2C)	16 bits	R	???? ???? ???? ????
TX11	Transmit data 11	082B (2B)	16 bits	R	???? ???? ???? ????
TX10	Transmit data 10	082A (2A)	16 bits	R	???? ???? ???? ????
TX9	Transmit data 9	0829 (29)	16 bits	R	???? ???? ???? ????
TX8	Transmit data 8	0828 (28)	16 bits	R	???? ???? ???? ????
TX7	Transmit data 7	0827 (27)	16 bits	R	???? ???? ???? ????
TX6	Transmit data 6	0826 (26)	16 bits	R	???? ???? ???? ????
TX5	Transmit data 5	0825 (25)	16 bits	R	???? ???? ???? ????
TX4	Transmit data 4	0824 (24)	16 bits	R	???? ???? ???? ????
TX3	Transmit data 3	0823 (23)	16 bits	R	???? ???? ???? ????
TX2	Transmit data 2	0822 (22)	16 bits	R	???? ???? ???? ????
TX1	Transmit data 1	0821 (21)	16 bits	R	???? ???? ???? ????
TX0	Transmit data 0	0820 (20)	16 bits	R	???? ???? ???? ????
Reserved	Reserved	–	–		–
STATUS_REG	Transmission status	0806 (06)	7 bits		0000 0000 0000 0000
CLOCK_ FREQUENCY_REG	Clock frequency	0805 (05)	11 bits	R/W	0000 0000 0000 0000
OVER_CLOCK_REG	Oversize clock periods	0804 (04)	10 bits	R/W	0000 0000 0000 0000

Table 3–126. MCSI Registers(XIO:0800) (Continued)

Name	Description	Address	Access	Type	Reset Value
CHANNEL_ USED_REG	Channel used	0803 (03)	16 bits	R/W	0000 0000 0000 0000
INTERRUPTS_REG	Interrupt masks	0802 (02)	11 bits	R/W	0000 0000 0000 0000
MAIN_ PARAMETERS_REG	Main transmission parameters	0801 (01)	14 bits	R/W	0000 0000 0000 0000
CONTROL_REG	Transmission control	0800 (00)	3 bits	R/W	0000 0000 0000 0000

3.16.1 Control Registers

The CHANNEL_USED_REG, CLOCK_FREQUENCY_REG, OVER_CLOCK_ REG, INTERRUPTS_REG, and MAIN_PARAMETERS_REG are write-protected if the MCSI is enabled (CONTROL_REG[0] = 1).

Table 3–127. Channel Used Register (CHANNEL_USED_REG)

Bit	Name	Function	Reset
15	USE_CH15	Data transmission on channel 15 0: Unselected 1: Selected	0x0
14	USE_CH14	Data transmission on channel 14 0: Unselected 1: Selected	0x0
13	USE_CH13	Data transmission on channel 13 0: Unselected 1: Selected	0x0
12	USE_CH12	Data transmission on channel 12 0: Unselected 1: Selected	0x0
11	USE_CH11	Data transmission on channel 11 0: Unselected 1: Selected	0x0
10	USE_CH10	Data transmission on channel 10 0: Unselected 1: Selected	0x0
9	USE_CH9	Data transmission on channel 9 0: Unselected 1: Selected	0x0

Table 3–127. Channel Used Register (CHANNEL_USED_REG) (Continued)

Bit	Name	Function	Reset
8	USE_CH8	Data transmission on channel 8 0: Unselected 1: Selected	0x0
7	USE_CH7	Data transmission on channel 7 0: Unselected 1: Selected	0x0
6	USE_CH6	Data transmission on channel 6 0: Unselected 1: Selected	0x0
5	USE_CH5	Data transmission on channel 5 0: Unselected 1: Selected	0x0
4	USE_CH4	Data transmission on channel 4 0: Unselected 1: Selected	0x0
3	USE_CH3	Data transmission on channel 3 0: Unselected 1: Selected	0x0
1	USE_CH1	Data transmission on channel 1 0: Unselected 1: Selected	0x0
0	USE_CH0	Data transmission on channel 0 0: Unselected 1: Selected	0x0

CHANNEL_USED_REG provides configuration for the channel selection register. This register is used only in multichannel mode.

Table 3–128. Clock Frequency Register (CLOCK_FREQUENCY_REG)

Bit	Name	Function	Access	Reset
15:11	Reserved	–	R	0x00
10:0	CLK_FREQ	Division factor of 13-MHz reference clock: Range: 2 to 2047	R/W	0x000

Note: The transmission clock frequency can be programmed from 6.3 kHz to 6.5 MHz in 76-ns steps: Clock frequency = 13 MHz/CLK_FREQ, with $2 \leq$ CLK_FREQ ≤ 2047.

In master mode, the clock frequency register defines the transmission baud rate from a frequency ratio based on a 13-MHz reference clock. This register is used only in master mode when the interface generates the serial clock.

Table 3–129. Oversize Frame Dimension Register (OVER_CLOCK_REG)

Bit	Name	Function	Access	Reset
15:10	Reserved	–	R	0x00
9:0	OVER_CLOCK	Over clock periods in frame duration: Range: 0 to 1023	R/W	0x000

The oversize frame dimension register holds the oversize frame dimension in clock periods.

Table 3–130. Interrupt Mask Register (INTERRUPTS_REG)

Bit	Name	Function	Access	Reset
15:11	Reserved	–	R	0x00
10	MASK_IT_ERROR	Frame duration error interrupt 0: Mask 1: Unmask	R/W	0x0
9	MASK_IT_TX	Transmit interrupt 0: Mask 1: Unmask	R/W	0x0
8	MASK_IT_RX	Receive interrupt 0: Mask 1: Unmask	R/W	0x0
7:4	NB_CHAN_IT_TX	Channel number for transmit interrupt Range: 0 to 15	R/W	0x0
3:0	NB_CHAN_IT_RX	Channel number for receive interrupt Range: 0 to 15	R/W	0x0

The interrupt mask register holds the interrupt masks.

Table 3–131. Main Parameters Register (MAIN_PARAMETERS_REG)

Bit	Name	Function	Access	Reset
15:14	Reserved	–	R	0x0
13:12	DAI_CONF	DAI mode selection: 00: Normal (no DAI) 01: Radio downlink 10: Radio uplink 11: Acoustic	R/W	0x0
11	DAI_SYN_REQ	Synchronization with audio frame: 0: No synchronization 1: Synchronized	R/W	0x0

Table 3–131. Main Parameters Register (MAIN_PARAMETERS_REG) (Continued)

Bit	Name	Function	Access	Reset
10	FSYNCH_POL	Frame synchronization pulse polarity: 0: Positive 1: Negative	R/W	0x0
9	FSYNCH_MODE	Frame synchronization pulse position: 0: Normal 1: Alternate	R/W	0x0
8	FSYNCH_SIZE	Frame synchronization pulse shape: 0: Short 1: Long	R/W	0x0
7	MULTI	Frame structure: 0: Single 1: Multiple	R/W	0x0
6	MCSI_MODE	Interface transmission mode: 0: Slave 1: Master	R/W	0x0
5	CONTINUOUS	Frame mode: 0: Burst 1: Continuous	R/W	0x0
4	CLOCK_POL	Clock edge selection: 0: Rising 1: Falling	R/W	0x0
3:0	WORD_SIZE	Word size in number of bits: Range: 2 to 15 (with 2 for 3 bits and 15 for 16 bits)	R/W	0x0

The main parameters register holds the main transmission parameters.

Table 3–132. Control Register (CONTROL_REG)

Bit	Name	Function	Access	SW Reset	HW Reset
15:3	Reserved	–	R	0x0000	0x0000
2	DAICLKEN	DAI interface activity: 0: Disabled 1: Enabled	R/W	0x0	0x0

Note: The software reset is applied as long as the MCSI software reset bit is set to 1. A software reset disables the mcsi (the MCSI CLK_ENABLE bit is cleared), initializes the status register, and does not modified the others registers.

Table 3–132. Control Register (CONTROL_REG) (Continued)

Bit	Name	Function	Access	SW Reset	HW Reset
1	SW_RESET	Asynchronous reset of module: 0: Disabled 1: Enabled	R/W	0x1	0x0
0	CLK_ENABLE	Clock of MCSI module: 0: Disabled 1: Enabled	R/W	0x0	0x0

Note: The software reset is applied as long as the MCSI software reset bit is set to 1. A software reset disables the mcsi (the MCSI CLK_ENABLE bit is cleared), initializes the status register, and does not modified the others registers.

CONTROL_REG is the transmission control register.

Table 3–133. Status Register (STATUS_REG)

Bit	Name	Function	Access	SW Reset	HW Reset
15:7	Reserved	–	R	0	0x000
6	DAI_READY	System simulator reset: 0: Not detected 1: Detected	R/W	0	0x0
5	TX_UNFLOW	Transmit underflow: 0: No error 1: Underflow error	R	0	0x0
4	TX_READY	Transmit interrupt: 0: None 1: Pending	R/W	0	0x0
3	RX_OVFLOW	Receive overflow error: 0: No error 1: Overflow	R	0	0x0
2	RX_READY	Receive interrupt: 0: None 1: Pending	R/W	0	0x0
1	ERROR_TYPE	Error type: 0: Frame too short 1: Frame too long	R	0	0x0
0	FRAME_ERROR	Frame duration error: 0: No error 1: Wrong frame duration	R/W	0	0x0

STATUS_REG is the transmission status register.

3.16.2 Data Registers

Table 3–134. Receive Word Register (RX_REG)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Acc.	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Note: The MCSI receives the most-significant bit first. For example, if the WORD_SIZE equals 11, the upper 12 bits of the RX registers contain the received data and the lower 4 bits are zeros.

The receive word register describes the individual receive word register bits.

Table 3–135. Transmit Word Register (TX_REG)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Acc.	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Note: The MCSI transmits the most-significant bit first. For example, if the WORD_SIZE equals 11, the upper 12 bits of the TX registers is transmitted.

The transmit word register describes the individual transmit word register bits.

3.17 DSP Interrupts

The DSP subchip has 17 interrupt lines, 11 of which are dedicated to external peripherals (INT0n to INT10n). Because the DSP megamodule uses four of the available 16 interrupts for internal purposes, the external interrupts nXIRQ(N) do not map directly to the DSP megamodule interrupts INTmN. These interrupts are mapped as shown in Table 3–136.

Table 3–136. DSP Interrupts Mapping (XIO:FA00)

Name	Sense	Location (hex)	Function
RSN			Reset (hardware or software)
nMIN		4	Abort on TIPB OR INT4n redirection
TINT		4C	Timer interrupts
RINT		50	SPI receive interrupt
XINT		54	SPI transmit interrupt
AINT		64	MPUI interrupts
INT0n	Level	40	RIF receive interrupt
INT1n	Level	44	RIF transmit interrupt
INT2n	Level	48	UART interrupt 1) Error on receiver line 2) Receive time-out 3) Received character 4) Character to transmit 5) Modem status change 6) Received XOFF 7) CTS/RTS deactivation
INT3n	Level	60	MCSI receive interrupt
INT4n	Level	58	MCSI transmit interrupt
INT5n	Level	5C	MCSI-frame duration error interrupt
INT6n	Level	68	MCSI DAI interrupt
INT7n	Edge	6C	Cypher interrupts 1) End of ciphering process 2) Processing error
INT8n	Edge	70	TPU frame interrupt
INT9n	Edge	74	TPU programmable interrupt
INT10n	Level	78	DMA interrupt

Note: The TPU interrupt (INT9n) is a facility offered to the DSP programmer in order to allow the generation of a DSP interrupt at a dedicated time with a 1/4-GSM bit accuracy. The interrupt is set in a scenario by using a time-stamped instruction.

3.17.1 Internal Registers

The XIO interrupt processor has one 16-bit control register and one nonimplemented command register. The control register is used exclusively for assigning edge-triggered/level-sensitive status to each of the 12 interrupt channels. The nonimplemented command register is a block of decoding logic that issues clear commands to the level-sensitive logic in each interrupt channel upon detecting a TIPB write transaction to an address that falls within the required address range. Table 3–137 shows the DSP interrupts registers.

Table 3–137. DSP Interrupts Registers (XIO:FA00 .. XIO:FA01)

Name	Description	Address	Access	Type	Reset Value
CNTRL_REG	Control	XIO:FA00	13 bits		???0 0000 0000 0000
CLEAR_REG	Clear commands	XIO:FA01	12 bits	W	

Table 3–138. Edge-Triggered/Level-Sensitive Control Register (CNTRL_REG)

Bit	Name	Function	Reset
15:13	Reserved	–	–
12	INT4_SWITCH	Channel 4 connection: 0: Channel 4 is connected to DSP INT4N (0x58). 1: Channel 4 is connected to DSP nNMI (0x4).	0x0
11:0	CHx trig/level	Channel CHx sense: 1: CHx is edge sensitive. 0: CHx is level sensitive.	0x000

The edge-triggered/level sensitive control register is mapped in DSP IO space on nXSTROBE(3).

3.17.1.1 Level-Sensitive Clear Commands

A C54x™ DSP write transaction in I/O space (nXSTROBE(3)) at an odd-valued address in the range of 0xFA01 through 0xFBFF results in a clear being issued to those interrupt channels whose assigned bit in the 16-bit word is being written as a logic 1.

Commands to clear interrupt channels are necessary for those channels assigned as level-sensitive interrupt channels and designated as shared channels. Table 3–138 illustrates the alignment of the channel clear assignments within the 16-bit word written to the XIO interrupt processor, and, in addition, gives the permissible range of addresses over which the write can take place.

3.17.1.2 NMI Interrupt

Bit 12 of the control register (int4_switch) allows to connect the interrupt number 4 to nIRQ(4) or to the NMI interrupt of the DSP.

INT4_SWITCH = 0 ≥ nXIRQ(4) connected to the INT4N of the DSP, NMI = 1
INT4_SWITCH = 1 ≥ nXIRQ(4) connected to the NMI of the DSP, INT4N = 1.

3.18 Memory Protection Unit (MPU)

3.18.1 Protection Mode Definition

Each of the four possible memory regions has a separate protection attribute. The type of protection associated with a region is defined by the PMn[2–0] bits in the MPU protection mode register.

The MPU recognizes several operating modes and uses this identification to build the protection scheme. Among the modes that can be detected are:

- User mode: This is the user mode as defined by the TMS470R1x user's guide. Most application programs run in user mode.
- Nonuser mode: This mode is also defined in the TMS470R1x user's guide. It is entered in order to service interrupts or exceptions or to access protected resources.
- Privileged region: This mode is specific to the MPU architecture. It allows the application program to set memory regions (regions 1 and 2, as defined by the MPU register frame) as privileged memory space where the operational codes (opcodes) located in it have write rights to a given memory region.

Table 3–139 summarizes the protection modes that can be selected.

Table 3–139. Protection Mode Definition

PMn 2	PMn 1	PMn 0	Protection Mode
0	0	0	Protection disabled: User, nonuser, and privileged-region read/write allowed
0	0	1	Nonuser read/write, user read-only (any fetched code location)
0	1	0	Reserved
0	1	1	ROM: Nonuser read-only, user read-only (any fetched code location)
1	0	0	Writes are authorized only when performed from code fetched in protected region 1 (any MPU operating mode).
1	0	1	Writes are authorized only when performed from code fetched in protected regions 1 or 2 (any MPU operating mode).
1	1	0	Reserved
1	1	1	Reserved

3.18.2 MPU Control Register Frame

The MPU module contains eleven 16-bit memory-mapped registers that can be accessed through a TIPB bus.

- The MPU register mapping to the MPU memory space is device-dependent, and hence is defined at the chip/system level. Within the regis-

ter frame, the physical address of registers is the start address (defined by the system) + the offset address.

- ❏ The status register saves unallowed accesses as well as out-of-protection fault signatures.
- ❏ The control register enables/disables assertion of the MPU_FAULT signal as well as the out-of-protection mechanism for each region.
- ❏ The protection mode register is dedicated to the selection of the protection mode.
- ❏ The remaining eight registers are partitioned by regions (two registers per region). Each register pair defines the base and start addresses and the end address for its associated region.
- ❏ The registers can be read at any time without affecting ongoing operations. All registers can be written according to their bit definition, as discussed in Section 3.18.4. The status register is a clear-only register (only write to 0).

Table 3–140. MPU Control and Status Register Frame

MPU Register	Offset Addr.	BITS															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ST	0x00	Reserved								OP4	OP3	OP2	OP1	IA_R4	IA_R3	IA_R2	IA_R1
PM	0x02	Reserved				PM42	PM41	PM40	PM32	PM31	PM30	PM2 2	PM2 1	PM20	PM1 2	PM1 1	PM10
– – –	0x04	Reserved								OP4	OP3	OP2	OP4_EN	OP3_EN	OP2_EN	OP1_EN	ABT_DIS
– – –	0x06	Reserved								OP4	OP3	OP2	OP4_EN	OP3_EN	OP2_EN	OP1_EN	ABT_DIS
CTL	0x08	Reserved								OP4	OP3	OP2	OP4_EN	OP3_EN	OP2_EN	OP1_EN	FAULT_EN
B&ST1	0x0A	Base1 1	Base1 0	Start1 13	Start1 12	Start1 11	Start1 10	Start1 9	Start18	Start1 7	Start16	Start15	Start14	Start13	Start12	Start1 1	Start1 0
END1	0x0C	Reserved		End113	End112	End111	End110	End19	End18	End17	End16	End1 5	End14	End13	End1 2	End1 1	End1 0
B&ST2	0x0E	Base2 1	Base2 0	Start2 13	Start2 12	Start2 11	Start2 10	Start2 9	Start28	Start2 7	Start26	Start25	Start24	Start23	Start22	Start2 1	Start2 0
END2	0x10	Reserved		End213	End212	End211	End210	End29	End28	End27	End26	End2 5	End24	End113	End2 2	End2 1	End2 0
B&ST3	0x12	Base3 1	Base3 0	Start3 13	Start3 12	Start3 11	Start3 10	Start3 9	Start38	Start3 7	Start36	Start35	Start34	Start33	Start32	Start3 1	Start3 0
END3	0x14	Reserved		End313	End312	End311	End310	End39	End38	End37	End36	End3 5	End34	End33	End3 2	End3 1	End3 0
B&ST4	0x16	Base4 1	Base4 0	Start4 13	Start4 12	Start4 11	Start4 10	Start4 9	Start48	Start4 7	Start46	Start45	Start44	Start43	Start42	Start4 1	Start4 0
END4	0x18	Reserved		End413	End412	End411	End410	End49	End48	End47	End46	End4 5	End44	End43	End4 2	End4 1	End4 0

3.18.3 Configuration Register Mapping

Table 3–141 contains the memory protection unit (MPU) configuration register mapping.

Table 3–141. MPU Register Mapping

Register Name	Address	Access	Reset Value
MPU_ST	FFFF:FF00	8 bits R	???? ???? 0000 0000
MPU_PM	FFFF:FF02	12 bits R/W	???? 0000 0000 0000

Table 3–141. MPU Register Mapping (Continued)

Register Name	Address	Access	Reset Value
MPU_CTL	FFFF:FF08	5 bits R/W	???? ???? ???0 0000
MPU_B&ST1	FFFF:FF0A	16 bits R/W	0000 0000 0000 0000
MPU_END1	FFFF:FF0C	14 bits R/W	??00 0000 0000 0000
MPU_B&ST2	FFFF:FF0E	16 bits R/W	0000 0000 0000 0000
MPU_END2	FFFF:FF10	14 bits R/W	??00 0000 0000 0000
MPU_B&ST3	FFFF:FF12	16 bits R/W	0000 0000 0000 0000
MPU_END3	FFFF:FF14	14 bits R/W	??00 0000 0000 0000
MPU_B&ST4	FFFF:FF16	16 bits R/W	0000 0000 0000 0000
MPU_END4	FFFF:FF18	14 bits R/W	??00 0000 0000 0000

3.18.4 Status Register

The MPU_ST register indicates which event initiated the fault. Table 3–142 describes the individual register bits.

Table 3–142. Status Register (MPU_ST)

Bit	Name	Function	Reset
15:8	Reserved	No effect	–
7	OP4	1: Out-of-protection access within region 4	0x0
6	OP3	1: Out-of-protection access within region 3	0x0
5	OP2	1: Out-of-protection access within region 2	0x0
4	OP1	1: Out-of-protection access within region 1	0x0
3	IA_R4	1: Illegal-access to the protected region 4	0x0
2	IA_R3	1: Illegal-access to the protected region 3	0x0
1	IA_R2	1: Illegal-access to the protected region 2	0x0
0	IA_R1	1: Illegal-access to the protected region 1	0x0

3.18.5 Control Register

The MPU_CTL register controls the MPU_FAULT signal assertion and enables out-of-protection monitoring for each region. Table 3–143 describes the individual register bits.

Table 3–143. Control Register (MPU_CTL)

Bit	Name	Function	Reset
15:5	Reserved	No effect	–
4	OP4_EN	Out-of-protection supervision within protected region 4 0: Disabled	0x0
3	OP3_EN	Out-of-protection supervision within protected region 3 0: Disabled	0x0
2	OP2_EN	Out-of-protection supervision within protected region 2 0: Disabled	0x0
1	OP1_EN	Out-of-protection supervision within protected region 1 0: Disabled	0x0
0	MPU_FAULT_EN	This read/write bit enables the MPU_FAULT signal. 0: Memory protection and out-of-protection supervision are active according to the value of the OPn_EN and PMn[2:0] bits. The status register is still recording the MPU events and remains available for reading; however, the MPU_FAULT signal is locked to a low level, not passing the fault indication to the processor (for example, abort not generated). 1: Any fault flagged to the status register initiates an MPU_FAULT signal transition (high level) indicating the fault occurrence to the processor (for example, abort generated).	0x0

3.18.6 Protection Mode Register

The MPU_PM register defines the protection associated with four possible protected regions. Table 3–144 describes the individual register bits.

Table 3–144. Protection Mode Register (MPU_PM)

Bit	Name	Function	Reset
15:12	Reserved	Reserved	–
11:9	PM4 (2:0)	Protection mode associated with region 4	0x0
8:6	PM3 (2:0)	Protection mode associated with region 3	0x0
5:3	PM2 (2:0)	Protection mode associated with region 2	0x0
2:0	PM1 (2:0)	Protection mode associated with region 1	0x0

See Table 3–139 for a description of protection modes.

3.18.7 Base and Start Address Region *n*

The MPU_B&ST(n) registers define the base and start address of the protected memory region (n). Table 3–145 describes the individual register bits.

Table 3–145. Base and Start Address Region n Registers (MPU_B&STn)

Bit	Name	Function	Reset
15:14	BASEn (1:0)	Base address for the corresponding protected memory region[n] The 2-bit (15:14) base address is compared to the MPU address bus (18:17).	0x0
13:0	STARTn (13:0)	Start address for the corresponding protected memory region[n] The 14-bit (13:0) start address is compared to the MPU address bus (16:3).	0x0000

3.18.8 End Address Definition, Region *n*

The MPU_End(n) registers define the end address of the protected memory region (n). Table 3–146 describes the individual register bits.

Table 3–146. End Address Definition Region n Registers (MPU_ENDn)

Bit	Name	Function	RESET
15:14	Reserved	No effect	–
13:0	ENDn (13:0)	End address for the corresponding protected memory region[n] The 14-bit (13:0) end address is compared to the MPU address bus (16:3).	0x0000

3.19 GSM-MPU Interrupts

The GSM-MPU owns two interrupt lines: nIRQ and nFIQ. The ABB fast interrupt is mapped on nFIQ. All peripheral interrupts are mapped as shown in Table 3–147.

Table 3–147. GSM-MPU Peripheral Interrupts Mapping (FFFF:FA00)

Name	Sense	IRQ	FIQ	Function
IRQ0	Edge	✓		Watchdog timer interrupts
IRQ1	Edge	✓		TIMER1 interrupt
IRQ2	Edge	✓		TIMER2 interrupt
IRQ3			✓	TSP receives interrupt
IRQ4	Edge	✓		TPU frame interrupt
IRQ5	Edge	✓		TPU page interrupt
IRQ6	Edge	✓		SIM interrupt 1) No answer to reset 2) Character underflow 3) Character overflow 4) Character to transmit 5) Received character 6) SIM card insertion/extraction
IRQ7	Level	✓		UART_MODEM interrupts 1) Error on receiver line 2) Receive time-out 3) Received character 4) Character to transmit 5) Modem status change 6) Received XOFF/special character detected 7) CTS/RTS/DSR deactivation 8) DSR/RXD/CTS activity detection (off mode only)
IRQ8	Level	✓		Keyboard or GPIO interrupt
IRQ9	Edge	✓		RTC periodical timer interrupt
IRQ10	Level	✓		RTC ALARM or I^2C data transfer error/completion
IRQ11	Edge	✓		ULPD end of gauging interrupt
IRQ12	Level	✓		External interrupt

Table 3–147. GSM-MPU Peripheral Interrupts Mapping (FFFF:FA00) (Continued)

Name	Sense	IRQ	FIQ	Function
IRQ13	Edge	✓		SPI interrupt 1) Received data 2) Data to transmit
IRQ14	Level	✓		DMA interrupt
IRQ15	Edge	✓		MPUI interrupts (nHINT)
IRQ16			✓	SIM card-detect fast interrupt
IRQ17			✓	Fast external interrupt
IRQ18		✓		Reserved
IRQ19	Level	✓		ULPD GSM timer
IRQ20	Level	✓		GEA interrupt
IRQ21–22				Reserved
IRQ23	Level or edge	✓		GSM edge extern MPU
IRQ24	Level		✓	GSM protect
IRQ25				Reserved
IRQ26	Edge			ICR interrupt
IRQ27	Edge			TCIF GSM-MPU memory access error

3.19.1 Interrupt Sequence

As IRQ and FIQ receive identical treatment, the following sequence is described only for IRQ interrupt.

1) One or several incoming interrupts go down, setting the corresponding ITR bits.

2) There are two possible cases:

   - There is only one incoming interrupt, which is active. If IRQ is not already active, the interrupt handler sends an IRQ.
   - There are several active incoming interrupts. In this case, the interrupt handler must determine which new interrupt is to be serviced. To do this, it compares the priority level of an interrupt with the one held in a dedicated register (N_IRQ) and stores the one having the highest priority in N_IRQ. It performs this until all the active interrupts have been processed. If IRQ is not already active, the interrupt handler sends an IRQ.

3) When an IRQ is sent, the SIR_IRQ register is updated (indicating the interrupt contained in N_IRQ) and the priority resolver is reset (and restarted if necessary).

4) To know which incoming interrupt requested an GSM-MPU action, the GSM-MPU must read the SIR_IRQ_CODE register. After that, it runs the corresponding subroutine.

5) To finish this sequence, the GSM-MPU software must set a dedicated bit (NEW_IRQ_AGR of control register) in order to reset the IRQ output and the SIR_IRQ register, and thus allow a new IRQ generation.

3.19.2 GSM-MPU Interrupt Registers

These registers are controlled directly by the internal TIPB. The handler is selected when CS(2:0) is equal to 2. Table 3–148 lists the GSM-MPU interrupt registers. Table 3–149 through Table 3–157 describe the register bits.

Table 3–148. GSM-MPU Interrupt Registers

Name	Description	Address	Size	Type	Reset Value
IT_REG1	Interrupt 1	FFFF:FA00	16 bits	R	0000 0000 0000 0000
IT_REG2	Interrupt 2	FFFF:FA02	4 bits	R	???? ???? ???0 0000
MASK_IT_REG1	Mask interrupt 1	FFFF:FA08	16 bits	R/W	1111 1111 1111 1111
MASK_IT_REG2	Mask interrupt 2	FFFF:FA0A	4 bits	R/W	1111 1111 1111 1111
SRC_IRQ_BIN_REG	Indicates interrupt number that requested GSM-MPU action	FFFF:FA10	5 bits	R	???? ???? ???0 0000
SRC_FIQ_BIN_REG	Indicates interrupt number that requested GSM-MPU action	FFFF:FA12	5 bits	R	???? ???? ???0 0000
INT_CTRL_REG	Interrupt control	FFFF:FA14	2 bits	R/W	???? ???? ???? ??00
ILR_IRQ0_REG	Interrupt level 0	FFFF:FA20	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ1_REG	Interrupt level 1	FFFF:FA22	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ2_REG	Interrupt level 2	FFFF:FA24	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ3_REG	Interrupt level 3	FFFF:FA26	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ4_REG	Interrupt level 4	FFFF:FA28	7 bits	R/W	???? ???? ?000 0000

Table 3–148. GSM-MPU Interrupt Registers (Continued)

Name	Description	Address	Size	Type	Reset Value
ILR_IRQ5_REG	Interrupt level 5	FFFF:FA2A	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ6_REG	Interrupt level 6	FFFF:FA2C	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ7_REG	Interrupt level 7	FFFF:FA2E	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ8_REG	Interrupt level 8	FFFF:FA30	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ9_REG	Interrupt level 9	FFFF:FA32	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ10_REG	Interrupt level 10	FFFF:FA34	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ11_REG	Interrupt level 11	FFFF:FA36	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ12_REG	Interrupt level 12	FFFF:FA38	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ13_REG	Interrupt level 13	FFFF:FA3A	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ14_REG	Interrupt level 14	FFFF:FA3C	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ15_REG	Interrupt level 15	FFFF:FA3E	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ16_REG	Interrupt level 16	FFFF:FA40	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ17_REG	Interrupt level 17	FFFF:FA42	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ18_REG	Interrupt level 18	FFFF:FA44	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ19_REG	Interrupt level 19	FFFF:FA46	7 bits	R/W	???? ???? ?000 0000
ILR_IRQ20_REG	Interrupt level 20	FFFF:FA48	7 bits	R/W	???? ???? ?000 0000

The GSM-MPU interrupt registers store an incoming interrupt in case of an edge-sensitive interrupt. When the GSM-MPU accesses the SIR_IRQ_CODE or SIR_FIQ_CODE register, the bit corresponding to the interrupt requesting GSM-MPU action is reset.

The GSM-MPU can also clear each bit individually. To do this, the GSM-MPU must write 0s to the corresponding bits (at ITR address); other bits keep their

previous values. This possibility can be used just before the GSM-MPU unmasks some interrupts and thus, allows to forget some interrupt occurrences.

The GSM-MPU can read this register. If an incoming interrupt is edge sensitive, the read value corresponds to the value held in the storage element. Otherwise, the read value corresponds to the inverted value of the incoming interrupt.

Table 3–149. Interrupt Register 1 (IT_REG1)

Bit	Name	Function	Reset
15:0	IRQ15...IRQ0	Indicates interrupt stored	0

Table 3–150. Interrupt Register 2 (IT_REG2)

Bit	Name	Function	Reset
15:5	Reserved	Reserved	0
4	IRQ20	Indicates interrupt stored	0
3	IRQ19	Indicates interrupt stored	0
2	IRQ18	Indicates interrupt stored	0
1	IRQ17	Indicates interrupt stored	0
0	IRQ16	Indicates interrupt stored	0

Table 3–151. Mask Interrupt Register 1 (MASK_IT_REG1)

Bit	Name	Function	Interrupt Source (Sec. 3.19.2)	Reset
15	IRQ_15_MSK	Disables IRQ_15 interrupt	MPUI	0x1
14	IRQ_14_MSK	Disables IRQ_14 interrupt	DMA	0x1
13	IRQ_13_MSK	Disables IRQ_13 interrupt	SPI	0x1
12	IRQ_12_MSK	Disables IRQ_12 interrupt	External	0x1
11	IRQ_11_MSK	Disables IRQ_11 interrupt	ULPD end of gauging	0x1
10	IRQ_10_MSK	Disables IRQ_10 interrupt	RTC alarm or I^2C	0x1
9	IRQ_9_MSK	Disables IRQ_9 interrupt	RTC periodical timer	0x1
8	IRQ_8_MSK	Disables IRQ_8 interrupt	Keyboard	0x1
7	IRQ_7_MSK	Disables IRQ_7 interrupt	UART modem	0x1
6	IRQ_6_MSK	Disables IRQ_6 interrupt	SIM	0x1
5	IRQ_5_MSK	Disables IRQ_5 interrupt	TPU page	0x1
4	IRQ_4_MSK	Disables IRQ_4 interrupt	TPU frame	0x1
3	IRQ_3_MSK	Disables IRQ_3 interrupt	TSP receive	0x1
2	IRQ_2_MSK	Disables IRQ_2 interrupt	TIMER2	0x1

Table 3–151. Mask Interrupt Register 1 (MASK_IT_REG1) (Continued)

Bit	Name	Function	Interrupt Source (Sec. 3.19.2)	Reset
1	IRQ_1_MSK	Disables IRQ_1 interrupt	TIMER1	0x1
0	IRQ_0_MSK	Disables IRQ_0 interrupt	Watchdog timer	0x1

Table 3–152. Mask Interrupt Register 2 (MASK_IT_REG2)

Bit	Name	Function	Interrupt Source	Reset
15:5	Reserved	Reserved	None	0xFFF
4	IRQ_20_MSK	Disables IRQ_20 interrupt	GEA module	0x1
3	IRQ_19_MSK	Disables IRQ_19 interrupt	ULPD GSM timer	0x1
2	Reserved	Reserved	None	0x1
1	IRQ_17_MSK	Disables IRQ_17 interrupt	Fast external	0x1
0	IRQ_16_MSK	Disables IRQ_16 interrupt	SIM card detect	0x1

Each incoming interrupt can be masked individually by MASK_IT_REG1 and MASK_IT_REG2. MIR operates after ITR: occurrences of incoming interrupts are always stored in ITR.

Table 3–153. Source IRQ Binary Coded Register (SRC_IRQ_BIN_REG)

Bit	Name	Function	Reset
4:0	IRQ_NUM	Active current IRQ interrupt	0x00

The source IRQ binary coded register indicates the active interrupt. In order to save software processing time, this register indicates the interrupt number having requested an GSM-MPU action.

Table 3–154. Source FIQ Binary Coded Register (SRC_FIQ_BIN_REG)

Bit	Name	Function	Reset
4:0	FIQ_NUM	Active current FIQ interrupt	0x00

The source FIQ binary coded register indicates the active interrupt. In order to save software processing time, this register indicates the interrupt number having requested an GSM-MPU action.

Table 3–155. Interrupt Control Register (INT_CTRL_REG)

Bit	Name	Function	Reset
1	NEW_FIQ_AGR	New FIQ agreement Resets FIQ output Clears source FIQ Enables a new FIQ generation Active at level 1 Reset by internal logic	0x0
0	NEW_IRQ_AGR	New IRQ agreement Resets IRQ output Clears source IRQ Enables a new IRQ generation Active at level 1 Reset by internal logic	0x0

Note:

IRQ (FIQ) output, SIR_IRQ, and SIR_IRQ_CODE (SIR_FIQ and SIR_FIQ_CODE) register are reset only if the bit of IT register corresponding to the interrupt having requested GSM-MPU action is already cleared or masked. The time when this bit is reset depends on the sensitivity of the incoming interrupt. In case of edge-sensitive interrupt, the IT register bit is deactivated when reading SIR_IRQ or SIR_IRQ_CODE (SIR_FIQ or SIR_FIQ_CODE) register. Otherwise, it is reset when the corresponding interrupt becomes inactive.

Table 3–156. Interrupt Level Registers and Sources

Name	Corresponding Interrupt	Interrupt Source	Offset Address (hex)
ILR_IRQ_0	IRQ_0	Watchdog timer	20
ILR_IRQ_1	IRQ_1	TIMER1	22
ILR_IRQ_2	IRQ_2	TIMER2	24
ILR_IRQ_3	IRQ_3	TSP receive	26
ILR_IRQ_4	IRQ_4	TPU frame	28
ILR_IRQ_5	IRQ_5	TPU page	2A
ILR_IRQ_6	IRQ_6	SIM	2C
ILR_IRQ_7	IRQ_7	UART modem	2E
ILR_IRQ_8	IRQ_8	Keyboard	30
ILR_IRQ_9	IRQ_9	RTC periodical timer	32
ILR_IRQ_10	IRQ_10	RTC alarm or I^2C	34
ILR_IRQ_1	IRQ_11	ULPD end of gauging	36
ILR_IRQ_12	IRQ_12	External	38

Table 3–156. Interrupt Level Registers and Sources (Continued)

Name	Corresponding Interrupt	Interrupt Source	Offset Address (hex)
ILR_IRQ_13	IRQ_13	SPI	3A
ILR_IRQ_14	IRQ_14	DMA	3C
ILR_IRQ_15	IRQ_15	MPUI	3E
ILR_IRQ_16	IRQ_16	SIM card detect	40
ILR_IRQ_17	IRQ_17	Fast external	42
ILR_IRQ_18	IRQ_18	Reserved	44
ILR_IRQ_19	IRQ_19	ULPD GSM timer	46
ILR_IRQ_20	IRQ_20	GEA module	48

Table 3–157. Interrupt Level Registers (ILR_IRQ0...ILR_IRQ20)

Bit	Name	Function	Reset
6:2	PRIORITY	Priority level when the corresponding interrupt is routed to IRQ: 0: Highest-priority level 1: Lowest-priority level	0x00
1	SENS_EDGE	1: The corresponding interrupt is falling-edge sensitive. 0: The corresponding interrupt is low-level sensitive.	0x0
0	FIQ	1: The corresponding interrupt is routed to FIQ. 0: The corresponding interrupt is routed to IRQ.	0x0

3.19.2.1 Predefined Order in Case of Identical Priority Level

Assuming that all interrupts have the same priority level and are active at the same time, the order of servicing is: IRQ_N–1 , IRQ_N–2, …. IRQ_0.

3.19.3 GSM-MPU To TIPB Registers

These registers are controlled directly by the internal TIPB. The GSM-MPU-TIPB bridge is selected when CS(2:0) is equal to 1. Table 3–158 lists the GSM-MPU TIPB registers. Table 3–159 through Table 3–163 describe the register bits.

Table 3–158. GSM-MPU to TIPB Registers

Name	Description	Address	Size	Type	HW Reset Value
RHEA_CNTL_REG	TIPB control	FFFF:F900	16 bits	R/W	1111 1111 0010 0000
API_WS_REG	MPUI wait state	FFFF:F902	10 bits	R/W	???? ??11 1110 0000

Table 3–158. GSM-MPU to TIPB Registers (Continued)

Name	Description	Address	Size	Type	HW Reset Value
ARM_RHEA_CNTL_REG	APU TIPB control	FFFF:F904	2 bits	R/W	???? ???? ???? ??11
ENHANCED_RHEA_CTL	Enhanced TIPB control	FFFF:F906	1 bit	R/W	???? ???? ???? ???1

Table 3–159. TIPB Control Register (RHEA_CNTL_REG)

Bit	Name	Function	Reset
15:8	TIMEOUT†	TIPB access time-out Allows limiting the maximum time a peripheral can stall the processor. When starting a cycle on TIPB, the time-out counter is loaded with this value. If the current cycle is not finished when the counter reaches 0, the cycle is aborted (by sending ARM_ABORT to the GSM-MPU or DMA_ABORT to the DMA and AABORT to the peripheral).	0xFF
7:4	ACCESS_ FACTOR1	Division factor of nASTROBE(1). Allows accessing slow peripherals by reducing the access frequency.	0x2
3:0	ACCESS_ FACTOR0	Division factor of nASTROBE(0). Allows accessing slow peripherals by reducing the access frequency.	0x0

† nASTROBE low-level pulse duration:
– Access_factor = 0 → bridge_clk low level
– Access_factor != 0 → access_factor ⇔ number of bridge_clk periods to use
– Maximum value is 256, which allows a time-out of 6.56 µs if bridge_clk is equal to 39 MHz

Once the GSM-MPU clock frequency has been selected, programmers must adapt the duration of GSM-MPU accesses to the timing of peripherals connected on the TIPB. For this purpose the RHEA_CNTL_REG defines an access factor, which allows adapting the TIPB access duration to a slow peripheral.

- ACCESS_FACTOR0(3:0) allows matching the access duration to a slow peripheral on strobe 0.
- ACCESS_FACTOR1(7:4) allows matching the access duration to a slow peripheral on strobe 1.

The value for the access factors is obtained using the following formula:

$$\text{Access\_Factor} \geq \frac{\text{ARM\_Access\_Freq}}{\text{Pheripheral\_Access\_Freq}} - 1$$

Note:

Peripheral_Access Freq value:

For rev. A: 39 MHz for all peripherals except GEA = 36 MHz and DMA = 34 MHz
For rev. B: 39 MHz for all peripherals

Table 3–160. MPUI Wait State Register (API_WS_REG)

Bit	Name	Function	Reset
9:5	API_WS_S	Indicates the number of wait states inserted for each MPUI access when DSP is in SAM	0x1F
4:0	API_WS_H	Indicates the number of wait states inserted for each MPUI access when DSP is in HOM	0x00

The MPUI memory is a dual-access memory that can be configured in two states:

- ❑ Host-only mode (HOM) with the memory dedicated to the GSM-MPU (no access possible from DSP)
- ❑ Shared access mode (SAM) with the memory access shared between GSM-MPU and DSP

In HOM, the GSM-MPU access to the memory is fully asynchronous and does not impose any time constraints on the GSM-MPU access frequency; nevertheless, the GSM-MPU access frequency must take care of the access time of the MPUI memory (technology dependent).

In SAM, the GSM-MPU access is resynchronized on the DSP cycle clock and the duration of the GSM-MPU access must obey the following rule:

$LEAD\_CLK\_freq \geq 4 \times ARM\_ACCESS\_freq$

If the GSM-MPU cycle frequency is not compliant with this rule, then wait states must be inserted during the GSM-MPU access to increase the access time duration.

Register API_WS_REG defines the number of wait states inserted during an GSM-MPU access to MPUI memory depending on DSP mode:

- ❑ API_WS_H defines the wait states inserted when DSP is in HOM mode.
- ❑ API_WS_S defines the wait states inserted when DSP is in SAM mode.

The number of wait states can be calculated using the following formulas:

$$WS\_S = \left(4 \times \frac{MCU\_frq}{DSP\_frq}\right) - 1 \qquad WS\_H = \left(\frac{MCU\_frq}{Mem\_frq}\right) - 1$$

Where: Mem_frq = 40 MHz

The HOM/SAM is selected by the DSP itself. Information is given to the GSM-MPU interface-to-MPUI by setting/clearing the 1 bit in the API_CONF register. The selected wait state register also depends of the value of bit 5 in CNTL_CLK register.

The number of wait-states is selected according to Table 3–161.

Table 3–161. Wait States

cntl_rst bit(1)	DSP in IDLE3	API_CONF bit(1)	Selected
1 (DSP reset)	x	x	API_WS_H
0 (DSP run)	Yes		
	No	1 (HOM)	API_WS_H
		0 (SAM)	API_WS_S

3.19.3.1 MPU TIPB Control Register

Table 3–162. MPU/TIPB Control Register (ARM_RHEA_CTL_REG)

Bit	Name	Function	Reset
1	W_BUF_EN_1	0: Write buffer is bypassed. 1: Write buffer is enabled for strobe domain 1.	0x1
0	W_BUF_EN_0	0: Write buffer is bypassed. 1: Write buffer is enabled for strobe domain 0.	0x1

Table 3–163. Enhanced TIPB Control Register (ENHANCED_RHEA_CTL)

Bit	Name	Function	Reset
0	TIMEOUT_EN	0: Time-out disabled 1: Time-out enabled	0x1

3.19.3.2 Maximum Latency for Each Peripheral

There are two formulas for latency calculation, depending on whether or not synchro-block is used in the design of the peripheral. If the peripheral uses synchro-read and/or synchro-write block, the formula is:

$$\text{MaxLatency} = \left\lceil \frac{2 \times \frac{\text{Fmcu}}{\text{Fperi}}}{\text{AF} + 1} \right\rceil + 3$$

Else the formula is:

MaxLatency = (AF + 1)

- The latency is in GSM-MPU clock periods.
- *Fmcu* is the GSM-MPU input frequency programmed through DPLL control and CNTL_ARM_CLK registers.
- AF is the TIPB access factor programmed through the Rhea_cntl_reg register.
- *Fperi* is the peripheral usage frequency (described in the following table).

Table 3–164 describes the peripheral frequency usage, the use/nonuse of synchro block, and the maximum latency for each peripheral.

Table 3–164. Maximum Peripheral Latency

Module Name	Synchro Block		Peripheral Frequency Fperiph (MHz)	Maximum Latency (GSM-MPU Cycle)		
	Read	Write		AF = 0	AF = 1	AF = 2
DMA	√		13	9	6	5
I^2C	√		13	9	6	5
MCSI	√	√	13	9	6	5
SPI_13	√	√	13	9xPVT	6xPVT	5xPVT
TIMER1	√	√	13	9	6	5
TIMER2	√		13	9	6	5
UART modem	√	√	13	9	6	5
ULPD	√	√	13/3	21	12	9
A51/2		√	13	9	6	5
GEA		√	13	9	6	5
RIF		√	13	9	6	5
TPU			13	1	2	3
DPLL			13	1	2	3
WATCHDOG			13	1	2	3
TIPB bridge			13	1	2	3
INTH			13	1	2	3
MEM. IF			13	1	2	3
CLKM			13	1	2	3
MPU			13	1	2	3
SIM			13	1	2	3
TSP			13	1	2	3
RTC			13	1	2	3
µWire			13	1	2	3
MPUIO			13	1	2	3
RTC			13	1	2	3
LPG			13	1	2	3

3.19.4 DPLL Register

Table 3–165. DPLL Register (FFFF:9800)

Name	Description	Address	Size	Type	Reset Value
DPLL_CTRL	DPLL control	FFFF:9800	16 bits	R/W	0x2806

3.19.5 DPLL Operation

The DPLL has two modes of operation: bypass mode and lock mode.

3.19.5.1 Bypass Mode

In the bypass mode, CLKOUT is equal to CLKREF divided by 1, 2, or 4. This mode can be used to save power, since the DPLL is disabled. This mode also provides an output clock while the DPLL circuitry is locking.

$$Clk_{out} = \frac{F_{in}}{k} \qquad k = 1,\ 2,\ \text{or}\ 4$$

3.19.5.2 Lock Mode

In the lock mode, the DPLL provides a synthesized output frequency which is locked to the input reference. The lock mode is entered if the PLL_ENABLE bit of the control register is set and the locking sequence is completed. In this mode, the CLKOUT contains a synthesized clock frequency as defined below:

$$Clk_{out} = F_{in} \times \frac{PLL\_MULT}{PLL\_DIV + 1} \qquad 1 < PLL\_MULT < 31 \qquad PLL\_DIV = 0,\ 1,\ \text{or}\ 2$$

3.19.6 Lock Times

The lock times depend on the values of PLL_MULT and PLL_DIV and the CLKOUT frequency as given by the following equation (in number of CLKREF cycles):

$$Lt = \frac{[4 \times (PLL\_DIV + 1) \times 11D] + 20}{F_{in}}$$

where: $$D = 1 + \log 2\left(\frac{PLL\_DIV + 1}{PLL\_MULT \times Clk_{out} \times \aleph\ \text{min}}\right)$$

where: $\aleph\ \text{min} = 5 \times 10^{-9}$

$F_{in} = 13$ MHz

3.19.7 Control Register Access

Every time the DPLL control register is written to, the mode in which the DPLL operates can change automatically. The DPLL recognizes that its control register has been modified on the rising edge of the nstrobe signal when the address matches that of the control register in the TIPB address space. If the DPLL is operating in the synthesized mode, it switches automatically to the bypass mode. Depending on the new control content, the DPLL may either initiate a new lock sequence or remain in the bypass mode.

Table 3–166. DPLL Control Register (DPLL_CTRL)

Bit	Name	Function	Access	Reset
15	Reserved		R	0x0
14	IAI	DPLL initialize after idle: 0: Attempt locking using the same internal delay chain setting which existed prior to entering the idle mode. 1: Start the entire locking sequence over after idle is deactivated.	R/W	0x0
13	IOB	DPLL initialize on break: 0: Continue to output the synthesized clock even if the core indicates it has lost the lock but BREAKLN is active low. 1: Switch to bypass mode and start a new locking sequence if the DPLL core indicates loss of lock.	R/W	0x1
12	TEST	Control test out clock on tclkout pin: 0: CLKOUT 1: CLKOUT/32 x: TCLKOUT is 0 when not in test mode.	R/W	0x0
11:7	PLL_MULT	DPLL multiply value: Range 0 to 31	R/W	0x10
6:5	PLL_DIV	DPLL divide value: 00: CLKREF 01: CLKREF/2 10: CLKREF/3 11: CLKREF/4 When PLL_MULT is equal to 0 or 1, the output clock is not synthesized by the PLL but is simply a divided-down version of clkref.	R/W	0x0
4	PLL_ENABLE	PLL enable: 0: Disabled (switchback bypass mode) 1: Enabled Requests the DPLL to enter the lock mode. It enters the lock mode only after it has synthesized the desired frequency.	R/W	0x0
3:2	BYPASS_DIV	Clock-out frequency in bypass mode: 00: CLKREF 01: CLKREF/2 1x: CLKREF/4	R/W	0x1

Table 3–166. DPLL Control Register (DPLL_CTRL) (Continued)

Bit	Name	Function	Access	Reset
1	BREAKLN	Lock status: 0: Broken lock 1: When lock restored or write to control occurs	R	0x1
0	LOCK	PLL mode: 0: Bypass 1: Locked	R	0x0

Writing to the control register causes the DPLL to immediately switch to the bypass mode if not in idle state. If the PLL_ENABLE signal is set, it begins its sequence to enter the locked mode. This prevents changing the multiply or divide value without reentering the DPLL lock sequence. The register bit positions in Table 3–166 are after the big-to-little-endian conversion in the TIPB interface. Externally, the position of bits 15:8 and 7:0 must be swapped before the internal TIPB interface converts them.

3.19.8 Timer Registers

Table 3–167 lists the timer registers. Table 3–168 through Table 3–173 describe the register bits.

Table 3–167. Timer Registers (FFFE:3800/FFFE:6800)

Name	Description	Address	Size	Type	Reset Value
CNTL_TIMER1	Timer1 control	FFFE:3800	8 bits	R/W	0000 0000
LOAD_TIM1	Timer1 load	FFFE:3802	16 bits	R/W	???? ???? ???? ????
READ_TIM1	Timer1 read	FFFE:3804	16 bits	R	???? ???? ???? ????
CNTL_TIMER2	Timer1 control	FFFE:6800	8 bits	R/W	0000 0000
LOAD_TIM2	Timer1 load	FFFE:6802	16 bits	R/W	???? ???? ???? ????
READ_TIM2	Timer1 read	FFFE:6804	16 bits	R	???? ???? ???? ????

Table 3–168. Timer 1 Control Register (CNTL_TIMER1)

Bit	Name	Function	Reset
15:8	Reserved		–
7	SOFT	Soft bit. Used in conjunction with the FREE bit to determine the state of the peripheral when a breakpoint is encountered. This bit is used in the emulation mode. 0: The peripheral halts immediately, either retaining or discarding the current state. 1: The peripheral stops after completion of the current task.	0x0
6	FREE	Free bit. Used in conjunction with the SOFT bit to determine the state of the peripheral when a breakpoint is encountered. This bit is used in the emulation mode. 0: The SOFT bit selects the emulation mode. 1: The peripheral clock runs free regardless of the SOFT bit.	0x0
5	CLOCK_ENABLE	External timer clock enable	0x0
4:2	PTV	Prescale clock timer value	0x0
1	AR	Reload TIMER1: 1: Autoreload TIMER1 0: One-shot TIMER1	0x0
0	ST†	Start TIMER1: 1: Start TIMER1 0: Stop TIMER1	0x0

† If one-shot mode is selected (AR=0), the ST bit is automatically reset by internal logic when the timer is equal to 0.

Table 3–169. Load Timer 1 Register (LOAD_TIM1)

Bit	Name	Function	Reset
15:0	LOAD_TIM1	This value is loaded when TIMER1 passes through 0 or when it starts.	Undefined

Table 3–170. Read Timer 1 Register (READ_TIM1)

Bit	Name	Function	Reset
15:0	VALUE_TIM1	Value of TIMER1	Undefined

Table 3–171. Timer 2 Control Register (CNTL_TIMER2)

Bit	Name	Function	Reset
15:8	Reserved		–
7	SOFT	SOFT bit. Used in conjunction with the FREE bit to determine the state of the peripheral when a breakpoint is encountered. This bit is used in the emulation mode. 0: The peripheral halts immediately, either retaining or discarding the current state. 1: The peripheral stops after completion of the current task	0x0
6	FREE	FREE bit. Used in conjunction with the SOFT bit to determine the state of the peripheral when a breakpoint is encountered. This bit is used in the emulation mode. 0: The SOFT bit selects the emulation mode. 1: The peripheral clock runs free regardless of the SOFT bit.	0x0
5	CLOCK_ENABLE	External TIMER2 clock enable	0x0
4:2	PTV	Prescale clock TIMER2 Value	0x0
1	AR	Reload TIMER2: 1: Autoreload timer 0: One-shot timer	0x0
0	ST†	Start TIMER2: 1: Start timer 0: Stop timer	0x0

† If one-shot mode is selected (AR=0), the ST bit is automatically reset by internal logic when the timer is equal to 0.

Table 3–172. Load Timer Register (LOAD_TIM2)

Bit	Name	Function	Reset
15:0	LOAD_TIM2	This value is loaded when the timer passes through 0 or when it starts.	Undefined

Table 3–173. Read Timer Register (READ_TIM2)

Bit	Name	Function	Reset
15:0	VALUE_TIM2	Value of TIMER2	Undefined

3.19.9 Watchdog Timer Registers

Table 3–174 shows the watchdog registers. Table 3–175 through Table 3–178 describe the register bits.

Table 3–174. Watchdog Registers (FFFF:F800)

Name	Description	Address	Access	Type	HW Reset Value
WATCHDOG_CNTL_TIM	Timer control	FFFF:F800	6 bits	R/W	???? 0000 0??? ??1?
WATCHDOG_LOAD_TIM	Load timer	FFFF:F802	16 bits	W	1111 1111 1111 1101
WATCHDOG_READ_TIM	Read timer	FFFF:F802	16 bits	R	1111 1111 1111 1101
WATCHDOG_TIM_MODE	Timer mode	FFFF:F804	9 bits	W	1??? ???? ???? ????

Table 3–175. Timer Control Register (WATCHDOG_CNTL_TIM)

Bit	Name	Function	Reset
15:12	Reserved	–	–
11:9	PTV	Prescale clock timer value	0x0
8	AR	Timer reload: 1: Autoreload timer 0: One-shot timer	0x0
7	ST	Timer start: 1: Start timer 0: Stop timer	0x0
6:0	Reserved		–

† If one-shot mode is selected (AR=0), the ST bit is automatically reset by internal logic when the timer is equal to 0.

Table 3–176. Load Timer Register (WATCHDOG_LOAD_TIM)

Bit	Name	Function	Reset
15:0	LOAD_TIM	General-purpose timer: This value is loaded when the timer passes through 0 or when it starts. Watchdog timer: Reload timer with this value.	0xFFFD

Table 3–177. Read Timer Register (WATCHDOG_READ_TIM)

Bit	Name	Function	Reset
15:0	VALUE_TIM	Value of timer	0xFFFD

Table 3–178. Timer Mode Register (WATCHDOG_TIM_MODE)

Bit	Name	Function	Reset
15	WATCHDOG	Write access:. 1: Switch back timer mode to watchdog Read access: Status of timer mode: 0: Timer is a general-purpose counter. 1: Timer is a watchdog timer.	0x1
7:0	WATCHDOG_DIS	Write access only Writing a predefined sequence (0xF5 followed by 0xA0) in this field disables watchdog function. After receiving 0xF5, if the second write access is different from 0xA0, the GSM-MPU core is reset (via RST_CMD output)	–

3.19.10 SPI Registers

Table 3–179 lists the SPI registers, mapped in the I/O port space of the MPU. Table 3–180 through Table 3–185 describe the register bits.

Table 3–179. SPI Registers (FFFE:3000)

Name	Description	Address	Size	Type	Reset Value
REG_SPI_SET1	Serial port 1 setup	FFFE:3000	6 bits	R/W	1111 1111 ??11 0000
REG_SPI_SET2	Serial port 2 setup	FFFE:3002	15 bits	R/W	1000 0000 0000 0000
REG_SPI_CTRL	Serial port interface control	FFFE:3004	10 bits	R/W	1111 1100 0000 0000
REG_STATUS	SPI status	FFFE:3006	2 bits	R	1111 1111 1111 1100
REG_TX_LSB	Data to transmit (word low)	FFFE:3008	16 bits	R/W	0000 0000 0000 0000
REG_TX_MSB	Data to transmit (word high)	FFFE:300A	16 bits	R/W	0000 0000 0000 0000
REG_RX_LSB	Data to receive (word low)	FFFE:300C	16 bits	R	0000 0000 0000 0000
REG_RX_MSB	Data to receive (word high)	FFFE:300E	16 bits	R	0000 0000 0000 0000

The serial port offers input and output registers for loading data to serialize (transmit) or reading paralleled data (receive), respectively.

Table 3–180. Serial Port 1 Setup Register (REG_SPI_SET1)

Bit	Name	Function	Reset
15:6	Reserved		–
5	MSK1	Enables interrupt for read/write cycle 0: Interrupt active 1: Interrupt disabled	0x1
4	MSK0	Enables interrupt for write cycle 0: Interrupt active 1: Interrupt disabled	0x1
3:1	PTV	Prescale clock divisor: 000: 1 001: 2 010: 4 011: 8 100: 16	0x0
0	EN_CLK	Clock enable 0: Clock is shut off. 1: Clock is running.	0x0

REG_SPI_SET1 is dedicated to the configuration of the serial port.

Table 3–181. Serial Port 2 Setup Register (REG_SPI_SET2)

Bit	Name	Function	Reset
15	Reserved		–
14:10	L	Format of enable signals nTSPEN 0: Level trigger 1: Edge trigger	0x00
9:5	P	Format of enable signals nTSPEN 0: Negative level 1: Positive level	0x00
4:0	C	Active edge of the clock for each device in TX 0: Falling 1: Rising	0x00

REG_SPI_SET2 is dedicated to the configuration of the serial port.

Table 3–182. Serial Port Interface Control Register (REG_SPI_CTRL)

Bit	Name	Function	Reset
15:10	Reserved		–
9:7	AD	Addressed device index (5 devices maximum)	0x0

Table 3–182. Serial Port Interface Control Register (REG_SPI_CTRL) (Continued)

Bit	Name	Function	Reset
6:2	NB	Transmission length of NB+1 bits: 00000: 1-bit transmit ..... 11111: 32-bit transmit	0x00
1	WR	Write process activation (toggle at 1)	0x0
0	RD	Read and write process activation (toggle at 1)	0x0

REG_SPI_CTRL is dedicated to the activation of the serial port and starts the operation of the interface as soon as one of its two bits is set. It defines:

- ❏ Write activation of the serial port (transmit only)
- ❏ Read activation of the serial port (simultaneously receive and transmit)
- ❏ Number of bits to transfer (in the range 1 to 32)
- ❏ External device address (between 0 and 5)

Table 3–183. Status Register (REG_STATUS)

Bit	Name	Function	Reset
15:2	Reserved		–
1	WE	Write end 1: The serialization is finished.	0x0
0	RE	Read end 1: Receives loaded	0x0

To read the status register or to write to setup register 2, the internal clock must be running (reg_set1 (0) = 1).

Note:

A reset value of 0 does not mean that transfer is ongoing.

Table 3–184. Data to Transmit Registers (REG_TX_LSB/MSB)

Bit	Name	Function	Reset
15:0	REG_TX_LSB	Data to transmit (word low)	0x0000
15:0	REG_TX_MSB	Data to transmit (word high)	0x0000

The data to transmit are loaded in two word registers (REG_TX_MSB and REG_TX_LSB). These registers are accessible to the TIPB in read or write. Table 3–184 describes the individual register bits.

Table 3–185. Data to Receive Registers (REG_RX_LSB/MSB)

Bit	Name	Function	Reset
15:0	REG_RX_LSB	Receive data (word low)	0x0000
15:0	REG_RX_MSB	Receive data (word high)	0x0000

The received data are accessible from the TIPB through two 16-bit registers (REG_RX_MSB and REG_RX_LSB).

This choice of implementation requires that, whatever its size, the word must be aligned on the MSB side.

3.19.11 µWire Registers

Table 3–186 lists the wire registers, mapped in the I/O port space of the GSM-MPU. Table 3–187 through Table 3–192 describe the register bits.

Table 3–186. µWire Registers (FFFE:4000)

Name	Description	Address	Size	Type	Reset Value
TDR	µWire transmit data	FFFE:4000	16 bits	W	???? ???? ???? ????
RDR	µWire receive data	FFFE:4000	16 bits	R	???? ???? ???? ????
CSR	µWire control and status	FFFE:4002	16 bits	R/W	0000 ???? ???? ????
SR1	µWire setup 1	FFFE:4004	12 bits	R/W	???0 ???? ?0??
SR2	µWire setup 2	FFFE:4006	12 bits	R/W	???0 ???? ?0??
SR3	µWire setup 3	FFFE:4008	3 bits	R/W	000

Table 3–187. µWire Transmit Data Register (TDR)

Bit	Name	Function	Reset
15:0	TD	Data to transmit	Undefined

The µWire transmit data register contains the data to transmit. MSB (bit 15) is the first transmitted bit. Whatever its size, the word must be aligned on the MSB side.

Table 3–188. µWire Receive Data Register (RDR)

Bit	Name	Function	Reset
15:0	RD	Received data	Undefined

The µWire receive data register contains the received data. LSB (bit 0) is the last received bit. Whatever its size, the word is aligned on the LSB side.

Table 3–189. µWire Control and Status Register (CSR)

Bit	Name	Function	Access	Reset
15	RDRB	1: Receive (RDR) is full. This bit is cleared when the controller reads the content of the RDR.	R	0x0
14	CSRB	0: The control and status (CSR) is ready to receive new data. This bit is set to 1 after writing to the control and status, either the CS_CMD or the START bit. CSRB is reset when the corresponding action has been performed.	R	0x0

Table 3–189. μWire Control and Status Register (CSR) (Continued)

Bit	Name	Function	Access	Reset
13	START	Start process: 1: Starts a write and/or a read process This bit is automatically reset by internal logic when a write or a read process is activated. Send NB_BITS_WR bits (contained in TDR) to the serial output DO. If NB_BITS_WR is equal to zero, then the write process is not started. Receive NB_BITS_RD bits from the serial input DI and store them in RDR,	R/W	0x0
12	CS_CMD	Chip-select: 1: Sets the chip-select of the selected device to its active level	R/W	0x0
11:10	INDEX	External device index: 00: CS0 10: CS2 01: CS1 11: CS3	R/W	Undefined
9:5	NB_BITS_WR	Number of bits to transmit	R/W	Undefined
4:0	NB_BITS_RD	Number of bits to receive	R/W	Undefined

Table 3–190. μWire Setup Register 1 (SR1)

Bit	Name	Function	Reset
11	CS1_CHK	Checks whether external device is ready before activating a write process: 1: If DI signal is low, the interface considers that the external component is busy, if DI is high, the interface considers that the first external component is ready and starts the write process. 0: No check is performed and the write process is immediately executed. Used when CS1 is selected.	Undefined
10:9	CS1_FRQ	SCLK clock frequency when the CS1 is selected: 00: F_INT/2 01: F_INT/4 10: F_INT/8 11: Undefined (F_INT is the frequency of the internal clock.)	Undefined
8	CS1CS_LVL	Defines the active level of the CS1 chip-select	0x0
7	CS1_EDGE_WR	Active edge of the serial clock SCLK used to write when CS1 is selected: 1: Falling 0: Rising (output data generated on this edge)	Undefined

Table 3–190. μWire Setup Register 1 (SR1) (Continued)

Bit	Name	Function	Reset
6	CS1_EDGE_RD	Active edge of the serial clock SCLK used to read when CS1 is selected: 1: Falling 0: Rising (input data strobed on this edge)	Undefined
5	CS0_CHK	Checks if external device is ready before activating a write process 1: If DI signal is low, the interface considers that the external component is busy. If DI is high, the interface considers that the first external component is ready and starts the write process. 0: No check is performed and the write process is immediately executed. Used when CS0 is selected	Undefined
4:3	CS0_FRQ	SCLK clock frequency when CS0 is selected: 00: F_INT/2 01: F_INT/4 10: F_INT/811 = Undefined (F_INT is the frequency of the internal clock.)	Undefined
2	CS0CS_LVL	Active level of the chip-select CS0	0x0
1	CS0_EDGE_WR	Active edge of the serial clock SCLK used to write when CS0 is selected: 1: Falling 0: Rising (output data generated on this edge)	Undefined
0	CS0_EDGE_RD	Active edge of the serial clock SCLK used to read when CS0 is selected: 1: Falling 0: Rising (input data strobed on this edge)	Undefined

The content of μWire setup register 1 must not be changed when a read or write process is running. This register sets up the serial interface for the first and second external components.

Table 3–191. μWire Setup Register 2 (SR2)

Bit	Name	Function	Reset
11	CS3_CHK	Checks whether external device is ready before activating a write process: 1: If DI signal is low, the interface considers that the external component is busy. If DI is high, the interface considers that the first external component is ready and starts the write process. 0: No check is performed and the write process is immediately executed. Used when CS3 is selected.	Undefined
10:9	CS3_FRQ	SCLK clock frequency when the CS3 is selected: 00: F_INT/2 01: F_INT/4 10: F_INT/8 11: Undefined (F_INT is the frequency of the internal clock.)	Undefined
8	CS3CS_LVL	Defines the active level of the CS3 chip-select	0x0
7	CS3_EDGE_WR	Active edge of the serial clock SCLK used to write when CS3 is selected: 1: Falling 0: Rising (output data generated on this edge)	Undefined
6	CS3_EDGE_RD	Active edge of the serial clock SCLK used to read when CS3 is selected: 1: Falling 0: Rising (input data strobed on this edge)	Undefined
5	CS2_CHK	Checks whether external device is ready before activating a write process: 1: If DI signal is low, the interface considers that the external component is busy. If DI is high, the interface considers that the first external component is ready and starts the write process. 0: No check is performed and the write process is immediately executed. Used when CS2 is selected.	Undefined
4:3	CS2_FRQ	SCLK clock frequency when the CS2 is selected: 00: F_INT/2 01: F_INT/4 10: F_INT/8 11: Undefined (F_INT is the frequency of the internal clock.)	Undefined
2	CS2CS_LVL	Defines the active level of the CS2 chip-select	0x0

Table 3–191. μWire Setup Register 2 (SR2) (Continued)

Bit	Name	Function	Reset
1	CS2_EDGE_WR	Active edge of the serial clock SCLK used to write when CS2 is selected 1: Falling 0: Rising (output data generated on this edge)	Undefined
0	CS2_EDGE_RD	Active edge of the serial clock SCLK used to read when CS2 is selected: 1: Falling 0: Rising (input data strobed on this edge)	Undefined

The content of μWire setup register 2 must not be changed when a read or write process is running. This register sets up the serial interface of the third and fourth external components.

Table 3–192. μWire Setup Register 3 (SR3)

Bit	Name	Function	Reset
2:1	CK_FREQ	Internal clock frequency (F_INT, CLK_EN = 1) All the internal logic is controlled by F_INT (F is the frequency of the external input clock). 00: F/2 01: F/4 10: F/7 11: F/10	0x0
0	CLK_EN	Clock enable: 0: Switches the clock off 1: Switches the clock on	0x0

The content of μWire setup register 3 must not be changed when a read or write process is running. This register sets up the serial interface for the internal clock.

3.19.12 MPU I/O Registers

Table 3–193 lists the MPUIO registers (*i* indicates GPIO or KBD external input values. Table 3–194 through Table 3–208 describe the register bits.

Table 3–193. MPUIO Registers (FFFE:4800)

Name	Description	Address	Access	Type	Reset Value
ARMIO_LATCH_IN	MPUIO latch input	FFFE:4800	16 bits	R	iiii iiii iiii iiii
ARMIO_LATCH_OUT	MPUIO latch output	FFFE:4802	16 bits	R/W	0000 0000 0000 0000

Table 3–193. MPUIO Registers (FFFE:4800) (Continued)

Name	Description	Address	Access	Type	Reset Value
IO_CNTL_REG	MPUIO input/output	FFFE:4804	16 bits	R/W	1111 1111 1111 1111
ARMIO_ CNTL_REG	MPUIO control	FFFE:4806	4 bits	R/W	1111 1111 1101 0011
ARMIO_LOAD_TIM	MPUIO load timer	FFFE:4808	16 bits	R/W	???? ???? ???? ????
KBR_LATCH_REG	MPUIO keyboard latch	FFFE:480A	5 bits	R	1111 1111 111i iiii
KBC_REG	MPUIO keyboard column	FFFE:480C	5 bits	R/W	???? ???? ???1 1111
BUZZER_ LIGHT_REG	MPUIO buzzer and light control	FFFE:480E	2 bits	R/W	???? ???? ???? ??00
LIGHT_ LEVEL_REG	MPUIO light power level	FFFE:4810	6 bits	R/W	???? ???? ??11 1111
BUZZER_ LEVEL_REG	MPUIO buzzer power level	FFFE:4812	6 bits	R/W	???? ???? ??11 1111
GPIO_EVENT_ MODE_REG	GPIO mode	FFFE:4814	6 bits	R/W	???? ???? ??00 0000
KBD_GPIO_INT	Keyboard/GPIO interrupt	FFFE:4816	2 bits		???? ???? ???? ??11
KBD_GPIO_ MASKIT	Keyboard/GPIO mask interrupt	FFFE:4818			???? ???? ???? ??00
GPIO_ DEBOUNCING_ REG	GPIO debouncing	FFFE:481A			???? ???? ???? 0000
GPIO_LATCH_REG	GPIO latch	FFFE:481C			???? ???? ???? ????

Table 3–194. MPUIO Input Register (ARMIO_LATCH_IN)

Bit	Name	Function	Reset
15:0	INPUT_LATCH	General-purpose inputs	Input pins

Table 3–195. MPUIO Output Register (ARMIO_LATCH_OUT)

Bit	Name	Function	Reset
15:0	INPUT_LATCH	General-purpose outputs	0x0000

Table 3–196. MPUIO Input/Output Control Register (IO_CNTL_REG)

Bit	Name	Function	Reset
15:0	IO_CNTL	Input/output control for general-purpose I/O 1: Input 0: Output	0xFFFF

Table 3–197. MPUIO Control Register (ARMIO_CNTL_REG)

Bit	Name	Function	Reset
15:6	Reserved		0x3FF
5	CLOCK_ENABLE	MPUIO module clock enable	0x0
4	Reserved		0x1
3	SOFT	Soft bit. Used in conjunction with the FREE bit to determine the state of the peripheral when a breakpoint is encountered. This bit is used in the emulation mode. 0: The peripheral halts immediately, either retaining or discarding the current state. 1: The peripheral stops after completion of the current task.	0x0
2	FREE	Free bit. Used in conjunction with the SOFT bit to determine the state of the peripheral when a breakpoint is encountered. This bit is used in the emulation mode. 0: The SOFT bit selects the emulation mode. 1: The peripheral clock runs free regardless of the SOFT bit.	0x0
1:0	Reserved		0x3

Table 3–198. MPUIO Load Timer Register (ARMIO_LOAD_TIM)

Bit	Name	Function	Reset
15:0	LOAD_TIM	This value is loaded when timer passes through 0 or when it starts.	Undefined

Table 3–199. MPUIO Keyboard Latch Register (KBR_LATCH_REG)

Bit	Name	Function	Reset
15:5	Reserved		–
4:0	KBR_LATCH	Keyboard row inputs	Input pins

Table 3–200. MPUIO Keyboard Column Register (KBC_REG)

Bit	Name	Function	Reset
15:5	Reserved		–
4:0	KBC_REG	Keyboard column outputs	0x1F

Table 3–201. MPUIO Buzzer and Light Control Register (BUZZER_LIGHT_REG)

Bit	Name	Function	Reset
15:2	Reserved		–
1	LIGHT	Light enable: 0: Off 1: On	0x0
0	BUZZER	Buzzer enable: 0: Off 1: On	0x0

Table 3–202. MPUIO Light Power Level Register (LIGHT_LEVEL_REG)

Bit	Name	Function	Reset
15:6	Reserved		–
5:0	LIGHT_LEVEL_REG	Value for light power level: 000000: Level 0 (no light) 000001: Level 1 111111: Level 63	0x3F

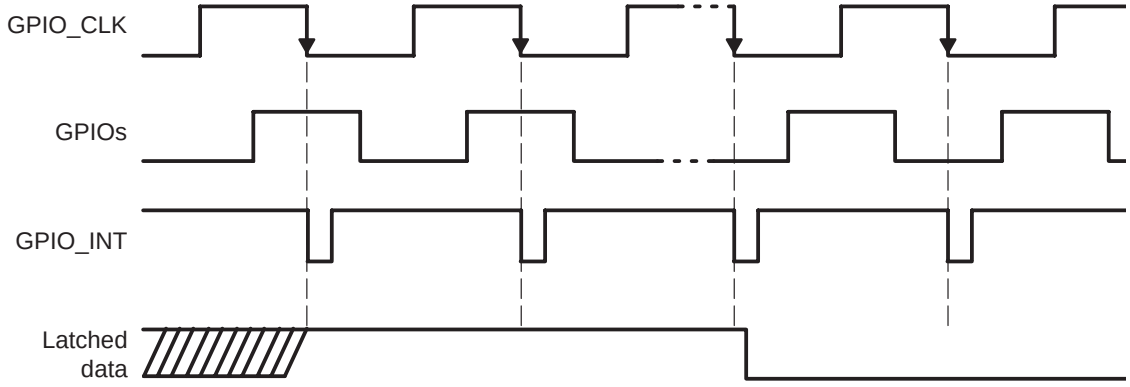
Table 3–203. MPUIO Buzzer Power Level Register (BUZZER_LEVEL_REG)

Bit	Name	Function	Reset
15:6	Reserved		–
5:0	BUZZER_ LEVEL_REG	Value for buzzer power level: 000000: Level 0 (no sound) 000001: Level 1 111111: Level 63	0x3F

3.19.12.1 GPIO Event Capture Module

The GPIO capture module allows latching of the input values present at the GPIO ports each time a rising of falling edge occurs on a selected GPIO port, here called GPIO_CLK. If not masked, the selected edge of the GPIO_CLK generates an interrupt to the processor.

Figure 3–2. GPIO Event Capture Timing



The GPIO_CLK can be generated from an external physical module. Consequently, it may become necessary to insert a debouncing processing on this signal. The debouncing time is programmable in GPIO_DEBOUNCING_REG register in 500-µs steps ((0 µs to 500 µs) → (7.5 ms to 8 ms)).

GPIO_EVENT_MODE_REG allows enabling or disabling of the GPIO event mode. It can also select the external pin to use as the GPIO_CLK. Finally, it can choose the GPIO_CLK falling or rising edge to generate a GPIO_INT interrupt. Table 3–204 describes the individual bits of the GPIO event mode register.

Table 3–204. GPIO Mode Register (GPIO_EVENT_MODE_REG)

Bit	Name	Function	Reset
15:6	Reserved	–	–
5	EDGE_SELECT	Set interrupt on falling/rising edge: 0: Falling edge 1: Rising edge	0x0
4:1	PIN_SELECT	Select ARMIO_IN[15:0] pin to generate IRQ: 0000: Pin 0 1111: Pin 15	0x0
0	SET_GPIO_EVENT_MODE	GPIO event mode: 0: Disabled 1: Enabled	0x0

Table 3–205. Keyboard/GPIO IRQ Register (KBD_GPIO_INT)

Bit	Name	Function	Reset
15:2	Reserved	–	–
1	GPIO_INT	GPIO interrupt (active low) Reset on read access to KBD_GPIO_INT	0x1
0	KBD_INT	Status bit keyboard interrupt (active low) Corresponding to the level of the interrupt signal	0x1

Table 3–206. Keyboard/GPIO Mask IRQ Register (KBD_GPIO_MASKIT)

Bit	Name	Function	Reset
15:2	Reserved	–	–
1	MASKIT_GPIO	Mask GPIO interrupt: 0: Unmasked 1: Masked	0x0
0	MASKIT_KBD	Mask keyboard interrupt: 0: Unmasked 1: Masked	0x0

Table 3–207. GPIO Debouncing Register (GPIO_DEBOUNCING_REG)

Bit	Name	Function	Reset
15:4	Reserved		–
3:0	GPIO_DEBOUNCING_REG	Debouncing time (step: 500 µs): 0000: 500 µs 1111: 8 ms	0x0

Table 3–208. GPIO Latch Register (GPIO_LATCH_REG)

Bit	Name	Function	Reset
15:0	GPIO_LATCH_REG	After debouncing time, ARMIO_IN bus is latched in this.	Undefined

3.20 SIM Registers

Table 3–209 lists the SIM registers, mapped in the I/O port space of the GSM-MPU. Table 3–210 through Table 3–218 describe the register bits.

Table 3–209. SIM Registers (FFFE:0000)

Name	Description	Address	Size	Type	Reset Value
REG_SIM_CMD	SIM control	FFFE:0000	5 bits	R/W	0 0000
REG_SIM_STAT	SIM status	FFFE:0002	4 bits	R	1010
REG_SIM_CONF1	SIM configuration	FFFE:0004	16 bits	R/W	0000 0000 0000 1100
REG_SIM_CONF2	SIM time-delay	FFFE:0006	16 bits	R/W	0000 1001 0100 0000
REG_SIM_IT	SIM interrupt status	FFFE:0008	5 bits	R	0 0000
REG_SIM_DRX	SIM receive byte	FFFE:000A	9 bits	R	0 ???? ????
REG_SIM_DTX	SIM transmit byte	FFFE:000C	8 bits	R/W	0000 0000
REG_SIM_ MASKIT	SIM interrupt mask	FFFE:000E	6 bits	R/W	11 1111
REG_SIM_IT_CD	SIM card-detect interrupt status	FFFE:0010	1 bit	R/W	0

Note:

Never perform an OR or AND operation while reading REG_SIM_CMD.

Table 3–210. SIM Control Register (REG_SIM_CMD)

Bit	Name	Function	Reset
15:5	Reserved		–
4	MODULE_CLK_EN	Module clock: 0: Disabled 1: Enabled	0x0
3	CMDSTART	SIM card start procedure: 0: No effect 1: Active	0x0
2	CMDSTOP	SIM card stop procedure: 1: Activation (toggle bit)	0x0

Note: These command bits are sensitive to the writing of a 1 (event generation). The writing of a 0 is inactive.

Table 3–210. SIM Control Register (REG_SIM_CMD) (Continued)

Bit	Name	Function	Reset
1	CMDIFRST	SIM interface software reset: 1: Activation (toggle bit)	0x0
0	CMDCARDRST	SIM card reset sequence: 0: Disabled 1: Enabled	0x0

Note: These command bits are sensitive to the writing of a 1 (event generation). The writing of a 0 is inactive.

Table 3–211. SIM Status Register (REG_SIM_STAT)

Bit	Name	Function	Reset
15:4	Reserved		–
3	STATFIFOEMPTY	FIFO content: 1: FIFO empty	0x1
2	STATFIFOFULL	FIFO content: 1: FIFO full	0x0
1	STATTXPAR	Parity check for transmit byte: 0: Parity error 1: Parity OK	0x1
0	STATNOCARD	Card presence: 0: No card 1: Card detected	0x0

Table 3–212. SIM Configuration Register 1 (REG_SIM_CONF1)

Bit	Name	Function	Reset
15	CONFSIOLOW	SIO: 0: No effect 1: Force low	0x0
14:11	CONFTRIG	FIFO trigger level: 0000: 1 ..... 1111: 16	0x0
10	CONFSRSTLEV	Logic level on SRST (used if CONFBYPASS = 1)	0x0
9	CONFSVCCLEV	Logic level on SVCC (used if CONFBYPASS = 1)	0x0
8	CONFBYPASS	Bypass hardware timers and start and stop sequences	0x0

Note: The CONFTXRX bit of the REG_SIM_CONF1 register defines the direction of the data transfer on the SIM I/O line at the system level (receive or transmit sequence of characters). Therefore, it does not define the direction of the line at the electrical level, which is monitored by the interface state machine.

Table 3–212. SIM Configuration Register 1 (REG_SIM_CONF1) (Continued)

Bit	Name	Function	Reset
7	CONFETUPER-IOD	ETU period 0: 372×1/F_{SCLK} 1: 512/8×1/F_{SCLK}	0x0
6	CONFSCLKLEV	SIM clock idle level: 0: Low 1: High	0x0
5	CONFSCLKDIV	SIM clock frequency: 0: 13/4 MHz 1: 13/8 MHz	0x0
4	Reserved	ETU period: 0: CONFETUPERIOD 1: 4×1/F_{SCLK}	0x0
3	CONFSCLKEN	SIM clock: 0: Standby mode 1: Normal	0x1
2	CONFTXRX	SIO line direction: 0: Receive mode 1: Transmit mode	0x1
1	CONFCODCONV	Coding convention (TS character): 0: Direct 1: Inverse	0x0
0	CONFCHKPAR	Enable parity check on reception: 0: Disabled 1: Enabled	0x0

Note: The CONFTXRX bit of the REG_SIM_CONF1 register defines the direction of the data transfer on the SIM I/O line at the system level (receive or transmit sequence of characters). Therefore, it does not define the direction of the line at the electrical level, which is monitored by the interface state machine.

Table 3–213. SIM Configuration Register 2 (REG_SIM_CONF2)

Bit	Name	Function	Reset
15:8	CONFWAITI	Overflow wait time between two received characters: Time unit = 960 × D × $T_{CKETU,}$ with D parameter = 1 or 8 (TA1 character)	0x09
7:4	CONFTDSIM	Time delay for contact activation/deactivation: Time unit = 8 × T_{CKETU}	0x4
3:0	CONFTFSIM	Time delay for filtering of SIM_CD: Time-unit = 1024 × T_{CK13M} (card extraction), or time-unit = 8192 × T_{CK13M} (card insertion)	0x0

Table 3–214. SIM Interrupt Status Register (REG_SIM_IT)

Bit	Name	Function (IRQ Flags)	Reset
15:5	Reserved	–	–
4	SIM_RX	0: On read access to REG_SIM_DRX 1: Waiting characters to be read	0x0
3	SIM_TX	0: On write access to REG_SIM_DTX or on switching from transmit to receive mode (CONFTXRX bit) 1: Waiting for character to transmit	0x0
2	SIM_OV	0: On read access to REG_SIM_IT 1: Receive overflow	0x0
1	SIM_WT	0: On read access to REG_SIM_IT 1: Character underflow	0x0
0	SIM_NATR	0: On read access to REG_SIM_IT 1: No answer to reset	0x0

Table 3–215. SIM Receive Byte Register (REG_SIM_DRX)

Bit	Name	Function	Reset
15:9	Reserved		–
8	STATRXPAR	Parity check for received byte: 0: Parity error 1: No parity error	0x0
7:0	SIM_DRX	Next data byte in FIFO available for reading	?

Table 3–216. SIM Transmit Byte Register (REG_SIM_DTX)

Bit	Name	Function	Reset
15:8	Reserved	–	–
7:0	SIM_DTX	Next data byte to be transmitted	0x00

Table 3–217. SIM Interrupt Mask Register (REG_SIM_MASKIT)

Bit	Name	Function	Reset
15:6	Reserved	–	–
5	MASK_SIM_CD	SIM card insertion/extraction interrupt: 0: Unmask 1: Mask	0x1
4	MASK_SIM_RX	Waiting characters to be read interrupt: 0: Unmask 1: Mask	0x1

Note: If mask is set to 1 the interrupt is disabled. However, the corresponding status bit remains sensitive.

Table 3–217. SIM Interrupt Mask Register (REG_SIM_MASKIT) (Continued)

Bit	Name	Function	Reset
3	MASK_SIM_TX	Waiting character to transmit interrupt: 0: Unmask 1: Mask	0x1
2	MASK_SIM_OV	Receive overflow interrupt: 0: Unmask 1: Mask	0x1
1	MASK_SIM_WT	Character wait-time overflow interrupt: 0: Unmask 1: Mask	0x1
0	MASK_SIM_NATR	No answer-to-reset interrupt: 0: Unmask 1: Mask	0x1

Note: If mask is set to 1 the interrupt is disabled. However, the corresponding status bit remains sensitive.

Table 3–218. SIM Card-Detect Interrupt Status Register (REG_SIM_IT_CD)

Bit	Name	Function (IRQ Flags)	Reset
15:1	Reserved	–	–
0	IT_CD	0: On read access to REG_SIM_IT_CD 1: SIM card insertion/extraction	0x0

3.21 Time Serial Port (TSP) Registers

3.21.1 Parallel Bit Interface

The parallel bit interface directly monitors 14 output signals TSPACTi with i = [0–13]. It can independently control the activation and deactivation of each output signal with a quarter of GSM bit time accuracy (923 ns).

The interface is based on two 8-bit registers, REG_TSP_ACT_L and REG_TSP_ACT_U, loaded by the TPU using a MOVE instruction. The two data registers of the parallel port are mapped in the addressable data-space corresponding to the address field of the MOVE instruction of the time processing unit (TPU). This field consists of five bits defining 32 potentially-addressable registers. Table 3–219 lists parallel port registers. Table 3–220 and Table 3–221 describe the register bits.

Table 3–219. Parallel Port Registers

Name	Description	TPU Address (MOVE Instruction)	Size	Value at Reset
REG_TSP_ACT_L	Lower TSP	0x06	8 bits	0000 0000
REG_TSP_ACT_U	Upper TSP	0x07	8 bits	0000 0000

Table 3–220. Lower TSP Register (REG_TSP_ACT_L)

Bit	Name	Description	Reset
7:0	REG_TSP_ACT_L	Activation of TSPACT[7:0] signals	0x00

Table 3–221. Upper TSP Register (REG_TSP_ACT_U)

Bit	Name	Description	Reset
7:5	Unused		0x0
4:0	REG_TSP_ACT_U	Activation of TSPACT[13:8] signals	0x00

This interface, based on REG_TSP_ACT_L and REG_TSP_ACT_U, is compatible with both the control signals of the transmit/receive sections of the AD7015 chip of Analog Devices and the timing interface of the Texas Instruments GSM codec.

3.21.2 TSP Receive and Transmit Registers

Both receive and transmit registers are mapped in the GSM-MPU address space. Transmit register access is limited to debug purposes, as these registers are controlled by the TPU only. All of these registers are accessible through the TIPB interface. Table 3–222 lists the TSP receive and transmit registers. Table 3–223 through Table 3–225 describe the register bits.

Table 3–222. TSP Receive and Transmit Registers (FFFE:0800)

Name	Description	GSM-MPU Address	Size	Type	Value at Reset
REG_RX_LSB	Receive lower bits	FFFE:0800 – (000)	16 bits	R/W	0x0000
REG_RX_MSB	Receive upper bits	FFFE:0802 – (001)	16 bits	R/W	0x0000
REG_TX_LSB	Transmit lower bits	FFFE:080C – (110)	16 bits	R	0x0000
REG_TX_MSB	Transmit upper bits	FFFE:080A – (101)	16 bits	R	0x0000

Note: REG_TX_LSB = REG_TX_3 && REG_TX_4; REG_TX_MSB = REG_TX_1 && REG_TX_2. If less than 16 bits are transferred, only the LSBs are significant.

The serial port offers input and output registers for loading data to serialize (TRANSMIT) or to read parallel data (RECEIVE), respectively. The characteristics of the serial port are related to configuration bits stored in two control registers. Updating the read-write control register sets off the operation of the serial port.

Table 3–223. TSP Receive LSB Registers (REG_RX_LSB)

Bit	Name	Address	Description	Reset
15:0	REG_RX_LSB	FFFE:0800	16 lower bits or received data	0x0000

Table 3–224. TSP Receive MSB Register (REG_RX_MSB)

Bit	Name	Address	Description	Reset
15:0	REG_RX_MSB	FFFE:0802	16 upper bits or received data	0x0000

Note: If less than 16 bits are transferred, only the REG_RX_LSB bits are significant.

The received data are accessible from the GSM-MPU through two 16-bit registers (REG_RX_MSB and REG_RX_LSB) but not from the TPU, which implements only a write access to the port (the TPU is unidirectional → write only).

Table 3–225. TSP Transmit Registers (REG_TX_1/2/3/4)

Bit	Name	Address	Description	Reset
7:0	REG_TX_1	04	1st byte to transmit	0x00
7:0	REG_TX_2	03	2nd byte to transmit	0x00
7:0	REG_TX_3	02	3rd byte to transmit	0x00
7:0	REG_TX_4	05	4th byte to transmit	0x00

For debug purposes, the transmit registers are mapped in the address space of the GSM-MPU. However, no synchronization is ensured in hardware between the TPU and the GSM-MPU.

Because the size of the registers in the external devices is variable and in the range 6 to 23, the serial port implements four byte registers (REG_TX_1,

REG_TX_2; REG_TX_3; REG_TX_4). Each of them is loaded through the use of the MOVE instruction of the TPU.

The data to transmit are loaded in the serial port directly by the TPU. A data transmission may require one, two, or three MOVE instruction(s) if the register size of the external device is less than or equal to 1, 2, 3, or 4 bytes, respectively. Table 3–225 describes the individual bits of the TSP transmit registers.

3.21.3 TPU Sequencer Internal Address Registers Mapping

The control and input data registers of the time serial port are mapped in the addressable data-space corresponding to the address field of the MOVE instruction of the time processing unit (TPU). This field is composed of 5 bits defining 32 potential addressable registers. Table 3–226 lists the TSP control and input registers. Table 3–227 through Table 3–233 describe the register bits.

Table 3–226. TSP Control and Input Data Registers

Name	Description	TPU Address Offset	Size	Type	Value at Reset
REG_TSP_CTRL1	TSP interface control 1	00	8 bits	W	???? ???? 0000 0000
REG_TSP_CTRL2	TSP interface control 2	01	2 bits	W	???? ???? ???? ??00
REG_TX_1	Transmit 1	04	8 bits	W	???? ???? 0000 0000
REG_TX_2	Transmit 2	03	8 bits	W	???? ???? 0000 0000
REG_TX_3	Transmit 3	02	8 bits	W	???? ???? 0000 0000
REG_TX_4	Transmit 4	05	8 bits	W	???? ???? 0000 0000
REG_TSP_SET1	TSP setup 1	09	8 bits	W	???? ???? ?000 ?000

Note: All TSP registers are frozen as long as TSP reset signal (see TPU register REG_TPU_CTRL) is active (level 1). TSP reset must be released to authorize access to registers.

Table 3–226. TSP Control and Input Data Registers (Continued)

Name	Description	TPU Address Offset	Size	Type	Value at Reset
REG_TSP_SET2	TSP setup 2	0A	8 bits	W	???? ???? ?000 ?000
REG_TSP_SET3	TSP setup 3	0B	3 bits	W	???? ???? ???? ?000
REG_GAUGING_ ENABLE	32-kHz oscillator gauging enable	11	1 bit	W	???? ???? ???? ???0

Note: All TSP registers are frozen as long as TSP reset signal (see TPU register REG_TPU_CTRL) is active (level 1). TSP reset must be released to authorize access to registers.

Table 3–227. TSP Interface Control Register 1 (REG_TSP_CTRL1)

Bit	Name	Description	Reset
7:5	AD	Index of the addressed device (0 to 5)	0x0
4:0	NB	Word size (1 to 32 bits): 00000: 1 bit to transmit or receive 11111: 32 bits to transmit or receive	0x00

REG_TSP_CTRL1 is dedicated to the configuration of the serial port. This register defines:

- ❏ Number of bits to transfer (in the range 1 to 32)
- ❏ External device address (between 0 and 5)

This register must be stable during a transfer and must be loaded before updating REG_TSP_CTRL2.

Table 3–228. TSP Interface Control Register 2 (REG_TSP_CTRL2)—0x01

Bit	Name	Description	Reset
1	WR	Write activation process	0x0
0	RD	Read activation process	0x0

Note: The RD and WR bits are automatically reset when the serial interface starts the corresponding action. No new requests can be executing until the end of the running one.

REG_TSP_CTRL2 is dedicated to the activation of the serial port and starts the operation of the interface as soon as one of its two bits is set. It defines:

- ❏ Write activation of the serial port (transmit only)
- ❏ Read activation of the serial port (simultaneous receive and transmit)

Table 3–229. TSP Transmit Registers (REG_TX_1...REG_TX_4)

Bit	Name	Address	Description	Reset
7:0	REG_TX_1	04	1st byte to transmit	0x00
7:0	REG_TX_2	03	2nd byte to transmit	0x00
7:0	REG_TX_3	02	3rd byte to transmit	0x00
7:0	REG_TX_4	05	4th byte to transmit	0x00

Because the size of the registers in the external devices is variable and in the range 6 to 23, the serial port implements four byte registers (REG_TX_1, REG_TX_2; REG_TX_3; REG_TX_4). Each of them is loaded through the use of the MOVE instruction of the TPU.

The data to transmit are loaded in the serial port directly by the TPU. A data transmission may require one, two, or three MOVE instruction(s) if the register size of the external device is less than or equal to 1, 2, 3, or 4 bytes, respectively. Table 3–225 describes the individual bits of the TSP transmit registers.

Table 3–230. TSP Setup Register 1 (REG_TSP_SET1)—0x09

Bit	Name	Description	Reset
7	Reserved	–	–
6	L	Format of enable signal of device 1: 0: Level trigger 1: Edge trigger	0x0
5	P	Format of enable signal of device 1: 0: Negative level 1: Positive level	0x0
4	C	Active edge clock of device 1: 0: Falling 1: Rising	0x0
3	Reserved		–
2	L	Format of enable signal of device 0: 0: Level trigger 1: Edge trigger	0x0
1	P	Format of enable signal of device 0: 0: Negative level 1: Positive level	0x0
0	C	Active edge clock of device 0: 0: Falling 1: Rising	0x0

The REG_TSP_SET1/2/3 registers are dedicated to the configuration of the signal waveforms of the serial port. They define for each device:

- ❏ The active edge clock (1 bit)
- ❏ The format of the enabled signals (2 bits)

Table 3–231. TSP Setup Register 2 (REG_TSP_SET2)

Bit	Name	Description	Reset
7	Reserved		–
6	L	Format of enable signal of device 3: 0: Level trigger 1: Edge trigger	0x0
5	P	Format of enable signal of device 3: 0: Negative level 1: Positive level	0x0
4	C	Active edge clock of device 3: 0: Falling 1: Rising	0x0
3	Reserved		–
2	L	Format of enable signal of device 2: 0: Level trigger 1: Edge trigger	0x0
1	P	Format of enable signal of device 2: 0: Negative level 1: Positive level	0x0
0	C	Active edge clock of device 2: 0: Falling 1: Rising	0x0

Table 3–232. TSP Setup Register (REG_TSP_SET3)

Bit	Name	Description	Reset
7:3	Reserved		–
2	L	Format of enable signal of device 4: 0: Level trigger 1: Edge trigger	0x0
1	P	Format of enable signal of device 4: 0: Negative level 1: Positive level	0x0
0	C	Active edge clock of device 4: 0: Falling 1: Rising	0x0

Table 3–233. TSP Gauging Enable Register (REG_GAUGING_ENABLE)

Bit	Name	Description	Reset
1	GAUGING_ENABLE	Activation of GAUGING_ENABLE signal	0x0

The GAUGING_ENABLE signal is necessary to gauge the 32-kHz oscillator that schedules the GSM time base during the deep-sleep mode. This bit is directly controllable through the REG_GAUGING loaded by the TPU with a MOVE instruction. It can control the activation or deactivation of the GAUGING_ENABLE signal.

3.22 TPU Registers

Both receive and transmit registers are mapped in the GSM-MPU address space. Table 3–234 lists the TPU registers. Table 3–235 through Table 3–241 describe the registers bits.

Table 3–234. TPU Registers (FFFF:1000)

Name	Description	Address (4:0)	Size	Type	Reset Value
REG_TPU_OFFSET	TPU offset	FFFF:100C (00110)	13 bits	R	0 0000 0000 0000
REG_TPU_SYNCHRO	TPU synchronization	FFFF:100E (00111)	13 bits	R	0 0000 0000 0000
REG_TPU_CTRL	TPU control and status	FFFF:1000 (00000)	12 bits	R/W	0000 10?0 ?001
REG_INT_CTRL	Interrupt control	FFFF:1002 (00001)	4 bits	W	0000
REG_INT_STAT	Interrupt status	FFFF:1004 (00010)	2 bits	R	11
REG_IT_DSP_PG	Interrupt occurrence	FFFF:1020 (10000)	1 bit	W	1

3.22.1 TPU RAM Memory Mapping

The two pages of the TPU communication buffer are addressable by the GSM-MPU through the TIPB interface.

Note:

The GSM-MPU sees only one page. The page selection management is handled by the TPU.

Memory	Address (10:0)
Page 0 or 1	1000000000 1111111111

Table 3–235. TPU Offset Register (REG_TPU_OFFSET)

Bit	Name	Description	Reset
15:13	Reserved	–	–
12:0	TPU_OFFSET	Value of OFFSET operand	0x0000

The interrupt control register allows the TPU to generate a DSP interrupt (IT_DSP_PG) by executing a move instruction to the TPU offset register.

Table 3–236. TPU Synchronization Register (REG_TPU_SYNCHRO)

Bit	Name	Description	Reset
15:13	Reserved		–
12:0	TPU_SYNCHRO	Value of SYNCHRO operand	0x0000

Note: The content of the register is not latched; therefore, its value can be corrupted if GSM-MPU read access is simultaneous with a TPU write operation**.**

Table 3–237. TPU Control and Status Register (REG_TPU_CTRL)

Bit	Name	Description	Reset	TPU Reset
11	FULL WRITE	Enables GSM-MPU to write anywhere in the RAM even if the TPU is executing a scenario. **CAUTION:** Handle with care. There is no protection against the GSM-MPU writing on the address read by the TPU.	0x0	
10	TPU_CK_ENABLE	TPU module clock: 0: Disabled 1: Enabled	0x0	
9	TPU_WAIT	WAIT or AT state of TPU: 1: Active	0x0	0x0
8	TPU_IDLE	TPU-scenario execution status: 0: None 1: Ongoing	0x0	0x0
7	TSP_RESET	Reset of TSP module: 0: No effect 1: Reset	0x1	
6	GSM-MPU_RAM_ACCESS	RAM read access: 0: Not allowed to GSM-MPU 1: Allowed to GSM-MPU	0x0	
5	Reserved	–	–	–
4	DSP_EN	IT_DSP generation on next it_frame: 0: Clear when IT_DSP occurs 1: Enabled (by GSM-MPU)	0x0	0x0
3	Reserved	–	–	–
2	TPU_EN	Execution new scenario loaded in the TPU buffer at page TPU_page: 0: None (by TPU). 1: enabled (by GSM-MPU*)*	0x0	0x0
1	TPU_PAGE	Page of TPU buffer: 0: Page0 1: Page1 (visible by TPU)	0x0	0x0
0	TPU_RESET	Reset TPU module: 0: No effect 1: Reset (except GSM timebase)	0x1	

The control and status register, REG_TPU_CTRL, allows the GSM-MPU to:

- ❏ Reset the TPU module
- ❏ Check the TPU activity
- ❏ Reset the TSP module
- ❏ Control the TPU communication buffer

Table 3–238. FULL_WRITE vs GSM-MPU_RAM_ACCESS Logic

GSM-MPU_RAM_ACCESS	FULL_WRITE	Mode
0	0	Write page access mode
0	1	Full write mode
1	X	Full read/write mode

Table 3–239. Interrupt Control Register (REG_INT_CTRL)

Bit	Name	Description	Reset	TPU Reset
3	ITD_F	Force frame interrupt for DSP: 0: No effect 1: Force	0x0	0x0
2	ITD_M	Frame interrupt for DSP: 0: Unmask 1: Mask	0x0	0x0
1	ITP_M	Page interrupt: 0: Unmask 1: Mask	0x0	0x0
0	ITF_M	Frame interrupt for GSM-MPU: 0: Unmask 1: Mask	0x0	0x0

The interrupt control register allows the GSM-MPU to:

- ❏ Mask the frame interrupt (IT_FRAME) generation
- ❏ Mask the page interrupt (IT_PAGE) generation
- ❏ Mask the DSP interrupt (IT_DSP) generation

Table 3–240. Interrupt Status Register (REG_INT_STAT)

Bit	Name	Description	Reset
1	ITP	Page interrupt occurrence 0: Execution of new scenarios	0x1
0	ITF	Frame interrupt occurrence 0: New frame occurrence	0x1

The interrupt status register informs the GSM-MPU of the origin of the interrupt:

- New frame occurrence (IT_FRAME)
- Execution of a new scenario (IT_PAGE)

Table 3–241. DSP Interrupt Occurrence Register (REG_IT_DSP_PG)

Bit	Name	Description	Active Value	Reset Value
0	IT_DSP_PG	DSP programmable interrupt occurrence	0x0	0x1

3.23 TPU Sequencer

3.23.1 Functional Description

The sequencer is a microprogrammable machine which executes an auto-scheduled program code. The instruction frequency adopted is: F_{inst} = 13/12 MHz. The corresponding time-period (923.1 ns) is the time accuracy for the execution of a command.

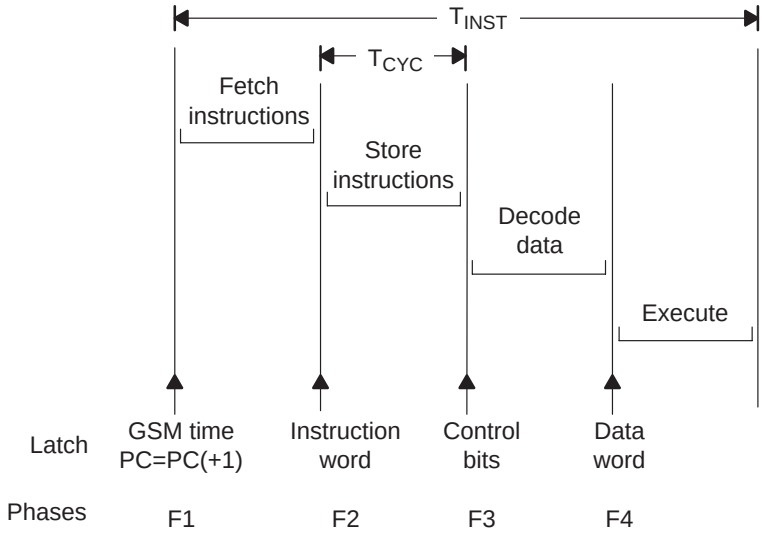
3.23.2 Instruction Execution Flow

The sequencer implements an instruction cycle based on four phases. The execution flow for each instruction is:

1) Instruction read (FETCH)
2) Instruction store (STORE)
3) Instruction decode (DECODE)
4) Data write or data compare (EXECUTE)

The network time (offset included) is latched at the beginning of each instruction cycle and remains stable for the whole cycle. The address pointer (PC) is updated at the beginning of the instruction cycle as a result of the execution phase of the previous instruction. The cycle frequency is: F_{cyc} = 13/3 MHz.

Figure 3–3. TPU Sequencer Instruction Flow



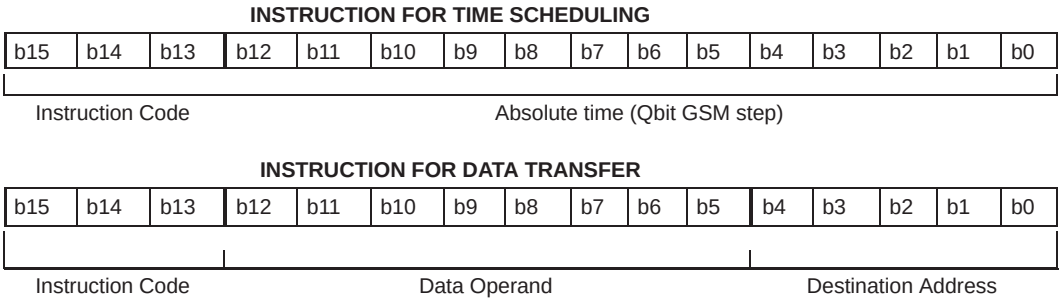
3.23.3 Microinstruction Set Definition

The set of microinstructions has been limited to the essential instructions in order to minimize the complexity of the decoder.

3.23.4 Structure of the Microinstruction

The microinstruction is coded on 16 bits to be consistent with the word format manipulated by the GSM-MPU and the DSP. It is split in several fields with a format specific to each category of instructions.

Figure 3–4. TPU Instruction Format



3.23.5 TPU Instruction Set

3.23.5.1 Microinstructions for Time Scheduling

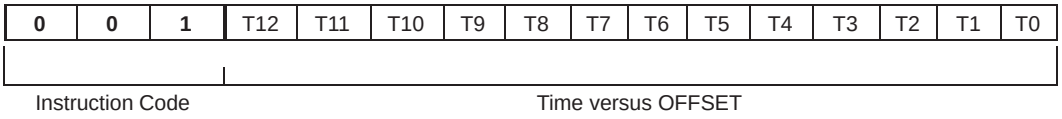
The instructions for time scheduling allow to:

- ❏ Start a process at a relative time in the frame: AT
- ❏ Load the offset value for the network time: OFFSET
- ❏ Load the offset value for the synchronization time: SYNCHRO
- ❏ Load the waiting time before execution of next instruction: WAIT
- ❏ Stop the sequencer: SLEEP

3.23.5.2 AT Instruction

The AT instruction allows starting a process at a specific time in the GSM TDMA frame. This time is relative to the network time. The time value is stored on 13 bits and expressed in quarter of GSM bit units. The dynamic range is of one frame (0 to 5000 qbit) with a quarter of bit time-step.

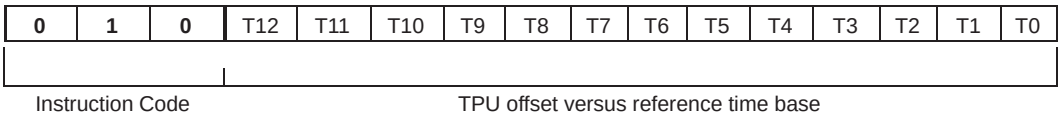
Figure 3–5. AT Instruction



3.23.5.3 OFFSET Instruction

The OFFSET instruction allows loading the offset value in the TPU offset register of the GSM time-base. The time value is stored on 13 bits and expressed in quarter of GSM bit units. The dynamic range is of one frame (0 to 5000 qbit) with a quarter of bit time-step.

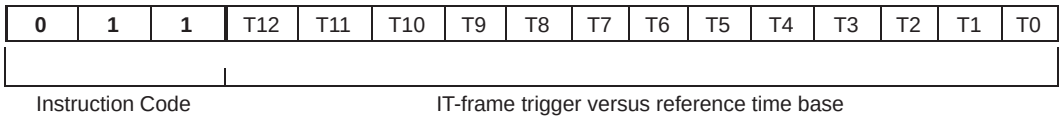
Figure 3–6. OFFSET Insruction



3.23.5.4 SYNCHRO Instruction

The SYNCHRO instruction allows loading the delta synchro-value in the TPU synchro-register and the offset register of the GSM time-base. Both registers are loaded simultaneously. The time value is stored on 13 bits and expressed n quarter of GSM bit units. The dynamic range is of one frame (0 to 5000 qbit) with a quarter of bit time-step.

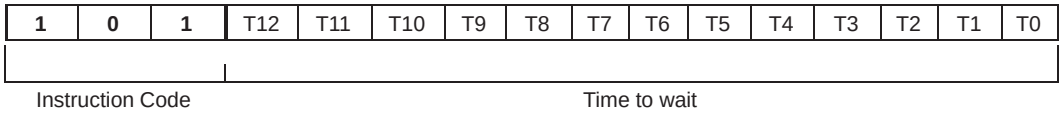
Figure 3–7. SYNCHRO Instruction



3.23.5.5 WAIT Instruction

The WAIT instruction allows introduction of a waiting-period between execution of two instructions. It can be used as a time relative AT. This time value is stored on 13 bits and expressed in quarter of GSM bit units. The total waiting time is equal to the programmed waiting period plus one qbit time interval that corresponds to the execution times of the instruction itself.

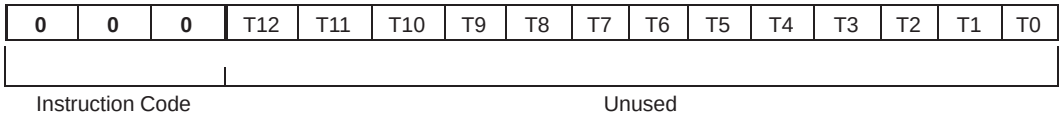
Figure 3–8. WAIT Instruction



3.23.5.6 SLEEP Instruction

The SLEEP instruction allows stopping the sequencer by disabling the TPU_ENABLE bit of the control register. To restart the TPU, the GSM-MPU must set the enable bit by writing in control register TPU_CTRL_REG.

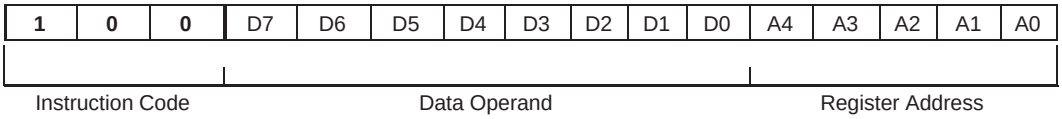
Figure 3–9. SLEEP Instruction



3.23.5.7 Microinstruction for Data Processing (MOVE)

The instruction for data processing (MOVE) allows to write a word of 8 bits maximum to a register of a peripheral. The address of the register is coded on 5 bits, thus defining 32 potentially-addressable registers.

Figure 3–10. Data Processing Instruction



The MOVE instruction can initiate two actions:

- A trigger event (toggle bit principle)
  - Bit DI = 1 → Event activated
  - Bit DI = 0 → No event
- Interrupt generation: A simple register loading with or without an associated trigger event; the writing of a word in a serial port and the start of the serialization of this word; a read operation with a bidirectional serial port.

3.24 ULPD Registers

Table 3–242 lists the the ULPD registers. Table 3–250 through Table 3–261 describe the register bits.

Table 3–242. ULPD Registers (FFFE:2000)

Name	Description	Address (FFFE:)	Size	Type	Reset Value
SETUP_RF_REG	RF setup	2024 (12)	8 bits	R/W	???? ???? 0000 0000
SETUP_FRAME_REG	Frame setup	2022 (11)	5 bits	R/W	???? ???? ???0 0000
SETUP_VTCXO_REG	VTCXO setup	2020 (10)	8 bits	R/W	???? 1111 1111 1111
SETUP_SLICER_REG	Slicer setup	201E (0F)	12 bits	R/W	???? 1111 1111 1111
SETUP_CLK13_REG	13-MHz clock setup	201C (0E)	6 bits	R/W	???? ???? ??11 1111
GSM_TIMER_IT_REG	GSM timer interrupt	201A (0D)	1 bit	R	???? ???? ???? ???0
GSM_TIMER_VALUE_REG	GSM timer value	2018 (0C)	16 bits	R	0000 0000 0000 0001
GSM_TIMER_INIT_REG	GSM timer initialization	2016 (0B)	16 bits	R/W	0000 0000 0000 0000
GSM_TIMER_CTRL_REG	GSM timer control	2014 (0A)	2 bits	R/W	???? ???? ???? ??10
GAUGING_STATUS_REG	Gauging status	2012 (09)	3 bits	R	???? ???? ???? ?000
GAUGING_CTRL_REG	Gauging control	2010 (08)	3 bits	R/W	???? ???? ???? ?000
COUNER_HI_FREQ_MSB_ REG	High-frequency MSB counter	200E (07)	6 bits	R	???? ???? ??00 0000
COUNTER_HI_FREQ_LSB_ REG	High-frequency LSB counter	200C (06)	16 bits	R	0000 0000 0000 0001
COUNTER_32_MSB_REG	32-kHz MSB counter	200A (05)	4 bits	R	???? ???? ???? 0000
COUNTER_32_LSB_REG	32-kHz LSB counter	2008 (04)	16 bits	R	0000 0000 0000 0001
SIXTEENTH_STOP_REG	Gauging stop timebase setup	2006 (03)	15 bits	R	?000 0000 0000 0000
SIXTEENTH_START_REG	Gauging start timebase setup	2004 (02)	15 bits	R	?000 0000 0000 0000

Table 3–242. ULPD Registers (FFFE:2000) (Continued)

Name	Description	Address (FFFE:)	Size	Type	Reset Value
INC_SIXTEENTH_REG	32-kHz period integer part setup	2002 (01)	12 bits	R/W	???? 0000 0000 0000
INC_FRAC_REG	32-kHz period fractional part setup	2000 (00)	16 bits	R/W	0000 0000 0000 0000

Table 3–243. RF Setup Register (SETUP_RF_REG)

Bit	Name	Function	Reset
7:0	SETUP_RF	Number of 32-kHz clock periods to enable the RF from the beginning of the wake-up phase	0x00

Note: The RF is enabled (SETUP_RF + 1) period at 32 kHz, after the GSM timer equals SETUP_FRAME.

Table 3–244. Frame Setup Register (SETUP_FRAME_REG)

Bit	Name	Function	Reset
4:0	SETUP_ FRAME	Number of frames to begin the wake-up phase (in TDMA frame unit)	0x00

Table 3–245. VTCXO Setup Register (SETUP_VTCXO_REG)

Bit	Name	Function	Reset
11:0	SETUP_ VTCXO	Number of 32-kHz clock periods to enable the VTCXO from the beginning of the wake-up phase	0xFFF

Note: The VTCXO is enabled (SETUP_VTCXO + 1) period at 32 kHz, after the GSM timer equals SETUP_FRAME.

Table 3–246. Slicer Setup Register (SETUP_SLICER_REG)

Bit	Name	Function	Reset
11:0	SETUP_ SLICER	Number of 32-kHz clock periods to enable the slicer from VTCXO ENABLE	0xFFF

Note: The slicer is enabled (SETUP_SLICER + 1) period at 32 kHz, after VTCXO goes active.

Table 3–247. 13-MHz Clock Setup Register (SETUP_CLK13_REG)

Bit	Name	Function	Reset
5:0	SETUP_CLK13	Number of 32-kHz clock periods to enable the 13-MHz clock from SLICER ENABLE	0x3F

Note: The 13-MHz clock is enabled after slicer goes active, plus delay. This delay is = (SETUP_CLK13 + 1) period of 32 kHz.

Table 3–248. GSM Timer Interrupt Register (GSM_TIMER_IT_REG)

Bit	Name	Function	Reset
0	IT_GSM_TIMER_ EVENT	GSM timer interrupt: 0: None 1: Pending	0x0

Note: The bit IT_GSM_TIMER_EVENT is cleared on reading this register.

Table 3–249. GSM Timer Value Register (GSM_TIMER_VALUE_REG)

Bit	Name	Function	Reset
15:0	TIMER_VALUE	Current value of the timer	0x0001

Table 3–250. GSM Timer Initialization Register (GSM_TIMER_INIT_REG)

Bit	Name	Function	Reset
15:0	TIMER_INIT	Load value of the timer	0x0000

Table 3–251. GSM Timer Control Register (GSM_TIMER_CTRL_REG)

Bit	Name	Function	Reset
1	FREEZE	GSM timer activity: 1: Frozen 0: Running	0x1
0	LOAD	Load the timer with TIMER_INIT value (toggle bit)	0x0

Note: The bit load is cleared after loading TIMER_INIT value. The bit freeze is set when the timer reaches zero.

Table 3–252. Gauging Status Register (GAUGING_STATUS_REG)

Bit	Name	Function	Reset
2	OVERFLOW_32	32-kHz counter-overflow error: 0: None 1: Overflow occurred	0x0
1	OVERFLOW_HI_FREQ	High-frequency counter-overflow error: 0: None 1: Overflow occurred (during gauging vs high-frequency clock)	0x0
0	IT_GAUGING	Gauging interrupt: 0: None 1: Pending	0x0

Note: The interrupt IT_GAUGING flag is cleared on reading of this register. OVERFLOW_32 and OVERFLOW_HI_FREQ are valid only after the interrupt IT_GAUGING occurrence. OVERFLOW_32 is cleared when a new gauging is started. OVERFLOW_HI_FREQ is cleared when a new gauging versus high-frequency clock is started.

Table 3–253. Gauging Control Register (GAUGING_CTRL_REG)

Bit	Name	Function	Reset
15:4	–	Reserved	–
3	TI internal test use	This bit must always be forced to 0.	0
2	SELECT_HI_ FREQ_CLOCK	High-frequency clock: 0: 13-MHz clock 1: DSP PLL clock	0x0

Note: The gauging is actually finished after the IT_GAUGING occurrence. After GAUGING_EN is cleared, if it is set before IT_GAUGING happens, the current gauging cannot be stopped. If GAUGING_EN is set and cleared in a short time (less than two periods at 32 kHz), the IT_GAUGING cannot be generated.

Table 3–253. Gauging Control Register (GAUGING_CTRL_REG) (Continued)

Bit	Name	Function	Reset
1	GAUGING_TYPE	Gauging versus: 0: GSM network time 1: High-frequency clock	0x0
0	GAUGING_EN	Gauging activity: 0: Stooped 1: Running	0x0

Note: The gauging is actually finished after the IT_GAUGING occurrence. After GAUGING_EN is cleared, if it is set before IT_GAUGING happens, the current gauging cannot be stopped. If GAUGING_EN is set and cleared in a short time (less than two periods at 32 kHz), the IT_GAUGING cannot be generated.

Table 3–254. High-Frequency MSB Counter Register (COUNTER_HI_FREQ_MSB_REG)

Bit	Name	Function	Reset
5:0	COUNTER_HI_ FREQ_MSB	Upper value of the number of high-frequency clocks during gauging time	0x00

Table 3–255. High-Frequency LSB Counter Register (COUNTER_HI_FREQ_LSB_REG)

Bit	Name	Function	Reset
15:0	COUNTER_HI_FREQ_LSB	Lower value of the number of high-frequency clocks during gauging time	0x0001

Table 3–256. 32-kHz MSB Counter Register (COUNTER_32_MSB_REG)

Bit	Name	Function	Reset
3:0	COUNTER_32_MSB	Upper value of the number of 32-kHz clocks during gauging time	0x0

Table 3–257. 32-kHz LSB Counter Register (COUNTER_32_LSB_REG)

Bit	Name	Function	Reset
15:0	COUNTER_32_LSB	Lower value of the number of 32-kHz clocks during gauging time	0x0001

Table 3–258. Gauging-Stop Time Base Setup Register (SIXTEENTH_STOP_REG)

Bit	Name	Function	Reset
14:0	SIXTEENTH_STOP	Value of GSM time base at the stop of gauging	0x0000

Table 3–259. Gauging-Start Time Base Setup Register (SIXTEENTH_START_REG)

Bit	Name	Function	Reset
14:0	SIXTEENTH_START	Value of GSM time base at the start of gauging	0x0000

Table 3–260. Integer Part of 32-kHz Period Setup Register (INC_SIXTEENTH_REG)

Bit	Name	Function	Reset
11:0	INC_SIXTEENTH	Integer part of the duration of the 32-kHz clock period in 1/16 of GSM bit units	0x000

Table 3–261. Fractional Part of 32-kHz Period Setup Register (INC_FRAC_REG)

Bit	Name	Function	Reset
15:0	INC_FRAC	Fractional part of the duration of the 32-kHz clock period in 1/16 of GSM bit units	0x0000

Note: The INC_FRAC value is the integer part of the fractional part of the duration of the 32-kHz clock period in 1/16 GSM bit units multiplied by 65536.

3.25 LPG Registers

Table 3–262 presents the LPG registers. Table 3–263 and Table 3–266 describe the individual register bits.

Table 3–262. LPG Registers (FFFE:7800)

Name	Description	Address	Access	Type	Reset Value
LCR_REG	LPG control	FFFE:7800	8 bits	R/W	0000 0000
PM_REG	Power management	FFFE:7801	1 bit	R/W	0

Table 3–263. LPG Control Register (LCR_REG)

Bit	Name	Function	Reset
7	PERM_ON†	Set high to force permanent light on	0x0
6	LPGRES†	LPG counter reset, active low	0x0
5:3	ONCTRL†	Time LED is on parameter	0x0
2:0	PERCTRL†	LED blink frequency	0x0

† Asynchronous writing and reading

The blinking period of the LED is determined by LCR bits 2:0, as shown in Table 3–264.

Table 3–264. LED Blinking Period

LCR Bit 2	LCR Bit 1	LCR Bit 0	LED Period (ms)	Clock Cycles
0	0	0	125	32
0	0	1	250	64
0	1	0	500	128
0	1	1	1000	256
1	0	0	1500	384
1	0	1	2000	512
1	1	0	2500	640
1	1	1	3000	768

The blinking period of the LED is determined by LCR bits 2:0, as shown in the LED blinking period table.

Table 3–265. LED On-Time

LCR Bit 5	LCR Bit 4	LCR Bit 3	Time LED On (ms)	Clock Cycles
0	0	0	3.889	4
0	0	1	7.789	8
0	1	0	15.59	12

Table 3–265. LED On-Time (Continued)

LCR Bit 5	LCR Bit 4	LCR Bit 3	Time LED On (ms)	Clock Cycles
0	1	1	31.39	16
1	0	0	46.59	20
1	0	1	62.59	24
1	1	0	78.39	28
1	1	1	93.59	32

The on-time of the LED is determined by LCR bits 5:3, as shown in the LED on-time table.

Table 3–266. LPG Power Management Register (PM_REG)

Bit	Name	Function	Reset
0	CLK_EN	Functional clock: 0: Disabled 1: Enabled	0x0

Note: Asynchronous writing and reading

3.25.1 Design Constraint

Clearing the LCR bit 6 (0) resets the whole PWM circuit (but not the control register) and switches off the LED. The LED can be switched independently from the PWM circuit by setting the LCR bit 7 (1 = permanent light). The reset PWRON is active low and resets the whole LPG (with the control register) and the output LPG_LED to zero asynchronously.

The LPG control register is written on a rising edge of nSTROBE. As nSTROBE is considered asynchronous, some metastability problems can happen. Consequently, the LED output could, in the worst case, be switched on at maximum intensity during one additional blink period.

3.26 UART 16C750 Registers

Each register is selected using its own combination of address lines A[4:0]. The programming combinations for register selection are shown in the following tables.

The registers listed in Table 3–267 are accessible to the GSM-MPU. The chip-select is defined by ARM_UART_CS.

Table 3–267. UART Modem Registers (FFFF:5000/6000)

Register	Address IRDA	Address GSM-MPU Modem	Address DSP Modem	Size	Type
UIR		FFFF:6000		2 bits	R/W
RHR	FFFF:5000	FFFF:5800	DATA:8000	8 bits	R
THR				8 bits	W
DLL				8 bits	R/W
IER	FFFF:5001	FFFF:5801	DATA:8001	8 bits	R/W
DLH				8 bits	R/W
IIR	FFFF:5002	FFFF:5802	DATA:8002	8 bits	R
FCR				8 bits	W
EFR				8 bits	R/W
LCR	FFFF:5003	FFFF:5803	DATA:8003	8 bits	R/W
MCR	FFFF:5004	FFFF:5804	DATA:8004	6 bits	R/W
XON1				8 bits	R/W
ADDR1				8 bits	R/W
LSR	FFFF:5005	FFFF:5805	DATA:8005	8 bits	R
XON2				8 bits	R/W
ADDR2				8 bits	R/W
MSR	FFFF:5006	FFFF:5806	DATA:8006	4 bits	R
TCR				8 bits	R/W
XOFF1				8 bits	R/W
SPR	FFFF:5007	FFFF:5807	DATA:8007	8 bits	R/W
TLR				8 bits	R/W
XOFF2				8 bits	R/W
MDR1	FFFF:5008	FFFF:5808	DATA:8008	6 bits	R/W
MDR2	FFFF:5009			2 bits	R/W
SFLSR	FFFF:500A			4 bits	R

Table 3–267. UART Modem Registers (FFFF:5000/6000) (Continued)

Register	Address IRDA	Address GSM-MPU Modem	Address DSP Modem	Size	Type
UIR		FFFF:6000		2 bits	R/W
TXFLL				8 bits	W
RESUME	FFFF:500B			8 bits	R
TXFLH				5 bits	W
SFREGL	FFFF:500C			8 bits	R
RXFLL				8 bits	R/W
SFREGH	FFFF:500D			4 bits	R
RXFLH				4 bits	R/W
BLR	FFFF:500E			8 bits	R/W
UASR		FFFF:580E	DATA:800E	8 bits	R
ACREG	FFFF:500F			5 bits	R/W
DIV1.6				8 bits	R/W
SCR	FFFF:5010	FFFF:5810	DATA:8010	6 bits	R/W
SSR	FFFF:5011	FFFF:5811	DATA:8011	2 bits	R
EBLR	FFFF:5012			8 bits	W

Table 3–268. UART Modem Registers Mapping

A[4:0]	Registers					
	LCR[7] = 0		LCR[7] = 1		LCR[7:0] = BF	
	READ	WRITE	READ	WRITE	READ	WRITE
0x00	RHR	THR	DLL	DLL	DLL	DLL
0x01	IER†	IER†	DLH	DLH	DLH	DLH
0x02	IIR	FCR†	IIR	FCR†	EFR	EFR
0x03	LCR	LCR	LCR	LCR	LCR	LCR
0x04	MCR†	MCR†	MCR†	MCR†	XON1	XON1
0x05	LSR	–	LSR	–	XON2	XON2
0x06	MSR/TCR‡	TCR‡	MSR/TCR‡	TCR‡	XOFF1/TCR‡	XOFF1/TCR‡
0x07	SPR/TLR‡	SPR/TLR‡	SPR/TLR‡	SPR/TLR‡	XOFF2/TLR‡	XOFF2/TLR‡
0x08	MDR1	MDR1	MDR1	MDR1	MDR1	MDR1

† MCR[7:5], FCR[5:4] and IER[7:4] can only be written when EFR[4] = 1.
‡ Transmission control register (TCR) and trigger level register (TLR) are accessible only when EFR[4] = 1 and MCR[6] = 1.

Table 3–268. UART Modem Registers Mapping (Continued)

A[4:0]	Registers					
	LCR[7] = 0		LCR[7] = 1		LCR[7:0] = BF	
	READ	WRITE	READ	WRITE	READ	WRITE
0x09	–	–	–	–	–	–
0x0A	–	–	–	–	–	–
0x0B	–	–	–	–	–	–
0x0C	–	–	–	–	–	–
0x0D	–	–	–	–	–	–
0x0E	–	–	UASR	–	UASR	–
0x0F	–	–	–	–	–	–
0x10	SCR	SCR	SCR	SCR	SCR	SCR
0x11	SSR	–	SSR	–	SSR	–

† MCR[7:5], FCR[5:4] and IER[7:4] can only be written when EFR[4] = 1.
‡ Transmission control register (TCR) and trigger level register (TLR) are accessible only when EFR[4] = 1 and MCR[6] = 1.

The registers are accessible to either the GSM-MPU or the DSP, depending on the value of UIR[0]. Either ARM_UART_CS or DSP_UART_CS defines the chip-select.

Table 3–269. UIR Register Mapping

A[4:0]	Registers	
	READ	WRITE
0000	UIR	UIR

The UIR register is permanently accessible only by the microcontroller and is mapped on a distinct chip-select (defined by the input ARM_SPECIAL_CS).

Table 3–270. Receiver Holding Register (RHR)

Bit	Name	Function	Reset
7:0	RHR	Receive holding	Undefined

The receiver section consists of the receiver holding register and the receiver shift register. The RHR is actually a 64-byte FIFO. The receiver shift register receives serial data from the RX input. The data is converted to parallel data and moved to the RHR. If the FIFO is disabled, location zero of the FIFO is used to store the single data character. If overflow occurs, data in the RHR is not overwritten.

Table 3–271. Transmit Holding Register (THR)

Bit	Name	Function	Reset
7:0	THR	Transmit holding	Undefined

The transmitter section consists of the transmit holding register and the transmit shift register. The transmit holding register is actually a 64-byte FIFO. The GSM-MPU writes data to the THR. The data is placed into transmit shift register where it is shifted out serially on the TX output. If the FIFO is disabled, location 0 of the FIFO is used to store the data.

Table 3–272. FIFO Control Register (FCR)

Bit	Name	Function	Reset
7:6	RX_FIFO_TRIG	RX FIFO trigger level (if TLR[7:4] = 0): 00: 8 characters 01: 16 characters 10: 56 characters 11: 60 characters	0
5:4	TX_FIFO_TRIG	TX FIFO trigger level (if TLR[3:0] = 0): 00: 8 spaces 01: 16 spaces 10: 32 spaces 11: 56 spaces	0
3	DMA_MODE	DMA mode (if SCR[0] = 0) 0: Mode 0 1: Mode 1	0
2	TX_FIFO_CLEAR	Transmit FIFO: 0: No change 1: Cleared and counter reset (returns to zero after clearing FIFO)	0
1	RX_FIFO_CLEAR	Receive FIFO 0: No change 1: Cleared and counter reset (returns to zero after clearing FIFO)	0
0	FIFO_EN	Transmit and receive FIFOs: 0: Disabled 1: Enabled	0

Note: Bits 4 and 5 can only be written when EFR[4] = 1.

Table 3–273. Supplementary Control Register (SCR)

Bit	Name	Function	Reset
5	DSR_IT	DSR interrupt 0: Disabled 1: Enabled	0x0
4	RX_CTS_WAKE_UP_ENABLE	Wake-up on RX or CTS: 0: Disabled, SSR[1] = 0 1: Waits for a falling edge of pins RX, CTS, or DSR to generate an interrupt	0x0

Table 3–273. Supplementary Control Register (SCR) (Continued)

Bit	Name	Function	Reset
3	TX_EMPTY_ CTL_IT	THR interrupt mode: 0: Normal mode 1: In UART mode, the THR interrupt is generated when TX FIFO and TX shifts are empty.	0x0
2:1	DMA_MODE_2	DMA mode (if SCR[0]= 1) 00: Mode 0 (no DMA) 01: Mode 1 TX: ARM_nDMA_REQ[0] RX: ARM_nDMA_REQ[1] 10: Mode 2 RX: ARM_nDMA_REQ[0] 11: Mode 3 TX: ARM_nDMA_REQ[0]	0x0
0	DMA_MODE_CTL	DMA mode set with: 0: FCR[3] 1: SCR[2:1]	0x0

Table 3–274. Line Control Register (LCR)

Bit	Name	Function	R/W	Reset
7	DIV_EN	Divisor latch: 0: Disabled 1: Enabled. Allows to access DLL, DLH, and others (refer to the mapping).	R/W	0x0
6	BREAK_EN	Break condition: 0: None 1: Break sending	R/W	0x0
5:4	PARITY_TYPE	Parity value (if LCR[3] = 1): 00: Odd 01: Even 10: Mark 11: Space	R/W	0x0
3	PARITY_EN	Parity bit (transmit and check): 0: None 1: Enable	R/W	0x0

Table 3–274. Line Control Register (LCR) (Continued)

Bit	Name	Function	R/W	Reset
2	NB_STOP	Number of stop bits: 0: 1 stop bit (word length = 5, 6, 7, 8) 1: 1.5 stop bits (word length = 5) 1: 2 stop bits (word length = 6, 7, 8)	R/W	0x0
1:0	CHAR_LENGTH	Word length (transmit and receive): 00: 5 bits 01: 6 bits 10: 7 bits 11: 8 bits	R/W	0x0

LCR[6:0] in the line control register define the transmission and reception parameters in UART mode.

3.26.1 Line Status Registers (LSR)

Table 3–275. UART Mode Line Status Register (LSR)

Bit	Name	Function	R/W	Reset
7	RX_FIFO_STS	Receive FIFO error: 0: Normal operation 1: At least one parity error, framing error or break indication in the receiver FIFO. Bit 7 is cleared when no more errors are present in the FIFO.	R	0x0
6	TX_SR_E	Transmit and hold status 0: Not empty 1: Empty	R	0x1
5	TX_FIFO_E	Transmit hold status: 0: Not empty 1: Empty	R	0x1
4	RX_BI	Break condition: 0: No break condition 1: Break detected Bit is set while the data being read from the RX FIFO is being received.	R	Undefined
3	RX_FE	Receiver framing error: 0: No framing 1: Framing error	R	Undefined
2	RX_PE	Receiver parity error: 0: No 1: Parity error	R	Undefined

Table 3–275. UART Mode Line Status Register (LSR) (Continued)

Bit	Name	Function	R/W	Reset
1	RX_OE	Receiver overrun error: 0: No error 1: Overrun error occurred When bit is set, character held in receive shift is not transferred to the RX FIFO. This case can occur only when receive FIFO is full.	R	0x0
0	RX_FIFO_E	RX FIFO contents status: 0: Empty 1: Not empty	R	0x0

The UART line status register shows that when the LSR is read, LSR[4:2] reflect the error bits (BI, FE, PE) of the character at the top of the RX FIFO (next character to be read). The LSR[4:2] registers do not physically exist as the data read from the RX FIFO is output directly onto the output data bus, DI[4:2], when the LSR is read. Therefore, reading the LSR and then reading the RHR identifies errors in a character. LSR[7] is set when there is an error anywhere in the RX FIFO and is cleared only when there are no more errors remaining in the FIFO.

Note:

Reading the LSR does not cause an increment of the RX FIFO read pointer. The RX FIFO read pointer is incremented by reading the RHR.

Table 3–276. SIR Mode Line Status Register (LSR) (UART/IrDA Module)

Bit	Name	Function	Reset
7	THR_EMPTY	Transmit hold: 0: Not empty 1: Empty When this bit is set, the processor can load up to 64 bytes of data into the THR if the TX FIFO is enabled.	0x1
6	STS_FIFO_FULL	Status FIFO: 0: Not full 1: Full	0x0
5	RX_LAST_BYTE	Last byte error: 0: Did not receive last byte of a frame from the FIFO 1: Received last byte from FIFO This bit is set when the last byte of a frame is read. Used to determine the frame boundary. Cleared on a read of the LSR.	0x0

Table 3–276. SIR Mode Line Status Register (LSR) (UART/IrDA Module) (Continued)

Bit	Name	Function	Reset
4	FRAME_TOO_LONG	Too long frame error: 0: No error 1: Error in the frame at the top of the STATUS FIFO (next character to be read) Bit is set when a frame exceeding the maximum length (set by RXFLH and RXFLLs) has been received. When this error is detected, current frame reception is terminated. Reception is stopped until the next START flag is detected.	?
3	ABORT	Receive frame abort: 0: None 1: Received	?
2	CRC	CRC frame error: 0: No error 1: CRC error in the frame at the top of the status FIFO (next character to be read)	?
1	STS_FIFO_E	Status FIFO: 0: Not empty 1: Empty	0x1
0	RX_FIFO_E	RX FIFO contents status: 0: Not Empty 1: Empty	0x1

The SIR mode line status register shows that when the LSR is read, LSR[4:2] reflect the error bits (FL, CRC, ABORT) of the frame at the top of the status FIFO (next frame status to be read). The LSR[4:2] registers do not physically exist as the data read from the status FIFO is output directly onto the output data bus, DI[4:2], when the LSR is read.

Table 3–277. Supplementary Status Register (SSR)

Bit	Name	Function	Reset
1	RX_CTS_DSR_WAKE_UP_STS	0: No falling edge event on RX, CTS, and DSR 1: A falling edge occurred on RX, CTS, or DSR.	0x0
0	TX_FIFO_FULL	TX FIFO: 0: Not full 1: Full	0x0

Table 3–278. Modem Control Register (MCR)

Bit	Name	Function	R/W	Reset
7	CLKSEL	Clock input divider: 0: None 1: Divide by 4	R/W	0x0
6	TCR_TLR	TCR, TLR access: 0: Disabled 1: Enabled	R/W	0x0
5	XON_EN	XON any: 0: Disables function 1: Enables function	R/W	0x0
4	LOOP-BACK_EN	Loopback mode: 0: Disabled (normal mode) 1: Enabled (internal) In this mode, the MCR[3:0] signals are looped back into MSR[7:4]. The transmit o/p is looped back to the receive input internally in both UART and SIR mode.	R/W	0x0
3	Reserved	Reserved	R/W	0x0
2	Reserved	Reserved	R/W	0x0
1	RTS	Force RTS output: 0: Forces $\overline{\text{RTS}}$ output to inactive (high). 1: Forces $\overline{\text{RTS}}$ output to active (low). In loopback, controls MSR[4]. If auto-$\overline{\text{RTS}}$ is enabled, the $\overline{\text{RTS}}$ output is controlled by hardware flow control.	R/W	0x0
0	Reserved	Reserved	R/W	0x0

† Bits 5, 6, and 7 can be written only when EFR[4] = 1.

MCR[3:0] controls the interface with the modem, data set, or peripheral device that is emulating the modem. The modem functions are only effective in UART mode. Table 3–278 describes the individual bits of the modem control register.

Table 3–279. Modem Status Register (MSR)

Bit	Name	Function	Reset
7:6	Reserved	Reserved	0x0
5	$\overline{\text{NDSR}}$_STS	This bit is the complement of the DSR input. In loopback mode, it is equivalent to MCR[0]	Input signal
4	$\overline{\text{NCTS}}$_STS	This bit is the complement of the $\overline{\text{CTS}}$ input. In loopback mode, it is equivalent to MCR[1]	Input signal
3:2	Reserved	Reserved	0x0

Table 3–279. Modem Status Register (MSR) (Continued)

Bit	Name	Function	Reset
1	$\overline{\text{DSR}}$_STS	DSR changed: 0: This bit is cleared on a read. 1: DSR input state has changed. (or MCR[0] in loopback mode)	0x0
0	$\overline{\text{CTS}}$_STS	CTS changed: 0: This bit is cleared on a read. 1: $\overline{\text{CTS}}$ input state has changed (or MCR[1]) in loopback mode).	0x0

The 8-bit modem status register provides information about the current state of the control lines from the modem, data set, or peripheral device to the GSM-MPU. It also indicates when a control input from the modem changes state.

3.26.2 Interrupt Enable Register (IER)

Table 3–280. UART Mode Interrupt Enable Register (IER)

Bit	Name	Function	Reset
7	$\overline{\text{CTS}}$_IT	$\overline{\text{CTS}}$ interrupt 0: Disabled 1: Enabled	0x0
6	$\overline{\text{RTS}}$_IT	$\overline{\text{RTS}}$ interrupt 0: Disabled 1: Enabled	0x0
5	XOFF_IT	XOFF interrupt 0: Disabled 1: Enabled	0x0
4	SLEEP_MODE	Sleep mode 0: Disabled 1: Enabled (baud rate clock stopped if module inactive)	0x0
3	MODEM_STS_IT	Modem status interrupt 0: Disabled 1: Enabled	0x0
2	LINE_STS_IT	Receiver line status interrupt 0: Disabled 1: Enabled	0x0

Note: Bits 4, 5, 6, and 7 can only be written when EFR[4] = 1 in UART mode.

Table 3–280. UART Mode Interrupt Enable Register (IER) (Continued)

Bit	Name	Function	Reset
1	THR_IT	THR interrupt 0: Disabled 1: Enabled	0x0
0	RHR_IT	RHR interrupt 0: Disabled 1: Enabled	0x0

Note: Bits 4, 5, 6, and 7 can only be written when EFR[4] = 1 in UART mode.

The UART mode interrupt enable register lists the seven types of interrupts in this mode: receiver error, RHR interrupt, THR interrupt, XOFF received, and $\overline{CTS}$/$\overline{RTS}$ change of state from low to high. They can be enabled/disabled individually. There is also a sleep mode enable bit.

Table 3–281. SIR Mode Interrupt Enable Register (IER) (UART IrDA Module)

Bit	Name	Function	Reset
7	EOF_IT	Received EOF interrupt: 0: Disabled 1: Enabled	0x0
6	LINE_STS_IT	Receiver line status interrupt: 0: Disabled 1: Enabled	0x0
5	TX_UNDERRUN_IT	TX underrun interrupt: 0: Disabled 1: Enabled	0x0
4	STS_FIFO_TRIG_IT	Status FIFO trigger level interrupt: 0: Disabled 1: Enabled	0x0
3	RX_OVERRUN_IT	RX overrun interrupt: 0: Disabled 1: Enabled	0x0
2	LAST_RX_BYTE_IT	Frame last-byte RX FIFO interrupt: 0: Disabled 1: Enabled	0x0

Table 3–281. SIR Mode Interrupt Enable Register (IER) (UART IrDA Module) (Continued)

Bit	Name	Function	Reset
1	THR_IT	THR interrupt: 0: Disabled 1: Enabled	0x0
0	RHR_IT	RHR interrupt: 0: Disabled 1: Enabled	0x0

The SIR mode interrupt enable register lists the eight types of interrupts in these modes: received EOF, LSR interrupt, TX underrun, status FIFO interrupt, RX overrun, last byte in RX FIFO, THR interrupt, and RHR interrupt. They can be enabled/disabled individually.

3.26.3 Interrupt Identification Register (IIR)

Table 3–282. UART Mode Interrupt Identification Register (IIR)

<table>
<tr><th>Bit</th><th>Name</th><th colspan="8">Function</th><th>Reset</th></tr>
<tr><td>7:6</td><td>FCR_MIRROR</td><td colspan="8">Mirror the contents of FCR (0).</td><td>0</td></tr>
<tr><td>5:1</td><td>IT_TYPE</td><td colspan="8"><table><tr><th rowspan="2">Priority</th><th colspan="6">Bits</th><th rowspan="2">Source</th></tr><tr><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>Receiver line status error</td></tr><tr><td>2</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>RX time-out</td></tr><tr><td>2</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>RHR interrupt</td></tr><tr><td>3</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>THR interrupt</td></tr><tr><td>4</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>Modem interrupt</td></tr><tr><td>5</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>XOFF/special character</td></tr><tr><td>6</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>CTS, RTS , DSR change state from active (low) to inactive (high)</td></tr></table></td><td>0</td></tr>
<tr><td>0</td><td>IT_PENDING</td><td colspan="8">Interrupt status:
0: IT pending
1: None</td><td>1</td></tr>
</table>

The UART mode interrupt identification register is a read-only 8-bit register that provides the source of the interrupt in a prioritized manner.

Table 3–283. SIR Mode Interrupt Identification Register (IIR)(UART IrDA Module)

Bit	Name	Function	Reset
7	EOF_IT	Received EOF interrupt 0: None 1: Pending	0x0
6	LINE_STS_IT	Receiver line status interrupt 0: None 1: Pending	0x0
5	TX_UE_IT	TX underrun interrupt 0: None 1: Pending	0x0
4	STS_FIFO_IT	Status FIFO trigger level interrupt 0: None 1: Pending	0x0
3	RX_OE_IT	RX overrun interrupt 0: None 1: Pending	0x0
2	RX_FIFO_LAST_BYTE_IT	Frame last-byte RX FIFO interrupt: 0: None 1: Pending	0x0
1	THR_IT	THR interrupt 0: None 1: Pending	0x0
0	RHR_IT	RHR interrupt 0: None 1: Pending	0x0

The nIRQ output is activated whenever one of the eight interrupts listed in the SIR mode interrupt identification register is active.

Table 3–284. Enhanced Feature Register (EFR)

Bit	Name	Function	Reset
7	AUTO_$\overline{CTS}$_EN	Auto-$\overline{CTS}$ flow control: 0: Disabled 1: Enabled Transmission is halted when the $\overline{CTS}$ pin is high (inactive).	0x0
6	AUTO_$\overline{RTS}$_EN	Auto-$\overline{RTS}$ flow control: 0: Disabled 1: Enabled $\overline{RTS}$ pin goes high (inactive) when the receiver FIFO halts trigger level, TCR(3:0), is reached, and goes low (active) when the receiver FIFO restores transmission trigger level.	0x0
5	SPECIAL_CHAR_DETECT	Special character detect: 0: Disabled 1: Enabled Received data is compared with XOFF2 data. If a match occurs, the received data is transferred to FIFO and IIR bit 4 is set to 1 to indicate detection of a special character.	0x0
4	ENHANCED_EN	Enhanced functions write enable: 0: Disables writing to IER bits 4:7 (in UART mode), FCR bits 4:5, and MCR bits 5:7. 1: Enables writing to IER bits 4:7 (in UART mode), FCR bits 4:5, and MCR bits 5:7.	0x0
3:2	TX_SWFLOW_CONTROL	Transmitter software flow control selection: 00: No flow control 01: Transmit $XON_2/XOFF_2$ 10: Transmit $XON_1/XOFF_1$ 11: Transmit $XON_1/XOFF_1$, $XON_2/XOFF_2$	0x0
1:0	RX_SWFLOW_CONTROL	Receiver software flow control selection: 00: No flow control 01: Compares $XON_2/XOFF_2$ 10: Compares $XON_1/XOFF_1$ 11: Compares $XON_1/XOFF_1$, $XON_2/XOFF_2$	0x0

Note: In SIR mode, EFR[1:0] can be used to enable/disable the automatic checking of the address of the incoming data frames. XON_1 and XON_2 must be set to different values if the software flow control is enabled.

EFR is an 8-bit register that enables or disables enhanced features. Most of the enhanced functions apply only to UART mode, but EFR[4] enables write accesses to FCR[5:4], the TX trigger level, which is also used in SIR mode.

Table 3–285. XON1/ADDR1 Register

Bit	Name	Function	Reset
7:0	XON_WORD1	Used to store the 8-bit XON1 character in UART mode and ADDR1 address 1 for SIR mode	Undefined

Table 3–286. XON2/ADDR2 Register

Bit	Name	Function	Reset
7:0	XON_WORD2	Used to store the 8-bit XON2 character in UART mode and ADDR2 address 2 for SIR mode	Undefined

Table 3–287. XOFF1 Register

Bit	Name	Function	Reset
7:0	XOFF_WORD1	Used to store the 8-bit XOFF1 character in used in UART mode	Undefined

Table 3–288. XOFF2 Register

Bit	Name	Function	Reset
7:0	XOFF_WORD2	Used to store the 8-bit XOFF2 character in used in UART mode.	Undefined

Table 3–289. Scratchpad Register (SPR)

Bit	Name	Function	Reset
7:0	SPR_WORD	Scratchpad	Undefined

SPR is an 8-bit read/write register that does not control the module in anyway. It is intended as a scratchpad register to be used by the programmer to hold temporary data.

3.26.3.1 Choosing the Appropriate Divisor Value

UART/SIR mode: Divisor value = Operating Frequency/(16 x Baud rate)

To achieve the required baud rate, DLL/DLH must be programmed with the integer part of the divisor value.

Table 3–290. Divisor Latch LSB Value Register (DLL)

Bit	Name	Function	Reset
7:0	CLOCK_LSB	Used to store the 8-bit LSB divisor value	Undefined

DLL stores the 8-bit LSB divisor for generation of the baud clock in the baud rate generator. DLL can only be written to before sleep mode is enabled (that is, before IER[4] is set).

Table 3–291. Divisor Latch MSB Value Register (DLH)

Bit	Name	Function	Reset
7:0	CLOCK_MSB	Used to store the 8-bit MSB divisor value	Undefined

DLH stores the 8-bit MSB divisor for generation of the baud clock in the baud rate generator. DLH can only be written to before sleep mode is enabled (that is, before IER[4] is set).

Table 3–292. Transmission Control Register (TCR)

Bit	Name	Function	Reset
7:4	RX_FIFO_TRIG_START	RCV FIFO trigger level to restore transmission (0–60)	0x0
3:0	RX_FIFO_TRIG_HALT	RCV FIFO trigger level to halt transmission (0–60)	0xF

TCR is an 8-bit register that is used to store the receive FIFO threshold levels to start/stop transmission during hardware/software flow control.

- TCR can be accessed only if EFR[4] = 1 and MCR[6] = 1.
- Trigger levels from 0–60 bytes are available with a granularity of four (trigger level = 4 x [4-bit register value])
- The programmer must ensure that TCR[3:0] > TCR[7:4] whenever auto-$\overline{\text{RTS}}$ or software flow control is enabled to avoid spurious operation of the device.

Table 3–293. Trigger Level Register (TLR)

Bit	Name	Function	Reset
7:4	RX_FIFO_TRIG_DMA	RCV FIFO trigger level (4–60)	0x0
3:0	TX_FIFO_TRIG_DMA	Transmit FIFO trigger level (4–60)	0x0

TLR is an 8-bit register that is used to store the programmable transmit and receive FIFO trigger levels used for DMA and IRQ generation. Trigger levels from 4 to 60 can be programmed with a granularity of 4, (that is, trigger level = 4 x [4-bit register value]).

- TLR can only be accessed if EFR[4] = 1 and MCR[6] = 1.
- If TLR[7:4] = 0000, the programmable RX trigger levels are disabled and the selectable trigger RX levels in FCR[7:6] are enabled.
- If TLR[3:0] = 0000, the programmable TX trigger levels are disabled and the selectable trigger TX levels in FCR[5:4] are enabled.

For the transmit FIFO, the TLR represents the number of empty spaces in the FIFO above which the THR interrupt is activated. For example, if TLR[3:0] = 1111, then if there are four or fewer bytes in the transmit FIFO, the THR interrupt is activated. If TLR[3:0] = 0001, then if there are 60 or less bytes in the transmit FIFO the interrupt is active.

Table 3–294. Mode Definition Register 1 (MDR1)

Bit	Name	Function	Reset
7	FRAME_END_MODE	Frame end mode: 0: Frame-length method 1: Set EOT bit method This bit is reserved in UART modem.	0x0
6	Reserved	Reserved	–
5	SCT	Start and control the SIR transmission: 0: As soon as a value is written to THR 1: Under control of ACREG[2] This bit is reserved in UART modem.	0x0
4	Reserved	Reserved	–
3	IR_SLEEP	SIR sleep mode: 0: Disabled 1: Enabled This bit is reserved in UART modem.	0x0
2:0	MODE_SELECT	Mode selection 000: UART 001: Slow infrared (reserved in UART modem) 010: UART with autobaud (reserved in IrDA) 111: Reset/default state All the other values are reserved	0x7

Writing to MDR1[2:0] can program the mode of operation. Therefore, the MDR1 must be programmed on start-up after configuring the configuration registers (DLL, DLH, LCR, etc.). The value of MDR1[2:0] must not be changed again during normal operation. To change the UART mode, MDR1[2:0] must be set to reset state and then to the new mode.

Table 3–295. UART Autobauding Status Register (UASR) (UART Modem Only)

Bit	Name	Function		Reset
7:6	PARITY_TYPE	Identified parity:		0x0
		00: No parity		
		01: Space		
		10: Even		
		11: Odd		
5	BIT_BY_CHAR	Identified character length:		0x0
		0: 7 bits		
		1: 8 bits		
4:0	SPEED	Identified speed:		0x00
		00000: Unknown	00110: 14 400 b/s	
		00001: 115 200 b/s	00111: 9 600 b/s	
		00010: 57 600 b/s	01000: 4 800 b/s	
		00011: 38 400 b/s	01001: 2 400 b/s	
		00100: 28 800 b/s	01010: 1 200 b/s	
		00101: 19 200 b/s	Others: Unknown	

The UART autobauding status register determines the speed, the number of bits by characters, and the type of parity. However, cases 5 and 6 bits are not considered. In autobauding mode, the input frequency of the UART modem must be fixed to 13 MHz.

To reset the autobauding hardware (to start a new AT detection) or to set the UART/modem in standard mode (no autobaud), MDR1[2:0] must be set to reset state 111 and then to the UART modem in autobaud mode 010 or UART/modem in standard mode 000.

Table 3–296. UART Interface Register (UIR) (UART Modem Only)

Bit	Name	Function	Reset
7:2	Reserved	Reserved	–
1	UART_MASK_IT	UART interrupt:	0x0
		0: Not masked	
		1: Masked	
0	UART_ACCESS	UART modems accesses by:	0x0
		0: GSM-MPU	
		1: DSP	

The UART interface register is accessed only by the microcontroller and is used to allow exclusive access of the IrDA/modem to the microcontroller or DSP. If access is given to the DSP, all the registers (except UIR) are accessible to the DSP (status, control register, and FIFO (RX, TX)). The interrupts are sent to the DSP and not to the microcontroller.

Steps required to perform a transition from the GSM-MPU TIPB to the DSP TIPB:

Step 1: GSM-MPU disables the DMA channel allocation to the UART modem in the DMA controller block.

Step 2: GSM-MPU disables DMA transfer since DMA bus is mapped on GSM-MPU TIPB only. This is done by setting the DMA in mode 0.

Step 3: GSM-MPU disables IT UART_MASK_IT to 1.

Step 4: GSM-MPU set UART_ACCESS to 1.

Step 5: GSM-MPU enables IT UART_MASK_IT to 0.

Steps required to perform a transition from the GSM-MPU TIPB to the DSP TIPB:

Step 1: GSM-MPU disables IT UART_MASK_IT to 1

Step 2: GSM-MPU sets UART_ACCESS to 0.

Step 3: GSM-MPU enables IT UART_MASK_IT to 0.

Step 4: The GSM-MPU can enable the DMA channel allocation to the UART/ modem in the DMA controller block.

Step 5: The GSM-MPU can then enable DMA transfers. This is done by setting the DMA in mode 1, 2, or 3.

There should be no activity on DSP_nSTROBE input (input stays high) between steps (1) and (3).

Table 3–297. Mode Definition Register 2 (MDR2) (ART IrDA Only)

Bit	Name	Function	Reset
7:5	Reserved	Reserved	–
4:3	DIV_1.6M	MSB part of DIV_1.6	0x0
2:1	STS_FIFO_TRIG	Status FIFO threshold select: 00: 1 character 01: 4 characters 10: 7 characters 11: 8 characters	0x0
0	Reserved	Reserved	–

The functions of MDR2 are only used in SIR mode. This register must be programmed before the mode is programmed in MDR1[2:0].

Table 3–298 and Table 3–299 describe the registers that hold the 13-bit transmit frame length. TXFLL holds the least-significant bits and TXFLH holds the most-significant bits. The frame length value is used if the frame length method of frame closing is used.

Table 3–298. Transmit Frame Length Low Register (TXFLL)(UART/IrDA Only)

Bit	Name	Function	R/W	Reset
7:0	TXFLL	LSB used to specify the frame length	W	0x00

Table 3–299. Transmit Frame Length High Register (TXFLH) (UART/IrDA Only)

Bit	Name	Function	R/W	Reset
7:5	Reserved	Reserved		–
4:0	TXFLH	MSB used to specify the frame length	W	0x00

Table 3–300 and Table 3–301 describe the registers that hold the 12-bit receive frame length. RXFLL holds the least-significant bits and RXFLH holds the most-significant bits. If the intended maximum receive frame length is n, then program RXFLL and RXFLH must be n + 3 in SIR mode.

Table 3–300. Receive Frame Length Low Register (RXFLL)(UART/IrDA Only)

Bit	Name	Function	R/W	Reset
7:0	RXFLL	LSB used to specify the frame length in reception	W	0x00

Table 3–301. Receive Frame Length High Register (RXFLH)(UART/IrDA Only)

Bit	Name	Function	R/W	Reset
7:4	–	Reserved		–
3:0	RXFLH	MSB used to specify the frame length in reception	W	0x0

Table 3–302. Status FIFO Line Status Register (SFLSR)(UART/IrDA Only)

Bit	Name	Function	R/W	Reset
7:5	Not used	Not used	–	–
4	OE_ERROR	Overrun error (in frame at top of RX FIFO): 0: No error 1: Overrun error	R	0x0
3	FRAME_LENGTH_ERROR	Frame-length error (in frame at top of FIFO): 0: No error 1: Frame error	R	0x0
2	ABORT_DETECT	Abort frame (in frame at top of FIFO): 0: None 1: Detected	R	0x0
1	CRC_ERROR	CRC error (in frame at top of FIFO): 0: No error 1: CRC Error	R	0x0
0	Not used	Not used	–	–

Reading SFLSR effectively reads frame status information from the status FIFO (that is, the register does not physically exist). Reading this register increments the status FIFO read pointer.

Table 3–303. Resume Register (RESUME)(UART/IrDA Only)

Bit	Name	Function	R/W	Reset
7:0	DI	Dummy read to restart TX or RX	R	0x00

The resume register is used to clear internal flags that halt transmission/reception when an underrun/overrun error occurs. Reading this register resumes the halted operation. This register does not physically exist and reading it results in all zeros being output on the data bus, DI[7:0].

Table 3–304 and Table 3–305 describe the status FIFO low and high registers. The frame lengths of received frames are written into the status FIFO. This information can be read by reading the SFREGL and SFREGH registers (that is, these registers do not physically exist). The least-significant bits are read from SFREGL and the most-significant bits are read from SFREGH. Reading these registers does not alter the status FIFO read pointer. These registers must be read before the pointer is incremented by reading the SFLSR register.

Table 3–304. Status FIFO Low Register (SFREGL)(UART/IrDA Only)

Bit	Name	Function	R/W	Reset
7:0	SFREGL	LSB part of the frame length	R	?

Table 3–305. Status FIFO High Register (SFREGH)(UART/IrDA Only)

Bit	Name	Function	R/W	Reset
3:0	SFREGH	MSB part of the frame length	R	?
7:4	Not used	Not used		

Table 3–306. BOF Length Register (BLR)(UART/IrDA Only)

Bit	Name	Function	R/W	Reset
7	STS_FIFO_RESET	Status FIFO reset. This bit is self-clearing.	R/W	0x0
6	BOF_TYPE	SIR start flag select 0: 0xFF 1: 0xC0	R/W	0x1
5:0	Not used	Not used	–	–

BLR[6] is used to select whether start patterns 0xC0 or 0xFF are used when multiple start flags are required in SIR mode. If only one start flag is required, this is always 0xC0. If (n) more than one start flag is required, then either (n–1) 0xC0 or (n–1) 0xFF flags are sent (if PLR (6) is 1 or 0, respectively), followed by a single 0xC0 flag (immediately preceding the first data byte).

NB_XBOF bits are used to specify how many xBOFs must be added at the beginning of a IrDA frame. The main purpose of the parameter is to provide a delay at the beginning of each frame for devices with long interrupt latency. The number of xBOFs to send depends on the baud rate and the time for the DTE to go from TX to RX state. According to the IrDA specification, the allowed number of additional values are: 0, 1, 2, 3, 4, 5, 6, 8, 10, 12, 16, 24, and 48.

Table 3–307. DIV1.6 Register(UART/IrDA Only)

Bit	Name	Function	Reset
7:0	DIV_1.6	Used to generate the 1.6-µs pulse	0x00

In SIR mode, the DIV1.6 register is used to generate 1.6-µs pulse encoding instead of 3/16 encoding when selected using ACREG[7].

The value of DIV1.6 is coded on 10 bits by MDR2[4:3] for its MSB and DIV–1.6[7:0] for its LSB.

In SIR mode, DIV1.6 must be programmed as follows:

DIV1.6 = (DLL,DLH) × 3 – 21 × (FCLK_frequency/13 MHz) + 1.

When the frequency of FCLK is 13 MHz, the formula becomes:

DIV1.6 = (DLL,DLH) × 3 – 20

- At 115200 b/s, DLH = 0x00, DLL = 0x07, MDR2[4:3] = 0x00, DIV1.6 = 0x01
- At 57600 b/s, DLH = 0x00, DLL = 0x0E, MDR2[4:3] = 0x00, DIV1.6 = 0x16
- At 38400 b/s, DLH = 0x00, DLL = 0x15, MDR2[4:3] = 0x00, DIV1.6 = 0x2B
- At 19200 b/s, DLH = 0x00, DLL = 0x2A, MDR2[4:3] = 0x00, DIV1.6 = 0x6A
- At 9600 b/s, DLH = 0x00, DLL = 0x55, MDR2[4:3] = 0x00, DIV1.6 = 0xEB
- At 2400 b/s, DLH = 0x01, DLL = 0x53, MDR2[4:3] = 0x03, DIV1.6 = 0xE5
- Etc.

Table 3–308. Auxiliary Control Register (ACREG)(UART/IrDA Only)

Bit	Name	Function	Reset
7	PULSE_TYPE	SIR pulse width select: 0: 3/16 of baud rate pulse width 1: 1.6 µs	0x0
6	SD_MOD	Primary o/p used to configure transceivers. Connected to the SD/MODE i/p of transceivers. 0: SD_MODE pin is set to high. 1: SD_MODE pin is set to low.	0x0
5	DIS_IR_RX	RXIR input: 0: Enabled 1: Disabled (for half-duplex purposes)	0x0
4	TX_UNDERRUN	TX underrun: 0: Enabled 1: Disabled (long stop bits can be transmitted)	0x0
3	Reserved	Reserved	–
2	SCTX_EN	Stores and controls TX start 0: Self-clearing bit 1: Start frame transmit (if MDR1[5] = 1)	0x0

Table 3–308. Auxiliary Control Register (ACREG)(UART/IrDA Only) (Continued)

Bit	Name	Function	Reset
1	ABORT_EN	Frame abort 0: No effect 1: Abort The GSM-MPU must reset the TX FIFO before the next frame is transmitted.	0x0
0	EOT_EN	End of transmission (EOT) 0: The GSM-MPU writes to the THR. 1: The GSM-MPU just writes the last byte to the TX FIFO.	0x0

Table 3–309. EBLR Register

Bit	Name	Function	R/W	Reset
7:0	EBLR	This allows to define up to 176 xBOFs, the maximum required by IrDA specification.	W	0x00

3.27 I²C Registers

The master I²C Interface module has 10 registers for communication between the TIPB and the I²C bus. The address of each register is equal to Start_ Address + Offset_Address where Start_Address is defined by the chip-select allocated to the I²C in the TIPB address space.

Table 3–310 lists the I²C registers. Table 3–311 through Table 3–320 describe the register bits.

Table 3–310. I²C Registers (FFFE:2800)

Name	Description	Address	Size	Type	Reset Value
DEVICE_REG	Device	FFFE:2800	7 bits	R/W	1000 0000
ADDRESS_REG	Address	FFFE:2801	8 bits	R/W	0000 0000
DATA_WR_REG	Data-write	FFFE:2802	8 bits	R/W	0000 0000
DATA_RD_REG	Data-read	FFFE:2803	8 bits	R	0000 0000
CMD_REG	Command	FFFE:2804	6 bits	R/W	1101 0001
CONF_FIFO_REG	FIFO configuration	FFFE:2805	4 bits	R/W	1111 1111
CONF_CLK_REG	Clock configuration	FFFE:2806	6 bits	R/W	1100 0000
CONF_CLK_FUNC_REF	Clock configuration functional reference	FFFE:2807	7 bits	R/W	1000 1010
STATUS_FIFO_REG	FIFO status	FFFE:2808	6 bits	R	1100 0010
STATUS_ACTIVITY_REG	Activity status	FFFE:2809	4 bits	R	1111 0000

3.27.1 I²C Features

The master I²C interface module provides an interface between TIPB and I²C bus. TIPB can control the external peripheral devices on the I²C bus via the master I²C interface module. The master I²C interface module is basically a parallel to serial and serial to parallel converter. The parallel data received from the TIPB must be converted into a suitable serial form for external peripheral devices on the I²C bus. Also, the serial data received from the I²C bus must be converted into a suitable parallel form for the TIPB. This master I²C interface module is compatible with TIPB specification and Philips master-only I²C specification.

- The master I²C interface module supports I²C master-only mode with:
  - 7-bit address device
  - 8-bit subaddress
  - Master write to slave receiver in single or multiple mode (data loop) 16-byte-deep transmit FIFO
  - Read combined cycle

- 3-bit programmable prescale internal clock divider and 7-bit programmable SCL clock divider to support wide clock-frequency range of module input clock signals. The I2C SCL clock frequencies are:
  - I2C standard mode:100 kHz
  - I2C fast mode: 400 kHz
- 3-bit programmable spike filter to provide I2C bus input signal noise filtering ability
- Asynchronous TIPB access with zero wait state insertion, except for the synchronization of TIPB read access to STATUS_ACTIVITY
- Error handling capability during I2C bus access

❑ The master I2C interface module does not support:

- TIPB abort
- TIPB suspend mode
- TIPB supervisor mode
- TIPB DMA access
- I2C bus 10-bit addressing
- I2C bus CBUS compatibility
- Multimaster I2C

Table 3–311. Device Register (DEVICE_REG)

Bit	Name	Function	Reset	
			HW	SW
7	Reserved	–	1	–
6:0	DEVICE	Identification code for I2C bus slave device	0x00	–

At the beginning of the I2C bus read/write access, the device register is loaded with information about 7-bit slave device identification.

Table 3–312. Address Register (ADDRESS_REG)

Bit	Name	Function	Reset	
			HW	SW
7:0	ADDRESS	I2C slave device internal address	0x00	–

Table 3–313. Data Write Register (DATA_WR_REG)

Bit	Name	Function	Reset	
			HW	SW
7:0	DATA_WRITE	Data to write on I2C bus	0x00	0x00

Table 3–314. Data Read Register (DATA_RD_REG)

Bit	Name	Function	Reset	
			HW	SW
7:0	DATA_READ	Data to read from I2C bus	0x00	0x00

Table 3–315. Command Register (CMD_REG)

Bit	Name	Function	Reset	
			HW	SW
7: 6	Reserved		0x3	–
5	IRQ_MSK	TIPB interrupt request 0: Disabled 1: Enabled	0x0	–
4	COMB_READ	Simple or combined read access 0: A master read immediately, if RW = 1 1: A combined read access, if RW = 1	0x1	–
3	RW	Read not write bit 0: I^2C bus write access 1: I^2C bus read access	0x0	–
2	START	Start the I^2C transmission (toggle bit)	0x0	–
1	EN_CLK	Clock enable 0: Clock is shut off 1: Clock is enabled	0x0	–
0	SOFT_RESET	Reset the FIFO 0: No reset soft 1: Reset soft	0x1	0x1

Note: TIPB START toggle bit is activated when writing a 1. This bit does not need to be reset to 0. Writing a 0 means no action.

Table 3–316. FIFO Configuration Register (CONF_FIFO_REG)

Bit	Name	Function	Reset	
			HW	SW
7:4	Reserved	–	0xF	–
3:0	FIFO_SIZE	FIFO Size (16 maximum) to generate the FIFO_FULL 000: A value of 1 is read in the FIFO n: A value of n+1 is value in the FIFO	0xF	–

Table 3–317. Clock Configuration Register (CONF_CLK_REG)

Bit	Name	Function	Reset	
			HW	SW
7:6	Reserved		0x3	–
5:3	SPK_F	Spike filter factor/check signal stability 000: No filtering 001: For two master clocks 010: For three master clocks 011: For four master clocks 100: For five master clocks 101: For six master clocks 110: For seven master clocks 111: For eight master clocks	0x0	–
2:0	PTV	Prescale clock divider factor: Divisor_1 000: 1 001: 2 010: 4 011: 8 100: 16	0x0	–

Table 3–318. Clock Configuration Functional Reference Register (CONF_CLK_FUNC_REF)

Bit	Name	Function	Reset	
			HW	SW
7	Reserved	–	0x1	–
6:0	CLK_REF	Functional clock reference: Divisor_2 0000001 = 1 ..... 0000010: 2 1111110: 126 0000011: 3 1111111: 127	0x0A	–

Note: The CLK_FUNC_REF is generated by:

$$\text{CLK\_FUNC\_REF} = \frac{\text{Master\_Clock\_Frequency}}{(\text{Divisor\_2} + 1)}$$

$$\text{Master\_Clock\_Frequency} = \frac{\text{External\_Clock\_Frequency}}{\text{Divisor\_1}}$$

$$\text{SCL\_OUT} = \frac{\text{CLK\_FUNC\_REF}}{3}$$

Table 3–319. FIFO Status Register (STATUS_FIFO_REG)

Bit	Name	Function	Reset	
			HW	SW
7:6	Reserved	–	0x3	–
5:2	READ_CPT	Indicates the read FIFO count value	0x0	0x0
1	FIFO_EMPTY	Indicates whether the FIFO is empty 0: FIFO not empty 1: FIFO empty	0x1	0x1
0	FIFO_FULL	Indicates whether the FIFO is full 0: FIFO not full 1: FIFO full	0x0	0x0

Table 3–320. Activity Status Register (STATUS_ACTIVITY_REG)

Bit	Name	Function	Reset	
			HW	SW
7:4	Reserved		0xF	–
3	INTERRUPT	Interrupt bit: 0: Transfer is not complete or module is in idle mode. 1: Transfer completed or aborted on nonacknowledged When the device is in interrupt mode, the interrupt bit is used to indicate that the I^2C module originated the interrupt request. For write access: New data is allowed. For read access: New data is received. For error: No ACK on device, address, or data	0x0	0x0
2	IDLE	Master I^2C mode 0: Idle 1: Transfer/receive When the device is in idle mode, the valid data is stored in read and a new access is allowed; else no new access is allowed.	0x0	0x0
1	ERROR_DEVICE	Device transmission error: 0: No error 1: Error	0x0	0x0
0	ERROR_DATA	Subaddress or data transmit flag error: 0: No error 1: Error	0x0	0x0

The internal clock must be running (EN_CLK of CMD_REG set to 1) to read the activity status register or to write in the setup register 2.

3.28 DSP Peripherals

3.28.1 Radio Interface (RIF)

The radio interface (RIF) module is a buffered serial port derived from the BSP peripheral module defined for the TMS320C54x DSP.

The external serial data transmission is supported by a full-duplex, double-buffered serial port interface. The TMS320C54x DSP exchanges data with the RIF through either its:

- ❏ XIO interface for configuration data and radio data in word by word protocol

or its

- ❏ MPUIF interface for radio data in DMA mode (buffered mode with data block transfer)

For each data transfer between TMS320C54x DSP and RIF, an interrupt is sent to the TMS320C54x DSP (XIO mode) or a DMA request and an END_DMA request is sent to GSM-MPU.

Transmit serial clock is either generated internally or externally. Receive serial clock is always generated externally. The RIF uses an internal 13-MHz clock to manage the autobuffering function.

3.28.2 Multichannel Serial Interface (MCSI)

The MCSI is a serial interface with a multichannel transmission capability. It expands the addressing capability of the parallel interface of a digital signal processor to connect external devices such as codec, DSP, and GSM system simulator.

All transmission parameters are configurable to cover the maximum number of operating conditions:

- ❏ Master or slave clock control (transmission clock and frame synchronization pulse)
- ❏ Programmable transmission clock frequency (from 6.3 kHz to 6.5 MHz)
- ❏ Single or multichannels (x16) frame structure
- ❏ Programmable word length: 3 to 16 bits
- ❏ Full-duplex transmission
- ❏ Programmable frame configuration
  - ■ Continuous or burst transmission
  - ■ Normal or alternate framing
  - ■ Normal or inverted frame polarity
  - ■ Short or long frame pulse
  - ■ Programmable oversize frame length
  - ■ Programmable frame length

- ❏ Programmable interrupt occurrence time (TX and RX)
- ❏ Error detection with interrupt generation on wrong frame length

Moreover, the interface supports the GSM DAI operating modes (radio uplink, radio downlink, and acoustic). In DAI mode, the MCSI interface is configured to connect directly to the GSM system simulator interface including the reset system simulator signal.

3.28.3 Ciphering Processor (CRYPT)

The ciphering processor implements both A51 and A52 algorithms as defined in GSM Rec 03.20 and detailed in GSM MoU documents.

3.28.4 Universal Asynchronous Receiver/Transmitter (16C750)

UART interface compatible with the NS 16C750 device. This UART is shared between the GSM-MPU and the TMS320C54x DSP processors. Only one processor at a time can control the UART. The allocation of the UART is defined by the GSM-MPU (configuration register). By default, the UART is connected to the GSM-MPU TIPB.

3.28.5 Direct Memory Access Controller (DMA)

The DMA module is shared between the DSP and GSM-MPU processors. The DMA have only two port DSP-MPUI memory and GSM-MPU-TIPB, so it allows TIPB peripherals to use the DMA channel between the TIPB peripherals and the MPUI memory.

3.28.6 Interrupt Handler (INTH)

The interrupt handler provides 21 interrupts to the DSP core. Each incoming interrupt can be configured as a low-level sensitive or falling-edge sensitive interrupt. The mask and the interruption level of the interrupts are configured in the DSP core itself.

3.28.7 DSP Program/Data Memory Extension

The C54x DSP program and data working space is extended through its XIO external interface.

The on-chip extended memory supports program and data read access up to the maximum DSP core operating cycle frequency. Write access require two-cycle access. All the extended memory is RAM-based to allow downloading of additional program code.

In order to ease the downloading of data into this memory space, this RAM is equally mapped in the GSM-MPU memory space with access validation controlled by a static configuration bit.

3.29 GSM-S Memory Mapping

GSM-MPU memory space is shared between the external memory interface and the TIPB. The memory interface provides six chip-select signals. All internal peripherals are mapped on GSM-MPU memory space with a range of 32K bytes.

The 8K bytes of internal RAM (0380:0000h to 0380:1000h) can overlay the first 8K-byte region 0000:0000h–0000:1000h of the GSM-MPU address space. In this case, the first 8K bytes of external memory are not accessible to the GSM-MPU. This overlay is controlled by GSM-MPU using a register of the GSM-MPU memory interface.

3.29.1 Memory Interface Mapping

Table 3–321. TIPB Memory Space

Device Name	nIBOOT	Start Address	Stop Address	Size (byte)	Data
nCS0: program†	1	0000:0000	007F:FFFF	8M	8/16/32
	0	0000:2000	007F:FFFF	8M – 8K	
nCS6	–	0080:0000	0084:FFFF	320K	8/16/32
nCS6 DSP-shared	–	0085:0000	0085:FFFF	64K	8/16/32
Not allocated	–	00C0:0000	00FF:FFFF	–	–
nCS1: data†	–	0100:0000	017F:FFFF	8M	8/16/32
nCS2 : random†	–	0180:0000	01FF:FFFF	8M	8/16/32
Not allocated	–	0200:0000	02BF:FFFF	–	–
nCS0 image	–	0300:0000	037F:FFFF	8M	8/16/32
nCS7	1	0380:0000	03FF:FFFF	8M	8/16/32
	0	0000:0000	0000:1FFF	8K	
Debug Unit (DU)	–	03C0:0000	03FF:FFFF	32	32
Not allocated	–	0400:0000	FFCF:FFFF	–	–
MPUI RAM	–	FFD0:0000	FFD0:3FFF	16K	16/32
MPUI control	–	FFE0:0000	FFE0:0001	2	16
Debug unit†	–				

† : External memory

3.29.2 External Flash/ROM Image

Regardless of the value of the nIBOOT signal, the external memory mapped on nCS0 is always accessible at an address defined in nCS0-image range.

3.29.3 TIPB Peripherals Mapping

Table 3–322. Memory Interface Mapping

Device	Description	Start Address	End Address	Size in Bytes	Data Access
		STROBE 0, FFFF:0000 ? FFFF:FFFF			
CS = 0	Reserved	FFFF:0000	FFFF:07FF	2K	
CS = 1	Reserved	FFFF:0800	FFFF:0FFF	2K	
CS = 2	TPU	FFFF:1000	FFFF:13FF	1K	16
	Reserved	FFFF:1400	FFFF:17FF	1K	
CS = 3 to 10	Reserved	FFFF:1800	FFFF:57FF	16K	
CS = 11	UART_MODEM	FFFF:5800	FFFF:5FFF	2K	8
CS = 12	Reserved	FFFF:6000	FFFF:67FF	2K	
CS = 13	Reserved	FFFF:6800	FFFF:6FFF	2K	
CS = 14 R	RIF	FFFF:7000	FFFF:77FF	2K	16
CS = 15 to 17	Reserved	FFFF:7800	FFFF:8FFF	6K	
CS = 18 TPU	TPU RAM	FFFF: 9000	FFFF:97FF	2K	16
CS = 19 DPLL	DPLL configura-tion	FFFF:9800	FFFF:9801	2	16
	Reserved	FFFF:9802	FFFF:9FFF	2046	
CS = 20 to 23	Not allocated	FFFF:9900	FFFF:BFFF	4K	
CS = 24	GEA	FFFF:C000	FFFF:C7FF	2K	8/16
CS = 25 to 30	Not allocated	FFFF:C800	FFFF:F7FF	12K	
CS = 31		FFFF:F800	FFFF:FFFF	2K	
	Watchdog timer	FFFF:F800	FFFF:F8FF	256	16
	TIPB bridge	FFFF:F900	FFFF:F9FF	256	16
	INTH	FFFF:FA00	FFFF:FAFF	256	16
	Memory Interface	FFFF:FB00	FFFF:FBFF	256	16
	DMA controller	FFFF:FC00	FFFF:FCFF	256	16
	CLKM	FFFF:FD00	FFFF:FDFF	256	16
	JTAG ID code	FFFF:FE00	FFFF:FE03	4	16
	MPU	FFFF:FF00	FFFF:FFFF	256	16

Table 3–322. Memory Interface Mapping (Continued)

Device	Description	Start Address	End Address	Size in Bytes	Data Access
		Strobe 1, FFFE:0000 ? FFFE:FFFF			
CS = 0	SIM	FFFE:0000	FFFE:07FF	2K	16
CS = 1	TSP	FFFE:0800	FFFE:0FFF	2K	16
CS = 2	Reserved	FFFE:1000	FFFE:17FF	2K	
CS = 3	RTC	FFFE:1800	FFFE:1FFF	2K	8
CS = 4	ULPD	FFFE:2000	FFFE:27FF	2K	16
CS = 5	I^2C	FFFE:2800	FFFE:2FFF	2K	8
CS = 6	SPI	FFFE:3000	FFFE:37FF	2K	16
CS = 7	TIMER1	FFFE:3800	FFFE:3FFF	2K	16
CS = 8	µWire	FFFE:4000	FFFE:47FF	2K	16
CS = 9	MPUIO	FFFE:4800	FFFE:4FFF	2K	16
CS = 10 to 12	Reserved	FFFE:5000	FFFE:67FF	6K	
CS = 13	Timer2	FFFE:6800	FFFE:6FFF	2K	16
CS = 14	Reserved	FFFE:7000	FFFE:77FF	2K	
CS = 15	LPG	FFFE:7800	FFFE:7FFF	2K	8
CS = 16	PWL	FFFE:8000	FFFE:87FF	2K	8
CS = 17	PWT	FFFE:8800	FFFE:8FFF	2K	8
CS = 18 to 20	Reserved	FFFE:9000	FFFE:A7FF	6K	
CS = 21	TCIF	FFFE:A800	FFFE:AFFF	2K	16
CS = 22	ICR	FFFE:B000	FFFE:B7FF	2K	16
CS = 23 to 29	Reserved	FFFE:B800	FFFE:EFFF	14K	
CS = 30	PERSEUS2_ CONF	FFFE:F000	FFFE:F7FF	2K	
CS = 31	Reserved	FFFE:F800	FFFE:FFFF	2K	

Table 3–323. TIPB Data Format

D32 ––––––––––> D24	D23 ––––––––––> D16	D15 ––––––––––>D8	D7 ––––––––––> D0
nCS0			
nCS1			
nCS2			
nCS3			

Table 3–323. TIPB Data Format (Continued)

D32 ––––––––––> D24	D23 ––––––––––> D16	D15 ––––––––––>D8	D7 ––––––––––> D0
CS4			
nCS6			
nCS7			
MPUI RAM			
Not mapped		MPU	
		GEA	
		MPUIC	
		SIM	
		TSP	
		TPU_REG	
		TPU_RAM	
		Not mapped	RTC
		ULPD	
		Not mapped	I2C
		SPI	
		TIMER1	
		Not mapped	LPG
		Not mapped	PWL
		Reserved	
		Not mapped	PWT
		µWire	
		MPUIO	
		Not mapped	Reserved
		Not mapped	UART_MODEM
		TIMER2	
		TIPB bridge	
		INTH	
		Memory interface	
		DMA controller	
		CLKM	
		JTAG ID code	
		Die ID code	

3.29.4 GSM-S DSP Memory Space

The DSP core is embedded with 28K words (16 bits) of RAM and 128K words (16 bits) of ROM.

3.29.4.1 Memory Type Definitions

DARAM: Dual-access data RAM. It is always mapped in data space and can be overlaid in program space using the OVLY bit.

MPUIRAM: Dual-access data RAM. It is always mapped in data space and can be overlaid in program space using the OVLY bit. The MPU host processor can also access this memory via the MPUI interface module. It behaves as a communication memory between the TMS320C54x DSP CPU and the MPU host processor.

PROM: Program ROM, always in program space

DROM: Data ROM, always in data space

PDROM: Program or data ROM. This ROM is always mapped in program space and can also be mapped in data space by setting the DROM control bit.

Shared PDRAM: Program/data RAM mapped on both the data space and the program space of the DSP XIO interface

The memory mapping for this S28C128 configuration is:

- 28K words of data memory (RAM based) mapped in both data spaces 0 and 1.
  - 2K words of dual access memory (DARAM)
  - 8K words of dual access memory (MPUI DARAM) shared between DSP and MPU/DMA
  - 18K words of dual access memory (DARAM)
- 128K words of program memory (ROM based)
  - 100K words of program memory (PROM) mapped in program space 0.
  - 20K words of data memory (DROM) mapped in data space 1.
  - 8K words of mixed program/data memory (PDROM) mapped in both program space 0 and data space 1.

Table 3–324. GSM-S TMS320C54x DSP Memory Space

	Data	Prog0	Prog1	Prog2	Prog3	Prog4	Prog5	Prog6
0000	DARAM overlay over the program area – 2K							
0800	MPUI overlay over the program area 8K							
1000								
1800								
2000								

Table 3–324. GSM-S TMS320C54x DSP Memory Space (Continued)

<table>
<tr><th></th><th>Data</th><th>Prog0</th><th>Prog1</th><th>Prog2</th><th>Prog3</th><th>Prog4</th><th>Prog5</th><th>Prog6</th></tr>
<tr><td>2800</td><td colspan="8" rowspan="8">DARAM overlay over the program area
18K</td></tr>
<tr><td>3000</td></tr>
<tr><td>3800</td></tr>
<tr><td>4000</td></tr>
<tr><td>4800</td></tr>
<tr><td>5000</td></tr>
<tr><td>5800</td></tr>
<tr><td>6000</td></tr>
<tr><td>6800</td><td colspan="2" rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>7000</td></tr>
<tr><td>7800</td><td rowspan="16">PDRAM
32K

DROM=0</td><td rowspan="2"></td><td rowspan="16">PROM
28K</td><td rowspan="16">PROM
32K</td><td rowspan="16">PROM
32K</td><td rowspan="2">PROM 8K</td><td rowspan="16">PDRAM
32K</td><td rowspan="16"></td><td rowspan="16"></td></tr>
<tr><td>8000</td></tr>
<tr><td>8800</td><td rowspan="7">DROM
20K

DROM=1</td><td rowspan="14"></td></tr>
<tr><td>9000</td></tr>
<tr><td>9800</td></tr>
<tr><td>A000</td></tr>
<tr><td>A800</td></tr>
<tr><td>B000</td></tr>
<tr><td>B800</td></tr>
<tr><td>C000</td></tr>
<tr><td>C800</td></tr>
<tr><td>D000</td></tr>
<tr><td>D800</td></tr>
<tr><td>E000</td><td colspan="2" rowspan="4">PDROM
8K</td></tr>
<tr><td>E800</td></tr>
<tr><td>F000</td></tr>
<tr><td>F800</td></tr>
</table>

Note: Grayed areas represent memory extension on XIO space

3.29.5 MPUIF

The MPUIF interface offers dual-access capability to 8K words of 16 bits of mixed data program memory. The MPUIF interface can be configured to manage data access of 8, 16, or 32 bits through the MPUIF control registers and the memory interface configuration registers.

The MPUIF dual-access capability is either enabled (SAM) or disabled (HOM) by the DSP. SAM is the default configuration when the DSP exits from a reset phase.

In shared-access mode (SAM), GSM-MPU (or DMA controller) and the TMS320C54x DSP can simultaneously access this shared memory space with GSM-MPU access resynchronized on the TMS320C54x DSP cycle clock (three times ratio required between GSM-MPU and TMS320C54x DSP cycle clocks).

In host-only mode (HOM), the MPUIF RAM is dedicated to external access under the control of either the GSM-MPU or the DMA controller; therefore, the access time is limited by the maximum access time of the used DARAM.

3.29.6 XIO Memory Mapping

All the data space is mapped on page 0 from address 0x8000 to 0xFFFF.

To avoid any overlay over the DROM memory space, the DROM bit is used to select either internal DROM (DROM = 1) or external PDRAM (DROM = 0).

The program space is mapped in the extended page 4, which has a size of 32K 16-bit words. This configuration prevents this DSP memory extension from overlaying the lower half-page of each extended page allocated to the existing DARAM and MPUIRAM.

According to the amount of memory targeted for program execution, the program space is mapped from 0x48000 to 0x4FFFF in page 4.

Sharing of this memory capacity between the DSP and the GSM-MPU is statically configurable through a dedicated register:

Case 1 → 0 bits for DSP/0.5M bits for GSM-MPU

Case 2 → 0.5M bits for DSP/0 bits for GSM-MPU

3.29.7 XIO–TIPB

Internal and external peripherals are mapped on XIO or data memory spaces. Theses spaces are accessible through nXSTROBE[3:0] with a range of 2K bytes for external peripherals allowing to connect up to:

- ❑ Six external devices on program space
- ❑ 26 external devices on data space
- ❑ 31 external devices on I/O space

Note:

32-bit internal peripherals are directly connected to the internal memory interface.

Table 3–325. TMS320C54x DSP XIO Memory Space

TMS320C54x DSP XIO-TIPB Mapping					
Device Name		**Start Address**	**Stop Address**	**Size in Bytes**	**Data Access**
External Peripherals Mapping—Program Space					
Strobe 0					
Not allocated	CS0	0000	07FF	2K	16
…	…	…	…	…	…
Not allocated	CS5	3000	37FF	2K	16
External Peripherals Mapping—Data Space 1					
Strobe 1					
Not allocated	CS6	3800	3FFF	2K	16
…	…	…	…	…	…
UART_MODEM	CS14	7000	77FF	2K	8
MCSI (Map1)	CS15	7800	7FFF	2K	16
External Peripherals Mapping—Data Space 2					
Strobe 2					
Not allocated	CS16	8000	87FF	2K	16
…	…	…	…	…	…
Not allocated	CS31	F800	FFFF	2K	16
External Peripherals Mapping—I/O Space					
Strobe 3					
RIF	CS0	0000	07FF	2K	16
MCSI	CS1	0800	0FFF	2K	16
Not allocated	CS2	1000	17FF	2K	
Not allocated	CS3	1800	1FFF	2K	
Not allocated	CS4	2000	27FF	2K	
A51/2	CS5	2800	2FFF	2K	16
Not allocated	CS6	3000	37FF	2K	
…	…	…	…	…	…
Not allocated	CS28	E000	E7FF	2K	
DMA controller	CS29	E800	EFFF	2K	16
Not allocated	CS30	F000	F7FF	2K	

Table 3–325. TMS320C54x DSP XIO Memory Space (Continued)

TMS320C54x DSP XIO-TIPB Mapping					
Device Name		**Start Address**	**Stop Address**	**Size in Bytes**	**Data Access**
External Peripherals Mapping—I/O Space (Continued)					
Strobe 3 (Continued)					
XI0–2-TIPB bridge	CS31	F800	F8FF	256	16
MPUI control		F900	F9FF	256	16
INTH		FA00	FAFF	256	16
NMI_ST_REG		FB00	FBFF	256	16
Not allocated		FC00	FCFF	256	
Not allocated		FD00	FDFF	256	
Not allocated		FE00	FEFF	256	
Not allocated		FF00	FFFF	256	

3.30 GSM-S Interrupt Mapping

The DSP subchip has 17 interrupt lines,11 of which are dedicated to external peripherals (INT0n to INT10n).

Table 3–326. DSP Interrupts Mapping

Name	Sense	DSP INT	Function
RSN	Level	RSN	DSP subsystem reset (HW or SW)
nMIN		nMIN	Abort on TIPB
	Level	INT0n	RIF receive interrupt
	Level	INT1n	RIF transmit interrupt
		INT2n	UART interrupt ❏ Error on receiver line ❏ Receive timeout ❏ Received character ❏ Character to transmit ❏ Modem status change ❏ Received XOFF/special character detected ❏ CTS/RTS deactivation
TINT		TINT	Timer interrupt
RINT		RINT	SPI receive interrupt
XINT		XINT	SPI transmit interrupt
	Level	INT3n	MCSI receive interrupt
	Level	INT4n	MCSI transmit interrupt
	Level	INT5n	MCSI frame duration error interrupt
	Level	INT6n	MCSI DAI interrupt
	Edge	INT7n	CYPHER interrupt ❏ End of ciphering process ❏ Error of processing
	Edge	INT8n	TPU frame interrupt
AINT		AINT	MPUI interrupts
	Edge	INT9n	TPU programmable interrupt
	Level	INT10n	DMA interrupt
	Edge	INT11n	External DSP interrupt

Note: The TPU programmable interrupt (INT9n) is a facility offered to the DSP programmer to allow the generation of a DSP interrupt at a dedicated time with a quarter of GSM bit accuracy. The interrupt is set in a scenario by using a time-stamped instruction.

3.30.1 MPU Interrupts

The GSM-MPU owns two interrupt lines: nIRQ and nFIQ. The TWL3016 and TWL3014 fast interrupt are mapped on nFIQ. All other peripheral interrupts are mapped on nIRQ as follows:

Table 3–327. GSM-S MPU Interrupt Mapping

Name	Sense	IRQ	FIQ	Function
IRQ0	Edge	X		Watchdog timer interrupts
IRQ1	Edge	X		Timer1 interrupt
IRQ2	Edge	X		Timer2 interrupt
IRQ3	Edge		X	TSP receive interrupt
IRQ4	Edge	X		TPU frame interrupt
IRQ5	Edge	X		TPU page interrupt
IRQ6	Edge	X		SIM interrupt ❑ No answer to reset ❑ Character underflow ❑ Character overflow ❑ Character to transmit ❑ Received character ❑ SIM card insertion/extraction
IRQ7	Level	X		UART_MODEM interrupts ❑ Error on receiver line ❑ Receive timeout ❑ Received character ❑ Character to transmit ❑ Modem status change ❑ Received XOFF/special character detected ❑ CTS/RTS deactivation ❑ DSR/RxD activity detection (off mode only)
IRQ8	Level	X		Keyboard or JogDial interrupt
IRQ9	Edge	X		RTC periodical timer interrupt
IRQ10	Level	X		RTC alarm or I^2C data transfer error/completion
IRQ11	Edge	X		ULPD end of gauging interrupt
IRQ12	Level	X		External interrupt (GSM_EXT_NIRQ pin) (See Note.)
IRQ13	Edge	X		SPI interrupt ❑ Received data ❑ Data to transmit
IRQ14	Level	X		DMA interrupt
IRQ15	Edge	X		MPUI interrupts (nHINT)

Note: The GSM_EXT_NIRQ signal is sent to both the GSM-S interrupt handler (IRQ12) and to the MPU-S GPIN(5).

Table 3–327. GSM-S MPU Interrupt Mapping (Continued)

Name	Sense	IRQ	FIQ	Function
IRQ16	Edge		X	SIM card-detect fast interrupt
IRQ17	Edge		X	Fast external power fail interrupt (NFIQ_PWR_FAIL pin)
IRQ18				Reserved
IRQ19	Level	X		ULPD GSM timer
IRQ20	Edge	X		GEA interrupt
IRQ21–22				Reserved
IRQ23	Level or Edge	X		GSM edge external MPU
IRQ24	Level		X	GSM protect
IRQ25				Reserved
IRQ26	Edge			ICR interrupt
IRQ27	Edge			TCIF GSM-MPU memory access error

Note: The GSM_EXT_NIRQ signal is sent to both the GSM-S interrupt handler (IRQ12) and to the MPU-S GPIN(5).

3.31 GSM-S DMA Mapping

3.31.1 GSM-S DMA Requests

The DMA controller manages access to the DSP MPUI 6K-word shared memory:

- The GSM-MPU GSM-MPU
- The radio interface (RIF)
- The modem UART
- Reserved

Each RIF RX and RIF TX has a dedicated channel. The following combinations are possible for UART:

- UART modem has two channels:
  - Ch#2: TX
  - Ch#3: RX
- UART modem has one channel:
  - Ch#2: RX or TX
- UARTs have no DMA.

All modules have DMA functions disabled after reset.

Table 3–328. DMA Channel Selection

DMA Request	Channel			
	0	**1**	**2**	**3**
RIF_DMA_REQ_X	?			
RIF_DMA_REQ_R		?		
nDMA_REQ_ARM(0) MODEM			?	
nDMA_REQ_ARM(1) MODEM				?

Note: Only one UART at a time should be allocated to a DMA channel (2 or 3). The potential conflicts between concurrent DMA requests must be solved at system level with only one peripheral configured in DMA mode.

3.32 GSM Memory Protection

This section discusses the GSM memory protection.

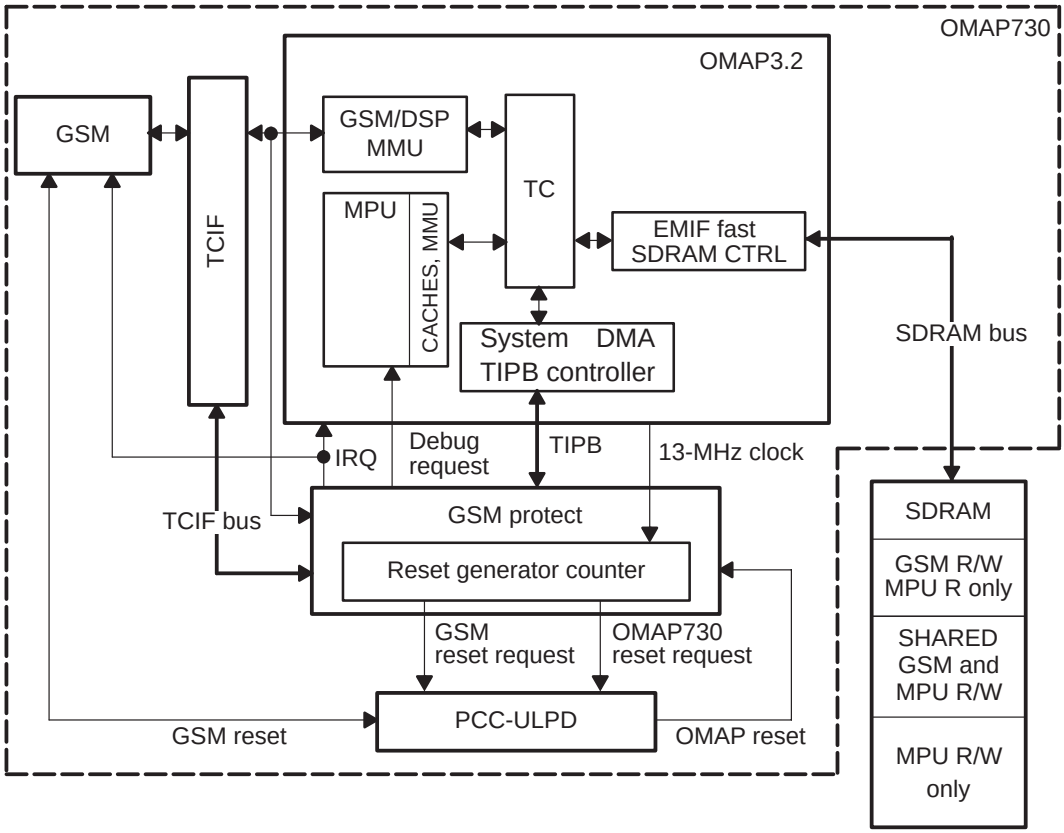
3.32.1 System Architecture Overview

In the OMAP730 device, the GSM-S has access to all MPU memories via the TCIF and GSM-DSP MMU components.

The GSM protect function prevents the MPU, DMA, and other components from writing into the part of the SDRAM that has been reserved for the GSM subsystem.

Figure 3–11 presents the system architecture overview for GSM memory protection.

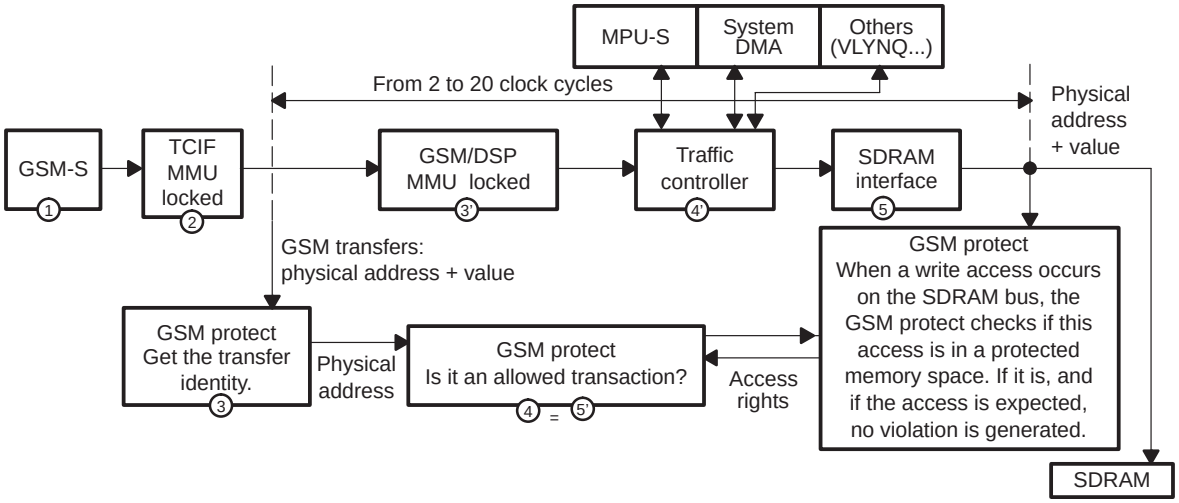
Figure 3–11. System Architecture Overview



3.32.2 Functional Overview

The protection is only intended to prevent the MPU, DMA, and other components from writing to specific areas in the SDRAM, which is a reserved memory space for GSM. Only the GSM can write to this area, but the MPU, DMA, and other components directly inserted in the OMAP traffic controller can read it. Whereas the TCIF MMU and GSM/DSP MMU mapping are designed to be locked after the boot process, the GSM cannot access the MPU-only memory space.

Figure 3–12. GSM Write to its Protected Area



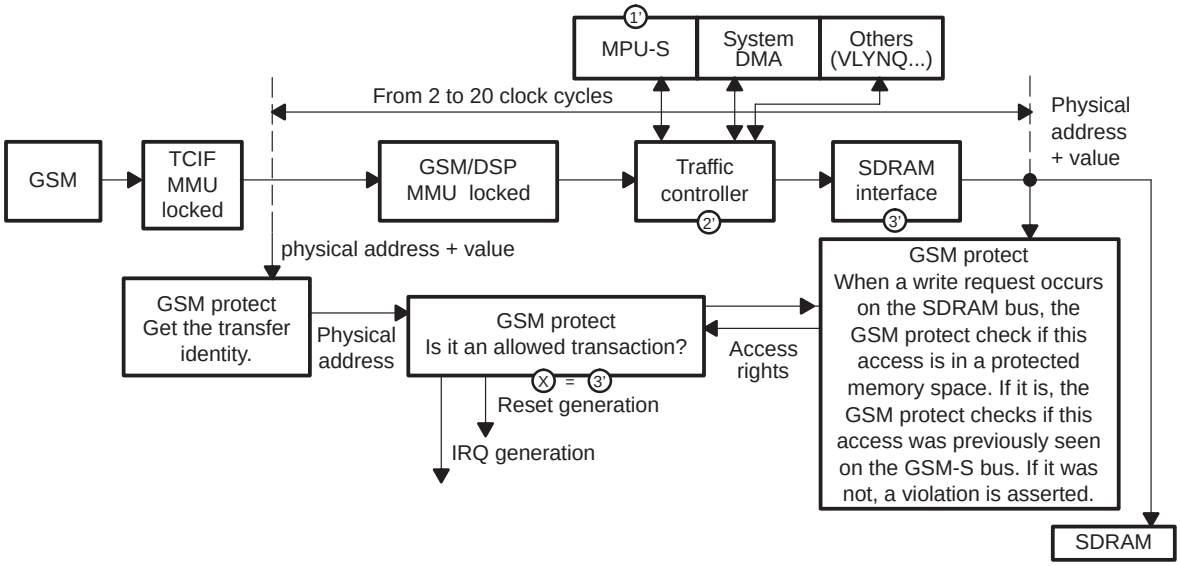
GSM write to its protected area:

1) The GSM-S generates a write access.
2) This access is translated into logical address by the TCIF.
3) The access is spied by the GSM protect.
4) The traffic controller dispatches the access to the SDRAM IF.
5) The SDRAM IF encodes the access in RAS/CAS format.

Steps 4 and 5 are compared by the GSM protect; if the access matches, no violation is generated.

Figure 3–13 shows an MPU write access to the GSM reserved memory space. The system DMA or another component such as VLYNQ can also generate the access. The case illustrated here shows where the GSM protect has not previously recorded the protected write access that is seen on SDRAM bus and, as a result, generates a violation.

Figure 3–13. MPU Write to GSM Protected Area



The MPU-S generates a write access:

1) The traffic controller dispatches the access to the SDRAM IF.
2) The SDRAM IF encodes the access in RAS/CAS format.

Because the protected SDRAM access does not correspond to a GSM-S access, a security violation is asserted.

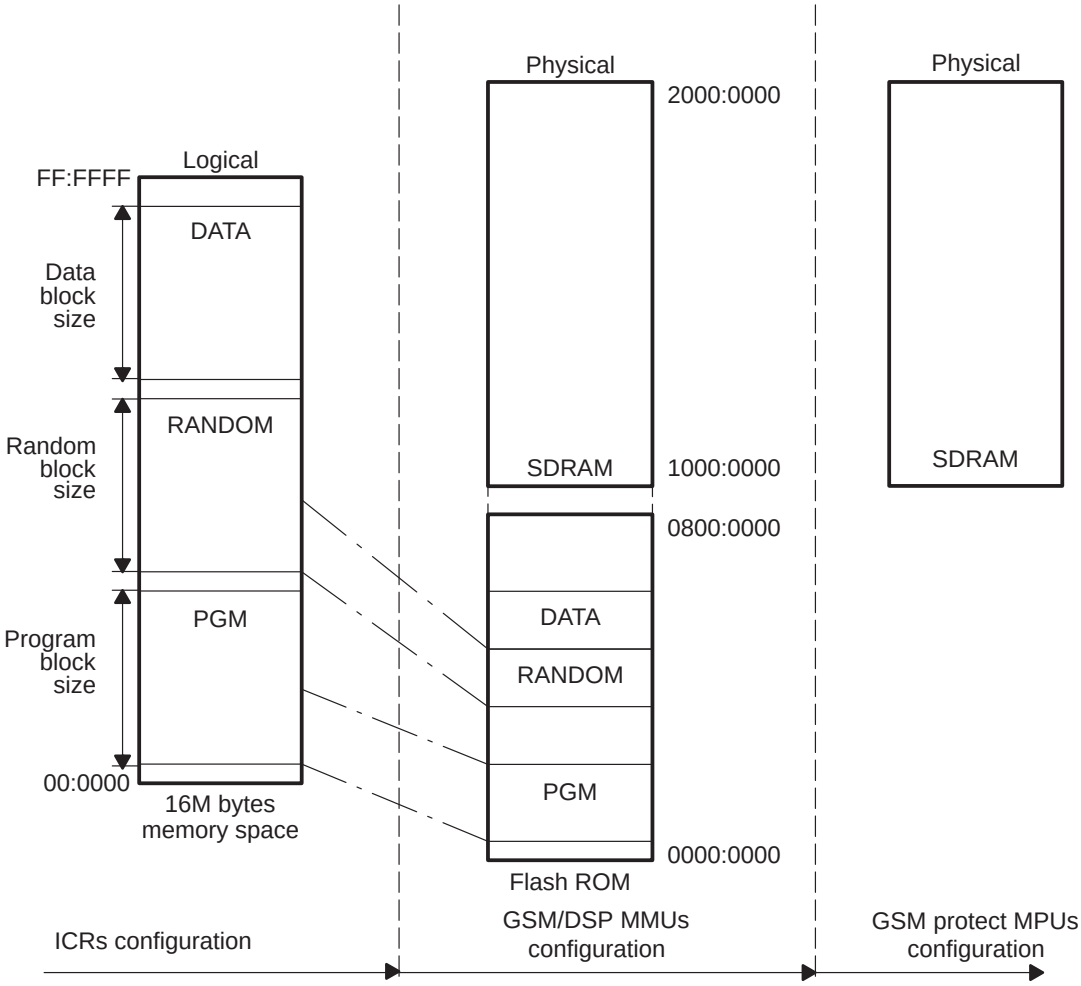
Note:

The write access to SDRAM is performed even if a violation is asserted.

3.32.3 GSM Memory Space Mapping and Protection Definition

Figure 3–14 illustrates the mapping of the GSM memory space on the flash component. This mapping can be done by configuring the GSM/DSP MMU component.

Figure 3–14. Mapping on Flash Component

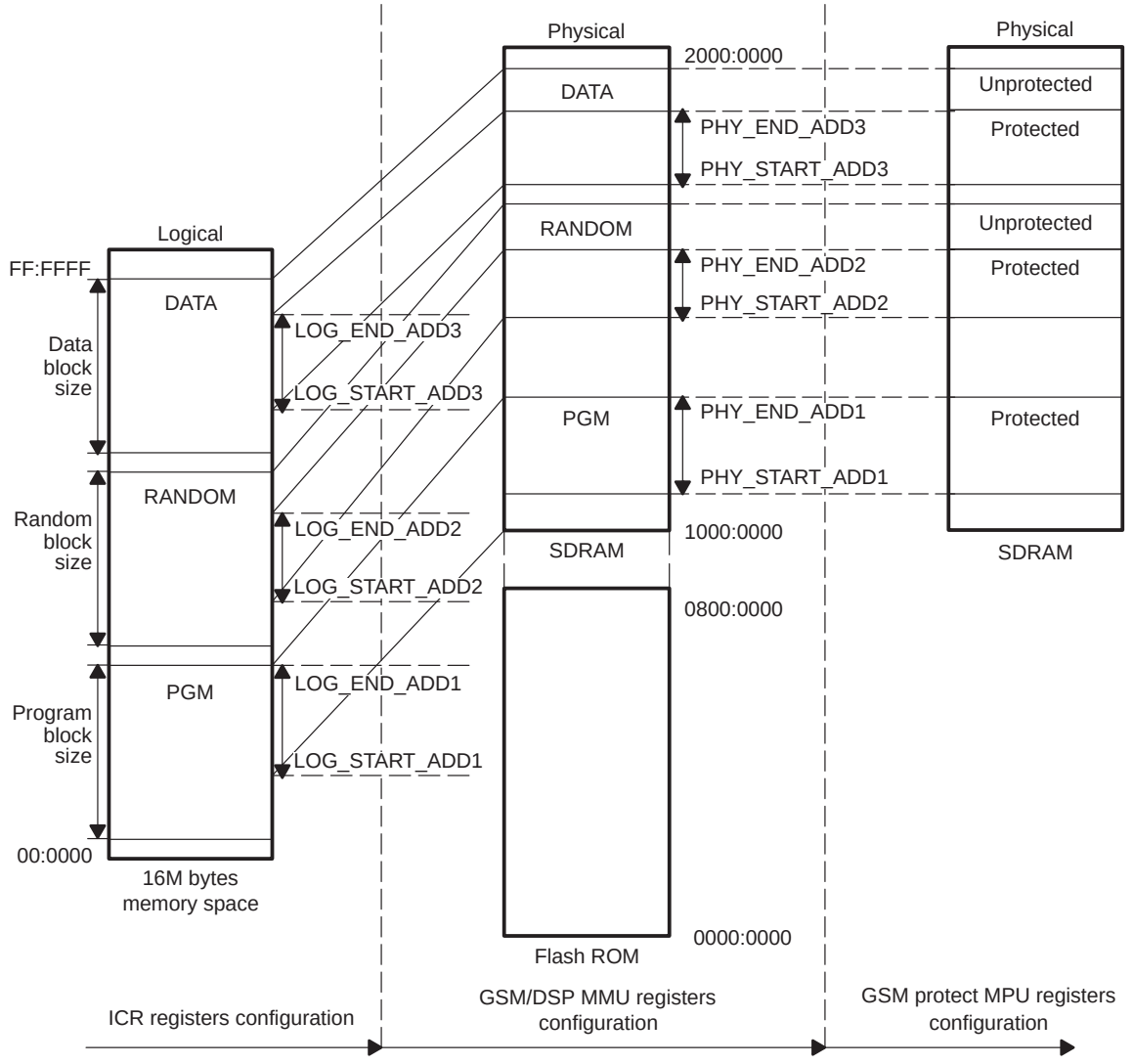


Note:

The GSM protect does not protect any part of the flash component. Refer to the flash protect component specification for additional information on this subject.

Figure 3–15 illustrates an example of the possible mapping of the GSM-S on the SDRAM component. As previously described, the GSM protect allows protection of only three zones. The logical and physical addresses described must be configured in the GSM protect registers in accordance with the TCIF/ICR and GSM-DSP MMU mapping configuration.

Figure 3–15. Mapping on SDRAM Component



Note: It is mandatory that no GSM instructions be stored in unprotected memory space on the SDRAM component (see security guidelines).

3.32.4 GSM/DSP MMU Restrictions

To formulate an efficient solution for the GSM_PROTECT, some limitations have been set on the use of the GSM/DSP MMU component.

To use the GSM_PROTECT properly and to ensure efficient protection, the three blocks (data, program, and random memory in the GSM section) can only be split into three parts on the SDRAM component (as shown in Figure 3–15). They can, however, consist of multiple sections: small pages, tiny pages, and others. This configuration is done using the GSM/DSP MMU.

Access permissions in the GSM/DSP MMU component are allowed to be set only for read (RAM_L_REG), but not for write , while the GSM_PROTECT does not take the GSM/DSP MMU access permissions into account.

Because of the necessity to lock the mapping of the GSM/DSP MMU and the ICR/TCIF by a write-once register after the boot process, an accurate configuration of this mapping is important because it cannot be changed after the boot process (see security guidelines).

3.32.5 Registers Implementation

As the GSM_PROTECT spies on the SDRAM bus, some registers of the SDRAM controller (SDRAM_IF) inside the traffic controller (TC) also act as spies. These registers spy to know and control the behavior of the SDRAM connected to the system. Thus, it is important that these registers be properly configured before activation of GSM_PROTECT.

These registers can be monitored for changes on the fly. Table 3–329 lists the SDRAM registers. Table 3–330 and Table 3–331 describe the register bits.

Table 3–329. SDRAM Registers Spied by GSM_PROTECT

Name	Description	Size	Offset	MPU-S Address (h)
EMIFF_CONF	EMIFF configuration	32 bits	0x20	FFFE:CC20
MRS_CONF	MRS configuration	32 bits	0x70	FFFE:CC70

These registers are directly duplicated from SDRAM_IF, which is the SDRAM controller interface into the GSM_PROTECT. The register replicas are done using the same TIPB chip-select and register memory space of the SDRAM IF; in other words, it is the exact image of the registers present in the GSM_PROTECT. The only way to modify them is to access the EMIF registers. Users must consider these two registers as part of the GSM_PROTECT configuration.

The main function of the register replicas is to keep track of all changes that can happen on the SDRAM type and timing configuration. If the SDRAM type changes when the GSM_PROTECT is running, some security-entry-point violation can take place.

For instance, the protection mechanism can be bypassed by changing the number of banks or the SDRAM interface timing. Therefore, special care must be taken in order to have reliable security protection.

Table 3–330. EMIFF Configuration Register (EMIFF_CONF)

Bit	Name	Function	R/W	Reset Value
31:30	Reserved			
29:28	SG SDRAM type	Used to define memories > 256 MB		
27:26	Reserved			
25:24	SDRAM_FREQ	**Encoding** **SDF0** **SDF1** **SDF2** **SDF3** Value [25:24] 00 01 10 11 **ac Parameters** **Timing (ns)** **SDF0 (Cycles)** **SDF1 (Cycles)** **SDF2 (Cycles)** **SDF3 (Cycles)** Trc 100 9 5 3 2 Tras 48 5 3 2 2 Trp 24 3 2 2 2 Trcd 24 3 2 2 2 Trrd*1 16 2 2 2 2 Tdpl (trwl*2) 8 – – – – Tdal 27 – – – – Trsc 2 2 2 2	R/W	0x00
23:8	Reserved			
7:4	SDRAM_TYPE	**Value [29:28] & [7:4]** **Memory Size (M bytes)** **Data Bus Size** **Number of Banks** 000000 16 8 2 000001 Reserved Reserved 000010 16 2 000011 Reserved Reserved 000100 64 8 2 000101 8 4 000110 16 2 000111 16 4 001000 128 Reserved Reserved 001001 8 4	R/W	0x0000 0000

Table 3–330. EMIFF Configuration Register (EMIFF_CONF) (Continued)

Bit	Name	Function				R/W	Reset Value
7:4	SDRAM_TYPE	**Value [29:28] & [7:4]**	**Memory Size (M bytes)**	**Data Bus Size**	**Number of Banks**	R/W	0x0000 0000
		001010		Reserved	Reserved		
		001011		16	4		
		001100	256	Reserved	Reserved		
		001101		8	4		
		001110		Reserved	Reserved		
		001111		16	4		
		010000	512	8	4		
		100000	1024	16	4		
3:0	Reserved						

Table 3–331. MRS Configuration Register (MRS_CONF)

Bit	Name	Function	R/W	Reset Value
31:6	Reserved			
6:4	CASL	CAS latency: 000: Reserved 001: Reserved 010: CAS latency = 2 cycles 011: CAS latency = 3 cycles	R/W	0x011
3:0	Reserved			

Table 3–332 lists the GSM protect registers. Table 3–333 through Table 3–346 describe the registers bits.

Table 3–332. GSM Protect Registers

Name	Description	Size	Offset	MPU-S Address (hex: FFFE:)
MPU_CTL	MPU control	32 bits	0x00	1800
LOG_START_ADD1	First block logical start address	32 bits	0x04	1804
LOG_END_ADD1	First block logical end address	32 bits	0x08	1808
LOG_START_ADD2	Second block logical start address	32 bits	0x0C	180C
LOG_END_ADD2	Second block logical end address	32 bits	0x10	1810
LOG_START_ADD3	Third block logical start address	32 bits	0x14	1814

Table 3–332. GSM Protect Registers (Continued)

Name	Description	Size	Offset	MPU-S Address (hex: FFFE:)
LOG_END_ADD3	Third block logical end address	32 bits	0x18	1818
PHY_START_ADD1	First block physical start address	32 bits	0x1C	181C
PHY_END_ADD1	First block physical end address	32 bits	0x20	1820
PHY_START_ADD2	Second block physical start address	32 bits	0x24	1824
PHY_END_ADD2	Second block physical end address	32 bits	0x28	1828
PHY_START_ADD3	Third block physical start address	32 bits	0x2C	182C
PHY_END_ADD3	Third block physical end address	32 bits	0x30	1830
SDRAM_CONF_VIOLATION	SDRAM configuration violation	32 bits	0x34	1834

Table 3–333. MPU Control Register (MPU_CTL)

Bit	Name	Function	R/W	Reset Value
31	VIOLATION_STATUS	Violation status 0: No violation 1: A violation occurred In case of a violation, this bit is cleared when the GSM reset is activated and then released, or when DEBUG_MODE is enabled; in this case, the bit is cleared after the end of the count set by field RESET_COUNTER.	R	0
30	VIOLATION_ TRACKER	Violation tracker: This special feature is needed when a violation occurs and the GSM reboots; no hardware flag remains to tell that a violation has occurred. This flag is used to fix security problems during the life cycle of the ASIC. 0: No violation had been recorded. 1: A violation occurred during a previous cession.	R	0
29	MPU_LOCK	When this bit is set, tGSM_PROTECT cannot be written until the next reset. 0: Not locked 1: Locked (the MPU configuration cannot be changed until the next reset)	R/W once	0
28	MPU_ACTIVATION	This bit activates GSM_PROTECT. 0: Not activated 1: Activated	R/W	0

Table 3–333. MPU Control Register (MPU_CTL) (Continued)

Bit	Name	Function	R/W	Reset Value
27:26	DEBUG_MODE	Debug mode activation: 00: A violation asserts a reset (GSM and MPU if selected) 01: A violation asserts nothing for the GSM part. If MPU_RESET_ENABLE is set, it sends a debug request to embedded OMAP3.2 ETM. Other: Reserved for future use	R/W	0
25	MPU_RESET_ENABLE	When a violation occurs, it asserts a reset to the MPU or a debug request to OMAP3.2 ETM if DEBUG_MODE is set. 0: Disable 1: Enable	R/W	0
24	GSM_RESET_MODE	GSM reset mode pulse or constant 0: Constant 1: Pulse	R/W	0
23:0	RESET_COUNTER	Reset counter parameter 24-bit counter on the 13-MHz clock Max value is 1.29 s. This counter delays the activation of the reset to allow the MPU and GSM to close their communications.	R/W	0

Table 3–334. First Block Logical Start Address Register (LOG_START_ADD1)

Bit	Name	Function	Reset Value
31:24	Reserved		
23:0	GSM_LOGICAL_START_ADD1	Logical start address for the GSM; 1st block area to protect	0x00000000

Table 3–335. First Block Logical End Address Register (LOG_END_ADD1)

Bit	Name	Function	Reset Value
31:24	Reserved		
23:0	GSM_LOGICAL_END_ADD1	Logical end address for the GSM; 1st block area to protect	0x00000000

Table 3–336. Second Block Logical Start Address Register (LOG_START_ADD2)

Bit	Name	Function	Reset Value
31:24	Reserved		
23:0	GSM_LOGICAL_START_ADD2	Logical start address for the GSM; 2nd block area to protect	0x00000000

Table 3–337. Second Block Logical End Address Register (LOG_END_ADD2)

Bit	Name	Function	Reset Value
31:24	Reserved		
23:0	GSM_LOGICAL_ END_ADD2	Logical end address for the GSM; 2nd block area to protect	0x00000000

Table 3–338. Third Block Logical Start Address Register (LOG_START_ADD3)

Bit	Name	Function	Reset Value
31:24	Reserved		
23:0	GSM_LOGICAL_ START_ADD3	Logical start address for the GSM; 3rd block area to protect	0x00000000

Table 3–339. Third Block Logical End Address Register (LOG_END_ADD3)

Bit	Name	Function	Reset Value
31:24	Reserved		
23:0	GSM_LOG- ICAL_END_ADD3	Logical end address for the GSM; 3rd block area to protect	0x00000000

Table 3–340. First Block Physical Start Address Register (PHY_START_ADD1)

Bit	Name	Function	Reset Value
31:28	Reserved		
27:0	SDRAM_PHY__ START_ADD1	Physical start address for the SDRAM corresponding to the 1st block area to protect	0x00000000

Table 3–341. First Block Physical End Address Register (PHY_END_ADD1)

Bit	Name	Function	Reset Value
31:28	Reserved		
27:0	SDRAM_PHY_END_ ADD1	Physical end address for the SDRAM corresponding to the 1st block area to protect	0x00000000

Table 3–342. Second Block Physical Start Address Register (PHY_START_ADD2)

Bit	Name	Function	Reset Value
31:28	Reserved		
27:0	SDRAM_PHY__ START_ADD2	Physical start address for the SDRAM corresponding to the 2nd block area to protect	0x00000000

Table 3–343. Second Block Physical End Address Register (PHY_END_ADD2)

Bit	Name	Function	Reset Value
31:28	Reserved		
27:0	SDRAM_PHY_END_ ADD2	Physical end address for the SDRAM corresponding to the 2nd block area to protect	0x00000000

Table 3–344. Third Block Physical Start Address Register (PHY_START_ADD3)

Bit	Name	Function	Reset Value
31:28	Reserved		
27:0	SDRAM_PHY__ START_ADD3	Physical start address for the SDRAM corresponding to the 3rd block area to protect	0x00000000

Table 3–345. Third Block Physical End Address Register (PHY_END_ADD3)

Bit	Name	Function	Reset Value
31:28	Reserved		
27:0	SDRAM_PHY_END_ ADD3	Physical end address for the SDRAM corresponding to the 3rd block area to protect	0x00000000

Table 3–346. SDRAM Configuration Violation Register (SDRAM_CONF_VIOLATION)

Bit	Name	Function	Reset Value
31:3	Reserved		
2	PROTECT_TYPE	If the GSM protect is locked when the value of the EMIFF configuration bits 29, 28, and 7:4 are changed, a violation is generated.	0x1
1	PROTECT_AC	If the GSM protect is locked when the value of the EMIFF configuration bits 25 and 24 are changed, a violation is generated.	0x1
0	PROTECT_MRS	If the GSM protect is locked when the value of the EMIFF configuration bits 6:4 are changed, a violation is generated.	0x1

3.32.6 Violation Handler

When a violation occurs, an interrupt is sent to the MPU (IRQ15 level 1) and an interrupt is sent to the GSM (IRQ24). The MPU can then read the MPU_CTL register to see the violation.

Depending on the register (MPU_CTL) configuration, when a violation occurs a counter can be set to delay the reset of the GSM. This delay also applies when MPU_RESET is selected in order to generate an OMAP730 reset.

Moreover, the type of reset for the GSM resets can be either pulse or constant.

The RESET_COUNTER allows some time to clean and clear the communication between the GSM and the MPU by delaying the reset generation.

Note:

The IRQ sent to the GSM must be handled as a violation interrupt bit while the GSM cannot access the MPU_CTL register through its TIPB.

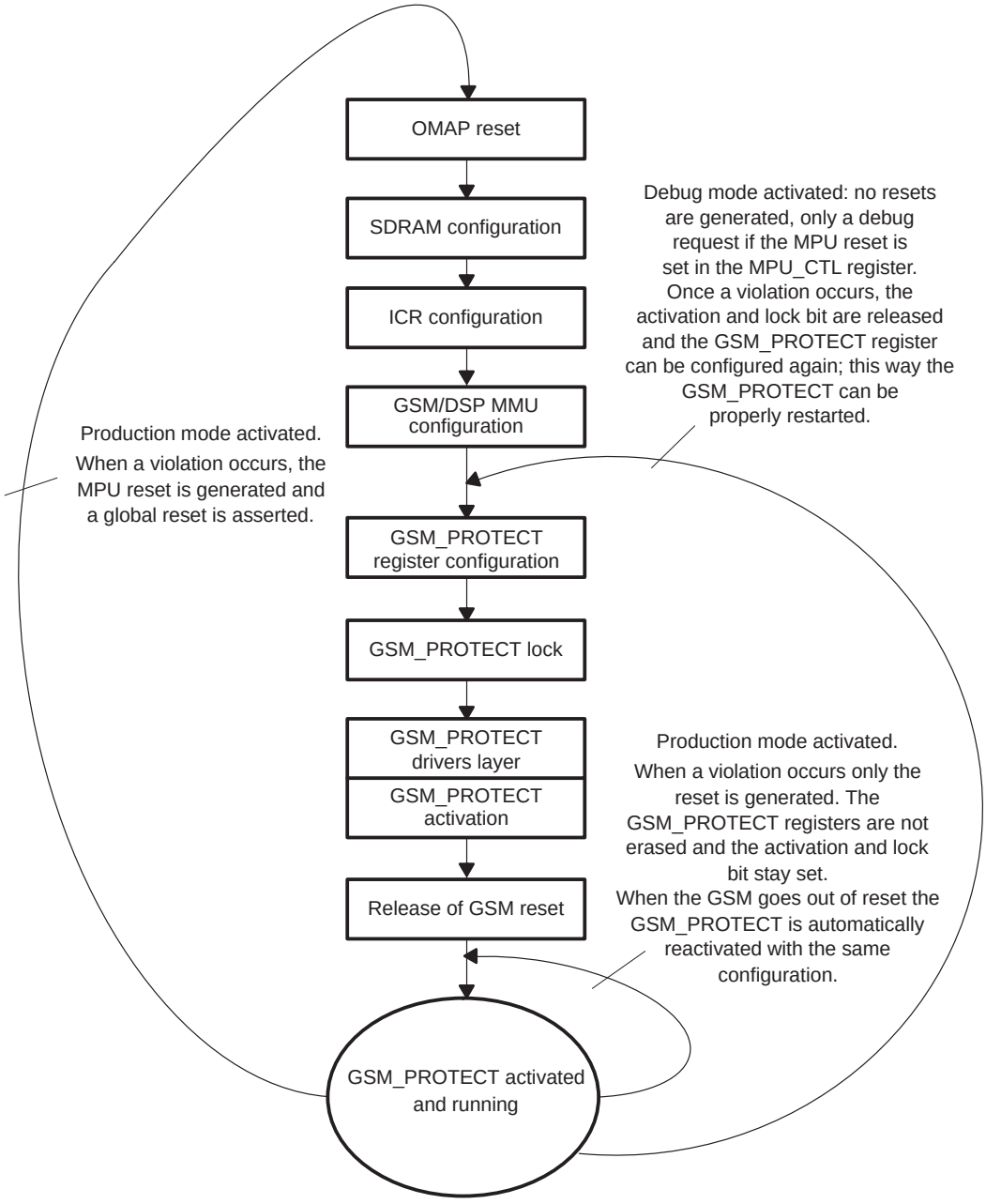
Note:

IRQs are also generated when in debug mode.

3.32.7 Reset Mode/Debug Mode State Machine

GSM_PROTECT supports three ways to handle violations. Two kinds of events are possible while in production mode: GSM reset and MPU reset. Only one event is possible while in debug mode. (See Figure 3–16.)

Figure 3–16. Reset Mode/Debug Mode State Machine



3.32.8 Security Guidelines

3.32.8.1 Configuring the GSM Protect for Maximum Security

The ICR/TCIF mapping must be set for each zone. This mapping must be locked by writing a 1 to bit 0 of the write-once register of the ICR at offset 0x10.

The DSP MMU mapping must be set according to ICR/TCIF previous mapping. Once set, this mapping must be locked by writing a 1 to bit 0 of the write-once register of the GSM/DSP MMU at offset 0x58.

The SDRAM_IF EMIFF_CONF and MRS_CONF registers must be configured before any activation of the GSM_PROTECT.

When the two mappings are done, the GSM_PROTECT mapping can be configured using the LOG_X and PHY_X registers (see Section 3.32.3 and Section 3.32.4).

It is recommended to activate the PROTECT_TYPE function in EMIFF_CONF violation register at offset 0x34 by setting bit 2 to 1.

- ❏ PROTECT_TYPE = SDRAM_CONF_VIOLATION[2] = 1

The protection of SDRAM ac characteristics and MRS configuration can be activated only if they are not supposed to change after the boot process. The PROTECT_AC function remains critical and it is recommended to activate it if the user programming model allows static configuration. The security goal is to enable as much protection as possible.

- ❏ PROTECT_AC = SDRAM_CONF_VIOLATION[1] =1
- ❏ PROTECT_MRS = SDRAM_CONF_VIOLATION[0] = 1

An easy example is to change the bank/row behavior of the SDRAM (different SDRAM type). Then, for an access found in a nonprotected area by the GSM_PROTECT, the SDRAM component stores this access in a protected area.

The protection on SDRAM registers is intended to prevent bypassing the GSM_PROTECT by creating faults on the SDRAM bus or by changing the characteristics of the SDRAM while running accesses in order to fool the TCIF, the SDRAM controller, and the GSM protect.

It is also important to configure the reset counter as soon as possible, because if a security failure generates a violation while the write access cannot be prevented (we only spy on the bus), the time allowed between the effective reset (reset counter) and the violation state can be used to make critical manipulations on the GSM content.

The authentication process on the GSM and full restore of GSM content must be done while rebooting.

3.32.8.2 General Security Requirements

GSM-MPU/DSP executable instructions must never be placed in nonprotected areas; otherwise, the security is fully compromised. Inserting a single assembly JMP instruction allows the GSM to self-transform its own code and compromise GSM behavior and its associated services.

3.32.8.3 Using the Violation Tracker—Extended Security Uses

When the GSM protect has generated a violation, the violation bit is cleared after rebooting the GSM, but the violation tracker remains high and cannot be cleared until the next hard reset.

For security matters, OS and native applications are never trusted because some assembly patches are easy to create and can be developed to hide any trace of the violation generated by GSM_PROTECT. The violation tracker was created in order to leave a hardware flag that can only be modified by a hard reset.

This flag does not have a predefined use, but it offers the valuable opportunity to synchronize the integrity check of the OS kernel, MMU mapping, and applications reauthentication and reauthoring when in OMAP secure mode (platform trusted area), for example.

Once a violation occurs, a security call can be created to determine its status and may be fed back on the attack. Countermeasures can be activated, such as security patch or Trojan/worm removers.

3.32.8.4 Software Constrains When Using GSM Protect

The use of a driver for good behavior of the GSM_PROTECT is mandatory. The SDRAM controller and especially the SDRAM state machine component impose constraints on the use of the GSM protect.

In fact, the SDRAM controller can keep the same row activated (see reference documents on SDRAM PC133) for several access.

When a violation occurs or when the GSM protect is activated, users must ensure that SDRAM row activation is done on the SDRAM bus. Otherwise, the GSM_PROTECT may see accesses with the wrong row (previously activated or noninitialized).

As a consequence, the GSM_PROTECT can generate nonjustified violations. This side effect can cause a violation loop and render the system unstable.

Drivers Description

The following functional sequence must be used to ensure that the GSM_PROTECT operation is stable:

If the MPU part is coming out of reset:

1) Disable SDRAM autorefresh.

2) Ensure that no module accesses the SDRAM component at any time. Take special care with active interrupts and the watchdog timer to prevent

unexpected results. The driver must mask all interrupts that can cause access to the SDRAM.

3) Activate the GSM_PROTECT.

4) Perform a READ access on each bank of the SDRAM in use. This causes an automatic TOW activation.

5) Reenable the SDRAM autorefresh.

6) Release the GSM-S reset.

If a violation occurs:

1) Disable SDRAM autorefresh.

2) Ensure that no module accesses the SDRAM. Take special care with active interrupts and the watchdog timer to prevent unexpected results. The driver must mask all interrupts that can cause access to the SDRAM.

3) Wait for the GSM reset after reset counter timing.

When the GSM reset is performed:

1) After a violation with GSM_PROTECT in normal mode, the GSM_PROTECT is already activated.

2) After a violation with GSM_PROTECT in debug mode, the GSM_PROTECT must be activated again.

   a) Perform a read access on each bank of the SDRAM in use. This provokes an automatic ROW activation.

   b) Reenable the SDRAM autorefresh.

   c) Release the GSM-S reset.

Note:

Before activation of the GSM protect, the row configuration is done in a way that automatically generates a violation if an access is performed without previous row activation.

3.32.9 Using the GSM Protect to Define a Read-Only SDRAM Platform Zone

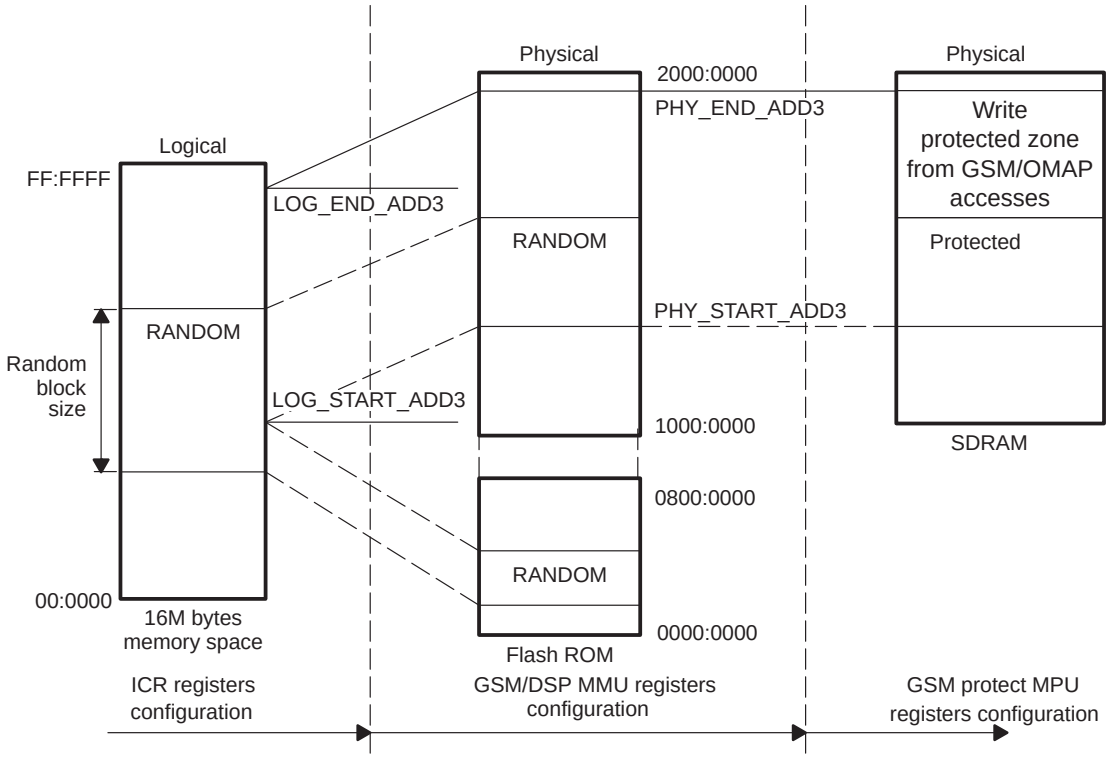
Because of the platform and GSM_PROTECT architectures, the GSM_PROTECT can provide an extra security feature to the OMAP platform.

If the ICR/TCIF and GSM/DSP MMU are properly configured, a write protected zone for OMAP and GSM on SDRAM memory can be set up.

The write-protected zone described must be filled by data before the activation of the GSM_PROTECT and must be a read-only zone for the entire platform.

Figure 3–17 shows the mapping that enables this extra security feature.

Figure 3–17. Mapping to Enable Extrasecurity Features



The zone defined is write-protected and cannot be accessed by the GSM or OMAP.

Chapter 4

Traffic Controller Interface

This chapter discusses the traffic controller interface (TCIF) module.

Topic **Page**

4.1 Traffic Controller Interface Module

The TCIF module allows the GSM-S memory interface to access OMAP730 external memory through the MPU-S traffic controller.

Four memory zones have been defined for GSM-S accesses: *program*, *data*, *constants,* and *random*. The choice between them is made by the GSM-S memory interface during the access by setting the appropriate chip-select and GSM-MPU NOPC (not op code) signal.

For all memory zones, the logical GSM-S address is transformed into a physical address before being sent to the MPU-S traffic controller.

4.1.1 Common Features of the Four Memory Zones

The MPU-S defines the size of each accessible zone by setting configuration values (these configuration registers are located outside the TCIF module). Program and constant zones are mapped into the same location.

If the GSM-S tries to perform an access outside this authorized zones, the data is written to the cache memory locations and, in some cases, to the external memory (a *line-flush*) and an interrupt is issued. In such cases, the TCIF must be reset by software.

The GSM-S address bus can not access more than 8M bytes, or less than the total external memory mapping. Consequently, the GSM-S addresses are remapped into the TC memory mapping via three base registers set by the MPU-S (one for each memory zone, located outside the TCIF module).

4.1.1.1 Program Memory Zone Features

This zone allows access to the program memory. It is defined as a read/write zone for debugging purposes (software break-points support).

In order to decrease external memory access count (especially for program loops) and duration (by using the TC burst capabilities), an 8-line cache (of 4×32-bit words each) is included in the module.

Due to specific memory restrictions on use, and to be compliant with the traffic controller module definition, all burst accesses to memories must be performed using a 32-bit aligned address (the four LSBs set to 0). This implies that the first byte address in a line cache must follow the same rule.

The external memory refresh after GSM-S write operations into this cache, as well as the cache refresh after MPU-S write operations into external memory, must be initiated by the GSM-S software by accessing a TCIF configuration register (all cache lines are flushed or completed with this software request). However, The TCIF may automatically initiate flush for one line in two cases:

- ❏ When it must free a line to use it for another address.
- ❏ When a line has been totally written by the GSM-S (flush performed in background).

In the same way, the TCIF automatically initiates a complete operation when the GSM-S reads from a memory location that is not present in the cache (not read from external memory and not already written by the GSM-S).

4.1.1.2 Data Memory Zone Features

The data memory zone allows access to the data memory. It is a read/write zone.

In order to decrease external memory access count when transferring large amounts of data as table or file contents, and to allow executing programs in this zone, an 8-line cache (of 4×32-bit words) is also included in the module.

Due to specific memory restrictions on use and to be compliant with the traffic controller module definition, all burst accesses to memories must be performed using a 32-bit aligned address (the four LSBs set to 0). This implies that the first byte address in the line cache must follow the same rule.

The external memory refresh after GSM-S write operations into this cache and the cache refresh after MPU-S write operations into the external memory must be initiated by GSM-S software by accessing a TCIF configuration register (with this software request, all cache lines are flushed or completed). However, The TCIF may automatically initiate the flush for one line in two cases:

- When it must free a line to use it for another address.
- When a line has been totally written by the GSM-S (flush performed in background).

In the same way, the TCIF automatically initiates a complete operation when the GSM-S reads from a memory location that is not present in the cache (not read from external memory and not already written by the GSM-S).

Simultaneous access to a single external data location from both *data* and *random* memory zones is not recommended. However, such accesses are possible, in which case the GSM-S must manage the data consistency between external memory and data cache. Indeed, an access through *data* memory zone may only update internal cache, whereas an access through *random* memory zone only updates external memory.

4.1.1.3 Constant Memory Zone Features

This zone allows access to the constant memory locations using program chip-select and NOPC signal. The constant memory is a read/write zone. In order to decrease latency when fetching constants (jump), a four-line cache (of 1×32 bits words) is also included in the module.

The external memory refresh after GSM-S write operations into this cache, as well as the cache refresh after MPU-S write operations into external memory must be initiated by GSM-S software by accessing a TCIF configuration register (with this software request, all cache lines are flushed or completed). Due to this zone's size, the TCIF automatically initiates flush for one line when a line has been totally written by the GSM-S (flush performed in background).

In the same way, the TCIF automatically initiates a complete operation when the GSM-S reads from a memory location that is not present in the cache (not read from external memory and not already written by the GSM-S).

4.1.1.4 Random Memory Zone Features

This zone also allows accessing the data memory. It is also a read/write zone.

The difference with the *data* memory zone is that accesses are considered to be performed word by word (as program variable accesses for example). In this case, there is no need to use a cache that would unnecessarily increase access time. Accesses are performed directly on the external memory through the traffic controller.

However, a single write buffer can be used to prevent stalling the GSM-S when it needs to perform a write access into external memory. The TCIF buffers all access parameters, releases the GSM-S, and performs the external access in the background.

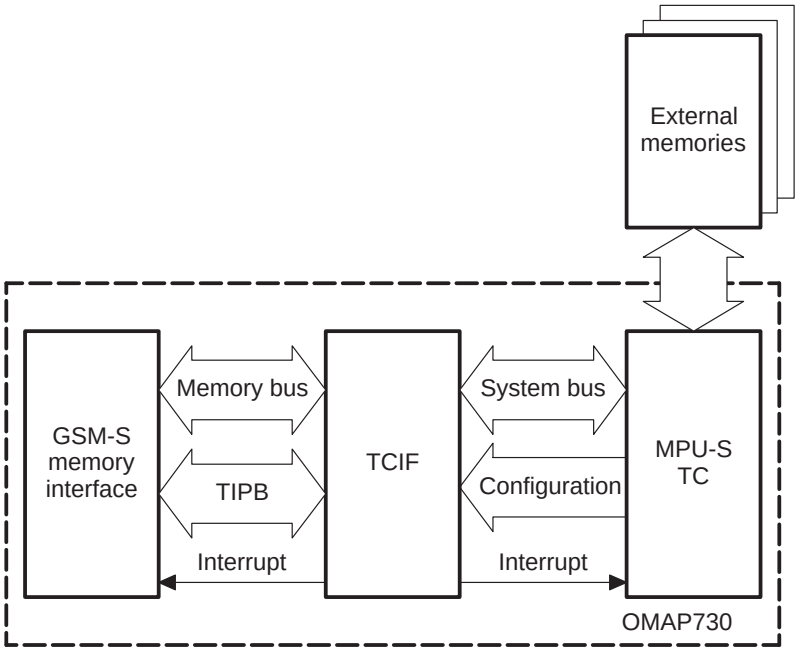
Simultaneous access to a single external data location from both *data* and *random* memory zones is not recommended. However, such accesses are possible, in which case the GSM-S must manage the data consistency between the external memory and the data cache. Indeed, an access through *the data* memory zone may only update internal cache, whereas an access through *random* memory zone only updates external memory.

4.2 TCIF Module Functionality

4.2.1 Module Behavior

The TCIF module allows the GSM-S memory interface to access external memories through the MPU-S traffic controller.

Figure 4–1. TCIF Behavior



Data transfers are performed through the memory and system buses.

The GSM-S TIPB bus allows configuring the TCIF module and flushing the memory caches.

The GSM-S interrupt line is activated when the GSM-S performs an access outside the authorized zones (controlled by MPU-S), or when the TC sends back an abort.

The configuration values from MPU-S define the TCIF memory management unit constants.

The MPU-S interrupt line is activated when the MPU-S configuration values are inconsistent with each other.

4.2.2 Module Functionality

The TCIF module functionality can be split into two major domains:

- ❏ Memory management: Dedicated to address translations and accesses rights control.
- ❏ Buffer and cache: Dedicated to all cache and buffer operations.

4.2.2.1 Memory Management

The process of conversion from logical to physical address is identical for all memory zones. Only block and size parameters differ between them, as shown in Table 4–1.

Table 4–1. Logical to Physical Address Conversion Parameters

Parameter	Program and Constant Zones	Data Zone	Random Zone
GSM-S address	AD_GSM_MEM		
Block size	PGM_BLK_SIZE	DATA_BLK_SIZE	RAND_BLK_SIZE
Block base address	PGM_BLK_ADDR	DATA_BLK_ADDR	RAND_BLK_ADDR
MPU-S address	ADD_MPU_TC		

Address Translation Process

When the GSM-S wants to perform an access into the external program memory, the TCIF module computes the MPU-S address using the following parameters:

- Memory zone block size: This configuration value is set by the MPU-S outside the TCIF module.
- Memory zone base address: This configuration value is set by the MPU-S outside the TCIF module. The useful address vector part depends on the memory-zone block-size parameter:
  - From 7 bits for an 8M memory zone block
  - To 14 bits for a 64K memory zone block
- GSM-S address: The useful address vector part depends on the memory zone block size parameter:
  - From 23 bits for an 8M memory zone block
  - Down to 16 bits for a 64K memory zone block

The useful address vector range is shown in Table 4–2.

Table 4–2. Useful Address Vector Range

Block Size	Useful Base Address Range	Useful GSM-S Address Range
64K	13:0	15:0
128K	13:1	16:0
256K	13:2	17:0
512K	13:3	18:0
1M	13:4	19:0
2M	13:5	20:0

Table 4–2. Useful Address Vector Range (Continued)

Block Size	Useful Base Address Range	Useful GSM-S Address Range
4M	13:6	21:0
8M	13:7	22:0

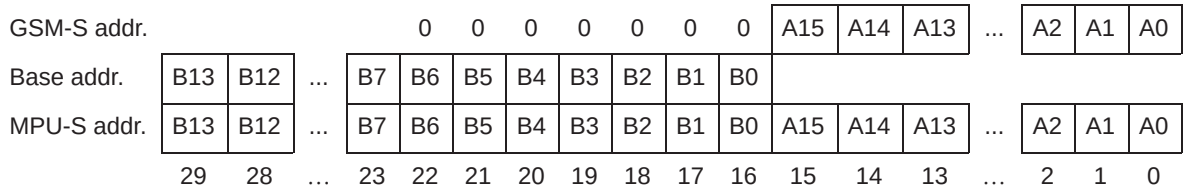
Notes: 1) For a given block size, all base address unused bits must be set to 0. A permanent control is done about this requirement as soon as the GSM-S reset becomes inactive. A nonmaskable level interrupt is set to the MPU-S in case of inconsistency.

2) Base and block size parameters must not be changed by the MPU-S while the GSM-S reset is inactive to prevent transient problems.

3) When the GSM-S performs an access, all GSM-S address unused bits must be set to 0. If this requirement is not met, the TCIF considers that the access to be out of bound and:
– Does not perform it
– Generates an interrupt to the GSM-S

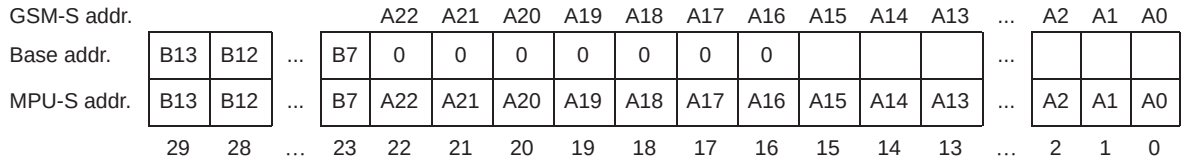
An example of a 64K-byte block is given in Figure 4–2.

Figure 4–2. 64K-Byte Block Size Example



An example of an 8M-byte block is given in Figure 4–3.

Figure 4–3. 8M-Byte Block Size Example



4.2.2.2 Buffer and Cache

Program Constants/NOPC or Data-Zone Access

These zones allow accessing external memory in read/write mode.

In order to decrease external memory accesses count (loops in program or table transfer), two independent caches are included in the module (one for each memory zone).

The caches are enabled or disabled by GSM-S software using programmable registers.

Note:

Simultaneous access of a single external data location from both data and random memory zone is not recommended. However, such access is possible. In this case, the GSM-S must manage the data consistency between the external memory and the data cache. An access through data memory zone only updates internal cache, whereas an access through random memory zone only updates external memory.

- Read access without cache
  - The GSM-S is stalled while the TCIF performs the read access directly from external memory.
- Write access without cache
  - The GSM-S is stalled while the TCIF performs the write access directly into the external memory.
- Read access with cache
  - The memory location is already in the cache:
    - The requested value stored in the cache is sent back to the GSM-S without consistency control between this value and the actual external memory value.
- The memory location is not yet in the cache, or the cache has been invalidated by software control access:
  a) The TCIF allocates to this access the cache line that has been unused for the longest period (LRU).
  b) If data have been written into this old page by the GSM-S but have not yet been automatically saved into the external memory (to free it in order to allocate it to another address, or when the GSM-S has written data into the entire page) or by software request:
     i) f necessary, the TCIF loads the external memory corresponding page to set in the cache all the data that have not been initialized by the GSM-S (byte granularity).
     ii) The TCIF saves the cache line into the external memory.
  c) The TCIF loads in this cache line the external memory page corresponding to this access.
  d) The TCIF sends back to the GSM-S the requested value.
- Write access with cache
  - The memory location is already in the cache:
  a) The TCIF writes the data into the cache.
  b) If the cache line has been fully written by the GSM-S, the TCIF saves it into external memory. This operation is performed in the background so the GSM-S may access another cache line at the same time.
  - The memory location is not yet in the cache, or the cache has been invalidated by software control access:

a) The TCIF allocates to this access the cache line that has not been used for the longest period (LRU).

b) If data have been written into this old page by the GSM-S but have not been saved into the external memory automatically (to free it in order to allocate it to another address, or when the GSM-S has written data into the entire page) or by software request:

   i) If necessary, the TCIF loads the external memory corresponding page to set in the cache all the data that have not been initialized by the GSM-S (byte granularity).

   ii) The TCIF saves the cache line into the external memory.

c) The TCIF writes the data into the cache.

Random Zone Access

This zone allows accessing external memory in read/write mode. The GSM-S can only access it word by word. A single buffer prevents stalling of the GSM-S processor when it performs a write access.

The register is enabled or disabled by GSM-S software using a TIPB register.

It is not recommended to access a single external data location simultaneously from both data and random memory zones. However, such access is possible. In this case, the GSM-S must manage the data consistency between the external memory and the data cache. An access through data memory zone may only update internal cache, whereas an access through random memory zone only updates external memory.

❏ Read access

The GSM-S is stalled while the TCIF performs the read access directly from external memory.

❏ Write access without buffer

The GSM-S is stalled while the TCIF performs the write access directly to external memory.

❏ Write access with buffer

a) The access parameters are buffered into the TCIF.

b) The GSM-S is released, while the TCIF performs the access in background.

c) If necessary, the GSM-S may read the access status (finished or not) from a TIPB register.

4.2.3 Module Registers

All TCIF registers (functional and debug) are accessed through the GSM-S TIPB bus with a chip-select value of 21.

Table 4–3 lists the 16-bit TCIF functional registers. Table 4–4 through describe the register bits.

Table 4–3. TCIF Functional Registers

Register	Description	R/W	Offset	Reset Value
TCIF_CTL	TCIF general control	R/W	000 h	0x0000
PGM_CACHE_CTL	Program memory cache control	R/W	002 h	0x0000
DATA_CACHE_CTL	Data memory cache control	R/W	004 h	0x0001
RAND_BUFFER_CTL	Random memory buffer control	R/W	006 h	0x0001
NOPC_BUFFER_CTL	Constant memory cache control	R/W	008 h	0x0001

Table 4–4. TCIF General Control Register (TCIF_CTL)

Bit	Name	Function	Reset Value
15:2	–	Reserved—Must be set to 0, read as 0	0
1	ABORT_IT_EN	When the TC sends back an abort for a GSM-S access: 0: No interrupt is generated. 1: An interrupt is generated to the GSM-S.	0
0	OUT_IT_EN	When the GSM-S wants to perform an access outside its authorized memory zone (address value greater than the memory zone block size), the TCIF does not enable it and… 0: No interrupt is generated. 1: An interrupt is generated to the GSM-S.	0

Table 4–5. Program Memory Cache Control Register (PGM_CACHE_CTL)

Bit	Name	Function	Reset Value
15:3	–	Reserved—Must be set to 0, read as 0	0
2	FLUSH_W	When this bit is set in write access, all data written into the cache by the GSM-S are sent into the external memory (for all cache lines). This bit is automatically reset when the flush is completed.	0
1	INVALIDATE	When this bit is set in write access, all cache data are invalidated. That means that for the next read access from the GSM-S, the TCIF refreshes cache contents from the external memory, even if data were present in it before setting this bit. This bit is automatically reset when the operation is completed. **Note:** All GSM-S data written into the cache before setting this bit are lost. To save the data into external memory before erasing them, it is also necessary to set the FLUSH_W bit (in the same or in a previous configuration access).	0
0	CACHE_EN	0: Cache is disabled. 1: Cache is enabled. **Note:** All GSM-S data written into the cache before disabling it are lost. To first save the data into the external memory, it is also necessary to set the FLUSH_W bit (in the same or in a previous configuration access). When the cache state changes from disable to enable in a write access to this register, the cache contents is automatically invalidated.	1

Table 4–6. Data Memory Cache Control Register (DATA_CACHE_CTL)

Bit	Name	Function	Reset Value
15:3	–	Reserved—Must be set to 0, read as 0.	0
2	FLUSH_W	When this bit is set in write access, all data written into the cache by the GSM-S are sent to the external memory (for all cache lines). This bit is automatically reset when the flush is completed.	0
1	INVALIDATE	When this bit is set in write access, all cache data are invalidated. This means that for the next read access by the GSM-S, the TCIF refreshes the cache contents from external memory, even if data were present in it before setting this bit. This bit is automatically reset when the operation is completed. **Note:** All GSM-S data written into the cache before setting this bit are lost. To save this data into external memory before erasing them, it is also necessary to set the FLUSH_W bit (in the same or in a previous configuration access).	0
0	CACHE_EN	0: Cache is disabled. 1: Cache is enabled. **Note:** All GSM-S data written into the cache before disabling it are lost. To first save this data into external memory, it is also necessary to set the FLUSH_W bit (in the same or in a previous configuration access). During a write access to this register, the cache state changes from disable to enable and the cache contents is automatically invalidated.	1

Table 4–7. Random Memory Buffer Control Register (RAND_BUFFER_CTL)

Bit	Name	Function	Reset Value
15:2	–	Reserved—Must be set to 0, read as 0	0
1	W_BUFFER_FULL	When the GSM-S performs a write with buffer access, this read only bit is set to 1 by the TCIF throughout the entire access to the external memory.	0
0	W_BUFFER_EN	When the GSM-S performs a write access: 0: The GSM-S is stalled until access completion. 1: The access is buffered to be executed in the background and the GSM-S is not stalled until access completion. However, the GSM-S can learn the access status (finished or not) by reading the W_BUFFER_FULL bit.	1

Table 4–8. Constant Memory Cache Control Register (NOPC_CACHE_CTL)

Bit	Name	Function	Reset Value
15:4	–	Reserved—Must be set to 0, read as 0.	0
3	NOPC_EN	0: NOPC is disabled (constants zone is enabled). 1: NOPC is enabled (constants zone is disabled).	0
2	FLUSH_W	When this bit is set to write access, all data that have been written into the cache by the GSM-S are sent to the external memory (for all cache lines). This bit is automatically reset when the flush is completed.	0
1	INVALIDATE	When this bit is set in write access, all cache data are invalidated. That means that for the next read access from the GSM-S, the TCIF refreshes cache contents from the external memory, even if data were present in it before setting this bit. This bit is automatically reset when the operation is completed. **Note:** All the GSM-S data written into the cache before setting this bit are lost. To save this data into external memory before erasing them, it is also necessary to set the FLUSH_W bit (in the same or in a previous configuration access).	0
0	CACHE_EN	0: Cache is disabled. 1: Cache is enabled. **Note:** This bit is only accessible in read when accessing the NOPC_CACHE_CTL register and is written when accessing the PROGRAM_CACHE_CTL register to enable the program cache.	1

Table 4–9 lists the 16-bit TCIF debug registers. Table 4–10 through Table 4–25 describe the register bits.

Table 4–9. TCIF Debug Registers

Register	Description	R/W	Offset	Reset Value
IT_DESCRIPTION	TCIF incoming interrupt description	R	010 h	0x0000
IT_ADDRESS_L	Last TCIF incoming interrupt high	R	012 h	0x0000
IT_ADDRESS_H	Last TCIF incoming interrupt low	R	014 h	0x0000
COUNT_MNGT	Debug counter control	R/W	020 h	0x0000
GSM_PGM_COUNT_L	GSM program counter low	R	030 h	0x0000
GSM_PGM_COUNT_H	GSM program counter high	R	032 h	0x0000
GSM_DATA_COUNT_L	GSM data counter low	R	034 h	0x0000
GSM_DATA_COUNT_H	GSM data counter high	R	036 h	0x0000
GSM_RAND_COUNT_L	GSM random memory counter low	R	038 h	0x0000
GSM_RAND_COUNT_H	GSM random memory counter high	R	03A h	0x0000
MPU_PGM_COUNT_L	MPU program counter low	R	040 h	0x0000
MPU_PGM_COUNT_H	MPU program counter high	R	042 h	0x0000
MPU_DATA_COUNT_L	MPU data counter low	R	044 h	0x0000
MPU_DATA_COUNT_H	MPU data counter high	R	046 h	0x0000
MPU_RAND_COUNT_L	MPU random memory counter low	R	048 h	0x0000
MPU_RAND_COUNT_H	MPU random memory counter high	R	04A h	0x0000

Table 4–10. TCIF Incoming Interrupt Description Register (IT_DESCRIPTION)

Bit	Name	Function	Reset Value
15:4	–	Reserved—Read as 0.	0
3	ERR_TC	If this bit is set, an abort has been sent back by the TC concerning this GSM-S access.	0
2	ERR_OUT	If this bit is set, the GSM-S address was out of bounds.	0
1:0	ZONE	The GSM-S zone of faulty access: 00: Program 01: Data 10: Random	0

IT_DESCRIPTION is a debug register used by the GSM-S to obtain details about the last TCIF incoming interrupt.

Table 4–11. Last TCIF Incoming Interrupt Low Register (IT_ADDRESS_L)

Bit	Name	Function	Reset Value
15:0	ADD_L	ERR_OUT error: GSM-S address of faulty access: Bits (15:0). ERR_TC error: MPU-S address of faulty access: Bits (15:0).	0

Table 4–12. Last TCIF Incoming Interrupt High Register (IT_ADDRESS_H)

Bit	Name	Function	Reset Value
15:12	–	Reserved—Read as 0	0
13:0	ADD_H	ERR_OUT error: Bits 13–7: Read as 0. Bits 6–0: GSM-S address of faulty access—Bits (22:16). ERR_TC error: MPU-S address of faulty access: Bits (29:16).	0

IT_ADDRESS_H is a debug register used by the GSM-S to obtain details about the last TCIF incoming interrupt.

Debug counters have been added to the TCIF module on all data access ports to measure their activities:

- On the GSM-S interface, three independent counters monitor the requested accesses (one for each memory zone).
- On the MPU-S interface, three independent counters monitor the resulting burst accesses (one for each memory zone).

Table 4–13. Debug Counter Control Register (COUNT_MNGT)

Bit	Name	Function	Reset Value
15:2	–	Reserved. Must be set to 0, read as 0.	0
1	RESET	Writing a 1 into this bit resets all counters. This bit is automatically reset by the TCIF. **Note:** Counters must be off before and after performing the reset to avoid synchronization problems (RESET and ON bit must not be modified in the same access).	0
0	ON	When high, all counters are on. **Note:** Because of internal synchronization when accessing this bit, counter states may not all change at exactly the same time (MPU-S counters can change after GSM-S counters).	0

The COUNT_MNGT debug register allows control of this test behavior.

Table 4–14 and Table 4–15 describe the GSM_PGM_COUNT_x debug registers, which allow to access the program zone GSM-S accesses counter.

Table 4–14. GSM Program Counter Low Register (GSM_PGM_COUNT_L)

Bit	Name	Function	Reset Value
15:0	COUNT_L	GSM-S access request in program memory zone count since last register reset: Bits (15–0).	0

Note: To avoid synchronization problems, these registers must be read only when the counter is off.

Table 4–15. GSM Program Counter High Register (GSM_PGM_COUNT_H)

Bit	Name	Function	Reset Value
15:8	–	Reserved—Read as 0	0
7:0	COUNT_H	GSM-S access request in program memory zone count since last register reset: Bits (23–16).	0

Table 4–16 and Table 4–17 describe the GSM_DATA_COUNT_x debug registers, which allow to access the data zone GSM-S accesses counter.

Table 4–16. GSM Data Counter Low Register (GSM_DATA_COUNT_L)

Bit	Name	Function	Reset Value
15:0	COUNT_L	GSM-S access request in data memory zone count since last register reset: Bits (15–0).	0

Note: To avoid synchronization problems, these registers must be read only when the counter is off.

Table 4–17. GSM Data Counter High Register (GSM_DATA_COUNT_H)

Bit	Name	Function	Reset Value
15:8	–	Reserved—Read as 0	0
7:0	COUNT_H	GSM-S access request in data memory zone count since last register reset: Bits (23–16).	0

Table 4–18 and Table 4–19 describe the GSM_RAND_COUNT_x debug registers, which allow to access the random zone GSM-S accesses counter.

Table 4–18. GSM Random Memory Counter Low Register (GSM_RAND_COUNT_L)

Bit	Name	Function	Reset Value
15:0	COUNT_L	GSM-S access request in random memory zone count since last register reset: Bits (15–0).	0

Note: To avoid synchronization problems, these registers must be read only when the counter is off.

Table 4–19. GSM Random Memory Counter High Register (GSM_RAND_COUNT_H)

Bit	Name	Function	Reset Value
15:8	–	Reserved—Read as 0	0
7:0	COUNT_H	GSM-S access request in random memory zone count since last register reset: Bits (23–16).	0

Table 4–20 and Table 4–21 describe the MPU_PGM_COUNT_x debug registers, whick allow to access the program zone MPU-S burst accesses counter.

Table 4–20. MPU Program Counter Low Register (MPU_PGM_COUNT_L)

Bit	Name	Function	Reset Value
15:0	COUNT_L	MPU-S burst accesses related to GSM-S access request in program memory zone count since last register reset: its (15–0).	0

Note: To avoid synchronization problems, these registers must be read only when the counter is off.

Table 4–21. MPU Program Counter High Register (MPU_PGM_COUNT_H)

Bit	Name	Function	Reset Value
15:8	–	Reserved—Read as 0.	0
7:0	COUNT_H	MPU-S burst accesses related to GSM-S access request in program memory zone count since last register reset: Bits (23–16).	0

Table 4–22 and Table 4–23 describe the MPU_DATA_COUNT_x debug registers, which allow to access the data zone MPU-S burst accesses counter.

Table 4–22. MPU Data Counter Low Register (MPU_DATA_COUNT_L)

Bit	Name	Function	Reset Value
15:0	COUNT_L	MPU-S burst accesses related to GSM-S access request in data memory zone count since last register reset: Bits (15–0).	0

Note: To avoid synchronization problems, these registers must be read only when the counter is off.

Table 4–23. MPU Data Counter High Register (MPU_DATA_COUNT_H)

Bit	Name	Function	Reset Value
15:8	–	Reserved—Read as 0.	0
7:0	COUNT_H	MPU-S burst accesses related to GSM-S access request in data memory zone count since last register reset: Bits (23–16).	0

Table 4–24 and Table 4–25 describe The MPU_RAND_COUNT_x debug registers, which allow to access the random zone MPU-S burst accesses counter.

Table 4–24. MPU Random Memory Counter Low Register (MPU_RAND_COUNT_L)

Bit	Name	Function	Reset Value
15:0	COUNT_L	MPU-S burst accesses related to GSM-S access request in random memory zone count since last register reset: Bits (15–0).	0

Note: To avoid synchronization problems, these registers must be read only when the counter is off.

Table 4–25. MPU Random Memory Counter High Register (MPU_RAND_COUNT_H)

Bit	Name	Function	Reset Value
15:8	–	Reserved—Read as 0.	0
7:0	COUNT_H	MPU-S burst accesses related to GSM-S access request in random memory zone count since last register reset: Bits (23–16).	0

4.3 Using TCIF Software

This section summarizes the use of the TCIF software.

4.3.1 Memory Management

4.3.1.1 Base and Block Size Configuration

The base and block size for all memory zones are defined by the MPU-S, via its TIPB bus, outside the TCIF module.

These parameters must be stable as long as the GSM-S reset is inactive.

A nonmaskable level-sensitive interrupt is generated by the TCIF if these parameters are not consistent with each other, as soon as the GSM-S reset is inactive.

4.3.1.2 GSM-S Accesses

Each memory zone (program, data, or random) has its own software memory space from the GSM-S code point of view.

If the GSM-S performs an out-of-bound access (the address value is greater than the selected memory zone block size), an edge-sensitive interrupt is sent back to it (this interrupt may be enabled or disabled via the OUT_IT_EN bit in the TCIF_CTL configuration register). Note that the GSM-S memory bus is hung during this operation.

If the MPU-S traffic controller sends back an abort for a GSM-S access, an edge-sensitive interrupt is also sent back to the GSM-S (this interrupt may be enabled or disabled via the ABORT_IT_EN bit in the TCIF_CTL configuration register).

4.3.2 Buffer and Cache

4.3.2.1 GSM-S Access in Program Constants or Data Zones

Disabling the Cache

To disable the cache without flushing it, all data written to it by the GSM-S without being saved to external memory (automatically when the cache management detects a fully written line or when it needs space or by software request) are lost.

The GSM-S disables the cache by resetting the CACHE_EN bit in the PGM/DATA_CACHE_CTL register.

Flushing and Disabling the Cache

To first save the cache contents into the external memory and then disable the cache:

1) The GSM-S flushes and disables the cache by setting the FLUSH_W and resetting the CACHE_EN bits in the ZONE_CACHE_CTL register (in the same or in consecutive accesses).

2) The GSM-S may wait for the end of flush by polling the FLUSH_W bit.

Enabling the Cache

To enable the cache for the next accesses:

1) The GSM-S enables the cache by setting the CACHE_EN bit in the ZONE_CACHE_CTL register. When performing this operation, if the cache status changes from disabled to enabled, its contents is automatically invalidated.

2) The GSM-S may wait to the end of the invalidate operation by polling the INVALIDATE bit.

Erasing the Cache

To invalidate the cache contents without flushing it: All data written to it by the GSM-S without being saved in external memory (automatically when the cache management detects a fully written line, when it needs space, or by software request) are lost.

1) The GSM-S invalidates the cache by setting the INVALIDATE bit in the ZONE_CACHE_CTL register.

2) The GSM-S may wait to the end of the invalidate operation by polling the INVALIDATE bit.

Flushing and Erasing the Cache

To first save the cache's contents into the external memory and then invalidate the cache:

1) The GSM-S flushes and invalidates the cache by setting the FLUSH_W and the INVALIDATE bits in the ZONE_CACHE_CTL register (in the same or in consecutive accesses).

2) The GSM-S may wait for the end of the previous operations by polling the FLUSH_W and the INVALIDATE bits.

Flushing the Cache

To save the cache's contents into the external memory (the data in the cache are still valid):

1) The GSM-S flushes the cache by setting the FLUSH_W bit in the ZONE_CACHE_CTL register.

2) The GSM-S may wait for the end of flush by polling the FLUSH_W bit.

Access Without Cache

When the cache is disabled, the GSM-S performs the access directly to the external memory. It remains stalled until access completion.

Access With Cache

To perform an access when the cache is enabled:

1) The GSM-S performs the access to the cache. It remains stalled until access completion. If the cache management unit has to free a cache line before executing the requested access (if the memory location is not already in the cache), it automatically initiates a flush in order to write its contents into external memory (if necessary).

2) If the access is a write operation and the GSM-S wants to be sure that the data is saved to external memory, it must request a cache flush by setting the FLUSH_W bit in the ZONE_CACHE_CTL register.

3) If a flush has been requested, the GSM-S may wait for its end by polling on the FLUSH_W bit.

4.3.2.2 GSM-S Access in Random Zone

Disabling the Write Buffer

The GSM-S disables the use of the buffer by resetting the W_BUFFER_EN bit in the RAND_BUFFER_CTL register.

Enabling the Write Buffer Use

The GSM-S enables the use of the buffer by setting the W_BUFFER_EN bit in the RAND_BUFFER_CTL register.

Read Access

The GSM-S performs the read access directly from external memory, whether the write buffer is used or not. It remains stalled until access completion.

Write Access Without Write Buffer

The GSM-S performs the write access directly into external memory when the write buffer is disabled. It remains stalled until access completion.

Write Access With Write Buffer

To perform a write access when the write buffer is enabled:

1) The TCIF buffers all GSM-S write access parameters and releases the processor.

2) The TCIF performs the access in the background.

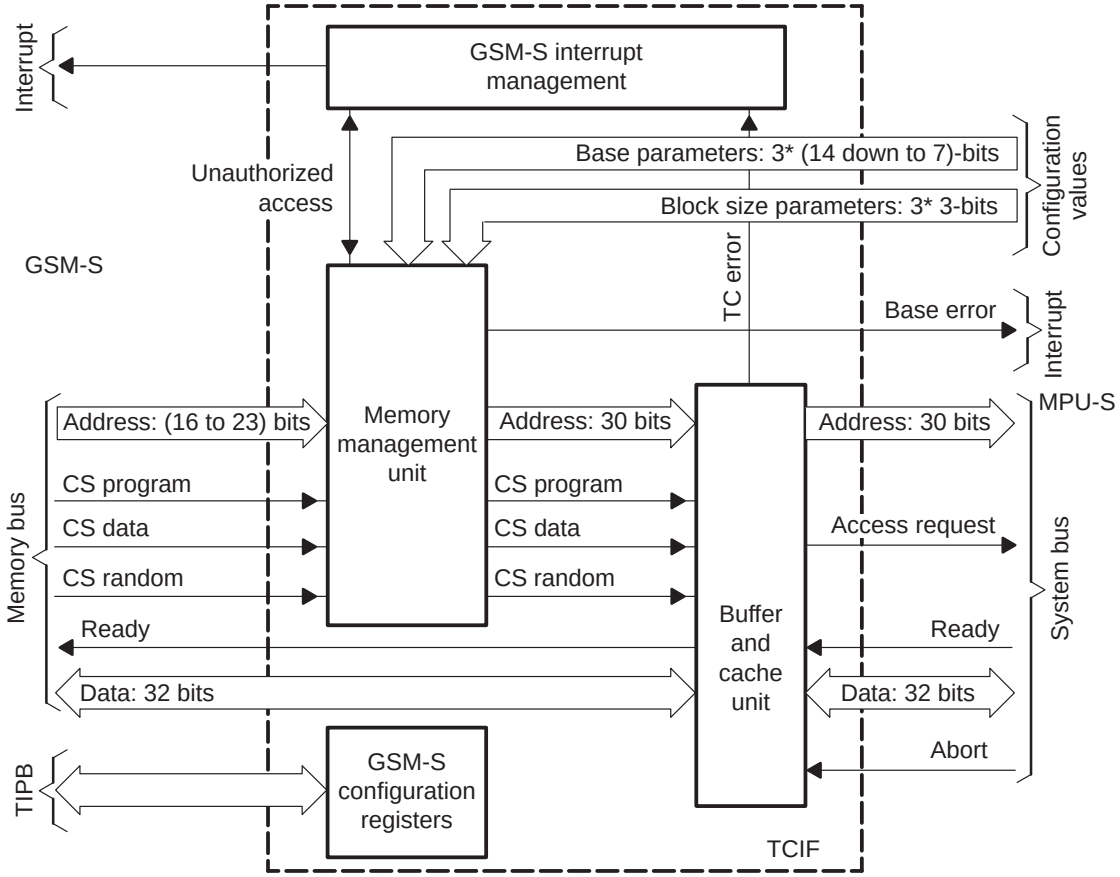
3) The GSM-S may wait for the end of access by polling on the W_BUFFER_FULL bit in the RAND_BUFFER_CTL register.

4.4 TCIF Module Architecture

The architecture of the TCIF module contains two main subblocks (see Figure 4–4):

- The memory management unit is dedicated to address translations and accesses rights control.
- The buffer and cache unit is dedicated to all buffer and cache operations.

Figure 4–4. TCIF Top-Level Diagram



4.4.1 Module Interface Description

4.4.1.1 I/O List and Description

Table 4–26. General-Purpose Pin List

Name	Bits	I/O	Description
CLK_MPU	1	I	MPU-S clock. One of its uses is for the MPU-S TC system bus.
CLK_GSM	1	I	GSM-S clock. One of its uses is for the GSM-S memory Interface bus.

Table 4–26. General-Purpose Pin List (Continued)

Name	Bits	I/O	Description
$\overline{\text{RESET}}$	1	I	Reset signal used to reset the TCIF module. Active low.
$\overline{\text{RESET\_GSM}}$	1	I	Reset signal used to reset the GSM-S. Used as data in TCIF module to enable the $\overline{\text{IRQ\_MPU}}$ interrupt generation. Active low.

Table 4–27. Scan Pin List

Name	Bits	I/O	Description
SCAN_MODE	1	I	Scan mode is set when high.

Table 4–28. BIST Pin List

Name	Bits	I/O	Description
BIST_EN	1	I	BIST enable command. Must be set to 1 to initiate tests.
BIST_CLK	1	I	Clock used for tests
BIST_LV_TM	1	I	
BIST_RST_MEM	1	I	
BIST_GO	1	O	Global result of test
BIST_GO_ID	1	O	
BIST_CMP_STAT	1	O	
BIST_CMP_STAT_ID	1	O	
BIST_ON	1	O	
BIST_DONE	1	O	End of test. Active high.

Table 4–29. MPU-S TC Pin List

Name	Bits	I/O	Description
ADD_MPU_TC	30	O	Address bus
DATA_W_MPU_TC	32	O	Data bus for write cycle
DATA_R_MPU_TC	32	I	Data bus for read cycle
$\overline{\text{REQ\_MPU\_TC}}$	1	O	Access request. Active low.
LOCK_MPU_TC	1	O	Lock request. Active high. Used to perform atomic accesses (when the bus must be reserved for several consecutive bursts).
RNW_MPU_TC	1	O	0: Write access 1: Read access

Table 4–29. MPU-S TC Pin List (Continued)

Name	Bits	I/O	Description
WIDTH_MPU_TC	2	O	Data bus width: 00: 8 bits 01: 16 bits 10: 32 bits
BURST_MPU_TC	3	O	Burst access size: 000: 1 word 011: 4 words 111: 8 words
$\overline{\text{READY\_MPU\_TC}}$	1	I	Ready signal comes from TC. Active low. Must be evaluated on CLK_MPU rising edge
$\overline{\text{ABORT\_MPU\_TC}}$	1	I	Abort signal comes from TC. Active low. Must be evaluated on CLK_MPU rising edge

Table 4–30. GSM-S TIPB Pin List

Name	Bits	I/O	Description
$\overline{\text{STROBE\_GSM\_RHEA}}$	1	I	TIPB bus strobe
CS_GSM_RHEA	5	I	TIPB CS signal
AD_GSM_RHEA	11	I	TIPB address bus
RNW_GSM_RHEA	1	I	0: Write access 1: Read access
DO_GSM_RHEA	16	I	Data bus for write cycle For use on $\overline{\text{STROBE\_GSM\_RHEA}}$ rising edge.
DI_GSM_RHEA	16	O	Data bus for read cycle Must be valid on $\overline{\text{STROBE\_GSM\_RHEA}}$ rising edge when $\overline{\text{READY\_GSM\_RHEA}}$ is set low.
$\overline{\text{READY\_GSM\_RHEA}}$	1	O	TCIF ready to accept or send data. Active low, evaluated on $\overline{\text{STROBE\_GSM\_RHEA}}$ rising edge.
$\overline{\text{OE\_GSM\_RHEA}}$	1	O	Set low when TCIF drives the $\overline{\text{DI\_GSM\_RHEA}}$ and $\overline{\text{READY\_GSM\_RHEA}}$ signals to a valid state.
PERHMAS_GSM_RHEA	2	O	Peripheral access size. Set to 01 (16 bits)

Table 4–31. GSM-S Memory Pin List

Name	Bits	I/O	Description
ADD_GSM_MEM	23	I	Address of access
$\overline{\text{CS\_PGM\_GSM\_MEM}}$	1	I	CS signal used for program memory accesses. Active low.
$\overline{\text{CS\_DATA\_GSM\_MEM}}$	1	I	CS signal used for data memory accesses. Active low.
$\overline{\text{CS\_RAND\_GSM\_MEM}}$	1	I	CS signal used for random memory accesses. Active low.
DIN_GSM_MEM	32	O	Data bus for read cycle
DOUT_GSM_MEM	32	I	Data bus for write cycle
RNW_GSM_MEM	1	I	0: Write access 1: Read access
$\overline{\text{READY\_GSM\_MEM}}$	1	O	Ready signal. Active low.
$\overline{\text{BE\_GSM\_MEM}}$	4	I	For the current transaction: – nbe (3) low → Data (31:24) is used. – nbe (2) low → Data (23:16) is used. – nbe (1) low → Data (15:8) is used. – nbe (0) low → Data (7:0) is used.

4.4.1.2 MMU Configuration Signals

Table 4–32. Output Registers Pin List

Name	Bits	I/O	Description
PGM_BLK_ADDR	14	I	Base address of external program memory accessible by GSM-S through TCIF module.
PGM_BLK_SIZE	3	I	Size of external program memory accessible by GSM-S through TCIF module.
DATA_BLK_ADDR	14	I	Base address of external data memory accessible by GSM-S through TCIF module.
DATA_BLK_SIZE	3	I	Size of external data memory accessible by GSM-S through TCIF module.
RAND_BLK_ADDR	14	I	Base address of external random memory accessible by GSM-S through TCIF module.
RAND_BLK_SIZE	3	I	Size of external random memory accessible by GSM-S through TCIF module.

Table 4–33. Interrupts Pin List

Name	Bits	I/O	Description
$\overline{\text{IRQ\_GSM}}$	1	O	Interrupt line. Falling-edge sensitive. Activated when: ❑ The GSM-S tries to perform an out of bound access (the address value is greater than the block size). ❑ The MPU-S traffic controller aborts a GSM-S access.
$\overline{\text{IRQ\_MPU}}$	1	O	Interrupt line. Low-level sensitive Activated when the GSM-S reset is inactive and the MMU configuration signals are not consistent with each other (block base vs block size for all memory zones).

4.5 Intersystem Communication Register

4.5.1 Module Overview

In the OMAP730 chip, the MPU, and the GSM subsystems may send synchronization flags to each other via two intersystem communication registers.

One of these registers (M_ICR) allows the MPU-S to send flags to the GSM-S, whereas the other one (G_ICR) allows the GSM-S to send flags to the MPU-S.

For a given direction, all flags are set by the source subsystem and reset by the destination subsystem when it so wishes (it is possible to consult flags for both subsystems without resetting them).

To avoid continuous polling on the ICR registers, each subsystem can receive an interrupt when operations that concern it (set or reset) are performed.

Some registers in this module also allow defining configuration values for intersystem communication and resource sharing.

These registers are defined in this module to be able to access them from both MPU-S and GSM-S but are not necessarily used inside it.

An internal dual-port RAM in this module allows transfer of data between the MPU and GSM subsystems without using the TCIF module.

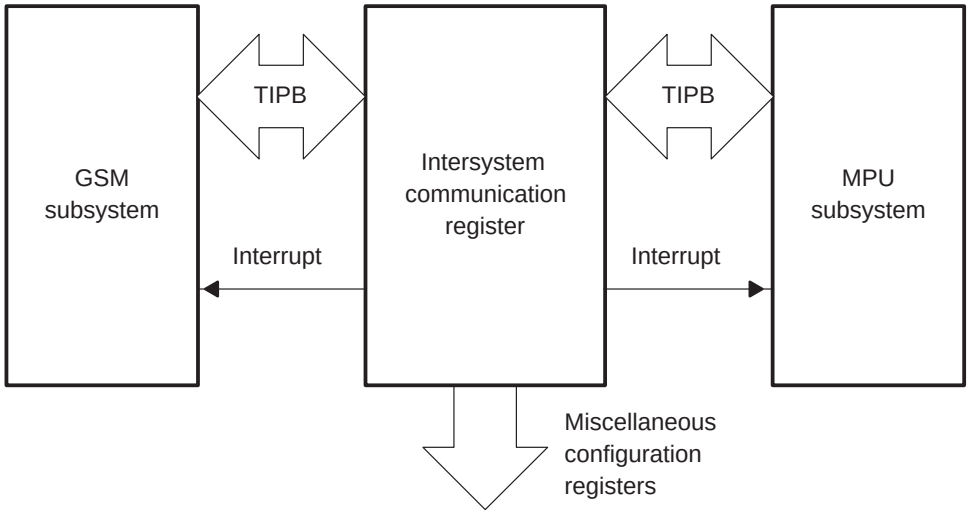
4.5.2 Module Behavior

The intersystem communication register module is a symmetrical interface between the GSM and MPU subsystems which allows them to exchange synchronization flags. It also allows definition of configuration values used by other modules (the advantage of defining them in this module is to make then accessible to both MPU and GSM subsystems).

To do this, the ICR module is connected to both subsystems with the following signals:

- A TIPB bus
- An interrupt line (from ICR to subsystem)

Figure 4–5. Intersystem Communication Register Behavior



4.5.2.1 Flag Management

A typical example of communication between MPU-S and GSM-S:

1) The MPU-S writes one or more flags to the M_ICR register.
2) Following this write operation, the ICR module sends an interrupt to the GSM-S (optional).
3) The GSM-S reads the M_ICR register to find out which flags have been set.
4) When it wants to, the GSM-S resets some flags in the M_ICR register.
5) Following this write operation, the ICR module sends an interrupt to the MPU-S (optional).
6) The MPU-S reads the M_ICR register to find out which flags have been reset.

4.5.2.2 Configuration Registers Management

These registers can be written by only one of the two outside subsystems (refer to register definitions), but can always be read by both. Values that are not used inside the ICR module are exported from it by entity signals.

4.5.2.3 Memory Management

The ICR module contains a dual-port memory RAM that can be accessed by both subsystems to exchange data.

Even if both subsystems have access to all memory locations for read and write operations, no hardware-exclusive access mechanism exists in the ICR module. Consequently, MPU-S and GSM-S software must define a communication protocol that ensures that both subsystems cannot write to the same memory location simultaneously.

4.5.3 Module Specification

4.5.3.1 Module Flags

All ICR flags may be accessed via the TIPB buses with the following chip select values:

- ❑ MPU-S: 6
- ❑ GSM-S: 22

Table 4–34 lists the ICR flag registers. Table 4–35 and Table 4–36 describe the register bits.

Table 4–34. ICR Flags

Register Name	Size	MPU-S		GSM-S	
		Access	Offset	Access	Offset
M_ICR	16 bits	Read/set	000 h	Read/reset	000 h
G_ICR	16 bits	Read/reset	002 h	Read/set	002 h

During access duration:

- ❑ The MPU and GSM subsystems TIPB buses are asynchronous. Consequently, to prevent concurrent access problems, all ICR flags read or write operations from both subsystems are resynchronized with a common clock.
- ❑ For this reason, the register access duration time (read or write operation) should be considered to be, in the worst case of consecutive accesses:
  - ■ Minimum: 1 strobe period
  - ■ Maximum: 3 strobe periods + 2 clock periods

Table 4–35. M_ICR Register

Bit	Name	Function	Reset Value
15:0	FLAGS	Software-dependent	0

The M_ICR register is a collection of MPU-S to GSM-S flags. There is no hardware bit definition for this register: Each bit may flag any event, on the assumption that MPU-S and GSM-S software have the same definitions for these bits.

Table 4–36. G_ICR Register

Bit	Name	Function	Reset Value
15:0	FLAGS	Software-dependent	0

The G_ICR register is a collection of GSM-S to MPU-S flags. There is no hardware bit definition for this register: Each bit may flag any event, on the assumption that MPU-S and GSM-S software have the same definitions for these bits.

4.5.3.2 Module Registers

All ICR registers can be accessed via the TIPB buses with the following chip-select values:

- MPU-S: 6
- GSM-S: 22

Table 4–37 lists the 16-bit ICR registers. Table 4–38 through Table 4–43 describe the register bits.

Table 4–37. ICR Registers

		MPU-S		GSM-S	
Name	**Description**	**Access**	**Offset**	**Access**	**Offset**
M_CTL	MPU-S control	R/W	004h	R	004h
G_CTL	GSM-S control	R	006h	R/W	006h
PM_BA	Program memory base address	R/W	00Ah	R	00Ah
DM_BA	Data memory base address	R/W	00Ch	R	00Ch
RM_BA	Random memory base address	R/W	00Eh	R	00Eh
SSPI_TAS	Syren SPI test and set	R (test and set)/W (reset)	012h	R (test and set)/W (reset)	012h

During access:

The MPU and GSM subsystem TIPB buses are asynchronous. Consequently, to prevent concurrent access problems, all ICR registers read or write operations from both subsystems are resynchronized with a common clock.

For this reason, the register access duration times (read or write operation) are, in the worst case of consecutive accesses:

- Minimum: 1 strobe period
- Maximum: 3 strobe periods + 2 clock periods

Table 4–38. MPU-S Control Register (M_CTL)

Bit	Name	Function	Reset Value
15:12	–	Reserved	–
11	G_ICR_INTEN	When the GSM-S sets a flag to MPU-S: 0: No interrupt is generated. 1: An interrupt is sent to the MPU-S.	0
10	M_ICR_INTEN	When the GSM-S resets a flag set by MPU-S: 0: No interrupt is generated. 1: An interrupt is sent to the MPU-S.	0
9:7	RAND_BLK_SIZE	This parameter specifies the size of the external random memory block accessible by GSM-S: 000: 64K bytes 001: 128K bytes 010: 256K bytes 011: 512K bytes 100: 1M bytes 101: 2M bytes 110: 4M bytes 111: 8M bytes	111
6:4	DATA_BLK_SIZE	This parameter specifies the size of the external data memory block accessible by the GSM-S: 000: 64K bytes 001: 128K bytes 010: 256K bytes 011: 512K bytes 100: 1M bytes 101: 2M bytes 110: 4M bytes 111: 8M bytes	111

Table 4–38. MPU-S Control Register (M_CTL)(Continued)

Bit	Name	Function	Reset Value
3:1	PGM_BLK_SIZE	This parameter specifies the size of the external program memory block accessible by the GSM-S: 000: 64K bytes 001: 128K bytes 010: 256K bytes 011: 512K bytes 100: 1M bytes 101: 2M bytes 110: 4M bytes 111: 8M bytes	111
0	GSM_RST	0: The GSM subsystem is reset 1: The GSM subsystem is in normal operation mode	0

The M_CTL register controls the MPU-S behavior.

Table 4–39.GSM-S Control Register (G_CTL)

Bit	Name	Function	Reset Value
15:3	–	Reserved	–
2	G_ICR_INTEN	When the MPU-S resets a flag set by GSM-S: 0: No interrupt is generated. 1: An interrupt is sent to the GSM-S.	0
1	M_ICR_INTEN	When the MPU-S sets a flag to GSM-S: 0: No interrupt is generated. 1: An interrupt is sent to the GSM-S.	0
0	–	Reserved	–

The G_CTL register controls the behavior of the GSM-S.

PM_BA (Program Memory Base Address)

The MPU-S sets the program memory base address register (PM_BA) to specify the 16 MSBs of the GSM-S program memory address in the external memory map. Depending on the block size of the program memory (refer to M_CTL register definition), the PM_BA uses 7 bits when the block size is 8M bytes, to 14 bits when the block size is 64K bytes.

64K-Byte Program Memory Block Size Example:

GSM-S addr.					0	0	0	0	0	0	0	A15	A14	A13		A2	A1	A0
Base addr.	B13	B12	…	B7	B6	B5	B4	B3	B2	B1	B0							
TC addr.	B13	B12	…	B7	B6	B5	B4	B3	B2	B1	B0	A15	A14	A13		A2	A1	A0
	29	28	…	23	22	21	20	19	18	17	16	15	14	13	…	2	1	0

8M-Byte Program Memory Block Size Example:

GSM-S addr.					A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	..	A2	A1	A0
Base addr.	B13	B12	..	B7	0	0	0	0	0	0	0							
TC addr.	B13	B12	..	B7	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	..	A2	A1	A0
	29	28	…	23	22	21	20	19	18	17	16	15	14	13	..	2	1	0

Table 4–40. Program Memory Base Address Register (PM_BA)

Bit	Name	Function	Reset Value
15:14	–	Reserved – Must be set to 0	–
13:0	PM_ADDR	Depending on the program memory block size, the following bits are used for memory base address: ❑ 64K Bits 13:0 Base address ❑ 128K Bits 13:1 Base address; Bit 0 Must be set to 0 ❑ 256K Bits 13:2 Base address; Bits 1:0 Must be set to 0 ❑ 512K Bits 13:3 Base address; Bits 2:0 Must be set to 0 ❑ 1M Bits 13:4 Base address; Bits 3:0 Must be set to 0 ❑ 2M Bits 13:5 Base address; Bits 4:0 Must be set to 0 ❑ 4M Bits 13:6 Base address; Bits 5:0 Must be set to 0 ❑ 8M Bits 13:7 Base address; Bits 6:0 Must be set to 0	0

DM_BA (Data Memory Base Address)

The MPU-S sets the data memory base address register (DM_BA) to specify the 16 MSBs of the GSM-S data memory address in the external memory map. Depending on the block size of the data memory (refer to M_CTL register definition), the DM_BA uses 7 bits when the block size is 8M bytes, to 14 bits when the block size is 64K bytes.

64K-Byte Data Memory Block Size Example:

GSM-S addr.					0	0	0	0	0	0	0	A15	A14	A13	…	A2	A1	A0
Base addr.	B13	B12	…	B7	B6	B5	B4	B3	B2	B1	B0							
TC addr.	B13	B12	…	B7	B6	B5	B4	B3	B2	B1	B0	A15	A14	A13	…	A2	A1	A0
	29	28	…	23	22	21	20	19	18	17	16	15	14	13	…	2	1	0

8M-Byte Data Memory Block Size Example:

GSM-S addr.					A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	..	A2	A1	A0
Base addr.	B13	B12	..	B7	0	0	0	0	0	0	0				..			
TC addr.	B13	B12	..	B7	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	..	A2	A1	A0
	29	28	…	23	22	21	20	19	18	17	16	15	14	13	..	2	1	0

Table 4–41. Data Memory Base Address Register (DM_BA)

Bit	Name	Function	Reset Value
15:14	–	Reserved. Must be set to 0.	–
13:0	DM_ADDR	Depending on the data memory block size, the following bits are used for memory base address: ❑ 64K Bits 13: 0 Base address ❑ 128K Bits 13:1 Base address Bit 0 Must be set to 0 ❑ 256K Bits 13: 2 Base address Bits 1:0 Must be set to 0 ❑ 512K Bits 13: 3 Base address Bits 2:0 Must be set to 0 ❑ 1M Bits 13: 4 Base address Bits 3:0 Must be set to 0 ❑ 2M Bits 13: 5 Base address Bits 4:0 Must be set to 0 ❑ 4M Bits 13: 6 Base address Bits 5:0 Must be set to 0 ❑ 8M Bits 13: 7 Base address Bits 6:0 Must be set to 0	0

RM_BA (Random Memory Base Address)

The MPU-S sets the random memory base address register (RM_BA) to specify the 16 MSBs of the GSM-S random memory address in the external memory map. Depending on the block size of the random memory (refer to M_CTL register definition), the RM_BA uses 7 bits when the block size is 8M bytes, to 14 bits when the block size is 64K bytes.

64K-Byte Random Memory Block Size Example:

GSM-S addr.					0	0	0	0	0	0	0	A15	A14	A13		A2	A1	A0
Base addr.	B13	B12		B7	B6	B5	B4	B3	B2	B1	B0							
TC addr.	B13	B12		B7	B6	B5	B4	B3	B2	B1	B0	A15	A14	A13		A2	A1	A0
	29	28	…	23	22	21	20	19	18	17	16	15	14	13	…	2	1	0

8M-Byte Random Memory Block Size Example:

GSM-S addr.					A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	..	A2	A1	A0
Base addr.	B13	B12	..	B7	0	0	0	0	0	0	0							
TC addr.	B13	B12	..	B7	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	..	A2	A1	A0
	29	28	..	23	22	21	20	19	18	17	16	15	14	13	..	2	1	0

Table 4–42. Random Memory Base Address Register (RM_BA)

Bit	Name	Function	Reset Value
15:14	–	Reserved – Must be set to 0	–
13:0	RM_ADDR	Depending on the random memory block size, the following bits are used for memory-base address: ❑ 64K Bits 13: 0 Base address ❑ 128K Bits 13:1 Base address Bit 0 Must be set to 0 ❑ 256K Bits 13: 2 Base address bits 1:0 Must be set to 0 ❑ 512K Bits 13: 3 Base address Bits 2:0 Must be set to 0 ❑ 1M Bits 13: 4 Base address Bits 3:0 Must be set to 0 ❑ 2M Bits 13: 5 Base address Bits 4:0 Must be set to 0 ❑ 4M Bits 13: 6 Base address Bits 5:0 Must be set to 0 ❑ 8M Bits 13: 7 Base address Bits 6:0 Must be set to 0	0

Table 4–43. TWL3016 SPI Test and Set Register (SSPI_TAS)

Bit	Name	Function	Reset Value
15:1	–	Unused	–
0	SSPI_TAS	If the SSPI interface is not already locked, the value read is 1 and the processor has SSPI exclusive access granted; otherwise, the value read is 0.	–

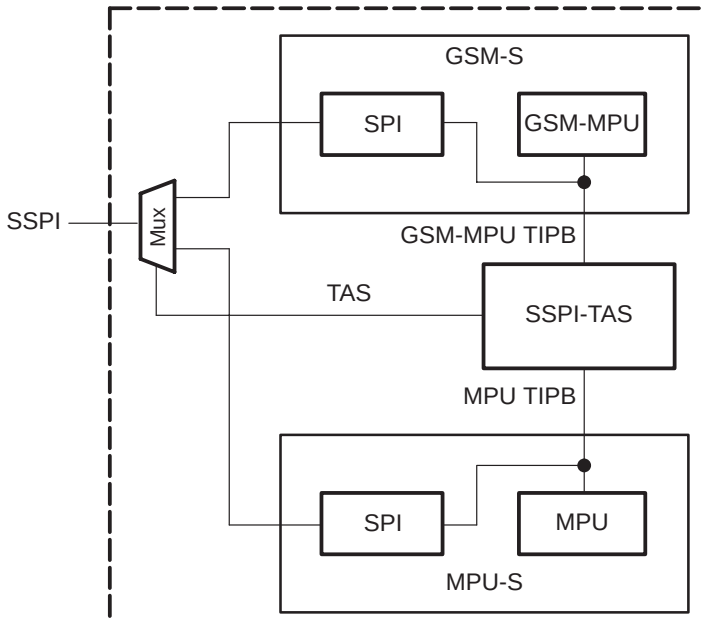
Notes: 1) If both subsystems request SSPI usage at the same time, the GSM-S has SSPI access granted.

2) After reset, GSM_S has access to SSPI. However, this access is not exclusive (until GSM_S reads SSPI_TAS register). This means MPU-S could possibly have exclusive access granted while SSPI is used by GSM-S.

3) When a subsystem reads the SSPI_TAS register and has exclusive access granted, a following read by the same subsystem returns a 0 value until it releases SSPI by writing to SSPI_TAS register.

The SSPI is a physical SPI interface shared between MPU-S and GSM-S SPI controllers. SSPI_TAS is a special register accessible by both MPU-S and GSM-S allowing to test the state of the SSPI and to give exclusive access to the interface with only one instruction. A write to this register releases the SSPI.

Figure 4–6. SSPI Block Diagram



4.5.3.3 TAS Functional Description

The SSPI_TAS register is accessible to the GSM-S and the MPU-S. When the GSM-S or the MPU-S needs to access the SSPI, it proceeds as follows:

1) The CPU reads the SSPI_TAS register:
   a) If the other processor is not currently using the SSPI, the value read is 1 and the processor gets SSPI access. The TAS signal then uses the mux to select the requested SPI controller.
   b) If the other processor is currently using the SSPI, the value returned is 0 and the processor gets SSPI access. In this case, the CPU must poll the SSPI_TAS register until it is 1.
2) Once the processor has finished accessing the SSPI, it must release the SSPI_TAS mechanism by either writing a 1 or a 0 into the TAS register.

4.5.3.4 Module RAM

A 640-byte dual-port RAM allows exchange of data between the MPU and GSM subsystems via their respective TIPB buses.

This memory can be used instead of external data RAM to store:

- ❏ All memory command registers M_CMD/G_CMD (2×64 bytes)
- ❏ At least one complete set of buffers M_BUFF/G_BUFF ($2 \times 2 \times 128$ bytes for the worst-case audio data transfers)
- ❏ Miscellaneous variables

Note:

There is no hardware-exclusive access management on this shared memory. Consequently, the GSM and MPU software must use a communication protocol that ensures that the two subsystems cannot write to the same memory location at the same time.

The RAM can be accessed via the TIPB buses with the following chip-select values:

- ❏ MPU-S: 6
- ❏ GSM-S: 22

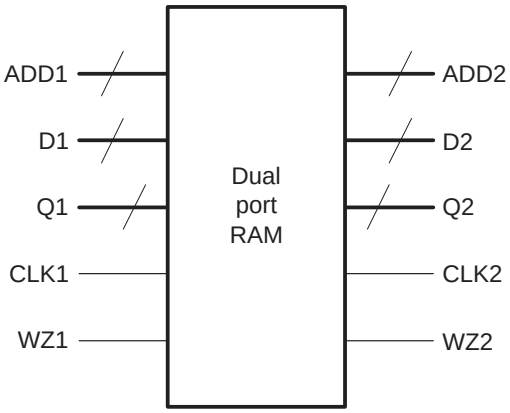
Table 4–44. ICR RAM [RAM Version]

RAM Name	Size	MPU-S		GSM-S	
		Access	Offset	Access	Offset
DUAL_PORT_RAM	16 bits	R/W	400 h … 67F h	R/W	400 h … 67F h

The RAM used is a TI MN series one. Its main characteristics are:

- ❏ Totally separate access ports
- ❏ Synchronous read and write operations

Figure 4–7. Dual Port RAM (RAM Version)

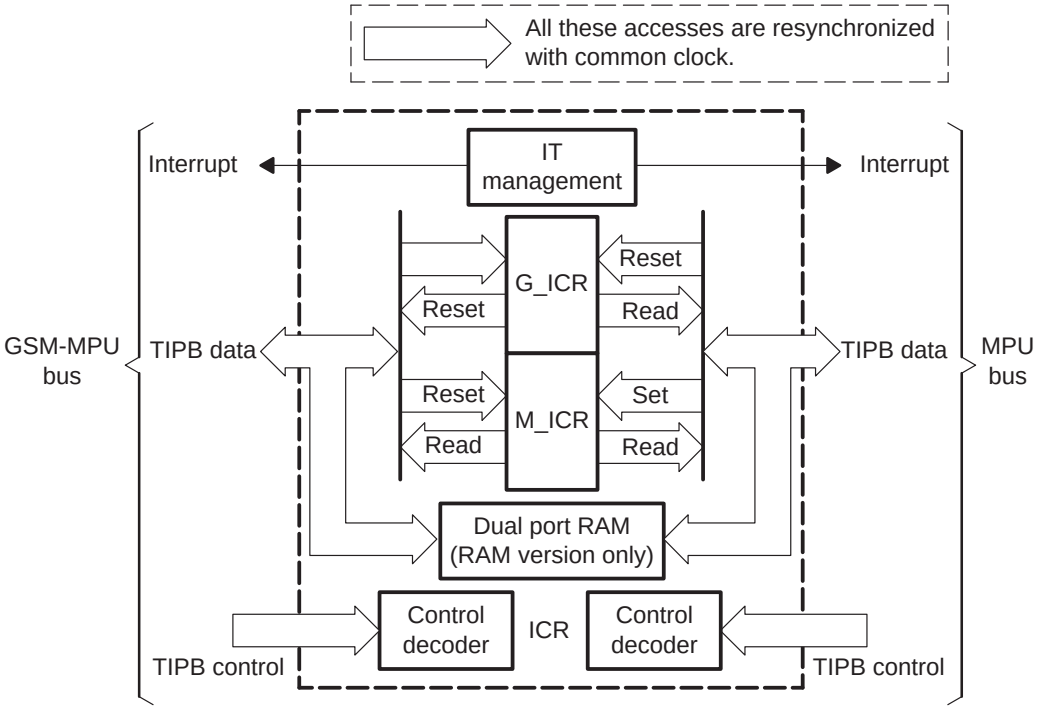


Access Duration:

- Read operation: Two TIPB cycles (one wait state)
- Write operation: One TIPB cycle (no wait state)

4.5.4 Top-Level Block Diagram

Figure 4–8. Intersystem Communication Register Overview



Note: Access Duration

The MPU and GSM subsystems TIPB buses are asynchronous. Consequently, to prevent concurrent access problems, all ICR registers' read or write operations from both subsystems are resynchronized with a common clock.

4.5.5 I/O Timing Diagram

4.5.5.1 TIPB Signals

All TIPB signals are compliant with the TIPB bus definition.

4.5.5.2 Interrupt Signals

The ICR is a symmetrical module. Thus, the falling-edge-sensitive interrupt signals to GSM and MPU subsystems generated when accessing the G_ICR and M_ICR registers have the same timing diagrams.

In the following example, accesses to the G_ICR register (flag is set from GSM-S to MPU-S) is assumed.

Figure 4–9. Intersystem Communication Register Overview

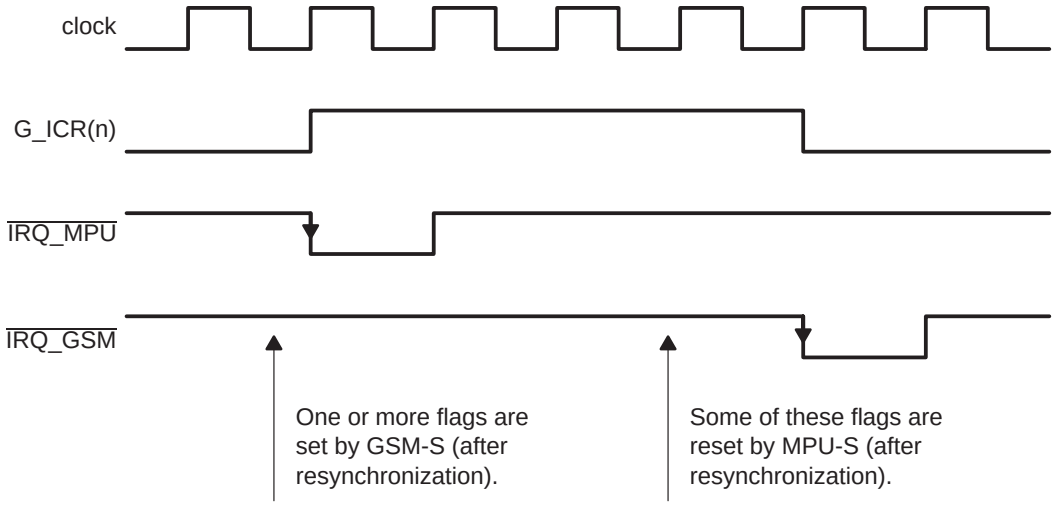


Figure 4–10. Interrupts Generation for G_ICR Register

Notes:
1) With the above definition for interrupts, problems may arise if one subsystem could perform two consecutive write accesses in two consecutive *clock* periods (to set or reset flags). But such accesses cannot occur due to the minimum duration of register-write access resynchronization.
2) Each flag inside M_ICR or G_ICR registers is independently managed. Consequently, MPU-S and GSM-S may access together (at the same *clock* clock rising edge after resynchronization) the same register to set and reset flags.
3) The following operations are not functional and should never occur:
   If only one of the subsystems (MPU-S or GSM-S) performs a write access that should set (or reset) a flag while it is already set (or reset):
   - The flag remains unchanged.
   - No interrupts are generated.

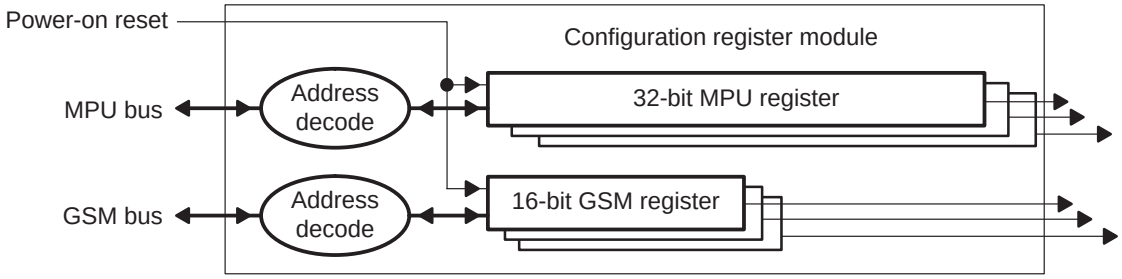
   In the case of a write accesses by MPU-S and GSM-S that simultaneously sets and resets a single flag at the same time (regardless of old flag value):
   - The flag is set,
   - An interrupt is generated to both MPU-S and GSM-S (if enabled).

4.6 OMAP730 Configuration Module

The OMAP730 configuration module allows software control of the various static modes supported by the device. This module is the primary control point for the following areas of the OMAP730 device:

- Functional I/O multiplexing
- Debug and observation I/O multiplexing
- Pullup/pulldown enable
- Interface voltage selection
- Static module configuration
- Clock multiplexing control
- Secure mode configuration

Figure 4–11. Configuration Register Module



4.6.1 Configuration Register Capabilities

The OMAP730 configuration module is functionally very simple. The module is primarily a bank of registers (32 bits on MPU side, 16 bits on GSM side) that can be accessed by software. There is no resynchronization on these registers, so the module is for use as static configuration (no changes after the release of peripheral reset). Considering that registers are programmed before release of the peripheral reset, the configuration registers are only power-on reset.

The access rights to these registers depends on the functionality: Read only, read/write, read/write once, or conditional rights, depending on bit/device type. All information concerning each of these registers can be found in the configuration register description section.

This bank of registers can be broken down into 10 primary sections:

- Functional multiplexing registers (dual modes) PCONF_D(2:0)
- Pullup/pulldown enable registers PCONF_PE
- Voltage control registers PCONF_VOLTAGE
- Test and debug registers PCONF_OBS
- Module configuration registers, no naming rules
- General-purpose registers (no predefined size), no naming rules
- Gating and Inhibiting registers, no naming rules
- Chip identifier registers, no naming rules
- Security control register, no naming rules
- Status, no naming rules

4.6.2 Configuring Pin Multiplexing and Pullups/Pulldowns

4.6.2.1 Programming Pin Multiplexing

Each pin that has a multiplexing function is assigned a three-bit register field in the register set PCONF_D(2:0) thus creating eight possible multiplexing options per pin (modes 0 to 6 enabled on OMAP730).

Reset values corresponding to the default configuration of the device: To avoid electrical contention, GPIOs configured as inputs are selected if not forbidden by system-level considerations (for example, EMIF is mandatory for boot, then selected by default).

The PCONF_ PE bit enables pullup or pulldown (depending on pad) on inputs. If PCONF_PE = 1, pullup/pulldown is active; else, PCONF_PE = 0 and pullup/pulldown is disabled. By default, pull is enabled to avoid floating nodes at reset.

When configuring the pinout of the device, follow this process:

1) Determine the desired values for each PCONF_D, PCONF_PE, and other global-purpose configuration registers.

2) Program the desired values by writing to the appropriate register (If not equal to reset value).

3) Reset on peripheral can be released.

This procedure enables you to make all multiplex configuration settings using a series of register writes.

Users must consult Appendix C.3, *Pin Multiplexing,* to determine the exact pin available on the device for pin functional multiplexing. This reference must be consulted to find out if a pullup/pulldown is actually implemented on a pin-by-pin basis. Not all pins have a pullup/pulldown macro implemented.

Table 4–45. Functional Multiplexing Modes

PCONF_D Three-Bit Register Value	Functional Multiplexing Mode (See Appendix C.3, *Pin Multiplexing)*
000	Primary function/functional multiplexing mode 0
001	Functional multiplexing mode 1
010	Functional multiplexing mode 2
011	Functional multiplexing mode 3
100	Functional multiplexing mode 4
101	Functional multiplexing mode 5
110	Functional multiplexing mode 6

When functionally possible, several pads are controlled by the same PCONF_D referring to a common function. These groups prevent redundancy between fields and inconsistent configurations, and decrease the number of accesses necessary for a configuration sequence.

For a given interface, the value of the PCONF_D register can vary from pin to pin. For example GSM_GPIO 4 and 2 can be accessed in mode 2, while GSM_GPIO 1 can be accessed in modes 1, 2, and 4.

The same module interface can be muxed to several pads. Hardware protection is implemented to avoid electrical contention. This feature can be useful when bad programming of a configuration register occurs and affects two or more input pads on the same internal module input.

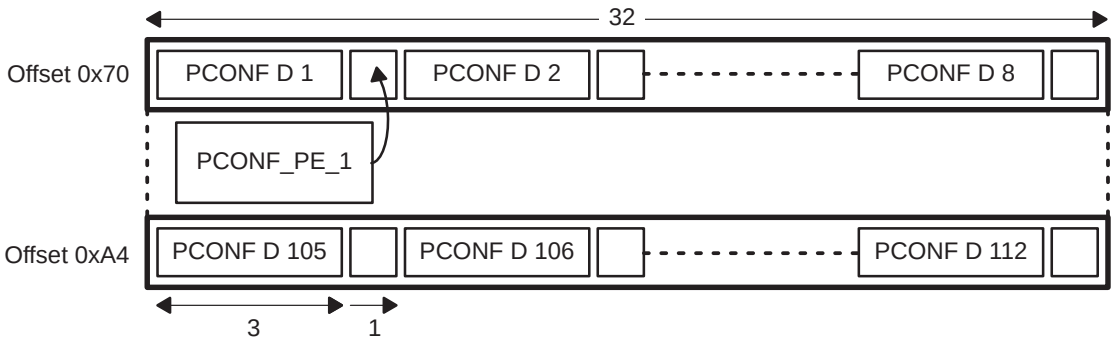
Programming a mux mode for a pin that is not defined in Appendix C.3, *Pin Multiplexing,* leads to undefined operation of the pin and must be strictly avoided.

4.6.2.2 Register Organization

The register length is 32 bits and each pad needs four bits to be controlled properly (three for PCONF_D and one for PCONF_PE). Consequently, fields are grouped by 8 in one register. A write to one address configures 8 dual modes:

- Eight different PCONF_Ds, each one enables one mode out of seven on at least one pin.
- Eight different PCONF_PEs enable or disable pullup/pulldown on at least one pin.

Figure 4–12. Configuration Register Format



Note that the name of PCONF<NAME> is not a number but a reference to the primary function.

4.6.2.3 OMAP710 Compatibility

Considering the new features embedded in OMAP730, seven modes are necessary to enable all the configurations required by different products with different requirements when using the compact OMAP730 package.

The module organization of the OMAP730 configuration registers is completely different from that of the OMAP710 module. There is no way to maintain software compatibility between the two devices considering the major changes on OMAP730 pinout and the fact that pullups/pulldowns are now controllable by software.

4.6.2.4 Customer Support

A dedicated software tool is available to users to aid in the generation of correct write sequences in the OMAP730 configuration register. This OMAP_GUI tool provides a friendly interface to developers.

4.6.3 Configuring the USB

There are three different ways to use the USB, each requiring specific configuration register settings. This is the topic of the following sections.

4.6.3.1 Interfacing the USB With an External OTG Transceiver

This solution requires:

- ❏ The use of an internal transceiver:

  PERSEUS2_MODE1(26): USB_TRANSCEIVER_SEL = 1

- ❏ Selection of USB pads in mode 0:

  PERSEUS2_IO_CONF2 (23:21): PCONF_D_DM = 0x0

- ❏ Optional configuration (depending on application):

  PERSEUS2_IO_CONF2 (27:25): PCONF_D_PU_EN = 0x0

  PERSEUS2_IO_CONF2 (31:29): PCONF_D_VBUSI = 0x0

4.6.3.2 Interfacing USB With an OTG Transceiver in Bidirectional Mode

This solution requires:

- ❏ No internal transceiver:

  PERSEUS2_MODE1(26): USB_TRANSCEIVER_SEL = 0

- ❏ Selection of USB pads in mode 3:

  PERSEUS2_IO_CONF2 (23:21): PCONF_D_DM = 0x3

  PERSEUS2_IO_CONF2 (27:25): PCONF_D_PU_EN = 0x3

  PERSEUS2_IO_CONF2 (31:29): PCONF_D_VBUSI = 0x3

- ❏ Optional configuration (depending on application):

  PERSEUS2_IO_CONF9 (31:29): PCONF_D_MPU_NIRQ = 0x1

4.6.3.3 Interfacing USB With a Non-OTG Transceiver in Unidirectional Mode

This solution requires:

- ❏ No internal transceiver:

  PERSEUS2_MODE1(26): USB_TRANSCEIVER_SEL = 0

- ❏ Selection of USB pads in mode 3:

  PERSEUS2_IO_CONF2 (27:25): PCONF_D_PU_EN = 0x3

  PERSEUS2_IO_CONF2 (31:29): PCONF_D_VBUSI = 0x3

  PERSEUS2_IO_CONF3 (3:1): PCONF_D_ MCLK_OUT = 0x3

  PERSEUS2_IO_CONF3 (7:5): PCONF_D_ CRESET = 0x3

- ❏ Selection of USB pads in mode 4:

PERSEUS2_IO_CONF3 (23:21): PCONF_D_ DM = 0x4

- ❏ Selection of USB pads in mode 5:

  PERSEUS2_IO_CONF5 (3:1): PCONF_D_ I2C_SDA = 0x5

  PERSEUS2_IO_CONF5 (7:5): PCONF_D_ I2C_SCK = 0x5

- ❏ Optional configuration (depending on application):

  PERSEUS2_IO_CONF9 (31:29): PCONF_D_MPU_NIRQ = 0x1

4.6.4 Programming Security Registers

Pay close attention to the register description (configuration register description section) when programming security registers. For security purposes, these registers' write access depends on device type and OMAP secure bit.

Be aware of:

- ❏ R/W1 in R/W column means this is a one-time programmable register.
- ❏ Bits SECCTRL(11:0) can be write-only if OMAP secure bit = 1.

4.6.5 Configuring Supply Voltage

Three power domains in OMAP730 have special pins supporting dual voltage: VDDSHV1 (SDRAM interface), VDDSHV3 (flash interface), and VDDSHV7 (USB interface).

4.6.5.1 Setting VDDSHV1 (SDRAM Interface)

- ❏ Select high voltage (default voltage): 2.5 V min, 2.75 V nom, 3.0 V max

  PERSEUS2_MODE2 (12): PCONF_VOLTAGE_SDRAM = 1

- ❏ Select low voltage: 1,65 V min, 1,8 V nom, 1,95 max

  PERSEUS2_MODE2 (12): PCONF_VOLTAGE_SDRAM = 0

4.6.5.2 Setting VDDSHV3 (Flash Interface)

- Select high voltage (default voltage): 2.5 V min, 2.75 V nom, 3.0 V max

  PERSEUS2_MODE2 (13): PCONF_VOLTAGE_FLASH = 1

- Select low voltage: 1,65 V min, 1,8 V nom, 1,95 max

  PERSEUS2_MODE2 (13): PCONF_VOLTAGE_FLASH = 0

4.6.5.3 Setting VDDSHV7 (USB Interface)

The right voltage is automatically selected when multiplexing mode is programmed.

4.7 OMAP730 Configuration Registers

This section presents the OMAP730 configuration registers. Table 4–47 through Table 4–91 describe the individual register bits.

Table 4–46. OMAP730 Configuration Registers

Name	Description	Size (bits)	R/W	Offset
PERSEUS2_MPU_DEV_ID	Device identification on MPU side	32	R	0x00
PERSEUS2_GSM_DEV_ID0	Device identification on number 0	16	R	0x00
PERSEUS2_GSM_DEV_ID1	Device identification on number 1	16	R	0x02
DSP_CONF	Software compatibility with EDGE	16	R/W	0x04
PERSEUS2_MPU_DIE_ID0	OMAP730 die identification 0	32	R	0x08
GSM_ASIC_CONF	Compatibility with TBB2100	16	R/W	0x08
PERSEUS2_MPU_DIE_ID1	OMAP730 die identification 1	32	R	0x0C
PERSEUS2_MODE1	OMAP730 mode configuration 1	32	R/W	0x10
PERSEUS2_GSM_DIE_ID0	OMAP730 die identification 0	16	R	0x10
PERSEUS2_GSM_DIE_ID1	OMAP730 die identification 1	16	R	0x12
PERSEUS2_MODE2	OMAP730 mode configuration 2	32	R/W	0x14
PERSEUS2_GSM_DIE_ID2	OMAP730 die identification 2	16	R	0x14
PERSEUS2_GSM_DIE_ID3	OMAP730 die identification 3	16	R	0x16
PERSEUS2_ ANALOG_CELLS_CONF	OMAP730 analog cells configuration	32	R/W	0x18
SECCTRL	Secure control		R/W	0x1C
SPARE1	ECO spare 1	32	R/W	0x20
SPARE2	ECO spare 2	32	R/W	0x24
GSM_PBG_IRQ	Edge—GSM domain	16	R/W	0x28
DMA_REQ_CONF	DMA mode configuration	32	R/W	0x30
PE_CONF_NO_DUAL	Edge—MPU domain	32	R/W	0x60
PERSEUS2_IO_CONF0	OMAP730 input/output configuration 0	32	R/W	0x70
PERSEUS2_IO_CONF1	OMAP730 input/output configuration 1	32	R/W	0x74
PERSEUS2_IO_CONF2	OMAP730 input/output configuration 2	32	R/W	0x78
PERSEUS2_IO_CONF3	OMAP730 input/output configuration 3	32	R/W	0x7C
PERSEUS2_IO_CONF4	OMAP730 input/output configuration 4	32	R/W	0x80
PERSEUS2_IO_CONF5	OMAP730 input/output configuration 5	32	R/W	0x84
PERSEUS2_IO_CONF6	OMAP730 input/output configuration 6	32	R/W	0x88

Table 4–46. OMAP730 Configuration Registers (Continued)

Name	Description	Size (bits)	R/W	Offset
PERSEUS2_IO_CONF7	OMAP730 input/output configuration 7	32	R/W	0x8C
PERSEUS2_IO_CONF8	OMAP730 input/output configuration 8	32	R/W	0x90
PERSEUS2_IO_CONF9	OMAP730 input/output configuration 9	32	R/W	0x94
PERSEUS2_IO_CONF10	OMAP730 input/output configuration 10	32	R/W	0x98
PERSEUS2_IO_CONF11	OMAP730 input/output configuration 11	32	R/W	0x9C
PERSEUS2_IO_CONF12	OMAP730 input/output configuration 12	32	R/W	0xA0
PERSEUS2_IO_CONF13	OMAP730 input/output configuration 10	32	R/W	0xA4
PERSEUS_PCC_CONF_REG	48-MHz input control	32	R/W	0xB4
BIST_STATUS_INTERNAL	BIST fail go	32	R	0xB8
BIST_CONTROL	BIST settings control	32	R/W	0xC0
BOOT_ROM_REG	Boot procedure	32	R/W	0xC4
PRODUCTION_ID_REG	Secure chip	32	R/W	0xC8
BIST_SECROM_ SIGNATURE1_INTERNAL	Secure ROM signature 1	32	R/W	0xD0
BIST_SECROM_ SIGNATURE2_INTERNAL	Secure ROM signature 2	32	R/W	0xD4
BIST_CONTROL_2	BIST settings control 2	32	R/W	0xD8
DEBUG1	Debug signal selection 1	32	R/W	0xE0
DEBUG2	Debug signal selection 2	32	R/W	0xE4
DEBUG_DMA_IRQ	DMA and IRQ selection	32	R/W	0xE8

Table 4–47. Device Identification on MPU Side Register (PERSEUS2_MPU_DEV_ID)

Bit	Name	Function	Reset Value
31:21	MANUFACTURER_IDENTITY	Texas Instruments IEEE ID = 0x17	0x17
20	UNUSED	Always 1	0x1
19:16	MPU_VERSION	Chip version number identical to GSM_VERSION	0x1
15:0	MPU_PART_NUMBER	Device part number in JTAG format Identical to GSM_PART_NUMBER	0xB55F

PERSEUS2_MPU_DEV_ID is for device identification on the MPU side register bits.

Table 4–48. Device Identification on Number 0 Register (PERSEUS2_GSM_DEV_ID0)

Bit	Name	Function	Reset Value
15:0	GSM_PART_NUMBER	Device part number in JTAG format Identical to MPU_PART_NUMBER	Unknown

PERSEUS2_GSM_DEV_ID0 is for device identification on the GSM side register bits.

Table 4–49. Device Identification on Number 1 Register (PERSEUS2_GSM_DEV_ID1)

Bit	Name	Function	Reset Value
15:4	RESERVED	Reserved	0X0
3:0	VERSION_GSM	Chip version number Identical to VERSION_MPU	0x1

PERSEUS2_GSM_DEV_ID1 is for device identification on the GSM side register bits.

Table 4–50. Software Compatibility With EDGE Register (DSP_CONF)

Bit	Name	Function	Reset Value
15:8	UNUSED	Unused	0x0
7	TPU_FRAME_INTERRUPT	0: TPU frame irq disable 1: TPU frame irq enable	0x0
6:0	UNUSED	Unused	0x0

Table 4–51. OMAP730 Die Identification Register (PERSEUS2_MPU_DIE_ID0)

Bit	Name	Function	Reset Value
31:0	PERSEUS2_MPU_DIE_ID0	OMAP730 DIE_ID0	Unknown

Table 4–52. Compatibility With TBB2100 Register (GSM_ASIC_CONF)

Bit	Name	Function	Reset Value
15	UNUSED	Unused	0x0
14	SPI_CLK_POL	0: SPI configuration clock RX (falling edge) 1: SPI configuration clock RX (rising edge)	0x0
13	RIF_CLK_POL	0: RIF configuration clock RX (falling edge) 1: RIF configuration clock RX (rising edge)	0x0
12:0	UNUSED	Unused	0x0

Table 4–53. OMAP730 Die Identification Number 1 Register (PERSEUS2_MPU_DIE_ID1)

Bit	Name	Function	Reset Value
31:0	PERSEUS2_MPU_DIE_ID1	OMAP730 die identification number 1	Unknown

Table 4–54. OMAP730 Mode Configuration Register (PERSEUS2_MODE1)

Bit	Name	Function	Reset Value
31:30	LED_OUT_GSM	00: LPG1/LPG2 disabled 01: LPG1 enabled on NPER_GSM_MLPG1 pin 10: LPG2 enabled on MPU_NBOOT_MLPG2 pin 11: LPG1 and LPG2 enabled	0X00
29:28	LED_OUT_MPU	00: LPG1/LPG2 disabled 01: LPG1 enabled on NPER_GSM_MLPG1 pin 10: LPG2 enabled on MPU_NBOOT_MLPG2 pin 11: LPG1 and LPG2 enabled	0X00
27	MPU_SYREN_SPI_CLK_POL	0: MPU_SPI configuration clock RX (falling edge) 1: MPU_SPI configuration clock RX (rising edge)	0X0
26	USB_TRANSCEIVER_SEL	0: External USB transceiver 1: On-chip USB transceiver	0X0
25	USB_VBUS_CTRL	When USB_VBUS_MODE = 1: drive VBUS signal of USB module 0: USB BUS is not plugged. 1: USB BUS is plugged.	0X0
24:23	USB_VBUS_MODE	00: Dual-mode pin USB_VBUS directly drives the power line of USB module 01: Register USB_VBUS_CTRL drives the power line 10: Power line driven by tactical cell UIS480	0X1
22:19	EXT_IO_CTRL	0: Control output register 1: Control input buffer EXT_IO_CTRL(5)→EXT_IO_3 EXT_IO_CTRL(4)→EXT_IO_2 EXT_IO_CTRL(3)→EXT_IO_1 EXT_IO_CTRL(2)→EXT_IO_0	0X0
18	EXT_IO_CS_SEL	0: CS2 1: CS3	0X0
17	EXT_IO_EN	1: Enable: CS3 (or CS2) disabled when Add = FFFFF8 – FFFFFE and EXT_IO enable 0: Disable EXT_IO EXT_IO_3 = Add 0xFFFFFE EXT_IO_2 = Add 0xFFFFFC EXT_IO_1 = Add 0xFFFFFA EXT_IO_0 = Add 0xFFFFF8	0x0
16	PIARMTDBGEN	Sets MPUTDMI TDBGEN to functional mode	0x0
15	USB_TRANSCEIVER_SPEED	To internal transceiver: 0: Low speed 1: High speed	0x0

Table 4–54. OMAP730 Mode Configuration Register (PERSEUS2_MODE1) (Continued)

Bit	Name	Function	Reset Value
14	MPU_SPI1_CLK_POL	0: MPU_SPI configuration clock RX (falling edge) 1: MPU_SPI configuration clock RX (rising edge)	0x0
13	TC_LRU_SEL	0: TC uses fixed-priority algorithm 1: TC uses least-recently used (LRU) mode priority algorithm	0x1
12	NANDF_CTRL_ENA	0: NAND-flash controller disable 1: NAND-flash controller enable	0x0
11:10	USB_PWRDN	D+ D– pulldown control 00:Pulldown enable on D+ D– 01:PWRDN1 active, pulldown on D+ 10:PWRDN2 active, pulldown on D– 11:Pulldown disable on D+ D–	0x0
9	EMULATION_CONFIG	0: MPU/GSM has separate signals /(MPU emu signals on nemu0/1 pins DSP/GSM-MPU emu signals on nemu0/1 dvpt) 1: Shared MPU-GSM EMU signals	0x0
8	PIARMTDBGRQ	Setting MPUTDMI TDBGRQ in functional mode.	0x0
7	ON_OFF	0: Signal ON_OFF (GSM power management) from pin ON_OFF 1: Signal ON_OFF (GSM power management) from ULPD	0x0
6	MCBSP2_CLKS_SEL	0: Internal oscillator from PCC (13/48 MHz) 1: External clock from PAD	0x00
5	MPU_SPI2_CLK_POL	0: MPU_SPI2 configuration clock RX (falling edge) 1: MPU_SPI2 configuration clock RX (rising edge)	0x0
4	MCBSP1_CLKS_SEL	Input MCLK of EAC and input CLKS of MCBSP: 0: Internal oscillator, from PCC (13/48 MHz) 1: External codec clock from pad	0X0
3:2	INTERNAL_EAC_BT_ AUSPI_SOURC	Selected source to EAC Bt AuSPI port module 00: Pins BT AuSPI: SCLK, SDI, FSYNC 01: Pins MPU_SPI: MPU_SCLK, MPU_SDI, MPU_SEN1 10: Reserved 11: Gated 0	0x3
1:0	INTERNAL_GSM_VOICE_ SOURCE	Selected source to TMS320C54x vspi (GSM voice), already slave 00: Pins BT AuSPI: SCLK, SDI, FSYNC 01: Internal EAC modem AuSPI 10: Internal VSPI modules 11: Nothing, gated 0	0x3

Table 4–55. OMAP730 Die Identification Number 0 Register (PERSEUS2_GSM_DIE_ID0)

Bit	Name	Function	Reset Value
15:0	GSM_DIE_ID_0	Die ID 0	Unknown

Table 4–56. OMAP730 Die Identification Number 1 Register (PERSEUS2_GSM_DIE_ID1)

Bit	Name	Function	Reset Value
15:0	GSM_DIE_ID_1	Die ID 1	Unknown

Table 4–57. OMAP730 Mode Configuration Register (PERSEUS2_MODE2)

Bit	Name	Function	Reset Value
31:18	UNUSED	Unused	0x0
17	RESERVED	Reserved	0x1
16	NFCS3HL_16_32M	0: NFCCS0L and NFCS0H for 16-MB flash 1: 32-MB flash	0x0
15	VLYNQ_CLKOUT_EN	VLYNQ clock enable	0x1
14	V2O_CLK_EN	OCP clock enable	0x1
13	VOLTAGE_FLASH	Dual-voltage control for flash domain 0: 1,65 V min, 1,8 V nom, 1,95 V max 1: 2.5 V min, 2.75 V nom, 3.0 V max	0x1
12	VOLTAGE_SDRAM	Dual-voltage control for SDRAM domain 0: 1,65 V min, 1,8 V nom, 1,95 V max 1: 2.5 V min, 2.75 V nom, 3.0 V max	0x1
11	RNG_TESTOSC	Used to get test feature 1: Ring oscillator output test	0x0
10	RNG_SELECTOSC	Ring oscillator 1 or 2 selection (0: Oscillator 1/1: Oscillator 2)	0x0
9:8	DLL_SELECT	Select DLL/DCDL: x0: PSTART, PFBK, and DCB[7:0] are selected from URD_DLL. x1: PSTART, PFBK and DCB[7:0] are selected from WRQ_DLL. 00: PMT_DSI and PMT_DSO are selected from URD_DCDL. 01: PMT_DSI and PMT_DSO are selected from WRQ_DCDL. 1x: PMT_DSI and PMT_DSO are selected from LRD_DCDL.	0x00
7	JTAG_EN	0: Disable OMAP JTAG access 1: Enable OMAP JTAG access	0x0

Table 4–57. OMAP730 Mode Configuration Register (PERSEUS2_MODE2) (Continued)

Bit	Name	Function	Reset Value
6	LCD_PSEUDO_18BIT	0: Reset value: 16-bit mode, the pseudo red and blue 6th bits are set to 0. Used to create a 64K color display. 1: Pseudo 18-bit mode, the pseudo red and blue 6th bits are a copy of 6th bit (LSB) green.	0x0
5	SD_REQ_BYPASS	0: SDRAM_REQ = not GPIO4 and ICR_GSM_RST 1: SDRAM_REQ = not GPIO4	0x0
4:3	TAP_CONF	Control the link of all OMAP730 TAPs: 00: MPU mono-emulation 01: GSM-MPU/TMS320C54x biemulation 10: MPU/GSM-MPU/TMS320C54x triemulation 11: Not used	0x2
2	OCP_INTERC_CGM_ENABLE	Enable gated clock feature	0x1
1	OCP_INTERC_SW_RESET	Software reset input signal (active high)	0x0
0	VLYNQ_DEFAULT_CLKDIR	The serial clock may be sourced by the internal VBUS clock (clkdir=1) or by an external clock (cllkdir=0). The default value of the clkdir bit is set during reset with the value of input signal VLYNQ_DEFAULT_CLKDIR	0x0

Table 4–58. OMAP730 Die Identification Number 2 Register (PERSEUS2_GSM_DIE_ID2)

Bit	Name	Function	Reset Value
15:0	GSM_DIE_ID_2	Die ID 2	Unknown

Table 4–59. OMAP730 Die Identification Number 2 Register (PERSEUS2_GSM_DIE_ID3)

Bit	Name	Function	Reset Value
15:0	GSM_DIE_ID_3	Die ID 3	Unknown

Table 4–60. OMAP730 Analog Cells Configuration Register (PERSEUS2_ANALOG_CELLS_CONF)

Bit	Name	Function	Reset Value
31:6	RESERVED	Reserved	0x0
5	SLICER_PWRSEL	1: The slicer is in high-power application 0: The slicer is in low-power application This bit selects either the low-power mode (PWRSEL low) or the high-performance mode (PWRSEL high). The high performance allows a better duty cycle than the low-power mode. The power cost is 300 µA. High performance mode is the default.	0x1
4	SLICER_PWRDN	When selected (equal 1) the cell does not consume any current if the bypass mode is not active.	0x0
3:0	CONTROL_ANALOG_SWITCHES	32k oscillator current gain control analog switches from SW1 to SW3 (see TI OS11V1 cell specification)	0x8

Table 4–61. Secure Register (SECCTRL)

Bit	Name	Function	R/W	Reset Value
31:17	RESERVED	Reserved	R/W	0x0
16	RNG_IDLE_MODE	RNG idle control: 0: RNG idle disabled 1: RNG idle enabled	R/W	0x1
15	PROTECT_CS3_ENABLE	Flash protect CS3 enable: Enable the mechanism protection of CS3. Reset value: The flash CS3 is not protected.	R/W1	0x0
14	PROTECT_CS0_ENABLE	Flash protect CS0 enable Enables CS0 protection mechanism. Reset value: The flash CS0 is not protected.	R/W1	0x0
13:12	BLOCK_SIZE	Flash protection Size of block protected: 00: 32K 01: 64K 10: 128K 11: 256K	R/W1	0x0

Table 4–61. Secure Register (SECCTRL) (Continued)

Bit	Name	Function	R/W	Reset Value
11	CONF_RNG_EN	This is the RNG module access control register. This bit is programmable only in secure mode 0: RNG module access in secure and non-secure modes is enabled. 1: RNG module access in secure mode only is enabled. Reset: 1: Reset condition for a normal production device. 1: Reset condition for an emulation device. This register can be configured as required and does not have a one-time-only configuration restriction.	R/W	0x1
10	CONF_DES_EN	This is the DES/3DES module access control register. This bit is programmable only in secure mode 0: DES/3DES module access in secure and nonsecure modes is enabled. 1: DES/3DES module access is enabled in secure mode only. Reset: 1: Reset condition for a normal production device. 1: Reset condition for an emulation device. This register can be configured as required and does not have a one-time-only configuration restriction.	R/W	0x1

Table 4–61. Secure Register (SECCTRL) (Continued)

Bit	Name	Function	R/W	Reset Value
9	CONF_MUX_CTRL	This bit controls access to the registers that configure multiplexing of the device pins and pullup/pulldown functions. This bit is programmable only in secure mode. 0: Registers FUNC_MUX_CTRL(3:12), PULL_DWN_CTRL(0:4), and PU_PD_SEL(0:4) are accessible at any time. 1: Registers FUNC_MUX_CTRL(3:12), PULL_DWN_CTRL(0:4), and PU_PD_SEL(0:4) are accessible only while the device is in secure mode. Reset: 0: Reset condition for a normal production device 1: Reset condition for an emulation device This register can be configured as required and does not have a one time only configuration restriction for either emulation devices or normal production devices.	R/W	0x0
8	CONF_SHA_EN	Normal production device: R/OTC R&D/SW development device: R SHA-1 module access control register This bit is programmable only in secure mode. 1: SHA-1 module access is enabled in secure mode only 0: SHA-1 module access is enabled in nonsecure and secure modes DMA access to SHA1 is dependant on CONF_SHA_EN and on the register DMA_BLOCK bit in OMAP3.2. Reset: Always 1	R/W	0x1
7	NORMAL_EMU_MODE_ INTERNAL	This bit has information about security type: This bit is programmable only in secure mode: 0: Security type is normal-secure 1: security type is debug-secure This bit is one-time-programmable for normal device and read-only for emulation device. Its reset value is 0 for both normal and emulation device types. Normal production device: R/OTC R&D/SW development device: R	R	0x0

Table 4–61. Secure Register (SECCTRL) (Continued)

Bit	Name	Function	R/W	Reset Value
6	CONF_JTAG_EN	Function: MPU JTAG enable control This bit is programmable only in secure mode: 0: MPU JTAG disabled 1: MPU JTAG enabled (the functionality of the MPU JTAG is not altered) Reset: 0: Normal production device 1: R&D/SW development device This register can be configured one time only for a normal production device. It cannot be configured at all for an emulation(R&D) device. Normal production device: R/OTC R&D/SW development device: R	R	0x0
5	CONF_ETM_EN	Function: ETM enable control This bit is programmable only in secure mode: 1: Trace is not affected 0: Trace is disabled Reset: 0: Reset condition for a normal production device 1: Reset condition for an R&D/SW development device This register can be configured one time only for a normal production device. It cannot be configured at all for an emulation device. Normal production device: R/OTC R&D/SW development device: R	R	0x0

Table 4–61. Secure Register (SECCTRL) (Continued)

Bit	Name	Function	R/W	Reset Value
4	CONF_CKEY_ACC	Function: CKEY access control register This bit is programmable only in secure mode: 1: eFuse access to field C allowed 0: eFuse access to field C forbidden Reset: 0: Reset condition for a normal production device 1: Reset condition for a an emulation (R&D) device This register can be configured one time only for both normal and emulation devices.	R/W1	0x1
3	CONF_RKEY_ACC	Reserved	R	0x1
2	CONF_ICE_EN	Function: MCU emulation enable control This bit is programmable only in secure mode. 0: MCU debug is disable 1: MCU debug is not affected Reset: 0: Reset condition for a normal production device 1: Reset condition for an emulation(R&D) device This register can be configured one time only for a normal production device. It cannot be configured at all for an emulation device. Normal device: R/OTC Development device: R	R	0x0

Table 4–61. Secure Register (SECCTRL) (Continued)

Bit	Name	Function	R/W	Reset Value
1	CONF_WDA_EN	Function: Secure watchdog operation enable control This bit is programmable only in secure mode. 0: OCP error generated/rejected 1: If in secure mode, OCP access done/ granted Reset: 0:Reset condition for an emulation(R&D) device 1:Reset condition for a normal production device This register can be configured one time only for both normal and emulation de-vices.	R/W1	0x1
0	CONF_WD_ACC	Watchdog access R/OTC register Function: Secure watchdog register update access control This bit is programmable only in secure mode. 0: Watchdog timer running 1: Watchdog timer frozen Reset: 0: Reset condition for an emulation(R&D) device 1: Reset condition for a normal production device This register can be configured one time only for both normal and emulation(R&D) devices.	R/W1	0x1

Table 4–62. ECO Spare Register 1 (SPARE1)

Bit	Name	Function	Reset Value
31:1	SPARE1	Spare flops for ECO needs	0x00000000
0	SPARE1_0	Reserved, must be set to 0	0x0

Table 4–63. ECO Spare Register 2 (SPARE2)

Bit	Name	Function	Reset Value
31:0	SPARE2	Spare flops for ECO needs	0x0

Table 4–64. Edge Register—GSM Domain (GSM_PBG_IRQ)

Bit	Name	Function	Reset Value
15:0	RESERVED	Reserved	0x0

Note: This register is compliant with TBB2100.

Table 4–65. DMA Mode Configuration Register (DMA_REQ_CONF)

Bit	Name	Function	Reset Value
31	RESERVED	Reserved	0x0
30:0	DMA_EDGE_EN	DMA request kind 0: Transition 1: Edge	0x7FFFFFFF

DMA_REQ_CONF describes the kinds of DMA requests. Each DMA request can be software programmed to trigger on edge or transition.

Table 4–66. Edge Register—MPU Domain (PE_CONF_NO_DUAL)

Bit	Name	Function	Reset Value
31:8	RESERVED	Reserved	0X0
7	RESERVED	Reserved	0x0
6	PE_MPU_NRST	Pull enable: 0: Disable 1: Enable	0x1
5	RESERVED	Reserved	0x0
4	PE_NBSCAN	Pull enable: 0: Disable 1: Enable	0x1
3	PE_TDI	Pull enable: 0: Disable 1: Enable	0x1
2	PE_TCK	Pull enable: 0: Disable 1: Enable	0x1
1	PE_NTRST	Pull enable: 0: Disable 1: Enable	0x1
0	PE_TMS	Pull enable: 0: Disable 1: Enable	0x1

Note: Special feature for signal with no dual modes. Other pull-enable controls are part of IO_CONF registers.

PE_CONF_NO_DUAL contains the pull-enable control signals for pads with no dual modes.

Table 4–67. OMAP730 Shared I/O Register 0 (PERSEUS_IO_CONF0)

Bit	Name	Function	Reset Value
31:29	D_TPU_TSPEN1	000: TSPEN_1 001: 010: 011: 100: 101: 110: GPIO_8	0x6
28	PE_TPU_TSPEN1	PE_TPU_TSPEN1 pull-enable control	0x1
27:25	D_TPU_TSPEN0	000: TSPEN_2 001: 010: 011: 100: 101: 110: GPIO_7	0x6
24	PE_TPU_TSPEN0	PE_TPU_TSPEN0 pull-enable control	0x1
23:21	D_TPU_TSPACT4	000: TSPACT_4, TSPDO 001: IO_GSM_1 010: 011: 100: 101: 110: GPIO_5, GPIO_6	0x6
20	PE_TPU_TSPACT4	PE_TPU_TSPACT4 pull-enable control	0x1
19:17	D_TPU_TSPACT3	000: TSPACT_3 001: IO_GSM_0 010: 011: 100: 101: 110: GPIO_4	0x6
16	PE_TPU_TSPACT3	PE_TPU_TSPACT3 pull-enable control	0x1
15:13	D_TPU_TSPACT2	000: TSPACT_2 001: 010: 011: 100: 101: 110: GPIO_3	0x6
12	PE_TPU_TSPACT2	PE_TPU_TSPACT2 pull-enable control	0x1

Table 4–67. OMAP730 Shared I/O Register 0 (PERSEUS_IO_CONF0) (Continued)

Bit	Name	Function	Reset Value
11:9	D_TPU_TSPACT1	000: TSPACT_1 001: 010: 011: 100: 101: 110: GPIO_2	0x6
8	PE_TPU_TSPACT1	PE_TPU_TSPACT1 pull-enable control	0x1
7:5	D_TPU_TSPACT0	000: TSPACT_0 001: 010: 011: 100: 101: 110: GPIO_1	0x6
4	PE_TPU_TSPACT0	PE_TPU_TSPACT0 pull-enable control	0x1
3:1	D_TSPCLKX_TSPDO	000: TSPCLKX 001: 010: 011: 100: 101: 110: GPIO_0	0x6
0	PE_TSPCLKX_TSPDO	PE_TSPCLKX_TSPDO pull-enable control	0x1

Table 4–68. OMAP730 Shared I/O Register 1 (PERSEUS_IO_CONF1)

Bit	Name	Function	Reset Value
31:29	D_RFEN	000: RFEN 001: IO_GSM_2 010: 011: 100: 101: 110: GPIO_19	0x0
28	PE_RFEN	PE_RFEN pull-enable control	0x1
27:25	D_TCXOEN	000: TCXOEN 001: 010: 011: 100: 101: 110: GPIO_18	0x0
24	PE_TCXOEN	PE_TCXOEN pull-enable control	0x1

Table 4–68. OMAP730 Shared I/O Register 1 (PERSEUS_IO_CONF1) (Continued)

Bit	Name	Function	Reset Value
23:21	D_IT_WAKEUP	000: IT_WAKEUP 001: 010: 011: 100: 101: 110:	0x0
20	PE_IT_WAKEUP	PE_IT_WAKEUP pull-enable control	0x1
19:17	D_BB_SIM_CD	000: SIM_CD, SIM_RST 001: , 010: IO_GSM_2, 011: MPU_I2C_SCK, 100: , 101: , 110: GPIO_16, GPIO_17	0x6
16	PE_BB_SIM_CD	PE_BB_SIM_CD pull-enable control	0x1
15:13	D_BB_SIM_PWR	000: SIM_PWRCTRL 001: 010: IO_GSM_4 011: MPU_I2C_SDA 100: 101: 110: GPIO_15	0x6
12	PE_BB_SIM_PWR	PE_BB_SIM_PWR pull-enable control	0x1
11:9	D_BB_SIM	000: SIM_IO, SIM_CLK 001: , 010: , 011: , 100: , 101: , 110: GPIO_13, GPIO_14	0x6
8	PE_BB_SIM	PE_BB_SIM pull-enable control	0x1
7:5	D_BB_IF	000: BFSR, BDR, BFSX, BDX 001: , , , 010: , , , 011: , , , 100: , , , 101: , , , 110: GPIO_10, GPIN_1, GPIO_11, GPIO_12	0x6
4	PE_BB_IF	PE_BB_IF pull-enable control	0x1

Table 4–68. OMAP730 Shared I/O Register 1 (PERSEUS_IO_CONF1) (Continued)

Bit	Name	Function	Reset Value
3:1	D_TPU_TSPEN2	000: TSPEN_0 001: 010: 011: 100: 101: 110: GPIO_9	0x6
0	PE_TPU_TSPEN2	PE_TPU_TSPEN2 pull-enable control	0x1

Table 4–69. OMAP730 Shared I/O Register 2 (PERSEUS_IO_CONF2)

Bit	Name	Function	Reset Value
31:29	D_VBUSI	000: USB_VBUSI 001: MPU_UART_RTS1 010: GSM_UART_RTS 011: USB_TXEN 100: GSM_MCSI_TXD 101: MPU_MCSI_TXD 110: GPIO_34	0x6
28	PE_VBUSI	PE_VBUSI pull-enable control	0x1
27:25	D_PU_EN	000: USB_PU_EN 001: MPU_UART_CTS1 010: GSM_UART_CTS 011: USB_RCV 100: GSM_MCSI_CLK 101: MPU_MCSI_CLK 110: GPIO_33	0x6
24	PE_PU_EN	PE_PU_EN pull-enable control	0x1
23:21	D_DM	000: USB_DM, USB_DP 001: MPU_UART_TX1, MPU_UART_RX1 010: MPU_UART_TX_IR2, MPU_UART_RX_IR2 011: USB_SEO_VM, USB_TXD_VP 100: USB_SEO, USB_TXD 101: , 110: GPIO_31, GPIO_32	0x6
20	PE_DM	PE_DM pull-enable control	0x1
19:17	D_SDMC_DAT3	000: SDMC_DAT_3 001: MPU_SPI1_SEN2 010: 011: 100: 101: 110: GPIO_30	0x6
16	PE_SDMC_DAT3	PE_SDMC_DAT3 pull-enable control	0x1

Table 4–69. OMAP730 Shared I/O Register 2 (PERSEUS_IO_CONF2) (Continued)

Bit	Name	Function	Reset Value
15:13	D_SDMC_DAT2	000: SDMC_DAT_2 001: MPU_SPI1_SEN1 010: CF_RESET 011: 100: 101: 110: GPIO_29	0x6
12	PE_SDMC_DAT2	PE_SDMC_DAT2 pull-enable control	0x1
11:9	D_SDMC	000: SDMC_CLK, SDMC_CMD, SDMC_DAT_0, SDMC_DAT_1 001: MPU_SPI1_SCLK, MPU_SPI1_SDO, MPU_SPI1_SDI, MPU_SPI1_SEN0 010: CF_nCD1, CF_nCD2, CF_nIOIS16, CF_INTREQ 011: , , , 100: , , , 101: , , , 110: GPIO_25, GPIO_26, GPIO_27, GPIO_28	0x6
8	PE_SDMC	PE_SDMC pull-enable control	0x1
7:5	D_SYREN_VOICE	000: SCLK, SDO, SDI, FSYNC 001: VCLKRX, VDX, VDR, VFSRX 010: MPU_MCSI_CLK, MPU_MCSI_TXD, MPU_MCSI_RXD, MPU_MCSI_FSYNCH 011: , , , 100: , , , 101: , , , 110: GPIO_22, GPIO_23, GPIN_3, GPIO_24	0x6
4	PE_SYREN_VOICE	PE_SYREN_VOICE pull-enable control	0x1
3:1	D_SYREN_SPI	000: MCUDI, MCUDO, MCUEN 001: , , 010: , , 011: , , 100: , , 101: , , 110: GPIN_2, GPIO_20, GPIO_21	0x6
0	PE_SYREN_SPI	PE_SYREN_SPI pull-enable control	0x1

Table 4–70. OMAP730 Shared I/O Register 3 (PERSEUS_IO_CONF3)

Bit	Name	Function	Reset Value
31:29	D_LCD_PXL_15_12	000: LCD_PIXEL_15, LCD_PIXEL_14, LCD_PIXEL_13, LCD_PIXEL_12 001: GSM_MCSI_FSYNCH, GSM_MCSI_CLK, GSM_MCSI_TXD, GSM_MCSI_RXD 010: MPU_SPI2_SCLK, MPU_SPI2_SDO, MPU_SPI2_SDI, MPU_SPI2_SEN0 011: , , , 100: , , , 101: , , , 110: GPIO_44, GPIO_45, GPIO_46, GPIO_47	0x6
28	PE_LCD_PXL_15_12	PE_LCD_PXL_15_12 pull-enable control	0x1
27:25	D_UART_RTS_CTS	000: MPU_UART_CTS1, MPU_UART_RTS1 001: VLYNQ_RXD1, VLYNQ_TXD1 010: MPU_UART_CTS2, MPU_UART_RTS2 011: GSM_UW_SCLK, GSM_UW_nSCS1 100: MPU_SPI1_SDI, MPU_SPI1_SEN0 101: TSPACT_7, TSPACT_8 110: GPIO_42, GPIO_43	0x6
24	PE_UART_RTS_CTS	PE_UART_RTS_CTS pull-enable control	0x1
23:21	D_UART_TX_RX	000: MPU_UART_TX1, MPU_UART_RX1 001: VLYNQ_TXD0, VLYNQ_RXD0 010: MPU_UART_TX2, MPU_UART_RX2 011: GSM_UW_SDO, GSM_UW_SDI 100: MPU_SPI1_SCLK, MPU_SPI1_SDO 101: TSPACT_5, TSPACT_6 110: GPIO_40, GPIO_41	0x6
20	PE_UART_TX_RX	PE_UART_TX_RX pull-enable control	0x1
19:17	D_UART_IRDA_SD	000: MPU_UART_SD2 001: HDQ1W 010: 011: 100: 101: 110: GPIO_39	0x6
16	PE_UART_IRDA_SD	PE_UART_IRDA_SD pull-enable control	0x1
15:13	D_UART_IRDA_RX	000: MPU_UART_RX_IR2 001: LCD_PIXEL_17 010: 011: 100: GSM_UART_RX 101: 110: GPIO_38	0x6
12	PE_UART_IRDA_RX	PE_UART_IRDA_RX pull-enable control	0x1

Table 4–70. OMAP730 Shared I/O Register 3 (PERSEUS_IO_CONF3) (Continued)

Bit	Name	Function	Reset Value
11:9	D_UART_IRDA_TX	000: MPU_UART_TX_IR2 001: LCD_PIXEL_16 010: 011: 100: GSM_UART_TX 101: 110: GPIO_37	0x6
8	PE_UART_IRDA_TX	PE_UART_IRDA_TX pull-enable control	0x1
7:5	D_CRESET	000: CRESET 001: MPU_UART_DSR1 010: GSM_UART_RX 011: USB_VP 100: GSM_MCSI_FSYNCH 101: MPU_MCSI_FSYNCH 110: GPIO_36	0x6
4	PE_CRESET	PE_CRESET pull-enable control	0x1
3:1	D_MCLK_OUT	000: MCLK_OUT 001: MPU_UART_DCD1 010: GSM_UART_TX 011: USB_VM 100: GSM_MCSI_RXD 101: MPU_MCSI_RXD 110: GPIO_35	0x6
0	PE_MCLK_OUT	PE_MCLK_OUT pull-enable control	0x1

Table 4–71. OMAP730 Shared I/O Register 4 (PERSEUS_IO_CONF4)

Bit	Name	Function	Reset Value
31:29	D_EAC_MCLK	000: MCLK 001: VLYNQ_CLK 010: CLKS1 011: TSPACT_10 100: IO_GSM_1 101: CLK13M_IN 110: GPIO_68	0x6
28	PE_EAC_MCLK	PE_EAC_MCLK pull-enable control	0x1
27:25	D_EAC_CDI	000: CDI 001: KBC_6 010: DR1 011: TSPACT_9 100: IO_GSM_0 101: 110: GPIO_67	0x6
24	PE_EAC_CDI	PE_EAC_CDI pull-enable control	0x1

Table 4–71. OMAP730 Shared I/O Register 4 (PERSEUS_IO_CONF4) (Continued)

Bit	Name	Function	Reset Value
23:21	D_EAC	000: CSYNC, CSCLK, CDO 001: KBR_5, KBR_6, KBC_5 010: FSRX1, CLK.1, DX1 011: , , 100: , , 101: , , 110: GPIO_64, GPIO_65, GPIO_66	0x6
20	PE_EAC	PE_EAC pull-enable control	0x1
19:17	D_LCD_VSYNC	000: LCD_VSYNC, LCD_AC 001: MPU_UW_nSCS2, MPU_UW_SDI 010: DR2, CLKS2 011: , 100: , 101: , 110: GPIO_62, GPIO_63	0x6
16	PE_LCD_VSYNC	PE_LCD_VSYNC pull-enable control	0x1
15:13	D_LCD_UWIRE	000: LCD_PIXEL_0, LCD_PCLK, LCD_HSYNC 001: MPU_UW_nSCS1, MPU_UW_SCLK, MPU_UW_SDO 010: FSRX2, CLK.2, DX2 011: , , 100: , , 101: , , 110: GPIO_59, GPIO_60, GPIO_61	0x6
12	PE_LCD_UWIRE	PE_LCD_UWIRE pull-enable control	0x1
11:9	D_LCD_PXL_9_2	000: LCD_PIXEL_9, LCD_PIXEL_8, LCD_PIXEL_7, LCD_PIXEL_6, LCD_PIXEL_5, LCD_PIXEL_4, LCD_PIXEL_3, LCD_PIXEL_2, LCD_PIXEL_1 001: , , , , , , , , 010: , , , , , , , , 011: , , , , , , , , 100: , , , , , , , , 101: , , , , , , , , 110: GPIO_50, GPIO_51, GPIO_52, GPIO_53, GPIO_54, GPIO_55, GPIO_56, GPIO_57, GPIO_58	0x6
8	PE_LCD_PXL_9_2	PE_LCD_PXL_9_2 pull-enable control	0x1
7:5	D_LCD_PXL_10	000: LCD_PIXEL_10 001: 010: MPU_SPI2_SEN2 011: 100: 101: 110: GPIO_49	0x6
4	PE_LCD_PXL_10	PE_LCD_PXL_10 pull-enable control	0x1

Table 4–71. OMAP730 Shared I/O Register 4 (PERSEUS_IO_CONF4) (Continued)

Bit	Name	Function	Reset Value
3:1	D_LCD_PXL_11	000: LCD_PIXEL_11 001: 010: MPU_SPI2_SEN1 011: 100: 101: 110: GPIO_48	0x6
0	PE_LCD_PXL_11	PE_LCD_PXL_11 pull-enable control	0x1

Table 4–72. OMAP730 Shared I/O Register 5 (PERSEUS_IO_CONF5)

Bit	Name	Function	Reset Value
31:29	D_EMIF_FADD22	000: FADD_22 001: 010: 011: 100: 101: 110: GPIO_78	0x0
28	PE_EMIF_FADD22	PE_EMIF_FADD22 pull-enable control	0x1
27:25	D_EMIF_FADD23	000: FADD_23 001: 010: 011: 100: 101: 110: GPIO_77	0x0
24	PE_EMIF_FADD23	PE_EMIF_FADD23 pull-enable control	0x1
23:21	D_EMIF_FADD24	000: FADD_24 001: 010: 011: 100: 101: 110: GPIO_76	0x0
20	PE_EMIF_FADD24	PE_EMIF_FADD24 pull-enable control	0x1
19:17	D_EMIF_FADD25	000: FADD_25 001: CLK_13M_REQ 010: NFCS_0 011: 100: 101: 110: GPIO_75	0x0
16	PE_EMIF_FADD25	PE_EMIF_FADD25 pull-enable control	0x1

Table 4–72. OMAP730 Shared I/O Register 5 (PERSEUS_IO_CONF5) (Continued)

Bit	Name	Function	Reset Value
15:13	D_DDR	000: DQSH, DQSL, SDCLKX 001: , , 010: , , 011: , , 100: , , 101: , , 110: GPIO_72, GPIO_73, GPIO_74	0x6
12	PE_DDR	PE_DDR pull-enable control	0x1
11:9	D_SDCS	000: nSDCS 001: 010: 011: 100: 101: 110: GPIO_71	0x0
8	PE_SDCS	PE_SDCS pull-enable control	0x1
7:5	D_I2C_SCK	000: MPU_I2C_SCK 001: EXT_IO_3 010: GSM_I2C_SCK 011: LCD_PIXEL_17 100: 101: USB_SPEED 110: GPIO_70	0x6
4	PE_I2C_SCK	PE_I2C_SCK pull-enable control	0x1
3:1	D_I2C_SDA	000: MPU_I2C_SDA 001: EXT_IO_2 010: GSM_I2C_SDA 011: LCD_PIXEL_16 100: 101: USB_SUSPEND 110: GPIO_69	0x6
0	PE_I2C_SDA	PE_I2C_SDA pull-enable control	0x1

Table 4–73. OMAP730 Shared I/O Register 6 (PERSEUS_IO_CONF6)

Bit	Name	Function	Reset Value
31:29	D_NFC_CE2	000: FADD_10, FADD_9, FADD_8, FADD_7, FADD_6, FADD_5, FADD_4, FADD_3, FADD_2, FADD_1 001: CE_2, I/O_0, WP, I/O_6, I/O_5, I/O_4, I/O_3, I/O_2, I/O_1, RE 010: , , , , , , , , , 011: , , , , , , , , , 100: , , , , , , , , , 101: , , , , , , , , , 110: GPIO_90, GPIO_91, GPIO_92, GPIO_93, GPIO_94, GPIO_95, GPIO_96, GPIO_97, GPIO_98, GPIO_99	0x0
28	PE_NFC_CE2	PE_NFC_CE2 pull-enable control	0x1
27:25	D_NFC	000: FADD_15, FADD_14, FADD_13, FADD_12, FADD_11 001: ALE, CLE, WE, I/O_7, CE_1 010: , , , , 011: , , , , 100: , , , , 101: , , , , 110: GPIO_85, GPIO_86, GPIO_87, GPIO_88, GPIO_89	0x0
24	PE_NFC	PE_NFC pull-enable control	0x1
23:21	D_EMIF_FADD16	000: FADD_16 001: RDY 010: 011: 100: MPU_SPI2_SEN1 101: 110: GPIO_84	0x0
20	PE_EMIF_FADD16	PE_EMIF_FADD16 pull-enable control	0x1
19:17	D_EMIF_FADD17	000: FADD_17 001: 010: 011: 100: MPU_SPI2_SEN0 101: 110: GPIO_83	0x0
16	PE_EMIF_FADD17	PE_EMIF_FADD17 pull-enable control	0x1
15:13	D_EMIF_FADD18	000: FADD_18 001: 010: 011: 100: MPU_SPI2_SDI 101: 110: GPIO_82	0x0
12	PE_EMIF_FADD18	PE_EMIF_FADD18 pull-enable control	0x1

Table 4–73. OMAP730 Shared I/O Register 6 (PERSEUS_IO_CONF6) (Continued)

Bit	Name	Function	Reset Value
11:9	D_EMIF_FADD19	000: FADD_19 001: 010: 011: 100: MPU_SPI2_SDO 101: 110: GPIO_81	0x0
8	PE_EMIF_FADD19	PE_EMIF_FADD19 pull-enable control	0x1
7:5	D_EMIF_FADD20	000: FADD_20 001: 010: 011: 100: MPU_SPI2_SCLK 101: 110: GPIO_80	0x0
4	PE_EMIF_FADD20	PE_EMIF_FADD20 pull-enable control	0x1
3:1	D_EMIF_FADD21	000: FADD_21 001: 010: 011: 100: 101: 110: GPIO_79	0x0
0	PE_EMIF_FADD21	PE_EMIF_FADD21 pull-enable control	0x1

Table 4–74. OMAP730 Shared I/O Register 7 (PERSEUS_IO_CONF7)

Bit	Name	Function	Reset Value
31:29	D_EMIF_FCLK	000: FCLK 001: 010: 011: 100: 101: 110: GPIO_124	0x0
28	PE_EMIF_FCLK	PE_EMIF_FCLK pull-enable control	0x1
27:25	D_EMIF_RDY	000: NFWAIT 001: RDY 010: 011: 100: 101: 110: GPIO_123	0x0
24	PE_EMIF_RDY	PE_EMIF_RDY pull-enable control	0x1

Table 4–74. OMAP730 Shared I/O Register 7 (PERSEUS_IO_CONF7) (Continued)

Bit	Name	Function	Reset Value
23:21	D_EMIF_NFBAA	000: NFBAA, NFWP 001: , 010: , 011: , 100: , 101: , 110: GPIO_121, GPIO_122	0x0
20	PE_EMIF_NFBAA	PE_EMIF_NFBAA pull-enable control	0x1
19:17	D_EMIF	000: NFWE, NFOE 001: , 010: , 011: , 100: , 101: , 110: GPIO_119, GPIO_120	0x0
16	PE_EMIF	PE_EMIF pull-enable control	0x1
15:13	D_EMIF_NFC3	000: NFCS_3 001: 010: NFCS3L 011: 100: 101: 110: GPIO_118	0x0
12	PE_EMIF_NFC3	PE_EMIF_NFC3 pull-enable control	0x1
11:9	D_EMIF_NFC1	000: NFCS_1 001: 010: NFCS3H 011: 100: 101: 110: GPIO_117	0x0
8	PE_EMIF_NFC1	PE_EMIF_NFC1 pull-enable control	0x1
7:5	D_EMIF_NFC2	000: NFCS_2 001: FADD_25 010: 011: 100: 101: 110: GPIO_116	0x0
4	PE_EMIF_NFC2	PE_EMIF_NFC2 pull-enable control	0x1

Table 4–74. OMAP730 Shared I/O Register 7 (PERSEUS_IO_CONF7) (Continued)

Bit	Name	Function	Reset Value
3:1	D_EMIF_NFDATA15_0	000: FDATA_15, FDATA_14, FDATA_13, FDATA_12, FDATA_11, FDATA_10, FDATA_9, FDATA_8, FDATA_7, FDATA_6, FDATA_5, FDATA_4, FDATA_3, FDATA_2, FDATA_1, FDATA_0 001: , , , , , , , , , , , , , , , , 010: , , , , , , , , , , , , , , , , 011: , , , , , , , , , , , , , , , , 100: , , , , , , , , , , , , , , , , 101: , , , , , , , , , , , , , , , , 110: GPIO_100, GPIO_101, GPIO_102, GPIO_103, GPIO_104, GPIO_105, GPIO_106, GPIO_107, GPIO_108, GPIO_109, GPIO_110, GPIO_111, GPIO_112, GPIO_113, GPIO_114, GPIO_115	0x0
0	PE_EMIF_NFDATA15_0	PE_EMIF_NFDATA15_0 pull-enable control	0x1

Table 4–75. OMAP730 Shared I/O Register 8 (PERSEUS_IO_CONF8)

Bit	Name	Function	Reset Value
31:29	D_SPI1_SEN0	000: MPU_SPI1_SEN0 001: FSYNC 010: MPU_MCSI_FSYNCH 011: GSM_UW_nSCS1 100: FSRX1 101: GSM_MCSI_FSYNCH 110: GPIO_132	0x6
28	PE_SPI1_SEN0	PE_SPI1_SEN0 pull-enable control	0x1
27:25	D_SPI1_SDI	000: MPU_SPI1_SDI 001: SDI 010: MPU_MCSI_RXD 011: GSM_UW_SDI 100: DR1 101: GSM_MCSI_RXD 110: GPIO_131	0x6
24	PE_SPI1_SDI	PE_SPI1_SDI pull-enable control	0x1
23:21	D_SPI1_SDO	000: MPU_SPI1_SDO 001: SDO 010: MPU_MCSI_TXD 011: GSM_UW_SDO 100: DX1 101: GSM_MCSI_TXD 110: GPIO_130	0x6
20	PE_SPI1_SDO	PE_SPI1_SDO pull-enable control	0x1

Table 4–75. OMAP730 Shared I/O Register 8 (PERSEUS_IO_CONF8) (Continued)

Bit	Name	Function	Reset Value
19:17	D_SPI1_SCLK	000: MPU_SPI1_SCLK 001: SCLK 010: MPU_MCSI_CLK 011: GSM_UW_SCLK 100: CLK.1 101: GSM_MCSI_CLK 110: GPIO_129	0x6
16	PE_SPI1_SCLK	PE_SPI1_SCLK pull-enable control	0x1
15:13	D_EMIF_NFRST	000: NFRST 001: 010: 011: 100: 101: 110: GPIO_128	0x0
12	PE_EMIF_NFRST	PE_EMIF_NFRST pull-enable control	0x1
11:9	D_EMIF_NFBE0	000: NFBE_0 001: 010: BCLKR 011: 100: 101: 110: GPIO_127	0x0
8	PE_EMIF_NFBE0	PE_EMIF_NFBE0 pull-enable control	0x1
7:5	D_EMIF_NFBE1	000: NFBE_1 001: 010: BCLKX 011: NFCS_0 100: 101: 110: GPIO_126	0x0
4	PE_EMIF_NFBE1	PE_EMIF_NFBE1 pull-enable control	0x1
3:1	D_EMIF_NFADV	000: NFADV 001: 010: NFCS_0 011: 100: 101: 110: GPIO_125	0x0
0	PE_EMIF_NFADV	PE_EMIF_NFADV pull-enable control	0x1

Table 4–76. OMAP730 Shared I/O Register 9 (PERSEUS_IO_CONF9)

Bit	Name	Function	Reset Value
31:29	D_MPU_NIRQ	000: MPU_EXT_NIRQ 001: USB_VBUSI 010: 011: 100: 101: 110: GPIN_4	0x6
28	PE_MPU_NIRQ	PE_MPU_NIRQ pull-enable control	0x1
27:25	D_SMC_PWR	000: SMC_PWCTRL 001: XF 010: TSPACT_9 011: KBC_6 100: GSM_UW_nSCS2 101: MPU_UW_nSCS2 110: GPIO_139	0x6
24	PE_SMC_PWR	PE_SMC_PWR pull-enable control	0x1
23:21	D_SMC_CD	000: SMC_CD 001: IT_FRAME 010: TSPACT_8 011: TSPACT_11 100: GSM_UW_nSCS1 101: MPU_UW_nSCS1 110: GPIO_138	0x6
20	PE_SMC_CD	PE_SMC_CD pull-enable control	0x1
19:17	D_SMC_RST	000: SMC_RST 001: GSM_IOx 010: TSPACT_7 011: KBC_5 100: GSM_UW_SDO 101: MPU_UW_SDO 110: GPIO_137	0x6
16	PE_SMC_RST	PE_SMC_RST pull-enable control	0x1
15:13	D_SMC_CLK	000: SMC_CLK 001: EXT_DSP_NIRQ 010: TSPACT_6 011: KBR_6 100: GSM_UW_SCLK 101: MPU_UW_SCLK 110: GPIO_136	0x6
12	PE_SMC_CLK	PE_SMC_CLK pull-enable control	0x1

Table 4–76. OMAP730 Shared I/O Register 9 (PERSEUS_IO_CONF9) (Continued)

Bit	Name	Function	Reset Value
11:9	D_SMC_IO	000: SMC_IO 001: EXT_ARM_NIRQ 010: TSPACT_5 011: KBR_5 100: GSM_UW_SDI 101: MPU_UW_SDI 110: GPIO_135	0x6
8	PE_SMC_IO	PE_SMC_IO pull-enable control	0x1
7:5	D_SPI1_SEN2	000: MPU_SPI1_SEN2 001: CLK13M_IN 010: CRESET 011: MPU_UART_DTR1 100: 101: IO_GSM_3 110: GPIO_134	0x6
4	PE_SPI1_SEN2	PE_SPI1_SEN2 pull-enable control	0x1
3:1	D_SPI1_SEN1	000: MPU_SPI1_SEN1 001: SEN1 010: 011: GSM_UW_nSCS2 100: CLKS1 101: IO_GSM_2 110: GPIO_133	0x6
0	PE_SPI1_SEN1	PE_SPI1_SEN1 pull-enable control	0x1

Table 4–77. OMAP730 Shared I/O Register 10 (PERSEUS_IO_CONF10)

Bit	Name	Function	Reset Value
31:29	D_CLK13MREQ	000: CLK_13M_REQ 001: IO_GSM_3 010: 011: PIPESTAT_3 100: 101: 110: GPIO_145	0x6
28	PE_CLK13MREQ	PE_CLK13MREQ pull-enable control	0x1
27:25	D_CLK32K	000: CLK32K 001: 010: 011: 100: 101: 110:	0x0
24	PE_CLK32K	PE_CLK32K pull-enable control	0x1

Table 4–77. OMAP730 Shared I/O Register 10 (PERSEUS_IO_CONF10) (Continued)

Bit	Name	Function	Reset Value
23:21	D_TEST_MODE	000: TEST_MODE 001: 010: 011: 100: 101: 110: GPIO_144	0x0
20	PE_TEST_MODE	PE_TEST_MODE pull-enable control	0x1
19:17	D_NEMU1	000: NEMU1 001: PWL 010: HDQ1W 011: PWT 100: 101: 110: GPIO_143	0x0
16	PE_NEMU1	PE_NEMU1 pull-enable control	0x1
15:13	D_NEMU0	000: NEMU0 001: Low_power 010: CLK48M_IN 011: TSPEN_3 100: 101: 110: GPIO_142	0x0
12	PE_NEMU0	PE_NEMU0 pull-enable control	0x1
11:9	D_MPU_LPG2	000: ARM_boot_MLPG2 001: GSM_LPG2 010: RTCK 011: EXT_IO_1 100: 101: 110: GPIO_141	0x2
8	PE_MPU_LPG2	PE_MPU_LPG2 pull-enable control	0x1
7:5	D_MPU_LPG1	000: Mux_mode_MLPG1 001: GSM_LPG1 010: 011: EXT_IO_0 100: NFIQ_PWRFAIL 101: 110: GPIO_140	0x0
4	PE_MPU_LPG1	PE_MPU_LPG1 pull-enable control	0x1

Table 4–77. OMAP730 Shared I/O Register 10 (PERSEUS_IO_CONF10) (Continued)

Bit	Name	Function	Reset Value
3:1	D_GSM_NIRQ	000: GSM_EXT_NIRQ 001: 010: 011: 100: 101: 110: GPIN_5	0x6
0	PE_GSM_NIRQ	PE_GSM_NIRQ pull-enable control	0x1

Table 4–78. OMAP730 Shared I/O Register 11 (PERSEUS_IO_CONF11)

Bit	Name	Function	Reset Value
31:29	D_CAM_DAT2	000: CAM_DATA_2 001: FSRX1 010: IO_GSM_1 011: TRACEPKT_2 100: GSM_UW_nSCS1 101: DEBUG_5 110: GPIO_153	0x6
28	PE_CAM_DAT2	PE_CAM_DAT2 pull-enable control	0x1
27:25	D_CAM_DAT1	000: CAM_DATA_1 001: DR1 010: IO_GSM_0 011: TRACEPKT_1 100: GSM_UW_SDO 101: DEBUG_6 110: GPIO_152	0x6
24	PE_CAM_DAT1	PE_CAM_DAT1 pull-enable control	0x1
23:21	D_CAM_DAT0	000: CAM_DATA_0 001: DX1 010: KBC_7 011: TRACEPKT_0 100: GSM_UW_SCLK 101: DEBUG_7 110: GPIO_151	0x6
20	PE_CAM_DAT0	PE_CAM_DAT0 pull-enable control	0x1
19:17	D_CAM_RSTZ	000: CAM_RSTZ 001: CLK.1 010: KBC_6 011: PIPESTAT_0 100: GSM_UW_SDI 101: DEBUG_8 110: GPIO_150	0x6
16	PE_CAM_RSTZ	PE_CAM_RSTZ pull-enable control	0x1

Table 4–78. OMAP730 Shared I/O Register 11 (PERSEUS_IO_CONF11) (Continued)

Bit	Name	Function	Reset Value
15:13	D_CAM_VS	000: CAM_VS 001: GSM_MCSI_FSYNCH 010: KBC_5 011: PIPESTAT_1 100: GSM_UW_nSCS2 101: DEBUG_9 110: GPIO_149	0x6
12	PE_CAM_VS	PE_CAM_VS pull-enable control	0x1
11:9	D_CAM_HS	000: CAM_HS 001: GSM_MCSI_RXD 010: KBR_7 011: PIPESTAT_2 100: ARMIO_2 101: DEBUG_10 110: GPIO_148	0x6
8	PE_CAM_HS	PE_CAM_HS pull-enable control	0x1
7:5	D_CAM_EXCLK	000: CAM_EXCLK 001: GSM_MCSI_TXD 010: KBR_6 011: TRACESYNC 100: ARMIO_1 101: DEBUG_11 110: GPIO_147	0x6
4	PE_CAM_EXCLK	PE_CAM_EXCLK pull-enable control	0x1
3:1	D_CAM_LCLK	000: CAM_LCLK 001: GSM_MCSI_CLK 010: KBR_5 011: TRACECLK 100: ARMIO_0 101: DEBUG_12 110: GPIO_146	0x6
0	PE_CAM_LCLK	PE_CAM_LCLK pull-enable control	0x1

Table 4–79. OMAP730 Shared I/O Register 12 (PERSEUS_IO_CONF12)

Bit	Name	Function	Reset Value
31:29	D_KB2	000: KBR_2 001: MPU_I2C_SDA 010: ARMIO_4 011: 100: TRACESYNCB 101: 110: GPIO_161	0x6
28	PE_KB2	PE_KB2 pull-enable control	0x1

Table 4–79. OMAP730 Shared I/O Register 12 (PERSEUS_IO_CONF12) (Continued)

Bit	Name	Function	Reset Value
27:25	D_KB1	000: KBR_1 001: EXT_IO_1 010: ARMIO_3 011: 100: PIPESTAT_4 101: 110: GPIO_160	0x6
24	PE_KB1	PE_KB1 pull-enable control	0x1
23:21	D_KB0	000: KBR_0 001: EXT_IO_0 010: 011: GSM_I2C_SCK 100: PIPESTAT_5 101: 110: GPIO_159	0x6
20	PE_KB0	PE_KB0 pull-enable control	0x1
19:17	D_CAM_DAT7	000: CAM_DATA_7 001: MPU_MCSI_FSYNCH 010: VLYNQ_TXD1 011: TRACEPKT_7 100: SEN1 101: DEBUG_0 110: GPIO_158	0x6
16	PE_CAM_DAT7	PE_CAM_DAT7 pull-enable control	0x1
15:13	D_CAM_DAT6	000: CAM_DATA_6 001: MPU_MCSI_RXD 010: VLYNQ_RXD1 011: TRACEPKT_6 100: FSYNC 101: DEBUG_1 110: GPIO_157	0x6
12	PE_CAM_DAT6	PE_CAM_DAT6 pull-enable control	0x1
11:9	D_CAM_DAT5	000: CAM_DATA_5 001: MPU_MCSI_TXD 010: VLYNQ_TXD0 011: TRACEPKT_5 100: SDI 101: DEBUG_2 110: GPIO_156	0x6
8	PE_CAM_DAT5	PE_CAM_DAT5 pull-enable control	0x1

Table 4–79. OMAP730 Shared I/O Register 12 (PERSEUS_IO_CONF12) (Continued)

Bit	Name	Function	Reset Value
7:5	D_CAM_DAT4	000: CAM_DATA_4 001: MPU_MCSI_CLK 010: VLYNQ_RXD0 011: TRACEPKT_4 100: SDO 101: DEBUG_3 110: GPIO_155	0x6
4	PE_CAM_DAT4	PE_CAM_DAT4 pull-enable control	0x1
3:1	D_CAM_DAT3	000: CAM_DATA_3 001: CLKS1 010: IO_GSM_2 011: TRACEPKT_3 100: SCLK 101: DEBUG_4 110: GPIO_154	0x6
0	PE_CAM_DAT3	PE_CAM_DAT3 pull-enable control	0x1

Table 4–80. OMAP730 Shared I/O Register 13 (PERSEUS_IO_CONF13)

Bit	Name	Function	Reset Value
31:28	RESERVED	Reserved	0x0
27:25	D_KB9	000: KBC_4 001: GSM_UW_SCLK 010: VLYNQ_TXD1 011: 100: MPU_SPI2_SEN0 101: EXTERN1_GSM 110: GPIO_168	0x6
24	PE_KB9	PE_KB9 pull-enable control	0x1
23:21	D_KB8	000: KBC_3 001: GSM_UW_SDI 010: VLYNQ_RXD1 011: 100: MPU_SPI2_SDI 101: EXTERN0_GSM 110: GPIO_167	0x6
20	PE_KB8	PE_KB8 pull-enable control	0x1
19:17	D_KB7	000: KBC_2 001: GSM_UW_SDO 010: 011: 100: MPU_SPI2_SDO 101: 110: GPIO_166	0x6

Table 4–80. OMAP730 Shared I/O Register 13 (PERSEUS_IO_CONF13) (Continued)

Bit	Name	Function	Reset Value
16	PE_KB7	PE_KB7 pull-enable control	0x1
15:13	D_KB6	000: KBC_1 001: EXT_IO_3 010: 011: 100: MPU_SPI2_SEN2 101: 110: GPIO_165	0x6
12	PE_KB6	PE_KB6 pull-enable control	0x1
11:9	D_KB5	000: KBC_0 001: EXT_IO_2 010: 011: GSM_I2C_SDA 100: MPU_SPI2_SEN1 101: 110: GPIO_164	0x6
8	PE_KB5	PE_KB5 pull-enable control	0x1
7:5	D_KB4	000: KBR_4 001: GSM_UW_nSCS1 010: VLYNQ_RXD0 011: TSPDI 100: MPU_SPI2_SCLK 101: EXTERN1_MPU 110: GPIO_163	0x6
4	PE_KB4	PE_KB4 pull-enable control	0x1
3:1	D_KB3	000: KBR_3 001: MPU_I2C_SCK 010: VLYNQ_TXD0 011: TSPEN_3 100: 101: EXTERN0_MPU 110: GPIO_162	0x6
0	PE_KB3	PE_KB3 pull-enable control	0x1

This register allows to control some PCC inputs for the 48-MHz clock muxing, APLL muxing, and backup clock muxing, as well as 48-MHz clock requests for modules that do not have hardware request and mux for PLL observability.

Table 4–81. 48-MHz Input Control Register (PERSEUS_PCC_CONF_REG)

Bit	Name	Function	R/W	Reset Value
31:10	RESERVED	Reserved	R	0x0
9	PLL_DIV_SEL	Allows selection of the divider ratio for the APLL or DPLL clock output 0: Register in PCC 1: Register in tap controller	R/W	0x0
8	PMT_MPU_SEL	Allows selection of either OMAP PLL output and lock, or PCC APLL48-MHz clock and lock for observability. 0: APLL 48-MHz selected 1: OMAP PLL selected	R/W	0x0
7	PCC_CAM_CLK_REQ	Allows to request the 48-MHz clock the camera interface. 1: Request active 0: Request inactive	R/W	0x0
6	MCBSP1_CLK_REQ	48-MHz or 13-MHz clock request for MCBSP1. The clock frequency must be programmed in the PCC. 0: Clock request inactive 1: Clock request active	R/W	0x0
5	MCBSP2_CLK_REQ	48-MHz or 13-MHz clock request for MCBSP2. The clock frequency must be programmed in the PCC. 0: Clock request inactive 1: Clock request active	R/W	0x0
4	UART3_DPLL_REQ	48-MHz clock request for UART3 0: Clock request inactive 1: Clock request active	R/W	0x0
3	UART1_DPLL_REQ	48-MHz clock request for UART1. 0: Clock request inactive 1: Clock request active	R/W	0x0
2	PCONF_MMC_DPLL_REQ	48-MHz clock request for MMC/SDIO in MMC mode. 0: Clock request inactive 1: Clock request active	R/W	0x0
1	PLL_NCLKEXT_SEL	Allows selection of the APLL48MHz and an external 48-MHz clock. 0: External 48-MHz clock 1: Internal APLL48 MHz	R/W	0x1
0	EXT13M_CLK_NAPLL13	Allows selection of either the APLL13MHz output clock or the ext13m_clk(backup). 0: APLL13MHz used 1: ext13m_clk used	R/W	0x0

Table 4–82. BIST_FAIL_GO Register (BIST_STATUS_INTERNAL)

Bit	Name	Function	Reset Value
31	DONE_COMBINED_INTERNAL	BIST status DONE bit. 1: BIST sequence complete	0x0
30	GLOBAL_FAIL_GO_COMBINED_INTERNAL	BIST status fail bit that combines all fail_go of PERSEUS2 BISTs	0x1
29:23	UNUSED	Unused	0x0
22	TPU_FAIL_GO_INTERNAL	BIST status fail bit	0x1
21	GEA_FAIL_GO_INTERNAL	BIST status fail bit	0x1
20	INTBOOTRAM_FAIL_GO_INTERNAL	BIST status fail bit	0x1
19	SECROM_FAIL_GO_INTERNAL	BIST status fail bit	0x1
18	SECRAM_FAIL_GO_INTERNAL	BIST status fail bit	0x1
17	OMAP_FAIL_GO_INTERNAL	BIST status fail bit	0x1
16	TCIF_FAIL_GO_INTERNAL	BIST status fail bit	0x1
15	ICR_FAIL_GO_INTERNAL	BIST status fail bit	0x1
14	EAC2_FAIL_GO_INTERNAL	BIST status fail bit	0x1
13	EAC1_FAIL_GO_INTERNAL	BIST status fail bit	0x1
12	MMC_FAIL_GO_INTERNAL	BIST status fail bit	0x1
11	FRAME_BUFFER_FAIL_GO_INTERNAL	BIST status fail bit	0x1
10	ARM7_CTRL_2_FAIL_GO_INTERNAL	BIST status fail bit (0.5M b)	0x1
9	ARM7_CTRL_1_FAIL_GO_INTERNAL	BIST status fail bit (2M b)	0x1
8:7	USB_FAIL_GO_INTERNAL	BIST status fail bit	0x1
6:4	UART_MOD_IRDA_FAIL_GO_INTERNAL	BIST status fail bit	0x1
3:1	UART_MOD_FAIL_GO_INTERNAL	BIST status fail bit	0x1
0	CAMERA_FAIL_GO_INTERNAL	BIST fail status bit	0x1

Note: All bits are undefined (U) when BIST mode is not activated.

The BIST_FAIL_GO register contains the BIST staus done and all BIST fail status bits from the OMAP730 BIST controller (OMAP3.2 included).

Table 4–83. BIST Settings Control Register (BIST_CONTROL)

Bit	Name	Function	Reset Value
31:30	UNUSED	Unused	0x0
29	BIST_MODE	BIST mode	0x0
28	MBIST_TCIF_CTRL_EN	BIST enable signal	0x0

Table 4–83. BIST Settings Control Register (BIST_CONTROL) (Continued)

Bit	Name	Function	Reset Value
27	MBIST_ICR_CTRL_EN	BIST enable signal	0x0
26	MBIST_EAC_CTRL_2_EN	BIST enable signal	0x0
25	MBIST_EAC_CTRL_1_EN	BIST enable signal	0x0
24	MBIST_MMC_CTRL_EN	BIST enable signal	0x0
23	MBIST_USB_HM_CTRL_EN	BIST enable signal	0x0
22	MBIST_USB_DM_CTRL_EN	BIST enable signal	0x0
21	MBIST_ARM7_CTRL_2_EN	BIST enable signal	0x0
20	MBIST_ARM7_CTRL_1_EN	BIST enable signal	0x0
19	MBIST_INTBOOTRAM_CTRL_EN	BIST enable signal	0x0
18	MBIST_GEA_CTRL_EN	BIST enable signal	0x0
17	MBIST_TPU_CTRL_EN	BIST enable signal	0x0
16:14	MBIST_UART_MOD_IRDA_CTRL_EN	BIST enable signal	0x0
13	MBIST_SECROM_CTRL_EN	BIST enable signal	0x0
12	MBIST_SECRAM_CTRL_EN	BIST enable signal	0x0
11	MBIST_FRAMBUF_CTRL_EN	BIST enable signal	0x0
10:8	MBIST_UART_MOD_CTRL_EN	BIST enable signal	0x0
7	MBIST_CAMERA_IF_CTRL_EN	BIST enable signal	0x0
6	RATIO_EN	0: Ratio value comes from JTAG or static value from input pad 1: Ratio value comes from configuration	0x0
5:4	RATIO_DOMAIN_MVE_2MB_RAM	BIST clock divider for GSM internal RAMs (2Mb + 0.5M b) 00: Divide by 1 01: Divide by 2 10: Divide by 4 11: Divide by 8	0x00
3:2	RATIO_DOMAIN_MPU_GSM	BIST clock divider for MPU peripherals and GSM. 00: Divide by 1 01: Divide by 2 10: Divide by 4 11: Divide by 8	0x00
1:0	RATIO_DOMAIN_OMAP	BIST clock divider for OMAP and its related peripherals. 00: Divide by 1 01: Divide by 2 10: Divide by 4 11: Divide by 8	0x00

Table 4–84. Boot Procedure Register (BOOT_ROM_REG)

Bit	Name	Function	R/W	Reset Value
31:16	GP_JTAG_REG	Status field for ROM and secure RAM tests	W	0x0000
15:12	UNUSED	Unused	R	0x0
11:9	TEST_SELECTION_INTERNAL	Test selection:· 0x0: Bypass branch (always with DPLL OFF) 0x1: BURNIN code (always with DPLL OFF) 0x2: TEST1 0x3: TEST2	R	0x0
8	DPLL_CONFIG_INTERNAL	DPLL configuration 0x0: DPLL OFF (DPLL bypassed) 0x1: DPLL ON. (x 10 → for ROMed tests only)	R	0x0
7:0	Perseus2_TI_TEST_SEL_INTERNAL	ROMed test cases, LED/EMIFS and selection: 0x00: OMAP730 BOOT ROM 0xA5: TI_TEST function selection	R	0x00

BOOT_ROM_REG contains the status for ROM and secure RAM tests.

Table 4–85. Secure Chip Register (PRODUCTION_ID_REG)

Bit	Name	Function	R/W	Reset Value
31:26	UNUSED	Unused	R	0x000
25	VBOX_EN	0: DFT read, DFT write values come from eFuse 1: DFT read, DFT write values are set by software	R/W	0x0
24:9	FUSE_COMPARE_REG_INTERNAL	This register is always set to 0x5555 by eFuse. If this value is identical to the reference value, the INITZ sequence is OK; if not, a signal blocks all the chip by setting the ULPD out reset to 0.	R	0x5555
8	PROTECT_CTLSECUREDATA_INTERNAL	Prevents reprogramming of the eFuses Gating of e-FUSE data in and data out	R	0x0
7	DFT_WRITE_OMAP	Default DFT_WRITE OMAP value	R/W	0x0
6	DFT_READ_OMAP	Default OMAP DFT_READ value	R/W	0x0
5	DFT_WRTE_MGS3	Not used on OMAP730	R	0x0

Table 4–85. Secure Chip Register (PRODUCTION_ID_REG) (Continued)

Bit	Name	Function	R/W	Reset Value
4	DFT_READ_MGS3	Not used on OMAP730	R	0x0
3:0	TST_DEVICE_TYPE	0000: Not programmed 0001: Normal 0010: Normal 0011: Normal 0100: Emulator 0101: Bad 0110: Bad 0111: Bad 1000: Emulator 1001: Bad 1010: Bad 1011: Bad 1100: Emulator 1101: Bad 1110: Bad 1111: Bad	R	0x0

The secure chip register determines whether the chip is secure or not.

Table 4–86. Secure ROM Signature Register 1 (BIST_SECROM_SIGNATURE1_INTERNAL)

Bit	Name	Function	Reset Value
31:0	BIST_SECROM_SIGNATURE1_INTERNAL	Secure ROM signature in BIST mode	0x0

The secure ROM signature register 1 contains the value of the secure ROM signature in BIST mode.

Table 4–87. Secure ROM Signature Register 2 (BIST_SECROM_SIGNATURE2_INTERNAL)

Bit	Name	Function	Reset Value
31:0	BIST_SECROM_SIGNATURE2_INTERNAL	Secure ROM signature in BIST mode	0X0

The secure ROM signature register 2 contains the value of the secure ROM signature in BIST mode.

Table 4–88. BIST Settings Control Register (BIST_CONTROL_2)

Bit	Name	Function	Reset Value
31:9	UNUSED	Unused	0x00000
8	MBIST_HOLD	Signal used for scan purposes	0x0
7:6	MBIST_SETUP	Signals used for IDDQ purposes	0x00
5:4	MBIST_ALGO_MODE	Signals used for IDDQ purposes	0x00
3	MBIST_TCK_MODE	Use TCK clock for data logging purposes Active at 1	0x0

Table 4–88. BIST Settings Control Register (BIST_CONTROL_2) (Continued)

Bit	Name	Function	Reset Value
2	MBIST_DIAG_EN	Mux COMPSTAT functionality on BIST_GO signals (this feature depends on the type of BIST controller) Active at 1	0x0
1	MBIST_DL_EN	Enable data logging logic Active at 1	0x1
0	MBIST_RST_MEM	Filler 0 of the BIST controller Active at 1	0x0

*Table 4–89. Debug Signal Selection Register (*DEBUG1*)*

Bit	Name	Function	Reset Value
31:28	OBS_MUX7	Selection of debug signals: 0000: Mode0 1011: Mode11	0x0
27:24	OBS_MUX6	Selection of debug signals: 0000: Mode0 1011: Mode11	0x0
23:20	OBS_MUX5	Selection of debug signals: 0000: Mode0 1011: Mode11	0x0
19:16	OBS_MUX4	Selection of debug signals: 0000: Mode0 1011: Mode11	0x0
15:12	OBS_MUX3	Selection of debug signals: 0000: Mode0 1011: Mode11	0x0
11:8	OBS_MUX2	Selection of debug signals: 0000: Mode0 1011: Mode11	0x0
7:4	OBS_MUX1	Selection of debug signals: 0000: Mode0 1011: Mode11	0x0
3:0	OBS_MUX0	Selection of debug signals: 0000: Mode0 1011: Mode11	0x0

*Table 4–90. Debug Signal Selection Register (*DEBUG2*)*

Bit	Name	Function	Reset Value
31:20	RESERVED	Reserved	0x0
19:16	OBS_MUX12	Selection of debug signals: 0000: Mode0 1011: Mode11	0x0
15:12	OBS_MUX11	Selection of debug signals: 0000: Mode0 1011: Mode11	0x0
11:8	OBS_MUX10	Selection of debug signals: 0000: Mode0 1011: Mode11	0x0
7:4	OBS_MUX9	Selection of debug signals: 0000: Mode0 1011: Mode11	0x0
3:0	OBS_MUX8	Selection of debug signals: 0000: Mode0 1011: Mode11	0x0

*Table 4–91. DMA and IRQ Selection Register (*DEBUG_DMA_IRQ*)*

Bit	Name	Function	Reset Value
31:22	RESERVED	Reserved	0X0
21:16	OBS_IRQ2_SEL	Selection of 1 IRQ (level 2) out of 64 for monitoring: 000000: IRQ 0 selected 111111: IRQ 63 selected	0x0
15	RESERVED	Reserved	0x0
14:10	OBS_IRQ1_SEL	Selection of 1 IRQ (level 1) out of 32 for monitoring: 00000: IRQ 0 selected 11111: IRQ 31 selected	0x0
9:8	RESERVED	Reserved	0x0
7:4	OBS_DMA_REQ_TX_SEL	Selection of 1 DMA TX request out of 16 for monitoring: 0000: DMA TX request 0 selected 1111: DMA TX request 15 selected	0x0
3:0	OBS_DMA_REQ_RX_SEL	Selection of 1 DMA RX request out of 16 for monitoring: 0000: DMA RX request 0 selected 1111: DMA RX request 15 selected	0x0

4.8 Inputs/Outputs

Table 4–92 through Table 4–94 present the input/output list and description.

Table 4–92. General-Purpose Signals

Name	Bits	I/O	Description
CLK	1	I	Clock used to resynchronize MPU-S and GSM-S accesses to share registers
CLK_DIE_ID	1	I	32-kHz clock used to access die_id register
$\overline{\text{RESPWRON}}$	1	I	Reset signal, active low
SCAN_MODE	1	I	Scan mode set when high
SCAN_CLK_CLK	1	I	Clock used in scan mode
SCAN_CLK_RHEA_MPU	1	I	Clock used in scan mode
SCAN_CLK_RHEA_GSM	1	I	Clock used in scan mode

Table 4–93. MPU-S Signals

Name	Bits	I/O	Description
$\overline{\text{STROBE\_MPU}}$	1	I	TIPB bus strobe
CS_MPU	5	I	TIPB CS signal
AD_MPU	11	I	TIPB address bus
rNW_MPU	1	I	Read access if 1, else write access
DO_MPU	32	I	Data bus for write cycle. For use on $\overline{\text{STROBE\_MPU}}$ rising edge.
DI_MPU	32	O	Data bus for read cycle. Must be valid on $\overline{\text{STROBE\_MPU}}$ rising edge when $\overline{\text{READY\_MPU}}$ is set low.
$\overline{\text{READY\_MPU}}$	1	O	Configuration register ready to accept or to send data. Active low, evaluated on $\overline{\text{STROBE\_MPU}}$ rising edge.
$\overline{\text{OE\_MPU}}$	1	O	Set low when ICR drives the DI_MPU and $\overline{\text{READY\_MPU}}$ signals to a valid state.

Table 4–94. GSM-S Signals

Name	Bits	I/O	Description
$\overline{\text{STROBE\_GSM}}$	1	I	TIPB bus strobe
CS_GSM	5	I	TIPB CS signal
AD_GSM	11	I	TIPB address bus
RNW_GSM	1	I	Read access if 1, else write access
DO_GSM	32	I	Data bus for write cycle. For use on $\overline{\text{STROBE\_GSM}}$ rising edge.
DI_GSM	32	O	Data bus for read cycle. Must be valid on $\overline{\text{STROBE\_GSM}}$ rising edge when $\overline{\text{READY\_GSM}}$ is set low.

Table 4–94. GSM-S Signals (Continued)

Name	Bits	I/O	Description
nREADY_GSM	1	O	Configuration register ready to accept or to sent data. Active low, evaluated on $\overline{\text{STROBE\_GSM}}$ rising edge.
nOE_GSM	1	O	Set low when ICR drives the DI_GSM and $\overline{\text{READY\_GSM}}$ signals to a valid state.

4.9 MPU/GSM Shared Port

Please see section 4.5, *Intersystem Communication Register*.

Chapter 5

Clock Generation and Reset Module

This chapter discusses the clock generation and reset module (CLKM), which is part of the microprocessor unit (MPU) subsystem in the OMAP730 hardware engine platform.

Topic **Page**

5.1 Module Description

The clock generation and system reset module is part of the MPU subsystem in the OMAP730 platform. This module manages the clock generation modes for the microprocessor unit (MPU), the modem part connection, and various other subsystems (memory interface, system DMA controller, etc.). These clocks can be controlled by software from registers described in Section 4.5. It also monitors the system reset and initiates the reset sequences for each clock domain. Finally, it controls the power-saving modes and generates wake-up controls to the processors and peripherals.

Clock generation modes include:

- Programmable clocking mode (synchronous, synchronous scalable, and mixed modes)
- Programmable clock for different clock domains (MPU, modem part connection, and traffic controller (TC) clock domains)
- Programmable clock for different peripherals (internal liquid crystal display (LCD) controller and external MPU TIPB peripherals)
- Programmable low-frequency clocks (derived from input reference clock) to supply the internal MPU and DSP timers
- Fixed low-frequency clocks to supply watchdog timers for the MPU
- Direct memory access (DMA) clock request mechanism (provides DMA clock during data transfer only)

System reset includes:

- Global software reset
- Reset control for the MPU and external TIPB peripherals
- System and reset status monitoring

Power-saving modes and wake-up control includes:

- Programmable power-saving mode and idle mode controls for the MPU, the traffic controller, and their respective subdomains
- Power control for external device reset/power on (flash memory)
- Wake-up functions initiated by MPU interrupts and DMA requests (traffic controller and TIPB) in the idle mode
- Initiation of the wake-up sequence by external devices during the idle mode

5.2 Clock Generation

The clock domains in the OMAP730 hardware engine platform are synthesized by the digital phase-locked loops (DPLL). The DPLL input clock source (CK_REF) is externally supplied from the CLKIN pin.

The DPLL1 output frequencies are programmable and can be further divided down to provide clocks to the MPU, the modem part connection, and the TC domains. The MPU domain, the modem part connection domain, and the TC domain are clocked from DPLL1.

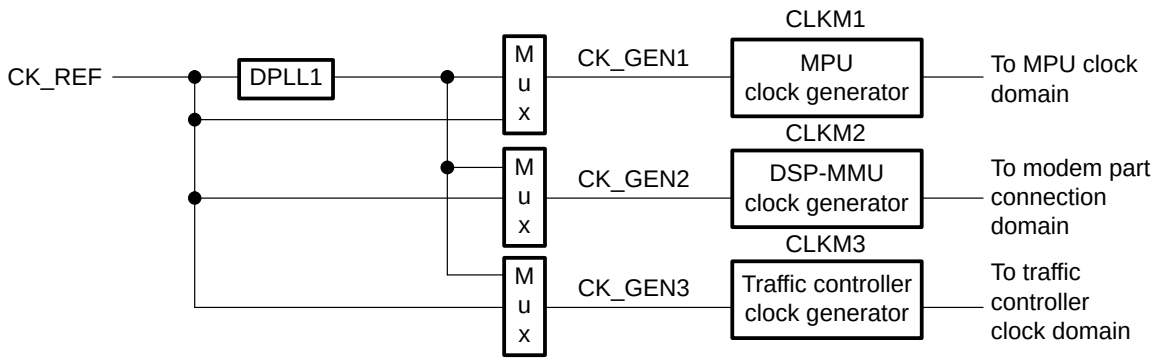
This implementation offers the clock rate selection flexibility to adjust the clock frequency of each clock domain and allows the OMAP730 hardware engine to adjust each clock domain to its optimal frequency. In addition, each domain is further subdivided into subdomains so that each subdomain can be independently activated/deactivated while the remaining part of the clock network is in an idle state. This clocking mode provides a scalable power-saving mechanism for general-purpose and power-hungry applications.

The OMAP clock system is organized around three main clock domains: MPU, modem part connection, and TC.

- The MPU clock domain contains:
  - MPU
  - MPU external peripheral clocks
  - MPU watchdog timer
  - MPU internal timers
  - MPU interrupt handler
- The modem part connection clock domain contains the DSP MMU.
- The TC clock domain contains:
  - TC
  - L3 OCP-I
  - MPUI port interface
  - System DMA controller
  - MPU TIPB bridges
  - LCD controller
  - OCP-T1 and OCP–T2

Figure 5–1 shows the clock generator module.

Figure 5–1. Clock Generator Module



5.2.1 Clock Generation Modes

The clock generation and system reset module of the OMAP730 hardware engine supports four kinds of clocking modes:

- Fully synchronous
- Synchronous scalable
- Bypass mode
- Mixed modes (3 and 4)

These clocking modes provide the system with the maximum flexibility for performance and power-saving capabilities. They are programmable by the CLOCK_SELECT field of the ARM_SYSST register, and the power-up default mode is the full synchronous mode.

Table 5–1 details the hardware engine clocking modes.

Table 5–1. OMAP730 Hardware Engine Clocking Modes

Clock Select	Clocking Operating Mode	MPU Clock Source	Modem Connection Clock Source	TC Clock Source	Remarks
000	Fully synchronous	DPLL1/N	DPLL1/N	DPLL1/N	See Section 5.2.1.1, *Fully Synchronous Mode.*
001	Reserved				
010	Synchronous scalable	DPLL1/M	DPLL1/N	DPLL1/O	See Section 5.2.1.2, *Synchronous Scalable Mode.*
011	Reserved				
100	Reserved				
101	Bypass mode	CK_REF	CK_REF	CK_REF	Input reference clock
110	Mixed mode 3: MPU synchronous to TC, modem connection synchronous scalable to TC and MPU	DPLL1/N	DPLL1/M	DPLL1/N	See Section 5.2.1.3, *Mixed Modes.*
111	Mixed mode 4: Modem connection synchronous to TC, MPU synchronous scalable to TC and modem connection	DPLL1/M	DPLL1/N	DPLL1/N	See Section 5.2.1.3, *Mixed Modes.*

5.2.1.1 Fully Synchronous Mode

In fully synchronous mode, the MPU, modem connection, and TC domains run at the same clock frequency derived from DPLL1. This is the power-up default mode. The fully synchronous mode is a special case of synchronous scalable mode, where the clock-divider bits for all domains are equal. However, there is separate clock select encoding for fully synchronous mode. It is the programmer's responsibility to ensure that all the clock-divider select bits are set to the same value.

When the fully synchronous mode is selected, you must program the divide-down bits of the ARM_CKCTL register so that ARMDIV, DSPMMUDIV, DSPDIV, and TCDIV are equal.

5.2.1.2 Synchronous Scalable Mode

In synchronous scalable mode, the MPU, modem connection, and TC domains are synchronous and run at different clock speeds. The clock feeding mechanism is similar to that of the fully synchronous mode, except that the clocks are multiples of one another.

In synchronous scalable mode, the divide-down bits ARMDIV, DSPDIV, and TCDIV of the ARM_CKCTL register define the prescaler value from the frequency of the DPLL.

When the synchronous scalable mode is selected, you must program the divide-down bits of the ARM_CKCTL register so that DSPMMUDIV = DSPDIV*2.

The TC clock frequency must be the same speed or slower than the MPU and DSP MMU clocks.

5.2.1.3 Mixed Modes

Clock generation supports two mixed modes:

- Mixed mode 3

  The MPU and TC clock domains are synchronous (same clock frequency), and the DSP is scaled synchronous (synchronous but with a frequency that is a multiple of the MPU/TC clock frequency). In this mode, the TC, MPU, and DSP receive clocks from the DPLL1 output.

  When mixed mode 3 is selected, you must program the divide-down bits of the ARM_CKCTL register so that ARMDIV = TCDIV.

  The TC clock frequency must be the same speed or slower than the DSP and the DSP MMU clock frequencies.

- Mixed mode 4

  The DSP and TC clock domains are synchronous (same clock frequency), while MPU is scaled synchronous (synchronous but with a frequency that is a multiple of the DSP/TC clock frequency). In this mode, the TC, MPU, and DSP receive clocks from the DPLL1 output.

When mixed mode 4 is selected, you must program the divide-down bits of the ARM_CKCTL register so that DSPDIV = DSPMMUDIV = TCDIV. Because ARM_CK supplies the host processor and TC_CK supplies different memory interfaces, the restriction on the speed of TC_CK ensures that the rate of instruction/data fetch is never more than the rate at which data can be processed.

The TC clock frequency must be the same speed or slower than the MPU clock frequency.

5.2.1.4 Bypass Mode

In bypass mode (CLOCK_SELECT = 101), the DPLL is bypassed and the input reference clock is directly fed to the MPU, DSP, and TC clock domains.

5.2.2 DPLL

The DPLL block synthesizes a frequency clock from the fixed reference input clock signal CK_REF using the digital phase-locked loop mechanism. Only the MPU can access the DPLL control register.

5.2.2.1 DPLL Modes

The DPLL can operate either in bypass mode or lock mode:

- Bypass mode

  In bypass mode (PLL_ENABLE bit of the DPLL1_CTL_REG register set to 0), the DPLL output clock can be CK_REF (input reference clock), CK_REF/2, or CK_REF/4, depending on the BYPASS_DIV bit-field value of the DPLL1_CTL_REG register.

- Lock mode

  In lock mode (PLL_ENABLE bit of DPLL1_CTL_REG register set to 1), the output frequency is an integer multiple or fractional multiple (m/n, respectively, in the PLL_MULT and PLL_DIV bit fields of DPLL1_CTL_REG) of the input reference clock CK_REF. With $1 \leq m \leq 31$ and $1 \leq n \leq 4$, the frequency output ranges from CK_REF/4 to 31×CK_REF.

5.2.2.2 Synthesizing a Clock

At reset, the DPLL is in bypass mode and the BYPASS_DIV bit field of DPLL1_CTL_REG is set to 0b00 (DPLL output clock = CK_REF).

The procedure to synthesize a clock at a desired frequency is:

1) Set the PLL_MULT and PLL_DIV bit fields of DPLL1_CTL_REG to the correct value to get the desired multiplication factor.

2) Set the PLL_ENABLE bit to 1 to enter the lock mode.

3) When the DPLL reaches the desired synthesized clock frequency, the bit LOCK bit of DPLL1_CTL_REG register goes to 1 and the output clock gets the synthesized clock.

The bit fields PLL_MULT and PLL_DIV can be modified on-the-fly even when the DPLL is in lock mode.

Polling can be done on the LOCK bit to determine when the DPLL locks on the desired synthesized frequency. The DPLL output clock is switched smoothly between bypass and locked frequency because it is not mandatory to wait for the DPLL to enter lock mode before running the DPLL output clock.

When idle mode of the DPLL is exited, the DPLL is set in bypass mode and the output signal is valid (locked) after a maximum of 10 input reference-clock cycles. The output is valid after a maximum of 12 input reference-clock cycles in bypass mode and switches to locked clock in another 32 maximum reference-clock cycles. If the DPLL was synthesizing a frequency prior to the idle state, the DPLL switches from bypass mode to synthesizer frequency when the lock state is reacquired.

5.2.3 MPU Clock Domain

The DPLL1 output frequency defines the speed of the MPU and the MPU external peripherals. The DPLL1 typically generates an optimal clock available within the OMAP730 hardware engine and provides a programmable clock (50% duty cycle) to the MPU and its peripherals. The 200-MHz range clock from DPLL1 output is supplied to the OMAP boundary. It has software gating in ARM_IDLECT1 (IDL_CLKOUT_ARM) and ARM_IDLECT2 (EN_CKOUT_ARM).

At reset, DPLL1 is in bypass mode (CK_GEN1 = CK_REF).

The MPU clock domain is divided into five subdomains:

- MPU (ARM_CK)

  You can program the divide-down ARMDIV bits of the ARM_CKCTL register to have the DPLL1 output clock (CK_GEN1) further divided by 1, 2, 4, or 8 to supply the clock signal driving the MPU. At reset, the highest frequency (divided by 1) is selected: ARM_CK = CK_GEN1 = CK_REF.

- MPU external peripheral (ARMPER_CK or ARMXOR_CK)

  You can program the divide-down PERDIV bits of the ARM_CKCTL register to have CK_GEN1 further divided by 1, 2, 4, or 8 to supply the MPU external peripheral clock ARMPER_CK signal at the OMAP boundary. At reset, the highest frequency (divided by 1) is selected but ARMPER_CK is inactive. ARMXOR_CK, a gated version of CK_REF, can also be used to supply the external peripherals. At reset, this clock is inactive.

- MPU internal timers (ARMTIM_CK)

  The ARM_TIMXO bit of the ARM_CKCTL register selects either CK_GEN1 divided by 1 or the input reference clock (CK_REF) to supply the internal MPU timers. At reset, CK_GEN1 is selected but the timer clock is inactive.

- MPU interrupt handler (ARM_INTH_CK)

  The MPU interrupt handler is supplied with a programmable clock, and the user can choose between the MPU clock or the divided-by-2 MPU clock using the ARM_INTHCK_SEL bit of the ARM_CKCTL register. The MPU clock is supplied as the default clock.

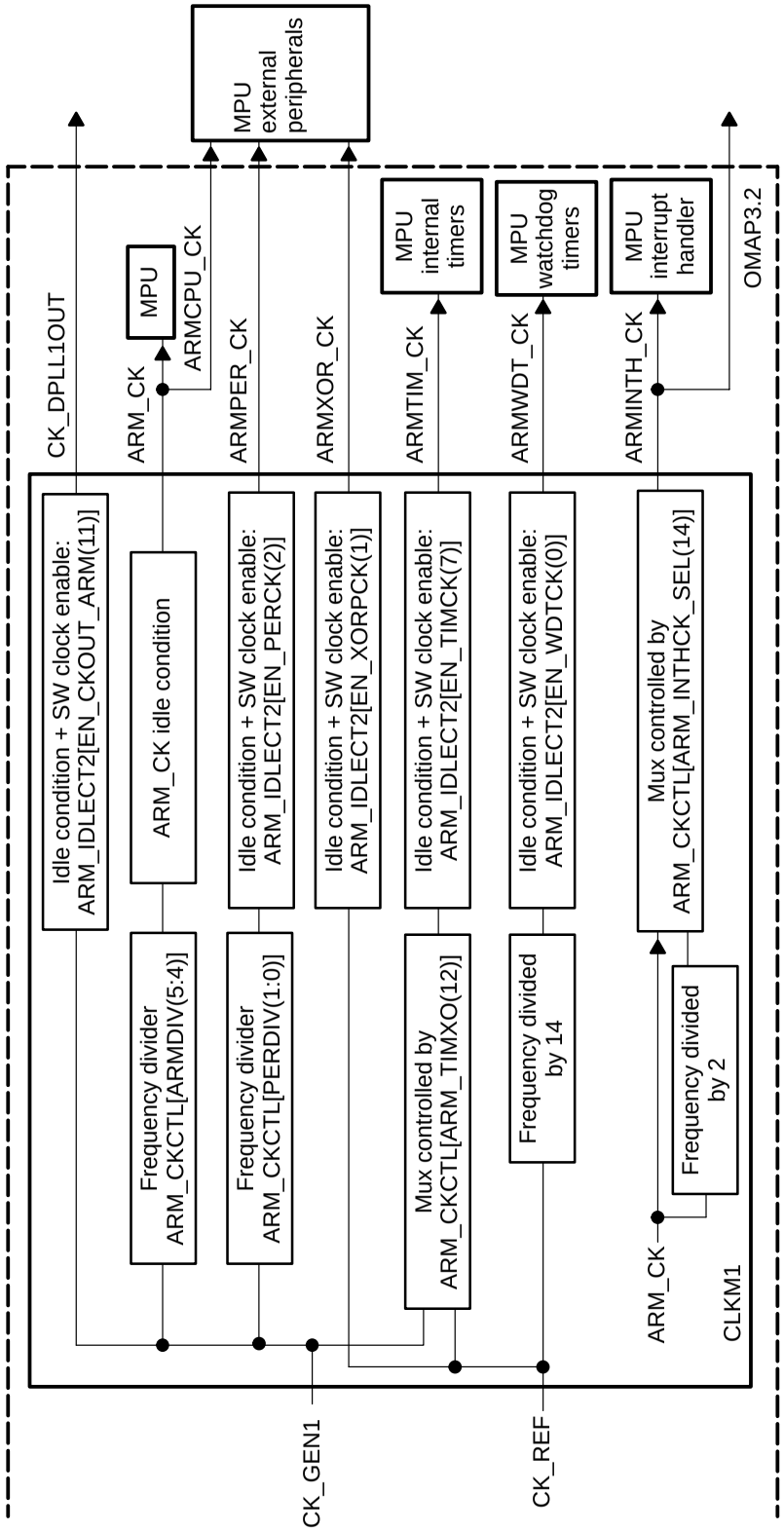
- MPU watchdog timer (ARMWDI_CK)

  The MPU watchdog timer is supplied with a low-frequency clock (CK_REF/14). This clock is active at reset.

Even when the MPU is not in idle mode, you have the option of individually disabling the clock to the MPU subdomains via the ARM_IDLECT2 register. This allows power saving when a module is not used.

Figure 5–2 shows the CLKM1.

Figure 5–2. CLKM1



5.2.4 Modem Connection Clock Domain

Depending on the OMAP clocking mode, the DPLL1 output frequency defines the speed of the modem connection domain. The clock output from DPLL1 (CK_GEN2) can be further divided to supply the modem connection clock. At reset, DPLL1 is in bypass mode (CK_GEN1 = CK_GEN2 = CK_REF).

The modem connection clock domain includes:

- DSP MMU (DSPMMU_CK)

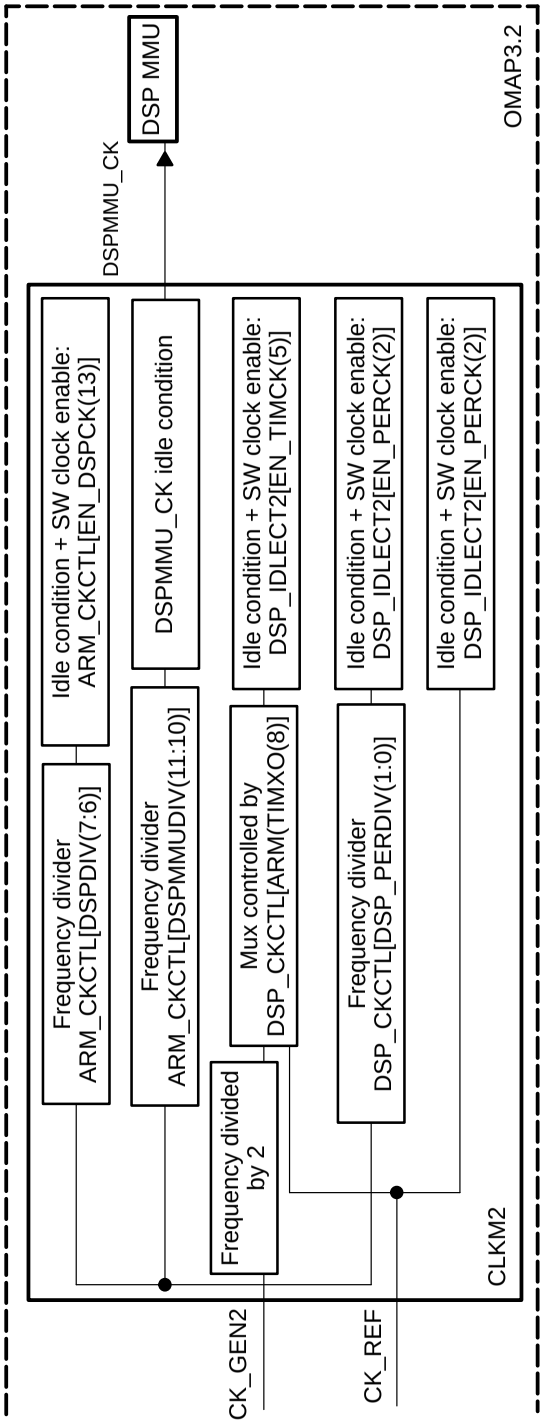
  The clock signal driving the DSP MMU (DSPMMU_CK) can be further divided by 2, 4, or 8 by programming the divide-down bits DSPMMUDIV of the ARM_CKCTL register. At reset, the highest frequency (divided by 1) is selected, but the DSP MMU clock is inactive.

Note:

The DSPDIV and DSPMMUDIV must be programmed to have: DSPDIV = ARMDIV, DSPMMUDIV = TC_DIV, and DSPMMUDIV $\geq 2 \times$ DSPDIV

Figure 5–3 shows the CLKM2.

Figure 5–3. CLKM2



5.2.5 Traffic Controller Clock Domain

The DPLL output frequency, which drives the traffic controller, generates the traffic controller clock (TC_CK). This TC_CK feeds:

- Traffic controller
- OCP initiator port (OCP-I)
- OCP Target1 (OCP-T1) and OCP Target2 (OCP-T2) ports
- System DMA controller
- LCD controller
- MPUI port interface
- MPU TIPB bridge

TC1_CK and TC2_CK are then broadcast outside the OMAP platform. They are identical to TC_CK and can be powered down independently in power-saving options.

The TC clock domain is divided into:

- First subdomain: Traffic controller, OCP-I port, OCP-T1 and OCP-T2 ports, MPUI port interface, system DMA controller, and MPU TIPB bridges

  The clock signal driving these modules is basically the same as the TC_CK, except that it can be gated independently of TC_CK.

  You can program the divide-down TCDIV bits of the ARM_CKCTL register to have the CK_GEN3 further divided by 2, 4, or 8 to generate the TC_CK. At reset, the highest frequency (divided by 1) is selected and TC_CK = CK_REF.

  At reset, the OCP-I port clock, MPUI port interface clock, system DMA controller clock, and OCP-T1 and OCP-T2 port clocks are inactive. The TC clocks and MPU TIPB bridge clocks are active.

- Second domain: LCD controller

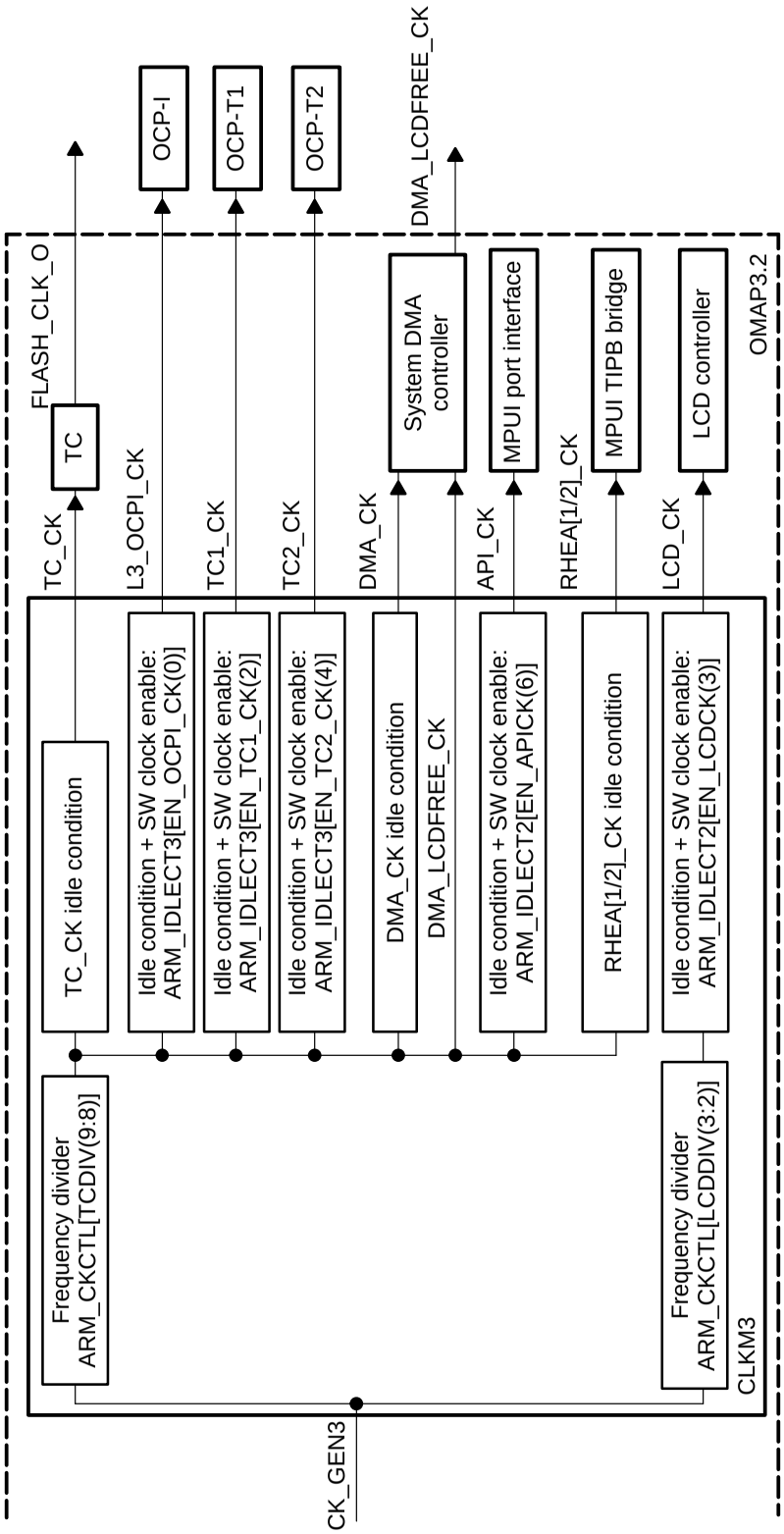
  You can program the divide-down LCDDIV bits of the ARM_CKCTL register to have CK_GEN3 further divided by 2, 4, or 8 to generate the LCD controller clock. At reset, the highest frequency (divided by 1) is selected, but the LCD controller clock is inactive.

These traffic-controller subdomain clocks can be disabled using the ARM_IDLECT2 register, even if the MPU, DSP, or TC are not in idle mode .

The DMA requires a free-running clock to be supplied to the external LCD controller even when the DMA clock is turned off so that the LCD controller can generate proper interrupts. This free-running clock for external LCD controller can only be cut off when the external LCD controller is in idle state.

Figure 5–4 shows the CLKM3.

Figure 5–4. CLKM3



Notes: 1) Rhea1_CK is used for the internal TIPB bridge, and Rhea2_CK is used for the external TIPB bridge.

2) *: FLASH_CLK_0 comes from the traffic controller and is derived from TC_CK divided by the FCLKDIV bit of the CS configuration register.

5.3 Power-Saving Modes and Wake-Up Control

This section describes the following power-saving features:

- MPU idle control
- DSP idle control
- Traffic controller idle control
- System DMA idle control
- MPU TIPB bridge idle control
- External device power control
- DPLL idle control
- Chip idle mode/deep sleep mode
- Wake-up control

5.3.1 MPU Idle Control

The OMAP730 hardware engine can operate in several power-saving modes that can reduce the operating current by stopping the clock signals of unused (inactive) domains without losing any data on operational context. When the idle state is entered, the MPU domain clocks are turned off according to the orderly sequenced events. The clock gating cell design ensures that clocks are properly stopped and restarted without parasitic pulses.

Activating the wait-for-interrupt MPU instruction initiates the MPU idle mode. It stops the MPU internal clocks, and then the STANDBYWFI signal from MPU megacell is asserted high, indicating that the MPU internal idle state is entered.

Before the idle mode is entered, you can stop the MPU internal timer clock, the LCD clock, the external peripheral clock, and the timer/watchdog clock by setting to 0 the corresponding bits of the ARM_IDLECT2 register, or you can set the corresponding bits of ARM_IDLECT1 register so that these peripherals automatically go to idle when the MPU goes to idle (except for the LCD).

Note that when the timer/watchdog timer is configured as a watchdog, its clock (CK_REF/14) is never shut down, regardless of the value of the IDLWDT_ARM bit in the ARM_IDLECT1 register, or the EN_WDTCK bit in the ARM_IDLECT2 register.

The idle command is forwarded to the MPU interrupt handler via the ARM_INTH_IDLE_REQ signal, and the MPU clock is stopped when receiving the acknowledge ARM_INTH_IDLE_ACK signal from the interrupt handler. Idle request is based on STANDBYWFI input and no pending interrupts.

When the MPU internal clocks are stopped, the MPU domain clocks are stopped if they were not already disabled using ARM_IDLECT2 before MPU went to idle.

5.3.2 Traffic Controller, System DMA Controller, and MPU TIPB Bridges Idle Control

Specific conditions must be met for the traffic controller, the system DMA controller, and the MPU TIPB bridges to enter the idle mode.

5.3.2.1 Traffic Controller Idle Control

To enter idle mode, the traffic controller must meet the following conditions:

- The MPU and modem connection must be set to global idle mode.
- L3 OCP-I is in idle mode. This can be done either by clearing the EN_OCPI_CK bit of the ARM_IDLECT3 register to 0, or by setting the IDLOCPI_ARM bit in the same register to 1 (which allows disabling of the OCPI clock in conjunction with the MPU clock). The OCP initiator bus enable signal (L3_OCPI_EN) must always be 0 before the OCPI clock stops.
- OCP-T1/T2 modules are in idle mode. This can be done by setting ARM_IDLECT3 (IDLTC1_CK) = 1 and ARM_IDLECT3 (IDLTC2_CK) = 1, which disables the TC1_CK and TC2_CK clocks when there is no activity and the idle request is acknowledged by the target. These modules can also be placed in idle mode by disabling TC1_CK and TC2_CK completely by setting EN_TC1_CK and EN_TC2_CK to 0.
- The idle interface (IDLIF_ARM) bit of the ARM_IDLECT1 register is set to logical 1.
- The MPUI clock is idle.
- There are no system DMA pending transfers.
- All the TC subdomains are stopped by setting the appropriate bits in the ARM_IDLECT2 and ARM_IDLECT3 registers.
- There are no MPU interrupt requests.
- Power-down enable bits PDE and PWD_EN of the EMIFS configuration register EMIF_SLOW_CONFIG are set to logical 1.
- Disable the SDRAM clock and set the power-down enable bit to 1 with the EMIFF configuration register. TC idle mode entry is affected by the RFRSH_STBY bit of the EMIFF_SDRAM_CONFIG_2REG TC register. When set to 1, the SDRAM must be put into self refresh state before going idle (SLRF bit of EMIFF_CONFIG register). Every time the SDRAM wakes up, the self-refresh state is cleared, so returning to idle requires that the SLRF bit of EMIFF_CONFIG_REG is set to 1 again.

Then the traffic controller completes its current operations and pulls the TCIDLE_ACK signal to a high level to indicate that TC_CK can now be safely stopped. In addition, the shut-down of the TC_CK clock indicates to the chip idle control logic to initiate the DPLL idle.

The TC_CK restarts upon:

- An MPU or modem connection interrupt request
- A DMA request
- L3_OCPI_EN pin set to logic 1 (enables restarting of the clock to L3 OCP initiator bus)
- DMA_LCD FREECK_NREQ pin set to logic 0 (system DMA request for a free running clock to be supplied to the external LCD controller)

5.3.2.2 System DMA Idle Control

The system DMA employs a built-in power-saving mechanism. The clock is only requested to the clock generator when DMA transfers are occurring.

The DMA clock can enter idle mode if one of the following conditions is true:

- ❏ DMACK_REQ = 1 and there are no DMA requests.
- ❏ DMACK_REQ = 0, the MPU clock is in idle, IDLIF_ARM = 1, there are no DMA requests, and the DMAIDLE_ACK signal is high.

5.3.2.3 MPU TIPB Bridges Idle Control

The TIPB bridges can enter idle mode only when all of the following conditions are true:

- ❏ MPU is set in idle mode.
- ❏ The idle interface bit IDLIF_ARM of ARM_IDLECT1 register is set to logical 1.
- ❏ There are no system DMA requests to the TIPB.
- ❏ There is no posted write (that is, posted write buffers are empty).

5.3.3 External Device Power Control

The RESPWR signal is an output pin that allows the reset/power-on control sequences of external devices such as flash memory.

Whenever the traffic controller enters the idle mode, the RESPWR pin switches from a high to a low level, allowing external components to be turned off. When a wake-up condition is detected, the pin is switched back to a high level and restores power to external devices.

Setting the bit REPWR_EN of ARM_EWUPCT to logical 0 enables this capability. At reset, this is disabled.

To allow the external device/component voltage to stabilize (ramp-up) when the power-down mode is released, the external power control is implemented with a programmable counter that delays the restart of all clocks from RESPWR signal going high. The EXTPWR bit field of the ARM_EWUPCT register permits the delay to be defined as:

$$WT_{(\text{wake-up time})} = (EXTPWR_{(\text{field value})} \pm 1) \times CK\_REF_{(\text{period})}$$

5.3.4 DPLL Idle Control

The DPLL can be set to idle mode if only the input reference clock (CK_REF) is needed.

The DPLL idle mode is entered when the IDLDPLL_ARM bit of the ARM_IDLECT1 register is set to logical 1 and all of the domains that use the DPLL clock are stopped. This means that the only domains running (domains that use CK_REF rather than the DPLL clock) are:

- MPU watchdog timer (CK_REF/14)
- Internal MPU timers when ARM_TIMXO bit of the ARM_CKCTL register is set to logical 0

The DPLL idle mode entry/exit time can be significant. The input reference clock must be active for at least 24 input clock cycles from the idle request (idle rising edge) before the idle setup is complete. Once the idle mode is exited, the DPLL is set in bypass mode and the output signal is valid after a maximum of 10 input reference clock cycles. The total DPLL idle entry-to-exit sequence takes no less than 34 reference clock cycles. Therefore, it may be preferable not to shutdown the DPLL when the MPU must be stopped for a short period of time or when critical operations are likely to occur (that is, DMA transfer, interrupt handling).

5.3.5 Chip Idle Mode, Deep Sleep Mode, and Wake-up Control

The OMAP730 hardware engine is considered to be in chip idle mode when the MPU, modem connection, peripherals, DPLL, and peripherals using CK_REF as their source are stopped. Then the external reference clock (CLKIN) can be stopped as well.

The signal at the CHIP_IDLE pin is asserted high when all internal system clocks are disabled and after the DPLL idle state has been acknowledged. It is deasserted when a wake-up condition is detected. It takes some synchronization CK_REF clock cycles for CHIP_IDLE to go low after a wake-up condition is detected.

Once the procedures for MPU idle, DSP idle, traffic controller idle, and DPLL idle have been followed, the chip idle state is reached and the CHIP_IDLE pin goes active 1. The following ordering is recommended to ensure that all of the clock domains can be made idle:

- Disable the MPU watchdog timer.
- Set the ARM_IDLECT1, ARM_IDLECT2, and ARM_IDLECT3 register bits in preparation for going to idle (MPU, DSP, TC, and DPLL idle entries are all affected by these registers).
- Configure the EMIFS and EMIFF modules as described in the traffic controller idle control section. If SDRAM contents must be maintained during the idle state, then the SDRAM self-refresh mode must be enabled before going to chip idle (set the SLFR bit of the EMIFF_SDRAM_CONFIG register to 1, and the RFRSH_STDBY bit

of the EMIFF_SDRAM_CONFIG2 register to 1). The self-refresh bit is cleared every time OMAP leaves chip idle.

- Prepare for wake up by enabling and unmasking MPU interrupts.
- Ensure that all interrupts and DMA status bits have been cleared. If all of the other idle conditions and controls have been met (as per the MPU, DSP, TC and DPLL descriptions), then activating the wait-for-interrupt instruction at this point leads to the full chip idle state, allowing the reference clock to be stopped.

In chip idle mode, the MPU, the DPLL, and peripherals that use CK_REF as their source are stopped, while the external clock source remains the only active clock signal.

The output CHIP_IDLE signal, available at the package pin, indicates that the internal system clock is not needed. It can be used externally along with the WKUP_nREQ output to switch off/on the reference clock source.

In deep sleep mode, all internal system clocks (MPU, DPLL, peripherals, and timers) and the external reference clock source are stopped, leaving the OMAP3 in a static state in which it consumes the lowest possible power. In this mode, it is recommended that the WKUP_MODE bit of the ARM_IDLECT1 register be set to 0 before going into IDLE. A complete handshake between the OMAP and external module turns off the OMAP input clock. This handshake ensures that OMAP wakes up properly.

Any unmasked interrupt request, any DMA clock request, or setting the L3_OCPI_EN signal to high exits the idle mode.

When the WKUP_MODE bit of ARM_IDLECT1 is set to logical 0, the wake-up procedure can be controlled from an external source/device via the CHIP_IDLE, WKUP_nREQ, and CHIP_nWKUP pins.

To enable the CHIP_nWKUP pin and allow external wake-up control, the bit WKUP_MODE must be cleared to 0 before entering the idle mode, the CHIP_IDLE pin must be asserted high, and the WKUP_nREQ pin must be asserted low.

When the WKUP_MODE bit value is set to logic 1, a single wake-up condition initiates a chip wake-up procedure. The wake-up condition can be caused by:

- An interrupt request from MPU interrupt handler. The MPU interrupt handler sets the nIRQ_SET signal to logic low and initiates the restarting of the ARM_CK, ARM_INTH_CK, RHEA_CK, DMA_CK, and TC_CK clocks. Depending on the setting of the ARM_IDLECT1/2 registers, peripherals clocks can also restart. This is a valid wake-up condition for MPU, TC, and DPLL.
- L3_OCPI_EN pin: When the L3_OCPI_EN pin is pulled high, the TC_CK, TC1_CK, and TC2_CK, and the L3_OCPI_CK restart and the TC_CK, L3_OCPI_CK, TC1_CK, and TC2_CK keep running as long as the pin remains asserted high. This is a valid wake-up condition for TC and DPLL only.

- TCLB_DMAREQ: When the system DMA controller receives an asynchronous request from the traffic controller, this signal is set high to enable the DMA_CK/TC_CK and DMA_CK/TC_CK to keep running as long as the DMA operates. This is a valid wake-up condition for TC and DPLL only.
- RHEA_DMAREQ: When the system DMA controller receives a request from the TIPB-bridge, this signal is asserted high to enable the TC_CK/Rhea_CK/DMA_CK and the TC_CK/RHEA_CK/DMA_CK to keep running as long as the DMA operates. This is a valid wake-up condition for TC and DPLL only.

5.4 System Reset

The system reset function ensures an orderly start-up sequence for the OMAP 3.2-based system.

The reset sequences for the MPU, the MPU subsystem, and the external peripherals are separately controlled. The reset control logic permits each of the clock domains or subdomains to be switched to a reset state while other parts are still running. The system reset input pin (CHIP_nRESET_IN) initializes the OMAP730 hardware engine, and the system reset output pin (CHIP_nRESET_OUT) indicates internally generated global system resets to the external system.

There are three causes of a system reset:

- External CHIP_nRESET_IN pin reset. A logic low at the CHIP_nRESET_IN pin causes a global system reset. An active-low level of 20 CK_REF cycles is required to guarantee a proper reset sequence. It can be pulled low at any time during the operation to start the reset sequence.
- MPU watchdog timer reset

  Assuming that the MPU timer/watchdog is configured as a watchdog timer, the reset is generated when the down-counter underflows.
- Software-generated reset

  System reset is initiated when the SW_RST bit of the ARM_RSTCT1 register is set to logical 1.

CHIP_nRESET_IN, MPU watchdog reset, and software reset are warm resets; external power-on reset nPOR is a cold reset. A warm reset differs from a cold reset in that the warm reset affects neither the SDRAM controller (EMIFF) nor the window tracers.

The CHIP_nRESET_OUT pin provides the visibility of the internal global system reset. The reset signal output can be used to initialize the external system. The reset circuitry drives the reset pin to a low level for 64 clock cycles after reset goes inactive.

After a system reset, reset is released for the MPU, internal OMAP peripherals, and all external peripherals (both MPU and DSP) controlled by external pins.

There are nine reset signals. All generated reset signals are active low and asynchronously asserted. Reset signals are released on the rising edge of the CK_REF clock.

Depending on the domain being controlled, the reset period is either automatically or software controlled. The automatic process holds the reset signals in an active state for 64 input clock cycles. The counter starts after the reset deassertion is detected. The other software-controlled signals are described below:

- COLD_nRST: This signal is asserted low when the nPOR (power-on reset) is asserted low at the chip boundary. The assertion of this signal is asynchronous but the deassertion is synchronous.

- WARM_nRST: This signal is asserted low when a system reset occurs.
  - External CHIP_nRESET_IN pin reset
  - MPU watchdog timer reset
  - Software-generated reset

The COLD_nRST and WARM_nRST signals have been provided to differentiate between the power-on reset and the resets from other sources.

- GLOBAL_nRST: This signal supplies the LCD controller, the system DMA controller, the MPU port interface, the window tracer, the L3 OCP initiator, the L4 controller, the traffic controller, the DSP MMU, the CLK and RESET, the MPU TIPB bridge and peripherals, and shared peripherals including mailboxes. The GLOBAL_nRST signal is set to its active low state upon either COLD_nRST or WARM_nRST.

  This signal is released to its inactive high state after 64 input clock cycles reset period delay.

- ARM_nRST: This signal controls the reset of the MPU. The ARM_nRST signal is set to its active low state:
  - Upon an external CHIP_nRESET_IN reset
  - When a watchdog reset is asserted (MPU controlled)
  - When a software global reset is issued
  - When the ARM_RST bit in the ARM_RSTCT1 control register is written to a logical 1

  This signal is released to its inactive high state after the 64 cycles delay.

- ARMPER_nRST: This signal, available at the OMAP3 boundary pins, is intended to provide reset to the external peripherals. The ARMPER_nRST signal is set in an active low state:
  - Upon an external CHIP_nRESET_IN reset
  - When a watchdog reset is asserted (MPU controlled)
  - When a software global reset is issued
  - When the corresponding peripheral-enable bits are set to a logical 0
    - ARM_PEREN bit in the ARM_RSTCT2 control register

  These signals are released to the inactive high state when the peripheral enable bit ARM_PEREN is written to a logical 1.

After a reset, the MPU can determine the causes of reset via reset status bit ARM_SYSST in the system status register (ARM_SYSST).

5.5 Registers

All registers are 16-bit registers, 32-bit accessed, and 32-bit aligned. MPU clock generation and system reset control registers are accessed by the MPU only.

5.5.1 MPU Registers

The MPU registers are listed in Table 5–2. Table 5–3 through Table 5–12 provide register bit descriptions.

Table 5–2. MPU Registers

Register Name	Offset
ARM_CKCTL	0x00
ARM_IDLECT1	0x04
ARM_IDLECT2	0x08
ARM_EWUPCT	0x0C
ARM_RSTCT1	0x10
ARM_RSTCT2	0x14
ARM_SYSST	0x18
ARM_CKOUT1	0x1C
ARM_CKOUT2	0x20
ARM_IDLECT3	0x24

Table 5–3. MPU Clock Control Prescaler Selection Register (ARM_CKCTL)

Bit	Name	Function	R/W	Reset Value
15	RESERVED	Reading this bit gives undefined values. Writing to it has no effect.	R/W	0
14	ARM_INTHCK_SEL	This bit controls which clock is used for the ARM_INTH_CK 0: ARM_INTH_CK clock is same as ARM_CK (default). 1: ARM_INTH_CK is half frequency of ARM_CK.	R/W	0
13	RESERVED	Must be kept at 1	R/W	1
12	ARM_TIMXO	Selects a subfrequency issued either from CK_GEN1 or from input reference clock (CK_REF) to supply internal MPU timers. 0: ARMTIM_CK clock frequency is the input reference clock (CK_REF). 1: ARMTIM_CK clock frequency is issued from CK_GEN1.	R/W	1

Table 5–3. MPU Clock Control Prescaler Selection Register (ARM_CKCTL) (Continued)

Bit	Name	Function	R/W	Reset Value
11:10	DSPMMUDIV	Define prescaler value from the frequency of CK_GEN2 to DSPMMU clock domain. 00: CK_GEN2 01: CK_GEN2/2 10: CK_GEN2/4 11: CK_GEN2/8	R/W	00
9:8	TCDIV	Define prescaler value from the frequency of CK_GEN3 to TC clock domain. 00: CK_GEN3 01: CK_GEN3/2 10: CK_GEN3/4 11: CK_GEN3/8	R/W	00
7:6	DSPDIV	Define the prescaler from the frequency of CK_GEN1 to MPU clock domain.00: CK_GEN1 01: CK_GEN1/2 10: CK_GEN1/4 11: CK_GEN1/8	R/W	00
5:4	ARMDIV	Define the prescaler value for the DSP clock domain. 00: CK_GEN2 01: CK_GEN2/2 10: CK_GEN2/4 11: CK_GEN2/8	R/W	00
3:2	LCDDIV	Define prescaler value from the frequency of CK_GEN3 to LCD controller clock signal 00: CK_GEN3 01: CK_GEN3/2 10: CK_GEN3/4 11: CK_GEN3/8	R/W	00
1:0	ARM_PERDIV	Define the prescaler value from the frequency of CK_GEN1 to MPU external peripheral clock domain. 00: CK_GEN1 01: CK_GEN1/2 10: CK_GEN1/4 11: CK_GEN1/8	R/W	00

Table 5–4. MPU Idle Enable Control Register 1 (ARM_IDLECT1)

Bit	Name	Function	R/W	Reset Value
15:13	RESERVED	Reading these bits gives undefined value. Writing to them has no effect.	R/W	000
12	IDL_CLKOUT_ARM	This read-write bit selects the idle entry mode for the external DPLL output clock. 0: The clock supplied to the external DPLL output clock remains active when the MPU enters the idle mode (ARM_CK stopped). 1: The clock supplied to the external DPLL O/P clock is stopped in conjunction with the MPU clock when the MPU enters the idle mode (ARM_CK stopped).	R/W	0
11	RESERVED	Reading this bit gives undefined value. Writing to it has no effect	R/W	0
10	WKUP_MODE	Controls how the MPU can exit the CHIP_IDLE state 0: After the interrupt has been asserted, the MPU idle mode is exited upon a low level at the external CHIP_nWKUP pin. Also, any of the wake-up conditions only wake up the OMAP out of CHIP_IDLE if CHIP_nWKUP is low. 1: Idle mode is exited upon an MPU interrupt (regardless the CHIP_nWKUP pin). Also, any wake-up condition wakes up the OMAP out of CHIP_IDLE regardless of the value on the CHIP_nWKUP pin.	R/W	1
9	IDLTIM_ARM	Selects the idle entry mode for internal MPU timer clock. 0: The clock supplied to the timers remains active when the MPU enters the idle mode. 1: The timer clock is stopped in conjunction with the MPU clock when the idle mode is entered.	R/W	0
8	RESERVED	Must always be 0	R/W	0
7	IDLDPLL_ARM	Enables the DPLL macro to enter idle mode when DSP is set to global_idle mode, MPU is in idle mode, no active DMA transaction or TCLB_EN pin is asserted low, no TIPB posted write is queued, and the peripheral clocks are stopped. 0: The DPLL remains active when the above conditions occur. 1: The DPLL enters the idle mode when the above conditions are met.	R/W	0

Table 5–4. MPU Idle Enable Control Register 1 (ARM_IDLECT1) (Continued)

Bit	Name	Function	R/W	Reset Value
6	IDLIF_ARM	Enables the TIPB bridge, the system DMA controller, and the TC to enter idle mode when the MPU processor executes the wait-for-interrupt instruction. 0: The clocks remain active when the MPU enters the idle mode. 1: The clocks are stopped in conjunction with the MPU clock when the idle mode is entered and the DSP is also in idle.	R/W	0
5	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0
4	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0
3	RESERVED	Reading this bit gives an undefined value. Writing to it has no effect.	R/W	0
2	IDLPER_ARM	Selects idle entry mode for external peripheral clock. (ARMPER_CK) 0: The peripheral clock remains active when the MPU and TC enter the idle mode. 1: The peripheral clock is stopped in conjunction with the MPU and TC clocks when the idle mode is entered. ARMPER_CK is no longer dependent only on ARM_IDLE conditions. As long as TC or MPU is active, ARMPER_CK clock is on. When both are inactive, ARMPER_CK is shutoff based on ARMPER IDLE/ACK.	R/W	0
1	IDLXORP_ARM	Selects idle entry mode for external reference peripheral clock ARMXOR_CK. 0: The external peripheral clock ARMXOR_CK remains active when the MPU enters the idle mode. 1: The external peripheral clock is stopped in conjunction with the MPU clock when the idle mode is entered.	R/W	0
0	IDLWDT_ARM	Selects the idle entry mode for internal timer/watchdog connected to MPU TIPB. When the timer/watchdog is configured as watchdog timer, the clock is never shutdown regardless of the IDLWDT_ARM bit. 0: The clock supplied to the timer/watchdog remains active when the MPU enters idle mode. 1: The timer/watchdog clock is stopped in conjunction with the MPU clock when the idle mode is entered.	R/W	0

Table 5–5. MPU Idle Enable Control Register 2 (ARM_IDLECT2)

Bit	Name	Function	R/W	Reset Value
15:12	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0000
11	EN_CKOUT_ARM	This read-write bit enables the free running clock from DPLL1 output 0: The clock generated from DPLL1 output is stopped. This bit must be set to logic 1 to resume clock activity 1: The clock generated from DPLL1 output is active	R/W	0
10	RESERVED	Reading this bits gives undefined values. Writing to it has no effect.	R/W	0
9	RESERVED	This bit must be set to 0.	R/W	0
8	DMACK_REQ	Disables the permanently-supplied-clock to the system DMA controller to function on a clock request basis. 0: The DMA clock is shutdown when the idle mode is entered if IDLIF_ARM bit of ARM_IDLECTL1 is set. 1: The DMA clock is stopped by default and is reactivated upon DMA request only.	R/W	1
7	EN_TIMCK	Enables the MPU internal timer clock connected to the MPU TIPB. 0: The MPU timer clock is stopped. 1: The MPU timer clock is active and can be stopped depending of the IDLTIM_ARM bit of ARM_ IDLECTL1.	R/W	0
6	EN_APICK	Enables the clock of the MPUI. 0: The MPUI clock is stopped. This bit must be set to logic 1 to enable clock activity 1: The MPUI clock is active. The clock ON/OFF is now controlled as per IDLAPI_ARM bit.	R/W	0
5:4	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	00
3	EN_LCDCK	Enables the clock of the LCD controller connected to MPU TIPB. 0: The LCD clock is stopped. 1: The LCD clock is active.	R/W	0

Table 5–5. MPU Idle Enable Control Register 2 (ARM_IDLECT2) (Continued)

Bit	Name	Function	R/W	Reset Value
2	EN_PERCK	Enables the external peripheral clock. 0: The external peripheral clock ARMPER_CK is stopped. 1: The external peripheral clock ARMPER_CK is active and can be stopped depending on the IDLLPER_ARM bit.	R/W	1
1	EN_XORPCK	Enables the clock of the OS timer connected to MPU TIPB and the external reference peripheral clock. 0: The OS timer clock and the external peripheral clock are stopped. 1: The OS timer clock and the external peripheral clock are active and can be stopped depending on the IDLXORP_ARM bit of ARM_IDLECTL1.	R/W	0
0	EN_WDTCK	Enables the clock of the timer/watchdog connected to MPU TIPB. (When the timer/watchdog is configured as watchdog timer, the clock is never shutdown regardless the value of IDLWDT_ARM and EN_WDTCK). 0: The timer/watchdog clock is stopped. 1: The clock supplied to timer/watchdog clock is active and can be stopped depending on the IDLWDT_ARM bit of ARM_IDLECTL1.	R/W	0

Table 5–6. MPU Restore Power Delay Register (ARM_EWUPCT)

Bit	Name	Function	R/W	Reset Value
15:6	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x000
5	REPWR_EN	Enables the external power control feature. 0: The RESPWR pin is set to logic low when TC is in idle mode. 1: The RESPWR pin stays high when the TC idle mode is entered.	R/W	1
4:0	EXTPWR	Define the delay from RESPWR pin going high to the clocks restarting. Reference clock is CK_REF.	R/W	11111

Table 5–7. Master Software Reset Register (ARM_RSTCT1)

Bit	Name	Function	R/W	Reset Value
15:4	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x000
3	SW_RST	Global system reset. Resets both the DSP and the MPU and peripherals. This bit is always read 0. 0: The DSP, the MPU, and the peripheral clock domains are enabled. 1: Resets the OMAP730 hardware engine. Once set to logic 1 by the MPU processor, this bit returns to logic 0 on the next cycles.	R/W	0
2:1	RESERVED	Must be kept to 00	R/W	00
0	ARM_RST	Resets the MPU. This bit is always read 0 0: The MPU clock domain is enabled. 1: Reset the MPU. Once set to 1 by the MPU, this bit returns to 0 on the next cycles	R/W	0

Note: Writing the DSP_EN bit to 0 and ARM_RST bit to 1 together initiates a global software reset.

Table 5–8. Peripherals Reset Register (ARM_RSTCT2)

Bit	Name	Function	R/W	Reset Value
15:1	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x0000
0	PER_EN	Controls the ARMPER_nRST output pin that can be used to reset and/or enable the external peripherals connected to MPU TIPB. 0: Sets the ARMPER_nRST pin to low-level output voltage 1: Sets the ARMPER_nRST pin to high-level output voltage	R/W	0

Table 5–9. MPU Clock Reset Status Register (ARM_SYSST)

Bit	Name	Function	R/W	Reset Value
15:14	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	00
13:11	CLOCK_SELECT	Reading these bits indicates the clock_select pins and indicates the current clocking mode selection. Writing to these bits enables switching the OMAP3.2 clocking scheme. These bits are at logic 0 after reset: 000: Fully synchronous 001: Reserved 010: Synchronous scalable 011: Reserved 100: Reserved 101: Bypass 110: Mixed mode 3, MPU synchronous to TC, DSP MMU synchronous scalar to MPU and TC 111: Mixed mode 4, DSP MMU synchronous to TC, MPU synchronous scalar to DSP MMU and TC	R/W	000
10:7	RESERVED	These read only bits are undefined.	R	0
6	RESERVED	Reserved	R	0
5	POR	Indicates (in conjunction with EXT_RST bit) whether or not a power-on reset (cold start) has occurred. Writing it to logic 0 clears this bit. This bit cannot be written to logic 1 from the TIPB interface. 0: No power-on-reset has been detected. 1: A power-on-reset has occurred.	R/C	1
4	EXT_RST	Indicates that external reset has been asserted. Writing it to logic 0 clears this bit. This bit cannot be written to logic 1 from the TIPB interface. 0: No external reset has been detected. 1: An external reset has occurred.	R/C	1
3	ARM_MCRST	Indicates whether or not an MPU reset has occurred. This bit is cleared to 0 upon an external reset pulse asserting at the CHIP_nRESET pin, or by writing to it a logic 0. This bit cannot be written to logic 1 from the TIPB interface. 0: The MPU processor has not been reset. 1: The MPU processor has been reset.	R/C	1

Table 5–9. MPU Clock Reset Status Register (ARM_SYSST) (Continued)

Bit	Name	Function	R/W	Reset Value
2	ARM_WDRST	Indicates whether or not the reset has been asserted due to an MPU timer/watchdog underflow. This bit is cleared to 0 upon an external reset pulse asserting at the CHIP_nRESET pin, or by writing to it a logic 0. This bit cannot be written to logic 1 from the TIPB interface. 0: An MPU timer/watchdog underflow has not occurred. 1: An MPU timer/watchdog underflow has generated the reset.	R/C	0
1	GLOB_SWRST	Indicates whether or not the reset has been asserted due to global software reset (DSP_EN set to 0 and ARM_RST set to 1). This bit is cleared to 0 upon an external reset pulse asserting at the CHIP_nRESET pin, or by writing to it a logic 0. This bit cannot be written to logic 1 from the TIPB interface. 0: Global software reset has not been requested. 1: Global software reset has been requested.	R/C	0
0	RESERVED	Reserved	R/C	0

Table 5–10. MPU Clock Out Definition Register (ARM_CKOUT1)

Bit	Name	Function	R/W	Reset Value
15:6	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x0000
5:4	TCLKOUT	The POCLKOUT3 pin functions are: 00: Reserved 01: POCLKOUT3 pin is an output and reflects CK_GEN3 clocking signal. 10: POCLKOUT3 pin is an output and reflects the TC_CK clock. 11: Reserved POCLKOUT3 is not a target clock; it is a free-running clock to select CK_GEN3 or TC_CK frequency.	R/W	01

Table 5–10. MPU Clock Out Definition Register (ARM_CKOUT1) (Continued)

Bit	Name	Function	R/W	Reset Value
3:2	DCLKOUT	The POCLKOUT2 pin functions are: 00: POCLKOUT2 pin is an output and reflects the DSPMMU_CK clock. 01: POCLKOUT2 pin is an output and reflects the CK_GEN2 clocking signal. 10: Reserved 11: POCLKOUT2 pin is an output and reflects low-frequency clock that supplies internal watchdog timers (CK_REF/14). POCLKOUT2 is not a target clock; it is a free-running clock to select DSPMMU_CK or CK_GEN2 frequency.	R/W	01
1:0	ACLKOUT	The POCLKOUT1 pin functions are: 00: Reserved 01: POCLKOUT1 pin is an output and reflects the CK_GEN1 clocking signal. 10: POCLKOUT1 pin is an output and reflects the ARM_CK clock. 11: POCLKOUT1 pin is an output and reflects the low-frequency clock that supplies the internal timers (CK_REF/14). POCLKOUT1 is not a target clock; it is a free-running clock to select CK_GEN1, ARM_CK, or CK_REF/14 clock frequency.	R/W	01

Table 5–11. MPU Reserved Register (ARM_CKOUT2)

Bit	Name	Function	R/W	Reset Value
15:2	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x0000
1:0	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R	0x0

Table 5–12. MPU Idle Enable Control Register 3 (ARM_IDLECT3)

Bit	Name	Function	R/W	Reset Value
15:6	RESERVED	Reading these bits gives undefined values. Writing to them has no effect.	R/W	0x000
5	IDLTC2_ARM	Selects the idle entry mode for TC2 clock. 0: The TC2 clock remains active when the MPU enters the idle mode (ARM_CK stopped). 1: The TC2 clock is stopped in conjunction with the MPU clock when the idle mode is entered. Cutting off is based on IDLE/ACK protocol between CLKRST and peripherals outside OMAP.	R/W	0
4	EN_TC2_CK	Enables the TC2 clock. This is a generic clock supplied to peripherals outside OMAP boundary and is at the same frequency as the TC clock. 0: The TC2_CK clock is stopped. Ensure that all peripherals connected to TC2_CK are inactive before setting 0 on EN_TC2_CK. 1: The TC2_CK clock is active.	R/W	1
3	IDLTC1_ARM	Selects the idle entry mode for TC1 clock. 0: The TC1 clock remains active when the MPU enters the idle mode (ARM_CK stopped). 1: The TC1 clock is stopped in conjunction with the MPU clock when the idle mode is entered. Cutting off is based on IDLE/ACK protocol between CLKRST and peripherals outside OMAP.	R/W	0
2	EN_TC1_CK	Enables the TC1 clock. This is a generic clock supplied to peripherals outside OMAP boundary and is at the same frequency as the TC clock. 0: The TC1_CK clock is stopped. Ensure that all peripherals connected to TC1_CK are inactive before setting 0 on EN_TC1_CK. 1: The TC1_CK clock is active.	R/W	1
1	IDLOCPI_ARM	Selects the idle entry mode for the L3 OCP initiator. 0: The L3 OCP-I clock remains active when the MPU enters the idle mode. 1: The L3 OCP-I clock is stopped in conjunction with the MPU clock when the idle mode is entered.	R/W	0
0	EN_OCPI_CK	Enables the L3 OCPI clock. 0: The L3 OCPI clock is stopped. 1: The L3 OCPI clock is active.	R/W	1

5.5.2 DPLL Registers

Table 5–13 provides bit descriptions of the DPLL control register.

Offset: 0xCF00

Table 5–13. DPLL1 Control Register (DPLL1_CTL_REG)

Bit	Name	Function	R/W	Reset Value
15	LS_DISABLE	Controls the level shifter power-down pin 0: Level shifter is in *transparent* mode; all signals between the wrapper and the DPLL core are connected 1: Level shifter is in *isolated* mode; the wrapper and the DPLL core are disconnected, so the DPLL core power supply (VDD_DPLL) can be turned off. There is no leakage current between VDD and VDD_DPLL.	R/W	0
14	IAI	Initialize after idle. Value of this bit must not be changed. Must be set to 0.	R/W	0
13	IOB	Initialize on break. When high, DPLL switches to bypass mode and starts a new locking sequence, even if the DPLL core indicates that it has lost the lock. When low, DPLL continues to output the synthesized clock, even if the core indicates it has lost the lock but the BREAKLN is active low.	R/W	1
12	TEST†	Controls the test output clock on the DPLL_TCLKOUT pin as given below: 0: DPLL_TCLKOUT = DPLL1 output clock when in test mode. 1: DPLL_TCLKOUT = DPLL1 output clock divided by 32 when in test mode. X: DPLL_TCLKOUT = 0 when not in test mode.	R/W	0
11:7	PLL_MULT	DPLL multiply value. The maximum clock out frequency is 31 * CK_REF.	R/W	00000
6:5	PLL_DIV	DPLL divide value. The minimum DPLL1 clock out frequency is CK_REF/4. 00: CLKOUTDPLL1 output clock = CK_REF 01: DPLL1 output clock CLKOUT = CK_REF/2 10: DPLL1 output clock CLKOUT = CK_REF/3 11: DPLL1 output clock CLKOUT = CK_REF/4	R/W	00

† POCLKOUT[1, 2, 3] is a nongated output from clock domain[1, 2, 3]. Using the [A/D/T] CLKOUT bits, select between different clocks used in the respective domains.

Table 5–13. DPLL1 Control Register (DPLL1_CTL_REG) (Continued)

Bit	Name	Function	R/W	Reset Value
4	PLL_ENABLE	Requests the DPLL to enter the lock mode. DPLL enters the lock mode only after it has synthesized the desired frequency. 0: DPLL enters the bypass mode. 1: DPLL enters the lock mode.	R/W	0
3:2	BYPASS_DIV	Determines the clock out frequency when in bypass mode. 00: DPLL1 output clock CLKOUT = CK_REF 01: DPLL1 output clock CLKOUT = CK_REF/2 1X: DPLL1 output clock CLKOUT = CK_REF/4	R/W	00
1	BREAKLN	Indicates whether DPLL has broken lock for some unknown reason. 0: DPLL has broken lock for some unknown reason. 1: Lock condition is restored or a write to a control register occurs.	R	0
0	LOCK	Indicates if DPLL is in lock mode and the clock out has the desired synthesized frequency. 0: DPLL is in bypass mode. 1: DPLL is in lock mode.	R	0

† POCLKOUT[1, 2, 3] is a nongated output from clock domain[1, 2, 3]. Using the [A/D/T] CLKOUT bits, select between different clocks used in the respective domains.

Chapter 6

MPU-S Interrupt Handler

This chapter discusses the MPU interrupt handler of the OMAP730 hardware engine.

Topic **Page**

6.1 Description

The MPU interrupt handler allows up to 32 hosts that generate interrupts to connect to the MPU, which can accept only two interrupts: fast interrupt request (FIQ) and low-priority interrupt request (IRQ). You can also program the interrupt handler to assign different priorities and mask each interrupt, and you can program each interrupt line to be either edge-triggered or level-sensitive.

Interrupts are handled through two cascaded interrupt controllers: The level 1 handler, which is internal to the OMAP, and the level 2 handler, which is external to the OMAP and functions similarly to the MPU interrupt handler.

- The 32 level 1 interrupts are handled by the MPU interrupt controller provided by the OMAP.
- The 64 level 2 interrupts are handled by the external interrupt controller and cascaded into INT0 and INT1 of the MPU internal interrupt controller.

For power management, CLK&RST can turn off the interrupt handler functional clock. A handshaking protocol is defined for the CLK&RST module to idle or wake up the interrupt handler.

6.1.1 Interrupt Control and Configuration

If an interrupt occurs, the ITR register stores the incoming interrupt in the corresponding bit. When there are several incoming interrupts, the MPU interrupt handler compares the priority level of the interrupts before sending an IRQ or FIQ to the MPU core. The selected interrupt's number is stored in SIR_IRQ or SIR_FIQ for the MPU to determine which interrupt service routine to execute. Reading either of these registers by the MPU resets the corresponding bit in ITR. The MPU can also clear each bit individually in ITR by writing a 0 to the corresponding bits. Writing a 1 keeps its previous value.

Each incoming interrupt can be masked individually by setting the corresponding bit in MIR to 1.

One interrupt level register (ILR) is associated with each incoming interrupt. The ILR determines whether the interrupt is to be edge-triggered or level-sensitive and assigns it a priority level: 0 (the highest priority), 1, ... 30, 31 (the lowest priority). If several interrupts have the same priority level assigned, they are serviced in a predefined order: IRQ_31, IRQ_30, ..., IRQ_1, IRQ_0. ILR also allows routing each of the 32 interrupts to either FIQ or IRQ.

The IRQ or FIQ outputs can be reset by writing a 1 to the corresponding bit of the CONTROL_REG to enable new IRQ or FIQ generation. The writing also clears the SIR_IRQ or SIR_FIQ register. Note that the corresponding bit in the ITR must be cleared before writing to the CONTROL_REG.

6.1.2 Software Interrupt

The interrupt handler also provides a 32-bit software interrupt register (SIR), which corresponds to the same 32-bit external interrupt lines. Writing a 1 to the targeted bit generates an interrupt if the corresponding ILR is set to edge-sensitive; otherwise, no interrupt is generated.

An external interrupt request and an internal software request are merged together before being sent to the interrupt handler to be serviced. The software interrupt register is always read back with a 0. You can use this software interrupt mechanism to simulate an external interrupt and test the corresponding interrupt driver as long as the interrupt line is programmed as edge-sensitive.

All internal interrupts are brought to the OMAP730 subchip level to provide maximum flexibility for system integration. You can reorganize, regroup, add, or delete the interrupt inputs for your applications.

Figure 6–1 shows the MPU interrupt handler.

Figure 6–1. MPU Interrupt Handler

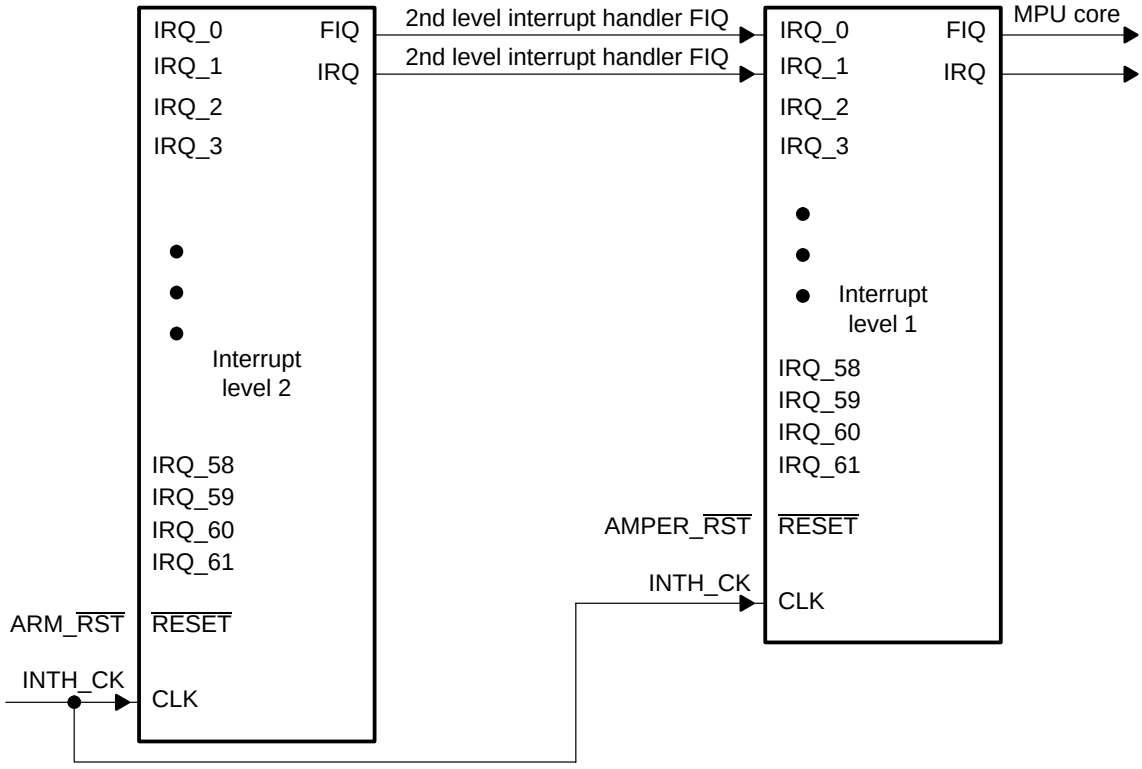
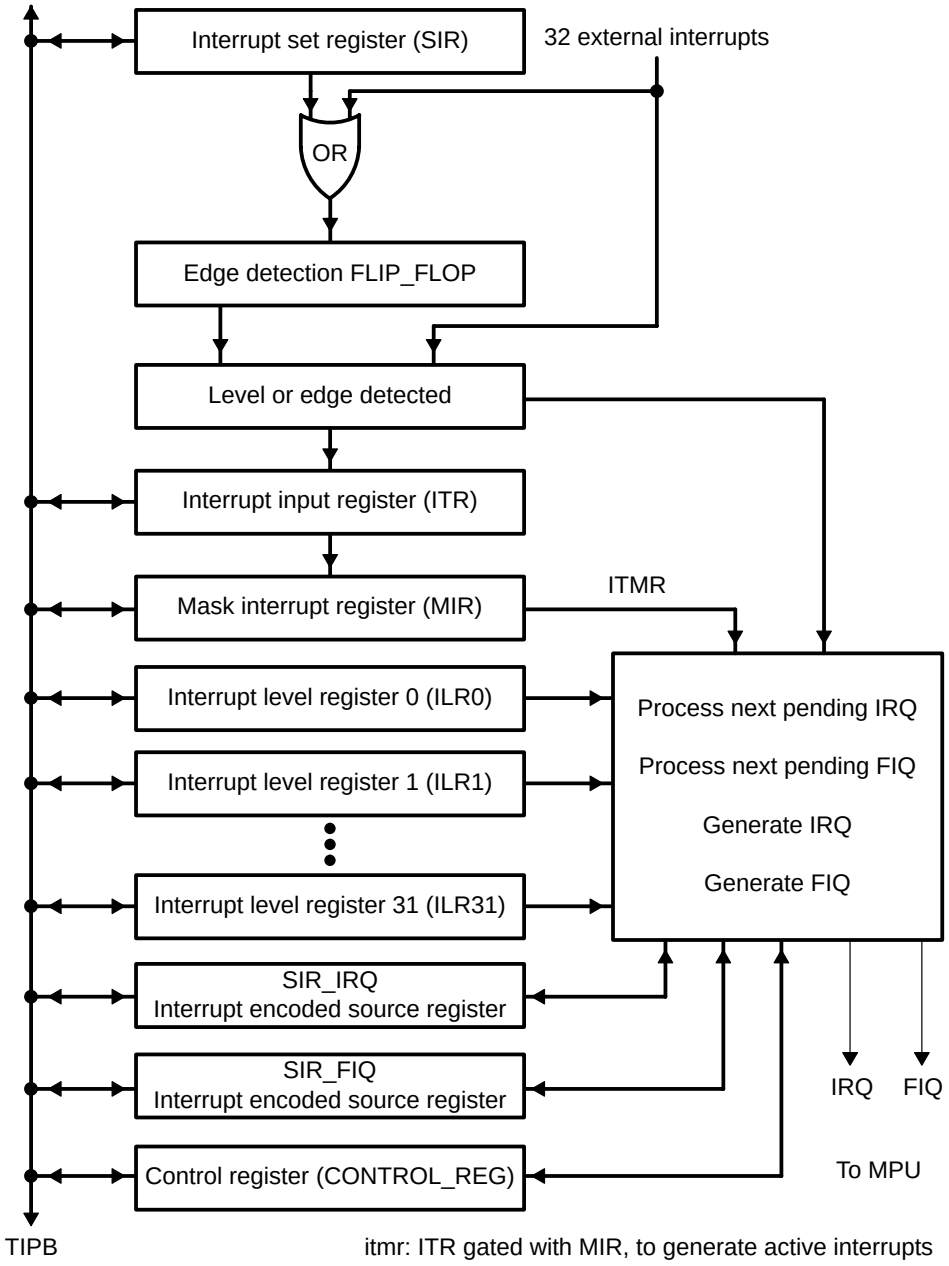


Figure 6–2. MPU Interrupt Handler Block Diagram



6.2 Interrupt Sequence

Table 6–1 shows the MPU interrupt sequence for an IRQ interrupt only. The FIQ interrupt sequence is identical.

Table 6–1. MPU Interrupt Sequence

Step	Interrupt Handler Action	MPU Action
One or several interrupts occur that set the corresponding bits in ITR	If one active interrupt occurs and the IRQ is not already active, the interrupt handler sends an IRQ. If several active interrupts occur, the interrupt handler must locate the interrupt with the highest priority. If an IRQ is not already active, the interrupt handler sends an IRQ.	
Processing the interrupt	When the IRQ is sent, SIR_IRQ is updated and the priority resolver is reset.	The MPU must read SIR_IRQ to determine the interrupt line being serviced. Then the MPU runs the corresponding subroutine.
Finish the interrupt		1. The MPU must first clear the interrupt bit in ITR (by writing a 0 in the corresponding bit or by reading SIR_IRQ. 2. For a level-sensitive interrupt, the level must be removed from the source that keeps the interrupt request low (active) for the next interrupt to occur. 3. Sets CONTROL_REG.NEW_IRQ_AGR to reset IRQ output and SIR_IRQ, thus allowing a new IRQ generation.

6.3 Interrupt Handler Software

To correctly process edge-triggered and level-sensitive interrupts, the following sequences must occur in the system. Only the IRQ treatment is described here. The FIQ treatment is identical.

6.3.1 Edge-Triggered Interrupts

1) The interrupt handler module receives one or more interrupts from outside the OMAP and registers them in the interrupt register (ITR).

2) If there are several active incoming interrupts, the interrupt handler determines the highest priority interrupt and puts it in the N_IRQ register, which is invisible to the software programmer.

3) If the IRQ (interrupt from the interrupt handler to the MPU) is not active, the interrupt handler sends the interrupt in N_IRQ register to the MPU as an IRQ signal. Then the source IRQ register (SIR_IRQ) is updated with the contents of the N_IRQ register (the SIR_IRQ contains encoded information that conveys the interrupt line number of the IRQ).

4) The MPU recognizes the interrupt and jumps to the interrupt service routine (ISR) code.

5) Within the ISR code, the MPU reads the SIR_IRQ in the interrupt handler to determine which interrupt line caused the interrupt. The MPU executes specific code appropriately.

6) When the MPU reads the SIR_IRQ, the corresponding bit is reset in ITR of the interrupt handler module. The IRQ is still active.

7) ,When the MPU is about to exit the ISR routine, it writes a 1 to CONTROL_REG.NEW_IRQ_AGR to deassert the IRQ going to the MPU and to enable a new IRQ generation.

8) The MPU exits the ISR and continues its normal code execution.

9) Once CONTROL_REG.NEW_IRQ_AGR is written, the process jumps to Step 2.

6.3.2 Level-Sensitive Interrupts

1) The interrupt handler module receives one or more incoming interrupts from outside the OMAP. Level-sensitive interrupts are not registered, but are used in the logic as is. The interrupt handler assumes that the peripheral will not deassert the level-sensitive incoming interrupts until it is told to do so by the MPU.

2) The interrupt handler determines the highest priority interrupt and puts it in the N_IRQ register.

3) If the IRQ (interrupt from interrupt handler to the MPU) is not active, the interrupt handler sends the highest priority interrupt (interrupt in N_IRQ

register) to the MPU as the IRQ signal. Then the SIR_IRQ is updated with the contents of N_IRQ register (the SIR_IRQ register contains encoded information that conveys the interrupt line number of IRQ). If the IRQ is active, which means CONTROL_REG.NEW_IRQ_AGR has not been set by the MPU, the incoming IRQ has to wait until the IRQ is not active.

4) The MPU recognizes the interrupt and jumps to the ISR code.

5) Within the ISR code, the MPU reads the SIR_IRQ in the interrupt handler to determine which interrupt line caused the interrupt.

6) The ISR code must be capable of doing one of the following things:

   - Letting the peripheral know that the interrupt generated by it has been serviced so the peripheral can deassert the interrupt request
   - Writing to interrupt handler mask interrupt register (MIR) to mask the level-sensitive interrupt

     Here the peripheral has to deassert the interrupt before the mask to the interrupt can be removed, so that the next interrupt can be recognized.

     If the peripheral deasserts the interrupt before the code in ISR tells it to, then the behavior is unpredictable and the interrupt may be lost.

7) When the MPU is about to exit the ISR routine, it must write a 1 to CONTROL_REG.NEW_IRQ_AGR to deassert the IRQ going to the MPU and to enable a new IRQ generation.

8) The MPU exits the ISR and continues its normal code execution.

9) When CONTROL_REG.NEW_IRQ_AGR is written to by MPU, the process jumps to Step 2.

6.4 Registers

6.4.1 Interrupt Registers

Table 6–2 lists the registers available to handle interrupts. Table 6–3 through Table 6–10 provide register bit descriptions. All these registers are 32 bits wide and are controlled directly by the private TIBP bus. To determine the base address of these registers, see Section 1.4, *Memory Mapping and Mapped Registers.*

Table 6–2. Interrupt Registers

Register	Offset (Byte)
ITR	0x00
MIR	0x04
SIR_IRQ	0x10
SIR_FIQ	0x14
CONTROL_REG	0x18
ILRx	0x1C + 0x4 * × interrupt number
SIR	0x9C
ENHANCED_CNTL_REG	0xA0

Table 6–3. Interrupt Register (ITR)

Bit	Name	Function	R/W	Reset Value
31:0	ACT_IRQ	Sets corresponding bit in ITR for edge-sensitive and level-sensitive interrupts	R/W	0x0000 0000

When the MPU accesses SIR_IRQ or SIR_FIQ, the ITR bit corresponding to the interrupt that has requested MPU action is reset. The MPU can also clear each bit individually by writing a 0 to the corresponding bits at the ITR address. Note that writing a 1 to a bit keeps its previous value. You can use the individual clearing just before the MPU unmasks some interrupts and thus ignore some interrupt occurrences.

Table 6–4. Mask Interrupt Register (MIR)

Bit	Name	Function	R/W	Reset Value
31:0	IRQ_MSK	Masks each incoming interrupt individually	R/W	0xFFFF FFFF

You can mask each incoming interrupt individually with the mask interrupt register by setting the corresponding bit to 1. This register operates after the interrupt register, which means that occurrences of incoming interrupts are always stored in the interrupt register.

Table 6–5. Interrupt Encoded Source Register for IRQ (SIR_IRQ)

Bit	Name	Function	R/W	Reset Value
31:5	RESERVED			
4:0	IRQ_NUM	Indicates encoded interrupt number that has an IRQ request. Reading this register clears the corresponding bit in the ITR register if the interrupt is set as edge-sensitive.	R	00000

Table 6–6. Interrupt Encoded Source Register for FIQ (SIR_FIQ)

Bit	Name	Function	R/W	Reset Value
31:5	RESERVED			
4:0	FIQ_NUM	Indicates the encoded interrupt number that has an FIQ request. Reading this register clears the corresponding bit in the ITR register if the interrupt is set as edge-sensitive.	R	00000

Table 6–7. Interrupt Control Register (CONTROL_REG)

Bit	Name	Function	R/W	Reset Value
31:2	RESERVED			
1	NEW_FIQ_AGR	New FIQ agreement. Writing a 1 resets the FIQ output, clears the SIR_FIQ, and enables a new FIQ generation. The corresponding bit of the ITR must be cleared first. Writing 0 has no effect.	R/W	0
0	NEW_IRQ_AGR	New IRQ agreement. Writing a 1 resets the IRQ output, clears the SIR_IRQ, and enables a new IRQ generation. The corresponding bit of the ITR must be cleared first. Writing 0 has no effect.	R/W	0

Table 6–8. Interrupt Level Register for Interrupt Number x (0 to 31) (ILRx)

Bit	Name	Function	R/W	Reset Value
31:7	RESERVED			
6:2	PRIORITY	Defines the priority level when the corresponding interrupt is routed to IRQ or FIQ. 0 is the highest priority level. 31 is the lowest priority level.	R/W	00000
1	SENS_LEVEL	0: The corresponding interrupt is falling-edge sensitive. 1: The corresponding interrupt is low-level sensitive.	R/W	1
0	FIQ	0: The corresponding interrupt is routed to IRQ. 1: The corresponding interrupt is routed to FIQ.	R/W	0

Table 6–9. Software Interrupt Set Register (SIR)

Bit	Name	Function	R/W	Reset Value
31:0	SIR	Writes a 1 to any bit that generates an interrupt to the MPU if the corresponding bit in the ILR is set as edge-triggered; otherwise, no interrupt is generated. The software interrupts are merged with external interrupts before being sent to the MIR.	W	0x0000 0000

Table 6–10. Enhanced Control Register (ENHANCED_CNTL_REG)

Bit	Name	Function	R/W	Reset Value
31:1	RESERVED			
0	GLOBAL_MASK	When 1, the interrupt handler module and the output signal INT_DIS are set to 1 (INT_DIS:output signal indicating that the interrupt handler has been disabled). For power management, CLK and RST can turn off the interrupt handler functional clock. A handshaking protocol is defined for the CLK and RST module to idle or wake up the interrupt handler.	R/W	0

6.4.2 Interrupt Level 2 Registers

Wait states: 0 Width: 32

Table 6–11 lists the interrupt level 2 registers. Table 6–12 through Table 6–21 provide register bit descriptions.

Table 6–11. Interrupt Level 2 Registers

Register	Domain	Offset (Byte)
ITRX	MPU	0x000
MIRX	MPU	0x004
SIR_IRQ_CODE	MPU	0x010
SIR_FIQ_CODE	MPU	0x014
CONTROL_REG	MPU	0x018
ILRX	MPU	0x01C
ISIRX	MPU	0x09C
STATUS	MPU	0x0A0
OCP_CFG	OCP	0x0A4
INTH_REV	All	0x0A8

Table 6–12. Interrupt Register (ITRX)

Bit	Name	Function	R/W	Reset Value
31:0	ACT_IRQ	Stores active line in case of edge-sensitive interrupt 0: Reset corresponding interruption 1: No effect	R/W	0x0000

There are 64 bits for the ITR, grouped in 32-bit sections. The offset between sections is 0x100. The first section contains the first 32 bits (0 to 31), and the second section contains the ITR from 32 to 63 bits.

Table 6–13. Mask Interrupt Register (MIRX)

Bit	Name	Function	R/W	Reset Value
31:0	IRQ_MSK	Masks the corresponding interrupt. Each bit corresponds to one interrupt.	R/W	0xFFFF

There are 64 bits for the MIR, grouped in 32-bit sections. The offset between sections is 0x100. The first section contains the first 32 bits (0 to 31), and the second section contains the MIR from 32 to 63 bits.

Table 6–14. Interrupt Encoded Source for IRQ Register (SIR_IRQ_CODE)

Bit	Name	Function	R/W	Reset Value
31:7	RESERVED	Reserved	R/W	0x0
6:0	IRQ_NUM	This field indicates the encoded interrupt number having an IRQ of FIQ request. Reading this register clears the corresponding bit in the ITR register if the interrupt is set as edge-sensitive.	R	0x00

Table 6–15. Interrupt Encoded Source for FIQ Register (SIR_FIQ_CODE)

Bit	Name	Function	R/W	Reset Value
31:7	RESERVED	Reserved	R/W	0x0
6:0	IRQ_NUM	This field indicates the encoded interrupt number having an IRQ of FIQ request. Reading this register clears the corresponding bit in the ITR register if the interrupt is set as edge-sensitive.	R	0x00

Table 6–16. Interrupt Control Register (CONTROL_REG)

Bit	Name	Function	R/W	Reset Value
31:4	RESERVED	Reserved	R/W	0x0
3	GLOBAL_MASK	1: IRQ and FIQ output lines are frozen. All incoming interrupts are stored. Resetting this bit reenables the treatment of the interrupt.	R/W	0x0
2	RESERVED	Reserved	R/W	0x0
1	NEW_FIQ_AGR	New FIQ agreement 1: Resets FIQ output. Enables a new generation. The treated interrupt must be previously reset.	R/W	0x0
0	NEW_IRQ_AGR	New IRQ agreement 1: Result IRQ output. Enables generation of a new IRQ. The treated interrupt must be reset before.	R/W	0x0

Table 6–17. Priority Level Register (ILRX)

Bit	Name	Function	R/W	Reset Value
31:7	RESERVED	Reserved	R/W	0x0
6:2	PRIORITY	Defines the priority level when the corresponding interrupt is routed to IRQ or FIQ.	R/W	0x00
1	SENS_EDGE	0: The corresponding interrupt is rising-edge sensitive 1: The corresponding interrupt is high-level sensitive	R/W	0x0
0	FIQ	0: The corresponding interrupt is routed to IRQ 1: The corresponding interrupt is routed to FIQ	R/W	0x0

There are 64 bits for the ILR, grouped into 32-bit sections. The offset between sections is 0x100. The first section contains the first 32 bits (0 to 31), and the second section contains bits 32 to 63.

Table 6–18. Software Interrupt Set Register (ISRX)

Bit	Name	Function	R/W	Reset Value
31:0	ISR	Writing a 1 to any bit generates an interrupt to the MPU in the corresponding ILR set as edge trigger. A read returns a 0.	R/W	0x0000

There are 64 bits for the ISR, grouped into 32-bit sections. The offset between sections is 0x100. The first section contains the first 32 bits (0 to 31), and the second section contains bits 32 to 63.

Table 6–19. OCP Status Register (STATUS)

Bit	Name	Function	R/W	Reset Value
31:1	RESERVED	Reserved	R	0x0
0	RESET_DONE	0: Reset has not occurred.	R	0x1

Table 6–20. OCP Configuration Register (OCP_CFG)

Bit	Name	Function	R/W	Reset Value
31:2	RESERVED	Reserved	R	0x0
1	SOFT_RESET	Soft reset: Writing a 1 soft resets the interrupt handler. It is automatically deasserted.	W	0x0
0	RESERVED2	Reserved (renamed to avoid name clash)	R	0x0

Table 6–21. Revision ID Register (INTH_REV)

Bit	Name	Function	R/W	Reset Value
31:8	RESERVED	Reserved	R	0x0
7:4	MAJOR_REVISION	Major revision number	R	0x0
3:0	MINOR_REVISION	Minor revision number	R	0x1

Chapter 7

MPU-S Direct-Memory Access

This chapter describes the direct memory access (DMA) controller for the OMAP730 hardware engine.

Topic **Page**

7.1 General Overview

The system DMA is designed to off-load the block data transfer function from the MPU.

The OMAP730 system DMA controller consists of:

- Sixteen logical channels plus one LCD logical channel
- Six physical ports plus one for configuration
- Three physical channels plus one LCD dedicated physical channel

The ports are connected to the L3 OCP targets, the external memory, the TIPB bridge, and one dedicated port connected to an LCD controller. The system DMA controller can be controlled via the MPU private TIPB or by an external host via the OCP-I port.

The system DMA controller is designed for low-power operation. It is partitioned into several clock domains where each clock domain is enabled only when it is used. All clocks are disabled when no DMA transfers are active.

Five different logical channel types are supported; each one represents a specific feature set.

- LCh-2D for memory to memory transfers, 1D and 2D
- LCh-P for peripheral transfers
- LCh-PD for peripheral transfers on a dedicated channel
- LCh-G for graphical transfers/operations
- LCh-D for display transfers

The features available are:

- Support for up to four address modes:
  - Constant
  - Postincremented
  - Single-indexed
  - Double-indexed
- Different indexing for source and destination
- Logical channel chaining
- Software enabling
- Hardware enabling—31 DMA request lines available
- Logical channel interleaving
- Logical channel preemption
- Two choices of logical channel arbitration of physical resources: Round robin or fixed
- Two levels of logical channel priority
- Constant fill
- Transparent copy

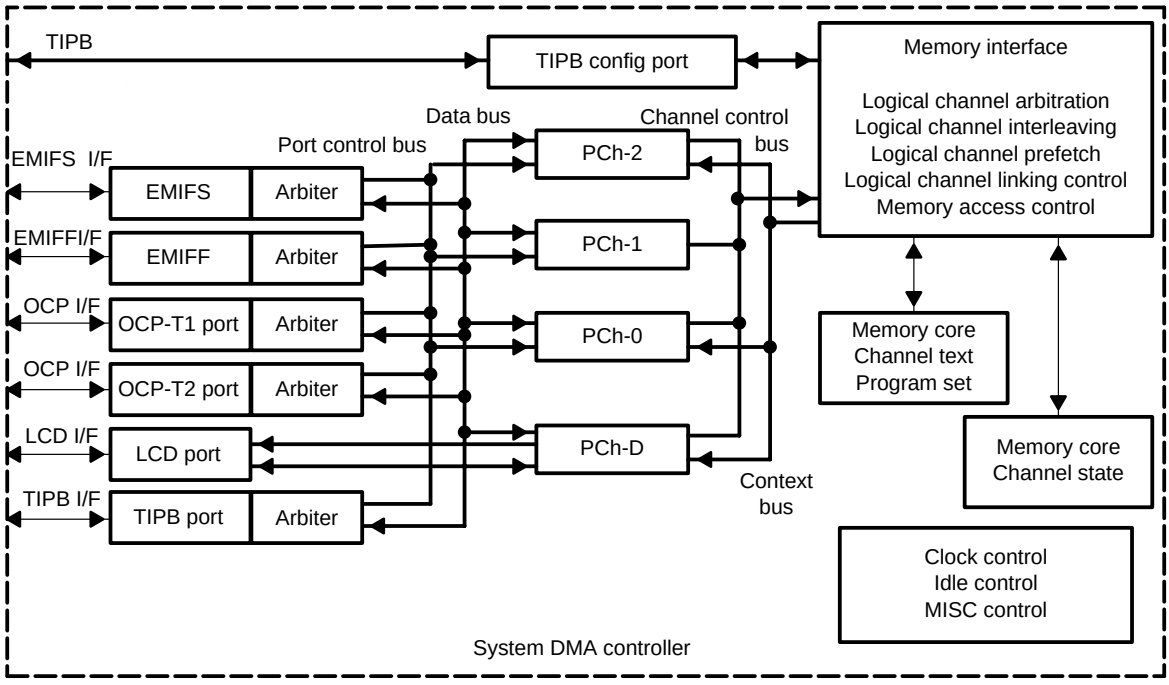
- ❏ Rotation: 0°, 90°, 180°, and 270°
- ❏ Six ports enabling:
  - Memory-to-memory transfers
  - Peripheral-to-memory transfers
  - Memory-to-peripheral transfers
  - Peripheral-to-peripheral transfers
- ❏ Binary backward-compatible by default configuration
- ❏ Up to four logical channels active in parallel

The logical channel dedicated to the display, LCh-D, has several additional features.

- ❏ Can be shared by two LCD controllers
- ❏ Supports both single and dual block modes
- ❏ Supports separate indexing and numbering for dual block mode for both elements and frames

A graphical overview of the system DMA and its external connections is shown in Figure 7–1.

Figure 7–1. System DMA Controller Simplified Block Diagram



7.2 Functional Description

This section describes the system DMA capabilities and programming.

The configuration of the system DMA logical channel registers can be done in any order with the exception of the enable bit in the channel control register, DMA_CCR. This bit enables all logical channel types, so this must be the last thing done when configuring a logical channel (LCh). While an LCh is enabled, it is not allowed to change its configuration registers, which causes undefined effects. All global system DMA configuration registers must be changed before any LChs are enabled; unpredictable effects may result otherwise.

The dedicated LCD channel has some additional features and different channel behavior compared to the generic channels. For more details, see Section 7.3.1, *Display Logical Channel*. Some channel features and behavior are common to both generic and LCD channels; this section describes them and indicates which are supported by the LCD channel.

7.2.1 Logical Channel Types

The OMAP system DMA is based on logical channels (LChs).

Each generic LCh can be configured to one of four different logical channel types. The dedicated LCD LCh can only be configured to LCh type D.

Logical channel types (LCh types) supported are:

- LCh-2D for nonsynchronized transfers (memory transfers, 1D and 2D)
- LCh-P for synchronized transfers (mostly peripheral transfers)
- LCh-PD similar to LCh-P but runs on a dedicated physical channel
- LCh-G for graphical transfers/operations
- LCh-D for display transfers

Note:

Logical channel LCh-D, dedicated to the display, must be configured to LCh type LCh-D.

Each logical channel type represents a specific subset of the system DMA features list. Even if the requirements of a specific transfer can fit into several LCh-types, it is important to select the correct LCh-type. Table 7–1 summarizes the features per logical channel type.

Table 7–1. Summary of Features Per Logical Channel Type (Without LCh-D)

Supported Transfer Features		LCh-2D (2D)	LCh-P (P)	LCh-PD (PD)	LCh-G (G)
LCh Control	*Transfer type*				
	❑ Synchronized	√	√	√	√
	Two levels of priority	√	√	√	√
	Preemption at element boundary	√	√		√
	LCh interleaving on DMA request		√	√	
	Linking logical channel capability	√	√	√	√
Addressing	*Types of addressing modes for SRC and DST*				
	❑ Constant mode	√	√	√	
	❑ Postincrement mode	√	√	√	
	❑ Single-indexed mode with separate SRC/DST index	√	See Note	See Note	
	❑ Double-indexed mode with separate SRC/DST index	√	See Note	See Note	√
Other	*Other LCh-type features*				
	Constant fill	√			√
	Transparent color				√

Note: Supported only on one end of the transfer (memory side—can be a source or a destination).

A separate table applies for the LCh-D (D), because several elements differ between that channel and these generic LChs. See Table 7–15, *Features Summary for LCh-D*, in section 7.3.1.

The abbreviations in parentheses in the Table 7–1 headings are used throughout this document as a guide to features supported for different LCh types.

7.2.2 OMAP730 System DMA Instances

The OMAP730 system DMA has 16 generic logical channels plus one logical channel dedicated to the LCD controller, as well as three physical channels plus one channel dedicated to the LCD controller (PCh-0, PCh-1, PCh-2 plus PCh-D).

PCh-0 and PCh-1 are always dynamically allocated to any active LCh of type LCh-2D, LCh-G, or LCh-P. Dynamic allocation means that the physical channel that becomes free first services the active logical channel. PCh-2 is different from PCh-0 and PCh-1, because it is possible to configure one or several synchronized LChs to only use this physical channel. For example, the physical channel can be dedicated to one LCh or to several interleaved synchronized channels. This is done by specifying an LCh as LCh-PD type. The fourth channel, PCh-D, is specifically designed for transfers to the display and can only be used by the dedicated LCh-D. LCh-D must always be configured as type LCh-D.

Table 7–2. Associated Physical Channels Per Logical Channel Type

LCh-Type	PCh Assigned
LCh-2D	PCh-0 or PCh-1 (dynamic allocation)
LCh-P	
LCh-G	
LCh-PD	PCh-2
LCh-D	PCh-D

There are six global registers: PCh status register, PCh ID register, and one PCh status register per PCh channel. These registers can be used to make DMA configurations independent of DMA hardware versions. See respective register descriptions for more details.

7.2.3 Synchronized Channel

LCh Types Supporting Synchronized Channels	2D	P	PD	G	D
	√	√	√	√	√

A synchronized channel (hardware activated) is a channel that only becomes active when it is enabled by software and subsequently receives a DMA request signal from outside the DMA. There are 31 possible DMA request signals, 1 to 31, which are hard wired at design. A hardware DMA request cannot be shared among several concurrent channels (enabled and active). However, a hardware DMA request can be shared by different channels if they are part of a chain. To understand which peripheral corresponds to a particular DMA request, see *Peripheral DMA Request Line Mapping* in the top-level specification for the design that incorporates OMAP 3.2.

Program the five SYNC bits located in the channel control register, DMA_CCR, to configure the external DMA request that activates the channel. The logical channel becomes a synchronized channel when this field is set to reflect the number of the request line. Remember that no DMA request can be mapped as 0, which is reserved to specify a nonsynchronized transfer.

Each time a DMA request is received for a synchronized channel, the logical channel is activated and a block of data is transferred when a physical channel is assigned to it. This block of data can be:

- An element

  A complete element, which is defined by DATA_TYPE. For example, 8/16/32 bits are transferred in response to a DMA request.

- An entire frame

  A complete frame of several elements is transferred in response to a DMA request.

- An entire block

  A complete block of several frames is transferred in response to a DMA request.

One DMA request can trigger several logical channels at the same time.

To configure an LCh to synchronize by element, frame, or block, program the frame synchronization (FS) bit in the DMA_CCR register and the block synchronization (BS) bit in the DMA_CCR2 register. If both bits are set to 0, the channel is synchronized by element. Setting both bits to 1 causes undefined effects.

❏ To configure an LCh to transfer *one element* per DMA request:

1) Configure the data type, also referenced as element size (ES), in the field Data_type in the channel source destination parameter register (DMA_CSDP).

2) Configure the number of transfers (elements) to take place before the LCh gets disabled again in the channel element number register (DMA_CEN).

3) Configure both FS and BS to 0.

❏ To configure an LCh to transfer *one frame* per DMA request:

1) Configure the element size as described before.

2) Configure the element number as described before. This represents the number of elements sent per frame, hence per DMA request.

3) Configure the number of transfers (frames) to take place before the LCh gets disabled again in the channel frame number registers (DMA_CFN).

4) Configure FS to 1 and BS to 0.

❏ To configure an LCh to transfer *one block* per DMA request:

1) Configure the element size as described before.

2) Configure the element number as described before. This represents the number of elements sent per frame.

3) Configure the frame number as described before. This represents the number of frames sent per block.

4) Configure FS to 0 and BS to 1.

It is also possible to stop a transfer by disabling the channel. This is done by resetting the ENABLE bit in the DMA_CCR register.

7.2.4 Physical Ports

The system DMA can access different data types, depending on which DMA port is used and what memory or peripheral is supporting. The system DMA ports can support different types of protocols and bus widths. This is important to understand, because it governs which type of transfer an LCh can perform. Table 7–3 summarizes the type of transfers supported by each port.

Table 7–3. OMAP3.2 System DMA Supported Interface Port Type Table

Port _Name	Port Functionality
OMAP EMIFF port	Supports: ❑ 8/16/32-bit, 4x32-bit burst serialized access ❑ 4x32-bit burst serialized access, if it is the source port of the LCD channel.
OMAP EMIFS port	
OMAP OCP-T1 port	
OMAP OCP-T2 port	
OMAP TIPB port	Supports: ❑ 8/16/32-bit serialized access
OMAP LCD port	Supports: ❑ 16-bit serialized access for OMAP LCD controller ❑ 32-bit serialized access for OMAP external LCD controller
OMAP TIPB configuration port	TIPB configuration interface for system DMA. It supports 16-bit serialized access.

Five of the physical ports can be selected to be the source and/or destination of a DMA transfer. The DMA LCD port can only function as a destination port. The TIPB configuration port cannot be used as a source or destination port.

- TIPB configuration interface:
  - Used by the MPU to control/configure the system DMA. Cannot be used as source or destination in a DMA transfer.
- External slow memory interface (flash/ROM):
  - All DMA accesses are done in burst mode of 4x32 bits, except for data packets that are not 16-byte aligned. The remaining data is transferred in single access mode (8/16/32-bit).
- External fast memory interface (SDRAM):
  - All DMA accesses (except for the channel dedicated for the LCD controller) are done in burst mode of 4x32 bits, except for data packets that are not 16-byte aligned. The remaining data is transferred in single access mode (8/16/32-bit).
- OCP-T1 and OCP-T2 interface (SRAM, SDRAM):
  - All DMA accesses are done in burst mode of 4x32 bits, except for data packets that are not 16-byte aligned. The remaining data is done in single accesses mode (8/16/32-bit).
- TIPB interface (to peripherals via TIPB bridge):
  - All DMA accesses are done in single access mode (8/16/32 bits).
- LCD interface
  - Port to an external LCD controller or to the OMAP LCD controller. Supports 32-bit access to the OMAP external LCD controller and 16-bit access for the OMAP LCD controller. Refer to the OMAP1610 chapter for additional details on the external LCD controller.

For all ports, only the number of programmed bytes are transferred; that is, there are no trailing or dirty bytes at the end of transfer.

Table 7–4 provides possible source ports (SRC), destination ports (DST), and data transfers.

Table 7–4. Possible Data Transfer

	DST					
SRC	**EMIFS**	**EMIFF**	**OCP-T1**	**OCP-T2**	**TIPB Bridge**	**LCD [1]**
EMIFS	Yes	Yes	Yes	Yes	Yes	No
EMIFF	Yes	Yes	Yes	Yes	Yes	Yes
OCP-T1	Yes	Yes	Yes	Yes	Yes	Yes
OCP-T2	Yes	Yes	Yes	Yes	Yes	Yes
TIPB bridge	Yes	Yes	Yes	Yes	Yes	No
LCD[1]	No	No	No	No	No	No

Note: [1] Used on both OMAP internal and external LCD controllers.

The port to use for source or destination is configured in the channel source destination parameters register, DMA_CSDP. The data type is also configured in this register.

Note:

It is the programmer's responsibility to ensure coherence among:

- Source port and start addresses
- All source addresses within the transfer and memory space of source port
- Destination port and start addresses
- All destination addresses within the transfer and memory space of destination port

Care must be exercised because no address space check is performed by the system DMA or at OMAP top level. All addressing outside the source and destination memory space causes undefined effects.

7.2.5 Port Channel Scheduling

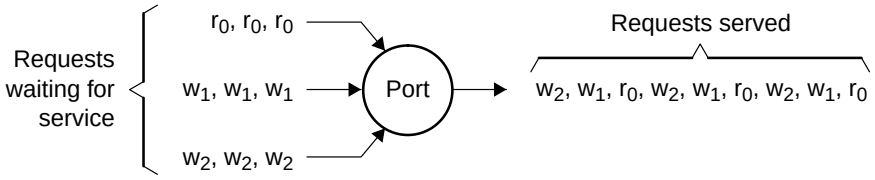
Each DMA logical channel can be configured independently of other logical channels. Each DMA physical channel can have its own port or share it with other physical channels. If physical channels share a DMA port, a port arbiter prioritizes physical channel access.

The system DMA port arbiters follow a round-robin scheme.

In Figure 7–2, a DMA port services three DMA physical channel requests:

- Physical channel 0 as a source port (read requests r_0)
- Physical channel 1 as a destination port (write requests w_1)
- Physical channel 2 as a destination port (write requests w_2)

Figure 7–2. Time Sharing Access on a System DMA Port



The system DMA port arbiters use a round-robin scheme, but are also dependent on the LCh priority. For more information about LCh priority, see Section 7.2.6.2, *Logical Channel Priorities.*

The arbiters use the following scheme each time previous DMA port access has finished:

1) Service all high-priority PChs using a round-robin scheme. A low-priority transfer currently being served by the same port is not aborted or suspended by a high-priority transfer. The high-priority transfer waits.

   Port arbitration is based on each access. If the current access is a burst access, the arbitration is on the boundary of burst. The higher priority PCh transfer takes the port as soon as the next available slot is open at the burst boundary.

2) If no highly prioritized PCh is present, then all PChs are arbitrated/served according to the round-robin scheme.

7.2.6 Logical Channel Scheduling

A logical channel is marked as an active channel only when either of the following conditions has been meet:

- Case 1**:**
  - The logical channel is a synchronized channel.
  - The logical channel enable field is set.
  - The associated DMA request is triggered.

❑ Case 2**:**

- The logical channel is a nonsynchronized channel.
- The logical channel enable field is set.

A logical channel can be assigned to a physical channel only when the logical channel is active.

When a physical channel is granted, the DMA controller loads the physical channel with the logical channel configuration register set, which controls the data transfer through the system DMA. At the end of a channel transfer, the current channel status is updated into the logical channel status register, and the current logical channel enable is cleared if it is a nonsynchronized channel.

7.2.6.1 Logical Channel Scheduling Scheme

Each physical channel can only serve one logical channel at a time. If several logical channels are active and waiting to be served, they are interleaved based on an arbitration scheme in a TDMA manner. The supported arbitration schemes are:

❑ Round-robin scheduling
❑ Fixed scheduling from low LCH ID to high LCH ID

The scheme to be used can be controlled by software on a global basis.

In the global control register, (DMA_GCR), the ROUND_ROBIN_DISABLE bit controls which scheme to follow. Note that this bit can only be changed when the DMA is quiescent; that is, when no LChs are enabled. Any change of this bit when a LCh is enabled causes undefined behavior.

7.2.6.2 Logical Channel Priorities

LCh Types Supporting this Feature	2D	P	PD	G	D
	√	√	√	√	N/A

Each logical channel can be given a low or high priority level. When a DMA physical channel receives requests from several logical channels, it looks at their priorities. The physical channel assignment to logical channels uses the following scheme:

1) Requests from high-priority logical channels are served first. A higher priority logical channel can preempt the current on-going low-priority logical transfer and start the higher priority logical channel transferring. The preempted logical channel continues the transfer as soon as all high-priority channels are served. A transfer can be preempted on an element boundary. See section 7.2.9, *Logical Channel Preempting*, for more information on channel preempting.

2) Requests from low-priority logical channels are served only if there are no requests from high-priority logical channels. This can occur if no high-priority logical channels are active, or if the high-priority logical channels are waiting for a synchronization event.

3) Requests of the same priority level are served in a round-robin or fixed scheduling scheme, as mentioned earlier in this chapter.

Use the PRIO bit in the logical channel register DMA_CCR to configure the LCh priority.

7.2.7 Logical Channel Interleaving For Synchronized Transfers

LCh Types Supporting this Feature	2D	P	PD	G	D
		√	√		

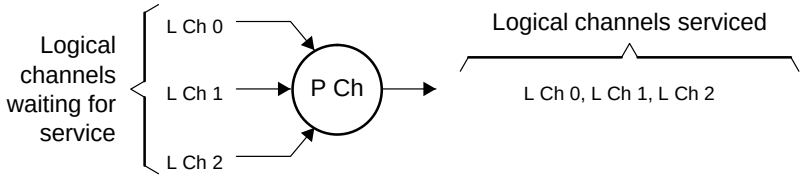
Logical channel interleaving is used when more than one synchronized channel shares the same physical channel. A synchronized channel is only active and requesting access to a physical channel when a DMA request is received.

When a DMA request is served but more data remains to be transmitted, the logical channel stays enabled waiting for the next DMA request. During this time, the physical channel is released if the LCh-type currently running supports logical channel interleaving.

A nonsynchronized LCh can use the physical channel as well (i.e., interleave), but it runs to finish before it releases the channel again. Therefore, it is the responsibility of the software to configure nonsynchronized transfers in small enough blocks so that synchronized LChs are served in time. This is true only if all PChs are occupied.

Figure 7–3 shows an example of a logical channel interleaving scheme for three synchronized LChs where the DMA requests are received in order: LCh 0 request, LCh 1 request, and LCh 2 request.

Figure 7–3. Logical Channel Interleaving on Channel Boundary With the Same Priority



It is possible to disable the interleaving on a logical channel basis. If this is done for a specific logical channel, that logical channel does not release the physical channel between the DMA requests until all the transfers are done or until the LCh is disabled by software. In this way a physical channel can be dedicated to a logical channel of type LCh-P or LCh-PD.

To disable LCh interleave for synchronized transfers, set the LCH_INTERLEAVE_DISABLE bit in the logical channel control register, DMA_LCH_CTRL.

A nonsynchronized transfer is not affected by the LCH_INTERLEAVE_DISABLE bit under any circumstances. A nonsynchronized transfer always releases the physical channel between transfers even if it is linked to another LCh using the logical channel linking feature.

7.2.8 Linking Logical Channels

LCh Types Supporting This Feature	2D	P	PD	G	D
	√	√	√	√	

Software can configure DMA logical channels to an LCh chain.

To configure and start a linked LCh chain:

1) To link the LChs, configure the NEXTLCH_ID bit in the logical channel link control register DMA_CLNK_CTRL to indicate the next logical channel to be enabled as soon as the current logical channel has finished the transfer.

2) To define the LCh as being part of a linked queue, set the ENABLE_LNK bit in the same register. In this way, the logical channel, defined by NEXTLCH_ID, is enabled after the current channel finishes transferring.

3) Start the chain by enabling the first LCh in the chain (set the ENABLE bit of the DMA_CCR register to 1).

In order to stop a linked chain, software can write to the STOP_LNK bit in logical channel link control register to disable the chain and queue. This must be done on each LCh that is part of the chain. Writing to the STOP_LNK bit resets both the ENABLE_LNK bit and the LCh enable bit in the DMA_CCR register. If a linked chain is run to finish without being stopped, the whole chain remains linked; that is, all LChs still have the ENABLE_LNK bit set to 1. To start the same linked chain again, just enable the first LCh in the chain.

An LCh can be linked to all LCH 0 to 15 but not to/from the LCH-D. An LCh can link to itself, and two LChs can link to each other. Linking several LChs in a circular fashion is also possible.

Note:

Be aware that when a linked LCh is stopped with STOP_LNK, the bit ENABLE_LNK is disabled and hence the LCh is no longer part of the chain.

The restrictions on linking logical channels are:

- Constraint 1**:**

  It is the responsibility of software to make sure that all NEXTLCH_ID fields and ENABLE_LNK bits are configured before the logical channel chain is started (first LCh is enabled).

  Undefined effects occur if software modifies these fields when the chain already is enabled.

- Constraint 2**:**

  It is the responsibility of software to ensure that only the head of a chained logical channel is enabled. Undefined effects occur if software enables a logical channel that is inside a chain.

- Constraint 3:

  It is the responsibility of software to ensure the chained logical channels have the same priority level; otherwise, undefined effects result.

- Constraint 4**:**

  It is the responsibility of software to make sure that the channel context does not change on the fly for a chained logic channel. The channel context must be set up by the software before the head of the chained logic channel is enabled. It can only be updated when the linked chain is disabled. Otherwise**,** undefined effects occur. An exception to this constraint is the STOP_LNK bit.

- Constraint 5:

  If the chain is configured as a looping chain (for example, the LCh is linked to itself or two LChs are linked to each other), it is the responsibility of software to disable/stop it.

- Constraint 6:

  If the bit STOP_LNK is set while the logical channel is running, then the logical channel is deactivated, the transfer is stopped and no further linked logical channel is activated.

- Constraint 7:

  It is the responsibility of software to set the bit ENABLE_LNK to 0 (or use STOP_LNK) for all the LCh of a chain, if the LCh is to be reused without the chaining capability. Otherwise, activating any of these LChs activates the chain starting from this logical channel.

  In other words, the ENABLE_LNK bit is not reset by hardware at the end of a linked logical channel.

  A linked chain remains linked after the chain has been executed (if STOP_LNK is not used).

- Constraint 8:

  If a synchronized LCh is part of the chain, it is the responsibility of the software to synchronize the LCh chain with the peripheral so DMA requests are not issued before the LCh is enabled.

  A DMA request is ignored if it is received before the synchronized LCh is enabled by the chain.

7.2.9 Logical Channel Preempting

LCh Types Supporting this Feature	2D	P	PD	G	D
	√	√		√	N/A

A logical channel can be preempted on the element boundary, so that the current element and any ongoing burst are fully transferred before the channel gets preempted.

Preemption occurs when a high-priority LCh suspends a low-priority LCh. This happens if no other PChs are free when the high-priority LCh gets active. For more details, see section 7.2.6, *Logical Channel Scheduling.*

A synchronized channel goes into an interleaved state once a DMA request is served, and it waits for a new DMA request. It goes into a preempted state once a higher priority logical channel is activated on the same physical channel.

A nonsynchronized channel goes into a preempted state once a higher logical channel is activated on the same physical channel.

A suspended (preempted) low-priority LCh always has a higher priority than a lower priority LCh to be scheduled later. In other words, the preempted LCh does not have to be rearbitrated again. However, a suspended LCh takes over the next free PCh. This means that the same PCh is not necessarily to be used if the preempted LCh is of an LCh type supported by several PChs.

An active, preempted, synchronized LCh (the preempted LCh that received a DMA request and was active before preemption) does not need a new DMA request. The LCh continues the transfer as soon as a PCh becomes free.

If a new DMA request is received when a synchronized LCh is preempted, an event drop is issued and the LCH is disabled. If the DROP_IE bit is set to 1 in the DMA_CICR register, an interrupt is generated and the DROP bit in the DMA_CSR register is set.

The content in the FIFOs in the physical channels is always transferred before the logical channel releases the physical channel. When a physical channel is preempted, it is always on the source element boundary. Therefore, the physical channel drains the FIFO data to the destination before the LCh releases the PCh.

7.2.10 Addressing Modes

An addressing mode is an address computation algorithm that a DMA channel uses to find where to access data. The system DMA supports four types of addressing modes:

- ❏ Constant index mode
- ❏ Postincremented mode
- ❏ Single-indexed (element Index) mode
- ❏ Double-indexed (element and frame index) mode

Based on LCh types, the summary of addressing modes is as follows:

Table 7–5. Logical Channel Type Address Mode Summary

Addressing Mode	LCh-P and LCh-PD		LCh-2D	LCh-G
	Peripheral Port	Memory Port		
Constant	√	√	√	√
Postincremented	√	√	√	√
Single-indexed		√	√	√
Double-indexed		√	√	√

The amount of data (block size) to be transferred is programmed in bytes.

The data block to transfer is split into frames and elements. The data block size in bytes can be expressed as:

BSi = FN x EN x ES

where:

BSi: The block size in bytes

FN: The number of frames in the block, $1 \leq FN \leq 65535$ (unsigned) which is defined in the DMA channel frame number register (DMA_CFN).

EN: The number of elements per frame, $1 \leq EN \leq 65535$ (unsigned) which is defined in each DMA channel element number register (DMA_CEN).

ES: The number of bytes per element, ES $\in$ {1, 2, 4}, which is defined with the field DATA_TYPE in register DMA channel source destination parameters (DMA_CSDP).

A frame size in bytes: FSi = ES x EN.

Setting FN or EN equal to 0 is not allowed, because it causes undefined effects.

An element (data type) can be:

- 8-bit scalar data, s8 (which means that ES = 1)
- 16-bit scalar data, s16 (which means that ES = 2)
- 32-bit scalar data, s32 (which means that ES = 4)

To set up a channel for a transfer, the software must program two addressing modes:

- The source addressing mode, source index size
- The destination addressing mode, destination index size

Both address modes for source and destination are independent. For example, to transfer data from TIPB port to internal memory, the source-addressing mode can be constant and the destination mode can be postincremented.

However, the number of frames, the number of elements, and the element size are the same for both source and destination.

Note:

The channel source and destination start addresses are configured in separate registers: DMA_CSSA_L/U and DMA_CDSA_L/U, respectively.

Frame index and element index are configured in separate registers for both source and destination: DMA_CSFI and DMA_CSEI, respectively, for source and DMA_CDFI and DMA_CDEI, respectively, for destination.

7.2.10.1 Data Alignment

During a transfer, the start address and all the addresses computed by the DMA must be aligned on the type of data transferred (data type):

- ❏ If the data type is s8 (8-bit scalar data), addresses can have any value.
- ❏ If the data type is s16 (16-bit scalar data), addresses must be aligned on 16-bit word boundary (the lowest bit of the address always 0).
- ❏ If the data type is s32 (32-bit scalar data), addresses must be aligned on 32-bit word boundary (the two lowest bit of the address always 00).
- ❏ If it is 4x32 burst access with s32 (32-bit scalar data), addresses must be aligned on burst boundary (the four lowest bit of the address always 0000).
- ❏ If frame index is used, it must always produce addresses aligned on data type boundary.
- ❏ If element index is used, it must always produce addresses aligned on data type boundary.
- ❏ Transfer block size must be aligned on the data type boundary.

Failure to follow these rules generates undefined operation due to wrong endianism switching.

In summary, the general constraints are:

Start address (SA), Element_Index (EI), and Frame_Index (FI) must be aligned on data type (ES):

SA mod ES= 0
(FI-1) mod ES= 0
(EI-1) mod ES= 0

7.2.10.2 Constant Addressing Mode

The address remains constant for each element to be transferred

A(n+1) = A(n)

where:

A(n): Byte address of the element n within the transfer

Note:

A(0) is always equal to the start address of the transfer. The same goes for all addressing modes.

7.2.10.3 Postincremented Addressing Mode

Address is postincremented by element size (ES).

$$A(n+1) = A(n) + ES$$

where:

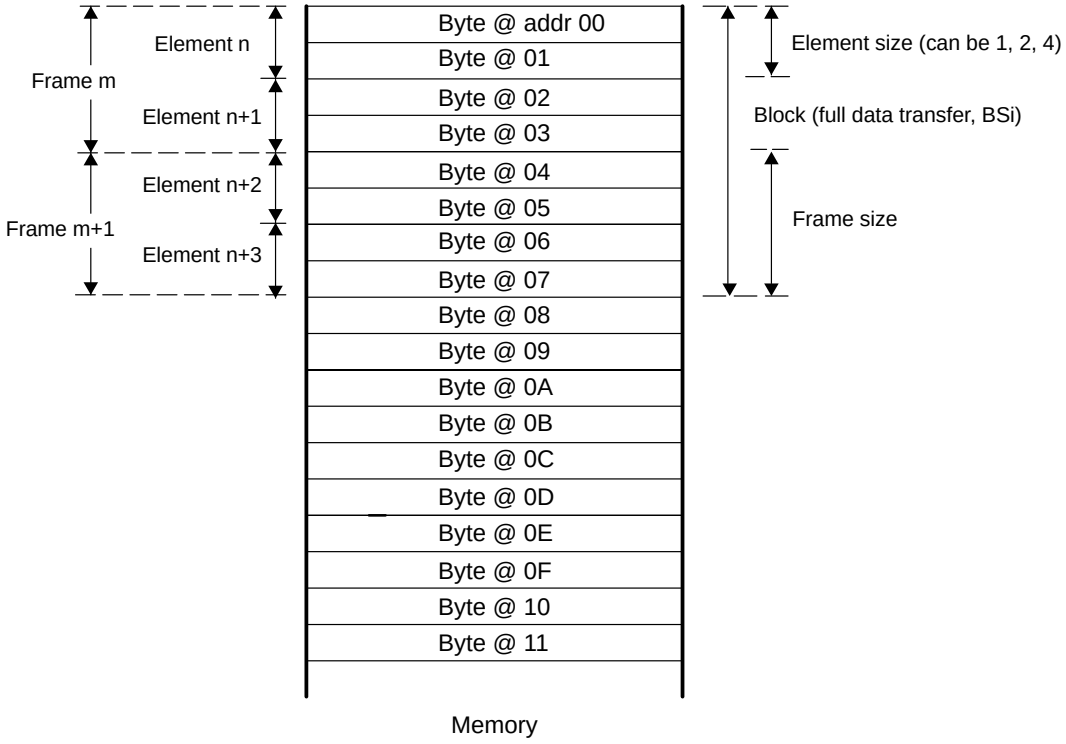
An: Byte address of the element n within the transfer

ES: Element size in bytes, ES $\in$ {1, 2, 4}

Figure 7–4 illustrates how memory accesses are performed if an LCh is configured with postincrement addressing mode and:

- Starting address: 00
- Element size: 2 (16 bits)
- Element number: 2
- Element index: Ignored
- Frame number: 2
- Frame index: Ignored

Figure 7–4. Postincremented Addressing Mode Memory Accesses



7.2.10.4 Single-Indexed Addressing Mode

Address is postincremented by element size and an element index (expressed in bytes).

$A(n+1) = A(n) + ES + (EI - 1)$

$EI = ((Stride\ EI - 1) * ES) + 1$

where:

A(n): Byte address of the element n within the transfer

ES: Element size in bytes, $ES \in \{1, 2, 4\}$

EI: Element index in bytes, specified in a configuration register, $-32768 \leq EI \leq 32767$

Stride EI: Number of elements between the beginning of current element n and the beginning of next element, n+1.

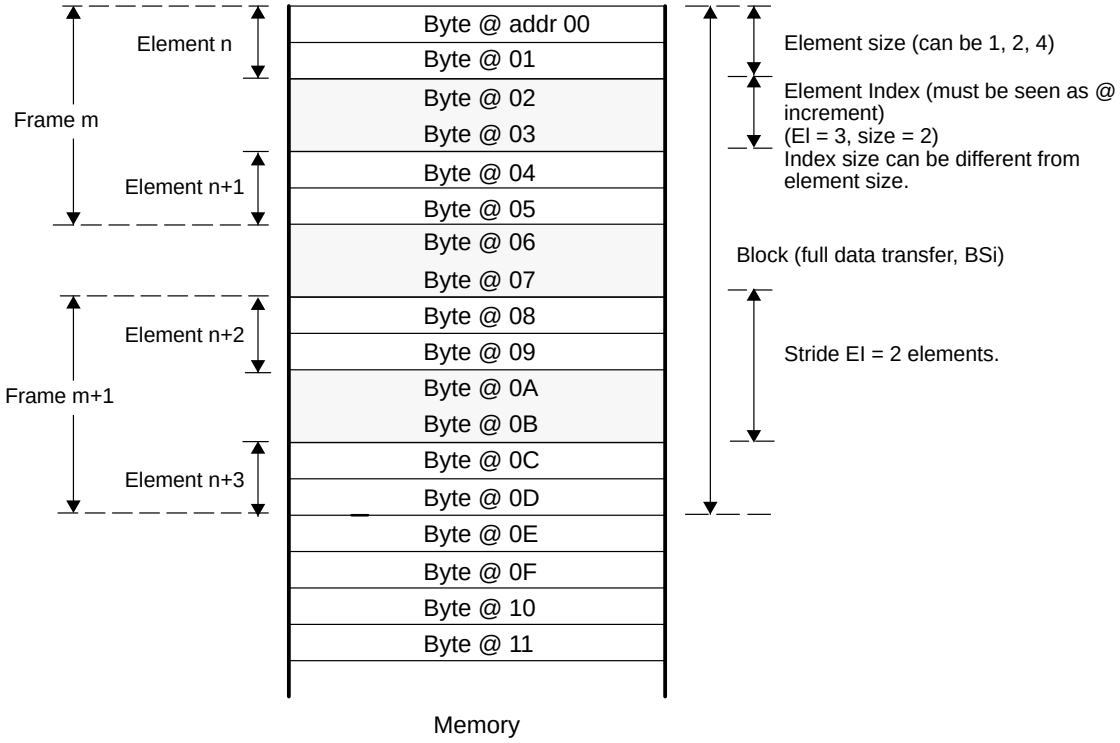
Note:

EI = 1 results in consecutive element accesses (the same behavior as with the postincremented addressing mode).

Figure 7–5 illustrates how the memory accesses are performed if a LCh is configured with single-indexed addressing mode and:

- Starting address: 00
- Element size: 2 (16 bits)
- Element number: 2
- Element index: 3
- Frame number: 2
- Frame index: Ignored

Figure 7–5. Single-Indexed Addressing Mode Memory Accesses



Note: EI is used between frames for single-indexed addressing mode. This gives this figure three equal, big strides between elements.

7.2.10.5 Double-Indexed Addressing Mode

The address is incremented by element size and a frame index if the end of the current frame is reached. The address is incremented by element size and an element index if the end of the current element is reached but the end of frame is not reached.

When not at end of a frame or transfer, that is, as long as element counter $\neq 0$:

$$A(n+1) = A(n) + ES + (EI - 1)$$

When at end of a frame but not at the end of the transfer, that is, as long as element counter = 0 and frame counter $\neq 0$:

$$A(n+1) = A(n) + ES + (FI - 1)$$

Calculate element and frame index as follows:

$$EI = ((\text{Stride EI} - 1) * ES) + 1$$
$$FI = ((\text{Stride FI} - 1) * ES) + 1$$

where:

A(n): Byte address of the element n within the transfer

ES: Element size in bytes, ES $\in$ {1, 2, 4}

EI: Element index in bytes, specified in a configuration register, $-32768 \leq EI \leq 32767$.

Stride EI: Number of elements between the beginning of current element n and the beginning of next element, n+1.

Element: Counter that is (re)initiated with the number of element per frame or per transfer. Decreased by 1 for each element transferred. Initial value is configured in register DMA channel element number, DMA_CEN.

FI: Frame index in bytes, specified in a configuration register, - 32768 ≤ FI ≤ 32767

Stride FI: Number of elements between the beginning of last element of the current element and the beginning of first element of the next frame.

Frame counter: Counter that is (re)initiated with the number of frames per transfer. Decreased by 1 for each frame transferred. Initial value is configured in register DMA channel frame number, DMA_CFN.

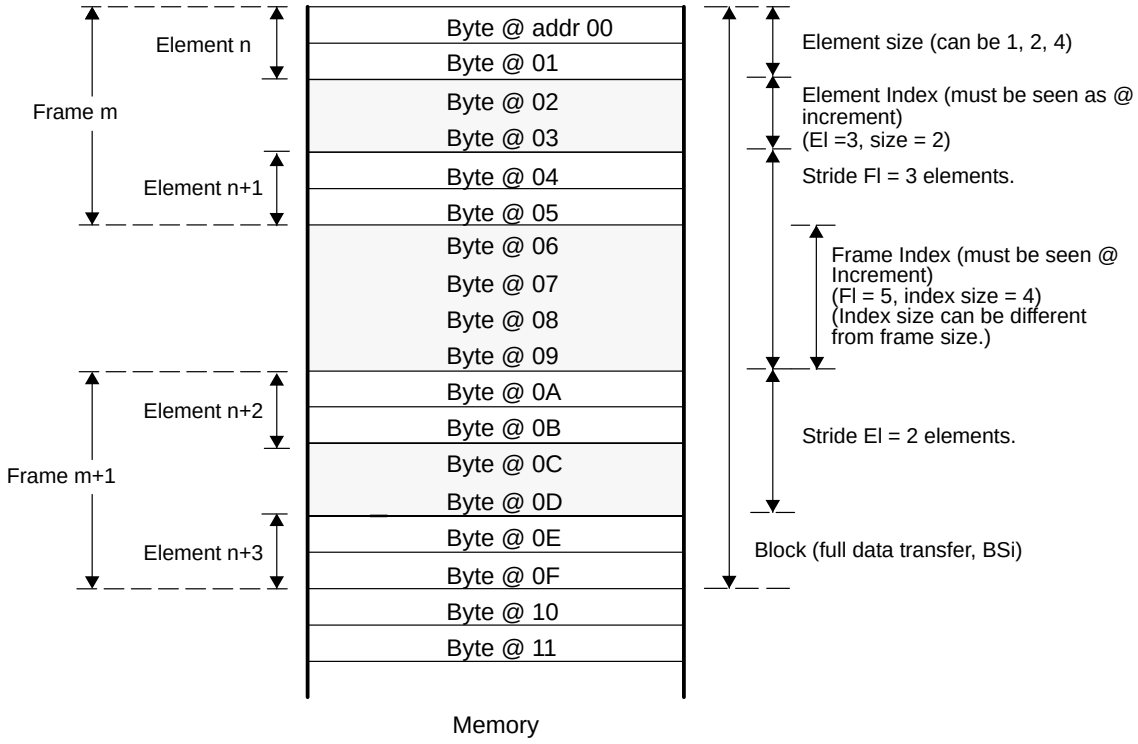
Note:

FI = 1 provides consecutive element accesses at end of frames. If EI = 1, the behavior is the same as with the postincremented addressing mode.

Figure 7–6 illustrates how the memory accesses are performed if an LCh is configured with double-indexed addressing mode and:

- Starting address: 00
- Element size: 2 (16 bits)
- Element number: 2
- Element index: 3
- Frame number: 2
- Frame index: 5

Figure 7–6. Double-Indexed Addressing Mode Memory Accesses



Note:

For all addressing modes, independent element and frame indexing were not supported on OMAP 3.0 and 3.1. The EI and FI configurations are dependent on the OMAP3_1_Compatible_Disable bit in the DMA_CCR register.

If DMA_CCR[OMAP3_1_Compatible_Disable] = 0:

- EI(source) = EI(destination) = DMA_CSEI register
- FI(source) = FI(destination) = DMA_CSFI register

If DMA_CCR[OMAP3_1_Compatible_Disable] = 1:

- EI(source) = DMA_CSEI register
- EI(destination) = DMA_CDEI register
- FI(source) = DMA_CSFI register
- FI(destination) = DMA_CDFI register

7.2.11 Data Packing and Bursting

LCh Types Supporting this Feature	2D	P	PD	G	D
	√	√	√	√	√

A DMA channel has the capacity to:

- Pack several consecutive byte accesses in a single word16 or word32 access. This increases the transfer rate. For a channel, the decision to pack to its source port is dependent on the source port access capability and whether source packing is enabled by software. The same goes for the destination port. Software can control packing for both destination and source using bits SRC_PACK and DST_PACK in the DMA channel source destination parameters register (DMA_CSDP).
- Split a single access, defined by its DMA port capability, into subset access size if the address of single access is not aligned on access size. For example:
  - Split a word16 transfer into several byte accesses, if access address is not aligned on word16 address
  - Split a word32 access into single word accesses, if access address is not aligned on word32 boundary
  - Split 4x32-bit burst access into several byte/word/double-word accesses, if access address is not aligned on 4x32-bit burst access boundary
- Burst 4×32 bits if bursting is enabled and the accessed source or destination port supports it. Software can enable or disable bursting using bitfields SRC_BURST_EN and DST_BURST_EN in the DMA_CSDP register. These bit-fields are ignored if the accessed port does not support bursting and hence result in single accesses.

Table 7–6 summarizes the possible transfer configurations and shows the cases where packing and splitting are performed.

Table 7–6. Packing and Splitting Summary

Data Type	Port Access Capability	Packing / Splitting
s8	s8	–
	s16	pack 2 x s8 => 16
	s32	pack 4 x s8 => 32
s16	s8	split s16 => 2 x 8
	s16	–
	s32	pack 2x s16 => 32
s32	s8	split s32 => 4 x 8
	s16	split s32 => 2 x 16
	s32	–

To compute the type of an access (8/16/32-bit or 4x32-bit burst) and decide to pack consecutive accesses, an address calculation unit checks:

- Its related DMA port capabilities:
  - Can the port perform bytes, 16-bit, or 32-bit access?
  - Can the port perform 4x32-bit burst access?
- What is allowed by the software in the configuration registers:
  - Is packing allowed?
  - Is bursting allowed?
- The last bits of the address:
  - Is the address even or odd?
  - Is the address word16, word32, 4x32-bit burst aligned?
- The number of elements remaining in the frame
- The synchronized transfer type for the LCh

The restrictions for packing and bursting are as follows:

- The constant address mode automatically disables packing and bursting on the DMA source or destination port where it is used. The other end of the DMA channel can still utilize packing and bursting if that DMA port is not configured to use constant address mode.
- All elements must be transferred continuously within a frame. That means that it cannot be synchronized on element boundary, and if single/dual-indexed addressing mode is used, element index (EI) must be set to 1.
- No packing or bursting can be done over frame boundary. It does not help to set frame index FI to 1.
- No packing or bursting can be done over block boundary.

Once the type of access is decided, the current byte address can be incremented by 1, 2, 4, or 16 to reach the next memory space location. Then, the DMA port checks its physical channel FIFO to see if there is enough data (write access) or enough space (read access) in the FIFO, before issuing the access.

It is assumed that:

- A 16-bit target memory or peripheral also has a byte-write/read capability.
- A 32-bit target memory or peripheral also has a byte and word16 write/read capability.

If this is not the case, then the software must carefully set up the transfer as follows:

- Follow the address alignment rules in Section 7.2.10.1, *Data Alignment.*
- Set the data_type to the same size as the peripheral register width that is the source or the destination of the DMA transfer.
- These rules apply only when the constant address mode is used (which is likely to be the case when a single register is source/destination), and no packing can be enabled.

Packing is controlled by the source and destination packing bits; it packs several elements into a larger access element (word16 or word32). For instance, if the data_type is set to s16, the source port supports 32-bit accesses, the source packing is enabled, and the address is aligned on word32, then the source port makes two 16-bit accesses and packs them to a word32 before sending word32 to the destination port. The packing bit is ignored if the source port uses the constant addressing mode and the packing bit is enabled.

Another sort of packing groups bytes or word16 accesses within an element. This is always done if the LCh is configured as such. Packing within an element is done if the access types are smaller than the element sizes (data_type). For instance, if the data type is set to s32 but the source target port only supports 16 bits, two consecutive word16 source accesses are packed to build the word32 element. This element is then sent to the destination port. This is done independently of the source and destination packing bits.

Packing within an element is supported even if the constant addressing mode is used or if EI is different from 1.

Example 7–1 shows a case of packing enabled.

Example 7–1. Packing Enabled

A detailed example (see Table 7–7 and Table 7–8) of packing 2 × s16 => 32 is:

- A logical channel is set up for a transfer with the following parameters for its source:
  - Number of frames in the block: FN = 2 elements
  - Number of elements per frame: EN = 5 elements
  - Type of data: s16
  - Frame index in bytes: FI = 13 bytes
  - Element index in bytes: EI = 1 byte
  - Source start address: SA = 2 bytes
  - Source addressing mode: Double-indexed addressing mode
  - The source port is a 32-bit port with byte/word16/word32 access capability.
  - Packing is enabled but burst is disabled.
  - Memory block to transfer is: element i, j (where j is the element number of frame i)
  - i and j start with the highest number and decreases, so the first element written is element 2,5.

Table 7–7. Channel Data Block to Transfer

Address	Byte 0	Byte 1	Byte 2	Byte 3
0			Element 2,5	
4	Element 2,4		Element 2,3	
8	Element 2,2		Element 2,1	
12				
16				
20				
24	Element 1,5		Element 1,4	
28	Element 1,3		Element 1,2	
32	Element 1,1			
36				
40				

The computed addresses and access type are given in Table 7–8:

Table 7–8. Channel Addresses and Access Types

Access	Frame Number j	Element Number i	Address	Access Type
0	2	5	2	16 bits
1	2	4	4	32 bits
2	2	2	8	32 bits
3	1	5	24	32 bits
4	1	3	28	32 bits
5	1	1	32	16 bits
6	End of transfer			

In this example, the first word16 is not packed to word32 because the address is not aligned on word32. The next accesses are packed because packing is enabled, constant addressing is not used, EI = 1, and the address is aligned on word32.

Example 7–2. Packing Plus Burst Enabled

Example 7–2 illustrates an example (see Table 7–9 and Table 7–10) with source packing 2 × s16=> 32 plus burst 4 × 32-bit enabled,. This example results in exactly the same behavior as Example 7–1 because the burst capability is ignored. It is ignored because when the address is on a 4 × 32-bit aligned boundary, less than 4 × 32 bits are left in the frame.

If the frame size is increased by doubling the element size, EN = 10, (assuming that the targeted source support 4 × 32-bit burst), a logical channel is set up for a transfer with the following parameters for its source:

- Number of frames in the block: FN = 2 elements
- Number of elements per frame: EN = 10 elements

- Type of data: s16
- Frame index in bytes: FI = 9 bytes
- Element index in bytes: EI = 1 byte
- Source start address: SA = 14 bytes
- Source addressing mode: Double indexed addressing mode
- The source port is a 32-bit port with byte/word16/word32/4x32-bit burst access capability.
- Source packing is enabled.
- Source burst is enabled.
- Memory block to transfer is: element i, j (where j is the element number of frame i).
- Note that i and j start with the highest number and decreases, so the first element written is element 2,10.

Table 7–9 shows the channel blocks to transfer.

Table 7–9. Channel Data Block to Transfer

Address	Byte 0	Byte 1	Byte 2	Byte 3
12			Element 2,10	
16	Element 2,9		Element 2,8	
20	Element 2,7		Element 2,6	
24	Element 2,5		Element 2,4	
28	Element 2,3		Element 2,2	
32	Element 2,1			
36				
40			Element 1,10	
44	Element 1,9		Element 1,8	
48	Element 1,7		Element 1,6	
52	Element 1,5		Element 1,4	
64	Element 1,3		Element 1,2	
68	Element 1,1			
72				

Table 7–10 lists the computed addresses and access types.

Table 7–10. Channel Addresses and Access Types

Access	Frame Number j	Element Number i	Address [byte]	Access Type
0	2	10	10	16 bits
1	2	9-2	12	4x32 bits
2	2	1	28	16 bits
3	1	10	42	16 bits
4	1	9,8	44	32 bits
5	1	7,6	48	32 bits
6	1	5,4	52	32 bits
7	1	3,2	64	32 bits
8	1	1	68	16 bits
9	End of transfer			

In this example, the first word16 is not packed to word32 because the address is not aligned on word32. The next accesses bursts 4×32 bits, because burst is enabled, constant addressing is not used, EI = 1, address is 4×32 bits burst aligned, and enough data is left in the frame (16 bytes). All other transfers are word16 single accesses or 32-bit packed accesses. There are no more bursts because the next time the address is burst aligned, there is not enough data left in the frame (less than 16 bytes).

7.2.12 Interrupt Generation

LCh Types Supporting this Feature	2D	P	PD	G	D
	√	√	√	√	√

Each generic LCh can generate six different interrupts. The following interrupt sources can be programmed:

- End of block: The last byte of the transfer has been written into destination, enabled with bit BLOCK_IE in register DMA_CICR.
- End of frame: The last byte of the current frame has been written into destination, enabled with bit FRAME_IE in register DMA_CICR.
- Half of frame: The middle byte of the current frame has been written into destination, enabled with bit HALF_IE in register DMA_CICR.
- Start of last frame: The first word of the last frame has been written into the destination, enabled with bit LAST_IE in register DMA_CICR.
- Request collision: A new DMA request occurred before the end of service of the previous request, enabled with DROP_IE bit in register DMA_CICR.
- Time-out error: An access error occurred in the transfer to the source or the destination, enabled with bit TOUT_IE in register DMA_CICR. The

countdown values are configured at the source and destination ports. No countdown value can be specified in the system DMA.

Note:

Configure the interrupt(s) to be generated for each generic LCh in the channel interrupt control register (DMA_CICR). The LCh-D supports only two kinds of interrupts. See Table 7–65, *DMA LCD Control (DMA_LCD_CTRL)* for more information.

Each DMA logical channel can generate an interrupt to the MPU to reflect the transfer status. Each system DMA logical channel has a dedicated interrupt line to the MPU. All DMA interrupts are level-sensitive interrupts; that is, an interrupt line is held active-low until the MPU reads the associated logical channel status register.

No new interrupts can be generated until the status register is read and thereby cleared. For each logical channel, all the interrupt sources are connected together to generate one interrupt. When an interrupt is issued by a logical channel, its status register, DMA_CSR, is set to record the interrupt cause if its associated DMA_CICR enable bit is set. The MPU interrupt service routine (ISR) can read this channel status register to find the sources of the interrupt. The status bits are automatically cleared after they are read by the MPU.

The interrupt enable bits are used to choose the events that trigger the DMA channel to send an interrupt to the processor. There are two classes of events:

- Error events: errors during the transfer. These are time-out and request collision.
- Status events: DMA transfer status during DMA channel transfers. These are start of last frame, half of frame, end of frame, and end-of-block.

When an error event occurs and the corresponding interrupt enable bit is enabled, the following happens:

- The status register bit is activated.
- An interrupt is generated.
- The logical channel is disabled.
- The physical channel is released.

If there is an error but error interrupt is not enabled, no event is generated, the status register bit is not set, and no interrupt is generated. However, the LCh is disabled.

When a status event occurs and the corresponding interrupt enable bit is enabled, the following happens:

- The status register bit is activated.
- An interrupt is generated.
- No new interrupts can be generated until the status register is read and thereby cleared.

Note:

It is *important* to be aware of the following:

- One read in the status register clears all the status bits.
- No read in the status register keeps all the status bits and the DMA interrupt output stays active low.

Therefore, always read the associated status register for each DMA interrupt received; otherwise, interrupts can be missed.

7.2.12.1 System DMA Interrupt Mapping

The OMAP730 system DMA has one interrupt line per channel. Table 7–11 shows the interrupt line per LCh mapping. This mapping is only true if the OMAP3_1_Compatible_Disable bit is set to 1 in the DMA_CCR register.

Table 7–11. Interrupt Mapping per LCh

Interrupt Line	LCh Mapping
Interrupt line 0	LCH_0
Interrupt line 1	LCH_1
Interrupt line 2	LCH_2
Interrupt line 3	LCH_3
Interrupt line 4	LCH_4
Interrupt line 5	LCH_5
Interrupt line 6	LCH_6
Interrupt line 7	LCH_7
Interrupt line 8	LCH_8
Interrupt line 9	LCH_9
Interrupt line 10	LCH_10
Interrupt line 11	LCH_11
Interrupt line 12	LCH_12
Interrupt line13	LCH_13
Interrupt line14	LCH_14
Interrupt line 15	LCH_15
LCD interrupt line	LCH_D

Each logical channel has an associated status register, DMA_CSR, where the source of the interrupt is shown.

7.2.13 DMA Request Input Protection

The system DMA uses either the edge-sensitive scheme or the transition-sensitive (edge + pulse_width) scheme to latch the DMA request input signals to trigger a synchronized logical channel transfer. The choice per DMA request to either use edge- or transition-sensitive scheme is controlled by OMAP input pins, gl_dmareqedgeen_tr[30:0], which are tied on/off on OMAP top level.

DMA edge-sensitive scheme is selected if input gl_dmareqedgeen_tr is tied to 1. Hence, it is not controlled by software. This section is included for information only.

7.2.14 DMA Idle Modes

The system DMA can automatically enter an idle mode dynamically as soon as it is not active, or it can be put into idle/suspend mode on request from MPU via the clock generator module.

7.2.14.1 Dynamic Idle Mode

To save power, the system DMA has a built-in dynamic idle mode that is enabled by setting the CLOCK_AUTOGATING_ON bit in the global control register, DMA_GCR.

The system DMA clock domain is split into several subdomains in this mode; each one of them becomes disabled if not used. This mode does not add any extra latency to the system DMA.

All internal clocks are in idle mode (disabled) when the following is fulfilled:

1) CLOCK_AUTOGATING_ON = 1 in DMA_GCR

2) No nonsynchronized LChs are enabled.

3) Either no synchronized LChs are enabled, or synchronized LChs are enabled but no DMA request is received or pending.

The system DMA wakes up if software enables a new LCh or if a DMA request is received. The system DMA also wakes up temporarily if the MPU or OCP-I wants to read or write to any of the registers.

7.2.14.2 System IDLE Request

The system DMA can be put into idle mode, that is, sleep mode, when the system needs to go into power saving mode. See Chapter 4, *Clock Generation and Reset Management* for more information on prerequisite steps to put the system and/or the system DMA in idle. An idle request signal is sent to the DMA when the system wants it to go idle.

As soon as the DMA detects an idle request, it enters idle mode when all PChs are free (including PCh-D). All nonsynchronized LChs, synchronized LChs without DMA request detected, and suspended LChs are unscheduled PChs. The DMA enters idle mode even if one of the LCD controllers is assigned PCh–D but is in sleep mode.

The DMA temporarily wakes up if:

- A synchronized LCh's DMA request is detected
- There are any writes to generic LCh registers
- Any of the LCD controllers gets active

The DMA goes back to idle mode when none of these are true. This also means that, as long as the system is requesting the system DMA to be in idle, nonsynchronized LChs can be enabled but never activated. Synchronized LChs can be enabled and hence activated at DMA request during this time.

The DMA finishes pending accesses of physical channels and resumes the logical channel transfer when released from idle. Therefore, it is the responsibility of the software to ensure that when software issues an IDLE instruction, all logical channels have been serviced.

The power-saving difference between the dynamic idle mode and the system idle request is provided by the clock tree between the clock generation module and the system DMA clock input.

7.2.15 DMA Debug State

During debug mode the MPU can send a request to suspend the DMA. This is useful if, for instance, the MPU is halted by a breakpoint. How the system DMA responds to this request is controlled by software by configuring the FREE bit in the DMA global control register, DMA_GCR. When the FREE bit is set to 0, all current transfers are suspended when a request is received from the MPU. The transfers resume when the MPU releases the debug request signal. If the FREE bit is set to 1, The DMA continues to run as usual, even if the MPU is sending a debug request signal. The channel status of the DMA interrupts is read on the DMA_CSR register bits. In contrast with the functional mode, the DMA_CSR bits are not cleared after an emulation read (identified by its emulation access qualifiers).

7.2.16 Other Logical Channel Features

The LCh-G type channel transfers data and also provides hardware with 2-D graphic data processing features to improve 2-D graphic processing speed and graphic quality. It supports the following graphic features:

- Transparent copy
- Constant fill (or constant solid color fill)
- Rotation

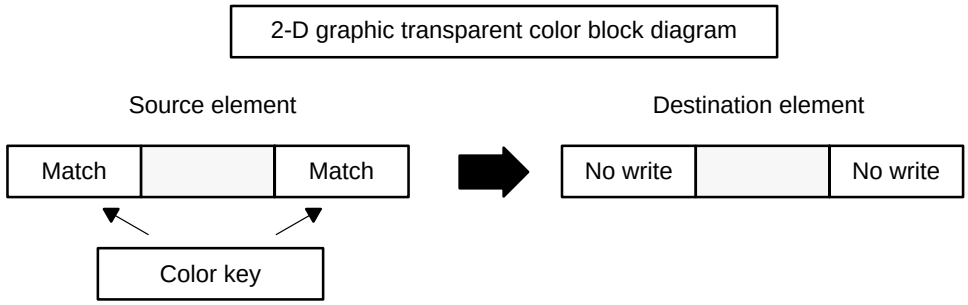
LCh-2D also supports constant fill and rotation. Rotation is supported by all LCh types supporting separate element and frame indexed addressing (2 dimension addressing)

7.2.16.1 Transparent Copy

LCh Types Supporting this Feature	2D	P	PD	G	D
				√	

It is often desirable to transfer irregular shapes, especially in game software. The system DMA supports the *color key* feature for 8 BPP, 16 BPP, and 32 BPP from source to destination; that is, each element of channel source is compared to a color key and those data bits (pixels) that match the color key are not written to the destination.

Figure 7–7. 2-D Transparent Color Block Diagram



This feature is enabled by setting the Transparent_Copy_Enable bit in the channel control 2 register, DMA_CCR2. If this bit is enabled, the DMA color parameter registers, DMA_COLOR_L and DMA_COLOR_U, are used to specify the color key.

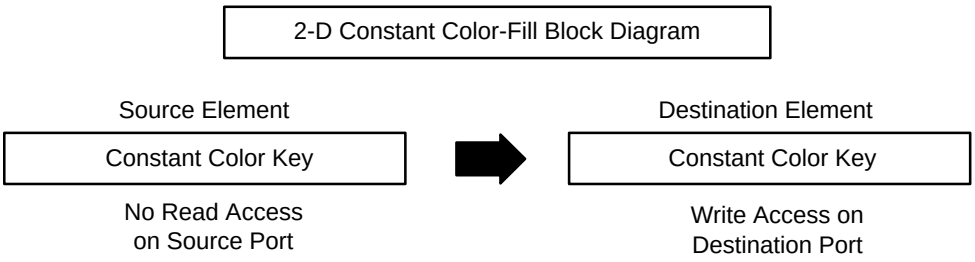
If the data type, ES, is 8 or 16 bits, only the DMA_COLOR_L register is used; if the data type is 32 bits, the DMA_COLOR_U register is also used. The system DMA always compares all bits (8, 16, or 32 bits). If other color depths (other than 8, 16, or 32) are used, it is the responsibility of the software to ensure that unused upper bits are constant. See the register description for more detailed information.

7.2.16.2 Constant Fill

LCh Types Supporting this Feature	2D	P	PD	G	D
	√			√	

This feature allows filling a region with a constant (solid color) or a pattern by repeating the data horizontally and/or vertically in the region. The system DMA writes to the destination a constant color/value defined in a register. There is no read access on any system DMA source port.

Figure 7–8. 2-D Constant Color Fill Block Diagram



This feature is enabled by setting the bit CONSTANT_FILL_ENABLE in the channel control register 2, DMA_CCR2. If this bit is enabled, the register DMA color parameter, DMA_COLOR_L and DMA_COLOR_U, is used to specify the color key.

If the data type, ES, is 8 or 16 bits, only the DMA_COLOR_L register is used; if the data type is 32 bits, register DMA_COLOR_U is also used. See the register description for more detailed information.

7.2.16.3 Rotation

LCh Types Supporting this Feature	2D	P	PD	G	D
	√	√	√	√	√

All LCh types with independent source and destination indexing support four types of rotation: 0°, 90°, 180°, and 270°. This can be established with the help of double-indexed addressing; that is, separate element/frame indexing. See section 7.2.10, *Addressing Modes*, for details of this addressing capability.

7.2.17 Compatibility with OMAP 3.0 and 3.1

After reset, the DMA and all LChs default to the OMAP 3.0/3.1 programming model. This is characterized by a reduced feature set, no LCH chaining, fewer LChs, program and active register sets per LCh, and a restricted number of interrupts (see Table 7–12).

New code for the DMA must use the programming model presented in previous sections of this document. Future versions of the DMA will be optimized for this programming model and new features will be added to it rather than the 3.0/1-compatible mode.

The DMA described here does not exhibit the same cycle-by-cycle behavior as previous implementations in OMAP 3.0/3.1 and hence time critical applications must be verified when upgrading.

Two orthogonal mechanisms control the compatibility modes of the system DMA:

- A global register bit controls the DMA register and interrupt mapping.
- A register bit per logical channel controls the disabling/enabling of the OMAP 3.0/1 programmer's model.

Note:

The OMAP730 is configured from reset to be OMAP 3.1 binary compatible. For more details of the 3.1 programming model, see the documentation for OMAP 3.1-based devices and the OMAP1509 TRM.

When in OMAP 3.1-compatible mode, PCh-0 and -1 are dynamically shared to serve the generic LChs.

To enable the new features introduced beyond OMAP P 3.1, the following configuration register bits must be configured:

- ❏ OMAP 3_1_MAPPING_DISABLE in DMA_GSCR
- ❏ OMAP3_1_COMPATIBLE_DISABLE in DMA_CCR/DMA_LCD_CCR

Table 7–12. Summary Table of Different Compatibility Modes

Global Register Bit OMAP3_1_ MAPPING_ DISABLE	Global Effects	LCh Register Bit OMAP3_1 COMPAT-IBLE_DIS-ABLE	Per-LCH Effects
0	Reset values 9 LCh + 1 LCh-D available OMAP 3.1 configuration register mapping OMAP 3.1 interrupt line mapping 7 interrupt lines	0	Only OMAP 3.1 programming model/feature set is supported. PCh-2 cannot be used, so only the LCD channel and two generic logical physical channels are available to these LChs.
		1	All features are available as. described in the remainder of this document. All four physical channels are available to these LChs.
1	16LCh + 1 LCh-D available OMAP3.2 configuration register mapping OMAP3.2 interrupt line mapping 17 interrupt lines	0	Only OMAP 3.1 .programming model/feature set is supported. PCh-2 cannot be used, so only the LCD channel and two generic physical channels are available to these LChs.
		1	All features are available as earlier described in this chapter. All four physical channels are available to these LChs.

Note: The LCh compatibility bit, OMAP3_1_compatible_disable, is configured per logical channel; so, it is possible to mix the programmer model for different logical channels.

The OMAP 3_1_MAPPING_DISABLE bit in register DMA_GSCR is a global register bit that enables/disables OMAP 3.1 register mapping and interrupt line mapping. DMA_GSCR is a global register and must therefore be configured before any LCh is enabled. The DMA_CCR/DMA_LCD_CCR registers are configured per LCh, which enables having different programming models for the LChs. The register and interrupt line mapping must be configured first and affect all the LChs, but the OMAP 3.0/3.1 and OMAP730 programming models can be mixed among the LChs.

7.2.17.1 Autoinitialization of Logical Channels

Autoinitialization is supported in OMAP 3.0/3.1 compatible mode. For more details, see the *OMAP1509 TRM.*

If a logical channel is autoinitialized, then the logical channel is automatically enabled again when a transfer ends. A new or old logical channel configuration register set is loaded, and a new block of data is transferred.

Autoinitialization means that the logical channel gets reenabled. It does not mean that the physical channel is kept for the logical channel. The logical channel must return to the same scheduling schemes as normal before it gets access to a physical channel.

The autoinitialization bit is used to set the DMA into autoinitialization mode. The end_prog and repeat bits are used to control DMA behavior while in autoinitialization mode. Bits AUTO_INIT, END_PROG, and REPEAT are located in the channel control register, DMA_CCR. Table 7–13 summaries the autoinitialization mode.

Table 7–13.Autoinitialization Configuration Bits Summary

Auto-init	Repeat	End_prog	Autoinitialization Behavior
0	x	x	No autoinitialization Waits until software sets enable = 1 to enable the LCh and loads the PCh with its programming register set.
1	0	0	As both repeat and end_prog bits equal 0, the LCh is still enabled, but pending. At the end of the current transfer, the LCh channel waits until repeat or end_prog = 1 to re-activate itself again. When end_prog = 1, the programming register set is copied to the active register set: A new context is programmed.
1	x	1	At the end of the current transfer, the LCh immediately loads the PCh with its programming register set when PCh is granted (end_prog = 1 allows loading the new LCh context, disregarding the repeat bit). The channel reinitializes itself and starts a new transfer with the new context.
1	1	0	The LCh reinitializes itself at the end of the current transfer and starts a new transfer with the previous channel context (active register set).

This table differs slightly from what is implemented in OMAP 3.0/3.1 hardware, where the programming register set is always loaded when End_prog or Repeat is set to 1.

7.2.17.2 OMAP3.0/3.1 System DMA Interrupt Mapping Rule

The OMAP 3.1 system DMA has nine channels plus one LCD channel which share seven interrupt lines; the OMAP730 system DMA has one interrupt line per channel. In order to be backward compatible, the following interrupt mapping modes have been implemented.

To control the mappings:

OMAP730 Mapping: DMA_CCR.OMAP_3_1_MAPPING DISABLE = 1

OMAP 3.1 Mapping: DMA_CCR.OMAP_3_1_MAPPING DISABLE = 0

Table 7–14. Interrupt Mapping per LCh for Both Compatible Modes

Interrupt Line	OMAP730 Mapping	OMAP 3.1 Mapping
Interrupt line 0	LCH_0	LCH_0 and LCH_6
Interrupt line 1	LCH_1	LCH_1 and LCH_7
Interrupt line 2	LCH_2	LCH_2 and LCH_8
Interrupt line 3	LCH_3	LCH_3
Interrupt line 4	LCH_4	LCH_4
Interrupt line 5	LCH_5	LCH_5
Interrupt line 6	LCH_6	N/A
Interrupt line 7	LCH_7	N/A
Interrupt line 8	LCH_8	N/A
Interrupt line 9	LCH_9	N/A
Interrupt line 10	LCH_10	N/A
Interrupt line 11	LCH_11	N/A
Interrupt line 12	LCH_12	N/A
Interrupt line13	LCH_13	N/A
Interrupt line14	LCH_14	N/A
Interrupt line 15	LCH_15	N/A
LCD interrupt line	LCH_D	LCH_12

In case of simultaneous events in two physical channels that share the same interrupt line, only one interrupt is generated and all the relevant status bits are set.

Each physical channel has a seven-bit status register. When an interrupt is shared by two logical channels, the MPU can read the status from the two channels in one TIPB access. The data read has the format shown in Figure 7–9.

Figure 7–9. DMA Packed Channel Status Register for Compatible Mode

	[13:7]	[6:0]
00	DMA_CSR [LCH_6] [6:0]	DMA_CSR [LCH_0] [6:0]

7.3 LCD Channel

7.3.1 Display Logical Channel

The display logical channel, LCh-D, transfers data to an LCD controller (either OMAP LCD controller or an external one) from a video block buffer stored in memory. In the OMAP730 system, the memory source for the transfer can be L3_OCP_T1, L3_OCP_T2, or EMIFF. These transfers can be arranged to have the source in one or two blocks.

Table 7–15 gives a summary of features for the LCD LCh type.

Table 7–15. Features Summary for LCh-D

Supported Transfer Features		LCh-D (D)
LCh Control	Transfer type:	
	❑ Memory to external LCD controller	√
	❑ Memory to OMAP LCD controller	√
	❑ Nonsynchronized	√
	❑ Synchronized	See Note 1
	Two levels of priority	
	Preemption at element boundary	
	LCh interleaving on DMA request	
	Linking logical channel capability	
	Automatic initialization	√
Addressing	Types of addressing modes for src and dst:	
	❑ Constant mode	
	❑ Postincrement mode	See Note 2
	❑ Single-indexed mode, with separate SRC/DST index	See Note 2
	❑ Double-indexed mode, with separate SRC/DST index	See Note 2
	❑ Supports single and dual block modes	See Note 2
	❑ Supports separate element/frame index and element/frame numbers for dual block mode	See Note 2

Notes: 1) LCh-D can be synchronized with the external LCD controller but not with the OMAP internal LCD controller.
2) The LCh-D destination port is the LCD port. The access address to the display is controlled by the LCD controller. Hence, these features only apply on the source.

The dual-block mode allows concurrent transfer and image processing (re-loading of one block while a second is being processed). Switching from one block to another is achieved by loading the configuration register of the second block buffer and then starting the second block transfer after the first block buffer has been fully transferred. This is automatic, enabling the dual-block feature by setting the BLOCK_MODE bit in the DMA_LCD_CONTROL register.

Separate element/frame index and number of elements between the two buffers are also supported in the dual-block mode.

The LCD channel sends the read request to the relevant port, defined by the LCD_SOURCE_PORT bit in the DMA LCD control register (DMA_LCD CTRL), just as the destination port is selected by the LCD_DESTINATION_PORT in the same register. The destination port can be either the OMAP internal LCD controller or the external LCD controller port.

The dedicated DMA channel contains a FIFO used as an elastic buffer to prevent underflow or overrun to/of the LCD.

Hardware ignores the bottom address register of blocks for dual- and single-block access mode. Each block size is defined by

Block_Size = ES x EN x FN in bytes

where:

ES = Element Size: Number of bytes within an element. Bit-field data_type is configured in register DMA LCD channel source destination parameters (DMA_LCD_CSDP)

EN = Element number (within one frame)

FN = Frame number

In order to support graphic functionality, this DMA LCD channel also supports rotation capability at 0°, 90°, 180°, and 270°, by utilizing its source double index mode.

7.3.2 LCD Channel Addressing Modes

Postincremented, single-indexed (element index), and double indexed (element and frame indexes) addressing modes are supported from the source port side. However, note that there is no write address to compute in the destination for the display channel. The read FIFO address is controlled by the LCD controller, so there is no write address to a port.

7.3.2.1 Source Address and Block Size Alignment

On the destination side, the accesses are fixed to 32 bits for external LCD controller and 16 bits for OMAP LCD controller. There is no address alignment because the LCD controller does not use addresses to access data from the LCD channel.

Source memory accesses can be 32/16/8-bit or 4x32-bit burst accesses. The LCD channel is compliant with the generic logical channel constraint; that is, address must be aligned on data_type.

Data block size must be a multiple of 32 bitS for external LCD controller, and a multiple of the 16 bitS for OMAP LCD controller.

However, the data_type can still be 32/16/8 bitS for both the OMAP and external LCD controllers.

7.3.2.2 Source Address Modes

The four different LCD-channel source-address algorithms use the following notations. The address mode descriptions are as follows:

A(n): Byte address of the element n within the transfer.

ES_B1: Element size in block 1 (in bytes): ES $\in$ {1, 2, 4}

ES_B2: Element size in block 2 (in bytes): ES $\in$ {1, 2, 4}

BS_B1: Block size of block 1 (in bytes)

$BS\_B1 = BB1 - TB1 = ES\_B1 \times EN\_B1 \times FN\_B1$

BS_B2: Block size of block 2 (in bytes)

$BS\_B2 = BB2 - TB2 = ES\_B2 \times EN\_B2 \times FN\_B2$

BB1: Bottom address of block 1

BB2: Bottom address of block 2

TB1: Top address of block 1

TB2: Top address of block 2

DBM: Dual-block mode

EN_B1: Number of elements within a frame of block 1

EN_B2: Number of elements within a frame of block 2

Element_counter_B1: Counter (re)initiated with the number of elements per frame or per transfer inside block 1. Decreased by one at each element transferred. Initial value EN_B1 is configured in register DMA channel element number, DMA_CEN

Element_counter_B2: Counter (re)initiated with the number of elements per frame inside block 2. Decreased by one at each element transferred. Initial value EN_B2 is configured in register DMA channel element number, DMA_CEN

Frame_counter_B1: Counter (re)initiated with the number of frames inside block 1. Decreased by one at each frame transferred. Initial value FN_B1 is configured in register DMA channel frame number, DMA_CFN

Frame_counter_B2: Counter (re)initiated with the number of frames inside block 2. Decreased by one at each frame transferred. Initial value FN_B2 is configured in register DMA channel frame number, DMA_CFN

EI_B1: Block 1 element index in bytes $-32768 \leq EI\_B1 \leq 32767$

EI_B2: Block 2 element index in bytes $-32768 \leq EI\_B1 \leq 32767$

Stride_EI_B1: Number of elements between the beginning of the current element (n) and the beginning of the next one (n+1) in block 1.

Stride_EI_B2: Number of elements between the beginning of the current element (n) and the beginning of the next one (n+1) in block 2.

Stride_FI_B1: Number of elements between the beginning of the last element of the current frame and the beginning of the first element of the next frame in block 1.

Stride_FI_B2: Number of elements between the beginning of the last element of the current frame and the beginning of the first element of the next frame in block 2.

FI_B1: block 1 frame index in bytes:

–2147483648 < FI_B1 < 2147483647

FI_B2: block 2 frame index in bytes:

–2147483648 < FI_B2 < 2147483647

Postincremented Addressing Mode

Address is postincremented by element size (ES_B1 or ES_B2 depending on which block is active if in dual-block mode).

DBM = 0 (one block mode)

Only block 1 is active:

A(0) = TB1
A(n+1) = A(n) + ES_B1 until the end-of-block 1.
Then, when A(n) reaches BB1, A(n+1) = TB1 again

DBM = 1 (dual block mode)

When block 1 is active:

A(0) = TB1
A(n+1) = A(n) + ES_B1 until the end-of-block 1.
Then, when A(n) reaches BB1, A(n+1) = TB2: block 2 becomes active.

When block 2 is active:

A(n) = TB2
A(n+1) = A(n) + ES_B2 until the end-of-block 2.
Then, when A(n) reaches BB2, A(n+1) = TB1: block 1 becomes active again.

Single-Indexed Addressing Mode

Address is incremented by element size and element index. All is expressed in bytes.

DBM = 0 (one block mode)

Only block 1 is active:

A(0) = TB1
A(n+1) = A(n) + ES_B1 + (EI_B1 – 1)
where EI_B1 = ((Stride_EI_B1 – 1) * ES_B1) + 1

A(n) is incremented in the same way until the end-of-block 1. Then, when A(n) reaches BB1, A(n+1) = TB1 again.

Note:

Stride_EI_B1 = 1 (equivalent to EI_B1 = 1) gives consecutive element accesses, hence the same behavior as with the postincremented addressing mode.

DBM = 1 (dual block mode)

When block 1 is active:

A(0) = TB1
A(n+1) = A(n) + ES_B1 + (EI_B1 – 1)
where EI_B1 = ((Stride_EI_B1 – 1) * ES_B1) + 1

A(n) is incremented in the same way until the end-of-block 1. Then, when A(n) reaches BB1, A(n+1) = TB2: block 2 becomes active.

When block 2 is active:

A(n) = TB2
A(n+1) = A(n) + ES_B2 + (EI_B2 – 1)
where EI_B2 = ((Stride_EI_B2 – 1) * ES_B2) + 1

A(n) is incremented in the same way until the end-of-block 2. Then, when A(n) reaches BB2, A(n+1) = TB1: block 1 becomes active again.

Note:

Stride_EI_B1/2 = 1 (equivalent to EI_B1/2 = 1) give consecutive element accesses, hence the same behavior as with the Post_Incremented addressing mode.

In dual-block mode, block 2 can be active first.

Double-Indexed Addressing Mode

Address is incremented by element size and element index if the end of the current frame is not reached.

Address is incremented by element size and frame index if the end of the current frame is reached.

Expressed in bytes.

DBM = 0 (one block mode)

Only block 1 is active:

A(0) = TB1
A(n+1) = A(n) + ES_B1 + (EI_B1 – 1)
where EI_B1 = ((Stride_EI_B1 – 1) * ES_B1) + 1

A(n) is incremented in the same way until the end of the current frame: as long as Element_counter_B1 ≠ 0.

When end of frame (but not end-of-block 1) is reached: Element_counter_B1 = 0 and Frame_counter_B1 ≠ 0:

A(n+1) = A(n) + ES_B1 + (FI_B1 – 1)
where FI_B1 = ((Stride_FI_B1 – 1) * ES_B1) + 1

When A(n) reaches BB1, A(n+1) = TB1 again

DBM = 1 (dual block mode)

When block 1 is active:

A(0) = TB1
A(n+1) = A(n) + ES_B1 + (EI_B1 – 1)
where EI_B1 = ((Stride_EI_B1 – 1) * ES_B1) + 1

A(n) is incremented in the same way until the end of the current frame: as long as Element_counter_B1 ≠ 0.

When end of frame (but not end-of-block 1) is reached: Element_counter_B1 = 0 and Frame_counter_B1 ≠ 0:

A(n+1) = A(n) + ES_B1 + (FI_B1 – 1)
where FI_B1 = ((Stride_FI_B1 – 1) * ES_B1) + 1

When A(n) reaches BB1, A(n+1) = TB2: block 2 becomes active.

When block 2 is active:

A(0) = TB2A
A(n+1) = A(n) + ES_B2 + (EI_B2 – 1)
where EI_B2 = ((Stride_EI_B2 – 1) * ES_B2) + 1

A(n) is incremented in the same way until the end of the current frame: as long as Element_counter_B2 ≠ 0.

When end of frame (but not end-of-block 2) is reached: Element_counter_B2 = 0 and Frame_counter_B2 ≠ 0:

A(n+1) = A(n) + ES_B2 + (FI_B2 – 1)
where FI_B2 = ((Stride_FI_B2 – 1) * ES_B2) + 1

When A(n) reaches BB2, A(n+1) = TB1: block 1 becomes active again.

Note:

Both Stride_EI_B1/2= 1 (equivalent to EI_B1/2 = 1) and Stride_FI_B1/2 = 1 (equivalent to EI_B1/2 = 1) give consecutive element accesses; hence resulting in the same behavior as with the postincremented addressing mode.

In dual-block mode, block 2 can be active first.

7.3.3 DMA LCD Channel Sharing Feature

The LCD channel in OMAP730 system DMA supports two LCD controllers. Only one LCD controller can use the DMA LCD channel at a time.

The LCD_DESTINATION_PORT bit in the DMA LCD control register (DMA_LCD_CTRL) indicates which controller is used.

The external LCD port supports 32-bit single access transfers, while the OMAP LCD controller only supports 16-bit single accesses. See the respective LCD controller for more detailed information.

LCh-D is enabled differently depending on which LCD controller used:

- In OMAP LCD controller mode, the LCD channel is enabled when the OMAP LCD controller is enabled; that is, when bit LcdEn = 1 in the LCD control register (LcdControl). For more information, see the LCD control register description in Section 7.4.3, *LCD Channel Dedicated Registers.*
- When using an external LCD controller, the LCD channel is enabled (as are regular channels) with the ENABLE bit in the DMA LCD channel control register (DMA_LCD_CCR).

The external LCD controller can be hardware synchronized with LCh-D. This feature is controlled by the bs bit in the DMA_LCD_channel_register (DMA_LCD_CCR). See the register description for more details.

7.3.4 DMA LCD Channel Rotation

This DMA LCD channel supports four types of rotation: 0°, 90°, 180°, and 270°, by using the double indexed addressing mode in LCD channel source port. This feature allows the use of display screens that are not oriented as they were designed. For example, a 320x240 screen can be used in a 240x320 application.

7.3.5 DMA LCD Channel Autoinitialization Feature

In order to support special requirements for LCD display, the detailed autoinitialization feature for the LCD channel is described below.

- If the DMA LCD channel is connected to the OMAP external LCD controller:

  When software sets the DMA_LCD_CCR.Enable bit to 1, the logical LCD channel is enabled. It loads the LCD channel programming register set to its channel active register set. Then the logical LCD channel (LCh_D) is active and data starts transferring.

- If the DMA LCD channel is connected to the OMAP LCD controller:

  When software enables the OMAP LCD controller, bit LcdControl.LcdEn is set to 1 and the DMA LCD channel is automatically enabled at the same time. It loads LCD channel programming register set to channel active register set. The logical LCD channel becomes active and data starts transferring.

At the end of the transfer, the LCD logical channel is enabled again, when autoinit is set to 1, and it loads the LCD physical channel with:

- Channel programming set, if end_prog = 1 and repeat = 1
- Channel active set, if end_prog = 0 and repeat = 1

Table 7–16 provides a bit summary.

Table 7–16. Autoinitialization Configuration Bits Summary for LCD Channel in Noncompatible Mode

Auto-init	Repeat	END_PROG	Autoinitialization Behavior
0	x	x	No autoinitialization Waits until enable = 1 to enable the LCD logical channel and loads the physical LCD channel with its programming register set. However, the LCD logical channel is active only when the LCD controller enables it.
1	0	0	As both repeat and end_prog bits equal 0, the channel is still enabled but pending. At the end of the current transfer, *the logical LCD channel waits until repeat or end_prog = 1 to reactivate itself again*. When end_prog = 1 the programming register set is copied to the active register set: *a new context is programmed.*
1	x	1	At the end of the current transfer, the logical LCD channel immediately loads the physical LCD channel with its programming register set, when physical LCD channel is granted (end_prog = 1 allows loading the new context, disregarding the repeat bit). *The channel reinitializes itself and starts a new transfer with the new context*.
1	1	0	The channel reinitializes itself at the end of the current transfer *and starts a new transfer with the previous context* (active register set).

7.3.6 DMA_LCD_Disable/Bus Error Feature

Software can disable the LCD channel on the fly. If the LCD channel is disabled, then transfer stops immediately.

During LCD channel transfer, if an underflow occurs, the DMA LCD channel resets its enable bit and the LCD channel stops immediately.

LCD PCh underflow is possible when the destination is the OMAP LCD controller (reads from external controller are stalled until data is ready).

If underflow occurs, a signal is sent to the OMAP LCD controller and the FUF bit is set in the OMAP LCD controller status register (LCSR). An interrupt is generated when a LCSR bit is set. See Section 8.3.5, *LCD Controller Status Register.*

If one hardware request is currently being serviced and another hardware request is triggered, the DMA_LCD controller stops and signals an event drop interrupt (bus error).

7.3.7 LCD Channel Usage Restrictions

7.3.7.1 Exclusive Blocks

The hardware design does not detect any overlap between two block buffers; that is, the start and stop addresses of each buffer must represent two different physical parts into the memory. In dual-block mode, the top address of the second block must be greater (and not equal to) than the bottom address of the first block.

7.3.7.2 Both Blocks Must Belong to a Single Source

In case of dual-block mode operation, it is not possible to have one block read from one source and one block read from a second source. To change from a source to another, the LCD_LCD_EN (enable transfer) signal must not be asserted and all pending LCD interrupts must be processed.

7.3.7.3 LCD Registers Can Be Configured During a Transfer

There is a shadow register set for the LCD channel, which allows the software to change the LCD channel context on the fly after the LCD channel enabled. This is not supported in the OMAP 3.0/3.1 system DMA.

Example 7–3 shows a transfer from a video block, located in SDRAM, to the OMAP LCD controller.

Example 7–3. LCD Transfer: EMIFF (SDRAM) → LCD, One Block

The size of the LCD display is 6 × 16 pixels with 16 bits per pixels. So the length of the video frame is 6 × 16x2 (in bytes) + 32 bytes for the palette = 224 bytes. If the video block starts at address 0x0B0000, the bottom address of the video block is 0x0B00DE.

Step 1: Registers are set as:

DMA_LCD_CTRL

block_mode = 0 (one block)
block_it_ie = 1
bus_error_it_ie = 1
lcd_source_Port = 0 (SDRAM)

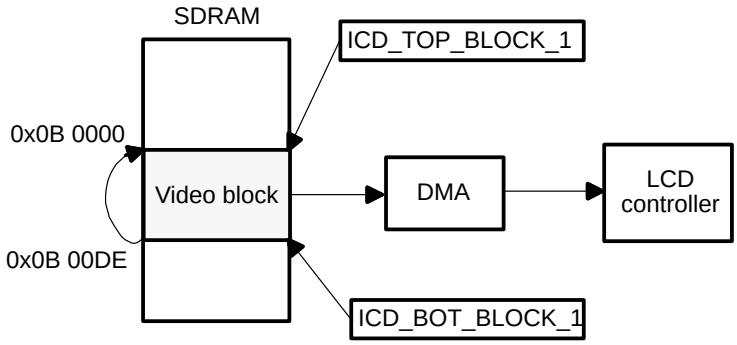
DMA_LCD_TOP_B1_U = 0x000B
DMA_LCD_TOP_B1_L = 0x0000
DMA_LCD_BOT_B1_U = 0x000B
DMA_LCD_BOT_B1_L = 0x00DE

DMA_LCD_TOP_B2_U = irrelevant
DMA_LCD_TOP_B2_L = irrelevant
DMA_LCD_BOT_B2_U = irrelevant
DMA_LCD_BOT_B2_L = irrelevant

Step 2: The transfer starts when the enable (hardware) signal from the OMAP LCD controller is asserted high.

The transfer runs and an interrupt is generated at the end of the block.

Figure 7–10. LCD One-Block Mode Transfer Scheme



Step 3: When an interrupt occurs, read the DMA_LCD_CTRL register to know the source of the interrupt.

If DMA_LCD_CTRL[3] = 1 (that is to say block_1_it_cond = 1), end-of-block 1 interrupt is detected.

If end-of-block is reached, the DMA restarts at the top of the block.

Step 4: Reset DMA_LCD_CTRL [3] and wait for another interrupt.

Example 7–4.LCD Transfer: OCP_T1 → LCD, Two Blocks

Example 7–4 shows a transfer from two video blocks located in memory connected to the OCP_T1 port to the LCD controller.

The size for the LCD display is 6 × 16 pixels with 16 bits per pixels. So the length of one video block is 6 × 16 × 2 (in bytes) + 32 bytes for the palette = 224 bytes. If video block 1 starts at address 0x0B0000, the bottom address of the video block is 0x0B00DE. If video block 2 starts at address 0x0C0000, the bottom address of the video block is 0x0C00DE.

Step 1: Registers are set as:

DMA_LCD_CTRL

block_mode = 1 (two blocks)
block_it_ie = 1
bus_error_it_ie = 1
lcd_source_Port = 1 (IMIF)

DMA_LCD_TOP_B1_U = 0x000B
DMA_LCD_TOP_B1_L = 0x0000
DMA_LCD_BOT_B1_U = 0x000B
DMA_LCD_BOT_B1_L = 0x00DE

DMA_LCD_TOP_B2_U = 0x000C
DMA_LCD_TOP_B2_L = 0x00000
DMA_LCD_BOT_B2_U = 0x000C
DMA_LCD_BOT_B2_L = 0x00DE

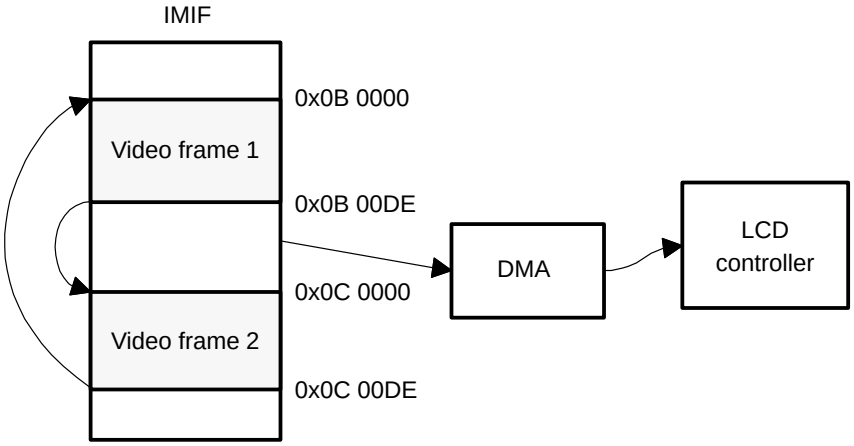
Step 2: The transfer starts when the enable (hardware) signal from the OMAP LCD controller is asserted high.

The transfer runs, and the interrupts are generated at the end of blocks 1 and 2.

DMA_LCD_CTRL [3] = block_1_it_cond = 1. This bit is a status bit that detects the interrupt.

When end-of-block 1 is reached, the DMA restarts at the top address of block 2 and DMA_LCD_CTRL [3] is reset to be able to detect the next interrupt.

Figure 7–11. LCD Dual-Block Mode Transfer Scheme



DMA_LCD_CTRL [4] = block_2_it_cond = 1: End-of-block 2 is reached. The DMA restarts at the top address of block 1, and DMA_LCD_CTRL [4] is reset to be able to detect the next interrupt.

Switching between the two frames is automatic; it requires no channel reconfiguration.

7.3.8 LCD Channel OMAP 3.0/3.1 Compatible Mode Programming

Users have the option of having new LCD channel features and maintaining compatibility with the previous LCD channel. To keep compatibility with the OMAP3.1 programming mode, the following points must be observed. There are no other compatibility obstacles.

7.3.8.1 Configuration Registers

DMA_LCD_CTRL is one of the LCD configuration registers. This register manages the operation of dual or single block mode, interrupt enable bits, and the source port for the next transfer. It then returns information by setting the status bits in its register. An interrupt can be sent at the end of the transfer of each block; this interrupt line is connected to the LCD interrupt line of the DMA.

7.3.8.2 Addressing Mode

In OMAP 3.1-compatible mode, the LCD channel only supports the *postincrement* addressing mode on the source side.

Address is postincremented by element size (ES_B1 or ES_B2 depending on which block is active if in dual-block mode)

$A(n+1) = A(n) + ES\_B1/2$ if $TB1/2 \leq A(n+1) \leq BB1/2$

where:

A(n) is the byte address of element n within the transfer.

ES_B1/2 is block1/2 element size

TB1/2 is top address for block1/2 and BB1/2 is bottom address for block1/2

Refer to postincrement details in section 7.2.10.3.

Hardware rotation is not supported in compatible mode due to restrictions on addressing modes.

7.3.8.3 DMA LCD Channel Sharing Feature

A second external LCD controller is not supported in compatible mode.

7.3.8.4 Disabling Feature

The LCD channel cannot be disabled by software. However, it stops transferring as soon as the OMAP LCD controller disables the LCD channel. Then it restarts from the beginning again, after the LCD controller enables the LCD channel.

7.3.8.5 LCD Channel Restriction

In OMAP 3.1-compatible mode, it is not possible to change any bit of any LCD channel register until a transfer has been fully completed, because no shadow registers exist in the LCD channel as in regular channels. To update registers, the LCD controller must not be enabled and all pending LCD interrupts must be processed.

The LCD channel source can only have 32-bit access from L3_OCP_T1 (IMIF) or EMIFF (SDRAM).

Selecting the LCD_SOURCE_PORT bit-field (in the DMA_LCD_CTRL register) to binary 10 or 11 causes unpredictable effects.

7.3.8.6 DMA LCD Channel Autoinitialization Feature

At the end of transfer, the LCD logical channel is enabled again if LCD_LCD_EN is set to 1 in the OMAP LCD controller. Table 7–17 summarizes the autoinitialization behavior in compatible mode.

Table 7–17. Autoinitialization Configuration Bits Summary for LCD Channel in Compatible Mode

Auto-init	Repeat	end_prog	Autoinitialization Behavior
Hardwired 1	Hardwired 1	Hardwired 1	At the end of the current transfer, the logical LCD channel immediately loads the physical LCD channel with its programming register set, when physical LCD channel is granted (end_prog = 1 allows loading the new context, disregarding the repeat bit). *The channel reinitializes itself and starts a new transfer with the new context.* However, the LCD logical channel is active only when the LCD controller enables it.
0 (software)	X (software)	X (software)	Not supported
1 (software)	X (software)	X (software)	Not supported

7.3.8.7 DMA Configuration Registers I/O Space

Each channel has its own address space. The configuration address is split into several fields. Each field is decoded to generate a channel select and a register select within the channel.

More configuration registers were added to the LCD channel in OMAP730. This results in two different LCD channel register mappings which are dependent on the mode selected, which is controlled by bit OMAP_3_1_MAPPING DISABLE in register DMA_GSCR.

Table 7–18 shows the register mapping in the two different modes (OMAP 3.0/3.1-compatible or not).

Table 7–18. LCD Channel Register Mapping for OMAP730 and OMAP 3.0/3.1 Compatible Modes

Offset	Register Mapping in OMAP730 Compatible Mode	Register Mapping in OMAP 3.0/3.1 Compatible Mode
E3C0	DMA_LCD_CSDP	DMA_LCD_CTRL
E3C2	DMA_LCD_CCR	DMA_LCD_TOP_B1_L
E3C4	DMA_LCD_CTRL	DMA_LCD_TOP_B1_U
E3C6	Reserved	DMA_LCD_BOT_B1_L
E3C8	DMA_LCD_TOP_B1_L	DMA_LCD_BOT_B1_U
E3CA	DMA_LCD_TOP_B1_U	DMA_LCD_TOP_B2_L
E3CC	DMA_LCD_BOT_B1_L	DMA_LCD_TOP_B2_U
E3CE	DMA_LCD_BOT_B1_U	DMA_LCD_BOT_B2_L
E3D0	DMA_LCD_TOP_B2_L	DMA_LCD_BOT_B2_U
E3D2	DMA_LCD_TOP_B2_U	N/A
E3D4	DMA_LCD_BOT_B2_L	N/A
E3D6	DMA_LCD_BOT_B2_U	N/A
E3D8	DMA_LCD_SRC_EI_B1	N/A
E3DA	DMA_LCD_SRC_FI_B1_L	N/A
E3DC	DMA_LCD_SRC_EI_B2	N/A
E3DE	DMA_LCD_SRC_FI_B2_L	N/A
E3E0	DMA_LCD_SRC_EN_B1	N/A
E3E2	DMA_LCD_SRC_EN_B2	N/A
E3E4	DMA_LCD_SRC_FN_B1	N/A
E3E6	DMA_LCD_SRC_FN_B2	N/A
E3EA	DMA_LCH_CTRL	N/A
E3F4	DMA_LCD_SRC_FI_B1_U	N/A
E3F6	DMA_LCD_SRC_FI_B2_U	N/A

7.4 System DMA Registers

- ❏ All reserved bits and all reserved registers must be written as 0x0000.
- ❏ All reserved bits are read as 0.
- ❏ The configuration and status registers are part of a superset that is designed for a 16-bit port and a 16-bit channels DMA. This superset enables optimal design reuse in hardware and software. This design reuse capability requires generic register mapping. These requirements cause some registers to appear almost empty at times.

7.4.1 DMA Global Registers

Table 7–19 lists the system DMA registers. Table 7–20 through Table 7–39 describe the registers bits. Global registers must be configured before any LCh is enabled to avoid unpredictable results.

Table 7–19. System DMA Registers

Name	Description	R/W	Offset
DMA Global Control Registers			
DMA_GCR	DMA global control		DC00
DMA_GSCR	DMA software compatible		DC04
DMA_GRST	DMA software reset control		DC08
DMA Identification Registers			
DMA_HW_ID	DMA version identification		DC42
DMA_PCH2_ID	DMA physical channel 2 version identification		DC44
DMA_PCH0_ID	DMA physical channel 0 version identification		DC46
DMA_PCH1_ID	DMA physical channel 1 version identification		DC48
DMA_PCHG_ID	DMA physical channel G version identification		DC4A
DMA_PCHD_ID	DMA physical channel D version identification		DC4C
DMA Capability Registers			
DMA_CAPS_0_U	DMA capability 0 upper		DC4E
DMA_CAPS_0_L	DMA capability 0 lower		DC50
DMA_CAPS_1_U	DMA capability 1 upper		DC52
DMA_CAPS_1_L	DMA capability 1 lower		DC54
DMA_CAPS_2	DMA capability 2		DC56
DMA_CAPS_3	DMA capability 3		DC58
DMA_CAPS_4	DMA capability 4		DC5A
DMA Status Registers			
DMA_PCH2_SR	DMA physical channel 2 status		DC60
DMA_PCH0_SR	DMA physical channel 0 status		DC80
DMA_PCH1_SR	DMA physical channel 1 status		DC82
DMA_PCHD_SR	DMA physical channel D status		DCC0

Table 7–20. DMA Global Control Register (DMA_GCR)

Bit	Name	Function	R/W	Reset Value
31:5	RESERVED	Reading this bit gives an undefined value; writing to it has no effect..	R/W	0x0
4	ROUND_ROBIN_DISABLE	GDMA physical channel scheduler ROUND_ROBIN scheduling disable 0: GDMA physical channel uses round-robin scheduling scheme. 1: GDMA physical channel uses fixed weighted scheduling scheme (from LCH0 to LCH1) to schedule next available logical channel.	R/W	0x0
3	CLK_AUTOGATING_ON	Allows the DMA to cut off its clocks according to its activity. 0: Clocks are always on. 1: Autogating enable.	R/W	0x1
2	FREE	DMA reaction to suspend the signal 0: DMA suspends all the current transfers when it receives the suspend signal from the processor 1: DMA continues running when it receives the suspend signal from the processor.	R/W	0x0
1:0	RESERVED	Reading this bit gives an undefined value; writing to it has no effect.	R/W	0x0

Table 7–21. DMA Software Compatible Register (DMA_GSCR)

Bit	Name	Function	R/W	Reset Value
15:4	RESERVED	Undefined	R	0x0
3	OMAP_3_1_MAPPING_DISABLE	OMAP3.1 mapping disable control 0: GDMA is compatible with OMAP3.1 GDMA, which maps interrupt lines and the LCD channel programming address. Therefore, GDMA logical channel can be configured as OMAP3.1 compatible channel or OMAP3.2 compatible channel by configuring GDMA_LCH_CCR[10]. 1: GDMA is compatible with OMAP3.2 GDMA, which maps the OMAP3.2 interrupt lines and the LCD channel programming address. Therefore, GDMA logical channel can be configured as OMAP3.1 compatible channel or OMAP3.2 compatible channel by configuring GDMA_LCH_CCR[10].	R/W	0x0
2:0	RESERVED	Undefined	R	0x0

Table 7–22. DMA Software Reset Control Register (DMA_GRST)

Bit	Name	Function	R/W	Reset Value
15:1	RESERVED	Undefined	R	0x0
0	SW_RESET	GDMA software reset control bit 1: Resets the entire GDMA when software writes a 1 to set this bit. Is automatically reset to 0 by hardware. Therefore, SW read to this register is 0.	R/W	0x0

Table 7–23. DMA Hardware Version ID Register (DMA_HW_ID)

Bit	Name	Function	R/W	Reset Value
15:0	GDMA_VERSION_ID_NUMBER	DMA version ID number	R	0x0001

This register contains the DMA subchip ID number for each spin of the DMA, regardless of what feature changes are made.

Table 7–24. Physical Channel 2 ID Register (DMA_PCH2_ID)

Bit	Name	Function	R/W	Reset Value
15:0	GDMA_PCH2_VERSION_ID_NUMBER	DMA PCH2 version ID number	R	0x0001

Table 7–25. Physical Channel 0 ID Register (DMA_PCH0_ID)

Bit	Name	Function	R/W	Reset Value
15:0	GDMA_PCH0_VERSION_ID_NUMBER	DMA PCH0 version ID number	R	0x0001

Table 7–26. Physical Channel 1 ID Register (DMA_PCH1_ID)

Bit	Name	Function	R/W	Reset Value
15:0	GDMA_PCH1_VERSION_ID_NUMBER	DMA PCH1 version ID number	R	0x0001

Table 7–27. Physical Channel G ID Register (DMA_PCHG_ID)

Bit	Name	Function	R/W	Reset Value
15:0	GDMA_PCHG_VERSION_ID_NUMBER	DMA PCHG version ID number	R	0x0001

The system DMA PCh-G version-ID number is not available in OMAP730.

Table 7–28. Physical Channel D ID Register (DMA_PCHD_ID)

Bit	Name	Function	R/W	Reset Value
15:0	GDMA_PCHD_VERSION_ID_NUMBER	DMA PCHD version ID number	R	0x0001

Table 7–29. DMA Capability 0 Upper Register (DMA_CAPS_0_U)

Bit	Name	Function	R/W	Reset Value
15:4	RESERVED	Undefined	R	0x0
3	CONSTANT_FILL_CAP	Constant fill capability: 1: PCH-G/PCH-M can do constant fill copy. 0: PCH-G/PCH-M can do constant fill copy.	R	0x1
2	TRANSPARENT_BLT_CAP	Transparent BLT capability: 1: PCH-G/PCH-M can do transparent BLT copy. 0: PCH-G/PCH-M can do transparent BLT copy.	R	0x1
1	OVERLAP_DETECTION_CAP	Overlap detection capability: 1: PCH-G can do overlap detection. 0: PCH-G can do overlap detection.	R	0x0
0	DIRECTIONAL_BLT_CAP	Directional blt capability: 1: PCH-G can do directional BLT copy. 0: PCH-G can do directional BLT copy.	R	0x0

Table 7–30. DMA Capability 0 Lower Register (DMA_CAPS_0_L) NIL

Bit	Name	Function	R/W	Reset Value
15:3	RESERVED	Undefined	R	0x0
2	SUB_BYTE_DESTINATION_CAP	Sub-byte destination capability: 1: PCH-G can do sub-byte adjust for expansion. 0: PCH-G can do sub-byte adjust for expansion.	R	0x0
1	RESERVED	Undefined	R	0x0
0	ORIGIN_COORDINATE_CAP	Origin coordinate capability: 1: PCH-G can do origin coordinate calculation. 0: PCH-G can do origin coordinate calculation.	R	0x0

Table 7–31. DMA Capability 1 Upper Register (DMA_CAPS_1_U)

Bit	Name	Function	R/W	Reset Value
15:0	RESERVED	Undefined	R	0x0

Table 7–32. DMA Capability 1 Lower Register (DMA_CAPS_1_L)

Bit	Name	Function	R/W	Reset Value
15:2	RESERVED	Undefined	R	0x0
1	ONE_BIT_COLOR_EXPANSION_CAP	1-bit palettized capability 1: PCH-G is able to do 1-bit color expansion. 0: PCH-G can do 1-bit color expansion.	R	0x0
0	RESERVED	Undefined	R	0x0

Table 7–33. DMA Capability 2 Register (DMA_CAPS_2)

Bit	Name	Function	R/W	Reset Value
15:9	RESERVED	Undefined	R	0x0
8	SEPARATE_SRC_DST_INDEX_CAP	Separate source/double-index capability 1: Supports separate SRC/DST index for 2-D addressing 0: Does not support separate SRC/DST index for 2-D addressing	R	0x1
7	DST_DOUBLE_INDEX_ADDRESS_CAP	Destination double-index address capability: 1: Supports double-index address mode in destination port 0: Does not support double-index address mode in destination port	R	0x1
6	DST_SINGLE_INDEX_CAP	Destination singl-index address capability 1: Supports single-index address mode in destination 0: Does not support single-index address mode in destination	R	0x1
5	DST_POST_INCREMENT_CAP	Destination post-increment address capability 1: Supports post-increment address mode in destination port 0: Does not support post-increment address mode in destination port	R	0x1
4	DST_CONSTANT_CAP	Destination constant address capability 1: Supports sconstant address mode in destination port 0: Does not support constant address mode in destination port	R	0x1
3	SOURCE_DOUBLE_INDEX_CAP	Source double-index address capability: 1: Supports double-index address mode in source port 0: Does not support double-index address mode in source port	R	0x1

Table 7–33. DMA Capability 2 Register (DMA_CAPS_2) (Continued)

Bit	Name	Function	R/W	Reset Value
2	SOURCE_SINGLE_INDEX_CAP	Source single-index address capability 1: Supports single-index address mode in source port 0: Does not support single-index address mode in source port	R	0x1
1	SOURCE_POST_INCREMENT_CAP	Source post-increment address capability 1: Supports post-increment address mode in source port 0: Does not support post-ncrement address mode in source port	R	0x1
0	SOURCE_CONSTANT_CAP	Source constant address capability 1: Supports constant address mode in source port 0: Does not support constant address mode in source port	R	0x1

Table 7–34. DMA Capability 3 Register (DMA_CAPS_3)

Bit	Name	Function	R/W	Reset Value
15:6	RESERVED	Undefined	R	0x0
5	CHANNEL_CHAINING_CAP	Channel channing capability 1: Supports logical channel chaining capability 0: Does not support logical channel chaining capability	R	0x1
4	LCH_INTERLEAVE_CAP	LCH interleave capability 1: Supports logical channel interleave capabiltiy 0: Does not support logical channel interleave capability	R	0x1
3	AUTOINIT_REPEAT_CAP	AUTOINIT repeat capability 1: Supports repeat feature in autoinit mode 0: Does not support repeat feature in autoinit mode	R	0x1
2	AUTOINIT_END_PROG_CAP	AUTOINIT_END_PROGRAM capability 1: Supports END_PROG feature in autoinit mode 0: Does not support END_PROG feature in autoinit mode	R	0x1

Table 7–34. DMA Capability 3 Register (DMA_CAPS_3) (Continued)

Bit	Name	Function	R/W	Reset Value
1	FRAME_SYNCHRONIZATION_CAP	Frame synchronization capability 1: Supports synchronization transfer on frame boundary 0: Does not support synchronization transfer on frame boundary	R	0x1
0	ELEMENT_SYNCHRONIZATION_CAP	Element synchronization capability 1: Support synchronization transfer on element boundary 0: Does not support synchronization transfer on element boundary	R	0x1

Table 7–35. DMA Capability 4 Register (DMA_CAPS_4)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Undefined	R	0x0
7	ACCESS_VIOLATION_INTERRUPT_CAP	Access violation interrupt capability 1:Supports access violation status bit generation 0:Does not support access violation status bit generation	R	0x0
6	SYNC_STATUS_CAP	Sync status capability 1:Supports synchronization transfer status bit generation 0:Does not support synchronization transfer status bit generation	R	0x1
5	BLOCK_INTERRUPT_CAP	Block interrupt capability 1: Supports block interrupt generation capability 0: Does not support block interrupt generation capability	R	0x1
4	LAST_FRAME_INTERRUPT_CAP	Last-frame interrupt capability 1:Supports last-frame interrupt generation capability 0:Does not support last-frame interrupt generation capability	R	0x1
3	FRAME_INTERRUPT_CAP	Frame interrupt capability 1: Supports frame-interrupt generation capability 0: Does not support frame-interrupt generation capability	R	0x1

Table 7–35. DMA Capability 4 Register (DMA_CAPS_4) (Continued)

Bit	Name	Function	R/W	Reset Value
2	HALF_FRAME_INTERRUPT_CAP	Half-frame interrupt capability 1: Supports Half-frame interrupt generation capability 0: Does not support half-frame interrupt generation capability	R	0x1
1	EVENT_DROP_INTERRUPT_CAP	Event drop interrupt generation capability 1: Supports event-drop interrupt generation capability 0: Does not support event-drop interrupt generation capability	R	0x1
0	TIMEOUT_INTERRUPT_CAP	Time-out interrupt capability 1: Supports time-out interrupt generation capability 0: Does not support time-out interrupt generation capability	R	0x1

Table 7–36. DMA Physical Channel 2 Status Register (DMA_PCh2_SR)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Undefined	R	0x0
7:0	LOGICAL_CHANNEL_NUMBER	The logical channel ID active in PHYSICAL_CHANNEL_2	R	0xFF

Table 7–37. DMA Physical Channel 0 Status Register (DMA_PCh0_SR)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Undefined	R	0x0
7:0	LOGICAL_CHANNEL_NUMBER	The logical channel ID active in PHYSICAL_CHANNEL_0	R	0xFF

Table 7–38. DMA Physical Channel 1 Status Register (DMA_PCh1_SR)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Undefined	R	0x0
7:0	LOGICAL_CHANNEL_NUMBER	The logical channel ID active in PHYSICAL_CHANNEL_1	R	0xFF

Table 7–39.DMA Physical Channel D Status Register (DMA_PChD_SR_0)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Undefined	R	0x0
7:0	LOGICAL_CHANNEL_NUMBER	The logical channel ID active in PHYSICAL_CHANNEL_D_0	R	0xFF

7.4.2 Logical Channel Registers

This set of registers is instantiated within each logical channel to the DMA. The registers for the display channel, LCh-D, are specific to that channel. Hence, these registers are collected in a dedicated section for the LCh-D, see section 7.4.3, *LCD Channel Dedicated Registers*.

All registers for a specific LCh must be configured before the LCh is enabled to avoid undefined results. There are some exceptions to this rule when in OMAP3.0/3.1 compatible mode. See the *OMAP1509 Multimedia Processor Technical Reference Manual* (SWPU004) for more details about active and program set registers.

Table 7–40 lists the DMA logical channel configuration registers. Table 7–42 through Table 7–82 describe the register bits.

Table 7–40. DMA Logical Channel Configuration Registers

Name	Description	R/W	Offset†
DMA_CSDP	Channel source destination parameters	R/W	D800
DMA_CCR	DMA channel control	R/W	D802
DMA_CICR	DMA channel interrupt control	R/W	D804
DMA_CSR	DMA channel status	R/W	D806
DMA_CSSA_L	DMA channel source start address lower	R/W	D808
DMA_CSSA_U	DMA channel source start address upper	R/W	D80A
DMA_CDSA_L	DMA channel destination start address lower	R/W	D80C
DMA_CDSA_U	DMA channel destination start address upper	R/W	D80E
DMA_CEN	DMA channel element number	R/W	D810
DMA_CFN	DMA channel frame number	R/W	D812
DMA_CSFI	DMA channel source frame index	R/W	D814
DMA_CSEI	DMA channel source element index	R/W	D816
DMA_CSAC	DMA channel source address counter	R	D818
DMA_CDAC	DMA channel destination address counter	R	D81A
DMA_CDEI	DMA channel destination element index	R/W	D81C

† The address for channel n: 0xFFFED8xx + (0x40 * n)

Table 7–40. DMA Logical Channel Configuration Registers (Continued)

Name	Description	R/W	Offset†
DMA_CDFI	DMA channel destination frame index	R/W	D81E
DMA_COLOR_L	DMA color parameter lower	R/W	D820
DMA_COLOR_U	DMA color parameter upper	R/W	D822
DMA_CCR2	DMA channel control 2	R/W	D824
DMA_CLNK_CTRL	DMA logical channel link control	R/W	D828
DMA_LCH_CTRL	DMA logical channel control	R/W	D82A
LCD Channel-Dedicated Registers			
DMA_LCD_CSDP	DMA LCD channel source destination parameters	R/W	E3C0
DMA_LCD_CCR	DMA LCD channel control	R/W	E3C2
DMA_LCD_CTRL	DMA LCD control	R/W	E3C4
TOP_B1_L	DMA LCD top address B1 L	R/W	E3C8
TOP_B1_U	DMA LCD top address B1 U	R/W	E3CA
BOT_B1_L	DMA LCD bottom address B1 L	R/W	E3CC
BOT_B1_U	DMA LCD bottom address B1 U	R/W	E3CE
TOP_B2_L	DMA LCD top address B2 L	R/W	E3D0
TOP_B2_U	DMA LCD top address B2 U	R/W	E3D2
BOT_B2_L	DMA LCD bottom address B2 L	R/W	E3D4
BOT_B2_U	DMA LCD bottom address B2 U	R/W	E3D6
DMA_LCD_SRC_EI_B1	DMA LCD source element index B1	R/W	E3D8
DMA_LCD_SRC_FI_B1_L	DMA LCD source frame index B1 lower	R/W	E3DA
DMA_LCD_SRC_FI_B1_U	DMA LCD source frame index B1 upper	R/W	E3F4
DMA_LCD_SRC_EI_B2	DMA LCD source element index B2	R/W	E3DC
DMA_LCD_SRC_FI_B2_L	DMA LCD source frame index B2 lower	R/W	E3DE
DMA_LCD_SRC_FI_B2_U	DMA LCD source frame index B2 upper	R/W	E3F6
DMA_LCD_SRC_EN_B1	DMA LCD source element number B1	R/W	E3E0
DMA_LCD_SRC_EN_B2	DMA LCD source element number B2	R/W	E3E2
DMA_LCD_SRC_FN_B1	DMA LCD source frame number B1	R/W	E3E4
DMA_LCD_SRC_FN_B2	DMA LCD source frame number B2	R/W	E3E6
DMA_LCH_CTRL	DMA logical channel control	R/W	E3EA

† The address for channel n: 0xFFFED8xx + (0x40 * n)

Table 7–41. DMA Logical Channel Configuration Register Set Summary Table

Registers (Number of Bits)	Offset (Address) n is the LCh number n = 0x0…0xF	Bit Position	Reset Value	Configuration Bits (Number of Bits)
DMA_CSDP (channel source destination parameters)	D800 + (n*0x40)	[1:0]	00	DATA_TYPE
		[5:2]	0000	SRC
		[6]	0	SRC_PACK
		[8:7]	00	SRC_BURST_EN
		[12:9]	0000	DST
		[13]	0	DST_PACK
		[15:14]	00	DST_BURST_EN
DMA_CCR (channel control register)	D802 + (n*0x40)	[4:0]	00000	SYNC
		[5]	0	FS
		[6]	0	PRIO
		[7]	0	ENABLE
		[8]	0	AUTO_INIT
		[9]	0	REPEAT
		[10]	0	OMAP3_1_COMPATIBLE_DISABLE
		[11]	0	END_PROG
		[13:12]	00	SRC_AMODE
		[15:14]	00	DST_AMODE
DMA_CSSA_U (channel source start address, upper bits)	D80A + (n*0x40)	[15:0]	ND	Source start address, upper word
DMA_CDSA_L (channel destination start address, lower bits)	D80C + (n*0x40)	[15:0]	ND	Destination start address, lower word
DMA_CDSA_U (channel destination start address, upper bits)	D80E + (n*0x40)	[15:0]	ND	Destination start address, upper word
DMA_CEN (channel element number)	D810 + (n*0x40)	[15:0]	ND	Element number
DMA_CFN (channel frame number)	D812 + (n*0x40)	[15:0]	ND	Frame number
DMA_CSFI (channel source frame index)	D814 + (n*0x40)	[15:0]	ND	Source frame index
DMA_CSEI (channel source element index)	D816 + (n*0x40)	[15:0]	ND	Source element index
DMA_CSAC (channel source address counter)	D818 + (n*0x40)	[15:0]	ND	Source address LSW

Table 7–41. DMA Logical Channel Configuration Register Set Summary Table (Continued)

Registers (Number of Bits)	Offset (Address) n is the LCh number n = 0x0…0xF	Bit Position	Reset Value	Configuration Bits (Number of Bits)
DMA_CICR (channel interrupt control register)	D804 + (n*0x40)	[0]	1	TOUT_IE
		[1]	1	DROP_IE
		[2]	0	HALF_IE
		[3]	0	FRAME_IE
		[4]	0	LAST_IE
		[5]	0	BLOCK_IE
		[15:6]	ND	Reserved
DMA_CSR (channel status register)	D806 + (n*0x40)	[0]	0	TOUT
		[1]	0	DROP
		[2]	0	HALF
		[3]	0	FRAME
		[4]	0	LAST
		[5]	0	BLOCK
		[6]	0	SYNC
		[15:7]	ND	Reserved
DMA_CSSA_L (channel source start address, lower bits)	D808 + (n*0x40)	[15:0]	ND	Source start address, lower word
DMA_CDAC (channel destination address counter)	D81A + (n*0x40)	[15:0]	ND	Destination address LSW
DMA_CDEI (channel destination element index)	D81C + (n*0x40)	[15:0]	ND	Destination element index
DMA_CDFI (channel destination frame index)	D81E + (n*0x40)	[15:0]	ND	Destination frame index
DMA_COLOR_L	D820 + (n*0x40)	[15:0]	ND	Color lower word [15:0]
DMA_COLOR_U	D822 + (n*0x40)	[15:0]	ND	Color upper word [31:16]
DMA_CCR2	D824 + (n*0x40)	[0]	ND	CONSTANT_FILL_ENABLE
		[1]	ND	TRANSPARENT_COPY_ENABLE
		[2]	ND	BS
		[15:3]	ND	Reserved

Table 7–41. DMA Logical Channel Configuration Register Set Summary Table (Continued)

Registers (Number of Bits)	Offset (Address) n is the LCh number n = 0x0…0xF	Bit Position	Reset Value	Configuration Bits (Number of Bits)
DMA_CLNK_CTRL	D828 + (n*0x40)	[4:0]	00000	NXTLCH_ID
		[13:5]	ND	Reserved
		[14]	0	STOP_LNK
		[15]	0	ENABLE_LNK
DMA_LCH_CTRL	D82A + (n*0x40)	[3:0]	0000	LCH Type
		[14:4]	ND	Reserved
		[15]	0	LCH_INTERLEAVE_ DISABLE

Table 7–42. Channel Source Destination Parameters Register (DMA_CSDP)

Bit	Name	Function	R/W	Reset Value
15:14	DST_BURST_EN	Destination burst enable	R/W	0x00
13	DST_PACK	Destination packing	R/W	0x0
12:9	DST	Destination port	R/W	0x0000
8:7	SRC_BURST_EN	Source burst enable 00: Single access (no burst) 01: Single access (no burst) 10: Burst 4 11: Burst 8	R/W	0x00
6	SRC_PACK	Source packing 0: Source port never makes packed accesses. 1: Source port makes packed accesses.	R/W	0x0
5:2	SRC	Source port 0000: SDRAM 0001: EMIF 0010: IMIF 0011: TIPB 0100: LOCAL 0101: API Others: Illegal	R/W	0x0000
1:0	DATA_TYPE	Type of data moved into channel 00: s8, 8 bits scalar 01: s16, 16 bits scalar 10: s32, 32 bits scalar 11: Illegal value	R/W	0x00

Table 7–43. DMA Channel Control Register (DMA_CCR)

Bit	Name	Function	R/W	Reset Value
15:14	DST_AMODE	Destination address mode 00: Constant address 01: Post incremented address 01: Single index (element index) 11: Double index (element index and frame index 0)	R/W	0x00
13:12	SRC_AMODE	Source address mode 00: Constant address 01: Post incremented address 10: Single index (element index) 11: Double index (element index and frame index 0)	R/W	0x00
11	END_PROG	End of programmation. Allows the channel to reinitialize itself if AUTO_INIT is enabled.	R/W	0x0
10	FIFO_FLUSH	FIFO flush 0: Nothing happens. 1: FIFO is flushed.	R/W	0x0
9	REPEAT	Repetitive operation 0: Once current transfer is complete, channel automatically reinitializes itself and starts a new transfer, disregarding END_PROG. 1: Once current transfer is complete, channel automatically reinitializes itself and starts a new transfer only if END_PROG =1	R/W	0x0
8	AUTO_INIT	AUTO_INITIALIZATION at the end of transfer 0: Channel stops at the end of current transfer. 1: Once current transfer is complete, the channel automatically reinitializes itself and starts a new transfer.	R/W	0x0
7	EN	Enable/disable transfer in DMA channel 0: Transfer stops and is reset. 1: Transfer starts.	R/W	0x0
6	PRIO	Channel priority 0: Channel has low-priority level. 1: Channel has high-priority level.	R/W	0x0
5	FS	Frame synchronization 0: An element is transferred each time a DMA request is made. 1: An entire frame is transferred each time a DMA request is made.	R/W	0x0
4	SYNC_PR	0: Synchronization is made with regard to TIPB port. 1: Synchronization is made with regard to the API_RHEA port.	R/W	0x0
3:0	SYNC	Synchronization control. Transfer synchronized on DMA_REQUEST{sync}, sync !=0.	R/W	0x0000

Table 7–44.DMA Channel Interrupt Control Register (DMA_CICR)

Bit	Name	Function	R/W	Reset Value
15:6	RESERVED	Reading this bit gives an undefined value; writing to it has no effect.	R/W	0x0
5	BLOCK_IE	End block interrupt enable 0: Channel does not interrupt the processor when the transfer of the block completes. 1: Channel sends an interrupt to the processor when the transfer of the block completes.	R/W	0x0
4	LAST_IE	Last frame interrupt enable 0: Channel does not interrupt the processor when the transfer of last frame starts. 1: Channel sends an interrupt to the processor when the when the transfer of the last frame starts.	R/W	0x0
3	FRAME_IE	Frame interrupt enable 0: Channel does not interrupt the processor when the transfer of current frame completes. 1: Channel sends an interrupt to the processor when the transfer of the current frame completes.	R/W	0x0
2	HALF_IE	Half frame interrupt enable 0: Channel does not interrupt the processor when the transfer of first half of the current frame completes. 1: Channel sends an interrupt to the processor when the synchronization event drop occurs.	R/W	0x0
1	DROP_IE	Synchronization event drop interrupt enable 0: Channel does not interrupt the processor when the synchronization event drop occurs 1: Channel sends an interrupt to the processor if the channel transfer is synchronized on DMA requests and two successive DMA requests drop.	R/W	0x1
0	TOUT_IE	Timeout interrupt enable 0: Channel does not interrupt the processor if a time-out error occurs. 1: Channel sends an interrupt to the processor if a time-out error occurs.	R/W	0x1

Table 7–45. DMA Channel Status Register (DMA_CSR)

Bit	Name	Function	R/W	Reset value
15:7	RESERVED	Reading this bit gives an undefined value; writing to it has no effect.	R/W	0x0
6	SYNC	Synchronization status 0: No DMA request is in service. 1: DMA request is made for this channel when it was in service.	R/W	0x0

Table 7–45. DMA Channel Status Register (DMA_CSR) (Continued)

Bit	Name	Function	R/W	Reset value
5	BLOCK	End block 0: Current transfer is not yet finished. 1: Current transfer is finished.	R/W	0x0
4	LAST	Last frame 0: Last frame did not start yet. 1: Transfer of last frame has started.	R/W	0x0
3	FRAME	Frame 0: Transfer of the current frame is still in progress. 1: A complete frame was transferred.	R/W	0x0
2	HALF	Half 0: First half of the current frame has not transferred yet. 1: First half of the current frame was transferred.	R/W	0x0
1	DROP	Drop 0: No event drop occurred during transfer. 1: Event drop occurred during transfer.	R/W	0x0
0	TOUT	Time out 0: No time-out error occurred in the channel. 1: Time-out error occurred in the channel.	R/W	0x0

Table 7–46. DMA Channel Source Start Address Lower Register (DMA_CSSA_L)

Bit	Name	Function	R/W	Reset Value
15:0	CSSA_L	Channel source start address, lower bits. The source start address output by DMA is up to 32-bit byte addressthat consists of the concatenation of DMA_CSSA_L and DMA_CSSA_U.	R/W	0x0

Table 7–47. DMA Channel Source Start Address Upper Bits Register (DMA_CSSA_U)

Bit	Name	Function	R/W	Reset Value
15:0	CSSA_U	Channel source start address, upper bits. The sources tart address consists of consists of the concatenation of DMA_CSSA_L and DMA_CSSA_U.	R/W	0x0

Table 7–48. DMA Channel Destination Start Address Lower Bits Register (DMA_CDSA_L)

Bit	Name	Function	R/W	Reset Value
15:0	CDSA_L	Lower bits for the destination start address, expressed in bytes. The destination start addresss is up to a 32-bit byte address that consists of the concatenation of DMA_CDSA_L and DMA_CDSA_U.	R/W	0x0

Table 7–49. DMA Channel Destination Start Address, Upper Bits Register (DMA_CDSA_U)

Bit	Name	Function	R/W	Reset Value
15:0	CDSA_U	Upper bits for destination start address.The destination start addresss is up to a 32-bit byte address that consists of the concatenation of DMA_CDSA_L and DMA_CDAS_U.	R/W	0x0

Table 7–50. DMA Channel Element Number Register (DMA_CEN)

Bit	Name	Function	R/W	Reset Value
15:0	CEN	Channel element number. Number of elements within a frame. The maximum frame number is 65535.	R/W	0x0

Table 7–51. DMA Channel Frame Number Register (DMA_CFN)

Bit	Name	Function	R/W	Reset Value
15:0	CFN	Channel frame number. Number of frames within the block to transfer. The maximum frame number is 65535.	R/W	0x0

The size in bytes of the data block to transfer is data_block_in_bytes = DMA_CFN × DMA_CEN × DMA_CSDP[data_type].

Table 7–52. DMA Channel Source Frame Index Register (DMA_CSFI)

Bit	Name	Function	R/W	Reset Value
15:0	CSFI	Source frame index	R/W	Undefined

DMA_CSFI contains the channel source frame index, expressed as a signed value in bytes, which is used to compute addresses when double-indexed addressing mode is used in the DMA source port.

Table 7–53. DMA Channel Source Element Index Register (DMA_CSEI)

Bit	Name	Function	R/W	Reset Value
15:0	CSEI	Source element index	R/W	Undefined

DMA_CSEI contains the channel source element index, expressed as a signed value in bytes, which is used to compute the addresses when single-index addressing mode is used.

Table 7–54. DMA Channel Destination Address Counter Register (DMA_CDAC)

Bit	Name	Function	R/W	Reset Value
15:0	CDAC	Channel destination element/frame address 16 LSB	R	Undefined

DMA_CDAC monitors the progress of a DMA transfer on channel destination port:

- It is a snapshot of the destination address generated by the channel destination address counter, which is scheduled in the channel destination port.
- It is incremented on each access made on the channel destination port (S8, S16 or S32).

Table 7–55. DMA Channel Source Address Counter Register (DMA_CSAC)

Bit	Name	Function	R/W	Reset Value
15:0	CSAC	Channel source element/frame address 16 LSB	R	Undefined

DMA_CSAC monitors the progress of a DMA transfer on channel source port:

- It is a snapshot of the source address generated by the channel source address counter, which is scheduled in the channel source port.
- It is incremented on each access made on the channel source port (S8, S16 or S32).

Table 7–56. DMA Channel Destination Element Index Register (DMA_CDEI)

Bit	Name	Function	R/W	Reset Value
15:0	CDEI	Channel destination element index	R/W	Undefined

DMA_CDEI contains the channel destination element index, expressed as a signed value in bytes, which is used to compute the addresses, when single/double-indexed addressing mode is used.

Note:

When DMA_ CCR[10] = 1, destination_element_index = DMA_CDEI.

When DMA_ CCR[10] = 0, destination_element_index = DMA_CSEI.

Table 7–57. DMA Channel Destination Frame Index Register (DMA_CDFI)

Bit	Name	Function	R/W	Reset Value
15:0	CDFI	Channel destination frame index	R/W	Undefined

DMA_CDFI contains the channel destination frame index, expressed as a signed value in bytes, which is used to compute the addresses when double indexed addressing mode is used.

Note:

When DMA_CCR[10] = 1, destination_frame_index = DMA_CDFI

When DMA_CCR[10] = 0, destination_frame_index = DMA_CSFI

Table 7–58. DMA Color Parameter Lower Register (DMA_COLOR_L)

Bit	Name	Function	R/W	Reset Value
15:0	COLOR_L	Channel BLT foreground color (least significant word)	R/W	Undefined

DMA_COLOR_L provides parameter for DMA constant fill and transparent copy features. It must be configured in big endian format.

- If DMA_CCR2[Constant_Fill_Enable] = 1, then it defines the parameter for constant filling.

  If data_type = 8 bit, then

  DMA_COLOR_L[7:0]: Defines parameter for constant filling

  If data_type = 16 bit, then

  DMA_COLOR_L[15:0]: Defines parameter for constant filling

  If data_type = 32 bit, then

  DMA_COLOR_L[15:0]: Defines parameter[15:0] (LSW) for constant filling

  DMA_COLOR_U[15:0]: Defines parameter[31:16] (MSW) for constant filling

- If DMA_CCR2[Transparent_Copy_Enable] = 1, then it defines color key parameter for transparent copy.

  If data_type = 8 bit, then

  DMA_COLOR_L[7:0]: Defines color key for transparent copy

  If data_type = 16 bit, then

  DMA_COLOR_L[15:0]: Defines color key for transparent copy

  If data_type = 32 bit, then

  DMA_COLOR_L[15:0]: Defines color key[15:0] (LSW) for transparent copy

  DMA_COLOR_U[15:0]: Defines color key[31:16] (MSW) for transparent copy

Table 7–59. DMA Color Parameter Upper Register (DMA_COLOR_U)

Bit	Name	Function	R/W	Reset Value
15:0	COLOR_U	Channel BLT foreground color (most significant word)	R/W	Undefined

For more details on this register, see Table 7–58, *DMA Color Parameter Lower Register (DMA_COLOR_U).*

Table 7–60. DMA Channel Control Register_2 (DMA_CCR2)

Bit	Name	Function	R/W	Reset Value
15:3	RESERVED	Undefined	R	0x0
2	BS	Block synchronization This bit is used to program the way a GDMA_REQUEST is serviced in a synchronized transfer: 1:An entire block is transferred each time a GDMA request is made. This frame can be interleaved on the GDMA ports with other channel requests. 0:An element/frame is transferred each time a GDMA request is made. The element can be interleaved on the GDMA port with other channel requests. Note: If DMA_CCR2.bs = 1 *and* DMA_CCR.fs = 1, the results are undefined because this is not a valid mode for OMAP 3.2.	R/W	0x0
1	TRANSPARENT_COPY_ENABLE	Transparent copy is enabled. 1: Transparent copy operation is enabled. During operation, any source data type that matches the GDMA_COLOR_U/L registers is not written to the destination. 0: Transparent copy operation is disabled.	R/W	0x0
0	CONSTANT_FILL_ENABLE	Constant fill operation enable 1:Constant fill operation is enabled. During the constant fill operation, it writes destination with GDMA_COLOR_U/L, instead of data from the source. 0: Constant fill operation is disabled. During operation, any source data is written to the destination.	R/W	0x0

Table 7–61. DMA Logical Channel Link Control Register (DMA_CLNK_CTRL)

Bit	Name	Function	R/W	Reset Value
15	ENABLE_LNK	Enable link defines the logical channel is on channel linked queue: 1: The logical channel, defined by NextLCH_ID, is enabled after the current channel finishes transferring. 0: No logical channel is chained after the current logical channel.	R/W	0x0
14	STOP_LNK	Stop link disables the logical channel on the channel-linked queue: 1: The logical channel, defined by NextLCH_ID, is disabled and ENABLE_LNK is disabled. 0: No logical channel in chained is disabled.	R/W	0x0
13:4	RESERVED	Undefined	R/W	0xU
3:0	NEXTLCH_ID	NextLCH_ID is used to build the logical channel chaining queue: x: The logical_channel x is enabled after the current logical channel finishes transfer.	R/W	0x0

Table 7–62. DMA Logical Channel Control Register (DMA_LCH_CTRL)

Bit	Name	Function	R/W	Reset Value
15	LCH_INTERLEAVE_DISABLE	Logical channel interleave disable defines the synchronized logical-channel interleave mode enable: 0: Synchronized logical channel interleave mode is enabled. The logical channel transfer can be interleaved at the end of each DMA request transfer. 1: Synchronized logical channel interleave mode is disabled. The logical channel takes control of the PCH until the entire DMA data has been transferred, regardless of the DMA request. However, to avoid suspending an LCH transfer (synchronized or not), the priority field of LCH has to be set to 1.	R/W	0x0
14:4	RESERVED	Undefined	R/W	0xU
3:0	LCH_TYPE	LCH_TYPE defines the logical channel assignment relationship to the physical channel For OMAP GDMA: 0000: LCH-2D dynamically shares the two PCH-M. 0001: LCH-G dynamically shares the two PCH-M. 0010: LCH-P dynamically shares the PCM-M (new). 0100: LCH-D uses PCH-D only. 1111: LCH-PD uses PCH-P only. LCD_CHANNEL (OMAP3.1 mode) LCH_TYPE is 1111.	R/W	0x0

7.4.3 LCD Channel Dedicated Registers

Table 7–63. DMA LCD Channel Source Destination Parameters Register (DMA_LCD_CSDP)

Bit	Name	Function	R/W	Reset Value
15:14	BURST_EN_B2†	Enables bursting on the source port, which performs bursts 4 × src_width access. When bursting isdisabled, the source port performs single accesses of src_width access. 00: Single access 01: Reserved 10: Burst 4x port_width (support 4x32bit accesses) 11: Reserved If the source port of the channel has no burst access capability, this field is ignored.	R/W	0
13	PACK_EN_B2	DMA ports can have a data bus width different from data type moved by DMA channel (s8 data type can be read on a 32-bit DMA port). DMA channel can pack four consecutive s8 data reads in a single 32-bit read access to increase transfer bandwidth. 0: Source port never makes packed accesses. 1: Source port makes packed accesses.	R/W	0
12:11	DATA_TYPE_B2†	Defines data type moved in the LCD channel from source port for BLOCK_1: 00: 8 bits scalar (s8) 01: 16 bits scalar (s16) 10: 32 bits scalar (s32) 11: illegal value Start_Address must be aligned on boundary of data type moved (if data type = s32, source start address must be aligned on word32). If data_type = s8, source start address can have any value. Software must ensure that start address is aligned with channel data type.	R/W	0
10:9	RESERVED	Reserved	R/W	Undefined
8:7	BURST_EN_B1†	Enables bursting on the source port, which performs bursts 4 × src_width access. When bursting isdisabled, the source port performs single accesses of src_width access. 00: Single access 01: Reserved 10: Burst 4x port_width (support 4x32bit accesses) 11: Reserved If the source port of the channel has no burst access capability, this field is ignored.	R/W	0

Table 7–63. DMA LCD Channel Source Destination Parameters Register (DMA_LCD_CSDP) (Continued)

Bit	Name	Function	R/W	Reset Value
6	PACK_EN_B1	DMA ports can have a data bus width different from data type moved by DMA channel (s8 data type can be read on a 32-bit DMA port). DMA channel can pack four consecutive s8 data reads in a single 32-bit read access to increase transfer bandwidth. 0: Source port never makes packed accesses. 1: Source port makes packed accesses.	R/W	0
5:2	RESERVED	Reserved	R/W	Undefined
1:0	DATA_TYPE_B1†	Defines data type moved in the LCD channel from source port for BLOCK_2:00: 8 bits scalar (s8) 01: 16 bits scalar (s16) 10: 32 bits scalar (s32) 11: illegal value Start_Address must be aligned on boundary of data type moved (if data type = s32, source start address must be aligned on word32). If data_type = s8, source start address can have any value. Software must ensure that start address is aligned with channel data type.	R/W	0

† OMAP3_1_MODE tie-off value = 10

The OMAP_3.1_MODE tie-off values are the values given to the bits in OMAP3.1-compatible mode, since the DMA_LCD_CCR, DMA_LCH_CTRL, and DMA_LCD_CSDP registers do not exist in the compatible mode. These values are tied off/on in hardware.

DMA_LCD_CSDP controls the LCD channel dual/single block transferring.

Table 7–64. DMA_LCD_Channel_Control_Register (DMA_LCD_CCR)

Bit	Name	Function	R/W	Reset Value
15:14	SRC_AMODE_B2	Source addressing mode for block2 transfer chooses the addressing mode on the source port of the channel. 00: Reserved 01: Postincremented address 10: Single index (element index) 11: Double index (element and frame indexes) There is no need for a DST_AMODE_B2 because there is no write address to a destination port in the LCD channel case.	R/W	01
13:12	SRC_AMODE_B1	Source addressing mode for block1 transfer chooses the addressing mode on the source port of the channel. 00: Reserved 01: Postincremented address 10: Single index (element index) 11: Double index (element and frame indexes) There is no need for a DST_AMODE_B1 because there is no write address to a destination port in the LCD channel case.	R/W	01
11	END_PROG	End of programmin status 0: If LCD channel is in autoinitialization mode and REPEAT bit = 1, DMA continues LCD next transfer with same channel context. If REPEAT = 0, channel does not reinitialize itself and does not start new transfer. 1: If LCD channel is in autoinitialization mode, it allows the channel to reinitialize itself with new channel context (program register set is copied to active register set) after the current DMA channel transfer has been finished. END_PROG bit automatically resets itself when the new context has been loaded. END_PROG bit not considered if AUTO_INIT = 0.	R/W	0
10	OMAP3_1_COMPATIBLE_DISABLE	Sets DMA LCD channel programming model 0: LCD channel is in OMAP3.1 compatible mode 1: LCD channel is in OMAP3.2-compatible mode	R/W	

† If software must use this bit as a flag, it must do a back-to-back read to ensure that correct value of bit is latched, because the enable bit can be asynchronously reset at the end of channel transfer: a wrong value may be sent back when this risky condition exists.

Table 7–64. DMA_LCD_Channel_Control_Register (DMA_LCD_CCR) (Continued)

Bit	Name	Function	R/W	Reset Value
9	REPEAT	Repetitive operation 0: If LCD channel in autoinitialization mode, and current transfer complete, channel automatically reinitializes itself and starts new transfer only if END_PROG = 1. 1: If LCD channel in autoinitialization mode, and current transfer complete, channel automatically reinitializes itself and starts new transfer with the previous or a new context, depending on END_PROG bit (0/1). REPEAT bit not considered if AUTO_INIT = 0.	R/W	1
8	AUTO_INIT	Autoinitialization at the end of the transfer 0: LCD channel in non-autoinitialization mode. The DMA channel stops at end of current transfer. 1: LCD channel in autoinitialization mode. Once current transfer complete, channel automatically reinitializes itself and starts new transfer if REPEAT or END_PROG bit is set. Else, if neither bit = 1, wait for END_PROG to equal 1 to reload channel with a new context and let transfer restart.	R/W	1
7	ENABLE†	Enables/disables transfer in the DMA LCD channel when OMAP external LCD controller is selected. 0: transfer stops and is reset 1: transfer starts Bit automatically cleared by DMA hardware once transfer completed. If auto_init = 1 and repeat = 1, the channel continues without being disabled. Clearing of this bit by the DMA has priority over a write by the processor. If both occur simultaneously, the configuration write is discarded.	R/W	1
6	PRIO_0	Channel priority: 0: channel has low-priority 1: channel has high-priority	R/W	1
5	RESERVED		R/W	0

† If software must use this bit as a flag, it must do a back-to-back read to ensure that correct value of bit is latched, because the enable bit can be asynchronously reset at the end of channel transfer: a wrong value may be sent back when this risky condition exists.

Table 7–64. DMA_LCD_Channel_Control_Register (DMA_LCD_CCR) (Continued)

Bit	Name	Function	R/W	Reset Value
4	BS	Block synchronize 0: If the external LCD controller is enabled, LCD is software-triggered channel, and transfer can start as soon as LCh is enabled. 1: If the external LCD controller is the destination, DMA LCD channel is synchronized on blocks, and block transfer is started each time LCh is enabled and a hardware synchronization signal is received from the external LCD controller. One DMA LCD channel request triggers one block transfer in both single and dual block modes. Two DMA LCD channel requests are required to trigger two block transfers, even if it is in dual block mode. If the OMAP LCD controller is the destination, BS is ignored. If AUTO_INIT and REPEST and/or END_PROG are set, hardware request for successive block transfers is required.	R/W	
3:0	RESERVED		R/W	Undefined

† If software must use this bit as a flag, it must do a back-to-back read to ensure that correct value of bit is latched, because the enable bit can be asynchronously reset at the end of channel transfer: a wrong value may be sent back when this risky condition exists.

The OMAP_3.1_MODE tie-off values are the values given to the bits in OMAP3.1-compatible mode, since the DMA_LCD_CCR, DMA_LCH_CTRL, and DMA_LCD_CSDP registers do not exist in the compatible mode. These values are tied off/on in hardware.

Table 7–65. DMA LCD Control Register (DMA_LCD_CTRL)

Bit	Name	Function	R/W	Reset Value
15:9	RESERVED	Reserved	R/W	Undefined
8	LCD_ DESTINATION_PORT	LCD controller destination port bit indicates which LCD controller is selected for next LCD transfer. 0: OMAP controller connected to DMA LCD channel 1: External LCD controller connected to DMA LCD channel	R/W	0
7:6	LCD_SOURCE_PORT	LCD memory source port indicates which memory source is selected for next LCD transfer: 00: SDRAM 01: L3_OCP_T1 (IMIF) 10: L3_OCP_T2 (LOCAL_ACCESS_PORT) 11: Reserved	R/W	00
5	BUS_IR_IT_COND	Bus error interrupt status 0: No bus error interrupt detected 1: Bus error interrupt detected	R/W	0
4	BLOCK_2_IT_COND	Block 2 interrupt status 0: No end-of-block 2 interrupt detected 1: End-of-block 2 interrupt detected	R/W	0
3	BLOCK_1_IT_COND	Block 1 interrupt status 0: No end-of-block 1 interrupt detected 1: End-of-block 1 interrupt detected	R/W	0
2	BUS_ERROR_IT_IE	Bus error interrupt enable 0: Interrupt disabled 1: Interrupt enabled	R/W	0
1	BLOCK_IT_IE	End block interrupt enable: enables an end-of-block interrupt for either block 1 or 2 when dual block mode is selected. 0: Interrupt disabled 1: Interrupt enabled	R/W	0
0	BLOCK_MODE	Type of block mode used for LCD transfer: 0: One block buffer—only registers relative to block 1 are used. 1: Two block buffers—LCD channel reads alternatively TOP_BLOCK_1 and TOP_BLOCK_2	R/W	0

† To enable the external LCD controller, you must have the external LCD clock active (even if not ready to send an image) by setting the following:
DMA_GSCR.OMAP31_MAPPING_DISABLE = 1
DMA_LCD_CTRL.ICD_DESTINATION_PORT = 1
DMA_LCD_CCR.OMAP31_COMPATIBLE_DISABLE = 1
To enter deep idle mode, you must disable DMA_LCD_CTRL.ICD_DESTINATION_PORT = 0.

The DMA LCD control register contains nine bit-fields which control the LCD channel operation. There are two cases of interrupt: End block buffer and abort

on the bus (bus error). Bits block_it_ie and bus_error_iie (interrupt enable) enable the generation of the interrupt.

If the status bits (xxx_cond bits and the corresponding interrupt enable bits xxx_ie bits) are all set, an interrupt signal is sent from the DMA LCD channel to the CPU. The CPU reads this register to find the cause of the interrupt.

Table 7–66. DMA LCD Top Address B1 L Register (TOP_B1_L)

Bit	Name	Function	R/W	Reset Value
15:1	ADD_L	LCD top address for block buffer 1 lower bits	R/W	Undefined
0	Reserved	Always tied to 0	R	0

Table 7–67. DMA LCD Top Address B1 U Register (TOP_B1_U)

Bit	Name	Function	R/W	Reset Value
15:0	ADD_H	LCD top address for block buffer 1 upper bits	R/W	Undefined

The LCD top address B1 registers are two 16-bit registers, which contain the start address for the video RAM buffer 1. The 32-bit address is obtained by the concatenation of the two word16 as follows:

LCD_TOP_B1 = DMA_LCD_TOP_B1_U & DMA_LCD_TOP_B1_L

Note:

The LSB of the word32 is equal to zero. Address of video buffer must always be even.

Table 7–68. DMA LCD Bottom Address B1 L Register (BOT_B1_L)

Bit	Name	Function	R/W	Reset Value
15:1	ADD_L	LCD bottom address for block buffer 1 lower bits	R/W	Undefined
0	Reserved	Always tied to 0	R	0

Table 7–69. DMA LCD Bottom Address B1 U Register (BOT_B1_U)

Bit	Name	Function	R/W	Reset Value
15:0	ADD_H	LCD bottom address for block buffer 1 upper bits	R/W	Undefined

The LCD bottom address B1 registers (see Table 7–68 and Table 7–69) are 16-bit registers that contain the bottom address for the video RAM buffer 1. The 32-bit address is obtained by the concatenation of the two word16 as follows:

LCD_BOTTOM_B1 = DMA_LCD_BOT_B1_U & DMA_LCD_BOT_B1_L

Note:

The LSB of the word32 is equal to zero. Address of video buffer must always be even.

Table 7–70. DMA LCD Top Address B2 L Register (TOP_B2_L)

Bit	Name	Function	R/W	Reset Value
15:1	ADD_L	LCD top address for block buffer 2 lower bits	R/W	Undefined
0	Reserved	Always tied to 0	R	0

Table 7–71. DMA LCD Top Address B2 U Register (TOP_B2_U)

Bit	Name	Function	R/W	Reset Value
15:0	ADD_H	LCD top address for block buffer 1 upper bits	R/W	Undefined

The LCD top address B2 registers (see Table 7–70 and Table 7–71) are 16-bit registers that contain the start address for the video RAM buffer 2. The 32-bit address is obtained by the concatenation of the two word16 as follows:

LCD_TOP_B2 = DMA_LCD_TOP_B2_U & DMA_LCD_TOP_B2_L

Note:

The LSB of the word32 is equal to zero. Address of video buffer must always be even.

7.4.3.1 DMA LCD Bottom Address B2 Registers

Table 7–72. DMA LCD Bottom Address B2 L Register (BOT_B2_L)

Bit	Name	Function	R/W	Reset Value
15:1	ADD_L	LCD bottom address for block buffer 2 lower bits	R/W	Undefined
0	Reserved	Always tied to 0	R	0

Table 7–73. DMA LCD Bottom Address B2 U Register (BOT_B2_U)

Bit	Name	Function	R/W	Reset Value
15:0	ADD_H	LCD bottom address for block buffer 2 upper bits	R/W	Undefined

The LCD bottom B2 address registers (see Table 7–72 and Table 7–73) are 16-bit registers that contain the bottom address for the video RAM buffer 2. The 32-bit address is obtained by the concatenation of the two word16 as follows:

LCD_BOTTOM_B2 = DMA_LCD_BOT_B2_U & DMA_LCD_BOT_B2_L

Note:

The LSB of the word32 is equal to zero. Address of video buffer must always be even.

Table 7–74. DMA LCD Source Element Index B1 Register (DMA_LCD_SRC_EI_B1)

Bit	Name	Function	R/W	Reset Value
15:0	SEI_B1	LCD source element index for block 1	R/W	Undefined

DMA_LCD_SRC_EI_B1 contains the channel source element index for LCD video RAM buffer 1, expressed as signed value in bytes, which is used to compute the addresses when single or double-indexed addressing mode is used.

Table 7–75. DMA LCD Source Frame Index B1 Register (DMA_LCD_SRC_FI_B1_L and DMA_LCD_SRC_FI_B1_U)

Bit	Name	Function	R/W	Reset Value
15:0	SFI_B1	LCD source frame index for block 1	R/W	Undefined

The DMA LCD source frame index B1 register contains the channel source frame index for video RAM buffer 1, expressed as signed value in bytes, which is used to compute addresses when double-indexed addressing mode is used.

Table 7–76. DMA LCD Source Element Index B2 Register (DMA_LCD_SRC_EI_B2)

Bit	Name	Function	R/W	Reset Value
15:0	SEI_B2	LCD source element index for block 2	R/W	Undefined

The DMA LCD source element index B2 register contains the channel source element index for LCD video RAM buffer 2, expressed as signed value in bytes, which is used to compute the addresses when single or double-indexed addressing mode is used.

Table 7–77. DMA LCD Source Frame Index B2 Register (DMA_LCD_SRC_FI_B2_L and DMA_LCD_SRC_FI_B2_U)

Bit	Name	Function	R/W	Reset Value
15:0	SFI_B2	LCD source frame index for block 2	R/W	Undefined

The DMA LCD source frame index B2 register contains the channel source frame index for video RAM buffer 2, expressed as signed value in bytes, which is used to compute addresses when double-indexed addressing mode is used.

Table 7–78. DMA LCD Source Element Number B1 Register (DMA_LCD_SRC_EN_B1)

Bit	Name	Function	R/W	Reset Value
15:0	SEN_B1	LCD channel source element number for block 1	R/W	Undefined

The DMA LCD source element number B1 register contains the number of elements within a frame (unsigned) for the video RAM buffer 1. The maximum element number is 65535.

Table 7–79. DMA LCD Source Frame Number B1 Register (DMA_LCD_SRC_FN_B1)

Bit	Name	Function	R/W	Reset Value
15:0	SFN_B1	LCD channel source frame number for block 1	R/W	Undefined

The DMA LCD source frame number B1 register contains the number of frames within a block (unsigned) for the video RAM buffer 1. The maximum frame number is 65535.

The size in bytes of the data block to transfer is as follows:

data_block_in_bytes = DMA_LCD_SRC_FN_B1 x DMA_LCD_SRC_EN_B1 x DMA_LCD_CDSP[Data_type_b1]

Table 7–80. DMA LCD Source Element Number B2 Register (DMA_LCD_SRC_EN_B2)

Bit	Name	Function	R/W	Reset Value
15:0	SEN_B2	LCD channel source element number for block 2	R/W	Undefined

The DMA LCD source element number B2 register contains the number of elements within a frame (unsigned) for the video RAM buffer 2. The maximum element number is 65535.

Table 7–81. DMA LCD Source Frame Number B2 Register (DMA_LCD_SRC_FN_B2)

Bit	Name	Function	R/W	Reset Value
15:0	SFN_B2	LCD channel source frame number for block 2	R/W	Undefined

The DMA LCD source frame number B2 register contains the number of frames within a block (unsigned) for the video RAM buffer 2. The maximum frame number is 65535.

The size in bytes of the data block to transfer is as follows:

data_block_in_bytes = DMA_LCD_SRC_FN_B2 x DMA_LCD_SRC_EN_B2 x DMA_LCD_CSDP[Data_type_b2]

Table 7–82. DMA Logical Channel Control Register (DMA_LCH_CTRL)

Bit	Name	Function	R/W	Reset Value
15:4	RESERVED	Reserved	R/W	Undefined
3:0	LCH_TYPE	Defines logical channel assignment relationship to associated features and physical channl (OMAP3.1 mode) Only possible configuration: 0100: LCh-D PCh-D (must be configured this way to secure forward compatibility)	R/W	0000

Chapter 8

LCD Controller

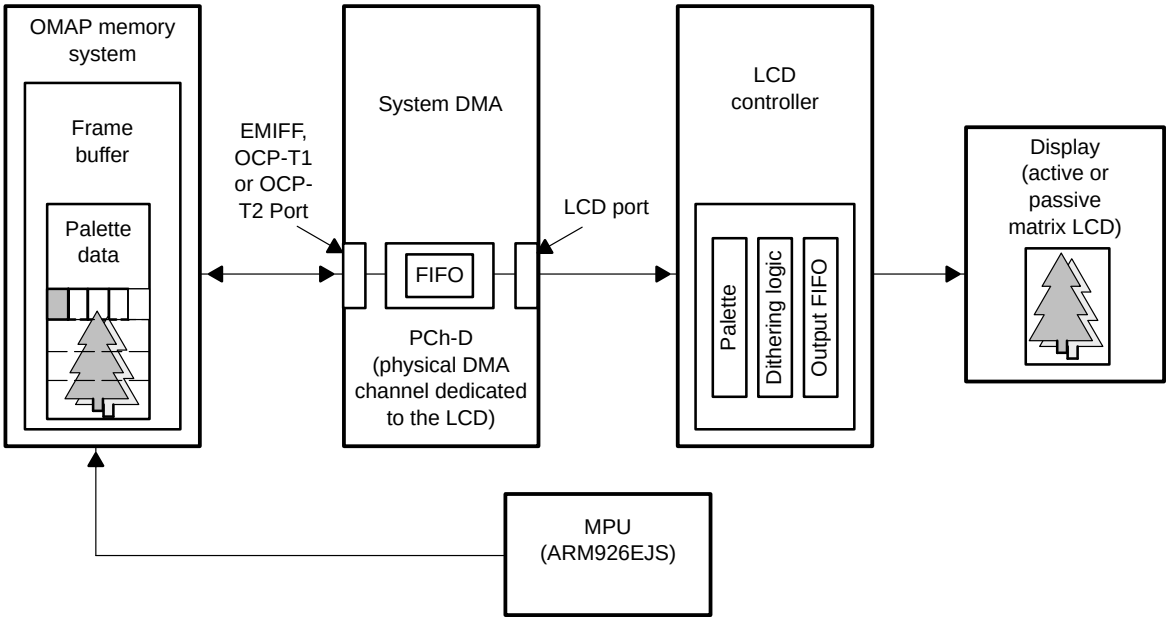
This chapter discusses the LCD controller module for the OMAP730 hardware engine.

Topic **Page**

8.1 LCD Controller Environment

The LCD controller provides the necessary control signals to interface the memory directly to the external display through a dedicated DMA channel, as seen in Figure 8–1.

Figure 8–1. Data Flow From Microprocessor to Display



The MPU stores the image to be displayed in a frame buffer. The frame buffer is a 16-bit-wide area used to supply enough encoded pixel values to fill the entire screen once.

The palette and the picture data are both in system memory. The palette loading mode (PLM) can be switched around so that only the picture data, only the palette data, or both are loaded at a given time (the palette is loaded only when it changes, then the PLM bit field returns to picture-only mode).

The working copy of the palette resides in the LCD controller itself. See Section 8.2, *LCD Controller Operation,* for descriptions of each block.

A specific system DMA channel dedicated to the LCD controller (LCh-D) is in charge of transferring data from the frame buffer to the LCD controller. Data is fetched and then transitions into a 64 $\times$ 33-bit FIFO. The 33rd bit is added for frame-synchronization purposes. However, the internal LCD controller receives 16-bit data from the DMA FIFO with each request.

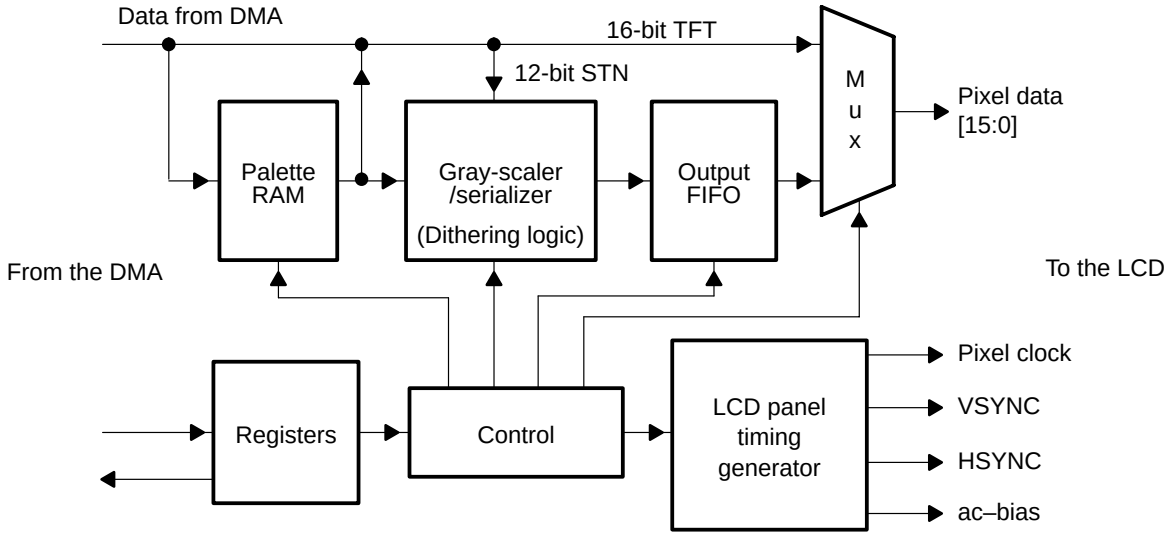
Section 8.2 describes the LCD controller operation in detail.

8.2 LCD Controller Operation

The LCD controller essentially consists of three blocks: a palette RAM, dithering logic, and an output FIFO, all associated with a control block operating through registers and a timing generator.

Figure 8–2 illustrates the LCD controller logic.

Figure 8–2. LCD Controller Operation Overview (Passive and Active Display Modes)



The LCD controller supports single-panel mode displays whose size is programmable and can be any width (line length) from 16 to 1024 pixels in 16-pixel increments (bit pixels per line, PPL, in the LCD timing 0 register). The number of lines is set by programming the lines per panel (LPP) bit in the LCD timing 1 register (see Section 8.3.3, *LCD Timing 1 Register*). The total video frame size is programmable up to 1024 ×1024.

The screen is intended to be mapped to the frame buffer as one continuous block where each line of pixels is mapped to a set of consecutive bytes or words in the frame buffer.

Two types of display technologies are supported: passive (super-twisted nematic, or STN) and active (thin film transistor, or TFT) panels (configured with the LCD TFT bit, LcdTFT, in the control register (see Section 8.3.1, *LCD Control Register*)). Both monochrome and color modes are supported (LCD monochrome bit, LcdBW, in the control register).

In passive STN mode, a total of 3375 possible colors is available, allowing 16, 256, or 3375 colors to be displayed in each frame, depending on the color depth (number of bits per pixel: BPP). Fifteen grayscale levels are available for monochrome screens. See Section 8.2.1.9, *Dithering Logic*, for information regarding number of colors displayed versus BPP and screen technology.

In active TFT mode, whatever the color depth, 4096 colors can be displayed, except in the 16-BPP mode, where up to 64K colors are supported. See Section 8.2.1.9, *Dithering Logic,* for information regarding number of colors displayed versus BPP and screen technology.

Note:

The active monochrome configuration is not considered in the remainder of this document.

8.2.1 Frame Buffer

The frame buffer is a memory area used to supply enough encoded pixel values to fill the entire screen one time. It is a part of the memory, which is connected either to EMIFF, OCP-T1, or OCP-T2 port. At the start (or lowest-order address) of the LCD controller frame buffer is a 32-byte buffer for 1-, 2-, 4- , 12-, and 16-BPP mode operation (or a 512-byte buffer for 8-BPP mode of operation), used to store the look-up palette data for each frame.

The 32-byte buffer is used to load the 16 entries of the palette for 1-, 2-, 4-, 12-, and 16-BPP encoding (or the 512-byte buffer is used to load the entire 256-entry palette for 8-BPP encoding).

Not all of the 16 entries of the palette are used in 1- and 2-BPP modes. However, all 16 palette entries are loaded regardless. The unused palette entries must be zero-filled.

The palette is not used in 12- or 16-BPP modes. The palette is only 32 bytes for these encodings to avoid wasting memory. The 32 bytes at the top of the frame buffer are zero-filled, except the first entry of the palette where a 3-bit field provides the information on the number of bits-per-pixel.

Each time a new frame is fetched from the frame buffer, the LCD controller palette is first loaded with data contained within the palette buffer. When the palette buffer is in data loading mode only, the palette does not have to be loaded each time (PLM = 10 in the control register; see Section 8.3.1.14, *Palette Loading (PLM))*.

Figure 8–3 and Figure 8–4 show the palette entry organization.

Figure 8–3. 16-Entry Palette/Buffer Format (1, 2, 4, 12, 16 BPP)

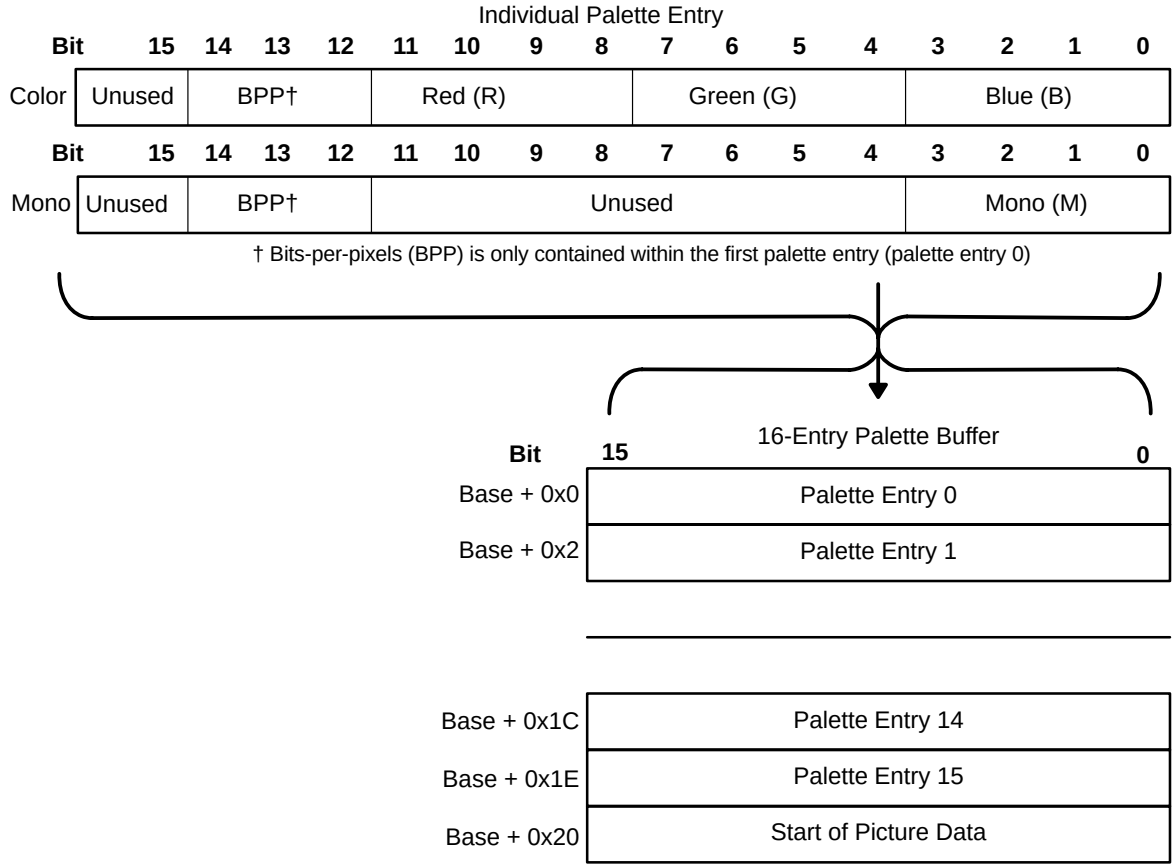
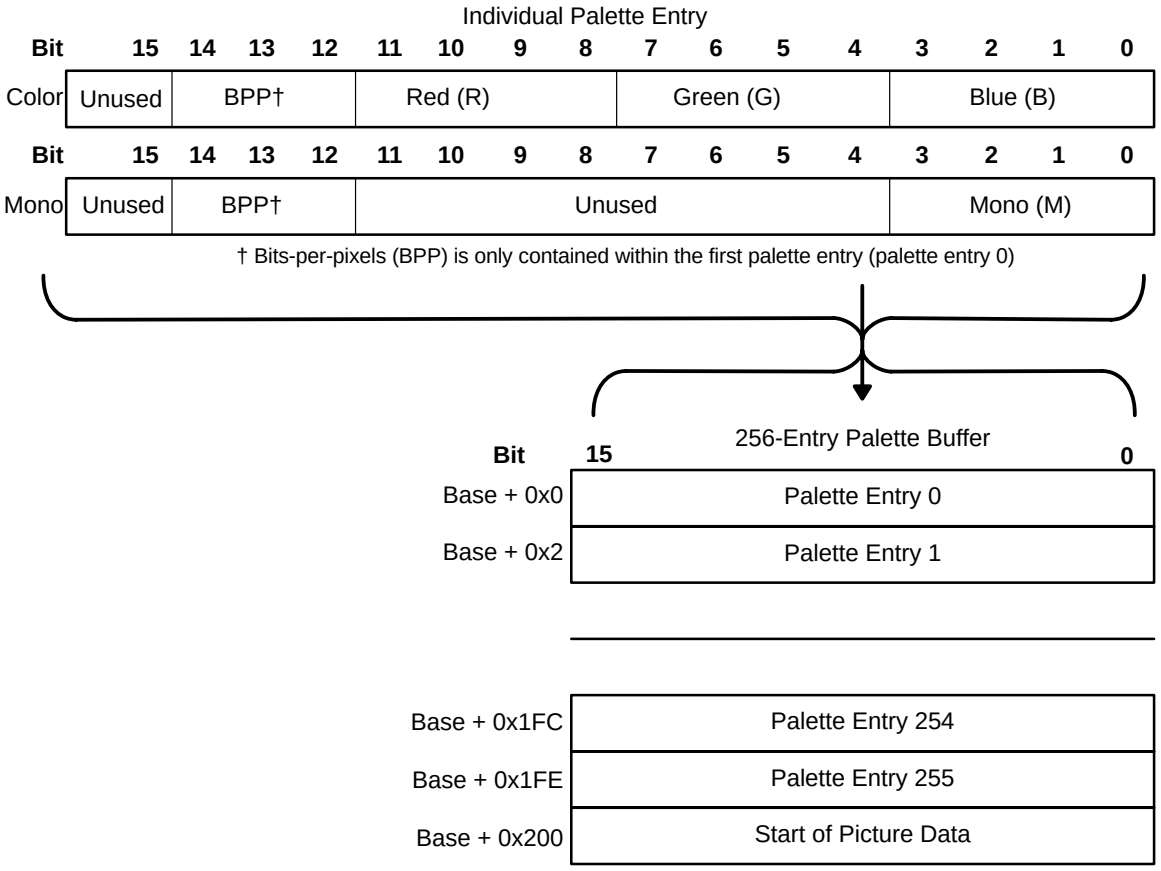


Figure 8–4. 256-Entry Palette/Buffer Format (8 BPP)



How the LCD views the ordering of frame buffer pixel entries can be set by programming the LCD big endian bit (LCDBE) in the LCD control register (LCDCONTROL). Note that the ordering of the 4-bit R, G, B, and mono-pixel data (and the BPP field, see Table 8–1) does not change between little and big endian modes.

Bits 12, 13, and 14 of the first palette entry select the number of bits-per-pixel to be used in the following frame and thus the number of palette RAM entries.

The bits-per-pixel (BPP) bit-field is decoded by the LCD to correctly unpack pixel data. It also configures the palette size to 16 or 256 entries.

Table 8–1 shows the BPP encoding in palette entry 0.

Table 8–1. Bits-Per-Pixel Encoding for Palette Entry 0 Buffer

Bit	Name	Description
14-12	BPP	Bits-per-pixel 000: 1 BPP 001: 2 BPP 010: 4 BPP 011: 8 BPP 1xx: 12 BPP in passive mode (LCDTFT=0 and 565 STN =0), 16 BPP in passive mode (LCDTFT=0 and 565 STN =1), 16 BPP in active mode (LCDTFT=1).

Notes: 1) Eight 1-bit pixels, four 2-bit pixels, and two 4-bit pixels are packed into each byte, and 12-bit pixels are right-justified on half-word32 boundaries (in the same format as palette entry).

2) For 565 STN, see the 16-BPP STN mode bit in the control register section.

The pixel data buffer directly follows the palette buffer. Palette and pixel data buffers make up the frame buffer. It contains one encoded pixel value for each pixel present on the display. Hence, the number of pixel data values depends on the size of the screen (that is, 1024 x 768 = 786,432 encoded pixel values). Again, each pixel data value can be 1, 2, 4, 8, 12, or 16 bits wide.

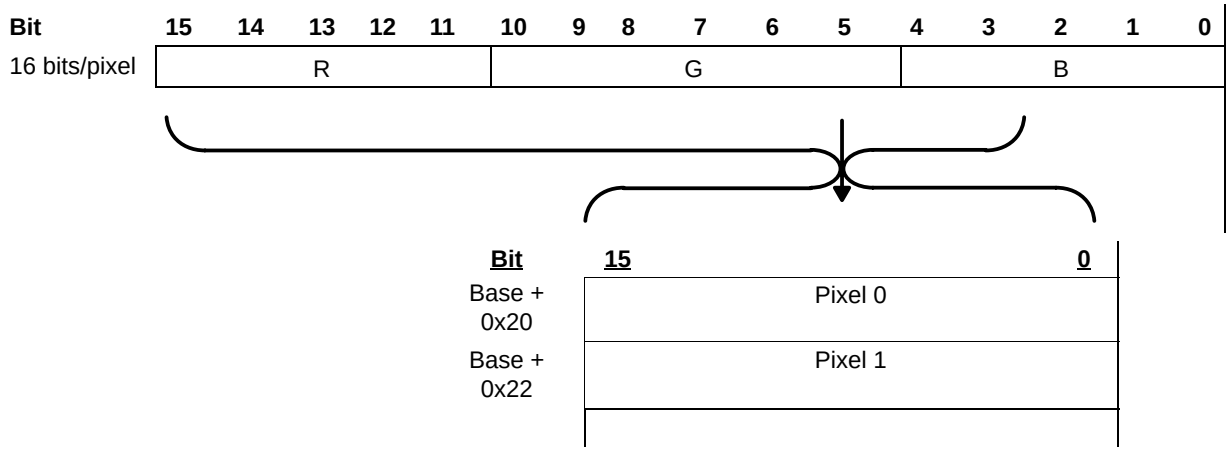
The following figures show the memory organization within the frame buffer for each pixel encoding size. You can select how the LCD views the ordering within each 16-bit frame buffer entry by programming the LCD big endian (LCDBE) or the nibble mode (NM) bit in the LCD control register.

The LCD controller is fed with a 16-bit data from the DMA. In 16- and 12-BPP modes, this entity is a scalar: it is unchanged whatever endianism convention is used.

8.2.1.1 16-BPP Mode (TFT)

Figure 8–5 shows one RGB representation in 16-BPP (TFT) mode.

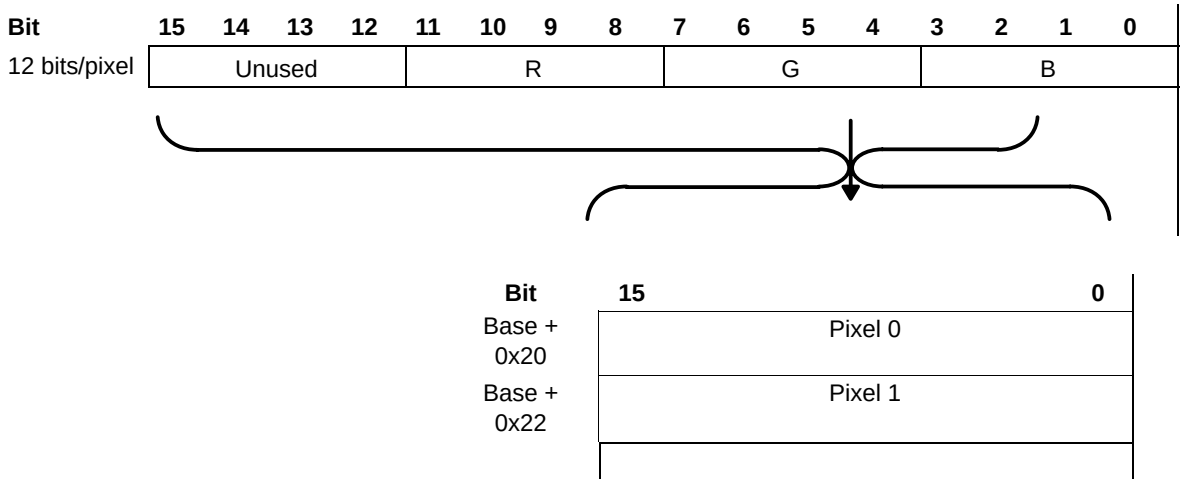
Figure 8–5. 16-BPP Data Memory Organization (TFT Mode Only)—Little or Big Endian, or Nibble



8.2.1.2 12-BPP Mode (STN)

Figure 8–6 shows one RGB representation in 12-BPP (STN) mode.

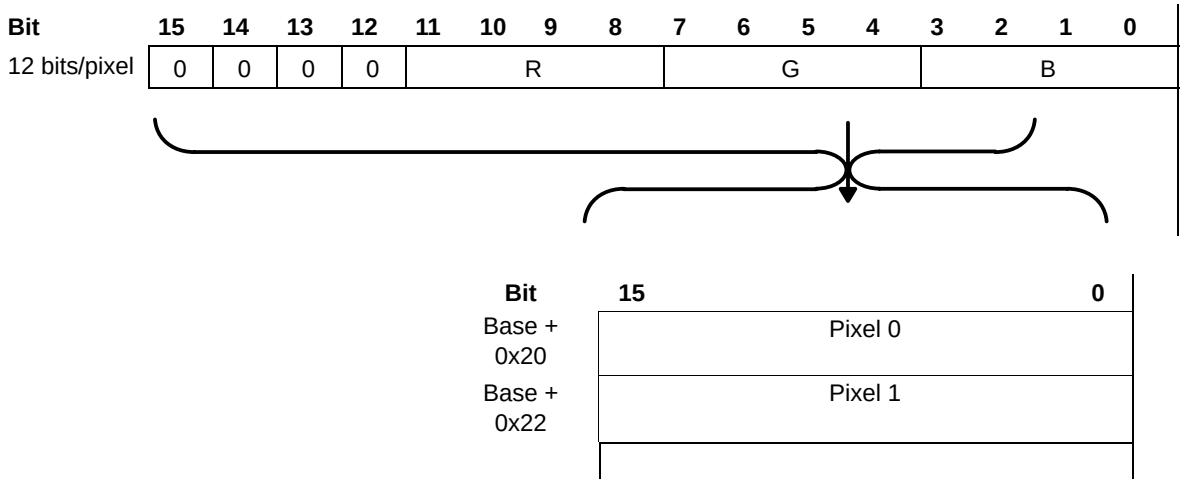
Figure 8–6. 12-BPP Data Memory Organization (STN Mode Only)—Little or Big Endian, or Nibble



8.2.1.3 12-BPP Mode (TFT)

Figure 8–7 shows one RGB representation in 12-BPP (TFT) mode.

Figure 8–7. 12-BPP Data Memory Organization (TFT Mode Only)—Little or Big Endian, or Nibble



8.2.1.4 8-BPP Mode (Little Endian or Nibble)

The conventional terminology of endianism is observed in 8-BPP mode. Figure 8–8 shows the 8-BPP organization in little endian or nibble modes. Figure 8–9 shows the 8-BPP organization in big endian mode.

Figure 8–8. 8-BPP Data Memory Organization (Little Endian or Nibble)

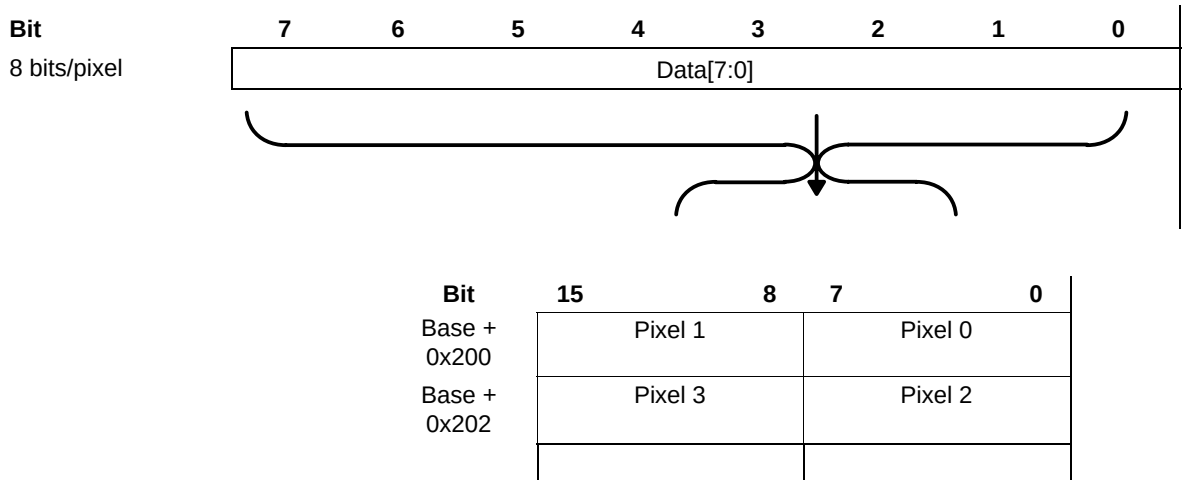
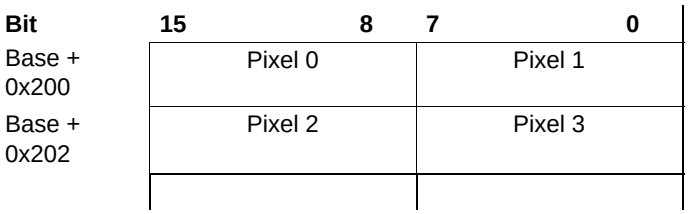


Figure 8–9. 8-BPP Data Memory Organization (Big Endian)



8.2.1.5 4-BPP Mode

From 4-BPP to 1-BPP, the endianism can be untraditional.

Figure 8–10. 4-BPP Data Memory Organization

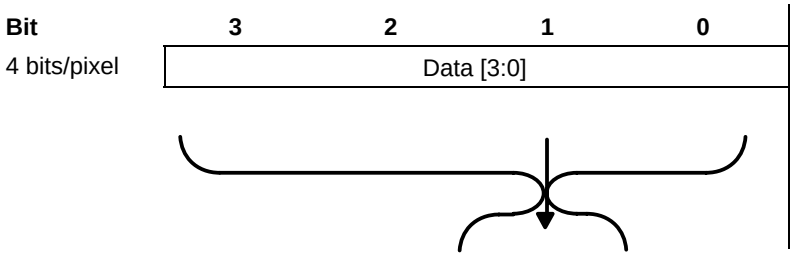


Figure 8–11. 4-BPP Data Memory Organization (Little Endian)

Bit	15 12	11 8	7 4	3 0
Base + 0x20	Pixel 3	Pixel 2	Pixel 1	Pixel 0
Base + 0x22	Pixel 7	Pixel 6	Pixel 5	Pixel 4

Figure 8–12. 4-BPP Data Memory Organization (Big Endian)

Bit	15 12	11 8	7 4	3 0
Base + 0x20	Pixel 0	Pixel 1	Pixel 2	Pixel 3
Base + 0x22	Pixel 4	Pixel 5	Pixel 6	Pixel 7

Figure 8–13. 4-BPP Data Memory Organization (Nibble Mode)

Bit	15 12	11 8	7 4	3 0
Base + 0x20	Pixel 2	Pixel 3	Pixel 0	Pixel 1
Base + 0x22	Pixel 6	Pixel 7	Pixel 4	Pixel 5

8.2.1.6 2-BPP Mode

Figure 8–14. 2-BPP Data Memory Organization (Little Endian)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 7	Pixel 7	Pixel 6		Pixel 5		Pixel 4		Pixel 3		Pixel 2		Pixel 1		Pixel 0	

Figure 8–15. 2-BPP Data Memory Organization (Big Endian)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 0		Pixel 1		Pixel 2		Pixel 3		Pixel 4		Pixel 5		Pixel 6		Pixel 7	

Figure 8–16. 2-BPP Data Memory Organization (Nibble Mode)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 4		Pixel 5		Pixel 6		Pixel 7		Pixel 0		Pixel 1		Pixel 2		Pixel 3	

8.2.1.7 1-BPP Mode

Figure 8–17. 1-BPP Data Memory Organization (Little Endian)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

Figure 8–18. 1-BPP Data Memory Organization (Big Endian)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15

Figure 8–19. 1-BPP Data Memory Organization (Nibble Mode)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P8	P9	P10	P11	P12	P13	P14	P15	P0	P1	P2	P3	P4	P5	P6	P7

The top and bottom addresses of the frame buffer (palette entries + pixel data) are programmed in the DMA controller.

The equations shown in Table 8–2 are used to calculate the total frame buffer size (in bytes) to program in the system DMA, based on varying pixel size encoding and screen sizes.

Table 8–2. Frame Buffer Size According to BPP

BPP	Frame Buffer Size
1	32 + (Lines × Columns) / 8
2	32 + (Lines × Columns) / 4
4	32 + (Lines × Columns) / 2
8	512 + (Lines × Columns)
12/16	32 + 2 * (Lines × Columns)

It is important to understand that BPP has two different meanings:

- In this section, BPP refers to how pixels are stored in memory. 1, 2, 4, 8, 12, or 16 are the different representations of the pixel within the frame buffer.
- BPP can also refer to how a panel views the pixels. This usage refers to which BPP the panels support (the actual interface, not the memory representation). The supported output panels are:
  - 1 BPP for monochrome panels, packed onto 8 (or 4) data lines
  - 3 BPP (1 bit each for red, green, and blue) for passive matrix technologies (output of dithering logic), packed onto 8 data lines
  - 12 BPP for STN (4, 4, 4) panels
  - 16 BPP for TFT (5, 6, 5) panels

8.2.1.8 Palette

The encoded pixel values stored in the frame buffer are used as pointers to index the 16-bit-wide palette. Palette entries are configured differently according to the mode used. See Figure 8–3 and Figure 8–4 for details.

The number of colors supported is given by $2^{\text{number of BPP}}$. These $2^{\text{number of BPP}}$ colors are so chosen within the palette that they are limited to $2^4 = 16$ grayscales in monochrome mode and $2^{12} = 4096$ colors in color mode, where 4 and 12 are the effective palette widths in each case (*effective* in the sense of being dedicated to the monochrome/color scales coding).

Sixteen grayscales and 4096 colors are numbers obtained after passing through the palette. A redundancy introduced at the dithering logic step reduces these numbers when displaying. As the dithering logic is bypassed in active mode, there is no redundancy and 4096 different colors are actually available. For more details, see Section 8.2.1.9, *Dithering Logic,* Table 8–3, *Color/Grayscale Intensities and Modulation Rates,* and Table 8–4, *Number of Colors/Grayscales Available on Screen.*

❑ ***Passive Matrix Technology***

The palette is bypassed in 12-BPP mode. In palette plus data or in palette-only modes (PLM = 00 or 01), the first entry is read anyway to acquire the number of BPP. All other entries or useless bits in the first entry are filled with zeros. But in data-only mode (PLM = 10), the palette is not loaded with every frame.

Note:

Henceforth, the palette is said to be *bypassed* in 12- and 16-BPP modes. The 12-bit values are directly supplied to the dithering logic when passive mode is enabled, whereas the 16-bit values are sent directly to the panel when active mode is enabled. The *bypass* term can be misleading in PLM = 00 or 01 configurations, considering that the palette must be read, at least for the first entry that contains the information of the color depth. The rest is zero-filled and not taken into consideration. In PLM = 10 (data-only mode), the palette is bypassed.

❑ ***Active Matrix Technology***

The palette is bypassed in 16 BPP allowing 2^{16} = 65536 = 64K colors to be displayed.

8.2.1.9 Dithering Logic

❑ ***Passive Matrix Technology***

Once a palette entry is selected by the encoded pixel value from the look-up palette, its content is sent to the color/grayscale space/time-based dither generator. The monochrome data, as well as each component, is encoded on 4 bits: Red, green, and blue (RGB) in color mode. See individual palette entry in Figure 8–3 and Figure 8–4. Each 4-bit value is processed by one dither block (three separate dither blocks are used in the color mode). These 4-bit values are used to select one of the 16 intensity levels. The gray/color intensity is controlled by turning individual pixels on and off at varying periodic rates. More intense grays/colors are produced by making the average time that the pixel is off longer than the average time that it is on. The dither generator also uses the intensity of adjacent pixels in its calculations to give the screen image a smooth appearance. The proprietary dither algorithm is optimized to provide a range of intensity values that match the eye's perception of color/gray gradations.

Table 8–3 summarizes the duty cycle and resultant intensity level for all 16 color/grayscale levels.

Table 8–3. Color/Grayscale Intensities and Modulation Rates

Dither Value (4-Bit Value from Palette)	Intensity (0% is White)	Modulation Rate (Ratio of On to On+Off Pixels)
0000	0.0%	0
0001	11.1%	1/9
0010	20.0%	1/5
0011	26.7%	4/15
0100	33.3%	3/9
0101	40.0%	2/5
0110	44.4%	4/9
0111	50.0%	1/2
1000	55.6%	5/9
1001	60.0%	3/5
1010	66.6%	6/9
1011	73.3%	11/15
1100	80.0%	4/5
1101	88.9%	8/9
1110	100.0%	1
1111	100.0%	1

Two of the 16 dither values (shaded in the table) are identical (most intense), which leads to redundancy in the colors.

This redundancy limits the choice to effectively 15 (instead of 16) grayscales and 3375 (instead of 4096) colors. Note that $3375 = 15^3$ which is the equivalent of 15-color scales for each component (R, G, and B).

❑ ***Active Matrix Technology***

The dithering logic is always bypassed in active displays. Hence, there is no redundancy introduced at this step, still allowing the choice of the $2^{\text{number of BPP}}$ within the whole 4096 colors.

Remember that monochrome mode is deliberately not considered in active mode.

Table 8–4 lists the number of colors/grayscales available on the screen according to both the display technology and the color depth.

Table 8–4. Number of Colors/Grayscales Available on Screen

	Passive Mode (LCDTFT = 0)		Active Mode (LCDTFT = 1)
Number of BPP	**Monochrome (LCDBW = 1)**	**Color (LCDBW = 0)**	**Color Only (LCDBW = 0)**
1	2 palette entries to select within 15 grayscales	2 palette entries to select within 3375 possible colors	2 palette entries to select within 4096 possible colors
2	4 palette entries to select within 15 grayscales	4 palette entries to select within 3375 possible colors	4 palette entries to select within 4096 possible colors
4	16 palette entries to select within 15 grayscales	16 palette entries to select within 3375 possible colors	16 palette entries to select within 4096 possible colors
8	Not relevant because it would consist in 256 palette entries to select within 15 grayscales, but exists anyway	256 palette entries to select 3375 possible colors	256 palette entries to select within 4096 possible colors
12	X	3375 possible colors	4096 possible colors
16 STN (565 STN_= 1)	X	3375 possible colors	X
16	X	X	Up to 65536 possible colors

8.2.1.10 The Output FIFO

- ***Passive Matrix Technology***

  The LCD controller contains a 2-entry by 8-bit wide output FIFO that is used to store pixel data before it is driven out to the LCD pins. Each time a modulated pixel value is output from the dither generator, it is placed into a serial shifter. The size of the shifter is controlled by programming the color/monochrome select bit (LcdBW) in the control register. The shifter can be configured to be 4 or 8 bits wide. Single-panel monochrome screens use either four or eight data lines; single-panel color screens use eight data pins. Once the correct number of pixels has been placed within the shifter, the value is transferred to the top of the output FIFO. The value is then transferred down until it reaches the last empty location within the FIFO. As values reach the bottom of the FIFO, they are driven out one-by-one onto the LCD data pins on the edge selected by the invert pixel clock (IPC) bit (see Section 8.3.4.6, *Invert Pixel Clock (IPC)*).

- ***Active Matrix Technology***

  The output FIFO is bypassed in TFT mode.

8.2.1.11 LCD Inputs

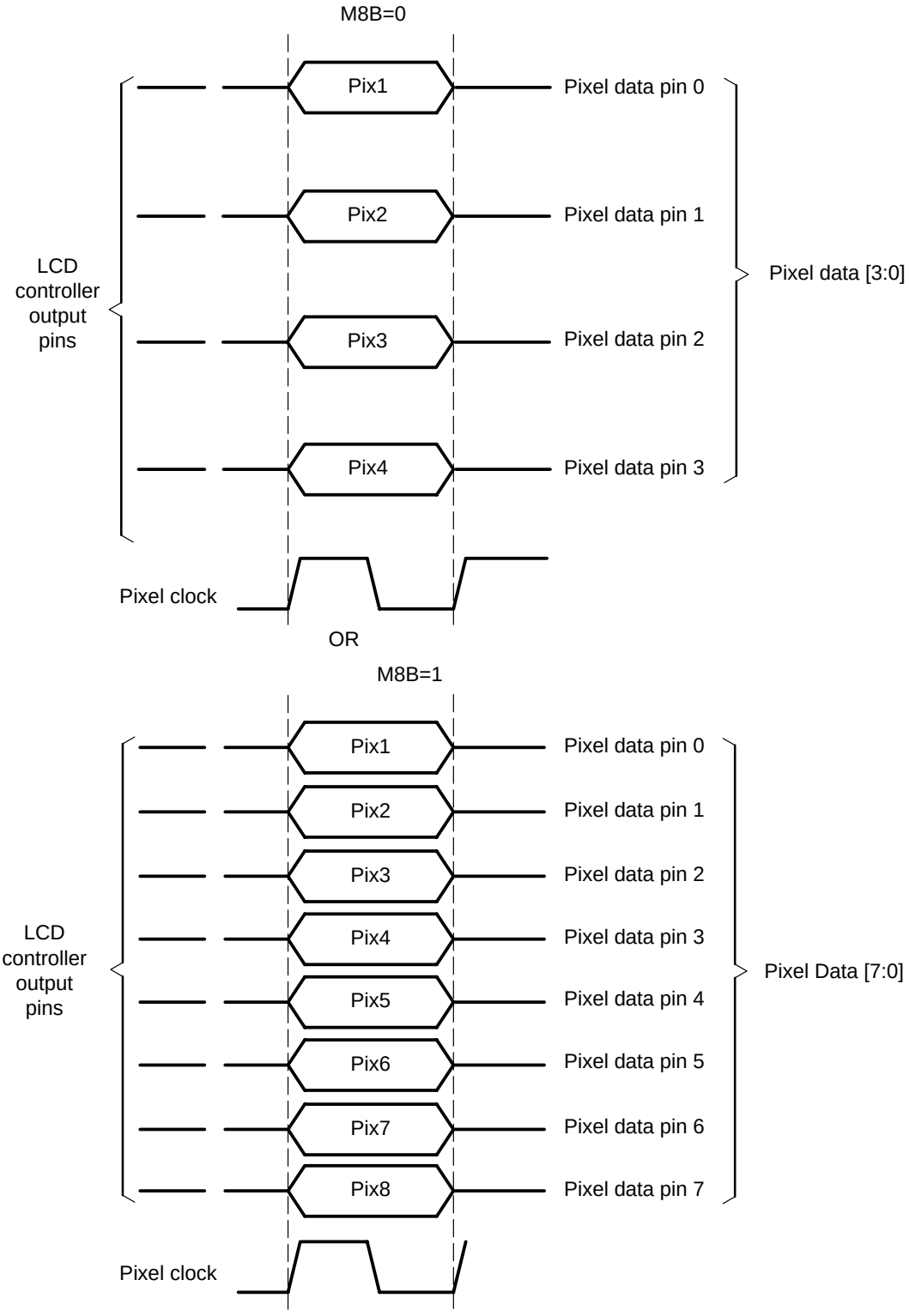
Depending on the type of panel used, the LCD controller is programmed to use either 4-, 8-, 12-, or 16-pixel data output pins. See Table 8–8, *LCD Controller Data Pin Utilization for Mono/Color Passive/Active Panels.*

- ***Passive Matrix Technology—Monochrome Mode***

  Monochrome displays use 4- or 8-bit data lines (according to the mono 8-bit mode (see Section 8.3.1.10, *Mono 8 Bit Mode (M8B)*). Each line represents one pixel (on or off), which means that, at each pixel clock, 4 or 8 pixels are sent to the screen.

  Figure 8–20 shows the passive monochrome mode (IPC = 0, see IPC bit in Section 8.3.4, *LCD Timing 2 Register*).

Figure 8–20. Passive Monochrome Mode



❑ ***Passive Matrix Technology—Color Mode***

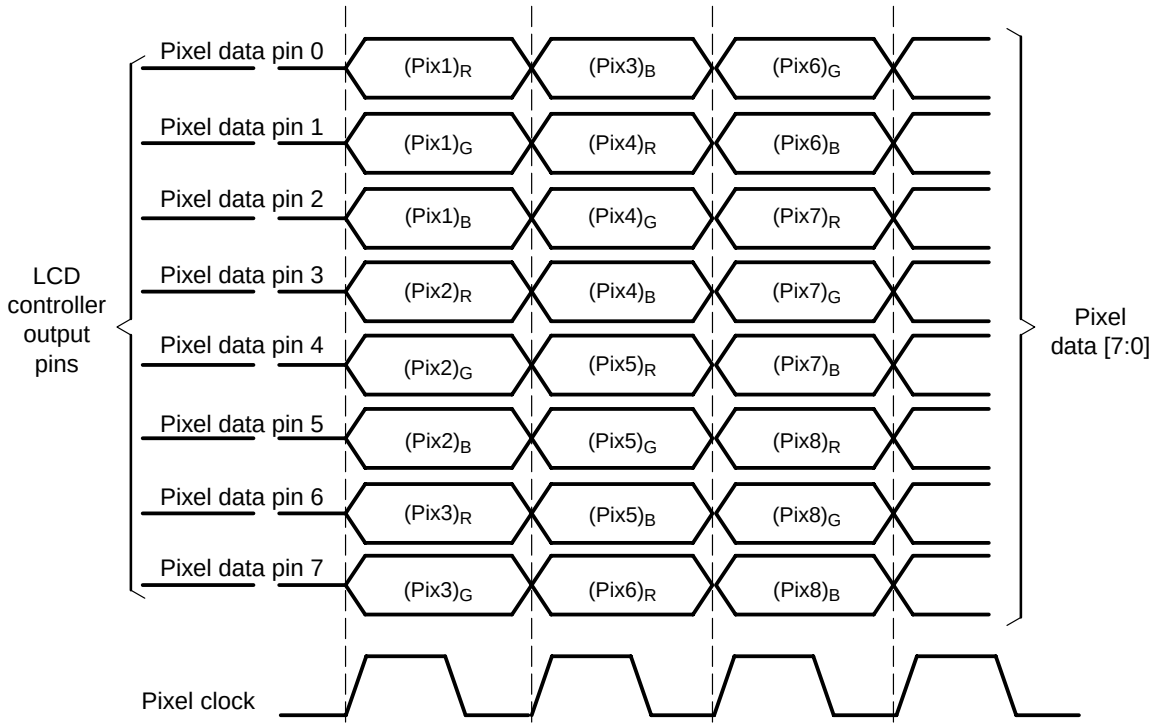
Color passive displays use eight data input lines. Each line represents one color component (red, green, or blue). 2 2/3 pixels are sent to the screen at each pixel clock.

Note:

8 data lines, 1 line per color component, 3 color components per pixel lead to 2 2/3 pixels on 8 data lines.

Figure 8–21 shows the passive color mode (IPC = 0, see IPC bit in Section 8.3.4, *LCD Timing 2 Register*).

Figure 8–21. Passive Color Mode



The situation returns to its initial state after the pixel clock toggles three times. At the fourth clock cycle, the figure becomes identical to itself (the number of pixels displayed is an integer).

❑ ***Active Matrix Technology***

In TFT displays, the dithering logic and the output FIFO are always bypassed. This means that data output from the palette is sent directly to the display. In 16-BPP mode, even the palette is bypassed so that data passes directly from the memory to the panel without being processed.

Consequently, at each pixel clock, only one pixel is sourced to the display.

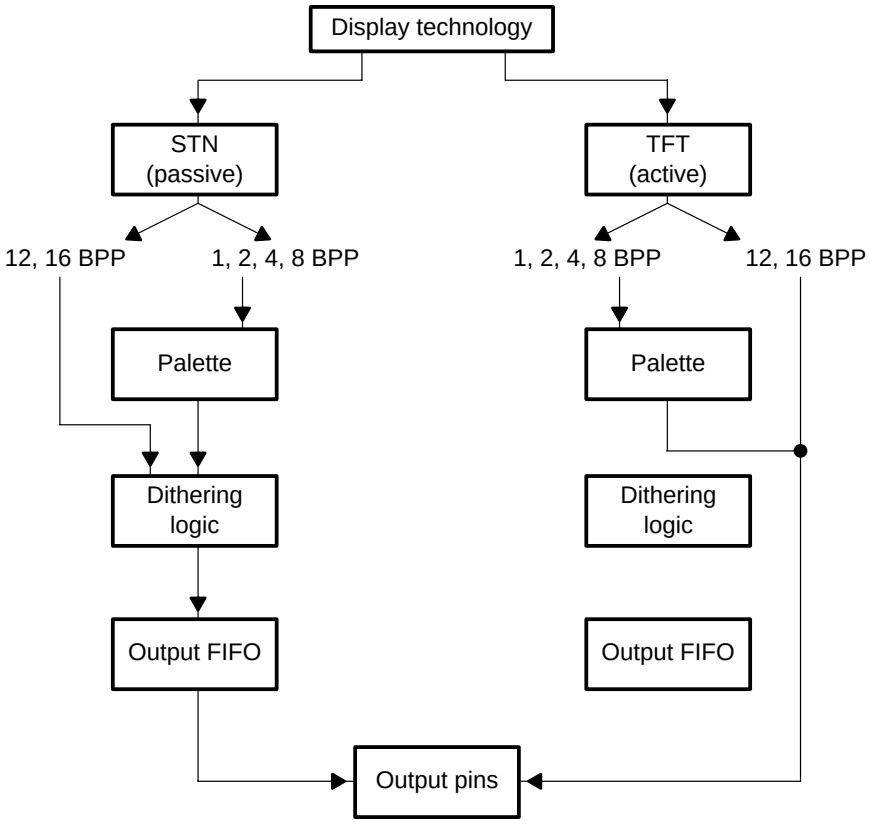
Table 8–5 is a summary of the number of pixels displayed on the screen at each pixel clock for different technologies.

Table 8–5. Number of Pixels Displayed per Pixel Clock

Number of Pixels	Display
1	TFT
2 2/3	STN color
4	Mono 4-bit
8	Mono 8-bit

Figure 8–22 shows the different data paths depending on the screen technology and the color depth.

Figure 8–22. LCD Controller Data Paths



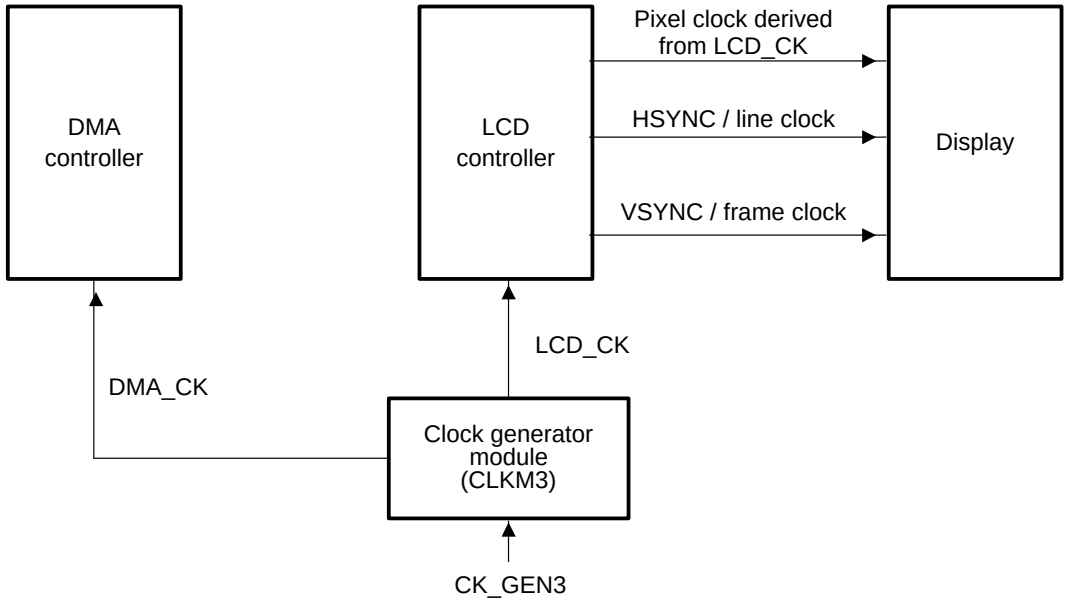
8.2.2 Control Blocks

The previous section explained the data flow from the memory to the LCD panel and the different modules they pass through. The whole process is governed by registers and timing, described in this section.

8.2.2.1 Timing

This section details the various clocks and signals. Figure 8–23 shows input and output LCD controller clocks.

Figure 8–23. Input and Output Clocks



Pixel Clock

The pixel clock frequency is derived from the LCD clock (LCD_CK), which belongs to the traffic controller (TC) domain. This frequency is the output of the on-chip PLL: LCD_CK = (CK_GEN3/1, 2, 4, or 8). See Chapter 5, *Clock Generation and Reset Module.*

You can program the pixel clock from LCD_CK / 2 to LCD_CK / 255. The divider is called PCD (see Section 8.3.4.1, *Pixel Clock Divider*).

The LCD_CK frequency must always be lower than or equal to the TC_CK frequency.

The pixel clock is used by the LCD display to clock the pixel data into the line shift register.

- **Passive Matrix Technology**

  In passive mode, the pixel clock transitions only when valid data is available on the data lines. It does not transition during wait state insertion or when the line clock is asserted.

- **Active Matrix Technology**

  In active mode, the pixel clock transitions continuously as long as the LCD is enabled, depending on the PXL_GATED bit in the LCD control register (see Section 8.3.1, *LCD Control Register*). Setting the PXL_GATED bit to 1 allows the pixel clock to toggle only when there is valid data to display.

Line Clock (HSYNC)

The line clock toggles after all pixels in a line have been transmitted to the LCD driver and a programmable number of pixel clock wait states has elapsed both at the beginning and end of each line.

For more information, see Section 8.3.2.2, *Horizontal Synchronization Pulse Width*.

❏ **Active Matrix Technology**

The line clock is also used by TFT displays as the horizontal synchronization signal (HSYNC).

The timings of the line clock pins are programmable to support:

❏ Delay insertion both at the beginning and end of each line (see Section 8.3.2.3, *Horizontal Front Porch (HFP) Bits*) and Section 8.3.2.4, *Horizontal Back Porch (HBP))* bit

❏ Line clock polarity (see Section 8.3.4.5, *Invert HSYNC (IHS)*) bit

❏ Line clock pulse width, driven on the rising or falling edge of the pixel clock (see Section 8.3.2.2, *Horizontal Synchronization Pulse Width (HSW))*, Section 8.3.4.8, *HSYNC/VSYNC Rise or Fall Programmable (RF)*, and Section 8.3.4.9, *HSYNC/VSYNC ON or OFF (ON_OFF) Bits*)

Frame Clock (VSYNC)

The frame clock toggles after all lines in a frame have been transmitted to the LCD driver and a programmable number of line clock cycles has elapsed both at the beginning and end of each frame.

For more information, see Section 8.3.3.2, *Vertical Synchronization Pulse Width*.

❏ **Passive Matrix Technology**

In passive mode, the frame clock toggles during the first line of the screen.

❏ **Active Matrix Technology**

The frame clock occurs between two frames. In active mode, it is asserted at the end of the previous one and after a programmable number of line clock wait states (VFP) has elapsed.

The frame clock is also used by TFT displays as the vertical synchronization signal (VSYNC).

The timing of the frame clock pins is programmable to support:

❏ Delay insertion both at the beginning and end of each frame (see Section 8.3.3.3, *Vertical Front Porch (VFP) Bits* and Section 8.3.3.4, *Vertical Back Porch (VBP)Bits)*

❏ Frame clock polarity (see Section 8.3.4.4, *Invert VSYNC (IVS) Bit)*

- Frame clock pulse width, driven on the rising or falling edge of the pixel clock (see Section 8.3.3.2, *Vertical Synchronization Pulse Width (VSW),* Section 8.3.2.2, *Horizontal Synchronization Pulse Width (HSW))*, Section 8.3.4.8, *HSYNC/VSYNC Rise or Fall Programmable (RF)*, and Section 8.3.4.9, *HSYNC/VSYNC ON or OFF(ON_OFF) Bits)*

ac-Bias

The ac-bias signal can be configured to transition each time a programmable number of line clocks occurs.

- **Passive Matrix Technology**

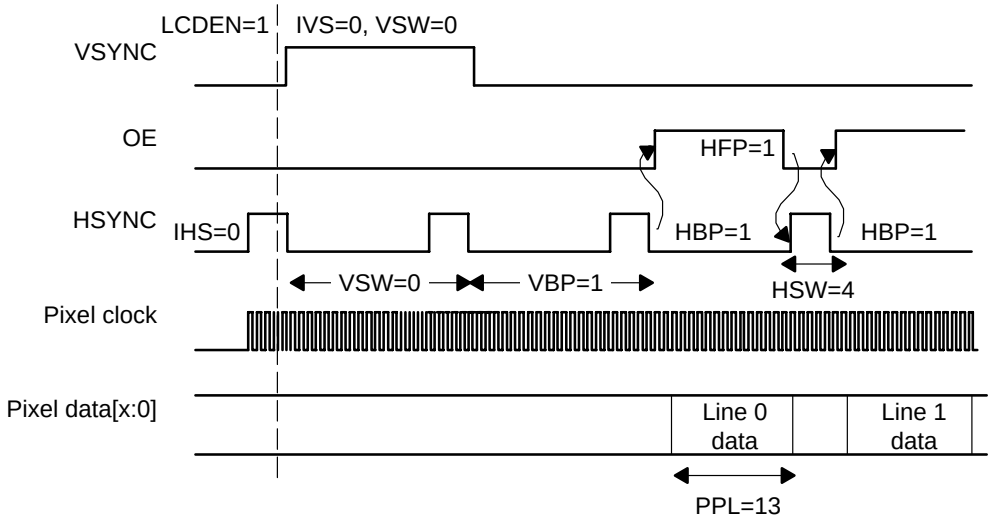
  To prevent a dc charge within the screen pixels, the power and ground supplies of the display are periodically switched. The LCD controller signals the display to switch the polarity by toggling the ac-bias pin.

- **Active Matrix Technology**

  Used in TFT mode, it acts as an output enable to signal when data must be latched from the data pins using the pixel clock.

  Figure 8–24 shows the different signals toggling in active mode.

Figure 8–24. Active (TFT) Mode Timing



Note: Ensure that HFP, HBP, PPL, and VSW values are programmed to the required value minus 1.

See the register sections, especially Sections 8.3.2, *LCD Timing 0 Register*, and 8.3.3*, LCD Timing 1 Register*, for more details on the notations.

8.2.3 Interrupts

8.2.3.1 Interrupt Sources

Several situations can generate an interrupt:

- ❏ Input and output FIFOs underrun errors
- ❏ Frame synchronization error
- ❏ When the last active frame has completed after the LCD is disabled (maskable)
- ❏ After a programmed number of transitions of the ac-bias pin (passive mode)
- ❏ When the display has reached the user-programmed line number (maskable)
- ❏ VSYNC interrupt after every end of frame (maskable)
- ❏ Palette loading (maskable)

Every hardware-detected event signals an interrupt request to the interrupt controller.

Each interrupt is signaled by a bit in the LCD controller status register. Each of the LCD status bits signals an interrupt request as long as the bit is set. Once the bit is cleared, the interrupt is cleared. Read/write bits are called status bits; read-only bits are called flags. Status bits are referred to as *sticky* (that is, once set by hardware, they must be cleared by software). Read-only flags are set and cleared by hardware; writes have no effect.

Some interrupts are also maskable. For masked bits, see Section 8.3.1, *LCD Control Register.*

Note:

A synchronization interrupt occurs if the programmed LCD displays information settings, such as pixels-per-line, lines-per-frame, color/monochrome mode, and bits-per-pixel, are not in accordance with the video buffer size programmed in the DMA.

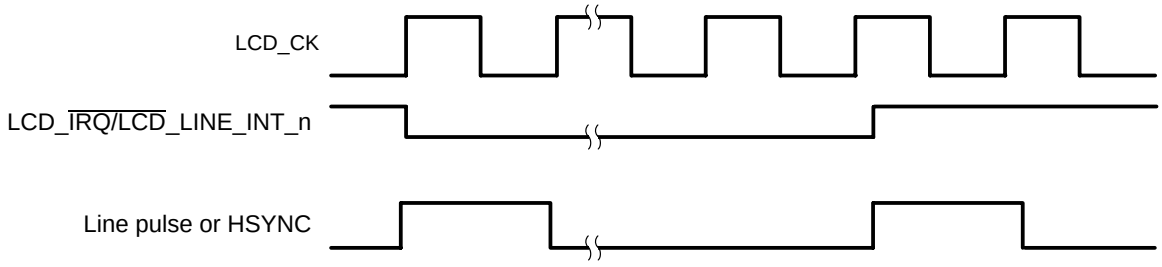
8.2.3.2 Tearing Effect

A synchronization mismatch between the frame buffer and the display refreshes can lead to images that appear to be stretched on the screen.

To prevent this, a synchronization mechanism is needed between the LCD controller and the logical channel (LCh) to update the frame buffer. For this purpose, a line comparator is implemented in the LCD controller. This comparator delivers an interrupt when the display reaches a predefined line number. This interrupt is a level signal that stays active during the programmed line of the display.

See Section 8.3.5.5, *Line Interrupt (LINE_INT*). Figure 8–25 shows the line interrupt output signal behavior for LINE_INT_CLR_SEL = 1 (see Section 8.3.1.11, *Line Interrupt Clear Select Bit (LINE_INT_CLR_SEL)*).

Figure 8–25. Line Interrupt Output Signal Transitions



Note that LCD_NIRQ and LCD_LINE_INT_n have the same behavior and are active-low signals.

The line interrupt must be connected as a DMA request line to the system DMA. This can then be used by any generic LCh (LCh 0-15) to synchronize a block transfer (BS).

Because all interrupts and DMA request inputs and outputs are brought out to OMAP top level boundary, the DMA request mapping is not predefined. See the chip top-level specification for the exact DMA request mapping.

8.2.4 LCD Subpanel Display Support

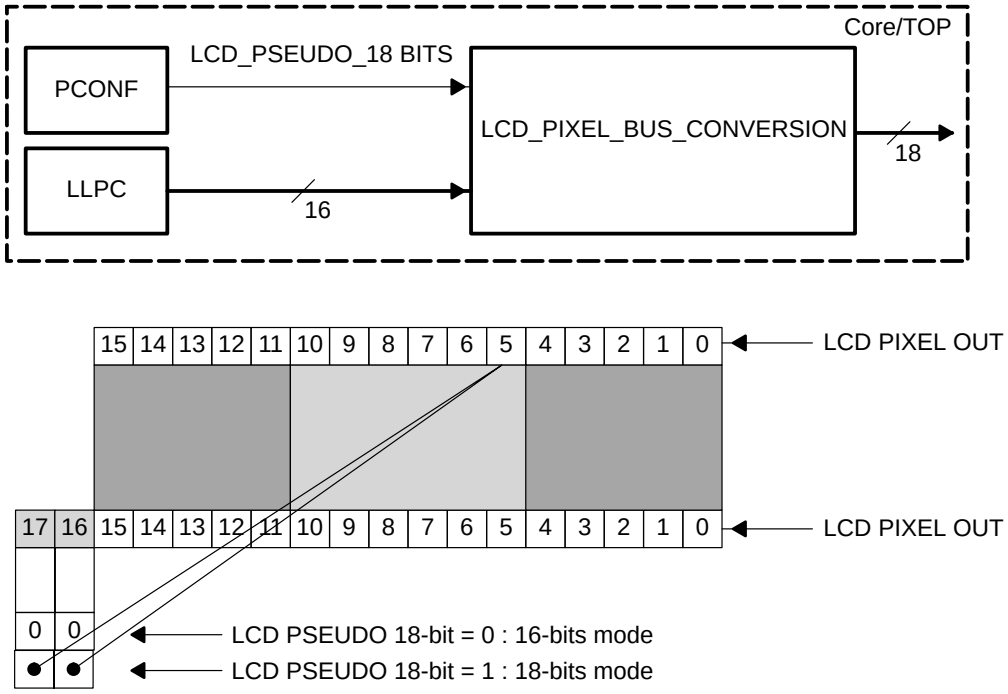
8.2.4.1 Principle

The subpanel display register supports the ability to display only the first or last x lines of the panel and send a fix contents for the others.

This function is used for power-saving. To display an image to a small portion of the screen (for example, from line 0 to line 15), the data is read from system memory (frame buffer) via the DMA. For line 16 up to line N, where n is the number of lines per panel (LPP), data is read from an LCD register (DPD: default pixel data) instead of through the DMA. Reading into a register requires no access to the frame buffer and saves power. There is no need to go off-chip and do memory reads. In addition, the DMA can shut off its clocks during this DPD value filling.

For more information on the subpanel functionality, see Section 8.3.6, *LCD Subpanel Display Support.*

Figure 8–26. LCD Pseudo-18-Bit



The LCD subpanel supports pseudo-18-bit pixel data. The LCD controller allows using the full LCD color space. That has been done in LCD_PIXEL_ BUS_CONVERSION (in MPU block).

The two MSBs are tied to 0 if 16-bit data are required. Else, in 18-bit data mode, they are equal to the two green-color LSBs.

8.3 Registers

The LCD controller contains eight registers:

- ❏ Four control registers
- ❏ Two status registers (including one display status register)
- ❏ One register related to the subpanel mode
- ❏ One specific register to define the line number where a line interrupt occurs

Table 8–6 shows the LCD controller registers and their physical addresses.

Table 8–6. LCD Controller Registers

Name	Offset	Description	Type
LCDCONTROL	0x 00	LCD control register	R/W
LCDTIMING0	0x 04	LCD timing 0 register	R/W
LCDTIMING1	0x 08	LCD timing 1 register	R/W
LCDTIMING2	0x 0C	LCD timing 2 register	R/W
LCDSTATUS	0x 10	LCD status register	R/W and R
LCDSUBPANEL	0x 14	LCD subpanel display register	R/W
LCDLINEINT	0x 18	Line interrupt register	R/W
LCDDISPLAYSTA-TUS	0x 1C	Display status register	R

8.3.1 LCD Control Register (LCDCONTROL)

The LCD control register (LCDCONTROL) contains bit-fields to enable/disable the LCD controller to define:

- ❏ The height and width of the screen being controlled
- ❏ Color or monochrome mode
- ❏ Passive or active display
- ❏ Polarity of the control lines
- ❏ Pulse width of the line and frame clocks
- ❏ The pixel clock and ac-bias frequency
- ❏ The number of delays to insert before/after each line and after each frame
- ❏ Interrupt mask bits

An additional control field exists to tune the DMA performance, based on the type of memory system in which the LCD controller is used. This field controls the placement of a minimum delay between each LCD palette request to ensure that enough bus bandwidth is given to other system accesses (see Section 8.3.1.13, *FIFO DMA Request Delay*). This field is used only for palette load.

Table 8–7 describes the LCD control register bits.

Table 8–7. LCD Control Register (LcdControl) Bit Descriptions

Bit	Name	Description	Reset
31:25	565 STN	12-BPP (565) mode 0: Off 1: On (16-bit data in frame buffer, but only 12 bits are dithered and sent out)	x
24	565 STN	12-BPP (565) mode 0: Off 1: On (16-bit data in frame buffer, but only 12 bits are dithered and sent out)	0
23	TFT MAP	TFT alternate signal mapping 0: Output pixel data for 1, 2, 4, 8, and 12 BPP modes is right-aligned on pixel data[11:0]. 1: Output pixel data for 1, 2, 4, 8, and 12 BPP modes is converted to 5, 6, 5 format and uses pins [15:0].	0
22	NM	Nibble mode 0: Nibble mode is disabled. 1: Nibble mode is enabled.	0
21-20	PLM	Palette loading mode 00: Palette and data loading 01: Palette loading 10: Data loading 11: Not connected	0
19-12	FDD	FIFO DMA request delay Encoded value (0-255) used to specify the number of LCD_CK cycles. The input FIFO DMA request must be disabled. The clock count starts after 16 words read in the input FIFO. Programming FDD = 00h disables this function.	0
11	PXL_GATED	Pixel gated (for TFT mode only) 0: Pixel clock always toggles. 1: Pixel clock only toggles when there is valid data to display.	0
10	LINE_INT_CLR_SEL	Line interrupt clear select bit 0: TIPB must write 0 to clear the line interrupt status register. 1: Line interrupt status register is reset at the end of the line.	0
9	M8B	Mono 8-bit mode 0: Pixel data [3:0] is used to output four pixel values to the panel at each pixel clock transition. 1: Pixel data [7:0] is used to output eight pixel values to the panel each pixel clock transition. This bit is ignored in all other modes.	0
8	LCDBE	LCD big endian 0: Little endian operation is selected; frame/pin buffer data is arranged into individual words of memory starting with the least significant nibble or byte. 1: Big endian operation is selected; frame/pin buffer data is arranged into individual words of memory starting with the most significant nibble or byte.	0
7	LCDTFT	LCD TFT 0: Passive or STN display operation is enabled; dither is logic enabled. 1: Active or TFT display operation enabled. External palette and DAC required. Dither logic and output FIFO bypassed. Pin timing changes to support continuous pixel clock, output enable, VSYNC, and HSYNC.	0

Note: Reserved bit reset values are undefined, but they return 1s when read.

Table 8–7. LCD Control Register (LcdControl) Bit Descriptions (Continued)

Bit	Name	Description	Reset
6	LINE_INT_MASK	Line interrupt mask (dedicated line) 0: Masks the dedicated line interrupt (LINE_INT), which is connected to the ICD_LINE_INT_n dedicated output line. 1: Mask not active	0
5	LINE_INT_NIRQ_ MASK	Line interrupt mask 0: The LINE_INT_NIRQ interrupt is masked. 1: The LINE_INT_NIRQ interrupt is unmasked.	0
4	LoadMask	Load mask 0: Masks the loaded palette interrupt, which is connected to the ICD_NIRQ shared output line. 1: Mask not active	0
3	DoneMask	Done mask 0: Masks the frame done (Done) interrupt, which is connected to the ICD_NIRQ shared output line. 1: Mask not active	0
2	VSYNC_MASK	LCD VSYNC interrupt mask 0: Interrupt to the LCD_NIRQ is masked. 1: Interrupt to the LCD_NIRQ is unmasked.	0
1	LCDBW	LCD monochrome 0: Color operation enabled 1: Monochrome operation enabled	0
0	LCDEn	LCD controller enable 0: LCD controller disabled 1: LCD controller enabled	0

Note: Reserved bit reset values are undefined, but they return 1s when read.

8.3.1.1 LCD Enable (LCDEN)

The LCD enable (LCDEN) bit is used to enable and disable all LCD controller operation.

When LCDEN = 0, the LCD controller is disabled.

When LCDEN = 1, the LCD controller is enabled.

Note:

You must program all other control bit-fields before setting LCDEN = 1 and must also disable the LCD controller when changing the state of any control bit within the LCD controller.

You can program the LCD control register (LCDCONTROL) last, and configure all 25 bit fields at the same time via a word32 write to the register. If you clear LCDEN bit while the LCD controller is enabled, you can complete transmission of the current frame before being disabled. Completion of the current frame is signaled by the LCD controller to the DMA by setting the frame done (DONE) bit within the LCD controller status register (see Section 8.3.5, *LCD Controller Status Register*), which generates an interrupt request.

If the LCD controller is disabled, the signals on pixel data [15:0] pins are set to 0 and the pixel clock, frame clock, line clock, and ac-bias signals are set to

their inactive state. This can be 0 or 1, depending on the inversions programmed in the timing 2 register (see Section 8.3.4, *LCD Timing 2 Register*).

8.3.1.2 LCD Monochrome (LCDBW)

The color/monochrome select (LCDBW) bit is used to determine whether the LCD controller operates in color or monochrome mode.

When LCDBW = 0:

- Color mode is selected.
- Palette entries are 12 bits wide *effective* (4 bits per color, which means 15 color levels for the red, green, and blue components, respectively).
- All three dither blocks are used (in passive mode only: LCDTFT = 0), one for each color component (R, G, B).

When LCDBW = 1:

- Monochrome mode is selected.
- Palette entries are 4 bits wide *effective* (15 levels of grayscale).
- 4 or 8 data lines are enabled, according to the mono 8-bit mode (M8B).

Table 8–8. LCD Controller Data Pin Utilization for Mono/Color Passive/Active Panels

Color/Mono BPP	Passive/Active Panel	Pins
Mono 1, 2, 4	Passive	Pixel data [3:0]
Mono 8	Passive	Pixel data [7:0]
Color 1, 2, 4, 8, 12	Passive	Pixel data [7:0]
Color 1, 2, 4, 8, 12	Active	Pixel data [11:0] or pixel data [15:0] according to TFT map bit in LCD control register
Color 16	Active	Pixel data [15:0]

Table 8–8 shows which set of LCD data pins (PIXEL DATA […]) is used for each mode of operation.

Note:

Unused pixel data bits always remain low.

8.3.1.3 LCD Vertical Synchronization Mask (VSYNC_MASK)

The LCD VSYNC_mask masks the VSYNC interrupt in the status register (see Section 8.3.5.1, *VSYNC Interrupt*) going to the LCD_NIRQ when VSYNC_MASK bit is 0. When it is 1, the VSYNC bit affects the LCD_NIRQ.

8.3.1.4 LCD Done Mask (DONEMASK)

The LCD done mask (DONEMASK) bit masks the path between the frame done interrupt and ICD_NIRQ (see Section 8.3.5.1., *Frame Done (Read-Only)*.

When DONEMASK = 0, the frame done interrupt is masked.

When DONEMASK = 1, the frame done interrupt is not masked.

8.3.1.5 LCD Loading Mask (LOADMASK)

The LCD loading mask (LOADMASK) bit masks the path between the palette loading interrupt signal (in the LCD status register) and ICD_NIRQ.

When LoadMask = 0, the loading interrupt is masked.

When LoadMask = 1, the loading interrupt is not masked.

8.3.1.6 Line Interrupt Mask (LINE_INT_NIRQ_MASK)

This LINE_INT_NIRQ_MASK bit masks the path going to the shared interrupt (ICD_NIRQ).

When LINE_INT_NIRQ_MASK = 0, the path between the line interrupt and ICD_NIRQ (shared line) is masked. When LINE_INT_NIRQ_MASK = 1, it is unmasked: Any interrupt—among those who share the LCD_NIRQ output line—can occur, including the line interrupt when the display reaches the programmed line number.

8.3.1.7 Line Interrupt Mask (LINE_INT_MASK)(Dedicated Line)

This LINE_INT_MASK bit masks or unmasks the connection to the dedicated top-level entity signal LCD_LINE_INT_n, which is set to 0 when the display reaches the user-programmed line number. See Section 8.3.5.5, *Line Interrupt,* and Figure 8–37, *Line Interrupt Path*).

When LINE_INT_MASK = 0, the connection is masked.

When LINE_INT_MASK = 1, the connection is unmasked.

Note:

When this bit is disabled, the LINE_INT still can be used as source to generate LCD_NIRQ.

8.3.1.8 LCD TFT (LCDTFT)

The LCD TFT (LCDTFT) bit selects whether the LCD controller operates in passive (STN) or active (TFT) display control mode.

When LCDTFT = 0, passive or STN mode is selected. LCD data flows from memory (frame buffer) to the palette (via the LCD dedicated DMA channel) to the dithering logic and the output FIFO before being output on the LCD data pins.

Figure 8–27 and Figure 8–28 describe the clocks and data pin behaviors in monochrome and color passive modes, respectively.

Figure 8–27. Monochrome Passive Mode Pixel Clock and Data Pin Timing

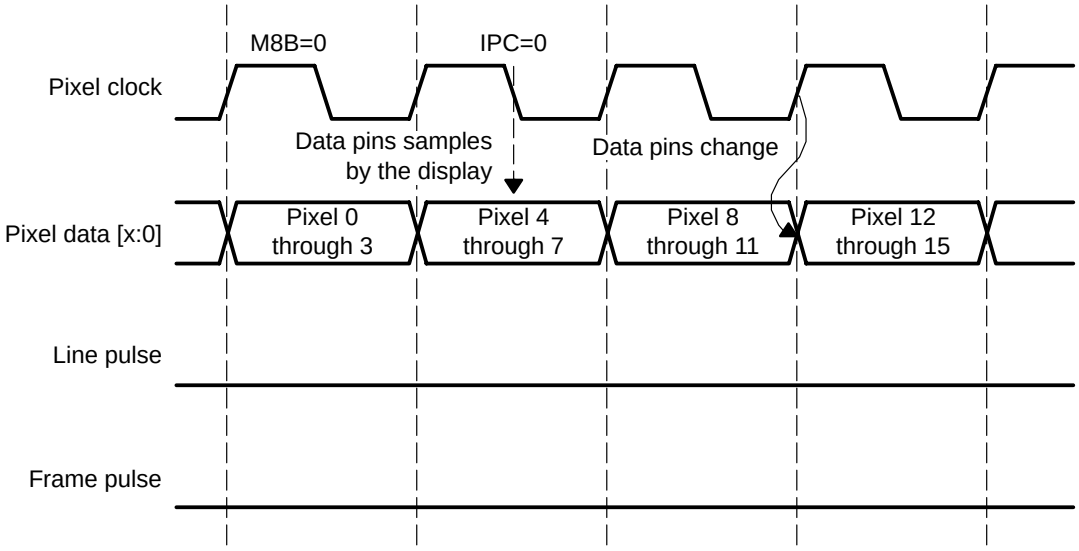
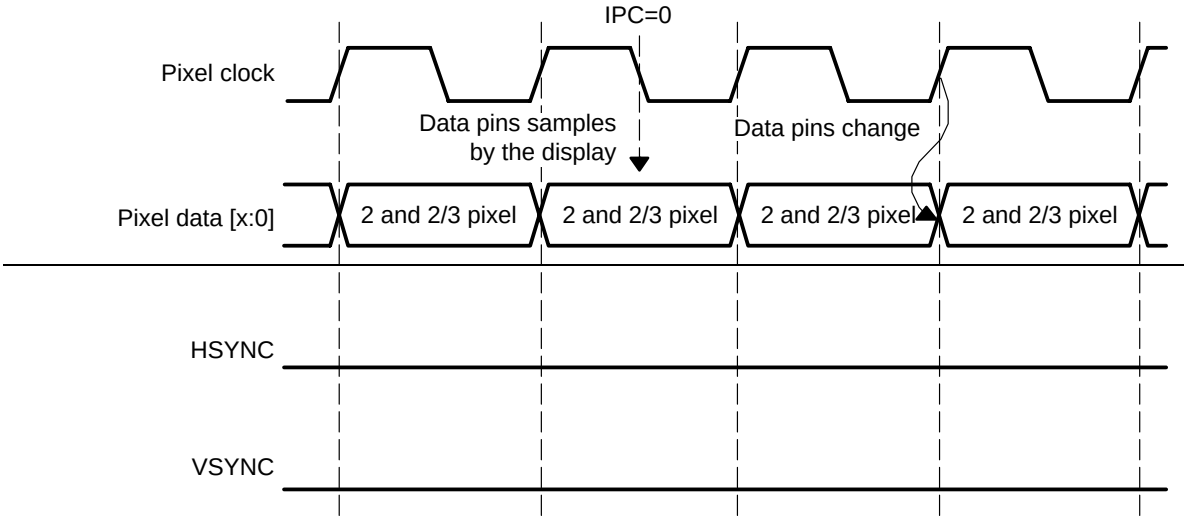


Figure 8–28. Color Passive Mode Pixel Clock and Data Pin Timing



When LCDTFT = 1, active or TFT mode is selected.

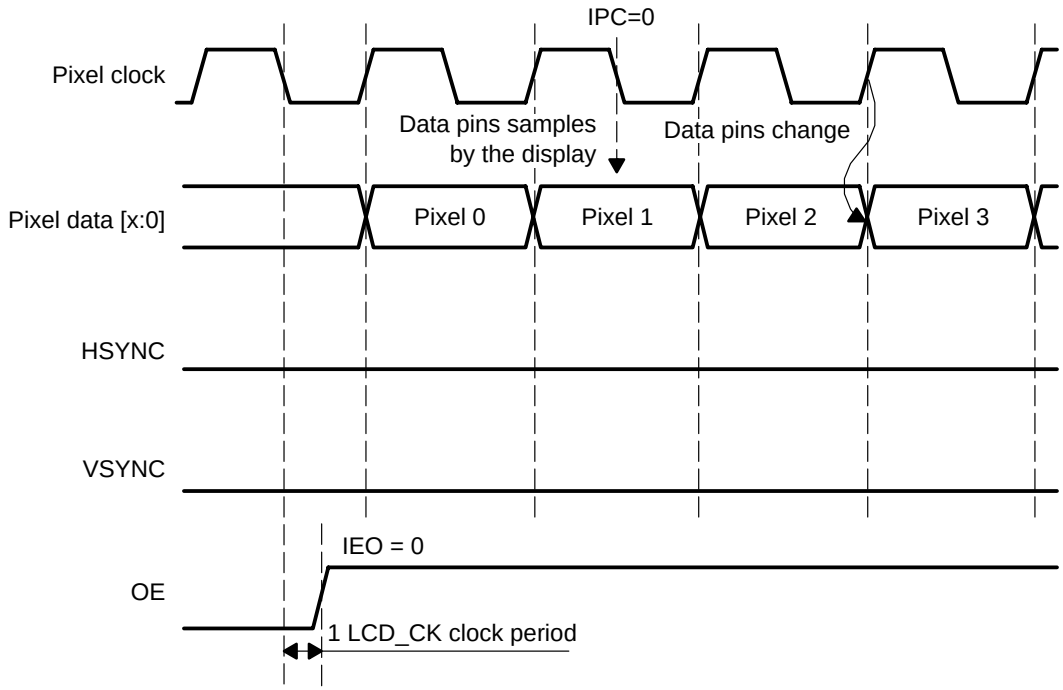
Video data is transferred via the DMA from memory to the input FIFO, and then is unpacked and used to select an entry from the palette (for 1, 2, 4, and 8 bits per-pixel modes), just as in passive mode.

The value read from the palette, however, bypasses both the LCD dither logic and the output FIFO to be output on the LCD data pins in TFT mode. The pixel size within the frame buffer is increased to 16 bits when 12- or 16-bit pixel encoding mode is enabled (BPP=1XX).

Remember that the palette is bypassed in 12-BPP for passive mode and 16-BPP for active mode The palette is also bypassed in 12-BPP TFT, because it is derived from the 16 BPP mode. See 8.3.1, *LCD Controller Data Paths*.

Figure 8–29 describes the clocks and data pin behavior in active mode.

Figure 8–29. Active Mode Pixel Clock and Data Pin Timing



The size of the pixel encoding is increased in TFT mode because the LCD dither logic is bypassed (the dither logic only supports 4 bits to encode each color component R, G, B that limits the pixel encoding size in passive mode). Increasing the size of the pixel representation allows a total of 64K colors to be addressed using an off-chip palette in conjunction with the LCD controller.

8.3.1.9 LCD Big Endian (LCDBE)

The LCD big endian (LCDBE) bit selects whether the LCD controller views the frame buffer organization as big or little endian.

When LCDBE = 0, little endian mode is selected, and pixels are packed into words starting with the least significant nibble, byte, or half-word.

When LCDBE = 1, big endian mode is selected, and pixel data is packed into words starting with the most significant nibble, byte, or half-word. See Figure 8–5 through Figure 8–19.

The big/little endian select bit also affects the ordering of palette buffer entries in the frame buffer.

When LCDBE = 0, half-word palette entries are packed into words starting with the least significant half-word. When LCDBE = 1, palette entries are packed into words starting with the most significant half-word.

Note:

LCDBE does not affect the ordering of the 4-bit red/green/blue bit-fields, the 4-bit mono field within each 16-bit palette entry, or the 3-bit field, dedicated to the BPP configuration, contained within palette entry 0.

8.3.1.10 Mono 8 Bit Mode (M8B)

The mono 8-bit mode (M8B) bit selects whether four or eight data lines are used to output pixel data to the LCD screen.

When M8B = 0, pixel data [3:0] is used to output four pixel values to the LCD panel at each pixel clock transition.

When M8B = 1, pixel data [7:0] is used to output eight pixel values to the LCD panel at each pixel clock transition.

Note:

M8B does not affect any of the color modes or TFT.

8.3.1.11 Line Interrupt Clear Select Bit (LINE_INT_CLR_SEL)

The LINE_INT_CLR_SEL bit selects between two methods that clear the bit in the status register. You can select either an automatic clear or a TIPB write.

When LINE_INT_CLR_SEL = 0, write 0 to the LINE_INT status bit to clear the interrupt.

When LINE_INT_CLR_SEL = 1, the line interrupt bit in the status register is reset at the end of the programmed line.

8.3.1.12 Gated Pixel Clock (PXL_GATED)

The PXL_GATED bit selects between gated and not gated pixel clocks when in TFT mode.

When PXL_GATED = 0, the pixel clock always toggles.

When PXL_GATED = 1, the pixel clock does not toggle when there is not valid data to display. This is a power-saving option.

8.3.1.13 FIFO DMA Request Delay (FDD)

The 8-bit FIFO DMA request delay (FDD) field is used to select the minimum number of LCD_CK cycles to wait between the servicing of each DMA request issued by the LCD controller, sending an address to the input FIFO.

The goal is to ensure enough bandwidth to other system accesses. A delay of FDD cycles is inserted every 16 words read from the input FIFO. This function is a concern only in 8 BPP mode, where the palette is 256 words.

When FDD = 00h, the FIFO DMA request delay function is disabled. This function is only used for palette loading.

8.3.1.14 Palette Loading (PLM)

The 2-bit palette loading field describes how the palette loading behaves when each new frame is loaded from memory.

- In 00 mode, the data in the frame buffer represents the palette data and the picture data. Both palette and picture data are loaded.
- 01 is the palette-only mode. The data in the frame buffer just represents a new palette to be loaded. This data is loaded and placed into the palette. Be sure to turn off the LCD after getting the loading interrupt; otherwise, the LCD behavior is unpredictable.
- When in data-loading mode (PLM = 10), the data in the frame buffer represents only the picture data (data-only). This data is then used as an index (in the palette) or sent directly out. This mode assumes the palette was previously loaded. There is no need to keep loading the palette if it is not changing. As a matter of fact, in data-only mode, the BPP is fixed and cannot change on-the-fly because the palette is not loaded at every frame.
- 11 mode is not connected.

8.3.1.15 Nibble Mode (NM)

This bit must be set to 1 if the operating system used is Windows CE. In this case, it overrides the LCD big endian (LCDBE) mode.

If this bit is set to 0, then the nibble mode is disabled and the endianism is given by the LCDBE control bit.

The nibble mode is compared to the little and big endian modes (see Figure 8–5 through Figure 8–19).

8.3.1.16 TFT Alternate Signal Mapping (TFT MAP)

This bit is relevant only if LCDTFT = 1.

This bit field controls how the TFT pixel data is output. Through this feature 12-BPP TFT data can be output to all 16-bit LCD pins (this also applies to 1-, 2-, 4-, and 8- BPP). This feature allows you to switch BPP modes on-the-fly, duplicating the 12-bit output data across the 16 data lines if they are already hardwired to the 16 data lines.

Figure 8–30 shows how the four red, four green, and four blue bits are mapped to all pixel data [15:0] output pins when this bit is set to 1.

Figure 8–30. TFT Alternate Signal Mapping Output

Pins	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data	R3	R2	R1	R0	R3	G3	G 2	G1	G0	G 3	G2	B3	B2	B1	B0	B3

When this bit is 0, the four red, four green, and four blue data are right-aligned on pixel data [11:0]. The upper pixel data [15:12] are set to 0. There is no duplication.

8.3.1.17 16 BPP STN Mode (565 STN)

This bit is relevant only if LCDTFT = 0, but has no effect in 1-, 2-, 4- and 8-BPP modes.

If 565 STN = 0, the frame buffer organization is in 12 BPP mode. In this mode, each color component is encoded in 4 bits, as shown in Table 8–9.

Table 8–9. 12-Bit STN Data in Frame Buffer

Unused	Red	Green	Blue
15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Data ignored	R3 R2 R1 R0	G3 G2 G1 G0	B3 B2 B1 B0

If 565 STN = 1, the 16-BPP STN mode is selected. The only difference from the 12-BPP mode is how the pixel data is organized in the frame buffer and which bits are sent to the dither logic.

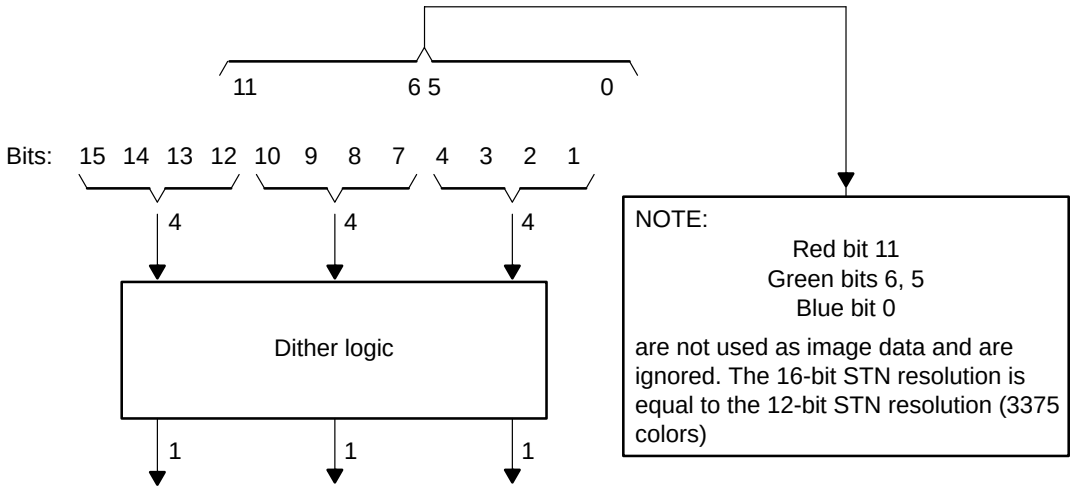
16-bit STN mode appears in the frame buffer memory as shown in Table 8–10.

Table 8–10. 16-Bit STN Data in Frame Buffer

Red	Green	Blue
15 14 13 12 11	10 9 8 7 6 5	4 3 2 1 0
R4 R3 R2 R1 R0	G5 G4 G3 G2 G1 G0	B4 B3 B2 B1 B0

Nevertheless, 16-bit STN mode only sends 12 bits to the dithering logic as well as the 12-BPP STN mode. The LSB bit of the red component (bit 11), the two LSBs of green (bits 6 and 5), and the LSB of blue (0) are not sent to the dithering logic.

Figure 8–31. 16 BPP STN Mode



This 12-BPP (5-6-5) mode can be used if the operating system does not support 12 BPP in the frame buffer. Data is arranged in 16 BPP instead, but only 12 bits are dithered and sent to the display.

8.3.2 LCD Timing 0 Register (LCDTIMING0)

The LCD timing 0 register (LCDTIMING0) contains four bit-fields that are used as modulus values for a collection of down counters. This register controls HSYNC-signal generation.

Table 8–11. LCD Timing 0 Register (LCDTIMING0) Bit Descriptions

Bit	Name	Description	Reset
31-24	HBP	Horizontal back porch	0x00
		Encoded value (from 1–256) used to specify the number of pixel clock periods to add to the beginning of a line transmission before the first set of pixels is output to the display (program to value required minus 1).	
		Note that the pixel clock is held in its inactive state during the beginning of line wait period in passive display mode and is permitted to transition in active display mode.	
23-16	HFP	Horizontal front porch	0x00
		Encoded value (from 1–256) used to specify the number of pixel clock periods to add to the end of a line transmission before line clock is asserted (program to value required minus 1).	
		Note that the pixel clock is held in its inactive state during the end of line wait period in passive display mode and is permitted to transition in active display mode.	
15-10	HSW	Horizontal synchronization pulse width	0x00
		Encoded value (from 1–64) used to specify the number of pixel clock periods to pulse the line clock at the end of each line (program to value required minus 1).	
		Note that pixel clock is held in its inactive state during the generation of the line clock in passive display mode and is permitted to transition in active display mode.	
9-0	PPL	Pixels per line	0x00F
		Encoded value (from 1–1024) used to specify the number of pixels contained within each line on the LCD display (program to value required minus 1).	

8.3.2.1 Pixels-Per-Line (PPL)

The pixels-per-line (PPL) bit-field is used to specify the number of pixels in each line on the screen. It represents the screen width. PPL is a 10-bit value. Taking into account that the bottom 4 bits of this register are not used and always read as 1, it is possible to support displays with the number of pixels-per-line ranging between 16 and 1024. PPL is used to count the correct number of pixel clocks that must occur before the line clock can be pulsed.

Notes:

PPL must be programmed to the value required minus 1 (0x27F for a 640-pixel-per-line LCD panel).

PPL must be a multiple of 16 pixels.

8.3.2.2 Horizontal Synchronization Pulse Width (HSW)

The 6-bit horizontal synchronization pulse width (HSW) field is used to specify the pulse width of the line clock in passive mode or horizontal synchronization pulse in active mode. The line clock (or HSYNC) is asserted each time a line or row of pixels is output to the display and a programmable number of pixel clock delays has elapsed. When line clock is asserted, the value in HSW is transferred to a 6-bit down counter that uses the programmed pixel clock frequency to decrement. When the counter reaches zero, the line clock is negated. HSW can be programmed to generate a line clock pulse width ranging from 1–64 pixel clock periods (program to value required minus 1).

Note:

The pixel clock does not transition during the line clock pulse in passive display mode, but it transitions in active display mode.

8.3.2.3 Horizontal Front Porch (HFP)

The 8-bit horizontal front porch (HFP) field is used to specify the number of dummy pixel clocks to insert at the end of each line or row of pixels before pulsing the line clock pin. Once a complete line of pixels is transmitted to the LCD driver, the value in HFP is used to count the number of pixel clocks to wait before pulsing the line clock. HFP generates a wait period ranging from 1–256 pixel clock cycles (program to value required minus 1).

Note:

The pixel clock does not transition during these dummy pixel clock cycles in passive display mode, but it transitions continuously in active display mode.

8.3.2.4 Horizontal Back Porch (HBP)

The 8-bit horizontal back porch (HBP) field is used to specify the number of dummy pixel clocks to insert at the beginning of each line or row of pixels. After the line clock for the previous line has been negated, the value in HBP is used to count the number of pixel clocks to wait before starting to output the first set of pixels in the next line. HBP generates a wait period ranging from 1–256 pixel clock cycles (program to value required minus 1).

Note:

The pixel clock does not transition during these dummy pixel clock cycles in passive display mode, but it transitions continuously in active display mode.

8.3.3 LCD Timing 1 Register (LCDTIMING1)

The LCD timing 1 register (LCDTIMING1) contains four bit-fields that are used as modulus values for a collection of down counters. This register controls VSYNC signal generation.

Table 8–12. LCD Timing 1 Register (LCDTIMING1) Bit Descriptions

Bit	Name	Description	Reset
31-24	VBP	Vertical back porch	0x00
		Value (from 0–255) used to specify number of line clock periods to add to the beginning of a frame before the first set of pixels is output to the display. Note that line clock transitions during the insertion of the extra line clock periods.	
23-16	VFP	Vertical front porch	0x00
		Value (from 0–255) used to specify number of line clock periods to add to the end of each frame. Note that the line clock transitions during the insertion of the extra line clock periods.	
15-10	VSW	Vertical synchronization pulse width	0x00
		In active mode (LCDTFT=1), encoded value (from 1–64) used to specify number of line clock periods (program to value required minus 1*)* to pulse the frame clock (VSYNC) pin at the end of each frame *after* the end of frame wait (VFP) period elapses. Frame clock used as VSYNC signal in active mode.	
		In passive mode (LcdTFT=0), encoded value (from 1–64) used to specify number of extra line clock periods to insert *after* the vertical front porch (VFP) period has elapsed. Note that the width of the frame clock (VSYNC) is not affected by VSW in passive mode, and that line clock transitions during the insertion of the extra line clock periods (program to value required minus 1*).*	
9-0	LPP	Lines per panel	0x000
		Encoded value (from 1–1024) used to specify number of lines per panel. It represents the total number of lines on the LCD.	

8.3.3.1 Lines Per Panel (LPP)

The lines per panel (LPP) bit-field is used to specify the number of lines or rows per LCD panel being controlled. It represents the total number of lines for the entire LCD display (the screen height). LPP is a 10-bit value, which represents between 1–1024 lines per panel. LPP is used to count the correct number of line clocks that must occur before the frame clock can be pulsed.

Note:

LPP must be programmed to the value required minus 1 (0xC7 for 200 lines per panel).

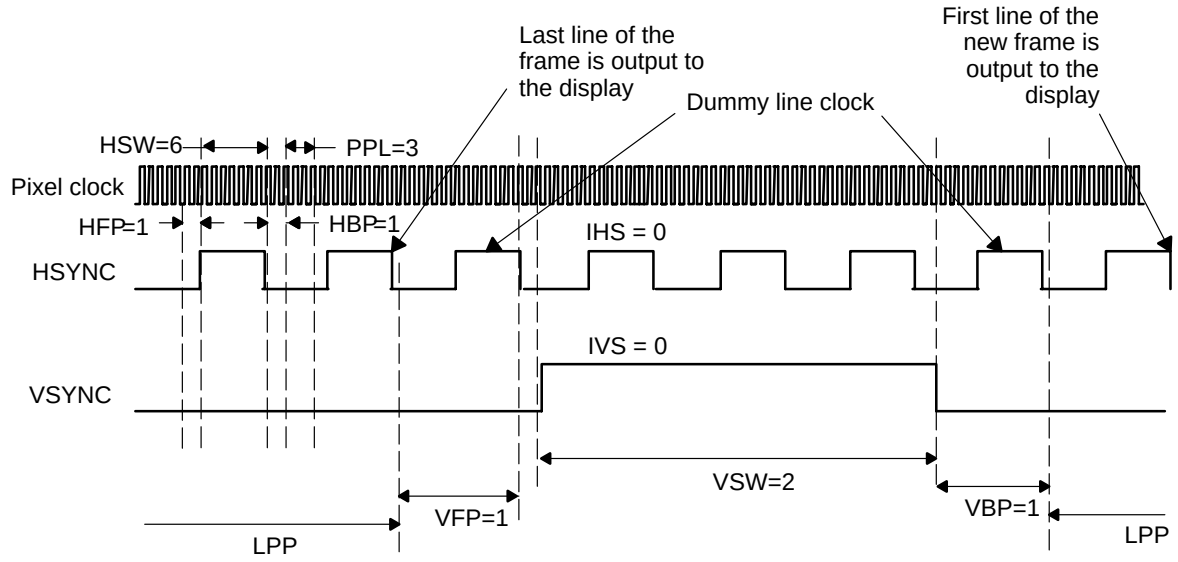
8.3.3.2 Vertical Synchronization Pulse Width (VSW)

The 6-bit vertical synchronization pulse width (VSW) field is used to specify the pulse width of the vertical synchronization pulse in active mode or is used to add extra dummy line clock cycles between the vertical front porch and vertical back porch in passive mode.

❑ **Active Matrix Technology**

In active mode (LCDTFT = 1), VSYNC is asserted each time the last line or row of pixels from the previous frame is output to the display and a programmable number of line clock delays (VFP) has elapsed. When the frame clock (VSYNC) is asserted, the value in VSW is transferred to a 6-bit down counter that uses the line clock frequency to decrement. When the counter reaches zero, the frame clock (VSYNC) is negated. VSW can be programmed to generate a vertical synchronization pulse width ranging from 1–64 line clock periods (program to value required minus 1—see Figure 8–32). The following frame starts after VSYNC is deasserted and a programmable number of line clock delays (VBP) has elapsed.

Figure 8–32. Active Matrix Timing



Note: Remember that most of the parameters (HSW, HFP, PPL, HBP) must be programmed to value required minus 1.

❑ **Passive Matrix Technology**

In passive mode (LCDTFT = 0), VSW does not affect the timing of the frame clock, but instead can be used to add extra line clock cycles between the end and beginning of frame line-clock cycle counts. The total number of line clock cycles that are inserted between each frame is equal to the sum of the values in VFP, VSW, and VBP. A counter is used to insert dummy line-clock cycles between frames by first using the value in VFP, then VSW, and then VBP. Ensure that the sum of the values in the three fields is equal to the total number of line clock cycles that are needed between frames.

The LCD controller frame clock pin is asserted on the rising-edge of the first pixel clock for each frame. The frame clock remains asserted for the

remainder of the first line as pixels are output to the display, also during the assertion of the first line clock for the frame, and then negated on the rising-edge of the first pixel clock of the second line of each frame.

Note:

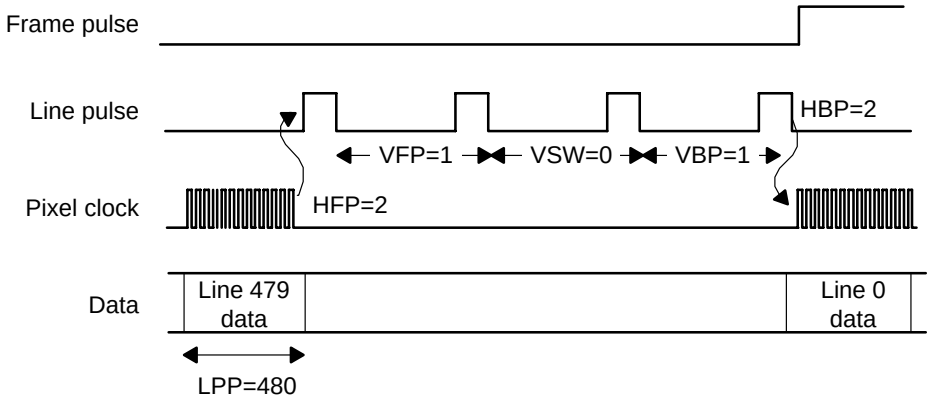
The pixel clock does not transition during the whole dummy line-clock periods that are inserted in passive mode before the frame pulse.

The line clock does transition during the insertion of the dummy line clock cycles. VSW must be long enough to load the palette.

8.3.3.3 Vertical Front Porch (VFP)

The 8-bit vertical front porch (VFP) field is used to specify the number of line clocks to insert at the end of each frame. Once a complete frame of pixels is transmitted to the LCD display, the value in VFP is used to count the number of line clock periods to wait. After the count has elapsed the VSYNC signal is pulsed in active mode or extra line clocks are inserted as specified by the VSW bit-field in passive mode. VFP generates from 0–255 line clock cycles (see Figure 8–33).

Figure 8–33. Passive Mode End of Frame Timing



Note: Remember that VSW must be programmed to value required minus 1.

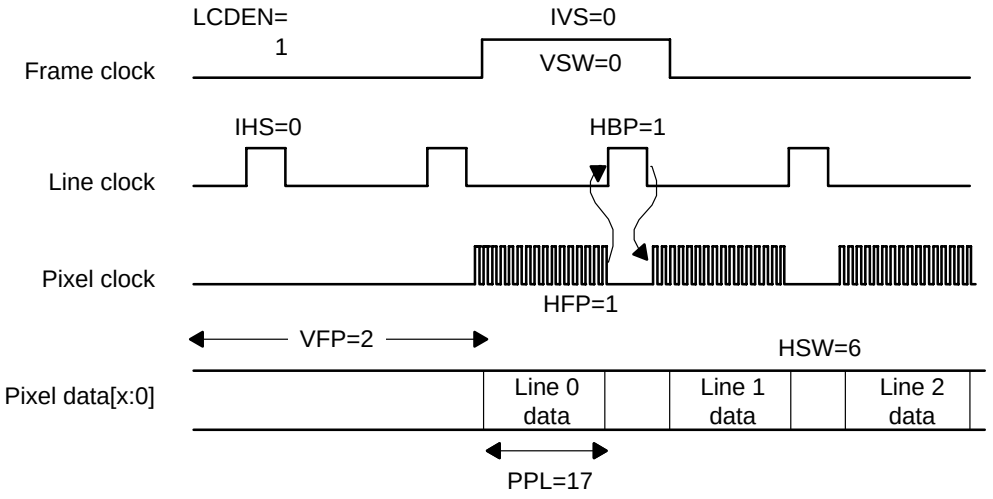
Note:

The line clock transitions during the generation of the VFP line clock periods.

8.3.3.4 Vertical Back Porch (VBP)

The 8-bit vertical back porch (VBP) field is used to specify the number of line clocks to insert at the beginning of each frame. The VBP count starts just after the VSYNC signal for the previous frame has been negated for active mode, or the extra line clocks have been inserted as specified by the VSW bit-field in passive mode. After this has occurred, the value in VBP is used to count the number of line clock periods to insert before starting to output pixels in the next frame. VBP generates from 0–255 extra line clock cycles (see Figure 8–34).

Figure 8–34. Passive Mode Beginning of Frame Timing



Note:

The line clock transitions during the generation of the VBP line clock wait periods. You must adjust the value of VBP appropriately so that enough line clock cycles are permitted to elapse. This allows the palette to be completely filled via the DMA and a sufficient number of encoded pixel values to be input from the frame buffer, processed by the dither logic, and then placed in the output FIFO, ready to be output to the LCD data lines.

8.3.4 LCD Timing 2 Register (LCDTIMING2)

The LCD timing 2 register (LCDTIMING2) contains nine different bit-fields that are used to control various functions associated with the timing of the LCD controller.

Table 8–13. LCD Timing 2 Register (LCDTIMING2) Bit Descriptions

Bit	Name	Description	Reset
31:26	RESERVED	Reserved	x
23	IEO	Invert output enable	0
		0: ac-bias pin is active high in active display mode.	
		1: ac-bias pin is active low in active display mode.	
		Active display mode: Data driven out to the LCD data lines on programmed pixel clock edge when ac-bias is active,according to the IPC bit (see Section 8.3.4.6, *Invert Pixel Clock (IPC)*. Note that IEO is ignored in passive display mode.	
22	IPC	Invert pixel clock	0
		0: Data is driven on the LCD data lines on the rising-edge of the pixel clock.	
		1: Data is driven on the LCD data lines on the falling-edge of the pixel clock.	
21	IHS	Invert HSYNC	0
		0: Line clock (HSYNC) pin is active high and inactive low.	
		1: Line clock (HSYNC) pin is active low and inactive high.	
		Active and passive mode: Horizontal synchronization pulse/line clock active between lines, after end of line wait period.	

Table 8–13. LCD Timing 2 Register (LCDTIMING2) Bit Descriptions (Continued)

Bit	Name	Description	Reset
20	IVS	Invert VSYNC 0: Frame clock (VSYNC) pin is active high and inactive low. 1: Frame clock (VSYNC) pin is active low and inactive high. Active mode: Vertical synchronization pulse is active between frames; after end of frame wait period. Passive mode: Frame clock is active during first line of each frame.	0
19-16	ACBI	ac-bias pin transitions per interrupt Value (from 0 to 15) used to specify the number of ac-bias pin transitions to count before setting the ac-bias count status bit (see Section 8.3.5.4), signaling an interrupt request. The counter is frozen when bit ABC is set and is restarted when bit ABC is cleared by software. This function is disabled when ACBI = 0x0000. This bit is relevant in passive mode only because it is used as an output enable in active mode. See Section 8.3.4.7, *Invert Output Enable (IEO)*. This bit is ignored in active mode.	0x0
24	RF	Program HSYNC/VSYNC RISE OR FALL 0: Line clock (HSYNC) and frame clock (VSYNC) are driven on falling edge of pixel clock (bit 25 must be set to 1). 1: Line clock (HSYNC) and frame clock (VSYNC) are driven on rising edge of pixel clock. (bit 25 must be set to 1).	0
25	ON_OFF	HSYNC/VSYNC pixel clock control on/off (should be ON only when in TFT mode) 0: Line clock (HSYNC) and frame clock (VSYNC) are driven on opposite edges of pixel clock than the pixel data. 1: Line clock (HSYNC) and frame clock (VSYNC) are driven according to bit 24.	0
15-8	ACB	ac-bias pin frequency (program to value required minus 1) Value (from 1 to 256) used to specify the number of line clocks to count before transitioning the ac-bias pin. This pin is used to periodically invert the polarity of the power supply to prevent dc charge build-up within the display. ACB = Number of line clocks/toggle of the ac-bias pin. This bit is relevant in passive mode only because ac-bias is used as an output enable in active mode. This bit is ignored in active mode.	0x00
7-0	PCD	Pixel clock divisor Value (from 2–255) used to specify the frequency of the pixel clock based on the LCD_CK frequency. Pixel clock frequency can range from LCD_CK /2 to LCD_CK /255. Pixel clock frequency = LCD_CK/PCD	0x00

8.3.4.1 Pixel Clock Divider (PCD)

The 8-bit pixel clock divider (PCD) field is used to select the frequency of the pixel clock. PCD can generate a range of pixel clock frequencies from LCD_CK/2 to LCD_CK/255, where LCD_CK is derived from CK_GEN3 (see Section 8.2.2.1, *Timing)*. The pixel clock frequency must be adjusted to meet the required screen refresh rate.

The refresh rate depends on:

- The number of pixels for the target display
- Whether monochrome or color mode is selected
- The number of pixel clock delays programmed at the beginning and end of each line
- The number of line clocks inserted at the beginning and end of each frame
- The width of the VSYNC signal in active mode or VSW line clocks inserted in passive mode
- The width of the line clock or HSYNC signal

All of these factors alter the time duration from one frame transmission to the next. Different display manufacturers require different frame refresh rates, depending on the physical characteristics of the display. PCD is used to alter the pixel clock frequency in order to meet these requirements. Pixel clock is used to synchronously signal the device to drive data to the LCD data pins and to signal the output FIFO to latch the data from the pins. The frequency of the pixel clock for a set PCD value or the required PCD value to yield a target pixel clock frequency can be calculated using the following equation:

PixelClock =LCD_CK/PCD

The pixel clock frequency is programmed taking into account the limitations shown in Table 8–14.

Table 8–14. Pixel Clock Frequency Programming Limitations

Type of Screen	Output (in Bits)	Min. Pixel Clock Divider
TFT 1, 2, 4, 8 BPP	12 (1 pixel)	2
TFT 12, 16 BPP	16 (1 pixel)	2
STN monochrome	4 (4 pixels)	4
STN monochrome	8 (8 pixels)	8
STN color	8 (2 2/3 pixels)	3

Note:

If PCD equals 0 or 1, the effect is undefined. Dividing the pixel clock frequency by an odd number distorts the duty cycle.

8.3.4.2 ac-Bias Pin Frequency (ACB)

The 8-bit ac-bias frequency (ACB) field is used to specify the number of line clock periods to count between each toggle of the ac-bias pin. After the LCD controller is enabled, the value in ACB is loaded to an 8-bit down counter, and the counter begins to decrement using the line clock. When the counter reaches zero it stops, the state of ac-bias pin is reversed, and the whole procedure starts again. The number of line clocks between each ac-bias pin transition ranges from 1–256 (program to value required minus 1). This line is used by the LCD display to periodically reverse the polarity of the power supplied to the screen to eliminate dc offset.

Note:

The ACB bit field has no effect in active mode. This is due to the fact that the pixel clock transitions continuously in active mode (when PIXEL_GATED = 0); the ac-bias line is used as an output enable signal. The ac-bias is asserted by the LCD controller in active mode; this occurs whenever pixel data is driven out to the data pins to signal to the display when it can latch pixels using the pixel clock.

8.3.4.3 ac-Bias Line Transitions Per Interrupt (ACBI)

The 4-bit ac-bias line transitions per interrupt (ACBI) field is used to specify the number of line transitions to count before setting the ac-bias count status (ABC) bit in the LCD controller status register, which signals an interrupt request. After the LCD controller is enabled, the value in ACBI is loaded to a 4-bit down counter, and the counter decrements each time the ac-bias line state is inverted. When the counter reaches zero it stops, and the ac-bias count (ABC) bit is set in the status register. Once ABC is set, the 4-bit down counter is reloaded with the value in ACBI and is disabled until ABC is cleared. Once ABC is cleared by the CPU, the down counter is enabled and again decrements each time the ac-bias line is flipped. The number of ac-bias line transitions between each interrupt request ranges from 0 to 15. Programming ACBI = 0h0000 disables the ac-bias line-transitions-per-interrupt function.

Note:

For the same reasons as explained in the note in Section 8.3.4.2, *ac Bias Pin Frequency (ABC)*, this bit-field has no effect in active mode.

8.3.4.4 Invert VSYNC (IVS)

The invert VSYNC(IVS) bit is used to invert the polarity of the frame clock (VSYNC).

When IVS = 1, the frame clock (VSYNC) is active low.

When IVS = 0, it is active high.

8.3.4.5 Invert HSYNC (IHS)

The invert HSYNC(IHS) bit is used to invert the polarity of the line clock (HSYNC).

When IHS = 1, the line clock (HSYNC) is active low.

When IHS = 0, it is active high.

8.3.4.6 Invert Pixel Clock (IPC)

The invert pixel clock (IPC) bit is used to select the edge of the pixel clock that drives pixel data out onto the LCD data lines.

When IPC = 1, data is driven onto the LCD data lines on the falling edge of the pixel clock.

When IPC = 0, data is driven onto the LCD data lines on the rising edge of the pixel clock.

8.3.4.7 Invert Output Enable (IEO)

The invert output enable (IEO) bit is used to select the active or inactive state of the output enable signal in active display mode. In this mode, the ac-bias pin is used as an enable that signals the device when data is actively being driven out using the pixel clock.

When IEO = 1, the ac-bias pin is active low. In active display mode, data is driven onto the LCD data lines on the programmed edge of the pixel clock when ac-bias pin is in its active state (see Section 8.3.4.6, *Invert Pixel Clock (IPC).*

When IEO = 0, the ac-bias pin is active high.

Note:

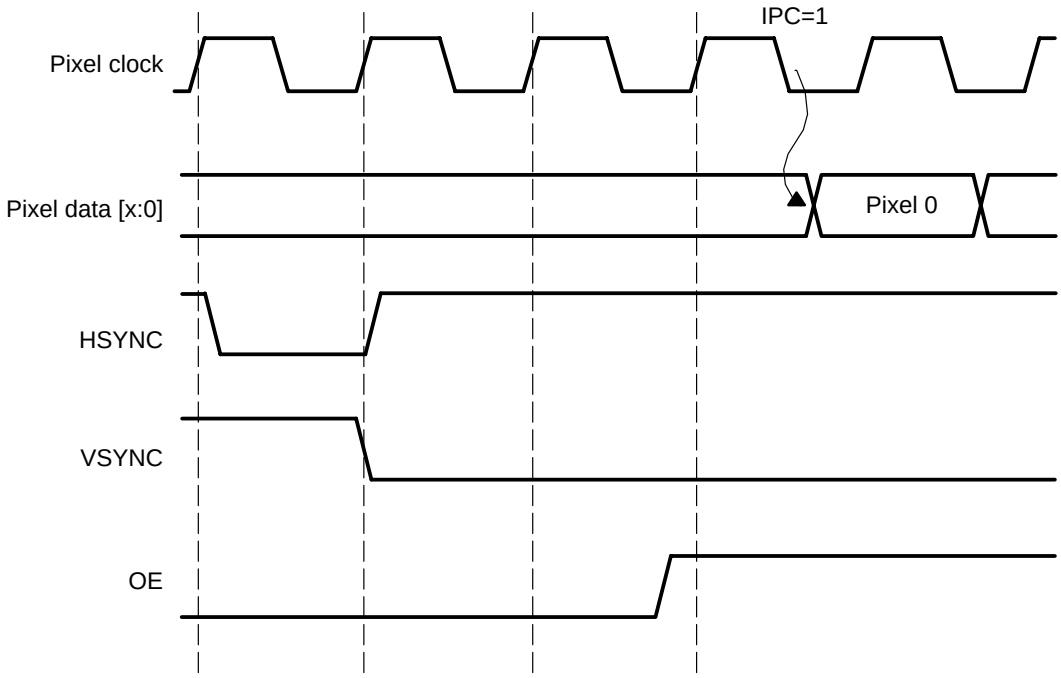
IEO does not affect the ac-bias pin in passive display mode.

8.3.4.8 HSYNC/VSYNC Rise or Fall Programmable (or not)(RF)

This bit determines whether the HSYNC/VSYNC is driven on the rising or falling edge of the pixel clock (see the HSYNC/VSYNC ON_OFF bit; ON_OFF must be turned on first). By default, the HSYNC and VSYNC signals are driven on the falling edge of the pixel clock, and the pixel data is driven on the rising edge of pixel clock. However, if the invert pixel clock (IPC) bit is set to 1, then the HSYNC and VSYNC signals are driven on the rising edge of pixel clock, and pixel data is driven on the falling edge. By setting the RF bit and enabling it (ON_OFF = 1), you can control on which edge the signals are driven.

Figure 8–35 shows the timing when ON_OFF = 0 and IPC =1 in TFT mode.

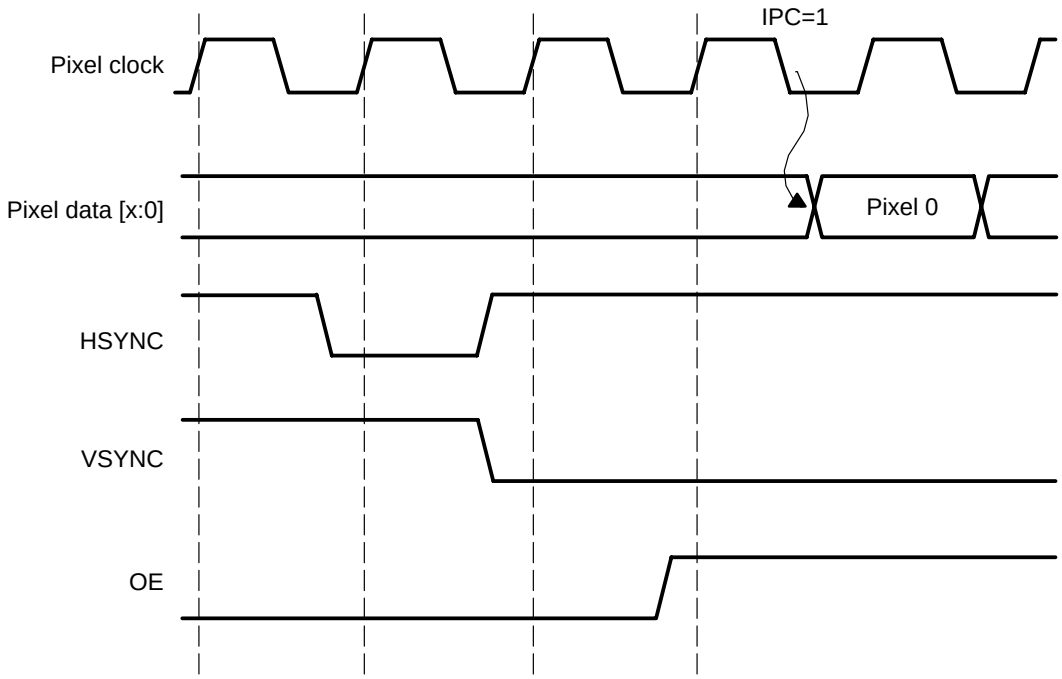
Figure 8–35. ON_OFF = 0, IPC = 1 in TFT Mode



- IPC = 1 means that pixel data is driven onto the LCD data lines on the falling edge of the pixel clock.
- ON_OFF = 0 means that HSYNC and VSYNC signals are driven on opposite edges of the pixel clock from pixel data (rising edge).

Figure 8–36 shows the timing when ON_OFF = 1, RF = 0, and IPC = 1.

Figure 8–36. ON_OFF = 1, RF = 0, and IPC = 1



- When ON_OFF = 1, HSYNC and VSYNC signals are driven according to the RF bit.
- When RF = 0, HSYNC and VSYNC signals are driven on the falling edge of the pixel clock.
- When IPC = 1, pixel data is driven on the falling edge of the pixel clock.

8.3.4.9 HSYNC/VSYNC ON or OFF (ON_OFF)

This bit enables/disables the possibility to make HSYNC and VSYNC programmable.

- When ON_OFF = 1, HSYNC and VSYNC are driven according to the RF bit.
- When ON_OFF = 0, HSYNC and VSYNC are driven on opposite edges of the pixel clock from pixel data.

8.3.5 LCD Controller Status Register (LCDSTATUS)

The LCD controller status register (LCDSTATUS) contains seven bits that detect different events. Each of these hardware-detected events signals an interrupt request to the interrupt controller.

Table 8–15. LCD Status Register (LCDSTATUS) Bit Descriptions

Bit	Name	Description	Reset
31-7	RESERVED	Reserved	x
6	LP	Loaded palette (read-only/clear-only) LP = 0 as long as the palette is not loaded. LP is set to 1 when the palette is loaded.	0
5	FUF	FIFO underflow status (read-only) FUF = 0 as long as FIFO has not underrun. FUF is set to 1 when LCD dither logic is not supplying data to FIFO at a sufficient rate, FIFO has completely emptied, and data pin driver logic has attempted to take added data from FIFO.	0
4	LINE_INT	Line interrupt (read-only/clear-only) LINE_INT = 0 as long as the display has not reached the programmed line yet, or if this interrupt has been cleared. LINE_INT is set to 1 when the display reaches the user-programmed line number and generates the interrupt.	0
3	ABC	ac-bias count status (read/clear only) ABC = 0 as long as ac-bias transition counter has not decremented to 0 (see Section 8.3.4.3, *ac-Bias Line Transitions Per Interrupt (ACBI)*. ABC is set to 1 when ac-bias transition counter has decremented to zero, indicating that the ac-bias output line has transitioned the number of times specified by the ACBI control bit-field. The counter is reloaded with value in ACBI but is disabled until ABC is cleared.	0
2	SYNC_ LOST	Synchronization lost (read-only) When SYNC_LOST = 0, no frame synchronization error occurred. When SYNC_LOST = 1, frame synchronization lost occurred.	0
1	VS	VSYNC interrupt (read/clear only) VS = 0 as long as VSYNC interrupt is not generated. VS is set to 1 when the VSYNC interrupt occurs at the end of frame.	0
0	DONE	Frame done (read-only) Done is set to 0 as long as LCD is enabled. Done is set to 1 when LCD is disabled and the active frame is just completed.	0

Note: Read-only/clear-only bits are cleared by writing 0 to them.

8.3.5.1 Frame Done (DONE) (Read-Only)

When the LCD is disabled by clearing the LCD enable bit (LCDEN = 0) in the LCDCONTROL register, the LCD allows the current frame to complete before it is disabled. After the last set of pixels is clocked out onto the LCD data pins by the pixel clock, the LCD is disabled and DONE is set.

- ❏ DONE = 1 when the frame is complete.
- ❏ DONE = 0 as long as the frame is not complete.

The frame done (DONE) bit is a read-only bit signaling that the frame is complete. It is cleared when LCDEN bit is set to 1 (turned on).

8.3.5.2 VSYNC Interrupt (VS) (Read/Clear-Only)

VSYNC interrupt occurs when the LCD reaches the end of the frame. This interrupt is shared with other LCD interrupts and is output on the LCD_NIRQ interrupt output line (see Figure 8–37, *Line Interrupt Path*). You can unmask the VSYNC interrupt by writing 1 to the VSYNC_MASK bit in the LCD control register (see Section 8.3.1, *LCD Control Register*).

To clear the VSYNC bit in the status register, you must write 0 to it.

- ❏ VSYNC = 1 signals that the end of frame occurred and generated the VSYNC interrupt.
- ❏ VSYNC = 0 if no VSYNC interrupt is generated.

8.3.5.3 Frame Synchronization Lost (SYNC_LOST) (Read-Only)

The frame synchronization lost (SYNC_LOST) bit is set if the LCD controller detects a frame synchronization error. A frame synchronization error happens when the LCD attempts to read what it believes to be the first word of the video buffer but cannot be recognized as such. This bit is cleared by disabling the LCD controller (LCDEN bit =0). This also resets the input FIFO in the DMA controller.

- ❏ SYNC_LOST = 1 when a frame synchronization lost occurred
- ❏ SYNC_LOST = 0 as long as no frame synchronization error occurs

8.3.5.4 ac-Bias Count Status (ABC) (Read/Clear-Only)

The ac-bias count status (ABC) bit is set each time the ac-bias line transitions a particular number of times as specified by the ac-bias line transitions per interrupt (ACBI) field in LCDTIMING2. If ACBI is programmed with a non-zero value, a counter is loaded with the value in ACBI and is decremented each time the ac-bias line reverses state. When the counter reaches zero, the ABC bit is set signaling an interrupt request to the interrupt controller. The counter reloads using the value in ACBI, but does not start to decrement again until ABC is cleared by writing 0 to the LCD status register.

- ❏ ABC = 1 when the ac–bias transition counter ACBI decrements to 0
- ❏ ABC = 0 as long as ACBI has not decremented to 0

8.3.5.5 Line Interrupt (LINE_INT) (Read/Clear-Only)

The LCD can be programmed to generate a line interrupt when the LCD reaches a certain line (n) in the display. This interrupt sets the LINE_INT bit in the status register. *This dedicated line interrupt can be connected to the hardware-synchronized channel of the DMA. It can be used to prevent tearing effect if double buffering is not implemented.* See Section 8.2.3.2, *Tearing Effect,* for information on the use of this feature.

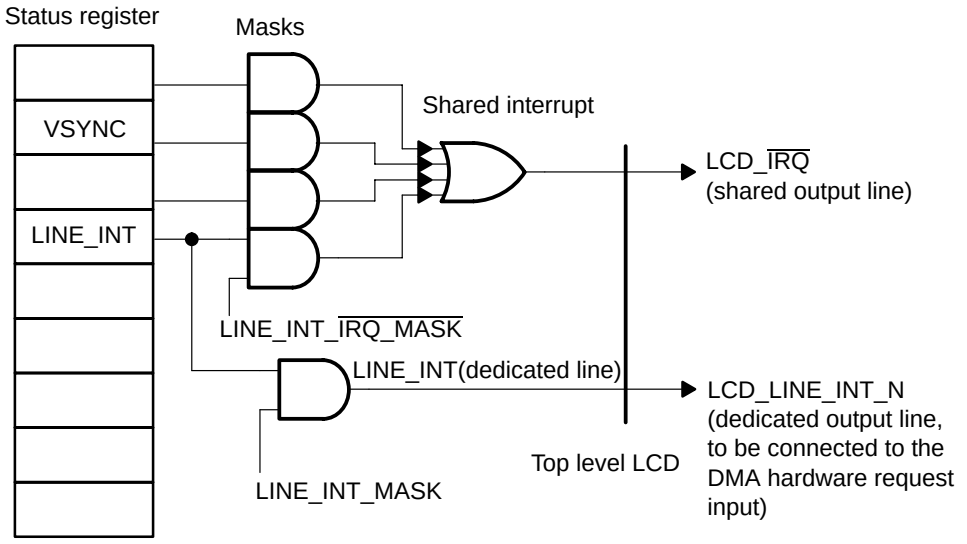
It is possible to connect the LINE_INT interrupt to both the LCD_NIRQ interrupt output line and to a dedicated LCD_LINE_INT_n output line.

The LINE_INT_MASK bit masks the dedicated line interrupt. You can unmask it by writing 1 to the control register. LINE_INT is a status bit that can be cleared by writing 0 to it.

The LINE_INT_NIRQ_MASK bit in the control register masks the shared interrupt.

See Figure 8–37 for details.

Figure 8–37. Line Interrupt Path



- LINE_INT is set to 1 when the display reaches the user-programmed line number and generates the interrupt.
- LINE_INT = 0 as long as the programmed line is not reached.

The LINE_INT bit is cleared if the TIPB writes 0 to it or at the end of the programmed line according to LINE_INT_CLR_SEL bit in the control register.

8.3.5.6 FIFO Underflow Status (FUF) (Read-Only)

The FIFO underflow status (FUF) bit is set when the input FIFO is completely empty and the LCD data pins driver logic attempts to fetch data from the FIFO. This bit is cleared by disabling the LCD controller (LCDEN =_0). This also resets the input FIFO in the DMA controller.

- FUF = 1 when the dithering logic is not supplying data to the FIFO at a sufficient rate.
- FUF = 0 as long as the FIFO has not underrun.

8.3.5.7 Loaded Palette (LP) (Read-Only/Clear-Only)

The loaded palette (LP) bit is a read-only bit that is set after the LCD finishes loading the palette into memory.

- LP = 1 when the palette is loaded.
- LP = 0 as long as the palette is not loaded.

In data-only (PLM = 10) and palette-plus-data (PLM = 00) modes, write 0 to clear the interrupt. However, in the palette only (PLM = 01) mode, the LCD must be turned off in order to reset/clear the interrupt. *In this particular mode*

*be sure not to turn off the LCD before getting the loading interrupt. S*ee Section 8.3.1.14, *Palette Loading (PLM).*

8.3.6 LCD Subpanel Display Register (LCDSUBPANEL)

The LCD subpanel display register (LCDSUBPANEL) enables displaying only the first or last X lines of the panel and sending fixed content for the other lines.

Table 8–16. LCD Subpanel (LCDSUBPANEL) Bit Descriptions

Bit	Name	Description	Reset
31	SPEN	Subpanel enable SPEN = 0: Subpanel function is disabled. SPEN = 1: Subpanel function is enabled.	0
30	RESERVED	Reserved	x
29	HOLS	High or low signal The field indicates the position of a subpanel compared to the LPPT value. HOLS = 0: The image from the frame buffer is displayed below the LPPT value. HOLS = 1: The image from the frame buffer is displayed above the LPPT value.	0
28-26	RESERVED	Reserved	x
25:16	LPPT	Line per panel threshold LPPT is a threshold value delimiting the subpanel and the DPD parts of the screen**.** It ranges from 1 to 1024 and must be programmed to value required minus one (0-1023).	0x000
15:0	DPD	Default pixel data DPD defines the default value of the pixel data sent to the panel for the lines until LPPT threshold is reached or after passing the LPPT, depending on HOLS mode.	0x0000

DPD, LPPT, HOLS, and SPEN bit fields and bits are not considered if SPEN = 0.

8.3.6.1 Default Pixel Data (DPD)

DPD defines a default value, which is sent to the display in either the top or bottom region of the screen delimited by the LPPT threshold. When displaying DPD value, there is no DMA activity.

8.3.6.2 Line-Per-Panel Threshold (LPPT)

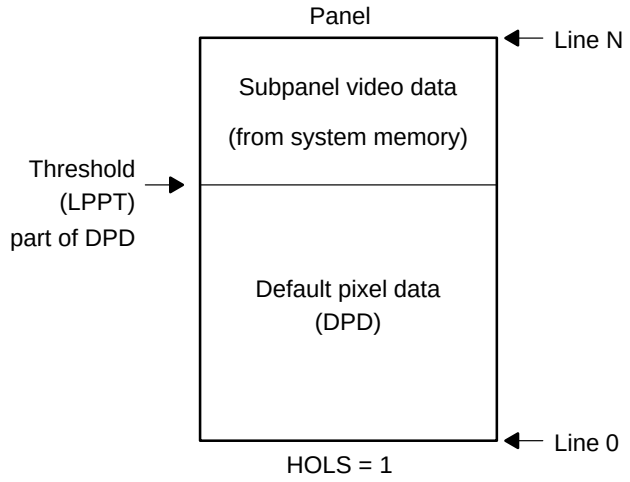
The line-per-panel threshold bit-field delimits the screen portion filled with data fetched from the frame buffer (the subpanel) and the rest of the screen filled with default pixel data (DPD). The LPPT line number points to a line filled with a DPD value when HOLS = 1 or to one filled with video data when HOLS = 0 (see Figure 8–38 and Figure 8–39).

8.3.6.3 High Or Low Signal (HOLS)

The HOLS bit indicates the position of the subpanel compared to the LPPT value.

When HOLS = 1, the image from the system memory is displayed above the threshold value. The threshold value is the line number where the DPD value begins to be displayed. The rest of the screen is filled with the DPD value.

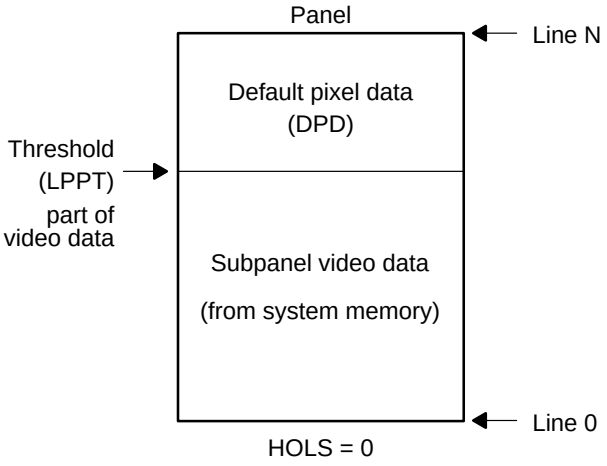
Figure 8–38. Subpanel Display: SPEN = 1, HOLS = 1



When HOLS = 0, the beginning of the screen is filled with the DPD value until the LPPT is excluded. From the LPPT line number, the rest of the screen1

(below LPPT) displays the image from the system memory.

Figure 8–39. Subpanel Display: SPEN = 1, HOLS = 0



Note:

The bottom of the panel is line 0, and the top line of the panel is line N (where N is the number of lines-per-panel).

For example, if you want to display four lines of video data at the bottom of the panel, the correct settings are HOLS = 0 and LPPT = 3. Here, the amount of video data to be transferred from the DMA_LCD channel is only four lines.

Note:

If the LPPT is above the number of LPPs, then:

When HOLS = 1: Panel with default data (whole panel is filled with DPD value).
When HOLS = 0: Normal panel (whole panel is filled with video data from the frame buffer).

8.3.6.4 Subpanel Enable (SPEN)

This bit enables or disables the subpanel mode.

When SPEN = 0, the subpanel mode is disabled.

When SPEN = 1, the subpanel mode is enabled.

8.3.7 Line Interrupt Register (LCDLINEINT)

Table 8–17. Line Interrupt Register (LCDLINE INT) Bit Descriptions

Bit	Name	Description	Reset
31:10	RESERVED	Reserved	x
9:0	LINE_INT_ NUMBER	Line number at which line interrupt occurs. Programmable from line 0 up to line 1023.	0x000

The line interrupt register (LCDLINEINT) enables programming of the line number in bits (0-9) where the line interrupt is to be generated.

8.3.8 Display Status Register (LCDDISPLAYSTATUS)

Table 8–18. Line Interrupt Register (LCDDISPLAYSTATUS) Bit Descriptions

Bit	Name	Description	Reset
31:10	RESERVED	Reserved	x
9:0	CURRENT_ LINE_ NUMBER	Line number being displayed. Because the number of lines can be programmed from 1 to 1024, the current line number varies between 0 and 1023.	0x3FF

The display status register (LCDDISPLAYSTATUS) contains the line number currently being displayed.

Chapter 9

UART Modem/IrDA

This chapter provides programmers with a functional presentation of the universal asynchronous receiver/transmitter (UART) infrared data association (IrDA) module. It includes a register description and a module configuration example. It also shows the basic UART module pins.

Topic **Page**

9.1 Signals and Block Diagram

Figure 9–1 illustrates the UART modem/IrDA signals, and Figure 9–2 presents the functional block diagram.

Figure 9–1. UART IrDA Signals

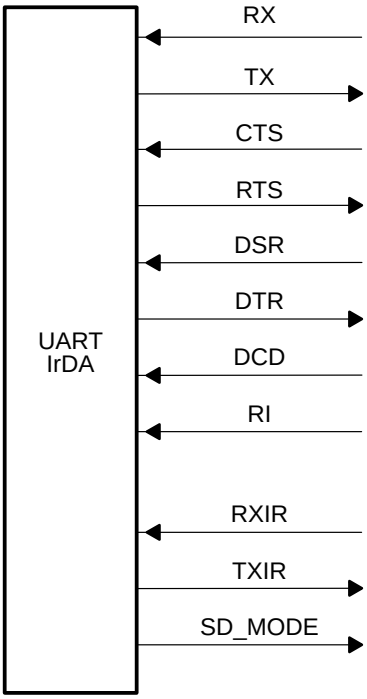
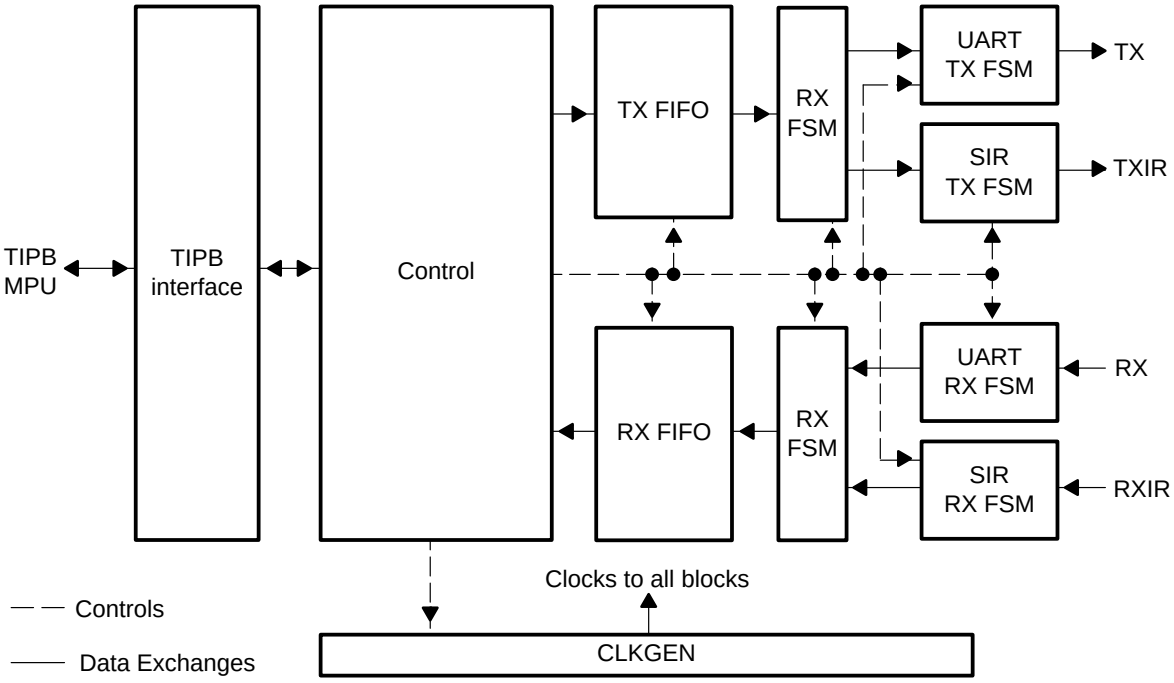


Figure 9–2. Functional Block Diagram



9.2 Main Features

- ❏ Selectable modem/IrDA modes
- ❏ Dual 64-entry FIFOs for received and transmitted data
- ❏ Programmable and selectable transmit and receive FIFO trigger levels for DMA and interrupt generation
- ❏ Programmable sleep mode
- ❏ Complete status reporting capabilities in both normal and sleep modes
- ❏ Frequency prescaler values from 0 to 16383 to generate the appropriate baud rates
- ❏ Single 48-MHz clock reference for baud setting
- ❏ Two DMA requests and one interrupt request to the system

9.2.1 Modem Functions

- ❏ Baud-rate from 300 bits/s up to 3.6864M bits/s
- ❏ Autobaud between 1200 bits/s and 115.2K bits/s
- ❏ Software/hardware flow control
  - ■ Programmable Xon/Xoff characters
  - ■ Programmable auto-$\overline{\text{RTS}}$ and auto-$\overline{\text{CTS}}$
- ❏ Programmable serial interface characteristics
  - ■ 5-, 6-, 7-, or 8-bit characters
  - ■ Even, odd, mark (always = 1), space (always = 0) or no parity (non parity bit frame) bit generation and detection
  - ■ 1, 1.5, or 2 stop-bit generation
- ❏ False start bit detection
- ❏ Line break generation and detection
- ❏ Fully prioritized interrupt system controls
- ❏ Internal test and loopback capabilities
- ❏ Modem control functions ($\overline{\text{CTS}}$, $\overline{\text{RTS}}$, $\overline{\text{DSR}}$, $\overline{\text{DTR}}$, $\overline{\text{RI}}$ and $\overline{\text{DCD}}$)

9.2.2 IrDA Functions

- ❏ Slow infrared (SIR 115.2K baud), medium infrared (MIR 0.576M baud) and fast infrared (FIR 4.0M baud) operations. Very fast infrared (VFIR) is not supported.
- ❏ Framing error, cyclic redundancy check (CRC) error, illegal symbol (FIR), abort pattern (SIR, MIR) detection
- ❏ 8-entry status FIFO (with selectable trigger levels) available to monitor frame length and frame errors

In Table 9–1 the module I/O description is at the module level.

Table 9–1. I/O Description

Signal	I/O	Description	Value At Reset
		UART/MODEM Signals	
RX	I	Serial data input	Unknown
TX	O	Serial data output	1
$\overline{\text{CTS}}$	I	Clear to send Active-low modem status signal. Reading bit 4 of the modem status register checks the condition of $\overline{\text{CTS}}$. Reading bit 0 of that register checks a change of state of $\overline{\text{CTS}}$ since the last read of the modem status register. $\overline{\text{CTS}}$ is used in auto-$\overline{\text{CTS}}$ mode to control the transmitter.	Unknown
$\overline{\text{RTS}}$	O	Request to send When active (low), the module is ready to receive data. Setting modem control register bit 1 activates $\overline{\text{RTS}}$. It becomes inactive as a result of a module reset, loopback mode, or by clearing the MCR[1]. In auto-$\overline{\text{RTS}}$ mode, it becomes inactive as a result of the receiver threshold logic.	1
$\overline{\text{DSR}}$	I	Data set ready Active-low modem status signal. Reading bit 5 of the modem status register checks the condition of $\overline{\text{DSR}}$. Reading bit 1 of that register checks a change of state of $\overline{\text{DSR}}$ since the last read of the modem status register.	Unknown
$\overline{\text{DTR}}$	O	Data terminal ready Active-low modem control signal. Reading bit 0 of the modem control register checks the condition of $\overline{\text{DTR}}$.	1
$\overline{\text{DCD}}$	I	Data carrier detect Active-low modem status signal. The condition of $\overline{\text{DCD}}$ is checked by reading bit 7 of the modem status register, and any change in its state can be detected by reading bit 3 of that register.	Unknown
$\overline{\text{RI}}$	I	Reading indicator Active-low modem status signal. The condition of $\overline{\text{RI}}$ is checked by reading bit 6 of the modem status register, and any change in its state is detected by reading bit 2 of that register.	Unknown
		IrDA Signals	
RXIR	I	Serial data input	Unknown
TXIR	O	Serial data output	0
SD	O	Signal used to configure transceivers	1

9.3 UART Modem/IrDA Registers

Table 9–2 lists the UART modem/IrDA registers. Each register is selected using a combination of address and some LCR register bit(s) setting as shown in Table 9–3. Table 9–4 through Table 9–48 describe the register bits.

The local host can access the following registers at address = module base address + address offset. The module base address is the module start address. Register address offsets depend on the module address alignment at the system top level. The address offsets (0x13 x S) and (0x18 x S to 0x31 x S (inclusive)) are reserved and must be read as 0x00 at all times:

- S = 1 for 8-bit aligned addresses
- 2 for 16-bit aligned addresses
- 4 for 32-bit aligned addresses

All UART registers are 8-bit, and their start addresses are:

- UART1 (or UART_MODEM): FFFB 0000
- UART2 (or UART_MODEM_IRDA): FFFB 0800

Table 9–2. UART Modem/IrDA Registers

Name	Description	R/W	Offset
RHR	Receive holding	R	
THR	Transmit holding	W	
FCR	FIFO control	W	
SCR	Supplementary control	R/W	
LCR	Line control	R/W	
LSR—MM	Line status—modem mode	R	
LSR—IR	Line status—IrDA mode	R	
SSR	Supplementary status	R	
MCR	Modem control	R/W	
MSR	Modem status	R	
IER—MM	Interrupt enable—modem mode	R/W	
IER—IrDA	Interrupt enable—IrDA mode	R/W	
IIR—MM	Interrupt Identification—modem mode	R	
IIR—IrDA	Interrupt Identification—IrDA mode	R	
EFR	Enhanced features	R/W	
XON1/ADDR1	XON1 address 1	R/W	
XON2/ADDR2	XON2 address 2	R/W	
XOFF1	XOFF1	R/W	
XOFF2	XOFF2	R/W	

Table 9–2. UART Modem/IrDA Registers (Continued)

Name	Description	R/W	Offset
SPR	Scratchpad	R/W	
DLL	Divisor latches low	R/W	
DLH	Divisor latches high	R/W	
TCR	Transmission control	R/W	
TLR	Trigger level	R/W	
MDR1	Mode definition 1	R/W	
MDR2	Mode definition 2	R/W	
UASR	UART autobauding status	R/W	
TCFLL	Transmit frame length low	W	
TCFLH	Transmit frame length high	R	
RXFLL	Received frame length low	W	
RXFLH	Received frame length high	R/W	
SFLSR	Status FIFO line status	R	
RESUME	Resume	R	
SFREGL	Status FIFO low	R	
SFREGH	Status FIFO high	R	
BLR	BOF control	R/W	
EBLR	BOF length	R/W	
ACREG	Auxiliary control	R/W	
MVR	Module version	R	
SYSC	System configuration	R/W	
SYSS	System status	R	
WER	Wake-up enable	R/W	

Table 9–3. UART Modem/IrDA Register Combinations

Address Offset	Registers					
	LCR[7] = 0		LCR[7] = 1 and LCR[7:0] is not 0xBF		LCR[7:0] = 0xBF	
	READ	WRITE	READ	WRITE	READ	WRITE
0x00 x S	RHR	THR	DLL	DLL	DLL	DLL
0x01 x S	IER §	IER §	DLH	DLH	DLH	DLH
0x02 x S	IIR	FCR‡	IIR	FCR‡	EFR	EFR
0x03 x S	LCR	LCR	LCR	LCR	LCR	LCR
0x04 x S	MCR‡	MCR‡	MCR‡	MCR‡	XON1/ADDR1	XON1/ADDR1
0x05 x S	LSR	–	LSR	–	XON2/ADDR2	XON2/ADDR2
0x06 x S	MSR/TCR†	TCR†	MSR/TCR†	TCR†	XOFF1/TCR†	XOFF1/TCR†
0x07 x S	SPR/TLR†	SPR/TLR†	SPR/TLR†	SPR/TLR†	XOFF2/TLR†	XOFF2/TLR†
0x08 x S	MDR1	MDR1	MDR1	MDR1	MDR1	MDR1
0x09 x S	MDR2	MDR2	MDR2	MDR2	MDR2	MDR2
0x0A x S	SFLSR	TXFLL	SFLSR	TXFLL	SFLSR	TXFLL
0x0B x S	RESUME	TXFLH	RESUME	TXFLH	RESUME	TXFLH
0x0C x S	SFREGL	RXFLL	SFREGL	RXFLL	SFREGL	RXFLL
0x0D x S	SFREGH	RXFLH	SFREGH	RXFLH	SFREGH	RXFLH
0x0E x S	BLR	BLR	UASR	–	UASR	–
0x0F x S	ACREG	ACREG	–	–	–	–
0x10 x S	SCR	SCR	SCR	SCR	SCR	SCR
0x11 x S	SSR	–	SSR	–	SSR	–
0x12 x S	EBLR	EBLR	–	–	–	–
0x14 x S	MVR	–	MVR	–	MVR	–
0x15 x S	SYSC	SYSC	SYSC	SYSC	SYSC	SYSC
0x16 x S	SYSS	SYSS	SYSS	SYSS	SYSS	SYSS
0x17 x S	WER	WER	WER	WER	WER	WER

† Transmission control register (TCR) and trigger level register (TLR) are accessible only when EFR[4] = 1 and MCR[6] = 1.
‡ MCR[7:5] and FCR[5:4] can only be written when EFR[4] = 1.
§ In UART modes, IER[7:4] can only be written when EFR[4] = 1. In IrDA modes, EFR[4] has no effect on access to IER[7:4].

Table 9–4. Receive Holding Register (RHR)

Bit	Name	Function	R/W	Value At Reset
7:0	RHR	Receive holding register	R	Unknown

If an overflow occurs the data in the receive holding register is not overwritten.

The receiver section consists of the RHR and the receiver shift register. The RHR is actually a 64-byte FIFO. The receiver shift register receives serial data from RX input. The data is converted to parallel data and moved to the RHR. If the FIFO is disabled, location 0 of the FIFO is used to store the single data character.

Table 9–5. Transmit Holding Register (THR)

Bit	Name	Function	R/W	Value At Reset
7:0	THR	Transmit holding register	W	Unknown

The transmitter section consists of the transmit holding register (THR) and the transmit shift register. The transmit holding register is actually a 64-byte FIFO. The LH writes data to the THR. The data is placed into the transmit shift register where it is shifted out serially on the TX output. If the FIFO is disabled, location 0 of the FIFO is used to store the data.

Table 9–6. FIFO Control Register (FCR)

Bit	Name	Function	R/W	Value At Reset
7:6	RX_FIFO_TRIG	Sets the trigger level for the RX FIFO: If SCR[7] = 0 and TLR[7:4] = 0000: 00: 8 characters 01: 16 characters 10: 56 characters 11: 60 characters If SCR[7] = 0 and TLR[7:4] non-0, RX_FIFO_TRIG is not considered. If SCR[7] = 1, RX_FIFO_TRIG is 2 LSBs of the trigger level (1–63 on 6 bits) with the granularity 1.	W	1
5:4	TX_FIFO_TRIG	Sets the trigger level for the TX FIFO: If SCR[6] = 0 and TLR[3:0] = 0000: 00: 8 spaces 01: 16 spaces 10: 32 spaces 11: 56 spaces If SCR[6] = 0 and TLR[3:0] non-0, TX_FIFO_TRIG is not considered. If SCR[6] = 1, TX_FIFO_TRIG is 2 LSBs of the trigger level (1–63 on 6 bits) with the granularity 1.	W	00
3	DMA_MODE	0: DMA_MODE 0 (No DMA) 1: DMA_MODE 1 (UART_nDMA_REQ[0] in TX, UART_nDMA_REQ[1] in RX) This register is considered if SCR[0] = 0.	W	0
2	TX_FIFO_CLEAR	0: No change 1: Clears the transmit FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.	W	0
1	RX_FIFO_CLEAR	0: No change 1: Clears the receive FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.	W	0
0	FIFO_EN	0: Disables the transmit and receive FIFOs 1: Enables the transmit and receive FIFOs	W	0

Notes: 1) Bits 4 and 5 can only be written to when EFR[4] = 1
2) Bits 0 to 3 can be changed only when the baud clock is not running (DLL and DLH set to 0)
3) See for FCR[5:4] setting restriction when SCR[6] = 1
4) See for FCR[7:6] setting restriction when SCR[7] = 1

Offset Address (hex): 0x02 x S and LCR not 0xBF (and EFR[4] = 1 for FCR[5:4]) and write.

Table 9–7. Supplementary Control Register (SCR)

Bit	Name	Function	R/W	Value At Reset
7	RX_TRIG_GRANU1	0: Disables the granularity of 1 for trigger RX level 1: Enables the granularity of 1 for trigger RX level	R/W	0
6	TX_TRIG_GRANU1	0: Disables the granularity of 1 for trigger TX level 1: Enables the granularity of 1 for trigger TX level	R/W	0
5	DSR_IT	0: Disables $\overline{\text{DSR}}$ interrupt 1: Enables $\overline{\text{DSR}}$ interrupt	R/W	0
4	RX_CTS_DSR_ WAKE_UP_ENABLE	0: Disables the wake-up interrupt and clears SSR[1]. 1: Waits for a falling edge of pins RX, $\overline{\text{CTS}}$ or $\overline{\text{DSR}}$ to generate an interrupt	R/W	0
3	TX_EMPTY_CTL_IT	0: Normal mode for THR interrupt (See UART mode interrupts table.) 1: The THR interrupt is generated when TX FIFO and TX shift register are empty.	R/W	0
2:1	DMA_MODE_2	Used to specify the DMA mode valid if SCR[0] = 1 00: DMA mode 0 (no DMA) 01: DMA mode 1 (UART_nDMA_REQ[0] in TX, UART_nDMA_REQ[1] in RX) 10: DMA mode 2 (UART_nDMA_REQ[0] in RX) 11: DMA mode 3 (UART_nDMA_REQ[0] in TX)	R/W	00
0	DMA_MODE_CTL	0: The DMA_MODE is set with FCR[3]. 1: The DMA_MODE is set with SCR[2:1].	R/W	0

Bit 4 of the supplementary control register enables the wake-up interrupt, but this interrupt is not mapped into the IIR register. Therefore, when an interrupt occurs and there is no interrupt pending in the IIR register, the SSR[1] bit must be checked. To clear the wake-up interrupt, bit SCR[4] must be reset to 0.

Table 9–8. Line Control Register (LCR)

Bit	Name	Function	R/W	Value At Reset
7	DIV_EN	0: Normal operating condition 1: Divisor latch enable. Allows access to DLL, DLH, and other registers (refer to the registers' mapping)	R/W	0
6	BREAK_EN	Break control bit 0: Normal operating condition 1: Forces the transmitter output to go low to alert the communication terminal	R/W	0

Table 9–8. Line Control Register (LCR) (Continued)

Bit	Name	Function	R/W	Value At Reset
5	PARITY_TYPE2		R/W	0
4	PARITY_TYPE1	0: Odd parity is generated (if LCR[3] = 1). 1: Even parity is generated (if LCR[3] = 1).	R/W	0
3	PARITY_EN	0: No parity 1: A parity bit is generated during transmission and the receiver checks for received parity.	R/W	0
2	NB_STOP	Specifies the number of stop bits: 0: 1 stop bits (word length = 5, 6, 7, 8) 1: 1.5 stop bits (word length = 5) 1–2 stop bits (word length = 6, 7, 8)	R/W	0
1:0	CHAR_LENGTH	Specifies the word length to be transmitted or received 00: 5 bits 01: 6 bits 10: 7 bits 11: 8 bits	R/W	00

LCR[6:0] of the line control register defines parameters of the transmission and reception.

Table 9–9. Line Status Register—Modem Mode (LSR—MM)

Bit	Name	Function	R/W	Value At Reset
7	RX_FIFO_STS	0: Normal operation 1: At least one parity error, framing error, or break indication in the receiver FIFO. Bit 7 is cleared when no more errors are present in the FIFO.	R	0
6	TX_SR_E	0: Transmitter hold and shift registers are not empty. 1: Transmitter hold and shift registers are empty.	R	1
5	TX_FIFO_E	0: Transmit hold register is not empty 1: Transmit hold register is empty. The processor can now load up to 64 bytes of data into the THR if the TX FIFO is enabled.	R	1
4	RX_BI	0: No break condition 1: A break was detected while the data being read from the RX FIFO was being received (that is, RX input was low for one character time frame).	R	0

Table 9–9. Line Status Register—Modem Mode (LSR—MM) (Continued)

Bit	Name	Function	R/W	Value At Reset
3	RX_FE	0: No framing error in data being read from RX FIFO 1: Framing error occurred in data being read from RX FIFO (received data did not have a valid stop bit)	R	0
2	RX_PE	0: No parity error in data being read from RX FIFO 1: Parity error in data being read from RX FIFO	R	0
1	RX_OE	0: No overrun error 1: Overrun error has occurred. Set when the character held in the receive shift register is not transferred to the RX FIFO. This case occurs only when receive FIFO is full.	R	0
0	RX_FIFO_E	0: No data in the receive FIFO 1: At least one data character in the RX_FIFO	R	0

Offset Address (hex): 0x05 x S and LCR is not 0xBF and read

When the LSR is read, LSR[4:2] reflects the error bits [BI, FE, PE] of the character at the top of the RX FIFO (next character to be read). Therefore, reading the LSR, and then reading the RHR, identifies errors in a character.

Reading RHR updates [BI, FE, PE] (see Section 9.5.2.1, *Modem Mode Interrupt*s).

LSR [7] is set when there is an error anywhere in the RX FIFO and is cleared only when there are no errors remaining in the FIFO.

Reading the LSR does not cause an increment of the RX FIFO read pointer. The RX FIFO read pointer is incremented by reading the RHR.

Reading LSR clears [OE] if set (see Table 9–49, *Modem Mode Interrupts*).

Table 9–10. Line Status Register—IR Mode(LSR—IR)

Bit	Name	Function	R/W	Value At Reset
7	THR_EMPTY	0: Transmit holding register is not empty. 1: Transmit holding register is empty. The processor can now load up to 64 bytes of data into the THR if the TX FIFO is enabled.	R	1
6	STS_FIFO_FULL	0: Status FIFO not full 1: Status FIFO full	R	0

Table 9–10. Line Status Register—IR Mode(LSR—IR) (Continued)

Bit	Name	Function	R/W	Value At Reset
5	RX_LAST_BYTE	0: The RX FIFO does not contain the last byte of the frame to be read. 1: The RX FIFO contains the last byte of the frame to be read. This bit is set only when the last byte of a frame is available to be read. It is used to determine the frame boundary. It is cleared on a single read of the LSR register.	R	0
4	FRAME_TOO_LONG	0: No frame-too-long error in frame 1: Frame-too-long error in the frame at the top of the STATUS FIFO, [next character to be read]. This bit is set to 1 when a frame exceeding the maximum length (set by RXFLH and RXFLL registers) has been received. When this error is detected, current frame reception is terminated. Reception is stopped until the next START flag is detected.	R	0
3	ABORT	0: No abort pattern error in frame 1: Abort pattern is received. SIR & MIR: Abort pattern FIR: Illegal symbol 0000	R	0
2	CRC	0: No CRC error in frame 1: CRC error in the frame at the top of the STATUS FIFO (next character to be read)	R	0
1	STS_FIFO_E	0: Status FIFO not empty 1: Status FIFO empty	R	1
0	RX_FIFO_E	0: At least one data character in the RX_FIFO 1: No data in the receive FIFO	R	1

When the LSR is read, LSR[4:2] reflects the error bits [FL, CRC, ABORT] of the frame at the top of the STATUS FIFO (next frame status to be read). See Section 9.5.2.2, *IrDA Mode Interrupts*.

Legacy software design (1510 and equivalent silicon) must take into account the above changes.

Table 9–11. Supplementary Status Register (SSR)

Bit	Name	Function	R/W	Value At Reset
7:2	–	Reserved	R	000000
1	RX_CTS_DSR_WAKE_UP_STS	0: No falling edge event on RX, $\overline{\text{CTS}}$ and $\overline{\text{DSR}}$	R	0
		1: A falling edge occurred on RX, $\overline{\text{CTS}}$ or $\overline{\text{DSR}}$		
0	TX_FIFO_FULL	0: TX FIFO not full	R	0
		1: TX FIFO full		

Offset Address (hex): 0x11 x S and read

Bit 1 in the supplementary status register is reset only when SCR[4] is reset to 0.

Table 9–12. Modem Control Register (MCR)

Bit	Name	Function	R/W	Value At Reset
7	–	Reserved	R	0
6	TCR_TLR	0: No action	R/W	0
		1: Enables access to the TCR and TLR registers		
5	XON_EN	0: Disable the XON-any function	R/W	0
		1: Enable the XON-any function		
4	LOOPBACK_EN	0: Normal operating mode	R/W	0
		1: Enable local loopback mode (internal)		
		In this mode, the MCR[3:0] signals are looped back into MSR[7:4]. The transmit output is looped back to the receive input internally.		
3	CD_STS_CH	0: In loopback, forces $\overline{\text{DCD}}$ input high and IRQ outputs to inactive state	R/W	0
		1: In loopback, forces $\overline{\text{DCD}}$ input low and IRQ outputs to inactive state		
2	RI_STS_CH	0: In loopback, forces $\overline{\text{RI}}$ input high	R/W	0
		1: In loopback, forces $\overline{\text{RI}}$ input low		
1	RTS	0: Force $\overline{\text{RTS}}$ output to inactive (high)	R/W	0
		1: Force $\overline{\text{RTS}}$ output to active (low)		
		In loopback, controls MSR[4]		
		If auto-RTS is enabled, the $\overline{\text{RTS}}$ output is controlled by hardware flow control.		
0	DTR	0: Force $\overline{\text{DTR}}$ output to inactive (high)	R/W	0
		1: Force $\overline{\text{DTR}}$ output to active (low)		

Offset Address (hex): 0x04 x S and LCR is not 0xBF (and EFR[4] = 1 for MCR[7:5])

MCR[3:0] controls the interface with the modem, data set, or peripheral device that is emulating the modem.

Bits 5 and 6 can be written only when EFR[4] = 1.

Table 9–13. Modem Status Register (MSR)

Bit	Name	Function	R/W	Value At Reset
7	NCD_STS	This bit is the complement of the $\overline{DCD}$ input. In loopback mode it is equivalent to MCR[3].	R	Unknown
6	NRI_STS	This bit is the complement of the $\overline{RI}$ input. In loopback mode it is equivalent to MCR[2].	R	Unknown
5	NDSR_STS	This bit is the complement of the $\overline{DSR}$ input. In loopback mode, it is equivalent to MCR[0].	R	Unknown
4	NCTS_STS	This bit is the complement of the $\overline{CTS}$ input. In loopback mode it is equivalent to MCR[1].	R	Unknown
3	DCD_STS	Indicates that $\overline{DCD}$ input (or MCR[3] in loopback) has changed. Cleared on a read.	R	0
2	RI_STS	Indicates that $\overline{RI}$ input (or MCR[2] in loopback) has changed state from low to high. Cleared on a read.	R	0
1	DSR_STS	1: Indicates that $\overline{DSR}$ input (or MCR[0] in loopback) has changed state. Cleared on a read.	R	0
0	CTS_STS	1: Indicates that $\overline{CTS}$ input (or MCR[1] in loopback) has changed state. Cleared on a read.	R	0

Offset Address (hex): 0x06 x S and LCR is not 0xBF and (EFR[4] = 0 or MCR[6] = 0) and read.

The modem status register provides information about the current state of the control lines from the modem, data set, or peripheral device to the LH. It also indicates when a control input from the modem changes state.

Table 9–14. Interrupt Enable Register—Modem Mode (IER—MM)

Bit	Name	Function	R/W	Value At Reset
7	CTS_IT	0: Disables the $\overline{CTS}$ interrupt 1: Enables the $\overline{CTS}$ interrupt	R/W	0
6	RTS_IT	0: Disables the $\overline{RTS}$ interrupt 1: Enables the $\overline{RTS}$ interrupt	R/W	0
5	XOFF_IT	0: Disables the XOFF interrupt 1: Enables the XOFF interrupt	R/W	0
4	SLEEP_MODE	0: Disables sleep mode 1: Enables sleep mode (stop baud rate clock when the module is inactive)	R/W	0

Table 9–14. Interrupt Enable Register—Modem Mode (IER—MM) (Continued)

Bit	Name	Function	R/W	Value At Reset
3	MODEM_STS_IT	0: Disables the modem status register interrupt 1: Enables the modem status register interrupt	R/W	0
2	LINE_STS_IT	0: Disables the receiver line status interrupt 1: Enables the receiver line status interrupt	R/W	0
1	THR_IT	0: Disables the THR interrupt 1: Enables the THR interrupt	R/W	0
0	RHR_IT	0: Disables the RHR interrupt and time-out interrupt 1: Enables the RHR interrupt and time-out interrupt	R/W	0

Offset Address (hex): 0x01 x S and LCR[7] = 0 (and EFR[4] = 1 for IER[7:4]—Modem modes only)

The interrupt enable register (IER) can be programmed to enable/disable any interrupt. This mode has seven types of interrupts: receiver error, RHR interrupt, THR interrupt, XOFF received, and $\overline{CTS}$/$\overline{RTS}$ change of state from low to high. Each interrupt can be enabled/disabled individually. The IER also has a sleep-mode enable bit.

Bits 4, 5, 6, and 7 can only be written when EFR[4] = 1.

Table 9–15. Interrupt Enable Register—IrDA (IER—IrDA Mode)

Bit	Name	Function	R/W	Value At Reset
7	EOF_IT	0: Disables the received EOF interrupt 1: Enables the received EOF interrupt	R/W	0
6	LINE_STS_IT	0: Disables the receiver line status interrupt 1: Enables the receiver line status interrupt	R/W	0
5	TX_STATUS_IT	0: Disables the TX status interrupt 1: Enables the TX status interrupt	R/W	0
4	STS_FIFO_TRIG_IT	0: Disables status FIFO trigger level interrupt 1: Enables status FIFO trigger level interrupt	R/W	0
3	RX_OVERRUN_IT	0: Disables the RX overrun interrupt 1: Enables the RX overrun interrupt	R/W	0
2	LAST_RX_BYTE_IT	0: Disables the last byte of frame in RX FIFO interrupt 1: Enables the last byte of frame in RX FIFO interrupt	R/W	0

Table 9–15. Interrupt Enable Register—IrDA (IER—IrDA Mode) (Continued)

Bit	Name	Function	R/W	Value At Reset
1	THR_IT	0: Disables the THR interrupt 1: Enables the THR interrupt	R/W	0
0	RHR_IT	0: Disables the RHR interrupt 1: Enables the RHR interrupt	R/W	0

This interrupt enable register describes the eight types of IER and IrDA modes interrupts: received EOF, LSR interrupt, TX status, status FIFO interrupt, RX overrun, last byte in RX FIFO, THR interrupt, and RHR interrupt. All can be enabled/disabled individually.

The TX_STATUS_IT interrupt reflects two possible conditions. The MDR2[0] must be read to determine the status in the event of this interrupt.

Table 9–16.Interrupt Identification Register—Modem Mode (IIR—MM)

Bit	Name	Function	R/W	Value At Reset
7:6	FCR_MIRROR	Mirror the contents of FCR[0] on both bits	R	00
5:1	IT_TYPE		R	00000
0	IT_PENDING	0: An interrupt is pending (UART_nIRQ active). 1: No interrupt is pending (UART_nIRQ inactive).	R	1

Offset Address (hex): 0x02 x S and LCR is not 0xBF and read.

The IIR is a read-only register that provides the source of the interrupt in a prioritized manner.

Table 9–17. Interrupt Identification Register IrDA Mode (IIR—IrDA)

Bit	Name	Function	R/W	Value At Reset
7	EOF_IT	0: Received EOF interrupt inactive. 1: Received EOF interrupt active	R	0
6	LINE_STS_IT	0: Receiver line status interrupt inactive 1: Receiver line status interrupt active	R	0
5	TX_STATUS_IT	0: TX status interrupt inactive 1: TX status interrupt active	R	0
4	STS_FIFO_IT	0: Status FIFO trigger level interrupt inactive 1: Status FIFO trigger level interrupt active	R	0
3	RX_OE_IT	0: RX overrun interrupt inactive 1: RX overrun interrupt active	R	0

Table 9–17. Interrupt Identification Register IrDA Mode (IIR—IrDA) (Continued)

Bit	Name	Function	R/W	Value At Reset
2	RX_FIFO_LAST_ BYTE_IT	0: Last byte of frame in RX FIFO interrupt inactive 1: Last byte of frame in RX FIFO interrupt active	R	0
1	THR_IT	0: THR interrupt inactive 1: THR interrupt active	R	0
0	RHR_IT	0: RHR interrupt inactive 1: RHR interrupt active	R	0

The UART_nIRQ output is activated whenever one of the eight interrupts described in the interrupt identification register IrDA mode is active.

Table 9–18.Enhanced Features Register (EFR)

Bit	Name	Function	R/W	Value At Reset
7	AUTO_CTS_EN	Auto-CTS enable bit 0: Normal operation 1: Auto-CTS flow control is enabled, that is, transmission is halted when the $\overline{CTS}$ pin is high (inactive).	R/W	0
6	AUTO_RTS_EN	Auto-RTS enable bit 0: Normal operation 1: Auto-RTS flow control is enabled. $\overline{RTS}$ pin goes high (inactive) when the receiver FIFO HALT trigger level, TCR[3:0], is reached, and goes low (active) when the receiver FIFO RESTORE transmission trigger level is reached.	R/W	0
5	SPECIAL_CHAR_ DETECT	0: Normal operation 1: Special character detect enable Received data is compared with XOFF2 data. If a match occurs the received data is transferred to FIFO, and IIR bit 4 is set to 1 to indicate that a special character has been detected.	R/W	0
4	ENHANCED_EN	Enhanced functions write enable bit 0: Disables writing to IER bits 4–7, FCR bits 4–5, and MCR bits 5–7 1: Enables writing to IER bits 4–7, FCR bits 4–5, and MCR bits 5–7	R/W	0
3:0	SW_FLOW_ CONTROL	Combinations of software flow control can be selected by programming bits 3–0. See Table 9–19.	R/W	0000

The enhanced features register enables or disables enhanced features. Most of the enhanced functions apply only to the modem mode, but EFR[4] enables write access to FCR[5:4], the TX trigger level, which is also used in IrDA modes (see Table 9–19).

Table 9–19. Software Flow Control Options

Bit 3	Bit 2	Bit 1	Bit 0	TX, RX Software Flow Controls
0	0	X	X	No transmit flow control
1	0	X	X	Transmit XON1, XOFF1
0	1	X	X	Transmit XON2, XOFF2
1	1	X	X	Transmit XON1, XON2: XOFF1, XOFF2
X	X	0	0	No receive flow control
X	X	1	0	Receiver compares XON1, XOFF1
X	X	1	1	Receiver compares XON2, XOFF2
X	X	1	1	Receiver compares XON1, XON2: XOFF1, XOFF2

Table 9–20 and Table 9–21 show the XON1 and XON2 register bits. XON1 and XON2 must be set to different values if software flow control is enabled.

Table 9–20. XON1/ADDR1 Register

Bit	Name	Function	R/W	Value At Reset
7:0	XON_WORD1	Used to store the 8-bit XON1 character in UART modes and ADDR1 address 1 for IrDA modes	R/W	0x00

Table 9–21. XON2/ADDR2 Register

Bit	Name	Function	R/W	Value At Reset
7:0	XON_WORD2	Used to store the 8-bit XON2 character in UART modes and ADDR2 address 2 for IrDA modes	R/W	0x00

Table 9–22. XOFF1 Register

Bit	Name	Function	R/W	Value At Reset
7:0	XOFF_WORD1	Used to store the 8-bit XOFF1 character used in UART modes	R/W	0x00

Table 9–23. XOFF2 Register

Bit	Name	Function	R/W	Value At Reset
7:0	XOFF_WORD2	Used to store the 8-bit XOFF2 character used in UART modes	R/W	0x00

Table 9–24. Scratchpad Register (SPR)

Bit	Name	Function	R/W	Value At Reset
7:0	SPR_WORD	Scratchpad register	R/W	0x00

SPR is a R/W register that does not control the module in any way. It is a scratchpad register to hold temporary data.

Table 9–25 and Table 9–26 describe the two registers that store the 14-bit divisor for the generation of the baud clock in the baud rate generator. DLL stores the least-significant part of the divisor. DLH stores the most-significant part of the divisor.

DLL and DLH can be written to only before sleep mode is enabled, that is, before IER[4] is set.

Table 9–25. Divisor Latches Low Register (DLL)

Bit	Name	Function	R/W	Value At Reset
7:0	CLOCK_LSB	Used to store the 8-bit LSB divisor value	R/W	0x00

Table 9–26. Divisor Latches High Register (DLH)

Bit	Name	Function	R/W	Value At Reset
7:6	–	Reserved	R	00
5:0	CLOCK_MSB	Used to store the 6-bit MSB divisor value	R/W	000000

Table 9–27. Transmission Control Register (TCR)

Bit	Name	Function	R/W	Value At Reset
7:4	RX_FIFO_TRIG_START	RCV FIFO trigger level to RESTORE transmission (0 – 60)	R/W	0x0
3:0	RX_FIFO_TRIG_HALT	RCV FIFO trigger level to HALT transmission (0 – 60)	R/W	0xF

Notes: 1) Trigger levels from 0 – 60 bytes are available with a granularity of 4. (Trigger level = 4 x [4-bit register value])

2) The programmer must ensure that TCR[3:0] > TCR[7:4] whenever auto-RTS or software flow control is enabled, to prevent device malfunction.

3) In FIFO interrupt mode with flow control, the programmer also must ensure that the trigger level to HALT transmission is greater than or equal to the receive FIFO trigger level (either TLR[7:4] or FCR[7:6]). Otherwise, the FIFO operation stalls. This problem does not exist in FIFO DMA mode with flow control because DMA request is sent each time a byte is received.

The transmission control register stores the receive FIFO threshold levels to start/stop transmission during hardware/software flow control.

Table 9–28. Trigger Level Register(TLR)

Bit	Name	Function	R/W	Value At Reset
7:4	RX_FIFO_TRIG_DMA	RCV FIFO trigger level	R/W	0x0
3:0	TX_FIFO_TRIG_DMA	Transmit FIFO trigger level	R/W	0x0

The trigger level register stores the programmable transmit and receive FIFO trigger levels used for DMA and IRQ generation.

Table 9–29 and Table 9–30 summarize the different ways to set the trigger levels for the transmit FIFO and the receive FIFO, respectively.

Table 9–29. TX FIFO Trigger Level Setting Summary

SCR[6]	TLR[3:0]	TX FIFO Trigger Level
0	0000	Defined by FCR[5:4] (8,16, 32, or 56 spaces)
0	Non-zero	Defined by TLR[3:0] (from 4 to 60 spaces with a granularity of 4 spaces)
1	Value	Defined by the concatenated value of TLR[3:0] and FCR [5:4] (from 1 to 63 spaces with a granularity of 1 space). Note: The combination of TLR[3:0] = 0000 and FCR [5:4] = 00 (all zeros) is not supported (min 1 space required). All zeros result in unpredictable behavior.

Table 9–30. RX FIFO Trigger Level Setting Summary

SCR[7]	TLR[7:4]	RX FIFO Trigger Level
0	0000	Defined by FCR[7:6] (8,16, 56, or 60 characters)
0	Non-zero	Defined by TLR[7:4] (from 4 to 60 characters with a granularity of 4 characters)
1	Value	Defined by the concatenated value of TLR[7:4] and FCR [7:6] (from 1 to 63 characters with a granularity of 1 character) Note: The combination of TLR[7:4] = 0000 and FCR [7:6] = 00 (all zeros) is not supported (min 1 character required). All zeros result in unpredictable behavior.

Table 9–31. Mode Definition Register 1 (MDR1)

Bit	Name	Function	R/W	Value At Reset
7	FRAME_END_MODE	0: Frame-length method 1: Set EOT bit method	R/W	0
6	SIP_MODE	MIR/FIR modes only 0: Manual SIP mode: SIP is generated with the control of ACREG[3]. 1: Automatic SIP mode: SIP is generated after each transmission.	R/W	0

Table 9–31. Mode Definition Register 1 (MDR1) (Continued)

Bit	Name	Function	R/W	Value At Reset
5	SCT	Store and control the transmission 0: Starts the IrDA transmission as soon as a value is written to THR 1: Starts the IrDA transmission with the control of ACREG[2]	R/W	0
4	SET_TXIR	Used to configure the IrDA transceiver 0: No action 1: TXIR pin output is forced high.	R/W	0
3	IR_SLEEP	0: IrDA sleep mode disabled 1: IrDA sleep mode enabled	R/W	0
2:0	MODE_SELECT	000: UART 16x mode 001: SIR mode 010: UART 16x autobaud 011: UART 13x mode 100: MIR mode 101: FIR mode 110: Reserved 111: Disable (default state)	R/W	111

The mode of operation is programmed by writing to MDR1[2:0]. Therefore, the MDR1 must be programmed on start-up after configuring registers DLL, DLH, and LCR. The value of MDR1[2:0] must not be changed again during normal operation.

Table 9–32. Mode Definition Register 2 (MDR2)

Bit	Name	Function	R/W	Value At Reset
7:3	–	Reserved	R	00000
2:1	STS_FIFO_TRIG	Frame status FIFO threshold select: 00: 1 entry 01: 4 entries 10: 7 entries 11: 8 entries	R/W	00
0	IRTX_UNDERRUN	IRDA transmission status interrupt When the IIR[5] interrupt occurs, the meaning of the interrupt is: 0: IRTX last bit of the frame has been transmitted successfully without error. 1: IRTX underrun has occurred. The last bit of the frame has been transmitted but with an underrun error present. The bit is reset to 0 when the RESUME register is read.	R	0

MDR2[0] describes the status of the interrupt in IIR[5]. The IRTX_UNDERRUN bit should be read after an IIR[5] TX_STATUS_IT interrupt has occurred. The bits [2:1] of this register set the trigger level for the frame status FIFO (8 entries) and must be programmed before the mode is programmed in MDR1[2:0].

Table 9–33. UART Autobauding Status Register (UASR)

Bit	Name	Function	R/W	Value At Reset
7:6	PARITY_TYPE	00: No parity identified 01: Parity space 10: Even parity 11: Odd parity	R	00
5	BIT_BY_CHAR	0: 7 bits character identified 1: 8 bits character identified	R	0
4:0	SPEED	Used to report the speed identified 00000: No speed identified 00001: 115 200 bauds 00010: 57 600 bauds 00011: 38 400 bauds 00100: 28 800 bauds 00101: 19 200 bauds 00110: 14 400 bauds 00111: 9 600 bauds 01000: 4 800 bauds 01001: 2 400 bauds 01010: 1 200 bauds	R	00000

Modem autobauding mode only.

The UART autobauding status register returns the speed, the number of bits by characters, and the type of the parity in modem autobauding mode.

In autobauding mode the input frequency of the UART modem must be fixed to 48 MHz. Any other module clock frequency results in incorrect baud rate recognition.

This register sets up transmission according to characteristics of previous reception instead of the LCR, DLL, and DLH registers used when UART is in autobauding mode.

To reset the autobauding hardware (to start a new AT detection) or to set the UART in standard mode (no autobaud), MDR1[2:0] must be set to reset state 111, and then to the UART in autobaud mode 010 or UART in standard mode 000.

Usage limitation:

- Only 7- and 8-bit character (5 and 6 bits not supported)
- 7-bit character with space parity not supported
- Baud rate between 1200 and 115200 bps (10 possibilities)

Table 9–34 and Table 9–35 show the registers TXFLL and TXFLH, which hold the 13-bit transmit frame length (expressed in bytes). TXFLL holds the least-significant bits, and TXFLH holds the most-significant bits. The frame length value is used if the frame length method of frame closing is used.

Table 9–34. Transmit Frame Length Low Register (TXFLL)

Bit	Name	Function	R/W	Value At Reset
7:0	TXFLL	LSB register used to specify the frame length	W	0x00

Table 9–35. Transmit Frame Length High Register (TXFLH)

Bit	Name	Function	R/W	Value At Reset
7:5	–	Reserved	R	000
4:0	TXFLH	MSB register used to specify the frame length	W	00000

Table 9–36 and Table 9–37 show the egisters RXFLL and RXFLH, which hold the 12-bit receive maximum frame length. RXFLL holds the least-significant bits, and RXFLH holds the most-significant bits. If the intended maximum-receive frame length is n bytes, program RXFLL and RXFLH to be *n* + 3 in SIR or MIR modes and *n* + 6 in FIR mode (+3 and +6 are due to frame format with CRC and stop flag; there are two bytes associated with the FIR stop flag).

Table 9–36. Received Frame Length Low Register (RXFLL)

Bit	Name	Function	R/W	Value At Reset
7:0	RXFLL	LSB register used to specify the frame length in reception	W	0x00

Table 9–37. Received Frame Length High Register (RXFLH)

Bit	Name	Function	R/W	Value At Reset
7:4	–	Reserved	R	0x0
3:0	RXFLH	MSB register used to specify the frame length in reception	W	0x0

Table 9–38. Status FIFO Line Status Register (SFLSR)

Bit	Name	Function	R/W	Value At Reset
7:5	–	Reserved	R	000
4	OE_ERROR	1: Overrun error in RX FIFO when frame at top of FIFO was received	R	Unknown
3	FRAME_TOO_LONG_ ERROR	1: Frame-length too long error in frame at top of FIFO	R	Unknown
2	ABORT_DETECT	1: Abort pattern detected in frame at top of FIFO	R	Unknown
1	CRC_ERROR	1: CRC error in frame at top of FIFO	R	Unknown
0	–	Reserved	R	0

Reading the status FIFO line status register in effect reads frame-status information from the status FIFO. This register does not physically exist. Reading this register increments the status FIFO read pointer (SFREGL and SFREGH must be read first).

Table 9–39. Resume Register (RESUME)

Bit	Name	Function	R/W	Value At Reset
7:0	RESUME	Dummy read to restart the TX or RX	R	0x00

The resume register clears internal flags that halt transmission/reception when an underrun/overrun error occurs. Reading this register resumes the halted operation. This register does not physically exist and reads always as 0x00.

Table 9–40 and Table 9–41 describe the frame lengths of received frames that are written into the status FIFO. This information can be read from the SFREGL and SFREGH registers. These registers do not physically exist. The least-significant bits are read from SFREGL, and the most-significant bits are read from SFREGH. Reading these registers does not alter the status FIFO read pointer. These registers must be read before the pointer is incremented by reading the SFLSR.

Table 9–40. Status FIFO Register Low (SFREGL)

Bit	Name	Function	R/W	Value at Reset
7:0	SFREGL	LSB part of the frame length	R	Unknown

Table 9–41. Status FIFO Register High (SFREGH)

Bit	Name	Function	R/W	Value at Reset
7:4	–	Reserved	R	0x0
3:0	SFREGH	MSB part of the frame length	R	Unknown

Table 9–42. BOF Control Register (BLR)

Bit	Name	Function	R/W	Value at Reset
7	STS_FIFO_RESET	Status FIFO reset. This bit is self-clearing.	R/W	0
6	XBOF_TYPE	SIR xBOF select 0: 0xFF 1: 0xC0	R/W	1
5:0	–	Reserved	R	000000

BLR[6] is used to select whether 0xC0 or 0xFF start patterns are to be used when multiple start flags are required in SIR mode. If only one start flag is required, the start pattern is always 0xC0. If *n* start flags are required, either (*n*–1) 0xC0 or (*n*–1) 0xFF flags are sent, followed by a single 0xC0 flag (immediately preceding the first data byte).

Table 9–43. BOF Length Register (EBLR)

Bit	Name	Function	R/W	Value at Reset
7:0	EBLR	This register allows definition up to 176 xBOFs, the maximum required by IrDA specification.	R/W	0x00

The BOF length register specifies the number of BOF + xBOFs to transmit in IrDA SIR operation. The value set into this register must take into account the BOF character. To send only one BOF with no XBOF, this register must be set to 1. To send one BOF with N XBOF, this register must be set to *n*+1. Furthermore, the value 0 sends 1 BOF plus 255 XBOF.

In IrDA MIR mode, this register specifies the number of additional start flags (MIR protocol mandates a minimum of two start flags).

Table 9–44. Auxiliary Control Register (ACREG)

Bit	Name	Function	R/W	Value at Reset
7	PULSE_TYPE	SIR pulse width select 0: 3/16 of baud-rate pulse width 1: 1.6 µs	R/W	0
6	SD_MOD	Primary output used to configure transceivers. Connected to the SD/MODE input pin of IrDA transceivers 0: SD pin is set to high 1: SD pin is set to low	R/W	0

Table 9–44. Auxiliary Control Register (ACREG) (Continued)

Bit	Name	Function	R/W	Value at Reset
5	DIS_IR_RX	0: Normal operation (Note: RXIR input automatically disabled during transmit but enabled outside of transmit operation.) 1: Disables RXIR input (permanent state—independent of transmit) Hence, RX_IR is disabled when either TX is active or ACREG[5] = 1.	R/W	0
4	DIS_TX_UNDERRUN	0: Long stop bits cannot be transmitted, and TX underrun is enabled. 1: Long stop bits can be transmitted, and TX underrun is disabled.	R/W	0
3	SEND_SIP	MIR/FIR modes only Send serial infrared interaction pulse (SIP). 0: No action 1: Send SIP pulse. If this bit is set during a MIR/FIR transmission, the SIP is sent at the end of it. This bit is automatically cleared at the end of the SIP transmission.	R/W	0
2	SCTX_EN	Store and controlled TX start When MDR1[5] = 1 and the LH writes 1 to this bit, the TX state machine starts frame transmission. This bit is self-clearing.	R/W	0
1	ABORT_EN	Frame abort The LH can intentionally abort transmission of a frame by writing 1 to this bit. Neither the end flag nor the CRC bits are appended to the frame.	R/W	0
0	EOT_EN	EOT (end of transmission) bit The LH writes 1 to this bit just before it writes the last byte to the TX FIFO in set-EOT bit frame closing method.This bit is automatically cleared when the LH writes to the THR (TX FIFO).	R/W	0

Table 9–45.Module Version Register (MVR)

Bit	Name	Function	R/W	Value at Reset
7:4	MAJOR_REV	Major revision number of the module	R	1
3:0	MINOR_REV	Minor revision number of the module	R	–

Note: UART/IRDA SIR only module is revision 1.x. UART/IRDA with SIR, MIR, and FIR support is revision 2.x.

Table 9–46. System Configuration Register (SYSC)

Bit	Name	Function	R/W	Value At Reset
7:5	–	Reserved	R	000
4:3	IdleMode	Power management request/acknowledge control 00: Force idle. An idle request is acknowledged unconditionally. 01: No idle. An idle request is never acknowledged. 10: Smart idle. Acknowledgement to an idle request is given based on the internal activity of the module. 11: Reserved Ref: OCP design guidelines version 1.1	R/W	00
2	EnaWakeUp	Wake-up feature control 0: Wake up is disabled. 1: Wake-up capability is enabled.	R/W	0
1	SoftReset	Software reset Set this bit to 1 to trigger a module reset. This bit is automatically reset by the hardware. During reads it always returns a 0. 0: Normal mode 1: The module is reset.	R/W	0
0	AutoIdle	Internal OCP clock gating strategy 0: Clock is running. 1: Automatic OCP clock gating strategy is applied, based on the OCP interface activity.	R/W	0

The autoidle bit in the system configuration register controls a power-saving technique to reduce the logic power consumption of the OCP interface. That is, when the feature is enabled the clock is gated off until an OCP command for this device has been detected.

When the software reset bit is set high it causes a full device reset.

Table 9–47. System Status Register (SYSS)

Bit	Name	Function	R/W	Value At Reset
7:1	–	Reserved	R	0000000
0	ResetDone	Internal reset monitoring 0: Internal module reset is ongoing. 1: Reset completed	R	0

Table 9–48. Wake-Up Enable Register (WER)

Bit	Name	Function	R/W	Value At Reset
7	–	Reserved	R	0
6	Event 6 Receiver line status interrupt	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
5	Event 5 RHR interrupt	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
4	Event 4 RX/ RXIR activity	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
3	Event 3 DCD (CD) activity	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
2	Event 2 RI activity	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
1	Event 1 DSR activity	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1
0	Event 0 CTS activity	0: Event is not allowed to wake up the system. 1: Event can wake up the system.	W/R	1

The UART wake-up enable register masks and unmasks a UART event that would subsequently notify the system. Such events are any activity in the logic that can cause an interrupt and/or any that require the system to wake up. Even if the wake-up is disabled for certain events, if these events also interrupt the UART, the UART still registers the interrupt as such.

9.4 Modes of Operation

The UART modem/IRDA module can operate in six different modes:

- ❏ Modem16x mode (≤230.4K bits/s)
- ❏ Modem16x mode with autobauding (≥1200 bits/s and ≤115.2K bits/s)
- ❏ Modem13x mode (≥460.8K bits/s)
- ❏ IrDA SIR mode (≤115.2K bits/s)
- ❏ IrDA MIR mode (0.576 and 1.152M bits/s)
- ❏ IrDA FIR mode (4M bits/s)

The module performs serial-to-parallel conversion on received data characters, and parallel-to-serial conversion on data characters the processor transmits. The complete status of each channel of the module and each received character/frame can be read at any time during functional operation via the line status register (LSR).

The module can be placed in an alternate mode (FIFO mode), relieving the processor of excessive software overhead by buffering received/transmitted characters. Both the receiver and transmitter FIFOs store up to 64 bytes of data (plus 3 additional bits of error status per byte for the receiver FIFO) and have selectable trigger levels.

Both interrupts and DMA are available to control the data flow between the LH and the module.

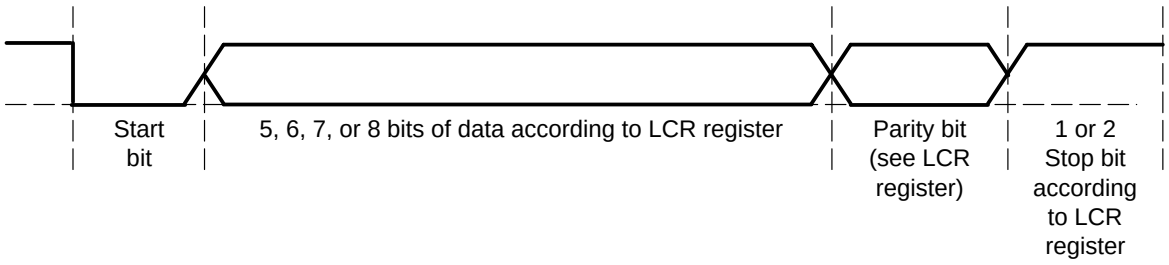
9.4.1 Modem Mode

The UART uses a wired interface for serial communication with a remote device.

The module can use hardware or software flow control to manage transmission/reception. Hardware flow control uses the RTS output and CTS input signals to automatically control serial data flow. This significantly reduces software overhead and increases system efficiency. Software flow control uses programmable XON/XOFF characters to automatically control data flow.

The UART modem module is enhanced with an autobauding functionality, which in control mode allows for automatic setting of the speed, number of bits per character, and parity.

Figure 9–3. UART Data Format



9.4.2 SIR Mode

In slow infrared (SIR) mode, data transfer takes place between the LH and peripheral devices at speeds up to 115200 bauds. A SIR transmit frame begins with start flags (either a single 0xC0, multiple 0xC0, or a single 0xC0 preceded by a number of 0xFF flags), followed by frame data, CRC–16, and ends with a stop flag (0xC1).

BLR[6] is used to select whether 0xC0 or 0xFF start patterns are to be used, when multiple start flags are required.

The SIR transmit state machine attaches start flags, CRC–16, and stop flags. It checks the outgoing data to determine whether data transparency is required.

SIR transparency is carried out if the outgoing data, between the start and stop flags, contains 0xC0, 0xC1, or 0x7D. If one of these is about to be transmitted, the SIR state machine sends an escape character (0x7D) first, then inverts the fifth bit of the real data to be sent, and sends this data immediately after the 0x7D character.

The SIR receive state machine recovers the receive clock, removes the start flags, removes any transparency from the incoming data, and determines frame boundary with reception of the stop flag. It also checks for errors such as frame abort (0x7D character followed immediately by a 0xC1 stop flag, without transparency), CRC error, and frame-length error. At the end of a frame reception, the LH reads the line status register (LSR) to find possible errors in the received frame.

The module can transfer data both ways, but when the device is transmitting, hardware automatically disables the IR RX circuitry. See Table 9–44, Auxiliary Control Register, for a description of the logical operation of all three modes, SIR, MIR, and FIR.

The infrared output in SIR mode can be either 1.6 µs or 3/16 encoding, selected by the PULSE_TYPE bit of the auxiliary control register (ACREG[7]). In 1.6 µs encoding, the infrared pulse width is 1.6 µs, and in 3/16 encoding the infrared pulse width is 3/16 of a bit duration (1/baud-rate).

The transmitting device must send at least 2 start flags at the start of each frame for back-to-back frames.

Reception supports variable-length stop bits.

9.4.2.1 Frame Format

Figure 9–4. IrDA SIR Frame Format

The CRC is applied on the address (A), control (C), and information (I) bytes.

The two words of CRC are written in the FIFO in reception.

9.4.2.2 Asynchronous Transparency

Before transmitting a byte, the UART IrDA controller examines each byte of the payload and the CRC field (between BOF and EOF). For each byte equal to 0xC0 (BOF), 0xC1 (EOF), or 0x7D (control escape) it does the following.

In transmission:

- ❏ Inserts a control escape (CE) byte preceding the byte
- ❏ Complements bit 5 of the byte (that is, exclusive ORs the byte with 0x20)

The byte sent for the CRC computation is the initial byte written in the TX FIFO (before the XOR with 0x20).

In reception:

For the A, C, I, and CRC fields:

- ❏ Compares the byte with the CE byte. If not equal, sends it to the CRC detector and stores it in the RX FIFO.
- ❏ If equal to CE, discards the CE byte
- ❏ Complements bit 5 of the byte following the CE
- ❏ Sends the complemented byte to the CRC detector and stores it in the RX FIFO

9.4.2.3 Abort Sequence

The transmitter may decide to prematurely close a frame. The transmitter aborts by sending the 0x7DC1 sequence. The abort pattern closes the frame without a CRC field or an ending flag.

It is possible to abort a transmission frame by programming the ABORT_EN bit of the auxiliary control register (ACREG [1]).

When this bit is set to 1, 0x7D and 0xC1 are transmitted and the frame is not terminated with CRC or stop flags.

The receiver treats a frame as an aborted frame when a 0x7D character followed immediately by a 0xC1 character is received without transparency.

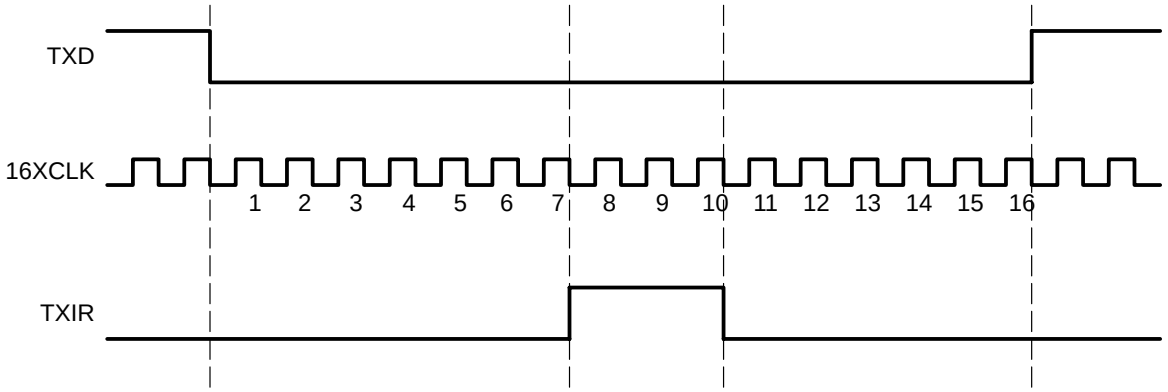
9.4.2.4 Pulse Shaping

In SIR mode both the 3/16 and the 1.6 ìs pulse duration methods are supported. ACREG[7] selects the pulse-width method in transmit mode.

9.4.2.5 Encoder

Serial data from the transmit state machine is encoded to transmit data to the optoelectronics. While the serial data input to the (TXD) is high, the output (TXIR) is always low, and the counter used to form a pulse on TXIR is continuously cleared. After TXD resets to 0, TXIR rises on the falling edge of the 7^{th} 16XCLK. On the falling edge of the 10^{th} 16XCLK pulse, TXIR falls, creating a 3-clock-wide pulse. While TXD stays low, a pulse is transmitted during the 7^{th} to the 10^{th} clocks of each 16-clock bit cycle.

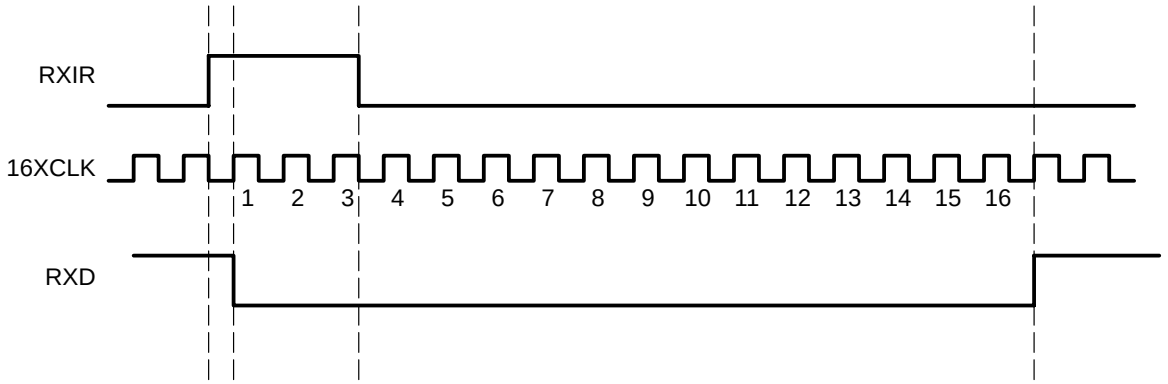
Figure 9–5. IrDA Encoder Mechanism



Decoder

After reset, RXD is high and the 4-bit counter is cleared. When a rising edge is detected on RXIR, RXD falls on the next rising edge of 16XCLK with sufficient setup time. RXD stays low for 16 cycles (16XCLK) and then returns to high as required by the IrDA specification. As long as no pulses (rising edges) are detected on the RXIR, RXD remains high.

Figure 9–6. IrDA Decoder Mechanism



The reception of RXIR input is disabled with DIS_IR_RX bits of the auxiliary control register (ACREG[5]).

9.4.2.6 IR Address Checking

In all IR modes, if address checking has been enabled, only frames intended for the device are written to the RX FIFO. This is to avoid receiving frames not meant for this device in a multi-point infrared environment. It is possible to program two frame addresses that the UART IrDA receives with XON1/ADDR1 and XON2/ADDR2 registers.

Selecting address1 checking is done by setting EFR[0] to 1, and address2 checking is done by setting EFR[1] to 1. Setting EFR[1:0] to 0 disables all address checking operations. If both bits are set, the incoming frame is checked for both private and public addresses.

If address checking is disabled, all received frames are written into the reception FIFO.

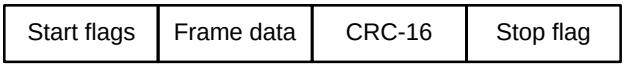
9.4.3 MIR Mode

In medium infrared (MIR) mode, data transfer takes place between the LH and peripheral devices at 0.576 or 1.152M bits/s speed. A MIR transmit frame begins with start flags (at least 2), followed by a frame data, CRC–16, and ends with a stop flag.

9.4.4 MIR Transmit Frame Format

On transmit, the MIR state machine attaches start flags, CRC–16, and stop flags. It also looks for 5 consecutive 1s in the frame data and automatically inserts 0 after them. (This is called bit stuffing.)

Figure 9–7. MIR Transmit Frame Format



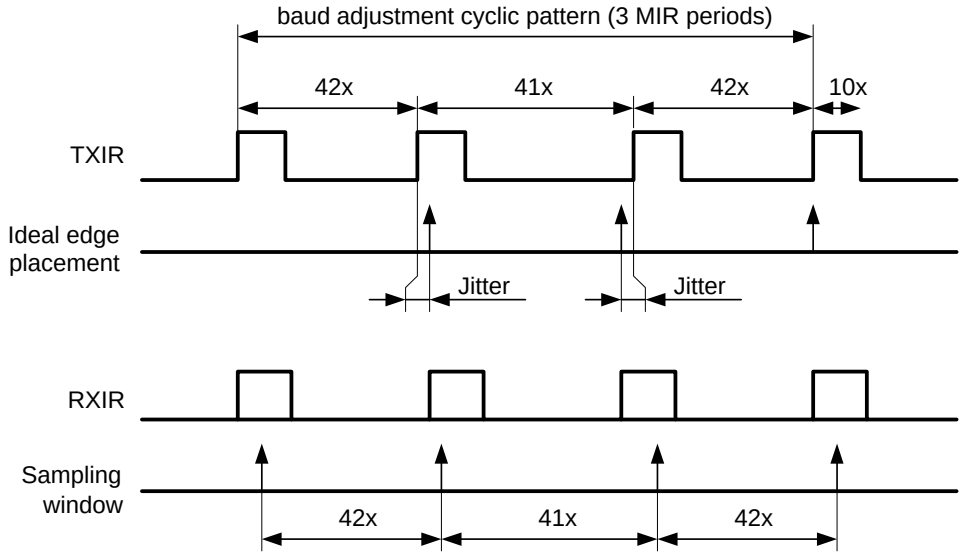
On receive, the MIR receive state machine recovers the receive clock, removes the start flags, destuffs the incoming data, and determines the frame boundary with reception of the stop flag. It also checks for errors such as frame abort, CRC error, or frame-length error. At the end of a frame reception, the LH reads the line status register (LSR) to discover possible errors in the received frame.

The module transfers data both ways, but when the device is transmitting, hardware automatically disables the IR RX circuitry. See Table 9–44, Auxiliary Control Register, for a description of the logical operation of all three modes, SIR, MIR, and FIR.

9.4.4.1 MIR Encoder/Decoder

In order to meet MIR baud-rate tolerance of ±0.1% with a 48-Mhz clock input, a 42–41–42 encoding/decoding adjustment is performed. The reference start point is 1st start flag, and the 42–41–42 cyclic pattern is repeated until the stop flag is sent or detected. The jitter created this way is within MIR tolerances. The pulse width is not exactly 1/4 but within tolerances defined by the IrDA specifications.

Figure 9–8. MIR Baud-Rate Adjustment Mechanism

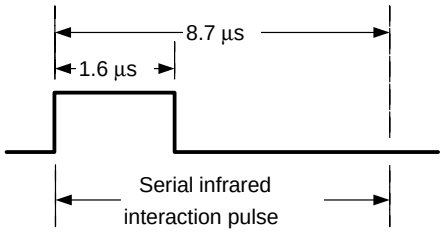


9.4.4.2 SIP Generation

In MIR and FIR operation modes, the transmitter needs to send a serial infrared interaction pulse (SIP) at least once every 500 ms. The purpose of the SIP is to let slow devices (operating in SIR mode) know that the medium is currently occupied.

The SIP pulse is shown below in Figure 9–9.

Figure 9–9. SIP Pulse

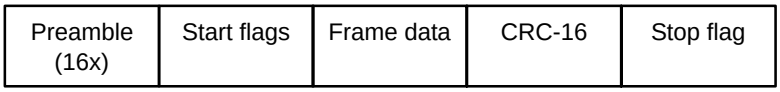


When the SIP_MODE bit of mode definition register 1 = 1 (MDR1[6]), the TX state machine always sends 1 SIP at the end of a transmission frame. But when MDR1[6] = 0, the transmission of the SIP depends on the SEND_SIP bit of the auxiliary control register (ACREG[3]). The local host can set ACREG[3] at once every 500 ms. The advantage of this approach over the default is that the TX state machine does not need to send the SIP at the end of each frame, which can reduce the overhead required.

9.4.5 FIR Mode

In fast infrared mode (FIR), data transfer takes place between the LH and peripheral devices at 4M bits/s. A FIR transmit frame starts with a preamble, followed by a start flag, frame data, CRC–32, and ends with a stop flag.

Figure 9–10. FIR Transmit Frame Format



On transmit, the FIR transmit state machine attaches the preamble, start flag, CRC–32, and stop flag. It also encodes the transmit data into 4PPM format and generates the serial infrared interaction pulse (SIP).

On receive, the FIR receive state machine recovers the receive clock, removes the start flag, decodes the 4PPM incoming data, and determines the frame boundary with reception of the stop flag. It also checks for errors such as illegal symbol, CRC error, and frame-length error. At the end of a frame reception, the LH reads the line status register (LSR) to discover possible errors in the received frame.

The module transfers data both ways, but when the device is transmitting, hardware automatically disables the IR RX circuitry. See Table 9–44, Auxiliary Control Register, for a description of the logical operation for all three modes, SIR, MIR, and FIR.

9.5 Functional Description

9.5.1 Trigger Levels

The UART provides programmable trigger levels for both receiver and transmitter DMA and interrupt generation. After reset, both transmitter and receiver FIFOs are disabled, so in effect the trigger level is the default value of 1 byte. The programmable trigger levels are an enhanced feature available via the trigger level register (TLR).

9.5.2 Interrupts

The UART modem/IrDA module generates interrupts on the UART_nIRQ output pin. All interrupts are enabled/disabled by writing to the appropriate bit in the interrupt enable register (IER). The interrupt status of the device can be checked at any time by reading the interrupt identification register (IIR).

The modem and IrDA modes have different interrupts in the UART modem/IrDA module, and therefore different IER and IIR mappings according to the selected mode.

9.5.2.1 Modem Mode Interrupts

The modem modes have seven possible interrupts. These interrupts are prioritized to six different levels.

When an interrupt is generated, the interrupt identification register (IIR) indicates that an interrupt is pending by bringing IIR[0] to 0 and provides the type of interrupt through IIR[5–1]. It also summarizes the interrupt control functions.

Table 9–49. Modem Mode Interrupts

IIR[5–0]	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Method
0 0 0 0 0 1	None	None	None	None
0 0 0 1 1 0	1	Receiver line status	OE, FE, PE, or BI errors occur in characters in the RX FIFO.	FE,PE,BI: Read RHR. OE: Read LSR
0 0 1 1 0 0	2	RX time-out	Stale data in RX FIFO	Read RHR
0 0 0 1 0 0	2	RHR interrupt	DRDY (data ready) (FIFO disable)	Read RHR until interrupt condition disappears.
			RX FIFO above trigger level (FIFO enable)	
0 0 0 0 1 0	3	THR interrupt	TFE (THR empty) (FIFO disable)	Write to THR until interrupt condition disappears.
			TX FIFO below trigger level (FIFO enable)	
0 0 0 0 0 0	4	Modem status	MSR[1:0] / = 0	Read MSR.

Table 9–49. Modem Mode Interrupts (Continued)

IIR[5–0]	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Method
0 1 0 0 0 0	5	XOFF interrupt/ special character interrupt	Receive XOFF characters(s)/special character	Receive XON character(s), if XOFF interrupt/read of IIR, if special character interrupt.
1 0 0 0 0 0	6	CTS,RTS, DSR	RTS pin, CTS pin or DSR pin change state from active (low) to inactive (high).	Read IIR.

It is important to note that for the receiver line status interrupt, RX_FIFO_STS bit (LSR[7]) generates the interrupt.

For the XOFF interrupt, if an XOFF flow character detection caused the interrupt, an XON flow character detection clears the interrupt. If special character detection caused the interrupt, a read of the IIR clears the interrupt.

9.5.2.2 IrDA Mode Interrupts

IrDA modes have eight possible interrupts. The UART_nIRQ output is activated when any of the eight interrupts is generated (there is no priority).

Table 9–50. IrDA Mode Interrupts

IIR Bit No.	Interrupt Type	Interrupt Source	Interrupt Reset Method
0	RHR interrupt	DRDY (data ready) (FIFO disable) RX FIFO above trigger level (FIFO enable)	Read RHR until interrupt condition disappears.
1	THR interrupt	TFE (THR empty) (FIFO disable) TX FIFO below trigger level (FIFO enable)	Write to THR until interrupt condition disappears.
2	Last byte in RX FIFO	Last byte of frame in RX FIFO	Read IIR.
3	RX overrun	Write to RHR when RX FIFO full.	Read RESUME register.
4	Status FIFO interrupt	Status FIFO triggers level reached.	Read STATUS FIFO.
5	TX status	1.THR empty before EOF sent. Last bit of transmission of the IRDA frame has occurred but with an underrun error. OR 2.Transmission of the last bit of the IRDA frame is finished successfully.	1. Read RESUME register. OR 2. Read IIR.

Table 9–50. IrDA Mode Interrupts(Continued)

IIR Bit No.	Interrupt Type	Interrupt Source	Interrupt Reset Method
6	Receiver line status interrupt	CRC, ABORT or frame-length error is written into STATUS FIFO.	Read STATUS FIFO [Read until empty—max 8 reads required].
7	Received EOF	Received end-of-frame.	Read IIR.

For IIR[5] the interrupt source 1 is used with interrupt reset method 1. The interrupt source 2 is used with interrupt reset method 2.

9.5.2.3 Wake-Up Interrupt

Wake-up interrupt is a special interrupt, not designed the same as the previous ones. It is enabled when the RX_CTS_DSR_WAKE_UP_ENABLE bit of the supplementary control register (SCR[4]) is set to 1. The IIR register is not modified when it occurs. SSR[1] must be checked to detect a wake-up event. When wake-up interrupt occurs, the only way to clear it is to reset SCR[4] to 0.

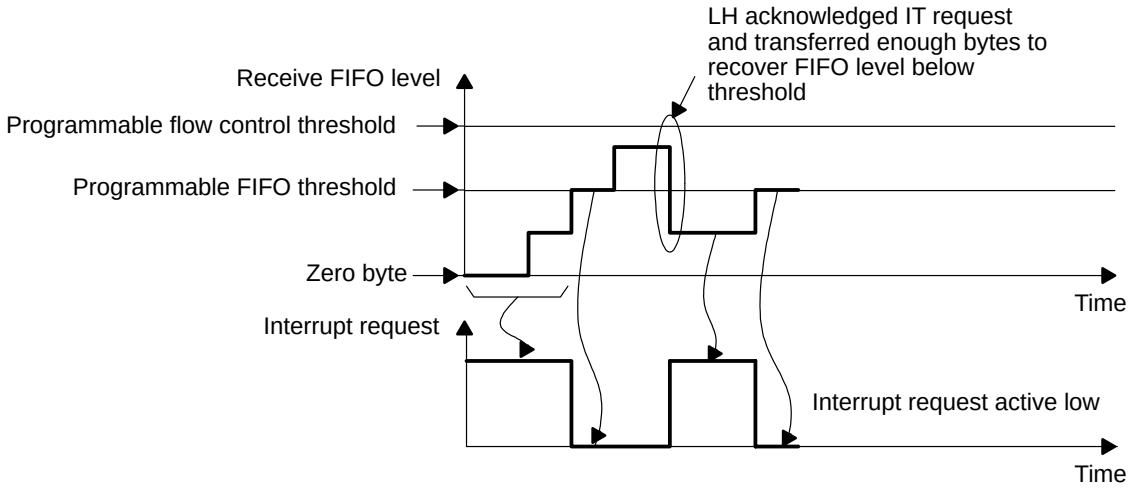
9.5.3 FIFO Interrupt Mode Operation

In FIFO interrupt mode (FIFO control register FCR[0] = 1, relevant interrupts enabled via IER), an interrupt signal (UART_nIRQ) informs the processor of the receiver and transmitter status. These interrupts are raised when receive/transmit FIFO threshold (respectively, TLR[7:4] and TLR[3:0] or FCR[7:6] and FCR[5:4]) are reached. The interrupt signals instruct the local host to transfer data to the destination (from the UART module in receive mode and/or from any source to the UART FIFO in transmit mode).

In the case of the UART flow control being enabled along with the interrupt capabilities, the user must ensure that the UART flow control FIFO threshold (TCR[3:0]) is greater than or equal to the receive FIFO threshold.

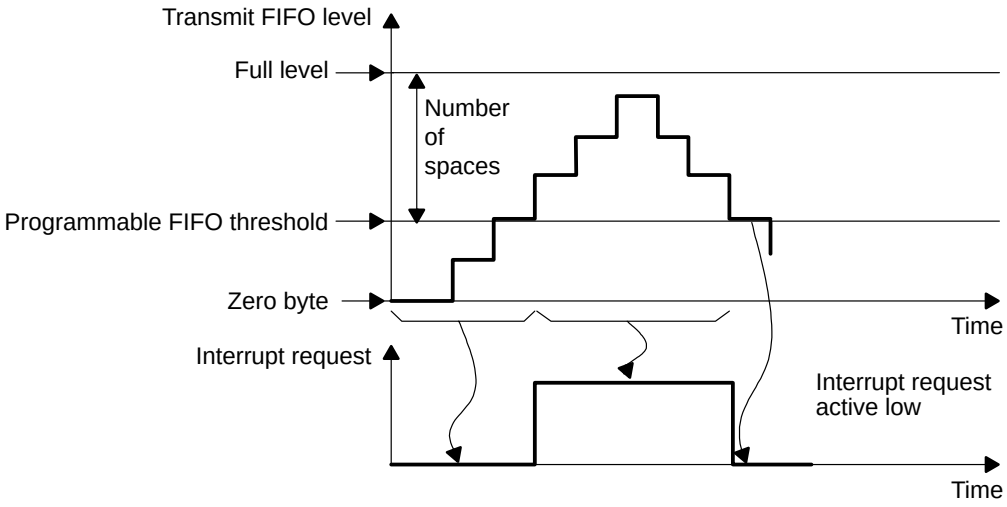
Figure 9–11 and Figure 9–12, respectively, depict receive and transmit operations.

Figure 9–11. Receive FIFO IT Request Generation



In receive, no interrupt is generated until receive FIFO reaches its threshold. Once low, the interrupt can only be deasserted when the local host has handled enough bytes to put the FIFO level below threshold. The flow control threshold is set at a higher value than FIFO threshold.

Figure 9–12. Transmit FIFO IT Request Generation



In transmit mode, an interrupt request is automatically asserted when FIFO is empty. This request is deasserted when the FIFO crosses the threshold level. The interrupt line is deasserted until a sufficient number of elements has been transmitted to go below FIFO threshold.

9.5.4 FIFO Polled Mode Operation

In FIFO polled mode (FCR [0] = 0, relevant interrupts disabled via the interrupt enable register (IER)), the status of the receiver and transmitter are checked by polling the line status register (LSR). This mode is an alternative to the FIFO interrupt mode of operation in which the status of the receiver and transmitter is automatically known by means of interrupts sent to the LH.

9.5.5 FIFO DMA Mode Operation

9.5.5.1 DMA Signaling

The four modes of DMA operation, DMA modes 0/1/2/3, are selected as follows:

When SCR[0] = 0: Setting FCR[3] to 0 enables DMA mode 0.

Setting FCR[3] to 1 enables DMA mode 1.

When SCR[0] = 1: SCR[2:1] determine DMA mode 0 to 3 according to supplementary control register (SCR) description.

For example:

- If no DMA operation is desired: Set SCR[0] to 1 and SCR[2:1] to 00 (FCR[3] is discarded).

- If DMA mode 1 is desired: Either set SCR[0] to 0 and FCR[3] to 1 or set SCR[0] to 1 and SCR[2:1] to 01 (FCR[3] is discarded).

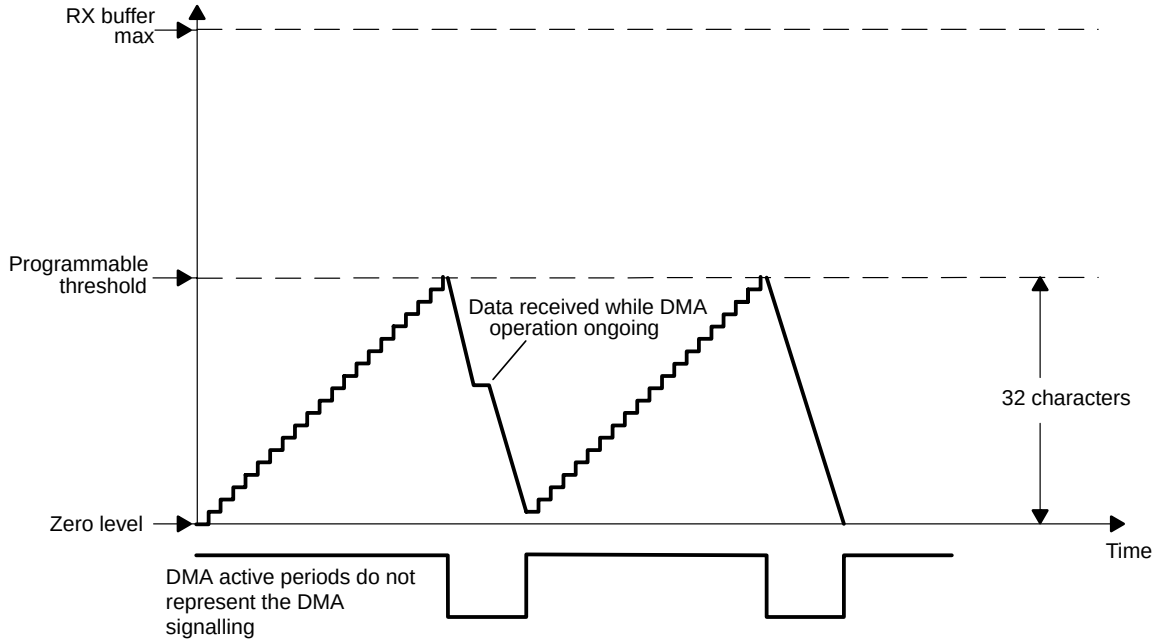
If the FIFOs are disabled (FCR[0] = 0), DMA occurs in single-character transfers.

When DMA mode 0 has been programmed, the signals associated with DMA operation are not active.

9.5.5.2 DMA Transfers (DMA Mode 1, 2, or 3)

Figure 9–13 through Figure 9–16 show the supported DMA operations.

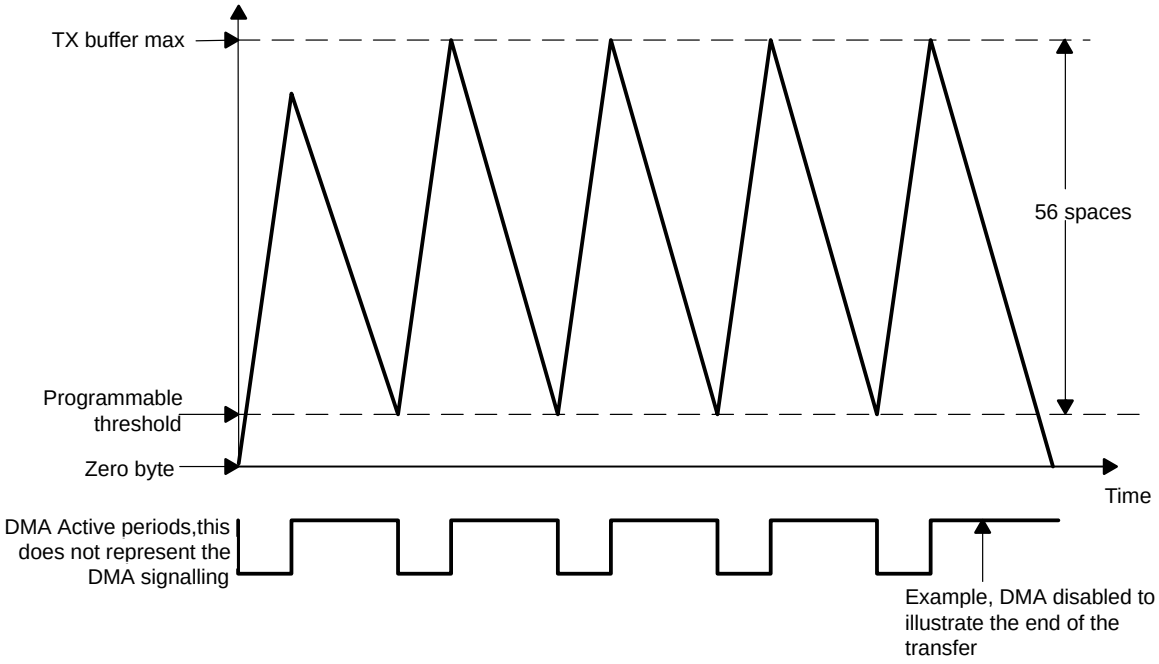
Figure 9–13. Receive FIFO DMA Request Generation (32 Characters)



In receive mode, a DMA request is generated as soon as the receive FIFO reaches its threshold level as defined in the trigger level register (TLR). (See Table 9–28.) This request is deasserted when the system DMA reads the number of bytes defined by the threshold level.

In transmit mode, a DMA request is automatically asserted when the FIFO is empty. This request is deasserted when the system DMA writes the number of bytes defined by the number of spaces in the trigger level register (TLR). If an insufficient number of characters is written, the DMA request remains active.

Figure 9–14. Transmit FIFO DMA Request Generation (56 Spaces)

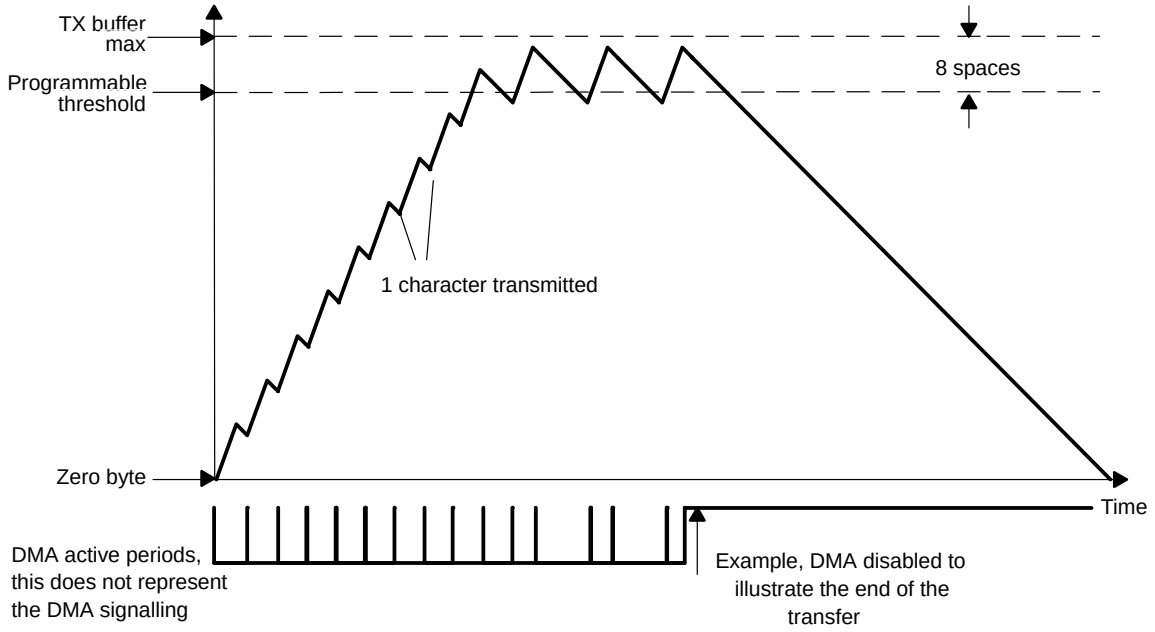


The DMA request is again asserted if the FIFO is able to receive the number of bytes defined by the TLR register. (See Table 9–28.)

The threshold can be programmed in a number of ways. See Figure 9–14 for an example of a DMA transfer that operates with a space setting of 56, which can arise from the use of the autosettings in the FCR[5:4] or the use of the TLR[3:0] concatenated with the FCR[5:4]. The setting of 56 spaces in the UART_IrDA must correlate with settings of the system DMA so that the buffer does not overflow (program the DMA request size of the local host controller to be equal to the number of spaces value in the UART).

Figure 9–15 shows another example with 8 spaces, to illustrate the buffer level crossing the space threshold. Again, the local host DMA controller settings must correspond to those of the UART_IrDA.

Figure 9–15. Transmit FIFO DMA Request Generation (8 Spaces)

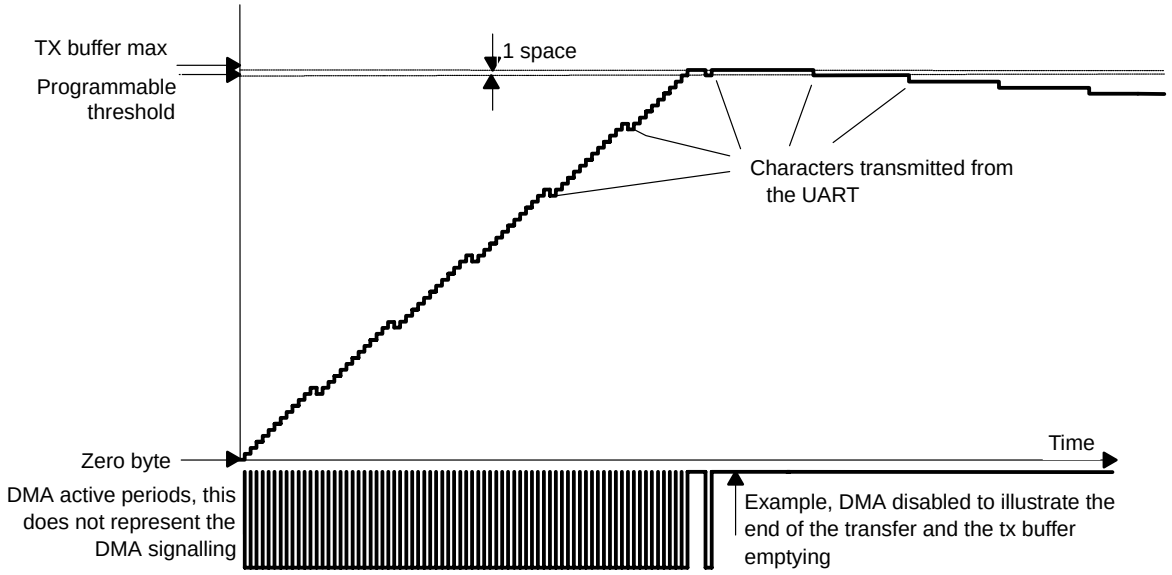


The final example in Figure 9–16 illustrates the setting of 1 space that uses the DMA for each transfer of 1 character to the transmit buffer. The buffer is filled at a faster rate than the BAUD rate transmits data to the TX pin. Eventually, the buffer is completely full and the DMA operation stops transferring data to the transmit buffer.

The buffer holds the maximum amount of data words on two occasions. Shortly after that the DMA is disabled to illustrate the slower transmission of the data words to the TX pin. Eventually, the buffer is emptied at the rate specified by the baud-rate settings of the DLL and DLH registers.

Again, the DMA settings must correspond to the local host DMA controller settings to ensure the correct operation of this logic.

Figure 9–16. Transmit FIFO DMA Request Generation (1 Space)



9.5.6 Sleep Mode

9.5.6.1 Modem Modes

In modem modes, sleep mode is enabled by writing a 1 to IER[4] (when EFR[4] = 1).

Sleep mode is entered when

- The serial data input line, RX, is idle.
- The TX FIFO and TX shift register are empty.
- The RX FIFO is empty.
- There are no interrupts pending except THR interrupts.

Sleep mode is a good way to lower power consumption of the UART, but this state can be achieved only when the UART is set in modem mode. Therefore, even if the UART has no functional key role, it must be initialized in a functional mode to take advantage of sleep mode.

In sleep mode the module clock and baud rate clock are stopped internally. Because most registers are clocked using these clocks, the power consumption is greatly reduced. The module wakes up when any change is detected on the RX line; if data is written to the TX FIFO, it occurs with any change in the state of the modem input pins. An interrupt is generated on a wake-up event by setting SCR[4] to 1.

Note: Writing to the divisor latches, DLL and DLH, to set the baud clock, BCLK, must not be done during sleep mode. Therefore, it is advisable to disable sleep mode using IER[4] before writing to DLL or DLH.

9.5.7 IrDA Modes

In IrDA modes, sleep mode is enabled by writing a 1 to MDR1[3].

Sleep mode is entered when

- ❏ The serial data input line, RXIR, is idle.
- ❏ The TX FIFO and TX shift register are empty.
- ❏ The RX FIFO is empty.
- ❏ There are no interrupts pending except THR interrupts.

The module wakes up when any change is detected on the RXIR line, if data is written to the TX FIFO.

9.5.8 Idle Modes

Sleep and autoIdle modes are embedded power-saving features. At the system level, power reduction techniques are applied by shutting down certain internal clock and power domains of the device.

The UART supports REQ_IDLE ACK handshaking protocol. This protocol is used at system level to shut down UART clocks in a clean and controlled manner and to switch the UART from the interrupt generation mode to a wake-up generation mode for unmasked events (Refer to SYSC[2] and WER.)

For a software programming guide, refer to the *OCP Design Guidelines for Idle Mode Control*.

9.5.9 Break and Time-Out Conditions

9.5.9.1 Time-Out Counter

An RX idle condition is detected when the receiver line, RX, has been high for a time equivalent to 4X programmed word length+12 bits. The receiver line is sampled midway through each bit.

For sleep mode, the counter is reset when there is activity on the RX line.

For the timeout interrupt, the counter only counts when there is data in the RX FIFO. The count is reset when there is activity on the RX line or when the RHR is read.

9.5.9.2 Break Condition

When a break condition occurs, the TX line is pulled low. A break condition is activated by setting LCR[6]. Be aware that the break condition is not aligned on word stream, that is, a break condition can occur in the middle of a character. The only way to send a break condition on a full character is:

1) Reset transmit FIFO (if enabled).
2) Wait for transmit shift register to become empty (LSR[6] = 1).
3) Take a guard time according to stop bit definition.
4) Set LCR[6] to 1.

Break condition is asserted as long as LCR[6] is set to 1.

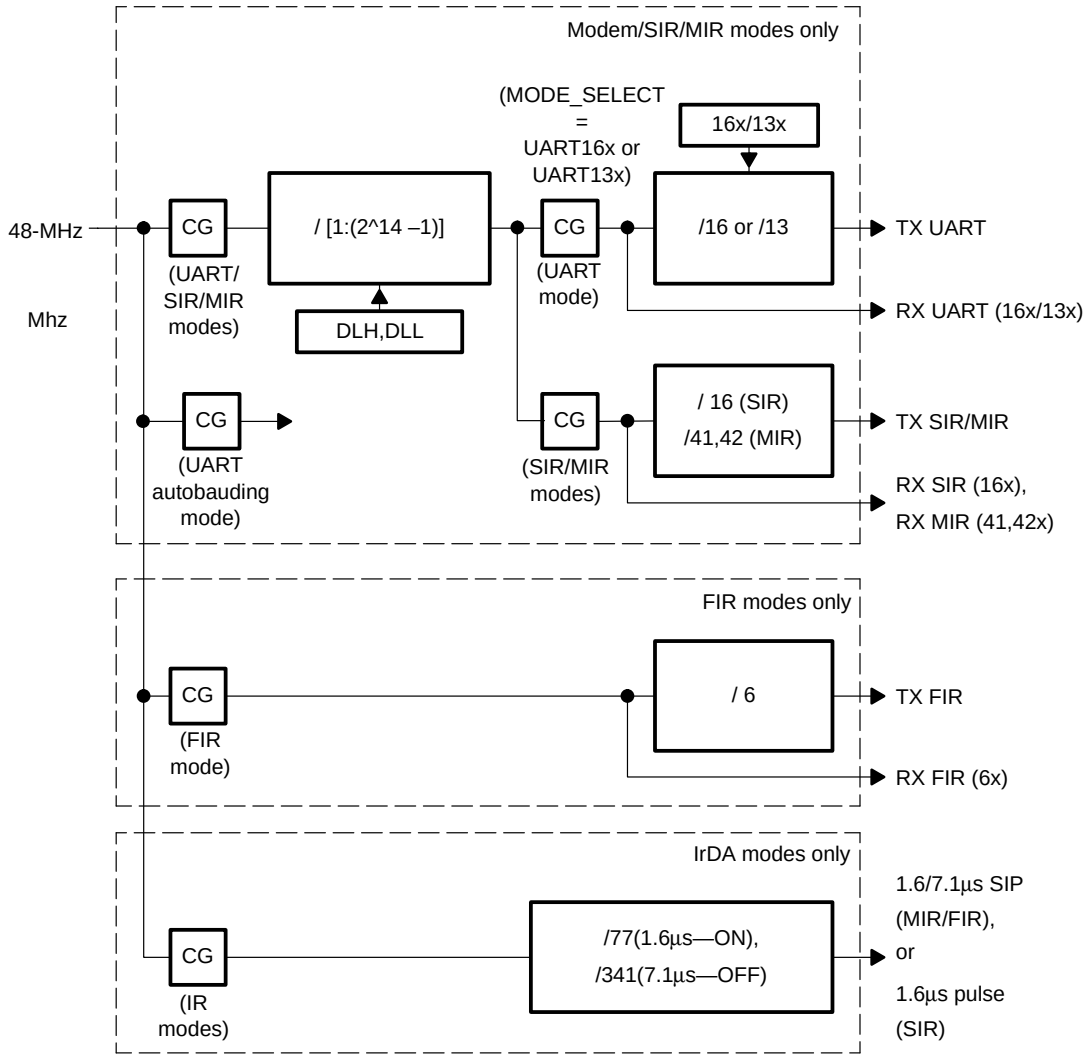
The above functionality (time-out counter and break condition) applies only to the UART modem operation and does not extend to the UART IrDA modes of operation.

9.5.10 Programmable Baud Rate Generator

The UART modem/IrDA module contains a programmable baud generator and a set of fixed dividers that take the 48-MHz clock input and divide it down to the expected baud rate.

The baud rate generator and associated controls are depicted in Figure 9–17.

Figure 9–17. Baud Rate Generator



Before Modifying Clock Parameters

It is recommended that MODE_SELECT = DISABLE (MDR1[2:0] = 111) be set before attempting to initialize or modify clock parameters controls (DLH, DLL). Nonobservance of this rule can result in an unpredictable behavior of the module.

Choosing the appropriate divisor value:

- ❏ Modem 16x mode: Divisor value = Operating Freq/(16x baud rate)
- ❏ Modem 13x mode:Divisor value = Operating Freq/(13x baud rate)
- ❏ SIR mode: Divisor value = Operating Freq/(16x baud rate)
- ❏ MIR mode:Divisor value = Operating Freq/(41x/42x baud rate)
- ❏ FIR mode:Divisor value = None

Table 9–51. UART BAUD Rate Settings (48-MHz Clock)

Baud Rate (b/s)	Baud Multiple	DLH,DLL (Decimal)	DLH,DLL (Hex)	Actual Baud Rate (b/s)	Error (%)[1]
0.3 K	16 x	10000	0x27, 0x10	0.3 K	0
0.6 K	16 x	5000	0x13, 0x88	0.6 K	0
1.2 K	16 x	2500	0x09, 0xC4	1.2 K	0
2.4 K	16 x	1250	0x04, 0xE2	2.4 K	0
4.8 K	16 x	625	0x02, 0x71	4.8 K	0
9.6 K	16 x	313	0x01, 0x39	9.6153 K	+0.16
14.4 K	16x	208	0x00, 0xD0	14.423 K	+0.16
19.2 K	16 x	156	0x00, 0x9C	19.231 K	+0.16
28.8 K	16 x	104	0x00, 0x68	28.846 K	+0.16
38.4 K	16 x	78	0x00, 0x4E	38.462 K	+0.16
57.6 K	16 x	52	0x00, 0x34	57.692 K	+0.16
115.2 K	16 x	26	0x00, 0x1A	115.38 K	+0.16
230.4 K	16 x	13	0x00, 0x0D	230.77 K	+0.16
460.8 K	13 x	8	0x00, 0x08	461.54 K	+0.16
921.6 K	13 x	4	0x00, 0x04	923.08 M	+0.16
1.8342 M	13 x	2	0x00, 0x02	1.1846 M	+0.16
3.6864 M	13 x	1	0x00, 0x01	3.6923 M	+0.16

Table 9–52. IrDA Baud Rate Settings (48-MHz Clock)

Baud Rate (b/s)	IR Mode	Baud Multi-ple	En-coding	DLH, DLL	Actual Baud Rate (* = Avg) (b/s)	Error (%)	Source Jitter (%)[1]	Pulse Duration
2.4 K	SIR	16x	3/16	1250	2.4 K	0	0	78.1 µs
9.6 K	SIR	16x	3/16	312	9.6153 K	+0.16	0	19.5 µs
19.2 K	SIR	16x	3/16	156	19.231 K	+0.16	0	9.75 µs
38.4 K	SIR	16x	3/16	78	38.462 K	+0.16	0	4.87 µs
57.6 K	SIR	16x	3/16	52	57.692 K	+0.16	0	3.25 µs
115.2 K	SIR	16x	3/16	26	115.38 K	+0.16	0	1.62 µs
0.576 M	MIR	41x/42 x	1/4	2	0.5756 M*	0	+1.63/ –0.80	416 ns
1.152 M	MIR	41x/42 x	1/4	1	1.1511 M*	0	+1.63/ –0.80	208 ns
4 M	FIR	6x	4 PPM	–	4 M	0	0	125 ns

9.5.11 Hardware Flow Control

Hardware flow control is composed of auto-CTS and auto-RTS. Auto-CTS and auto-RTS can be enabled/disabled independently by programming EFR[7:6].

With auto-CTS, $\overline{\text{CTS}}$ must be active before the module can transmit data.

Auto-RTS only activates the $\overline{\text{RTS}}$ output when there is enough room in the FIFO to receive data and deactivates the $\overline{\text{RTS}}$ output when the RX FIFO is sufficiently full. The HALT and RESTORE trigger levels in the TCR determine the levels at which $\overline{\text{RTS}}$ is activated/deactivated.

If both auto-CTS and auto-RTS are enabled, data transmission does not occur unless the receiver FIFO has empty space. Thus, overrun errors are eliminated during hardware flow control. If they are not enabled, overrun errors occur when the transmit data rate exceeds the receive FIFO latency.

9.5.11.1 Auto-RTS

Auto-RTS data flow control originates in the receiver block (see Figure 9–2, Functional Block Diagram). The receiver FIFO trigger levels used in auto-RTS are stored in the TCR. $\overline{\text{RTS}}$ is active if the RX FIFO level is below the HALT trigger level in TCR[3:0]. When the receiver FIFO HALT trigger level is reached, $\overline{\text{RTS}}$ is deasserted. The sending device (for example, another UART) may send an additional byte after the trigger level is reached because it may not recognize the deassertion of $\overline{\text{RTS}}$ until it has begun sending the additional byte. $\overline{\text{RTS}}$ is automatically reasserted once the receiver FIFO reaches the RESUME trigger level programmed via TCR[7:4]. This reassertion requests the sending device to resume transmission.

9.5.11.2 Auto-CTS

The transmitter circuitry checks $\overline{CTS}$ before sending the next data byte. When $\overline{CTS}$ is active, the transmitter sends the next byte. To stop the transmitter from sending the following byte, $\overline{CTS}$ must be deasserted before the middle of the last stop bit that is currently being sent. The auto-CTS function reduces interrupts to the host system. When auto-CTS flow control is enabled, the $\overline{CTS}$ state changes need not trigger host interrupts because the device automatically controls its own transmitter. Without auto-CTS, the transmitter sends any data present in the transmit FIFO and a receiver overrun error can result.

9.5.12 Software Flow Control

The enhanced feature register (EFR) and the modem control register (MCR) enable software flow control. Different combinations of software flow control are enabled by setting different combinations of EFR[3:0].

Two other enhanced features relate to software flow control:

- XON-any function (MCR[5]): The operation resumes after receiving any character, after recognizing the XOFF character.

  The XON-any character is written into the RX FIFO even if it is a software flow character.

- Special character (EFR[5]): Incoming data is compared to XOFF2. Detection of the special character sets the XOFF interrupt (IIR[4]) but does not halt transmission. A read of the IIR clears the XOFF interrupt. The special character is transferred to the RX FIFO.

9.5.12.1 Receive (RX)

When the software flow control operation is enabled, the UART compares incoming data with XOFF1/2 programmed characters (in certain cases XOFF1 and XOFF2 must be received sequentially). When the correct XOFF characters are received, transmission is halted after completing transmission of the current character. XOFF detection also sets IIR[4] (if enabled via IER[5]) and causes UART_nIRQ to go low.

To resume transmission an XON1/2 character must be received (in certain cases XON1 and XON2 must be received sequentially). When the correct XON characters are received, IIR[4] is cleared and the XOFF interrupt disappears.

If a parity, framing, or break error occurs while receiving a software flow control character, the character is treated as normal data and is written to the RX FIFO.

When XON-any and special character detect are disabled, and software flow control is enabled, no valid XON or XOFF characters are written to the RX FIFO. For example, in EFR[1:0] = 10, if the XON1 and XOFF1 characters are received they are not written to the RX FIFO.

In the case where pairs of software flow characters are programmed to be received sequentially (EFR[1:0] = 11), the software flow characters are not

written to the RX FIFO if they are received sequentially. However, received XON1/XOFF1 characters must be written to the RX FIFO if the subsequent character is not XON2/XOFF2.

9.5.12.2 Transmit (TX)

XOFF1: Two characters are transmitted when the RX FIFO passes the programmed trigger level TCR[3:0].

XON1: Two characters are transmitted when the RX FIFO reaches the programmed trigger level via TCR[7:4].

After an XOFF character has been sent, software flow control is disabled and the module transmits XON characters automatically to enable normal transmission to proceed.

The transmission of XOFF/XON (s) follows the same protocol as transmission of an ordinary byte from the FIFO. This means that even if the word length is set to be 5, 6, or 7 characters, then the 5, 6, or 7 least-significant bits of XOFF1,2/XON1,2 are transmitted. The transmission of 5, 6, or 7 bits of a character is seldom done, but this functionality is included to maintain compatibility with earlier designs.

It is assumed that software flow control and hardware flow control are never enabled simultaneously.

9.5.13 Autobauding Mode

In autobaud mode, UART extracts transfer characteristics (speed, length, and parity) from an AT command. These characteristics are used to receive data following an *at* and to send data.

Valid AT commands are:

AT DATA <CR>
AT DATA <CR>
A/
a

The line break during the acquisition of the sequence AT is not recognized and echo functionality is not implemented in hardware.

A/ and *a/* are not used to extract characteristics, but they must be recognized because of their special meaning. They are used to instruct the software to repeat the last received AT command. Therefore, an *a/* always comes after an AT, and transfer characteristics are not expected to change between an AT and an *a/*.

As soon as a valid *at* (AT) is received, it and all subsequent data are saved into FIFO, including final CR (0x0D). Then, the autobaud state machine waits for the next valid AT command. If an *a/* (AI) is received, it is saved into FIFO and the state machine waits for next valid AT command.

Upon the first successful detection of the baud rate, the UART activates an interrupt to signify that the AT(upper- or lowercase) sequence has been

detected. The UASR register reflects the correct settings for the baud rate that has been detected. The interrupt activity continues in this fashion each time a subsequent character is received. Therefore, it is recommended that the software enable the RHR interrupt when using the autobaud mode.

The following settings are detected in autobaud mode with a module clock of 48 MHz:

- Speed:115.2K bauds, 57.6K bauds, 38.4K bauds, 28.8K bauds, 19.2K bauds, 14.4K bauds, 9.6K bauds, 4.8K bauds, 2.4K bauds, or 1.2K bauds
- Length: 7 or 8 bits
- Parity: Odd, even, or space

Note:

The combination of 7-bit character + space parity is not supported.

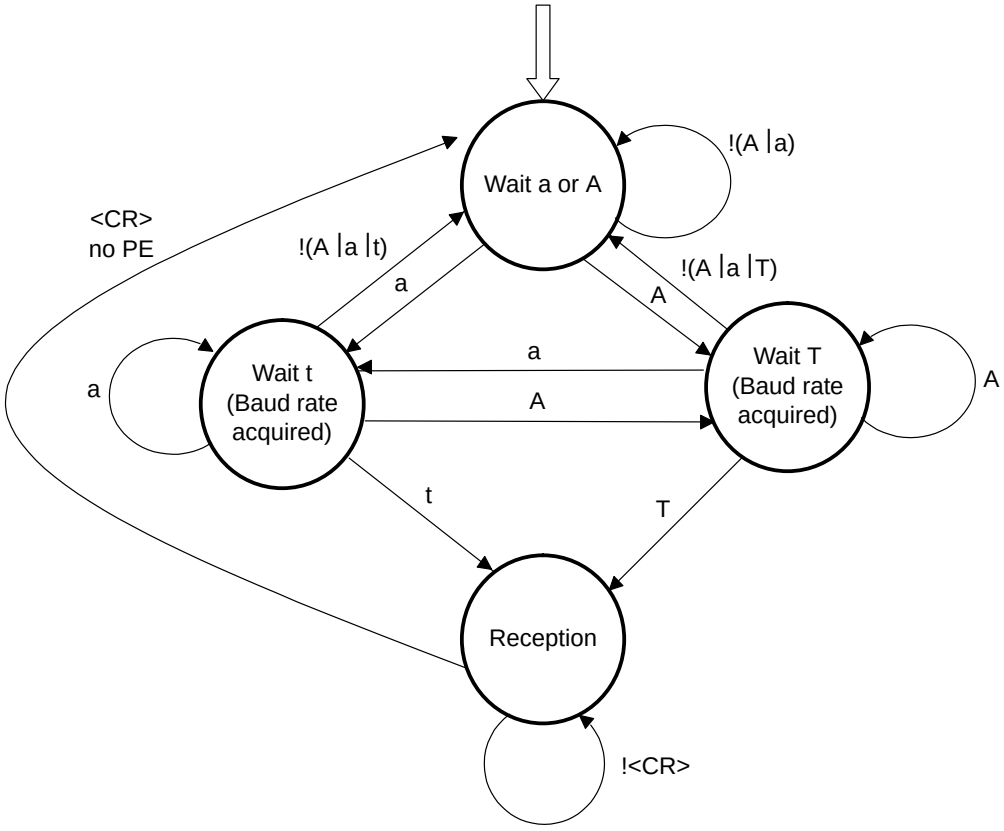
The method used to identify the speed is:

1) Detect the transition 1 $\rightarrow$ 0 on the received data. This happens as soon as a stop-to-start bit transition occurs. The transition is valid after a majority vote on 3 sampling periods.
2) Sample the start bit duration with 115 200 *16-Hz clock frequency as long as there is no rising edge. A transition 0 $\rightarrow$ 1 is considered as valid after a majority vote on 3 sampling periods.
3) Compare the sampled value with a table. If the sampled value is outside a valid range, an error is reported (no speed identified) and the hardware goes back to the first state (1).
4) Else, store the first data bit in the received register (for serial to parallel conversion) and go to frame format identification.

The next received bits are sampled according to the programmed baud rate. However, after receiving 7 bits, the speed identification must be restarted because several *a* or *A* characters may have been received before a valid *t* or *T* character.

The autobauding mode is selected when MDR1[2:0] = 010. In the UART auto-bauding mode, DLL, DLH, and LCR[5:0] settings are not used. Instead, UASR is updated with the configuration detected by the autobauding logic.

Figure 9–18. Autobaud State Machine



9.5.14 Frame Closing

A transmission frame is properly terminated in one of two ways:

- Frame-length method: The frame-length method is selected when MDR1[7] = 0. The local host writes the frame-length value to the TXFLH and TXFLL registers. The device automatically attaches end flags to the frame once the number of bytes transmitted becomes equal to the frame-length value.
- Set-EOT bit method: The set-EOT bit method is selected when MDR1[7] = 1. The local host writes 1 to ACREG[0] (EOT bit) just before it writes the last byte to the TX FIFO. When the local host writes the last byte to the TX FIFO, the device internally sets the tag bit for that particular character in the TX FIFO. As the TX state machine reads data from the TX FIFO it uses this tag-bit information to attach end flags and properly terminate the frame.

9.5.15 Store and Controlled Transmission (SCT)

In SCT, the local host first starts writing data into the TX FIFO. Then, after it writes a part of a frame (for a bigger frame) or a whole frame (a small frame, that is, supervisory frame), it writes a 1 to ACREG[2] (deferred TX start) to start transmission. SCT is enabled when MDR1[5] = 1. This method of transmission

differs from the normal mode, where transmission of data starts immediately after data is written to the TX FIFO. SCT is useful for sending short frames without TX underrun.

9.5.16 Underrun During Transmission

Underrun in transmission occurs when the TX FIFO becomes empty before the end of the frame is transmitted. When underrun occurs, the device closes the frame with end flags but attaches an incorrect CRC value. The receiving device detects a CRC error and discards the frame; it can then ask for a re-transmission. Underrun also causes an internal flag to be set, which disables further transmission. Before the next frame can be transmitted, the system (LH) must:

1) Reset the TX FIFO.
2) Read the RESUME register. This clears the internal flag.

This functionality is disabled with ACREG[4], compensated by the extension of the stop bit in transmission in case the TX FIFO is empty.

9.5.17 Overrun During Receive

Overrun occurs during receive if the RX state machine tries to write data into the RX FIFO when it is already full. When overrun occurs, the device interrupts the local host with IIR[3] and discards the remaining portion of the frame. Overrun also causes an internal flag to be set, which disables further reception. Before the next frame can be received the system (LH) must:

1) Reset the RX FIFO.
2) Read the RESUME register. This clears the internal flag

9.5.18 Status FIFO

In IrDA modes, a status FIFO is used to record the received frame status. When a complete frame is received, the length of the frame and the error bits associated with it are written into the status FIFO.

The frame length and error status can be determined by reading SFREGL/H and SFLSR. Reading the SFLSR causes the read pointer to be incremented. The status FIFO is eight entries deep and, therefore, can hold the status of eight frames.

The LH uses the frame-length information to locate the frame-boundary in the received frame data. The LH screens bad frames using the error-status information and later requests the sender to resend only the bad frames.

This status FIFO is used effectively in DMA, as the LH need not be interrupted each time a frame is received but only when the programmed status FIFO trigger level is reached.

9.6 UART Configuration Example

This section outlines the programming stages for operating one UART module with FIFO, interrupt, and no DMA capabilities. This is a three-step procedure that ensures a quick start of these modules (obviously, it does not cover every UART module feature). The first stage covers the software reset of the module (interrupts, status, and controls). The second stage deals with FIFO configuration and enable. The last stage with deals with the baud rate data and stop configuration. The procedure below is programming language agnostic.

9.6.1 UART Software Reset

Goal:

To clear IER and MCR registers, remove UART breaks (LCR[6] = 0) and put module in reset (MDR1[2:0] = 0x07).

Procedure:

To write into both the IER and MCR register EFR[4] must first be set to 1.

To be able to access the EFR register, 0xBF must be first be written to LCR register. Hence,

1) LCR = 0xBF: First write to the LCR register.
2) EFR[4] = 1: When LCR = 0xBF, enable the enhanced feature register.
3) LCR[7] = 0: Here, access to IER and MCR is allowed.
4) IER = 0x00: Disable interrupt.
5) MCR = 0x00: Force control signals inactive.
6) LCR[6] = 0: Here, remove UART breaks.
7) MDR1 = 0x07: Here, UART is in reset or disabled.

Alternatively, the SYSC[1] can be set to one to instigate a hardware reset from the generic synchronous reset module. The reset progress can be monitored via the SYSS[0]. Once complete, the above sequence ensures that the UART is in the equivalent disabled mode with reference to MDR1[2:0].

9.6.2 UART FIFO Configuration

Goal:

To set trigger level for halt/restore (TCR register), set trigger level for transmit/receive (TLR register), and configure FIFO (FCR register).

Procedure:

To write into both the TLR and TCR registers, EFR[4] must be set to 1 and MCR[6] to 1. To write into FCR, EFR[4] must be set to 1. Note that EFR[4] = 1 was already done in the previous section. Hence, a simple write to MCR[6] is necessary.

MCR[6] = 1; Sets TCR TLR and FCR to the desired value.

Here accesses to TCR, TLR, and FCR must be disabled to avoid any further undesired write to these registers. Hence,

- ❏ LCR = 0xBF: Provides access to EFR
- ❏ EFR[4] = 0
- ❏ LCR[7] = 0
- ❏ MCR[6] = 0

9.6.3 Baud Rate Data and Stop Configuration

Goal:

To configure UART data, stop (LCR register) baud rate (DLH and DLL registers), and enable UART operation. In case interrupt capability is added, configuration must be added just before UART enable.

Procedure:

- ❏ Set LCR to desired value.

  LCR[7] to 1: Gives access to DLH and DLL registers

- ❏ Set DLH and DLL.

  LCR[7] = 0: Removes access to DLH and DLL registers

- ❏ Set IER to desired value. Sets interrupts.

  MDR1[2:0] = 0: Enables UART without autobauding

The UART module is operational.

Chapter 10

Universal Serial Bus

This chapter describes the universal serial bus (USB) host on the OMAP730 multimedia processor.

Topic **Page**

10.1 Overview

The OMAP730 processor provides several varieties of USB functionality, including:

- USB host: OMAP730 provides a one-port *USB Specification Revision 1.1*-compliant host controller, which is based on the *OHCI Specification for USB Release 1.0a*.
- USB device: OMAP730 provides a full-speed USB device.
- USB On-The-Go (OTG): OMAP730 acts as an OTG dual-role device; the USB device functionality and one port of the USB host controller act in concert to provide an OTG port.

Flexible multiplexing of signals from the OMAP730 USB host controller, the OMAP730 USB function controller, and other OMAP730 peripherals allow a wide variety of system-level USB capabilities. Many of the OMAP730 pins can be used for USB-related signals or for signals from other OMAP730 peripherals. The OMAP730 top-level pin multiplexing controls each pin individually to select one of several possible internal pin signal interconnections. When these shared pins are programmed for use as USB signals, the OMAP730 USB signal multiplexing selects how the signals associated with the OMAP730 USB host port and the OMAP730 USB function controller can be brought out to OMAP730 pins.

10.2 USB Host Controller

The OMAP730 USB host controller (HC) is a one-port controller that communicates with USB devices at the USB low-speed (1.5M bit-per-second maximum) and full-speed (12M bit-per-second maximum) data rates. It is compatible with the *Universal Serial Bus Specification Revision 2.0* and the *Open HCI—Open Host Controller Interface Specification for USB,* Release 1.0a*,* available through the Compaq Computer Corporation web site, and hereafter called the *OHCI Specification for USB.* It is assumed that users of the OMAP730 USB host controller are already familiar with the *USB Specification* and *OHCI Specification for USB*.

When used for an OTG connection, the host controller port acts as the upstream device when OMAP730 controls the OTG link, and the USB function controller acts as the downstream device when OMAP730 acts as an OTG downstream device. The OMAP730 OTG controller is described in Section 15.4, *USB OTG Controller*.

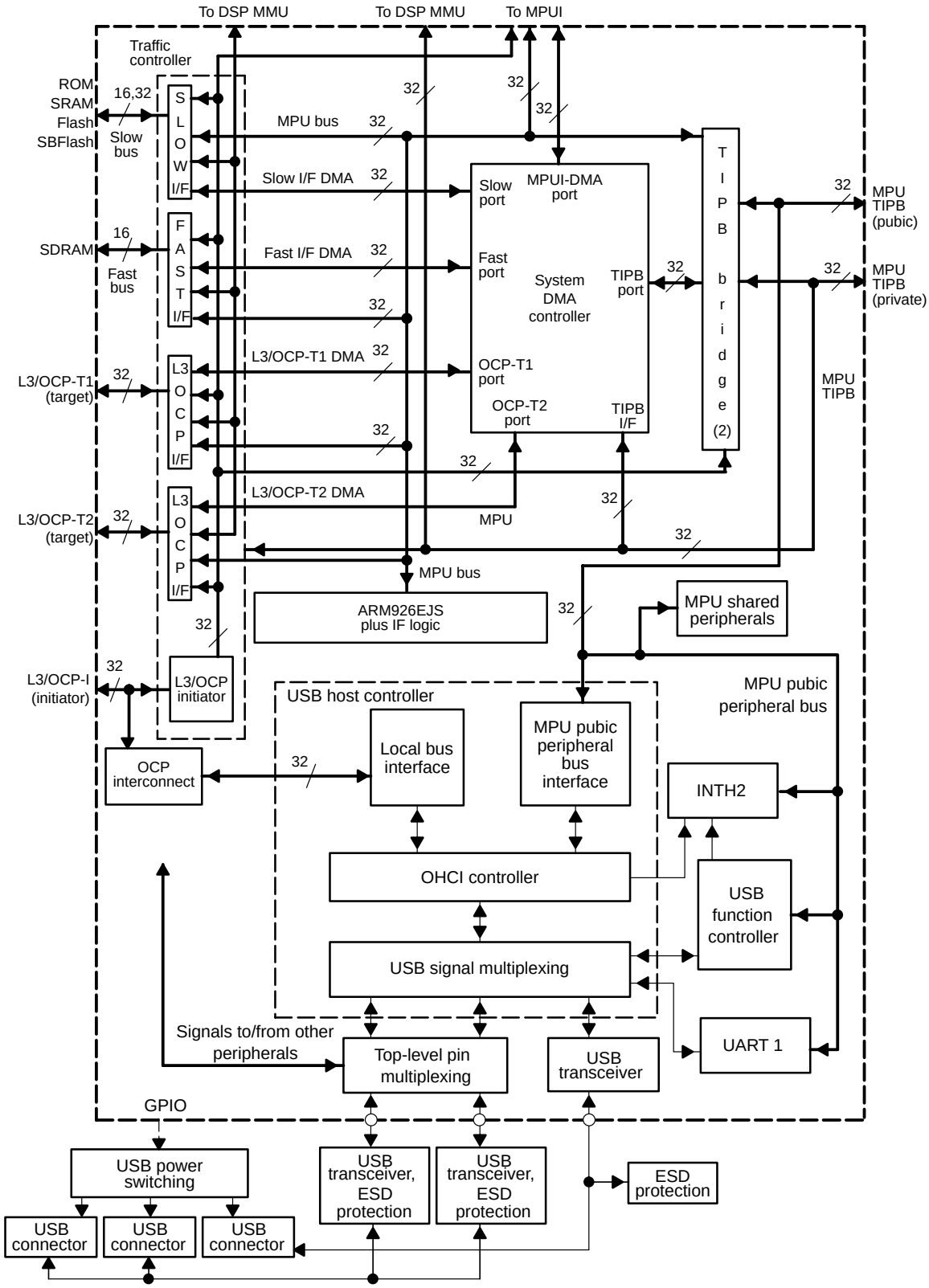
The OMAP730 USB host controller implements the register set and makes use of the memory data structures defined in the *OHCI Specification for USB*. These registers and data structures are the mechanism by which a USB host controller driver software package can control the OMAP730 USB host controller. The *OHCI Specification for USB* also defines how the USB host controller implementation must interact with those registers and data structures in system memory. The OMAP730 MPU accesses these registers via the OMAP730 MPU public peripheral bus.

To reduce processor software and interrupt overhead, the USB host controller generates USB traffic based on data structures and data buffers stored in system memory. The OMAP730 USB host controller accesses these data structures without direct intervention by the processor using the OMAP730 open-core protocol (OCP) bus. These data structures and data buffers can be located in internal or external system RAM.

The USB host controller provides an interrupt to the MPU level 2 interrupt handler to signal certain hardware events to the host controller driver software.

Figure 10–1 shows the OMAP730 USB host controller.

Figure 10–1. USB Host Controller



10.2.1 USB Open Host Controller Interface Functionality

10.2.1.1 OHCI Controller Overview

The *Open HCI—Open Host Controller Interface Specification for USB*, Release 1.0a, defines a set of registers and data structures stored in system memory that define how a USB host controller interfaces to system software. This specification, in conjunction with the *Universal Serial Bus Specification Version 2.0*, defines most of the USB functionality that the OMAP730 USB host controller provides.

The *OHCI Specification for USB* focuses on two main aspects of the hardware implementation of a USB host controller: its register set and the memory data structures that define the activity to appear on the USB bus. Also discussed are issues such as interrupt generation, USB host controller state, USB frame management, and the methods that the hardware must use to process the lists of data structures in system memory.

This document does not duplicate the information presented in the *OHCI Specification for USB* or the *USB Specification*. OMAP730 USB host controller users can refer to the *USB Specification* and the *OHCI Specification for USB* for detailed discussions of USB requirements and OHCI controller operation.

10.2.2 OMAP730 USB Host Controller Differences From OHCI Specification for USB

The OMAP730 USB host controller implementation does not implement every aspect of the functionality defined in the *OHCI Specification for USB*. The differences focus on power switching, overcurrent reporting, and the OHCI ownership change interrupt. Other restrictions are imposed by the effects of OMAP730 pin multiplexing options.

10.2.2.1 Power Switching Output Pins Not Supported

The OMAP730 device does not provide pins that can be controlled directly by the USB host controller OHCI port power control features. The OHCI $\overline{\text{RHPORTSTATUS}}$ register port power control bits can be programmed by the USB host controller driver software, but this does not have any direct effect on any VBUS switching implemented on the board.

Users can use software control of GPIO pins or other implementation-specific control mechanisms to control VBUS switching.

10.2.2.2 Overcurrent Protection Input Pins Not Supported

The OMAP730 device does not provide any pins that allow the USB host controller OHCI $\overline{\text{RHPORTSTATUS}}$ overcurrent protection status bits to be directly controlled by external hardware.

Users can use software monitoring of GPIO pins or other implementation-specific control mechanisms to report port overcurrent information to the USB host controller driver.

10.2.2.3 HMC_MODE and Top-Level Pin Multiplexing and OHCI Registers

The USB signal multiplexing modes selected by HMC_MODE provide selections where the USB host controller port can be brought to OMAP730 pins. The OHCI RHDESCRIPTORA register always reports one available USB host port, regardless of the HMC_MODE setting (see Table 10–20) or top-level pin multiplexing settings. When the HMC_MODE setting disables a USB host controller port, the USB host controller sees that port as unattached.

When OMAP730 top-level pin multiplexing configures a pin for functionality other than the USB, the USB host controller is disconnected from that pin and that pin does not affect the USB host controller.

10.2.2.4 No Ownership Change Interrupt

The OMAP730 USB host controller does not implement the OHCI ownership change interrupt.

10.2.3 OMAP730 Implementation of *OHCI Specification for USB*

10.2.3.1 Isochronous Transmit Descriptor (TD) OFFSETX/PSWX Values

The OMAP730 USB host controller implements the *OHCI Specification for USB* optional feature of checking isochronous OFFSETX/PSWX values. If either OFFSETX or OFFSET(X+1) does not have a condition code of *Not Accessed*, or if the OFFSET(X+1) value is not greater than or equal to OFFSETX, then an unrecoverable error is reported. Unrecoverable errors issued for these reasons do not cause an update of the HOSTUEADDR, HOSTUESTATUS, or HOSTTIMEOUTCTRL registers.

10.2.3.2 OMAP730 USB Host Controller Endpoint Descriptor (ED) List Head Pointers

The OHCI *Specification for USB* provides a specific sequence of operations for the host controller driver to perform when setting up the host controller. Failure to follow that sequence can result in malfunction. As a specific example, the HCCONTROLHEADED and HCBULKHEADED pointer registers and the 32 HCCAINTERRUPTTABLE pointers must all point to valid physical addresses of valid endpoint descriptors.

The OMAP730 USB host controller does not check HCCONTROLHEADED registers, HCBULKHEADED registers, or the values in the 32 HCCAINTERRUPTTABLE pointers before using them to access EDs. If any of these pointers are NULL when the corresponding list enable bit is set, the OMAP730 USB host controller attempts to access using the physical address of 0, which causes an unrecoverable error to be signaled. HOSTUEADDR, HOSTUESTATUS, and HOSTTIMEOUTCTRL registers are updated in this case.

10.2.3.3 OHCI USB Suspend State

The OMAP730 USB host controller ignores upstream traffic from downstream devices for about 3 ms after the host controller state (HCCONTROL.HCFS) changes from USB resume state to USB operational state. If any TDs cause generation of downstream packets during that time, the downstream packets

are sent, but any response provided by the downstream device is ignored. Any such TDs are aborted with completion codes marked as *Device Not Responding*. TDs on any of the lists (periodic, control, bulk, and isochronous) can cause such an occurrence.

The USB specification requires that system software must provide a 10-ms resume recovery time (T_{RSMRCY}) after a bus segment transitions from resume signaling to normal operational mode. During that time, only start of frame packets are to be sent on the bus segment. It is recommended that system software disable all list enable bits (HCCONTROL.PLE, HCCONTROL.IE, HCCONTROL.CLE, and HCCONTROL.BLE) and then wait for at least 1 ms before setting the host controller into USB suspend state (via HCCONTROL.HCFS). When restoring from suspend, system software must set the host controller into USB resume state, and wait for the host controller to transition into USB operational state. System software must then wait 10 ms before enabling the host controller list enable bits.

When the host controller has been placed into the USB suspend state under software control, but is brought out by a remote wake-up, system software must monitor the HCRHPORTSTATUS[x].PSS and HCRHPORTSTATUS[x].PSSC bits. The HCRHPORTSTATUS[x].PSS bit changes to 0 only after completion of resume signaling on the bus segment completes and completion of the 3-ms period where packets from downstream devices are ignored.

When using port-specific suspend, it is not necessary to disable the host controller lists so long as there are no active EDs and TDs directed toward devices that are downstream of the suspended port. For port-specific suspend operations, the host controller does not issue a root hub status change interrupt with the HCRHPORTSTATUS[n].PSSC bit = 1 and HCRHPORTSTATUS[n].PSS = 0 until after the approximately 3-ms delay after resume signaling completes.

When using port-specific suspend, system software must ensure that there are no active EDs for devices that are downstream of the suspended port before setting the port into suspend mode. While the port is in suspend or being resumed, system software must not enable any EDs for any devices downstream of the suspended port. Once the root hub status change interrupt occurs as a result of the suspended port PSS bit changing to 0, EDs can be enabled for devices downstream of the port that is now operational.

10.2.4 USB Host Controller Registers

Most of the OMAP730 host controller (HC) registers are the OHCI operational registers, which are defined by the *OHCI Specification for USB*. Four additional registers not specified by the *OHCI Specification for USB* provide additional information about the USB host controller state. USB host controller registers can be accessed in user and supervisor modes.

Table 10–1 lists the 32-bit OMAP730 USB host controller registers. Table 10–2 through Table 10–27 describe specific register bits.

Table 10–1. USB Host Controller Registers

Name	Description	R/W	Size†	Address (FFFB:)
HCREVISION	OHCI revision number	R	32	A000h
HCCONTROL	HC operating mode	R/W	32	A004h
HCCOMMANDSTATUS	HC command and status	R/W	32	A008h
HCINTERRUPTSTATUS	HC interrupt status	R/W	32	A00Ch
HCINTERRUPTENABLE	HC interrupt enable	R/W	32	A010h
HCINTERRUPTDISABLE	HC interrupt disable	R/W	32	A014h
HCHCCA	Physical address of HCCA‡	R/W	32	A018h
HCPERIODCURRENTED	Physical address of current periodic endpoint descriptor‡	R/W	32	A01Ch
HCCONTROLHEADED	Physical address of head of control endpoint descriptor list‡	R/W	32	A020h
HCCONTROLCUR-RENTED	Physical address of current control endpoint descriptor‡	R/W	32	A024h
HCBULKHEADED	Physical address of head of bulk end-point descriptor list‡	R/W	32	A028h
HCBULKCURRENTED	Physical of current bulk endpoint descriptor‡	R/W	32	A02Ch
HCDONEHEAD	Physical address of head of list of retired transfer descriptors‡	R	32	A030h
HCFMINTERVAL	HC frame interval	R/W	32	A034h
HCFMREMAINING	HC frame remaining	R	32	A038h
HCFMNUMBER	HC frame number	R	32	A03Ch
HCPERIODICSTART	HC periodic start	R/W	32	A040h
HCLSTHRESHOLD	HC low speed threshold	R/W	32	A044h
HCRHDESCRIPTORA	HC root hub A	R, R/W	32	A048h
HCRHDESCRIPTORB	HC root hub B	R/W	32	A04Ch
HCRHSTATUS	HC root hub status	R, R/W	32	A050h
HCRHPORTSTATUS1	HC port 1 control and status§	R, R/W	32	A054h
Reserved	Reserved	None	32	A058h
Reserved	Reserved	None	32	FFFBA05Ch

† Access to these registers must be by 32-bit reads or 32-bit writes. Use of other access sizes may result in undefined operation.
‡ Restrictions apply to the physical addresses used in these registers. See Section 10.2.9, *Physical Addressing*.
§ This register provides control and status for the OMAP730 pins normally associated with USB port 0 (the integrated USB transceiver) for some HMC_MODE values.

Table 10–1. USB Host Controller Registers (Continued)

Name	Description	R/W	Size†	Address (FFFB:)
Reserved	Reserved	None		A060h to A0DFh
HOSTUEADDR	Host UE address	R	32	A0E0h
HOSTUESTATUS	Host UE status	R	32	A0E4h
HOSTTIMEOUTCTRL	Host time-out control	R/W	32	A0E8h
HOSTREVISION	Host revision	R	32	A0ECh
Reserved	Reserved	None		A0F0h to AFFFh

† Access to these registers must be by 32-bit reads or 32-bit writes. Use of other access sizes may result in undefined operation.
‡ Restrictions apply to the physical addresses used in these registers. See Section 10.2.9, *Physical Addressing*.
§ This register provides control and status for the OMAP730 pins normally associated with USB port 0 (the integrated USB transceiver) for some HMC_MODE values.

Table 10–2. OHCI Revision Number Register (HCREVISION)

Bit	Name	Description	Type	Reset Value
31:8	Reserved	Reserved		0x00 0000
7:0	REV	OHCI specification revision—the OHCI revision number upon which the USB host controller is based. Write has no effect.	R	0x10

The OCHI revision number register reports the revision number of the *OHCI Specification for USB* upon which the USB host controller is based.

Table 10–3. HC Operating Mode Register (HCCONTROL)

Bit	Name	Value	Description	Type	Reset Value
31:11	Reserved		Reserved		
10	RWE		Remote wake-up enable. This bit has no effect in OMAP730. The OMAP730 USB host controller does not provide a processor wake-up mechanism.	R/W	0
9	RWC		Remote wake-up connected. This bit has no effect in OMAP730. The OMAP730 USB host controller does not provide a processor wake-up mechanism.	R/W	0
8	IR		Interrupt routing. The OMAP730 USB host controller does not provide an SMI interrupt. This bit must be 0 to allow the USB host controller interrupt to propagate to the MPU level 2 interrupt controller.	R/W	0
7:6	HCFS		Host controller functional state:	R/W	00
		00	USB reset		
		01	USB resume		

Table 10–3. HC Operating Mode Register (HCCONTROL)(Continued)

Bit	Name	Value	Description	Type	Reset Value
		10	USB operational		
		11	USB suspend A transition to USB operational causes SOF generation to begin in 1 ms. The USB host controller can automatically transition from USB suspend to USB resume if a downstream resume is received. The USB host controller enters USB suspend after a software reset. The USB host controller enters USB reset after a hardware reset. The USB reset state resets the root hub and causes downstream signaling of USB reset.		
5	BLE		Bulk list enable:	R/W	0
		0	Bulk ED list not processed in the next 1-ms frame. Host controller driver can modify the list. If driver removes the ED pointed to by the HCBULKCURRENTED from the ED list, it must update HCBULKCURRENTED to point to an ED still on the list before it re-enables the bulk list.		
		1	Enables processing of bulk ED List. HCBULKHEADED must be 0 or point to a valid ED before setting this bit. HCBULKCURRENTED must point to a valid ED or be 0 before setting this bit.		
4	CLE		Control list enable:	R/W	0
		0	Control ED list is not processed in the next 1-ms frame. Host controller driver can modify the control ED list. If driver removes the ED pointed to by the HCCONTROLCURRENTED from the ED list, it must update HCCONTROLCURRENTED to point to an ED still on the list before it re-enables the control list.		
		1	Enables processing of the control ED list. HCCONTROLHEADED must be 0 or point to a valid ED before setting this bit. HCCONTROLCURRENTED must be 0 or point to a valid ED before setting this bit.		
3	IE		Isochronous enable	R/W	0
		0	Isochronous EDs are not processed. The USB host controller checks this bit every time it finds an isochronous ED in the periodic list. When this bit is written to 1, processing of isochronous EDs can be enabled in the next frame, if not in the current frame.		
		1	Enables processing of isochronous EDs		
2	PLE		Periodic list enable	R/W	0

Table 10–3. HC Operating Mode Register (HCCONTROL)(Continued)

Bit	Name	Value	Description	Type	Reset Value
		0	The periodic ED lists are not processed. When written to 0, periodic list processing is disabled beginning with the next frame.		
		1	Enables processing of the periodic ED lists. When written to 1, periodic list processing begins in the next frame.		
1:0	CBSR		Control/bulk service ratio Specifies the ratio between control and bulk EDs processed in a frame	R/W	00
		00	One control ED per bulk ED		
		01	Two control EDs per bulk ED		
		10	Three control EDs per bulk ED		
		11	Four control EDs per bulk ED		

The HC operating mode register controls the operating mode of the USB host controller.

Table 10–4. HC Command and Status Register (HCCOMMANDSTATUS)

Bit	Name	Description	Type	Reset Value
31:18	Reserved	Reserved		
17:16	SOC	Scheduling overrun count Counts the number of times a scheduling overrun occurs. This count is incremented even if the host controller driver has not acknowledged any previous pending scheduling overrun interrupt.	R	00
15:4	Reserved	Reserved		
3	OCR	Ownership change request This bit is set by the host controller driver to gain ownership of the host controller. OMAP730 does not support SMI interrupts, so no ownership change interrupt occurs.	R/W	0
2	BLF	Bulk list filled The host controller driver must set this bit if it modifies the bulk list to include new TDs. If HCBULKCURRENTED is 0, the USB host controller does not begin processing bulk list EDs unless this bit is set. When the USB host controller sees this bit set and begins processing the bulk list, it clears this bit.	R/W	0

Table 10–4. HC Command and Status Register (HCCOMMANDSTATUS) (Continued)

Bit	Name	Description	Type	Reset Value
1	CLF	Control list filled The host controller driver must set this bit if it modifies the control list to include new TDs. If HCCONTROLHEADED is 0, the USB host controller does not begin processing control list EDs unless this bit is set. When the USB host controller sees this bit set and begins processing the control list, it clears this bit.	R/W	0
0	HCR	Host controller reset Write of 0 has no effect. 1: This bit initiates a software reset of the USB host controller. This transitions the USB host controller to the USB suspend state. This resets most USB host controller OHCI registers. OHCI register accesses must not be attempted until a read of this register returns a 0. A write of 1 to this bit does not reset the root hub and does not signal USB reset to downstream USB functions.	R/W	0

The HC command and status register shows the current state of the host controller and accepts commands from the host controller driver.

Table 10–5. HC Interrupt and Status Register (HCINTERRUPTSTATUS)

Bit	Name	Description	Type	Reset Value
31	Reserved	Reserved		
30	OC	Ownership change The OMAP730 USB host controller does not implement ownership change interrupts.	R	0
29:7	Reserved	Reserved		
6	RHSC	Root hub status change When 1, indicates a root hub status change has occurred. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
5	FNO	Frame number overflow When 1, indicates a frame number overflow has occurred. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0

Table 10–5. HC Interrupt and Status Register (HCINTERRUPTSTATUS) (Continued)

Bit	Name	Description	Type	Reset Value
4	UE	Unrecoverable error When 1, indicates that an unrecoverable error has occurred on the OCP bus or that an isochronous TD PSW field condition code was not set to *Not Accessed* when the USB host controller attempted to perform a transfer using that PSW/offset pair. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
3	RD	Resume detected When 1, indicates that a downstream device has issued a resume request. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
2	SF	Start of frame When 1, indicates that a SOF has been issued. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
1	WDH	Write done head When 1, indicates that the USB host controller has updated the HCDONEHEAD register. Write of 0 has no effect. Write of 1 clears this bit. The host controller driver must read the value from HCDONEHEAD before writing 1 to this bit.	R/W	0
0	SO	Scheduling overrun When 1, indicates that a scheduling overrun has occurred. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0

The HC interrupt status register reports the status of the USB host controller internal interrupt sources.

Table 10–6. HC Interrupt Enable Register (HCINTERRUPTENABLE)

Bit	Name	Description	Type	Reset Value
31	MIE	Master interrupt enable When 1, allows other enabled OHCI interrupt sources to propagate to the OMAP730 level 2 interrupt controller. When 0, OHCI interrupt sources are ignored and no USB host controller interrupts are propagated to the OMAP730 level 2 interrupt controller. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0
30	OC	Ownership change This bit has no effect on OMAP730.	R	0
29:7	Reserved	Reserved		
6	RHSC	Root hub status change When 1 and MIE is 1, allows root hub status change interrupts to propagate to the OMAP730 level 2 interrupt controller. When 0, or when MIE is 0, root hub status change interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0
5	FNO	Frame number overflow When 1 and MIE is 1, allows frame number overflow interrupts to propagate to the OMAP730 level 2 interrupt controller. When 0, or when MIE is 0, frame number overflow interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W:	0
4	UE	Unrecoverable error When 1 and MIE is 1, allows unrecoverable error interrupts to propagate to the OMAP730 level 2 interrupt controller. When 0, or when MIE is 0, unrecoverable error interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0

Table 10–6. HC Interrupt Enable Register (HCINTERRUPTENABLE) (Continued)

Bit	Name	Description	Type	Reset Value
3	RD	Resume detected When 1 and MIE is 1, allows resume detected interrupts to propagate to the OMAP730 level 2 interrupt controller. When 0, or when MIE is 0, resume detected interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0
2	SF	Start of frame When 1 and MIE is 1, allows start of frame interrupts to propagate to the OMAP730 level 2 interrupt controller. When 0, or when MIE is 0, start of frame interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0
1	WDH	Write done head When 1 and MIE is 1, allows write done head interrupts to propagate to the OMAP730 level 2 interrupt controller. When 0, or when MIE is 0, write done head interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0
0	SO	Scheduling overrun When 1 and MIE is 1, allows scheduling overrun interrupts to propagate to the OMAP730 level 2 interrupt controller. When 0, or when MIE is 0, scheduling overrun interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0

The HC interrupt enable register (Table 10–6) enables various OHCI interrupt sources to generate interrupts to the OMAP730 level 2 interrupt handler.

Table 10–7.HC Interrupt Disable Register (HCINTERRUPTDISABLE)

Bit	Name	Description	Type	Reset Value
31	MIE	Master interrupt enable Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE MIE bit.	R/W	0
30	OC	Ownership change This bit has no effect on OMAP730.	R	0
29:7	Reserved	Reserved		
6	RHSC	Root hub status change Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE RHSC bit.	R/W	0
5	FNO	Frame number overflow Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE FNO bit.	R/W	0
4	UE	Unrecoverable error Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE UE bit.	R/W	0
3	RD	Resume detected Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE RD bit.	R/W	0
2	SF	Start of frame Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE SF bit.	R/W	0
1	WDH	Write done head Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE WDH bit.	R/W	0
0	SO	Scheduling overrun Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE SO bit.	R/W	0

The HC interrupt disable register is used to clear bits in the HCINTERRUPTENABLE register.

Table 10–8. HC HCAA Address Register (HCHCCA)

Bit	Name	Description	Type	Reset Value
31:8	HCCA	Physical address of the beginning of the HCCA	R/W	0x000000
7:0	Reserved	Reserved	R	x00

The HCAA address register defines the physical address of the beginning of the HCCA.

Table 10–9. HC Current Periodic Register (HCPERIODCURRENTED)

Bit	Name	Description	Type	Reset Value
31:4	PCED	Physical address of current ED on the periodic ED list. This field represents bits 31:4 of the physical address of the next ED on the periodic ED List. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 10.2.9, *Physical Addressing*, for the restrictions on physical addresses.	R	0x0000000
3:0	Reserved	Reserved	R	0x0

The HC current periodic register defines the physical address of the next endpoint descriptor (ED) on the periodic ED List.

Table 10–10. HC Head Control Register (HCCONTROLHEADED)

Bit	Name	Description	Type	Reset Value
31:4	CHED	Physical address of head ED on the control ED list This field represents bits 31:4 of the physical address of the head ED on the control ED list. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 10.2.9, *Physical Addressing*, for the restrictions on physical addresses.	R/W	0x0000000
3:0	Reserved	Reserved	R	0x0

The HC head control register defines the physical address of the head ED of the control ED list.

Table 10–11. HC Current Control Register (HCCONTROLCURRENTED)

Bit	Name	Description	Type	Reset Value
31:4	CCED	Physical address of current ED on the control ED list This field represents bits 31:4 of the physical address of the next ED on the control ED list. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 10.2.9, *Physical Addressing*, for the restrictions on physical addresses. A value of 0x0000000 indicates that the USB host controller has reached the end of the control ED list without finding any transfers to process. This register is automatically updated by the USB host controller.	R/W	0x0000000
3:0	Reserved	Reserved	R	0x0

The HC current control register defines the physical address of the next ED on the control ED list.

Table 10–12. HC Head Bulk Register (HCBULKHEADED)

Bit	Name	Description	Type	Reset Value
31:4	BHED	Physical address of head ED on the bulk ED list This field represents bits 31:4 of the physical address of the head ED on the bulk ED list. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 10.2.9, *Physical Addressing*, for the restrictions on physical addresses.	R/W	0x0000000
3:0	Reserved	Reserved	R	0x0

The head bulk register defines the physical address of the head ED on the bulk ED list.

Table 10–13. HC Current Bulk Register (HCBULKCURRENTED)

Bit	Name	Description	Type	Reset Value
31:4	BCED	Physical address of current ED on the bulk ED list This field represents bits 31:4 of the physical address of the next ED on the bulk ED list. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 10.2.9, *Physical Addressing*, for the restrictions on physical addresses. A value of 0x0000000 indicates that the USB host controller has reached the end of the bulk ED list without finding any transfers to process. The USB host controller automatically updates this register.	R/W	0x0000000
3:0	Reserved	Reserved	R	0x0

The current bulk register defines the physical address of the next ED on the bulk ED list.

Table 10–14. HC Head Done Register (HCDONEHEAD)

Bit	Name	Description	Type	Reset Value
31:4	DH	Physical address of the last TD that has added to the done queue. This field represents bits 31:4 of the physical address of the top TD on the done TD queue. TDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. A value of 0x00000000 indicates that there are no TDs on the done queue. The USB host controller automatically updates this register.	R	0x0000000
3:0	Reserved	Reserved	R	0x0

The head done register defines the physical address of the current head of the done TD queue.

Table 10–15. HC Frame Interval Register (HCFMINTERVAL)

Bit	Name	Description	Type	Reset Value
31	FIT	Frame interval toggle The host controller driver must toggle this bit any time it changes the frame interval field.	R/W	0
30:16	FSMPS	Largest data packet Largest data packet size allowed for full-speed packets, in bit times.	R/W	0x0000
15:14	Reserved	Reserved		
13:0	FI	Frame interval Number of 12-MHz clocks in the USB frame. Nominally, this is set to 11,999, to give a 1-ms frame. The host controller driver can make minor changes to this field to attempt to manually synchronize with another clock source.	R/W	0x2EDF

The frame interval register defines the number of 12-MHz clock pulses in each USB frame.

Table 10–16. HC Frame Remaining Register (HCFMREMAINING)

Bit	Name	Description	Type	Reset Value
31	FRT	Frame remaining toggle This bit is loaded with the frame interval toggle bit every time the USB host controller loads the frame interval field into the frame remaining field.	R	0
30:14	Reserved	Reserved		
13:0	FR	Frame remaining The number of full-speed bit times remaining in the current frame. This field is automatically reloaded with the frame interval field value at the beginning of every frame.	R	0x0000

The HC frame remaining register reports the number of full-speed bit times remaining in the current frame.

Table 10–17. HC Frame Number Register (HCFMNUMBER)

Bit	Name	Description	Type	Reset Value
31:16	Reserved	Reserved		
15:0	FN	Frame number This field reports the current USB frame number. It is incremented when the frame remaining field is reloaded with the frame interval field value. Frame number automatically rolls over from 0xFFFF to 0x0000. After frame number is incremented, its new value is written to the HCCA and the USB host controller sets the SOF interrupt status bit and begins processing the ED lists.	R	0x0000

The HC frame number register reports the current USB frame number.

Table 10–18. HC Periodic Start Register (HCPERIODICSTART)

Bit	Name	Description	Type	Reset Value
31:14	Reserved	Reserved		
13:0	PS	Periodic start The host controller driver must program this value to be about 10% less than the frame interval field value so that control and bulk EDs have priority for the first 10% of the frame; then periodic EDs have priority for the remaining 90% of the frame.	R/W	0x0000

The HC periodic start register defines the position within the USB frame where EDs on the periodic list have priority over EDs on the bulk and control lists.

Table 10–19. HC Low-Speed Threshold Register (HCLSTHRESHOLD)

Bit	Name	Description	Type	Reset Value
31:14	Reserved	Reserved		
13:0	LST	Low-speed threshold This field defines the number of full-speed bit times in the frame after which the USB host controller cannot start an 8-byte low-speed packet. The USB host controller only begins a low-speed transaction if the frame remaining field is greater than the low-speed threshold. The host controller driver must set this field to a value that ensures that an 8-byte low-speed TD completes before the end of the frame. When set, the host controller driver must not change the value.	R/W	0x0628

The HC low-speed threshold register defines the latest time in a frame that the USB host controller can begin a low-speed packet.

Table 10–20. HC Root Hub A Register (HCRHDESCRIPTORA)

Bit	Name	Value	Description	Type	Reset Value
31:24	POTPG		Power-on to power-good time This field defines the minimum amount of time (2 ms × POTPG) between the USB host controller turning on power to a downstream port and when the USB host can access the downstream device. This field has no effect on USB host controller operation. After turning on power to a port, the USB host controller driver must delay the amount of time implied by POTPG before attempting to reset an attached downstream device. The required amount of time is implementation-specific and must be calculated based on the amount of time the VBUS supply takes to provide valid VBUS to a worst-case downstream USB function controller. The implementation-specific value must be computed and then written to this register before the USB host controller driver is initialized. Because OMAP730 does not provide a direct control from the USB host controller to switch VBUS on and off, this value must take into account any delays caused by other methods of controlling VBUS externally. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	0xA
23:13	Reserved		Reserved		

Table 10–20. HC Root Hub A Register (HCRHDESCRIPTORA) (Continued)

Bit	Name	Value	Description	Type	Reset Value
12	NOCP		No overcurrent protection	R/W	1
			When 1, this bit indicates that the USB host controller does not implement overcurrent protection inputs. OMAP730 does not provide signals to allow connection of external overcurrent indication signals to the USB host controller, so this bit defaults to 1.		
			This bit has no relationship to the OTG controller register bits that relate to VBUS.		
11	OCPM		Overcurrent protection mode	R/W	0
			OMAP730 does not provide host controller overcurrent protection input signals, so this bit has no effect. This bit has no relationship to the OTG controller register bits that relate to VBUS.		
10	DT		Device type	R	0
			This bit is always 0, which indicates that the USB host controller implemented is not a compound device.		
9	NPS		No power switching	R/W	1
		0	Indicates that VBUS power switching is supported and is either per-port or all-port switched per the power switching mode field.		
		1	Indicates that VBUS power switching is not supported and that power is available to all downstream ports when the USB host controller is powered.		
			Because OMAP730 does not provide connections from the USB host controller to control external VBUS switching, this bit defaults to 1.		
			This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.		
8	PSM		Power switching mode	R/W	0
		0	Indicates that all ports are powered at the same time		
		1	Indicates that individual port power switching is supported		
			Because OMAP730 does not provide signals from the USB host controller to control external VBUS switching, this bit defaults to 0.		

Table 10–20. HC Root Hub A Register (HCRHDESCRIPTORA) (Continued)

Bit	Name	Value	Description	Type	Reset Value
7:0	NDP		Number of downstream ports This register defaults to 3 to indicate three downstream ports. The USB signal multiplexing mode and OMAP730 top-level pin multiplexing features can place the OMAP730 device in a mode where 0, 1, 2, or 3 of the USB host controller downstream ports are usable. This register reports three ports, regardless of USB signal multiplexing mode and OMAP730 top-level pin multiplexing mode.	R	0x03

The HC root hub A register defines several aspects of the USB host controller root hub functionality.

Table 10–21. HC Root Hub B Register (HCRHDESCRIPTORB)

Bit	Name	Description	Type	Reset Value
31:16	PPCM	Port power control mask Each bit defines whether a corresponding downstream port has port power controlled by the global power control. If set, per-port power control is implemented for the corresponding port. If clear, global power control is implemented for the corresponding port. PPCM bit 0 is reserved. PPCM bit 1 is the port power control mask for downstream port 1. PPCM bit 2 is the port power control mask for downstream port 2. PPCM bit 3 is the port power control mask for downstream port 3. PPCM bits 4 through 15 are reserved. OMAP730 does not provide connections from the USB host controller to pins to provide external port power switching. Systems that implement port power switching must use other mechanisms to control port power. System software can update these bits to simplify host controller driver and/or OTG driver coding.	R/W	0x0000

Table 10–21. HC Root Hub B Register (HCRHDESCRIPTORB) (Continued)

Bit	Name	Description	Type	Reset Value
15:0	DR	Device removable	R/W	0x0000
		Each bit defines whether a corresponding downstream port has a removable or non-removable device. A cleared bit indicates the corresponding port may have a removable device attached. A set bit indicates that the corresponding port has a non-removable device attached.		
		DR bit 0 is reserved.		
		DR bit 1 is the device removable bit for downstream port 1.		
		DR bit 2 is the device removable bit for downstream port 2.		
		DR bit 3 is the device removable bit for downstream port 3.		
		DR bits 4 through 15 are reserved.		

The HC root hub B register defines several aspects of the USB host controller root hub functionality.

Table 10–22. HC Root Hub Status Register (HCRHSTATUS)

Bit	Name	Description	Type	Reset Value
31	CRWE	Clear remote wake-up enable	R/W	0
		Write of 0 has no effect.		
		Write of 1 clears the device remote wake-up enable bit.		
30:18	Reserved	Reserved		
17	OCIC	Overcurrent indication change	R/W	0
		This bit is automatically set when the overcurrent indicator bit changes.		
		Write of 0 has no effect.		
		Write of 1 clears this bit.		
		This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.		
16	LPSC	Local power status change	R/W	0
		This bit defaults to 0 because the root hub does not support the local power status feature.		
		Write of 0 has no effect.		
		Write of 1 sets the port power status bits for all ports if power switching mode is 0. A write of 1 sets port power status bits for ports with their corresponding port power control mask bits cleared if power switching mode is 1.		
		This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.		

Table 10–22. HC Root Hub Status Register (HCRHSTATUS) (Continued)

Bit	Name	Description	Type	Reset Value
15	DRWE	Device remote wake-up enable When 1, this bit enables a connect status change event to be treated as a resume event, which causes a transition from USB suspend to USB resume state and sets the resume detected interrupt status bit. When 0, connect status change events do not cause a transition from USB suspend to USB resume state and the resume detected interrupt is not changed. Write of 0 has no effect. Write of 1 sets the device remote wake-up enable bit.	R/W	0
14:2	Reserved	Reserved		
1	OCI	Overcurrent indicator This bit reports global overcurrent indication if global overcurrent reporting is selected. When 1, this bit indicates that an overcurrent condition has been sensed. When 0, no overcurrent condition has been sensed. Because OMAP730 does not provide signals for external hardware to report overcurrent status to the USB host controller, this bit is always 0. This bit has no relationship to the OTG controller register bits that relate to VBUS.	R	0
0	LPS	Local power status The root hub does not support the local power status feature. This bit always reads as 0. Write of 0 has no effect. Write of 1 when in global power mode (power switching mode = 0), turns off power to all ports. If in per-port power mode (power switching mode = 1), a write of 1 turns off power to those ports whose corresponding port power control mask bit is 0. Because OMAP730 does not provide signals from the USB host controller to external VBUS switching circuitry, this bit has no effect. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	0

The HC root hub status register reports the USB host controller root hub status.

Table 10–23 describes the HC port 1 status and control register, which reports and controls the state of USB host port 1. HCRHPORTSTATUS1 can provide status and control for the host controller port that is usually associated with the OMAP730 integrated USB transceiver and the associated OMAP730 pins:

- USB.DP
- USB.DM

Alternately, OMAP730 top-level pin multiplexing can also configure the host controller port associated with this register to provide status and control for a USB host port connected to the following pins:

- USB.VBUSI/USB.TXEN
- USB.DP/USB.TXD
- USB.DM/USB.SE0
- USB.PU.EN/USB.RCV.
- EAC.CRESET/USB.VP
- EAC.MCLK.OUT/USB.VM
- I2C.SCK/USB.SPEED
- I2C.SDA/USB.SUSPEND

The host controller port associated with this register can also be held in a state where it always appears to be disconnected, depending on the HMC_MODE value and top-level pin multiplexing. See Table 10–22 and 10.4.3, *Pin Multiplexing*.

HC port 1 can act as the host portion of an OTG controller. See Section 10.4, *USB OTG Controller*.

Table 10–23. HC Port 1 Status and Control Register (HCRHPORTSTATUS1)

Bit	Name	Description	Type	Reset Value
31:21	Reserved	Reserved		
20	PRSC	Port 1 reset status change This bit indicates, when 1, that the port 1 port reset status bit has changed. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
19	OCIC	Port 1 overcurrent indicator change This bit indicates, when 1, that the port 1 port overcurrent indicator has changed. Write of 0 has no effect. Write of 1 clears this bit. The OMAP730 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism. This bit has no relationship to the OTG controller register bits that relate to VBUS.	R/W	0

Table 10–23. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) (Continued)

Bit	Name	Description	Type	Reset Value
18	PSSC	Port 1 suspend status change This bit indicates, when 1, that the port 1 port suspend status has changed. Suspend status is considered to have changed only after the resume pulse, low-speed EOP, and 3-ms synchronization delays have been completed. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
17	PESC	Port 1 enable status change This bit indicates, when 1, that the port 1 port enable status changed. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
16	CSC	Port 1 connect status change This bit indicates, when 1, that the port 1 current connect status has changed due to a connect or disconnect event. If current connect status is 0 when a set port reset, set port enable, or set port suspend write occurs, then this bit is set. A write of 1 clears this bit. A write of 0 has no effect. If the HCRHDESCRIPTORB.DR[1] bit is set to indicate a non-removable USB device on port 1, this bit is set only after a root hub reset to inform the system that the device is attached.	R/W	0
15:10	Reserved	Reserved		
9	LSDA/CPP	Port 1 low-speed device attached/clear port power This bit indicates, when read as 1, that a low-speed device is attached to port 1. A 0 in this bit indicates a full-speed device. This bit is valid only when port 1 current connect status is 1. The host controller driver can write a 1 to this bit to clear the port 1 port power status. A write of 0 to this bit has no effect. OMAP730 USB host controller does not control external port power using OHCI mechanisms, so, if required, USB host port power must be controlled through other means. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	0

Table 10–23. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) (Continued)

Bit	Name	Description	Type	Reset Value
8	PPS/SPP	Port 1 port power status/set port power This bit indicates, when read as 1, that the port 1 power is enabled. When read as 0, port 1 power is not enabled. The OMAP730 does not provide signals from the USB host controller to control external port power, so if required, USB host port power control signals must be controlled through other means. Software can track the current power state using the port power status bit and other power control bits, but those bits have no direct effect on external port power control. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding. A write of 1 to this bit sets the port 1 port power status bit. A write of 0 has no effect. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	1
7:5	Reserved	Reserved		
4	PRS/SPR	Port 1 port reset status/set port reset When read as 1, indicates that port 1 is signalling the USB reset. When read as 0, USB reset is not being sent to port 1. A write of 1 to this bit sets the port 1 port reset status bit and causes the USB host controller to begin signaling USB reset to port 1. A write of 0 to this bit has no effect.	R/W	0
3	POCI/CSS	Port 1 port overcurrent indicator/clear suspend status When read as 1, indicates a port 1 port overcurrent condition has occurred. When 0, no port 1 port overcurrent condition has occurred. OMAP730 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism. A write of 1 to this bit when port 1 port suspend status is 1 causes resume signaling on port 1. A write of 1 when port 1 port suspend status is 0 has no effect. A write of 0 has no effect.	R/W	0

Table 10–23. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) (Continued)

Bit	Name	Description	Type	Reset Value
2	PSS/SPS	Port 1 port suspend status/set port suspend When read as 1, indicates that port 1 is in the USB suspend state or is in the resume sequence. When 0, indicates that port 1 is not in the USB suspend state. This bit is cleared automatically at the end of the USB resume sequence and also at the end of the USB reset sequence. If port 1 current connect status is 1, a write of 1 to this bit sets the port 1 port suspend status bit and places port 1 in USB suspend state. If current connect status is 0, a write of 1 instead sets connect status change to inform the USB host controller driver software of an attempt to suspend a disconnected device. A write of 0 to this bit has no effect.	R/W	0
1	PES/SPE	Port 1 port enable status/set port enable When read as 1, indicates that port 1 is enabled. When read as 0, this bit indicates that port 1 is not enabled. This bit is automatically set at completion of port 1 USB reset if it was not already set before the USB reset completed, and is automatically set at the end of a USB suspend if the port was not enabled when the USB resume completed. A write of 1 to this bit when port 1 current connect status is 1 sets the port 1 port enable status bit. A write of 1 when port 1 current connect status is 0 has no effect. A write of 0 has no effect.	R/W	0
0	CCS/CPE	Port 1 current connection status/clear port enable When read as 1, indicates that port 1 currently has a USB device attached. When 0, indicates that no USB device is attached to port 1. This bit is set to 1 after root hub reset if the HCRHDESCRIPTORB.DR[1] bit is set to indicate a non-removable device on port 1. A write of 1 to this bit clears the port 1 port enable bit. A write of 0 to this bit has no effect.	R/W	0

Table 10–24. Host UE Address Register (HOSTUEADDR)

Bit	Name	Description	Type	Reset Value
31:0	UE_ADDR	Unrecoverable error address This register captures the physical address of any OCP bus operation that is started by the USB host controller that encounters an unrecoverable error condition. This information, along with the information in HOSTUESTATUS, can help a developer determine why the USB host issued an OCP access to a physical address that resulted in an unrecoverable error.	R	0x0000 0000

The host UE address register reports the physical address of the last OCP bus access that caused an unrecoverable error (UE). This register has no meaning until an unrecoverable error has occurred. It also has no meaning if the USB host controller issues an unrecoverable error because the offset checking fault occurred while processing an isochronous TD. This register is not defined by the OHCI specification.

Table 10–25. Host UE Status Register (HOSTUESTATUS)

Bit	Name	Description	Type	Reset Value
31:1	Reserved	Reserved	R	xxxxxxxx
0	UEAccess	Access type when unrecoverable error occurred When an unrecoverable error occurs because of time-out of a OCP bus write, this bit is set. When an unrecoverable error occurs because of time-out of a OCP bus read, this bit is cleared. This bit has no meaning before an unrecoverable error occurs. This information, along with the information in HOSTUEADDR, can help a developer determine why the USB host issued an OCP bus access that resulted in an unrecoverable error.		0

The host UE status register reports the OCP bus cycle-type for the last unrecoverable error that occurred. This register has no meaning until an unrecoverable error has occurred. It also has no meaning if the USB host controller issues an unrecoverable error because the offset checking fault occurred while processing an isochronous TD. This register is not defined by the OHCI specification.

Table 10–26. Host Time-out Control Register (HOSTTIMEOUTCTRL)

Bit	Name	Description	Type	Reset Value
31:1	Reserved	Reserved		
0	TO_DIS	OCP bus time-out disable When 1, the USB host controller OCP bus time-out counter is disabled and the host controller waits indefinitely for completion of a USB host controller access to system memory. When 0, the USB host controller waits indefinitely to access system memory. When cleared (the default state), the USB host controller waits no more than 4096 OCP bus clocks for completion of a OCP bus access to system memory. If the OCP bus cycle does not complete in that time, the USB host controller signals an unrecoverable error.	R/W	0

The host time-out control register controls the USB host controller OCP bus time-out mechanism. This register is not defined by the OHCI specification.

Table 10–27. Host Revision Register (HOSTREVISION)

Bit	Name	Description	Type	Reset Value
31:8	Reserved	Reserved	R	xxxxxx
7:4	MAJORREV	Major revision number	R	xx
		MAJORREV indicates the major revision number of the USB host controller. The original OMAP730 USB host controller version implements a major revision number of 0.		
3:0	MINORREV	Minor revision number	R	xx
		MINORREv indicates the minor revision number of the USB host controller. The original OMAP730 USB host controller version implements a minor revision number of 0.		

The host revision register returns the revision number for the OMAP730 USB host controller. This register is not defined by the OHCI specification.

10.2.5 USB Host Controller Reserved Registers and Reserved Bit Fields

To enhance code reusability with possible future versions of the USB host controller, reads and writes to reserved USB host controller register addresses are to be avoided. Unless otherwise specified, when writing registers that have reserved bits, read-modify-write operations must be used so that the reserved bits are written with their previous values.

10.2.6 USB Host Controller Registers, USB Reset, and USB Clocking

When the USB host controller clock is disabled, or when the ULPD does not provide 48 MHz to the USB host controller, reads from and writes to the USB host controller registers do not occur correctly. To properly access the USB host controller registers, the USB host controller must be clocked and must be out of reset.

USB host controller clock enable is controlled as described in Table 10–28**.**

Table 10–28. USB Host Controller Clock Control

OTG_SYS-CON_ 2 OTG_ PADEN	MOD_CONF_CTRL_0. CONF_MOD_USB_HOST_ HHC_UHOST_EN_R	OTG_ SYSCON2 UHOST_EN	USB Host Clock Enabled?	USB Host in Reset?
0	Don’t care	0	No	Yes
0	Don’t care	1	Yes	No
1	0	Don’t care	No	Yes
1	1	Don’t care	Yes	No

The USB host controller hardware reset is controlled by the ARM_RSTCT2.PER_EN bit and the OTG_SYSCON_1.SOFT_RESET bit. The USB host controller is held in reset whenever ARM_RSTCT2.PER_EN is

0 or OTG_SYSCON_1.SOFT_RESET is 1. The USB host controller can transition out of reset whenever the clock is enabled and ARM_RSTCT2.PER_EN is 1 and OTG_SYSCON_1.SOFT_RESET is 0.

The USB host controller completes its reset within about 72 cycles of the 48-MHz clock after the host controller clock is transitioned from disabled to enabled using the mechanisms shown in Table 10–28 and the host controller reset is removed. After system software turns on the clock to the USB host controller and removes it from reset, it is necessary to wait until the USB host controller internal reset completes. To ensure that the USB host controller has completely reset, system software must wait until reads of both the HCREVISION register and the HCHCCA register return their correct reset default values.

10.2.7 OHCI Interrupts

The OMAP730 USB host controller provides an interrupt output to the MPU level 2 interrupt handler on its IRQ_27 interrupt input. This is a level-sensitive interrupt signal, and the MPU level 2 interrupt handler IRQ_27 must be programmed as a level-sensitive input.

10.2.7.1 OHCI Scheduling Overrun Interrupt

The OHCI scheduling overrun interrupt is supported as described in the *OHCI Specification for USB*.

10.2.7.2 OHCI HCDONEHEAD Writeback Interrupt

The OHCI HCDONEHEAD writeback interrupt is supported as described in the *OHCI Specification for USB*.

10.2.7.3 OHCI Start Of Frame Interrupt

The OHCI start of frame interrupt is supported as described in the *OHCI Specification for USB*.

10.2.7.4 OHCI Resume Detect Interrupt

The OHCI resume detect interrupt is supported as described in the *OHCI Specification for USB*.

10.2.7.5 OHCI Unrecoverable Error Interrupt

The OHCI unrecoverable error interrupt is supported as described in the *OHCI Specification for US*B. This interrupt occurs if the USB host controller is unable to complete an OCP bus read or OCP write within 4096 OCP bus clocks when the USB host OCP bus time-out feature is enabled [see Table 15-28, *Host Time-out Control Register (HOSTTIMEOUTCTRL*)]. When an OCP bus time-out causes an unrecoverable error, HOSTUEADDR and HOSTUESTATUS are updated. When an isochronous TD is processed with an OFFSET/PSW field that is not set for *Not Accessed*, an unrecoverable error interrupt is generated, but HOSTUEADDR and HOSTUESTATUS are not updated.

10.2.7.6 OHCI Frame Number Overflow

The OHCI frame number overflow interrupt is supported as described in the *OHCI Specification for USB*.

10.2.7.7 OHCI Root Hub Status Change

The OHCI root hub status change interrupt is supported as described in the *OHCI Specification for USB*. The OMAP730 does not provide a connection between the USB host controller and USB port overcurrent detection hardware, so the root hub status change interrupt does not occur because of a port overcurrent event.

10.2.7.8 OHCI Ownership Change Interrupt

The optional OHCI ownership change interrupt is not supported.

10.2.8 USB Host Controller Access to System Memory

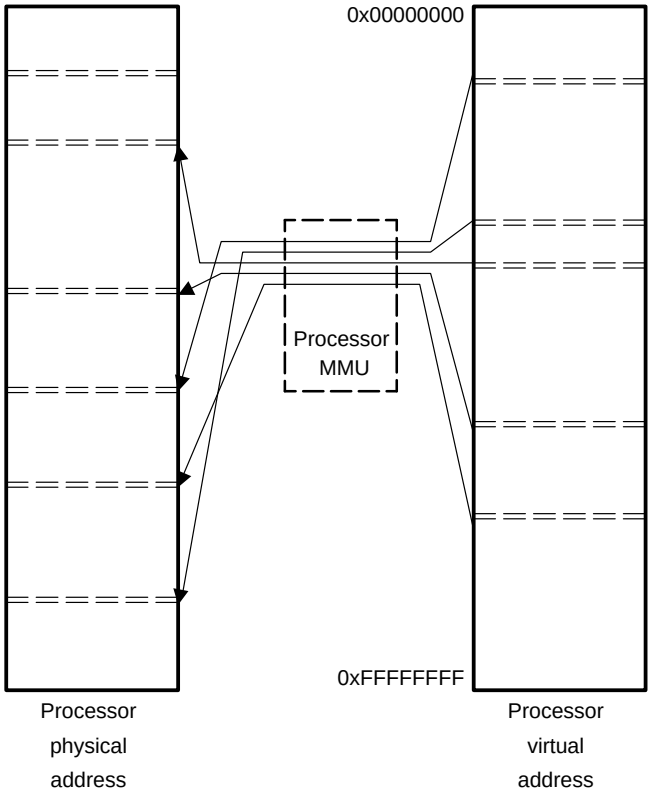
The USB host controller must have access to system memory to read and write the OHCI data structures and data buffers associated with USB traffic. The OMAP730 OCP bus allows the USB host controller to access OMAP730 system memory, as shown in Figure 10–1.

10.2.9 Physical Addressing

Transactions on the OMAP730 OCP bus use physical addresses, so all system memory accesses initiated by the USB host controller must use physical addresses. The OMAP730 MPU can be configured to use virtual addressing. In this case, MPU software manipulates virtual addresses that may or may not be identical to physical addresses. When virtual addressing is used, system software must perform the appropriate virtual address to physical address and physical address to virtual address conversions when manipulating the USB host controllers data structures and pointers to those data structures.

Figure 10–2 shows the MPU virtual address to physical address conversion.

Figure 10–2. Relationships Between Virtual Address Physical Address



10.2.10 Cache Coherency in OHCI Data Structures and Data Buffers

The OMAP730 traffic controller does not provide mechanisms to flush (or writeback) the MPU cache when a DMA controller or OCPI access to system memory occurs. Because there is no forced coherency mechanism, the system implementation must ensure that the OMAP730 USB host controller can access the correct data from system memory and that the MPU accesses that same data. This requires that any system memory accessed by the USB host controller be allocated in non-cached system memory.

If the OHCI data structures and/or data buffers are allocated in cached portions of system memory, a cache coherency problem can exist because the MPU can read from, and, if in writeback mode, write to the cache; but the USB host controller accesses are always directly to the physical system memory. If the data structures are in a cached portion of system memory and writeback mode is enabled, it is possible that the USB host controller can read stale data that has not been updated by a cache writeback.

Similarly, if the data structure is in memory that is currently in the MPU cache (either writeback or writethrough mode) and the OHCI controller modifies the information in physical memory, the MPU can read stale data from the cache.

Cache coherency problems can be avoided by allocating the OHCI data structures (HCCA, EDs, and TDs) and the USB data buffers in non-cacheable system memory. In this case, every MPU access directly accesses physical

memory, so there is no coherency issue. Configuration of cacheable portions of the MPU virtual address space is provided via the MPU memory management unit. See the description of the MPU MMU in the *OMAP3.2 Hardware Engine Reference Guide (SWPU019C)*.

10.2.11 OCP Bus Addressing and OHCI Data Structure Pointers

The USB host controller OHCI (open host controller interface) registers that point to the HCCA and the ED lists must be programmed with values that correspond to the physical addresses of the particular data structure. System software must use physical addresses and not processor addresses when manipulating the OHCI control registers that point to the HCCA, and to the ED and TD lists. System software must also use physical addresses for the ED and TD pointers that are stored within ED and TD entries.

The USB host controller driver software must also be able to examine the list of completed transfer descriptors that the host controller creates as it retires transfer descriptors. This list is pointed to by the HCDONEHEAD register, which contains a physical address that points to the most recent transfer descriptor that has been retired. This requires that the host controller driver software must be able to convert from the physical address back to an MPU virtual address.

Several address conversion functions are helpful to enable proper addressing by the MPU software and the USB host controller. The functionality required for proper address conversion depends on the settings used in the MPU MMU, so it is system-dependent. The conversion functions are described in general terms in the following subsections.

10.2.11.1MPUVAtoPA()—MPU Virtual Address to Physical Address Conversion Function

This function converts an MPU virtual address to the equivalent system physical address. This function must understand the way that the system software has configured the MPU MMU. This function must provide a conversion that has a result that is identical to the conversion done in hardware by the MPU MMU.

10.2.11.2PAtoMPUVA()—Physical Address to MPU Virtual Address Conversion Function

This function is a reverse version of the MPU virtual address to physical address conversion. It accepts a physical address as an argument and returns the equivalent MPU virtual address.

It is possible to program the MPU MMU to allow two (or more) different MPU virtual addresses to map to the same physical address. This is especially common in systems that use linear addressing within a task. System software implementers must be careful to avoid that situation or to perform the conversion in a way that understands the task-specific conversion requirements.

10.2.12 NULL Pointers

The *OHCI Specification for USB* uses NULL pointers to indicate the end of a list. The OMAP USB host controller compares the ED and TD pointers against

the value 0x00000000 to determine if the pointer is a null pointer. Address conversion routines must understand this usage.

10.2.13 OMAP730 OCP Bus and the USB Host Controller

The OMAP730 OCP bus provides a mechanism whereby OMAP730 peripheral modules can access system memory. Unlike peripherals that use the DMA controller, peripherals connected to the OCP bus can generate the address, byte enables, and read/write indication for each OCP access. This functionality allows the USB host controller to implement a scatter-gather engine to access the OHCI data structures from system memory in an efficient manner.

10.2.14 OCPI Registers

The USB host controller connects to an OCPI arbiter, which then connects to the transfer controller OCPI port. There are no OCPI arbiter register settings that affect USB host controller access to system memory, but the transfer controller OCPI port must be properly configured to allow access to system memory.

Because the USB host must be able to access system memory, the OMAP730 security features must be properly configured to allow USB host controller access to system memory.

10.2.15 USB Host Controller Clock Control

The OMAP730 clock generation and system reset management module (ULPD) provides a 48-MHz clock to the USB OTG controller, USB device controller, and USB host controller. This clock can be stopped by software to reduce USB OTG controller, USB device controller, and USB host controller power consumption when USB functionality is not needed.

The ULPD controls the 48-MHz clock to the USB host controller via:

- ❑ CLOCK_CTRL_REG.USB_MCLK_EN
- ❑ SOFT_REQ_REG.SOFT_USB_OTG_DPLL_REQ
- ❑ SOFT_DISABLE_REQ_REG.DIS_USB_HOST_DPLL_REQ

See Chapter 17, *Power and Clock Control*, for more details. When these PCC register bits are properly configured, the USB host controller receives a clock when the UHOST_EN signal is active. See discussions of OT_SYSCON_2.OTG_PADEN and OTG_SYSCON_2.HOST_EN in Section 10.4.

The USB host controller is held in reset and receives no clock at any time that UHOST_EN is inactive.

10.2.16 USB Host Controller Hardware Reset

The ULPD module provides reset of the USB host controller. The PER_EN bit in the MPU reset control 2 register controls the reset to many OMAP730 peripherals, including the USB host controller. When held in reset, the USB host controller does not generate any USB activity on its USB ports. The USB host controller requires that its 48-MHz clock input (from the OMAP730 ULPD module) be active and that UHOST_EN be set (see Table 10–62, *OTG System Configuration Register 2 (OTG_SYSCON_2)*) in order to complete its reset sequence.

There is a delay of approximately 72 cycles of the ULPD USB host controller 48-MHz clock before the USB host controller is successfully reset. This delay starts at the latest of the PER_EN bit set, UHOST_EN set, or 48-MHz clock start. When the USB host controller is in hardware reset, read or write accesses to its registers have no effect. It is recommended that USB host controller software read the HCREVISION and HCHCCA registers after deasserting reset to verify the proper reset values. If the read values for both HCREVISION and HCHCCA are not correct, reset the values and continue reading until the proper reset values are seen.

The UHOST_EN bit, when cleared, also holds the USB host controller in a hardware reset. While the USB host controller is in reset, reads from the USB host controller registers do not return valid data, and writes to the USB host controller registers have no effect. When UHOST_EN is cleared, all USB host controller internal state information is lost.

Software that initializes the USB host controller must ensure that the reset is turned off, that the ULPD 48-MHz clock for the USB host controller is enabled, and that UHOST_EN is set. It must then wait until reads of both the HCREVISION register and the HCHCCA register return their correct reset default values.

10.2.17 USB Host Controller OHCI Reset

The *OHCI Specification for USB* provides the HCR bit in the HCCOMMANDSTATUS register, which resets the OHCI controller. This reset can be used to reset the OHCI functionality and has no effect on the USB host controller OCPI and the MPU public peripheral bus interfaces. The OHCI reset does not affect the USB host controller clock.

10.2.18 USB Host Controller Power Management

Power management of the OMAP730 USB host controller is limited to disabling the clock to the USB host by clearing UHOST_EN (see (see Table 10–62, *OTG System Configuration Register 2 (OTG_SYSCON_2)*). When UHOST_EN is 0, the USB host controller clocks are disabled and the USB host controller is held in reset. The USB signal multiplexing controlled by HMC_MODE is not affected, so a HMC_MODE setting that multiplexes USB function controller and/or UART1 signals to OMAP730 top-level multiplexing can still make use of the USB function controller and/or UART1.

When the OMAP730 host controller 48-MHz clock is disabled or UHOST_EN is 0, all USB host controller OHCI registers and the HOSTUEADDR, HOSTUESTATUS, HOSTTIMEOUTCTRL, and HOSTREVISION registers are inaccessible.

10.2.19 OCPI Clocking

The OCPI clock must be active to allow USB host controller access to system memory. The OCPI clock is controlled by the ARM_IDLECT3.EN_OCPI_CK and IDLOCPI_ARM bits (see Chapter 5, *CLock and Reset Architecture*).

10.3 USB Device Controller

The USB device controller supports the implementation of a full-speed device fully compliant with the USB 1.1 standard.

It provides an interface between the MPU and the USB wire and handles USB transactions with minimal MPU software intervention.

The USB device controller module supports one control endpoint (EP0), up to 15 IN endpoints, and up to 15 OUT endpoints. The exact endpoint configuration is software programmable. The specific items of a configuration are for each endpoint, the size in bytes, the direction (IN, OUT), the type (bulk/interrupt or ISO), and the associated number.

The USB device controller module also supports three DMA channels for IN endpoints and three DMA channels for OUT endpoints for either bulk/interrupt or ISO transactions.

10.3.1 USB Device Controller Registers

Table 10–29 lists the USB device controller registers. Table 10–30 through Table 10–52 describe the register bits.

Table 10–29. USB Device Controller Registers

Register	Description	R/W	Offset (OxFFFB:)
REV	Revision	R	4000
	Endpoint		
EP_NUM	Endpoint selection	R/W	4004
DATA	Data	R/W	4008
CTRL	Control	W	400C
STAT_FLG	Status	R	4010
RXFSTAT	Receive FIFO status	R	4014
SYSCON1	System configuration 1	R/W	4018
SYSCON2	System configuration 2	R/W	401C
DEVSTAT	Device status	R	4020
SOF	Start of frame	R	4024
IRQ_EN	Interrupt enable	R/W	4028
DMA_IRQ_EN	DMA interrupt enable	R/W	402C
IRQ_SRC	Interrupt source	R/C†	4030
EPN_STAT	Non-ISO endpoint interrupt enable	R	4034
DMAN_STAT	Non-ISO DMA interrupt enable	R	4038
Reserved			403C
	DMA Configuration		
RXDMA_CFG	DMA receive channels configuration	R/W	4040
TXDMA_CFG	DMA transmit channels configuration	R/W	4044
DATA_DMA	DMA FIFO data	R/W	4048
Reserved			404C
TXDMA0	Transmit DMA control 0	R/W	4050
TXDMA1	Transmit DMA control 1	R/W	4054
TXDMA2	Transmit DMA control 2	R/W	4058
Reserved			405C
RXDMA0	Receive DMA control 0	R/W	4060
RXDMA1	Receive DMA control 1	R/W	4064

Table 10–29. USB Device Controller Registers(Continued)

Register	Description	R/W	Offset (OxFFFB:)
RXDMA2	Receive DMA control 2	R/W	4068
Reserved			406C...407F
	Endpoint Configuration		
EP0	Endpoint configuration 0	R/W	4080
EP1_RX...EP15RX	Receive endpoint configuration 1...15	R/W	4084...40BC
Reserved			40C0
EP1_TX...EP15TX	Transmit endpoint configuration 1...15	R/W	40C4...40FC

† Read/Clear only register

- ❑ 16-bit access: all bits of the register are accessed.
- ❑ 8-LSB bit access: the 8 least significant bits are accessed.
- ❑ 8-MSB bit access: the 8 most significant bits are accessed.

Table 10–30. Revision Register (REV)

Bit	Name	Description
15:8	-	Reserved
7:0	REV_NB	This 8-bit field indicates revision number of current USB device controller module—value fixed by hardware: 0x01: Revision 0.1 0x02: Revision 0.2 0x21: Revision 2.1

This read-only register contains the revision number of the module. A write to this register is denied.

MPU and USB reset have no effect on this register.

Table 10–31. Endpoint Selection Register (EP_NUM)

Bit	Name	Description
15:7	-	Reserved
6	SETUP_SEL	The setup FIFO select bit is set by the USB device controller to access the status (STAT_FLG, RXFSTAT) and data (DATA) registers for the endpoint selected. If EP_NUM.EP_DIR bit is set to 0, the MPU can read data from endpoint RX FIFO by reading DATA register. If EP_NUM.EP_DIR bit is set to 1, the MPU can write data into endpoint TX FIFO by writing into the DATA register. After each access to an endpoint during interrupt handling, the USB device controller must absolutely clear this bit: 0: No access 1: Access permitted Note: When the USB device controller sets this bit, it must set SETUP_SEL bit to 0. After having accessed the endpoint FIFO either for read or for write access, the USB device controller must clear this bit by writing a 0 to it. The value after MPU or USB reset is low.
5	EP_SEL	The TX/RX FIFO select bit is set by the USB device controller in response to a set-up general USB interrupt, to access the EP0 read-only setup FIFO when reading DATA register. Setting this bit clears the IRQ_SRC.SETUP interrupt bit. When this bit is set, the value of other EP_NUM register bits must be 0. 0: No access 1: Access permitted Note: After having read the setup FIFO, the USB device controller must clear this bit by writing a 0 to it. Value after MPU or USB reset is low.
4	EP_DIR	The endpoint direction bit gives the direction associated with the endpoint number selected in EP_NUM.EP_NUM: 0: OUT endpoint 1: IN endpoint Value after MPU or USB reset is low.
3:0	EP_NUM	The endpoint number binary encoded in these four bits, associated with the direction given by EP_NUM.EP_DIR bit, is the current endpoint selected. All reads and writes to the endpoint status and the control and data locations are for this endpoint. 0000: EP0 0001: EP1 … 1111: EP15 Value after MPU or USB reset is low.

The endpoint selection register is a read/write register that selects and enables the endpoint that can be accessed by the USB device controller.

Table 10–32. Data Register (DATA)

Bit	Name	Description
15:0	DATA†	Transmit/receive FIFO data: EP_NUM.EP_DIR = 1: This register contains the data written by the USB device controller to be sent to the USB host during the next IN transaction. Data can be written successfully only if the EP_NUM.EP_SEL bit is asserted. EP_NUM.EP_DIR = 0: This register contains the data received by the USB core from USB host OUT or SETUP transactions. Data can be read successfully only if the EP_NUM.EP_SEL bit is asserted, or if EP_NUM.SETUP_SEL bit is asserted (for setup data).

† A write into the data register (DATA) when EP_NUM.EP_DIR = 0 and a read from the register when EP_NUM.EP_DIR = 1 are denied.

The data register is the entry point to write into a selected TX endpoint or to read data from a selected RX endpoint, or to read data from setup FIFO. If selected endpoint direction is OUT, this register is read-only and a write into it is denied. If selected endpoint direction is IN, this register is write-only and a read to this register is denied.

Table 10–33. Control Register (CTRL)

Bit	Name	Description
15:8	–	Reserved
7	CLR_HALT†	The clear halt endpoint (non-ISO) bit only concerns non-ISO endpoints—used by the USB device controller to clear an endpoint halt condition: 0: No action 1: Clear halt condition Always read 0
6	SET_HALT†	The set halt endpoint (non-ISO) only concerns non-ISO endpoints—used by the USB device controller to halt the selected endpoint. The halted endpoint returns STALL handshakes to the USB host. The USB device controller can disable the endpoint interrupt if it does not want to be informed of STALL handshakes. Note: If the endpoint to halt is used by a DMA channel, the USB device controller must disable the DMA channel before setting halt condition for this endpoint. 0: No action 1: Halt endpoint Always read 0
5:3	–	Reserved

Table 10–33. Control Register (CTRL) (Continued)

Bit	Name	Description
2	SET_FIFO_EN†	The set FIFO enable (non-ISO) bit only concerns non-ISO endpoints. If the selected endpoint direction is IN, the USB device controller uses this bit to enable the USB device to transmit data from the FIFO at the next valid IN token. If the selected endpoint direction is OUT, the USB device controller uses this bit to enable the USB device to receive data from the USB host at the next valid OUT transaction. If not, setting the device returns a NAK handshake. ISO endpoints FIFO are always enabled. Note: The USB device controller must never enable endpoint 0 FIFO out of control transfers. For bulk and interrupt endpoints, FIFO must never be enabled when the halt feature is set or when RX FIFO is not empty. Furthermore, during EP interrupts handling, the USB device controller must have cleared the interrupt bit before setting CTRL.SET_FIFO_EN bit (to avoid masked ACK interrupts). 0: No action 1: FIFO enabled Always read 0
1	CLR_EP	Clear endpoint: the USB device controller sets this bit to clear the selected endpoint FIFO pointers and flags. This bit resets the FIFO pointers, the FIFO empty status bit is set, and the FIFO enable bit and other FIFO flags are cleared upon completion of the FIFO reset. It also clears the previous transaction handshake status. For ISO endpoints or non-ISO double-buffered endpoints, both foreground and background FIFO are cleared. 0: No action 1: Clear endpoint Always read 0
0	RESET_EP	The endpoint reset (non-control) bit only concerns non-control endpoints.The USB device controller sets this bit to reset the selected endpoint. It forces an interrupt or a bulk endpoint data PID to DATA0, clears the halt condition, and clears the FIFO (both foreground and background if endpoint is double-buffered) and previous transactions handshake status. For an ISO endpoint, it only clears the FIFO (both foreground and background). 0: No action 1: Reset endpoint Always read 0

† It is not required to set EP_NUM.EP_SEL bit before setting this bit. If this bit is set during the handling of an interrupt to the endpoint, however, the USB device controller must not set EP_NUM.EP_SEL bit before setting CTRL.CLR_HALT bit, in order to avoid a possible effect on interrupts . The USB device controller must check that FIFO is empty before setting the halt feature for the endpoint. A STALL transaction clears the FIFO.

CTRL is a set-only register that controls the FIFO and the status of the selected endpoint. A read access to this register always returns 0.

Endpoint 0 setup FIFO is always enabled and ready to accept setup data. No control register CTRL is implemented for this FIFO because the USB device controller cannot control it.

Table 10–34. Status Register (STAT_FLG)

Bit	Name	Description
15	NO_RXPACKET	The isochronous no packet received (ISO OUT) bit only concerns ISO OUT endpoints.This bit notifies the USB device controller that the core has not received any isochronous packet on the endpoint. This bit is updated on a SOF (start of frame). 0: The endpoint received a packet on the previous frame. 1: The endpoint did not receive a packet on the previous frame. Value after MPU or USB reset is high.
14	MISS_IN	The isochronous missed IN token for the previous frame (ISO IN) bit only concerns ISO IN endpoints.This bit notifies the USB device controller that the core missed a valid ISO IN token during the previous frame and that TX data were flushed from the FIFO instead of being transmitted to the USB host. This bit is updated on a SOF (start of frame). 0: The endpoint received an IN token the previous frame. 1: The endpoint did not receive an IN token the previous frame and TX data were flushed. Value after MPU or USB reset is low.
13	DATA_FLUSH	The isochronous receive data flush (ISO OUT) bit only concerns ISO OUT endpoints. When set, this bit indicates that data were flushed from the isochronous FIFO that was moved from the foreground to the background. This happens when the USB device controller does not read all of the data from the foreground FIFO in a frame. This bit is updated in every frame. 0: Not significant 1: Data was flushed. Value after MPU or USB reset is low.
12	ISO_ERR	The isochronous receive data error (ISO OUT) bit only concerns ISO OUT endpoints. When set, this bit indicates that the ISO data packet was received incorrectly. This happens when the core detects an error in the data packet (CRC, bit stuffing, PID check) or when there is an overrun condition in the FIFO. When this bit is set, the FIFO contents are automatically flushed by the core and the FIFO status is empty. This bit is updated in every frame. 0: Not significant 1: ISO packet received with errors Value after MPU or USB reset is low.
11:10	-	Reserved

Table 10–34. Status Register (STAT_FLG) (Continued)

Bit	Name	Description
9	ISO_FIFO_EMPTY	The ISO FIFO empty bit only concerns ISO endpoints.This bit is set when the FIFO for the selected ISO endpoint is empty, either via an appropriate CTRL.CLR_EP bit or CTRL.RESET_EP bit or after successful reads from the selected FIFO. 0: ISO FIFO is not empty. 1: ISO FIFO is empty. Value after MPU or USB reset is high (FIFO empty).
8	ISO_FIFO_FULL	The ISO FIFO full bit only concerns ISO endpoints.This bit is set when the FIFO for the selected ISO endpoint is full. This condition is cleared by setting the CTRL.CLR_EP bit or CTRL.RESET_EP bit, or after one successful read (by the USB device controller or the USB host). 0: ISO FIFO is not full. 1: ISO FIFO is full. Value after MPU or USB reset is low (FIFO empty).
7	-	Reserved
6	EP_HALTED	The endpoint halted flag (non-ISO) bit only concerns non-ISO endpoints.This bit when asserted 1 indicates that the selected endpoint is halted. The endpoint can be put into the halt state only by the USB device controller writing the endpoint halt control bit (in response to a SET_FEATURE request, for instance). 0: The selected endpoint is not halted. 1: The selected endpoint is halted. Value after MPU or USB reset is low.
5	STALL	The transaction stall (non-ISO) bit only concerns non-ISO endpoints.This status bit is set at the end of a transaction if a STALL handshake packet was returned to the USB host, and if no non-handled interrupt is pending on the current buffer (see Section 10.3.11, *Important Note on USB Device Interrupts*). The core automatically returns a STALL packet if a valid IN token is received by a halted TX endpoint, if a valid OUT transaction is received by a halted RX endpoint, or if there is a request error (endpoint 0). The bit is cleared when the USB device controller has finished handling the corresponding interrupt (at EP_NUM.EP_SEL bit deselection). 0: No STALL handshake was returned. 1: A STALL handshake packet was returned. Value after MPU or USB reset is low.

Table 10–34. Status Register (STAT_FLG) (Continued)

Bit	Name	Description
4	NAK	The transaction non-acknowledge (non-ISO) bit only concerns non-ISO endpoints with SYSCON1.NAK_EN bit asserted.This status bit is set at the end of a transaction if a NAK handshake is returned to the USB host, and if no non-handled interrupt is pending on the current buffer. The USB core automatically returns a NAK handshake to the USB host if a valid IN token is received by a TX endpoint or if a valid OUT transaction is received by an RX endpoint, and the STAT_FLG.FIFO_EN bit is not set for the endpoint. The bit is cleared when the USB device controller has finished handling the corresponding interrupt (at EP_NUM.EP_SEL bit deselection). 0: No NAK handshake was returned (SYSCON1.NAK_EN bit is set). 1: A NAK handshake packet was returned and SYSCON1.NAK_EN bit is set. Value after MPU or USB reset is low.
3	ACK	The transaction acknowledge (non-ISO) bit only concerns non-ISO endpoints.This bit is set at the end of a non-transparent valid IN transaction if the data packet was sent successfully to the USB host and the ACK handshake was received, or at the end of a non-transparent valid OUT transaction if the data packet was received successfully by the USB device and the ACK handshake was returned. The bit is cleared when the USB device controller has finished handling the corresponding interrupt (at EP_NUM.EP_SEL bit deselection). 0: No ACK handshake packet was returned. 1: An ACK handshake packet was returned. Value after MPU or USB reset is low.
2	FIFO_EN	The FIFO enable status (non-ISO) bit only concerns non-ISO endpoints.This bit is asserted when CTRL.SET_FIFO_EN is set to 1 and is cleared automatically after a transaction completes with an ACK, STALL. 0: The non-ISO endpoint FIFO is disabled. 1: The non-ISO endpoint FIFO is enabled. Value after MPU or USB reset is low.

Table 10–34. Status Register (STAT_FLG) (Continued)

Bit	Name	Description
1	NON_ISO_FIFO_EMPTY	The non-ISO FIFO empty bit only concerns non-ISO endpoints.This bit is set when the FIFO for the selected non-ISO endpoint is empty, either via an appropriate CTRL.CLR_EP bit or CTRL.RESET_EP bit or after successful reads from the selected FIFO. 0: Non-ISO FIFO is not empty. 1: Non-ISO FIFO is empty. Value after MPU or USB reset is high (FIFO empty).
0	NON_ISO_FIFO_FULL	The non-ISO FIFO full bit only concerns non-ISO endpoints.This bit is set when the FIFO for the selected non-ISO endpoint is full. This condition is cleared by setting the CTRL.CLR_EP bit or CTRL.RESET_EP bit, or after one successful read (by the USB device controller or the USB host). 0: Non-ISO FIFO is not full. 1: Non-ISO FIFO is full. Value after MPU or USB reset is low (FIFO empty).

STAT_FLG is a read-only register that provides a status of the FIFO and the results of the transactions handshakes for the selected endpoint. The 8 MSB are reserved for ISO endpoints, whereas the 8 LSB are reserved for non-ISO endpoints. This register cannot be read if EP_NUM.EP_SEL bit is not asserted for the endpoint. No status flag exists for the read-only set-up FIFO, which is always enabled.

Note:

The updates for non-ISO transactions are done at the end of each non-transparent and valid transaction to a given endpoint, if no non-handled interrupt is pending on the endpoint.

The definition of a non-transparent, non-ISO IN transaction is one that responds with an ACK handshake or a STALL handshake, or optionally a NAK handshake if SYSCON1.NAK_EN is asserted to 1. An ERR handshake or a NAK handshake when SYSCON1.NAK_EN is 0 is considered transparent.

A write to this register has no effect.

Table 10–35. Receive FIFO Status Register (RXFSTAT)

Bit	Name	Description
15:10	–	Reserved
9:0	RXF_COUNT	The receive FIFO byte count 10-bit field indicates the number of bytes currently in the receive FIFO. Values after MPU or USB reset is low (all 10 bits).

RXFSTAT is a read-only register that indicates how many bytes are in the receive FIFO for the selected endpoint. A write to this register has no effect. The USB device controller cannot read this register if the EP_NUM.EP_SEL bit is not set for the endpoint. No receive FIFO status exists for the setup FIFO, because 8 bytes are always expected.

Table 10–36. System Configuration Register 1 (SYSCON1)

Bit	Name	Description
15:9	-	Reserved
8	CFG_LOCK	Device configuration locked bit: after the USB device controller has entered the device configuration (registers 0x20 to 0x3F), it must set the CFG_LOCK bit so that the device can be used. When the device configuration is not locked, the device is not ready to be used: 0: Device configuration is not locked. Device is not ready. 1: Device configuration is locked. The value after the USB device controller hardware reset is low; after USB reset, it is unchanged (keep previous configuration).
7	DATA_ENDIAN	The data endian bit can be set by the USB device controller to select little- or big-endian format on data access (read or write): 0: Little endian 1: Big endian The value after USB device controller hardware reset is low; after USB reset, it is unchanged (keep previous configuration).
6	DMA_ENDIAN	The DMA data endian bit can be set by the USB device controller to select little- or big-endian format on data access (read or write): 0: Little endian 1: Big endian The value after USB device controller hardware reset is low; after USB reset, it is unchanged (keep previous configuration).
5	–	Reserved
4	NAK_EN	The NAK enable bit can be set by the USB device controller to be signaled for NAK transaction handshake response. When this bit is set, STAT_FLG.NAK is set on a NAK handshake if no non-handled interrupt is pending on the endpoint, and the endpoint interrupt is asserted. In the normal mode, when cleared, NAK handshake response to the USB host is made transparent to the USB device controller and no interrupt is asserted: 0: NAK is disabled. 1: NAK is enabled. The value after MPU or USB reset is low. Note: If the USB device controller sets this bit, it must wait for a NAK interrupt before selecting TX endpoint to write TX data.

Table 10–36. System Configuration Register 1 (SYSCON1) (Continued)

Bit	Name	Description
3	AUTODEC_DIS	The autodecode process disabled can be set to force software to handle all EP0 transactions (except the SET_ADDRESS transaction): 0: Autodecode process is activated (see Table 10–53, *Autodecoded Versus Non-Autodecoded Control Requests*). 1: Autodecode process is deactivated. The value after USB device controller hardware reset is low; after USB reset, it is unchanged.
2	SELF_PWR	The self-powered bit indicates to the USB host whether the device is bus-powered or self-powered. This information is needed for an autodecoded request. The USB device controller must update this bit after a SET_CONFIGURATION according to the self-powered bit D6 given in the configuration descriptor (see USB 1.1 specification chapter 9). 0: Bus powered 1: Self-powered The value after USB device controller hardware reset is low; after USB reset, it is unchanged.
1	SOFF_DIS	When the shutoff disable bit is set, it disables the power shutoff circuitry: 0: Power shutoff circuitry is enabled. 1: Power shutoff circuitry is disabled. The value after USB device controller hardware reset is low; after USB reset, it is unchanged.
0	PULLUP_EN	The external pullup enable bit allows the device to disconnect itself from the USB bus, forcing the host to reset and reconfigure the device. This bit can be used to prevent USB traffic when the device is not ready: 0: Pull-up is disabled. USB host cannot detect the device. In this mode, the 48-MHz USB clock is forced off. 1: Pull-up is enabled. The value after USB device controller hardware reset is low; after USB reset, it is high; and after detach, it is unchanged.

SYSCON1 is a read/write register that provides control functions for power management and miscellaneous control for the core.

Values depend on whether the reset action comes from the USB device controller (MPU) or the USB host.

15 8	7 0
Byte 1	Byte 0
Little Endian Format (16 bits)	

15 8	7 0
Byte 0	Byte 1
Big Endian Format (16 bits)	

Table 10–37. System Configuration Register 2 (SYSCON2)

Bit	Name	Description
15:7	–	Reserved
6	RMT_WKP	The set-only remote wake-up bit, when written with a 1, initiates the remote wake-up sequence even if DEVSTAT.R_WK_OK bit was not previously set to 1 by the USB host. Reading this bit always returns 0. Writing 0 into this bit has no effect. To generate a resume, the software must check the remote wake-up enable value before initiating any wake-up sequence: 0: No action 1: Initiates the remote wake-up sequence Always read 0
5	STALL_CMD	The set-only stall command bit only concerns non-autodecoded requests on control endpoint (EP0).This is asserted in response to a USB command where either the command itself or its data is invalid. Asserting this bit forces the non-autodecoded command to complete with a STALL handshake. It has no effect for autodecoded requests. 0: No action 1: Stall current USB command Always read 0
4	Reserved	Reserved
3	DEV_CFG	If the USB device controller receives a SET_CONFIGURATION with a valid configuration value, and the device is in addressed state, it must write a 1 to the device configured (DEV_CFG) bit to inform the command decode that the device moves to configured state. The core sets the DEVSTAT.CFG bit to 1. If the device is already configured when the SET_CONFIGURATION request is received, the USB device controller does not have to set this bit. If the new configuration value is 0, USB device controller must set the SYSCON2.CLR_CFG bit to move to addressed state. Reading this bit always returns 0. Writing 0 into this bit has no effect. 0: No action 1: Allows DEVSTAT.CFG to be set Always read 0
2	CLR_CFG	If the USB device controller receives a SET_CONFIGURATION with a configuration value of 0, and if the device is configured, it must write a 1 to the clear configured (CLR_CFG) bit to inform the command decode that the device becomes deconfigured (moves to addressed state). The core clears the DEVSTAT.CFG bit. Reading this bit always returns 0. Writing 0 into this bit has no effect. 0: No action 1: Allows DEVSTAT.CFG to be cleared Always read 0
1:0	Reserved	Reserved

SYSCON2 is a set-only register that provides miscellaneous controls for the function. A read to this register always returns 0.

Table 10–38. Device Status Register (DEVSTAT)

Bit	Name	Description
15:10	Reserved	Reserved
9	B_HNP_ENABLE	The HNP enable for B-device bit is used for On-The-Go feature selection purposes. See the *On-The-Go Supplement* to the *USB 2.0 Specification*. 0: Not significant 1: HNP enable for B-device Value after MPU or USB reset is low.
8	A_HNP_SUPPORT	The B-device is directly connected to an A-device port that supports HNP. This A_HNP_SUPPORT bit is used for On-The-Go feature selection purposes. See the *On-The-Go Supplement* to the *USB 2.0 Specification*. 0: Not significant 1: B-device connection to HNP capable A-device Value after MPU or USB reset is low.
7	A_ALT_HNP_SUPPORT	The B-device is connected to an A-device port that is not capable of HNP, but the A-device has an alternate port that is capable of HNP. This A_ALT_HNP_SUPPORT bit is used for On-The-Go feature selection purposes. See the *On-The-Go Supplement* to the *USB 2.0 Specification*. 0: Not significant 1: B-device connection to not HNP capable A-device Value after MPU or USB reset is low.
6	R_WK_OK	The remote wake-up granted bit is automatically set and cleared when the core receives a valid set/clear device feature request from the USB host. It returns a 1 when the USB host grants the function the ability to assert remote wake-up. 0: Not significant 1: Remote wake-up granted Value after MPU or USB reset is low.

Table 10–38. Device Status Register (DEVSTAT) (Continued)

Bit	Name	Description
5	USB_RESET	USB reset signaling is active. The USB_RESET bit returns 1 when the USB host resets the USB bus. A valid USB reset resets the endpoint FIFOS, the other control register bits, except for SYSCON1.CFG_LOCK and the associated configuration registers (0x20 to 0x3F), IRQ_EN.DS_CHG_IE and IRQ_SRC.DS_CHG, and forces the device to the default state for DEVSTAT (this register). This bit is cleared at the end of reset. This bit, as well as other DEVSTAT bits, is double-buffered. If a pending interrupt has not been handled when a USB reset occurs and is handled only when USB reset is finished, the USB device controller does not see the USB_RESET bit going high and then low. 0: Device is not being reset by USB host. 1: Device is being reset by USB host. The value after the USB device controller hardware reset is low, and during USB reset is high (low after USB reset).
4	SUS	Suspended state bit: the device in suspended state is, at a minimum, attached to the USB, powered, has been reset by the USB host, and has not seen bus activity for 5 ms. It can also have a unique address and be configured for use. Because the device is suspended, however, the host cannot use the device function. This bit returns 1 when the USB device is in a suspended state. 0: Not suspended 1: Suspended Value after MPU or USB reset is low.
3	CFG	Configured state bit: the device is attached to the USB, powered, has been reset, has a unique address, and is configured. The host can use the function provided by the device. This bit returns 1 when the USB device has been configured after a set SYSCON2.DEV_CFG = 1. This bit remains set to 1 until the device becomes deconfigured. This bit is cleared when the core receives a valid SET_CONFIGURATION request and the USB device controller sets the SYSCON2.CLR_CFG bit. During the time this bit is not set to 1, transactions that are not for control EP0 are ignored. A GET_ENDPOINT_STATUS to a non-control endpoint is stalled. 0: Not configured 1: Configured Value after MPU or USB reset is low.

Table 10–38. Device Status Register (DEVSTAT) (Continued)

Bit	Name	Description
2	ADD	Addressed state bit: the device is attached to the USB, powered, has been reset, and a unique device address has been assigned. This bit (ADD) returns 1 after an ET_ADDRESS standard request. This bit remains set to 1 until the device becomes deaddressed. 0: Not addressed 1: Addressed Value after MPU or USB reset is low.
1	DEF	The default state bit returns 1 when the USB device is attached to the USB and powered, and has been reset. This bit remains set to 1 until the device becomes depowered. The device moves into default state as soon as the USB reset is effective. 0: Not in default 1: Default The value after MPU is low, and after USB reset is high.
0	ATT	The attached state bit returns 1 when the device is attached to the USB and is powered. This bit remains set to 1 until the device powers down. 0: Not attached 1: Attached The value after USB device controller hardware reset is low (unattached) or high (attached), and after USB reset is high.

DEVSTAT is a read-only register that provides a status reflecting the visible device states as defined in *USB1.1 Specification*, Chapter 9. A write to this register has no effect.

Note:

This register is double-buffered. If IRQ_EN.DS_CHG_IE is set (interrupt enabled), the background register is moved into foreground position only after clearing any pending DS_CHG interrupt. Therefore, if there is a state change and a pending DS_CHG interrupt has not been serviced yet, then the recent state change is not visible, because the background register was only updated and not moved into the foreground position.

The values depend on whether the reset action comes from the USB device controller (MPU) or USB host. They also depend on whether the device is attached or not when the USB device controller hardware reset event occurs.

Table 10–39. Start of Frame Register (SOF)

Bit	Name	Description
15:13	Reserved	Reserved
12	FT_LOCK	Frame timer locked bit: the USB host sends out a start of frame (SOF) packet every millisecond. When the device does not receive a SOF packet because of a bus error, a local start of-frame that is used for ISO FIFO switch is generated. Once the core receives two valid SOF, separated by TF (time frame), it sets SOF.FT_LOCK to 1, only if TF is in frame interval IF allowed by the USB 1.1 specification (IF = [11964:12036] USB bit time). If TF is out of this interval, SOF.FT_LOCK value remains 0, and a local SOF is generated by the core. When SOF.FT_LOCK is set and the frame timer is locked to the timing TF, a local SOF is generated, if no valid SOF has been received in an interval of TF since the last valid SOF. The SOF.FT_LOCK bit is cleared if a valid SOF is received out of the interval IF. If the core receives a valid SOF in this interval, the frame timer locks to the new frame time. If the core does not receive a valid SOF, the frame timer remains locked to TF. When SOF.FT_LOCK is cleared, a local SOF is generated after the 12036 USB bit time, if no valid SOF has been received, and SOF.FT_LOCK remains 0. 0: Frame timer is not locked. 1: Frame timer is locked. The value after MPU or USB reset is low. Note: Each time the core receives a valid SOF out of the allowed interval IF, a local SOF is generated and the ISO FIFO switches.
11	TS_OK	The timestamp OK bit indicates that the timestamp in SOF.TS is valid for the current frame. It returns a 1 if a valid SOF packet was received from the USB host and a 0 otherwise. 0: Timestamp is invalid. 1: Timestamp is valid. Value after MPU or USB reset is low.
10:0	TS	The timestamp number field returns the timestamp from the last USB host-valid SOF packet. The frame number is valid if SOF.TS_OK is 1. In case of an SOF miss, this value is not updated and TS_OK is cleared. The value after MPU or USB reset is low (all 11 bits).

SOF is a read-only register that provides a frame timer status for use in ISO communications. A write to this register is denied.

Table 10–40. Interrupt Enable Register (IRQ_EN)

Bit	Name	Description
15:8	Reserved	Reserved
7	SOF_IE	Start of frame interrupt enable
6	Reserved	Reserved
5	EPn_RX_IE	Receive endpoint n interrupt enable (non-ISO)
4	EPn_TX_IE	Transmit endpoint n interrupt enable (non-ISO)

Table 10–40. Interrupt Enable Register (IRQ_EN) (Continued)

Bit	Name	Description
3	DS_CHG_IE	Device state changed interrupt enable
2:1	Reserved	Reserved
0	EP0_IE	EP0 transactions interrupt enable

IRQ_EN is a read/write register that enables all non-DMA interrupts (control, state changed, ISO, and non-ISO).

The values depend on whether the reset action comes from the USB device controller (MPU) or the USB host.

When the USB device controller sets a bit position to 1, an interrupt is signaled to the USB device controller if the corresponding IRQ_SRC bit is asserted to 1 by the core for any IRQ_SRC bit controlled by this bit. If reset to 0, the interrupt is masked and not signaled to the USB device controller.

- 0: Interrupt is disabled.
- 1: Interrupt is enabled.

The value after MPU or USB reset is low for all bits except IRQ_EN.DS_CHG_IE bit, which remains unchanged after a reset from the USB host.

Table 10–41. DMA Interrupt Enable Register (DMA_IRQ_EN)

Bit	Name	Description
15:11	Reserved	Reserved
10	TX2_DONE_IE	Transmit DMA channel 2 done interrupt enable
9	RX2_CNTt_IE	Receive DMA channel 2 transactions count interrupt enable
8	RX2_EOT_IE	Receive DMA channel 2 end of transfer interrupt enable
7	Reserved	Reserved
6	TX1_DONE_IE	Transmit DMA channel 1 done interrupt enable
5	RX1_CNT_IE	Receive DMA channel 1 transactions count interrupt enable
4	RX1_EOT_IE	Receive DMA channel 1 end of transfer interrupt enable
3	Reserved	Reserved
2	TX0_DONE_IE	Transmit DMA channel 0 done interrupt enable
1	RX0_CNT_IE	Receive DMA channel 0 transactions count interrupt enable
0	RX0_EOT_IE	Receive DMA channel 0 end of transfer interrupt enable

DMA_IRQ_EN is a read/write register that enables all DMA interrupts.

When the USB device controller sets a bit position to 1, an interrupt is signaled to the USB device controller if the corresponding bit in the IRQ_SRC register

is asserted to 1 by the core. If reset to 0, the interrupt is masked and not signaled to the USB device controller.

- ❏ 0: Interrupt is disabled.
- ❏ 1: Interrupt is enabled.

The value after MPU or USB reset is low for all bits.

Table 10–42. Interrupt Source Register (IRQ_SRC)

Bit	Name	Description
15:11	Reserved	Reserved
10	TXn_DONE	The transmit DMA channel n done interrupt flag (non-ISO) bit is only for non-ISO DMA transfer. This bit is never set for ISO DMA transfer. The core automatically sets this bit when a transmit DMA channel has completed the programmed transfer by servicing the last IN transaction from the USB host. This is when TXDMAn.TXn_TSC (transfer size counter) equals 0, and the last IN transaction completes with an ACK. When this bit is asserted, the USB device controller must read the DMAN_STAT register to identify the endpoint number for which the transfer completed. The endpoint interrupt IRQ_SRC.EPn_TX is never set for the assigned endpoint to TX DMA channel n. 0: No action 1: Non-ISO transmit DMA transfer for a channel has ended. The value after MPU or USB reset is low.
9	RXn_CNT	The receive DMA channel n transactions count interrupt flag (non-ISO) bit is only for non-ISO DMA transfer. This bit is never set for ISO DMA transfer and never set if the interrupt enable bit associated with it is not set. It means that no poll operation is possible on the DMA. The core automatically sets this bit during an active Receive DMA transfer each time RXDMAn.RXn_TC equals 0 after an OUT transaction with ACK status. This bit is set after the RX DMA data have been read (end of DMA request). When this bit is asserted, the USB device controller must read DMAN_STAT register to identify the endpoint number for which the transfer completed. An RXn_CNT IT is asserted also if RXDMAn.RXn_STOP is set; in this case, both IRQ_SRC.RXn_EOT and IRQ_SRC.RXn_CNT are asserted. 0: No action 1: Non-ISO receive DMA transfer for a channel has reached transactions count level. The value after MPU or USB reset is low.

Table 10–42. Interrupt Source Register (IRQ_SRC) (Continued)

Bit	Name	Description
8	RXn_EOT	The receive DMA channel n end of transfer interrupt flag (non-ISO) bit is for non-ISO DMA transfer only. This bit is never set for ISO DMA transfer and never set if the interrupt enable bit associated with it is not set. It means that no poll operation is possible on the DMA. The core automatically sets this bit when a receive DMA channel detects an end-of-transfer (EOT) packet during the last OUT transaction from the USB host. This bit is set after the RX DMA data have been read (end of DMA request). When this happens, the DMA assigned endpoint FIFO is kept disabled (STAT_ FLG.FIFO_EN = 0) to avoid receiving a new packet data from the USB host. The USB device controller can grant DMA transfer again to the same endpoint by enabling the FIFO again (STAT_FLG.FIFO_EN = 1). An end of transfer is detected when the core receives a data packet that is less than the configured endpoint FIFO size (or is empty), or when RXDMAn.RXn_TC equals 0 after an OUT transaction with ACK status and the RXDMAn.RXn _STOP bit is set. When this bit is asserted, the USB device controller must read DMAN_STAT.DMAn_RX_IT_SRC to identify the endpoint number for which the transfer completed, and DMAN_STAT.DMAn_RX_SB must be informed of an odd number of bytes received during the last transaction (useful for 16-bit read access from the DATA_DMA register). The endpoint interrupt IRQ_SRC.EPn_RX is never set to RX DMA channel for the assigned endpoint. 0: No action 1: Non-ISO receive DMA transfer for a channel has ended. The value after MPU or USB reset is low.
7	SOF	Start of frame interrupt flag bit: every millisecond, the USB host outputs a start-of-frame packet to the functions. The SOF bit reflects when a new SOF is received. Writing a 1 in the SOF bit location clears the flag. Writing a 0 has no effect. In accordance with the USB1.1 specification, if SOF is received corrupted or is not received, the core still sets this flag at the same rate (if bit SOF.FT_LOCK = 1) or after 12043 USB bit time (if bit SOF.FT_LOCK = 0). 0: No action 1: Start-of-frame packet received (or internal SOF) The value after MPU or USB reset is low.
6	Reserved	Reserved
5	EPn_RX	The EPn OUT transactions interrupt flag bit only concerns non-ISO endpoints. The core automatically sets the EPn_RX bit when a handshake sequence occurs for an OUT transaction to an interrupt of the bulk endpoint (NAK with SYS-CON1.NAK_EN set, ACK or STALL). The USB device controller must read the EPn_STAT register to identify the endpoint causing the interrupt. 0: No action 1: OUT transaction detected on an endpoint The value after MPU or USB reset is low.

Table 10–42. Interrupt Source Register (IRQ_SRC) (Continued)

Bit	Name	Description
4	EPn_TX	The EPn IN transactions interrupt flag bit only concerns non-ISO endpoints. The core automatically sets this bit when a handshake sequence occurs for an IN transaction to an interrupt of bulk endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). The USB device controller must read the EPn_STAT register to identify the endpoint causing the interrupt. 0: No action 1: IN transaction detected on an endpoint value after MPU or USB reset is low.
3	DS_CHG	Device state changed interrupt flag bit: the core automatically resets the DS_CHG bit when the state of the device changes. This is when the core modifies any of the bits present in the DEVSTAT register. When this bit is cleared, the background DEVSTAT register moves into foreground position. 0: No action 1: Device state change detected Value after USB device controller hardware reset is low and after USB reset is high.
2	SETUP	Setup transaction interrupt flag bit: the core automatically sets the SETUP bit when a valid setup transaction completes on control endpoint for a non-autodecoded control request and automatically clears it when the USB device controller sets EP_NUM.SETUP_SEL bit when reading setup data. A write 1 to it has no effect. 0: No action 1: Valid setup transaction occurred on endpoint 0. Value after MPU or USB reset is low.
1	EP0_RX	EP0 OUT transactions interrupt flag bit: the core automatically sets the EP0_RX bit when a handshake sequence occurs for a non-autodecoded OUT transaction to control endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). 0: No action 1: OUT transaction on EP0 Value after MPU or USB reset is low.
0	EP0_TX	EP0 IN transactions interrupt flag bit: the core automatically sets this bit when a handshake sequence occurs for a non-autodecoded IN transaction to control endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). 0: No action 1: IN transaction on EP0 Value after MPU or USB reset is low.

IRQ_SRC is a read/clear only register that identifies and clears the source of the interrupt signaled by a set flag.

The value depends on whether the reset action comes from the USB device controller (MPU) or the USB host.

The USB device controller can clear a set bit location only by writing a 1 into the bit location (except for the setup bit, which is automatically cleared by the core). A write 0 has no effect.

When the core sets a bit location to 1, an interrupt is signaled to the USB device controller if the interrupt was enabled.

- 0: No interrupt
- 1: Interrupt signaled

The value after the MPU or USB reset is low except for the IRQ_SRC.DS_CHG bit, which is high after a USB reset.

Table 10–43. Non-ISO Endpoint Interrupt Status Register (EPN_STAT)

Bit	Name	Description
15:12	Reserved	Reserved
11:8	EPn_RX_IT_SRC	The receive endpoint interrupt source (non-ISO) bit only concerns non-ISO endpoints. When the IRQ_SRC.EPn_RX flag is set, the endpoint causing the interrupt condition is encoded in these four register bits. When the IRQ_SRC.EPn_RX flag is cleared, the four bits read as 0. 0000: No receive endpoint interrupt is pending. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all four bits).
7:4	Reserved	Reserved
3:0	EPn_TX_IT_SRC	Transmit endpoint interrupt source (non-ISO) bit only concerns non-ISO endpoints.When the IRQ_SRC.EPn_TX flag is set, the endpoint that causes this flag to be set is encoded in these four register bits. When the IRQ_SRC.EPn_TX flag is cleared, the four bits read as 0. 0000: No transmit endpoint interrupt is pending. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).

EPN_STAT is a read-only register that identifies the non-ISO endpoint causing an EPn interrupt. A write into it is denied.

Note:

If a nontransparent transaction occurs before a previous one on another endpoint in the same direction, the second interrupt is asserted only after the USB device controller clears the first one and EPN_STAT is updated with the corresponding interrupt assertion.

Table 10–44. Non-ISO DMA Interrupt Status Register (DMAN_STAT)

Bit	Name	Description
15:11	Reserved	Reserved
12	DMAn_RX_SB	The DMA receive single byte (non-ISO) bit only concerns non-ISO endpoints only (ISO endpoints receive a constant number of bytes). This bit is set when an IRQ_SRC.RXn_EOT interrupt is asserted and the core received an odd number of bytes during the last transaction. It is used to know the exact number of bytes received in case of a 16-bit read access from DATA_DMA register. When the IRQ_SRC.RXn_EOT flag is cleared, this bit reads as 0. 0: No EOT DMA interrupt is pending or the core received an even number of bytes during the last transaction. 1: An EOT DMA interrupt is pending and an odd number of bytes was received during the last transaction. Value after MPU or USB reset is low.
11:8	DMAn_RX_IT_SRC	The DMA receive interrupt source (non-ISO) bit only concerns non-ISO endpoints. When the IRQ_SRC.EPn_RX flag is set, the endpoint causing this flag to be set is encoded in these four register bits. When the IRQ_SRC.EPn_RX flag is cleared, the four bits read as 0. 0000: No receive DMA interrupt is pending. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).
7:4	Reserved	Reserved
3:0	DMAn_TX_IT_SRC	The DMA transmit interrupt source (non-ISO) bit only concerns non-ISO endpoints only. When the IRQ_SRC.EPn_TX flag is set, the endpoint that causes this flag to be set is encoded in these four register bits. When the IRQ_SRC.EPn_TX flag is cleared, the four bits read as 0. 0000: No transmit DMA interrupt is pending. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).

DMAN_STAT is a read-only register that identifies the endpoint that causes a DMAn interrupt. A write into it is denied.

Note:

If a DMA interrupt occurs before a previous one on another endpoint in the same direction, the second interrupt is asserted only after the USB device controller clears the first one. DMAn_STAT is updated when the corresponding interrupt is asserted.

Table 10–45. DMA Receive Channels Configuration Register (RXDMA_CFG)

Bit	Name	Description
15:13	Reserved	Reserved
12	RX_REQ	The RX DMA request active level or pulse bit allows the RXDMA request to be configurable level or pulse-sensitive. When pulse-sensitive, the request is active during 2 cycles. 0: RX DMA request active level 1: RX DMA request active pulse Value after MPU or USB reset is low.
11:8	RXDMA2_EP	Receive endpoint number for DMA channel 2. The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 2. A zero value indicates that the DMA channel 2 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint. 0000: Receive DMA channel 2 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).
7:4	RXDMA1_EP	Receive endpoint number for DMA channel 1. The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 1. A zero value indicates that the DMA channel 1 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint. 0000: Receive DMA channel 1 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).
3:0	RXDMA0_EP	Receive endpoint number for DMA channel 0. The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 0. A zero value indicates that the DMA channel 0 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint. 0000: Receive DMA channel 0 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).

RXDMA_CFG is a read/write register that enables the three possible DMA receive channels and selects the endpoint number that is assigned to each of these DMA channels. An endpoint used by an RX DMA channel must have been configured (through register EPn_RX). The RXDMA_CFG register can be filled when SYSCON1.CFG_LOCK is set.

Only one channel is serviced at a time, so it is better to configure ISO endpoints on the first channel (RXDMA0_EP). In this case, the ISO endpoint is configured on channel 2,and it can never be serviced with low software and a fast USB host.

The USB device controller transmit endpoint DMA channels 0, 1, and 2 are associated with OMAP1610 controller inputs.

CAUTION

System software must not program RXDMAx_EP to endpoint numbers that are not configured as DMA endpoints.

Table 10–46. DMA Transmit Channels Configuration Register (TXDMA_CFG)

Bit	Name	Description
15:13	Reserved	Reserved
12	TX_REQ	The TX DMA request active level or pulse bit allows the TXDMA request to be configurable level or pulse-sensitive. When pulse-sensitive, the request is active for two cycles. 0: TX DMA request active level 1: TX DMA request active pulse Value after MPU or USB reset is low.
11:8	TXDMA2_EP	Transmit endpoint number for DMA channel 2: the endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 2. A zero value indicates that the DMA channel 2 is deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint. 0000: Transmit DMA channel 2 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all four bits).

Table 10–46. DMA Transmit Channels Configuration Register (TXDMA_CFG) (Continued)

Bit	Name	Description
7:4	TXDMA1_EP	Transmit endpoint number for DMA channel 1: the endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 1. A zero value indicates that the DMA channel 1 is deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint. 0000: Transmit DMA channel 1 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all four bits).
3:0	TXDMA0_EP	Transmit endpoint number for DMA channel 0: the endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 0. A zero value indicates that the DMA channel 0 is deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint. 0000: Transmit DMA channel 0 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all four bits).

TXDMA_CFG is a read/write register that enables the three possible DMA transmit channels and selects the endpoint number that is assigned to each. An endpoint used by a TX DMA channel must have been configured (through register EPn_TX). TXDMA_CFG register can be filled when SYSCON1.CFG_LOCK is set.

Only one channel is serviced at a time, so it is better to configure ISO endpoints on the first channel (TXDMA0_EP). In this case, your ISO endpoint is configured on the channel 2 and it can never be serviced with low software and a fast USB host.

USB device controller transmit endpoint DMA channels 0, 1, and 2 are associated with OMAP1610 controller inputs.

Table 10–47. DMA FIFO Data Register (DATA_DMA)

Bit	Name	Description
15:0	DATA_DMA	DMA FIFO data. When an RX DMA request is active for a channel (only one active at a given time), this register contains the data that the core received from the USB host OUT transaction using this channel. Data can be accessed by the main DMA controller engine (read access) in response to the DMA request for the channel. When a TX DMA request is active for a channel (only one active at a given time), this register contains the data written by the main DMA controller engine (write access) in response to a DMA request for the transmit channel to be sent to the USB host during the next IN transaction. Warning: It is possible for both an RX DMA request and a TX DMA request to be active at the same time. In this case, the main DMA controller engine can access both transmit endpoint and receive endpoint FIFO. A read access to DATA_DMA register affects the endpoint that caused the RX DMA request to be active, and a write access affects the endpoint that caused the TX DMA request to be active. **Caution:** The USB device controller must not access this register directly; however, there is no hardware mechanism to prevent such access. No access into this register must be made out of DMA request handling.

The DMA FIFO data register is the entry point to write or to read data into/from an endpoint used in a DMA transfer through DMA channel 0, 1 or 2.

Table 10–48. Transmit DMA Control Register n (TXDMAn)

Bit	Name	Description
15	TXn_EOT	Transmit DMA channel n end of transfer: the TXn_EOT bit can be either 0 or 1 for BULK DMA transfer. When the USB device controller sets it to 1, this bit signals the core that the transfer size set in TXDMAn.TXn_TSC is in bytes. A TX one interrupt (IRQ_SRC.TXn_DONE) is asserted with the last IN transaction. If the number of bytes set in TXDMAn.TXn_TSC is a multiple of the endpoint buffer size, the TX done interrupt is asserted only after an IN transaction with an empty data packet. When cleared, the transfer size set in TXn_TSC is in full buffer size for the endpoint selected (BULK only). A TX done interrupt is asserted when the last buffer is sent with the last IN transaction. This mode is to be used for a partial bulk transfer of a large file exceeding 1023 bytes. 0: DMA transfer size is in buffers. 1: DMA transfer size is in bytes. Value after MPU or USB reset is low.
14	TXn_START	Transmit DMA channel n start. The USB device controller sets this bit to tell the device that the main DMA system is ready to transmit the number of bytes or buffers. Once set, the DMA transfer cannot be interrupted, unless the USB device controller clears the endpoint in the TXDMA_CFG register. A write 0 into this bit has no effect and a read to this bit always returns 0. The IRQ_SRC.TXn_DONE interrupt bit is asserted when the DMA transfer ends. 0: No action 1: DMA transfer start Always reads 0

Table 10–48. Transmit DMA Control Register n (TXDMAn) (Continued)

Bit	Name	Description
13:10	Reserved	Reserved
9:0	TXn_TSC	Transmit DMA channel n transfer size counter. The binary encoded value from 0 to 1023, which the USB device controller writes into this register, corresponds to the number of bytes or number of buffer transfers (function of TXDMAn.TXn_EOT) to be transmitted by the transmit DMA channel n. When read, the register reflects the number of bytes/buffers the USB device has still to transmit. Read mode is only provided for software debug purposes. **Caution:** For ISO transfer, the user must verify that the set value does not exceed the ISO FIFO size for the endpoint. There is no hardware mechanism to prevent this situation. If this situation occurs, results are unpredictable. **Caution:** For bulk transfer, when TXDMAn.TXn_EOT = 0, a set value of TXDMAn.TXn_TSC = 0 means 1024 buffers and not 0. The counter then operates in the following way: 000, 3FF, 3FE, 0001, 000, stop. When TXDMAn.TXn_EOT = 1, a set value of TXDMAn.TXn_TSC = 0 a NULL packet is sent in response to the next IN token. Values after MPU or USB reset are low (all 10 bits).

TXDMAn is a read/write register that controls the operation of the transmit DMA channel n (n = 0, 1, 2).

Table 10–49. Receive DMA Control Register n (RXDMAn)

Bit	Name	Description
15	RXn_STOP	Receive DMA channel n transfer stop. When this bit is set, an RXn_EOT interrupt is asserted to the USB device controller after n OUT transactions where n is the encoded binary value + 1 programmed into RXDMAn.RXn_TC field. This register is used when no smaller than buffer size packet is received at an end-of-transfer (EOT), and the USB device controller expects a given amount of data for the transfer. **Caution:** At end of transfer, the DMA channel is disabled and all OUT transactions received to the assigned endpoint are sent NAK by the core. The USB device controller must set CTRL.SET_FIFO_EN for the endpoint to reenable the channel. Values after MPU or USB reset are low.
14:8	Reserved	Reserved
7:0	RXn_TC	Receive DMA channel n transactions count. The USB device controller can ask for an interrupt each n OUT transactions where n is the encoded binary value + 1 programmed into RXDMAn.RXn_TC field. This register must be programmed to the desired transaction watermark limit prior to enabling the DMA transfer for the receive DMA channel n. **Caution:** A reached watermark does not disable an active DMA transfer if RXDMAn.RXn_STOP was not set. If RXDMAn.RXn_STOP was set for the transfer both RXn_CNT and RXn_EOT interrupts are asserted. A read to that register returns the number of transactions remaining before the IRQ_SRC.RXn_CNT interrupt flag is asserted. This read mode is only provided for software debug purposes. Values after MPU or USB reset are low (all 8 bits).

RXDMAn is a read/write register that monitors incoming OUT transactions during DMA transfer on channel n (n=0,1,2).

Table 10–50. Endpoint 0 Configuration Register (EP0)

Bit	Name	Description
15:14	Reserved	Reserved
13:12	EP0_SIZE	Endpoint 0 FIFO size. This field contains the endpoint 0 FIFO size value and must match the value sent by the USB device controller to the USB host during the GET_DEVICE_DESCRIPTOR request preceding configuration phase. Status flags (STAT_FLG.NON_ISO_FIFO_EMPTY and STAT_FLG.NON_ISO_FIFO_FULL) and overrun and underrun conditions are based on this value for all IN and OUT transactions to endpoint 0. The USB device controller must fill this field before setting SYSCON1.CFG_LOCK. 00: 8 bytes 01: 16 bytes 10: 32 bytes 11: 64 bytes Values after USB device controller hardware reset or USB reset are unchanged (which means that value is unknown until first write access).
11	Reserved	Reserved
10:0	EP0_PTR	Endpoint 0 pointer. This field contains the address of the endpoint 0 pointer. Value 0x000 is not permitted (reserved for setup FIFO). 0x000: address = BASE (not permitted) 0x001: address = BASE + 8 bytes 0x002: address = BASE + 16 bytes 0x003: address = BASE + 24 bytes ... 0x0FF: address = BASE + 2040 bytes Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access). Note: The pointer value must be set to a value < 0xFF, because the memory size is 2K bytes and a pointer coded value = 0xFF corresponds to 2040 bytes. Addressing upper bytes results in memory overlap.

EP0 is a read/write register that gives the device configuration for control endpoint 0.

Values depend on whether the reset action comes from USB device controller (MPU) or the USB host.

Table 10–51. Receive Endpoint n Configuration Register (EPn_RX)

Bit	Name	Description
15	EPn_RX_VALID	The receive endpoint n valid bit must be set by the USB device controller to allow receive endpoint n to be used for USB transfers as part of the device configuration. If not set, all transactions to this endpoint are ignored by the core. 0: Receive endpoint n does not exist for this configuration. 1: Receive endpoint n is part of the device configuration. Values after USB device controller hardware reset are low, after USB reset is unchanged.
14	EPn_RX_SIZE/DB	Receive non-ISO (or ISO) endpoint n double-buffer (DB).This bit is only for non-ISO endpoints. For ISO endpoints, which are always double-buffered, this bit is endpoint size MSB. This bit must be set by the USB device controller to allow double buffering for receive non-ISO endpoint n. This is used to reduce number of transactions resulting in NAK handshake. 0: No double buffer for non-ISO receive endpoint n. 1: Double buffer used for non-ISO receive endpoint n. Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access).
13:12	EPn_RX_SIZE	The receive endpoint n size field contains the endpoint n FIFO size value. Status flags (STAT_FLG.NON_ISO_FIFO_EMPTY, STAT_FLG.NON_ISO_FIFO_FULL, STAT_FLG. ISO_FIFO_EMPTY, and STAT_FLG.ISO_FIFO_FULL) and overrun and underrun conditions are based on this value for all OUT transactions to endpoint n. Non-ISO: 00: 8 bytes ISO: 000: 8 bytes [13:12] 01: 16 bytes [14:12] 001: 16 bytes 10: 32 bytes 010: 32 bytes 11: 64 bytes 011: 64 bytes 100: 128 bytes 101: 256 bytes 110: 512 bytes 111: 1023 bytes Warning: For ISO endpoints, a size of 1023 bytes takes up the whole memory and prevents it from being programmed with a 2K-byte USB device controller. Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access).

Table 10–51. Receive Endpoint n Configuration Register (EPn_RX) (Continued)

Bit	Name	Description
11	EPn_RX_ISO	The receive ISO endpoint n field must be set if the receive endpoint n type is isochronous in the desired device configuration. If not set, the endpoint type is bulk or interrupt (the hardware does not distinguish bulk type from interrupt). 0: Receive endpoint n type is isochronous. 1: Receive endpoint n type is bulk or interrupt. Values after USB device controller hardware reset or USB reset are unchanged.
10:0	EPn_RX_PTR	The receive endpoint n pointer field contains the address of the receive endpoint n pointer. Value 0x000 is not permitted (reserved for setup FIFO). Caution: For ISO endpoints or for non-ISO endpoints that allow double-buffering, 2*RX buffer size must be reserved for ping-pong. 0x000: address = BASE (not permitted) 0x001: address = BASE + 8 bytes 0x002: address = BASE + 16 bytes 0x003: address = BASE + 24 bytes ... 0x0FF: address = BASE + 2040 bytes Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access). Note: Pointer value must be set to a value < 0xFF, because the memory size is 2K bytes and a pointer coded value = 0xFF corresponds to 2040 bytes. Addressing upper bytes results in memory overlap.

EPn_RX is a read/write register that gives the device configuration for non-control receive endpoint n (n: 115). The endpoints size fields must match values sent by the USB device controller to the USB host in response to the GET_CONFIGURATION_DESCRIPTOR during configuration phase.

The USB device controller must fill this field before setting SYSCON1.CFG_LOCK and must not change the values once SYSCON1.CFG_LOCK is set.

Values depend on whether the reset action comes from USB device controller (MPU) or USB host.

Table 10–52. Transmit Endpoint n Configuration Register (EPn_TX)

Bit	Name	Description
15	EPn_TX_VALID	The transmit endpoint n valid bit must be set by the USB device controller to allow transmit endpoint n to be used for USB transfers as part of the device configuration. If not set, all transactions to this endpoint are ignored by the core. 0: Transmit endpoint n does not exist for this configuration. 1: Transmit endpoint n is part of the device configuration. Values after USB device controller hardware reset are low, after USB reset is unchanged.
14	EPn_TX_SIZE/Db	Transmit non-ISO (or ISO) endpoint n double buffer. This bit is only for non-ISO endpoints used in DMA mode. For ISO endpoints, which are always double buffered, this bit is endpoint size MSB. For non-ISO endpoints that are not used in a DMA transfer, double buffering is not provided.This bit must be set by the USB device controller to allow double buffering for transmit non-ISO endpoint n, when used in a DMA transfer. This is used to reduce the number of transactions resulting in NAK handshake. 0: No double buffer for non-ISO transmit endpoint n. 1: Double buffer used for non-ISO transmit endpoint n. Values after MPU or USB reset is unchanged. Or transmit ISO endpoint n size[2]
13:12	EPn_TX_SIZE	Transmit endpoint n size

Table 10–52. Transmit Endpoint n Configuration Register (EPn_TX) (Continued)

Bit	Name	Description
11	EPn_TX_ISO	The transmit ISO endpoint n field must be set if the transmit endpoint n type is isochronous in the desired device configuration. If not set, the endpoint type is bulk or interrupt (the hardware does not distinguish bulk type from interrupt). 0: Transmit endpoint n type is isochronous. 1: Transmit endpoint n type is bulk or interrupt. Values after MPU or USB reset is unchanged.
10:0	EPn_TX_PTR	The transmit endpoint n pointer field contains the address of the transmit endpoint n pointer. Caution: For ISO endpoints or for non-ISO endpoints that allow double-buffering, (2*TX buffer size) must be reserved for ping-pong. 0x000: address = BASE 0x001: address = BASE + 8 bytes 0x002: address = BASE + 16 bytes 0x003: address = BASE + 24 bytes ... 0x0FF: address = BASE + 2040 bytes Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access). Pointer value must be set to a value < 0xFF, because the memory size is 2K bytes and a pointer coded value = 0xFF corresponds to 2040 bytes. Addressing upper bytes results in memory overlap.

EPn_TX is a read/write register that gives the device configuration for non-control transmit endpoint n (n: 115). The endpoints size fields must match the values sent by the USB device controller to the USB host in response to the GET_CONFIGURATION_DESCRIPTOR during configuration phase.

Note:

The USB device controller must fill this field before setting SYSCON1.CFG_LOCK and must not change the values once SYSCON1.CFG_LOCK is set.

10.3.1.1 EPn_TX[14:12].EPn_TX_SIZE: Transmit Endpoint n Size

EPn_TX.[14] bit description only applies for ISO endpoints.

This field contains the endpoint n FIFO size value. Status flags (FIFO_EMPTY, FIFO_FULL) and underrun condition are based on this value for all IN transactions to endpoint n.

Non-ISO: [13:12]	00:	8 bytes	ISO: [14:12]	000:	8 bytes
	01:	16 bytes		001:	16 bytes
	10:	32 bytes		010:	32 bytes
	11:	64 bytes		011:	64 bytes
				100:	128 bytes
				101:	256 bytes
				110:	512 bytes
				111:	1023 bytes

ISO Endpoints

For ISO endpoints, a size of 1023 bytes takes up the whole memory and prevents programming with a 2K-byte USB device controller.

Values after USB device controller hardware reset or USB reset are unchanged (value is unknown until first write access).

10.3.2 USB Device Transactions

There is an interrupt to the MPU at the end of an USB transaction if the MPU has actions to perform. Isochronous transactions are an exception because isochronous interrupt information is available at start of frame interrupts. The MPU ISR code determines which endpoint and direction caused the interrupt and acts appropriately. The following subsections describe in detail the activities surrounding USB transactions that are not part of a DMA transfer. Cases where a transaction occurs before the previous one has been handled by the MPU are not taken into account in this part. The information is organized so that each section deals with one type and direction of transaction, such as:

- Non-isochronous, non-setup OUT transactions
- Non-isochronous IN transactions
- Isochronous OUT transactions
- Isochronous IN transactions

This allows each section to focus only on a specific style of transaction without adding in the confusion of special cases related to other styles.

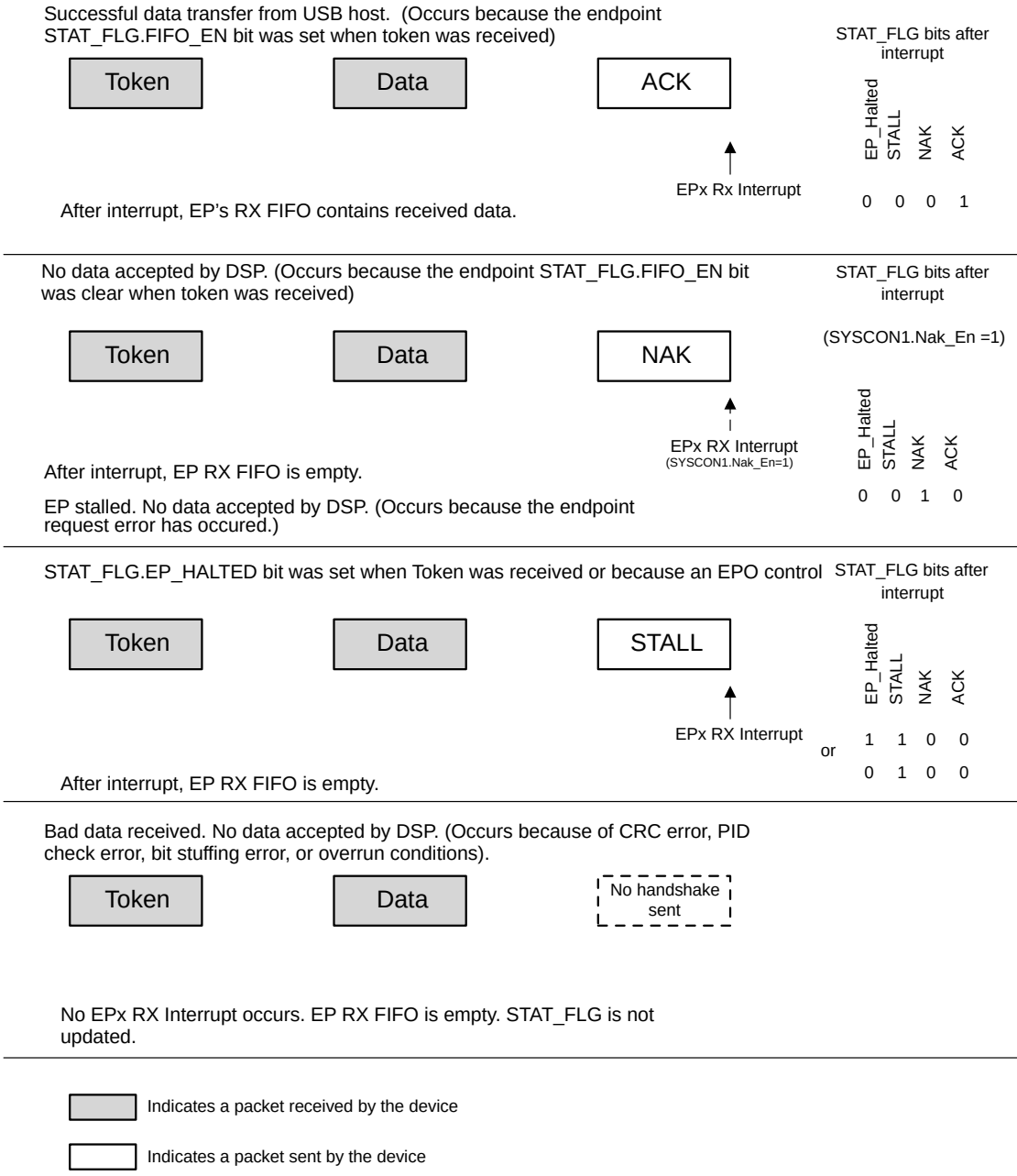
10.3.3 Non-Isochronous, Non-Setup OUT (USB HOST -> MPU) Transactions

Non-isochronous, non-setup OUT transactions refer to USB transactions where data is moved from the USB host to the MPU, the USB handshaking protocols are in effect, and data transmission is ensured. These types of transactions apply to all OUT transactions on bulk and interrupt endpoint types, and to non-setup transactions on control endpoints.

Figure 10–3 shows the various USB protocol conditions that can occur during non-isochronous, non-setup OUT transactions. The diagram shows the three phases that can occur in an OUT transaction, the direction of information flow

for each phase, when endpoint interrupts are generated, and the resulting STAT_FLG bits for the endpoint. The top three cases show the normal USB handshaking modes: acknowledge (good data received), NAK (device not ready to receive data), and STALL (device in a condition where the endpoint cannot handle OUT transactions). The last case is an abnormal instance where the token packet or the data packet was received with errors. The RX FIFO only contains valid receive data under the first (acknowledge) case.

Figure 10–3. Non-Isochronous, Non-Control OUT Transaction Phases and Interrupts



10.3.3.1 Non-Isochronous, Non-Control OUT Endpoint Handshaking Conditions

An endpoint CTRL.SET_FIFO_EN bit provides the main control for the endpoint ability to allow successful OUT transaction data reception for the end-

point. If at the beginning of an OUT transaction to an endpoint, the endpoint STAT_FLG.FIFO_EN bit is 1, the USB module is allowed to accept the OUT transaction data to the RX FIFO, and, when the transaction completes, the USB module can return ACK to the PC to indicate that the data was received correctly (this is the top case shown in Figure 10–3). If, however, the endpoint STAT_FLG.FIFO_EN bit was 0 at the beginning of an OUT transaction to the endpoint, the USB module returns NAK during the handshake phase to indicate that the endpoint did not accept the data (the second case shown in Figure 10–3).

The USB host need not send a whole RX FIFO worth of data to the endpoint during an OUT transaction. In this case, the RX FIFO is not full when the endpoint RX interrupt is generated. The MPU code must be careful not to read too much data. MPU code must read RXFSTAT.RXF_COUNT value before reading data from the RX FIFO.

At the end of an USB OUT transaction to an endpoint where the data is accepted (ACKed), the hardware clears the endpoint STAT_FLG.FIFO_EN bit. Once the MPU software has dealt with the OUT transaction data in the endpoint RX FIFO, it must re-enable the endpoint OUT transaction reception by setting the endpoint CTRL.SET_FIFO_EN bit. MPU software can use the endpoint CTRL.SET_FIFO_EN bit as a receive flow control mechanism.

Acknowledged Transactions (ACK)

At completion of an OUT transaction to an endpoint, the USB module issues an endpoint-specific interrupt to the MPU and STAT_FLG is updated. In response to the endpoint interrupt, the MPU must read EPN_STAT register to identify the endpoint causing the interrupt, then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.ACK is set to indicate that the endpoint received a transaction to which the USB module signaled ACK handshaking.

If STAT_FLG.FIFO_EMPTY is cleared, the transaction sent 1 or more bytes of data (but less than or equal to the physical size of the endpoint RX FIFO), and the data is in the endpoint RX FIFO. The MPU knows the number of bytes to read from RX FIFO by reading RXFSTAT.RXF_COUNT value. The MPU can then read RX data from DATA register. Once the MPU has read the data from the FIFO, it sets the CTRL.SET_FIFO_EN bit to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO, and then clears the EP_NUM.EP_SEL bit. This clears the STAT_FLG.ACK bit for this endpoint to allow next transaction status to be written into the STAT_FLG register.

Non-Acknowledged Transactions (NAK)

The device can be configured via the SYSCON1.NAK_EN either to inform the MPU of a NAKed transaction or not. If SYSCON1.NAK_EN is cleared, no interrupt is asserted to the MPU if an OUT transaction completes with a NAK handshake and STAT_FLG.NAK bit is not set. If SYSCON1.NAK_EN is set, the USB module issues an endpoint-specific interrupt to the MPU at completion

of an OUT transaction to an endpoint, and STAT_FLG.NAK bit is set. In response to the endpoint interrupt, the MPU must read EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.NAK is set to indicate that the endpoint received a transaction to which the USB module signaled NAK handshaking.

The MPU must set the CTRL.SET_FIFO_EN bit to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO, and then clear the EP_NUM.EP_SEL bit. This clears the STAT_FLG.NAK bit for this endpoint to allow the next transaction status to be written into the STAT_FLG register.

10.3.3.2 Non-Isochronous, Non-Control OUT Transaction Error Conditions

STALLed Transactions

The USB module responds to an endpoint OUT transaction with a STALL handshake to indicate an error condition on the endpoint either if the endpoint STAT_FLG.EP_HALTED bit is set or if a request error occurs (control transactions only). When an endpoint OUT transaction is given a STALL handshake, the endpoint STAT_FLG.STALL bit is set and an endpoint-specific interrupt is generated for the endpoint. STAT_FLG.FIFO_EN is of lower priority than STAT_FLG.EP_HALTED; when the STAT_FLG.EP_HALTED bit is set and transactions to the RX endpoint are stalled, regardless of the STAT_FLG.FIFO_EN value. If STAT_FLG.FIFO_EN is set, the STAT_FLG.FIFO_EN bit is automatically cleared at the end of the STALLed transaction and RX FIFO is cleared.

In response to the endpoint interrupt, the MPU must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.STALL is set to indicate that the endpoint received a transaction to which the USB module signaled STALL handshaking.

If STAT_FLG.EP_HALTED has been set by the MPU and can be removed, the MPU must set CTRL.CLR_HALT to clear the condition and set CTRL.SET_FIFO_EN to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO. If STAT_FLG.EP_HALTED has been set in response to a SET_FEATURE request sent by the USB host, or if the bit is cleared (control transaction only), the MPU has no action to perform and must clear the EP_NUM.EP_SEL bit. This clears the STAT_FLG.STALL bit for this endpoint and allows the next transaction status to be written into the STAT_FLG register.

Packet Errors

In case of a receive data error during an endpoint OUT transaction (token or data packet), the USB module does not provide a handshake during the handshake phase of the transaction and no interrupt is asserted to the MPU (the

fourth case shown in Figure 10–3). Additionally, the endpoint RX FIFO is not filled, and STAT_FLG.FIFO_EN bit is not cleared. If the MPU clears the RX FIFO during the data packet of an OUT transaction, no handshake is returned to the USB host to signal an error.

Sequence Bit Errors

If the core does not receive expected DATA PID during an OUT transaction, the module automatically returns ACK handshake to the USB host, regardless of the STAT_FLG.FIFO_EN bit (per the USB specification). Data is ignored, and no interrupt is asserted to the MPU.

This error occurs if ACK handshake from previous OUT transaction is received corrupted by the USB host.

10.3.3.3 Non-Isochronous, Non-Control OUT Endpoint FIFO Error Conditions

If the USB host attempts to fill more data into an endpoint RX FIFO than the FIFO can hold, a FIFO overrun occurs. The USB module does not provide a handshake during the handshake phase of the transaction and no interrupt is asserted to the MPU. Additionally, the endpoint RX FIFO is not filled, and STAT_FLG.FIFO_EN bit is not cleared.

The MPU must not read more data from RX FIFO than the value indicated by RXFSTAT.RXF_COUNT.

10.3.4 Non-Isochronous IN (MPU->USB HOST) Transactions

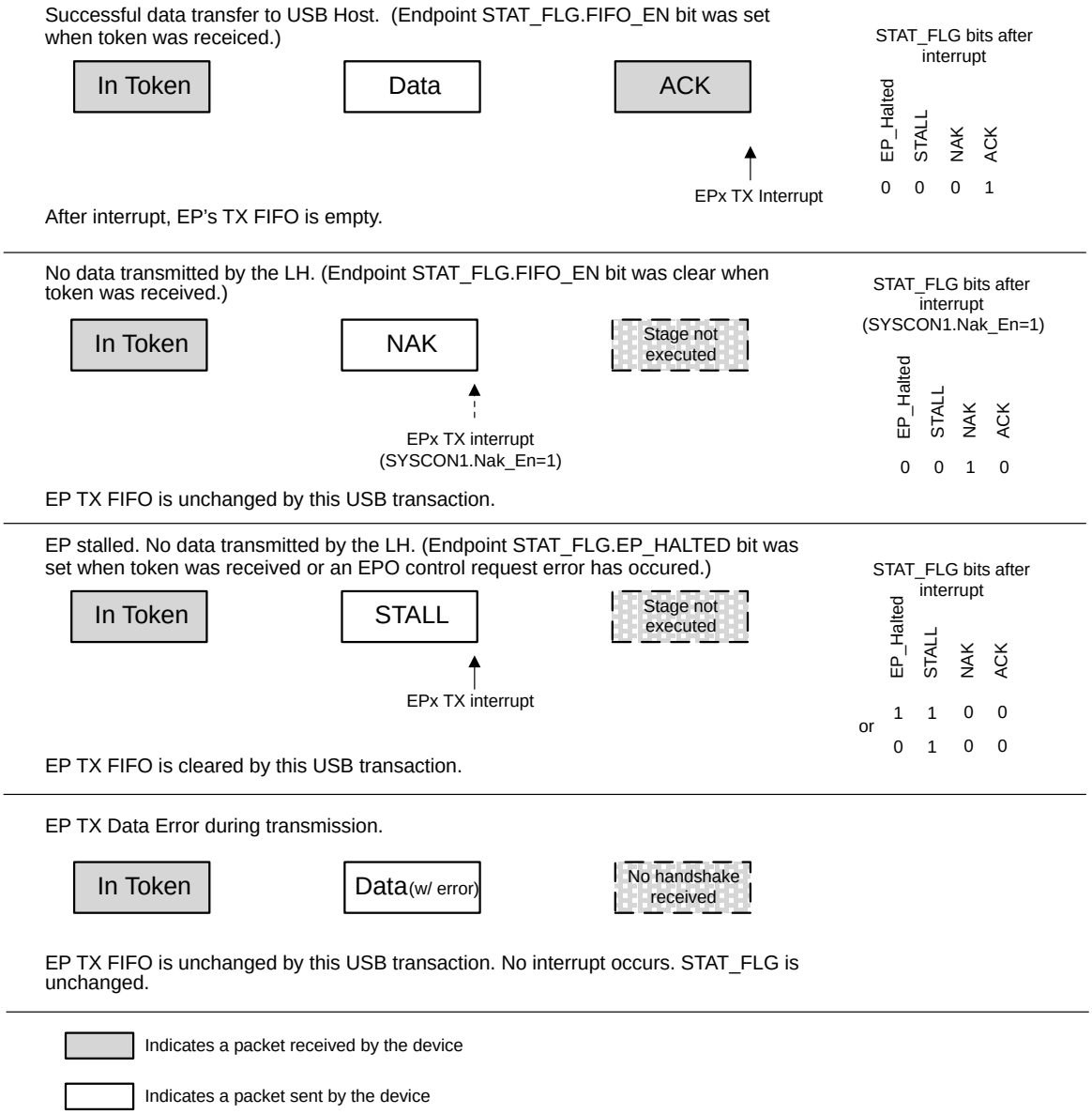
Non-isochronous IN transactions refer to USB transactions in which data is moved from the MPU to the USB host, where the USB handshaking protocols are in effect, and data transmission is ensured. These transactions are the IN transactions that occur on control, bulk, and interrupt endpoints. These transactions do not ensure USB bandwidth.

To provide data for an endpoint IN transaction, the MPU code writes the transmit data into the endpoint transmit FIFO. MPU code must first wait until the USB is done with any previous TX data for the endpoint (if data had previously been written to the TX FIFO). This must be done by proper response to endpoint-specific transmit interrupts. When an IN transaction to the endpoint occurs, if the endpoint STAT_FLG.FIFO_EN bit is set, the USB module sends any data that is in the endpoint TX FIFO during the data phase. If the TX FIFO is empty and STAT_FLG.FIFO_EN is set when an IN transaction to the endpoint occurs, a 0-byte data packet is sent.

Once the endpoint previous transmit activity is taken care of, the MPU code gains access to endpoint FIFO and status by setting the EP_NUM.EP_SEL bit. Then the MPU can write the new transmit data to the endpoint TX FIFO via the DATA register (being careful not to overflow the FIFO). Once all of the transmit data has been written to the endpoint FIFO, MPU code sets the CTRL.SET_FIFO_EN bit to allow the USB to use the endpoint TX FIFO, and then clears the EP_NUM.EP_SEL bit. The data in the endpoint TX FIFO is sent to the USB host the next time an IN transaction to the endpoint occurs.

Figure 10–4 shows the various USB protocol conditions that can occur during non-isochronous IN transactions. It diagrams the three phases of the IN transaction, the direction of information flow for each phase, when endpoint-specific interrupts are generated, and the resulting STAT_FLG bits for the endpoint. The top three cases show the normal USB handshaking: acknowledge (data sent by USB module and received properly by the USB host), NAK (device not ready to send data to USB host), and STALL (device in a condition where the endpoint cannot handle IN transactions). The last case shows an abnormal case in which there is an error either in the token packet received by the core or in the data packet received by the USB host.

Figure 10–4. Non-Isochronous IN Transaction Phases and Interrupts



10.3.4.1 Non-Isochronous IN Endpoint Handshaking

Per USB specifications for IN transactions, the USB host can provide only one of two handshakes to the USB function during the handshake phase: ACK or no handshake at all. The first indicates successful transfer (first case shown in Figure 10–4), and the second indicates that the host received a garbled data packet (last case shown in Figure 10–4).

Acknowledged Transactions (ACK)

When the endpoint IN transaction completes on the USB bus with an ACK handshake, the endpoint generates an endpoint-specific interrupt to the MPU (see first case in Figure 10–4). In response to the endpoint interrupt, the MPU must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number, EP_NUM.EP_DIR to 1 (to signal an IN endpoint), and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.ACK is set to indicate that the endpoint received an ACK handshake from the USB host, and the TX FIFO is empty (because any data that was in the TX FIFO was transmitted during the IN transaction).

If the MPU has more data to transmit to the USB host, it must fill the TX FIFO following the process indicated above. It must then clear EP_NUM.EP_SEL bit. This clears the STAT_FLG.ACK bit for this endpoint to allow next transaction status to be written into the STAT_FLG register.

Non-Acknowledged Transactions (NAK)

For the case in which the MPU is not ready to provide transmit data for transactions to an IN endpoint, the core provides a NAK handshake to the host for any USB IN transaction to that endpoint . Readiness to transmit data is signaled via the endpoint STAT_FLG.FIFO_EN bit; when 1 it indicates that data in the TX FIFO can be sent to the USB host. When the endpoint STAT_FLG.FIFO_EN bit is 0 and an IN transaction to the endpoint occurs, NAK handshake is sent, indicating that the MPU is not ready to handle the request.

If SYSCON1.NAK_EN bit is cleared, when the NAK handshake is sent in the data packet portion of the transaction to the IN endpoint, STAT_FLG is not updated and no endpoint-specific interrupt to the MPU is generated. If the SYSCON1.NAK_EN bit is set, when the NAK handshake is sent in the data packet portion of the transaction to the IN endpoint, the STAT_FLG.NAK bit is set and an endpoint-specific interrupt to the MPU is generated.

In response to the endpoint interrupt, the MPU must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number, EP_NUM.EP_DIR to 1 (to signal an IN endpoint), and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.NAK is set to indicate that the endpoint sent a NAK handshake to the USB host. If the MPU has data to transmit to the USB host,

it must fill the TX FIFO following the process indicated above. The MPU must then clear the EP_NUM.EP_SEL bit. That clears the STAT_FLG.NAK bit for this endpoint to allow the next transaction status to be written into the STAT_FLG register. Signaling NAK does not cause the endpoint TX FIFO to be cleared (because the MPU still retains control of the FIFO).

Signaling NAK handshake for several endpoint transactions in a row can cause the PC host to discard the transaction, so NAK is not necessarily a good mechanism in cases where the MPU is not able to service a request for long periods of time.

10.3.4.2 Non-Isochronous IN Transaction Error Conditions

STALLed Transactions

The USB module sends a STALL handshake to the USB host during the data phase of the transaction to the IN endpoint either if the endpoint STAT_FLG.EP_HALTED flag is set, or if a request error occurs (control transaction only). A USB STALL handshake indicates that the device endpoint is in a condition in which it is not able to transfer data and instructs the USB host not to retry the transaction. The device typically requires intervention via some other mechanism to clear the condition, usually a control transfer via endpoint 0. The MPU can set the endpoint EP_HALTED bit by selecting the endpoint by writing the appropriate value in EP_NUM register, and then setting the endpoint CTRL.Set_HALT bit, and clear it by selecting the endpoint, and then setting the endpoint CTRL.Clr_HALT bit. When the endpoint EP_HALTED bit is set, the endpoint signals STALL for its IN transactions until the HALT condition is cleared. When the STALL handshake is sent in response to a transaction to the endpoint, the STAT_FLG.STALL bit is set, and an endpoint-specific interrupt to the MPU is generated.

In response to the endpoint interrupt, the MPU must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number, EP_NUM.EP_DIR to 1 (to signal an IN endpoint), and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.STALL is set to indicate that the endpoint sent a STALL handshake to the USB host. The MPU must then clear EP_NUM.EP_SEL bit. This clears the STAT_FLG.STALL bit for this endpoint and allows the next transaction status to be written into the STAT_FLG register.

Except for control endpoint 0, separate endpoint halt bits are defined for each direction; so for a given endpoint number, the TX can be halted when the RX is not.

Packet Errors

If an error (CRC, bit stuffing, or PID check) occurs during the token packet of a USB IN transaction to a non-isochronous endpoint, the USB block ignores the transaction. No endpoint-specific interrupt to the MPU occurs for transac-

tions with corrupted packets. If the MPU clears the TX FIFO during the data packet of an IN transaction, a bit stuffing error is forced.

If the USB host returns no handshake after an IN transaction (in case of an error during transmission), the USB device controller module detects after a time-out that an error has occurred. The data to transmit is still in the TX FIFO to be re-sent during next IN transaction, STAT_FLG.FIFO_EN is not cleared, and no interrupt is asserted to the MPU.

10.3.4.3 Non-Isochronous IN Endpoint FIFO Error Conditions

The MPU cannot write more data into the TX FIFO than the configured FIFO size.

10.3.5 Isochronous OUT (USB HOST-> MPU) Transactions

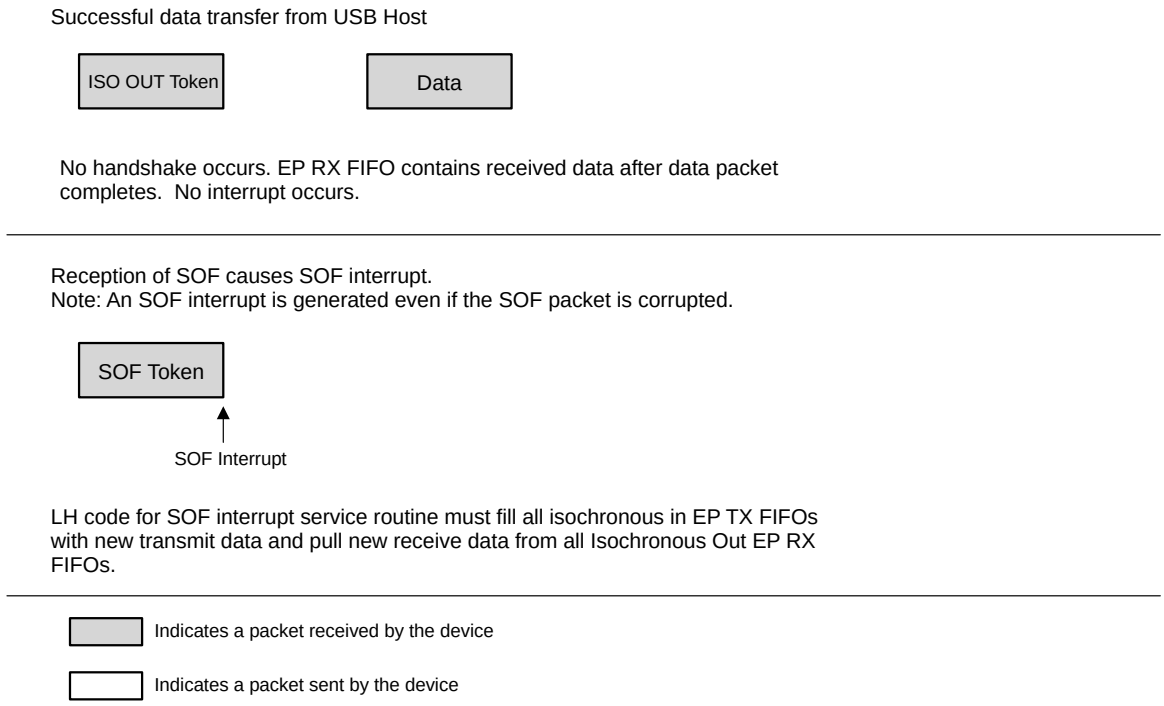
Isochronous OUT transactions are USB transactions in which a given amount of data is transferred from the USB host to the USB device controller module device every 1-ms USB frame. No USB handshaking is provided, and no endpoint-specific interrupt to the MPU is generated at completion of an isochronous OUT transaction. The MPU is responsible for handling isochronous OUT data at each start of frame (SOF) interrupt.

At every SOF interrupt, for each isochronous OUT endpoint, MPU code must select the endpoint by writing the appropriate value in the EP_NUM register and check STAT_FLG.ISO_FIFO_EMPTY. If the RX FIFO contains data, code must read the RXFSTAT.RXF_COUNT value (if the number of bytes to read from RX FIFO is not known), read all the bytes from RX FIFO via the data register, and then clear the EP_NUM.EP_SEL bit.

Because the USB transaction for the isochronous endpoint can occur at any time during the USB 1-ms frame, the USB interface implements a double-buffering of the endpoint receive data FIFO. The endpoint includes two FIFOs, each of which is the length of the configured isochronous endpoint. At all times, one of the two FIFOs is foreground and the other is background. The USB interface side of the USB module is allowed to write to the background RX FIFO, and the MPU is allowed to read to the foreground RX FIFO. The designations foreground and background are swapped at each start of frame (SOF). Isochronous endpoint FIFOs in the background are always enabled to the USB, whereas the foreground FIFOs are enabled to the MPU.

Figure 10–5 shows the two phases (ISO OUT token and data) of an isochronous OUT data transfer in the top portion of the figure. No endpoint-specific interrupt to the MPU is generated for the isochronous OUT transaction. The data for isochronous endpoints are instead handled by the MPU at each start of frame (SOF) interrupt, which is shown as the second case in Figure 10–5.

Figure 10–5. Isochronous OUT Transaction Phases and Interrupts



10.3.5.1 Isochronous OUT Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STAT_FLG.STALL, STAT_FLG.NAK, and STAT_FLG.ACK bits for isochronous endpoints always return 0. Because there is no handshake, the endpoint-specific interrupt for isochronous endpoints is not used.

10.3.5.2 Isochronous OUT Transaction Error Conditions

If the MPU fails to read all of the data in the ISO OUT endpoint foreground FIFO by the time the foreground and background FIFOs are switched (at the next SOF), the endpoint FIFO that is being switched to the background is flushed, and the STAT_FLG.DATA_FLUSH bit is asserted for the duration of the next frame.

There is no special indication for the case in which the USB host does not provide a transaction to an ISO OUT endpoint during a frame, but once the FIFO that was background in that frame is foreground, the FIFO is empty (a 0-length data ISO OUT transaction also results in an empty FIFO and cannot be distinguished from a missed ISO OUT transaction).

If an ISO OUT transaction occurs with data error (CRC, PID check, or bit stuffing), the RX FIFO is empty at the next SOF interrupt, and the STAT_FLG.ISO_ERR bit is asserted for the duration of the next frame.

10.3.5.3 Isochronous OUT Endpoint FIFO Error Conditions

The MPU must never read more data than the value given by RXFSTAT.RXF_COUNT.

If the USB host sends more data than the FIFO can contain, the FIFO is cleared and the STAT_FLG.ISO_ERR is set at the next SOF interrupt. A properly configured USB system does not do this.

Note:

Both foreground and background isochronous FIFOs are cleared when the CTRL.CLR_EP bit is set.

10.3.6 Isochronous IN (MPU->USB HOST) Transactions

Isochronous IN transactions are USB transactions in which a given amount of data is transferred from the USB device controller module device to the USB host every 1-ms USB frame. No handshaking is provided.

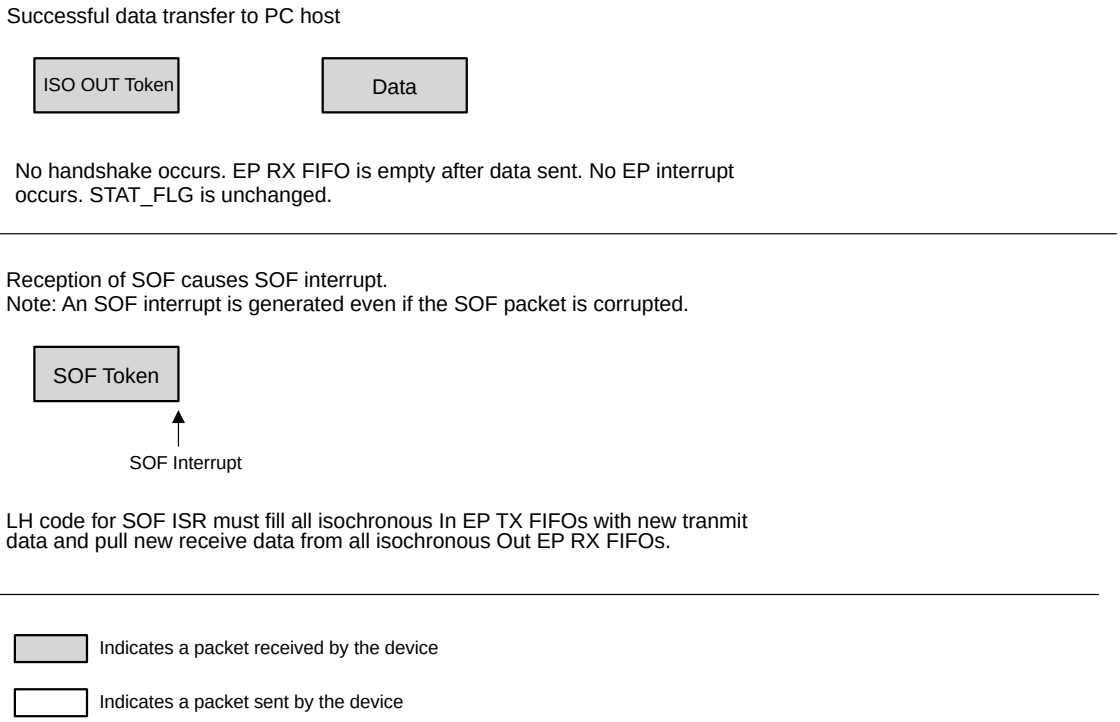
The USB module provides double-buffering of data for ISO IN endpoints; the background FIFO is used as the source of data for IN transactions to the ISO endpoint, and the foreground FIFO can be written to by the MPU. When an IN transaction to an ISO endpoint occurs, the USB module sends all data found in the endpoint background TX FIFO. The MPU is responsible for providing new data to the isochronous IN endpoint foreground TX FIFO at each start of frame interrupt.

In response to the SOF interrupt, for each isochronous IN endpoint, MPU code selects the endpoint (via the EP_NUM register), and then fills the endpoint TX FIFO (via the DATA register). Once all the transmit data have been written to the FIFO, the MPU code must clear the EP_NUM.EP_SEL bit.

Because the USB transaction for the isochronous endpoint can occur at any time during the USB 1-ms frame, the USB interface implements a double-buffering of the endpoint transmit data FIFO. The endpoint includes two FIFOs, each of which is the length of the configured isochronous endpoint. At all times, one of the two FIFOs is foreground and the other is background. The USB interface side of the USB module is allowed to read from the background TX FIFO, and the MPU is allowed to write to the foreground TX FIFO. The designations foreground and background are swapped, and the new background TX FIFO is cleared at each start of frame (SOF). Because ISO endpoints implement double-buffering, ISO endpoints do not control access to the FIFOs via a CTRL.SET_FIFO_EN bit; the CTRL.SET_FIFO_EN and the STAT_FLG.FIFO_EN bits are not implemented for ISO IN endpoints.

Figure 10–6 shows the transaction phases associated with isochronous IN transactions and the SOF transaction. No endpoint-specific interrupt to the MPU is generated as a result of an isochronous IN transaction, and there is no handshake phase. The SOF transaction causes an SOF interrupt to the MPU; it is assumed that the MPU refills the isochronous IN endpoint transmit FIFO at each SOF interrupt.

Figure 10–6. Isochronous IN Transaction Phases and Interrupts



10.3.6.1 Isochronous IN Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STAT_FLG.STALL, STAT_FLG.NAK, and STAT_FLG.ACK bits for isochronous endpoints always return 0. Because there is no handshake, there is no endpoint-specific interrupt to the MPU to report handshake results for isochronous endpoints.

10.3.6.2 Isochronous IN Transaction Error Conditions

If the USB host did not successfully complete an ISO IN transaction in the previous frame, and if data were present in TX FIFO to be sent at the IN transaction, the STAT_FLG.MISS_IN bit is asserted for the duration of the following frame. If the ISO IN endpoint is cleared in the middle of a USB transaction to the background FIFO, the macro forces a bit stuffing error for the ISO transaction.

10.3.6.3 Isochronous IN Endpoint FIFO Error Conditions

If the MPU attempts to overfill the configured endpoint FIFO, data written to DATA register after the TX FIFO is full is lost, but any data that was successfully put into the FIFO is transmitted when that FIFO is the background FIFO and an IN transaction for that endpoint occurs. Because an ISO TX FIFO is cleared automatically on the toggle from background to foreground, there is no reason to clear the FIFO. However, if the MPU does not wish to send the data it wrote, clearing the endpoint is the only mechanism to do this.

10.3.7 Control Transfers on Endpoint 0

Control transfers on endpoint 0 include control write and control read transfers. Control write and control read transfers are each composed of two or more transactions to endpoint 0. Additionally, the USB device controller module is capable of autodecoding some control write and control read transfers. These operations are summarized in Figure 10–7 and Figure 10–8. An IN or an OUT transaction is received out of a control request. This transaction is automatically stalled by the core.

Figure 10–7. Stages and Transaction Phases of Autodecoded Control Transfers

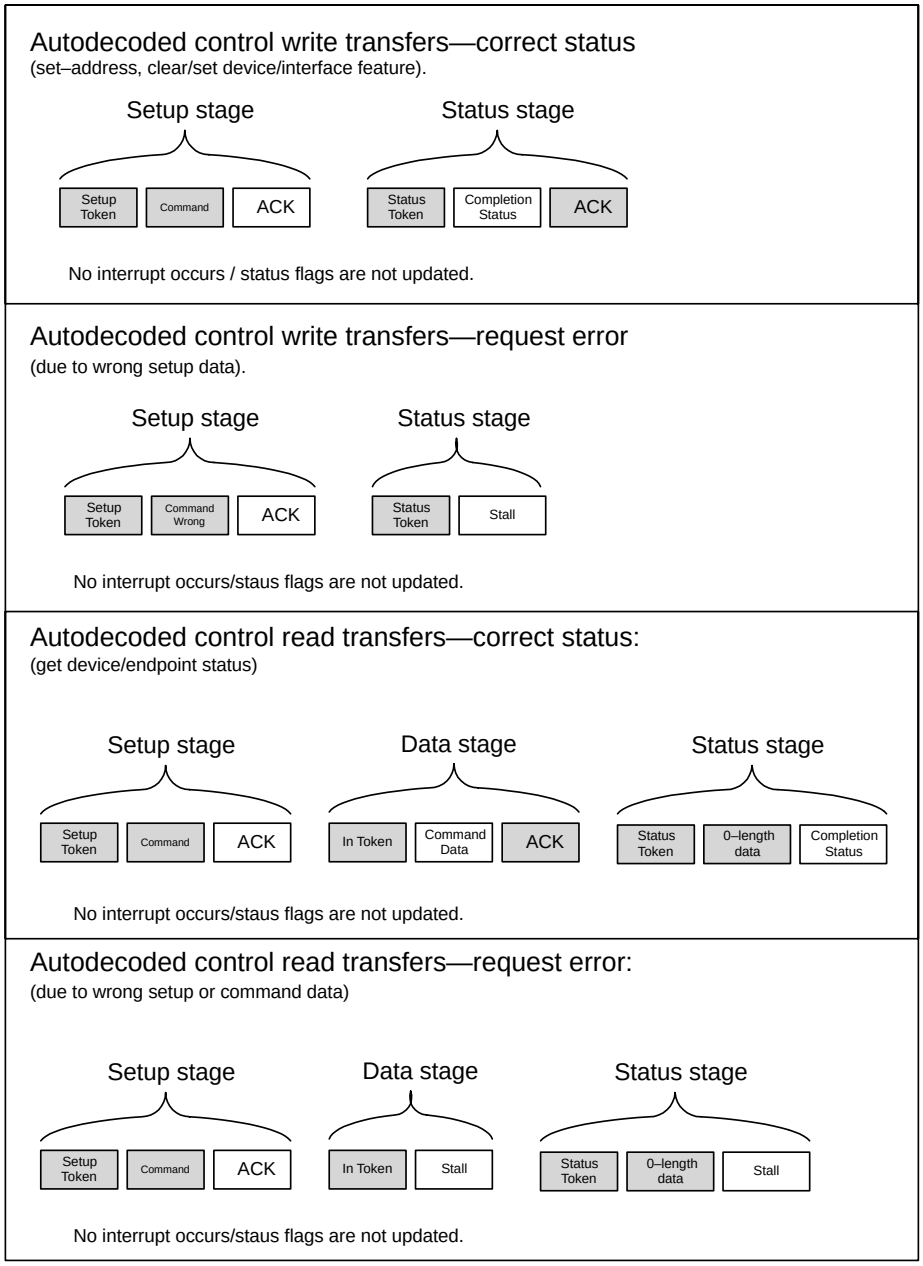
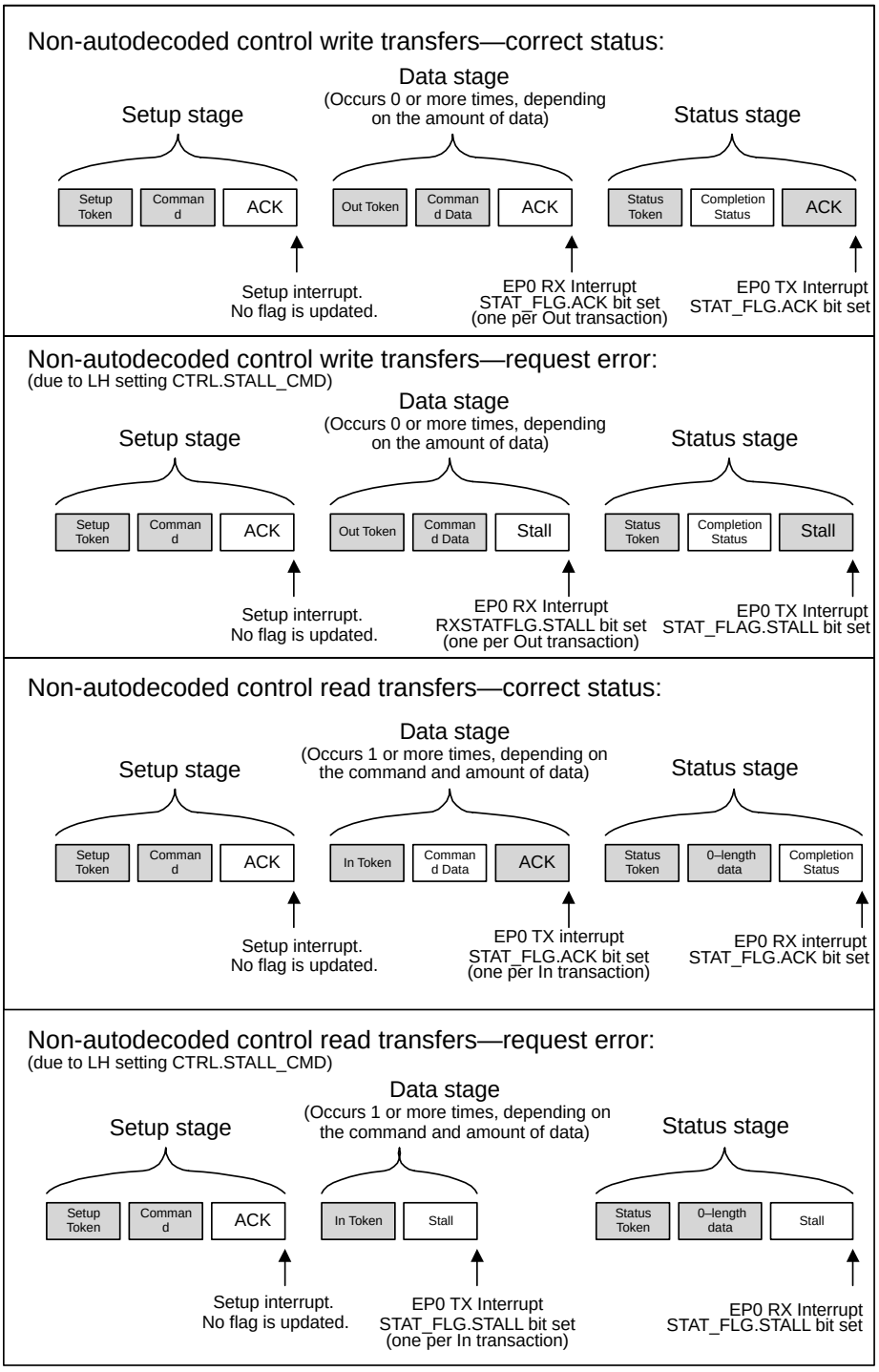


Figure 10–8. Stages and Transaction Phases of Non-Autodecoded Control Transfers



Non-autodecoded control read and control write transfers are sets of transactions that occur on endpoint 0 and have specific USB protocol meaning but are not handled automatically by the core. The USB device controller block automatically provides an ACK handshake for the setup stage transaction, but the data and status stage transaction handshaking is accomplished using the normal RX and TX control bits that affect transaction handshaking. A general USB

interrupt to the MPU occurs at the end of each transaction of each stage of a control transfer. The MPU must perform the following actions to act on non-autodecoded control transfers:

- Process the setup phase setup USB interrupt. The MPU reads the control transfer command from the setup FIFO and decodes the command. For control reads, the MPU fetches the requested read data and places it (or as much of the read data as fits) into the endpoint 0 FIFO, and then enables the endpoint 0 FIFO. For control writes, the MPU code only enables the endpoint 0 FIFO. MPU code also sets any flags needed for processing endpoint 0 USB interrupts during the control transfer.
- Process the data phase endpoint 0 general USB interrupt(s). For control reads, the data phase general USB endpoint 0 TX interrupt indicates that the previously provided transmit data has been sent. Any additional data must be written to the endpoint 0 FIFO. For control writes, the write data must be pulled from the endpoint 0 FIFO, and when all control write data is available, interpret the write data and act on the write request. After handling the last data phase interrupt, the MPU must set the endpoint 0 control bits to signal the desired status to the host.
- Process the status stage endpoint 0 general USB interrupt. The MPU provides its completion status back to the USB host during this stage, either via status in the data phase of the transaction (for control write transfers) or via the handshake phase of the transaction (for control read transfers).

Autodecoded control read and control write transfers are sets of transactions that occur on endpoint 0 that have specific USB protocol meaning and are handled automatically by the USB device controller block without any intervention by the MPU. The USB device controller block handles all handshaking automatically and without regard to the endpoint 0 control bits that affect normal (non-control transfer) transaction handshaking. No interrupt is asserted to the MPU during autodecoded control transfers.

If no USB1.1 specification defined request is associated with the data of the setup phase, request is stalled by the core and the MPU is not informed of its occurrence (autodecoded).

When a setup token is identified, the USB decode module must monitor the setup stage data packet, decode it, and determine whether it is an autodecoded or a non-autodecoded transfer and a control read or a control write. If it is a valid non-autodecoded request, the setup FIFO is immediately cleared and control of the FIFO is immediately taken away from the MPU (if the MPU had control of the FIFO). New setup data are placed into the setup FIFO, and the setup interrupt flag is set (IRQ_SRC.setup).

In response to the setup interrupt, the MPU must select the setup FIFO by setting the EP_NUM.SETUP_SEL bit. This clears the IRQ_SRC.SETUP flag. The MPU must then read 8 bytes from the setup FIFO, clear the EP_NUM.SETUP_SEL bit, and confirm that IRQ_SRC.SETUP bit has not been reset by a new setup transaction. If the IRQ_SRC.SETUP flag is asserted, the MPU must discard the previously read data and handle the new

setup packet as explained above. Thus the MPU never misses a new occurring setup transaction (per USB 1.1 specification).

10.3.7.1 Autodecoded Control Write Transfers

For set address control write transfers, the USB address provided in the setup token is captured to the USB module device address register. If new address is different from 0, the device moves into addressed state (DEVSTAT.ADD set) if it was not already addressed.

For set and clear feature control writes, the appropriate feature information bit is set or cleared. When a set or clear feature transfer occurs to set or clear the device remote wake-up feature, the DEVSTAT.R_WK_OK bit is set or cleared, as appropriate. If a set or clear interface feature occurs, the request is automatically stalled by the core because no feature is defined for interface (see the USB 1.1 specification).

Per the USB 1.1 specifications, a SET_ADDRESS request is effective after the status stage of the request, even if the status stage does not end with an ACK handshake. SET/CLEAR_FEATURE requests are effective after setup stage, even if no status stage occurs.

Autodecoded Control Write Transfer Handshaking

The USB device controller module automatically provides ACK handshaking for all transactions of all stages of autodecoded control write transfers, except if a corrupted packet is received, which the USB module ignores. The CTRL.SET_FIFO_EN and SYSCON2.STALL_CMD bits have no effect on handshaking.

Autodecoded Control Write Transfer Error Conditions

If the token packet or the data packet of a setup stage transaction has an error (bad CRC, PID check, or bit stuffing error), the USB block ignores the transaction. The USB block does not provide ACK handshaking in this case.

10.3.7.2 Autodecoded Control Read Transfers

Autodecoded control reads include the standard device requests GET_ENDPOINT and DEVICE_STATUS. These control read transfers access information that is kept in registers inside the USB module, so MPU code is not involved in filling the read data into the TX FIFO.

The USB module returns the currently selected appropriate status information (depending on the wIndex value in the setup stage data packet) during the data phase of the single IN transaction of the data stage, and provides ACK as the handshake for the status stage handshake phase. The MPU receives no interrupt.

Autodecoded Control Read Transfer Handshaking

The USB device controller module automatically provides ACK handshaking for all transactions of all stages of autodecoded control read transfers, except

if a corrupted token packet is received, which the USB module ignores. The CTRL.FIFO_EN and SYSCON2.STALL_CMD bits have no effect on the handshaking. If the status packet has a DATA0 PID instead of a DATA1 PID, status is STALLed and no interrupt is asserted to the MPU. If the setup packet has a DATA1 PID instead of a DATA0 PID, setup transaction is ignored (error).

Autodecoded Control Read Transfer Error Conditions

If the token phase or the data phase of a setup stage transaction has an error (bad CRC, PID check, or bit stuffing error), the USB block ignores the transaction. The USB block does not provide ACK handshaking in this case.

Data errors during the data stage of autodecoded control write transfers are handled in the standard way; any data stage transaction from the host where a data error occurs is ignored.

It is possible that the USB host sends a GET_ENDPOINT/DEVICE_STATUS request with a bad parameter. If the autodecode mechanism senses a bad parameter in the setup stage data phase, the autodecode mechanism causes a STALL handshake to be signaled during the data phase of the data stage and during the status stage.

10.3.7.3 Non-Autodecoded Control Write Transfers

Non-autodecoded control write transfers include the SET_/CLEAR_ENDPOINT feature, SET_CONFIGURATION, SET_INTERFACE, SET_DESCRIPTOR and class- or vendor-specific control write transfers. Non-autodecoded control write transfers consist of two or three stages [setup, data (optional), and status].

The setup stage of a valid non-autodecoded control write transfer consists of one SETUP transaction from USB host to USB device. At the end of the setup stage handshake, the USB module generates an MPU general USB interrupt with the IRQ_SRC.SETUP flag set. The MPU must respond to this general USB interrupt by setting EP_NUM.SETUP_SEL bit, which clears the setup interrupt flag. The MPU must then read 8 bytes from the setup FIFO via the DATA register, clear EP_NUM.EP_SEL bit, and check the IRQ_SRC.SETUP flag. If the IRQ_SRC.SETUP flag is set, the MPU must discard the setup data it has just read and handle the new setup data packet following the same scheme. If the IRQ_SRC.SETUP flag is cleared, the MPU code interprets this request information and performs any application-specific activity needed because of the setup stage request. If there is one or more data stage for the transfer, the MPU must set the CTRL.SET_FIFO_EN bit for endpoint 0 to allow the core to accept RX data from the coming OUT transaction.

The data stage for non-autodecoded control writes consists of zero or more OUT transactions. Transaction handshaking and interrupt generation as for non-isochronous, non-control OUT endpoints applies. The MPU can cause NAK, STALL, or ACK signaling for the data stage transactions. If ACK was signaled on a given general USB interrupt, the MPU must respond by reading the data from the endpoint 0 RX FIFO and saving it for processing.

After completion of the data stage, a status stage IN transaction occurs. The USB module provides handshaking to the USB host based on the endpoint 0

handshaking control bit STAT_FLG.FIFO_EN. The MPU can delay signaling completion of the control write transfer by forcing NAK handshaking to the host during the status stage (by holding STAT_FLG.FIFO_EN 0), or causing ACK handshaking by setting the CTRL.SET_FIFO_EN bit with an empty endpoint 0 FIFO. An endpoint 0 TX general USB interrupt is sent to the MPU at completion of the status stage.

After a SET_CONFIGURATION request, the device moves in addressed or configured state as soon as the MPU sets the SYSCON2.DEV_CFG or SYSCON2.CLR_CFG bits.

Specific MPU Required Actions

If the device receives a valid set endpoint halt feature request, it must set the appropriate CTRL.SET_HALT control bit.

If the device receives a valid CLEAR_ENDPOINT halt feature request, it must set the appropriate CTRL.RESET_EP bit to clear the halt condition, FIFO flags, and reset data PID to DATA0 for the endpoint. If the specified endpoint number is 0, the MPU has only to set CTRL.CLR_HALT bit to clear the halt condition.

If the device receives a valid SET_CONFIGURATION request, it must reset all endpoints by setting the CTRL.RESET_EP control bits, set the SYSCON1.SELF_PWR bit to the appropriate value, and then set halt conditions for endpoints not used by the default interface set for the configuration. If the device was addressed when the SET_CONFIGURATION was received, the MPU must write 1 to the SYSCON2.DEV_CFG bit to allow the device to move into the configured state (DEVSTAT.CFG bit set). If the device was configured when the SET_CONFIGURATION was received, and the new configuration value is 0, the MPU must write 1 to the SYSCON2.CLR_CFG bit to allow the device to move back into the addressed state (DEVSTAT.CFG bit cleared).

If the device receives a valid set interface request, it must reset all endpoints used by the interface set, by setting CTRL.RESET_EP control bits, and then set halt conditions for endpoints not used by this interface.

Other MPU required actions are specific to the request and not detailed in this document.

Non-Autodecoded Control Write Transfer Handshaking

Setup stage transactions that are valid are signaled ACK. Transactions with invalid setup stage token or data packets are ignored and receive no handshake packet from the USB module, and there is no interrupt generated.

Data stage handshaking for non-autodecoded control write transfers is dependent on the endpoint 0 STAT_FLG.FIFO_EN, STAT_FLG.EP_HALTED, and SYSCON2.STALL_CMD bits. The MPU can delay completion of any transaction of the data stage by signaling NAK (via CTRL.SET_FIFO_EN bit not set). The USB specification requires that once STALL is signaled in a control transfer, it must be signaled on that endpoint until the next setup token is received.

Either the SYSCON2.STALL_CMD or the CTRL.SET_HALT (reflected in the STAT_FLG.EP_HALTED register bit) register bits provide this functionality. STAT_FLG.EP_HALTED does not reflect the forced STALL caused by SYSCON2.STALL_CMD; it retains its previous value.

Status stage handshaking is controlled by the endpoint 0 STAT_FLG.FIFO_EN and SYSCON2.STALL_CMD bits. Successful completion of a non-autodecoded control write transfer is indicated by the USB device controller module returning a zero length data payload for the data phase of the status stage and an ACK handshake from the host for the handshake phase of the status stage. Although NAK handshaking can be used to indicate delays in completion of the requested control write, the USB host can choose to abort the control write after some number of NAKs.

Non-Autodecoded Control Write Transfer Error Conditions

If an error occurs while dealing with the control write, which the MPU cannot deal with itself, it must signal STALL to the USB host for all subsequent transactions until a new setup token to endpoint 0 occurs. This is true for both data stage and status stage transactions. This is most conveniently done by setting endpoint 0 SYSCON2.STALL_CMD bit, which causes stalling of all the remaining transactions of all remaining stages of a non-autodecoded control transfer, up to the reception of the next valid SETUP command.

Error conditions are handled as for BULK/INTERRUPT transactions. If a packet is received corrupted, the core ignores the transaction and no interrupt is asserted.

10.3.7.4 Non-Autodecoded Control Read Transfers

Non-autodecoded control read transfers include the GET_INTERFACE_STATUS, GET_CONFIGURATION, GET_INTERFACE, GET_DESCRIPTOR, SYNCH_FRAME and class- or vendor-specific control read transfers. Non-autodecoded control read transfers consist of three stages (setup, data, and status).

The setup stage of a valid non-autodecoded control read transfer consists of one SETUP transaction from USB host to USB device. At the end of the setup stage handshake, the USB module generates a MPU general USB interrupt with the IRQ_SRC.SETUP flag set. The MPU must respond to this general USB interrupt by setting the EP_NUM.SETUP_SEL bit, which clears the setup interrupt flag. The MPU must then read 8 bytes from the setup FIFO via the DATA register, clear the EP_NUM.EP_SEL bit, and check the IRQ_SRC.SETUP flag. If the IRQ_SRC.SETUP flag is set, the MPU must discard the setup data it has just read and handle the new setup data packet following the same scheme. If the IRQ_SRC.SETUP flag is cleared, the MPU code interprets this request information and then prepares data for the IN transaction that follow. This includes placing the data being requested (or the first few bytes, if more than one FIFO worth of data is being returned) into the endpoint 0 FIFO, and setting the CTRL.SET_FIFO_EN bit.

The data stage of a control read transfer consists of one or more IN transactions. Transaction handshaking and interrupt generation as for non-isochrono-

us, non-control IN endpoints applies; the MPU can cause NAK, STALL, or ACK signaling for the data stage transactions. At endpoint 0 TX general USB interrupts, MPU code must move more data to the endpoint 0 FIFO, until the last bytes of the requested data have been provided. Although SETUP packets have a defined payload length, the USB host can cancel the transaction at any time, without the status stage, and resend another SETUP command. The MPU code must be able to operate correctly in this situation.

After completion of the data stage, a status stage OUT transaction occurs. The USB host sends a zero-length data packet, and the MPU code must return its completion status for the control read standard request via standard handshaking mechanisms.

Note:

In the case of returning exactly what the host requested and that request was a multiple of the maximum packet size, no zero-length packet is required. A zero-length packet is required only when the amount of data the device has to return is less than the amount requested by the host and the amount returned is a multiple of the maximum packet size (source USB forum).

Non-Autodecoded Control Read Transfer Handshaking

Handshaking for the setup stage of non-autodecoded control read transfers is forced by the USB module to always be ACK, unless there is a data error in the packet, in which case the USB module ignores the transaction. If the setup packet has a DATA1 PID instead of a DATA0 PID, the setup transaction is ignored (error).

Data stage handshaking for non-autodecoded control read transfers is dependent on the endpoint 0 STAT_FLG.FIFO_EN, STAT_FLG.EP_HALTED, and SYSCON2.STALL_CMD bits. The handshaking information is used during the data phase of the data stage transaction. The USB specification requires that once STALL is signaled in a control transfer, it must be signaled until the next setup token is received. The SYSCON2.STALL_CMD and CTRL.SET_HALT (reflected through the STAT_FLG.EP_HALTED register bit) register bits provide this functionality. STAT_FLG.EP_HALTED does not reflect the forced STALL caused by SYSCON2.STALL_CMD; it retains its previous value.

The status stage is controlled by the CTRL.FIFO_EN and SYSCON2.STALL_CMD bits.

Successful completion of non-autodecoded control read transfers is indicated by the host sending an OUT token followed by an empty packet and the USB device controller responding with ACK. If the data packet sent by the USB host during the status stage of a control read request is not empty, the OUT transaction is accepted by the core, but OUT data is not put into the endpoint 0 RX FIFO. If the status packet has a DATA0 PID instead of a DATA1 PID, a STALLed is returned by the core and an interrupt is asserted.

Non-Autodecoded Control Read Transfer Error Conditions

If an error occurs while dealing with the control read, which the MPU cannot deal with itself, it must signal STALL to the USB host for all subsequent trans-

actions until a new setup token to endpoint 0 occurs. This is true for both data stage and status stage transactions. This is most conveniently done by setting endpoint 0 SYSCON2.STALL_CMD bit, which causes stalling of all the remaining transactions of all remaining stages of a non-autodecoded control transfer, up to the reception of the next valid SETUP command.

Error conditions are handled as for BULK/INTERRUPT transactions. The USB device controller module responds to control read status stage transactions that have a bad token or bad data by not sending a handshake packet. In both cases, the transaction is ignored and no general USB interrupt is generated to the MPU.

10.3.7.5 Autodecoded Versus Non-Autodecoded Control Requests

Table 10–53. Autodecoded Versus Non-Autodecoded Control Requests

Request	Recipient	Status	MPU Required Action	Device Behavior if Device Is Not Configured
GET_ STATUS	Device	Autodecoded (function of AUTODEC_ DIS register bit)	None	Device status is returned (SYSCON1.SELF_PWR and DEVSTAT.R_WK_OK bits).
	Interface	Non-autodecoded	The MPU must stall the command (via SYSCON2.STALL_ CMD bit) if interface number is not correct. No feature is defined for interface.	Command is passed to the MPU.
	Endpoint	Autodecoded (function of AUTODEC_ DIS register bit)	None	The core automatically stalls the command if endpoint number is different from 0.
CLEAR/SET FEATURE	Device	Autodecoded (function of AUTODEC_ DIS register bit)	None (DS_CHG IT is asserted to the MPU after any DEVSTAT.R_ WK_OK bit modification).	The core handles the request.

Notes: 1) Transactions on endpoints other than zero are ignored if the device is not configured (addressed state).

2) If some endpoints are not used by the interface currently set, transactions on these endpoints are not ignored; the MPU must set halt feature for the endpoint. This does not happen if USB host works correctly.

3) If endpoint 0 is halted, per *USB 1.1 Specification* (see 9.4.5: Get_Status), all requests are stalled except GET_STATUS, CLEAR_FEATURE, and SET_FEATURE requests.

4) Requests are handled with respect to USB 1.1 when specified as such, but many device reactions are not specified by USB 1.1.

5) During a SET_ADDRESS autodecoded command, only the 7 LSBs are significant for the new address (decimal value from 0 to 127); all others are discarded.

Table 10–53. Autodecoded Versus Non-Autodecoded Control Requests (Continued)

Request	Recipient	Status	MPU Required Action	Device Behavior if Device Is Not Configured
	Interface	Autodecoded (function of AUTODEC_ DIS register bit)	None. (No feature is defined in USB 1.1 spec for interface. These requests are stalled).	Command is stalled anyway.
	Endpoint	Non-autodecoded	The MPU must stall the command (via SYSCON2.STALL_ CMD bit) if endpoint number/ type/direction is not correct. The MPU must reset the EP after having handled the pending transactions (if CLEAR) or set halt condition (if SET). For EP 0, MPU must only clear or set halt condition; FIFO and data PID is always correct for next setup.	Command is passed to the MPU.
SET_ ADDRESS	Device	Autodecoded	None (see Note 5) (Whether the device is addressed or not is available in DEVSTAT register. A valid SET_ADDRESS request with address number from 0 generates a DS_CHG interrupt to the MPU).	Default: device moves in the addressed state if address number is different from 0. Addressed: device takes the new address value or moves in default state if address number is 0. Configured: request is STALLed.
GET_ DESCRIPTOR	All	Non-autodecoded	The MPU must write descriptor data into endpoint 0 FIFO.	Command is passed to the MPU.
SET_ DESCRIPTOR	All	Non-autodecoded	The MPU must stall the command (via SYSCON2.STALL_ CMD bit) if it does not support set descriptor requests.	Command is passed to the MPU.

Notes:
1) Transactions on endpoints other than zero are ignored if the device is not configured (addressed state).
2) If some endpoints are not used by the interface currently set, transactions on these endpoints are not ignored; the MPU must set halt feature for the endpoint. This does not happen if USB host works correctly.
3) If endpoint 0 is halted, per *USB 1.1 Specification* (see 9.4.5: Get_Status), all requests are stalled except GET_STATUS, CLEAR_FEATURE, and SET_FEATURE requests.
4) Requests are handled with respect to USB 1.1 when specified as such, but many device reactions are not specified by USB 1.1.
5) During a SET_ADDRESS autodecoded command, only the 7 LSBs are significant for the new address (decimal value from 0 to 127); all others are discarded.

Table 10–53. Autodecoded Versus Non-Autodecoded Control Requests (Continued)

Request	Recipient	Status	MPU Required Action	Device Behavior if Device Is Not Configured
GET/SET CONFIGURATION	Device	Non-autodecoded	The MPU must stall the command (via SYSCON2.STALL_CMD bit) if configuration number is not correct. If the request is SET_CONFIG, the MPU must reset all endpoints, halt endpoints not used by the default interface setting, set SYSCON1.SELF_PWR value if device is self-powered for the configuration set, and then set SYSCON2.DEV_CFG bit (if config nb is not 0), or set SYSCON2.CLR_CFG bit (if config nb is 0) before allowing status stage to complete. The device moves to configured state (if DEV_CFG set), or moves to addressed state (if CLR_CFG set) and a DS_CHG interrupt is asserted to the MPU.	Command is passed to the MPU.
GET/SET INTERFACE	Interface	Non-autodecoded	The MPU must stall the command (via SYSCON2.STALL_CMD bit) if interface/setting number is not correct. If the request is SET_INTER-FACE, the MPU must reset endpoints used by the interface and halt endpoints not used by the interface setting before allowing status stage to complete.	Command is passed to the MPU.
SYNCH_FRAME	Endpoint	Non-autodecoded	The MPU must stall the command if it does not support SYNCH_FRAME request, else write requested data in the endpoint 0 FIFO.	Command is passed to the MPU.

Notes:
1) Transactions on endpoints other than zero are ignored if the device is not configured (addressed state).
2) If some endpoints are not used by the interface currently set, transactions on these endpoints are not ignored; the MPU must set halt feature for the endpoint. This does not happen if USB host works correctly.
3) If endpoint 0 is halted, per *USB 1.1 Specification* (see 9.4.5: Get_Status), all requests are stalled except GET_STATUS, CLEAR_FEATURE, and SET_FEATURE requests.
4) Requests are handled with respect to USB 1.1 when specified as such, but many device reactions are not specified by USB 1.1.
5) During a SET_ADDRESS autodecoded command, only the 7 LSBs are significant for the new address (decimal value from 0 to 127); all others are discarded.

10.3.7.6 Note on Control Transfers Data Stage Length

The control transfer data stage length is indicated in the setup data packet.

During control reads, if the USB host requests more data than indicated in the setup packet, unexpected IN transaction is STALLed, causing STALL handshake for all remaining transactions of the transfer until next SETUP. If the USB host requires less data than indicated in the setup packet, the transfer is not STALLed. However, if the host moves to status stage earlier than expected for a non-autodecoded request, the OUT status stage is NAKed because the MPU has not enabled the RX FIFO.

During control writes, if the USB host sends more bytes than indicated in the setup packet, the transfer is STALLed. If the USB host sends fewer bytes than were expected, the request is accepted. But if the USB host moves to status stage earlier than expected for a non-autodecoded request, the IN status stage is NAKed because the MPU has not enabled the TX FIFO.

10.3.8 USB Device Initialization

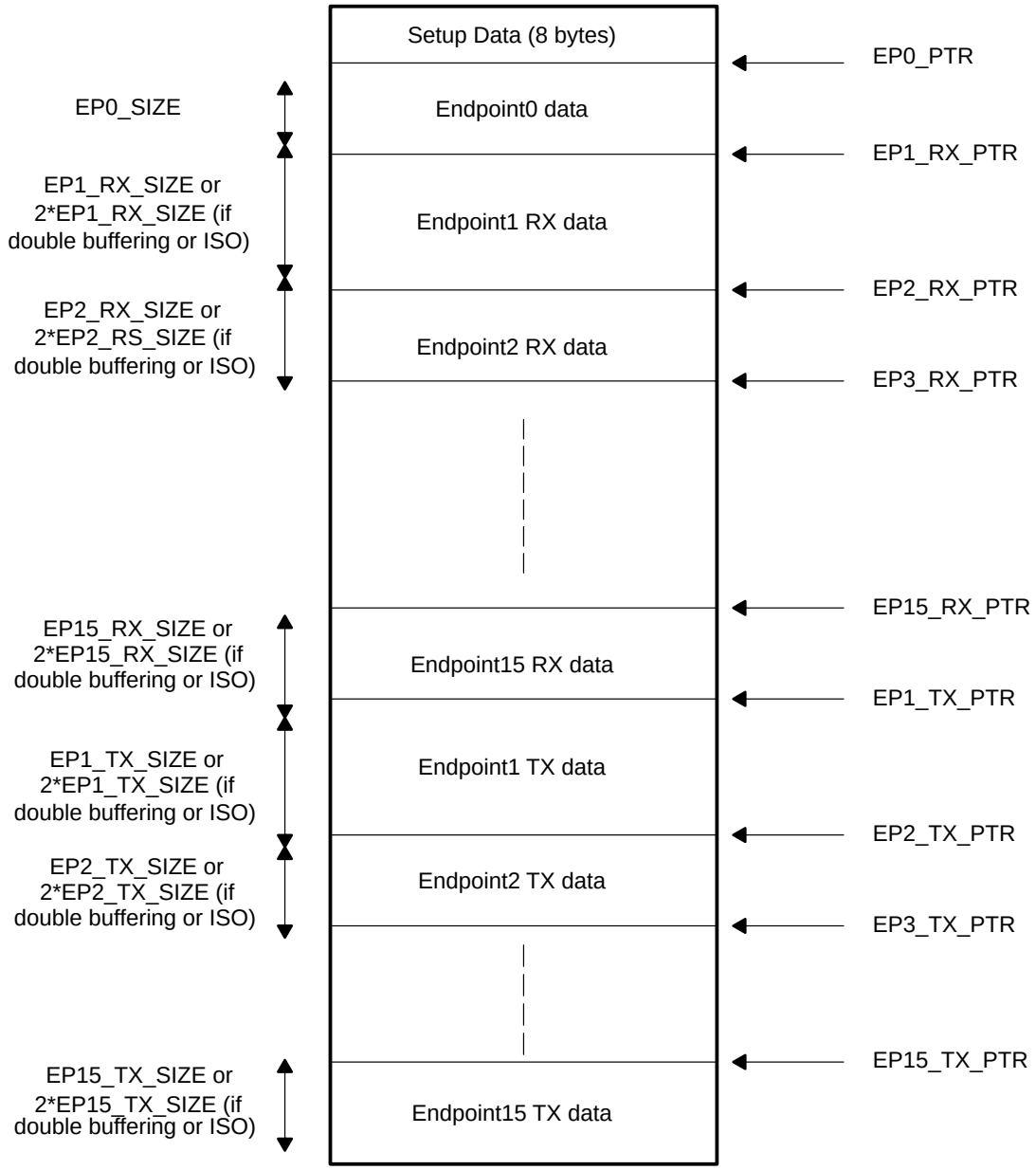
To allow communication between the device and a USB host, the MPU must configure the device by filling the configuration registers.

For each endpoint, the MPU must write on the dedicated register:

- ❏ Endpoint size
- ❏ Whether double-buffering is allowed for endpoint or not
- ❏ Endpoint type (ISO or non-ISO)
- ❏ Address of the pointer

System software must choose how to allocate the 2040 available bytes of USB device controller RAM to the USB endpoints. Receive endpoint size and type are configured using the EP1_RX through EP15_RX registers. Transmit endpoint size and type are configured using the EP1_TX through EP15_TX registers. Figure 10–9 shows an example of the RAM organization, obtained by following the flowchart shown in Figure 10–10.

Figure 10–9. Example of RAM Organization



Once the endpoints are configured, the MPU must set the SYSCON1.CFG_LOCK bit. If this bit is not set, all transactions are ignored by the core. Then, when the MPU is ready to communicate with the USB host, it must set the SYSCON1.PULLUP_EN bit. The MPU can wait until the DS_CHG attach interrupt has been detected and handled before setting the SYSCON1.PULLUP_EN bit. The USB host cannot detect the device until this bit is set.

Figure 10–10 and Figure 10–11 show flowcharts for the configuration phase.

Figure 10–10. Device Configuration Routine

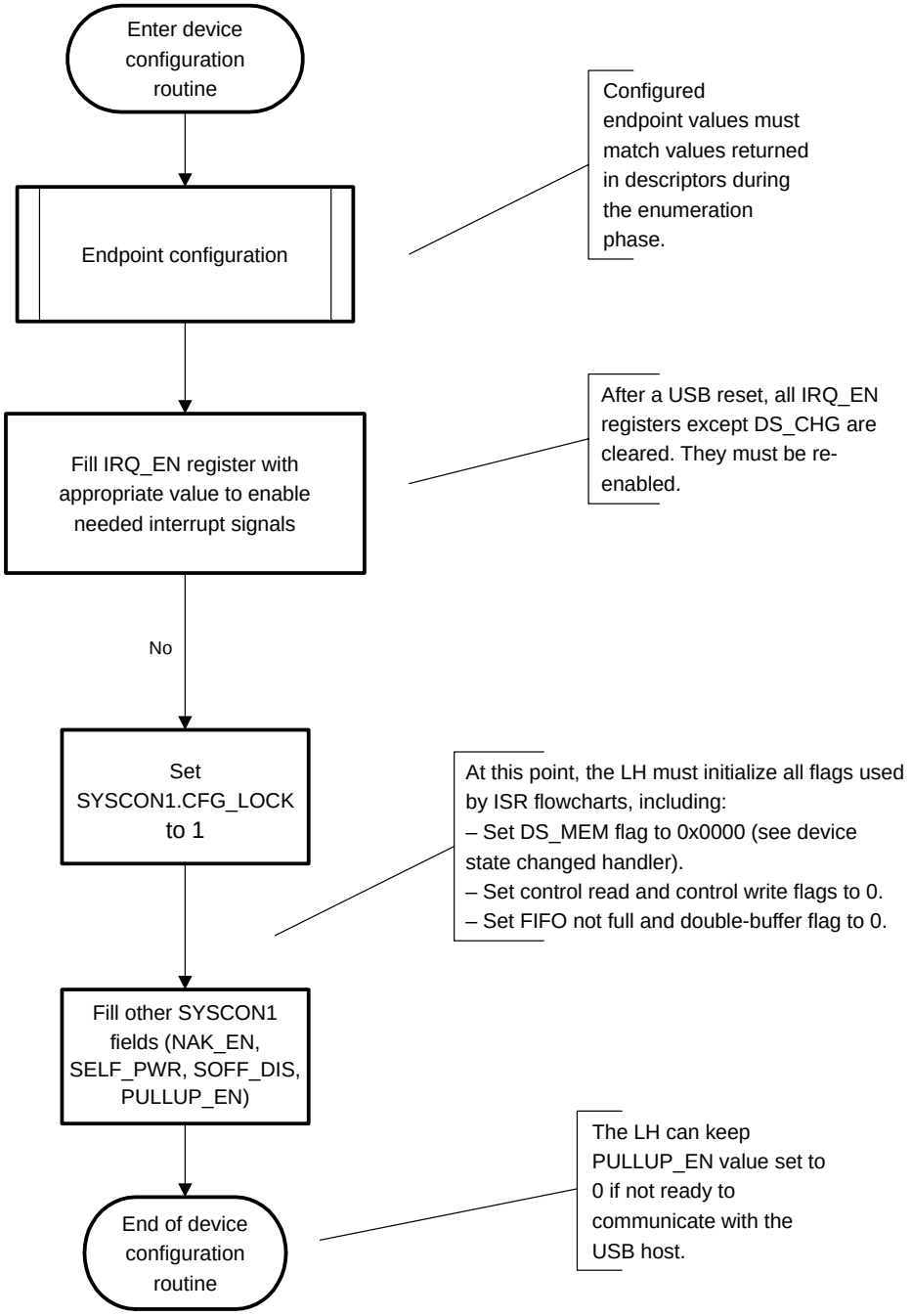
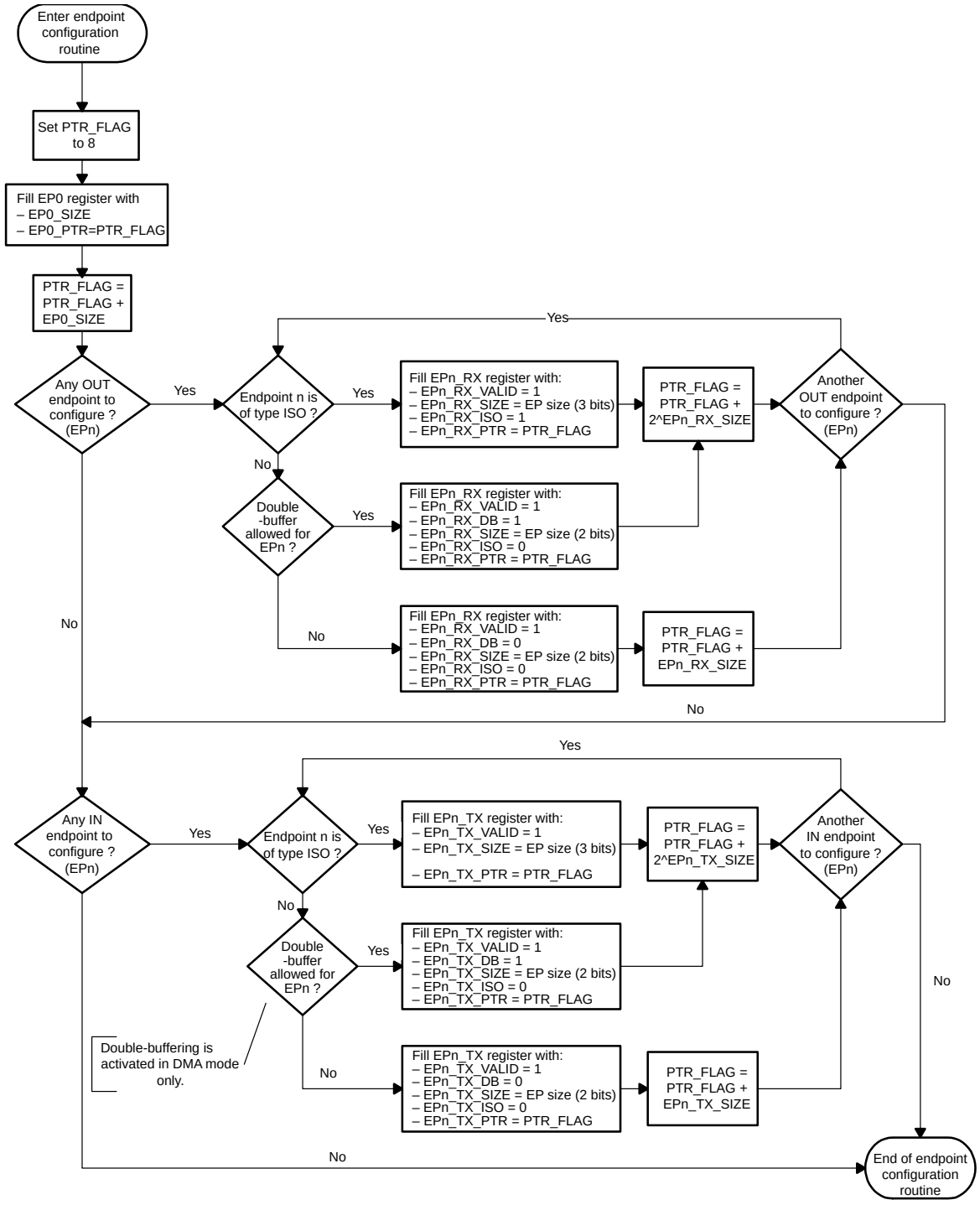


Figure 10–11. Endpoint Configuration Routine



10.3.9 Preparing for Transfers

To avoid NAK handshakes for the first transaction on an endpoint, the MPU must prepare the endpoint FIFO for receiving or transferring data. After the first transaction, the FIFO is enabled during the interrupt handling (ISR flowchart).

For receive endpoints, this phase consists of enabling the FIFO to receive data from the USB host. If double buffering is allowed for the endpoint, setting the CTRL.SET_FIFO_EN bit enables both FIFOs. Therefore, it is not possible to allow a single transaction when double buffering is used.

The MPU enters the prepare for USB RX transfers routine, presented once after the enumeration phase, and then properly reacts to EP interrupts. Whether double-buffering is allowed or not is transparent to the MPU, unless both FIFOs are cleared through a CTRL.CLR_EP or CTRL.RESET_EP. In that case, and in the case where the MPU finishes to handle an interrupt without having set the CTRL.SET_FIFO_EN bit, the MPU must reenter the prepare for USB RX transfers routine.

For transmit endpoints, the MPU enters the prepare for endpoint n TX transfer routine, presented each time a new file must be transmitted from endpoint n to USB host. The MPU must not enter this routine until data written into TX FIFO from the previous transfer have all been received successfully by the USB host (ACK interrupt received), unless TX FIFO is cleared through the CTRL.CLR_EP or CTRL.RESET_EP bits (see Figure 10–12 and Figure 10–13).

Note:

This does not apply to endpoint 0, which is not used before a setup interrupt occurs. At setup interrupt, the MPU reacts appropriately, and enables EP0 FIFO only if necessary.

To ensure proper usage of the module, you cannot prepare data on different endpoints at the same time. You can enter a routine once the routine is not being used by another endpoint (no parallelism); the EP_NUM register can be accessed via different routines.

Figure 10–12. Prepare for USB RX Transfers Routine

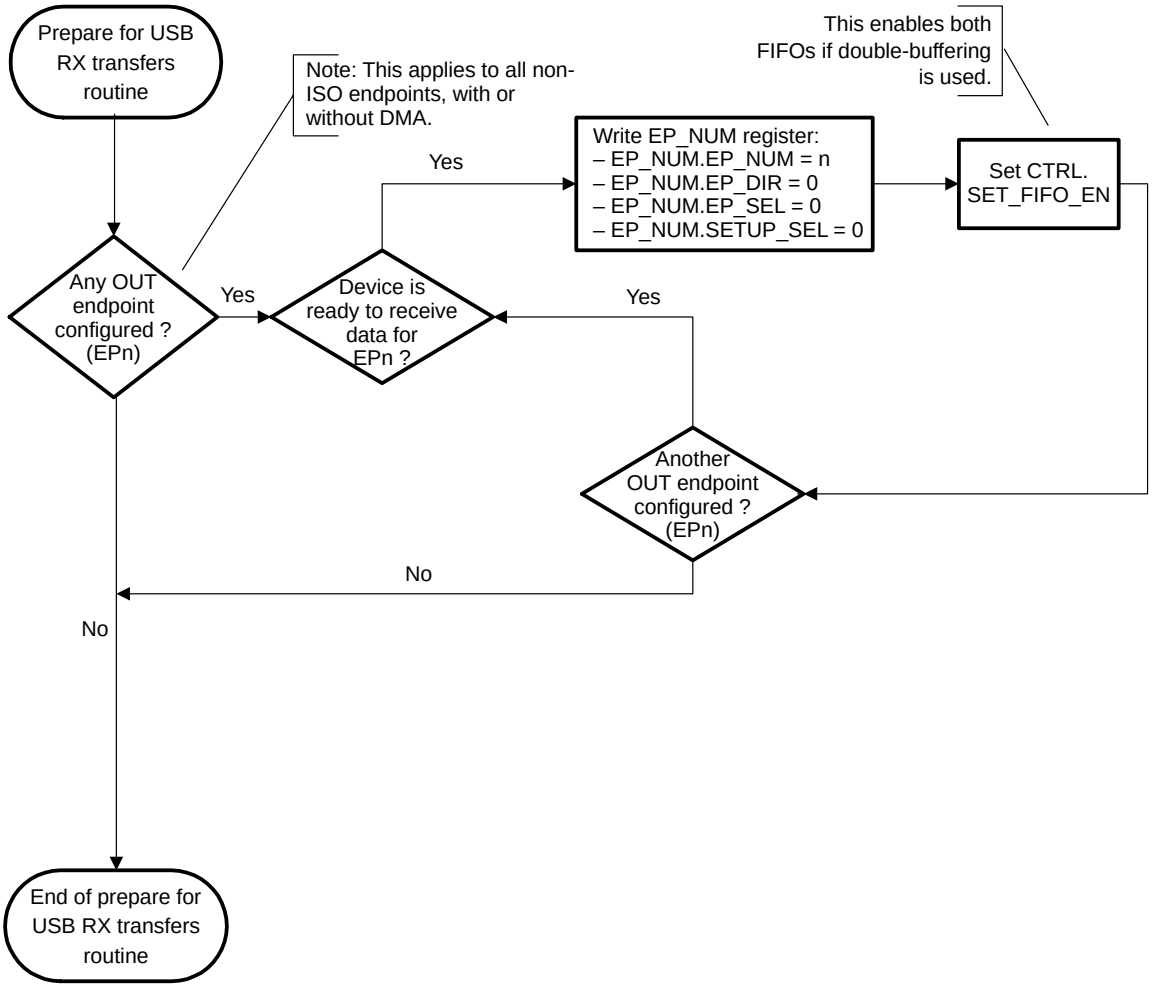
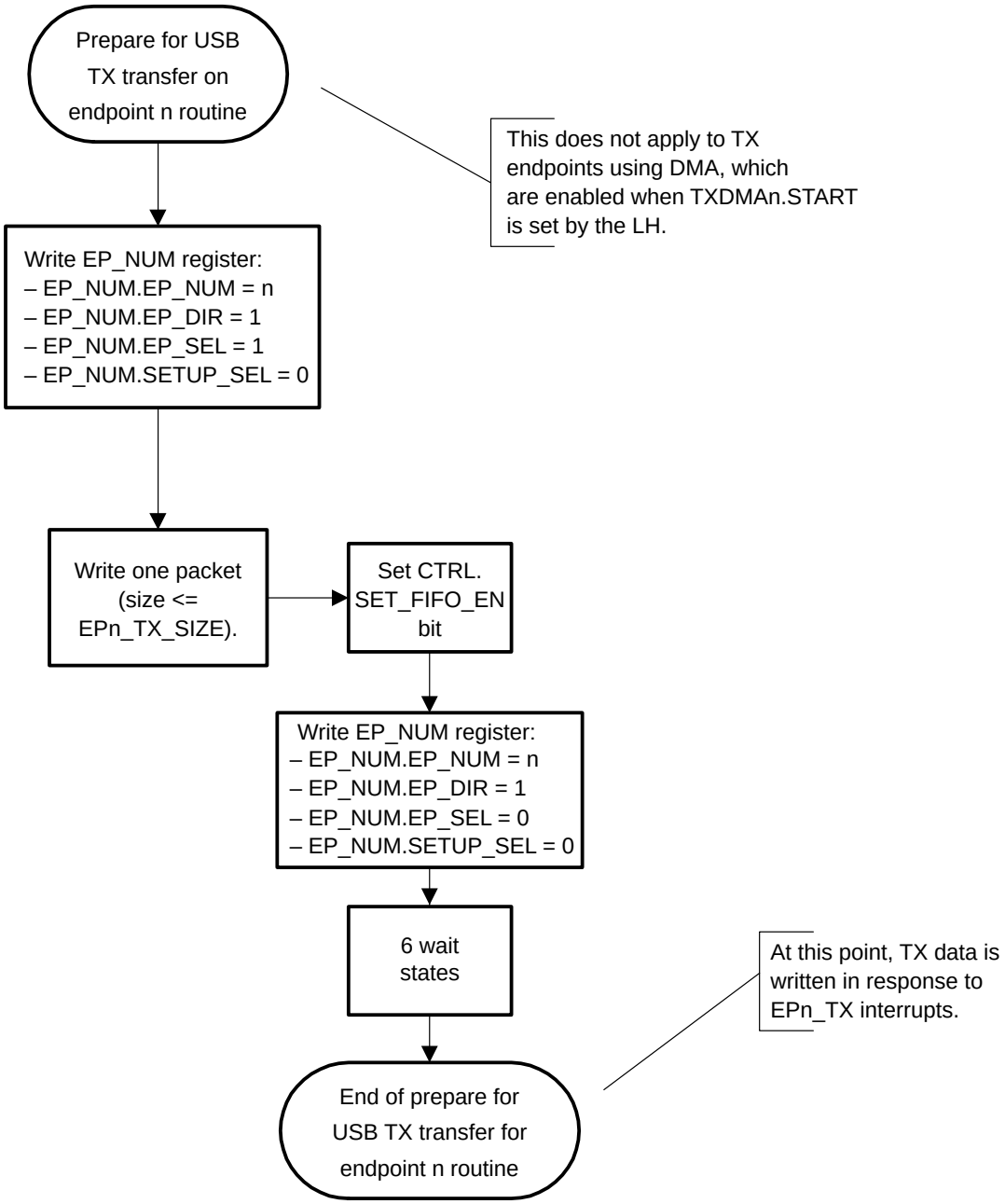


Figure 10–13. Prepare for TX Transfer on Endpoint n Routine



10.3.10 USB Device Interrupt Service Routine (ISR) Flowcharts

The flowcharts in this section give general operational guidelines for USB device ISR processing. System-architecture-specific details are left to the engineers who write the MPU and USB host code. One USB-specific interrupt register is provided (IRQ_SRC), including:

- ❏ General USB interrupts (including endpoint 0, DMA and device states interrupts) on MPU level 2 IRQ_20
- ❏ Non-ISO endpoint-specific interrupt on MPU level 2 IRQ_30
- ❏ Start of frame (SOF) interrupt for ISO transactions on MPU level 2 IRQ_29

The general USB interrupt ISR must handle non-autodecoded control transfers on endpoint 0 and some specialty interrupts generated because of USB device state modifications or DMA transfers. The ISR for the endpoint-specific interrupt must handle interrupts from the USB module that are generated because of USB activity for non-isochronous endpoints. The SOF ISR is responsible for handling isochronous endpoints and, if needed by the application, tracking the USB frame number. Many flowcharts are presented in this chapter to provide guidelines for how to handle the interrupts related to the USB device controller module. The flowcharts in this part suppose that the SYSCON1.NAK_EN bit is cleared.

Note:

A key assumption behind the flowcharts presented here is that the application provides separate buffers for each direction of endpoint, except for endpoint 0. The flowcharts read from these application buffers for IN transactions on TX endpoints and write to these application buffers for OUT transactions on RX endpoints.

The USB device controller does not support reentrant interrupts. Each USB device controller must be handled completely before handling another USB device controller interrupt. This restriction occurs because there is only one EP_NUM register, so endpoint control operations must be completed before working with another endpoint or endpoint direction.

10.3.11 Important Note on USB Device Interrupts

When an endpoint interrupt is asserted, the MPU writes the EP_NUM register with the EP_NUM.EP_SEL bit set to 1. The MPU must finish the interrupt handling before clearing the EP_NUM.EP_SEL bit, because clearing this bit clears the corresponding status bit in the STAT_FLG register (ACK, NAK, STALL). When an interrupt is pending on an endpoint, the MPU must not select and then unselect the endpoint without handling the interrupt, because this clears the pending transaction status flags. The MPU does not need to set EP_NUM.EP_SEL to 1 when setting CTRL.SET_FIFO_EN, CTRL.SET_HALT, and CTRL.CLR_HALT bits.

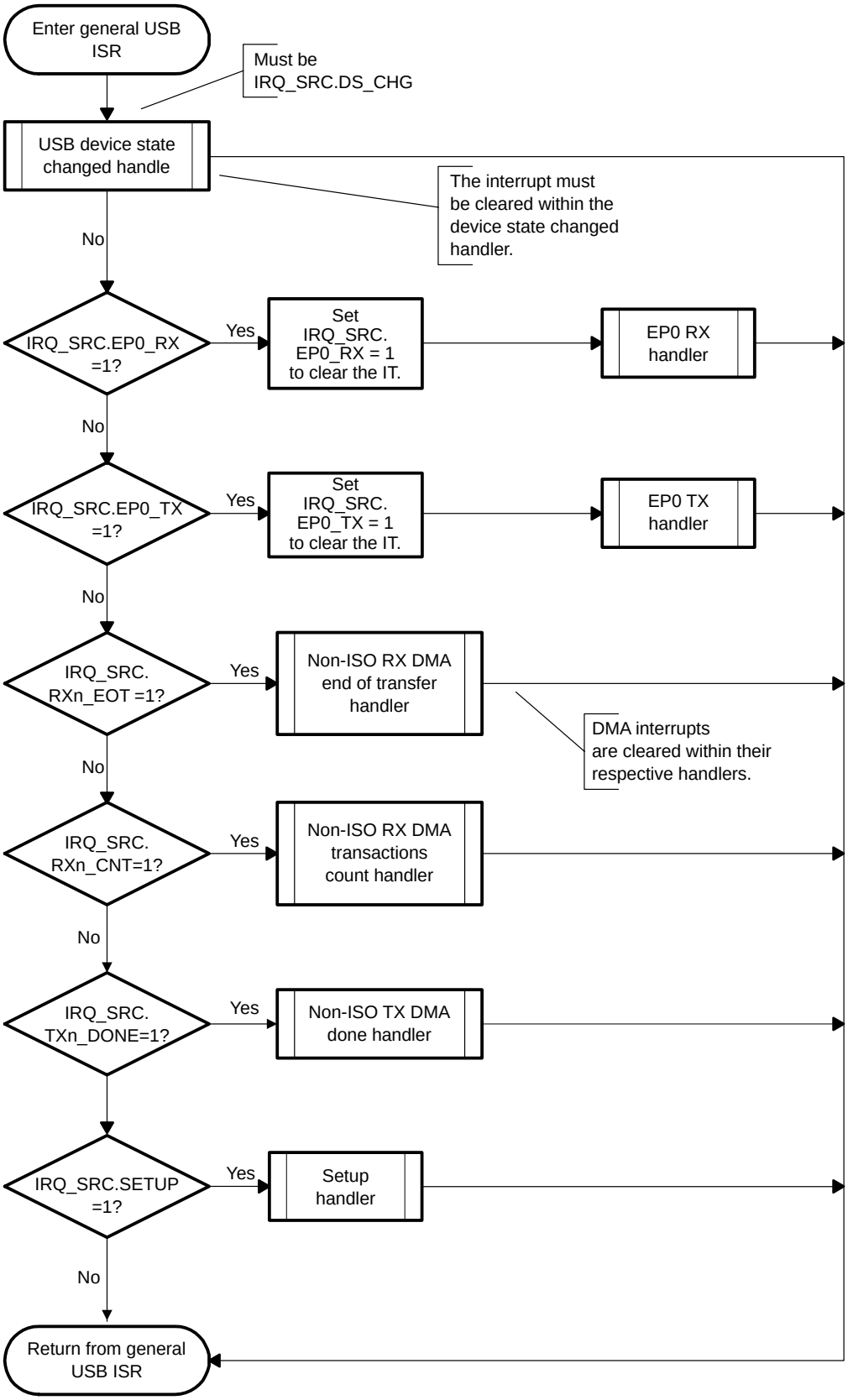
The endpoint status (STAT_FLG register) is updated at the end of each USB transaction if the previous transaction has been handled. If a pending interrupt has not been handled when a new non-transparent transaction occurs, status flags are not updated (and NAK is returned, even if FIFO was enabled, or STALLed if the EP halt feature was set), so that the MPU never misses an ACKed transaction. If double-buffering is used for an endpoint, STAT_FLG is updated if there is zero or one interrupt pending for the endpoint, and is not updated if there are already two interrupts pending on the endpoint.

The MPU does not need to set the SYSCON1.NAK_EN bit during normal operation. However, for debugging process, this bit can be set when the MPU finishes handling an EP interrupt without having set the corresponding CTRL.SET_FIFO_EN bit. During TX transaction, if the SYSCON1.NAK_EN bit is set, the MPU must wait for a NAK interrupt to write the TX data, to avoid a possible conflict caused by the NAK interrupt received while the MPU was writing the TX data.

10.3.12 Parsing General USB Device Interrupt

The general USB interrupt (MPU level 2 IRQ_20) ISR must parse the interrupt identifier register IRQ_SRC to determine the types of general USB interrupts that are active. These include interrupts relating to USB device state modifications (USB reset, suspend/resume, during enumeration phase) and control transfers on endpoint 0 or non-ISO DMA transfers in either receive or transmit mode. Multiple interrupts can be active at any time, and all must be dealt with by the ISR before returning from the ISR. Figure 10–14 shows an appropriate flowchart for parsing the general USB interrupts.

Figure 10–14. General USB Interrupt ISR Source Parsing Flowchart



10.3.13 Setup Interrupt Handler

A separate interrupt flag exists for setup transactions so that the MPU cannot miss a setup transaction, even if it occurs during the data or status phase of another transfer (case of an aborted transfer). The setup parsing function captures the control transfer request information for use in determining which USB bus activity is needed and in controlling how the MPU must generate or respond to the control transfer. This information includes the following:

- bmRequestType
- bmRequest
- wValue
- wIndex
- wLength

The setup interrupt handler shown in Figure 10–15 is responsible for processing setup transactions occurring on endpoint 0. It calls the routine that parses the control transfer request information, shown in Figure 10–16 to set flags that the rest of the ISR code can use to control proper response to control transfers. Two flags are set by the setup interrupt handler, to be used during endpoint 0 interrupt handlers:

- Control read flag
- Control write flag

Figure 10–15. Setup Interrupt Handler

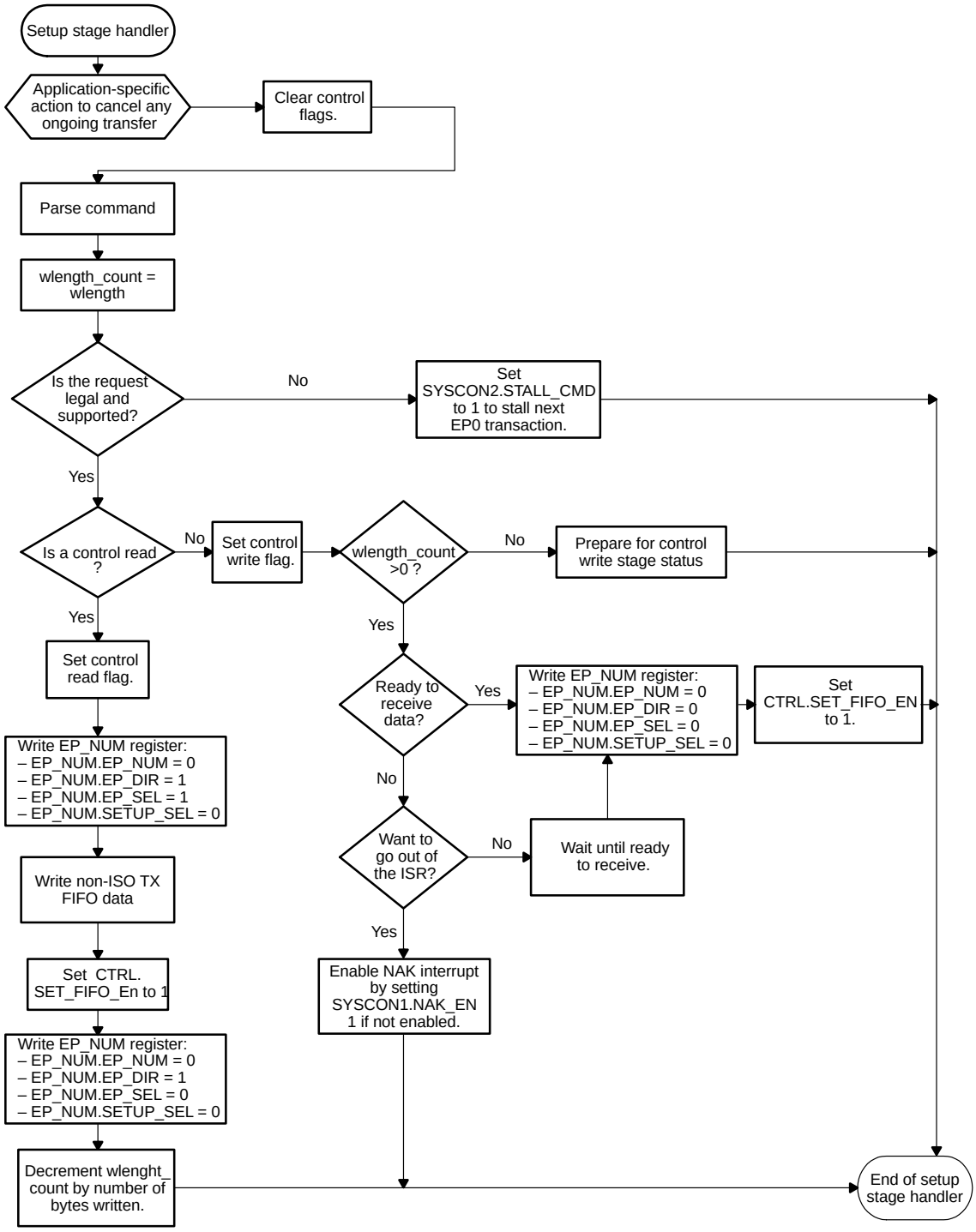
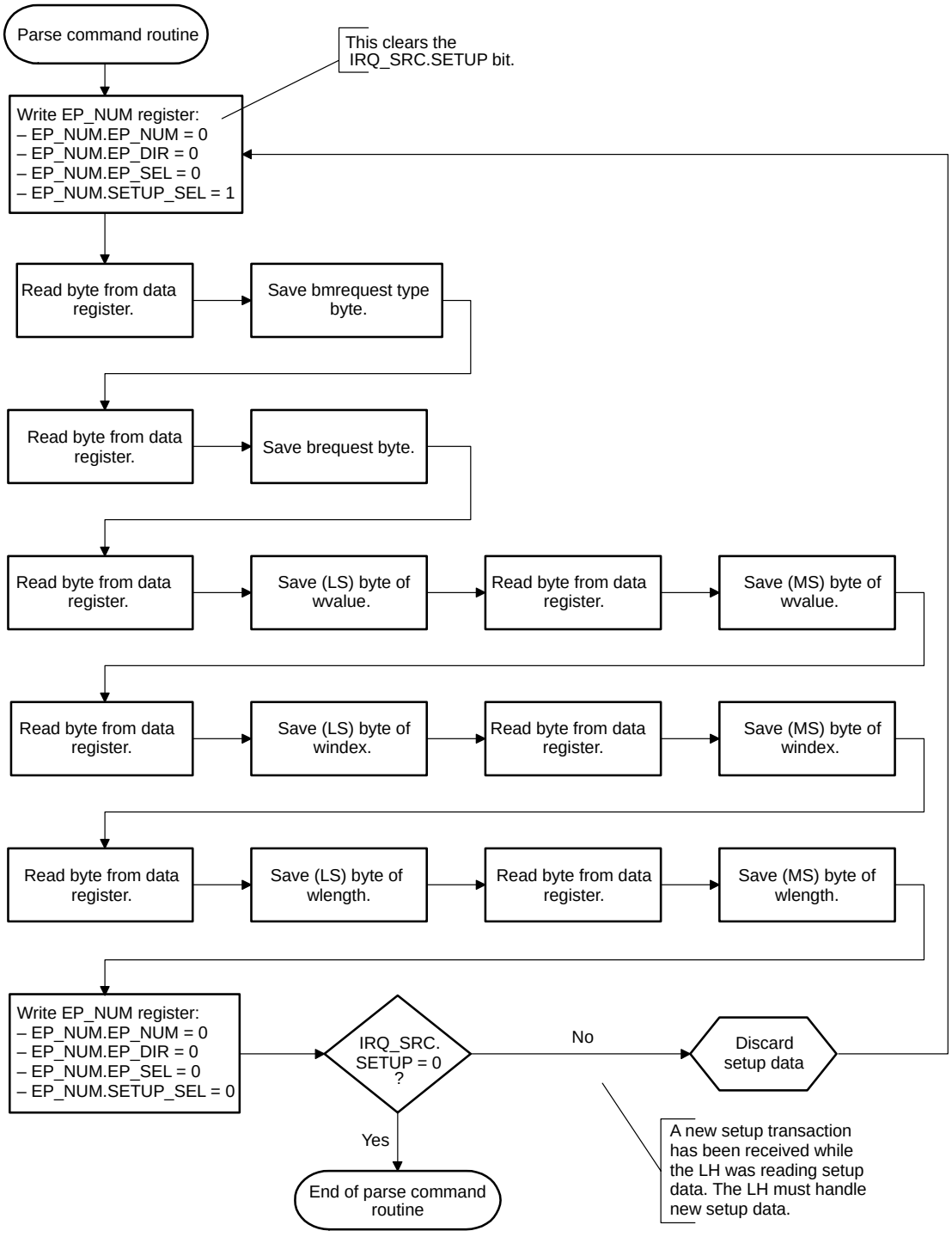


Figure 10–16. Parse Command Routine (Setup Stage Control Transfer Request)



10.3.14 Endpoint 0 RX Interrupt Handler

Figure 10–17 shows the endpoint 0 RX portion of the general USB interrupt handler, which must handle general USB interrupts related to control OUT transactions on endpoint 0. No EP0 interrupt is generated for autodecoded control transfers.

Figure 10–17. Endpoint 0 RX Interrupt Handler

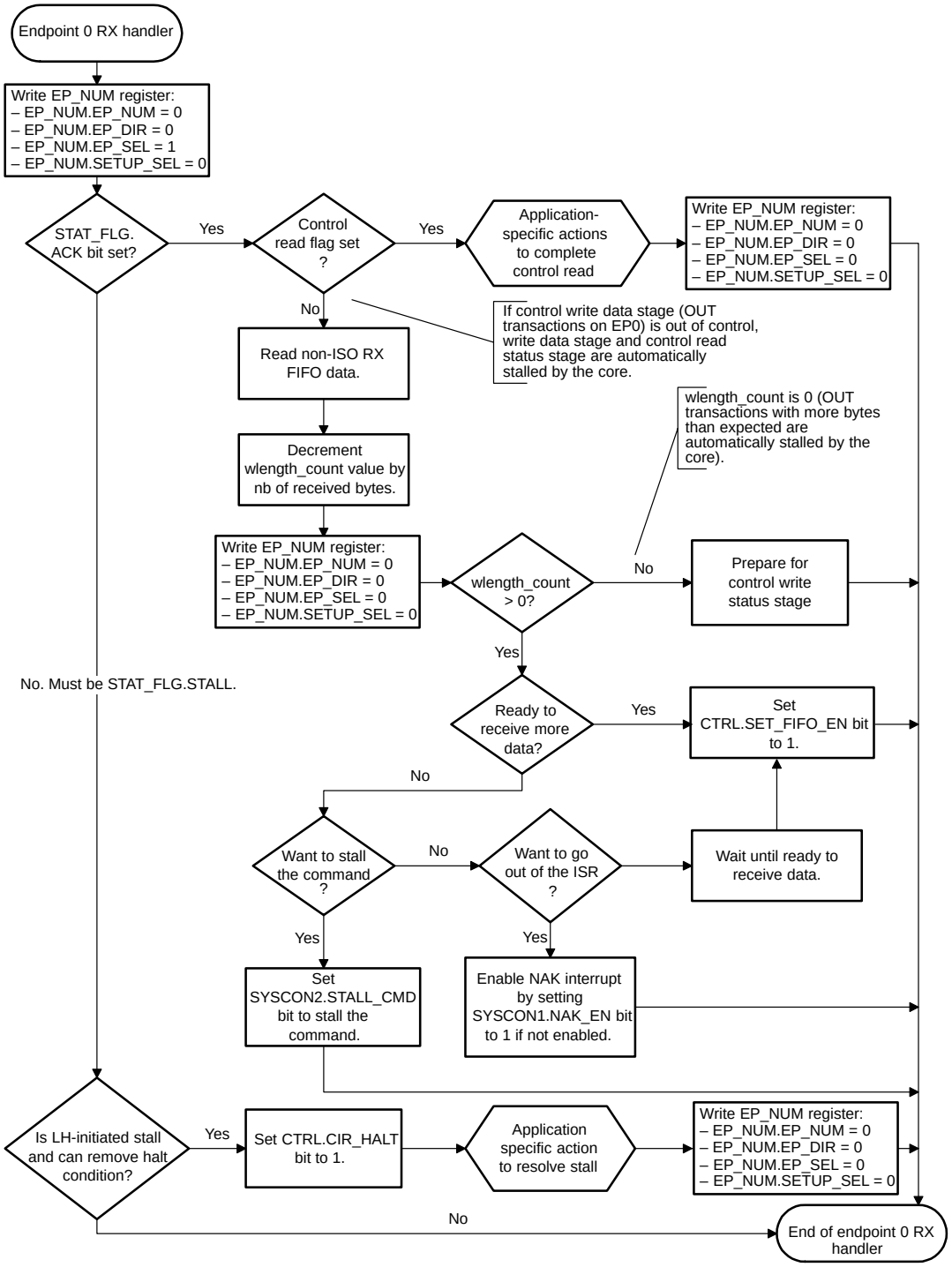
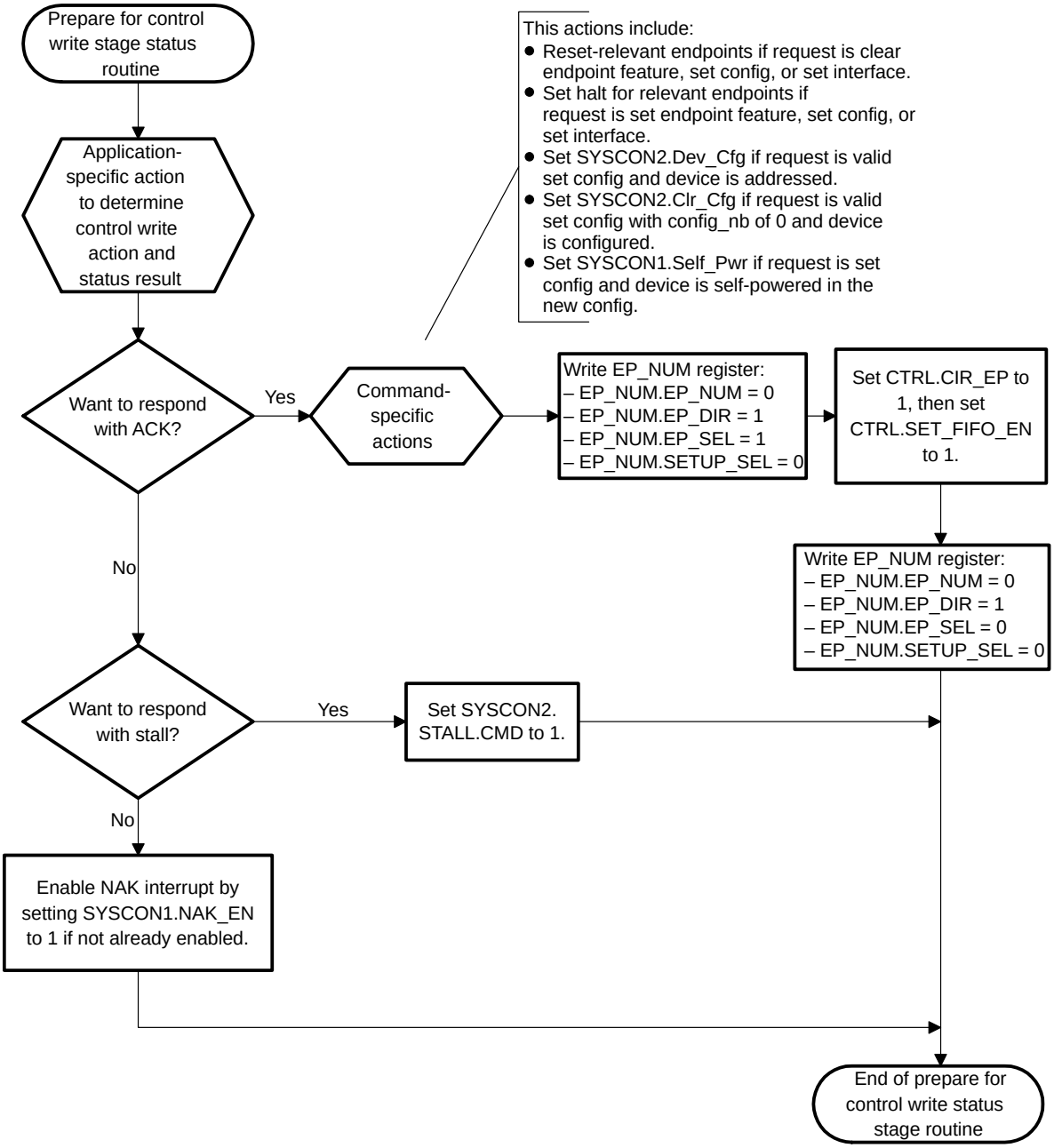


Figure 10–18. Prepare for Control Write Status Stage Routine



To support the SET_DESCRIPTOR command, when in commspecific actions, you must first reset all endpoints by selecting the endpoint and then clear it, unlock the configuration, charge the new one, lock the configuration, go in prepare for transfers, and enable all interrupts (see Figure 10–18).

10.3.15 Endpoint 0 TX Interrupt Handler

Figure 10–19 shows the TX portion of the general USB interrupt handler, which must handle general USB interrupts related to control IN transactions on endpoint 0.

The endpoint 0 TX interrupt handler must be able to move data into the endpoint 0 TX FIFO when the application buffer for endpoint 0 TX data is not empty and an endpoint 0 TX interrupt occurs signaling an ACKed non-autodecoded endpoint 0 IN transaction. This data can be control read data stage information or control write status stage handshaking information (see Figure 10–20).

Figure 10–19. Endpoint 0 TX Interrupt Handler

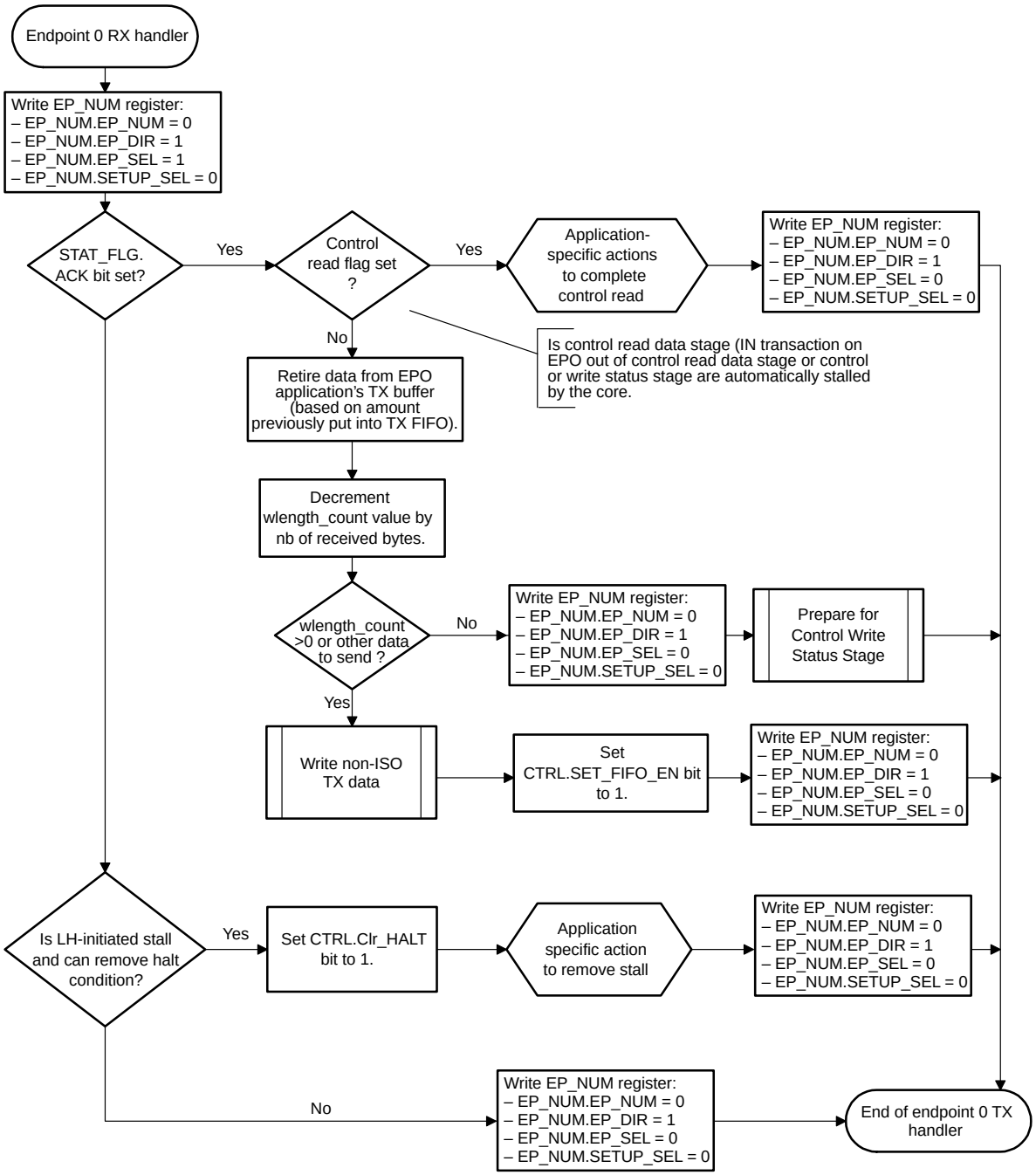
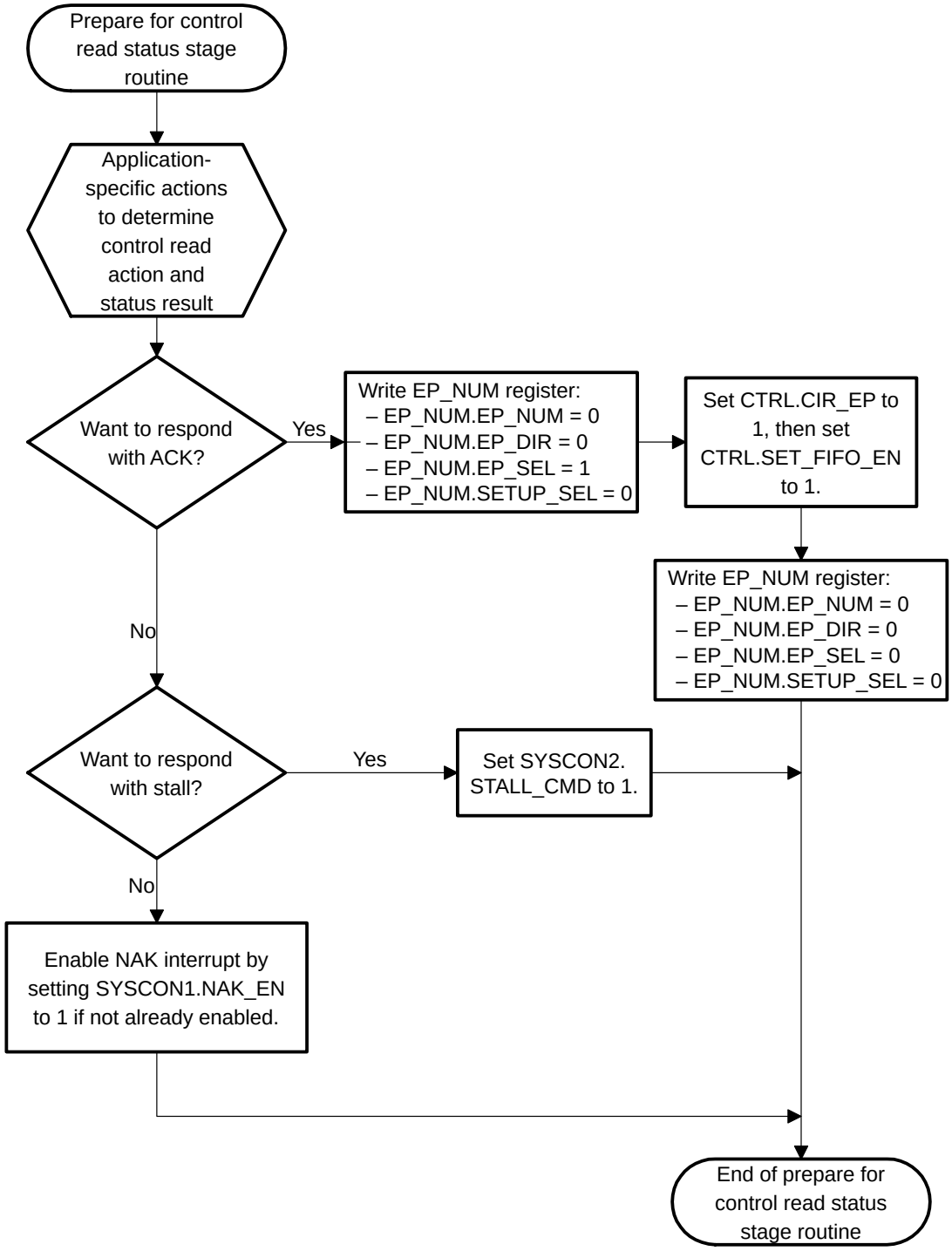


Figure 10–20. Prepare for Control Read Status Stage Routine



10.3.16 Device States Changed Handler

This section describes how USB device states and transitions states are decoded by the USB device controller and how they can be handled.

The state machine (see Figure 10–21) shows how the USB device controller device moves from one state to another state with respect to the USB1.1 specification. Attach/unattach transition is not shown in the transition flow.

Because SET_CONFIGURATION is not decoded by the core, the MPU has the responsibility to distinguish a SET_CONFIGURATION with a nonvalid configuration value from other SET_CONFIGURATION requests and to set SYSCON2.DEV_CFG only if the configuration value is valid (value 0 is nonvalid), when the device is in addressed state. When the device is in configured state, the MPU has the responsibility to set SYSCON2.CLR_CFG if the configuration number is 0 so that the device moves to addressed state.

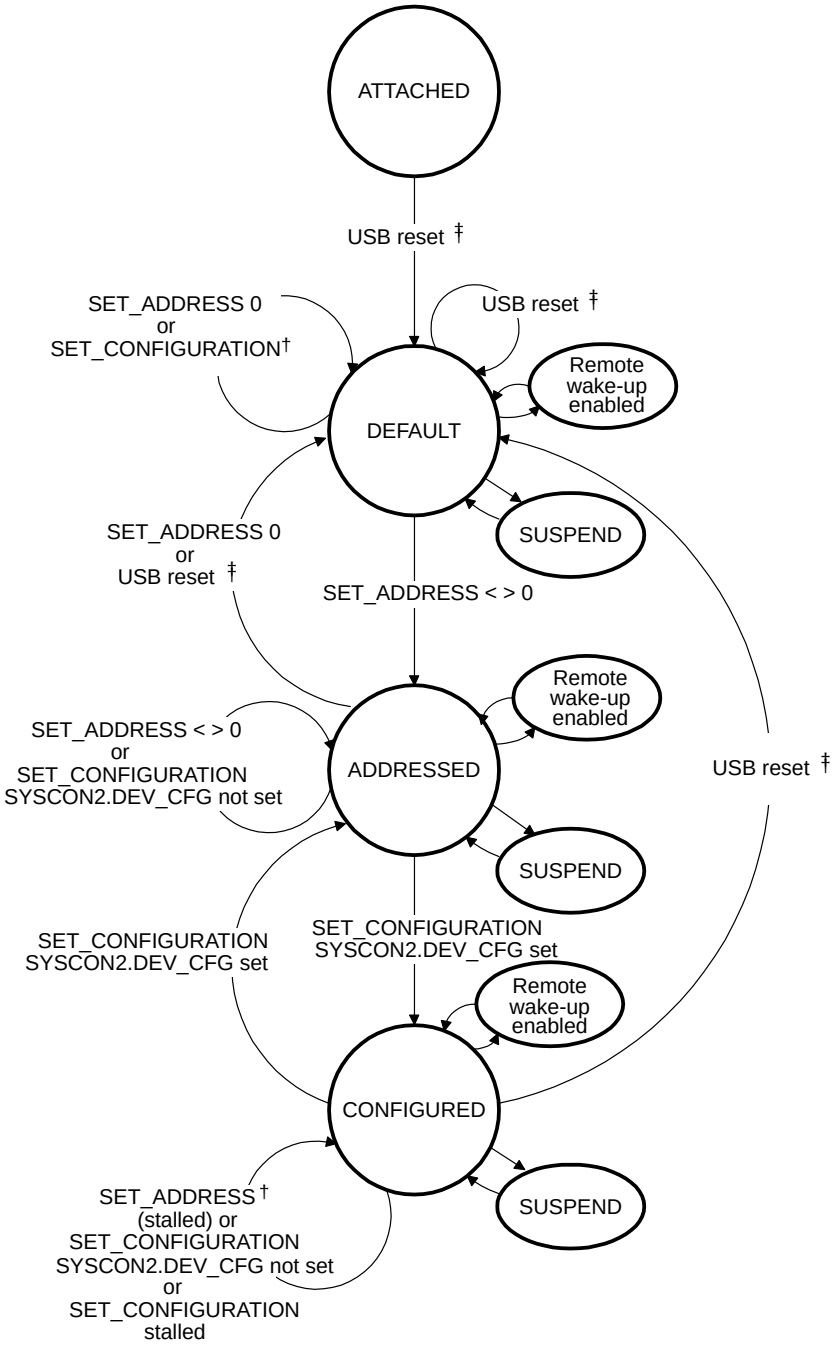
Device states are visible in the DEVSTAT register and are decoded as follows.

- Attached: The device is attached to the USB and powered.
- Default: The device is attached to the USB, is powered, and has been reset.
- Addressed: The device is attached to the USB, is powered, has been reset, and an address has been assigned. The device moves into the addressed state after a SET_ADDRESS request with an address number other than 0.
- Configured: The device is attached to the USB, is powered, has been reset, has an address other than 0, and is configured. The device moves into the configured state after a valid SET_CONFIGURATION request, only if the MPU has set the SYSCON2.DEV_CFG bit (meaning the configuration is valid).
- Suspended: The device is at minimum default and has not seen bus activity for 5 ms.
- Reset: When set, the device is receiving a valid USB host reset.
- R_WK_OK: This bit is set (cleared) automatically after a valid SET_DEVICE_FEATURE (CLEAR_DEVICE_FEATURE) request from the USB host.

Any change in the DEVSTAT register bits triggers a device change interrupt (IRQ_SRC.DS_CHG), if enabled.

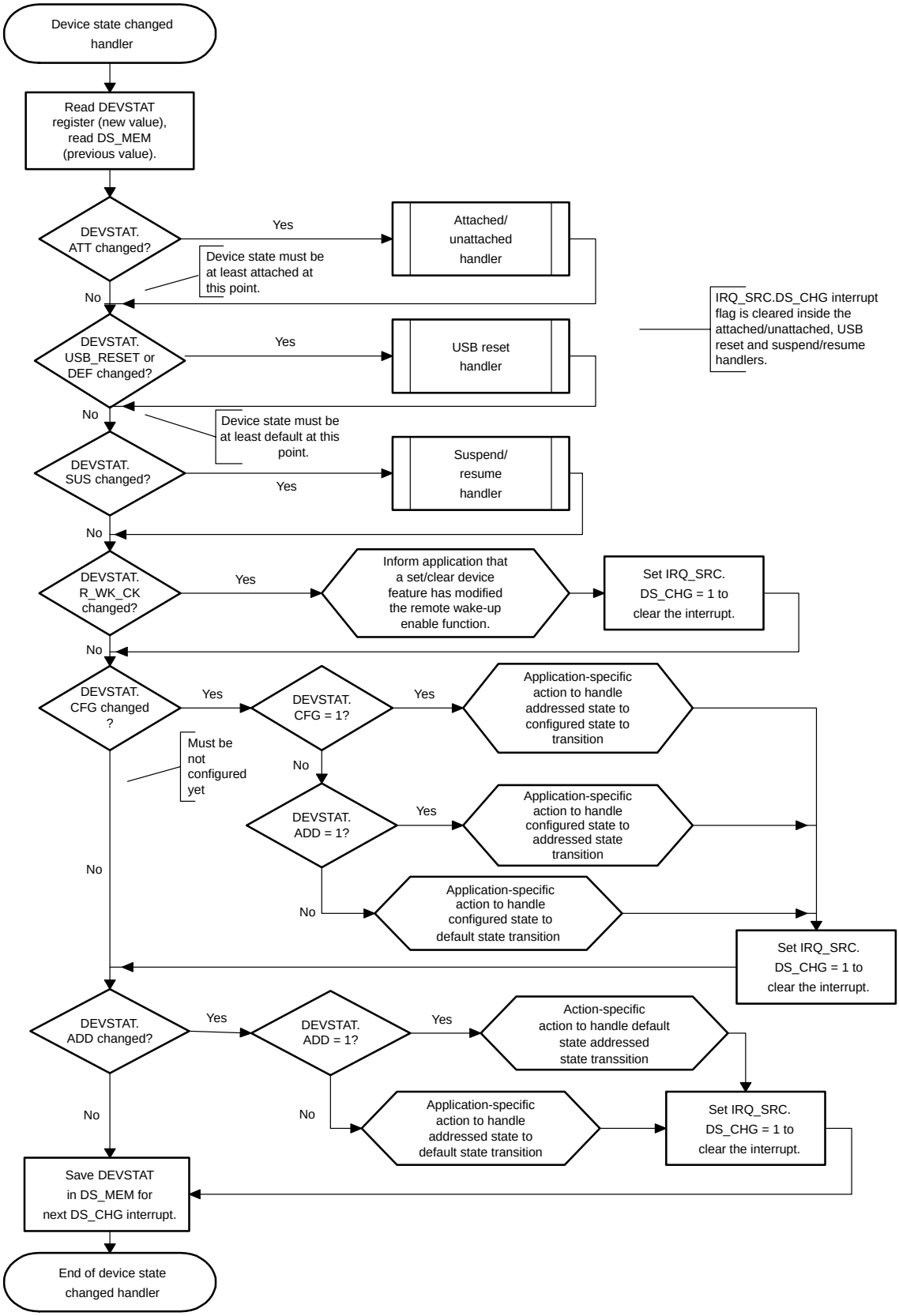
The device moves to addressed state after the status stage of a valid SET_ADDRESS, even if the status stage ACK handshake is received corrupted or not sent by the USB host. A SET_DEVICE_FEATURE or a CLEAR_DEVICE_FEATURE is effective after setup transaction, even if no status stage occurs. A SET_CONFIGURATION request is effective before status stage, when the MPU sets the SYSCON2.CLR_CFG or SYSCON2.DEV_CFG bit (see Figure 10–22).

Figure 10–21. USB Device Controller Device State Transitions



†Behavior not specified by USB 1.1 specifications (see chapter 9)
‡USB reset generates two interrupts (when USB reset is asserted and then when USB reset completes).
No interrupt is asserted by the core for tansitions shown with dashed lines.

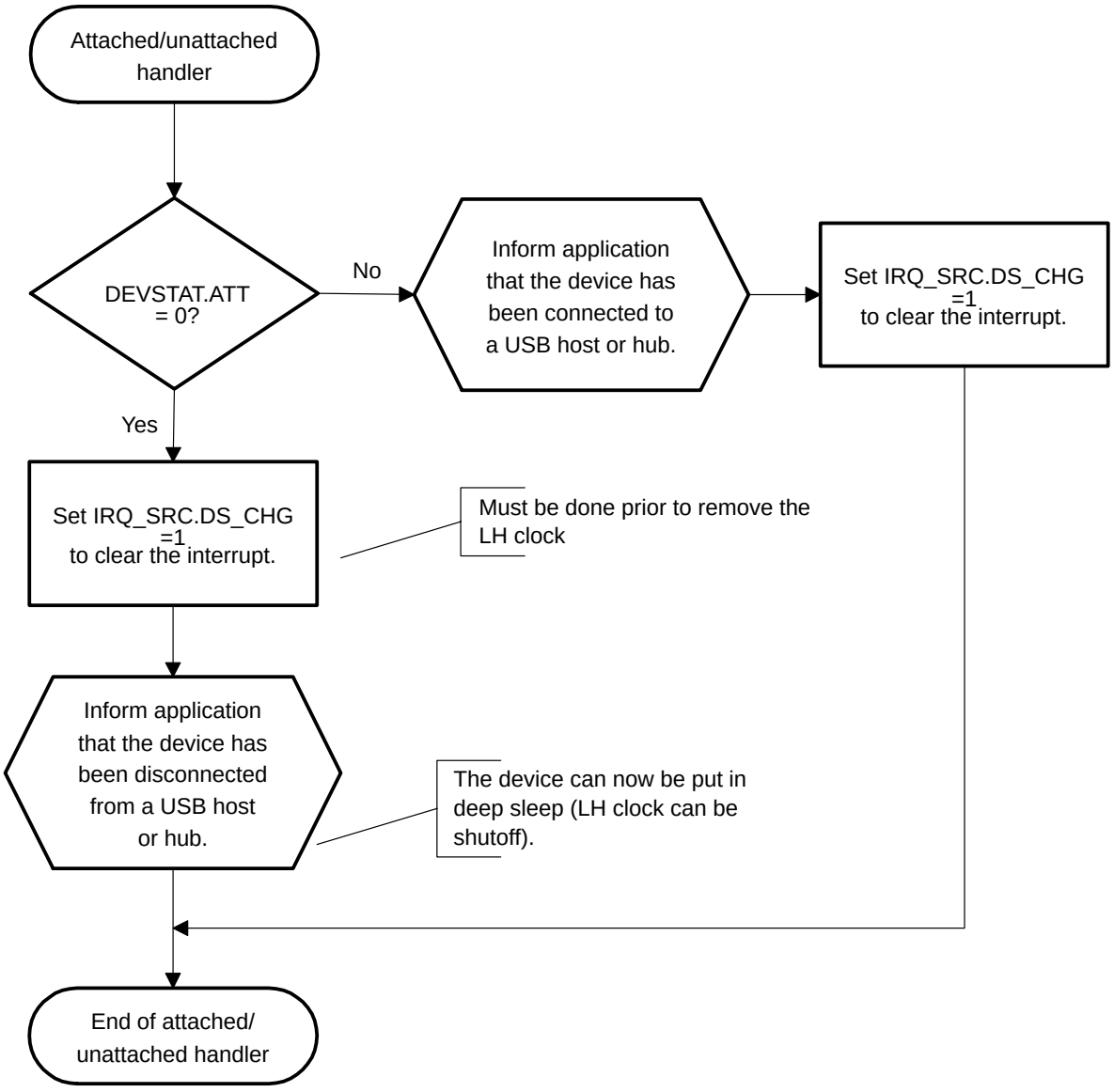
Figure 10–22. Typical Operation for USB Device State Changed Interrupt Handler



10.3.17 Device States Attached/Unattached Handler

Device attached/unattached interrupts occur when the device detects that its VBUS has changed. System software can disable the USB device controller clock after IRQ_SRC.DS_CHG is cleared after servicing an unattached interrupt. Disabling the USB device controller clock before IRQ_SRC.DS_CHG is cleared can result in improper functionality for future USB device controller interrupts (see Figure 10–23).

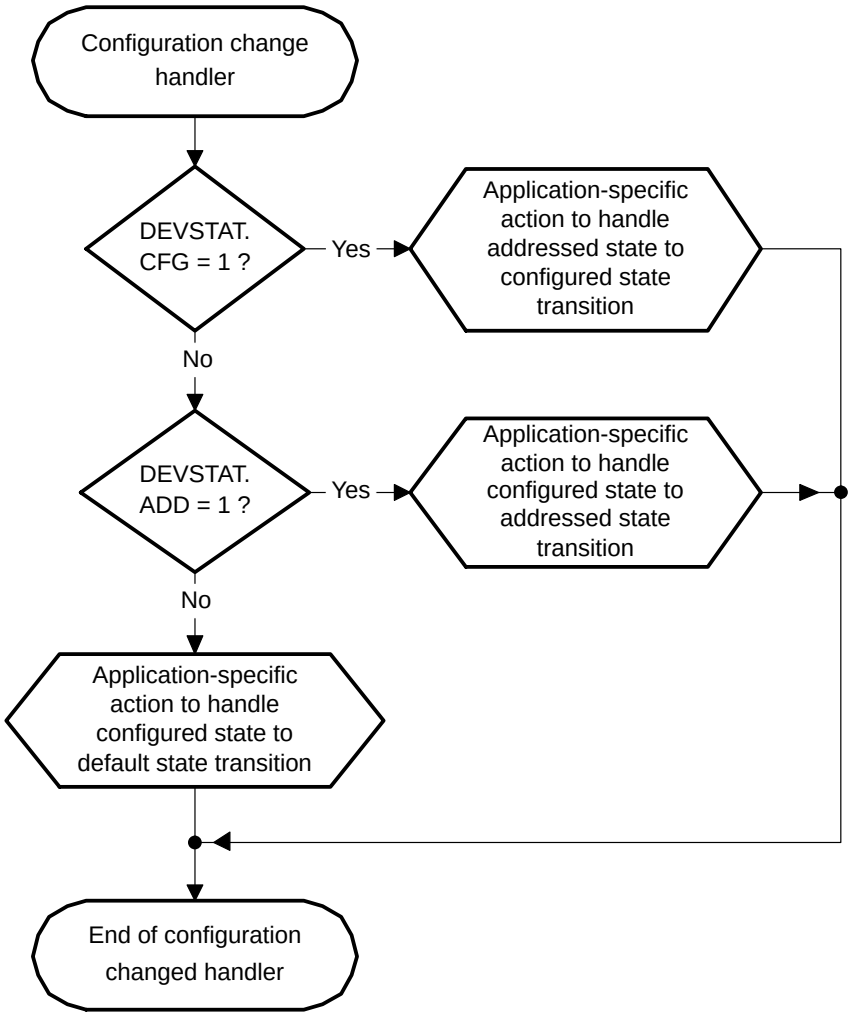
Figure 10–23. Attached/Unattached Handler



10.3.18 Device State Configuration Changed Handler

When a configuration changed interrupt occurs, the USB device has received a set configuration operation. When this occurs, the configuration-changed handler performs the operations shown in Figure 10–24.

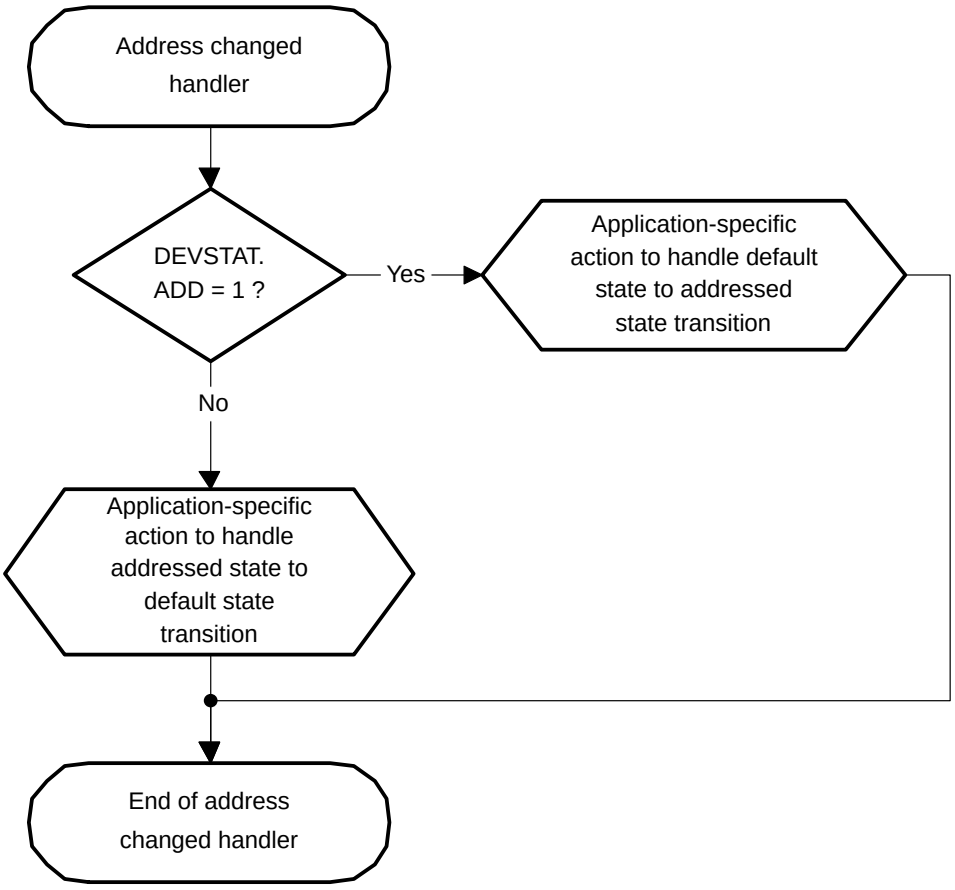
Figure 10–24. Configuration Changed Handler



10.3.19 Device State Address Changed Handler

When an address changed interrupt occurs, the USB device has received a set address operation. When this occurs, the address-changed handler performs the operations shown in Figure 10–25.

Figure 10–25. Address Changed Handler

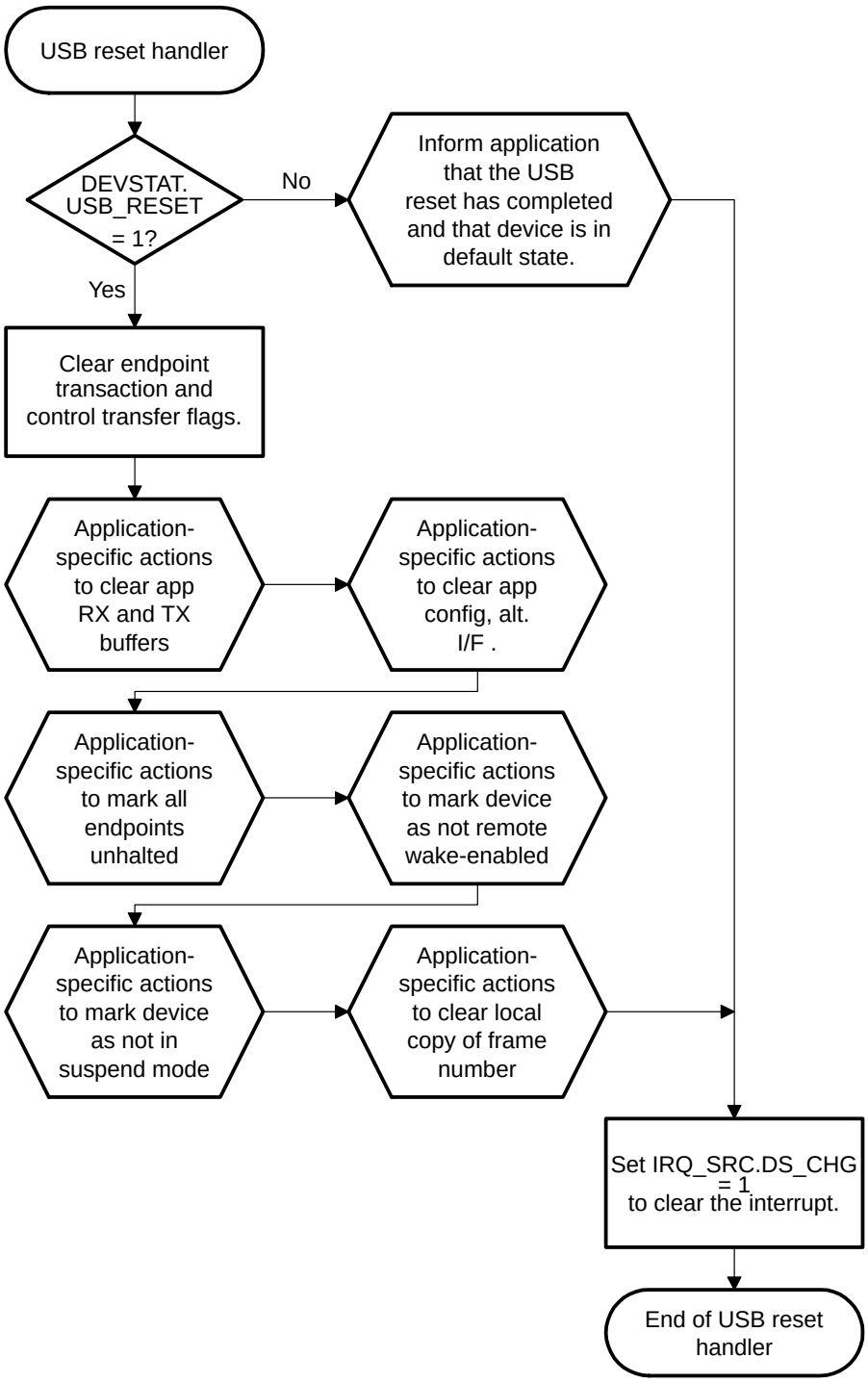


10.3.20 USB Device Reset Interrupt Handler

When a USB reset occurs, the USB module generates a general USB interrupt to the MPU (see Figure 10–26). The MPU responds to this interrupt by performing the following operations:

- ❏ Cancels any ongoing USB transaction and/or control transfer handling
- ❏ Clears any copies that the application has of configuration number or alternate interface numbers
- ❏ Clears any application-specific information relating to halted endpoints
- ❏ Clears any application-specific information relating to the remote wake enable flag
- ❏ Clears any application-specific information relating to the suspend mode flag
- ❏ Clears any application-specific copy of the frame number

Figure 10–26. USB Device Reset Handler Flowchart



10.3.21 Suspend/Resume Interrupt Handler

When a USB device suspend/resume general USB interrupt occurs, the USB module has either entered or left suspend mode. The MPU code must determine which and react appropriately (see Figure 10–27).

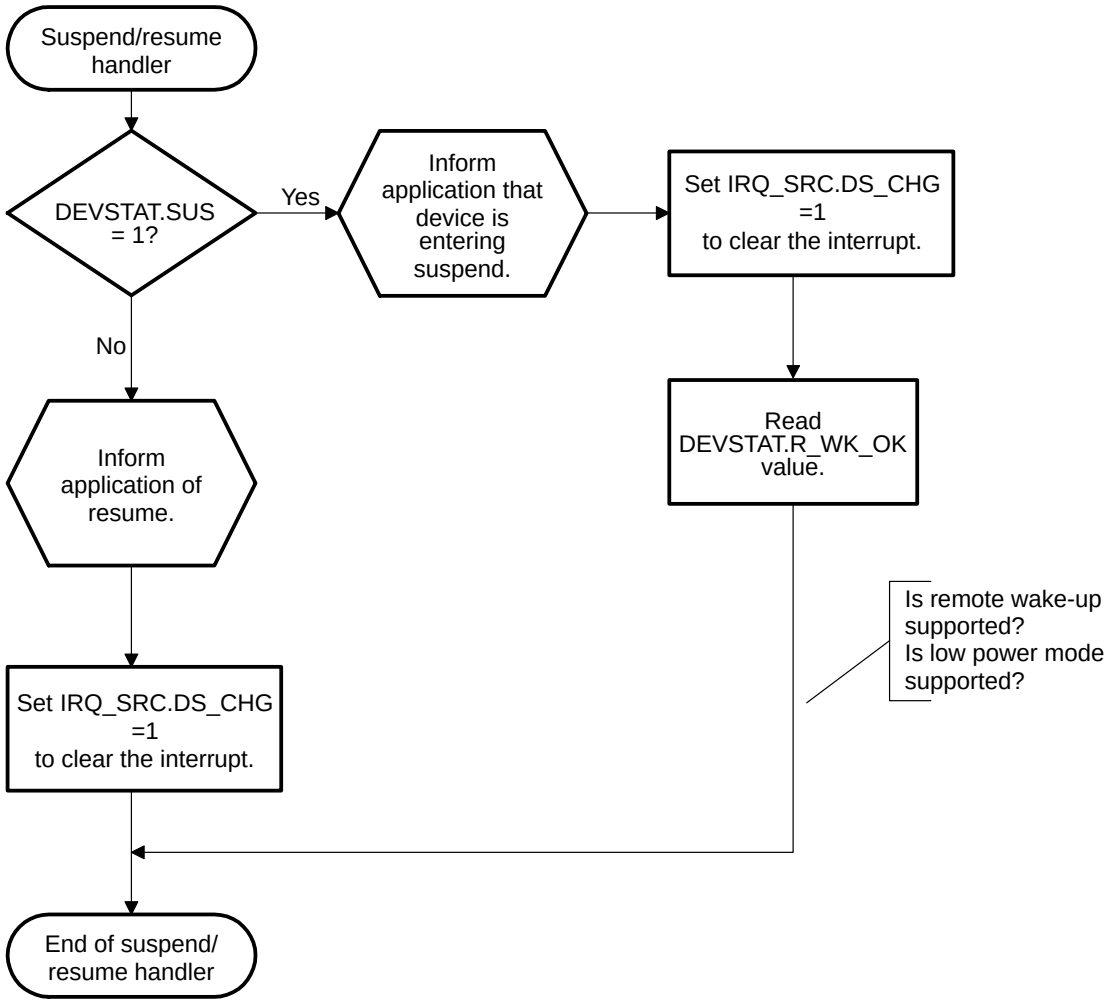
The suspend sense hardware is implemented to trigger only after 5 ms of bus idle. This forces compliance with the *USB Specification Version 1.1* T_{WTRSM} timing parameter (3 ms of IDLE to identify suspend, 2 ms before the remote device can signal resume).

If the MPU wants to wake the device from suspend mode and remote wake-up enable is set (DEVSTAT.R_WK_OK = 1), it must first turn its clock on (if stopped), and then set SYSCON2.RMT_WKP. The device then drives resume.

If shutoff is enabled (SYSCON1.SOFF_DIS=0), the 48-MHz clock is automatically shut off at suspend and turned on at resume (USB host or MPU driven). Setting or not setting the SYSCON1.SOFF_DIS bit is part of the device configuration. However, the MPU can modify its value at suspend interrupt time if necessary.

A USB reset is also a valid way to exit suspend mode. But the suspend/resume handler and the USB reset handler do not have to take this into account, because three interrupts are generated in that case (1 for resume, 1 for reset, and 1 for end of reset).

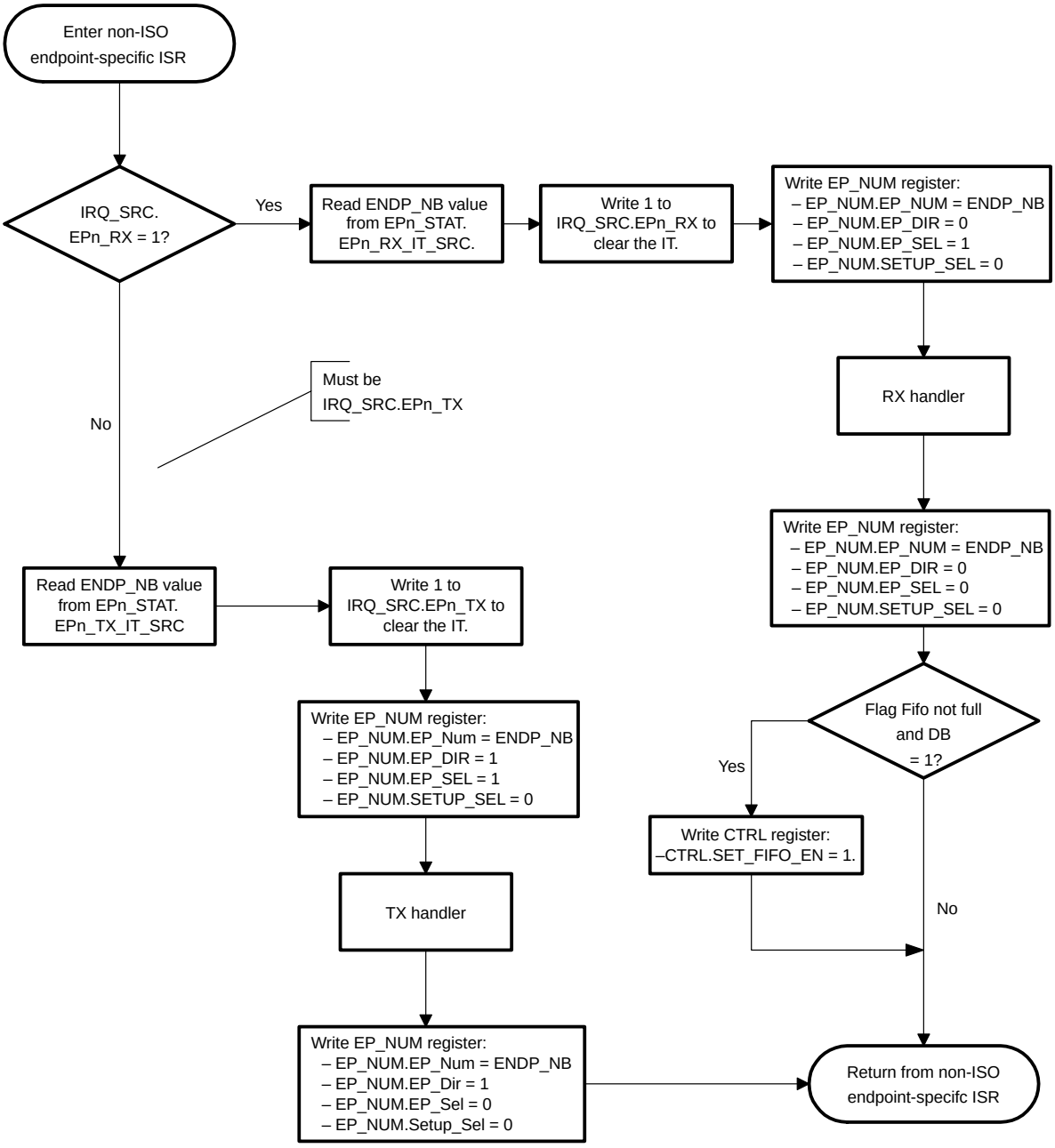
Figure 10–27. Typical Operation for USB Suspend/Resume General USB Interrupt Handler



10.3.22 Parsing Non-ISO Endpoint-Specific Interrupt

The endpoint-specific interrupt ISR (also known as the non-isochronous interrupt, on MPU Level 2 IRQ_30) must parse the interrupt identifier register IRQ_SRC to determine the interrupts that are active (IRQ_SRC.EPN_RX, IRQ_SRC.EPN_TX, or both). The two interrupts can be active at any time. The ISR must then read the EPN_STAT register to determine the endpoint causing the interrupt. For each direction, only one endpoint interrupt can be active at a time (see Figure 10–28).

Figure 10–28. Non-ISO Endpoint-Specific (Except EP 0) ISR Flowchart



10.3.23 Non-ISO, Non-Control OUT Endpoint Receive Interrupt Handler

Figure 10–29 shows the operations necessary to handle non-ISO, non-control OUT endpoint-specific receive interrupts. This flowchart shows two different RX transaction handshaking interrupts. There is a third interrupt handshaking possibility when NAK interrupts are enabled, which is not depicted here. Depending on the application-specific actions needed for various endpoints in the real system, it is possible to use one routine that is common to all of the non-ISO, non-control receive endpoints, where the only differences are in the

EP_NUM register value set and the selection of the proper application RX data buffer in the read non-ISO RX FIFO data routine (see Figure 10–30).

This flowchart does not attempt to document control endpoint 0 receive interrupts, which are discussed separately because of the more complex three-stage transfer mechanism used for control writes.

Figure 10–29. Non-Isochronous Non-Control Endpoint Receive Interrupt Handler

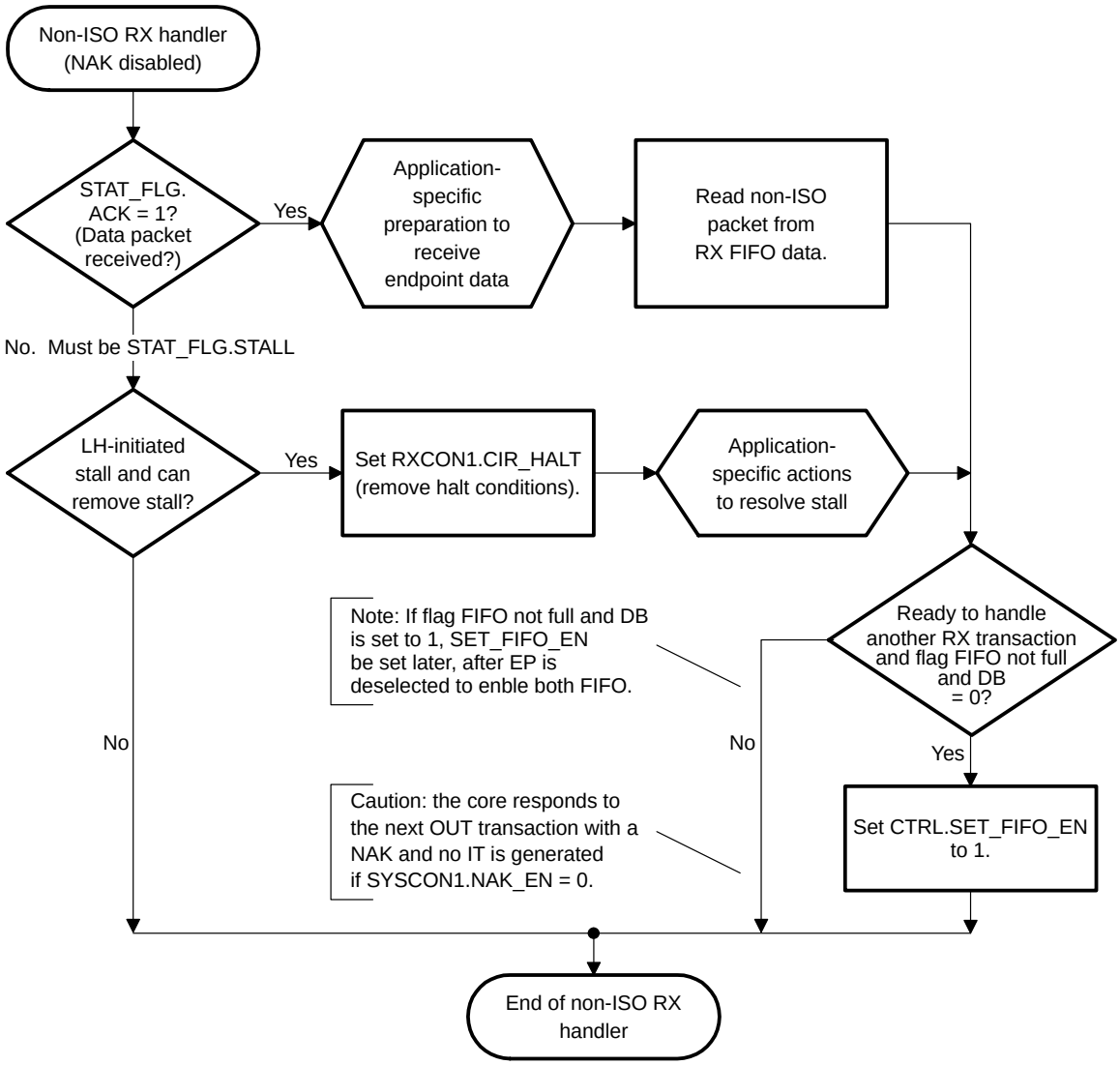
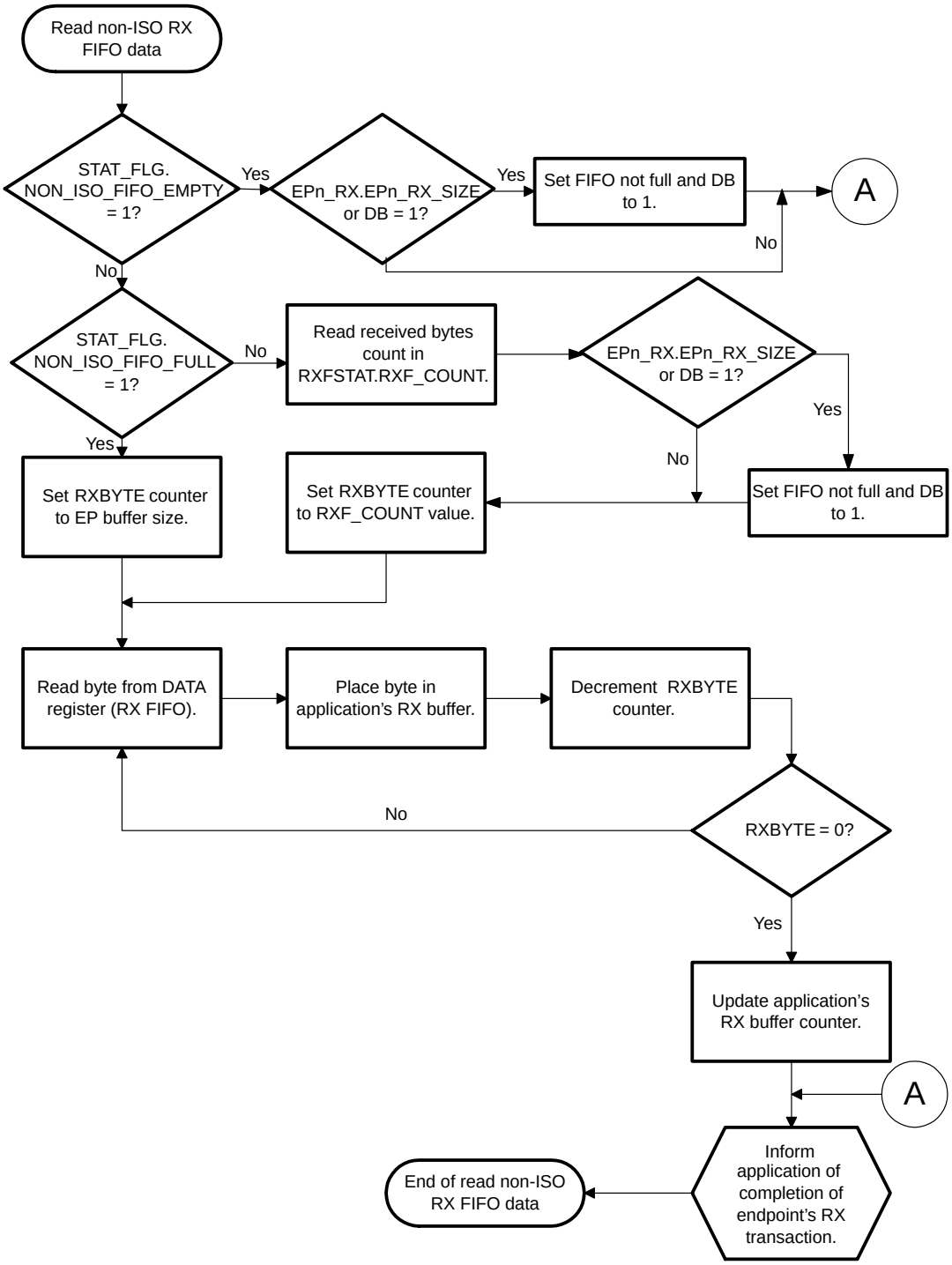


Figure 10–30. Read Non-Isochronous RX FIFO Data Flowchart



10.3.24 Non-ISO, Non-Control IN Endpoint Transmit Interrupt Handler

Figure 10–31 shows the operations necessary to handle non-ISO, non-control IN endpoint-specific transmit interrupts. This flowchart shows two TX transaction handshaking interrupts. There is a third interrupt handshaking possibility when NAK interrupts are enabled, which is not depicted here. Depending on

the application-specific actions needed for various endpoints in the real system, it is possible to use one routine that is common to all of the non-ISO, non-control transmit endpoints, where the only differences are in the EP_NUM register value set and in the routine selection of the application TX buffer (see Figure 10–32).

This flowchart does not attempt to document control endpoint 0 transmit interrupts, which are discussed separately because of the more complex three-stage transfer mechanism used for control reads.

Figure 10–31. Non-Isochronous Non-Control Endpoint Transmit Interrupt Handler

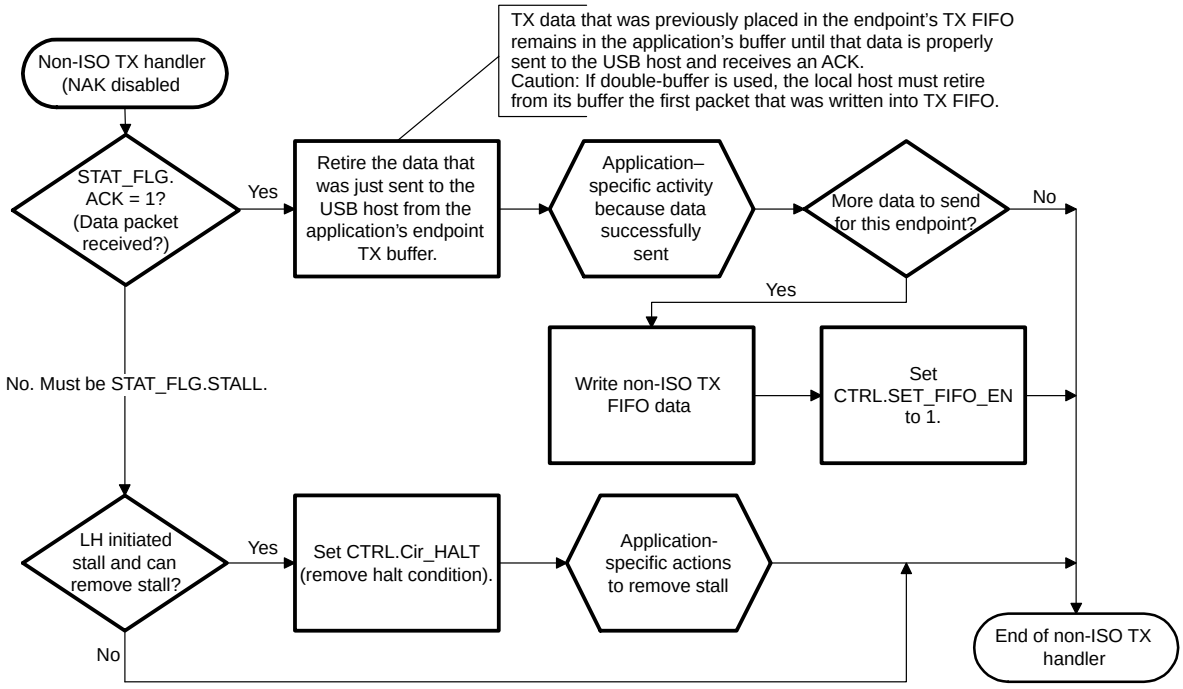
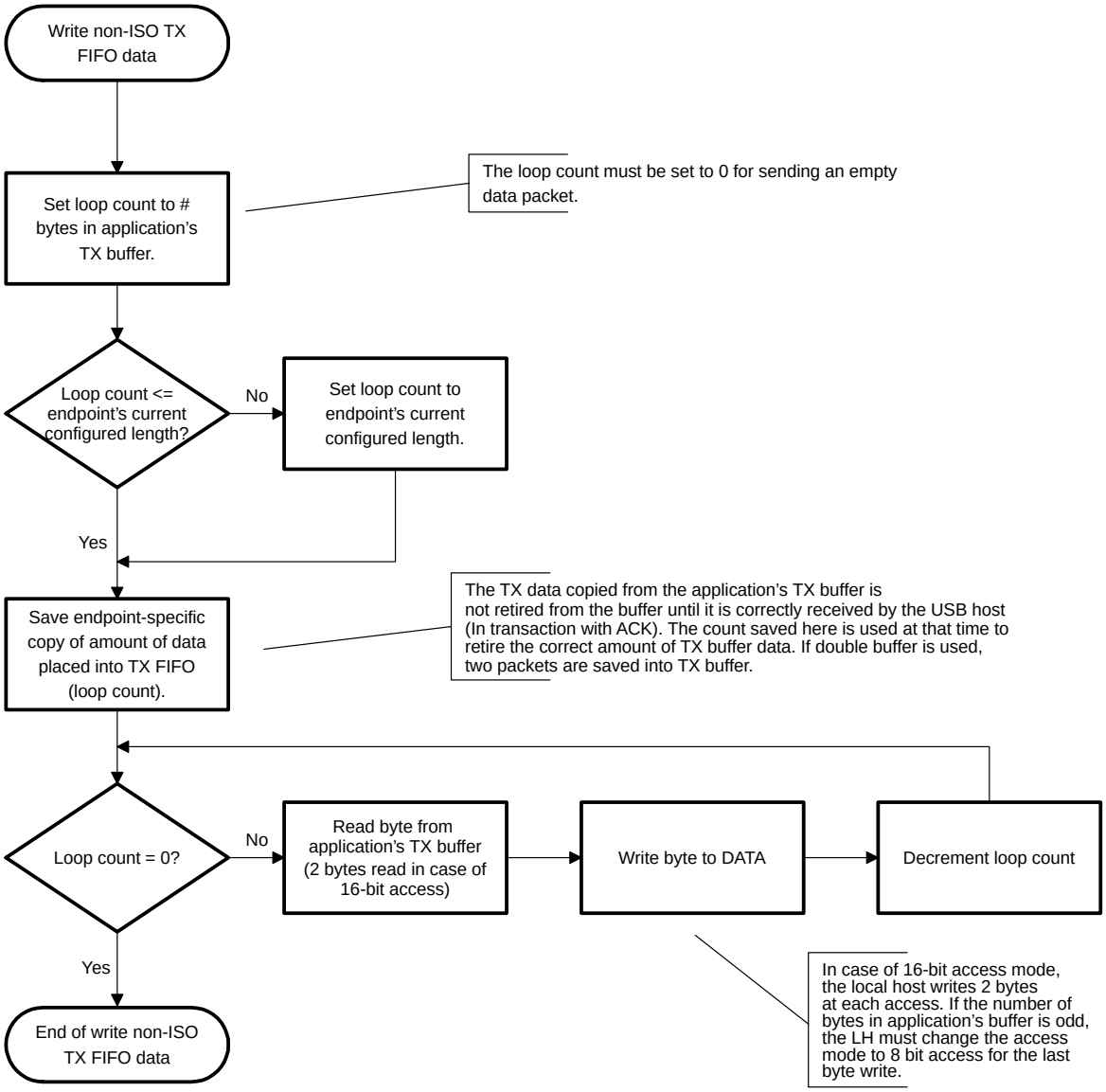


Figure 10–32. Write Non-Isochronous TX FIFO Data Flowchart



10.3.25 SOF Interrupt Handler

SOF interrupts to the MPU (also known as isochronous interrupts on MPU level 2 IRQ_29) occur once per USB frame. The MPU must handle data traffic for the isochronous endpoints at each SOF interrupt. In addition, the SOF ISR can handle any application-specific activity related to the implicit timing of the SOF interrupt. Figure 10–33 shows the SOF ISR flowchart. The read ISO RX FIFO data and write ISO TX FIFO data procedures are shown Figure 10–34 and Figure 10–35, respectively.

Figure 10–33. SOF Interrupt Handler Flowchart

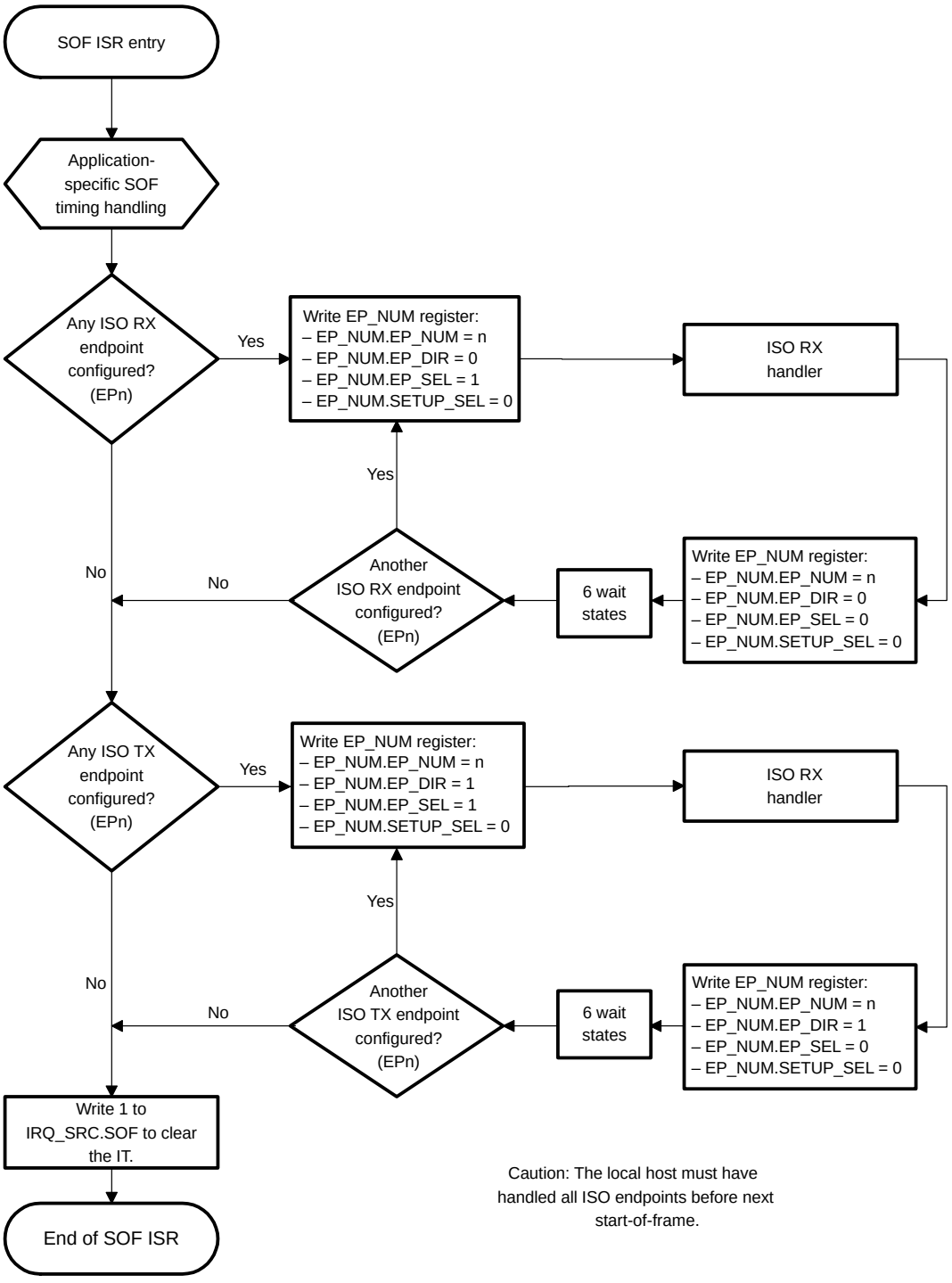


Figure 10–34. Read Isochronous RX FIFO Data Flowchart

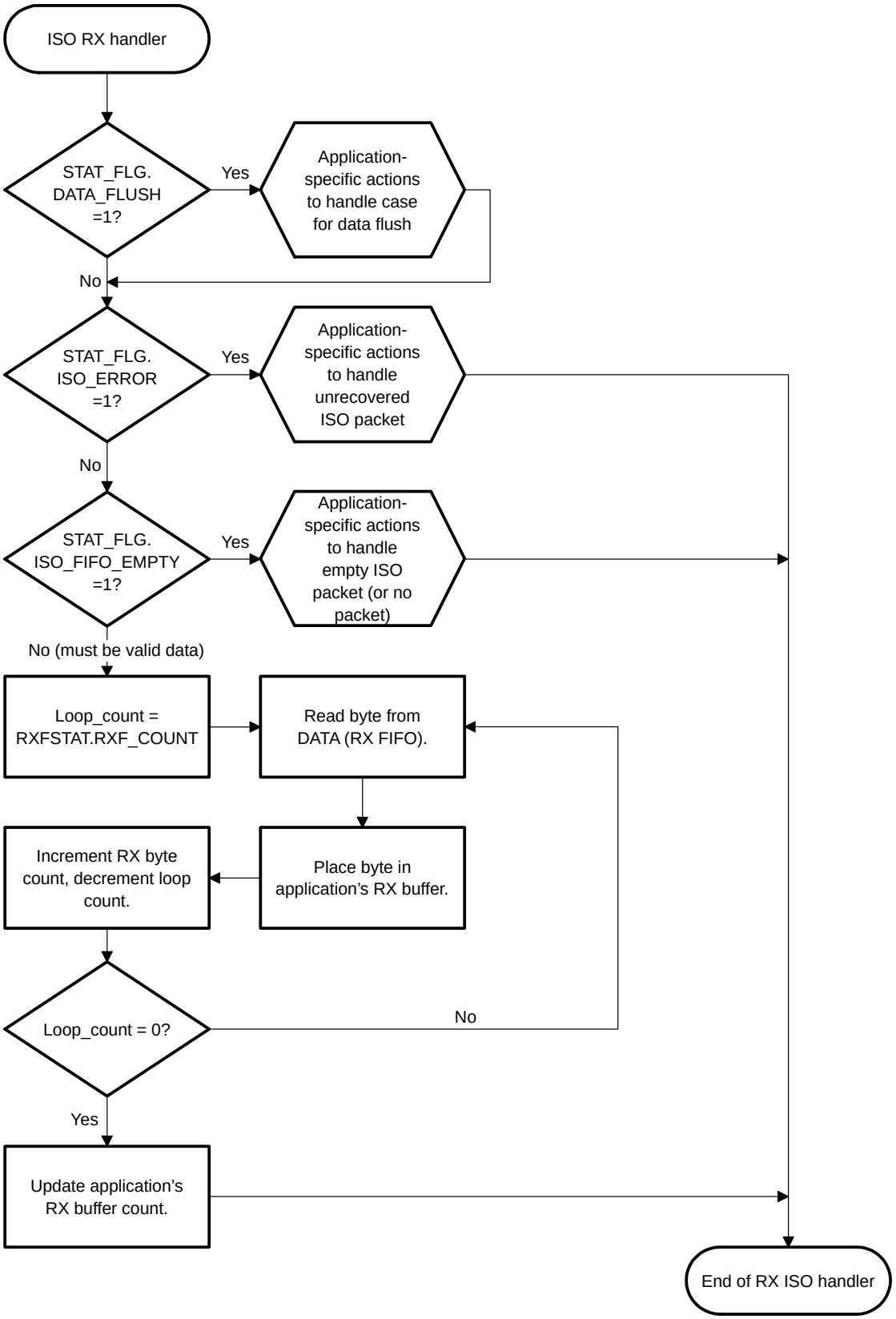
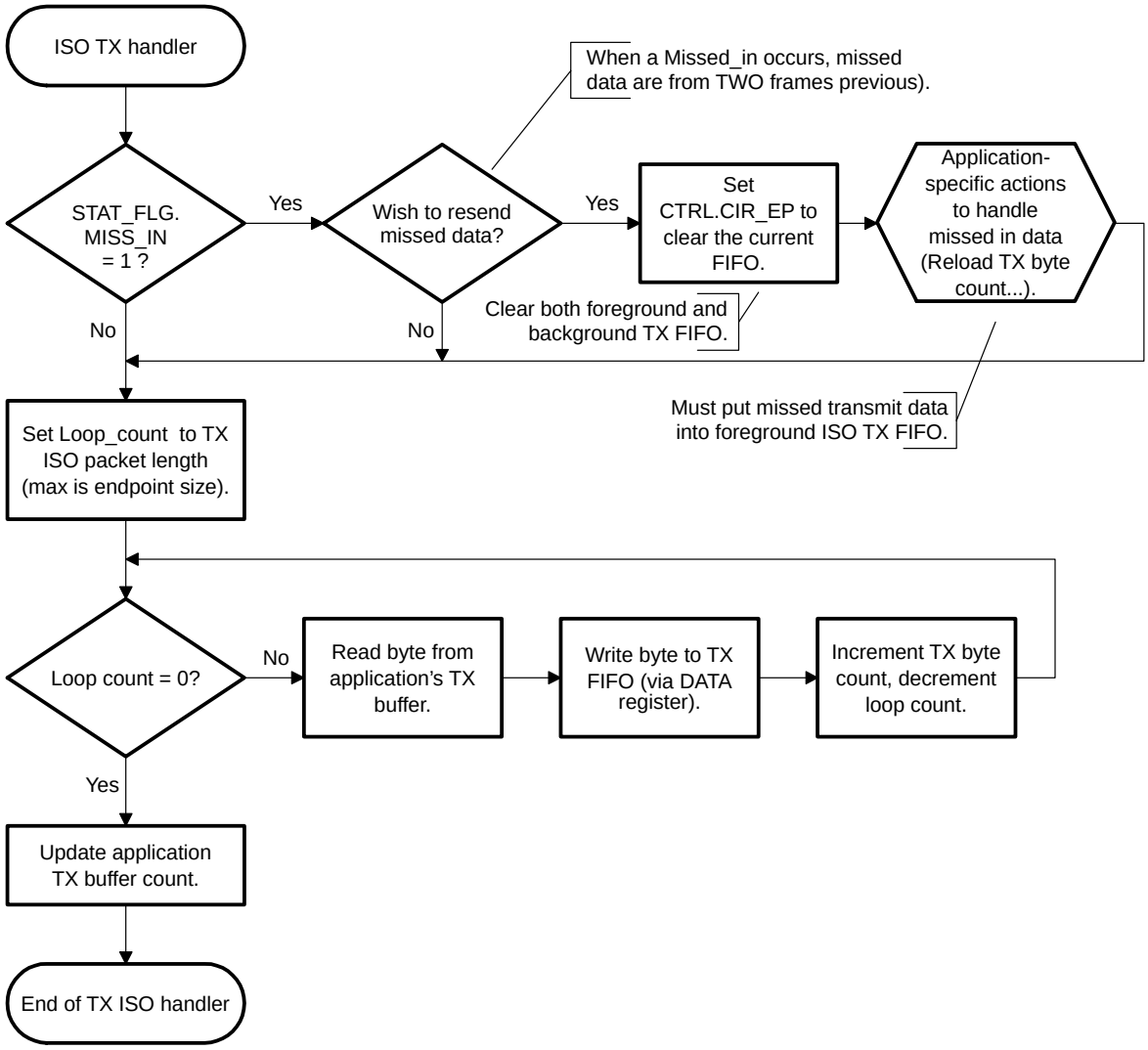


Figure 10–35. Write Isochronous TX FIFO Data Flowchart



10.3.26 Summary of USB Device Controller Interrupts

Table 10–54 lists the interrupt types by endpoint types.

Table 10–54. USB Device Controller Interrupt Type by Endpoint Type

Interrupt Type	General USB IRQs (MPU Level 2 IRQ_29)				EP-Specific IRQs (Non-ISO Interrupt on MPU Level 1 IRQ_2)		SOF ISO Interrupt on MPU Level 1 IRQ_3)
	Setup (EP0)	Control (EP0) Out	Control (EP0) In	Other	Bulk or Interrupt Out	Bulk or Interrupt In	(Isochronous) SOF
Transaction ACKed		X	X		X	X	
Transaction NAKed (if enabled)		X	X		X	X	
Transaction STALLed		X	X		X	X	
Setup	X						
SOF							X
Device state changed				X			
RX DMA EOT (non_ISO)				X			
RX DMA trans count (non_ISO)				X			
TX DMA done (non_ISO)				X			

10.3.26.1 USB Device Controller Clock Control

The OMAP730 clock generation and system reset management module(ULPD) provides a single 48-MHz clock to the USB OTG controller, USB device controller, and USB host controller. This clock can be stopped by software to reduce USB OTG controller, USB host, and USB device controller power consumption when USB functionality is not needed. The ULPD controls the 48-MHz clock to the USB device controller via:

- ❏ CLOCK_CTRL_REG.USB_MCLK_EN
- ❏ SOFT_REQ_REG.SOFT_USB_OTG_DPLL_REQ
- ❏ SOFT_DIS–ABLE_REQ_REG.DIS_USB_DEVICE_DPLL_REQ

See Chapter 17, *Power and Clock Control*.

The OTG module allows separate control of USB device controller clocking via OTG_SYSCON_1.DEV_IDLE_EN.

10.3.26.2 USB Device Controller Hardware Reset

Reset of the USB device controller is provided by the ULPD module. The PER_EN bit in the MPU reset control 2 register controls the reset to many OMAP730 peripherals, including the USB device controller. When held in reset, the USB device controller does not recognize any USB activity.

When the USB device controller is in reset, its registers have no effect on USB functionality. Software must wait until OTG_SYSCON_1.RESET_DONE is 1.

10.3.27 DMA Operation

The USB device controller provides support for six DMA channels. Three receive DMA channels are reserved to OUT transfers (ISO or non-ISO) and three transmit DMA channels are reserved to IN transfers (ISO or non-ISO). It is not possible to operate DMA transactions on control EP0.

The MPU must not access an endpoint used in a DMA transfer through the EP_NUM, CTRL, and STAT_FLG registers (in DMA, this remark applies after the MPU has set the CTRL.SET_FIFO_EN bit to enable the RX DMA transfer). In particular, the MPU must not set the halt feature while the endpoint is selected in the RXDMA_CFG register.

Note:

To use the DMA channels properly, you must set the DMA configuration during the address state interrupt (DS_CHANGE).

The parameters used for DMA transactions (FIFO size, ISO endpoint, double-buffering, and pointers) are those defined for the associated endpoint.

Receive DMA Channels Overview

Receive DMA channels are programmed via the three RXDMA control registers. Each channel is assigned to a given endpoint number by assigning a non-zero value in RXDMA_CFG.RXDMAn_EP fields (a 0 value means the DMA channel is deselected). Received OUT data must be read when an RX DMA request is active, through the register DATA_DMA. The RX FIFO accessed is that of the endpoint for which the DMA request is active (only one RX DMA request is active at a time).

The USB device controller transmit DMA channels 0 through 2 are connected to OMAP730 DMA controller requests DMA_REQ_26, DMA_REQ_27, and DMA_REQ_29, respectively.

Non-Isochronous OUT (USB HOST –> MPU) DMA Transactions

During non-ISO transfers to a DMA operated OUT endpoint, a request to the MPU DMA controller is generated when data have been placed into endpoint

FIFO and must be read. ACK and NAK interrupts are always disabled automatically by the core for DMA operated endpoints.

There are two dedicated maskable interrupts per DMA channel to control non-ISO OUT transfers.

End of Transfer Interrupt (IRQ_SRC.RXn_EOT)

This interrupt signals that the core has detected an end-of-transfer (EOT). EOT occurs in the two following cases:

- When the last valid transaction to the endpoint is either an empty packet (ACK and buffer empty) or a packet whose size is less than the physical endpoint buffer size (ACK and buffer not full)
- When the number of received transactions has reached the programmed value in the RXDMAn.RXn_TC field, if the RXDMAn.RXn_STOP bit has been set by the MPU

After an end of transfer interrupt, the MPU must set CTRL.SET_FIFO_EN for the endpoint to reenable the channel.

The MPU must not initiate a new RX DMA transfer until it receives an end-of-transfer interrupt.

Transaction Count Interrupt (IRQ_SRC.RXn_CNT)

The intent of this interrupt is watermark control. It can be used by the MPU to monitor the file size of incoming transfers and take appropriate actions if, for instance, the file being received exceeds an expected size.

A transaction count interrupt does not disable the ongoing DMA transfer.

A transaction count interrupt occurs each time the number of received transactions (and not bytes) has reached the programmed value in the receive transaction counter for the DMA channel. One transaction has a size equal to the buffer size of the selected non-ISO endpoint. RXn_COUNT interrupt is asserted even if RXDMAn.RXn_STOP has been set; in that case, both RXn_COUNT and RXn_EOT interrupts are asserted (see Figure 10–36 through Figure 10–39).

The transaction count watermark is programmed in the RXDMAn register.

Figure 10–36. Non-ISO RX DMA Transaction Example (RX_TC=2)

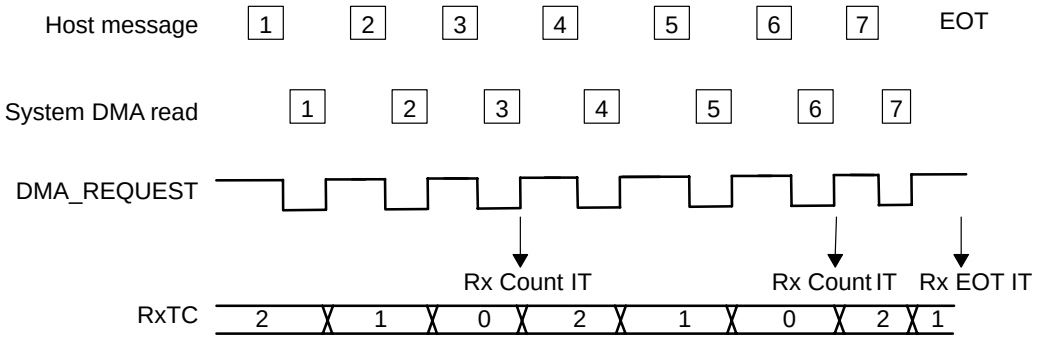


Figure 10–37. Non-ISO RX DMA Start Routine

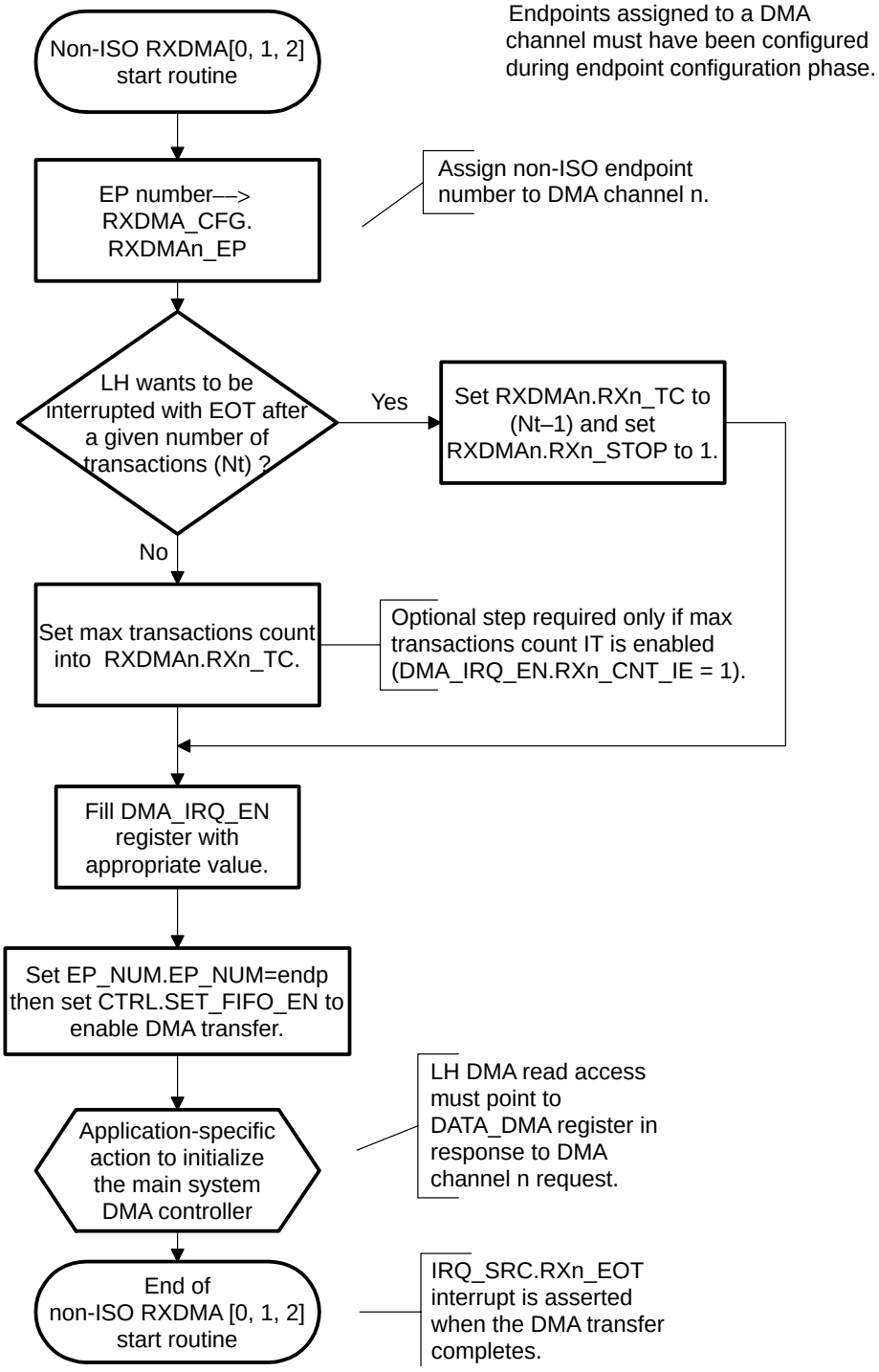


Figure 10–38. Non-ISO RX DMA EOT Interrupt Handler

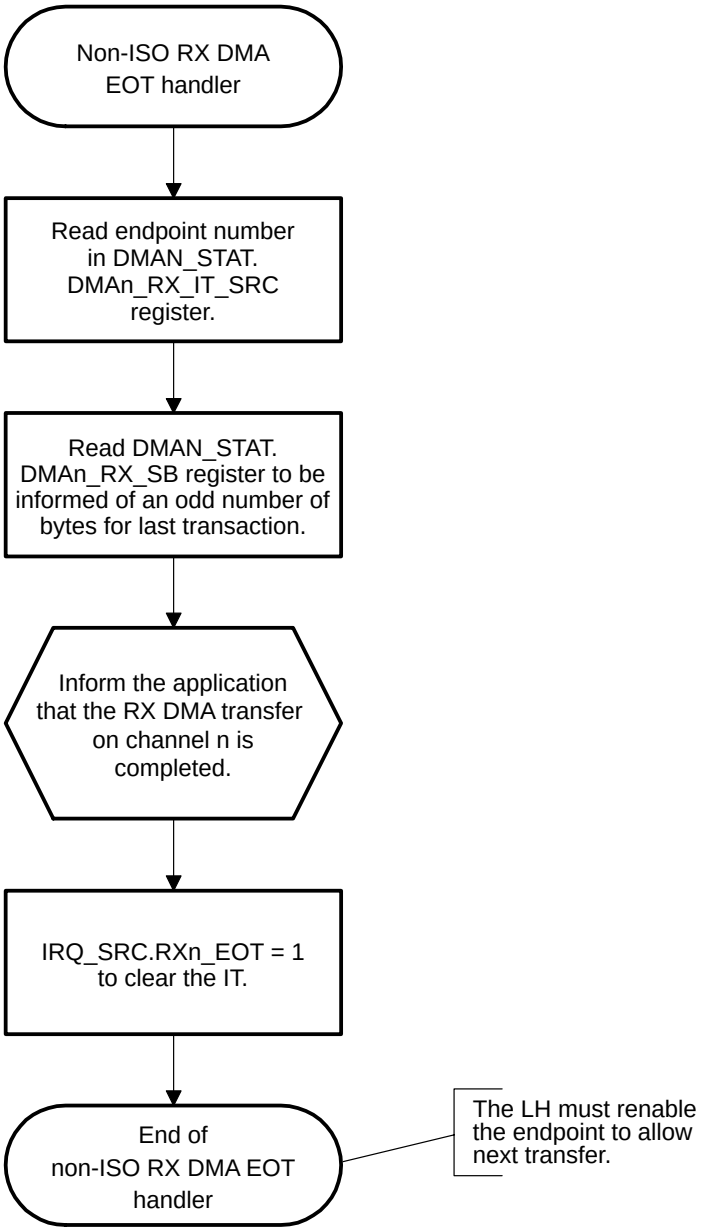
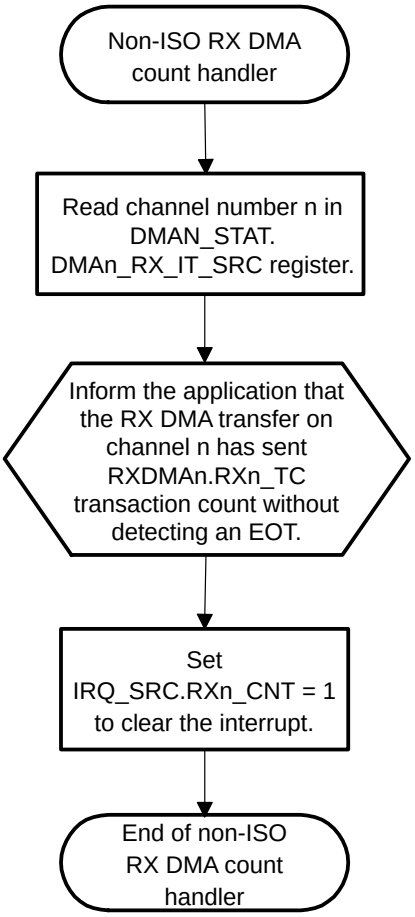


Figure 10–39. Non-ISO RX DMA Transactions Count Interrupt Handler



Isochronous OUT (USB HOST –> MPU) DMA Transactions

During ISO transfers to a DMA operated OUT endpoint, a request to the MPU DMA controller is generated every 1-ms frame when an isochronous data packet is received with no error. There is no interrupt associated with DMA transfer to ISO OUT endpoints (see Figure 10–40 and Figure 10–41).

Figure 10–40. ISO RX DMA Transaction

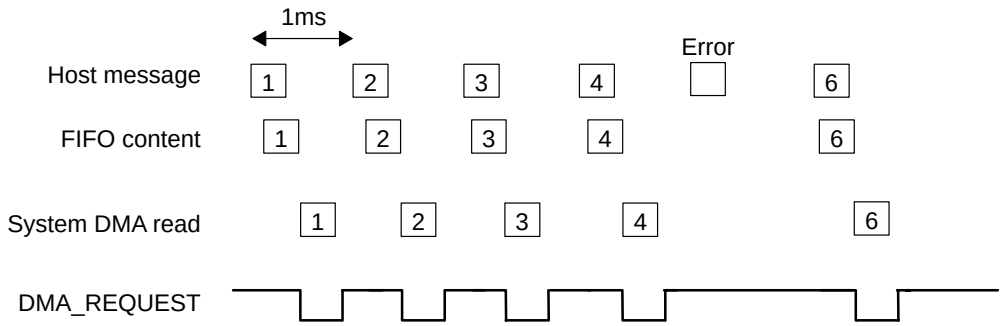
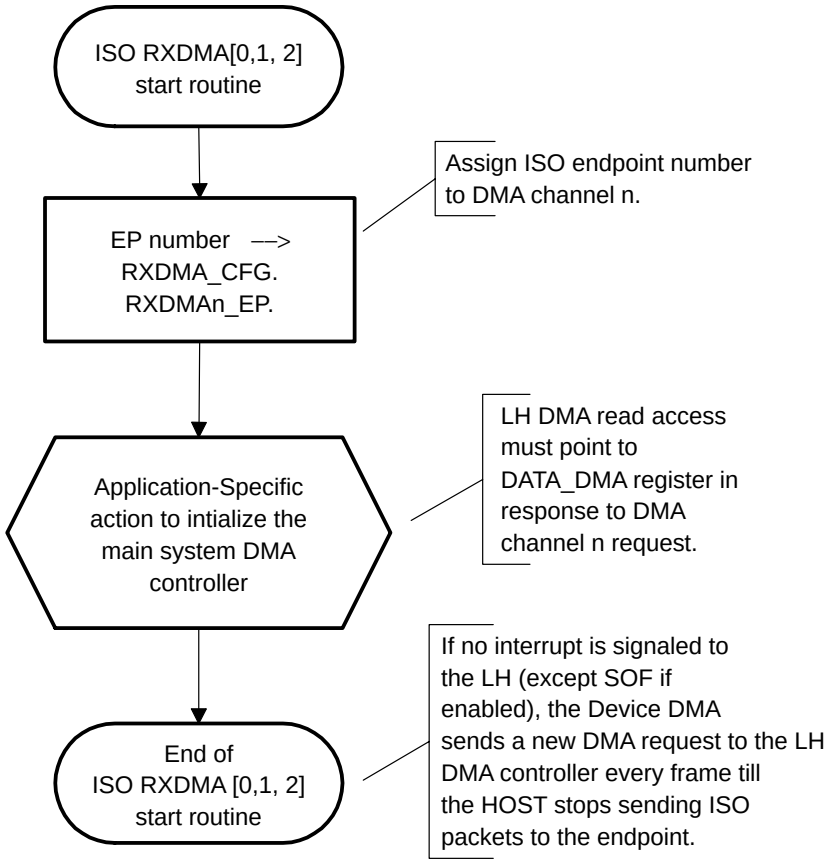


Figure 10–41. ISO RX DMA Start Routine



Transmit DMA Channels Overview

Transmit DMA channels are programmed via the three TXDMA control registers. Each channel can be assigned to a given endpoint number by assigning

a non-zero value in TXDMA_CFG.TXDMAn_EP (a 0 value means the DMA channel is deselected). The other three control registers (TXDMA0, TXDMA1, and TXDMA2) operate in a different manner for ISO and non-ISO endpoints. Transmitted data must be written into the DATA_DMA when a TX DMA request is active. They are written into the TX FIFO of the endpoint associated with active request (only one TX DMA request active at a given time).

The USB device controller transmit DMA channels 0 through 2 are connected to OMAP730 DMA controller requests DMA_REQ_29, DMA_REQ_30, and DMA_REQ_31, respectively.

Non-Isochronous IN (MPU –> USB HOST) DMA Transactions

Non-ISO (bulk) TX DMA file transfers are virtually unlimited in size. The flowcharts depicted in Figure 10–42 and Figure 10–43 show how to handle small, medium, or large file transfers.

TXDMA0, TXDMA1, and TXDMA2 registers operate for non-ISO endpoints in the following manner. The transfer size counter (TXDMAn.TXN_TSC) corresponds to either the number of bytes to transmit (EOT bit set) or the number of buffers to transmit (EOT bit cleared). The buffer size corresponds to the programmed size of the TX endpoint.

A request to the MPU main DMA controller is generated when the endpoint buffer is empty initially after that the START bit is set and then each time there is space free in TX FIFO for other TX packet to be written, until TXDMAn.TXn_TSC counts down to zero. The request is removed when the buffer is full or when there are no more bytes of data to be sent.

A DMA transmit transfer done interrupt is signaled to the MPU after the last IN transaction completes successfully. This is after START bit was set and after TXDMAn.TXn_TSC equals 0 for the selected DMA channel.

The MPU must not initiate a new TX DMA transfer until it receives a TX_DONE interrupt.

A small file transfer less than 1024 bytes can be achieved in a single pass DMA signaled by a single interrupt completion. A file size equal to or greater than 1024 bytes needs two or more DMA passes, signaled by an interrupt completion after each pass.

Figure 10–42 shows the necessary steps to prepare and permit a TX DMA transfer of any size. It also effectively starts the initial DMA transfer. The completion of this DMA task is signaled to the MPU via a DONE interrupt, whose handler is shown in Figure 10–43. The start routine and the associated interrupt handler are tightly coupled.

Figure 10–42. Non-ISO TX DMA Start Routine

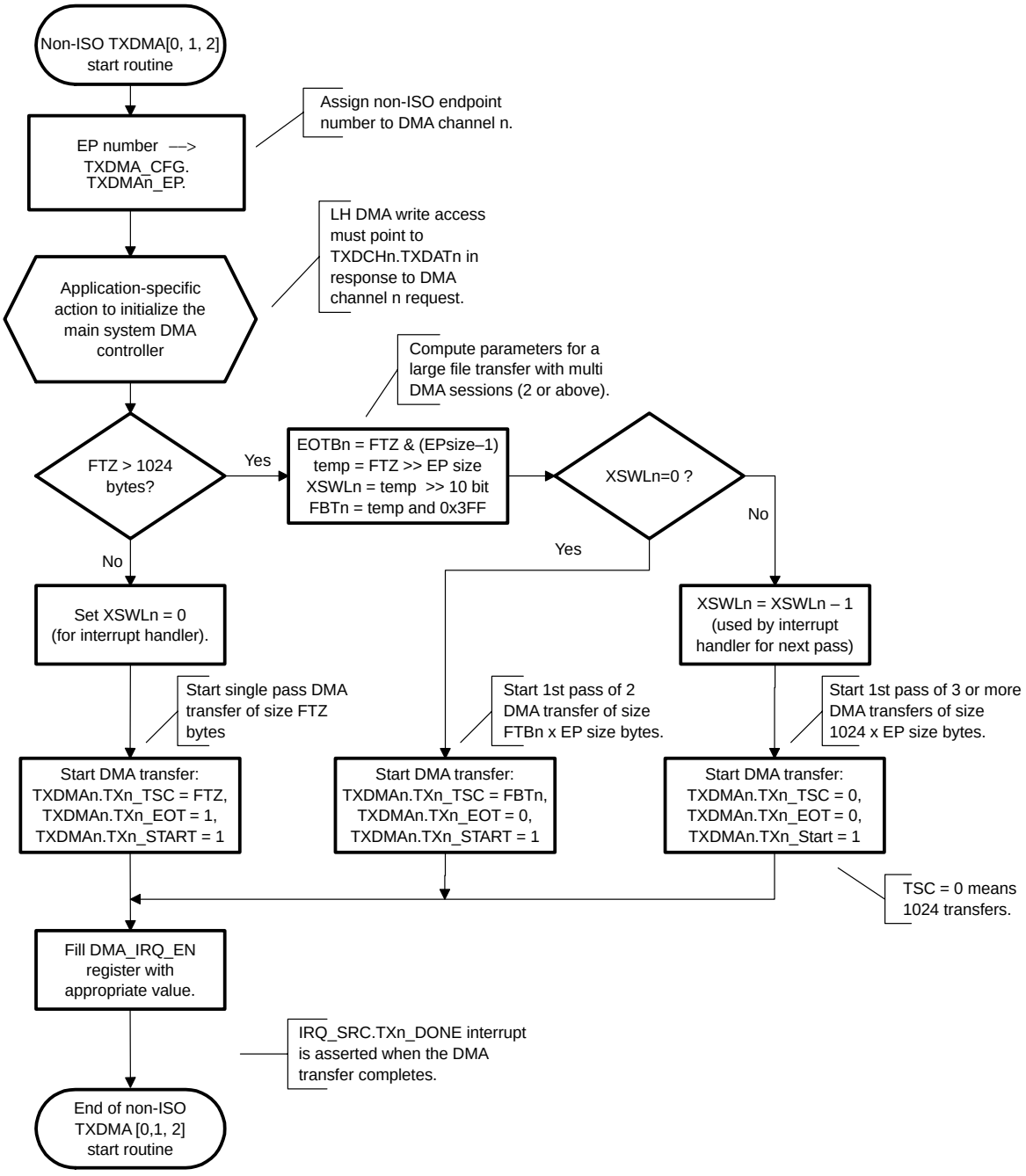
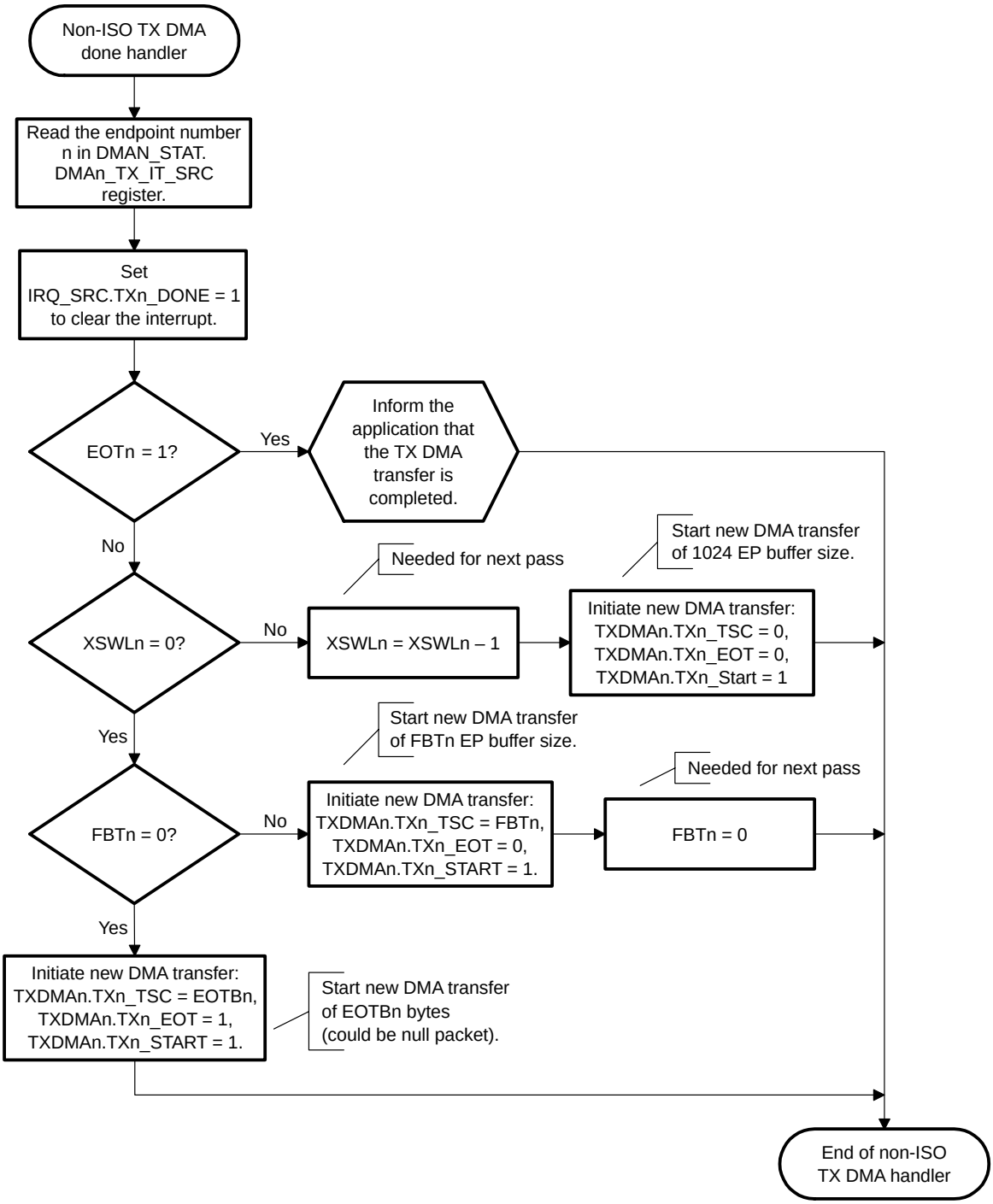


Figure 10–43. Non-ISO TX DMA Done Interrupt Handler



Example 10–1. Example: 100603 Bytes to Transfer via 32 Bytes IN Bulk Endpoint

This gives XSWL=0x3, FBT=0x47, EOTB=0x1b,

which means five passes of DMA transfer, signaled by 5 TXn_DONE interrupts, are required:

1)	EOT=0, FBT=0, loop=3	32768 bytes transferred (1024 x 32 bytes)
2)	EOT=0, FBT=0, loop=2	32768 bytes
3)	EOT=0, FBT=0, loop=1	32768 bytes
4)	EOT=0, FBT=0x47, loop=0	2272 bytes (71 x 32 bytes)
5)	EOT=1, FBT=0x1B, loop=0	27 bytes

Isochronous IN (USB HOST –> MPU) DMA Transactions

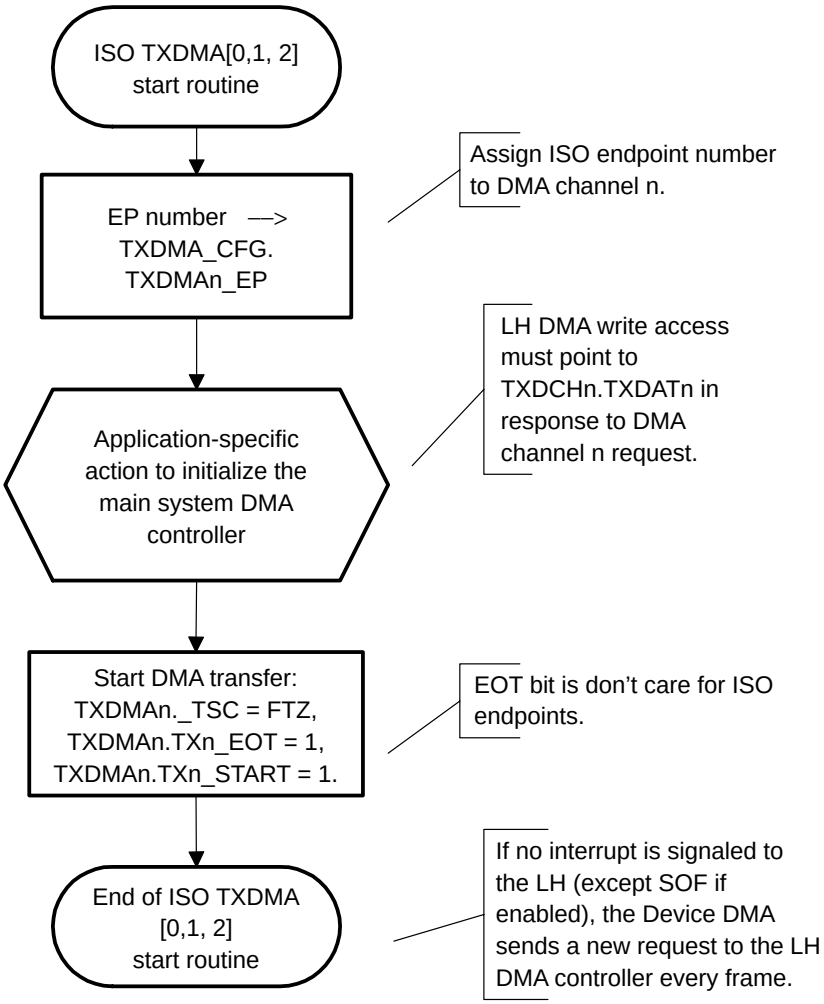
For ISO endpoints, the transfer size counter (TXDMAn.TXn_TSC) corresponds to the number of bytes to transmit. The programmed size must not exceed the programmed buffer size of the endpoint. Otherwise, the result is unpredictable (see Figure 10–44).

A request to the MPU main DMA controller is generated when the endpoint buffer is empty initially after the START bit is set, and then after each SOF (every 1 ms). The request is removed when the number of bytes written in the buffer matches the TXDMAn.TXn_TSC value.

During ISO transfers to a DMA operated IN endpoint, a request to the MPU system DMA controller is generated every 1-ms frame when an isochronous data packet is received with no error. There is no special interrupt associated with the DMA transfer.

No interrupt is signaled to the MPU during DMA operation to ISO IN endpoints.

Figure 10–44. ISO TX DMA Start Routine



Important Note on DMA Requests

For each direction, only one DMA request can be active at any time. A request must then be serviced to allow the next pending request on the same direction to be asserted. In particular, a TX DMA request is asserted at each start-of-frame if a TX DMA channel is configured for an isochronous endpoint; this request must be serviced imperatively.

Note on DMA Channels Deconfiguration

It is recommended that the MPU wait for an EOT (RX) or a DONE (TX) interrupt before disabling the channel by writing a value 0 in the TX/RXDMA_CFG register. However, if needed by the application, the MPU can unselect the endpoint number in the TX/RXDMA_CFG register during a DMA transfer. The resulting behavior is:

- For RX transfer:
  - If RX DMA request is active for the endpoint when the endpoint is unselected, deconfiguration is effective only at the end of the RX DMA

request (that is, after all the DMA data have been read). When double-buffering is used, the deconfiguration is effective after both buffers have been read (if both buffers were not empty at unselection). An EOT interrupt is asserted if an end-of-transfer is detected.

- If the RX DMA request is not active when unselection occurs, the effect is immediate.

❏ For TX transfer:

- If the request is active when the endpoint is unselected, deconfiguration is effective after the TX DMA request has been handled and the TX data have been sent through an IN transaction (both buffers in case of double-buffering with both buffers full). No TX_DONE interrupt is asserted even if TXDMAn.TSC bit value is 0 after the transaction.
- If the TX DMA request is inactive when the endpoint is unselected, deconfiguration is effective when all data available in TX buffer(s) have been sent through IN transaction(s). If the TXDMAn_TSC value is 0 at this point, no TX_DONE interrupt is asserted. If TX_DONE interrupt had already been asserted when the endpoint is deselected, configuration is effective only after the TX_DONE interrupt handling.

TX/RXDMA_CFG.TX/RXDMAn_EP reflects the endpoint value until deconfiguration is effective. The MPU must read this register to know if the channel has been disabled yet or not. It must wait until the read value is 0 before performing other actions to the endpoint. After effective deconfiguration, all transactions to the endpoint generate an endpoint-specific interrupt (if non-transparent).

If the selected endpoint is of isochronous type, deconfiguration is effective after the TX/RX request has been serviced, and the subsequent isochronous transactions are handled at SOF interrupt through the endpoint registers (EP_NUM and STAT_FLG).

10.3.28 Power Management

Figure 10–45 shows the values assigned to the USB device controller signals concerned with power management, in the function of the device state. These signals are:

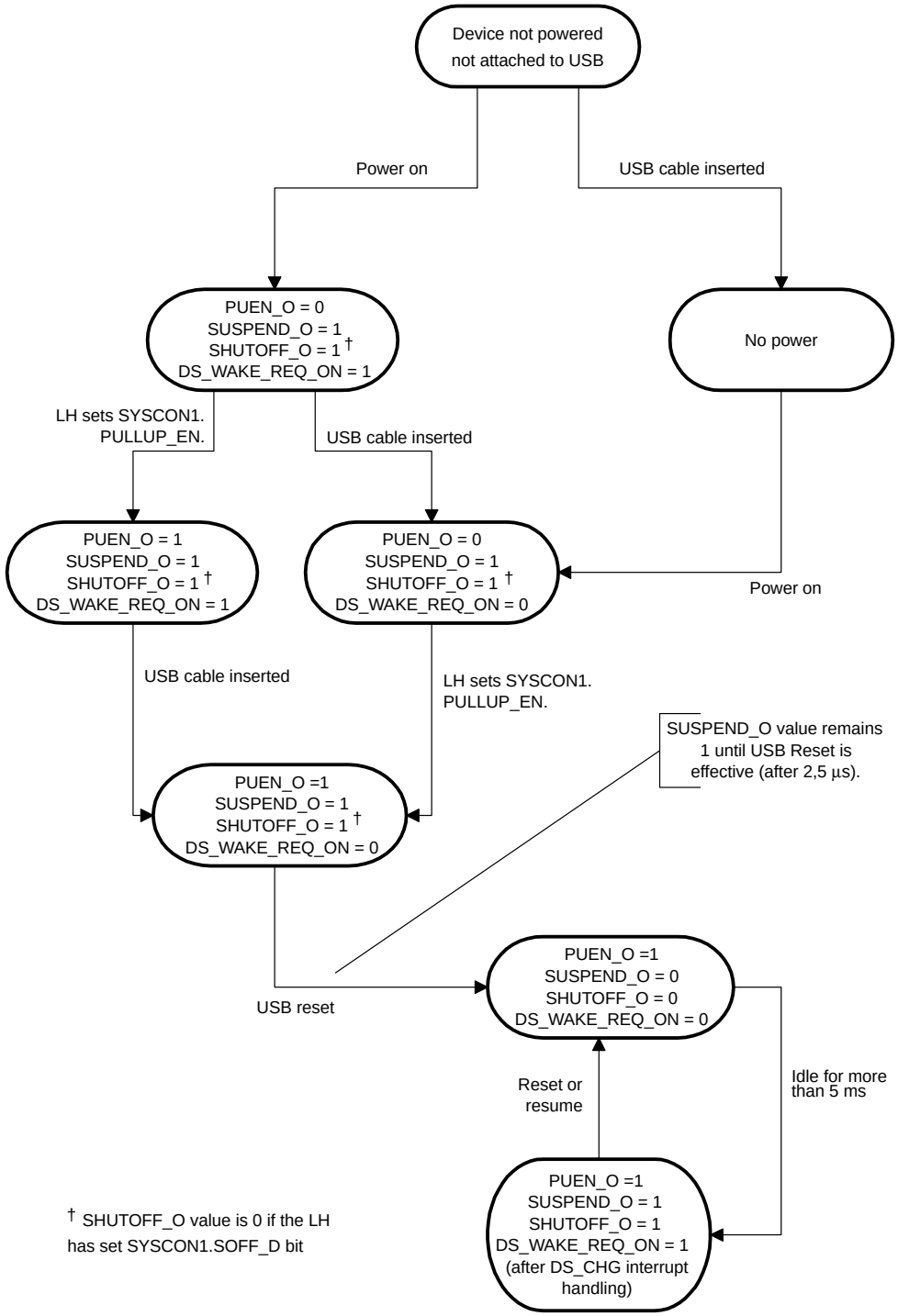
❏ PUEN_O: Pullup enable signal, always reflecting the SYSCON1.PULLUP_EN register bit

❏ SHUTOFF_O: Power circuitry shutoff signal, controlled by the core and the SYSCON1.SOFF_DIS bit

❏ DS_WAKE_REQ_ON: Deep-sleep wake request, asserted low when the interface clock is needed

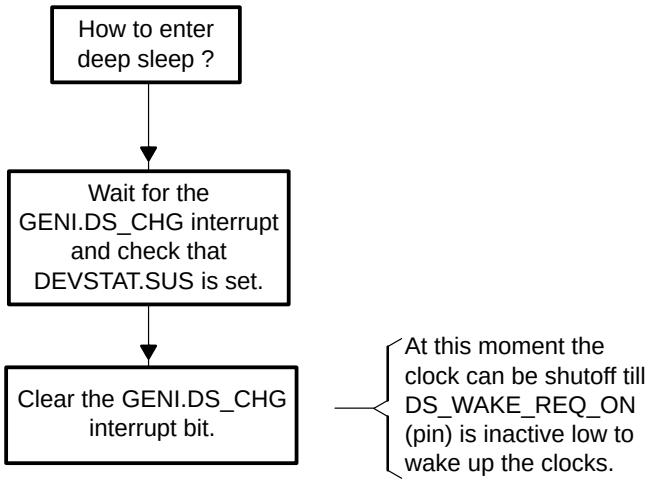
❏ SUSPEND_O: Suspend signal, asserted high when the device is in suspend mode

Figure 10–45. Power Management Signal Values



From a software point of view, Figure 10–46 shows the reaction. This flowchart does not need to be implemented; it only reflects the way the module can enter deep sleep.

Figure 10–46. Power Management Flowchart



10.4 USB OTG Controller

The OMAP730 USB OTG controller works in conjunction with the OMAP730 USB device controller and one port of the OMAP730 host controller to provide USB On-The-Go functionality. The OMAP730 OTG controller includes various control and status logic that switches between the two controllers as required by the USB On-The-Go protocol. The combination allows OMAP730 to act as a USB OTG dual-role device.

The USB OTG controller provides a dual-role device capability that is compatible with the On-The-Go supplement to the *USB 2.0 Specification* (Revision 1.0). This dual-role device can provide a 12M bit-per-second connection between OMAP730 and other USB OTG dual-role devices. The OMAP730 OTG solution does not support 48M bit-per-second OTG connectivity.

In addition, the USB OTG controller provides control and status information for various aspects of the OMAP730 USB host controller and the OMAP730 USB device controller operation when On-The-Go functionality is not enabled.

10.4.1 OTG Controller Features

The main features of the OMAP730 USB OTG controller include:

- ❏ USB specification version 2.0 compatibility
- ❏ When acting as an OTG a-peripheral or an OTG b-peripheral: a 12M bit-per-second communication link with configurable data transfer type, data buffer size, single- or double-buffering for each endpoint, and up to three IN endpoints using DMA and up to three OUT endpoints using DMA to support streaming USB data.
- ❏ When acting as an OTG a-host or an OTG b-host: an OHCI Rev 1.0-compatible USB host controller capable of USB full-speed (12M bits-per-second) and USB low-speed (1.5M bits-per-second) communication.

These are basic features of the OMAP730 USB device controller and the OMAP730 USB host controller. Additional OTG-specific features provided by the OMAP730 OTG controller include:

- ❏ Control and status logic that allows implementation of an OTG dual-role device (DRD)
- ❏ Ability to implement an OTG peripheral-only device (that is, one that cannot act as an OTG A-device)
- ❏ Support for generation and detection of the OTG host negation protocol (HNP) and both types of OTG session request protocol
- ❏ Support for OTG transceivers compatible with the OTG working group OTG transceiver specification

10.4.2 OTG Controller Registers

Table 10–55 lists the 32-bit OTG controller registers. Table 10–56 through Table 10–68 describe the register bits.

Table 10–55. OTG Controller Registers

Name	Description	R/W	Address
OTG_REV	OTG controller revision number	R/O	0400h
OTG_SYSCON_1	OTG system configuration group 1	R/W	0404h
OTG_SYSCON_2	OTG system configuration group 2	R/W	0408h
OTG_CTRL	OTG control	R/W	040Ch
OTG_IRQ_EN	OTG interrupt enable	R/W	0410h
OTG_IRQ_SRC	OTG interrupt source identification	R/W	0414h
Reserved	Reserved	None	0418h to 04FBh
VC	USB vendor code	R/O	04FCh

All bits defined as reserved must be written by software with 0s, for preserving future compatibility. When read, any reserved bit returns 0. It is good software practice to use complete mask patterns for setting or testing bit fields individually within a register.

Table 10–56. OTG Revision Number Register (OTG_REV)

Bit	Name	Description
31-8		
7-0	OTG_REV_NB	OTG revision number: this 8-bit field indicates the revision number of the current OTG controller, in binary-coded digital (BCD), with the major revision number in bits 7–4, and the minor revision number in bits 3–0. This value is hardware fixed. 0x10: Revision 1.0 0x11: Revision 1.1 0x12: Revision 1.2 0x13: Revision 1.3 0x14: Revision 1.4 0x15: Revision 1.5 0x16: Revision 1.6 .... 0xF2: Revision 15.2

OTG_REV is a read-only register that reflects the OTG controller revision number.

Table 10–57. OTG System Configuration Register 1 (OTG_SYSCON_1)

Bit	Name	Description
31:27	Reserved	Reserved
26:24	USB2_TRX_MODE†	USB port 2 transceiver mode. These bits configure the transceiver signaling type used by the USB host, USB device, and USB OTG controllers when communicating with the transceiver connected to USB pin group 2. Transceiver signaling type depends on the signaling mode used by the transceiver. The supported transceiver signaling types are 3-pin DAT/SE0 mode bidirectional transceiver signaling, 4-pin VP/VM mode bidirectional transceiver signaling, and 6-pin DAT/SE0 unidirectional transceiver signaling (see Section 10.4.5, *Transceiver Signaling Types*).
23	Reserved	Reserved
22:20	USB1_TRX_MODE‡	USB port 1 transceiver mode. These bits configure the transceiver signaling type used by the USB host, USB device, and USB OTG controllers when communicating with the transceiver connected to USB pin group 1. Transceiver signaling type depends on the signaling mode used by the transceiver. The supported transceiver signaling types are 3-pin DAT/SE0 mode bidirectional transceiver signaling, 4-pin VP/VM mode bidirectional transceiver signaling, and 6-pin DAT/SE0 unidirectional transceiver signaling (see Section 10.4.5, *Transceiver Signaling Types*).
19	Reserved	Reserved
18:16	USB0_TRX_MODE§	USB port 0 transceiver mode. These bits configure the transceiver signaling type used by the USB host, USB device, and USB OTG controllers when communicating with the transceiver connected to USB pin group 0 or USB alternate pin group 2. Transceiver signaling type depends on the signaling mode used by the transceiver. The supported transceiver signaling types are 3-pin DAT/SE0 mode bidirectional transceiver signaling, 4-pin VP/VM mode bidirectional transceiver signaling, and 6-pin DAT/SE0 unidirectional transceiver signaling (see Section 10.4.5, *Transceiver Signaling Types*).
15	OTG_IDLE_EN	OTG controller clock-gating control. This bit disables the clock to the OTG controller. 0: The OTG controller clock is not gated. Register settings in the ULPD can prevent the USB OTG controller from getting a clock even if this bit is 0. 1: The OTG controller is idled by disabling the OTG controller. Accesses to OTG controller registers are allowed, but the OTG controller clock is not gated. This register is cleared 0 by soft reset or hardware reset. For more information about the power saving circuitry, see Section 10.4.2.2.
14	Reserved	Reserved

† See Table 10–58, *Pin Group 2 Transceiver Type Selection*.
‡ See Table 10–59, *Pin Group 1 Transceiver Type Selection*.
§ See Table 10–60, *Pin Group 0 Transceiver Type Selection* and Table 10–61, *Alternate Pin Group 2 Transceiver Type Selection*

Table 10–57. OTG System Configuration Register 1 (OTG_SYSCON_1) (Continued)

Bit	Name	Description
13	DEV_IDLE_EN	Device controller clock-gating control. This bit defines the device power-saving circuitry. 0: The USB device controller clock is not gated. Other registers in OMAP730 can prevent the USB device controller from getting a clock even if this bit is 0. 1: The USB device controller clock is disabled if the USB device controller does not need an active clock (such as when the USB device controller is in suspense). Accesses to USB device controller registers are allowed, but the USB device controller state machines does not function. If the USB device controller sees USB resume signaled when DEV_IDLE_EN is 1, a USB device controller general interrupt is generated. The interrupt service routine must clear DEV_IDLE_EN before performing the operations necessary to service the interrupt. This register is set to1 by soft reset or hardware reset. For more information about the USB device controller, see Section 10.3, *USB Device Controller*.
12:3	Reserved	Reserved
2	RESET_DONE	Reset status information: This bit reflects the reset status of the controller (hardware and software reset, internal reset monitoring of the USB device and the USB OTG controller sections). 0: Internal reset for OTG is ongoing (includes OTG controller and USB device) 1: Reset completed This register is cleared 0 by soft reset or hardware reset. This reset done information is important for proper use of the controller. After a hardware reset or a soft reset, this bit must be checked to determine that the reset has completed.This register does not reflect the reset status of the OMAP730 USB host controller. Software must query certain USB host controller registers to determine when the USB host controller reset is complete. See Section 10.2.16,*USB Host Controller Hardware Reset*.

† See Table 10–58, *Pin Group 2 Transceiver Type Selection*.
‡ See Table 10–59, *Pin Group 1 Transceiver Type Selection*.
§ See Table 10–60, *Pin Group 0 Transceiver Type Selection* and Table 10–61, *Alternate Pin Group 2 Transceiver Type Selection*

Table 10–57. OTG System Configuration Register 1 (OTG_SYSCON_1) (Continued)

Bit	Name	Description
1	SOFT_RESET	Software reset for the OTG, device, and host controllers: set this bit to 1 to trigger a reset of the OTG controller, USB host controller, and USB device controller. The bit is automatically cleared by the hardware. During reads, it always returns 0. 0: Normal mode 1: The controller is reset. Software must monitor the RESET_DONE to determine when the OTG controller and the USB device controller have completed reset. Section 10.2.16 *USB Host Controller Hardware Reset* describes software activities required to determine when the USB host controller completes its reset.
0	Reserved	–

† See Table 10–58, *Pin Group 2 Transceiver Type Selection*.
‡ See Table 10–59, *Pin Group 1 Transceiver Type Selection*.
§ See Table 10–60, *Pin Group 0 Transceiver Type Selection* and Table 10–61, *Alternate Pin Group 2 Transceiver Type Selection*

OTG_SYSCON_1 is a read-write register that controls the USB transceiver interface, the USB controller clock gating, and USB controller reset generation.

Table 10–58. Pin Group 2 Transceiver Type Selection

USB2_TRX_MODE	OMAP730 Pin Group 2 Transceiver Signaling
0x0	6-pin DAT/SE0 mode unidirectional signaling Normal functionality is available if USB_TRANSCEIVER_CTRL.CONF_USB2_UNI_R = 1. If USB_TRANSCEIVER_CTRL.CONF_USB2_UNI_R = 0, OMAP730 places pins USB2.TXD and USB2_TXSE0 in high-impedance mode.
0x1	4-pin VP/VM mode bidirectional signaling
0x2	3-pin DAT/SE0 mode bidirectional signaling
0x3	6-pin DAT/SE0 mode unidirectional signaling
Other values	Reserved

USB2_TRX_MODE is cleared to 0x0 by software reset or hardware reset.

When both

- ❏ USB2_TRX_MODE = 0
- ❏ USB_TRANSCEIVER_CTRL.CONF_USB2_UNI_R = 0

OMAP730 places pins USB2.TXD and USB2.TXSE0 in high-impedance mode to prevent contention with any signals driven by the external transceiver. Mode 0 can be used for normal 6-pin DAT/SE0 mode functionality when USB_TRANSCEIVER_CTRL.CONF_USB2_UNI_R = 1.

This register does not have an effect on the transceiver signaling type when top-level pin multiplexing selects alternate pin group 2 (see the USB0_TRX_MODE subsection). When HMC_MODE selects a transceiver-

less link logic connection using USB pin group 2, this register must be set to 3.

Table 10–59. Pin Group 1 Transceiver Type Selection

USB1_TRX_MODE	OMAP730 Pin Group 1 Transceiver Signaling
0x0	6-pin DAT/SE0 mode unidirectional signaling. Normal functionality is available if USB_TRANSCEIVER_CTRL.CONF_USB1_UNI_R = 1. If USB_TRANSCEIVER_CTRL.CONF_USB1_UNI_R = 0, OMAP730 places pins USB1.TXD and USB1_TXSE0 in high-impedance mode.
0x1	4-pin VP/VM mode bidirectional signaling
0x2	3-pin DAT/SE0 mode bidirectional signaling
0x3	6-pin DAT/SE0 mode unidirectional signaling
Other values	Reserved

USB1_TRX_MODE is cleared to 0x0 by soft reset or hardware reset.

When both:

- ❑ USB1_TRX_MODE = 0
- ❑ USB_TRANSCEIVER_CTRL.CONF_USB1_UNI_R = 0

OMAP730 places pins USB1.TXD and USB1_TXSE0 in high-impedance mode to prevent contention with any signals driven by the external transceiver. Mode 0 can be used for normal 6-pin DAT/SE0 mode functionality when USB_TRANSCEIVER_CTRL.CONF_USB1_UNI_R = 1.

Table 10–60. Pin Group 0 Transceiver Type Selection

USB0_TRX_MODE	OMAP730 Pin Group 0 Transceiver Signaling
0x0	6-pin DAT/SE0 mode unidirectional signaling
0x1	4-pin VP/VM mode bidirectional signaling
0x2	3-pin DAT/SE0 mode bidirectional signaling
0x3	6-pin DAT/SE0 mode unidirectional signaling
Other values	Reserved

USB0_TRX_MODE is cleared to 0x0 by soft reset or hardware reset.

When using the OMAP730 integrated USB transceiver, USB0_TRX_MODE is set to 3. Software can avoid USB0_TRX_MODE = 0 to improve software compatibility with future devices.

When OMAP730 top-level signal multiplexing selects alternate pin group 2, the USB0_TRX_MODE signal affects the transceiver type for the transceiver connected to the following OMAP730 pins:

- MCSI2.DOUT/USB0.TXEN
- UART2.TX/USB0.TXD
- UART1.RTS/USB0.SE0
- UART2.CTS/USB0.RCV
- MCSI2.DIN/USB0.VP
- UART2.RX/USB0.VM

Table 10–61. Alternate Pin Group 2 Transceiver Type Selection

USB0_TRX_MODE	OMAP730 Alternate Pin Group 2 Transceiver Signaling
0x0	6-pin DAT/SE0 mode unidirectional signaling.
0x1	4-pin VP/VM mode bidirectional signaling
0x2	3-pin DAT/SE0 mode bidirectional signaling
0x3	6-pin DAT/SE0 mode unidirectional signaling
Other values	Reserved

Table 10–62. OTG System Configuration Register 2 (OTG_SYSCON_2)

Bit	Name	Description
31	OTG_EN	The OTG enable bit selects the OTG controller functionality: 0: The OTG controller circuitry is not activated. The USB device controller and the USB host controller can be used, but an OTG link cannot be implemented. 1: The OTG controller circuitry is activated. The OTG state machine (HNP) can be used on one of the USB ports. The USB device controller and one of the USB host controller ports can be used to implement an OTG link. This register is cleared 0 by soft reset or hardware reset. When an OTG link is not needed, this bit can be cleared and OTG_ SYSCON_1.OTG_IDLE_EN can be set to reduce power consumption. A driver switch interrupt is internally generated whenever a 1 is written to OTG_EN. If OTG_IRQ_EN.DRIVER_SWITCH_EN is 1, this internal interrupt propagates to the MPU level 2 interrupt controller.
30	USBx_SYNCHRO	The USB output signals synchronized bit must be set to 1 to allow proper signal timing on OMAP730 USB pins. This register is cleared 0 by soft reset or hardware reset.
29	OTG_MST16	This bit enables compatibility with the 16-bit OTG master controller, and it must be set to 0 to allow proper USB host controller access to OMAP730 system memory. 0: Host controller is connected to system memory via a 32-bit bus. 1: Host controller is connected to system memory via a 16-bit bus. OMAP730 does not support this setting. This register is cleared to 0 by soft reset or hardware reset.

† See Table 10–63, *OTG_PADEN Source Status*.
‡ See Table 10–64, *HMC_PADEN: USB Signal Multiplexing Control Source*.

Table 10–62. OTG System Configuration Register 2 (OTG_SYSCON_2) (Continued)

Bit	Name	Description
28	SRP_GPDATA	Session request protocol generation pulse width on D+. This bit allows the OTG controller to extend the duration of the D+ pulse during the data line pulsing portion of a session request protocol activity. This bit is only used when the OTG controller is configured to act as an OTG default B device. 0: Data line pulse during SRP is 6 ms (nominal). 1: Data line pulse during SRP is 9 ms (nominal). This register is cleared 0 by soft reset or hardware reset. When the OTG controller performs the SRP process, it first generates a data line pulse SRP request and then a VBUS pulse SRP request.
27	SRP_GPDVBUS	Session request protocol generation discharge VBUS enable. This bit enables the OTG controller to provide a VBUS discharge sequence as part of its SRP generation activities. This bit enables the VBUS discharge functionality after any VBUS charge process. When enabled, the duration of the VBUS discharge is defined by SRP_GPUVBUS. 0: VBUS is not discharged after the SRP VBUS pulse. 1: VBUS is discharged after the SRP VBUS pulse. This register is cleared 0 by soft reset or hardware reset.

† See Table 10–63, *OTG_PADEN Source Status*.
‡ See Table 10–64, *HMC_PADEN: USB Signal Multiplexing Control Source*.

Table 10–62. OTG System Configuration Register 2 (OTG_SYSCON_2) (Continued)

Bit	Name	Description
26:24	SRP_GPUVBUS	Session request protocol generation charge VBUS duration. This bit controls the duration that the OTG controller attempts to charge VBUS when acting as a default-B OTG device and requesting SRP. 0x0: OMAP730 does not issue a VBUS charge pulse as part of its SRP generation sequence. 0x1: VBUS is charged for 0.5 ms nominal. 0x2: During SRP generation, VBUS is charged for .5 ms nominal. 0x3: During SRP generation, VBUS is charged for 2 ms nominal. 0x4: During SRP generation, VBUS is charged for 4 ms nominal. 0x5: During SRP generation, VBUS is charged for 6 ms nominal. 0x6: During SRP generation, VBUS is charged for 10 ms nominal. 0x7: During SRP generation, VBUS is charged for 40 ms nominal. This register is cleared 000 by soft reset or hardware reset. The current sourcing capabilities of the hardware that drives the SRP VBUS pulse determines the required duration of the VBUS pulse. The value chosen must ensure that VBUS rises higher than 2 V within the selected VBUS pulse duration when attached to an OTG dual-role device. The value chosen must also ensure that VBUS does not rise higher than 2 V within the selected VBUS pulse duration when attached to a non-OTG port, such as a USB host port or a downstream port of a USB hub. SRP_GPUVBUS also controls the duration of a VBUS discharge pulse when SRP_GPDBVBUS is set to 1. If SRP_GPUVBUS is 0x0, the OTG controller does not request a VBUS discharge. The minimum time for SRP signalling completion is (the duration of the data line pulse defined by SRP_GPDATA) + (2 * VBUS pulse time defined by SRP_GPUVBUS). Disabling the VBUS discharge does not affect that minimum time. The interrupt OTG_IRQ_SRC:B_SRP_DONE is generated upon completion of the entire SRP.
23	Reserved	Reserved

† See Table 10–63, *OTG_PADEN Source Status*.
‡ See Table 10–64, *HMC_PADEN: USB Signal Multiplexing Control Source*.

Table 10–62. OTG System Configuration Register 2 (OTG_SYSCON_2) (Continued)

Bit	Name	Description
22:20	A_WAIT_VRISE	Offset for A_WAIT_VRISE timer. These bits define the maximum duration that the OTG controller must wait for VBUS to rise above the A-device VBUS valid threshold. A_WAIT_VRISE is used only when OMAP730 acts as a default-A device. The A_WAIT_VRISE counter begins counting upon transition from the A_IDLE state. (See the *On-The-Go Supplement to the USB 2.0 Specification Revision 1.0* description of the dual-role A-device state diagram). 0x0: Maximum delay in A_WAIT_VRISE state is 200 ms nominal. 0x1: Maximum delay in A_WAIT_VRISE state is 287.04 ms nominal. 0x2: Maximum delay in A_WAIT_VRISE state is 374.08 ms nominal. 0x3: Maximum delay in A_WAIT_VRISE state is 548.16 ms nominal. This register is cleared 0x0 by soft reset or hardware reset. If the A_WAIT_VRISE counter expires before VBUS rises above the A-device VBUS valid threshold, it indicates that there is a heavy load on VBUS. The OTG controller state machine transitions to A_VBUS_ERR and generates a VBUS error interrupt.
19	Reserved	Reserved
18:16	B_ASE0_BRST	Offset for B_ASE0_BRST timer. These bits control how the USB OTG controller manages disabling the D+ pullup when acting as a default-B OTG dual-role device and transitioning from state b_peripheral to b_wait_acon, and when acting as a default-A OTG dual-role device and transitioning from state a_peripheral to a_wait_bcon. The setting of this register controls the mechanism by which the OMAP730 OTG controller knows that the D+ pullup is disabled. Because OMAP730 implementations control the D+ pullup using an I^2C link to an OTG transceiver, the only allowed setting is 0x4. When in this mode of operation, system software must write a 1 to OTG_CTRL.OTG_PU immediately after completion of the I^2C operation that disables the D+ pullup. This field is cleared 0x0 by soft reset or hardware reset. This field has no effect when OTG is disabled. This field must not be changed when OTG functionality is enabled (OTG_SYSCON_2.OTG_EN=1).
15	Reserved	Reserved
14	SRP_DPW	Session request protocol detection—pulse width. This bit selects the width for the SRP detection when acting as a default-A OTG dual-role device. This value applies on both DATA and VBUS pulsing detections. 0: SRP pulse must be greater than 1 ms (nominal) to be sensed as a valid D+, D–, or VBUS SRP pulse. 1: SRP pulse must be greater than 167 ns (nominal) to be sensed as a valid D+, D–, or VBUS SRP pulse. This register is cleared 0 by soft reset or hardware reset. This bit has no effect when the OTG feature is disabled (OTG_SYSCON_2.OTG_EN = 0).

† See Table 10–63, *OTG_PADEN Source Status*.
‡ See Table 10–64, *HMC_PADEN: USB Signal Multiplexing Control Source*.

Table 10–62. OTG System Configuration Register 2 (OTG_SYSCON_2) (Continued)

Bit	Name	Description
13	SRP_DATA	Session request protocol detection—data pulsing detection enable. This bit determines whether the OMAP730 OTG controller considers D+ or D– pulses as SRP requests. 0: OTG controller does not consider D+ or D– pulses as SRP requests. 1: OTG controller treats a D+ or D– pulse of appropriate minimum duration as an SRP request. This register is cleared 0 by soft reset or hardware reset. This register is used only when the OMAP730 OTG controller acts as a default-A dual-role device. SRP detection requires that either OTG_SYSCON_2.SRP_DATA or OTG_SYSCON_2.SRP_VBUS (or both) is set.
12	SRP_VBUS	Session request protocol detection—VBUS pulsing detection enable. This bit determines whether the OMAP730 OTG controller considers VBUS pulses as SRP requests. 0: OTG controller does not consider VBUS pulses as SRP requests. 1: OTG controller treats a VBUS pulse of appropriate minimum duration as an SRP request. This register is cleared 0 by soft reset or hardware reset. This register is only used when the OMAP730 OTG controller acts as a default-A dual-role-device. SRP detection requires that either OTG_SYSCON_2.SRP_DATA or OTG_SYSCON_2.SRP_VBUS (or both) is set.
11	Reserved	Reserved
10	OTG_PADEN†	OTG transceiver control and status information selector. This bit determines how OTG_CTRL register bits ID, VBUSVLD, BSESSVLD, BSESSEND, and ASESSEND are controlled. When using OMAP730 OTG controller functionality to implement an On-The-Go dual-role device, this bit must be set to 0. In this case, those OTG_CTRL bits are read/write bits and are controlled by system software. When using OMAP730 USB device controller functionality without enabling OTG functionality, this bit must be set to 1. In this case, OTG_CTRL register bits ID, VBUSVLD, BSESSEND, and ASESSEND are held at 0, and OTG_CTRL bit BSESSVLD is controlled by the OMAP730 GPIO0/USB.VBUS pin (if top-level pin multiplexing selects the USB_VBUS mode for pin GPIO0). Software writes to those OTG_CTRL bits are ignored when OTG_PADEN = 1.
9	HMC_PADEN‡	USB pin multiplexing control selector. This bit determines whether USB signal multiplexing is controlled by registers in OTG_SYSCON_2 or by registers in the OMAP730 configuration register module. 0 = OTG_SYSCON_2 provides the controls. 1 = OMAP730 configuration register module registers provide the controls.

† See Table 10–63, *OTG_PADEN Source Status*.
‡ See Table 10–64, *HMC_PADEN: USB Signal Multiplexing Control Source*.

Table 10–62. OTG System Configuration Register 2 (OTG_SYSCON_2) (Continued)

Bit	Name	Description
8	UHOST_EN	The host USB controller enable bit controls the clock enable and hardware reset for the OMAP730 USB host controller when HMC_PADEN is 0. When HMC_PADEN is 1, writes to this bit have no effect on the clock enable. 0: USB host controller is not clocked and is held in hardware reset. 1: USB host controller clock is not held inactive and a USB host controller hardware reset is not forced. The USB host controller is clocked if the ULPD 48-MHz clock output to the USB controllers is active and if the OMAP730 peripheral reset is inactive. This register is cleared 1 by soft reset or hardware reset. A read of this register gives the internal value used for UHOST_EN regardless of the setting of HMC_PADEN. See Section 15.1.20 USB host controller hardware reset for information on USB host controller access restrictions relating to UHOST_EN.
7	HMC_TLLSPEED	The HMC TLL SPEED configuration bit controls the way that the OMAP730 transceiverless link logic models the speed of the transceiverless link when HMC_PADEN is 0. When HMC_PADEN is 1, this bit has no effect on the transceiverless link logic and writes to this bit have no effect. 0: Transceiverless link logic models a low-speed (1.5M bits-per-second) USB link. 1: Transceiverless link logic models a full-speed (12M bits-per-second) USB link. This register is cleared 0 by soft reset or hardware reset. Proper operation of the transceiverless link logic requires that the USB host controller is clocked when the transceiverless link logic is used, even if the transceiverless link logic is not connected to a USB host controller port.
6	HMC_TLLATTACH	The HMC TLL ATTACH configuration bit controls the OMAP730 transceiverless link logic attach/detach status when HMC_PADEN is 0. When HMC_PADEN is 1, this bit has no effect on the transceiverless link logic and writes to this bit have no effect. 0: Transceiverless link logic models a detached link. 1: Transceiverless link logic models an attached link. This register is cleared 0 by soft reset or hardware reset. Proper operation of the transceiverless link logic requires that the USB host controller is clocked when the transceiverless link logic is used, even if the transceiverless link logic is not connected to a USB host controller port.
5:0	HMC_MODE	The HMC mode configuration bits control the OMAP USB signal multiplexing selection when HMC_PADEN is 0. When HMC_PADEN is 1 this field is unused and writes to this register have no effect. HMC_MODE values are discussed in Section 15.2.1, *Pin Multiplexing*. This register is cleared 0x0 by soft reset or hardware reset.

† See Table 10–63, *OTG_PADEN Source Status*.
‡ See Table 10–64, *HMC_PADEN: USB Signal Multiplexing Control Source*.

OTG_SYSCON_2 is a read-write register that provides control and status for various aspects of OTG controller, USB pin multiplexing, and USB host controller functionality.

Table 10–63. OTG_PADEN Source Status

OTG_PADEN	Value	Source
0	ID	OTG_CTRL.ID
	VBUSVLD	OTG_CTRL.VBUSVLD
	BSESSEND	OTG_CTRL.BSESSEND
	ASESSEND	OTG_CTRL.ASESSEND
	BSESSVLD	OTG_CTRL.BSESSVLD
	VBUS value to USB device controller	
1	ID	Tied to 0
	VBUSVLD	Tied to 0
	BSESSEND	Tied to 0
	ASESSEND	Tied to 0
	BSESSVLD	Follows GPIO0/USB.VBUS if top-level pin multiplexing selects the USB.VBUS mode for pin GPIO_0. Is tied to 0 if top-level pin multiplexing selects a mode other than USB.VBUS for pin GPIO0.
	VBUS value to USB device controller	BSESSVLD

OTG_PADEN is cleared 0 by soft reset or hardware reset.

Table 10–64. HMC_PADEN: USB Signal Multiplexing Control Source

HMC_PADEN	Value	Source
0	HMC_MODE	OTG_SYSCON_2.HMC_MODE
	HMC_TLLATTACH	OTG_SYSCON_2.HMC_TLLATTACH
	HMC_TLLSPEED	OTG_SYSCON_2.HMC_TLLSPEED
	UHOST_EN	OTG_SYSCON_2.UHOST_EN
1	HMC_MODE	MOD_CONF_CTRL_0. CONF_MOD_USB_HOST_HHC_HMC_MODE_R
	HMC_TLLATTACH	MOD_CONF_CTRL_0. CONF_MOD_USB_HOST_HHC_HMC_TLL_ATTACH_R
	HMC_TLLSPEED	MOD_CONF_CTRL_0. CONF_MOD_USB_HOST_HHC_HMC_TLL_SPEED_R
	UHOST_EN	MOD_CONF_CTRL_0.CONF_MOD_USB_HOST_HHC_UHOST_ EN_R

HMC_PADEN is cleared 0 by soft reset or hardware reset.

For best software compatibility with future devices, it is recommended that system software set this register to 0.

Table 10–65. OTG Control Register (OTG_CTRL)

Bit	Name	Description
31:21	Reserved	Reserved
20	ASESSVLD	Current A-device session valid value. When OTG is enabled (OTG_EN = 1), this bit must be programmed to reflect the OTG transceiver VBUS voltage comparator that compares against $V_{A\_SESS\_VLD}$. When VBUS is above $V_{A\_SESS\_VLD}$, ASESSVLD must be set to 1. When VBUS is below $V_{A\_SESS\_VLD}$, ASESSVLD should be set to 0. This information is only relevant when connected as an A-device (OTG_CTRL.ID = 0) and OTG is enabled (OTG_EN = 1). 0: VBUS voltage is below $V_{A\_SESS\_VLD}$. 1: VBUS voltage is above $V_{A\_SESS\_VLD}$. This register is cleared 0x0 by soft reset or hardware reset.
19	BSESSEND	Current B-device session end value. When OTG is enabled (OTG_EN = 1), this bit must be programmed to reflect the OTG transceiver VBUS voltage comparator that compares against $V_{B\_SESS\_END}$. When VBUS is below $V_{B\_SESS\_END}$, BSESSEND must be set to 1. When VBUS is above $V_{B\_SESS\_END}$, BSESSEND must be set to 0. This information is only relevant when connected as B-device (OTG_CTRL.ID = 1) and OTG is enabled (OTG_EN = 1). 0: VBUS voltage is above $V_{B\_SESS\_END}$. 1: VBUS voltage is below $V_{B\_SESS\_END}$. This register is cleared 0x0 by soft reset or hardware reset.
18	BSESSVLD	Current B-device session valid value. When OTG is enabled (OTG_EN = 1) and OMAP730 is connected as a dual-role B-device (OTG_CTRL.ID = 1), this bit must be programmed to reflect the OTG transceiver VBUS voltage comparator that compares against $V_{B\_SESS\_VLD}$. When VBUS is above $V_{B\_SESS\_VLD}$, BSESSVLD must be set to 1. When VBUS is below $V_{B\_SESS\_VLD}$, BSESSVLD must be set to 0. 0: VBUS voltage is below $V_{B\_SESS\_VLD}$. 1: VBUS voltage is above $V_{B\_SESS\_VLD}$. This register is cleared 0x0 by soft reset or hardware reset. When OTG is disabled, OTG_PADEN is 0, and the USB device controller is being used, this bit must be programmed to reflect whether VBUS is high enough to allow the OMAP730 USB device controller to function properly. 0: VBUS voltage is below $V_{A\_VBUS\_VLD}$ (if OTG is enabled), or VBUS is insufficient to allow OMAP730 USB device controller functionality (if OTG is disabled and OTG_PADEN is 0). 1: VBUS voltage is above $V_{A\_VBUS\_VLD}$ (if OTG is enabled), or VBUS is sufficient to allow OMAP730 USB device controller functionality (if OTG is disabled and OTG_PADEN is 0).

Table 10–65. OTG Control Register (OTG_CTRL) (Continued)

Bit	Name	Description
17	VBUSVLD	Current VBUS valid value. When OTG is enabled (OTG_EN = 1), this bit must be programmed to reflect the OTG transceiver VBUS voltage comparator that compares against $V_{A\_VBUS\_VLD}$. When VBUS is above $V_{A\_VBUS\_VLD}$, VBUSVLD must be set to 1. When VBUS is below $V_{A\_VBUS\_VLD}$, VBUSVLD must be set to 0. This bit has no meaning when acting as an OTG default-B device (OTG ID = 1 and OTG_EN = 1). This register is cleared 0x0 by soft reset or hardware reset.
16	ID	Current ID pin value. When OTG is enabled (OTG_EN = 1), system software must program this bit to reflect the OTG transceiver ID pin sensor status any time ID changes. When OTG is disabled (OTG_EN = 0) this bit has no meaning. 0: ID pin is grounded. 1: ID pin is high-impedance. This register is cleared 0x0 by soft reset or hardware reset. The OTG controller does not understand resistive ID connectivity as can be found in a car kit environment. When an OTG transceiver sees ID pin resistively tied, OTG is not possible and OTG_EN must not be set.
15	DRIVER_SEL	Active controller/driver software. When OTG is enabled (OTG_EN = 1), this read-only bit determines which OMAP730 USB controller (and therefore software driver) has ownership of the OTG link. When HNP occurs, the OTG controller updates this bit and issues a driver change interrupt. When OTG is disabled (OTG_EN = 0), this bit has no meaning. When a write changes OTG_EN from 0 to 1, this bit is updated to reflect the value of OTG_CTRL.ID. 0: Host driver has control of the OTG link. 1: Device driver has control of the OTG link. This bit is set to 1 by soft reset or hardware reset.
14:13	Reserved	Reserved
12	A_SETB_HNPEN	B-device HNP indication (when acting as default-A device). When OTG is enabled (OTG_EN = 1) and OMAP730 acts as a default-A device (OTG_CTRL.ID=0), this bit must be programmed to reflect whether or not the B-device has been enabled to issue HNP. This bit has no effect when OMAP730 acts as a default-B device (OTG_CTRL.ID=1) or when OTG is disabled (OTG_EN = 0). 0: The B-device has not been enabled to issue HNP. The OMAP730 OTG controller does not respond to HNP issued by the B-device. 1: The B-device has been enabled to issue HNP. The OMAP730 OTG controller responds to HNP issued by the B-device. This bit is cleared 0 by soft reset or hardware reset.

Table 10–65. OTG Control Register (OTG_CTRL) (Continued)

Bit	Name	Description
11	A_BUSREQ	Bus request (when acting as default-A device). When OTG is enabled (OTG_EN = 1) and OMAP730 is acting as a default-A device (OTG_CTRL.ID = 0), system software must set this bit when it wishes to begin an OTG session. When acting as the OTG A-device, system software can suspend the appropriate USB host controller port and then clear this bit to allow HNP requests from the B-device. If no HNP occurs, system software can set this bit to resume ownership of the OTG link by the OMAP730 USB host controller. This bit has no effect if OTG_EN = 0 or if OTG_CTRL.ID = 1. 0: A-device does not request host role on the bus. Depending on OTG state machine state, either the session can end or the default-B device can issue an HNP. 1: A-device requests host role on the bus. OTG state machine begins a session (if the OTG session is ended) or attempts to return to ownership of the OTG link to the OMAP730 USB host controller as soon as possible. This bit is cleared 0 by soft reset, hardware reset, or when the A_REQ_TMROUT interrupt is set.
10	Reserved	Reserved
9	B_HNPEN	B-device HNP enable (when acting as default-B device). When OTG is enabled (OTG_EN = 1) and OMAP730 is acting as a default-B device (OTG_CTRL.ID = 1), system software must set this bit when the USB device receives a set feature operation with feature selector B_HNP_ENABLE. System software clears this bit whenever it sends USB reset to the default–B device. This bit has no meaning when OTG_EN = 0 or when OTG_CTRL.ID = 0. 0: B-device is not HNP enabled. The OTG controller does not issue HNP. 1: B-device is HNP enabled. The OTG controller can issue HNP when B_BUSREQ is set. This bit is cleared 0 by soft reset or hardware reset.
8	B_BUSREQ	Bus request (when acting as default-B device). When OTG is enabled (OTG_EN = 1) and OMAP730 is acting as a default-B device (OTG_CTRL.ID = 1), system software sets this bit when the application wishes to take ownership of the OTG link and begin acting as a host. When set, this allows the OTG controller to issue SRP if a session is not currently valid, and to begin HNP at the next available opportunity if HNP is enabled (OTG_CTRL.B_HNPEN = 1). This bit is ignored by the OTG controller when OTG_CTRL.ID = 0. 0: System software does not currently wish to begin acting as host. 1: System software (acting as a default-B device) wishes to begin acting as host. HNP and, if necessary, SRP are issued. This bit is cleared 0 by soft reset or hardware reset. The OTG controller clears this bit if there is not a valid session and the SRP fails. The OTG controller clears this bit if there is a valid session but the HNP fails. If SRP or HNP fails, system software repeats the request several times. If the SRP or HNP request fails several times, system software must inform the user of the failure.

Table 10–65. OTG Control Register (OTG_CTRL) (Continued)

Bit	Name	Description
7	OTG_BUSDROP	OTG bus drop request. When OTG is enabled (OTG_EN = 1) and OMAP730 is acting as a default-A device (OTG_CTRL.ID = 0), system software requests the end of a session by setting this bit. This bit has no effect when OTG_CTRL.ID = 1. 0: System software does not require the end of the OTG session. 1: System software requires that the OTG session be ended. This bit is cleared 0 by soft reset or hardware reset.
6	Reserved	Reserved
5	OTG_PD	D+ pulldown enable. When OTG is enabled (OTG_EN = 1), this read-only register indicates whether the OTG transceiver applies a pulldown to D+. When OTG is disabled, this bit has no meaning. 0: OTG transceiver does not activate the D+ pulldown. 1: OTG transceiver activates the D+ pulldown. This bit is cleared 0 by soft reset or hardware reset. This register can be polled when the OTG_IRQ_EN:OPRT_CHG_EN = 0. When OTG_IRQ_EN.OPRT_CHG_EN = 1 and OTG_IRQ_SRC:OPRT_CHG = 1, OTG_PD retains its value until after OTG_IRQ_SRC.OPRT_CHG is cleared.
4	OTG_PU	D+ pullup enable. When OTG is enabled (OTG_EN = 1), this register indicates whether the OTG transceiver applies a pullup to D+. When OTG is disabled, this bit has no meaning. 0: OTG transceiver disables the D+ pullup. 1: OTG transceiver activates the D+ pullup. This bit is cleared 0 by soft reset or hardware reset. This bit can be polled when the OTG_IRQ_EN:OPRT_CHG_EN = 0. When OTG_IRQ_EN.OPRT_CHG_EN = 1 and OTG_IRQ_SRC:OPRT_CHG =1, OTG_PU retains its value until after OTG_IRQ_SRC.OPRT_CHG is cleared. When OTG_SYSCON_2.B_ASE0_RST = 4 and acting as default-B device (OTG_CTRL.ID = 1), and system software is performing HNP, the software must write a 1 to this register when it knows that the OTG controller D+ pullup is active (typically upon completion of I^2C activity).

Table 10–65. OTG Control Register (OTG_CTRL) (Continued)

Bit	Name	Description
3	OTG_DRV_VBUS	VBUS drive enable. When OTG is enabled (OTG_EN = 1), this read-only bit indicates whether the OTG transceiver drives VBUS. When OTG is disabled (OTG_EN = 0), this bit has no meaning. Driving VBUS is requested only when OMAP730 acts as a default-A device and an OTG session is needed. 0: OTG transceiver does not drive VBUS. 1: OTG transceiver drives VBUS. This bit is cleared 0 by soft reset or hardware reset. This bit can be polled when the OTG_IRQ_EN:OPRT_CHG_EN = 0. When OTG_IRQ_EN.OPRT_CHG_EN = 1 and OTG_IRQ_SRC:OPRT_CHG = 1, OTG_DRV_VBUS retains its value until after OTG_IRQ_SRC.OPRT_CHG is cleared.
2	OTG_PD_VBUS	VBUS pulldown enable. When OTG is enabled (OTG_EN = 1), this read-only bit indicates whether the OTG transceiver is to discharge VBUS when generating SRP. Driving VBUS is requested only when OMAP730 acts as a default-A device and an OTG session is needed or is in progress. When OTG is disabled (OTG_EN = 0), this bit has no meaning. Discharge of VBUS is requested only when OMAP730 is acting as a default-B device, an OTG session needs to be requested, and OTG_SYSCON_2.SRP_GPDVBUS = 1. 0: OTG transceiver does not discharge VBUS. 1: OTG transceiver discharges VBUS. This bit is cleared 0 by soft reset or hardware reset. This bit can be polled when the OTG_IRQ_EN:OPRT_CHG_EN = 0. When OTG_IRQ_EN.OPRT_CHG_EN = 1 and OTG_IRQ_SRC:OPRT_CHG =1, OTG_PD_VBUS retains its value until after OTG_IRQ_SRC.OPRT_CHG is cleared.

Table 10–65. OTG Control Register (OTG_CTRL) (Continued)

Bit	Name	Description
1	OTG_PU_VBUS	VBUS pullup enable. When OTG is enabled (OTG_EN = 1), this read-only bit indicates whether the OTG transceiver charges VBUS. Charging VBUS is only used when OMAP730 acts as a default-B device and is performing SRP. When OTG is disabled (OTG_EN = 0), this bit has no meaning. 0: OTG transceiver does not charge VBUS. 1: OTG transceiver charges VBUS. This bit is cleared 0 by soft reset or hardware reset. This bit can be polled when the OTG_IRQ_EN:OPRT_CHG_EN = 0. When OTG_IRQ_EN.OPRT_CHG_EN = 1 and OTG_IRQ_SRC:OPRT_CHG =1, OTG_DRV_VBUS retains its value until after OTG_IRQ_SRC.OPRT_CHG is cleared.
0	OTG_PU_ID	ID signal pullup enable. When OTG is enabled (OTG_EN = 1), this read-only bit indicates whether the OTG transceiver applies a pullup to the ID pin to assist in detection of the ID pin level. System software for implementations using typical OTG transceivers with I^2C interfaces ignore this bit. 0: OTG transceiver does not apply a pullup to ID. 1: OTG transceiver applies a pullup to ID. This bit is cleared 0 by soft reset or hardware reset. This register can be used for polling when the OTG_IRQ_EN:OPRT_CHG_EN is cleared 0. Otherwise, this information is frozen when the interrupt status OTG_IRQ_SRC:OPRT_CHG is active 1.

OTG_CTRL is a read/write register that reflects the status of some USB OTG control bits. System software uses these register bits to convey OTG transceiver control and status between the transceiver and the OTG controller.

The bits OTG_PU_ID, OTG_PU_VBUS, OTG_PD_VBUS, OTG_DRV_VBUS, OTG_PU, and OTG_PD are controlled by the OTG controller and determine how software configures the OTG transceiver. Whenever one of these bits is changed by the OTG controller, an OPRT_CHG interrupt is generated (if enabled). It is best for system software to implement an interrupt handler that updates the OTG transceiver control registers when the OPRT_CHG interrupt occurs.

The bits ID, VBUSVLD, BSESSVLD, BSESSEND, and ASESSVLD must be updated to reflect the OTG transceiver status. Typically, the OTG transceiver provides an interrupt that can be configured to assert when the OTG transceiver status changes. This interrupt output is generally connected to an OMAP730 GPIO input that is configured as an MPU interrupt source. The interrupt service routine for the transceiver interrupt must query the OTG transceiver interrupt and appropriate status bits via I^2C operations and update the status bits in OTG_CTRL[31:27].

The bits OTG_BUSDROP, B_BUSREQ, B_HNPEN, A_BUSREQ, and A_SETB_HNPEN control the OTG controller state machines and provide functionality defined in the OTG supplement. System software must manage these bits as described below.

The DRIVER_SEL bit shows whether the USB host controller or the USB device controller has control of the OTG link.

Table 10–66. OTG Interrupt Enable Register (OTG_IRQ_EN)

Bit	Name	Description
15	DRIVER_SWITCH_EN	Driver switch interrupt enable. This bit enables interrupt generation when the OTG_IRQ_SRC: DRIVER_SWITCH bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC: DRIVER_SWITCH bit is set. This bit is cleared to 0 by soft reset or hardware reset.
14	Reserved	Reserved
13	A_VBUS_ERR_EN	A-device Vbus error interrupt enable: This bit enables interrupt generation when the OTG_IRQ_SRC: A_VBUS_ERr bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC: A_VBUS_ERR bit is set. This bit is cleared to 0 by soft reset or hardware reset.
12	A_REQ_TMROUT_EN	A-device request time-out interrupt enable. This bit enables generation of an interrupt when the OTG_IRQ_SRC: A_REQ_TMROUT bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC: A_REQ_TMROUT bit is set. This bit is cleared 0 by soft reset or hardware reset.
11	A_SRP_DETECT_EN	A-device SRP detection interrupt enable. This bit enables interrupt generation when the OTG_IRQ_SRC: A_SRP_DETECT bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC: A_SRP_DETECT bit is set. This bit is cleared to 0 by soft reset or hardware reset.
10	B_HNP_FAIL_EN	B-device HNP failed interrupt enable. This bit enables interrupt generation when the OTG_IRQ_SRC:B_HNP_FAIL bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC:B_HNP_FAIL bit is set. This bit is cleared to 0 by soft reset or hardware reset.

Table 10–66. OTG Interrupt Enable Register (OTG_IRQ_EN) (Continued)

Bit	Name	Description
9	B_SRP_TMROUT_EN	B-device SRP time-out interrupt enable. This bit enables interrupt generation when the OTG_IRQ_SRC:B_SRP_TMROUT bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC:B_SRP_TMROUT bit is set. This bit is cleared to 0 by soft reset or hardware reset.
8	B_SRP_DONE_EN	B-device SRP done interrupt enable. This bit enables interrupt generation when the OTG_IRQ_SRC:B_SRP_DONE bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC:B_SRP_DONE bit is set. This bit is cleared 0 by soft reset or hardware reset.
7	B_SRP_STARTED_EN	B-device SRP started interrupt enable. This bit enables interrupt generation when the OTG_IRQ_SRC:B_SRP_STARTED bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC:B_SRP_STARTED bit is set. This bit is cleared 0 by soft reset or hardware reset.
6:1	Reserved	Reserved
0	OPRT_CHG_EN	OTG output port status change interrupt enable: this bit enables interrupt generation when the OTG_IRQ_SRC:OPRT_CHG bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC:OPRT_CHG bit is set. This bit is cleared 0 by soft reset or hardware reset.

OTG_IRQ_EN is a read/write register that controls which OTG controller interrupts are passed to the MPU level 2 interrupt controller IRQ_8.

Table 10–67. OTG Interrupt Status Register (OTG_IRQ_SRC)

Bit	Name	Description
15	DRIVER_SWITCH	Driver switch interrupt status. This bit reflects the status of the driver switch interrupt source. Driver switch interrupts occur when HNP completes and control of the OTG link must switch between the OMAP730 USB host controller driver and the OMAP730 USB device controller driver (or vice versa).This interrupt also occurs upon enabling the OTG functionality by writing OTG_SYSCON_2.OTG_EN = 1. In this case, OGT_CTRL.DRIVER_SEL selects the controller driver that is appropriate, given the value of OTG_CTRL.ID. 0: Driver switch interrupt is inactive. 1: Driver switch interrupt is active. This bit is cleared either by writing a 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.
14	Reserved	Reserved
13	A_VBUS_ERR	A-device Vbus error interrupt status. This bit reflects the status of the A_VBUS_ERR interrupt. VBUS errors occur when the OMAP730 OTG controller state machine acts as a default-A dual-role device and transitions into state A_VBUS_ERR. Usually, the state machine transitions into state A_VBUS_ERR if VBUS voltage drops below the A_VBUS_VALID threshold because of low battery conditions or heavy loading by the attached device. Transitions into this state can unintentionally occur if system software turns off VBUS power and writes 0 to OTG_CTRL.VBUSVLD before writing 1 to OTG_CTRL.OTG _BUS_DROP. Writing 1 to OTG_CTRL.OTG_BUS_DROP clears the source of this interrupt. 0: VBUS error interrupt is inactive. 1:VBUS error interrupt is active. This bit is cleared either by writing a 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.
12	A_REQ_TMROUT	A-device request time-out interrupt status. This bit reflects the status of the A_REQ_TMROUT interrupt. This interrupt occurs when the OMAP730 OTG controller state machine acts as a default-A dual-role device and transitions from state a_wait_bcon to a_wait_vfall because the state machine did not see an attach. This interrupt can occur if system software attempts to power up an OTG link when the cable is not attached at both ends, or if the device at the far end of the cable does not provide a pullup resistor. 0: A-device request time-out interrupt is inactive. 1: A-device request time-out interrupt is active. This bit is cleared either by writing a 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.

Table 10–67. OTG Interrupt Status Register (OTG_IRQ_SRC) (Continued)

Bit	Name	Description
11	A_SRP_DETECT	A-device SRP detection interrupt status. This bit reflects the status of SRP detection when the OMAP730 OTG controller acts as an OTG default-A device. When the OTG controller sees a valid and enabled SRP method, this interrupt is generated. SRP is not detected if OTG_CTRL.OTG_BUS_DROP is 1 or if OTG_CTRL.A_BUSREQ is active. 0: SRP detection interrupt is inactive. 1: An SRP has been detected, using the VBUS pulsing and/or data pulsing detection mechanism. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.
10	B_HNP_FAIL	B-device HNP failed interrupt status. This bit reflects interrupts that are generated when the OMAP730 OTG controller acts as an OTG default-B device and an HNP it attempts to issue fails. This interrupt is generated when the default-A device does not enable its D+ pullup in response to the HNP request within the allotted time. This interrupt is also generated if the OMAP730 OTG controller acting as a default-B device issues HNP, but the default-A device issues a USB resume. A third instance where this interrupt can be generated is if OMAP730 acting as a default-B device issues an HNP request, but VBUS fails before completion of the HNP. 0: B-device HNP failure interrupt is inactive. 1: B-device HNP failure interrupt is active. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.
9	B_SRP_TMROUT	B-device SRP time-out interrupt status. This bit reflects interrupts that are generated when the OMAP730 OTG controller acts as an OTG default-B device and issues an SRP request, but the default-A device does not power VBUS within 5.5 seconds of the beginning of the SRP request. When this interrupt occurs, system software must inform the user that something is wrong (such as bad or unconnected cabling). 0: SRP time-out interrupt is inactive. 1: SRP time-out interrupt is active. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.The SRP timer starts when the B-device bus request is activated.
8	B_SRP_DONE	B-device SRP done interrupt status. This bit reflects interrupts that are generated upon completion of an SRP when the OMAP730 OTG controller acts as an OTG default-B device. The actual duration of SRP depends on the values programmed into OTG_SYSCON_2.SRP_GPDATA and OTG_SYSCON_2.SRP_GPUVBUS. 0: SRP done interrupt is inactive. 1: SRP done interrupt is active. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.

Table 10–67. OTG Interrupt Status Register (OTG_IRQ_SRC) (Continued)

Bit	Name	Description
7	B_SRP_STARTED	B-device SRP started interrupt status. This bit reflects interrupts that are generated when the OMAP730 OTG controller acts as a default-B device and begins to pulse D+ at the beginning of an SRP request. This interrupt does not occur when OTG_CTRL.BSESSVLD is 1. 0: SRP begin interrupt is inactive. 1: SRP begin interrupt is active. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.
6:1	Reserved	Reserved
0	OPRT_CHG	OTG output port status change interrupt status. This bit reflects interrupts that are generated when the OMAP730 OTG controller requires a change in the OTG transceiver control signals. Any change in OTG_CTRL.[5:0] causes this interrupt. These interrupts include changes to D+ pullup or pulldown, VBUS drive, pullup, and pulldown, or ID pin pullup. Generally, system software responds to this interrupt by issuing I^2C operations to change the control registers in the OTG transceiver. When this bit is 1, OTG_CTRL.[5:0] reflect their values at the time that the interrupt is generated. Other changes on OTG_CTRL.[5:0] can occur before the interrupt is processed but these changes are not shown in OTG_CTRL.[5:0] until after the output port status change interrupt status is cleared. 0: Output port status change interrupt is inactive. 1: An interrupt is generated for a change on OTG output ports status. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.

OTG_IRQ_SRC is a read/write register that reflects all OTG interrupt status.

Table 10–68. OTG Vendor Code Register (OTG_ VC)

Bit	Name	Description
31–16	Reserved	Reserved
15–0	VC	Vendor code identifier: this read-only register reflects the Texas Instruments vendor code identifier: 0x5449 for TI in ASCII.

The OTG vendor code register reflects the binary coded decimal equivalent of the USB vendor code identifier assigned to Texas Instruments.

10.4.2.1 OTG Clock and Reset Requirements

The OTG module is clocked mainly by the 48-MHz input clock from the OMAP730 ULPD module. ULPD module registers CLOCK_CTRL_REG.USB_MCLK_EN, SOFT_REQ_REG.SOFT_USB_OTG_DPLL_REQ, and SOFT_DISABLE_REQ_REG.DIS_USB_HOST_DPLL_REQ control the 48 MHz.

When the OTG module receives a 48-MHz clock from the ULPD module, the OTG controller logic can have its clocks disabled locally via the

OTG_SYSCON_1.OTG_IDLE_EN bit. The OTG module register interface is clocked separately, so OTG module registers other than OTG_SYSCON_2 and OTG_CTRL can be accessed when the 48-MHz clock input is inactive. Accesses to OTG_SYSCON_2 and OTG_CTRL require that the OTG controller 48-MHz clock be active.

The OTG controller 48-MHz clock must be enabled at both the ULPD level and the OTG module level whenever USB On-The-Go functionality is needed. It is not appropriate to disable the OTG controller 48-MHz clock when a USB On-The-Go link is established with another USB On-The-Go dual-role device.

The OTG controller uses the same 48-MHz clock input from the ULPD module to provide clocks to the USB device controller and the USB host controller. When the ULPD 48-MHz clock to the OTG module is disabled, the USB host controller registers cannot be accessed and the USB host controller cannot respond to any downstream USB bus activity.

The OTG module can disable the USB device controller clock using OTG_SYSCON_1.DEV_IDLE_EN. When the ULPD 48-MHz clock to the OTG module is disabled, or when OTG_SYSCON_1.DEV_IDLE_EN is 1, the USB device controller registers cannot be accessed, but the USB device controller can respond to USB bus resume signaling by issuing a USB device controller general interrupt. System software can read USB device controller registers when OTG_SYSCON_1.DEV_IDLE_EN is 1, but must enable the 48-MHz clock to the USB device controller before writing to the USB device controller registers.

Hardware reset of the USB OTG module is provided by the ULPD module. The PER_EN bit in the MPU reset control 2 register controls the reset to many OMAP730 peripherals, including the USB OTG module. When held in hardware reset, the USB OTG controller cannot perform USB On-The-Go SRP or HNP protocols, and its registers cannot be accessed.

The OTG controller provides a soft reset mechanism. When OTG_SYSCON_1.SOFT_RESET is written with a 1, the OTG controller, the USB device controller, and the USB host controller are reset. Soft reset is complete when OTG_SYSCON_1.RESET_DONE is 1. OTG_SYSCON_1.SOFT_RESET automatically clears itself.

10.4.2.2 OTG Controller Power Management

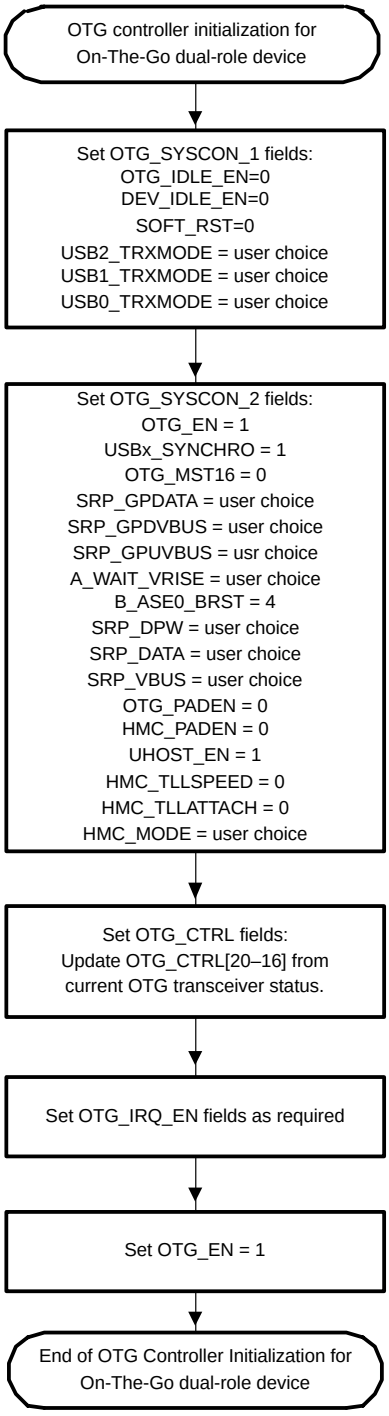
OTG controller power management is provided using the OTG_SYSCON_2.OTG_EN bit. When OTG_SYSCON_2.OTG_EN is 0, OTG controller power consumption is reduced, because the OTG controller logic is not clocked. The OTG controller cannot perform On-The-Go HNP or SRP when it is not clocked.

10.4.2.3 OTG Controller Initialization

OTG controller initialization operations depend on whether or not the system implements an On-The-Go dual-role device.

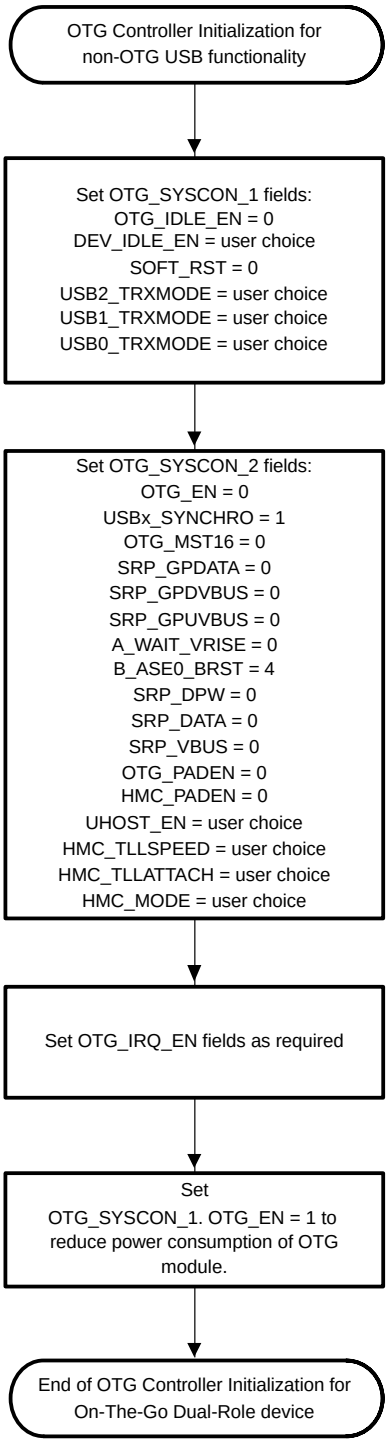
If the application implements an OTG dual-role device, follow the initialization shown in Figure 10–47.

Figure 10–47. OTG Controller Initialization When Implementing an OTG Dual-Role Device



Systems that implement USB functionality without implementing an OTG dual-role device are recommended to follow the initialization shown in Figure 10–48.

Figure 10–48. OTG Controller Initialization When Not Implementing an OTG Dual-Role Device



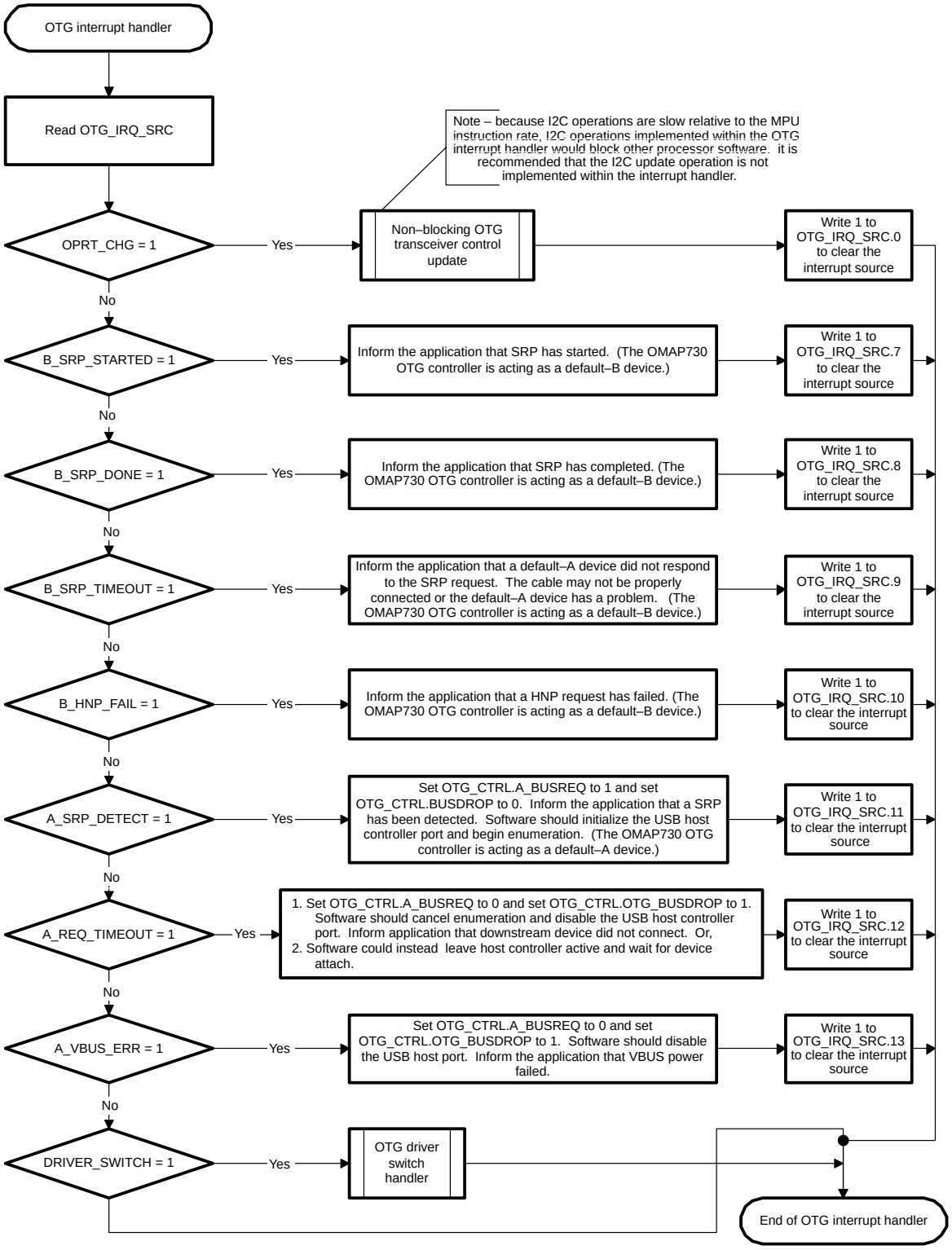
User choice values shown in Figure 10–47 and Figure 10–48 depend on the system implementation, including transceiver types being used, top-level pin multiplexing choice, and connector type. Pin multiplexing options and transceiver types are discussed in Section 10.4.3, *Pin Multiplexing*.

10.4.2.4 OTG Controller Interrupt Handler and Related Software

The OTG controller implements several interrupt sources that are separately enabled via bits in OTG_IRQ_EN. The OTG controller provides a single interrupt output that provides OTG controller interrupts to the MPU level 2 interrupt controller IRQ_30 input.

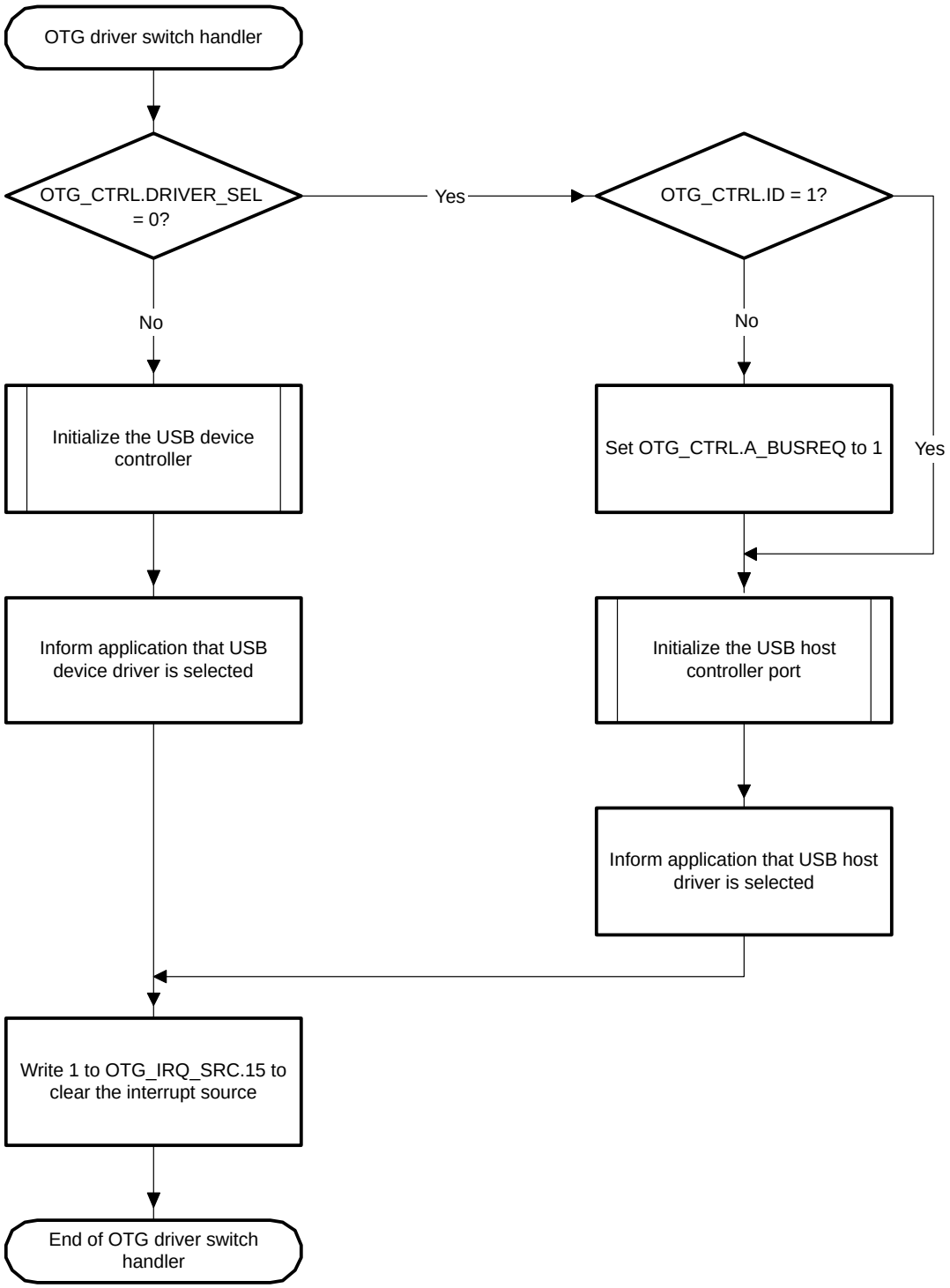
Figure 10–49 shows the operations required in the USB OTG controller interrupt handler. This flowchart references several other operations, which are described in flowcharts later in this section.

Figure 10–49. OTG Interrupt Handler



The OTG driver switch handler is shown in Figure 10–50. It is responsible for switching control of the OTG link between the USB host controller and the USB device controller. This switch occurs mainly because of HNP operations, but can also occur when OTG_SYSCON_2.OTG_EN is set to 1.

Figure 10–50. OTG Driver Switch Handler



The effects of On-The-Go functionality on OMAP730 USB device controller operation are documented in the USB device controller flowcharts. Because USB host controller operations are mainly defined in the open host controller for USB specification, some On-The-Go functionality issues are discussed here.

Most OMAP730 USB host controller software is not affected by USB On-The-Go functionality. A host controller driver written for non-OTG functionality needs few modifications to support On-The-Go functionality. In general, On-The-Go events precede USB host controller driver operations, so the On-The-Go event can cause the OTG driver software to pass the appropriate message to the USB host controller driver. It is never necessary for the USB host controller driver to access OTG controller registers.

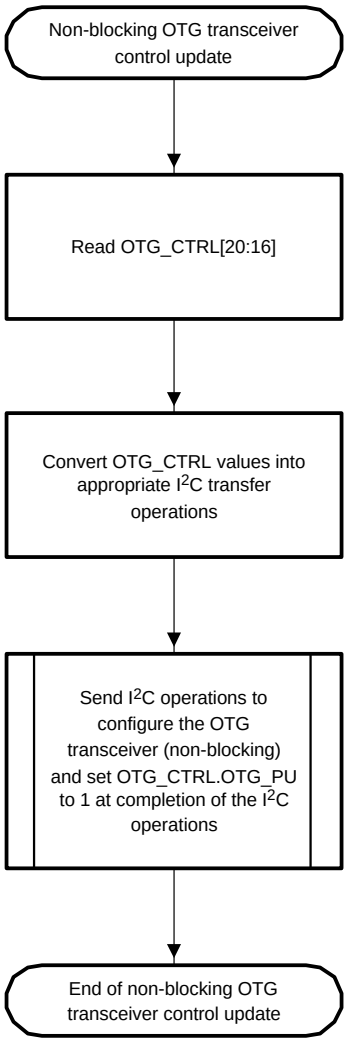
For an On-The-Go connection, the SRP and HNP operations can be considered to replace the attach, detach, suspend, resume, and remote wake functions. This means that it can be beneficial for system software to control the USB host controller driver differently for an OTG link than for a typical USB host port.

The USB host controller driver must implement a mechanism where the application can specify that the OTG port is no longer needed. The host controller driver software must respond to this message by suspending the port associated with the OTG link and cancelling all EDs and TDs associated with the OTG connection. These ED and TD removal activities are ones usually associated with a disconnect event, so the host controller driver already has these functions available. The application typically sends this message and then informs the OTG controller driver that its use of the host is complete and that the OTG controller can now release the bus for a possible HNP.

When the USB device controller owns the USB OTG link, the USB host controller port need not remain active. If no other USB host controller ports are being used, it is possible to disable the USB host controller clock for the duration while the USB device controller owns the OTG link. System software must re-enable the USB host controller clock and re-initialize the USB host on an OTG driver switch interrupt to USB host controller control of the OTG link if the host clock is dynamically disabled.

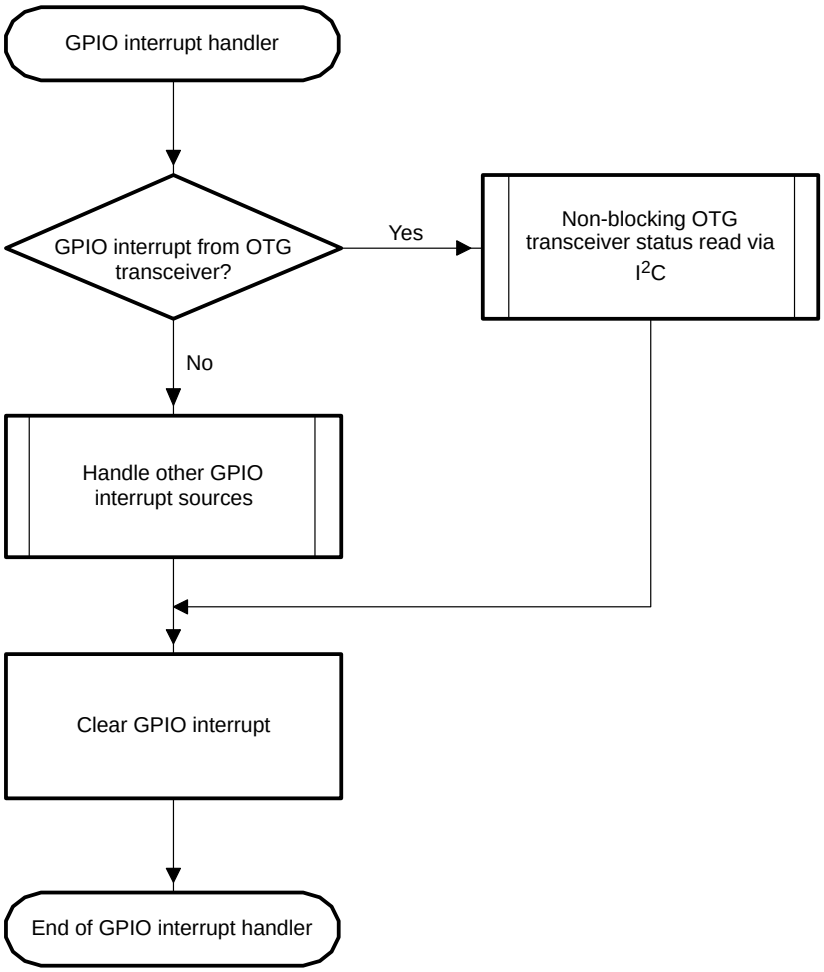
Software to support communication between the USB OTG controller and the external OTG transceiver via I^2C is described in Figure 10–51. This software must handle control information from the OTG controller to the external OTG. These software operations apply directly to the OTG controller OPRT_CHG interrupt. Because I^2C operations are slow (relative to the MPU instruction speed), it is best if these functions are not implemented as part of an interrupt handler; interrupt handlers block other processor software operations until the handler completes. System software must implement a non-blocking mechanism, such as a message-based system, to allow the I^2C operations to be completed outside of the OTG interrupt handler, but then allow updating of the OTG controller OTG_CTRL.OTG_PU bit on completion of the I^2C operations.

Figure 10–51. OTG Transceiver I²C Control Handler



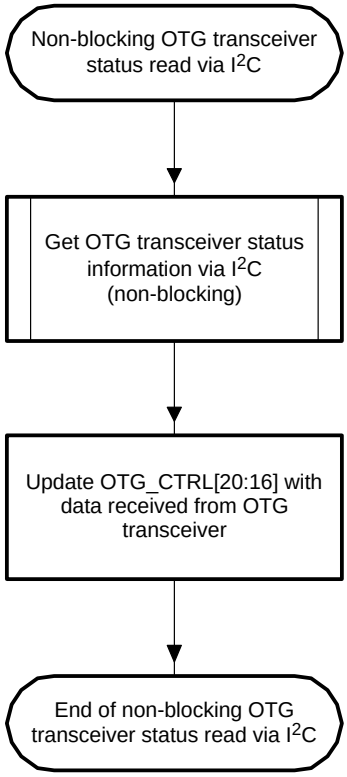
The OTG transceiver signals the need for a status transfer from the OTG transceiver to the OTG controller, using an interrupt output signal. This interrupt is generally connected to an OMAP730 GPIO input pin, and that pin is configured to provide an interrupt to the MPU Level 2 interrupt controller. Figure 10–52 shows the basic GPIO interrupt handler functionality required.

Figure 10–52. GPIO Interrupt Handler Support for OTG Transceiver Interrupt Input



The operations in Figure 10–53 show the non-blocking OTG transceiver status read and update to the OTG controller registers. To reduce the effect on system performance, it is important that the I^2C read operations are non-blocking.

Figure 10–53. OTG Transceiver Status Read



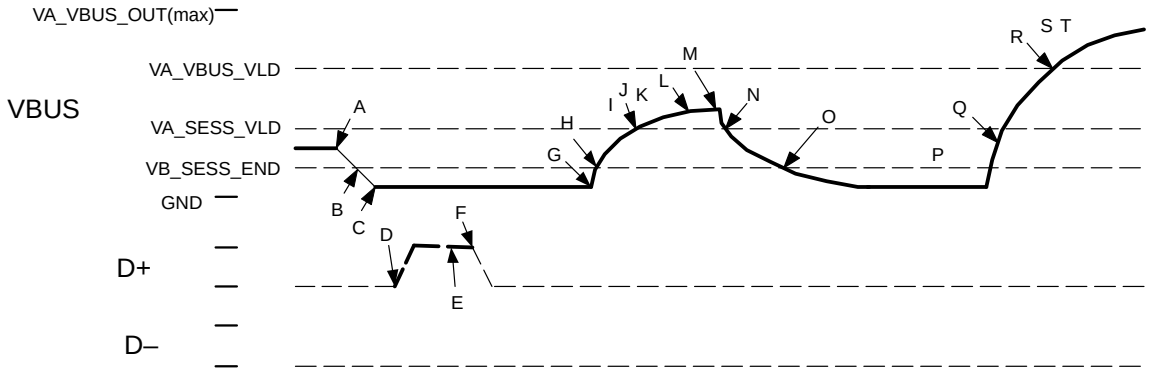
10.4.2.5 Typical SRP and HNP Events

This section describes the typical sequence of events for SRP and HNP events with the OMAP730 OTG controller.

Careful system software design can optimize system performance by managing interrupt source enables at both the OTG controller and at the OTG transceiver. System software developers must carefully weigh the advantages brought by disabling unneeded OTG transceiver interrupt sources versus the performance cost associated with the I^2C activity to enable and disable the transceiver interrupt sources.

Figure 10–54 shows the typical events that occur when OMAP730 acts as an OTG default-A dual-role device and the default-B device issues an SRP. Figure 10–54 shows that the default-B device pulses its D+ pullup resistor. The OMAP730 OTG controller accepts D– pullup pulses as well as the D+ pullup pulse shown as SRP data line.

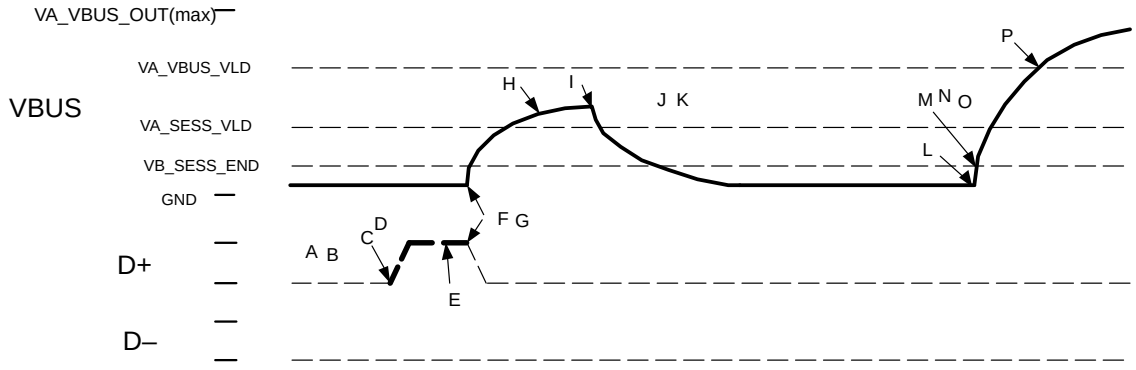
Figure 10–54. OMAP730 OTG Controller Response To SRP When Acting as a Default-A Dual-Role OTG Device



A. OTG dual-role default-B device begins discharging VBUS (optional).
B. VBUS at default-B device falls below VB_SESS_END.
C. OTG dual-role default-B device stops discharging VBUS (optional).
D. Default-B device enables SRP D+ pullup.
E. OMAP730 OTG controller recognizes D+ pulse SRP request if D+ pulse is longer than the duration programmed in OTG_SYSCON_2.SRP_DPW and if OTG_SYSCON_2.SRP_DATA =1. If recognized, OTG_CTRL.OTG_DRV_VBUS is set to 1 and OPRT_CHG and SRP_DETECT interrupts are generated.
F. Default-B device disables D+ pullup (to end SRP Data Line Pulse).
G. Initial conditions for SRP are satisfied. Default-B device enables VBUS SRP pulse.
H. VBUS voltage crosses VB_SESS_END at default-B.
I. VBUS voltage crosses VA_SESS_VLD at OMAP730 OTG transceiver. OTG transceiver issues an interrupt.
J. OMAP730 GPIO interrupt handler identifies OTG transceiver interrupt. Handler initiates I^2C operation to get transceiver interrupt source information.
K. OMAP730 I^2C operations to get transceiver interrupt source information completes. System software sees VBUS > VA_SESS_VLD and sets OTG_CTRL.ASESSVLD to 1.
L. OMAP730 OTG controller recognizes VBUS pulse SRP request if OTG_CTRL.ASESSVLD is 1 for longer than the duration programmed in OTG_SYSCON_2.SRP_DPW and if OTG_SYSCON_2.SRP_VBUS=1. If recognized, OTG_CTRL.OTG_DRV_VBUS is set to 1 and OPRT_CHG and SRP_DETECT interrupts are generated.
M. Default-B device stops driving VBUS pulse.
N. VBUS voltage discharges through various leakage sources.
O. VBUS voltage drops below VB_SESS_END at default-B device.
P. OMAP730 OTG Interrupt handler sees OPRT_CHG interrupt and sees OTG_CTRL.OTG_DRV_VBUS set to 1 and initiates I^2C operations to configure OTG transceiver to drive VBUS. OTG interrupt handler also sees SRP_DETECT interrupt and begins initialization of OTG session as a default-A dual-role device.
Q. OMAP730 I^2C operations to configure OTG transceiver to drive VBUS complete. System software writes a 1 to OTG_CTRL.OTG_PU to indicate that the I^2C operation has completed.
R. VBUS rises above VA_VBUS_VLD. OTG transceiver issues interrupt.
S. OMAP730 GPIO interrupt handler sees OTG transceiver interrupt, initiates I^2C operations to get OTG interrupt source information.
T. OMAP730 I2C operations complete. System software sees OTG status showing that VBUS voltage is above VA_SESS_VLD and sets OTG_CTRL.ASESSVLD to 1. OMAP730 OTG dual-role device is now ready for default-B device to enable its pullup.

Figure 10–55 shows the typical events that occur when OMAP730 acts as an OTG default-B dual-role device and issues an SRP to the default-A device. If OMAP730 completes its SRP request but the default-A device does not drive VBUS within 5.5 seconds, the OMAP730 OTG controller issues a B_SRP_TIMEOUT interrupt. System software must provide a message to the user that says the SRP request failed and that the user needs to check the cable and the A-device. The OMAP730 OTG controller only implements D+ data line SRP pulses and does not request a D– pulse as a data line SRP pulse.

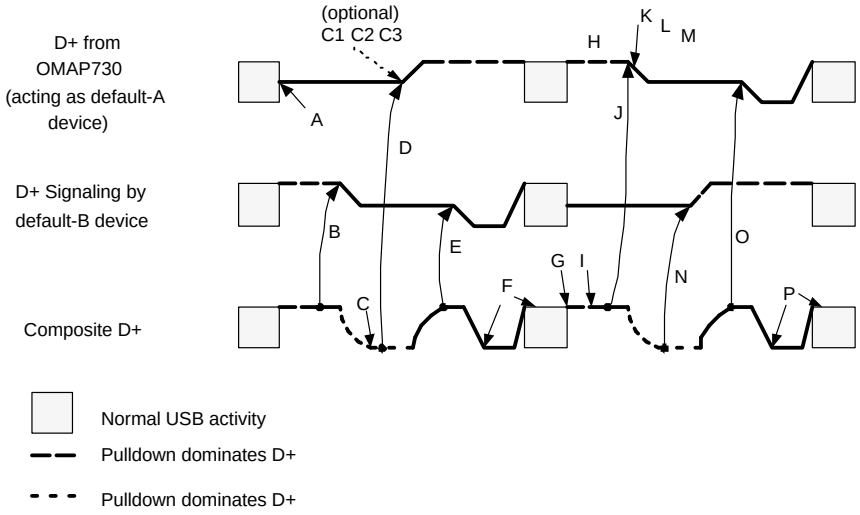
Figure 10–55. OMAP730 OTG Controller SRP Generation When Acting as a Default-A Dual-Role OTG Device



A. OMAP730 is acting as a default-B device. OTG_CTRL.BSESSEND=1. Application decides that it wants to communicate with the default-B dual-role device. System software sets OTG_CTRL.B_BUSREQ.
B. OMAP730 OTG controller sees write of 1 to OTG_CTRL.B_BUSREQ and sees OTG_CTRL.BSESSEND=1. OTG controller waits until USB bus has been SE0 for the minimum time and then begins SRP process by setting OTG_CTRL.OTG_PU and issues the OPRT_CHG interrupt.
C. OMAP730 OTG interrupt handler sees OPRT_CHG interrupt and sees OTG_PU = 1. Handler initiates I^2C operation to enable OTG transceiver D+ pullup.
D. OMAP730 I^2C operations to enable the D+ pullup complete. OMAP730 software sets OTG_CTRL.OTG_PU to 1 to indicate that the D+ pullup enable operation has completed.
E. OMAP730 OTG controller changes OTG_CTRL.OTG_PU to 0 approximately 9 ms after it had set it to 1. OTG controller also sets OTG_CTRL.OTG_VBUS_PU at the same time. OTG controller issues OPRT_CHG interrupt.
F. OMAP730 OTG interrupt handler sees OPRT_CHG interrupt and sees OTG_PU = 0 and sees OTG_PU_VBUS = 1. Handler initiates I^2C operation to disable OTG transceiver D+ pullup and to enable OTG transceiver VBUS pullup.
G. OMAP730 I^2C operations to disable the D+ pullup and enable the VBUS pullup complete. OMAP730 software sets OTG_CTRL.OTG_PU to 1 to indicate that the D+ pullup disable operation has completed.
H. OMAP730 OTG controller changes OTG_CTRL.OTG_PU_VBUS to 0 after a time specified by OTG_SYSCON_2.SRP_GPUVBUS from the time the controller had set OTG_PU_VBUS to 1. If OTG_SYSCON_2.SRP_GPDVBUS = 1, OTG controller sets OTG_CTRL.OTG_PD_VBUS to 1. OTG controller issues OPRT_CHG interrupt.
I. OMAP730 OTG interrupt handler sees OPRT_CHG interrupt and sees OTG_CTRL.OTG_PU_VBUS = 0 and sees current value in OTG_CTRL.OTG_PD_VBUS. Handler initiates I^2C operations to disable the OTG transceiver VBUS pullup and to enable or disable the OTG transceiver VBUS pulldown depending on the value of OTG_PD_VBUS.
J. OMAP730 OTG controller waits for the time specified by OTG_SYSCON_2.SRP_GPUVBUS from the time the controller had set OTG_PU_VBUS to 0. The OTG controller issues a B_SRP_DONE interrupt. If OTG_CTRL.OTG_PD_VBUS =1, the controller sets OTG_PD_VBUS to 0 and issues an OPRT_CHG interrupt.
K. OMAP730 OTG interrupt handler sees OTG interrupt. If OPRT_CHG interrupt is active, interrupt handler sees that OTG_CTRL.OTG_PD_VBUS is 0 and initiates I^2C operations to disable the OTG transceiver VBUS pulldown.
L. Some time later, the default–A device responds to the SRP request by driving VBUS active.
M. The OTG transceiver sees VBUS rise above VB_SESS_VLD and issues an interrupt.
N. The OMAP730 GPIO interrupt handler sees the OTG transceiver interrupt and initiates the I^2C operations which query the OTG transceiver interrupt status.
O. The I^2C operation completes, and OMAP730 system software updates OTG_CTRL.ASESSVLD, OTG_CTRL.BSESSEND, OTG_CTRL.BSESSVLD, OTG_CTRL.VBUSVLD, and OTG_CTRL.ID. If BSESSVLD is 1, then the OMAP730 USB device controller will see VBUS go active and can begin preparations for enumeration.
P. The default-A device sees VBUS rise above VA_VBUS_VLD and can begin USB reset and enumerating the OMAP730 default-B device.

Figure 10–56 shows the typical events that occur when OMAP730 acts as an OTG default-A dual-role device and transitions from acting as an A-host to acting as an A-peripheral via HNP, and then transitions back to acting as an A-host via HNP. The figure shows the optional activities associated with the OTG transceiver autoconnect feature.

Figure 10–56. OMAP730 OTG Controller HNP Events When Acting as a Default-A Dual-Role OTG Device



A: OMAP730 application has no more traffic for default-B device. System software suspends the USB host port which is being used for the OTG link, sets OTG_CTRL.A_BUSREQ to 0. If the OTG transceiver supports the autoconnect feature, system software should initiate the appropriate I^2C operations to enable it before suspending the OTG link.

B: Default-B device sees OTG link suspended by default-A device. Because the default-B device is enabled for HNP, and desires to issue HNP, it disables its D+ pullup, allowing D+ to float. Default-B controller waits for OMAP730 to enable its D+ pullup.

C: OMAP730 OTG controller sees OTG_CTRL.A_SETB_HNPEN=1 and sees SE0 on the bus. OTG controller sets OTG_CTRL.OTG_PU to 0 and issues both the OPRT_CHG interrupt and the driver switch interrupt.

C1. If the OTG transceiver autoconnect feature was enabled in step A, the OTG transceiver will automatically enable its D+ pullup upon seeing SE0 on the bus, and issue an interrupt.

C2.OMAP730 GPIO interrupt handler sees an OTG transceiver interrupt and issues I^2C operations to query the transceiver interrupt source.

C3. OMAP730 I^2C operation completes and the interrupt status shows autoconnect occurred. It is not necessary to update any OTG controller registers when auto-HNP occurs.

D. OMAP730 OTG interrupt handler sees OPRT_CHG interrupt and OTG_PU set to 1, and (if auto-HNP is not used) causes I_2C operation to enable OTG transceiver D+ pullup. OMAP730 OTG interrupt handler sees driver switch interrupt and begins initialization of USB device controller.

E. Default-B device sees D+ pullup as HNP operation and begins acting as a USB host. It signals USB reset and enumerates the OMAP730 dual-role device as a peripheral, and begins normal USB bus activity.

F. OMAP730 USB device controller sees and responds appropriately to USB reset, enumeration and normal USB bus activity.

G. Default-B device finishes its bus activity and suspends the USB link.

H. OMAP730 application decides that it wants to communicate with the default-B device. System software sets OTG_CTRL.A_BUSREQ.

I. OMAP730 OTG controller sees suspend signaled on the OTG link, sets OTG_CTRL.OTG_PU to 0, issues OPRT_CHG interrupt.

J. OMAP730 OTG interrupt handler sees OPRT_CHG interrupt, sees OTG_CTRL.OTG_PU set to 0, issues I^2C operations to disable the OTG transceiver D+ pullup.

K. OMAP730 I^2C operations to disable the D+ pullup complete. OMAP730 software sets OTG_CTRL.OTG_PU to 1 to indicate that the D+ pullup disable operation has completed.

L. OMAP730 OTG controller sees write of 1 to OTG_CTRL.OTG_PU and issues driver switch interrupt.

M. OMAP730 OTG interrupt handler sees driver switch interrupt, begins initialization of host port.

N. Default-B device sees SE0 on the OTG link, turns on its D+ pullup.

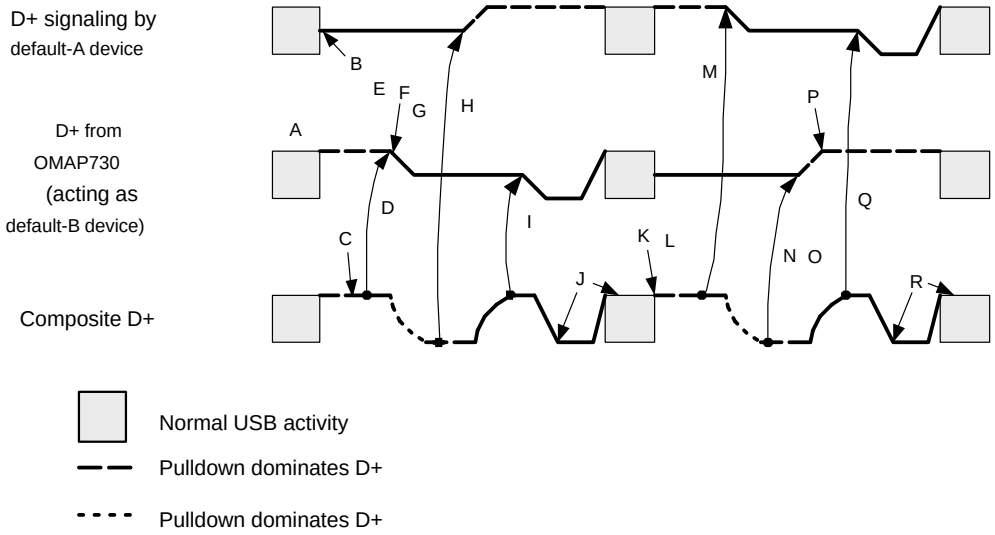
O. OMAP730 OTG controller and USB host controller see D+ pullup as HNP operation. OMAP730 USB host controller signals USB reset, sends the set feature signal with B_HNPEN feature enabled, and enumerates the default-B dual-role device as a peripheral, and begins normal USB bus activity.

P. Default-B device sees and responds appropriately to USB reset, the set feature signal, enumeration and normal USB bus activity.

Figure 10–57 shows the typical events that occur when OMAP730 acts as an OTG default-B dual-role device and transitions from acting as a B-peripheral to acting as a B-host via HNP, and then transitions back to acting as a B-peripheral via HNP.

If the default-A device does not enable its pullup resistor within the required time, if the default-A device issues a USB resume during the HNP process, or if the VBUS voltage falls below VB_SESS_VLD threshold during the HNP process, the OMAP730 OTG controller issues a B_HNP_FAIL interrupt.

Figure 10–57. OMAP730 OTG Controller HNP Events When Acting as a Default-B Dual-Role OTG Device



A: OMAP730 application decides that it wishes to communicate with the default-A device. System software sets OTG_CTRL.B_BUSREQ.

B: Default-A device has no more traffic for the default-B device (OMAP730). Default-A device suspends the OTG link.

C: OMAP730 OTG controller (acting as the OTG dual-role default-B device) sees OTG link suspended, sees OTG_CTRL.B_HNPEN set to 1, sees OTG_CTRL.B_BUSREQ set to 1. OTG controller sets OTG_CTRL.OTG_PU to 0 and issues OPRT_CHG interrupt.

D. OMAP730 OTG interrupt handler sees OPRT_CHG interrupt and sees OTG_PU = 0. Handler initiates I^2C operation to disable OTG transceiver D+ pullup.

E. OMAP730 I^2C operations to disable the D+ pullup complete. OMAP730 software sets OTG_CTRL.OTG_PU to 1 to indicate that the D+ pullup disable operation has completed.

F. OMAP730 OTG controller sees write of 1 to OTG_CTRL.OTG_PU and issues driver switch interrupt.

G. OMAP730 OTG interrupt handler sees driver switch interrupt, begins initialization of host port.

H. Default-A device sees SE0 on the OTG link, turns on its D+ pullup.

I. OMAP730 OTG controller sees D+ pullup as HNP operation and OMAP730 USB host controller sees D+ pullup as an attach. OMAP730 USB host controller signals USB reset, and enumerates the default-A dual-role device as a peripheral, and begins normal USB bus activity.

J. Default-A device sees and responds appropriately to USB reset, enumeration and normal USB bus activity.

K. OMAP730 application decides it has no more information to communicate to the default-A device. System software suspends the USB host port which is being used for the OTG link, sets OTG_CTRL.A_BUSREQ to 0.

L. OMAP730 OTG controller sees OTG_CTRL.A_BUSREQ set to 0. OTG controller sets OTG_CTRL.OTG_PU to 1 and issues both OPRT_CHG and driver switch interrupts.

M. Default-A device sees OTG link signal USB Suspend. Default-A device disables its D+ pullup.

N. OMAP730 OTG interrupt handler sees OPRT_CHG interrupt, sees OTG_CTRL.OTG_PU set to 1, initiates I^2C operations to enable OTG transceiver D+ pullup.

O. OMAP730 OTG interrupt handler sees Driver Switch and begins initialization of OMAP730 USB device controller.

P. OMAP730 I^2C operations to enable OTG transceiver D+ pullup complete.

Q. Default-A device sees D+ pullup, signals USB reset, issues Set Feature of the B_HNP_ENABLE feature, and enumerates the OMAP730 default-B dual-role device as a peripheral, and begins normal USB bus activity.

R. OMAP730 USB device controller sees and responds appropriately to USB reset, set feature of the B_HNP_ENABLE feature, enumeration, and normal USB bus activity.

10.4.2.6 System-Level OTG Considerations

The USB OTG specification states that an OTG dual-role device has only one USB connector.

An OTG implementation using an OTG transceiver that is based on the *OTG Transceiver Interface Specification* can use the transceiver autoconnect feature. Enabling this OTG transceiver feature when OMAP730 is acting as a default-A dual-role On-The-Go device eases the timing requirements on system software during an HNP transition from OMAP730 acting as an A-host to OMAP730 acting as an A-peripheral.

A system that implements an OTG controller has a USB mini-AB receptacle. This receptacle is used with an OTG cable to connect the system to another OTG system. It is possible to connect a non-OTG USB device to the OMAP730-based OTG system mini-AB receptacle using an adapter cable as defined in the On-The-Go supplement to the USB specification. If the system supports this option, it may be necessary to provide a VBUS source with larger current source capability than is typically available through an OTG transceiver.

10.4.3 Pin Multiplexing

OMAP730 USB signal multiplexing determines which USB functionality is available at which OMAP730 pins.

A USB transceiver is needed for each USB port used in the system. It converts between signaling appropriate for the OMAP730 USB controllers and signaling appropriate for the USB wire. OMAP730 USB functionality includes support for several types of USB transceivers. OMAP730 provides one integrated USB transceiver suitable for a single USB host or USB device port, but it is not capable of acting as a USB OTG transceiver.

Several different types of external transceiver signaling are supported. Signaling between the OMAP730 USB controller and the external USB transceiver for monitoring and controlling the differential USB signal can be via a 6-wire, 4-wire, or 3-wire signaling interface, with two or more additional control signals provided either by additional signals or via an I^2C link. OTG transceivers can support the 6-wire, 4-wire, and/or 3-wire basic signaling but generally provide an interrupt output and an I^2C link for the additional control and status reporting required by OTG functionality.

10.4.4 Selecting and Configuring USB Connectivity

The process of selecting desired USB connectivity and configuring OMAP730 for that connectivity can be done using the steps listed below.

10.4.4.1 Select Desired USB Functionality

Choose the desired USB functionality. OMAP730 can bring one USB port to device pins. Options include:

- ❏ One USB device port
- ❏ One USB OTG port
- ❏ One USB host port

10.4.4.2 Select How USB Functionality Is Multiplexed to OMAP730 Pins

OMAP730 provides pin interfaces for the USB port. The USB functional port can be mapped to the different OMAP730 USB pin groups.

- ❏ Pin group 0 is associated with the OMAP730 integrated USB transceiver, which is connected to OMAP730 pins USB.DP and USB.DM. The transceiver can be used as a USB host or USB device port, but is not for use as a USB OTG port without additional external hardware. Pin group 0 is related to USB signal multiplexing port 0.

The other two USB-related pin groups are associated with standard CMOS input and output pins. USB connectivity that uses these pin groups must use USB external transceivers. OMAP730 top-level pin multiplexing allows a wide variety of functionality to be provided via these pins, so selection of these pins for use as USB functionality limits designer ability to use those pins for other (non-USB) functionality.

- ❏ Pin group 3 maps signals associated with USB signal multiplexing port 0 to the same OMAP730 pins used in pin group 3. The pin names for alternate pin group 3 are as follows:
  - ■ USB.VBUSI/USB.TXEN
  - ■ USB.DP/USB.TXD
  - ■ USB.DM/USB.SE0
  - ■ USB.PU.EN/USB.RCV.
  - ■ EAC.CRESET/USB.VP
  - ■ EAC.MCLK.OUT/USB.VM
  - ■ I2C.SCK/USB.SPEED
  - ■ I2C.SDA/USB.SUSPEND

When a USB host controller port is disabled, the port appears to the controller as a disconnected port. When the USB device controller is not available to a pin group and OTG is not enabled, the USB device controller sees pin signaling equivalent to USB_RESET.

When an HMC_MODE is selected that can connect a USB multiplexing port to OMAP730 device pins, but all of the pins associated with that pin group are set for non-USB functionality, the associated USB multiplexing port sees 6-/8-wire transceiver signaling, which implies that both D+ and D- are low. If

that pin group is associated with a host controller port, that host controller port appears as a disconnected USB link. If that pin group is associated with the device controller, the device controller sees USB reset signaling.

Functionality is undefined when top-level pin multiplexing selects non-USB signalling of one or more of the pins that are required by the selected HMC_MODE and transceiver signalling type.

Select USB and/or USB OTG Transceiver Type

USB connectivity on OMAP730 USB pin group 3 requires external components, except when it is configured for a TLL mode. Several different types of external transceiver signaling are supported. Signaling between the OMAP730 USB controller and the external USB transceiver for monitoring and controlling the differential USB signal can be via a 6-wire, 4-wire, or 3-wire signaling interface, with two or more additional control signals provided either by additional signals or via an I^2C link. OTG transceivers can support the 6-wire, 4-wire, and/or 3-wire basic signaling, but generally provide an interrupt output and an I^2C link for the additional control and status reporting required by OTG functionality.

Consider which OMAP730 pins are required in each diagram and determine if any particular choice (3-wire, 4-wire, or 6-wire) is advantageous in terms of other non-USB functionality that is available for each pin group. Choose a transceiver type.

A USB OTG transceiver can generally be used in place of a USB transceiver; using an OTG transceiver with an I^2C interface can reduce the number of OMAP730 pins that are required. This can assist in making non-USB functionality available on the USB pin groups.

Determine Proper Top-level Multiplexing Settings

Top-level pin multiplexing settings are determined primarily by which pins perform USB functions and the type of transceiver connected to the pins.

Pin multiplexing is controlled on a pin-by-pin basis via the top-level pin multiplexing. Each pin, referenced by default pin name, must be configured to provide the correct functional signal. To configure a pin for its USB functionality, determine the default pin name to be configured. For example, the default pin name for I2C.SCK/USB.SPEED is I2C.SCK. See Table 10–69 to determine the name of the register and the name of the bit field in that register that controls multiplexing for the pin. Choose the desired USB signal name to be used on that pin, and use the associated value under pin multiplexing configuration value as the value to be programmed into the specified bit field in the specified pin multiplexing control register.

Table 10–69. Top-Level Pin Multiplexing Configuration for OMAP730 USB-Related Pins

Signal	Mode 0			Mode 1			Mode 3			Mode 4			Mode 5		
	Pad Name	†	‡	Pad Name	†	‡	Pad Name	†	‡	Pad Name	†	‡	Pad Name	†	‡
USB_DP	usb_dp	2	5												
USB_DM	usb_dm	2	5												
USB_TXD_VP							usb_dp	2	5						
USB_SEO_VM							usb_dm	2	5						
USB_TXD										usb_dp	2	5			
USB_SEO										usb_dm	2	5			
USB_VP							creset	3	1						
USB_VM							mclk_out	3	0						
USB_VBUSI	usb_vbusi	2	7	mpu_ext_nirq	9	7									
USB_TXEN							usb_vbusi	2	7						
USB_PU_EN	usb_pu_en	2	6												
USB_RCV							usb_pu_en	2	6						
USB_ SUSPEND													mpu_i2c_sda	5	0
USB_SPEED													mpu_i2c_sck	5	1

† Configuration register = n PERSEUS2_IO_CONFn with n = 0 to 13
‡ Register field = n Bits [4 x n : 3 + 4 x n] with n = 0 to 7

Determine OTG Module Control Register Settings

The OTG module provides registers that control several aspects of the USB pin signaling. These registers must be properly configured to allow proper USB operation, even when the OTG feature is not being used.

OTG_SYSCON_1.USB0_TRX_MODE must be 0 or 3 to allow proper operation of the integrated USB transceiver on USB pin group 0. For cases where USB alternate pin group 2 is used, OTG_SYSCON_1.USB0_TRX_MODE must be set to match the type of transceiver used on USB alternate pin group 3 (3 for a 6-wire transceiver, 1 for a 4-wire transceiver, or 2 for 3-wire transceiver).

OTG_SYSCON_2.USBx_SYNCHRO must be set to 1 for proper transceiver operation.

OTG_SYSCON_2.OTG_PADEN must be set to 0 whenever OTG functionality is required. Writing a 1 to this bit prevents proper operation of the OTG_CTRL register. When OTG_PADEN is 0 and OTG_EN = 0, the GPIO0/USB.VBUS input is not provided to the USB device controller. In this case, software monitors GPIO0 and updates OTG_CTRL.BSESSVLD with the value from GPIO0.

When OTG_PADEN is 1 and GPIO_0/USB.VBUS is configured for USB.VBUS functionality, GPIO0/USB.VBUS propagates to the USB device controller without software intervention, but the OTG_CTRL register bits ASESSVLD, BSESSEND, BSESSVLD, VBUSVLD, and ID are read-only and are always 0. Because of this, it is impossible to implement USB-OTG functionality when OTG_PADEN is 1.

OTG_SYSCON_2.HMC_PADEN determines whether values for UHOST_EN, HMC_MODE, TLL_ATTACH, and TLL_SPEED are provided by bits in OTG_SYSCON_2 or by bits in MOD_CONF_CTRL_0, as shown in Table 10–70. For software compatibility with future devices, HMC_PADEN must be set to 0.

Table 10–70. UHOST_EN, HMC_MODE, TLL_ATTACH, TLL_SPEED Source Selection

Register	Field	Value	
OTG_SYSCON2	HMC_PADEN	0	1
MOD_CONF_CTRL_0	UHOST_EN	Don't care	Valid
	HMC_MODE		
	HMC_TLLATTACH		
	HMC_TLLSPEED		
OTG_SYSCON_2	CONF_MOD_USB_HOST_HHC_UHOST_EN_R	Valid	Don't care
	CONF_MOD_USB_HOST_HMC_MODE_R		
	CONF_MOD_USB_HOST_HMC_TLL_ATTACH_R		
	CONF_MOD_USB_HOST_HMC_TLL_SPEED_R		
Values used	UHOST_EN	Values from OTG_SYSCON_2	Values from MOD_CONF_CTRL_0
	HMC_MODE		
	TLL_ATTACH		
	TLL_SPEED		

If OTG_SYSCON_2.HMC_PADEN is 0, OTG_SYSCON_2.HMC_MODE must be programmed with the HMC_MODE value chosen. If OTG_SYSCON_2.HMC_PADEN is 1, the chosen HMC_MODE value must be written to MOD_CONF_CTRL_0.CONF_MOD_USB_HOST_HMC_MODE_R.

10.4.5 Transceiver Signaling Types

USB Transceiver Unidirectional Signaling

When a USB or USB OTG transceiver is connected to OMAP730 and is used in unidirectional DAT/SE0 signaling mode, the signaling shown in Table 10–71 is used.

Table 10–71. Signaling Between USB Controller and Unidirectional USB Transceiver Using DAT/SE0 Signaling

Logical Signal Name(s)	OMAP730 Pin Direction	Transceiver Pin Direction	Description
$\overline{OE}$	Output	Input	When low, USB transceiver drives D+ and D-.
DAT and SE0	Output	Input	Controls the values output by the USB transceiver on D+ and D- when $\overline{OE}$ is low. Ignored when $\overline{OE}$ is high.

OEn	DAT	SE0	D+	D-
0	0	0	0	1
	1	0	1	0
	X	1	0	0
1	X	X	Undriven	Undriven

RCV	Input	Output	Output from transceiver differential receiver.

D+	D-	RCV
0	0	X
0	1	0
1	0	1
1	1	X

RCVVP	Input	Output	Output from transceiver single-ended D+ signal receiver .

D+	RCVVP
0	0
1	1

RCVVM	Input	Output	Output from transceiver single-ended D- signal receiver.

D-	RCVVM
0	0
1	1

SPEED	Output	Input	Determines transceiver D+, D- output characteristics. 0 = Low speed. 1 = Full speed. Transceivers with I^2C interfaces can implement this functionality as a control bit rather than a pin.
SUSPEND	Output	Input	Controls transceiver powerdown modes. 0 = Active mode. 1 = Low-power mode Transceivers with I^2C interfaces can implement this functionality as a control bit rather than a pin.

OMAP730 does not support unidirectional transceivers that implement VPO/VMO signaling.

USB Transceiver 3-Wire Bidirectional Signaling

When a USB or USB OTG transceiver is connected to OMAP730 and is used in 3-wire bidirectional TXDAT/TXSE0 signaling mode, the signaling shown in Table 10–72 is used.

Table 10–72. Signaling Between USB Controller and 3-Wire Bidirectional USB Transceiver Using TXDAT/TXSE0 Signaling

Logical Signal Name	OMAP730 Pin Direction	Transceiver Pin Direction	Description				
$\overline{OE}$	Output	Input	When low, USB transceiver drives D+ and D-.				
TXDAT/ RCVDAT and TXSE0/ RCVSE0			When $\overline{OE}$ is low, OMAP730 drives TXDAT and TXSE0 and the transceiver drives D+ and D- based on the values of TXDAT and TXSE0.				
	Output	Input	OEn	TXDAT	TXSE0	D+	D-
			0	0	0	0	1
				1	0	1	0
				X	1	0	0
	Input	Output	OEn	D+	D-	RCVDAT	RCVSE0
			1	0	0	0	1
				0	1	0	0
				1	0	1	0
				1	1	Undefined	Undefined
SPEED	Output	Input	Determines transceiver D+, D- output characteristics. 0 = Low speed 1 = Full speed Transceivers with I^2C interfaces can implement this functionality as a control bit rather than a pin.				
SUSPEND	Output	Input	Controls transceiver powerdown modes. 0 = Active mode 1 = Low-power mode Transceivers with I^2C interfaces can implement this functionality as a control bit rather than a pin.				

OMAP730 does not support 3-wire bidirectional signaling using VP/VM signals.

USB Transceiver 4-Wire Bidirectional Signaling

When a USB or USB OTG transceiver is connected to OMAP730 and is used in 4-wire bidirectional DAT/SE0 signaling mode, the signaling shown in Table 10–73 is used.

Table 10–73. Signaling Between USB Controller and 4-Wire Unidirectional USB Transceiver Using VP/VM Signaling

Logical Signal Name	OMAP730 Pin Direction	Transceiver Pin Direction	Description		
$\overline{OE}$	Output	Input	When low, USB transceiver drives D+ and D-.		
TXVM/ RCVVM			Value driven to or received from D-		
	Output	Input	OEn	TXVM	D-
			0	0	0
			0	1	1
	Input	Output	OEn	D-	RCVVM
			1	0	0
			1	1	1
TXVP/ RCVVP			Value driven to or received from D+		
	Output	Input	OEn	TXVP	D+
			0	0	0
			0	1	1
	Input	Output	OEn	D+	RCVVP
			1	0	0
			1	1	1
RCV	Input	Output	Output from transceiver single-ended D- signal receiver.		
			D+	D-	RCV
			0	0	X
			0	1	0
			1	0	1
			1	1	X
SPEED	Output	Input	Determines transceiver D+, D- output characteristics. 0 = Low speed 1 = Full speed Transceivers with I^2C interfaces can implement this functionality as a control bit rather than a pin.		
SUSPEND	Output	Input	Controls transceiver power-down modes. 0 = Active mode 1 = Low-power mode Transceivers with I^2C interfaces can implement this functionality as a control bit rather than a pin.		

OMAP730 does not support 4-wire bidirectional signaling using DAT/SE0 signals.

10.4.6 USB OTG External Connectivity

To provide USB OTG connectivity, a dual-role device system that provides a USB OTG controller must implement certain features. These features include a USB mini-AB receptacle, VBUS monitoring and control, transient suppression, ID monitoring, controllable D+ and D- series, pullup and pulldown resistors, and a D+ and D- driver/receiver. USB OTG transceivers that implement many of these features are commercially available.

Some USB OTG transceivers implement a 6-wire connection to the OTG controller, whereas others require only 4 wires or 3 wires. The OMAP730 device must be properly configured to support the signaling required by the attached transceiver. See Section 10.4.4, *Selecting and Configuring USB Connectivity.*

Many OTG transceiver control and status functions are provided using I^2C registers. This reduces the number of connections between the OTG transceiver and the OTG controller. OTG transceiver I^2C registers control a variety of functions, including D+ and D- pullups, D+ and D- pulldowns, and the VBUS output. OTG transceiver I^2C registers provide transceiver status including status of the VBUS and ID pins and interrupt conditions.

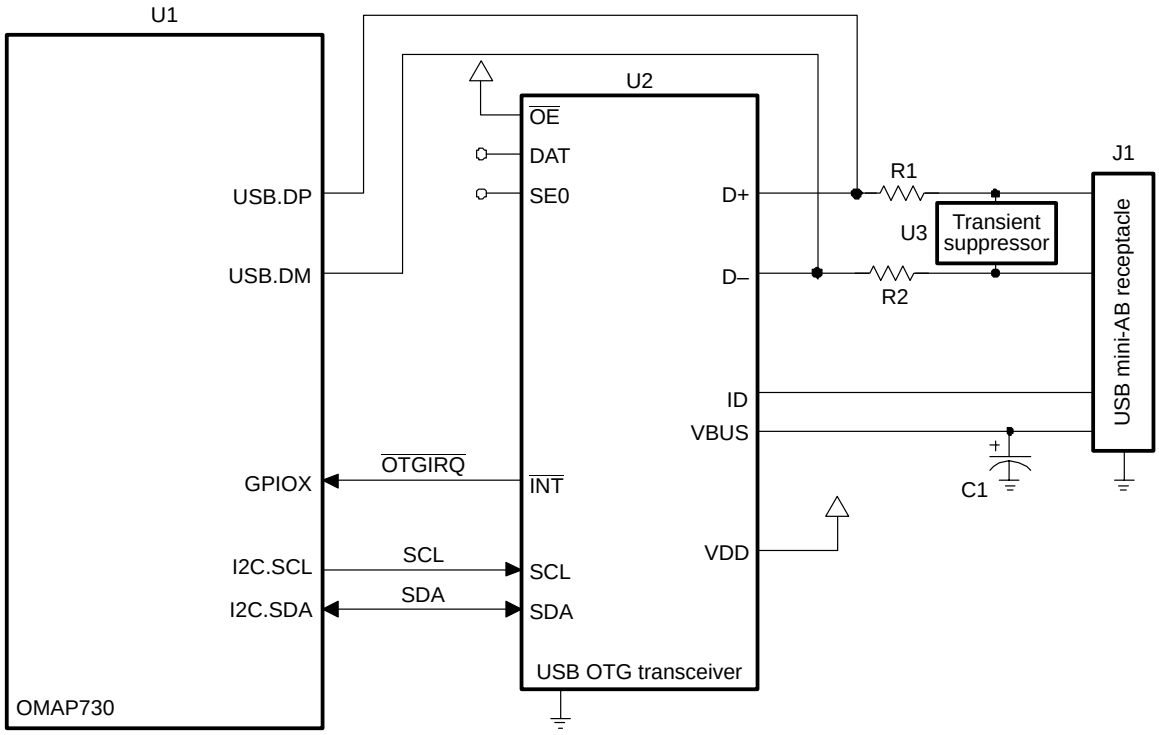
The typical OTG transceiver provides an interrupt output to the processor. This interrupt informs the processor when VBUS state changes, when ID state changes, or when any number of other transceiver-dependent conditions occur. When OTG transceiver interrupts occur, system software must query the OTG transceiver status using I^2C and update the OTG controller registers as appropriate.

Some system applications require that the USB Mini_AB receptacle can be connected to downstream non-OTG USB devices, such as mice, keyboards, or printers. In such cases, the USB OTG specification allows use of a converter cable with a mini-A plug on one end and a standard type-A receptacle on the other end. The USB OTG transceiver, which is capable of supplying the amount of VBUS current required by the USB OTG specification, might not be able to meet the VBUS current supply requirement of the USB specification. In such cases, it is necessary to provide some alternate VBUS source that meets the USB specification requirements.

Pin Group 0 OTG

The integrated USB transceiver on OMAP730 USB pin group 0 does not provide the functionality specific to OTG transceivers. To implement an OTG dual-role device using OMAP730 USB pin group 0, an external OTG transceiver is necessary to provide that additional functionality. In such a system, the external OTG transceiver provides the VBUS detect and drive, D+ and D- pullup and pulldown, and ID detection features, while the OMAP730 integrated USB transceiver provides the D+ and D- signal drive and receive functions (see Figure 10–58).

Figure 10–58. OTG on USB Pin Group 0



R1, R2 27 Ohm 5%
C1 VBUS capacitance must meet OTG spec C_{DRD_VBUS}
U1 OMAP730
U2 USB OTG transceiver
U3 Transient suppressor, like SN65220, SN65240, or SN75240
J1 USB mini-AB receptacle

Proper initialization of the OMAP730 device to support this mode of operation requires proper setting of the top-level pin multiplexing for pins USB.DP and USB.DM and selection of an HMC_MODE value that routes the OTG controller port to pin group 0. OTG_SYSCON_1.USB0_TRX_MODE must be set to 3 to allow proper operation of the integrated USB transceiver.

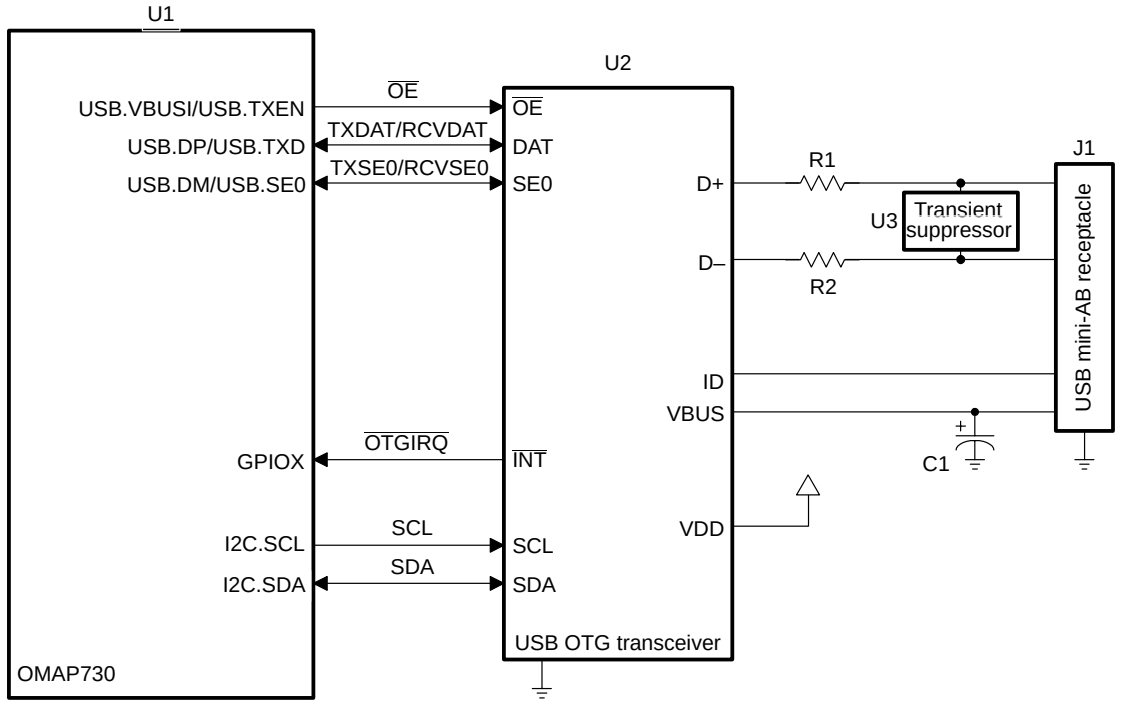
System software is responsible for transferring control signals from the OMAP730 OTG controller to the OTG transceiver and status information from the OTG transceiver to the OMAP730 OTG controller registers.

When acting as an OTG default-A dual-role device, system software must make use of the OTG transceiver for autoconnect mode. System software must also configure the OTG transceiver to suspend mode whenever OMAP730 acts as a default-A device and is placing the OTG link into suspend so that the remote-B device can issue an HNP.

If you want to use the external OTG transceiver ability to provide I^2C signaling to the transceiver D+ and D- pins, it is necessary to control the OTG transceiver OEn input using an OMAP730 GPIO pin. It is also necessary to set USB_TRANSCEIVER_CTRL.CONF_USB0_ISOLATE to 1 to ensure that the I^2C signals have no effect on the OMAP730 USB controllers.

OTG on Pin Group 3 Using 3-Wire OTG Transceiver

Figure 10–59. OTG on USB Pin Group 3 Using 3-Wire OTG Transceiver



R1, R2 Value depends on transceiver
C1 VBUS capacitance must meet OTG spec
U1 OMAP730
U2 USB OTG transceiver
U3 Transient suppressor, like SN65220, SN65240, or SN75240
J1 USB mini-AB receptacle

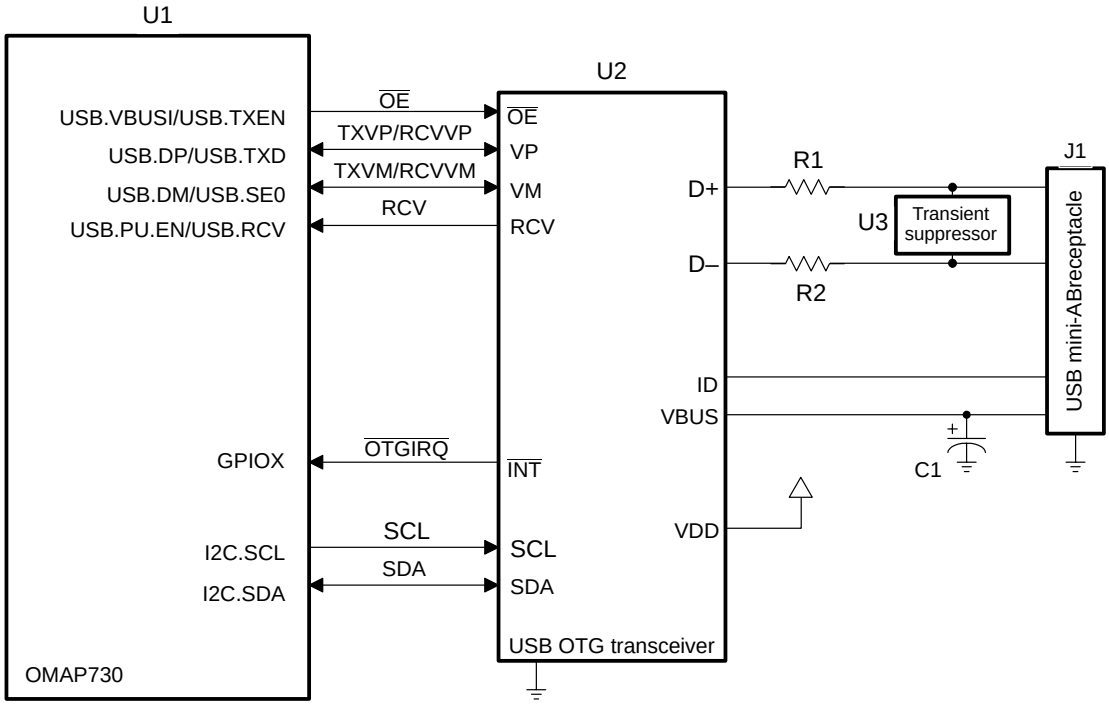
Proper initialization of the OMAP730 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❏ USB.VBUSI/USB.TXEN
- ❏ USB.DP/USB.TXD
- ❏ USB.DM/USB.SE0

HMC_MODE must be set to a value that routes OTG functionality to USB pin group 1. OTG_SYSCON_1.USB1_TRX_MODE must be set to 2 to allow proper bidirectional operation of the 3-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins USB.DP/USB.TXD and USB.DM/USB.SE0.

OTG on Pin Group 1 Using 4-Wire OTG Transceiver

Figure 10–60. OTG on USB Pin Group 1 Using 4-Wire OTG Transceiver



R1, R2 Value depends on transceiver
C1 VBUS capacitance must meet OTG spec C DRD_VBUS
U1 OMAP730
U2 USB OTG transceiver
U3 Transient suppressor, like SN65220, SN65240, or SN75240
J1 USB mini-AB receptacle

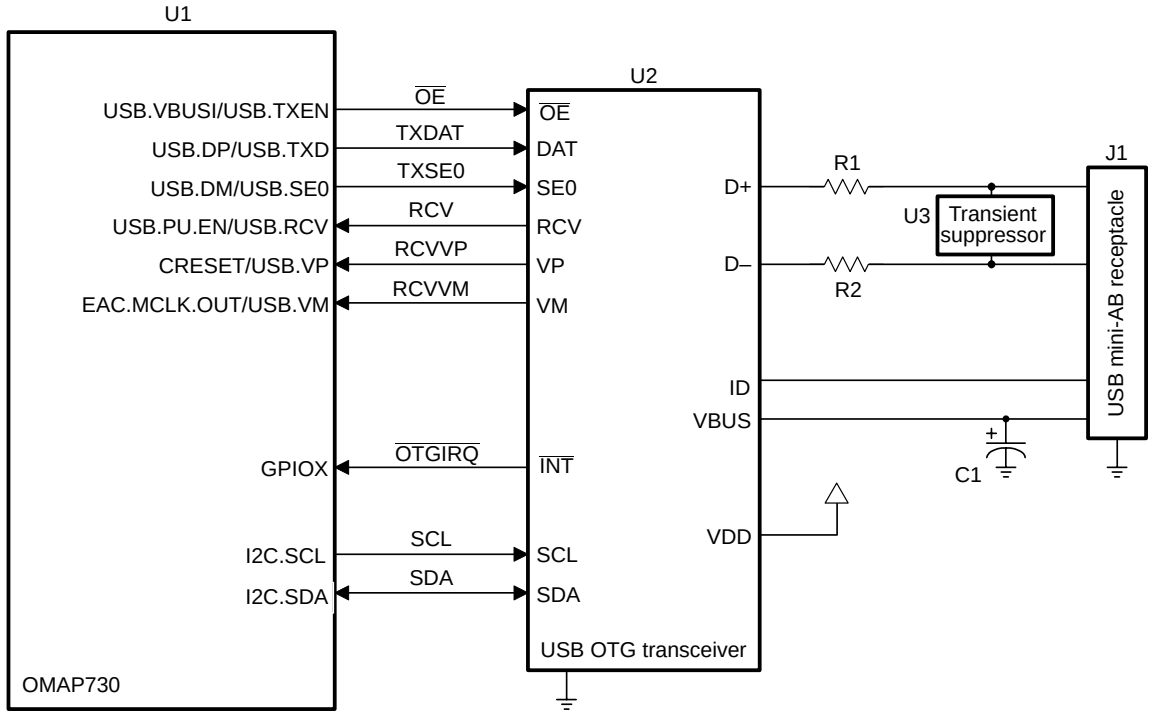
Proper initialization of the OMAP730 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❏ USB.VBUSI/USB.TXEN
- ❏ USB.DP/USB.TXD
- ❏ USB.DM/USB.SE0
- ❏ USB.PU.EN/USB.RCV

HMC_MODE must be set to a value that provides OTG functionality on USB pin group 1. OTG_SYSCON_1.USB1_TRX_MODE must be set to 1 to allow proper bidirectional operation of the 4-wire USB transceiver. OTG_SYS-CON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins USB.DP/USB.TXD and USB.DM/USB.SE0.

OTG on Pin Group 1 OTG Using 6-Wire OTG Transceiver

Figure 10–61. OTG on USB Pin Group 1 Using 6-Wire OTG Transceiver



R1, R2 Value depends on transceiver
C1 VBUS capacitance must meet OTG spec C DRD_VBUS
U1 OMAP730
U2 USB OTG transceiver
U3 Transient suppressor, like SN65220, SN65240, or SN75240
J1 USB mini-AB receptacle

Proper initialization of the OMAP730 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❏ USB.VBUSI/USB.TXEN
- ❏ USB.DP/USB.TXD
- ❏ USB.DM/USB.SE0
- ❏ CRESET/USB.VP
- ❏ EAC.MCLK.OUT/USB.VM
- ❏ USB.PU.EN/USB.RCV

HMC_MODE must be set to a value that provides OTG functionality on USB Pin group 1. OTG_SYSCON_1.USB1_TRX_MODE must be set to 3 to allow proper operation of the 6-wire USB transceiver.

10.4.7 Host Controller Connectivity With USB Transceivers

To provide a robust USB solution, a system that provides a USB host controller must implement certain features. These features include a type-A receptacle, power on the VBUS signal (can be switched or unswitched power), transient suppression, pulldown resistors, and USB-compatible downstream port transceiver.

Because OMAP730 does not provide a pin that connects to the USB host controller port power control registers, some other mechanism must be used if VBUS switching is required. Similarly, OMAP730 does not provide any pins that connect to the USB host controller overcurrent status bits, so some other mechanism must be used if overcurrent sensing is required.

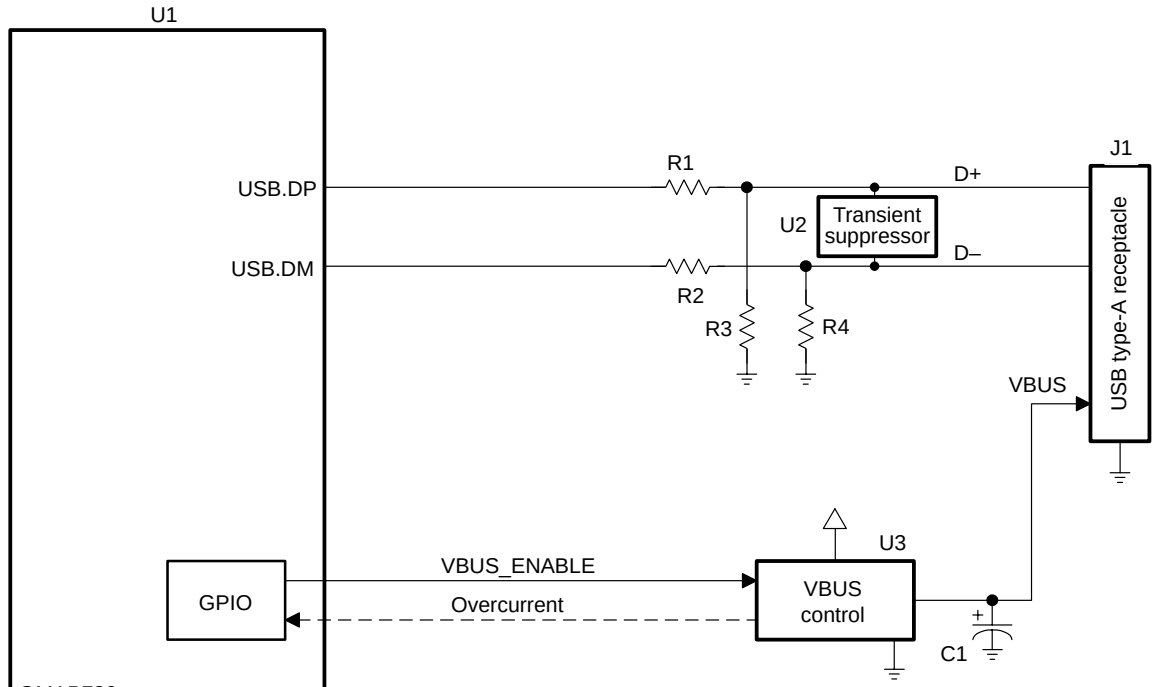
Some USB transceivers implement a 6-wire connection to the host controller, whereas others require only 4 wires or 3 wires. The OMAP730 device must be properly configured to support the signaling required by the attached transceiver. See Section 10.4.4, *Selecting and Configuring USB Connectivity.*

It is possible to use OTG transceiver connectivity to provide a USB host-only port. The OTG transceiver meets all of the requirements of a USB upstream transceiver with the exception of VBUS current sourcing. OTG transceiver VBUS current is typically limited to a maximum value that is lower than the minimum required by the USB specification for a Type-A receptacle. Systems that use an OTG transceiver to control a host-only Type-A receptacle must provide some VBUS source that is capable of meeting USB specification current requirements. Systems that implement a standard Type-A USB receptacle and use an OTG transceiver must ground the OTG transceiver ID input. If the OTG transceiver is used to monitor VBUS for a standard Type-A USB receptacle, a series resistor is recommended between the OTG transceiver VBUS pin and the connector so that the OTG transceiver cannot drive any significant current onto VBUS.

USB Host Connections Using the OMAP730 Integrated USB Transceiver

The OMAP730 integrated USB transceiver can provide connectivity for a host-controller port. Figure 10–62 shows a way to connect the integrated USB transceiver as a host-only port.

Figure 10–62. USB Host Connections Using the OMAP730 Integrated USB Transceiver



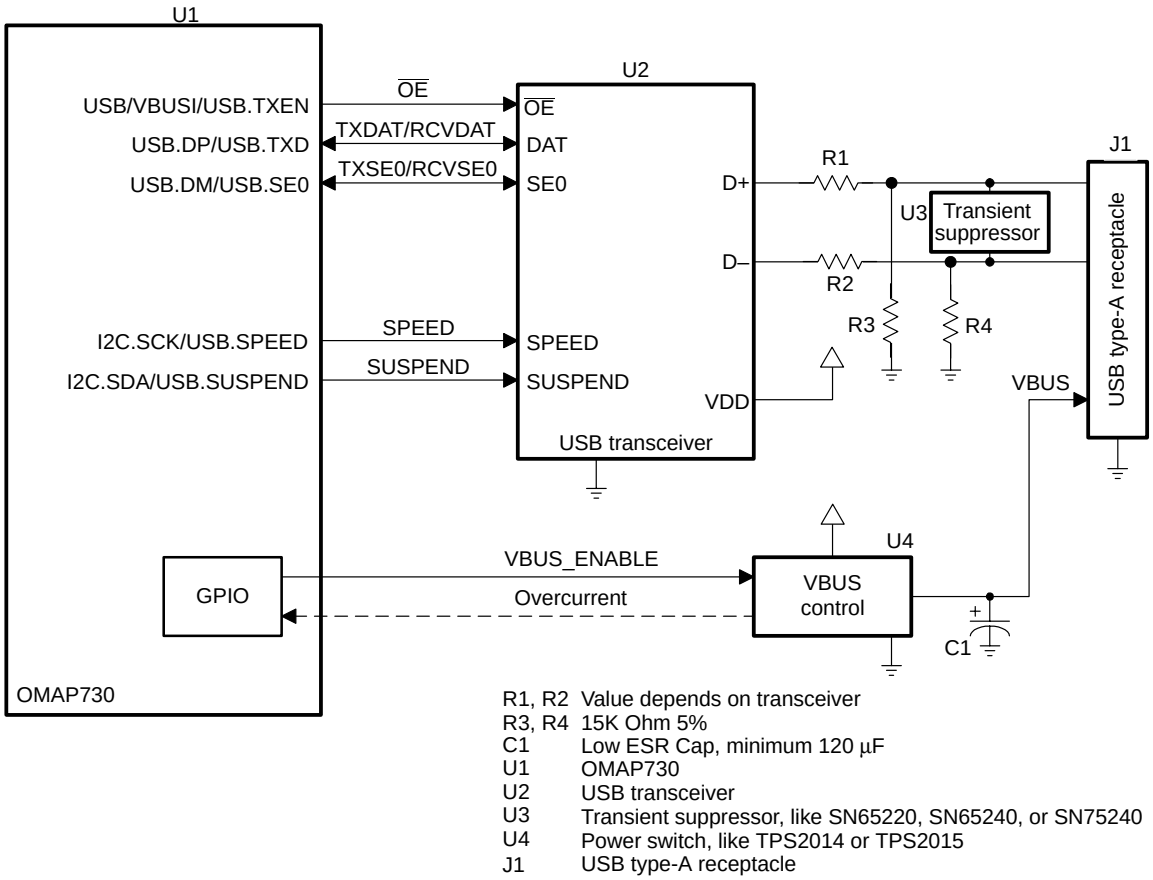
R1, R2 Value depends on transceiver
R3, R4 15K Ohm 5%
C1 Low ESR Cap, minimum 120 µF
U1 OMAP730
U2 Transient suppressor, like SN65220, SN65240, or SN75240
U3 Power switch, like TPS2014 or TPS2015
J1 USB type-A receptacle

Proper initialization of the OMAP730 device to support this mode of operation requires proper setting of the top-level pin multiplexing for pins USB.DP and USB.DM and selection of an HMC_MODE value, which routes a host controller port to pin group 0. OTG_SYSCON_1.USB0_TRX_MODE must be set to 3 to allow proper operation of the integrated USB transceiver.

OMAP730 integrated pulldowns for pins USB.DP and USB.DM do not meet the USB specifications for D+ and D– pulldowns. The external resistors shown in Figure 10–62 must be implemented when using USB.DP and USB.DM for a host connection.

Pin Group 1 USB Host Using 3-Wire USB Transceiver

Figure 10–63. USB Host Connections On USB Pin Group 1 Using 3-Wire Transceiver



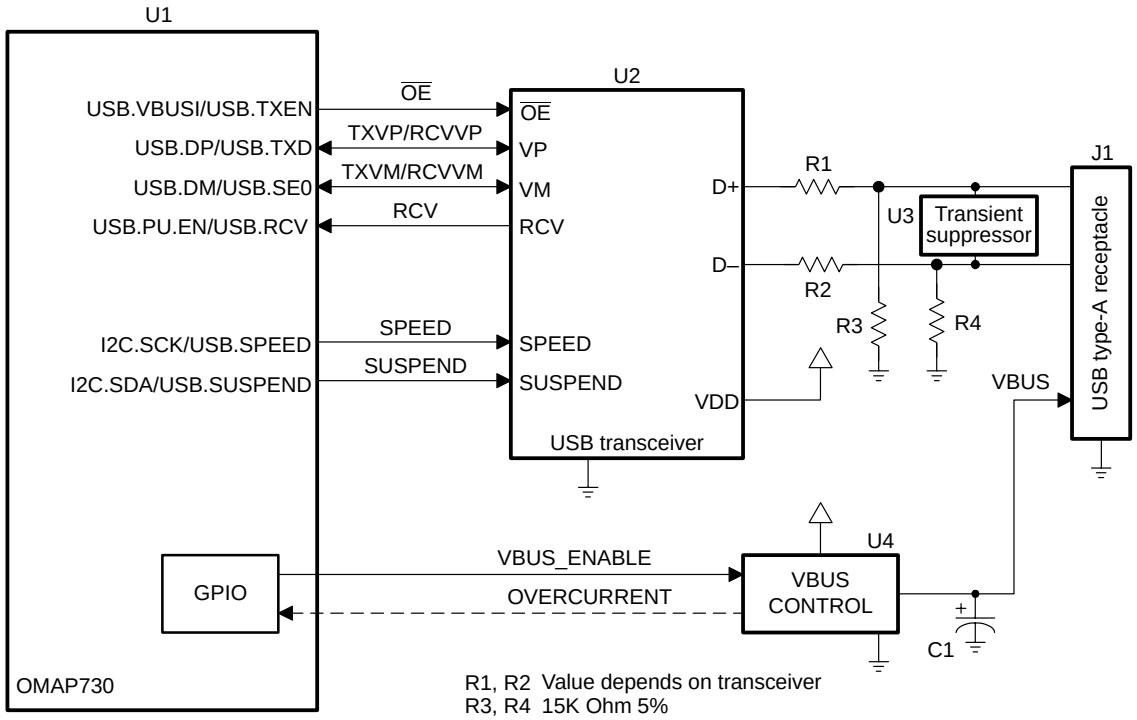
Proper initialization of the OMAP730 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❏ USB.VBUSI/USB.TXEN
- ❏ USB.DP/USB.TXD
- ❏ USB.DM/USB.SE0
- ❏ I2C.SCK/USB.SPEED
- ❏ I2C.SDA/USB.SUSPEND

HMC_MODE must be set to a value that routes a host controller port to USB pin group 1. OTG_SYSCON_1.USB1_TRX_MODE must be set to 2 to allow proper bidirectional operation of the 3-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins USB.DP/USB.TXD and USB.DM/USB.SE0.

Pin Group 1 USB Host Using 4-Wire USB Transceiver

Figure 10–64. USB Host Connections on USB Pin Group 1 Using 4-Wire Transceiver



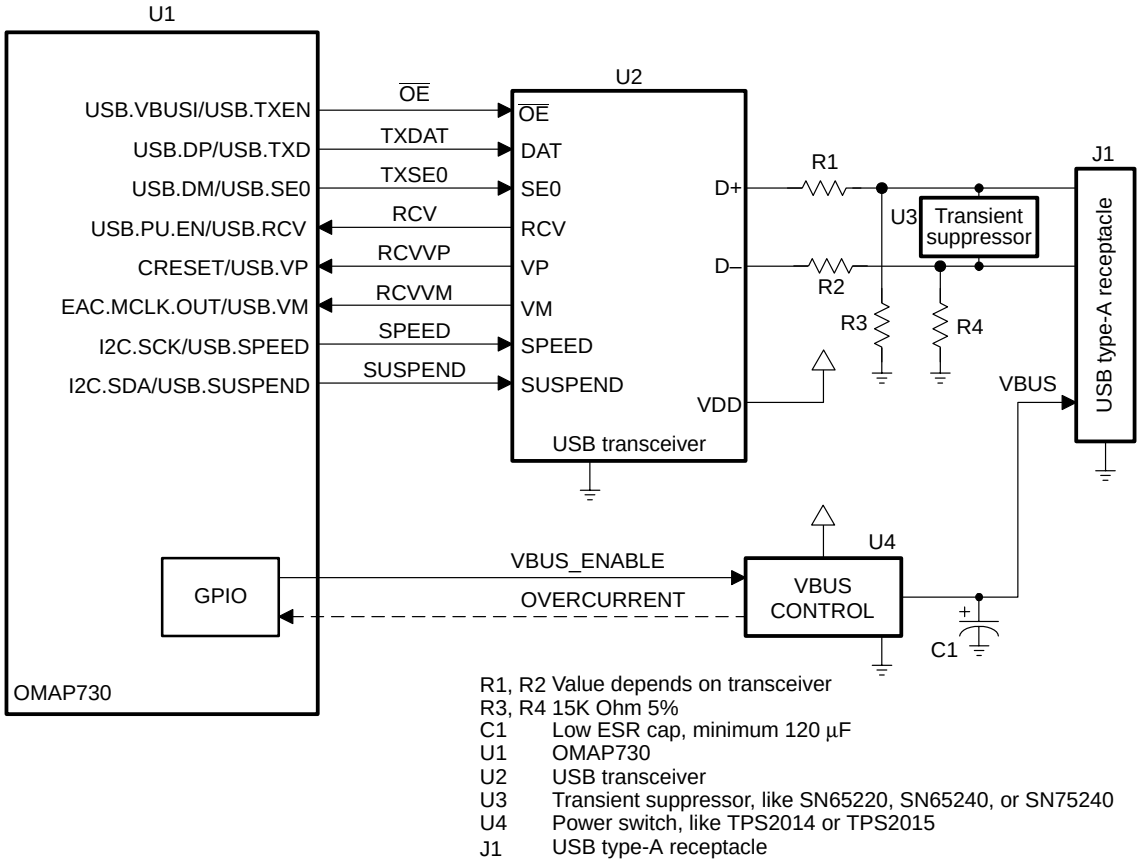
Proper initialization of the OMAP730 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❑ USB.VBUSI/USB.TXEN
- ❑ USB.DP/USB.TXD
- ❑ USB.DM/USB.SE0
- ❑ USB.PU.EN/USB.RCV
- ❑ I2C.SCK/USB.SPEED
- ❑ I2C.SDA/USB.SUSPEND

HMC_MODE must be set to a value that routes a host controller port to USB pin group 1. OTG_SYSCON_1.USB1_TRX_MODE must be set to 1 to allow proper bidirectional operation of the 4-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins USB.DP/USB.TXD and USB.DM/USB.SE0.

Pin Group 1 USB Host Using 6-Wire USB Transceiver

Figure 10–65. USB Host Connections on USB Pin Group 1 Using 6-Wire OTG Transceiver



Proper initialization of the OMAP730 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❏ USB.VBUSI/USB.TXEN
- ❏ USB.DP/USB.TXD
- ❏ USB.DM/USB.SE0
- ❏ CRESET/USB.VP
- ❏ EAC.MCLK.OUT/USB.VM
- ❏ USB.PU.EN/USB.RCV
- ❏ I2C.SCK/USB.SPEED
- ❏ I2C.SDA/USB.SUSPEND

HMC_MODE must be set to a value that routes a host controller port to USB pin group 1. OTG_SYSCON_1.USB1_TRX_MODE must be set to 3 to allow proper unidirectional operation of the 6-wire USB transceiver.

10.4.8 USB Function Controller Connectivity With USB Transceivers

To provide a robust USB solution, a system that provides a non-OTG USB device controller port must implement certain features. These features include a USB type-B or mini-B receptacle, VBUS power detection, transient suppression, a controllable pullup resistor to the D+ or D- line, and USB–compatible upstream port transceiver. These elements are shown in Figure 10–66.

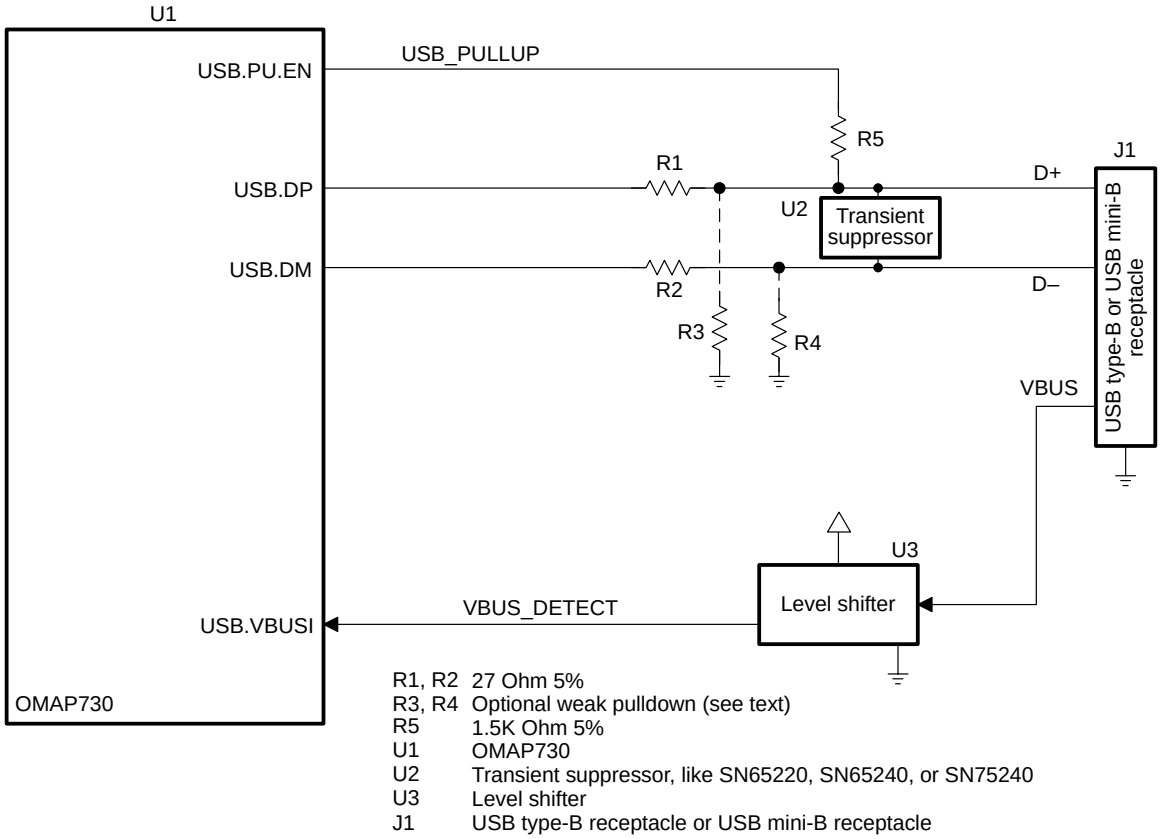
Optional weak pulldown resistors can be used to hold the D+ and D– signals at voltages below the USB transceiver VIL level when there is no USB host connected to the USB Type B connector. By keeping D+ and D– voltages below VIL, the USB transceiver IDDQ can be reduced. Choice of value for the pulldown resistors must be made carefully so that the circuit meets the requirements of the *USB Specification*.

The OMAP730 USB function controller only supports implementation as a full-speed USB device. As such, the pullup resistor must be connected to the D+ signal to indicate implementation of a full-speed USB device.

Some USB transceivers implement a 6-wire connection to the host controller, whereas others require only 4 wires or 3 wires. Figure 10–66 through Figure 10–77 show the signal connectivity options for the integrated transceiver and external 3-, 4-, and 6-wire USB transceivers on each of the available OMAP730 USB pin groups.

Pin Group 0 USB Device

Figure 10–66. USB Device Connections Using OMAP730 Integrated USB Transceiver

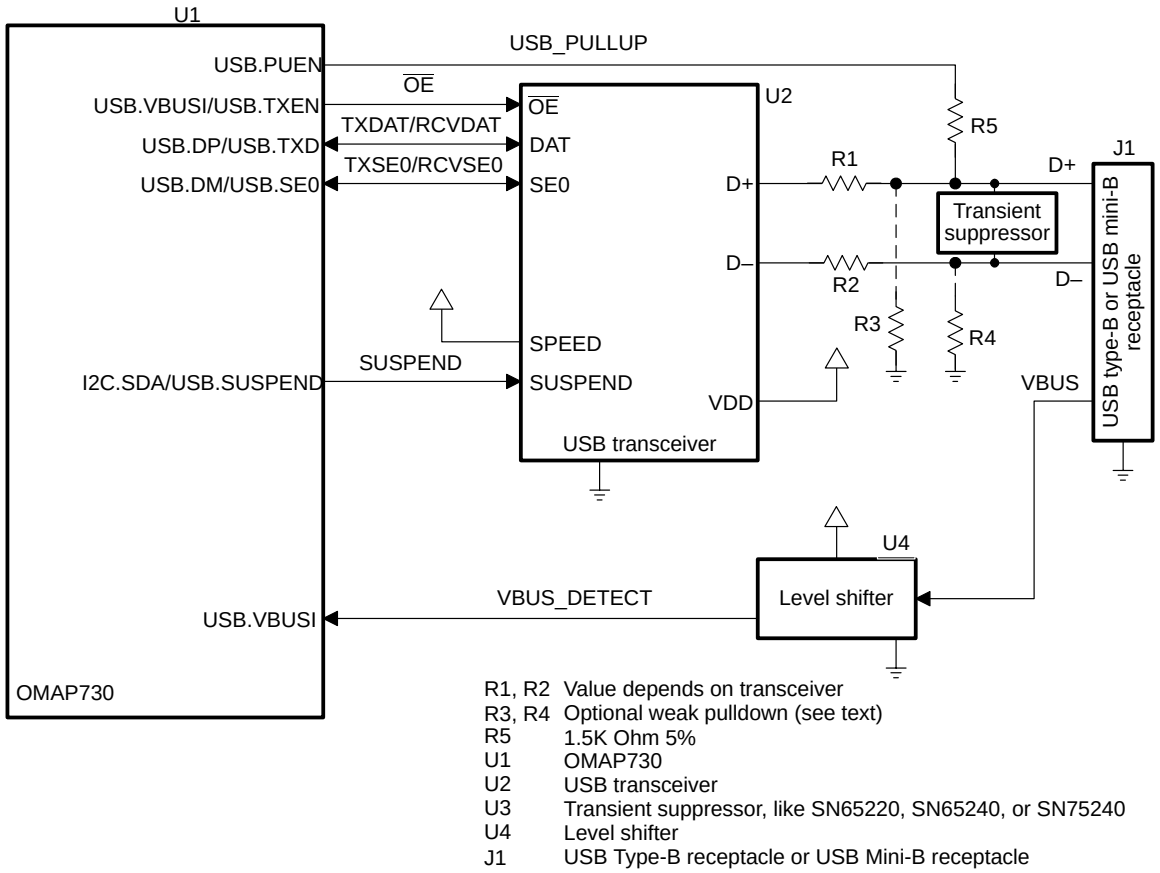


Proper initialization of the OMAP730 to support this mode of operation requires proper setting of the top-level multiplexing for USB.DP and USB.DM and selection of an HMC_MODE value that routes the USB device controller to pin group 0. OTG_SYSCON_1. USB0.TRX_MODE must be set to 3 to allow proper operation of the integrated USB transceiver.

When OTG_SYSCON_1.OTG_PADEN is 1, the status of GPIO0/ USB.VBUS is automatically provided to the USB device controller via hardware mechanisms. When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only sees the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown), and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

Pin Group 1 USB Device Using 3-Wire USB Transceiver

Figure 10–67. USB Device Connections on USB Pin Group 1 Using 3-Wire Transceiver



Proper initialization of the OMAP730 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❏ USB.VBUSI/USB.TXEN
- ❏ USB.DP/USB.TXD
- ❏ USB.DM/USB.SE0
- ❏ I2C.SDA/USB.SUSPEND

HMC_MODE must be set to a value that routes the OMAP730 USB device controller to USB pin group 2. OTG_SYSCON_1.USB1_TRX_MODE must be set to 2 to allow proper bidirectional operation of the 3-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins USB.DP/USB.TXD and USB.DM/USB.SE0.

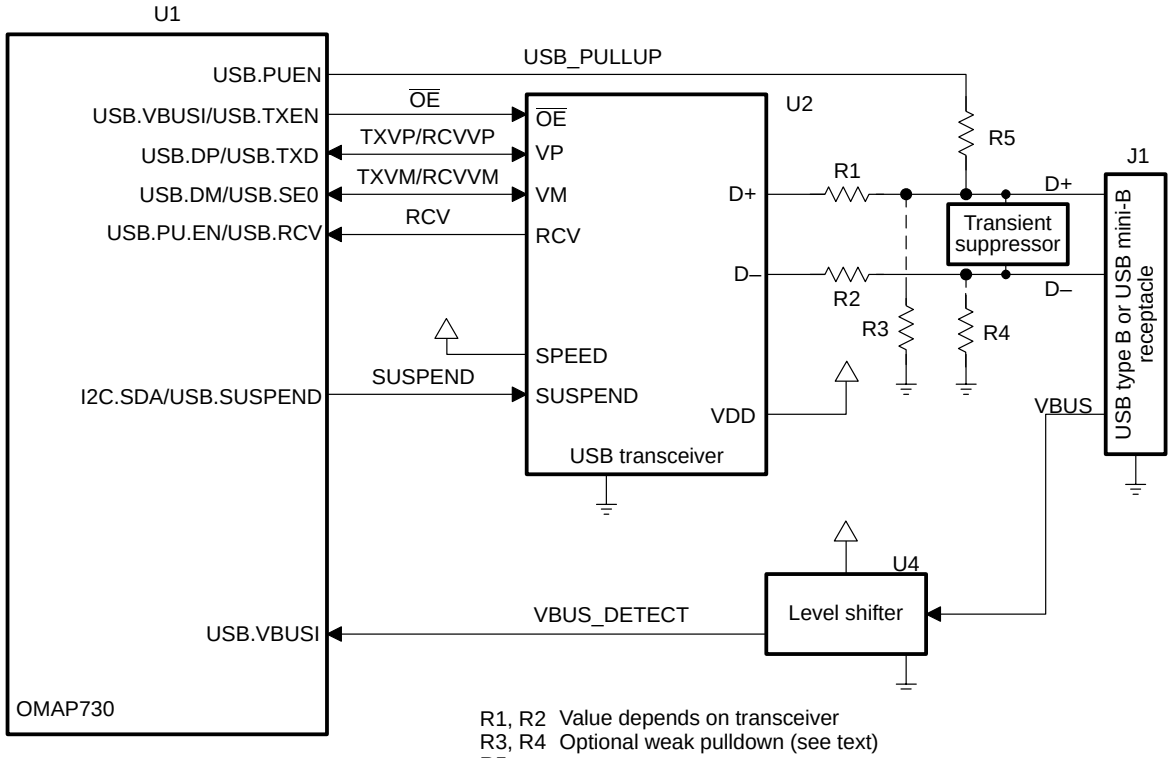
A USB OTG transceiver can be used to provide USB device-only functionality instead of a USB transceiver, with appropriate I^2C and interrupt connectivity. In this case, OTG_SYSCON_2.OTG_PADEN must be 0 and software passes VBUS validity information from the USB OTG transceiver to the device controller by reading VBUS status via the I^2C link and updating the value sent to the device controller via OTG_CTRL.BSESSVLD. The OTG transceiver ID pin must be left unconnected, and OTG functionality is not available. Because the

OTG transceiver provides integrated D+ pullup capability, suspend mode control, and D+/D- pulldown controls, those hardware connections shown can be removed when an OTG transceiver is used. System software must then control those features via the OTG transceiver I^2C register set.

When OTG_SYSCON_2.OTG_PADEN is 1, the status of GPIO0/ USB.VBUS is automatically provided to the USB device controller via hardware mechanisms (when HMC_MODE provides USB device functionality to USB pin group). When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only ses the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown), and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

Pin Group 1 USB Device Using 4-Wire USB Transceiver

Figure 10–68. USB Device Connections on USB Pin Group 1 Using 4-Wire Transceiver



R1, R2 Value depends on transceiver
R3, R4 Optional weak pulldown (see text)
R5 1.5K Ohm 5%
U1 OMAP730
U2 USB transceiver
U3 Transient suppressor, like SN65220, SN65240, or SN75240
U4 Level shifter
J1 USB Type-B receptacle or USB Mini-B receptacle

Proper initialization of the OMAP730 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- USB.VBUSI/USB.TXEN
- USB.DP/USB.TXD
- USB.DM/USB.SE0
- I2C.SCK/USB.SPEED
- I2C.SDA/USB.SUSPEND

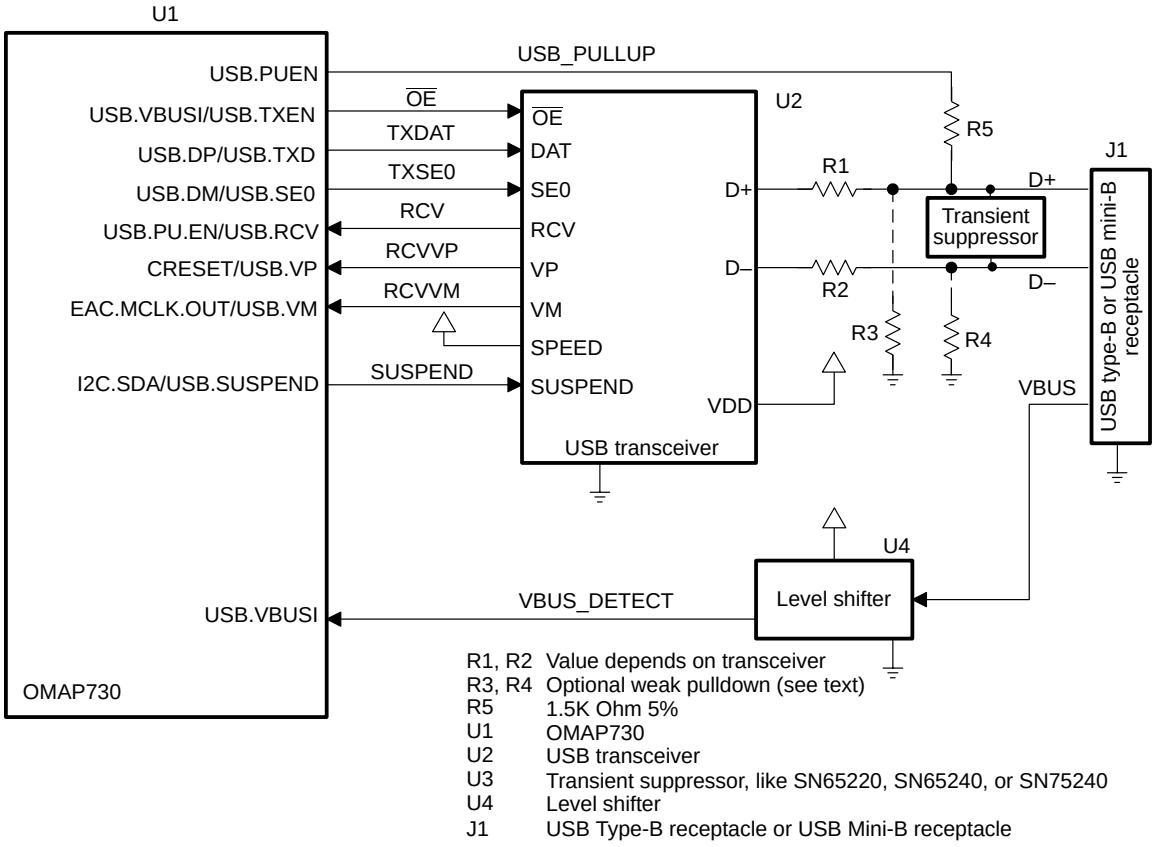
HMC_MODE must be set to a value that routes the USB device controller to USB pin group 1. OTG_SYSCON_1.USB1_TRX_MODE must be set to 1 to allow proper bidirectional operation of the 4-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins USB.DP/USB.TXD and USB.DM/USB.SE0.

A USB OTG transceiver can be used to provide USB device-only functionality instead of a USB transceiver, with appropriate I^2C and interrupt connectivity. In this case, OTG_SYSCON_2.OTG_PADEN must be 0 and software passes VBUS validity information from the USB OTG transceiver to the device controller by reading VBUS status via the I^2C link and updating the value sent to the device controller via OTG_CTRL.BSESSVLD. In this case, the OTG transceiver ID pin is left unconnected and OTG functionality is not available. Because the OTG transceiver provides integrated D+ pullup capability, suspend mode control, and D+/D- pulldown controls, those hardware features shown can be removed when an OTG transceiver is used. System software needs to control those features via the OTG transceiver I^2C register set.

When OTG_SYSCON_2.OTG_PADEN is 1, the status of GPIO0/ USB.VBUS is automatically provided to the USB device controller via hardware mechanisms (when HMC_MODE provides USB device functionality to a USB pin group). When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only sees the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown), and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

Pin Group 1 USB Device Using 6-Wire USB Transceiver

Figure 10–69. USB Device Connections on USB Pin Group 1 Using 6-Wire Transceiver



Proper initialization of the OMAP730 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❑ USB.VBUSI/USB.TXEN
- ❑ USB.DP/USB.TXD
- ❑ USB.DM/USB.SE0
- ❑ CRESET/USB.VP
- ❑ EAC.MCLK.OUT/USB.VM
- ❑ USB.PU.EN/USB.RCV
- ❑ I2C.SDA/USB.SUSPEND

HMC_MODE must be set to a value that routes the USB device controller to USB pin group 1. OTG_SYSCON_1.USB1_TRX_MODE must be set to 3 to allow proper unidirectional operation of the 6-wire USB transceiver.

A USB OTG transceiver can be used to provide USB device-only functionality instead of a USB transceiver, with appropriate I^2C and interrupt connectivity. In this case, OTG_SYSCON_2.OTG_PADEN must be 0 and software passes VBUS validity information from the USB OTG transceiver to the device controller (USB_OTG) by reading VBUS status via the I^2C link and updating the value sent to the device controller via OTG_CTRL.BSESSVLD. In this case, the OTG transceiver ID pin is left unconnected, and OTG functionality is not avail-

able. Because the OTG transceiver provides integrated D+ pullup capability, suspend mode control, and D+/D- pulldown controls, the hardware features shown can be removed when an OTG transceiver is used. System software must control those features via the OTG transceiver I^2C register set.

When OTG_SYSCON_2.OTG_PADEN is 1, the status of GPIO0/ USB.VBUS is automatically provided to the USB device controller via hardware mechanisms (when HMC_MODE provides USB device functionality to a USB pin group). When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only sees the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown), and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

10.4.9 Onboard Transceiverless Connection Using OMAP730 Transceiverless Link

The transceiverless link logic feature of the OMAP730 USB signal multiplexing enables connection of the OMAP730 USB device controller to an external, onboard USB host controller or connection of the OMAP730 USB host controller, without the use of USB transceivers or associated circuitry. When the transceiverless link logic is used, both of the USB transceivers, the series resistors, pullup and pulldown resistors, VBUS switching components, and USB connectors and cables that normally are used between a USB host controller and the downstream USB device controller are removed. The OMAP730 transceiverless link logic feature does not support OTG session request protocol or OTG host negotiation protocol, so it cannot be used for a transceiverless OTG link.

The transceiverless link logic signaling system is not suitable for use across a cable. It is intended only for use when the OMAP730 device is used with an external USB integrated circuit that is on the same board.

When using the transceiverless link logic, six of the external USB integrated circuit pins that typically connect to a USB transceiver connect instead directly to OMAP730 device pins. Signaling on these pins use CMOS levels. Transceiverless link logic can only be used with external devices that support 6-wire transceiver connectivity.

Figure 10–70 and Figure 10–71 show how transceiverless link logic can be compared to a typical USB implementation. Figure 10–70 shows OMAP730 being used as a USB host controller, with the top portion of the diagram showing a transceiver-based solution and the bottom portion showing a transceiverless solution using the OMAP730 transceiverless link logic. Figure 10–71 shows OMAP730 used as a USB device controller, with the top portion of the diagram showing a transceiver-based solution and the bottom portion showing a transceiverless solution using the OMAP730 transceiverless link logic.

Figure 10–70. OMAP730 USB Host Controller Connection—With and Without the OMAP730 Transceiverless Link Logic

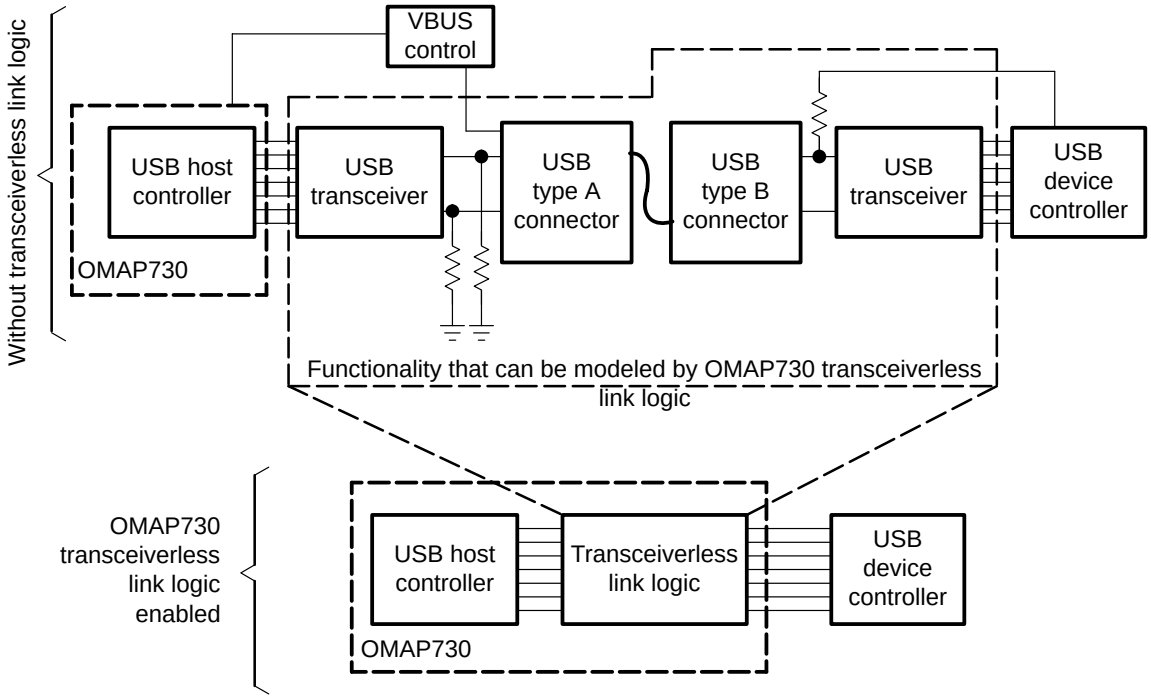
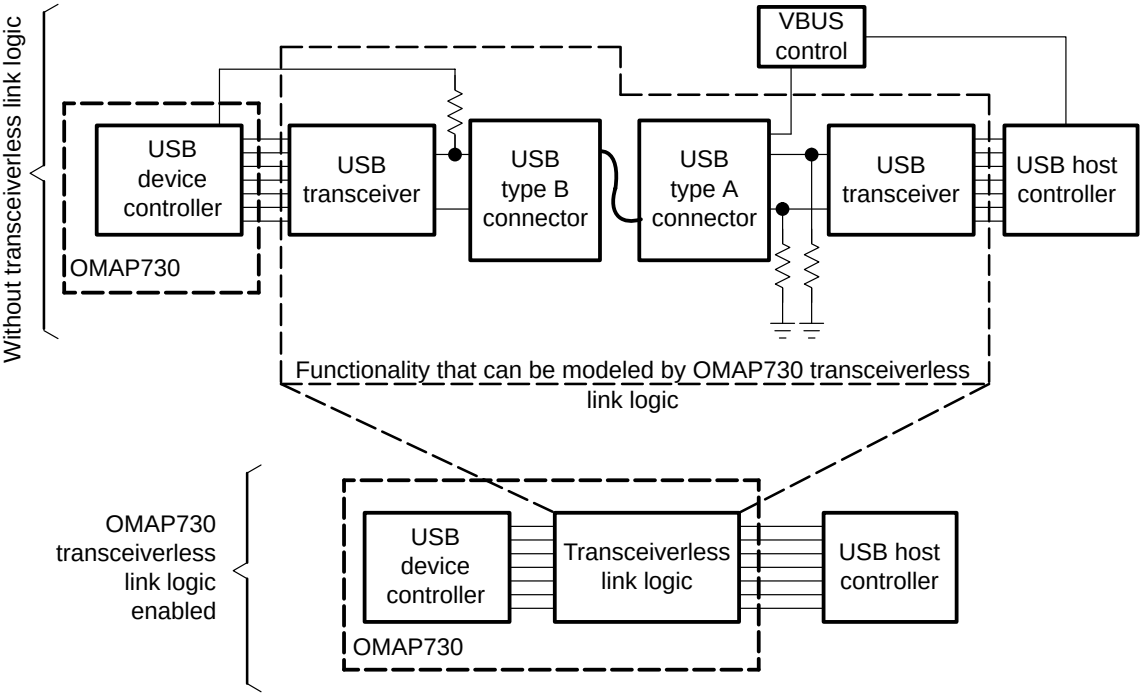


Figure 10–71. OMAP730 USB Device Connection—With and Without the OMAP730 Transceiverless Link Logic



The transceiverless link logic function in the OMAP730 device interprets the transmit control signals from the external USB integrated circuit and similar

signals from the OMAP730 USB host controller or OMAP730 USB device controller and computes the equivalent USB differential pair state. The computed differential pair state is interpreted and the appropriate transceiver output signals are provided to the external USB integrated circuit and to the OMAP730 USB host controller or OMAP730 USB device controller.

Two control bits help determine the proper differential pair state. TLL_ATTACH and TLL_SPEED provide these inputs and are controlled by OMAP730 register bits (see Table 10–70). TLL_ATTACH is used to control when the differential pair models a pullup resistor as is implemented in a transceiver-based USB device. TLL_SPEED is used solely to determine whether the modeled pullup resistor is modeled on the internal version of D+ (for a full-speed transceiverless link) or D- (for a low-speed transceiverless link).

Caution

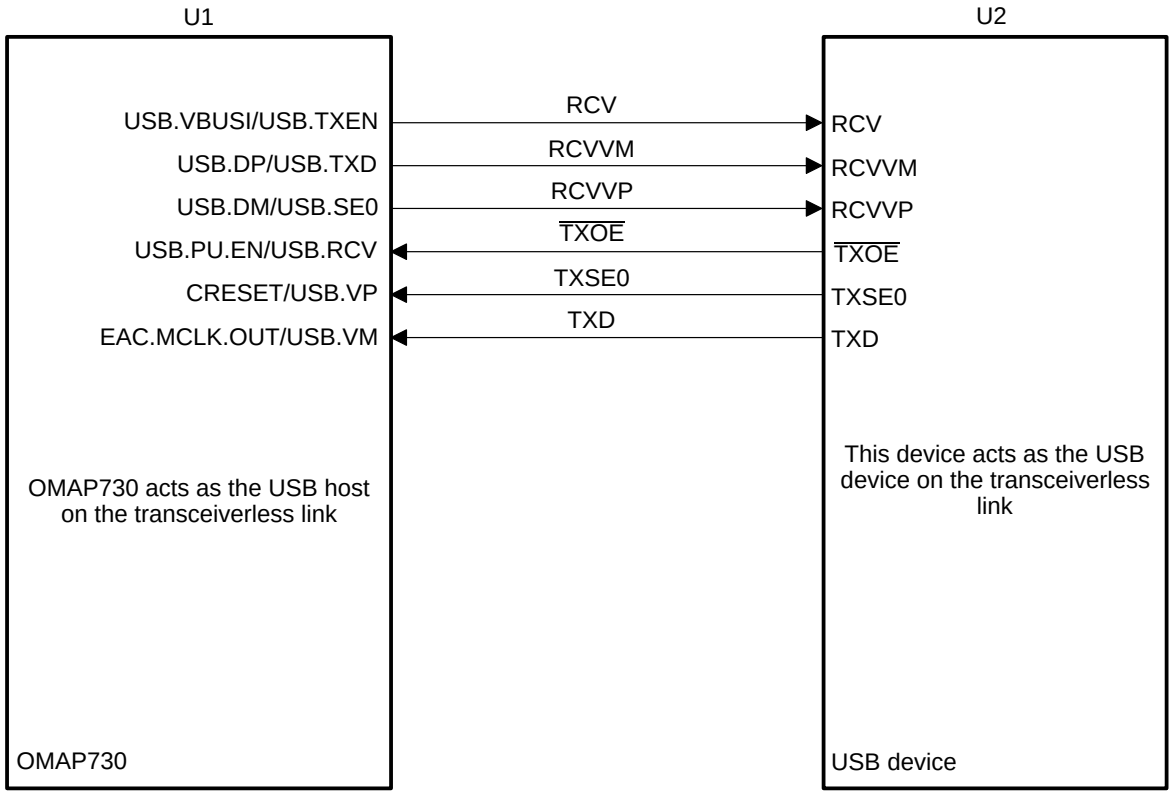
The OMAP730 USB device cannot operate as a low-speed USB device. Do not set TLL_SPEED to 0 when HMC_MODE connects the OMAP730 USB device controller to the transceiverless link logic.

The transceiverless link logic cannot be used as part of an OTG connection to an on-board OTG device. An OTG connection always requires an OTG transceiver. Some modes of operation simultaneously support a transceiver-based OTG connection and an additional transceiverless link logic connection to an on-board downstream USB device. OMAP730 does not provide any modes where both a transceiverless link logic connection to an on-board USB host and a transceiver-based OTG connection are simultaneously available.

USB Host With Transceiverless Link Logic (TXD/TXSE0)

Figure 10–72 shows connectivity when OMAP730 acts as the USB host controller on a transceiverless link connection to an on-board USB device controller.

Figure 10–72. OMAP730 as USB Host on Transceiverless Link Using TXD/TXSE0 Signaling



Proper initialization of the OMAP730 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

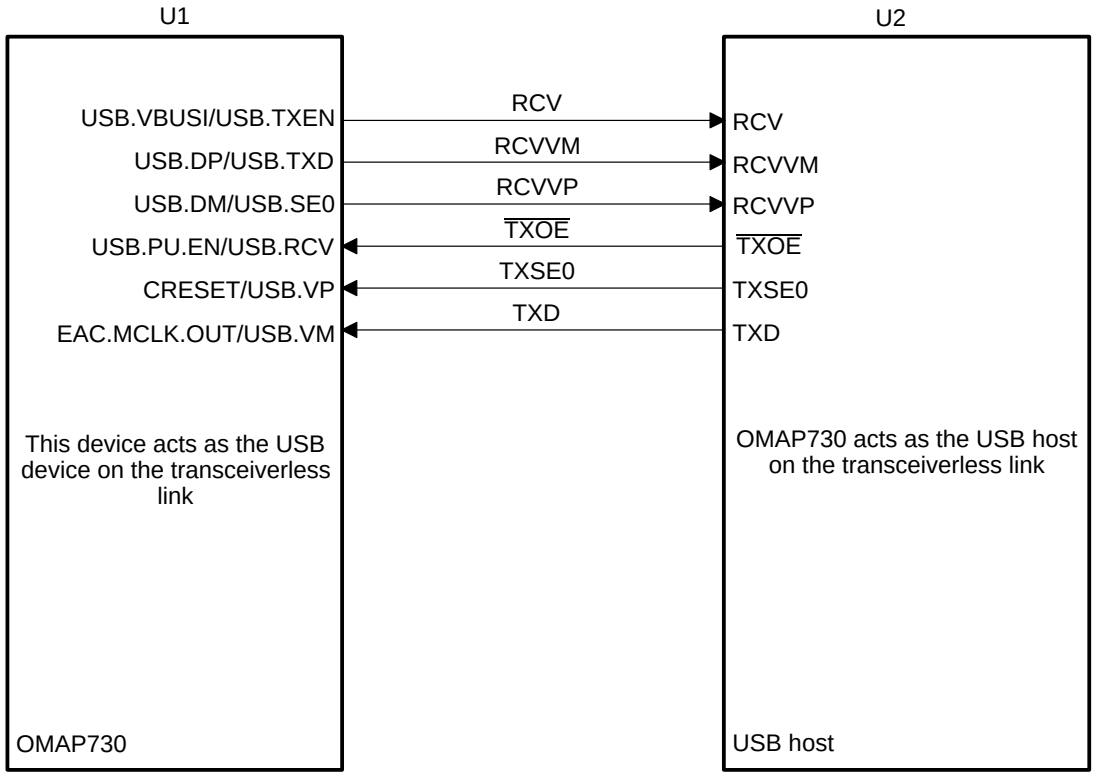
- ❏ USB.VBUSI/USB.TXEN
- ❏ USB.DP/USB.TXD
- ❏ USB.DM/USB.SE0
- ❏ USB.PU.EN/USB.RCV
- ❏ CRESET/USB.VP
- ❏ EAC.MCLK.OUT/USB.VM

HMC_MODE must be set to a value that routes a host controller port through the TLL (in TXD/TXSE0 mode) to USB pin group 2. OTG_SYSCON_1.USB2_TRX_MODE must be set to 3 to allow proper operation of the transceiverless link logic connection.

USB Device With Transceiverless Link Logic (TXD/TXSE0)

Figure 10–73 shows connectivity when OMAP730 acts as the USB device on a transceiverless link connection to an on-board USB host controller.

Figure 10–73. OMAP730 as USB Device Controller on Transceiverless Link Using TXD/TXSE0 Signaling



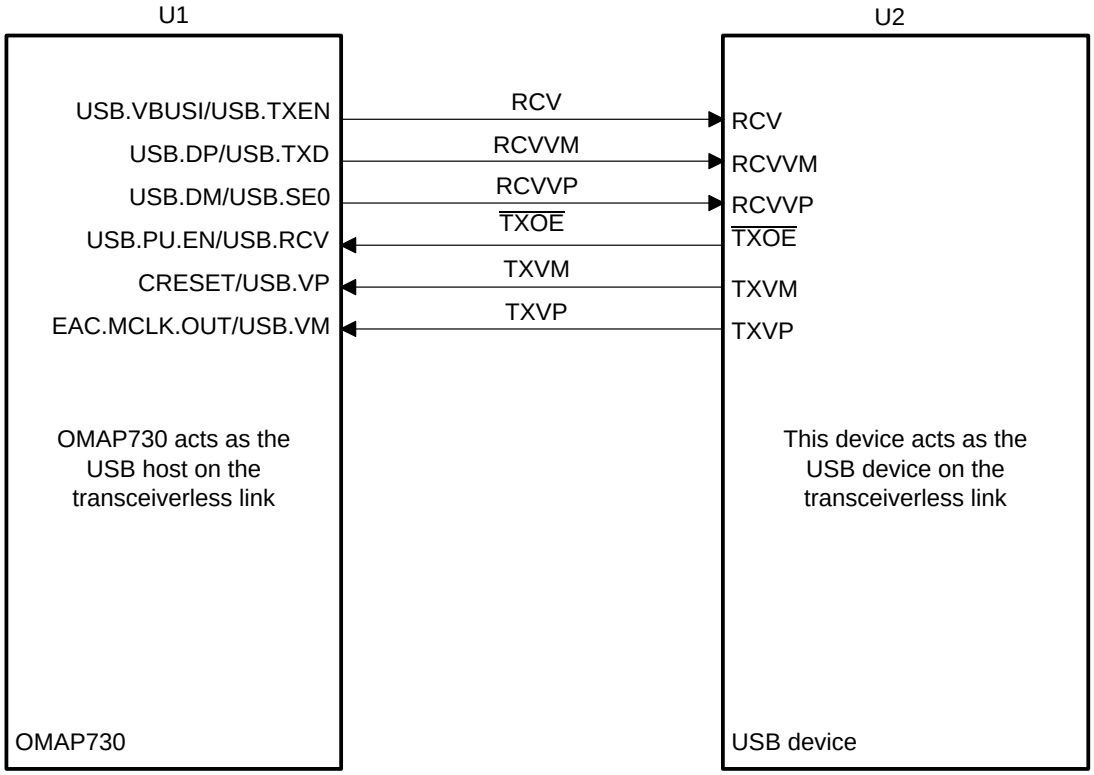
Proper initialization of the OMAP730 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❏ USB.VBUSI/USB.TXEN
- ❏ USB.DP/USB.TXD
- ❏ USB.DM/USB.SE0
- ❏ USB.PU.EN/USB.RCV
- ❏ CRESET/USB.VP
- ❏ EAC.MCLK.OUT/USB.VM

HMC_MODE must be set to a value that routes the USB device controller through the TLL (in TXD/TXSE0 mode) to USB pin group 2. OTG_SYSCON_1.USB2_TRX_MODE must be set to 3 to allow proper operation of the transceiverless link logic connection. TLL_SPEED must be 1 when using the OMAP730 USB device controller with transceiverless link logic.

USB Host With Transceiverless Link Logic (TXVP/TXVM)

Figure 10–74. OMAP730 as USB Host Controller on Transceiverless Link Using TXVP/TXVM Signaling



Proper initialization of the OMAP730 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

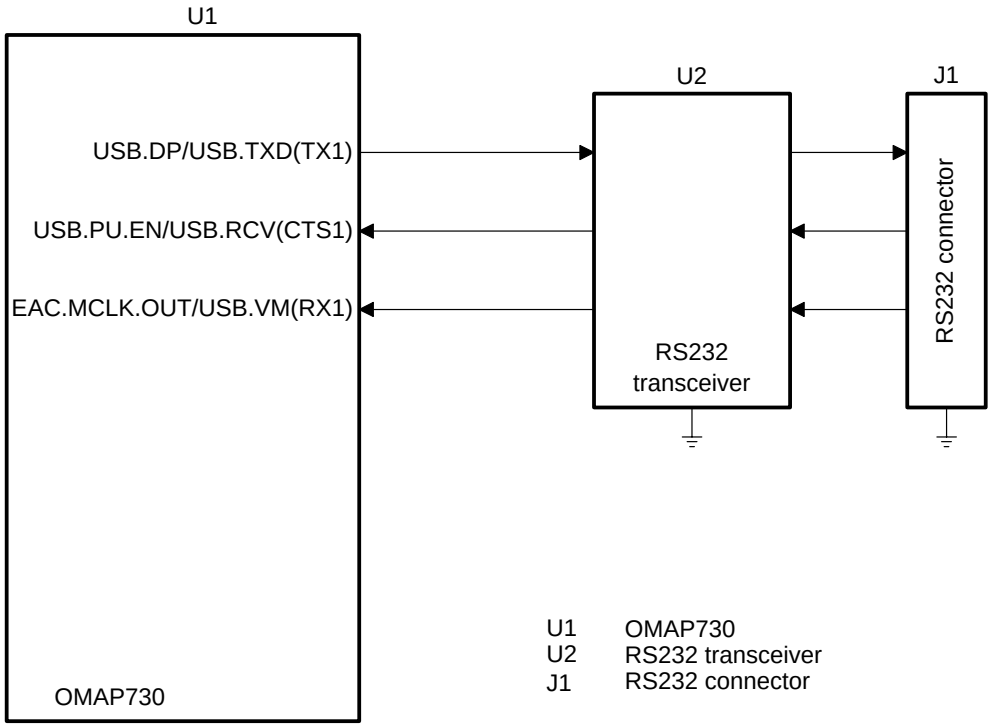
- ❏ USB.VBUSI/USB.TXEN
- ❏ USB.DP/USB.TXD
- ❏ USB.DM/USB.SE0
- ❏ USB.PU.EN/USB.RCV
- ❏ CRESET/USB.VP
- ❏ EAC.MCLK.OUT/USB.VM

HMC_MODE must be set to a value that routes a host controller port through the TLL (in TXVP/TXVM mode) to USB pin group 2. OTG_SYSCON_1.USB2_TRX_MODE must be set to 3 to allow proper operation of the transceiverless link logic connection.

UART 1 on USB Pin Group 2 Pins

When configured for some HMC_MODE values, the USB signal multiplexing mechanism routes UART1 signals to some OMAP730 USB pin group 1 pins (see Figure 10–75).

Figure 10–75. External Connectivity When USB Pin Group 1 Configured for UART1 Signalling



Proper initialization of the OMAP730 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

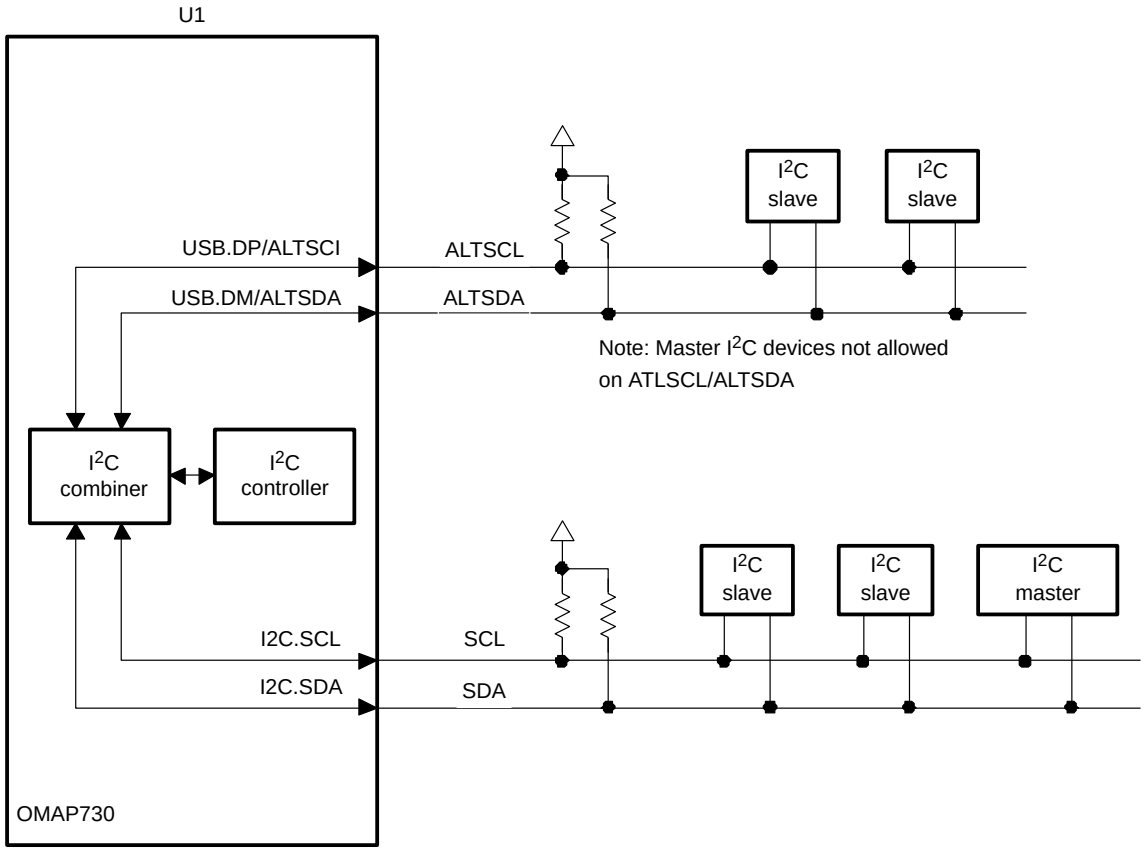
- ❏ USB.DP/USB.TXD
- ❏ EAC.MCLK.OUT/USB.VM
- ❏ USB.PU.EN/USB.RCV

HMC_MODE must be set to a value that routes the UART1 to USB pin group 1. OTG_SYSCON_1.USB1_TRX_MODE must be set to 3 to allow proper operation of the 3-wire connection to the external transceiver.

I²C on Pin Group 0

OMAP730 supports a mode in which the OMAP730 I²C signals are brought out via USB pin group 0 pins. In this mode of operation, top-level pin multiplexing is configured to provide the SCL signal on the USB.DP pin and the SDA signal on the USB.DM pin. This is configured using the USB_TRANSCEIVER_CTRL.CONF_USB0 register, (see Figure 10–76). You must also set USB_TRANSCEIVER_CTRL.CONF_USB0_ISOLATE to 1 to ensure that the I²C signals have no effect on the OMAP730 USB controllers.

Figure 10–76. I²C on USB Pin Group 0



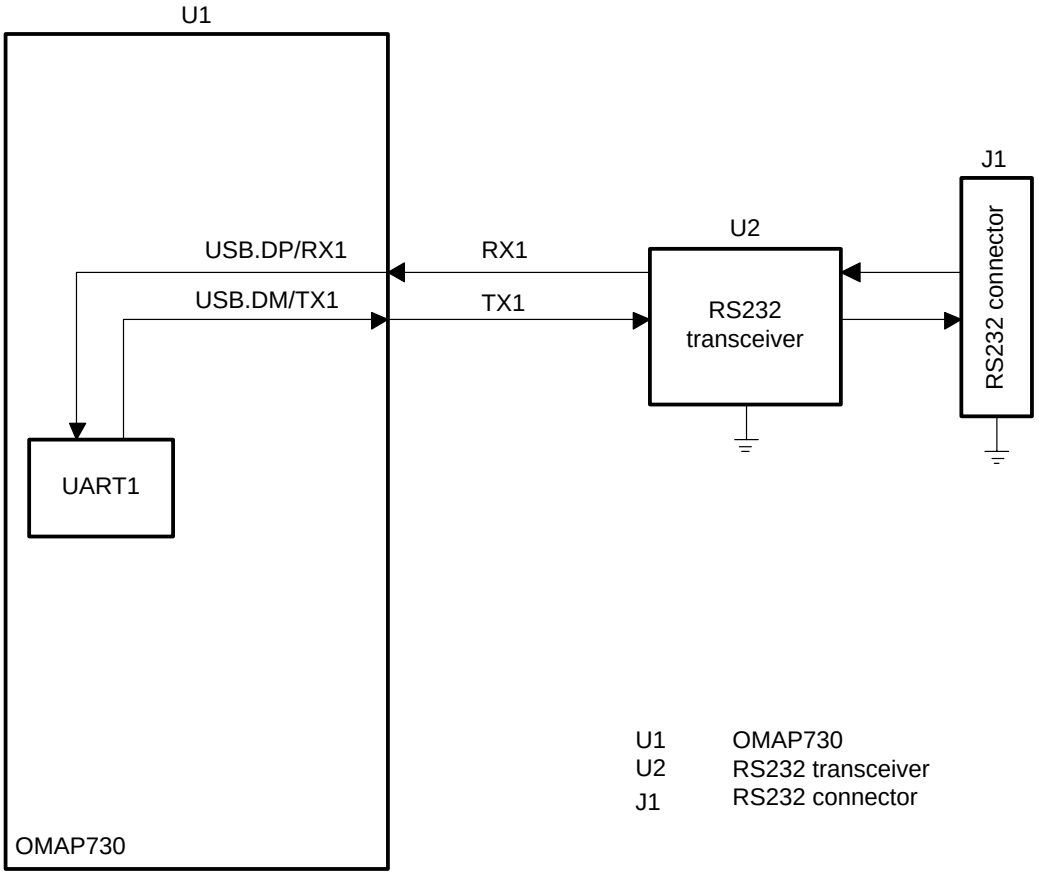
This mode of operation does not allow mastering I²C devices on the I²C link on pins USB.DP and USB.DM. Mastering I²C is allowed on SCL and SDA. A master I²C device connected externally to SCL and SDA is only able to access the I²C slaves on SCL and SDA or the OMAP730 I²C controller slave function. An external master I²C device on SCL and SDA cannot access I²C slaves on ALTSCL and ALTSDA.

This mode of operation implements only one I²C controller. OMAP730 internally combines ALTSCL and SCL, and ALTSDA and SDA. This means that it is not possible to implement I²C slaves with identical I²C addresses on both links.

UART1 on Pin Group 0

OMAP730 supports a mode in which the OMAP730 UART1.TX and UART1.RX signals are brought out via USB pin group 0 pins. In this mode of operation, top-level pin multiplexing is configured to provide the RX input on USB.DP/UART1.RX and provide the TX output on USB.DM/UART1.TX. This is configured using the USB_TRANSCEIVER_CTRL.CONF_USB0 register (see Figure 10–77). When using this mode, set USB_TRANSCEIVER_CTRL.CONF_USB0_ISOLATE to 1 to ensure that the UART1 signals have no effect on the OMAP730 USB controllers.

Figure 10–77. UART1 on USB Pin Group 0



10.4.10 Conflicts Between USB Signal Multiplexing and Top-Level Multiplexing

When OMAP730 top-level signal multiplexing selects non-USB functionality for a pin but USB signal multiplexing is set to use that pin as an output, the signal from the USB signal multiplexing is ignored and the source selected by the OMAP730 top-level signal multiplexing is used.

When OMAP730 top-level signal multiplexing selects non-USB functionality for a pin but the USB signal multiplexing is set to use that pin as an input, the OMAP730 top-level signal multiplexing presents a low level to the USB signal multiplexer.

It may be useful to select a HMC_MODE value that brings some USB signals to the OMAP730 top-level signal multiplexing, but then set the top-level signal multiplexing to ignore those USB signals.

10.4.11 OMAP730 USB Hardware Considerations

10.4.11.1 VBUS Power Switching for USB Type A Host Receptacles

The USB specification places several VBUS requirements on USB hosts, including current capability, droop, and other important characteristics. Circuits that meet the USB specification requirements can be implemented using Texas Instruments devices such as the TPS2014 and TSP2015 power distribution switch devices. Further information, including data sheets and application notes, can be found at the Texas Instruments web site.

Although an OTG transceiver has the ability to power VBUS within the OTG specification limits for an OTG dual role device, it may not have sufficient current sourcing capability to meet the *USB 1.1 Specification*, which requires far greater output current on VBUS. If an OTG downstream port must also support a downstream non-OTG USB device (via a standard-A receptacle to mini-A plug as defined in the OTG specification), a VBUS switch as mentioned above can be used to meet those requirements.

10.4.11.2 Transient Suppression for USB Connectors

It is important to provide transient suppression for USB connectors. Electrostatic discharge that occurs when a user connects or disconnects a USB cable can have a dramatic effect on a system if not suppressed. Texas Instruments offers several devices for transient suppression on USB connections, such as the SN65220, SN65240, and SN75240 universal serial bus port transient suppressor devices. Further information, including data sheets and application notes, can be found at the Texas Instruments web site.

10.4.11.3 VBUS Monitoring for USB Function Controller

A USB function controller must be capable of monitoring the VBUS voltage provided by the upstream USB host controller. The OMAP730 device provides the input pin USB.VBUS, which is provided to the OMAP730 USB function controller (when HMC_MODE provides USB device functionality to a USB pin group). This input is a CMOS input that is not rated for the full VBUS range defined by the USB specification. An external signal level shifter is required to convert the VBUS signal range to a range suitable for the OMAP730 USB.VBUS pin.

10.4.11.4 USB D+ Pullup Enable for USB Function Controller

When using a USB signal multiplexing mode that provides USB function controller signals to OMAP730 pins, the OMAP730 top-level pin multiplexing options lead to several possible USB pullup implementations.

When the USB.PUEN pin is set for top-level multiplexing configuration 0, the USB.PUEN pin is driven low when the pullup is active and is driven high when

the pullup is inactive. In this mode of operation, an external inverter or an external 3-statable device can be used to provide a nominal 3.3-V signal to the supply end of the USB D+ pullup.

When the USB.PUEN pin is set for top-level multiplexing configuration 1, the USB.PUEN pin provides a clock output and cannot be used to control the USB pullup.

When the USB.PUEN pin is set for top-level multiplexing configuration 3, the USB.PUEN pin is driven high when the pullup is active and is driven low when the pullup is inactive. In this mode of operation, the pullup resistor can be connected directly between the OMAP730 USB.PUEN and the D+ signal.

10.4.11.5 MPU_BOOT Signal Sharing

The OMAP730 device implements shared functionality on the I2C.SDA/USB.SUSPEND pin. The MPU_BOOT pin is sampled at hardware reset. If low, the MPU processor boots from memory connected to CS0 on the EMIFS; if high, the MPU processor enters the boot overlay mode, causing it to boot from memory connected to CS3 on the EMIFS. After reset, the pin can be configured for other functionality, such as the USB1.SUSP output. The MPU_BOOT signal has an internal pulldown resistor that is enabled by default. The boot overlay mode requires an external pullup resister. The internal pulldown can be disabled in the OMAP configuration registers.

10.4.11.6 USB D+, D- Pulldown for USB Function Controller

System implementations that use the OMAP730 USB function controller and are sensitive to supply current requirements can implement weak pulldown resistors on the USB D+ and D- signals associated with the USB Type-B receptacle. When there is no host controller attached upstream of the USB Type-B receptacle, the undriven D+ and D- wires can float to voltages that cause excessive current consumption by the USB transceiver. Weak pulldowns can help prevent this problem. Selection of pulldown resistors depends on transceiver characteristics, D+ pullup resistor implementation, and board layout, and must be designed to meet USB D+ and D- signal requirements. The OMAP730 integrated USB transceiver includes programmable weak pulldowns:

- When USB_TRANSCEIVER_CTRL.CONF_USB0_PWRDN_DN is 0, the internal weak pulldown is enabled for OMAP730 pin USB.DM.
- When USB_TRANSCEIVER_CTRL.CONF_USB0_PWRDN_DP is 0, the internal weak pulldown is enabled for OMAP160 pin USB.DP.

Chapter 11

Multimedia Card (MMC/SD/SDIO) Interface

This chapter describes the multimedia card (MMC) interface of the OMAP730 multimedia processor.

Topic **Page**

11.1 MMC Overview

The multimedia card/secure data/secure digital IO (MMC/SD/SDIO) host controller provides an interface between a local host, such as a microprocessor unit (MPU) or digital signal processor (DSP), and either an MMC or SD memory card, plus up to four serial flash cards. The host controller handles MMC/SD/SDIO or serial port interface (SPI) transactions with minimal local host intervention. Figure 11–1 shows an overview of the system.

The host controller supports the following combination of external devices:

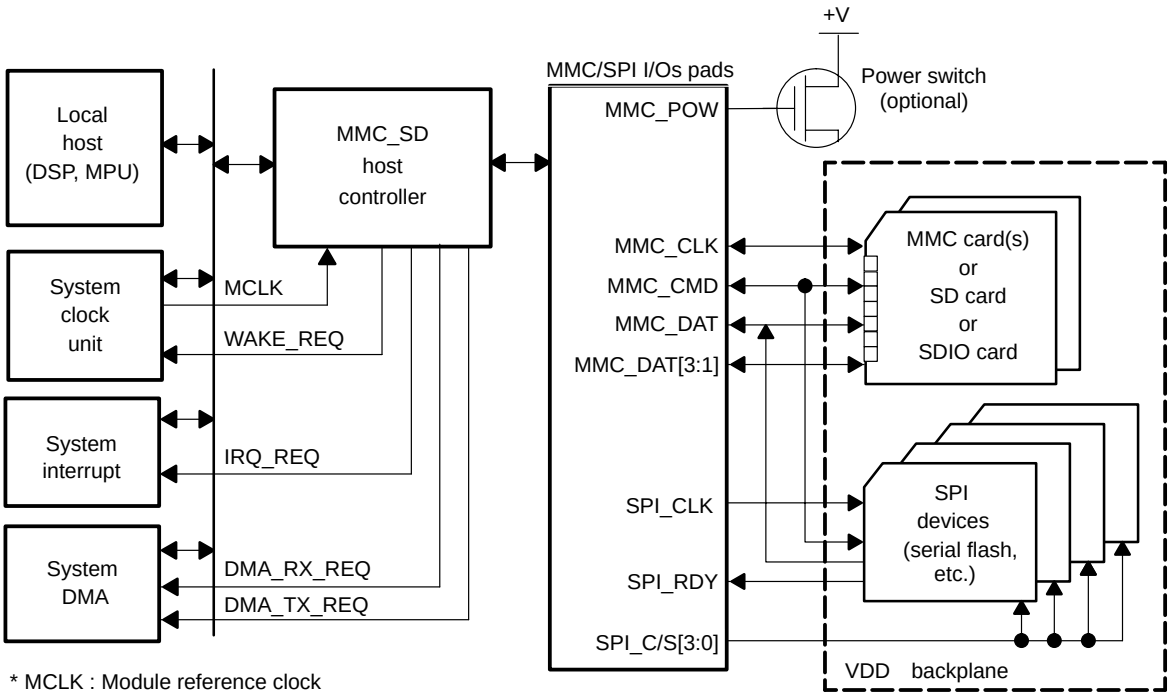
- One or more MMC memory cards sharing the same bus, plus up to four devices with 8-bit SPI protocol interface (serial flash memories)
- One SD memory card or SDIO card, plus up to four devices with 8-bit SPI protocol interface

Other combinations, such as two SD cards or one MMC card plus one SD card, are not supported through a single controller.

The application interface manages transaction semantics. The MMC/SD/SDIO host controller handles the MMC/SD protocol at the transmission level, including data packing, adding the cyclic redundancy check (CRC) and start/end bits, and checking for syntactical correctness. It also supports SD mode wide-bus width.

The application interface can send every MMC/SD/SDIO command and either poll for the status of the adapter, wait for an interrupt request, which is sent back in case of exceptions, or warn of the end of the operation. The application interface reads card responses and flag registers, and masks interrupt sources individually. These operations are performed by reading and writing control registers. The MMC/SD/SDIO module also supports two direct memory access (DMA) channels.

Figure 11–1. MMC/SD/SDIO System Overview



11.1.1 MMC/SD/SDIO Host Controller Features

The main features of the controller are:

- Full compliance with MMC command/response sets as defined in *The Multimedia Card-System Specification*, MMCA Technical Committee, Version 3.1, June 2001
- Full compliance with SD command/response sets as defined in *SD Memory Card Specification–Part 1, Physical Layer Specification*, SD Group, Version 1.0, March 2000, and *Supplementary Notes–Part 1, Physical Layer Specification*, SD Group, June 2000
- Full compliance with SDIO command/response sets and interrupt/read-wait mode as defined in *SDIO Card Specification Part E1*, SDIO Working Group, Version 1.0, October 2001
- Flexible architecture that allows support for new command structure
- Separate SPI interface with four CS. Provides support for up to four serial flash devices
- Built-in 64-byte FIFO for buffered read or write
- 16-bit-wide access bus to maximize bus throughput
- Low-power design
- Wide-interrupt capability
- Programmable clock generation

- Two DMA channels
- Big- /little-endian mode for data

Known limitations:

- No built-in hardware support for error correction codes (ECC)

11.1.2 MMC/SD Host Controller Signal Pads

The signal pads listed in Table 11–1 describe the physical interface between the driving IC (the transceiver) and the target MMC/SD memory cards, SDIO device, or serial flash memories.

The transceiver provides a dc-level adaptation function between the controller core and the target devices. It can be integrated either on-chip with the controller or implemented off-chip (system-dependent issue).

Table 11–1. Signal Pads

Name	Type	Pullup	Reset Value	Description
MMC.CLK	Out	–	0	MMC/SD/SDIO card CLK signal Only active during active command to MMC/SD/SDIO card using MMC or SPI protocols
MMC.CMD/ SPI.DO (SPI_SO)	In/out	Yes	Input	MMC/SD card CMD signal in MMC/SD mode SPI serial out signal in SPI modes (output—goes to serial-in of target device(s))
MMC.DAT0 (SPI_SI)	In/out	Yes	Input	MMC card DAT signal or SD/SDIO card DAT[0] signal in MMC/SD mode SPI serial in signal in SPI modes (input—comes from serial-out of target device(s))
MMC.DAT1 (SDIO_IRQ)†	In/out	Yes	Input	SD/SDIO card DAT[1] signal Interrupt (IRQ) for SDIO card (SD and SPI protocol)
MMC.DAT2 (SDIO_RW)†	In/out	Yes	Input	SD/SDIO card DAT[2] signal Read wait (RW) for SDIO card
MMC.DAT3 (MMC_CS, SD_CD)¶	In/out	Yes	Input	SD/SDIO card DAT[3] signal Chip-select (CS) for MMC/SD/SDIO cards using SPI protocol Chip detect (CD) for SD/SDIO cards
SPI.CLK‡	Out	–	0	SPI CLK signal Only active in SPI mode during active SPI transfers, except when MMC_CLK is selected
SPI_C/Sn[3:0]‡	Out	–	b1111	Four SPI chip-select signals. Active low Only active in SPI mode during active SPI transfers
SPI.RDY‡	In	Yes	Input	SPI ready/busy signal When low, denotes a busy condition. Only active in SPI mode during active SPI transfers
MMC_POW§	Out	–	0	MMC/SD cards on/off power supply control When high, denotes power-on condition

† Optional signals. Only needed for SD/SDIO cards.
‡ Optional signals. Only needed for devices with SPI interfaces (serial flash, additional MMC, SD, or SDIO cards).
§ Optional signal. Only needed if power supply (VDD) of cards or other SPI devices is to be switched on and off in the application.
¶ Optional signals. Only needed for SD/SDIO cards or for MMC card operated in SPI mode.

11.1.3 MMC.CLK and SPI.CLK Signal ac Characteristics

The core internally gates the MMC or SPI clock signals to be active only during a valid transaction to the selected target device (memory cards or serial flash). The duty cycle of the clock depends on the clock division factor and the polarity setting.

Figure 11–2. MMC.CLK and SPI.CLK Signal ac Characteristics

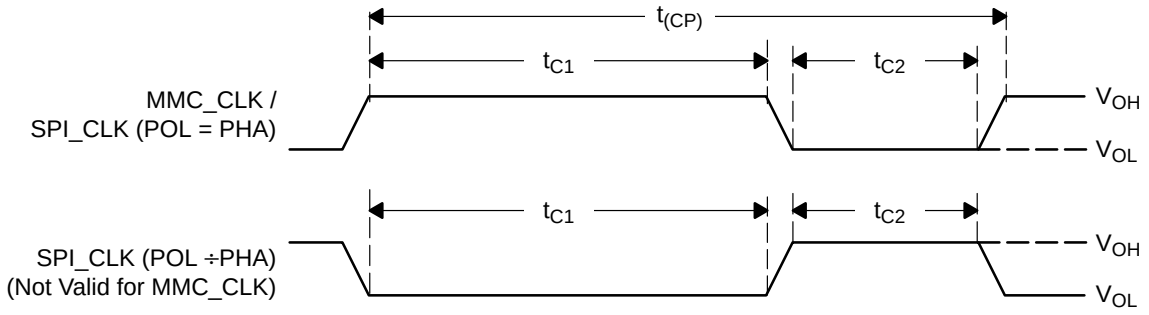


Table 11–2. MMC.CLK and SPI.CLK Signal ac Parameters

Parameter	Description	Min	Max	Unit
t_{CP}	Clock period	40	–	ns
t_{C1}	MMC mode: Clock high time	–	–	ns
	SPI mode: Clock high time (POL = PHA), clock low time (POL ≠ PHA)			
t_{C2}	MMC mode: Clock low time	–	–	ns
	SPI mode: Clock low time (POL = PHA), clock high time (POL ≠ PHA)			

The clock period and the high and low times specified in Table 11–2, as well as all timings in subsequent pages, are controller capabilities.

The real clock period must be computed as a function of the system reference clock and adjusted to the target device used in the application. All derived timings must be checked against selected target device specifications.

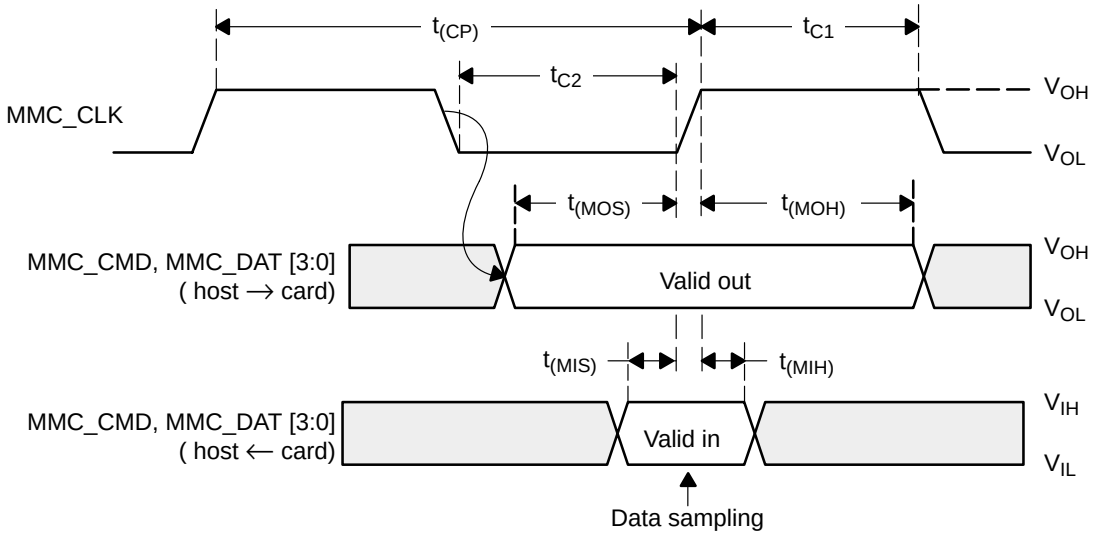
For example: MMC card: max 20 MHz (min 50 ns); SD card: max 25 MHz (min 40 ns); SPI serial flash: max 13 MHz (min 77 ns)

11.1.4 MMC/SD/SDIO Modes—Interface Signal ac Characteristics

Figure 11–3 depicts the ac characteristics of the interface signals when the interface is configured for MMC/SD/SDIO operation.

SPI-specific output signals (SPI_C/Sn[3:0], SPI.CLK) are held to their inactive state, and SPI-specific input signals are don't care (SPI.RDY).

Figure 11–3. MMC/SD/SDIO ac Characteristics



Data is sampled on the rising edge of the clock.

Table 11–3. MMC/SD/SDIO ac Parameters

Parameter	Description	Min	Max	Unit
t_{MOS}	Data output setup to rising edge of clock	t_{C2}–5	–	ns
t_{MOH}	Data output hold to rising edge of clock	t_{C1}–5	–	ns
t_{MIS}	Data input setup to rising edge of clock	4	–	ns
t_{MIH}	Data input hold to rising edge of clock	4	–	ns

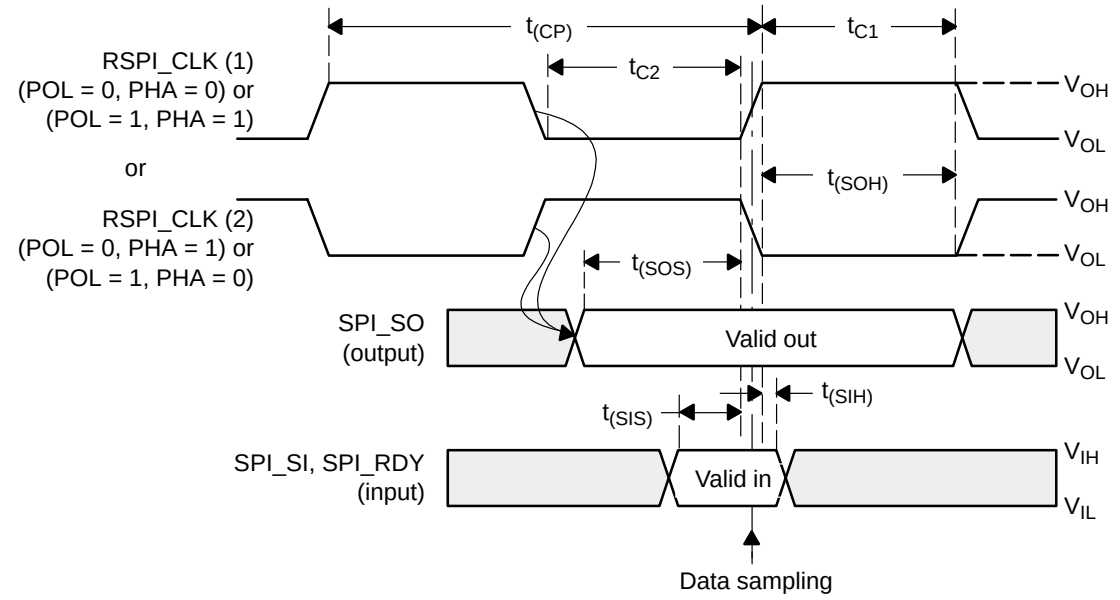
11.1.5 SPI Mode—Interface Signal ac Characteristics

Figure 11–4 depicts the ac characteristics of the interface signals when the interface is configured for SPI operation.

MMC-specific input/output signals (MMC.DAT3) are held to their inactive state.

The SPI interface is master only.

Figure 11–4. SPI ac Characteristics



Data is sampled on the rising or falling edge of the clock, depending on the polarity/phase setting. See Table 11–4.

Table 11–4. SPI ac Parameters

Parameter	Description	Min	Max	Unit
t_{SOS}	Data output set up to rising edge of clock (1) or falling edge of clock (2)	t_{C2}–5	–	ns
	Data output hold to rising edge of clock (1) or falling edge of clock (2)	t_{C1}–5	–	ns
	Data input set up to rising edge of clock (1) or falling edge of clock (2)	16	–	ns
t_{SOH}	Data input hold to rising edge of clock (1) or falling edge of clock (2)	0	–	ns
t_{SIS}				

11.2 MMC Registers

Table 11–5 lists the MMC registers. Table 11–6 through Table 11–36 describe the registers bits.

Table 11–5. MMC Registers

Register	Description	Offset
MMC.CMD/SPI.DO	MMC command	0x00
MMC_ARGL	MMC argument low	0x04
MMC_ARGH	MMC argument high	0x08
MMC_CON	MMC module configuration	0x0C
MMC_STAT	MMC module status	0x10
MMC_IE	MMC system interrupt enable	0x14
MMC_CTO	MMC command time-out	0x18
MMC_DTO	MMC data read time-out	0x1C
MMC_DATA	MMC data access	0x20
MMC_BLEN	MMC block length	0x24
MMC_NBLK	MMC number of blocks	0x28
MMC_BUF	MMC buffer configuration	0x2C
MMC_SPI	MMC SPI configuration	0x30
MMC_SDIO	MMC SDIO mode configuration	0x34
MMC_SYST	MMC system test	0x38
MMC_REV	MMC module revision	0x3C
MMC_RSP0	MMC/SD command response 0	0x40
MMC_RSP1	MMC/SD command response 1	0x44
MMC_RSP2	MMC/SD command response 2	0x48
MMC_RSP3	MMC/SD command response 3	0x4C
MMC_RSP4	MMC/SD command response 4	0x50
MMC_RSP5	MMC/SD command response 5	0x54
MMC_RSP6	MMC/SD command response 6	0x58
MMC_RSP7	MMC/SD command response 7	0x5C
MMC_IOSR	MMC SDIO suspend/resume control	0x60
MMC_SYSC	MMC system control	0x64
MMC_SYSS	MMC system status	0x68
Reserved	Reserved	0x6C–7C

Table 11–6. MMC Command Register (MMC_CMD)

Bit	Name	Description
15	DDIR	Data direction [write, read] This bit specifies whether the data transfer is a read or a write and is valid only if the command type is adtc. This bit has the same polarity as the RD/WR argument bit 0 for a GEN_CMD command (CMD56): 0: Data write 1: Data read Value after reset is low.
14	SHR	Stream command or broadcast host response [normal, stream/host] MMC card only The SD card does not support stream operation or host-generated response. This bit must be set to 1 in two cases: ❑ Associated with adtc command type, if the command is a stream data transfer (read or write). Stream read is a class 1 command (CMD11: READ_DAT_UNTIL_STOP). Stream write is a class 3 command (CMD20: WRITE_DAT_UNTIL_STOP). ❑ Associated with a bc command type, the host generates a 48-bit response instead of a command. It can be used to terminate the interrupt mode by generating a CMD40 response by the core (see Section 4.3, *Interrupt Mode*, *The Multimedia Card–System Specification*). In order for the host response to be generated in open drain mode, MMC_CMD[ODTO] must be set to 1. This bit is value only if the command type is adtc or bc: 0: Normal mode 1: Stream mode (MMC_CMD[TYPE] = adtc), host response (MMC_CMD[TYPE] = bc) Value after reset is low.
13:12	TYPE	Command type [bc, bcr, ac, adtc] Encoded bits that define the type of command passed by the core to the MMC/SD memory card (see Section 4.7.1, *Command Types*, *The Multimedia Card–System Specification*, or *SD Memory Card Specification–Part 1 Physical Layer Specification* and *Supplementary Notes–Part 1 Physical Layer Specification).* 00: Broadcast commands (bc), no response 01: Broadcast commands with response (bcr) 10: Addressed commands (ac), no data transfer 11: Addressed data transfer commands (adtc). *Caution: Also reset the FIFO.* Value after reset is low (both bits).
11	BUSY	Command with busy response This bit must be set to 1 if the response to the command sent is type R1b (R1 + busy): 0: Response without busy (R1, R2, R3, R4, R5, R6) 1: Response with busy (R1b) Value after reset is low.

Table 11–6. MMC Command Register (MMC_CMD) (Continued)

Bit	Name	Description
10:8	RSP	Command responses [no response, R1/R1b, R2, R3, R4, R5, R6, reserved] Encoded bits that define the response for the command that the core passes to the MMC/SD memory card (see Section 4.9, *Responses*, *The Multimedia Card–System Specification*, or *SD Memory Card Specifications–Part 1 Physical Layer Specification* and *Supplementary Notes–Part 1 Physical Layer Specification*). 000: No response 001: R1/R1b (normal response command) 010: R2 (CID, CSD registers) 011: R3 (OCR register) 100: R4 (fast I/O—MMC card only) 101: R5 (interrupt request—MMC card only/IO_RW_DIRECT—SDIO card only) 110: R6 (published RCA response—SD card only) 111: Reserved Value after reset is low (3 bits).
7	INAB	Send initialization stream/data abort command This bit must only be set in two particular cases: ❑ When the card is idle, to send an initialization sequence. This option simplifies the acquisition of new cards. An initialization sequence consists of setting the CMD line to 1 during 80 clock cycles (see Section 6.3, *Power-Up*. *The Multimedia Card–System Specification*, or Section 6.4, *SD Memory Card Specifications–Part 1 Physical Layer Specification* and *Supplementary Notes–Part 1 Physical Layer Specification SD Memory Card Specifications–Part 1 Physical Layer Specification* and *Supplementary Notes–Part 1 Physical Layer Specification*). In this mode, no command is sent to the card and no response is expected. ❑ When the card is in the data transfer stage, to stop or abort an ongoing data transfer. The card is said to be in such state when the previous command was of type adtc and has not yet completed (MMC_STAT[BRS] = 0). A stop or aborted data command: ❑ Freezes the MMC_BLEN[BLEN] and MMC_NBLK[NBLK] values according to the last valid byte written to or read from the card ❑ Sets the MMC_STAT[BRS] status bit as follows: ❑ 0: No action 1: Initialization (80 clock cycles)/data abort command Value after reset is low.

Table 11–6. MMC Command Register (MMC_CMD) (Continued)

Bit	Name	Description
6	ODTO	Card open drain mode/extended command time-out mode This bit has a dual function, depending on the value set in the MMC_SDIO[XDTS] bit. Open drain control function (MMC_SDIO[XDTS] = 0) —MMC card only This bit must be set to 1 if the MMC card bus is operating in open-drain mode during the response phase to the command sent. Typically, it occurs during card identification mode when the card is in idle, ready, or identification state. It is also necessary to set this bit to 1 for a broadcast host response (see MMC.CMD[SHR]). This bit must be set for MMC card commands 1, 2, 3, and 40. For the SD card, this bit must always be kept low because SD cards do not have open drain capability. 0: Push-pull 1: Open-drain Extended command time-out function (MMC_SDIO[XDTS] = 1)—SDIO card only This bit must be set to 1 if the SDIO command response requires a long time-out (typically an IO_RW_DIRECT CMD52). When set, the command time-out is set to the data time-out value (see MMC_DTO[DTO]). When clear, the usual command time-out applies (see MMC_CTO[CTO]). 0: command time-out equals CTO 1: Command time-out equals DTO Value after reset is low.
5:0	INDX	Command index [CMD0, …, CMD63] Binary encoded value from 0 to 63 specifying the command number sent to the card 000000: CMD0 000001: CMD1 … 111111: CMD63 Value after reset is low (all 6 bits).

A write to the MMC command register sends a command to the card.

If the local host accesses this register byte-wise, the card receives the command only after a write access to the least significant (LS) byte (bits 7:0). Therefore, the most significant (MS) byte must always be written first in a byte-accessed situation.

A read has no effect except to return the last command that was sent.

Note: Writes

A write into this register with MMC_CMD[TYPE] = adtc resets the FIFO. Writes with other type of values (bc, bcr, ac) do not affect the FIFO contents. Hence, data must be written into the FIFO after sending a single- or multiple block-write command.

Value after reset is low (all 6 bits).

Table 11–7. System Argument Low Register (MMC_ARGL)

Bit	Name	Description
15:0	ARGL	Command argument bits [15:0]

In the system argument low register, the value after reset is low (all 16 bits).

Table 11–8. System Argument High Register (MMC_ARGH)

Bit	Name	Description
15:0	ARGH	Command argument bits [31:16]

In the system argument high register, the value after reset is low (all 16 bits).

MMC_ARGL and MMC_ARGH specify the 32-bit argument value that is passed with the command. These registers must be initialized before sending the command to the card (write action into the MMC_CMD register). The only exception making a write unnecessary is a command index specifying stuff bits in arguments.

Table 11–9. Module Configuration Register (MMC_CON)

Bit	Name	Description
15	DW	Data bus width SD mode only This bit must be set following a valid SET_BUS_WIDTH command (ACMD6) with the value written in bit 1 of the argument. Prior to this command, the SD card configuration register (SCR) must be verified for the bus width supported by the SD card. 0: 1-bit data width (DAT[0] used) 1: 4-bit data width (DAT[3:0] used, SD card only) Value after reset is low. This bit must always be set to 0 for MMC cards or during SPI transfer. Failing to set this bit correctly results in unpredictable behavior.
14	–	Reserved

Table 11–9. Module Configuration Register (MMC_CON) (Continued)

Bit	Name	Description
13:12	MODE	Operating mode select [MMC/SD, SPI, SYSTEST] These two bits select among MMC/SD, SPI, and SYSTEST modes. In MMC/SD mode, transfers to the MMC/SD/SDIO card follow the MMC protocol. The MMC clock is enabled and the SPI clock is disabled. MMC/SD transfers are operated under the control of the MMC.CMD register. In SPI mode, transfers to as many as four SPI controlled devices (serial flash, MMC/SD/SDIO cards) are supported. SPI transfers are operated under the control of the MMC_SPI register. In SYSTEST mode, the signal pins are configured as general-purpose input/output, and the 64-byte FIFO is configured as a stack memory accessible only by the local host or system DMA. The pins retain their default type (input, output, or in-out). The YSTEST mode is operated under the control of the MMC_SYST register. 00: MMC/SD mode (MMC/SD/SDIO cards using MMC?SD protocol) 01: SPI mode (for serial flash or others SPI slave devices) 10: SYSTEST mode 11: Reserved Value after reset is low (both bits).
11	POWER_UP	Power-up control This bit must be set to 1 before any valid transaction to either MMC/SD or SPI memory cards. When 1, the card is considered powered-up and the controller core is enabled. When 0, the card is considered powered-down (system dependent), and the controller core logic is in pseudo-reset state. That is, the MMC_STAT flags and the FIFO pointers are reset, any access to MMC_DATA[DATA] has no effect, a write into the MMC.CMD register is ignored, and a setting of MMC_SPI[STR] to 1 is ignored. This bit directly controls the MMC_POW signal (if implemented as a device pin): 0: Powered-down/pseudo-reset state 1: Powered-up/usual operation mode Value after reset is low.

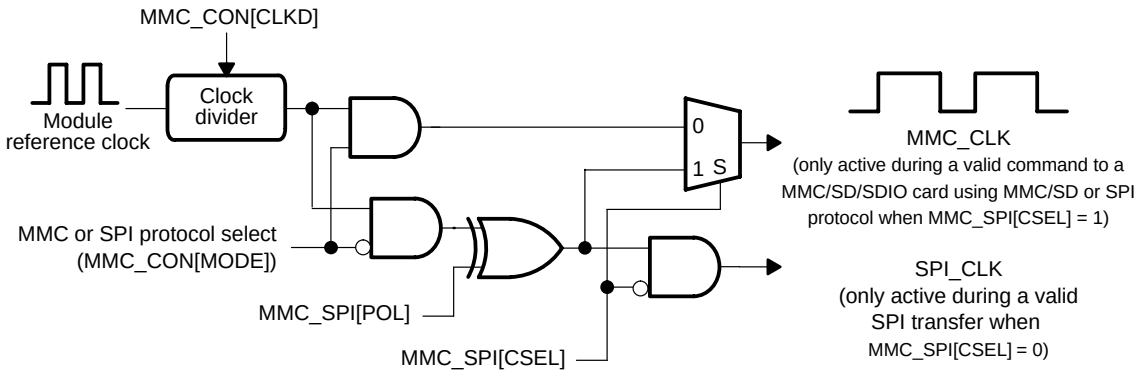
Table 11–9. Module Configuration Register (MMC_CON) (Continued)

Bit	Name	Description
10	BE	Big-endian mode [little, big] When this bit is 0 (default), the FIFO is accessed in little-endian format. In transmit mode, the LS byte (MMC_DATA[7:0]) is transmitted first, and the MS byte (MMC_DATA[15:8]) is transmitted to the card in second position. Conversely, in receive mode, the first or odd byte received (1, 3, 5, …) is stored in the LS byte position, and the second or even byte received is stored in the MS byte position. When the local host sets this bit to a 1, the FIFO is accessed in big-endian format. In transmit mode, the MS byte (MMC_DATA[15:8]) is transmitted first and the LS byte (MMC_DATA[7:0]) is transmitted to the card in second position. Conversely, in receive mode, the first or odd byte received (1, 3, 5, …) is stored in the MS byte position, and the second or even byte received is stored in the LS byte position. 0: Little-endian mode 1: Big-endian mode Value aftrer reset is low.
9:0	CLKD	Clock divider [disabled, 1:1023] These bits define the ratio between a reference clock frequency (system dependent) and the output clock frequency on the CLK pin of either the memory card (MMC or SD) or other 8-bit mode SPI-controlled device. The division factor is equal to the binary-encoded decimal value for values between 1 and 1023. A value of 0 disables the clock: 0x00: Clock disabled 0x01: Ref clock/1 .... 0x3FF: Ref clock/1023 Value after reset is low (all 10 bits).

Note: Active Transfer Phase

The module configuration register must never be written during an active transfer phase. Changing it may result in unpredictable behavior.

Figure 11–5. Clock Control



Notes: 1) During the identification phase, the maximum frequency on the MMC CLK line is 400 kHz (see Section 6.7, *Bus Timing*, *The Multimedia Card–System Specification*, Section 6.8, *SD Memory Card Specifications–Part 1, Physical Layer Specification,* or *Supplementary Notes–Part 1, Physical Layer Specification*). A value of 50 must be set into the frequency ratio register if the reference clock frequency to the MMC/SD host controller is 20 MHz.

2) During the data-transfer phase, the maximum frequency is 20 MHz for the MMC card and 25 MHz for SD cards.

3) The duty cycle of the generated MMC.CLK and SPI.CLK signals depends on the clock divider value (MMC_CON[CLKD]) and on the polarity setting (MMC_SPI[POL]) in SPI mode only. The low- and high times approximate values can be computed using the rules set out in Table 11–10.

4) In MMC/SD mode, the idle value of the MMC.CLK signal is low. In SPI mode, the idle value of either the MMC.CLK (MMC_SPI[CSEL] = 1) or the SPI.CLK (MMC_SPI[CSEL] = 0) is a function of the polarity setting (low if MMC_SPI[POL] = 0, high if MMC_SPI[POL] = 1).

Table 11–10. MMC.CLK/SPI.CLK High/Low Time Computation

MMC_CON[CLKD]	MMC.CLK/SPI.CLK High Time	MMC.CLK/SPI.CLK Low Time
1	REF_CLK_HIGH_TIME	REF_CLK_LOW_TIME
Even ≥ 2	REF_CLK_PER (CLKD/2)	REF_CLK_PER (CLKD/2)
Odd ≥ 3 (POL = PHA)	REF_CLK_PER (TRUNC[CLKD/2] + 1)	REF_CLK_PER (TRUNC[CLKD/2])
Odd ≥ 3 (POL ≠ PHA)	REF_CLK_PER (TRUNC[CLKD/2])	REF_CLK_PER (TRUNC[CLKD/2] + 1)

Notes: 1) REF_CLK_PER is the reference clock period (in ns) to the module (end-system dependent).

2) TRUNC is the truncate function to an integer number (round down).

Example 1:

Module reference clock = 48 MHz (20.83 ns); target is the MMC card.

a) (MMC_CON[CLKD] = 3 (because the MMC card is 20 MHz max)

b) MMC_CLK period = 62.5 ns (> 50 ns OK)

c) Ideal MMC_CLK high time = 41.66 ns (>>10 ns OK)

d) Ideal MMC_CLK low time = 20.83 ns (>>10 ns OK)

Example 2:

Module reference clock = 60 MHz (16.67 ns); target is the 13-MHz serial flash requiring polarity 1 programming.

a) (MMC_CON[CLKD] = 5 (because 13-MHz serial flash)

b) SPI_CLK period = 83.33 ns (> 77 ns OK)

c) Ideal SPI_CLK high time = 33.3 ns (<35 ns FAIL, use (MMC_CON[CLKD] = 6)

d) Ideal SPI_CLK low time = 50 ns (>>35 ns OK)

e) (MMC_CON[CLKD] = 6 (because the clock-high time minimum is 35 ns)

f) SPI_CLK period = 100 ns (> 77 ns OK)

g) Ideal SPI_CLK high time = 50 ns (>>35 ns OK)

h) Ideal SPI_CLK low time = 50 ns (>>35 ns OK)

Table 11–11. Module Status Register (MMC_STAT)

Bit	Name	Description
15	–	Reserved
14	CERR	Card status error in response (see Table 11–12) MMC/SD mode only The core automatically sets this bit when there is at least one error in a response of type R1, R1b, R6, or R5 if enabled. Only bits referenced as type E (error) set a card status error. The error handler must parse the response registers to understand the source of the error. Other responses (type R2, R3, or R4) do not trigger a card status error. This bit has no meaning and always reads 0 in SPI or SYSTEST modes. 0: No action or no error 1: Error occurred Value after reset is low.

Table 11–11. Module Status Register (MMC_STAT) (Continued)

Bit	Name	Description
13	CIRQ	MMC card IRQ received (following CMD40) or SDIO card interrupt MMC card only The core automatically sets this bit when a card is in interrupt mode and exits WAIT_IRQ state (IRQ) by asserting a low level on the CMD line (cards are in open-drain mode). Only Class 9 MMC cards can be put into interrupt mode when in standby state using a GO_IRQ_STATE (CMD40) command. SDIO card only The core automatically sets this bit when an SDIO card has signaled an interrupt on DAT1 line and if the MMC_SDIO[IRQE] bit was set to 1. The interrupt condition is detected in either 1-bit or 4-bit transfer mode and for either MMC/SD or SPI operation mode. SD memory cards do not support interrupt mode. This bit has no meaning and always reads 0 in SPI or SYSTEST mode. 0: No action or idle 1: Card exits IRQ state (MMC card), card interrupt detected (SDIO card). Value after reset is low.
12	OCRB	Operation condition register (OCR) busy (following CMD1 or ACMD41) MMC/SD mode only The core automatically sets this bit after a SEND_OP_COND (CMD1) or a SD_APP_OP_COND (ACMD1) command when one or more cards have not yet completed power-up. When this bit is set, the CMD1/ACMD1 command must be repeated until the card stops responding with a busy condition (a low value on bit 31 of OCR register indicates a busy condition) (see Section 6.3, *Power-Up*, in *The MultiMediaCard–System Specification* or Section 6.4, *SD Memory Card Specifications–Part 1, Physical Layer Specification* or *Supplementary Notes–Part 1, Physical Layer Specification*). This bit has no meaning and always reads 0 in SPI or SYSTEST mode. 0: No action or card powered up 1: OCR busy Value after reset is low.
11	AE	Buffer almost empty The core automatically sets this bit during a write operation to the card (see class 4 block-oriented write command in Section 4.7.3, *Command Classes,* in *The MultiMediaCard–System Specification–Part 1, Physical Layer Specification*) when the level equals or is below the threshold value (in 16-bit words) set in MMC_BUF[AEL]. It indicates that the memory card has emptied the buffer to the specified level and that the local host is able to write more data into the buffer. If the DMA transmit mode is enabled, this bit is never set. Instead, a DMA TX request is generated to the main DMA controller in the system. **NOTE:** The almost-empty status bit and DMA TX request are generated under the same conditions. This bit is set initially when a new block write command is sent to the card. Once the bit is set, the core internally masks a new set condition until the local host has performed MMC_BUF[AEL] 16-bit word write access(es) to the FIFO. 0: No action or buffer is equal or above almost-empty level. 1: Buffer is almost empty. Value after reset is low.

Table 11–11. Module Status Register (MMC_STAT) (Continued)

Bit	Name	Description
10	AF	Buffer almost full The core automatically sets this bit during a read operation to the card (see class 2 block-oriented read commands in Section 4.7.3, *Command Classes,* in *The Multi-MediaCard–System Specification–Part 1, Physical Layer Specification*) when the level equals or is above the threshold value (in 16-bit words) set in MMC_BUF[AFL]. It indicates that the memory card has filled out the buffer to the specified level and that the local host needs to empty the buffer by reading it. If the DMA-receive mode is enabled, this bit is never set. Instead, a DMA RX request is generated to the main DMA controller in the system. **NOTE:** The almost full status bit and DMA RX request are generated under the same conditions. Once set, the core internally masks a new set condition until the local host has performed MMC_BUF[AFL] 16-bit word read access(es) from the FIFO. 0: No action or the buffer is below or equal to the almost full level. 1: Buffer is almost full. Value after reset is low.
9	CRW	Card read wait SDIO card only The core automatically sets this bit when an SDIO card has entered read wait. It indicates that the previous read multiple transfer (CMD53) has been temporarily stalled and that a new command without data stage (such as CMD52) can be sent to the SDIO card. This bit is set on the condition that the core has requested a wait to the card (MMC_SDIO[RW] = 1) and the read-wait mode is enabled (MMC_SDIO[RWE] = 1). The read-wait condition is detected in either 1- or 4-bit transfer mode. 0: No action 1: SDIO card in read-wait Value after reset is low
8	CCRC	Command CRC error MMC/SD mode onlyThe core automatically sets this bit if there is a CRC7 error in the command response (bits 7:1 of response). CRC7 is checked for all command response types (R1 through R6) with the exception of type R3, and conditionally for type R4 if MMC_SDIO[DCR4] = 1. In SPI or SYSTEST modes, this bit has no meaning and always reads 0. 0: No action or no CRC7 error 1: CRC7 error Value after reset is low.

Table 11–11. Module Status Register (MMC_STAT) (Continued)

Bit	Name	Description
7	CTO	Command response time-out (no response) MMC/SD mode only The core automatically sets this bit if the card does not respond to any command requiring a response within the specified number of command time-out clock cycles set in MMC_CTO[CTO]. **NOTE:** If this bit is set after a command time-out, clearing this bit automatically stops the MMC clock and forces the controller FSM to its default state. 0: No action or no command time-out 1: Command time-out Value after reset is low.
6	DCRC	Data CRC error MMC/SD mode only The core automatically sets this bit if there is a CRC16 error in the data phase response following a block read command (single or multiple), or if a 3-bit CRC status differs from a positive 010 token during a block-write command (single or multiple). A token error can be either a data transmission error 101 or a no-CRC response 111 in the case of a programming error (SD card only). Every block of the CRC is checked in a multiple-block transfer. This bit has no meaning and always reads 0 in SPI or SYSTEST mode. 0: No action or no CRC error 1: CRC16 error (read), 3-bit CRC token error (write) Value after reset is low.
5	DTO	Data response time-out (no response) The core automatically sets this bit if the card does not respond within the specified number of data time-out clock cycles (DTO) set in MMC_DTO[DTO]. This bit is also set in SPI mode if the RDY/BUSY signal remains asserted in busy condition for MMC_DTO[DTO] consecutive clock cycles. **NOTE:** If this bit is set after a data time-out, clearing this bit automatically stops the MMC or SPI clock and forces the controller FSM to its default state. This bit has no meaning and always reads 0 in SYSTEST mode. 0: No action or no data time-out 1: Data time-out Value after reset is low.
4	EOFB	Card exit busy state MMC/SD mode only The core automatically sets this bit when the addressed card releases the DAT line from its busy state (low level = busy). This bit can only be set during a programming phase (write operation) to an MMC or SD memory card. This bit has no meaning and always reads 0 in SPI or SYSTEST mode. 0: No action 1: Data line released/exit busy state Value after reset is low.

Table 11–11. Module Status Register (MMC_STAT) (Continued)

Bit	Name	Description
3	BRS	Block received/sent The core automatically sets this bit at the end of a block transfer (read or write). In MMC or SD mode, this bit is set when the block transfer completes without error. If a CRC error occurs, this bit is not set. Instead, a data CRC error is set to 1. For either multiple block or stream transfer, this bit is set only once after the last successful block transfer (when MMC_NBLK[NBLK] decrements down to 0) or until interrupted by a stop command. In SPI mode, this bit is set when either the read or write command completes (MMC_BLEN[BLEN] decrements down to 0). The differences between a DMA and a non-DMA receive operation are: ❑ In non-DMA RX mode, this bit is set after the last byte has been received in the FIFO. At this stage, the FIFO is not empty and must be read by the LH until it is emptied before sending a new command. ❑ In DMA RX mode, this bit is set after both the last byte has been received and the FIFO is empty. This bit has no meaning and always reads 0 in SYSTEST mode. 0: No action 1: Block received/sent Value after reset is low.
2	CB	Card enter busy state MMC/SD mode only The core automatically sets this bit when the addressed card asserts the DAT line to a low level during a programming phase (write operation) to an MMC or SD memory card. For an MMC card only, users can optionally use this interrupt to deselect the card, which continues to program, and select another card. This bit has no meaning and always reads 0 in SPI or SYSTEST mode. 0: No action 1: Data line asserted low/card busy Value after reset is low.

Table 11–11. Module Status Register (MMC_STAT) (Continued)

Bit	Name	Description
1	CD	Card detect on DAT3 MMC/SD mode only The core automatically sets this bit after it has detected a card-detect condition on the DAT3 line and if MMC_SDIO[CDE] = 1. This bit has no meaning and always reads 0 in SPI or SYSTEST mode. 0: No action 1: Card-detect event Value after reset is low.
0	EOC	End of command phase MMC/SD mode only The core automatically sets this bit at the end of a successful command/response sequence or at the end of a command without response. This bit is not set in case of a card status error (MMC_STAT[CERR] = 1), command CRC error (MMC_STAT[CCRC] = 1), or command time-out (MMC_STAT[CTO] = 1). This bit has no meaning and always reads 0 in SPI or SYSTEST mode. 0: No action 1: End of command/response sequence Value after reset is low.

The following are common to all bits:

- The local host can clear a set bit location only by writing a 1 into that bit location. Writing 0 has no effect.
- When the core sets a bit location to 1, the local host receives an interrupt signal if the interrupt was enabled.

Table 11–12. Card Status Error (CERR)

Response Type	Card Status Bits With Error	Response Register Significant Bits	Comments
R1 (MMC, SD, SDIO)	31–26, 24–16, 4^2 3^2, 2^2 (opt)	MMC_RSP7[15–10, 8–0]† MMC_RSP6[4, 3, 2]‡	Bit 4 if MMC_SDIO[C14E] = 1 (SDIO) Bit 3 if MMC_SDIO[C13E] = 1 (SD/app spec) Bit 2 if MMC_SDIO[C12E] = 1 (app spec)
R6 (SD, SDIO)	15–13, 3	MMC_RSP6[15–13, 3]	These correspond to 23, 22, 19, 3 card status errors (SDIO card does not generate error on bit 3 force 0—superset)
R5 (SDIO)	7, 6, 3, 1, 0[3]	MMC_RSP6[15,14, 11, 9, 8]	[3] Only if MMC_SDIO[15] = 1

† These 15 bits can all generate errors (SDIO spec specifies 31, 23–22, 19 whereas others are 0—superset).
‡ These 3 bits can also generate errors if enabled.

Table 11–13 lists the characteristics of the system interrupt enable register.

Table 11–13. System Interrupt Enable Register (MMC_IE)

Bit	Name	Description
15	-	Reserved
14	CERR	Card status error interrupt enable
13	CIRQ	Card IRQ interrupt enable
12	OCRB	OCR busy interrupt enable
11	AE	Buffer almost empty interrupt enable
10	AF	Buffer almost full interrupt enable
9	CRW	Card read wait enable
8	CCRC	Command CRC error interrupt enable
7	CTO	Command response time-out interrupt enable
6	DCRC	Data CRC error interrupt enable
5	DTO	Data response time-out interrupt enable
4	EOFB	Card exit busy state interrupt enable
3	BRS	Block received/sent interrupt enable
2	CB	Card enter busy state interrupt enable
1	CD	Card-detect interrupt enable
0	EOC	End of command interrupt enable

Common to all bits:

When the local host sets a bit location to 1, it is notified of an interrupt if the core asserted the corresponding bit location in MMC_STAT register to 1.

If set to 0, the interrupt is masked and the local host is not signaled.

- 0: Interrupt disabled
- 1: Interrupt enabled

Value after reset is low (all bits).

Table 11–14. Command Time-Out Register(MMC_CTO)

Bit	Name	Description
15:8	–	Reserved
7:0	CTO	MMC command time-out value MMC/SD mode only The local host sets this field based on N_{CR} clock cycles. The MMC and SD cards specify N_{CR} to be between 2 and 64 clock cycles. If the card does not respond within the specified number of cycles, command time-out is set to 1 in the MMC_STAT[CTO] register bit. For MMC card-interrupt mode support, this time-out is disabled when the command passes with an R5 response (CMD40) if register bit MMC_SDIO[C5E] = 0. For the SDIO card, this time-out value is not valid for command passes with MMC_CMD[ODTO] = 1 if MMC_SDIO[XDTS] = 1; MMC_DTO[DTO] applies instead. 0x00: Command time-out disabled 0x01: 1 clock cycle … 0xFF: 255 clock cycles (2^8–1) Value after reset is low (all 8 bits).

This 16-bit register specifies the maximum number of clock cycles before a command time-out condition occurs.

Table 11–15 lists the characteristics of the data read time-out register.

Table 11–15. Data Read Time-Out Register (MMC_DTO)

Bit	Name	Description
15:0	DTO	Data read time-out. See Table 11–16. In MMC/SD mode, the local host must set this field based on N_{AC} clock cycles. N_{AC} is computed from the parameters TAAC and NSAC and the operating clock frequency. TAAC and NSAC are CSD card parameters and are obtained by reading the response register after successful execution of a SEND_CSD command (CMD9). If the card does not respond within the specified number of cycles, data time-out is set to 1 in the MMC_STAT[DTO] register bit. The effective number of clock cycles for the time-out value is multiplied by 1024 if MMC_SDIO[DPE] = 1 and by 1 if MMC_SDIO[DPE] = 0. For the SDIO card, this time-out value is also valid for command without data stage passes with MMC_CMD[ODTO] = 1 and MMC_SDIO[XDTS] = 1. In SPI mode, a data time-out condition is also generated if the RDY/BUSY signal is asserted low (BUSY) for DTO consecutive clock cycles.

MMC_DTO is a 16-bit register that specifies the maximum number of clock cycles before a data time-out condition occurs.

Table 11–16. Clock Cycles for Time-out Value

MMC_DTO[DTO]	MMC_SDIO[DPE] = 0	MMC_SDIO[DPE] = 1	Units
0x0000	No time-out	No time-out	
0x0001	1	1024	MMC clock cycles
0x0002	2	2048	“
…	…	…	“
0xFFFF	65535 (2^{16}–1)	67107840 (2^{26}–2^{10})	“

Value after reset is low (all 16 bits).

Table 11–17. Data Access Register (MMC_DATA)

Bit	Name	Description
15:0	DATA	Transmit/receive FIFO data In MMC/SD mode, this register contains either the data packet associated with block transfer (read or write), the CID contents for a PROGRAM_CID (CMD26) command, or the CSD contents for a PROGRAM_CSD (CMD27) command. Because the block length is passed as an argument, the local host can perform only 16-bit access (read or write) into the buffer even if the block length is not an even number. In case of an odd number of bytes to read, the upper byte of the last access always reads as 0x00. Conversely, for an odd number of bytes to write, the upper byte must be filled with 0x00 for the last data value. In SPI mode, the register contains both the command (opcode and address for a serial flash) and the data. In SYSTEST mode, the FIFO behaves as a stack accessible only by the local host (push and pop operations). In this mode, the set FIFO threshold values are active, as are the associated interrupts and DMA if enabled. This special mode can be used for system test purposes. Value after reset is low (all 16 bits).

The data access register is the entry point for the local host to read data from, or write data into, the FIFO buffer. The FIFO size is 32x16 bits (64 bytes). Bytes within a word are stored and read in little-endian format (MMC_CON[BE] = 0) or big-endian format (MMC_CON[BE] = 1).

If the local host accesses this register byte-wise, the MS byte (bits [15:8]) must always be written/read first. A byte access to the LS byte without a prior write into the MS byte results in the MS byte being filled with 0x00.

Note: Read/Write Access

A read access when the buffer is empty returns the previous read data value. A write access when the buffer is full is ignored. In both events, the FIFO pointers are not updated and a remote access error (hardware error) is generated (access qualifier). No remote error is generated when the local host performs a 16-bit access if the buffer contains a single byte. A debugger read access (SUSPEND operation) returns the current read data value but does not update the FIFO pointers or generate an access error.

Figure 11–6 shows the behavior of the data access to/from the FIFO as a function of the MMC_CON[BE] bit value.

Figure 11–6. Little-/Big-Endian Mode FIFO Access

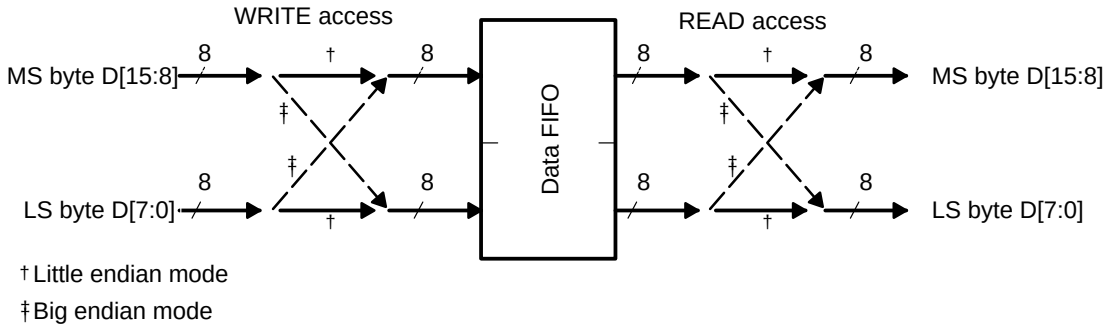


Table 11–18. Block Length Register (MMC_BLEN)

Bit	Name	Description
15:11	–	Reserved
10:0	BLEN	Block length value A write into this register initializes an 11-bit counter that decrements by 1 after each byte is transferred. A read of the register returns the number of bytes remaining to be transferred. When the counter reaches 0 and the last byte transfer completes, the core automatically reloads the block length to its programmed value, except in the case of an aborted/stopped data transfer. In MMC/SD mode, this 11-bit value specifies the data block length. This value must be set with a maximum $2^{READ\_BL\_LEN}$-1 for a block read or a maximum $2^{WRITE\_BL\_LEN}$-1 for a block write, respectively. READ_BL_LEN and WRITE_BL_LEN are CSD register settings of the card returned in an R2 response following a SEND_CSD command (CMD9). In SPI mode and for a read transaction, the block length must be initialized with the exact byte count to read –1, excluding the opcode and address arguments. Opcode and address arguments passed to the SPI device must be written into the FIFO buffer before starting the SPI transfer. BLEN starts to decrement as soon as the buffer contents have been shifted out to the SPI device. The buffer then starts to be filled with the received data from the SPI device. In SPI mode and for a write transaction, the block length must be initialized with the exact byte count to write –1, including the number of bytes needed to pass the opcode and address arguments. It is recommended to have opcodes and address that are passed to the SPI module written into the FIFO buffer before starting the SPI transfer. This allows the DMA write operation to access only the data portion. The block length starts to decrement for every byte shifted out to the SPI device. 0x000: 1 byte … 0x7FF: 2048 bytes Value after reset is low (all 11 bits).

The block length register (Table 11–18) configures the core for the number of bytes to read or write. It must be initialized at least once before starting an MMC, SD, or SPI block data transfer (read or write).

Table 11–19. Number of Blocks Register (MMC_NBLK)

Bit	Name	Description
15:11	–	Reserved
10:0	NBLK	Number of blocks value MMC/SD mode onlyIn MMC/SD mode, this 11-bit value specifies the number of blocks for a multiple-block data transfer (read or write). This value must be set with the number of blocks –1. Each block is of size MMC_BLEN[BLEN]. This register must be programmed before any multiple block data transfer. A write into this register initializes an 11-bit counter that decrements by 1 after each block transfer. A read of this register returns the number of blocks remaining to be transferred to the card. When the counter reaches 0, the transfer stops after the last transfer completes. For stream or multiple block transfers, a block received/sent interrupt is generated only once after the last successful transfer when MMC_NBLK[NBLK] reaches 0. In stream mode, once the block receive/sent interrupt is received, MMC_NBLK[NBLK] can be reprogrammed to continue the transfer from the point where it was interrupted. NOTE: This value must be 0x000 for a single block transfer. In stream mode, the minimum allowable number of blocks is 2. Finally, if the transfer is interrupted by a STOP_TRANSMISSION command (CMD12) before the counter reaches 0, reprogram this register before starting any new single- or multiple-block data transfer. 0x000: 1 block … 0x7FF: 2048 blocks Value after reset is low (all 11 bits).

MMC_NBLK configures the number of blocks for a multiple block data transfer (read or write) operation for MMC/SD cards. This register is not used for SPI transfers.

Table 11–20. Buffer Configuration Register (MMC_BUF)

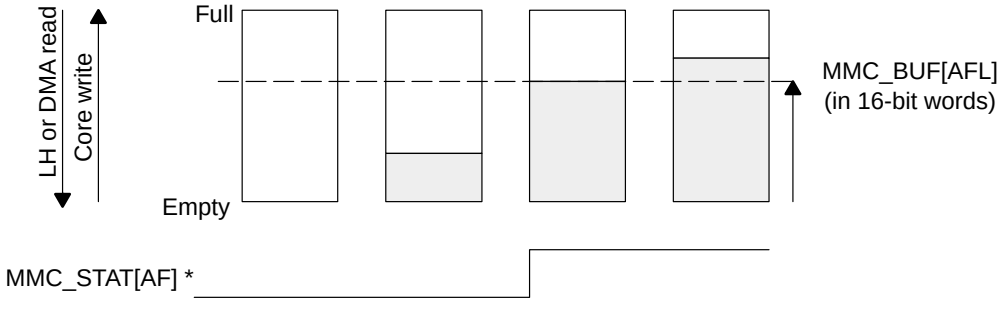
Bit	Name	Description
15	RXDE	Receive DMA channel enable When this bit is set to 1, the receive DMA channel is enabled and the core forces the MMC_STAT[AF] status bit to 0 regardless of the buffer almost-full level setting (MMC_BUF[AFL]). See *Buffer Almost Full (AFL),* below, and Section 11.4, *DMA Operations*. 0: Receive DMA channel disabled 1: Receive DMA channel enabled Value after reset is low.
14:13	–	Reserved

Table 11–20. Buffer Configuration Register (MMC_BUF) (Continued)

Bit	Name	Description
12:8	AFL	Buffer almost-full level This register holds the programmable almost-full level value used to determine the almost-full buffer condition. If users want an interrupt or a DMA read request to be issued during a read operation when the data buffer holds *at least* n 16-bit words, the buffer MMC_BUF[AFL] must be set with n–1. 0x00: 1 16-bit word (2 bytes) … 0x1F: 32 16-bit words (64 bytes) Value after reset is low (all 5 bits).
7	TXDE	Transmit DMA channel enable When this bit is set to 1, the transmit DMA channel is enabled and the core forces the MMC_STAT[AE] status bit to 0 regardless of the buffer almost-empty level (MMC_BUF[AEL]). See Section *Buffer Almost Empty (AEL),* below*,* and Section 11.4, *DMA Operations*. 0: Receive DMA channel disabled 1: Receive DMA channel enabled Value after reset is low.
6:5	–	Reserved
4:0	AEL	Buffer almost-empty level This register holds the programmable almost-empty level value used to determine an almost-empty buffer condition. If users want an interrupt or a DMA write request to be issued during a write operation when the data buffer is able to receive n words of 16 bits, then MMC_BUF[AEL] must be set with n–1. 0x00: 1 16-bit word (2 bytes) … 0x1E: 31 16-bit words (62 bytes) 0x1F: 32 16-bit words (64 bytes) Value after reset is low (all 5 bits).

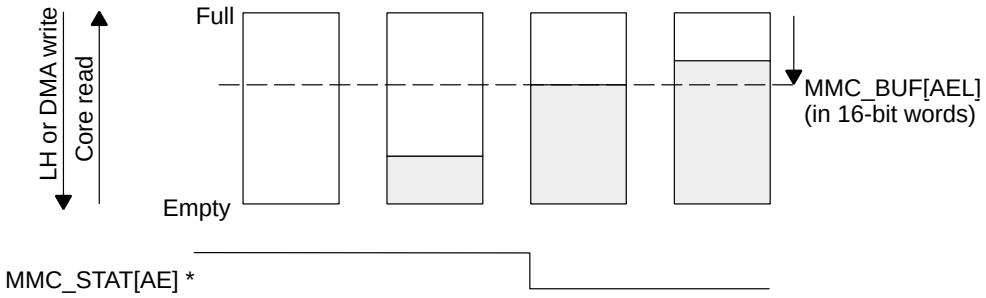
The buffer configuration register configures the buffer threshold level of the 32 16-bit-word FIFO entries and enables DMA transfers.

Figure 11–7. Buffer Almost Full Level (AFL)



* Non-DMA mode only. In DMA mode, the DMA TX request is asserted to its active level under identical conditions.

Figure 11–8. Buffer Almost-Empty Level (AEL)



* Non–DMA mode only. In DMA mode, the DMA RX request is asserted to its active level under identical conditions.

Table 11–21. SPI Configuration Register (MMC_SPI)

Bit	Name	Description
15	STR	Start SPI transfer Set only bit This bit can be set only if MMC_SPI[CSTR] = 0. This bit always reads as 0. A write to 0 has no effect. When the local host sets the bit to 1, an SPI transfer automatically starts. Users must initialize MMC_BLEN[BLEN] before starting an SPI transfer. The SPI transfer automatically stops when the size programmed in MMC_BLEN[BLEN] decrements to 0 (in read and in write). 0: No action 1: SPI transfer starts (condition: MMC_SPI[CSTR] = 0) Value after reset is low.
14	WNR	Write/not read This bit instructs the 11-bit block length counter (MMC_BLEN[BLEN]) to decrement either on byte read when MMC_SPI[WNR] = 0 or on byte write when MMC_SPI[WNR] = 1. 0: Decrement on byte received 1: Decrement on byte sent Value after reset is low.
13	SODV	SPI_SO serial-out pin default value This bit determines the default value for the serial-out signal before an SPI start and during the data phase of an SPI read transfer. This bit must be set to 1 for MMC/SD/SDIO cards using SPI protocol. 0: SPI_SO default value = 0 1: SPI_SO default value = 1 Value after reset is low.

Table 11–21. SPI Configuration Register (MMC_SPI) (Continued)

Bit	Name	Description
12	CSTR	SPI transfer controlled start When this bit is set, the FIFO is disabled and a write in the MMC_DATA register automatically triggers an SPI transfer of 1 byte or one 16-bit word, depending on the local host write access. Because the FIFO is disabled, the local host must wait for the SPI transfer completion signaled by MMC_STAT[AF] = 1 before attempting to write again in the MMC_DATA register. 0: No action 1: A write in MMC_DATA starts an SPI transfer. Value after reset is low.
11:10	TCSH	Chip-select hold time control This 2-bit field defines the number of interface clock cycles that the core waits after the last serial-clock edge before asserting the chip-select signal to its inactive high level. 00: 0.5 clock cycle 01: 1.5 clock cycles 10: 2.5 clock cycles 11: 3.5 clock cycles Value after reset is low (2 bits).
9:8	TCSS	Chip-select setup time control This 2-bit field defines the number of interface clock cycles that the core waits after asserting the chip-select signal to its active-low level before asserting the first serial-clock edge. 00: 1 clock cycle 01: 2 clock cycles 10: 3 clock cycles 11: 4 clock cycles Value after reset is low (2 bits).
7	CSEL	Card socket connector select When this bit is set, the core outputs the serial clock on the MMC.CLK signal instead of the SPI.CLK signal. It also outputs the chip-select 0 on the MMC.DAT3 signal instead of the SPI_C/Sn[0] signal. This can be used to communicate with MMC/SD/SDIO cards using the SPI protocol on the default MMC/SD card connector. 0: Use the SPI.CLK and MMC_C/Sn[0] signals during SPI transfer (MMC.CLK inactive low and MMC.DAT3 inactive high though pull-up). 1: Use the MMC.CLK and MMC.DAT3 signals during SPI transfer (SPI.CLK inactive low and SPI_C/Sn[0] inactive high). Value after reset is low.
6	–	Reserved

Table 11–21. SPI Configuration Register (MMC_SPI) (Continued)

Bit	Name	Description
5:4	CS	Chip-select control The encoded value selects the device being targeted for SPI transfer. 00: C/S 0 01: C/S 1 10: C/S 2 11: C/S 3 Value after reset is low (2 bits).
3	CSM	Chip-select mode. See Table 11–22. When this bit is set to 0 and enabled (MMC_SPI[CSD] = 0), the selected chip-select signal pin is brought active (low) only when the SPI transfer starts, and returns automatically to its inactive state (high) when the SPI transfer completes. When this bit is set to 1, automatic control of the chip-select signal is disabled, and the signal pin is then manually controlled by the chip-select disable register bit (MMC_SPI[CSD]). This mode provides support for a complex SPI transfer scheme that requires chip-select to be kept active during the entire transfer (for example, MMC card write with busy condition). 0: Automatic mode 1: Manual mode (controlled by CSD) Value after reset is low.
2	CSD	Chip-select disable. See Table 11–22. When this bit is set to 0, the selected chip-select signal is asserted to its active (low) state either automatically when MMC_SPI[CSM] = 0 or manually when MMC_SPI[CSM] = 1. When this bit is set to 1, the selected chip-select signal is forced to its inactive (high) state. This bit can be used to send dummy clocks with chip-select inactive to an MMC or SD card. 0: Selected chip-select conditionally asserted (low). See Table 11–22. 1: Selected chip-select deasserted (high) Value after reset is low.

Table 11–21. SPI Configuration Register (MMC_SPI) (Continued)

Bit	Name	Description
1	PHA	Phase control The clock polarity and clock phase bits select four different clocking schemes for the serial clock pin (SPI.CLK or MMC.CLK). The clock phase bit (MMC_SPI[PHA]) selects a half-cycle delay for the clock. When the clock phase = 0: ❏ MSB data is ready one-half cycle of the serial clock before the SPI clock starts. ❏ Data is shifted in during reception on the first edge transition of the serial clock. ❏ Data is shifted out in transmission on the second edge transition of the serial clock. When the clock phase = 1: ❏ Data is shifted out in transmission on the first edge transition of the serial clock. ❏ Data is shifted in during reception on the second edge transition of the serial clock. 0: Phase 0 1: Phase 1 Value after reset is low.
0	POL	Polarity control The clock polarity bit (MMC_SPI[POL]) selects the active edge of the clock, either rising or falling. When this bit is 0, the idle value of the serial clock signal (SPI.CLK or MMC.CLK) is low, and the rising edge of the clock is active. When this bit is 1, the idle value of the serial clock signal is high, and and the falling edge of the clock is active. 0: Rising edge active 1: Falling edge active Value after reset is low.

MMC_SPI is used to configure the SPI interface and to start an SPI transfer if the SPI mode has been enabled.

Figure 11–9. SPI Mode C/S Timing Controls (POL = 0)

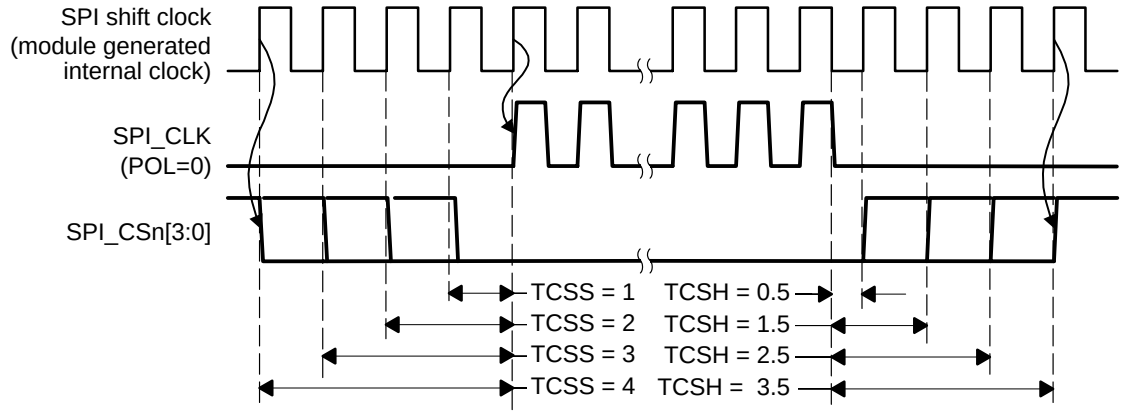


Figure 11–10. SPI Mode C/S Timing Controls (POL = 1)

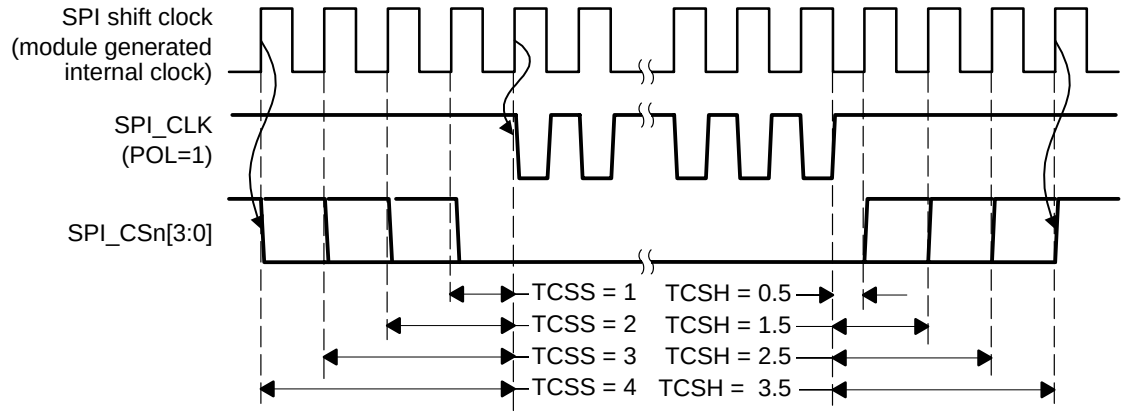


Table 11–22. Chip-Select Control (SPI Mode)

CSM	CSD	Selected CS	Comment
0	0	High → Low → High	Automatic mode: CS asserted active (low) during SPI transfer
0	1	High	Automatic mode: CS forced inactive (high)
1	0	Low	Manual mode: CS asserted active (low)
1	1	High	Manual mode: CS asserted inactive (high)

Table 11–23. SDIO Mode Configuration Register (MMC_SDIO)

Bit	Name	Description
15	C5E	Card status error on bit 5 of response 1 enable This bit must be set to 1 for SDIO cards only. If this bit is set to 1, the R5 response generates card status errors and enables the command time-out. See SDIO CMD52. See Section 5.1, IO_RW_DIRECT command (CMD52), SDIO Card Specification. By default, when this bit is set to 0, the R5 response does not generate errors or disable the command time-out for MMC interrupt mode support. 0: MMC default mode 1: Card status errors on response R5 enabled (SDIO card only) Value after reset is low.
14	C14E	Card status error on bit 4 of response 1 enable This bit must be set to 1 for SDIO cards only. When this bit is set to 1, a card status error is generated if bit 4 of the status is 1 for an R1 or R1b response. 0: Error on bit 4 masked 1: Card status errors on bit 4 of response R1 enabled (SDIO card only) Value after reset is low.

Table 11–23. SDIO Mode Configuration Register (MMC_SDIO)(Continued)

Bit	Name	Description
13	C13E	Card status error on bit 3 of response 1 enable This bit must be set to 1 for SD cards only or for an application-specific command that generates an error. When this bit is set to 1, a card status error is generated if bit 3 of the status is 1 for an R1 or R1b response. 0: Error on bit 3 masked 1: Card status errors on bit 3 of response R1 enabled (SD card or application-specific only) Value after reset is low.
12	C12E	Card status error on bit 2 of response 1 enable This bit must be set to 1 for an application-specific command that generates an error. When this bit is set to 1, a card status error is generated if bit 2 of the status is 1 for an R1 or R1b response. 0: Error on bit 2 masked 1: Card status errors on bit 2 of response R1 enabled (application specific only) Value after reset is low.
11	D3PS	DAT3 polarity control This bit determines the active edge/level condition for DAT3 signal. 0: Not inverted 1: Inverted Value after reset is low.
10	D3ES	DAT3 edge/level detection mode This bit determines whether DAT3 is an edge- or level-sensitive signal. 0: Level sensitive 1: Edge sensitive Value after reset is low.
9	CDWE	Card-detect wake request enable SD/SDIO card only This bit must be set to allow a valid card-detect condition to assert the WAKE_REQ signal active. 0: Masked 1: Unmasked Value after reset is low.

Table 11–23. SDIO Mode Configuration Register (MMC_SDIO)(Continued)

Bit	Name	Description
8	IWE	Interrupt wake request enable SDIO card only This bit must be set to allow a valid SDIO interrupt condition to assert the WAKE_REQ signal active. 0: Masked 1: Unmasked Value after reset is low.
7	DCR4	Disable CRC7 check in R4 response SDIO card only This bit must be set to disable the CRC7 check on R4 response. This prevents the MMC_STAT[CCRC] from being set following an IO_SEND_OP_COND command (see *SDIO Card Specification Part E1*). 0: CRC7 check enabled on R4 response 1: CRC7 check disabled on R4 response Value after reset is low.
6	XDTS	Extended data time-out mode select (default OD) This bit determines the default action of the MMC_CMD[ODTO] register bit when passing the command to the card (see section). The local host must set this bit to 1 after the card has been identified as an SDIO card in order to support the IO_RW_DIRECT command with long time-out values.This bit can remain clear in all other cases. 0: Open-drain control mode 1: Time-out control mode Value after reset is low.
5	DPE	Data time-out prescaler enable When the local host sets this bit to 1, the data time-out value (MMC_DTO[DTO]) is x1024 the number of MMC.CLK cycles. 0: x1 (prescaler off) 1: x1024 (prescaler on) Value after reset is low.
4	RW	Assert read wait condition to the card SDIO card only The local host can set this bit to stall a read multiple data transfer (CMD53) temporarily. After setting this bit, the host must wait until the wait becomes effective (MMC_STAT[CRW] = 1) before sending a new command without data phase (such as CMD52) to the card. After completion of CMD52, the local host can resume the read multiple data transfer by clearing this bit. 0: No action or read wait released 1: Read wait requested (if MMC_SDIO[RWE] = 1) Value after reset is low.

Table 11–23. SDIO Mode Configuration Register (MMC_SDIO)(Continued)

Bit	Name	Description
3	–	Reserved
2	CDE	Card-detect mode enable SD or SDIO card only The local host must set this bit to 1 to permit the card detection operation. When this bit is set, the core sets the MMC_STAT[CD] register bit to 1 if a valid card-detection condition occurs on the DAT3 line. This happens when a selectable level/edge sensitive event occurs on DAT3 and the card is in idle state. The MMC_SDIO[D3ES] register bit controls edge/level, and the MMC_SDIO[D3PS] register bit controls polarity. This event also asynchronously asserts the WAKE_REQ signal event to warn the system to wake up its clocks if needed. The MMC_SDIO[CDWE] register bit masks the WAKE_REQ signal for this event. 0: Card-detect mode disabled 1: Card-detect mode enabled Value after reset is low.
1	RWE	SDIO read wait mode enable SDIO card only The local host must set this bit to 1 to permit the read wait operation. When this bit is set to 0, the read wait operation is disabled, and setting a 1 in the MMC_SDIO[RW] register bit results in no action. 0: Read wait mode disabled 1: Read wait mode enabled Value after reset is low.
0	IRQE	SDIO interrupt mode enable SDIO card only The local host must set this bit to 1 to permit the interrupt operation. When this bit is set, the core sets the MMC_STAT[CIRQ] register bit to 1 if an interrupt condition is detected on the DAT1 line. A detected interrupt condition when the card is in idle state also asynchronously asserts the WAKE_REQ signal event to warn the system to wake up its clocks if needed. The MMC_SDIO[IWE] register bit masks the WAKE_REQ signal for this event. 0: Interrupt mode disabled 1: Interrupt mode enabled Value after reset is low.

The SDIO mode configuration register configures the MMC/SD interface for SDIO operation and the card-detection mechanism on DAT3. If the card-detection mechanism is based on a mechanical switch, its control/sense is the responsibility of another system module.

Table 11–24. System Test Register (MMC_SYST)

Bit	Name	Description
15	WAKD	WAKE_REQ output-signal data value (internal signal to system) The WAKE_REQ signal is driven high or low according to the value written into this register bit. Value after reset is low.
14	SSB	Set status bits Writing 1 into this bit sets to 1 all status bits contained in the MMC_STAT register. Writing 0 into this bit does not clear already set status bits; only writing 1 into a set status bit can clear it (see MMC_STAT operation). This bit must be cleared before attempting to clear a status bit. 0: No action 1: Set all status bits Value after reset is low.
13	RDYD	Ready/busy input signal data value This read-only bit returns the value of the signal on the input pad (high or low). 0: Ready/busy low 1: Ready/busy high Value after reset is high.
12	DDIR	DAT[3:0] signals direction When set, this bit places all of the in/out MMC_DAT[3:0] pins in output mode. 0: Input 1: Output Value after reset is low.
11:8	D3D–D0D	DAT3–DAT0 input/output signal data value[†] If MMC_SYST[DDIR] = 0 (input mode direction), these bits return the value on the corresponding MMC.DAT pins (high or low). A write into these bits has no effect. If MMC_SYST[DDIR] = 1 (output mode direction), the MMC.DAT pins are driven high or low according to the value written into these register bits. Value after reset is low (all 4 bits).
7	CDIR	CMD/SO signal direction When set, this bit places the in/out MMC.CMD/SPI.DO pin in output mode. 0: Input 1: Output Value after reset is low.

[†] SI input/output signal data value applies to bit 8.

Table 11–24. System Test Register (MMC_SYST) (Continued)

Bit	Name	Description
6	CDAT	CMD/SO input/output signal data value If MMC_SYST[CDIR] = 0 (input mode direction), these bits return the value on the MMC.CMD/SPI.DO pin (high or low). A write into this bit has no effect. If MMC_SYST[CDIR] = 1 (output mode direction), the MMC_CMD pin is driven high or low according to the value written into this register bit. Value after reset is low.
5	MCKD	MMC clock output signal data value The MMC.CLK pin is driven high or low according to the value written into this register bit. Value after reset is low.
4	SCKD	SPI clock output signal data value The SPI.CLK pin is driven high or low according to the value written into this register bit. Value after reset is low.
3:0	CS3D–CS0D	C/S3–C/S0 output signal data value The CS[3:0] pins are driven high or low according to the values written into these register bits. Value after reset is low (all 4 bits).

† SI input/output signal data value applies to bit 8.

The system test register controls the signals that connect to I/O pins when the module is configured in system test (SYSTEST) mode.

Table 11–25. Module Revision Register (MMC_REV)

Bit	Name	Description
15:8	–	Reserved
7:0	REV	Module revision number This 8-bit field indicates the revision number of the RTL for this module. This value is fixed by hardware. ❑ The 4 LSBs indicate a minor revision. ❑ The 4 MSBs indicate a major revision. ❑ 0x30: Revision 3.0 ❑ 0x31: Revision 3.1 A reset has no effect on the value returned.

Notes: 1) MMC/SD host controller without SDIO support is revision 1.x or 2.x (WMU_020_1 spec)

2) MMC/SD/SDIO host controller is revision 3.x.

MMC_REV is a read-only register that contains the revision number of the module. A write to this register has no effect.

Table 11–26. MMC/SD Command Response Register 0 (MMC_RSP0)

Bit	Name	Description
15:0	RSP0	CMD response (R2[15:0]) MMC_RSP0 is a 16-bit register that holds bit positions [15:0] for a 128-bit response of type R2. This register is also reset after a write into the MMC.CMD/SPI.DO register (command sent).

Table 11–27. MMC/SD Command Response Register 1 (MMC_RSP1)

Bit	Name	Description
15:0	RSP1	CMD response (R2[31:16]) MMC_RSP1 is a 16-bit register that holds bit positions [31:16] for a 128-bit response of type R2. MMC_RSP1 is also reset after a write into the MMC.CMD/SPI.DO register (command sent).

Table 11–28. MMC/SD Command Response Register 2 (MMC_RSP2)

Bit	Name	Description
15:0	RSP2	CMD response (R2[47:32]) MMC_RSP2 is a 16-bit register that holds bit positions [47:32] for a 128-bit response of type R2. This register is also reset after a write into the MMC.CMD/SPI.DO register (command sent).

Table 11–29. MMC/SD Command Response Register 3 (MMC_RSP3)

Bit	Name	Description
15:0	RSP3	CMD response (R2[63:48]) MMC_RSP3 is a 16-bit register that holds bit positions [63:48] for a 128-bit response of type R2. This register is also reset after a write into the MMC.CMD/SPI.DO register (command sent).

Table 11–30. MMC/SD Command Response Register 4 (MMC_RSP4)

Bit	Name	Description
15:0	RSP4	CMD response (R2[79:64]) MMC_RSP4 is a 16-bit register that holds bit positions [79:64] for a 128-bit response of type R2. This register is also reset after a write into the MMC.CMD/SPI.DO register (command sent).

Table 11–31. MMC/SD Command Response Register 5 (MMC_RSP5)

Bit	Name	Description
15:0	RSP5	CMD response (R2[95:80]) MMC_RSP5 is a 16-bit register that holds bit positions [95:80] for a response of type R2. This register is also reset after a write into the MMC.CMD/SPI.DO register (command sent).

Table 11–32. MMC/SD Command Response Register 6 (MMC_RSP6)

Bit	Name	Description
15:0	RSP6	CMD response (R2[111:96], R1/R1b/R3/R4/R5/R6[23:8]) MMC_RSP6 is a 16-bit register that holds bit positions [111:96] for a 128-bit response of type R2 and bit positions [23:8] for a 32-bit response of types R1/R1b/R3/R4/R5/R6. This register is also reset after a write into the MMC_CMD register (command sent).

Table 11–33. MMC/SD Command Response Register 7 (MMC_RSP7)

Bit	Name	Description
15:0	RSP7	CMD response (R2[127:112], R1/R1b/R3/R4/R5/R6[39:24]) MMC_RSP7 is a 16-bit register that holds bit positions [127:112] for a 128-bit response of type R2 and bit positions [39:24] for a 32-bit response of types R1/R1b/R3/R4/R5/R6. This register is also reset after a write into the MMC.CMD/SPI.DO register (command sent).

Table 11–34. SDIO Suspend/Resume Control Register (MMC_IOSR)

Bit	Name	Description
15:4	–	Reserved
3	STOP	Stop core data operation request (after card grants suspend) SDIO cards only The local host can set this bit to 1 only when the SDIO card function has been suspended. When this bit is set, the core immediately stops signaling the data transfer request (either DMA request or interrupt) to the system if no request is pending, or immediately after it has been serviced if a request is pending. Once the bit is set, the core automatically clears it to signal that FIFO operations are effectively suspended. This happens when all data requests have been disabled and serviced. The local host must poll this bit until cleared to 0 before attempting to save the FIFO contents. A write to 0 has no action. 0: No action, or FIFO operations suspended 1: Stop core data operation request Value after reset is low.

Table 11–34. SDIO Suspend/Resume Control Register (MMC_IOSR) (Continued)

Bit	Name	Description
2	SAVE	Save FIFO contents of suspended function SDIO cards only This bit must be set to 1 when the core acknowledges the stop data operation request MMC_IOSR[STOP] = 0. When this bit is set, the FIFO is placed in read mode, and the local host downloads the FIFO contents of the suspended function until totally empty (signaled by MMC_STAT[AF] = 0). In this mode, the actual threshold value set in MMC_BUFF[AFL] is a don't care. MMC_STAT[AF] equals 0 only when the FIFO is empty. 0: No action 1: Save action (SDIO card only) Value after reset is low.
1	RESU	Next SD command is a resume request SDIO cards only This bit must be set to 1 when the next command passed to the card is a resume request for a suspended function. 0: No action 1: Next command is resume (SDIO card only).
0	SUSP	Next SD command is a suspend request SDIO cards only This bit must be set to 1 when the next command passed to the card is a suspend request of the current active function. When this bit is set for a multiple block read- or write operation, the core automatically stops its data transfer operation at the end of the current active block. It also stops requesting new data from the local host or system DMA for a new block write. 0: No action 1: Next command is suspend (SDIO card only). Value after reset is low.

MMC_IOSR implements the necessary controls for SDIO suspend/resume operations.

Table 11–35 lists the characteristics of the system control register.

Table 11–35. System Control Register(1.1MMC_SYSC)

Bit	Name	Description
15:2	–	Reserved
1	SRST	Software reset When this bit is set to 1, the entire module is reset as for the hardware reset. The core automatically sets this bit to 0, and it is only reset by the hardware reset. It always returns 0 during reads. The module can be also partially reset using the MMC_CON[POW] bit. 0: No action 1: Module is reset. Value after reset is low.
0	–	Reserved

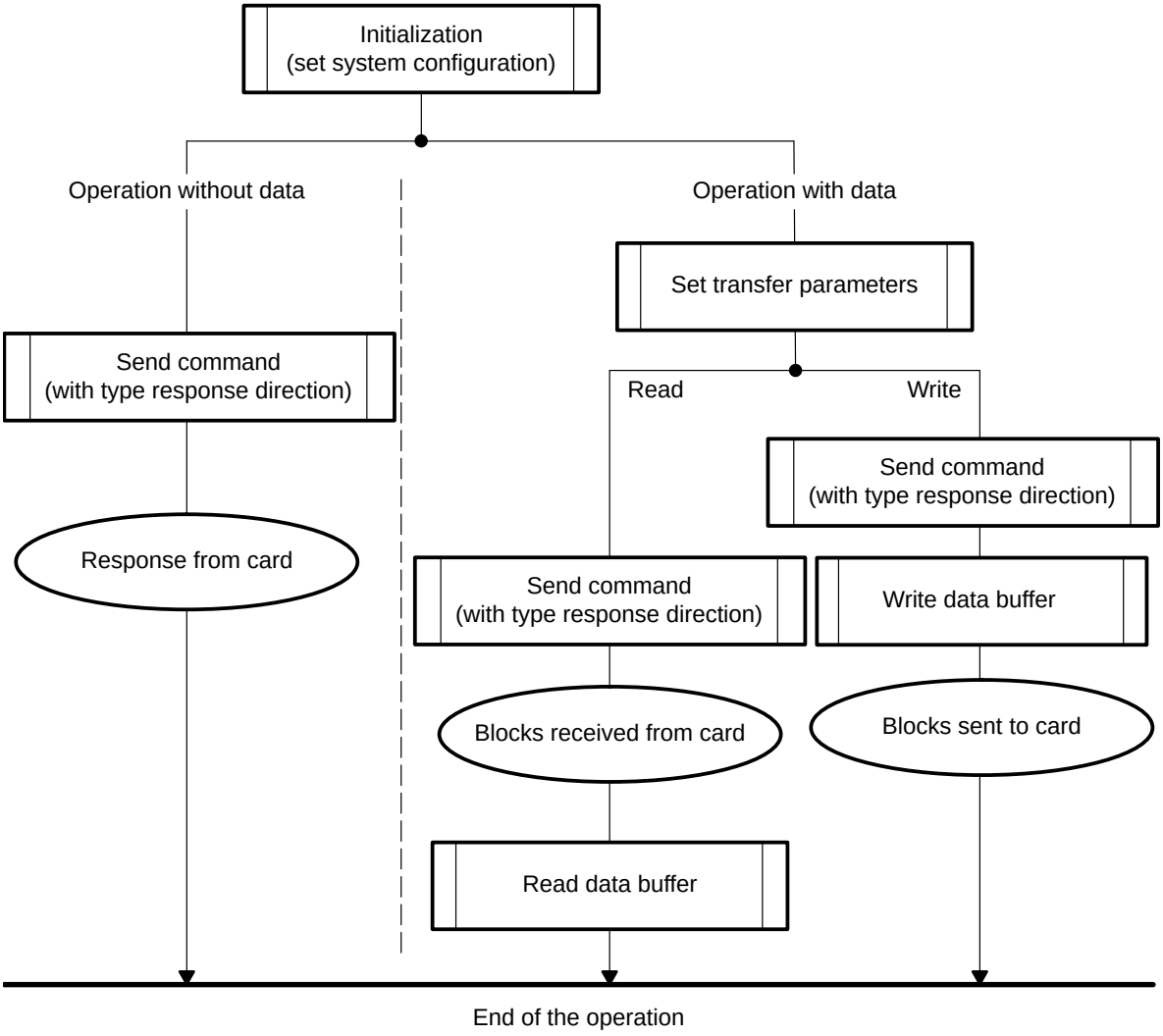
Table 11–36. System Status Register(MMC_SYSS)

Bit	Name	Description
15:1	–	Reserved
0	RSTD	Reset done This read-only bit indicates the state of the reset in case of hardware reset, global software reset MMC_SYSC[SRST], or partial software reset MMC_CON[POW]). The module must receive all of its clocks before it can grant a reset completed status. 0: Internal module reset in ongoing or partially held in reset 1: Reset completed Value after reset is low.

11.3 MMC Command Flow

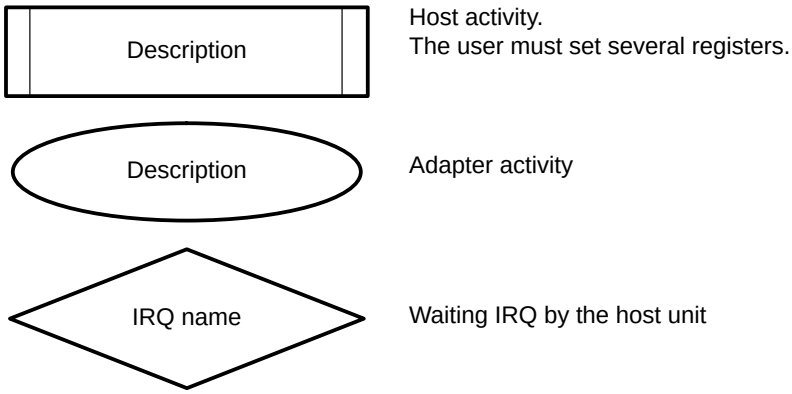
To drive the MMC/SD/SDIO correctly for the execution of a command, the host must process as shown in Figure 11–11 and Figure 11–12.

Figure 11–11. General Command Flow



In all modes (MMC, SD, SPI), an initialization phase is necessary at the beginning. After the initialization, the MMC/SD adapter works differently depending on whether the host sends a command without data or with data, and also according to the type, the index of the response, and the direction.

Figure 11–12. Flow Conventions



11.3.1 Basic Operations

Figure 11–13 depicts initialization.

Figure 11–13. Initialization

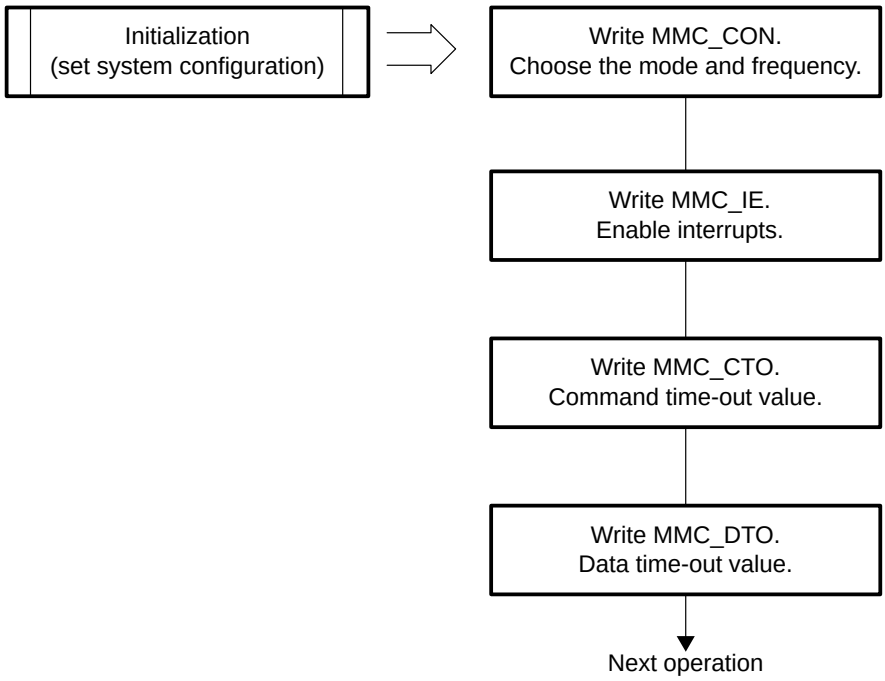


Figure 11–14 depicts the command transfer.

Figure 11–14. Command Transfer

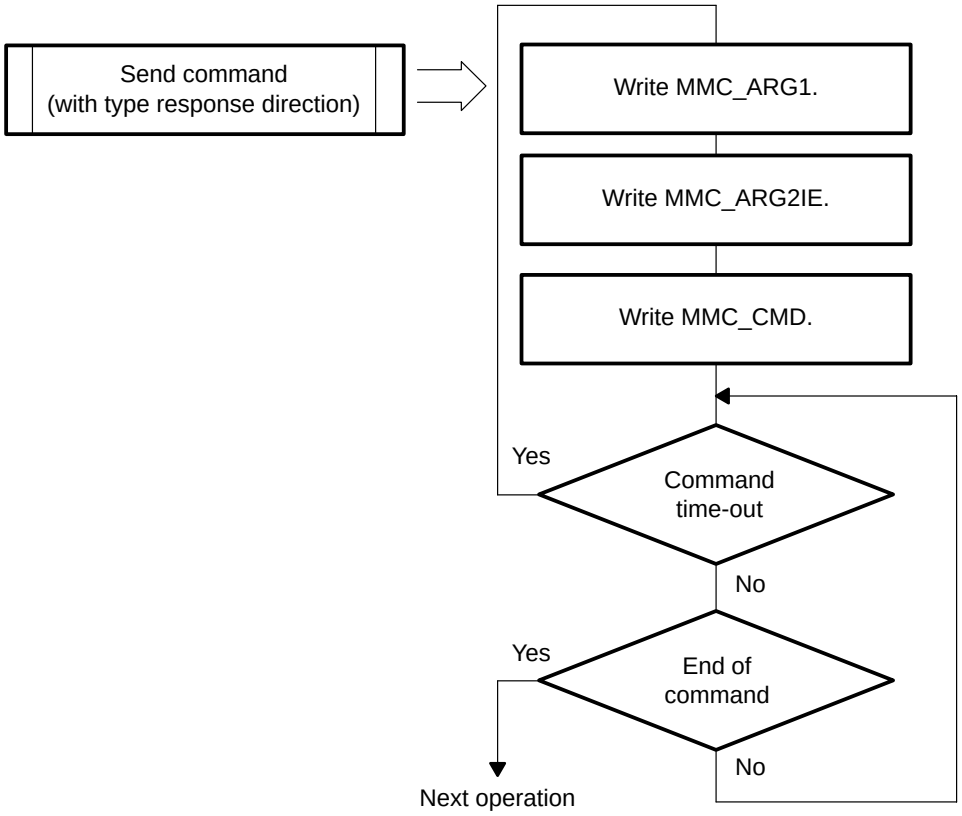
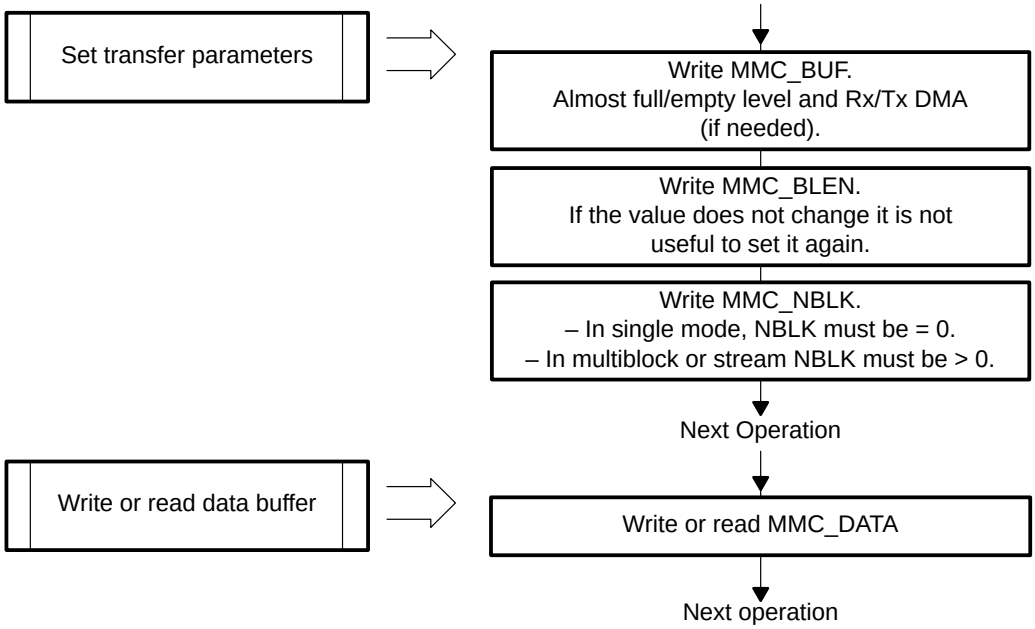


Figure 11–15 illustrates the transfer of data.

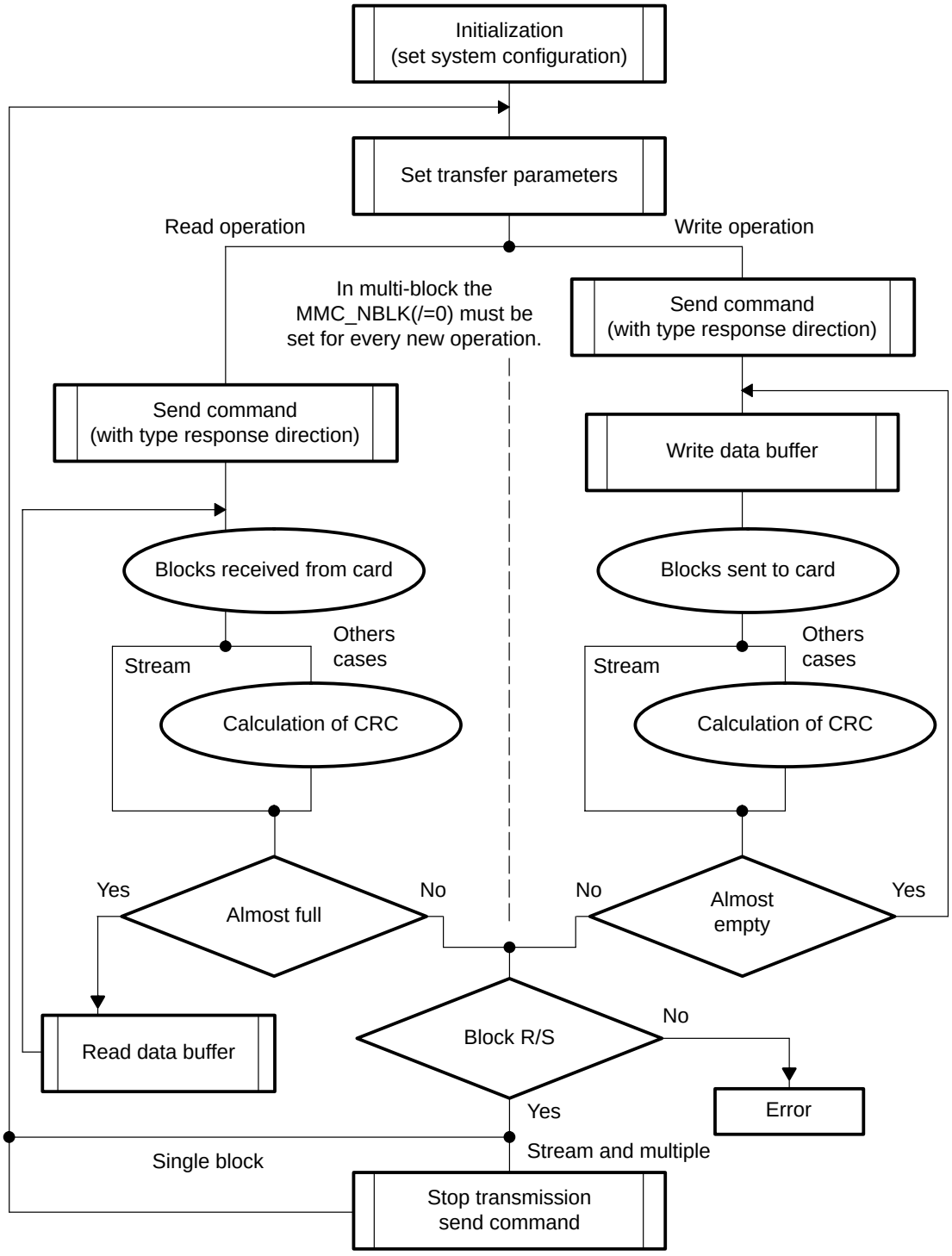
Figure 11–15. Data Transfer



Mode MMC/SD (MMC_CON[MODE] = 00) is selected for this example.

Figure 11–16 illustrates the transfer of data in MMC/SD mode.

Figure 11–16. Data Transfer in MMC/SD Mode



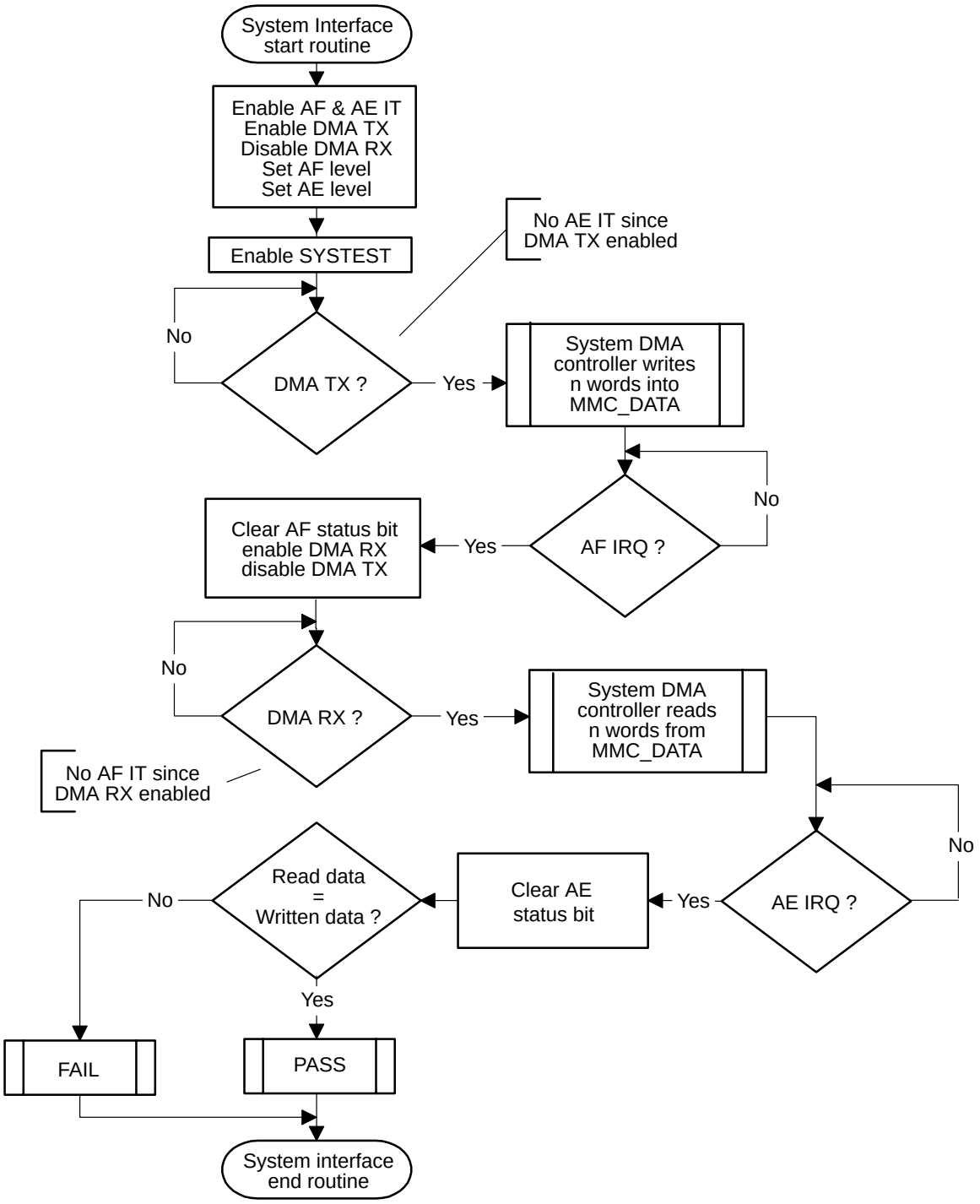
11.3.2 System Test Mode

The system test (SYSTEST) mode implemented in the MMC/SD host controller easily checks the correctness of the system interconnect either internally to the peripheral bus, central DMA, and interrupt handler, or externally to device I/O pads.

I/O verification is not depicted here but can be performed in SYSTEST mode by toggling the outputs and capturing the logic state of the inputs.

Figure 11–17 depicts a scenario to validate data DMA transfers and interrupt assertions in a one-pass flow. All data transfers are performed under DMA control.

Figure 11–17. System Interface Test Flow



11.3.3 SPI Mode

In write operations, the block length register (MMC_BLEN[BLEN]) is loaded with the data transfer dimension, opcode, and address.

In read operations, the block length register is loaded only with the data transfer dimension.

11.4 DMA Operations

11.4.1 MMC DMA Receive Mode

In a DMA block read operation (single or multiple), the DMA RX request signal is asserted to its active level when the FIFO level becomes greater than or equal to the threshold value (in 16-bit words) set in MMC_BUF[AFL]. The DMA RX request is deasserted to its inactive level when the system DMA has read one single word from the FIFO.

Because the request lasts one 16-bit word read cycle, it is recommended that the threshold value in MMC_BUF[AFL] (expressed in 16-bit words) be equal to the DMA burst access size (n). If the system DMA does not support more than one 16-bit word read access, MMC_BUF[AFL] must be set to 0.

New DMA requests are internally masked until the system DMA has performed exactly n reads.

Note: Block Size

Because each DMA transfer is of equal size, the block size of the transfer must be a multiple of the DMA read access size.

Summary:

- DMA transfer size = n ≤ FIFO size (max 32 16-bit words)
- MMC_BUF[AFL] = n – 1 (FIFO threshold level)
- n = Submultiple of block size

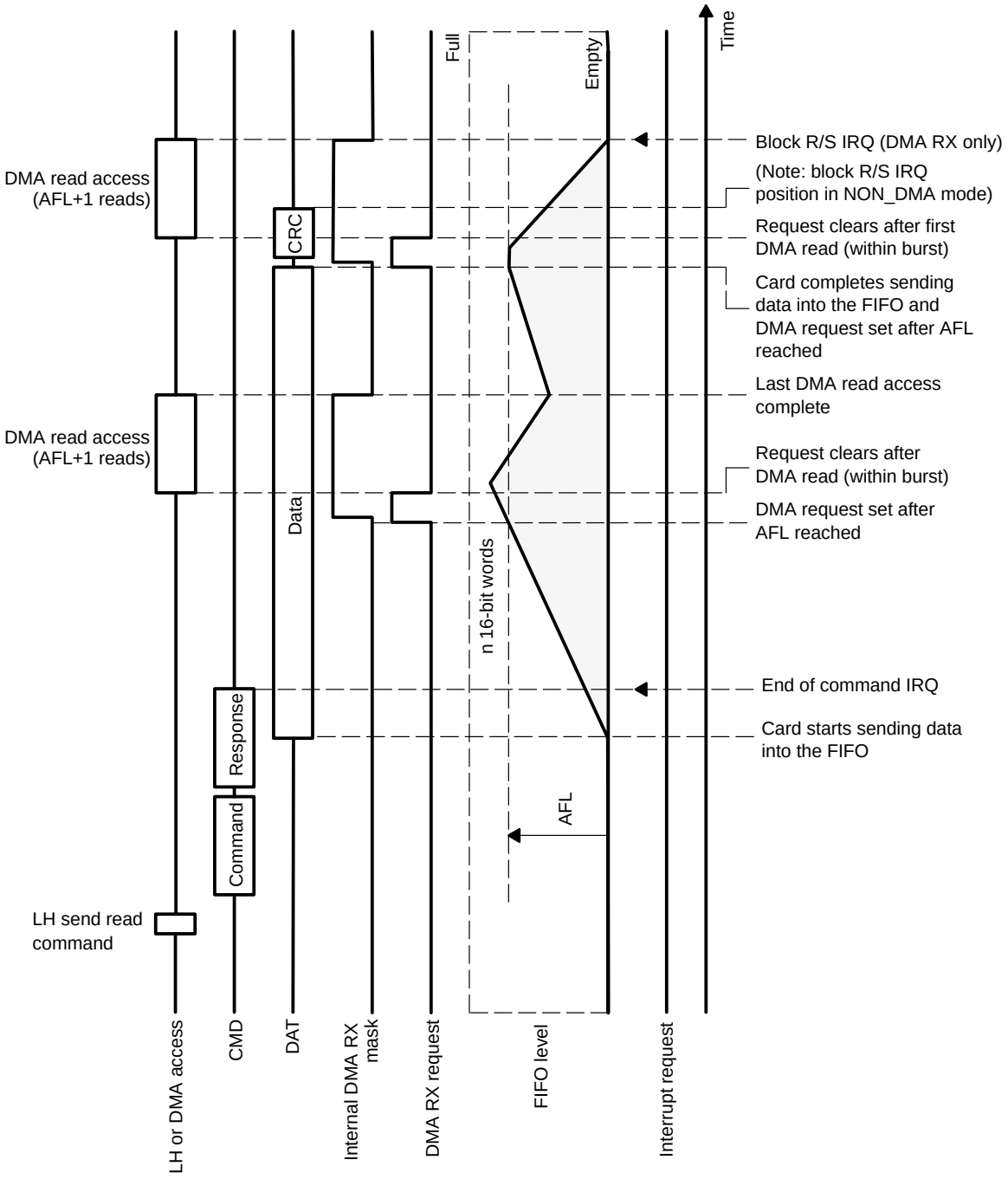
Example: Multiple block read of 7 blocks of 256 bytes each.

The DMA transfer size n can be set to 4 16-bit words (8 bytes) and MMC_BUF[AFL] = 0x3. The read transfer operation completes after 224 system DMA RX requests.

The receive FIFO never overflows. If the FIFO becomes full, the MMC.CLK signal is momentarily stopped until the system DMA or the local host performs a read access, which starts emptying the FIFO.

Figure 11–18 shows a typical DMA read operation to an MMC card.

Figure 11–18. MMC Mode DMA RX Transfer



Signals are represented in a symbolic form (a high level for the DMA mask or the request signal denotes an ON condition).

11.4.2 MMC DMA Transmit Mode

In a DMA block write operation (single or multiple), the DMA TX request signal is asserted to its active level when the FIFO level becomes less than or equal to the threshold value (in 16-bit words) set in MMC_BUF[AEL] after the block write command has been set (write action into MMC.CMD/SPI.DO). The DMA TX request is deasserted to its inactive level when the system DMA has written one single word into the FIFO.

Because the request lasts one 16-bit word write cycle, it is recommended that the threshold value in MMC_BUF[AEL] (expressed in 16-bit words) be equal to the DMA burst size (n). If the system DMA does not support more than one 16-bit word write access, MMC_BUF[AEL] must be set to 0.

New DMA requests are internally masked until the system DMA has performed exactly n writes.

Note: Block Size

Because each DMA transfer is of equal size, it is necessary to have the block size of the transfer be a multiple of the DMA write access size.

Summary:

- DMA transfer size = n ≤ FIFO size (maximum 32 16-bit words)
- MMC_BUF[AEL] = n – 1 (FIFO threshold level)
- n = Submultiple of block size

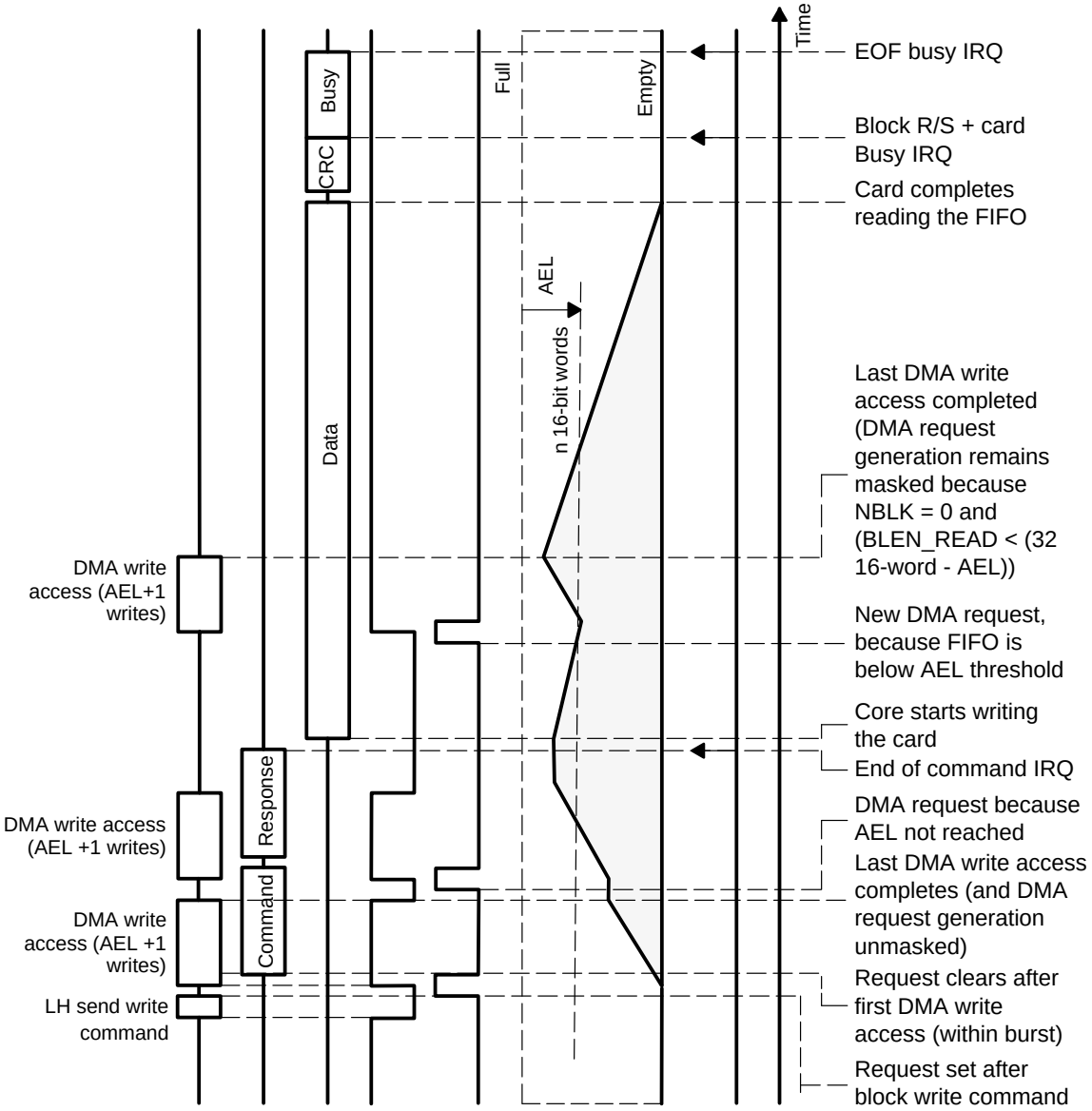
Example: Multiple block write of 10 blocks of 512 bytes each

The DMA transfer size n can be set to 16 16-bit words (32 bytes) and MMC_BUF[AEL] = 0xF. The write transfer operation completes after 160 system DMA TX requests.

The transmit FIFO never underflows. If the FIFO is emptied, the MMC.CLK clock signal is momentarily stopped until the system DMA or the local host performs a write access, which starts filling the FIFO.

Figure 11–19 shows a typical DMA write operation to an MMC card.

Figure 11–19. MMC Mode DMA TX Transfer



Signals are represented in a symbolic form (a high level for the DMA mask or the request signals denotes an ON condition).

11.4.3 SDIO Suspend/Resume

Suspend/resume only apply to multiple-block read or write operations.

To suspend a function, the local host must:

1) Save the MMC_CMD[DDIR] value.
2) Set MMC_IOSR[SUSP] to 1.
3) Send the suspend command (CMD52) and check the SDIO card status for acknowledgment. Repeat the command as long as necessary.
4) Set MMC_IOSR[STOP] to 1 and read MMC_IOSR[STOP] until 0.
5) Disable DMA channel if enabled.
6) Save the MMC_NBLK register.
7) Set the MMC_IOSR[SAVE] bit to 1 and empty the FIFO by reading the MMC_DATA register until empty (signaled by MMC_STAT[AF] = 0).
8) Clear the MMC_IOSR[SUSP] and MMC_IOSR[SAVE] bits to 0.

A new command can be sent to another function at this point.

To resume the suspended function, the local host must:

1) Restore the FIFO contents by writing into the MMC_DATA register.
2) Restore the MMC_NBLK register with the saved value.
3) Reenable the DMA channel if in DMA mode.
4) Set MMC_IOSR[RESU] bit to 1.
5) Send the resume command (CMD52) with MMC_CMD[DDIR] set according to the suspended function (needed to resume the function as a multiple-block read or as a multiple-block write)
6) Clear the MMC_IOSR[RESU] bit to 0.

Table 11–37 contains information for programming the CMD register.

Table 11–37. Programming Aid for CMD Register (MMC_CMD)

Command	DDIR	SHR	TYPE	BUSY	Resp	Hex. Value (MSB of MMC.CMD/ SPI.DO)
bc: no resp	0	0	00	0	000	0x00
bcr: R3	0	0	01	0	011	0x13
bcr – R6	0	0	01	0	110	0x16
bcr: R2	0	0	01	0	010	0x12
bcr: R5	0	0	01	0	101	0x15
ac: R1	0	0	10	0	001	0x21
ac: R1b	0	0	10	1	001	0x29
ac: R2	0	0	10	0	010	0x22
ac: no resp	0	0	10	0	000	0x20
ac: R4	0	0	10	0	100	0x24
adtc: R1	1	0	11	0	001	0xB1
(no stream)	0					0x31
adtc: R1b	1	0	11	1	001	0xB9
(no stream)	0					0x39
adtc: R1	1	1	11	0	001	0xF1
(stream)	0					0x71
adtc: R1b	1	1	11	1	001	0xF9
(stream)	0					0x79
Host response	0	1	00	0	000	0x40

11.4.4 Programming Model Incompatibility

Software developed for the previous WMU_020_1 MMC/SD host controller version is compatible with this version except for the listed changes:

- ❏ MMC_CMD[INAB]: The previous core sent an 80-clock initialization sequence followed by a command. This core sends 80 clocks without any commands.
- ❏ MMC_CON[MODE]: The previous core defines an SPI mode 2 operation for SPI-operated MMC/SD/SDIO cards on top of the SPI mode 1 operation for serial flash card. This is now replaced by a single SPI operation with a new SPI control bit MMC_SPI[CSEL].
- ❏ MMC_CMD[SHR]: The previous core did not send the host response in open-drain mode. The new core does, so it must be performed at the proper device identification speed.
- ❏ MMC_BUF[AFL]: The previous core defines a reset value of 0x1F. The reset value for this core is 0x00.

Chapter 12

McBSPs

This chapter describes the two multichannel buffered serial ports (McBSPs) available on the OMAP730 device.

Topic **Page**

12.1 Introduction to McBSPs

The OMAP730 device provides multiple high-speed multichannel buffered serial ports (McBSPs) that allow direct interface to codecs and other devices in a system.

Section 12.1 through Section 12.11 describe McBSP functionality common to the McBSPs. Section 12.12, *McBSP in the OMAP730 Device,* and Section 12.13, *McBSP Registers,* provide specific application information.

12.1.1 Key Features of the McBSPs

The McBSPs feature:

- Full-duplex communication
- Double-buffered transmission and triple-buffered reception, which allow a continuous data stream
- Independent clocking and framing for reception and for transmission
- The capability to send interrupts to the CPU and to send DMA events to the DMA controller
- 128 channels for transmission and for reception
- Multichannel selection modes that enable or disable block transfers in each of the channels
- Direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected A/D and D/A devices
- Support for external generation of clock signals and frame-synchronization signals
- A programmable sample-rate generator for internal generation and control of clock signals and frame-synchronization signals
- Programmable polarity for frame-synchronization pulses and clock signals
- Direct interface to:
  - T1/E1 framers
  - Multivendor integration protocol (MVIP) switching-compatible and ST-BUS-compliant devices including:
    - MVIP framers
    - H.100 framers
    - SCSA framers
  - IOM-2-compliant devices
  - I2S-compliant devices
  - SPI devices

- A wide selection of data sizes: 8, 12, 16, 20, 24, and 32 bits

Note:

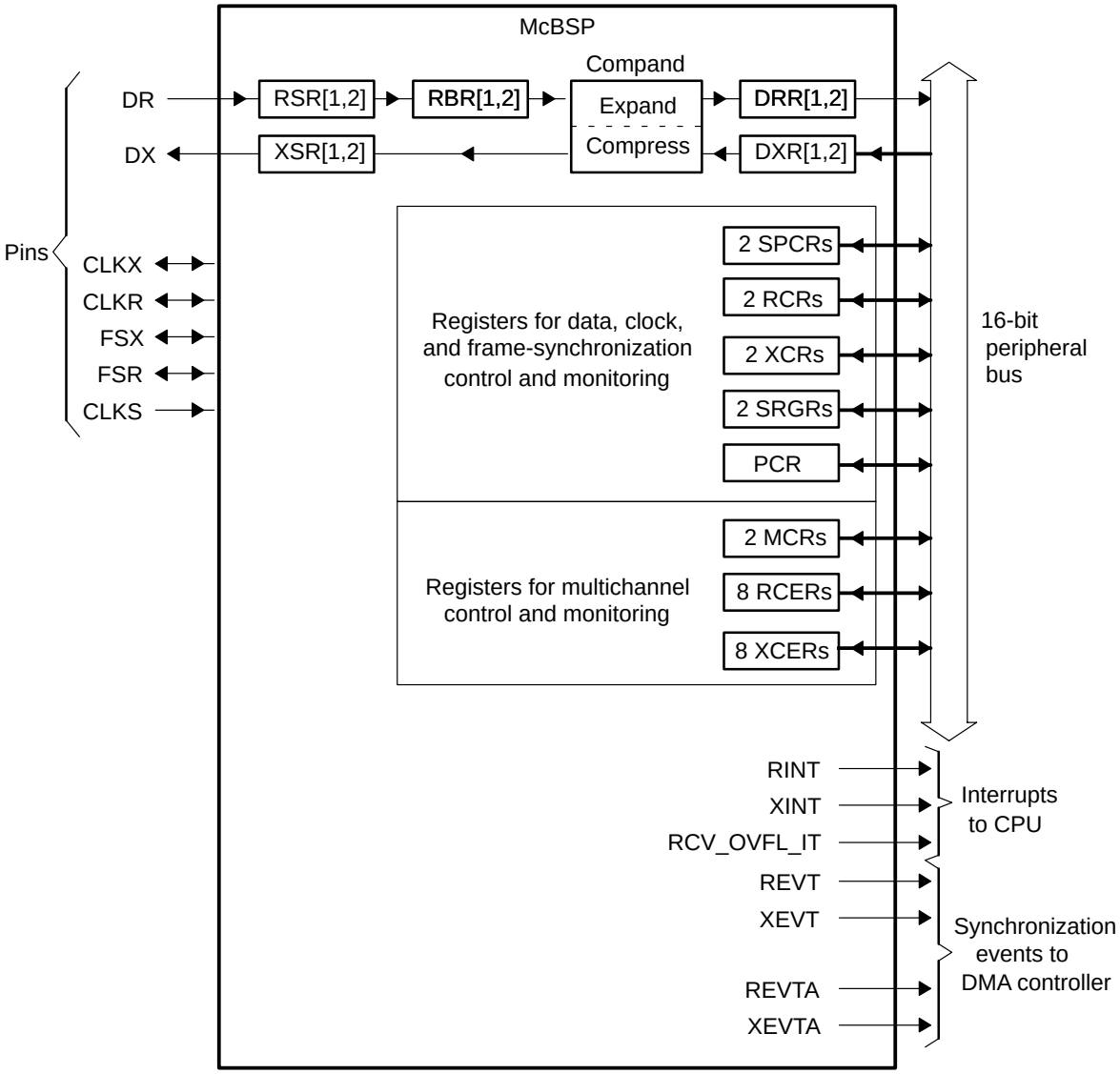
A value of the chosen data size is referred to as a *serial word* or *word* throughout the McBSP documentation. Elsewhere, *word* is used to describe a 16-bit value.

- μ-law and A-law companding
- The option of transmitting/receiving 8-bit data with the LSB first
- Status bits for flagging exception/error conditions
- The capability to use the McBSP pins as general-purpose I/O pins
- Continuous drive by OMAP730 McBSPs of their transmit data line (DX)—no support of any application with multiple drivers on this line by OMAP730

12.1.2 McBSP Generic Block Diagram

The McBSP consists of a data-flow path and a control path connected to external devices by seven pins, as shown in Figure 12–1. The figure and the text in this section use generic pin names. For the OMAP730 signal names on each McBSP, see Section 12.1.3.

Figure 12–1. Conceptual Block Diagram of the McBSP



Data is communicated to devices interfaced with the McBSP via the data transmit (DX) pin for transmission and via the data receive (DR) pin for reception. Control information in the form of clocking and frame synchronization is communicated via the following pins: CLKX (transmit clock), CLKR (receive clock), FSX (transmit frame synchronization), and FSR (receive frame synchronization).

The CPU and the DMA controller communicate with the McBSP through 16-bit-wide registers accessible via the internal peripheral bus. The CPU or

the DMA controller writes the data to be transmitted to the data transmit registers (DXR1, DXR2). Data written to the DXRs is shifted out to DX via the transmit shift registers (XSR1, XSR2). Similarly, receive data on the DR pin is shifted into the receive shift registers (RSR1, RSR2) and copied into the receive buffer registers (RBR1, RBR2). The contents of the RBRs is then copied to the DRRs, which can be read by the CPU or the DMA controller. This allows simultaneous movement of internal and external data communications.

DRR2, RBR2, RSR2, DXR2, and XSR2 are not used (written, read, or shifted) if the serial word length is 8 bits, 12 bits, or 16 bits. For larger word lengths, these registers are needed to hold the most significant bits.

The remaining registers in Figure 12–1 are registers for controlling McBSP operation.

12.1.3 McBSP Pins

Table 12–1 describes the McBSP interface pins. For information on using these pins for general-purpose input/output (GPIO), see Section 12.9.

Table 12–1. McBSP Interface Pins

Pin	McBSP1	McBSP2	Direction	Possible Uses
CLKRX	MCBSP1.CLKRX1	MCBSP2.CLKRX2	I/O	Supplying or reflecting the receive/transmit clock; supplying the input clock of the sample rate generator; general-purpose I/O
CLKS	MCBSP1.CLKS1	MCBSP2.CLKRX2	I	Supplying the input clock of the sample rate generator; general-purpose input
DR	MCBSP1.DR1	MCBSP2.DR2	I	Receiving serial data; general-purpose input
DX	MCBSP1.DX1	MCBSP2.DX2	O	Transmitting serial data; general-purpose output
FSRX	MCBSP.FSRX1	MCBSP2.FSRX2	I/O	Supplying or reflecting the receive/transmit frame-sync signal; controlling sample rate generator synchronization for the case when GSYNC = 1 (see Section 12.3.3); general-purpose I/O

12.1.4 McBSP Register Addresses

See Section 12.13, *McBSP Registers*, for McBSP register memory maps and detailed register descriptions.

12.2 McBSP Operation

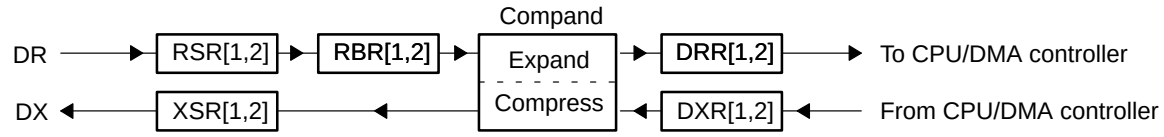
This section addresses the following topics:

- Data transfer process
- Clocking and framing data
- Frame phases
- McBSP reception
- McBSP transmission
- Interrupts and DMA events generated by McBSPs

12.2.1 Data Transfer Process of McBSPs

Figure 12–2 shows a diagram of the McBSP data transfer paths. The McBSP receive operation is triple-buffered, and transmit operation is double-buffered. The use of registers varies, depending on whether the defined length of each serial word is 16 bits.

Figure 12–2. McBSP Data Transfer Paths



12.2.1.1 Data Transfer Process for Word Length of 8, 12, or 16 Bits

If the word length is 16 bits or smaller, only one 16-bit register is needed at each stage of the data transfer paths. The registers DRR2, RBR2, RSR2, DXR2, and XSR2 are not used (written, read, or shifted).

Receive data arrives on the DR pin and is shifted into receive shift register 1 (RSR1). Once a full word is received, the content of RSR1 is copied to receive buffer register 1 (RBR1) if RBR1 is not full with previous data. RBR1 is then copied to data receive register 1 (DRR1), unless the previous content of DRR1 has not been read by the CPU or the DMA controller. If the companding feature of the McBSP is implemented, the required word length is 8 bits and receive data is expanded into the appropriate format before being passed from RBR1 to DRR1. For more details about reception, see Section 12.2.5.

Transmit data is written by the CPU or the DMA controller to data transmit register 1 (DXR1). If there is no previous data in the transmit shift register (XSR1), the value in DXR1 is copied to XSR1; otherwise, DXR1 is copied to XSR1 when the last bit of the previous data is shifted out on the DX pin. If selected, the companding module compresses 16-bit data into the appropriate 8-bit format before passing it to XSR1. After transmit frame synchronization, the transmitter begins shifting bits from XSR1 to the DX pin. For more details about transmission, see Section 12.2.6.

12.2.1.2 Data Transfer Process for Word Length of 20, 24, or 32 Bits

If the word length is larger than 16 bits, two 16-bit registers are needed at each stage of the data transfer paths. The registers DRR2, RBR2, RSR2, DXR2, and XSR2 are needed to hold the most significant bits.

Receive data arrives on the DR pin and is shifted first into RSR2 and then into RSR1. Once the full word is received, the contents of RSR2 and RSR1 are copied to RBR2 and RBR1, respectively, if RBR1 is not full. Then the contents of RBR2 and RBR1 are copied to DRR2 and DRR1, respectively, unless the previous content of DRR1 has not been read by the CPU or the DMA controller. The CPU or the DMA controller must read data from DRR2 first and then from DRR1. When DRR1 is read, the next RBR-to-DRR copy occurs. For more details about reception, see Section 12.2.5.

For transmission, the CPU or the DMA controller must write data to DXR2 first and then to DXR1. When new data arrives in DXR1, if there is no previous data in XSR1, the contents of DXR2 and DXR1 are copied to XSR2 and XSR1, respectively; otherwise, the contents of the DXRs are copied to the XSRs when the last bit of the previous data is shifted out on the DX pin. After transmit frame synchronization, the transmitter begins shifting bits from the XSRs to the DX pin. For more details about transmission, see Section 12.2.6.

12.2.2 Companding (Compressing and Expanding) Data

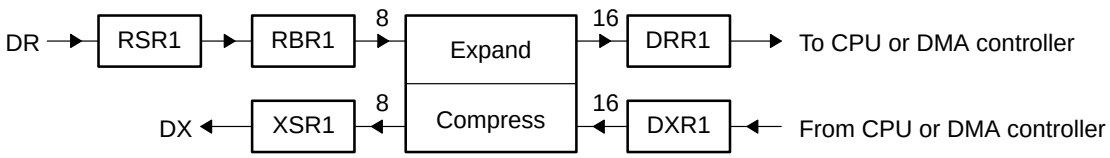
Companding (COMpressing and exPANDing) hardware allows compression and expansion of data in either μ-law or A-law format. The companding standard employed in the United States and Japan is μ-law. The European companding standard is referred to as A-law. The specifications for the μ-law and A-law log PCM are part of the CCITT G.711 recommendation.

A-law and μ-law allow 13 bits and 14 bits of dynamic range, respectively. Any values outside this range are set to the most positive or most negative value. Thus, for companding to work best, the data transferred to and from the McBSP via the CPU or DMA controller must be at least 16 bits wide.

The μ-law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word-length bits (RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

Figure 12–3 illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or μ-law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to twos-complement format.

Figure 12–3. Companding Processes

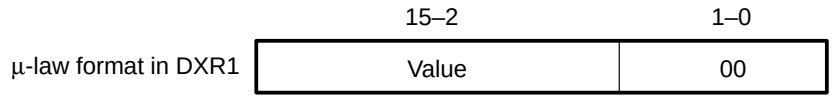


12.2.2.1 Companding Formats

For reception, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1. The receive sign-extension and justification mode specified in RJUST is ignored when companding is used.

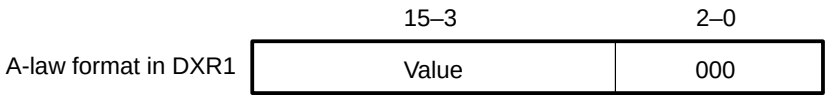
For transmission using μ-law compression, the 14 data bits must be left justified in DXR1 and the remaining two low-order bits are filled with zeros, as shown in Figure 12–4.

Figure 12–4. μ-Law Transmit Data Companding Format



For transmission using A-law compression, the 13 data bits must be left-justified in DXR1, with the remaining three low-order bits filled with zeros, as shown in Figure 12–5.

Figure 12–5. A-Law Transmit Data Companding Format



12.2.2.2 Capability to Compand Internal Data

If the McBSP is otherwise unused (the serial port transmit and receive sections are reset), the companding hardware can compand internal data. This can be used to:

- Convert linear to the appropriate μ-law or A-law format
- Convert μ-law or A-law to the linear format
- Observe the quantization effects in companding by transmitting linear data and compressing and reexpanding this data. This is useful only if both XCOMPAND and RCOMPAND enable the same companding format.

Figure 12–6 shows two methods by which the McBSP can compand internal data. Data paths for these two methods are used to indicate:

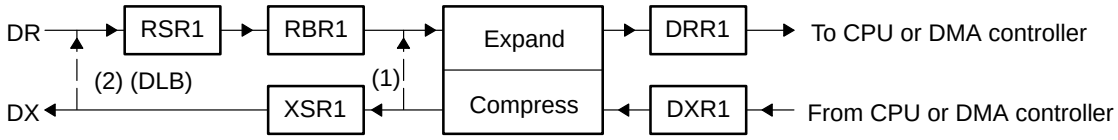
- When both the transmit and receive sections of the serial port are reset, DRR1 and DXR1 are connected internally through the companding logic. Values from DXR1 are compressed, as selected by XCOMPAND, and then expanded, as selected by RCOMPAND. RRDY and XRDY bits are not set. However, data is available in DRR1 within four CPU clocks after being written to DXR1.

  The advantage of this method is its speed. The disadvantage is that there is no synchronization available to the CPU and DMA to control the flow.

DRR1 and DXR1 are internally connected if the (X/R)COMPAND bits are set to 10b or 11b (compand using μ-law or A-law).

- The McBSP is enabled in digital loopback mode with companding appropriately enabled by RCOMPAND and XCOMPAND. Receive and transmit interrupts (RINT when RINTM = 0 and XINT when XINTM = 0) or synchronization events (REVT and XEVT) allow synchronization of the CPU or DMA to these conversions, respectively. Here, the time for this companding depends on the serial bit rate selected.

Figure 12–6. Two Methods by Which the McBSP Can Compand Internal Data



12.2.2.3 Reversing Bit Order: Option to Transfer LSB First

Generally, the McBSP transmits or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols (that do not use companded data) require the least significant bit (LSB) to be transferred first. If you set XCOMPAND = 01b in XCR2, the bit ordering of 8-bit words is reversed (LSB first) before being sent from the serial port. If you set RCOMPAND = 01b in RCR2, the bit ordering of 8-bit words is reversed during reception. Similar to companding, this feature is enabled only if the appropriate word length bits are set to 0, indicating that 8-bit words are to be transferred serially. If either phase of the frame does not have an 8-bit word length, the McBSP assumes the word length is 8 bits, and LSB-first ordering is done.

12.2.3 Clocking and Framing Data

This section explains basic concepts and terminology important for understanding how McBSP data transfers are timed and delimited.

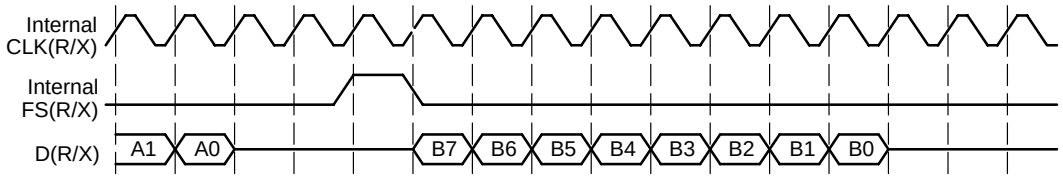
12.2.3.1 Clocking

Data is shifted one bit at a time from the DR pin to the RSR(s) or from the XSR(s) to the DX pin. The time for each bit transfer is controlled by the rising or falling edge of a clock signal.

The receive clock signal (CLKR) controls bit transfers from the DR pin to the RSR(s). The transmit clock signal (CLKX) controls bit transfers from the XSR(s) to the DX pin. CLKR or CLKX can be derived from a pin at the boundary of the McBSP or derived from inside the McBSP. The polarities of CLKR and CLKX are programmable.

In the example in Figure 12–7, the clock signal controls the timing of each bit transfer on the pin.

Figure 12–7. Example—Clock Signal Control of Bit Transfer Timing



Note:

The McBSP cannot operate at a frequency faster than 1/2 of the CPU clock frequency. When driving CLKX or CLKR at the pin, choose an appropriate input clock frequency. When using the internal sample rate generator for CLKX and/or CLKR, choose an appropriate input clock frequency and divide-down value (CLKDV).

12.2.3.2 Serial Words

Bits traveling between a shift register (RSR or XSR) and a data pin (DR or DX) are transferred in a group called a serial word. You can define how many bits are in a word.

Bits coming in on the DR pin are held in RSR until RSR holds a full serial word. Only then is the word passed to RBR (and ultimately to the DRR).

During transmission, XSR does not accept new data from DXR until a full serial word has been passed from XSR to the DX pin.

In the example in Figure 12–7, an 8-bit word size was defined (see bits 7 through 0 of word B being transferred).

12.2.3.3 Frames and Frame Synchronization

One or more words are transferred in a group called a frame. You can define how many words are in a frame.

All of the words in a frame are sent in a continuous stream. However, there can be pauses between frame transfers. The McBSP uses frame-synchronization signals to determine when each frame is received/transmitted. When a pulse occurs on a frame-synchronization signal, the McBSP begins receiving/transmitting a frame of data. When the next pulse occurs, the McBSP receives/transmits the next frame, and so on.

Pulses on the receive frame-synchronization signal (FSR) initiate frame transfers on DR. Pulses on the transmit frame-sync signal (FSX) initiate frame transfers on DX. FSR or FSX can be derived from a pin at the boundary of the McBSP or derived from inside the McBSP.

In the example in Figure 12–7, a one-word frame is transferred when a frame-synchronization pulse occurs.

In McBSP operation, the inactive-to-active transition of the frame-synchronization signal indicates the start of the next frame. For this reason, the frame-synchronization signal may be high for an arbitrary number of clock cycles. Only after the signal is recognized to have gone inactive, and then active again, does the next frame synchronization occur.

12.2.3.4 Detecting Frame-Synchronization Pulses, Even in Reset State

The McBSP can send receive and transmit interrupts to the CPU to indicate specific events in the McBSP. To facilitate detection of frame synchronization, these interrupts can be sent in response to frame-synchronization pulses. Set the appropriate interrupt mode bits to 10b (for reception, RINTM = 10b; for transmission, XINTM = 10b).

Unlike other serial port interrupt modes, this mode can operate while the associated portion of the serial port is in reset (such as activating RINT when the receiver is in reset). In this case, FSRM/FSXM and FSRP/FSXP still select the appropriate source and polarity of frame synchronization. Thus, even when the serial port is in the reset state, these signals are synchronized to the CPU clock and then sent to the CPU in the form of RINT and XINT at the point at which they feed the receiver and transmitter of the serial port. Consequently, a new frame-synchronization pulse can be detected, and after this occurs the CPU can take the serial port out of reset safely.

12.2.3.5 Ignoring Frame-Synchronization Pulses

The McBSP can be configured to ignore transmit and/or receive frame-synchronization pulses. To have the receiver or transmitter recognize frame-synchronization pulses, clear the appropriate frame-synchronization ignore bit (RFIG = 0 for the receiver, XFIG = 0 for the transmitter). To have the receiver or transmitter ignore frame-synchronization pulses until the desired frame length or number of words is reached, set the appropriate frame-synchronization ignore bit (RFIG = 1 for the receiver, XFIG = 1 for the transmitter). For more details on unexpected frame-synchronization pulses, see Section 12.4.2, *Unexpected Receive Frame-Synchronization Pulse*, or Section 12.4.5, *Unexpected Transmit Frame-Synchronization Pulse.*

You can also use the frame-synchronization ignore function for data packing (for more details, see Section 12.11.2).

12.2.3.6 Frame Frequency

The frame frequency is determined by the period between frame-synchronization pulses and is defined as shown by Equation 12–1.

Equation 12–1. McBSP Frame Frequency

$$\text{Frame Frequency} = \frac{\text{Clock Frequency}}{\text{Number of Clock Cycles Between Frame–Sync Pulses}}$$

The frame frequency can be increased by decreasing the time between frame-synchronization pulses (limited only by the number of bits per frame). As the frame transmit frequency increases, the inactivity period between the data packets for adjacent transfers decreases to zero.

12.2.3.7 Maximum Frame Frequency

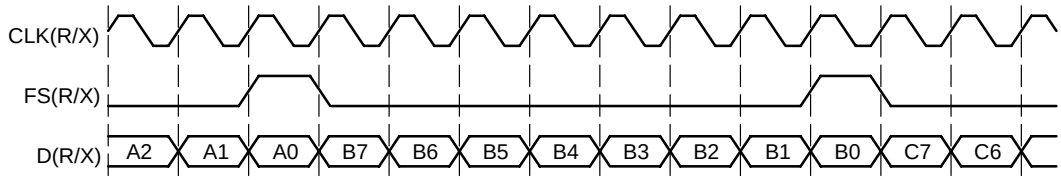
The minimum number of clock cycles between frame synchronization pulses is equal to the number of bits transferred per frame. The maximum frame frequency is defined as shown by Equation 12–2.

Equation 12–2. McBSP Maximum Frame Frequency

$$\text{Maximum Frame Frequency} = \frac{\text{Clock Frequency}}{\text{Number of Bits Per Frame}}$$

Figure 12–8 shows the McBSP operating at maximum packet frequency. At maximum packet frequency, the data bits in consecutive packets are transmitted contiguously with no inactivity between bits.

Figure 12–8. McBSP Operating at Maximum Packet Frequency



If there is a 1-bit data delay as shown in Figure 12–8, the frame-synchronization pulse overlaps the last bit transmitted in the previous frame. Effectively, this permits a continuous stream of data, making frame-synchronization pulses redundant. Theoretically, only an initial frame-synchronization pulse is required to initiate a multipacket transfer.

The McBSP supports operation of the serial port in this fashion by ignoring the successive frame-synchronization pulses. Data is clocked into the receiver or clocked out of the transmitter during every clock cycle.

Note:

For XDATDLY = 0 (0-bit data delay), the first bit of data is transmitted asynchronously to the internal transmit clock signal (CLKX). For more details, see Section 12.8.12, *Set the Transmit Data Delay*.

12.2.4 Frame Phases

The McBSP allows you to configure each frame to contain one or two phases. The number of words per frame and the number of bits per word can be specified differently for each of the two phases of a frame, allowing greater flexibility in structuring data transfers. For example, a user might define a frame as consisting of one phase containing two words of 16 bits each, followed by a second phase consisting of ten words of 8 bits each. This configuration permits the user to compose frames for custom applications or, in general, to maximize the efficiency of data transfers.

12.2.4.1 Number of Phases, Words, and Bits Per Frame

Table 12–2 shows which bit-fields in the receive control registers (RCR1 and RCR2) and in the transmit control registers (XCR1 and XCR2) determine the number of phases per frame, the number of words per frame, and the number of bits per word for each phase, for the receiver and transmitter. The maximum number of words per frame is 128 for a single-phase frame and 256 for a dual-phase frame. The number of bits per word can be 8, 12, 16, 20, 24, or 32 bits.

Table 12–2. McBSP Register Bits

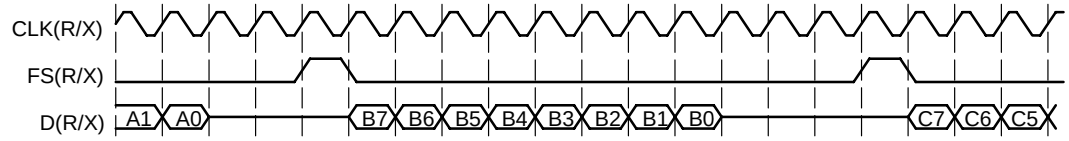
Operation	Number of Phases	Words per Frame Set With	Bits per Word Set With
Reception	1 (RPHASE = 0)	RFRLEN1	RWDLEN1
Reception	2 (RPHASE = 1)	RFRLEN1 and RFRLEN2	RWDLEN1 for phase 1 RWDLEN2 for phase 2
Transmission	1 (XPHASE = 0)	XFRLEN1	XWDLEN1
Transmission	2 (XPHASE = 1)	XFRLEN1 and XFRLEN2	XWDLEN1 for phase 1 XWDLEN2 for phase 2

12.2.4.2 Single-Phase Frame Example

Figure 12–9 shows an example of a single-phase data frame containing one 8-bit word. Because the transfer is configured for 1-bit data delay, the data on the DX and DR pins are available one clock cycle after FS(R/X) goes active. The figure makes the following assumptions:

- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLEN1 = 0b: 1 word per frame
- (R/X)WDLEN1 = 000b: 8-bit word length
- (R/X)FRLEN2 and (R/X)WDLEN2 are ignored
- CLK(X/R)P = 0: Receive data clocked on falling edge; transmit data clocked on rising edge
- FS(R/X)P = 0: Active-high frame-synchronization signals
- (R/X)DATDLY = 01b: 1-bit data delay

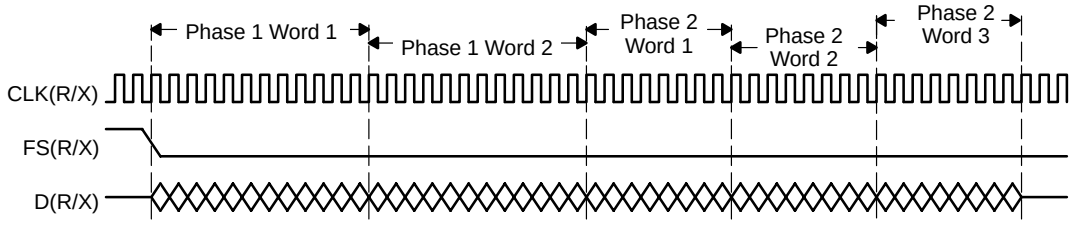
Figure 12–9. Single-Phase Frame for a McBSP Data Transfer



12.2.4.3 Dual-Phase Frame Example

Figure 12–10 shows an example of a frame where the first phase consists of two words of 12 bits each, followed by a second phase of three words of 8 bits each. The entire bit stream in the frame is contiguous. There are no gaps either between words or between phases.

Figure 12–10. Dual-Phase Frame for a McBSP Data Transfer

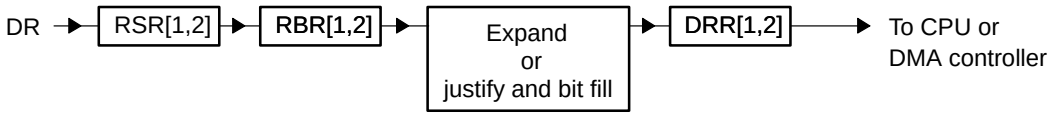


12.2.5 McBSP Reception

This section explains the fundamental process of reception in the McBSP. For details about how to program the McBSP receiver, see Section 12.7, *Receiver Configuration*.

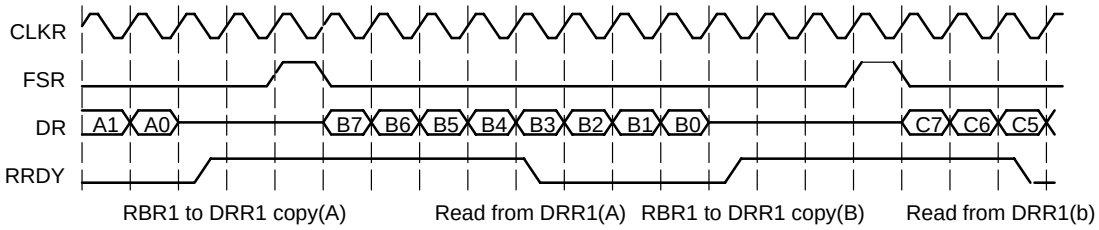
Figure 12–11 and Figure 12–12 show how reception occurs in the McBSP. Figure 12–11 shows the physical path for the data. Figure 12–12 is a timing diagram showing signal activity for one possible reception scenario. A description of the process follows the figures.

Figure 12–11. McBSP Reception Physical Data Path



RSR[1,2]: Receive shift registers 1 and 2
RBR[1,2]: Receive buffer registers 1 and 2
DRR[1,2]: Data receive registers 1 and 2

Figure 12–12. McBSP Reception Signal Activity



CLKR: Internal receive clock
FSR: Internal receive frame-synchronization signal
DR: Data on DR pin
RRDY: Status of receiver ready bit (high is 1)

The following process describes how data travels from the DR pin to the CPU or to the DMA controller:

1) The McBSP waits for a receive frame-synchronization pulse on internal FSR.

2) When the pulse arrives, the McBSP inserts the appropriate data delay that is selected with the RDATDLY bits of RCR2.

   In the preceding timing diagram (Figure 12–12), a 1-bit data delay is selected.

3) The McBSP accepts data bits on the DR pin and shifts them into the receive shift register(s).

   If the word length is 16 bits or smaller, only RSR1 is used. If the word length is larger than 16 bits, RSR2 and RSR1 are used and RSR2 contains the most significant bits. For details on choosing a word length, see Section 12.7.8, *Set the Receive Word Length(s)*.

4) When a full word is received, the McBSP copies the contents of the receive shift register(s) to the receive buffer register(s), provided that RBR1 is not full with previous data.

   If the word length is 16 bits or smaller, only RBR1 is used. If the word length is larger than 16 bits, RBR2 and RBR1 are used and RBR2 contains the most significant bits.

5) The McBSP copies the contents of the receive buffer register(s) into the data receive register(s), provided that DRR1 is not full with previous data. When DRR1 receives new data, the receiver ready bit (RRDY) is set in SPCR1. This indicates that receive data is ready to be read by the CPU or the DMA controller.

If the word length is 16 bits or smaller, only DRR1 is used. If the word length is larger than 16 bits, DRR2 and DRR1 are used and DRR2 contains the most significant bits.

If companding is used during the copy (RCOMPAND = 10b or 11b in RCR2), the 8-bit compressed data in RBR1 is expanded to a left-justified 16-bit value in DRR1. If companding is disabled, the data copied from RBR[1,2] to DRR[1,2] is justified and bit filled according to the RJUST bits.

6) The CPU or the DMA controller reads the data from the data receive register(s). When DRR1 is read, RRDY is cleared and the next RBR-to-DRR copy is initiated.

Note:

If both DRRs are required (word length larger than 16 bits), the CPU or the DMA controller must read from DRR2 first and then from DRR1. As soon as DRR1 is read, the next RBR-to-DRR copy is initiated. If DRR2 is not read first, the data in DRR2 is lost.

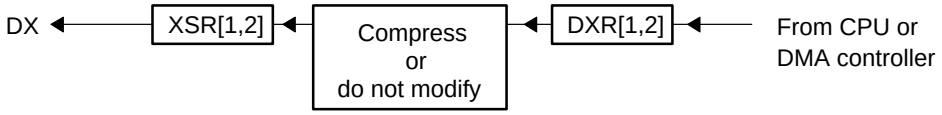
When activity is not properly timed, errors can occur. See Section 12.4.1, *Overrun in the Receiver*, or Section 12.4.2, *Unexpected Receive Frame-Synchronization Pulse*, for more details.

12.2.6 McBSP Transmission

This section explains the fundamental process of transmission in the McBSP. For details about how to program the McBSP transmitter, see Section 12.8, *Transmitter Configuration*.

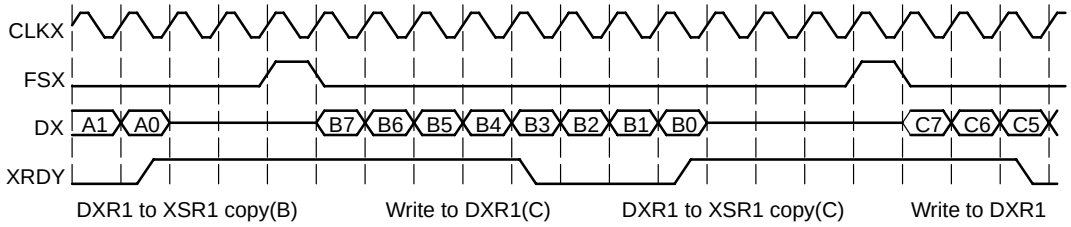
Figure 12–13 and Figure 12–14 show how transmission occurs in the McBSP. Figure 12–13 shows the physical path for the data. Figure 12–14 is a timing diagram showing signal activity for one possible transmission scenario. A description of the process follows the figures.

Figure 12–13. McBSP Transmission Physical Data Path



XSR[1,2]: Transmit shift registers 1 and 2 DXR[1,2]: Data transmit registers 1 and 2

Figure 12–14. McBSP Transmission Signal Activity



CLKX: Internal transmit clock
FSX: Internal transmit frame-synchronization signal
DX: Data on DX pin
XRDY: Status of transmitter ready bit (high is 1)

1) The CPU or the DMA controller writes data to the data transmit register(s). When DXR1 is loaded, the transmitter ready bit (XRDY) is cleared in SPCR2 to indicate that the transmitter is not ready for new data.

   If the word length is 16 bits or smaller, only DXR1 is used. If the word length is larger than 16 bits, DXR2 and DXR1 are used and DXR2 contains the most significant bits. For details on choosing a word length, see Section 12.8.8, *Set the Transmit Word Length(s)*.

Note:

If both DXRs are needed (word length larger than 16 bits), the CPU or the DMA controller must load DXR2 first and then load DXR1. As soon as DXR1 is loaded, the contents of both DXRs are copied to the transmit shift registers (XSRs), as described in the next step. If DXR2 is not loaded first, the previous content of DXR2 is passed to the XSR2.

2) When new data arrives in DXR1, the McBSP copies the content of the data transmit register(s) to the transmit shift register(s). In addition, the transmit ready bit (XRDY) is set. This indicates that the transmitter is ready to accept new data from the CPU or the DMA controller.

   If the word length is 16 bits or smaller, only XSR1 is used. If the word length is larger than 16 bits, XSR2 and XSR1 are used and XSR2 contains the most significant bits.

   If companding is used during the transfer (XCOMPAND = 10b or 11b in XCR2), the McBSP compresses the 16-bit data in DXR1 to 8-bit data in μ-law or A-law format in XSR1. If companding is disabled, the McBSP passes data from the DXR(s) to the XSR(s) without modification.

3) The McBSP waits for a transmit frame-synchronization pulse on internal FSX.

4) When the pulse arrives, the McBSP inserts the appropriate data delay that is selected with the XDATDLY bits of XCR2.

   In the preceding timing diagram (Figure 12–14), a 1-bit data delay is selected.

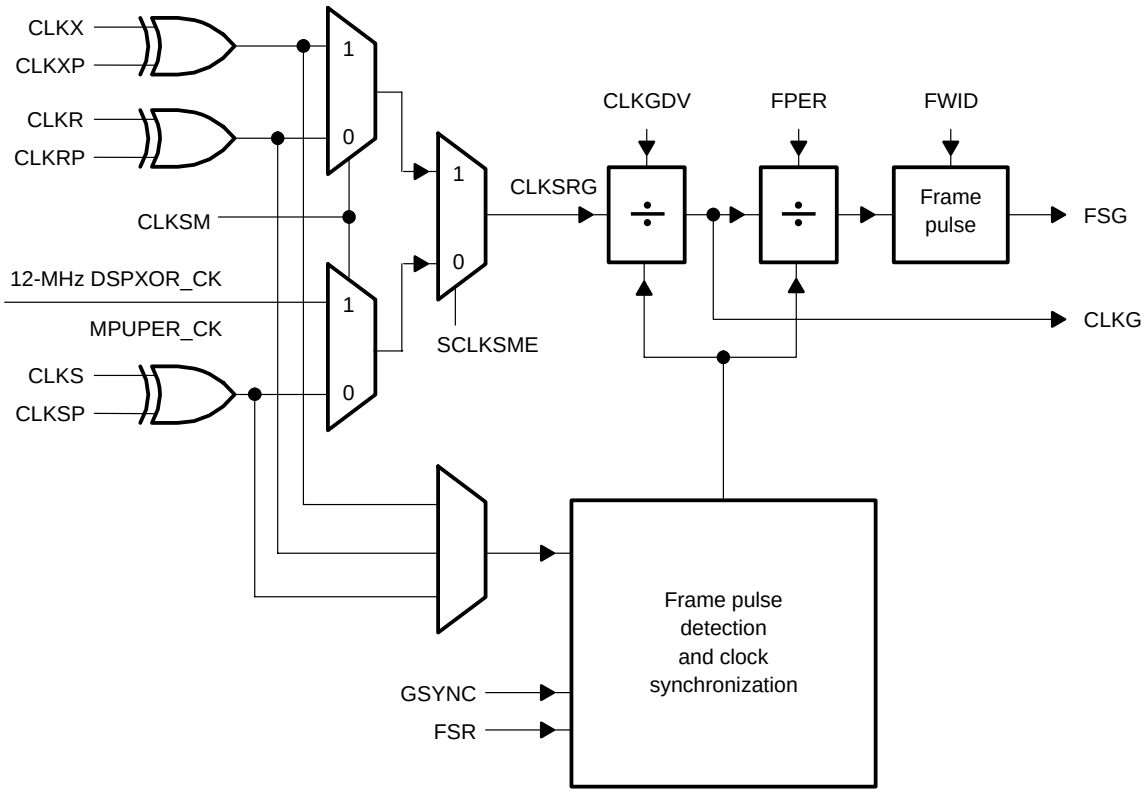
5) The McBSP shifts data bits from the transmit shift register(s) to the DX pin.

When activity is not properly timed, errors can occur. See Section 12.4.3, *Overwrite in the Transmitter*, Section 12.4.4, *Underflow in the Transmitter*, or Section 12.4.5, *Unexpected Transmit Frame-Synchronization Pulse*, for more details.

12.3 McBSP Sample Rate Generator

Each McBSP contains a sample rate generator that can be used to generate an internal data clock (CLKG) and an internal frame-synchronization signal (FSG). CLKG can be used for bit shifting on the data receive (DR) pin and/or the data transmit (DX) pin. FSG can be used to initiate frame transfers on DR and/or DX. Figure 12–15 is a conceptual block diagram of the sample rate generator.

Figure 12–15. Conceptual Block Diagram of the Sample Rate Generator



The source clock for the sample rate generator (labeled CLKSRG in the diagram) can be supplied by either the 12-MHz DSPXOR_CK clock, the ARMPER_CK clock, or by an external pin (CLKS, CLKX, or CLKR). The source is selected with the SCLKME bit of PCR and the CLKSM bit of SRGR2. If a pin is used, the polarity of the incoming signal can be inverted with the appropriate polarity bit (CLKSP of SRGR2, CLKXP of PCR, or CLKRP of PCR).

The sample rate generator has a three-stage clock divider that gives CLKG and FSG programmability. The three stages provide:

- Clock divide-down. The source clock is divided according to the CLKGDV bits of SRGR1 to produce CLKG.
- Frame period divide-down. CLKG is divided according to the FPER bits of SRGR2 to control the period from the start of a frame-pulse to the start of the next pulse.

- Frame-synchronization pulse-width countdown. CLKG cycles are counted according to the FWID bits of SRGR1 to control the width of each frame-synchronization pulse.

Note:

The McBSP cannot operate at a frequency faster than 1/2 of the clock frequency. Choose an input clock frequency and a CLKDV value such that CLKG is less than or equal to 1/2 of the clock frequency.

In addition to the three-stage clock divider, the sample rate generator has a frame-synchronization pulse detection and clock synchronization module that allows synchronization of the clock divide-down with an incoming frame-synchronization pulse on the FSR pin. This feature is enabled or disabled with the GSYNC bit of SRGR2.

For details on getting the sample rate generator ready for operation, see Section 12.3.4, *Reset and Initialization Procedure*.

12.3.1 Clock Generation in the Sample Rate Generator

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both. Use of the sample rate generator to drive clocking is controlled by the clock mode bits (CLKRM and CLKXM) in the pin control register (PCR). When a clock mode bit is set to 1 (CLKRM = 1 for reception, CLKXM = 1 for transmission), the corresponding data clock (CLKR for reception, CLKX for transmission) is driven by the internal sample rate generator output clock (CLKG).

The effects of CLKRM = 1 and CLKXM = 1 on the McBSP are partially affected by the use of the digital loopback mode and the clock stop (SPI) mode, respectively, as described in Table 12–3. The digital loopback mode (described in Section 12.7.4) is selected with the DLB bit of SPCR1. The clock stop mode (described in Section 12.6) is selected with the CLKSTP bits of SPCR1.

When using the sample rate generator as a clock source, make sure the sample rate generator is enabled ($\overline{\text{GRST}}$ = 1).

Table 12–3. Effects of DLB and CLKSTP on Clock Modes

Mode Bit Settings		Effect
CLKRM = 1	DLB = 0 (Digital loopback mode disabled)	CLKR is an output pin driven by the sample rate generator output clock (CLKG).
	DLB = 1 (Digital loopback mode enabled)	CLKR is an output pin driven by internal CLKX. The source for CLKX depends on the CLKXM bit.
CLKXM = 1	CLKSTP = 00b or 01b (Clock stop (SPI) mode disabled)	CLKX is an output pin driven by the sample rate generator output clock (CLKG).
	CLKSTP = 10b or 11b (Clock stop (SPI) mode enabled)	The McBSP is a master in an SPI system. Internal CLKX drives internal CLKR and the shift clocks of any SPI-compliant slave devices in the system. CLKX is driven by the internal sample rate generator.

12.3.1.1 Choosing an Input Clock

The sample rate generator must be driven by an input clock signal from one of the four sources selectable with the SCLKME bit of PCR and the CLKSM bit of SRGR2 (see Table 12–4). When CLKSM = 1, the minimum divide-down value in CLKGDV bits is 1. CLKGDV is described in Section 12.3.1.3.

Note:

The McBSP cannot operate at a frequency faster than 1/2 of the CPU clock frequency. Choose an input clock frequency and a CLKDV value such that CLKG is less than or equal to 1/2 of the CPU clock frequency.

Table 12–4. Choosing an Input Clock for the Sample Rate Generator with the SCLKME and CLKSM Bits

SCLKME	CLKSM	Input Clock for Sample Rate Generator
0	0	Signal on CLKS pin
0	1	CPU clock
1	0	Signal on CLKR pin
1	1	Signal on CLKX pin

12.3.1.2 Choosing a Polarity for the Input Clock

As shown in Figure 12–16, when the input clock is received from a pin, you can choose the polarity of the input clock. The rising edge of CLKSRG generates CLKG and FSG, but you can determine which edge of the input clock causes a rising edge on CLKSRG. The polarity options and their effects are described in Table 12–5.

Figure 12–16. Possible Inputs to the Sample Rate Generator and the Polarity Bits

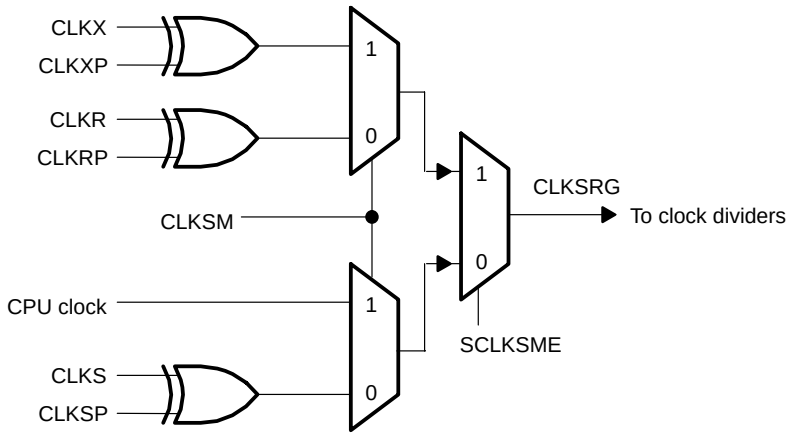


Table 12–5. Polarity Options for the Input to the Sample Rate Generator

Input Clock	Polarity Option	Effect
Signal on CLKS pin	CLKSP = 0 in SRGR2	Rising edge on CLKS pin generates transitions on CLKG and FSG.
	CLKSP = 1 in SRGR2	Falling edge on CLKS pin generates transitions on CLKG and FSG.
CPU clock	Always positive polarity	Rising edge of CPU clock generates transitions on CLKG and FSG.
Signal on CLKR pin	CLKRP = 0 in PCR	Falling edge on CLKR pin generates transitions on CLKG and FSG.
	CLKRP = 1 in PCR	Rising edge on CLKR pin generates transitions on CLKG and FSG.
Signal on CLKX pin	CLKXP = 0 in PCR	Rising edge on CLKX pin generates transitions on CLKG and FSG.
	CLKXP = 1 in PCR	Falling edge on CLKX pin generates transitions on CLKG and FSG.

12.3.1.3 Choosing a Frequency for the Output Clock (CLKG)

The input clock (CPU clock or external clock) can be divided down by a programmable value to drive CLKG. Regardless of the source to the sample rate generator, the rising edge of CLKSRG (see Figure 12–15) generates CLKG and FSG.

The first divider stage of the sample rate generator creates the output clock from the input clock. This divider stage uses a counter that is preloaded with the divide down value in the CLKGDV bits of SRGR1. The output of this stage is the data clock (CLKG). CLKG has the frequency represented by the following equation:

$$\text{CLKG frequency} = \frac{\text{Input clock frequency}}{(\text{CLKGDV} + 1)}$$

Thus, the input clock frequency is divided by a value between 1 and 256. When CLKGDV is odd or equal to 0, the CLKG duty cycle is 50%. When CLKGDV is an even value, 2p, representing an odd divide-down, the high-state duration is p+1 cycles and the low-state duration is p cycles.

Note:

The McBSP cannot operate at a frequency faster than 1/2 of the CPU clock frequency. Choose an input clock frequency and a CLKDV value such that CLKG is less than or equal to 1/2 of the CPU clock frequency.

12.3.1.4 Keeping CLKG Synchronized to an External Input Clock

When an external signal is selected to drive the sample rate generator (see Section 12.3.1.1), the GSYNC bit in SRGR2 and the FSR pin can be used to configure the timing of the output clock (CLKG) relative to the input clock.

GSYNC = 1 ensures that the McBSP and an external device are dividing down the input clock with the same phase relationship. If GSYNC = 1, an inactive-to-active transition on the FSR pin triggers a resynchronization of CLKG and generation of FSG.

For more details about synchronization, see Section 12.3.3, *Synchronizing Sample Rate Generator Outputs to an External Clock*.

12.3.2 Frame-Synchronization Generation in the Sample Rate Generator

The sample rate generator can produce a frame-synchronization signal (FSG) for use by the receiver, the transmitter, or both.

If you want the receiver to use FSG for frame synchronization, make sure FSRM = 1. (When FSRM = 0, receive frame synchronization is supplied via the FSR pin.)

If you want the transmitter to use FSG for frame synchronization, you must set:

- FSXM = 1 in PCR. This indicates that transmit frame synchronization is supplied by the McBSP itself rather than by the FSX pin.
- FSGM = 1 in SRGR2. This indicates that when FSXM = 1, transmit frame synchronization is supplied by the sample rate generator. (When FSGM = 0 and FSXM = 1, the transmitter uses frame-synchronization pulses generated every time data is transferred from DXR[1,2] to XSR[1,2].)

In either case, the sample rate generator must be enabled ($\overline{\text{GRST}}$ = 1) and the frame-synchronization logic in the sample rate generator must be enabled ($\overline{\text{GRST}}$ = 0).

12.3.2.1 Choosing the Width of the Frame-Synchronization Pulse on FSG

Each pulse on FSG has a programmable width. You program the FWID bits of SRGR1, and the resulting pulse width is (FWID + 1) CLKG cycles, where CLKG is the output clock of the sample rate generator.

12.3.2.2 Controlling the Period Between the Starting Edges of Frame-Synchronization Pulses on FSG

You can control the amount of time from the starting edge of one FSG pulse to the starting edge of the next FSG pulse. This period is controlled in one of two ways, depending on the configuration of the sample rate generator:

- If the sample rate generator is using an external input clock and GSYNC = 1 in SRGR2, FSG pulses in response to an inactive-to-active transition on the FSR pin. Thus, the frame-synchronization period is controlled by an external device.
- Otherwise, you program the FPER bits of SRGR2, and the resulting frame-synchronization period is (FPER + 1) CLKG cycles, where CLKG is the output clock of the sample rate generator.

12.3.2.3 Keeping FSG Synchronized to an External Clock

When an external signal is selected to drive the sample rate generator (see Section 12.3.1.1), the GSYNC bit of SRGR2 and the FSR pin can be used to configure the timing of FSG pulses.

GSYNC = 1 ensures that the McBSP and an external device are dividing down the input clock with the same phase relationship. If GSYNC = 1, an inactive-to-active transition on the FSR pin triggers a resynchronization of CLKG and generation of FSG.

See Section 12.3.3 for more details about synchronization.

12.3.3 Synchronizing Sample Rate Generator Outputs to an External Clock

The sample rate generator can produce a clock signal (CLKG) and a frame-synchronization signal (FSG) based on an input clock signal that is either the CPU clock signal or a signal at the CLKS, CLKR, or CLKX pin. When an external clock is selected to drive the sample rate generator, the GSYNC bit of SRGR2 and the FSR pin can be used to control the timing of CLKG and the pulsing of FSG relative to the chosen input clock.

Make GSYNC = 1 when you want the McBSP and an external device to divide down the input clock with the same phase relationship. If GSYNC = 1:

- ❏ An inactive-to-active transition on the FSR pin triggers a resynchronization of CLKG and a pulsing of FSG.
- ❏ CLKG always begins with a high state after synchronization.
- ❏ FSR is always detected at the same edge of the input clock signal that generates CLKG, no matter how long the FSR pulse is.
- ❏ The FPER bits of SRGR2 are ignored because the frame-synchronization period on FSG is determined by the arrival of the next frame-synchronization pulse on the FSR pin.

If GSYNC = 0, CLKG runs freely and is not resynchronized, and the frame-synchronization period on FSG is determined by FPER.

12.3.3.1 Operating the Transmitter Synchronously with the Receiver

When GSYNC = 1, the transmitter can operate synchronously with the receiver, provided that:

- ❏ FSX is programmed to be driven by FSG (FSGM = 1 in SRGR2 and FSXM = 1 in PCR). If the input FSR has appropriate timing so that it can be sampled by the falling edge of CLKG, it can be used, instead, by setting FSXM = 0 and connecting FSR to FSX externally.
- ❏ The sample rate generator clock drives the transmit and receive clocking (CLKRM = CLKXM = 1 in PCR). Therefore, the CLK(R/X) pin must not be driven by any other driving source.

12.3.3.2 Synchronization Examples

Figure 12–17 and Figure 12–18 show the clock and frame-synchronization operation with various polarities of CLKS (the chosen input clock) and FSR. These figures assume FWID = 0 in SRGR1, for an FSG pulse that is one CLKG cycle wide. The FPER bits of SRGR2 are not programmed; the period from the start of a frame-synchronization pulse to the start of the next pulse is determined by the arrival of the next inactive-to-active transition on the FSR pin. Each of the figures shows what happens to CLKG when it is initially synchronized and GSYNC = 1, and when it is not initially synchronized and GSYNC = 1. The second figure has a slower CLKG frequency (it has a larger divide-down value in the CLKGDV bits of SRGR1).

Figure 12–17. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 1

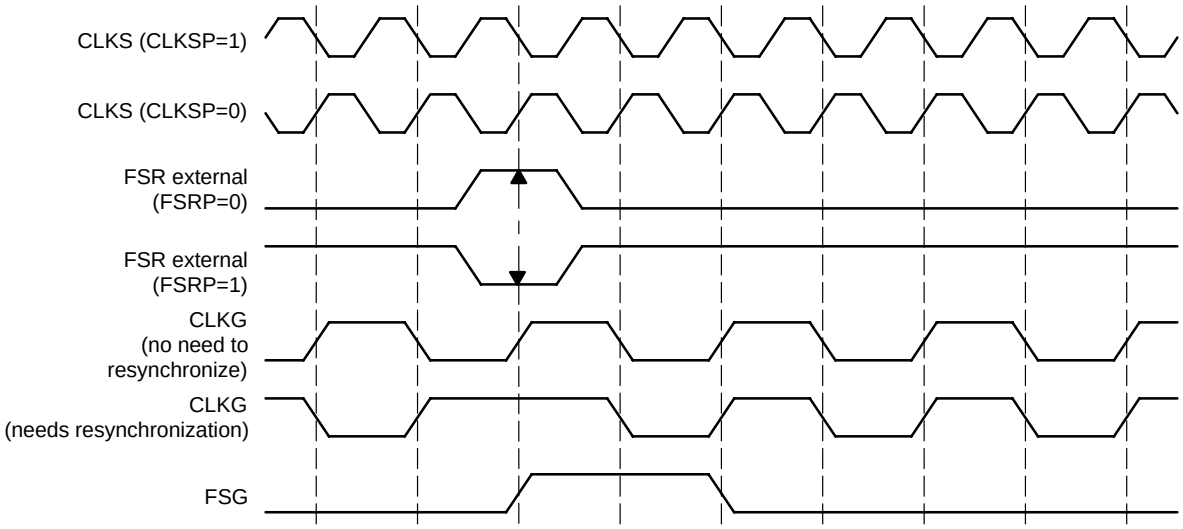
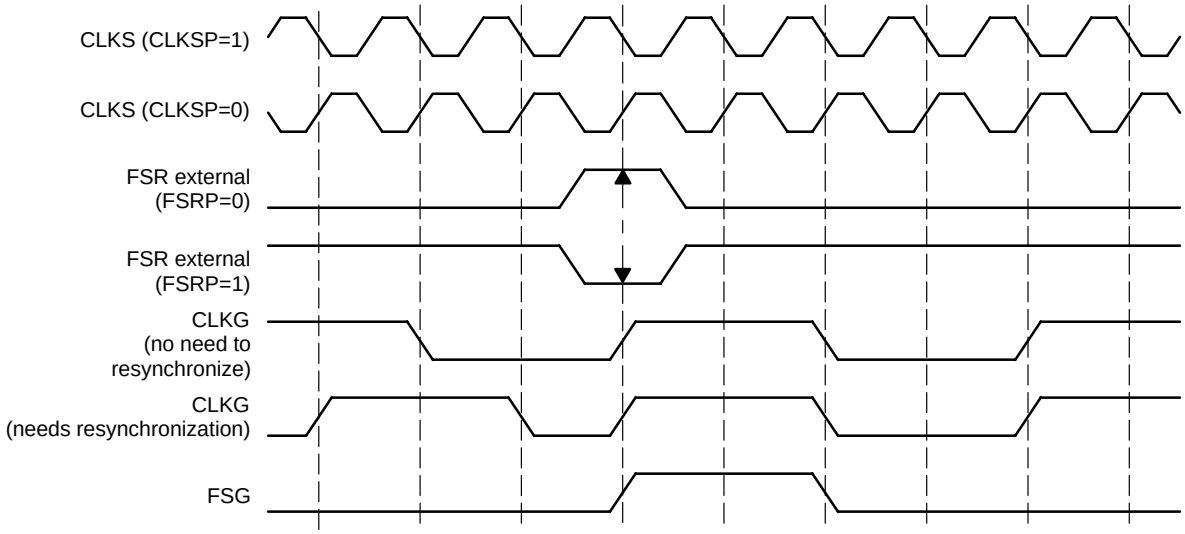


Figure 12–18. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 3



12.3.4 Reset and Initialization Procedure for the Sample Rate Generator

To reset and initialize the sample rate generator:

Step 1: Place the McBSP/sample rate generator in reset.

During an OMAP730 reset, the sample rate generator, the receiver, and the transmitter reset bits ($\overline{\text{GRST}}$, $\overline{\text{RRST}}$, and $\overline{\text{XRST}}$) are automatically forced to 0. Otherwise, during normal operation, the sample rate generator can be reset by making $\overline{\text{GRST}}$ = 0 in SPCR2, provided that CLKG and/or FSG is not used by any portion of the McBSP. Depending on your system you may also want to reset the receiver ($\overline{\text{RRST}}$ = 0 in SPCR1) and reset the transmitter ($\overline{\text{XRST}}$ = 0 in SPCR2).

If $\overline{\text{GRST}}$ = 0 because of an OMAP730 reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven inactive-low. If $\overline{\text{GRST}}$ = 0 because of program code, CLKG and FSG are driven low (inactive).

Step 2: Program the registers that affect the sample rate generator.

Program the sample-rate generator registers (SRGR1 and SRGR2) as required for your application. If necessary, other control registers can be loaded with desired values, provided the respective portion of the McBSP (the receiver or transmitter) is in reset.

After the sample-rate generator registers are programmed, wait 2 CLKSRG cycles. This ensures proper synchronization internally.

Step 3: Enable the sample rate generator (take it out of reset).

In SPCR2, make $\overline{\text{GRST}}$ = 1 to enable the sample rate generator.

After the sample rate generator is enabled, wait two CLKG cycles for the sample-rate generator logic to stabilize.

On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to

$$\text{CLKG frequency} = \frac{\text{Input clock frequency}}{(\text{CLKGDV} + 1)}$$

where the input clock is selected with the SCLKME bit of PCR and the CLKSM bit of SRGR2 in one of the configurations in Table 12–6.

Table 12–6. Input Clock Selection for Sample Rate Generator

SCLKME	CLKSM	Input Clock for Sample Rate Generator
0	0	Signal on CLKS pin
0	1	CPU clock
1	0	Signal on CLKR pin
1	1	Signal on CLKX pin

Step 4: If necessary, enable the receiver and/or the transmitter.

If necessary, remove the receiver and/or transmitter from reset by setting $\overline{\text{RRST}}$ and/or $\overline{\text{XRST}}$ = 1.

Step 5: If necessary, enable the frame-synchronization logic of the sample rate generator.

After the required data acquisition setup is done (DXR[1/2] is loaded with data), set $\overline{\text{GRST}}$ = 1 in SPCR2 if an internally generated frame-synchronization pulse is required. FSG is generated with an active-high edge after the programmed number of CLKG clocks (FPER + 1) has elapsed.

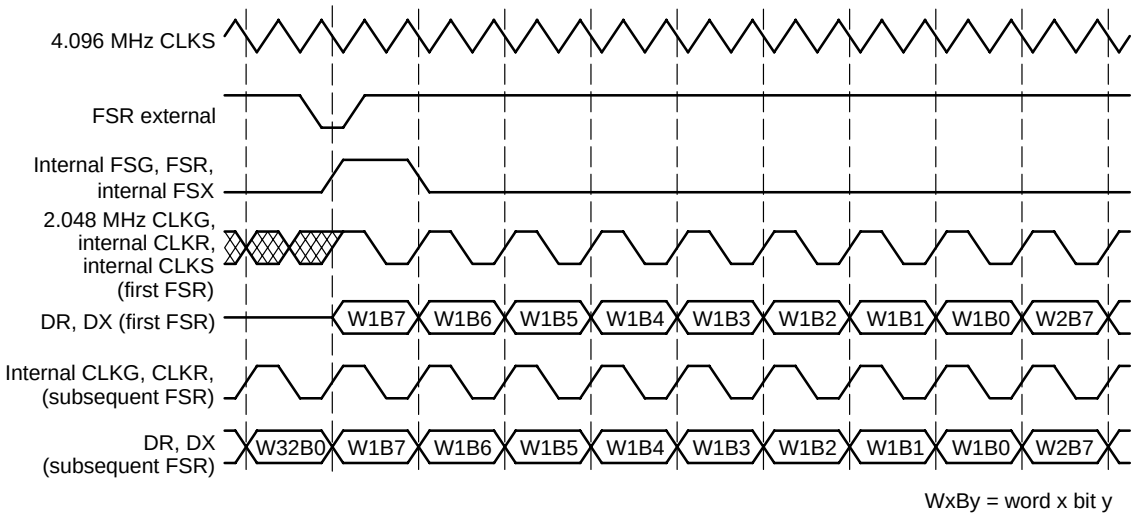
12.3.5 Sample Rate Generator Clocking Examples

This section shows three examples of using the sample rate generator to clock data during transmission and reception.

12.3.5.1 Double-Rate ST-Bus Clock

Figure 12–19 shows McBSP configuration to be compatible with the Mitel ST-Bus. This operation is running at maximum frame frequency.

Figure 12–19. ST-Bus and MVIP Clocking Example



For this McBSP configuration:

- DLB = 0: Digital loopback mode off, CLKSTP = 00b: clock stop mode off, and CLKRM/CLKXM = 1: internal CLKR/CLKX generated internally by sample rate generator
- GSYNC = 1: Synchronize CLKG with the external frame-synchronization signal input on the FSR pin. CLKG is not synchronized until the frame-synchronization signal is active. FSR is regenerated internally to form a minimum pulse width.
- SCLKME = 0 and CLKSM = 1: External clock signal at CLKS pin drives the sample rate generator.
- CLKSP = 1: Falling edge of CLKS generates CLKG and thus internal CLK(R/X).

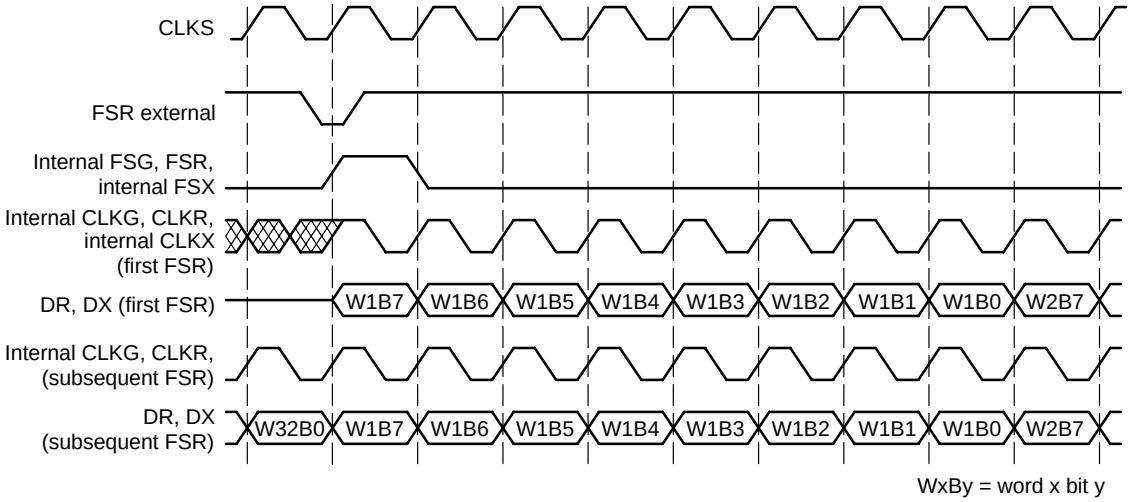
- ❏ CLKGDV = 1: Frequency of receive clock (shown as CLKR) is 1/2 CLKS frequency.
- ❏ FSRP/FSXP = 1: Active-low frame-synchronization pulse
- ❏ RFRLEN1/XFRLEN1 = 11111b: 32 words per frame
- ❏ RWDLEN1/XWDLEN1 = 0: 8 bits per word
- ❏ RPHASE/XPHASE = 0: Single-phase frame and thus (R/X)FRLEN2 and (R/X)WDLEN2 are ignored
- ❏ RDATDLY/XDATDLY = 0: No data delay

12.3.5.2 Single-Rate ST-Bus Clock

The example in Figure 12–20 is the same as the double-rate ST-bus clock example in Section 12.3.5.1 except that:

- ❏ CLKGDV = 0: CLKS drives internal CLK(R/X) without any divide down (single-rate clock).
- ❏ CLKSP = 0: Rising edge of CLKS generates CLKG and internal CLK(R/X).

Figure 12–20. Single-Rate Clock Example



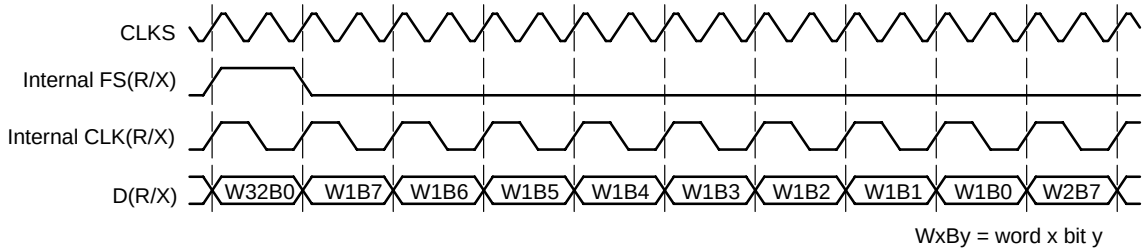
The rising edge of CLKS is used to detect the external FSR pulse, which is used to resynchronize internal McBSP clocks and generate a frame-synchronization pulse for internal use. The internal frame-synchronization pulse is generated so that it is wide enough to be detected on the falling edge of internal clocks.

12.3.5.3 Other Double-Rate Clock

The example in Figure 12–21 is the same as the double-rate ST-bus clock example in Section 12.3.5.1 except that:

- CLKSP = 0: Rising edge of CLKS generates CLKG and thus CLK(R/X).
- CLKGDV = 1: Frequency of CLKG (and thus internal CLKR and internal CLKX) is1/2 CLKS frequency
- FSRM/FSXM = 0: Frame synchronization is externally generated. The frame-synchronization pulse is wide enough to be detected.
- GSYNC = 0: CLKS drives CLKG. CLKG runs freely; it is not resynchronized by a pulse on the FSR pin.
- FSRP/FSXP = 0: Active-high input frame-synchronization signal
- RDATDLY/XDATDLY = 1: Data delay of 1 bit

Figure 12–21. Double-Rate Clock Example



12.4 McBSP Exception/Error Conditions

There are five serial port events that can constitute a system error:

- Receiver overrun (RFULL = 1)

  This occurs when DRR1 has not been read since the last RBR-to-DRR copy. Consequently, the receiver does not copy a new word from the RBR(s) to the DRR(s) and the RSR(s) are now full with another new word shifted in from DR. Therefore, RFULL = 1 indicates an error condition wherein any new data that can arrive at this time on DR replaces the contents of the RSR(s), and the previous word is lost. The RSRs continue to be overwritten as long as new data arrives on DR and DRR1 is not read. For more details about overrun in the receiver, see Section 12.4.1.

- Unexpected receive frame-synchronization pulse (RSYNCERR = 1)

  This occurs during reception when RFIG = 0 and an unexpected frame-synchronization pulse occurs. An unexpected frame-synchronization pulse is one that begins the next frame transfer before all the bits of the current frame have been received. Such a pulse causes data reception to abort and restart. If new data has been copied into the RBR(s) from the RSR(s) since the last RBR-to-DRR copy, this new data in the RBR(s) is lost. This is because no RBR-to-DRR copy occurs; the reception has been restarted. For more details about receive frame-synchronization errors, see Section 12.4.2.

- Transmitter data overwrite

  This occurs when the CPU or DMA controller overwrites data in the DXR(s) before the data is copied to the XSR(s). The overwritten data never reaches the DX pin. For more details about overwrite in the transmitter, see Section 12.4.3.

- Transmitter underflow ($\overline{\text{XEMPTY}}$ = 0)

  If a new frame-synchronization signal arrives before new data is loaded into DXR1, the previous data in the DXR(s) is sent again. This procedure continues for every new frame-synchronization pulse that arrives until DXR1 is loaded with new data. For more details about underflow in the transmitter, see Section 12.4.4.

- Unexpected transmit frame-synchronization pulse (XSYNCERR = 1)

  This occurs during transmission when XFIG = 0 and an unexpected frame-synchronization pulse occurs. An unexpected frame-synchronization pulse is one that begins the next frame transfer before all the bits of the current frame have been transferred. Such a pulse causes the current data transmission to abort and restart. If new data has been written to the DXR(s) since the last DXR-to-XSR copy, the current value in the XSR(s) is lost. For more details about transmit frame-synchronization errors, see Section 12.4.5.

12.4.1 Overrun in the Receiver

RFULL = 1 in SPCR1 indicates that the receiver has experienced overrun and is in an error condition. RFULL is set when all of the following conditions are met:

1) DRR1 has not been read since the last RBR-to-DRR copy (RRDY = 1).
2) RBR1 is full and an RBR-to-DRR copy has not occurred.
3) RSR1 is full and an RSR1-to-RBR copy has not occurred.

As described in Section 12.2.5, *McBSP Reception*, data arriving on DR is continuously shifted into RSR1 (for word length of 16 bits or smaller) or RSR2 and RSR1 (for word length larger than 16 bits). Once a complete word is shifted into the RSR(s), an RSR-to-RBR copy can occur only if the previous data in RBR1 has been copied to DRR1. The RRDY bit is set when new data arrives in DRR1 and is cleared when that data is read from DRR1. Until RRDY = 0, the next RBR-to-DRR copy does not take place, and the data is held in the RSR(s). New data arriving on the DR pin is shifted into RSR(s), and the previous content of the RSR(s) is lost.

You can prevent the loss of data if DRR1 is read no later than 2.5 cycles before the end of the third word is shifted into the RSR1.

Note:

If both DRRs are needed (word length larger than 16 bits), the CPU or the DMA controller must read from DRR2 first and then from DRR1. As soon as DRR1 is read, the next RBR-to-DRR copy is initiated. If DRR2 is not read first, the data in DRR2 is lost.

After the receiver starts running from reset, a minimum of three words must be received before RFULL is set. Either of the following events clears the RFULL bit and allows subsequent transfers to be read properly:

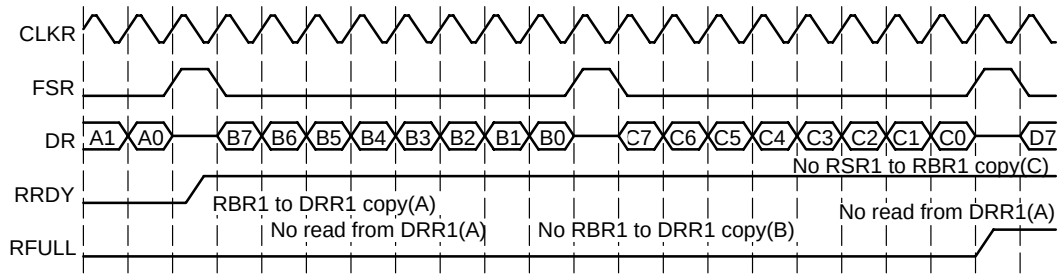
- The CPU or DMA controller reads DRR1.
- The receiver is reset individually ($\overline{\text{RRST}}$ = 0) or as part of an OMAP730 reset.

Another frame-synchronization pulse is required to restart the receiver.

12.4.1.1 Example of Overrun Condition

Figure 12–22 shows the receive overrun condition. Because serial word A is not read from DRR1 before serial word B arrives in RBR1, B is not transferred to DRR1 yet. Another new word (C) arrives and RSR1 is full with this data. DRR1 is finally read, but not earlier than 2.5 cycles before the end of word C. Therefore, new data (D) overwrites word C in RSR1. If DRR1 is not read in time, the next word can overwrite D.

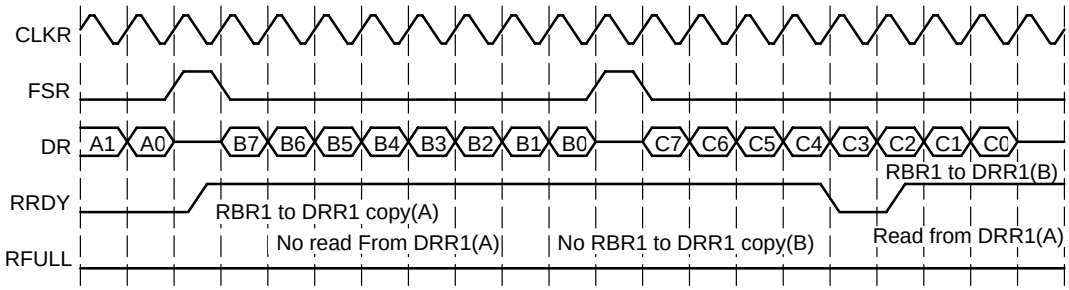
Figure 12–22. Overrun in the McBSP Receiver



12.4.1.2 Example of Preventing Overrun Condition

Figure 12–23 shows the case where RFULL is set, but the overrun condition is prevented by a read from DRR1 at least 2.5 cycles before the next serial word (C) is completely shifted into RSR1. This ensures that an RBR1-to-DRR1 copy of word B occurs before receiver attempts to transfer word C from RSR1 to RBR1.

Figure 12–23. Overrun Prevented in the McBSP Receiver



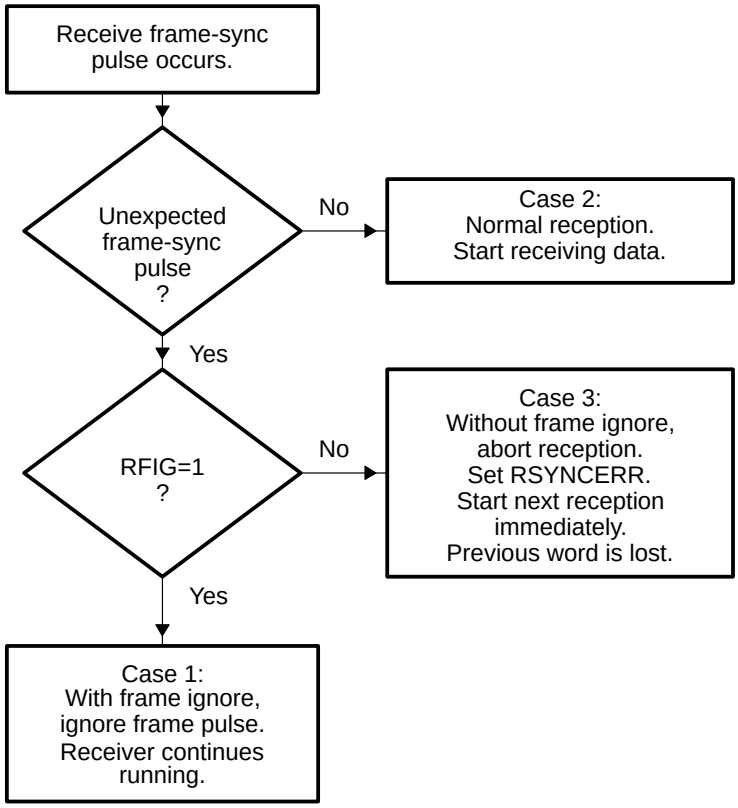
12.4.2 Unexpected Receive Frame-Synchronization Pulse

Section 12.4.2.1 shows how the McBSP responds to any receive frame-synchronization pulses, including an unexpected pulse. Sections 12.4.2.2 and 12.4.2.3 show an example of a frame-synchronization error and how to prevent such an error, respectively.

12.4.2.1 Possible Responses to Receive Frame-Synchronization Pulses

Figure 12–24 shows the decision tree that the receiver uses to handle all incoming frame-synchronization pulses. The figure assumes that the receiver has been started ($\overline{\text{RRST}}$ = 1 in SPCR1). Case 3 in the figure is the case in which an error occurs.

Figure 12–24. Possible Responses to Receive Frame-Synchronization Pulses



Any one of three cases can occur:

- Case 1: Unexpected internal FSR pulses with RFIG = 1 in RCR2. Receive frame-synchronization pulses are ignored, and the reception continues.
- Case 2: Normal serial port reception. Reception continues normally because the frame-synchronization pulse is not unexpected. There are three possible reasons why a receive operation might *not* be in progress when the pulse occurs:
  - The FSR pulse is the first after the receiver is enabled ($\overline{\text{RRST}}$ = 1 in SPCR1).
  - The FSR pulse is the first after DRR[1,2] is read, clearing a receiver full (RFULL = 1 in SPCR1) condition.
  - The serial port is in the interpacket intervals. The programmed data delay for reception (programmed with the RDATDLY bits in RCR2) may start during these interpacket intervals for the first bit of the next word to be received. Thus, at maximum frame frequency, frame synchronization can still be received 0 to 2 clock cycles before the first bit of the synchronized frame.
- Case 3: Unexpected receive frame synchronization with RFIG = 0 (frame-synchronization pulses not ignored). Unexpected frame-synchronization pulses can originate from an external source or from the internal sample rate generator.

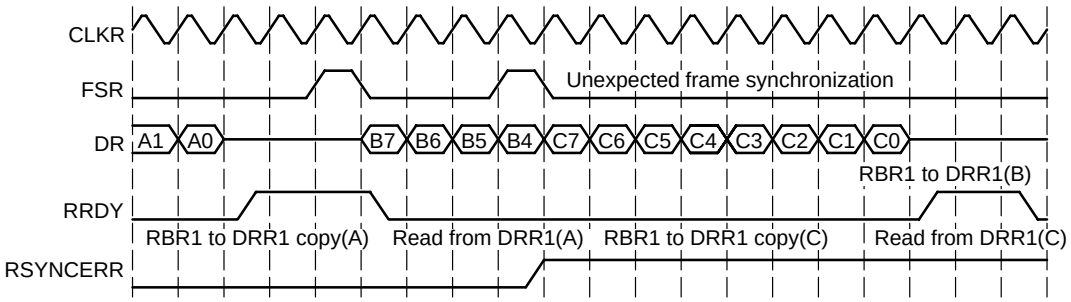
If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-synchronization pulse, and the receiver sets the receive frame-synchronization error bit (RSYNCERR) in SPCR1. RSYNCERR can be cleared only by a receiver reset or by a write of 0 to this bit.

If you want the McBSP to notify the CPU of receive frame-synchronization errors, you can set a special receive interrupt mode with the RINTM bits of SPCR1. When RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU each time that RSYNCERR is set.

12.4.2.2 Example of Unexpected Receive Frame-Synchronization Pulse

Figure 12–25 shows an unexpected receive frame-synchronization pulse during normal operation of the serial port, with time intervals between data packets. When the unexpected frame-synchronization pulse occurs, the RSYNCERR bit is set, the reception of data B is aborted, and the reception of data C begins. In addition, if RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU.

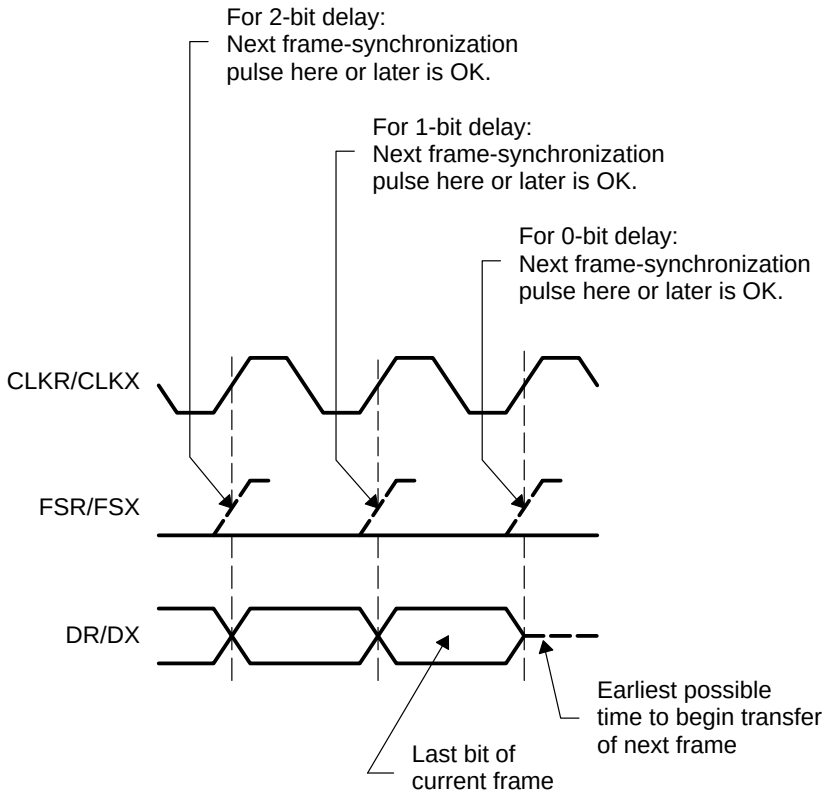
Figure 12–25. An Unexpected Frame-Synchronization Pulse During a McBSP Reception



12.4.2.3 Preventing Unexpected Receive Frame-Synchronization Pulses

Each frame transfer can be delayed by 0, 1, or 2 CLKR cycles, depending on the value in the RDATDLY bits of RCR2. For each possible data delay, Figure 12–26 shows when a new frame-synchronization pulse on FSR can safely occur relative to the last bit of the current frame.

Figure 12–26. Proper Positioning of Frame-Synchronization Pulses



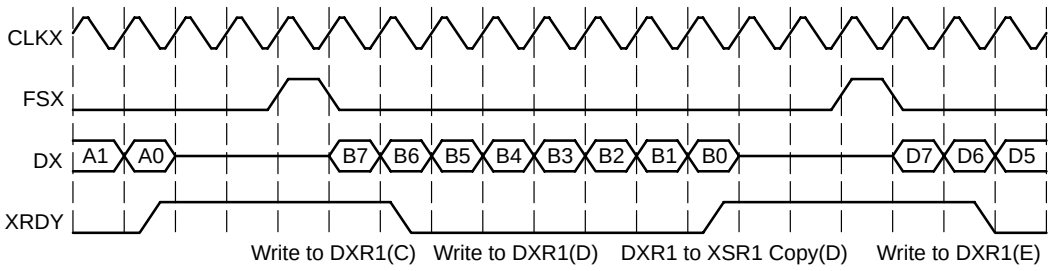
12.4.3 Overwrite in the Transmitter

As described in Section 12.2.6, *McBSP Transmission*, the transmitter must copy the data previously written to the DXR(s) by the CPU or DMA controller into the XSR(s) and then shift each bit from the XSR(s) to the DX pin. If new data is written to the DXR(s) before the previous data is copied to the XSR(s), the previous data in the DXR(s) is overwritten and thus lost.

12.4.3.1 Example of Overwrite Condition

Figure 12–27 shows what happens if the data in DXR1 is overwritten before being transmitted. Initially, DXR1 is loaded with data C. A subsequent write to DXR1 overwrites C with D before C is copied to XSR1. Thus, C is never transmitted on DX.

Figure 12–27. Data in the McBSP Transmitter Overwritten and Thus Not Transmitted



12.4.3.2 Preventing Overwrites

You can prevent CPU overwrites by making the CPU:

- Poll for XRDY = 1 in SPCR2 before writing to the DXR(s). XRDY is set when data is copied from DXR1 to XSR1 and is cleared when new data is written to DXR1.
- Wait for a transmit interrupt (XINT) before writing to the DXR(s). When XINTM = 00b in SPCR2, the transmitter sends XINT to the CPU each time XRDY is set.

You can prevent DMA overwrites by synchronizing DMA transfers to the transmit synchronization event XEVT. The transmitter sends an XEVT signal each time XRDY is set.

12.4.4 Underflow in the Transmitter

The McBSP indicates a transmitter empty (or underflow) condition by clearing the $\overline{\text{XEMPTY}}$ bit in SPCR2. Either of the following events activates $\overline{\text{XEMPTY}}$ ($\overline{\text{XEMPTY}}$ = 0):

- DXR1 has not been loaded since the last DXR-to-XSR copy, and all bits of the data word in the XSR(s) have been shifted out on the DX pin.
- The transmitter is reset (by forcing $\overline{\text{XRST}}$ = 0 in SPCR2, or by an OMAP730 reset) and is then restarted.

In the underflow condition, the transmitter continues to transmit the old data that is in the DXR(s) for every new transmit frame-synchronization signal until a new value is loaded into DXR1 by the CPU or the DMA controller.

Note:

If both DXRs are needed (word length larger than 16 bits), the CPU or the DMA controller must load DXR2 first and then load DXR1. As soon as DXR1 is loaded, the contents of both DXRs are copied to the transmit shift registers (XSRs). If DXR2 is not loaded first, the previous content of DXR2 is passed to the XSR2.

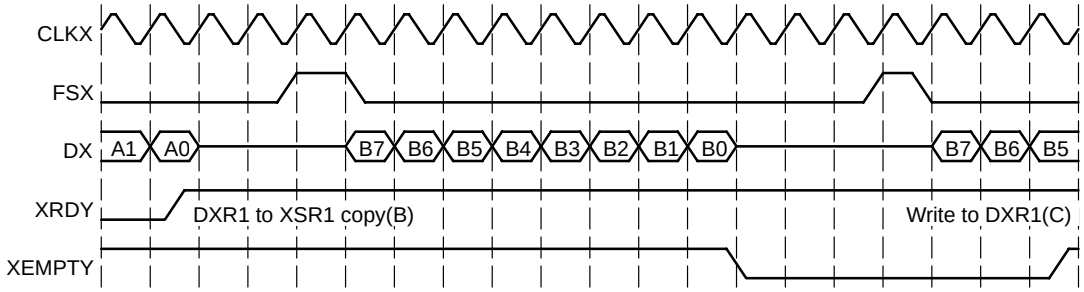
$\overline{\text{XEMPTY}}$ is deactivated ($\overline{\text{XEMPTY}}$ = 1) when a new word in DXR1 is transferred to XSR1. If FSXM = 1 in PCR and FSGM = 0 in SRGR2, the transmitter generates a single internal FSX pulse in response to a DXR-to-XSR copy. Otherwise, the transmitter waits for the next frame-synchronization pulse before sending out the next frame on DX.

When the transmitter is taken out of reset ($\overline{\text{XRST}}$ = 1), it is in a transmitter ready (XRDY = 1 in SPCR2) and transmitter empty ($\overline{\text{XEMPTY}}$ = 0) state. If DXR1 is loaded by the CPU or the DMA controller before internal FSX goes active high, a valid DXR-to-XSR transfer occurs. This allows for the first word of the first frame to be valid even before the transmit frame-synchronization pulse is generated or detected. Alternatively, if a transmit frame-synchronization pulse is detected before DXR1 is loaded, zeros are output on DX.

12.4.4.1 Example of the Underflow Condition

Figure 12–28 shows an underflow condition. After B is transmitted, DXR1 is not reloaded before the subsequent frame-synchronization pulse. Thus, B is again transmitted on DX.

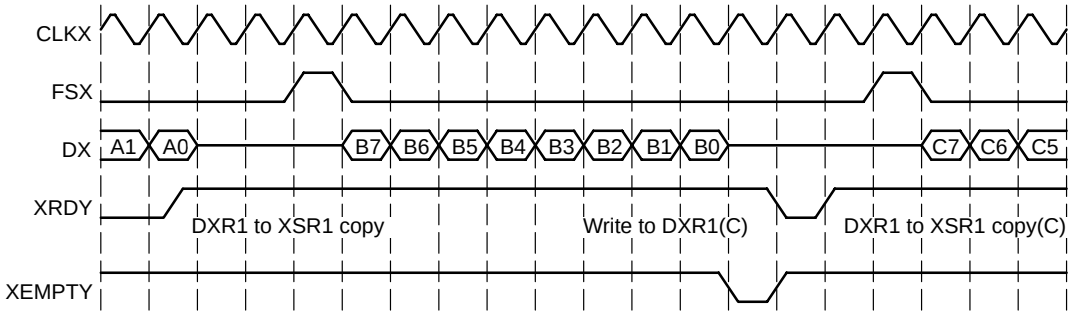
Figure 12–28. Underflow During McBSP Transmission



12.4.4.2 Example of Preventing Underflow Condition

Figure 12–29 shows the case of writing to DXR1 just before an underflow condition would otherwise occur. After B is transmitted, C is written to DXR1 before the next frame-synchronization pulse. As a result, there is no underflow; B is not transmitted twice.

Figure 12–29. Underflow Prevented in the McBSP Transmitter



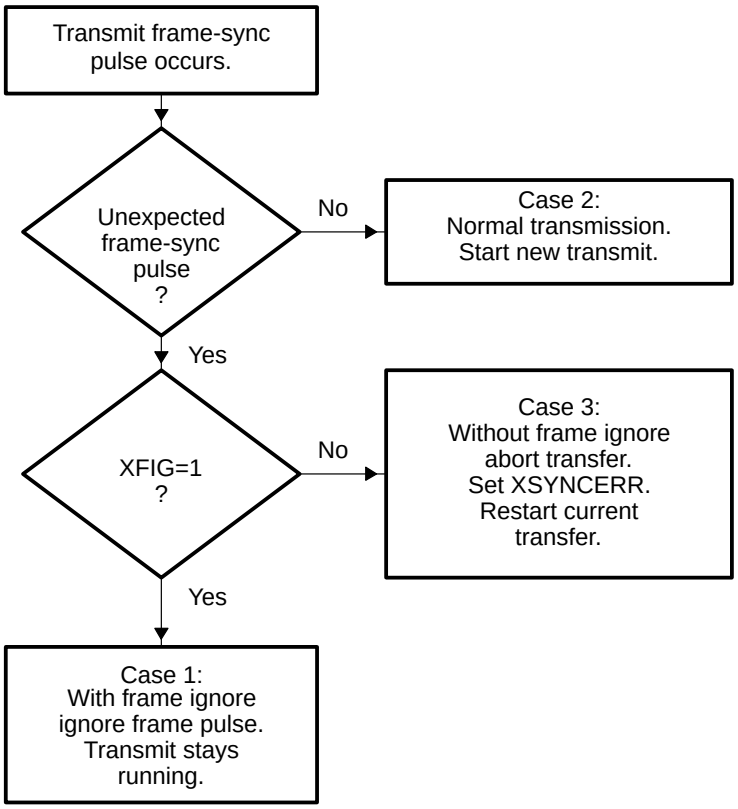
12.4.5 Unexpected Transmit Frame-Synchronization Pulse

Section 12.4.5.1 shows how the McBSP responds to any transmit frame-synchronization pulses, including an unexpected pulse. Sections 12.4.5.2 and 12.4.5.3 show examples of a frame-synchronization error and an example of how to prevent such an error, respectively.

12.4.5.1 Possible Responses to Transmit Frame-Synchronization Pulses

Figure 12–30 shows the decision tree that the transmitter uses to handle all incoming frame-synchronization pulses. The figure assumes that the transmitter has been started ($\overline{\text{XRST}}$ = 1 in SPCR2). Case 3 in the figure is the case in which an error occurs.

Figure 12–30. Possible Responses to Transmit Frame-Synchronization Pulses



Any one of three cases can occur:

- Case 1: Unexpected internal FSX pulses with XFIG = 1 in XCR2. Transmit frame-synchronization pulses are ignored, and the transmission continues.
- Case 2: Normal serial port transmission. Transmission continues normally because the frame-synchronization pulse is not unexpected. There are two possible reasons why a transmit operation might *not* be in progress when the pulse occurs:
  - This FSX pulse is the first after the transmitter is enabled ($\overline{\text{XRST}}$ = 1).
  - The serial port is in the interpacket intervals. The programmed data delay for transmission (programmed with the XDATDLY bits of XCR2) may start during these interpacket intervals before the first bit of the previous word is transmitted. Thus, at maximum packet frequency, frame synchronization can still be received 0 to 2 clock cycles before the first bit of the synchronized frame.
- Case 3: Unexpected transmit frame synchronization with XFIG = 0 (frame-synchronization pulses not ignored). Unexpected frame-synchronization pulses can originate from an external source or from the internal sample rate generator.

  If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unex-

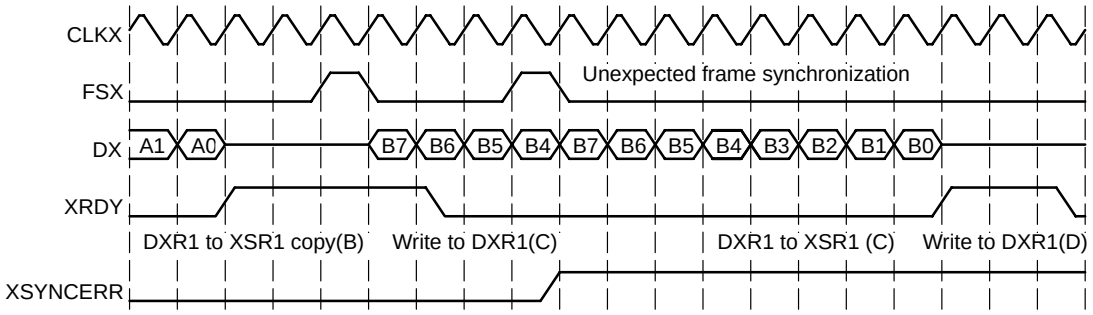
pected frame-synchronization pulse, and the transmitter sets the transmit frame-synchronization error bit (XSYNCERR) in SPCR2. XSYNCERR can be cleared only by a transmitter reset or by a write of 0 to this bit.

If you want the McBSP to notify the CPU of frame-synchronization errors, you can set a special transmit interrupt mode with the XINTM bits of SPCR2. When XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU each time that XSYNCERR is set.

12.4.5.2 Example of Unexpected Transmit Frame-Synchronization Pulse

Figure 12–31 shows an unexpected transmit frame-synchronization pulse during normal operation of the serial port with intervals between the data packets. When the unexpected frame-synchronization pulse occurs, the XSYNCERR bit is set and the transmission of data B is restarted because no new data has been passed to XSR1 yet. In addition, if XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU.

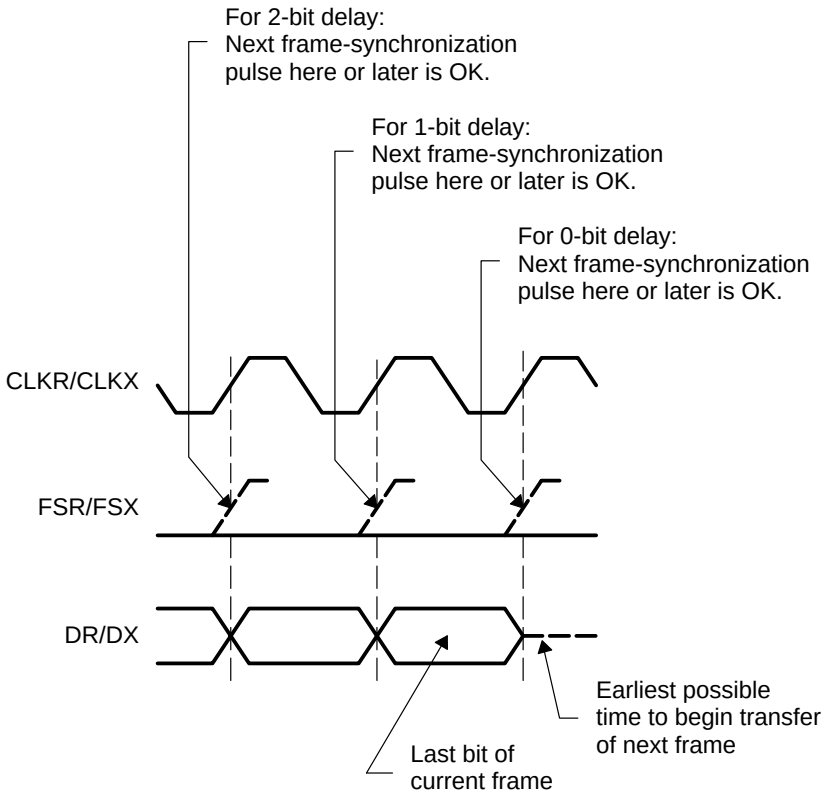
Figure 12–31. An Unexpected Frame-Synchronization Pulse During a McBSP Transmission



12.4.5.3 Preventing Unexpected Transmit Frame-Synchronization Pulses

Each frame transfer can be delayed by 0, 1, or 2 CLKX cycles, depending on the value in the XDATDLY bits of XCR2. For each possible data delay, Figure 12–32 shows when a new frame-synchronization pulse on FSX can safely occur relative to the last bit of the current frame.

Figure 12–32. Proper Positioning of Frame-Synchronization Pulses



12.5 Multichannel Selection Modes

This section discusses the multichannel selection modes for the McBSP.

12.5.1 Channels, Blocks, and Partitions

A McBSP channel is a time slot for shifting in/out the bits of one serial word. Each McBSP supports up to 128 channels for reception and 128 channels for transmission.

In the receiver and in the transmitter, the 128 available channels are divided into eight blocks that each contain 16 contiguous channels:

Block 0: Channels 0–15	Block 4: Channels 64–79
Block 1: Channels 16–31	Block 5: Channels 80–95
Block 2: Channels 32–47	Block 6: Channels 96–111
Block 3: Channels 48–63	Block 7: Channels 112–127

The blocks are assigned to partitions according to the selected partition mode. In the two-partition mode (described in section 12.5.4), you assign one even-numbered block (0, 2, 4, or 6) to partition A and one odd-numbered block (1, 3, 5, or 7) to partition B. In the 8-partition mode (described in Section 12.5.5), blocks 0 through 7 are automatically assigned to partitions, A through H, respectively.

The number of partitions for reception and the number of partitions for transmission are independent. For example, it is possible to use two receive partitions (A and B) and eight transmit partitions (A–H).

12.5.2 Multichannel Selection

When a McBSP uses a time-division multiplexed (TDM) data stream while communicating with other McBSPs or serial devices, the McBSP may need to receive and/or transmit on only a few channels. To save memory and bus bandwidth, you can use a multichannel selection mode to prevent data flow in some of the channels.

Each channel partition has a dedicated channel enable register. If the appropriate multichannel selection mode is on, each bit in the register controls whether data flow is allowed or prevented in one of the channels that is assigned to that partition.

The McBSP has one receive multichannel selection mode (described in Section 12.5.6) and three transmit multichannel selection modes (described in Section 12.5.7).

12.5.3 Configuring a Frame for Multichannel Selection

Before you enable a multichannel selection mode, ensure that you properly configure the data frame:

- ❑ Select a single-phase frame (RPHASE/XPHASE = 0). Each frame represents a TDM data stream.

- Set a frame length (in RFRLEN1/XFRLEN1) that includes the highest-numbered channel to be used. For example, if you plan to use channels 0, 15, and 39 for reception, the receive frame length must be at least 40 (RFRLEN1 = 39). If XFRLEN1 = 39 in this case, the receiver creates 40 time slots per frame but only receives data during time slots 0, 15, and 39 of each frame.

12.5.4 Using Two Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use two partitions or eight partitions (described in Section 12.5.5). If you choose the two-partition mode (RMCME = 0 for reception, XMCME = 0 for transmission), McBSP channels are activated using an alternating scheme. In response to a frame-synchronization pulse, the receiver or transmitter begins with the channels in partition A and then alternates between partitions B and A until the complete frame has been transferred. When the next frame-synchronization pulse occurs, the next frame is transferred beginning with the channels in partition A.

12.5.4.1 Assigning Blocks to Partitions A and B

For reception, any two of the eight receive-channel blocks can be assigned to receive partitions A and B, which means up to 32 receive channels can be enabled at any given time. Similarly, any two of the eight transmit-channel blocks (up 32 enabled transmit channels) can be assigned to transmit partitions A and B.

For reception:

- Assign an even-numbered channel block (0, 2, 4, or 6) to receive partition A by writing to the RPABLK bits. In the receive multichannel selection mode (described in Section 12.5.6), the channels in this partition are controlled by receive-channel enable register A (RCERA).

- Assign an odd-numbered block (1, 3, 5, or 7) to receive partition B with the RPBBLK bits. In the receive multichannel selection mode, the channels in this partition are controlled by receive-channel enable register B (RCERB).

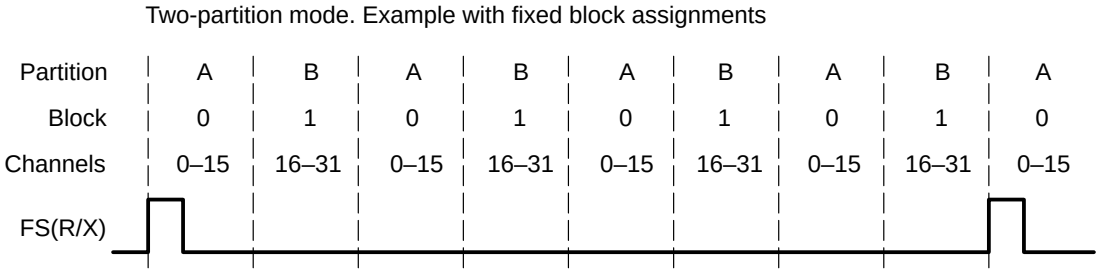
For transmission:

- Assign an even-numbered channel block (0, 2, 4, or 6) to transmit partition A by writing to the XPABLK bits. In one of the transmit-multichannel selection modes (described in Section 12.5.7), the channels in this partition are controlled by transmit-channel enable register A (XCERA).

- Assign an odd-numbered block (1, 3, 5, or 7) to transmit partition B with the XPBBLK bits. In one of the transmit-multichannel selection modes, the channels in this partition are controlled by transmit channel enable register B (XCERB).

Figure 12–33 shows an example of alternating between the channels of partition A and partition B. Channels 0–15 have been assigned to partition A, and channels 16–31 have been assigned to partition B. In response to a frame-synchronization pulse, the McBSP begins a frame transfer with partition A and then alternates between partitions B and A until the complete frame is transferred.

Figure 12–33. Alternating Between the Channels of Partition A and the Channels of Partition B



As explained in Section 12.5.4.2, you can dynamically change which blocks of channels are assigned to the partitions.

12.5.4.2 Reassigning Blocks During Reception/Transmission

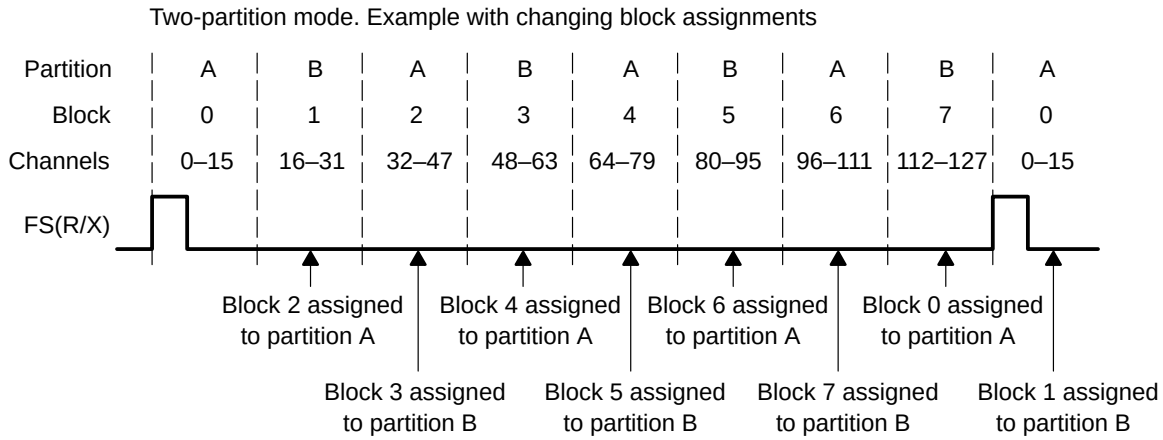
If you want to use more than 32 channels, you can change which channel blocks are assigned to partitions A and B during the course of a data transfer. However, these changes must be carefully timed. While a partition is being transferred, its associated block assignment bits cannot be modified and its associated channel enable register cannot be modified. For example, if block 3 is being transferred and block 3 is assigned to partition A, you can modify neither (R/X)PABLK to assign different channels to partition A nor (R/X)CERA to change the channel configuration for partition A. Several features of the McBSP help you time the reassignment:

- The block of channels currently involved in reception/transmission (the current block) is reflected in the RCBLK/XCBLK bits. Your program can poll these bits to determine which partition is active. When a partition is not active, it is safe to change its block assignment and channel configuration.

- At the end of every block (at the boundary of two partitions), an interrupt can be sent to the CPU. In response to the interrupt, the CPU can then check the RCBLK/XCBLK bits and update the inactive partition. See Section 12.5.8, *Using Interrupts Between Block Transfers*.

Figure 12–34 shows an example of reassigning channels throughout a data transfer. In response to a frame-synchronization pulse, the McBSP alternates between partitions A and B. Whenever partition B is active, the CPU changes the block assignment for partition A. Whenever partition A is active, the CPU changes the block assignment for partition B.

Figure 12–34. Reassigning Channel Blocks Throughout a McBSP Data Transfer



12.5.5 Using Eight Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use eight partitions or two partitions (described in Section 12.5.4). If you choose the eight-partition mode (RMCME = 1 for reception, XMCME = 1 for transmission), McBSP channels are activated in the following order: A, B, C, D, E, F, G, H. In response to a frame-synchronization pulse, the receiver or transmitter begins with the channels in partition A and then continues with the other partitions in order until the complete frame has been transferred. When the next frame-synchronization pulse occurs, the next frame is transferred, beginning with the channels in partition A.

In the eight-partition mode, the (R/X)PABLK and (R/X)PBBLK bits are ignored and the 16-channel blocks are assigned to the partitions as shown in Table 12–7 and Table 12–8. These assignments cannot be changed. The tables also show the registers used to control the channels in the partitions.

Table 12–7. Receive Channel Assignment and Control With Eight Receive Partitions

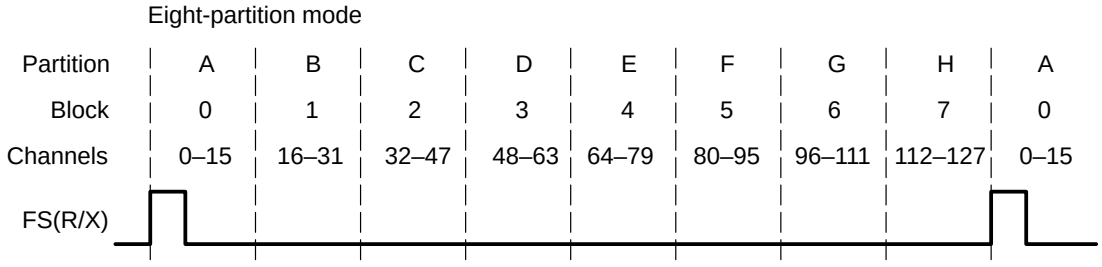
Receive Partition	Assigned Block of Receive Channels	Register Used for Channel Control
A	Block 0: Channels 0 through 15	RCERA
B	Block 1: Channels 16 through 31	RCERB
C	Block 2: Channels 32 through 47	RCERC
D	Block 3: Channels 48 through 63	RCERD
E	Block 4: Channels 64 through 79	RCERE
F	Block 5: Channels 80 through 95	RCERF
G	Block 6: Channels 96 through 111	RCERG
H	Block 7: Channels 112 through 127	RCERH

Table 12–8. Transmit Channel Assignment and Control When Eight Transmit Partitions Are Used

Transmit Partition	Assigned Block of Transmit Channels	Register Used for Channel Control
A	Block 0: Channels 0 through 15	XCERA
B	Block 1: Channels 16 through 31	XCERB
C	Block 2: Channels 32 through 47	XCERC
D	Block 3: Channels 48 through 63	XCERD
E	Block 4: Channels 64 through 79	XCERE
F	Block 5: Channels 80 through 95	XCERF
G	Block 6: Channels 96 through 111	XCERG
H	Block 7: Channels 112 through 127	XCERH

Figure 12–35 shows an example of the McBSP using the eight-partition mode. In response to a frame-synchronization pulse, the McBSP begins a frame transfer with partition A and then activates B, C, D, E, F, G, and H to complete a 128-word frame.

Figure 12–35. McBSP Data Transfer in Eight-Partition Mode



12.5.6 Receive Multichannel Selection Mode

The RMCM bit of MCR1 determines whether all channels or only selected channels are enabled for reception. When RMCM = 0, all 128 receive channels are enabled and cannot be disabled. When RMCM = 1, the receive multichannel selection mode is enabled. In this mode:

- Channels can be individually enabled or disabled. The only channels enabled are those selected in the appropriate receive channel enable registers (RCERs). The way channels are assigned to the RCERs depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit of MCR1.
- If a receive channel is disabled, any bits received in that channel are passed only as far as the receive buffer register(s) (RBR(s)). The receiver does not copy the content of the RBR(s) to the DRR(s), and as a result, does not set the receiver ready bit (RRDY). Therefore, no DMA synchronization event (REVT) is generated and, if the receiver interrupt mode depends on RRDY (RINTM = 00b), no interrupt is generated.

As an example of how the McBSP behaves in the receive multichannel selection mode, suppose you enable only channels 0, 15, and 39 and that the frame length is 40. The McBSP:

1) Accepts bits shifted in from the DR pin in channel 0
2) Ignores bits received in channels 1–14
3) Accepts bits shifted in from the DR pin in channel 15
4) Ignores bits received in channels 16–38
5) Accepts bits shifted in from the DR pin in channel 39

12.5.7 Transmit Multichannel Selection Modes

The XMCM bits of XCR2 determine whether all channels or only selected channels are enabled and unmasked for transmission. More details on enabling and masking are in Section 12.5.7.1. The McBSP has three transmit multichannel selection modes (XMCM = 01b, XMCM = 10b, and XMCM = 11b), which are described in Table 12–9.

Table 12–9. Selecting a Transmit Multichannel Selection Mode With the XMCM Bits

XMCM	Transmit Multichannel Selection Mode
00b	No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.
01b	All channels are disabled unless they are selected in the appropriate transmit channel enable registers (XCERs). If enabled, a channel in this mode is also unmasked. The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in XCERs.
10b	All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in XCERs.
11b	This mode is used for symmetric transmission and reception. All channels are disabled for transmission unless they are enabled for reception in the appropriate receive channel enable registers (RCERs). Once enabled, they are masked unless they are also selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in RCERs and XCERs.

As an example of how the McBSP behaves in a transmit-multichannel selection mode, suppose that XMCM = 01b (all channels disabled unless individually enabled) and that you have enabled only channels 0, 15, and 39. Suppose also that the frame length is 40. The McBSP:

1) Shifts data to the DX pin in channel 0
2) Places the DX pin in high impedance state in channels 1–14
3) Shifts data to the DX pin in channel 15
4) Places the DX pin in high impedance state in channels 16–38
5) Shifts data to the DX pin in channel 39

12.5.7.1 Disabling/Enabling Versus Masking/Unmasking

For transmission, a channel can be:

- ❏ Enabled and unmasked (transmission can begin and can be completed)
- ❏ Enabled but masked (transmission can begin but cannot be completed)
- ❏ Disabled (transmission cannot occur)

The following definitions explain the channel control options:

Enabled channel	A channel that can begin transmission by passing data from the data transmit register(s) (DXR(s)) to the transmit shift registers (XSR(s)).
Masked channel	A channel that cannot complete transmission. The DX pin is held in high impedance state; data cannot be shifted out on the DX pin. In systems where symmetric transmit and receive provides software benefits, this feature allows transmit channels to be disabled on a shared serial bus. A similar feature is not needed for reception because multiple receptions cannot cause serial bus contention.
Disabled channel	A channel that is not enabled. A disabled channel is also masked. Because no DXR-to-XSR copy occurs, the XRDY bit of SPCR2 is not set. Therefore, no DMA synchronization event (XEVT) is generated, and if the transmit interrupt mode depends on XRDY (XINTM = 00b in SPCR2), no interrupt is generated. The $\overline{\text{XEMPTY}}$ bit of SPCR2 is not affected.
Unmasked channel	A channel that is not masked. Data in the XSR(s) is shifted out on the DX pin.

12.5.7.2 Activity on McBSP Pins for Different Values of XMCM

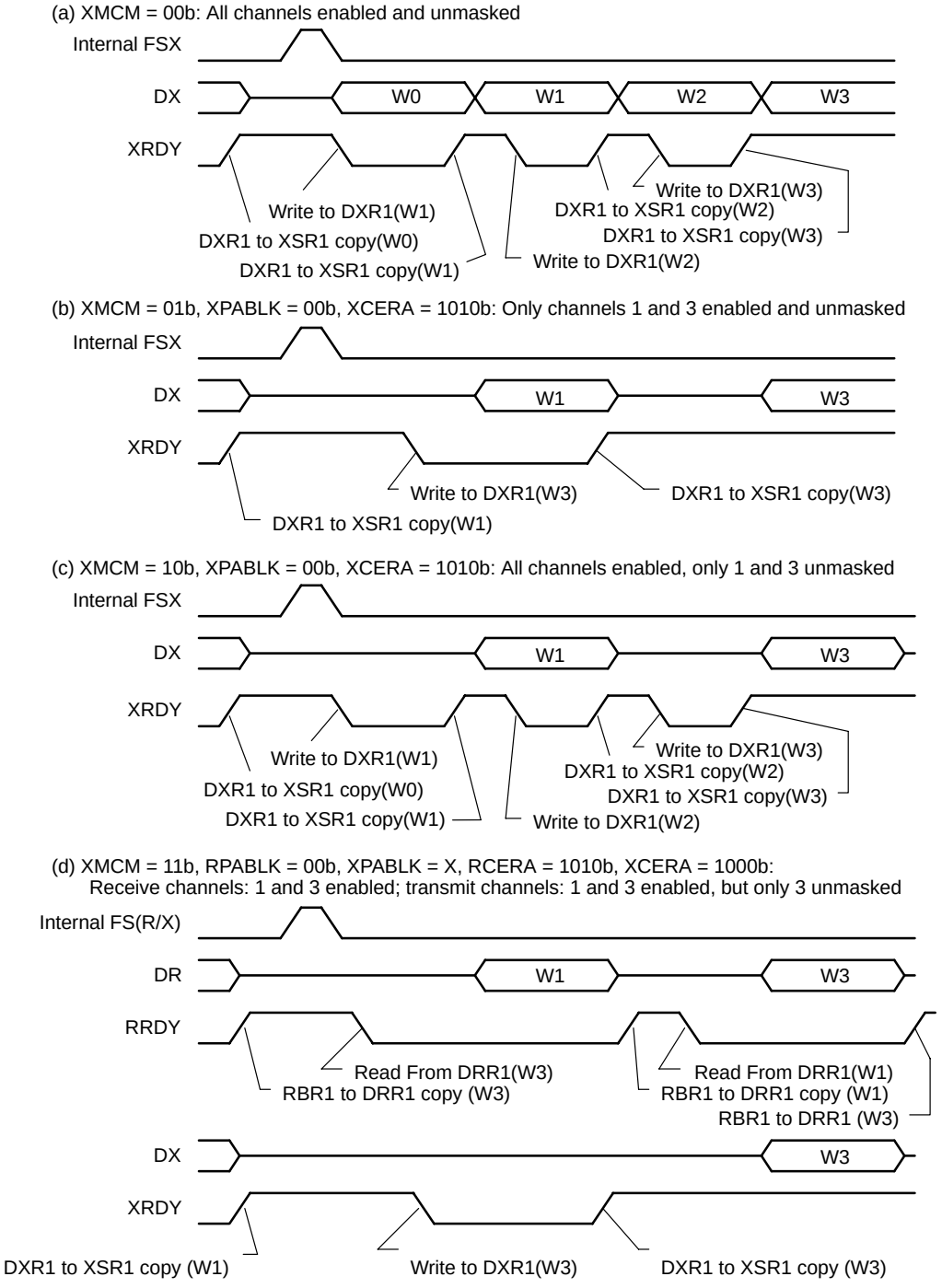
Figure 12–36 shows the activity on the McBSP pins for the various XMCM values. In all cases, the transmit frame is configured as follows:

- XPHASE = 0: Single-phase frame (required for multichannel selection modes)
- XFRLEN1 = 0000011b: 4 words per frame
- XWDLEN1 = 000b: 8 bits per word
- XMCME = 0: 2-partition mode (only partitions A and B used)

In the case where XMCM = 11b, transmission and reception are symmetric, which means the corresponding bits for the receiver (RPHASE, RFRLEN1, RWDLEN1, and RMCME) must have the same values as XPHASE, XFRLEN1, and XWDLEN1, respectively.

In the figure, the arrows showing where the various events occur are only sample indications. Wherever possible, there is a time window in which these events can occur.

Figure 12–36. Activity on McBSP Pins for the Possible Values of XMCM



12.5.8 Using Interrupts Between Block Transfers

When a multichannel selection mode is used, an interrupt request can be sent to the CPU at the end of every 16-channel block (at the boundary between partitions and at the end of the frame). In the receive multichannel selection mode, a receive interrupt (RINT) request is generated at the end of each block transfer if RINTM = 01b. In any of the transmit multichannel selection modes, a transmit interrupt (XINT) request is generated at the end of each block transfer if XINTM = 01b. When RINTM/XINTM = 01b, no interrupt is generated unless a multichannel selection mode is on.

These interrupt pulses are active high and last for two CPU clock cycles.

This type of interrupt is especially helpful if you are using the two-partition mode (described in Section 12.5.4) and you want to know when you can assign a different block of channels to partitions A or B.

12.6 SPI Operation Using the Clock Stop Mode

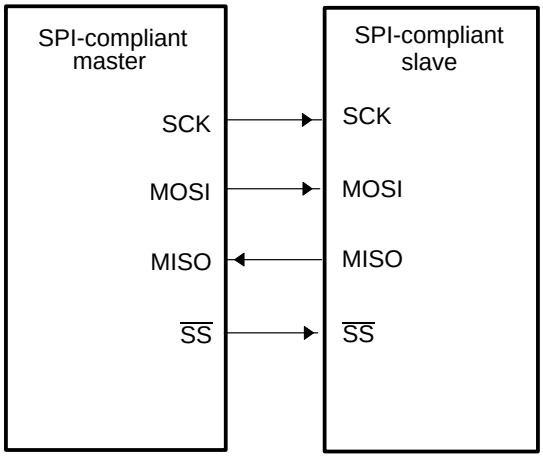
12.6.1 SPI Protocol

The SPI protocol is a master-slave configuration with one master device and one or more slave devices. The interface consists of the following four signals:

- Serial data input (also referred to as master in/slave out, or MISO)
- Serial data output (also referred to as master out/slave in, or MOSI)
- Shift-clock (SCK)
- Slave-enable signal (SS)

A typical SPI interface with a single slave device is shown in Figure 12–37.

Figure 12–37. Typical SPI Interface



The master device controls the flow of communication by providing shift-clock and slave-enable signals. The slave-enable signal is an optional active-low signal that enables the serial data input and output of the slave device (device not sending out the clock).

In the absence of a dedicated slave-enable signal, communication between the master and slave is determined by the presence or absence of an active shift-clock. When the McBSP is operating in SPI master mode and the $\overline{SS}$ signal is not used by the slave SPI port, the slave device must remain enabled at all times, and multiple slaves cannot be used.

12.6.2 Clock Stop Mode

The clock stop mode of the McBSP provides compatibility with the SPI protocol. When the McBSP is configured in clock stop mode, the transmitter and receiver are internally synchronized so that the McBSP functions as an SPI master or slave device. The transmit clock signal (CLKX) corresponds to the serial clock signal (SCK) of the SPI protocol, whereas the transmit frame-synchronization signal (FSX) is used as the slave-enable signal (SS_).

The receive clock signal (CLKR) and receive frame-synchronization signal (FSR) are not used in the clock stop mode because these signals are internally connected to their transmit counterparts, CLKX and FSX.

12.6.3 Bits Used to Enable and Configure the Clock Stop Mode

The bits required to configure the McBSP as an SPI device are introduced in Table 12–10. Table 12–11 shows how the various combinations of the CLKSTP bit and the polarity bits CLKXP and CLKRP create four possible clock-stop mode configurations. The timing diagrams in Section 12.6.4 show the effects of CLKSTP, CLKXP, and CLKRP.

Table 12–10. Bits Used to Enable and Configure the Clock Stop Mode

Bit Field	Description
CLKSTP bits of SPCR1	Use these bits to enable the clock stop mode and to select one of two timing variations. (See also Table 12–11.)
CLKXP bit of PCR	This bit determines the polarity of the CLKX signal. (See also Table 12–11.)
CLKRP bit of PCR	This bit determines the polarity of the CLKR signal. (See also Table 12–11.)
CLKXM bit of PCR	This bit determines whether CLKX is an input signal (McBSP as slave) or an output signal (McBSP as master).
XPHASE bit of XCR2	You must use a single-phase transmit frame (XPHASE = 0).
RPHASE bit of RCR2	You must use a single-phase receive frame (RPHASE = 0).
XFRLEN1 bits of XCR1	You must use a transmit frame length of 1 serial word (XFRLEN1 = 0).
RFRLEN1 bits of RCR1	You must use a receive frame length of 1 serial word (RFRLEN1 = 0).
XWDLEN1 bits of XCR1	The XWDLEN1 bits determine the transmit packet length. XWDLEN1 must be equal to RWDLEN1 because in the clock stop mode the McBSP transmit and receive circuits are synchronized to a single clock.
RWDLEN1 bits of RCR1	The RWDLEN1 bits determine the receive packet length. RWDLEN1 must be equal to XWDLEN1 because in the clock stop mode the McBSP transmit and receive circuits are synchronized to a single clock.

Table 12–11. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme

Bit Settings	Clock Scheme
CLKSTP = 00b or 01b CLKXP = 0 or 1 CLKRP = 0 or 1	Clock stop mode disabled. Clock enabled for non-SPI mode.
CLKSTP = 10b CLKXP = 0 CLKRP = 0	Low inactive state without delay: The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of CLKR.
CLKSTP = 11b CLKXP = 0 CLKRP = 1	Low inactive state with delay: The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of CLKR.
CLKSTP = 10b CLKXP = 1 CLKRP = 0	High inactive state without delay: The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of CLKR.
CLKSTP = 11b CLKXP = 1 CLKRP = 1	High inactive state with delay: The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of CLKR.

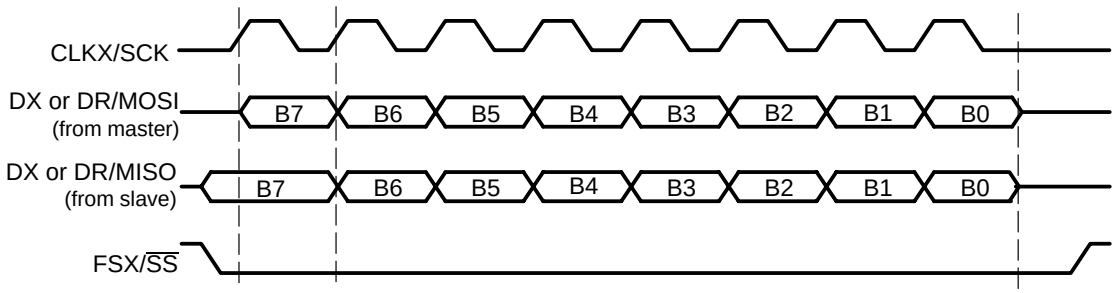
12.6.4 Clock Stop Mode Timing Diagrams

The timing diagrams for the four possible clock-stop mode configurations are shown in Figure 12–38 through Figure 12–41. Notice that the frame-synchronization signal used in clock stop mode is active throughout the entire transmission as a slave-enable signal. Although the timing diagrams show 8-bit transfers, the packet length can be set to 8-, 12-, 16-, 20-, 24-, or 32 bits per packet. The receive packet length is selected with the RWDLEN1 bits of RCR1, and the transmit packet length is selected with the XWDLEN1 bits of XCR1. For clock stop mode, the values of RWDLEN1 and XWDLEN1 must be the same because the McBSP transmit and receive circuits are synchronized to a single clock.

Note:

Even if multiple words are consecutively transferred, the CLKX signal is always stopped and the FSX signal returns to the inactive state after a packet transfer. When consecutive packet transfers are performed, this leads to a minimum idle time of two bit-periods between each packet transfer.

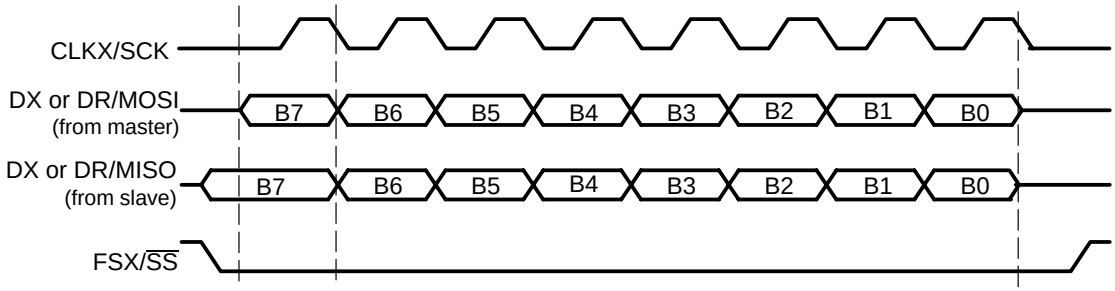
Figure 12–38. SPI Transfer With CLKSTP = 10b (No Clock Delay), CLKXP = 0, and CLKRP = 0



Notes: 1) If the McBSP is the SPI master (CLKXM = 1), MOSI = DX. If the McBSP is the SPI slave (CLKXM = 0), MOSI = DR.

2) If the McBSP is the SPI master (CLKXM = 1), MISO = DR. If the McBSP is the SPI slave (CLKXM = 0), MISO = DX.

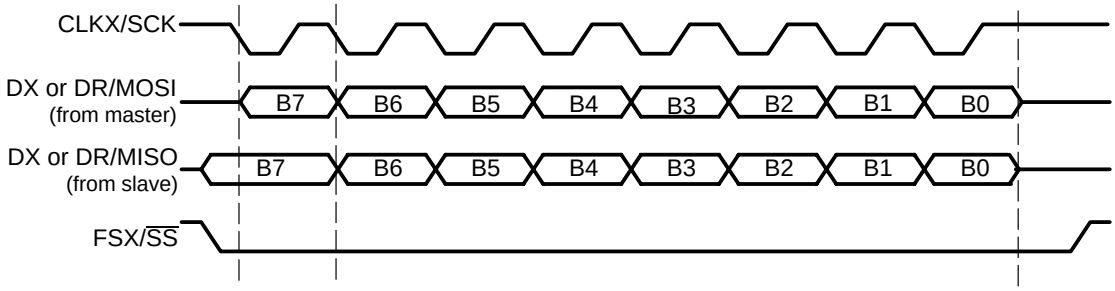
Figure 12–39. SPI Transfer With CLKSTP = 11b (Clock Delay), CLKXP = 0, CLKRP = 1



Notes: 1) If the McBSP is the SPI master (CLKXM = 1), MOSI = DX. If the McBSP is the SPI slave (CLKXM = 0), MOSI = DR.

2) If the McBSP is the SPI master (CLKXM = 1), MISO = DR. If the McBSP is the SPI slave (CLKXM = 0), MISO = DX.

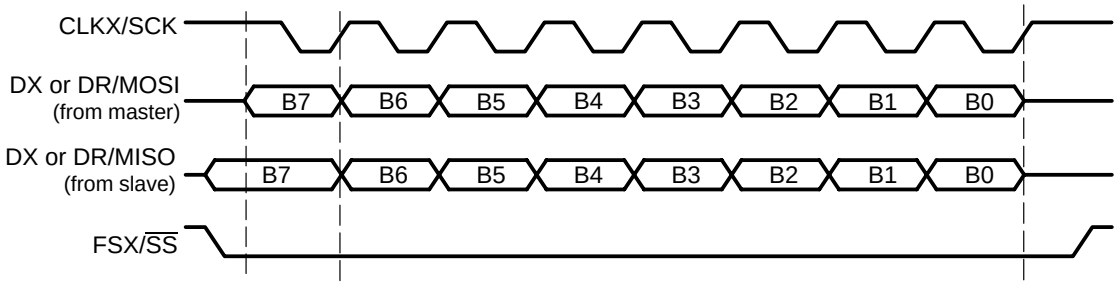
Figure 12–40. SPI Transfer With CLKSTP = 10b (No Clock Delay), CLKXP = 1, and CLKRP = 0



Notes: 1) If the McBSP is the SPI master (CLKXM = 1), MOSI = DX. If the McBSP is the SPI slave (CLKXM = 0), MOSI = DR.

2) If the McBSP is the SPI master (CLKXM = 1), MISO = DR. If the McBSP is the SPI slave (CLKXM = 0), MISO = DX.

Figure 12–41. SPI Transfer With CLKSTP = 11b (Clock Delay), CLKXP = 1, CLKRP = 1



Notes: 1) If the McBSP is the SPI master (CLKXM = 1), MOSI=DX. If the McBSP is the SPI slave (CLKXM = 0), MOSI = DR.

2) If the McBSP is the SPI master (CLKXM = 1), MISO=DR. If the McBSP is the SPI slave (CLKXM = 0), MISO = DX.

12.6.5 Procedure for Configuring a McBSP for SPI Operation

To configure the McBSP for SPI master or slave operation:

Step 1: Place the transmitter and receiver in reset.

Clear the transmitter reset bit ($\overline{\text{XRST}}$ = 0) in SPCR2 to reset the transmitter. Clear the receiver reset bit ($\overline{\text{RRST}}$ = 0) in SPCR1 to reset the receiver.

Step 2: Place the sample rate generator in reset.

Clear the sample rate generator reset bit ($\overline{\text{GRST}}$ = 0) in SPCR2 to reset the sample rate generator.

Step 3: Program registers that affect SPI operation.

Program the appropriate McBSP registers to configure the McBSP for proper operation as an SPI master or an SPI slave. For a list of important bit settings, see Section 12.6.6, *McBSP as the SPI Master* and Section 12.6.7, *McBSP as an SPI Slave.*

Step 4: Enable the sample rate generator.

To release the sample rate generator from reset, set the sample rate generator reset bit ($\overline{\text{GRST}}$ = 1) in SPCR2.

Ensure that during the write to SPCR2, you modify only $\overline{\text{GRST}}$. Otherwise, you modify the McBSP configuration you selected in the previous step.

Step 5: Enable the transmitter and receiver.

After the sample rate generator is released from reset, wait two sample rate generator clock periods for the McBSP logic to stabilize.

If the CPU services the McBSP transmit and receive buffers, then you can immediately enable the transmitter ($\overline{\text{XRST}}$ = 1 in SPCR2) and enable the receiver ($\overline{\text{RRST}}$ = 1 in SPCR1).

If the DMA controller services the McBSP transmit and receive buffers, then you must first configure the DMA controller (this includes enabling the channels that service the McBSP buffers). When the DMA controller is ready, make $\overline{\text{XRST}}$ = 1 and $\overline{\text{RRST}}$ = 1.

Ensure that you change only $\overline{\text{XRST}}$ and $\overline{\text{RRST}}$ when you write to SPCR2 and SPCR1. Otherwise, you modify the bit settings you selected earlier in this procedure.

After the transmitter and receiver are released from reset, wait two sample rate generator clock periods for the McBSP logic to stabilize.

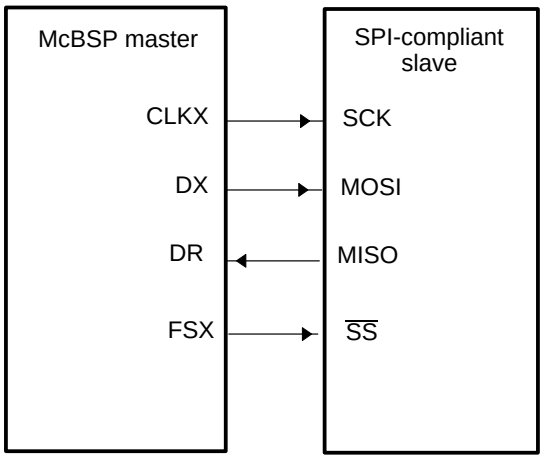
Step 6: If necessary, enable the frame-synchronization logic of the sample rate generator.

After the required data acquisition setup is done (DXR[1/2] is loaded with data), set $\overline{\text{FRST}}$ = 1 if an internally generated frame-synchronization pulse is required (that is, if the McBSP is the SPI master).

12.6.6 McBSP as the SPI Master

An SPI interface with the McBSP used as the master is shown in Figure 12–42. When the McBSP is configured as a master, the transmit output signal (DX) is used as the MOSI signal of the SPI protocol, and the receive input signal (DR) is used as the MISO signal.

Figure 12–42. SPI Interface With McBSP as Master



The register bit values required to configure the McBSP as a master are listed in Table 12–12. Following the table are more details about the configuration requirements.

Table 12–12. Bit Values Required to Configure the McBSP as an SPI Master

Required Bit Setting	Description
CLKSTP = 10b or 11b	The clock stop mode (without or with a clock delay) is selected.
CLKXP = 0 or 1	The polarity of CLKX as seen on the CLKX pin is positive (CLKXP = 0) or negative (CLKXP = 1).
CLKRP = 0 or 1	The polarity of CLKR as seen on the CLKR pin is positive (CLKRP = 0) or negative (CLKRP = 1).
CLKXM = 1	The CLKX pin is an output pin driven by the internal sample rate generator. Because CLKSTP is equal to 10b or 11b, CLKR is driven internally by CLKX.
SCLKME = 0 CLKSM = 1	The clock generated by the sample rate generator (CLKG) is derived from the CPU clock.
CLKGDV is a value from 0 to 255	CLKGDV defines the divide-down value for CLKG.
FSXM = 1	The FSX pin is an output pin driven according to the FSGM bit.
FSGM = 0	The transmitter drives a frame-synchronization pulse on the FSX pin every time data is transferred from DXR1 to XSR1.
FSXP = 1	The FSX pin is active low.
XDATDLY = 01b RDATDLY = 01b	This setting provides the correct setup time on the FSX signal.

When the McBSP functions as the SPI master, it controls the transmission of data by producing the serial clock signal. The clock signal on the CLKX pin is enabled only during packet transfers. When packets are not being transferred, the CLKX pin remains high or low depending on the polarity used.

For SPI master operation, the CLKX pin must be configured as an output. The sample rate generator is then used to derive the CLKX signal from the CPU clock. The clock stop mode internally connects the CLKX pin to the CLKR signal so that no external signal connection is required on the CLKR pin and both the transmit and receive circuits are clocked by the master clock (CLKX).

The data delay parameters of the McBSP (XDATDLY and RDATDLY) must be set to 1 for proper SPI master operation. A data delay value of 0 or 2 is undefined in the clock stop mode.

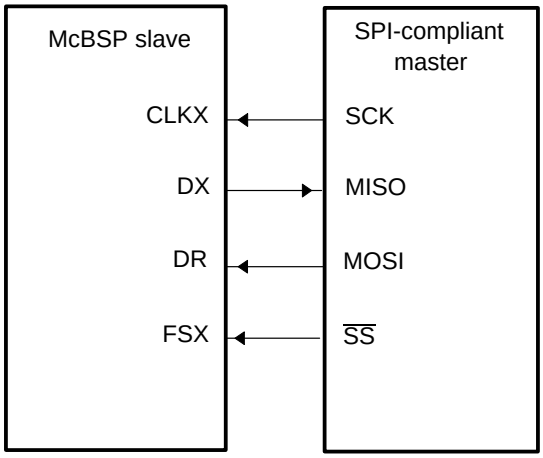
The McBSP can also provide a slave-enable signal (SS_) on the FSX pin. If a slave-enable signal is required, the FSX pin must be configured as an output and the transmitter must be configured so that a frame-synchronization pulse is generated automatically each time a packet is transmitted (FSGM = 0). The polarity of the FSX pin is programmable high or low; however, in most cases the pin must be configured active low.

When the McBSP is configured as described for SPI-master operation, the bit fields for frame-synchronization pulse width (FWID) and frame-synchronization period (FPER) are overridden, and custom frame-synchronization waveforms are not allowed. To see the resulting waveform produced on the FSX pin, see the timing diagrams in Section 12.6.4. The signal becomes active before the first bit of a packet transfer and remains active until the last bit of the packet is transferred. After the packet transfer is complete, the FSX signal returns to the inactive state.

12.6.7 McBSP as an SPI Slave

An SPI interface with the McBSP used as a slave is shown in Figure 12–43. When the McBSP is configured as a slave, DX is used as the MISO signal and DR is used as the MOSI signal.

Figure 12–43. SPI Interface With McBSP as Slave



The register bit values required to configure the McBSP as a slave are listed in Table 12–13. Following the table are more details about configuration requirements.

Table 12–13. Bit Values Required to Configure the McBSP as an SPI Slave

Required Bit Setting	Description
CLKSTP = 10b or 11b	The clock stop mode (without or with a clock delay) is selected.
CLKXP = 0 or 1	The polarity of CLKX as seen on the CLKX pin is positive (CLKXP = 0) or negative (CLKXP = 1).
CLKRP = 0 or 1	The polarity of CLKR as seen on the CLKR pin is positive (CLKRP = 0) or negative (CLKRP = 1).
CLKXM = 0	The CLKX pin is an input pin so that it can be driven by the SPI master. Because CLKSTP = 10b or 11b, CLKR is driven internally by CLKX.
SCLKME = 0 CLKSM = 1	The clock generated by the sample rate generator (CLKG) is derived from the CPU clock. (The sample rate generator is used to synchronize the McBSP logic with the externally-generated master clock.)
CLKGDV = 1	The sample rate generator divides the CPU clock by 2 before generating CLKG.
FSXM = 0	The FSX pin is an input pin so that it can be driven by the SPI master.
FSXP = 1	The FSX pin is active low.
XDATDLY = 00b RDATDLY = 00b	These bits must be zeros for SPI slave operation.

When the McBSP is used as an SPI slave, the master clock and slave-enable signals are generated externally by a master device. Accordingly, the CLKX and FSX pins must be configured as inputs. The CLKX pin is internally connected to the CLKR signal so that both the transmit and receive circuits of the McBSP are clocked by the external master clock. The FSX pin is also internally connected to the FSR signal, and no external signal connections are required on the CLKR and FSR pins.

Although the CLKX signal is generated externally by the master and is asynchronous to the McBSP, the sample rate generator of the McBSP must be enabled for proper SPI slave operation. The sample rate generator must be programmed to its maximum rate of one-half the CPU clock rate. The internal sample rate clock is then used to synchronize the McBSP logic to the external master clock and slave-enable signals.

The McBSP requires an active edge of the slave-enable signal on the FSX input for each transfer. This means that the master device must assert the slave-enable signal at the beginning of each transfer and deassert the signal after the completion of each packet transfer; the slave-enable signal cannot remain active between transfers.

The data delay parameters of the McBSP must be set to 0 for proper SPI slave operation. A value of 1 or 2 is undefined in the clock stop mode.

12.7 Receiver Configuration

To configure the McBSP receiver, perform the following procedure:

1) Place the McBSP/receiver in reset (see Section 12.7.2).

2) Program the McBSP registers for the desired receiver operation (see Section 12.7.1).

3) Take the receiver out of reset (see Section 12.7.2).

12.7.1 Programming the McBSP Registers for the Desired Receiver Operation

The following are important tasks to be performed when you are configuring the McBSP receiver. Each task corresponds to one or more McBSP register bit fields.

It may be helpful to first photocopy the McBSP Register Worksheet (see Section 12.14) and to fill in the photocopy of the worksheet as you read the tasks.

- ❑ Global behavior:
  - ■ Set the receiver pins to operate as McBSP pins.
  - ■ Enable/disable the digital loopback mode.
  - ■ Enable/disable the clock stop mode.
  - ■ Enable/disable the receive multichannel selection mode.
- ❑ Data behavior:
  - ■ Choose 1 or 2 phases for the receive frame.
  - ■ Set the receive word length(s).
  - ■ Set the receive frame length.
  - ■ Enable/disable the receive frame-synchronization ignore function.
  - ■ Set the receive companding mode.
  - ■ Set the receive data delay.
  - ■ Set the receive sign-extension and justification mode.
  - ■ Set the receive interrupt mode.
- ❑ Frame-synchronization behavior:
  - ■ Set the receive frame-synchronization mode.
  - ■ Set the receive frame-synchronization polarity.
  - ■ Set the sample rate generator (SRG) frame-synchronization period and pulse width.

- Clock behavior:
  - Set the receive clock mode.
  - Set the receive clock polarity.
  - Set the SRG clock divide-down value.
  - Set the SRG clock synchronization mode.
  - Set the SRG clock mode (choose an input clock).
  - Set the SRG input clock polarity.

12.7.2 Resetting and Enabling the Receiver

The first step of the receiver configuration procedure is to reset the receiver, and the last step is to enable the receiver (to take it out of reset). Table 12–14 describes the bits used for both of these steps.

Table 12–14. Register Bits Used to Reset or Enable the McBSP Receiver

Register	Bit	Name	Function		Type	Reset Value
SPCR1	0	$\overline{\text{RRST}}$	Receiver reset		R/W	0
			$\overline{\text{RRST}}$ = 0	The serial port receiver is disabled and in the reset state.		
			$\overline{\text{RRST}}$ = 1	The serial port receiver is enabled.		
SPCR2	6	$\overline{\text{GRST}}$	Sample rate generator reset		R/W	0
			$\overline{\text{GRST}}$ = 0	Sample rate generator is reset. If $\overline{\text{GRST}}$ = 0 because of an OMAP730 reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven low (inactive). If $\overline{\text{GRST}}$ = 0 because of program code, CLKG and FSG are both driven low (inactive).		
			$\overline{\text{GRST}}$ = 1	Sample rate generator is enabled. CLKG is driven according to the configuration programmed in the sample rate generator registers (SRGR[1,2]). If $\overline{\text{FRST}}$ = 1, the generator also generates the frame-synchronization signal FSG as programmed in the sample rate generator registers.		
SPCR2	7	$\overline{\text{FRST}}$	Frame-synchronization logic reset		R/W	0
			$\overline{\text{FRST}}$ = 0	Frame-synchronization logic is reset. The sample rate generator does not generate frame-synchronization signal FSG, even if $\overline{\text{GRST}}$ = 1.		
			$\overline{\text{FRST}}$ = 1	If $\overline{\text{GRST}}$ = 1, frame-synchronization signal FSG is generated after (FPER + 1) number of CLKG clock cycles. All frame counters are loaded with their programmed values.		

12.7.2.1 Reset Considerations

The serial port can be reset in the following two ways:

- An OMAP730 reset ($\overline{\text{RESET}}$ signal driven low) places the receiver, transmitter, and sample rate generator in reset. When the device reset is removed ($\overline{\text{RESET}}$ signal released), $\overline{\text{GRST}}$ = $\overline{\text{FRST}}$ = $\overline{\text{RRST}}$ = $\overline{\text{XRST}}$ = 0 keep the entire serial port in reset state.
- The serial port transmitter and receiver can be reset directly using the $\overline{\text{RRST}}$ and $\overline{\text{XRST}}$ bits in the serial port control registers. The sample rate generator can be reset directly using the $\overline{\text{GRST}}$ bit in SPCR2.

Table 12–15 shows the state of McBSP pins when the serial port is reset due to an OMAP730 reset and a direct receiver/transmitter reset.

Table 12–15. Reset State of Each McBSP Pin

Pin	Possible State(s)	State Forced By OMAP730 Reset	State Forced By Receiver/Transmitter Reset
			Receiver reset ($\overline{\text{RRST}}$ = 0 and $\overline{\text{GRST}}$ = 1)
DR	I	Input	Input
CLKRX	I/O/Z	Input	Known state if input; CLKRX running if output
FSRX	I/O/Z	Input	Known state if input; FSRX inactive state if output
CLKS	I/O/Z	Input	Input
			Transmitter reset ($\overline{\text{XRST}}$ = 0 and $\overline{\text{GRST}}$ = 1)
DX	O/Z	High/Low impedance	Low impedance after transmit bit clock provided

Note: In Possible State(s) column, I = input, O = output, Z = high impedance

For more details about McBSP reset conditions and effects, see Section 12.10.3, *Resetting and Initializing a McBSP*.

12.7.3 Set the Receiver Pins to Operate as McBSP Pins

The RIOEN bit, described in Table 12–16, determines whether the receiver pins are McBSP pins or general-purpose I/O pins.

Table 12–16. Register Bit Used to Set Receiver Pins to Operate as McBSP Pins

Register	Bit	Name	Function		Type	Reset Value
PCR	12	RIOEN	Receive I/O enable		R/W	0
			This bit is applicable only when the receiver is in the reset state ($\overline{RRST}$ = 0 in SPCR1).			
			RIOEN = 0	The DR, FSRX, CLKRX, and CLKS pins are configured as serial port pins and do not function as general-purpose I/O pins.		
			RIOEN = 1	The DR pin is a general-purpose input pin. The FSRX and CLKRX pins are general-purpose I/O pins. These serial port pins do not perform serial port operation. The CLKS pin is a general-purpose input pin if RIOEN = XIOEN = 1 and $\overline{RRST}$ = $\overline{XRST}$ = 0. For more information on using these pins as general-purpose I/O pins, see Section 12.9.		

12.7.4 Enable/Disable the Digital Loopback Mode

The DLB bit determines whether the digital loopback mode is on. DLB is described in Table 12–17.

Table 12–17. Register Bit Used to Enable/Disable the Digital Loopback Mode

Register	Bit	Name	Function		Type	Reset Value
SPCR1	15	DLB	Digital loopback mode		R/W	0
			DLB = 0	Digital loopback mode is disabled.		
			DLB = 1	Digital loopback mode is enabled.		

12.7.4.1 Digital Loopback Mode

In the digital loopback mode, the receive signals are connected internally through multiplexers to the corresponding transmit signals, as shown in Table 12–18. This mode allows testing of serial port code with a single DSP device; the McBSP receives the data it transmits.

Table 12–18. Receive Signals Connected to Transmit Signals in Digital Loopback Mode

This Receive Signal	Is Fed Internally by This Transmit Signal
DR (receive data)	DX (transmit data)
FSR (receive frame synchronization)	FSX (transmit frame synchronization)
CLKR (receive clock)	CLKX (transmit clock)

12.7.5 Enable/Disable the Clock Stop Mode

The CLKSTP bits determine whether the clock stop mode is on. CLKSTP is described in Table 12–19.

Table 12–19. Register Bits Used to Enable/Disable the Clock Stop Mode

Register	Bit	Name	Function		Type	Reset Value
SPCR1	12–11	CLKSTP	Clock stop mode		R/W	00
			CLKSTP = 0Xb	Clock stop mode disabled; normal clocking for non-SPI mode		
			CLKSTP = 10b	Clock stop mode enabled, without clock delay		
			CLKSTP = 11b	Clock stop mode enabled, with clock delay		

12.7.5.1 Clock Stop Mode

The clock stop mode supports the SPI master-slave protocol. If you do not plan to use the SPI protocol, you can clear CLKSTP to disable the clock stop mode.

In the clock stop mode, the clock stops at the end of each data transfer. At the beginning of each data transfer, the clock starts immediately (CLKSTP = 10b) or after a half-cycle delay (CLKSTP = 11b). The CLKXP bit determines whether the starting edge of the clock on the CLKX pin is rising or falling. The CLKRP bit determines whether receive data is sampled on the rising or falling edge of the clock shown on the CLKR pin.

Table 12–20 summarizes the effect of CLKSTP, CLKXP, and CLKRP on serial port operation. In the clock stop mode, the receive clock is tied internally to the transmit clock, and the receive frame-synchronization signal is tied internally to the transmit frame-synchronization signal.

Table 12–20. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme

Bit Settings	Clock Scheme
CLKSTP = 00b or 01b CLKXP = 0 or 1 CLKRP = 0 or 1	Clock stop mode disabled. Clock enabled for non-SPI mode.
CLKSTP = 10b CLKXP = 0 CLKRP = 0	Low inactive state without delay: The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of CLKR.
CLKSTP = 11b CLKXP = 0 CLKRP = 1	Low inactive state with delay: The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of CLKR.
CLKSTP = 10b CLKXP = 1 CLKRP = 0	High inactive state without delay: The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of CLKR.
CLKSTP = 11b CLKXP = 1 CLKRP = 1	High inactive state with delay: The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of CLKR.

12.7.6 Enable/Disable the Receive Multichannel Selection Mode

The RMCM bit determines whether the receive multichannel selection mode is on. RMCM is described in Table 12–21.

Table 12–21. Register Bit Used to Enable/Disable the Receive Multichannel Selection Mode

Register	Bit	Name	Function		Type	Reset Value
MCR1	0	RMCM	Receive multichannel selection mode		R/W	0
			RMCM = 0	The mode is disabled. All 128 channels are enabled.		
			RMCM = 1	The mode is enabled. Channels can be individually enabled or disabled. The only channels enabled are those selected in the appropriate receive channel enable registers (RCERs). The way channels are assigned to the RCERs depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit.		

For more details, see Section 12.5.6, *Receive Multichannel Selection Mode*.

12.7.7 Choose One or Two Phases for the Receive Frame

The RPHASE bit (see Table 12–22) determines whether the receive data frame has one or two phases.

Table 12–22. Register Bit Used to Choose One or Two Phases for the Receive Frame

Register	Bit	Name	Function		Type	Reset Value
RCR2	15	RPHASE	Receive phase number		R/W	0
			Specifies whether the receive frame has 1 or 2 phases			
			RPHASE = 0	Single-phase frame		
			RPHASE = 1	Dual-phase frame		

12.7.8 Set the Receive Word Length(s)

The RWDLEN1 and RWDLEN2 bit fields (see Table 12–23) determine how many bits are in each serial word in phase 1 and in phase 2, respectively, of the receive data frame.

Table 12–23. Register Bits Used to Set the Receive Word Length(s)

Register	Bit	Name	Function	Type	Reset Value
RCR1	7:5	RWDLEN1	Receive word length 1	R/W	000
			Specifies the length of every serial word in phase 1 of the receive frame.		
			RWDLEN1 = 000 8 bits		
			RWDLEN1 = 001 12 bits		
			RWDLEN1 = 010 16 bits		
			RWDLEN1 = 011 20 bits		
			RWDLEN1 = 100 24 bits		
			RWDLEN1 = 101 32 bits		
			RWDLEN1 = 11X Reserved		
RCR2	7:5	RWDLEN2	Receive word length 2	R/W	000
			If a dual-phase frame is selected, RWDLEN2 specifies the length of every serial word in phase 2 of the frame.		
			RWDLEN2 = 000 8 bits		
			RWDLEN2 = 001 12 bits		
			RWDLEN2 = 010 16 bits		
			RWDLEN2 = 011 20 bits		
			RWDLEN2 = 100 24 bits		
			RWDLEN2 = 101 32 bits		
			RWDLEN2 = 11X Reserved		

12.7.8.1 Word Length Bits

Each frame can have one or two phases, depending on the value that you load into the RPHASE bit. If a single-phase frame is selected, RWDLEN1 selects the length for every serial word received in the frame. If a dual-phase frame is selected, RWDLEN1 determines the length of the serial words in phase 1 of the frame, and RWDLEN2 determines the word length in phase 2 of the frame.

12.7.9 Set the Receive Frame Length

The RFRLEN1 and RFRLEN2 bit fields (see Table 12–24) determine how many serial words are in phase 1 and in phase 2, respectively, of the receive data frame.

Table 12–24. Register Bits Used to Set the Receive Frame Length

Register	Bit	Name	Function	Type	Reset Value
RCR1	14:8	RFRLEN1	Receive frame length 1	R/W	000 0000
			(RFRLEN1 + 1) is the number of serial words in phase 1 of the receive frame.		
			RFRLEN1 = 000 0000 1 word in phase 1		
			RFRLEN1 = 000 0001 2 words in phase 1		
			| |		
			| |		
			RFRLEN1 = 111 1111 128 words in phase 1		
RCR2	14:8	RFRLEN2	Receive frame length 2	R/W	000 0000
			If a dual-phase frame is selected, (RFRLEN2 + 1) is the number of serial words in phase 2 of the receive frame.		
			RFRLEN2 = 000 0000 1 word in phase 2		
			RFRLEN2 = 000 0001 2 words in phase 2		
			| |		
			| |		
			RFRLEN2 = 111 1111 128 words in phase 2		

12.7.9.1 Selected Frame Length

The receive frame length is the number of serial words in the receive frame. Each frame can have one or two phases, depending on the value that you load into the RPHASE bit.

If a single-phase frame is selected (RPHASE = 0), the frame length is equal to the length of phase 1. If a dual-phase frame is selected (RPHASE = 1), the frame length is the length of phase 1 plus the length of phase 2.

The 7-bit RFRLEN fields allow up to 128 words per phase. See Table 12–25 for a summary of how to calculate the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization pulse.

Program the RFRLEN fields with [*w minus 1*], where *w* represents the number of words per phase. For the example, if you want a phase length of 128 words in phase 1, load 127 into RFRLEN1.

Table 12–25. How to Calculate the Length of the Receive Frame

RPHASE	RFRLEN1	RFRLEN2	Frame Length
0	0 ≤ RFRLEN1 ≤ 127	Don't care	(RFRLEN1 + 1) words
1	0 ≤ RFRLEN1 ≤ 127	0 ≤ RFRLEN2 ≤ 127	(RFRLEN1 + 1) + (RFRLEN2 + 1) words

12.7.10 Enable/Disable the Receive Frame-Synchronization Ignore Function

The RFIG bit (see Table 12–26) controls the receive frame-synchronization ignore function.

Table 12–26. Register Bit Used to Enable/Disable the Receive Frame-Synchronization Ignore Function

Register	Bit	Name	Function		Type	Reset Value
RCR2	2	RFIG	Receive frame-synchronization ignore		R/W	0
			RFIG = 0	An unexpected receive frame-synchronization pulse causes the McBSP to restart the frame transfer.		
			RFIG = 1	The McBSP ignores unexpected receive frame-synchronization pulses.		

12.7.10.1 Unexpected Frame-Synchronization Pulses and the Frame-Synchronization Ignore Function

If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-synchronization pulse.

When RFIG = 1, reception continues, ignoring the unexpected frame-synchronization pulses.

When RFIG = 0, an unexpected FSR pulse causes the McBSP to discard the contents of RSR[1,2] in favor of the new incoming data. Therefore, if RFIG = 0 and an unexpected frame-synchronization pulse occurs, the serial port:

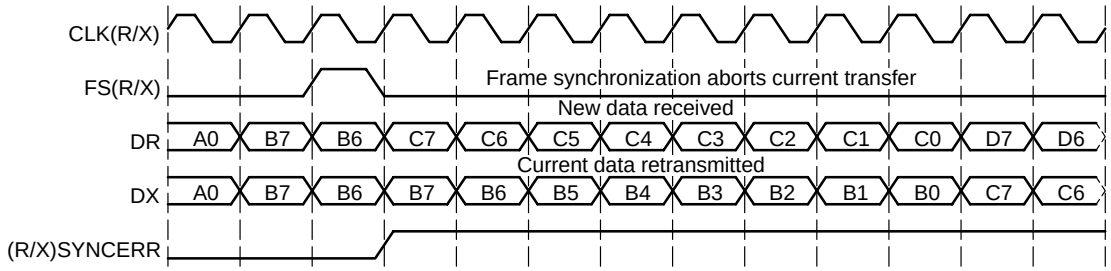
1) Aborts the current data transfer
2) Sets RSYNCERR in SPCR1 to 1
3) Begins the transfer of a new data word

For more details about the frame-synchronization error condition, see Section 12.4.2, *Unexpected Receive Frame-Synchronization Pulse*.

12.7.10.2 Examples of Effects of RFIG

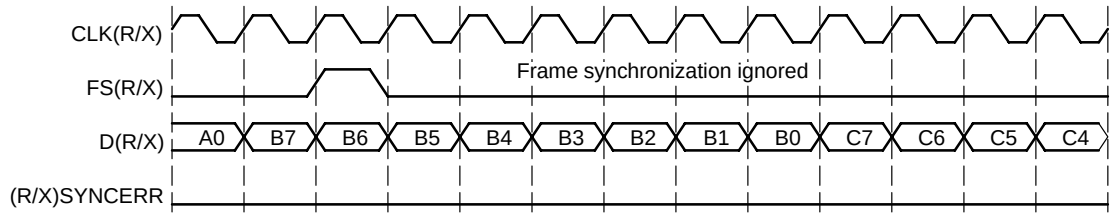
Figure 12–44 shows an example in which word B is interrupted by an unexpected frame-synchronization pulse when (R/X)FIG = 0. In the case of reception, the reception of B is aborted (B is lost), and a new data word (C in this example) is received after the appropriate data delay. This condition is a receive synchronization error, which sets the RSYNCERR bit.

Figure 12–44. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 0



In contrast to Figure 12–44, Figure 12–45 shows McBSP operation when unexpected frame-synchronization signals are ignored (when (R/X)FIG = 1). Here, the transfer of word B is not affected by an unexpected pulse.

Figure 12–45. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 1



12.7.11 Set the Receive Companding Mode

The RCOMPAND bits (see Table 12–27) determine whether companding or another data transfer option is chosen for McBSP reception.

Table 12–27. Register Bits Used to Set the Receive Companding Mode

Register	Bit	Name	Function	Type	Reset Value
RCR2	4:3	RCOMPAND	Receive companding mode	R/W	00
			Modes other than 00b are enabled only when the appropriate RWDLEN is 000b, indicating 8-bit data.		
			RCOMPAND = 00 No companding, any size data, MSB received first		
			RCOMPAND = 01 No companding, 8-bit data, LSB received first (for details, see Section 12.7.11.4, *Option to Receive LSB First*)		
			RCOMPAND = 10 µ-law companding, 8-bit data, MSB received first		
			RCOMPAND = 11 A-law companding, 8-bit data, MSB received first		

12.7.11.1 Companding

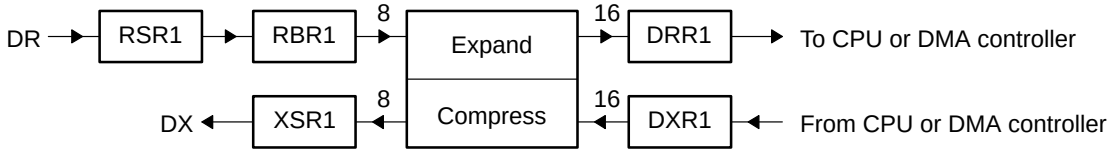
Companding (COMpressing and exPANDing) hardware allows compression and expansion of data in either μ-law or A-law format. The companding standard employed in the United States and Japan is μ-law. The European companding standard is referred to as A-law. The specifications for μ-law and A-law log PCM are part of the CCITT G.711 recommendation.

A-law and μ-law allow 13 bits and 14 bits of dynamic range, respectively. Any values outside this range are set to the most positive or most negative value. Thus, for companding to work best, the data transferred to and from the McBSP via the CPU or DMA controller must be at least 16 bits wide.

The μ-law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word length bits (RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

Figure 12–46 illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or μ-law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to twos-complement format.

Figure 12–46. Companding Processes for Reception and for Transmission



12.7.11.2 Format of Expanded Data

For reception, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1. The RJUST bit of SPCR1 is ignored when companding is used.

12.7.11.3 Companding Internal Data

If the McBSP is otherwise unused (the serial port transmit and receive sections are reset), the companding hardware can compand internal data. See Section 12.2.2.2, *Capability to Compand Internal Data*.

12.7.11.4 Option to Receive LSB First

Usually, the McBSP transmits or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols (that do not use companded data) require the least significant bit (LSB) to be transferred first. If you set RCOMPAND = 01b in RCR2, the bit ordering of 8-bit words is reversed during reception. Similar to companding, this feature is enabled only if the appropriate

word length bits are set to 0, indicating that 8-bit words are to be transferred serially. If either phase of the frame does not have an 8-bit word length, the McBSP assumes the word length is 8 bits and LSB-first ordering is done.

12.7.12 Set the Receive Data Delay

The RDATDLY bits (see Table 12–28) determine the length of the data delay for the receive frame.

Table 12–28. Register Bits Used to Set the Receive Data Delay

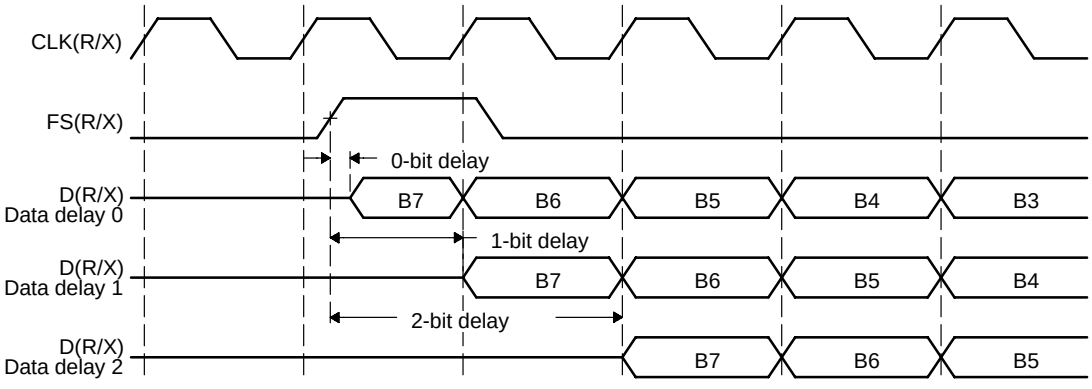
Register	Bit	Name	Function	Type	Reset Value
RCR2	1:0	RDATDLY	Receive data delay	R/W	00
			RDATDLY = 00 0-bit data delay		
			RDATDLY = 01 1-bit data delay		
			RDATDLY = 10 2-bit data delay		
			RDATDLY = 11 Reserved		

12.7.12.1 Data Delay

The start of a frame is defined by the first clock cycle in which frame-synchronization is found to be active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if required. This delay is called data delay.

RDATDLY specifies the data delay for reception. The range of programmable data delay is 0 to 2 bit-clocks (RDATDLY = 00b–10b), as described in Table 12–28 and shown in Figure 12–47. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on. Typically, a 1-bit delay is selected, because data often follows a 1-cycle active frame-synchronization pulse.

Figure 12–47. Range of Programmable Data Delay



12.7.12.2 0-Bit Data Delay

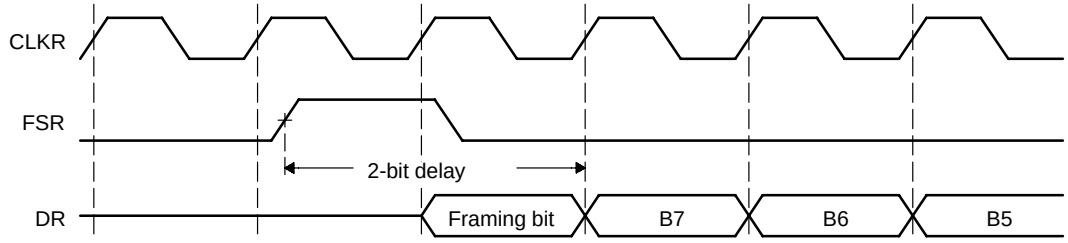
Usually, a frame-synchronization pulse is detected or sampled with respect to an edge of internal serial clock CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data can be received or transmitted. However, in the case of 0-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle.

For reception, this problem is solved because receive data is sampled on the first falling edge of CLKR where an active-high internal FSR is detected. However, data transmission must begin on the rising edge of the internal CLKX clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in XSR1, and thus on DX. The transmitter then asynchronously detects the frame-synchronization signal (FSX) going active high and immediately starts driving the first bit to be transmitted on the DX pin.

12.7.12.3 2-Bit Data Delay

A data delay of two bit-periods allows the serial port to interface to different types of T1 framing devices where the data stream is preceded by a framing bit. During reception of such a stream with data delay of 2 bits (framing bit appears after a 1-bit delay and data appears after a 2-bit delay), the serial port essentially discards the framing bit from the data stream, as shown in Figure 12–48. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on.

Figure 12–48. 2-Bit Data Delay Used to Skip a Framing Bit



12.7.13 Set the Receive Sign-Extension and Justification Mode

The RJUST bits (see Table 12–29) determine whether data received by the McBSP is sign-extended and how it is justified.

Table 12–29. Register Bits Used to Set the Receive Sign-Extension and Justification Mode

Register	Bit	Name	Function		Type	Reset Value
SPCR1	14:13	RJUST	Receive sign-extension and justification mode		R/W	00
			RJUST = 00	Right-justify data and zero fill MSBs in DRR[1,2]		
			RJUST = 01	Right-justify data and sign extend it into the MSBs in DRR[1,2]		
			RJUST = 10	Left-justify data and zero fill LSBs in DRR[1,2]		
			RJUST = 11	Reserved		

12.7.13.1 Sign-Extension and the Justification

RJUST in SPCR1 selects whether data in RBR[1,2] is right- or left-justified (with respect to the MSB) in DRR[1,2] and whether unused bits in DRR[1,2] are filled with zeros or with sign bits.

Table 12–30 and Table 12–31 show the effects of various RJUST values. The first table shows the effect on an example 12-bit receive-data value 0xABC. The second table shows the effect on an example 20-bit receive-data value 0xABCDE.

Table 12–30. Example: Use of RJUST Field With 12-Bit Data Value 0xABC

RJUST	Justification	Extension	Value in DRR2	Value in DRR1
00b	Right	Zero fill MSBs	0x0000	0x0ABC
01b	Right	Sign extend data into MSBs	0xFFFF	0xFABC
10b	Left	Zero fill LSBs	0x0000	0xABC0
11b	Reserved	Reserved	Reserved	Reserved

Table 12–31. Example: Use of RJUST Field With 20-Bit Data Value 0xABCDE

RJUST	Justification	Extension	Value in DRR2	Value in DRR1
00b	Right	Zero fill MSBs	0x000A	0xBCDE
01b	Right	Sign extend data into MSBs	0xFFFA	0xBCDE
10b	Left	Zero fill LSBs	0xABCD	0xE000
11b	Reserved	Reserved	Reserved	Reserved

12.7.14 Set the Receive Interrupt Mode

The RINTM bits (see Table 12–32) determine which event generates a receive interrupt request to the CPU.

The receive interrupt (RINT) informs the CPU of changes to the serial port status. Four options exist for configuring this interrupt. The options are set by the receive interrupt mode bits, RINTM, in SPCR1.

Table 12–32. Register Bits Used to Set the Receive Interrupt Mode

Register	Bit	Name	Function		Type	Reset Value
SPCR1	5:4	RINTM	Receive interrupt mode		R/W	00
			RINTM = 00	RINT generated when RRDY changes from 0 to 1. Interrupt on every serial word by tracking the RRDY bit in SPCR1. Regardless of the value of RINTM, RRDY can be read to detect the RRDY = 1 condition.		
			RINTM = 01	RINT generated by an end-of-block or end-of-frame condition in the receive multichannel selection mode. In the multichannel selection mode, interrupt after every 16-channel block boundary has been crossed within a frame and at the end of the frame. For details, see Section 12.5.8, *Using Interrupts Between Block Transfers*. In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated.		
			RINTM = 10	RINT generated by a new receive frame-synchronization pulse. Interrupt on detection of receive frame-synchronization pulses. This generates an interrupt even when the receiver is in its reset state. This is done by synchronizing the incoming frame-synchronization pulse to the CPU clock and sending it to the CPU via RINT.		
			RINTM = 11	RINT generated when RSYNCERR is set. Interrupt on frame-synchronization error. Regardless of the value of RINTM, RSYNCERR can be read to detect this condition. For information on using RSYNCERR, see Section 12.4.2, *Unexpected Receive Frame-Synchronization Pulse*.		

12.7.15 Set the Receive Frame-Synchronization Mode

The bits described in Table 12–33 determine the source for receive frame synchronization and the function of the FSR pin.

Table 12–33. Register Bits Used to Set the Receive Frame Synchronization Mode

Register	Bit	Name	Function		Type	Reset Value
PCR	10	FSRM	Receive frame-synchronization mode		R/W	0
			FSRM = 0	Receive frame synchronization is supplied by an external source via the FSR pin.		
			FSRM = 1	Receive frame synchronization is supplied by the sample rate generator. FSR is an output pin reflecting internal FSR, except when GSYNC = 1 in SRGR2.		
SRGR2	15	GSYNC	Sample rate generator clock synchronization mode		R/W	0
			If the sample rate generator creates a frame-synchronization signal (FSG) that is derived from an external input clock, the GSYNC bit determines whether FSG is kept synchronized with pulses on the FSR pin.			
			GSYNC = 0	No clock synchronization is used: CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles.		
			GSYNC = 1	Clock synchronization is used. When a pulse is detected on the FSR pin: ❑ CLKG is adjusted as necessary so that it is synchronized with the input clock on the CLKS, CLKR, or CLKX pin. ❑ FSG pulses FSG only pulses in response to a pulse on the FSR pin. The frame-synchronization period defined in FPER is ignored.		
			For more details, see Section 12.3.3, *Synchronizing Sample Rate Generator Outputs to an External Clock*.			
SPCR1	15	DLB	Digital loopback mode		R/W	0
			DLB = 0	Digital loopback mode is disabled.		
			DLB = 1	Digital loopback mode is enabled. The receive signals, including the receive frame-synchronization signal, are connected internally through multiplexers to the corresponding transmit signals.		

Table 12–33. Register Bits Used to Set the Receive Frame Synchronization Mode (Continued)

Register	Bit	Name	Function		Type	Reset Value
SPCR1	12:11	CLKSTP	Clock stop mode		R/W	00
			CLKSTP = 0Xb	Clock stop mode disabled; usual clocking for non-SPI mode.		
			CLKSTP = 10b	Clock stop mode enabled without clock delay. The internal receive clock signal (CLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.		
			CLKSTP = 11b	Clock stop mode enabled with clock delay. The internal receive clock signal (CLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.		

12.7.15.1 Receive Frame-Synchronization Modes

Table 12–34 shows how you can select various sources to provide the receive frame-synchronization signal and the effect on the FSR pin. The polarity of the signal on the FSR pin is determined by the FSRP bit.

In digital loopback mode (DLB = 1), the transmit frame-synchronization signal is used as the receive frame-synchronization signal.

Also in the clock stop mode, the internal receive clock signal (CLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.

Table 12–34. Select Sources to Provide the Receive Frame-Synchronization Signal and the Effect on the FSR Pin

DLB	FSRM	GSYNC	Source of Receive Frame Synchronization	FSR Pin Status
0	0	0 or 1	An external frame-synchronization signal enters the McBSP through the FSR pin. The signal is then inverted as determined by FSRP before being used as internal FSR.	Input
0	1	0	Internal FSR is driven by the sample rate generator frame-synchronization signal (FSG).	Output. FSG is inverted as determined by FSRP before being driven out on the FSR pin.
0	1	1	Internal FSR is driven by the sample rate generator frame-synchronization signal (FSG).	Input. The external frame-synchronization input on the FSR pin is used to synchronize CLKG and generate FSG pulses.
1	0	0	Internal FSX drives internal FSR.	High impedance
1	0 or 1	1	Internal FSX drives internal FSR.	Input. If the sample rate generator is running, external FSR is used to synchronize CLKG and generate FSG pulses.
1	1	0	Internal FSX drives internal FSR.	Output. Receive (same as transmit) frame-synchronization is inverted as determined by FSRP before being driven out on the FSR pin.

12.7.16 Set the Receive Frame-Synchronization Polarity

The FSRP bit (see Table 12–35) determines whether frame-synchronization pulses are active high or active low on the FSR pin.

Table 12–35. Register Bit Used to Set Receive Frame-Synchronization Polarity

Register	Bit	Name	Function		Type	Reset Value
PCR	2	FSRP	Receive frame-synchronization polarity		R/W	0
			FSRP = 0	Frame-synchronization pulse FSR is active high.		
			FSRP = 1	Frame-synchronization pulse FSR is active low.		

12.7.16.1 Frame-Synchronization Pulses, Clock Signals, and Their Polarities

Receive frame-synchronization pulses can be generated internally by the sample rate generator (see Section 12.3.2) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSRM, in PCR. FSR is also affected by the GSYNC bit in SRGR2. For information about the effects of FSRM and GSYNC, see Section 12.7.15, *Set the Receive Frame-Synchronization Mode*. Similarly, receive clocks can be selected to be inputs or outputs by programming the mode bit, CLKRM, in the PCR (see Section 12.7.18, *Set the Receive Clock Mode*).

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from an external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of the internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

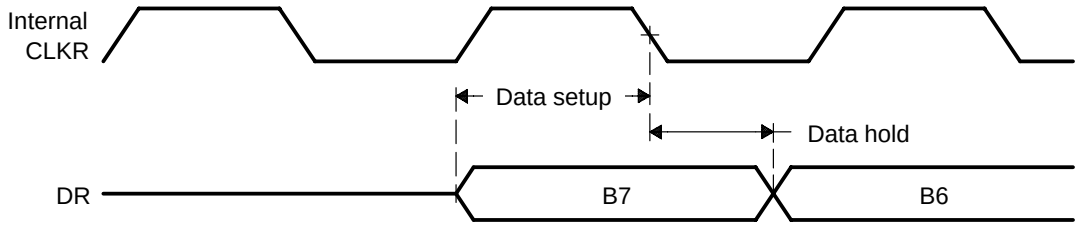
FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1, and internal clocking is selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the CLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal CLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising-edge triggered input clock on CLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the CLKR pin.

CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge. Figure 12–49 shows how data clocked by an external serial device using a rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.

Figure 12–49. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



12.7.17 Set the SRG Frame-Synchronization Period and Pulse Width

The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. If the sample rate generator supplies receive or transmit frame synchronization, you must program the bit fields FPER and FWID.

On FSG, the period from the start of a frame-synchronization pulse to the start of the next pulse is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-synchronization period of 1 to 4096 CLKG cycles, which allows up to 4096 data-bits per frame. When GSYNC = 1, FPER is a don't care value.

Each pulse on FSG has a width of (FWID + 1) CLKG cycles. The 8 bits of FWID allow a pulse width of 1 to 256 CLKG cycles. It is recommended that FWID be programmed to a value less than the programmed word length.

The values in FPER and FWID are loaded into separate down-counters. The 12-bit FPER counter counts down the generated clock cycles from the programmed value (4095 maximum) to 0. The 8-bit FWID counter counts down from the programmed value (255 maximum) to 0.

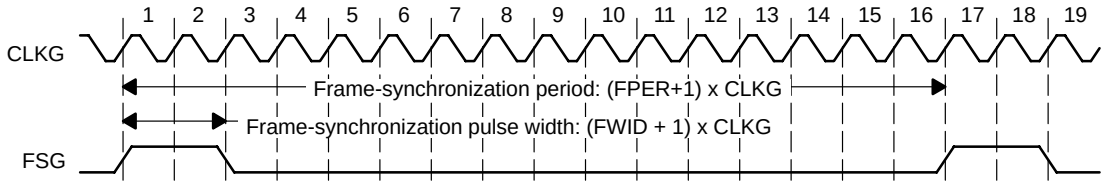
Table 12–36 shows settings for FPER and FWID.

Table 12–36. Register Bits Used to Set the SRG Frame-Synchronization Period and Pulse Width

Register	Bit	Name	Function	Type	Reset Value
SRGR2	11–0	FPER	Sample rate generator frame-synchronization period For the frame-synchronization signal FSG, (FPER + 1) determines the period from the start of a frame-synchronization pulse to the start of the next frame-synchronization pulse. Range for (FPER + 1): 1 to 4096 CLKG cycles	R/W	0000 0000 0000
SRGR1	15–8	FWID	Sample rate generator frame-synchronization pulse width This field plus 1 determines the width of each frame-synchronization pulse on FSG. Range for (FWID + 1): 1 to 256 CLKG cycles	R/W	0000 0000

Figure 12–50 shows a frame-synchronization period of 16 CLKG periods (FPER = 15 or 00001111b) and a frame-synchronization pulse with an active width of 2 CLKG periods (FWID = 1).

Figure 12–50. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods



When the sample rate generator comes out of reset, FSG is in its inactive state. Then, when $\overline{\text{GRST}}$ = 1 and FSGM = 1, a frame-synchronization pulse is generated. The frame width value (FWID + 1) is counted down on every CLKG cycle until it reaches 0, at which time FSG goes low. At the same time, the frame period value (FPER + 1) is also counting down. When this value reaches 0, FSG goes high, indicating a new frame.

12.7.18 Set the Receive Clock Mode

Table 12–37 shows the settings for bits used to set the receive clock mode.

Table 12–37. Register Bits Used to Set the Receive Clock Mode

Register	Bit	Name	Function		Type	Reset Value
PCR	8	CLKRM	Receive clock mode		R/W	0
			Case 1: Digital loopback mode not set (DLB = 0) in SPCR1.			
			CLKRM = 0	The CLKR pin is an input pin that supplies the internal receive clock (CLKR).		
			CLKRM = 1	Internal CLKR is driven by the sample rate generator of the McBSP. The CLKR pin is an output pin that reflects internal CLKR.		
			Case 2: Digital loopback mode set (DLB = 1) in SPCR1.			
			CLKRM = 0	The CLKR pin is in high impedance state. The internal receive clock (CLKR) is driven by the internal transmit clock (CLKX). Internal CLKX is derived according to the CLKXM bit of PCR.		
			CLKRM = 1	Internal CLKR is driven by internal CLKX. The CLKR pin is an output pin that reflects internal CLKR. Internal CLKX is derived according to the CLKXM bit of PCR.		
SPCR1	15	DLB	Digital loopback mode		R/W	00
			DLB = 0	Digital loopback mode is disabled.		
			DLB = 1	Digital loopback mode is enabled. The receive signals, including the receive frame-synchronization signal, are connected internally through multiplexers to the corresponding transmit signals.		
SPCR1	12:11	CLKSTP	Clock stop mode		R/W	00
			CLKSTP = 0Xb	Clock stop mode disabled; usual clocking for non-SPI mode.		
			CLKSTP = 10b	Clock stop mode enabled without clock delay. The internal receive clock signal (CLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.		
			CLKSTP = 11b	Clock stop mode enabled with clock delay. The internal receive clock signal (CLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.		

12.7.18.1 Selecting a Source for the Receive Clock and a Data Direction for the CLKR Pin

Table 12–38 shows how you can select various sources to provide the receive clock signal and affect the CLKR pin. The polarity of the signal on the CLKR pin is determined by the CLKRP bit.

In digital loopback mode (DLB = 1), the transmit clock signal is used as the receive clock signal.

Also, in clock stop mode, the internal receive clock signal (CLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.

Table 12–38. Receive Clock Signal Source Selection

DLB in SPCR1	CLKRM in PCR	Source of Receive Clock	CLKR Pin Status
0	0	The CLKR pin is an input driven by an external clock. The external clock signal is inverted as determined by CLKRP before being used.	Input
0	1	The sample rate generator clock (CLKG) drives internal CLKR.	Output. CLKG, inverted as determined by CLKRP, is driven out on the CLKR pin.
1	0	Internal CLKX drives internal CLKR. To configure CLKX, see Section 12.8.18, *Set the Transmit Clock Mode*.	High impedance
1	1	Internal CLKX drives internal CLKR. To configure CLKX, see Section 12.8.18, *Set the Transmit Clock Mode*.	Output. Internal CLKR (same as internal CLKX) is inverted as determined by CLKRP before being driven out on the CLKR pin.

12.7.19 Set the Receive Clock Polarity

Table 12–39. Register Bit Used to Set Receive Clock Polarity

Register	Bit	Name	Function		Type	Reset Value
PCR	0	CLKRP	Receive clock polarity		R/W	0
			CLKRP = 0	Receive data sampled on falling edge of CLKR		
			CLKRP = 1	Receive data sampled on rising edge of CLKR		

12.7.19.1 Frame Synchronization Pulses, Clock Signals, and Their Polarities

Receive frame-synchronization pulses can be generated internally by the sample rate generator (see Section 12.3.2) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSRM, in PCR. FSR is also affected by the GSYNC bit in SRGR2. For informa-

tion about the effects of FSRM and GSYNC, see Section 12.7.15, *Set the Receive Frame-Synchronization Mode*. Similarly, receive clocks can be selected to be inputs or outputs by programming the mode bit, CLKRM, in the PCR (see Section 12.7.18, *Set the Receive Clock Mode*).

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from an external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of the internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame-synchronization (FSR/FSX are inputs to McBSP) and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

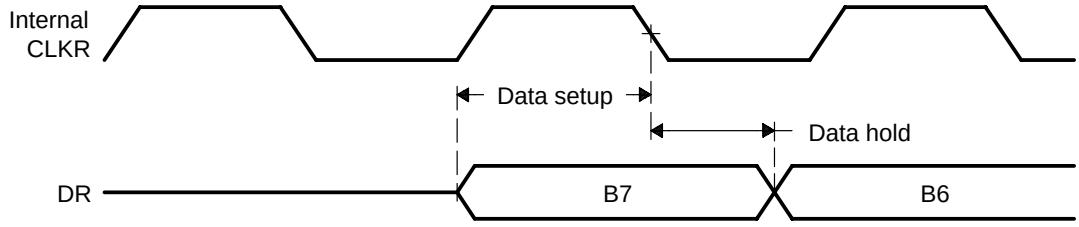
On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1 and internal clocking is selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the CLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal CLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising-edge triggered input clock on CLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the CLKR pin.

CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge. Figure 12–51 shows how data clocked by an external serial device using a

rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.

Figure 12–51. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



12.7.20 Set the SRG Clock Divide-Down Value

Table 12–40. Register Bits Used to Set the Sample Rate Generator (SRG) Clock Divide-Down Value

Register	Bit	Name	Function	Type	Reset Value
SRGR1	7:0	CLKGDV	Sample rate generator clock divide-down value The input clock of the sample rate generator is divided by (CLKGDV + 1) to generate the required sample rate generator clock frequency. The default value of CLKGDV is 1 (divide input clock by 2).	R/W	0000 0001

12.7.20.1 Sample Rate Generator Clock Divider

The first divider stage generates the serial data bit clock from the input clock. This divider stage uses a counter, preloaded by CLKGDV, that contains the divide ratio value.

The output of the first divider stage is the data bit clock, which is output as CLKG and which serves as the input for the second and third stages of the divider.

CLKG has a frequency equal to 1/(CLKGDV + 1) of the sample-rate generator input clock. Thus, the sample-rate generator input clock frequency is divided by a value between 1 and 256. When CLKGDV is odd or equal to 0, the CLKG duty cycle is 50%. When CLKGDV is an even value, 2p, representing an odd divide-down, the high-state duration is p + 1 cycles and the low-state duration is p cycles.

12.7.21 Set the SRG Clock Synchronization Mode

Table 12–41. Register Bit Used to Set the SRG Clock Synchronization Mode

Register	Bit	Name	Function		Type	Reset Value
SRGR2	15	GSYNC	Sample-rate generator clock synchronization		R/W	0
			GSYNC is used only when the input clock source for the sample rate generator is external—on the CLKS, CLKR, or CLKX pin.			
			GSYNC = 0	The sample rate generator clock (CLKG) is free-running. CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles.		
			GSYNC = 1	Clock synchronization is performed when a pulse is detected on the FSR pin: ❑ CLKG is adjusted as necessary so that it is synchronized with the input clock on the CLKS, CLKR, or CLKX pin. ❑ FSG pulses. FSG only pulses in response to a pulse on the FSR pin. The frame-synchronization period defined in FPER is ignored.		

For more details on using the clock synchronization feature, see Section 12.3.3, *Synchronizing Sample Rate Generator Outputs to an External Clock*.

12.7.22 Set the SRG Clock Mode (Choose an Input Clock)

Table 12–42. Register Bits Used to Set the SRG Clock Mode (Choose an Input Clock)

Register	Bit	Name	Function		Type	Reset Value
PCR SRGR2	7 13	SCLKME CLKSM	Sample-rate generator clock mode		R/W R/W	0 1
			SCLKME = 0 CLKSM = 0	Sample-rate generator clock derived from CLKS pin		
			SCLKME = 0 CLKSM = 1	Sample-rate generator clock derived from CPU clock (This is the condition forced by an OMAP730 reset.)		
			SCLKME = 1 CLKSM = 0	Sample-rate generator clock derived from CLKR pin		
			SCLKME = 1 CLKSM = 1	Sample-rate generator clock derived from CLKX pin		

12.7.22.1 SRG Clock Mode

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both, but CLKG is derived from an input clock. Table 12–42 shows the four possible sources of the input clock. For more details on generating CLKG, see Section 12.3.1, *Clock Generation in the Sample Rate Generator*.

12.7.23 Set the SRG Input Clock Polarity

Table 12–43. Register Bits Used to Set the SRG Input Clock Polarity

Register	Bit	Name	Function		Type	Reset Value
SRGR2	14	CLKSP	CLKS pin polarity		R/W	0
			CLKSP determines the input clock polarity when the CLKS pin supplies the input clock (SCLKME = 0 and CLKSM = 0).			
			CLKSP = 0	Rising edge on CLKS pin generates CLKG and FSG.		
			CLKSP = 1	Falling edge on CLKS pin generates CLKG and FSG.		
PCR	1	CLKXP	CLKX pin polarity		R/W	0
			CLKXP determines the input clock polarity when the CLKX pin supplies the input clock (SCLKME = 1 and CLKSM = 1).			
			CLKXP = 0	Rising edge on CLKX pin generates transitions on CLKG and FSG.		
			CLKXP = 1	Falling edge on CLKX pin generates transitions on CLKG and FSG.		
PCR	0	CLKRP	CLKR pin polarity		R/W	0
			CLKRP determines the input clock polarity when the CLKR pin supplies the input clock (SCLKME = 1 and CLKSM = 0).			
			CLKRP = 0	Falling edge on CLKR pin generates transitions on CLKG and FSG.		
			CLKRP = 1	Rising edge on CLKR pin generates transitions on CLKG and FSG.		

12.7.23.1 Using CLKSP/CLKXP/CLKRP to Choose an Input Clock Polarity

The sample rate generator can produce a clock signal (CLKG) and a frame-synchronization signal (FSG) for use by the receiver, the transmitter, or both. To produce CLKG and FSG, the sample rate generator must be driven by an input clock signal derived from the CPU clock or from an external clock on the CLKS, CLKX, or CLKR pin. If you use a pin, choose a polarity for that pin by using the appropriate polarity bit (CLKSP for the CLKS pin, CLKXP for the CLKX pin, CLKRP for the CLKR pin). The polarity determines whether the rising or falling edge of the input clock generates transitions on CLKG and FSG.

12.8 Transmitter Configuration

To configure the McBSP transmitter, perform the following procedure:

1) Place the McBSP/transmitter in reset (see Section 12.8.2).
2) Program the McBSP registers for the desired transmitter operation (see Section 12.8.1).
3) Take the transmitter out of reset (see Section 12.8.2).

12.8.1 Programming the McBSP Registers for the Desired Transmitter Operation

The following are important tasks to be performed when you are configuring the McBSP transmitter. Each task corresponds to one or more McBSP register bit fields.

It may be helpful to print the McBSP Register Worksheet at the end of this chapter first and to fill in the worksheet as you read the tasks.

- ❏ Global behavior:
  - ■ Set the transmitter pins to operate as McBSP pins.
  - ■ Enable/disable the digital loopback mode.
  - ■ Enable/disable the clock stop mode.
  - ■ Enable/disable transmit multichannel selection.
- ❏ Data behavior:
  - ■ Choose 1 or 2 phases for the transmit frame.
  - ■ Set the transmit word length(s).
  - ■ Set the transmit frame length.
  - ■ Enable/disable the transmit frame-synchronization ignore function.
  - ■ Set the transmit companding mode.
  - ■ Set the transmit data delay.
  - ■ Set the transmit DXENA mode.
  - ■ Set the transmit interrupt mode.
- ❏ Frame-synchronization behavior:
  - ■ Set the transmit frame-synchronization mode.
  - ■ Set the transmit frame-synchronization polarity.
  - ■ Set the SRG frame-synchronization period and pulse width.
- ❏ Clock behavior:
  - ■ Set the transmit clock mode.
  - ■ Set the transmit clock polarity.
  - ■ Set the SRG clock divide-down value.
  - ■ Set the SRG clock synchronization mode.
  - ■ Set the SRG clock mode (choose an input clock).
  - ■ Set the SRG input clock polarity.

12.8.2 Resetting and Enabling the Transmitter

The first step of the transmitter configuration procedure is to reset the transmitter, and the last step is to enable the transmitter (to take it out of reset). Table 12–44 describes the bits used for both of these steps.

Table 12–44. Register Bits Used to Place Transmitter in Reset

Register	Bit	Name	Function		Type	Reset Value
SPCR2	0	$\overline{\text{XRST}}$	Transmitter reset		R/W	0
			$\overline{\text{XRST}}$ = 0	The serial port transmitter is disabled and is in the reset state.		
			$\overline{\text{XRST}}$ = 1	The serial port transmitter is enabled.		
SPCR2	6	$\overline{\text{GRST}}$	Sample rate generator reset		R/W	0
			$\overline{\text{GRST}}$ = 0	Sample rate generator is reset. If $\overline{\text{GRST}}$ = 0 because of an OMAP730 reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven low (inactive). If $\overline{\text{GRST}}$ = 0 because of program code, CLKG and FSG are both driven low (inactive).		
			$\overline{\text{GRST}}$ = 1	Sample rate generator is enabled. CLKG is driven according to the configuration programmed in the sample-rate generator registers (SRGR[1,2]). If $\overline{\text{FRST}}$ = 1, the generator also generates the frame-synchronization signal FSG as programmed in the sample-rate generator registers.		
SPCR2	7	$\overline{\text{FRST}}$	Frame-synchronization logic reset		R/W	0
			$\overline{\text{FRST}}$ = 0	Frame-synchronization logic is reset. The sample rate generator does not generate frame-synchronization signal FSG, even if $\overline{\text{GRST}}$ = 1.		
			$\overline{\text{FRST}}$ = 1	If $\overline{\text{GRST}}$ = 1, frame-synchronization signal FSG is generated after (FPER + 1) number of CLKG clock cycles; all frame counters are loaded with their programmed values.		

12.8.2.1 Reset Considerations

The serial port can be reset in the following two ways:

- An OMAP730 reset ($\overline{\text{RESET}}$ signal driven low) places the receiver, transmitter, and sample rate generator in reset. When the device reset is removed ($\overline{\text{RESET}}$ signal released), $\overline{\text{GRST}}$ = $\overline{\text{FRST}}$ = $\overline{\text{RRST}}$ = $\overline{\text{XRST}}$ = 0, keeps the entire serial port in the reset state.
- The serial port transmitter and receiver can be reset directly using the $\overline{\text{RRST}}$ and $\overline{\text{XRST}}$ bits in the serial port control registers. The sample rate generator can be reset directly using the $\overline{\text{GRST}}$ bit in SPCR2.

Table 12–45 shows the state of McBSP pins when the serial port is reset due to an OMAP730 reset and a direct receiver/transmitter reset.

Table 12–45. Reset State of Each McBSP Pin

Pin	Possible State(s)	State Forced by OMAP730 Reset	State Forced by Receiver/Transmitter Reset
Receiver Reset ($\overline{RRST}$ = 0 and $\overline{GRST}$ = 1)			
DR	I	Input	Input
CLKRX	I/O/Z	Input	Known state if input; CLKRX running if output
FSRX	I/O/Z	Input	Known state if input; FSRX inactive state if output
CLKS	I/O/Z	Input	Input
Transmitter Reset ($\overline{XRST}$ = 0 and $\overline{GRST}$ = 1)			
DX	O/Z	High impedance	High impedance
CLKRX	I/O/Z	Input	Known state if input; CLKRX running if output
FSRX	I/O/Z	Input	Known state if input; FSRX inactive state if output

For more details about McBSP reset conditions and effects, see Section 12.10.3, *Resetting and Initializing a McBSP*.

12.8.3 Set the Transmitter Pins to Operate as McBSP Pins

Table 12–46. Register Bit Used to Set Transmitter Pins to Operate as McBSP Pins

Register	Bit	Name	Function		Type	Reset Value
PCR	13	XIOEN	Transmit I/O enable		R/W	0
			This bit is only applicable when the transmitter is in reset state ($\overline{XRST}$ = 0 in SPCR2).			
			XIOEN = 0	The DX, FSX, CLKX, and CLKS pins are configured as serial port pins and do not function as general-purpose I/Os.		
			XIOEN = 1	The DX pin is a general-purpose output pin. The FSX and CLKX pins are general-purpose I/O pins. These serial port pins do not perform serial port operation. The CLKS pin is a general-purpose input pin if RIOEN = XIOEN = 1 and $\overline{RRST}$ = $\overline{XRST}$ = 0. For more information on using these pins as general-purpose I/O pins, see Section 12.9.		

12.8.4 Enable/Disable the Digital Loopback Mode

The DLB bit determines whether the digital loopback mode is on. DLB is described in Table 12–47.

Table 12–47. Register Bit Used to Enable/Disable the Digital Loopback Mode

Register	Bit	Name	Function		Type	Reset Value
SPCR1	15	DLB	Digital loopback mode		R/W	0
			DLB = 0	Digital loopback mode is disabled.		
			DLB = 1	Digital loopback mode is enabled.		

12.8.4.1 Digital Loopback Mode

In the digital loopback mode, the receive signals are connected internally through multiplexers to the corresponding transmit signals, as shown in Table 12–48. This mode allows testing of serial port code with a single DSP device; the McBSP receives the data it transmits.

Table 12–48. Receive Signals Connected to Transmit Signals in Digital Loopback Mode

This Receive Signal	Is Fed Internally by This Transmit Signal
DR (receive data)	DX (transmit data)
FSR (receive frame synchronization)	FSX (transmit frame synchronization)
CLKR (receive clock)	CLKX (transmit clock)

12.8.5 Enable/Disable the Clock Stop Mode

The CLKSTP bits determine whether the clock stop mode is on. CLKSTP is described in Table 12–49.

Table 12–49. Register Bits Used to Enable/Disable the Clock Stop Mode

Register	Bit	Name	Function		Type	Reset Value
SPCR1	12:11	CLKSTP	Clock stop mode		R/W	00
			CLKSTP = 0Xb	Clock stop mode disabled; normal clocking for non-SPI mode.		
			CLKSTP = 10b	Clock stop mode enabled without clock delay		
			CLKSTP = 11b	Clock stop mode enabled with clock delay		

12.8.5.1 Clock Stop Mode

The clock stop mode supports the SPI master-slave protocol. If you do not plan to use the SPI protocol, you can clear CLKSTP to disable the clock stop mode.

In the clock stop mode, the clock stops at the end of each data transfer. At the beginning of each data transfer, the clock starts immediately (CLKSTP = 10b)

or after a half-cycle delay (CLKSTP = 11b). The CLKXP bit determines whether the starting edge of the clock on the CLKX pin is rising or falling. The CLKRP bit determines whether receive data is sampled on the rising or falling edge of the clock shown on the CLKR pin.

Table 12–50 summarizes the effect of CLKSTP, CLKXP, and CLKRP on serial port operation. In the clock stop mode, the receive clock is tied internally to the transmit clock, and the receive frame-synchronization signal is tied internally to the transmit frame-synchronization signal.

Table 12–50. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme

Bit Settings	Clock Scheme
CLKSTP = 00b or 01b CLKXP = 0 or 1 CLKRP = 0 or 1	Clock stop mode disabled. Clock enabled for non-SPI mode.
CLKSTP = 10b CLKXP = 0 CLKRP = 0	Low inactive state without delay: The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of CLKR.
CLKSTP = 11b CLKXP = 0 CLKRP = 1	Low inactive state with delay: The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of CLKR.
CLKSTP = 10b CLKXP = 1 CLKRP = 0	High inactive state without delay: The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of CLKR.
CLKSTP = 11b CLKXP = 1 CLKRP = 1	High inactive state with delay: The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of CLKR.

12.8.6 Enable/Disable Transmit Multichannel Selection

Table 12–51. Register Bits Used to Enable/Disable Transmit Multichannel Selection

Register	Bit	Name	Function		Type	Reset Value
MCR2	1:0	XMCM	Transmit multichannel selection		R/W	00
			XMCM = 00b	No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.		
			XMCM = 01b	All channels are disabled unless they are selected in the appropriate transmit channel enable registers (XCERs). If enabled, a channel in this mode is also unmasked.		
				The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERs.		
			XMCM = 10b	All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (XCERs).		
				The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERs.		
			XMCM = 11b	This mode is used for symmetric transmission and reception.		
				All channels are disabled for transmission unless they are enabled for reception in the appropriate receive channel enable registers (RCERs). Once enabled, they are masked unless they are also selected in the appropriate transmit channel enable registers (XCERs).		
				The XMCME bit determines whether 32 channels or 128 channels are selectable in RCERs and XCERs.		

For more details, see Section 12.5.7, *Transmit Multichannel Selection Modes*.

12.8.7 Choose One or Two Phases for the Transmit Frame

Table 12–52. Register Bit Used to Choose 1 or 2 Phases for the Transmit Frame

Register	Bit	Name	Function		Type	Reset Value
XCR2	15	XPHASE	Transmit phase number		R/W	0
			Specifies whether the transmit frame has 1 or 2 phases.			
			XPHASE = 0	Single-phase frame		
			XPHASE = 1	Dual-phase frame		

12.8.8 Set the Transmit Word Length(s)

Table 12–53. Register Bits Used to Set the Transmit Word Length(s)

Register	Bit	Name	Function	Type	Reset Value
XCR1	7:5	XWDLEN1	Transmit word length of frame phase 1	R/W	000
			XWDLEN1 = 000b 8 bits		
			XWDLEN1 = 001b 12 bits		
			XWDLEN1 = 010b 16 bits		
			XWDLEN1 = 011b 20 bits		
			XWDLEN1 = 100b 24 bits		
			XWDLEN1 = 101b 32 bits		
			XWDLEN1 = 11Xb Reserved		
XCR2	7:5	XWDLEN2	Transmit word length of frame phase 2	R/W	000
			XWDLEN2 = 000b 8 bits		
			XWDLEN2 = 001b 12 bits		
			XWDLEN2 = 010b 16 bits		
			XWDLEN2 = 011b 20 bits		
			XWDLEN2 = 100b 24 bits		
			XWDLEN2 = 101b 32 bits		
			XWDLEN2 = 11Xb Reserved		

12.8.8.1 Word Length Bits

Each frame can have one or two phases, depending on the value that you load into the RPHASE bit. If a single-phase frame is selected, XWDLEN1 selects the length for every serial word transmitted in the frame. If a dual-phase frame is selected, XWDLEN1 determines the length of the serial words in phase 1 of the frame, and XWDLEN2 determines the word length in phase 2 of the frame.

12.8.9 Set the Transmit Frame Length

Table 12–54. Register Bits Used to Set the Transmit Frame Length

Register	Bit	Name	Function	Type	Reset Value
XCR1	14:8	XFRLEN1	Transmit frame length 1	R/W	000 0000
			(XFRLEN1 + 1) is the number of serial words in phase 1 of the transmit frame.		
			XFRLEN1 = 000 0000 1 word in phase 1		
			XFRLEN1 = 000 0001 2 words in phase 1		
			| |		
			| |		
			XFRLEN1 = 111 1111 128 words in phase 1		
XCR2	14:8	XFRLEN2	Transmit frame length 2	R/W	000 0000
			If a dual-phase frame is selected, (XFRLEN2 + 1) is the number of serial words in phase 2 of the transmit frame.		
			XFRLEN2 = 000 0000 1 word in phase 2		
			XFRLEN2 = 000 0001 2 words in phase 2		
			| |		
			| |		
			XFRLEN2 = 111 1111 128 words in phase 2		

12.8.9.1 Selected Frame Length

The transmit frame length is the number of serial words in the transmit frame. Each frame can have one or two phases, depending on the value that you load into the XPHASE bit.

If a single-phase frame is selected (XPHASE = 0), the frame length is equal to the length of phase 1. If a dual-phase frame is selected (XPHASE = 1), the frame length is the length of phase 1 plus the length of phase 2.

The 7-bit XFRLEN fields allow up to 128 words per phase. See Table 12–55 for a summary of how to calculate the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization pulse.

Note:

Program the XFRLEN fields with [*w minus 1*], where *w* represents the number of words per phase. For example, if you want a phase length of 128 words in phase 1, load 127 into XFRLEN1.

Table 12–55. How to Calculate Frame Length

XPHASE	XFRLEN1	XFRLEN2	Frame Length
0	0 ≤ XFRLEN1 ≤ 127	Don't care	(XFRLEN1 + 1) words
1	0 ≤ XFRLEN1 ≤ 127	0 ≤ XFRLEN2 ≤ 127	(XFRLEN1 + 1) + (XFRLEN2 + 1) words

12.8.10 Enable/Disable the Transmit Frame-Synchronization Ignore Function

Table 12–56. Register Bit Used to Enable/Disable the Transmit Frame-Synchronization Ignore Function

Register	Bit	Name	Function		Type	Reset Value
XCR2	2	XFIG	Transmit frame-synchronization ignore		R/W	0
			XFIG = 0	An unexpected transmit frame-synchronization pulse causes the McBSP to restart the frame transfer.		
			XFIG = 1	The McBSP ignores unexpected transmit frame-synchronization pulses.		

12.8.10.1 Unexpected Frame-Synchronization Pulses and Frame-Synchronization Ignore

If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-synchronization pulse.

When XFIG = 1, normal transmission continues with unexpected frame-synchronization signals ignored.

When XFIG = 0 and an unexpected frame-synchronization pulse occurs, the serial port:

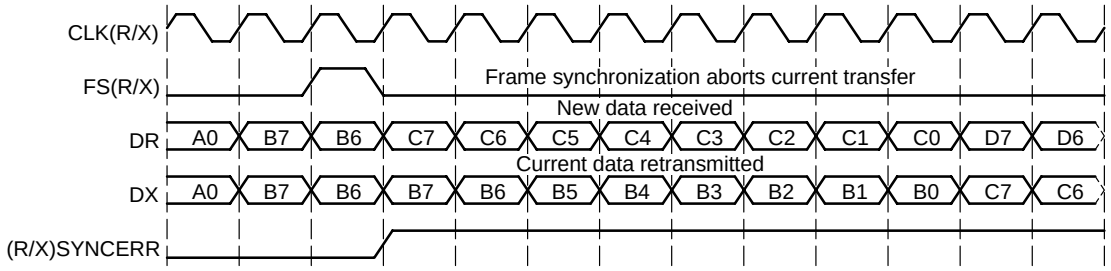
1) Aborts the present transmission
2) Sets XSYNCERR to 1 in SPCR2
3) Reinitiates transmission of the current word that was aborted

For more details about the frame-synchronization error condition, see Section 12.4.5, *Unexpected Transmit Frame-Synchronization Pulse*.

12.8.10.2 Examples Showing the Effects of XFIG

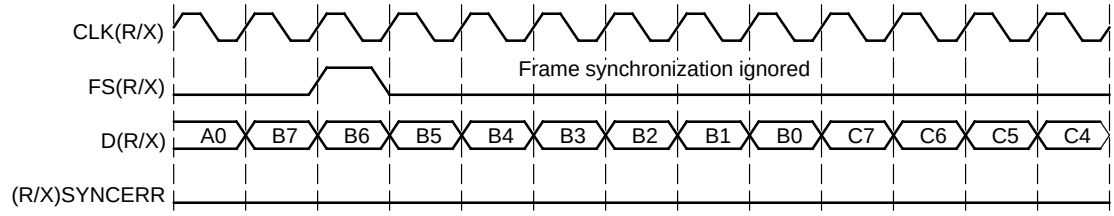
Figure 12–52 shows an example in which word B is interrupted by an unexpected frame-synchronization pulse when (R/X)FIG = 0. In the case of transmission, the transmission of B is aborted (B is lost). This condition is a transmit synchronization error, which sets the XSYNCERR bit. No new data has been written to DXR[1,2]; therefore, the McBSP transmits B again.

Figure 12–52. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 0



In contrast to Figure 12–52, Figure 12–53 shows McBSP operation when unexpected frame-synchronization signals are ignored (when (R/X)FIG = 1). Here, the transfer of word B is not affected by an unexpected frame-synchronization pulse.

Figure 12–53. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 1



12.8.11 Set the Transmit Companding Mode

Table 12–57. Register Bits Used to Set the Transmit Companding Mode

Register	Bit	Name	Function		Type	Reset Value
XCR2	4:3	XCOMPAND	Transmit companding mode		R/W	00
			Modes other than 00b are enabled only when the appropriate XWDLEN is 000b, indicating 8-bit data.			
			XCOMPAND = 00b	No companding, any size data, MSB transmitted first		
			XCOMPAND = 01b	No companding, 8-bit data, LSB transmitted first (for details, see Section 12.7.11.4, *Option to Receive LSB First*)		
			XCOMPAND = 10b	μ-law companding, 8-bit data, MSB transmitted first		
			XCOMPAND = 11b	A-law companding, 8-bit data, MSB transmitted first		

12.8.11.1 Companding

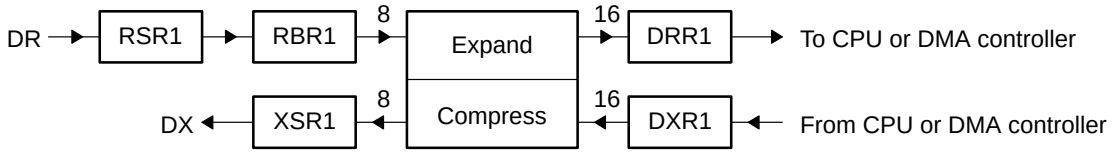
Companding (COMpressing and exPANDing) hardware allows compression and expansion of data in either μ-law or A-law format. The companding standard employed in the United States and Japan is μ-law. The European companding standard is referred to as A-law. The specifications for μ-law and A-law log PCM are part of the CCITT G.711 recommendation.

A-law and μ-law allow 13 bits and 14 bits of dynamic range, respectively. Any values outside this range are set to the most positive or most negative value. Thus, for companding to work best, the data transferred to and from the McBSP via the CPU or DMA controller must be at least 16 bits wide.

The μ-law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word length bits (RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

Figure 12–54 illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or μ-law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to twos-complement format.

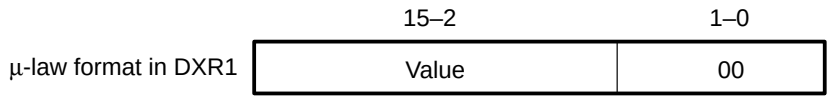
Figure 12–54. Companding Processes for Reception and for Transmission



12.8.11.2 Format for Data To Be Compressed

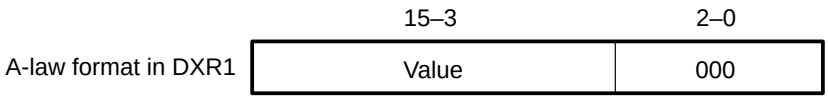
For transmission using μ-law compression, ensure that the 14 data-bits are left-justified in DXR1, with the remaining two low-order bits filled with zeros, as shown in Figure 12–55.

Figure 12–55. μ-Law Transmit Data Companding Format



For transmission using A-law compression, ensure that the 13 data-bits are left-justified in DXR1, with the remaining three low-order bits filled with zeros, as shown in Figure 12–56.

Figure 12–56. A-Law Transmit Data Companding Format



12.8.11.3 Capability to Compand Internal Data

If the McBSP is otherwise unused (the serial port transmit and receive sections are reset), the companding hardware can compand internal data. See Section 12.2.2.2, *Capability to Compand Internal Data*.

12.8.11.4 Option to Transmit LSB First

Usually, the McBSP transmits or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols (that do not use companded data) require the least significant bit (LSB) to be transferred first. If you set XCOMPAND = 01b in XCR2, the bit ordering of 8-bit words is reversed (LSB first) before being sent from the serial port. Similar to companding, this feature is enabled only if the appropriate word length bits are set to 0, indicating that 8-bit words are to be transferred serially. If either phase of the frame does not have an 8-bit word length, the McBSP assumes the word length is 8 bits and LSB-first ordering is done.

12.8.12 Set the Transmit Data Delay

Table 12–58. Register Bits Used to Set the Transmit Data Delay

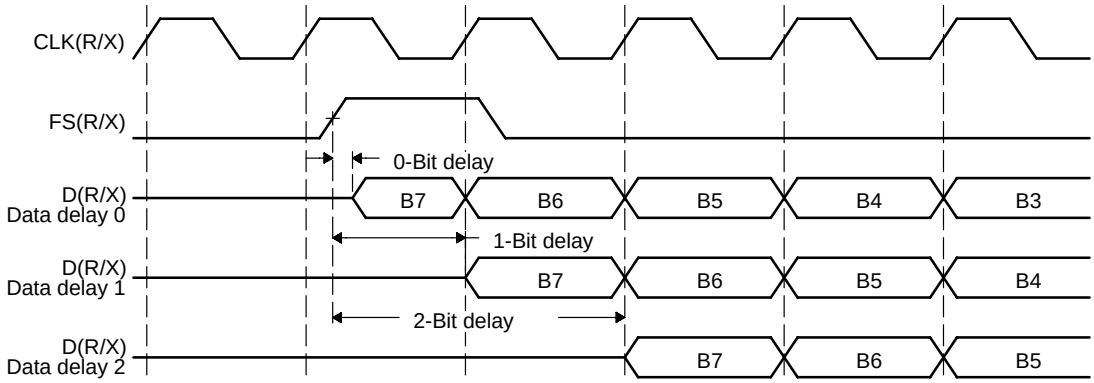
Register	Bit	Name	Function		Type	Reset Value
XCR2	1:0	XDATDLY	Transmitter data delay		R/W	00
			XDATDLY = 00	0-bit data delay		
			XDATDLY = 01	1-bit data delay		
			XDATDLY = 10	2-bit data delay		
			XDATDLY = 11	Reserved		

12.8.12.1 Data Delay

The start of a frame is defined by the first clock cycle in which frame synchronization is found to be active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if necessary. This delay is called data delay.

XDATDLY specifies the data delay for transmission. The range of programmable data delay is 0 to 2 bit-clocks (XDATDLY = 00b–10b), as described in Table 12–58 and Figure 12–57. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on. Typically, a 1-bit delay is selected, because data often follows a 1-cycle active frame-synchronization pulse.

Figure 12–57. Range of Programmable Data Delay



12.8.12.2 0-Bit Data Delay

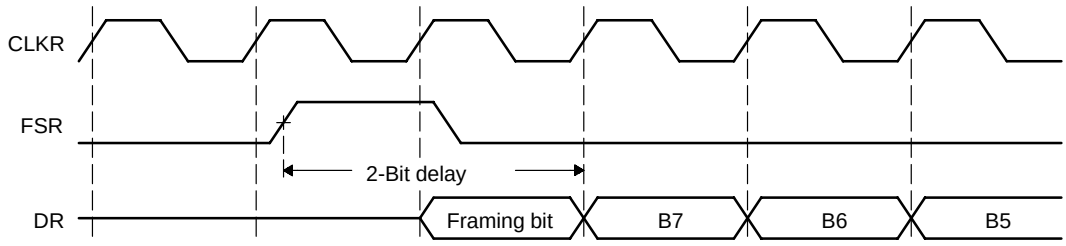
Usually, a frame-synchronization pulse is detected or sampled with respect to an edge of an internal serial clock CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data can be received or transmitted. However, in the case of 0-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle.

For reception, this problem is solved because receive data is sampled on the first falling edge of CLKR where an active-high internal FSR is detected. However, data transmission must begin on the rising edge of the internal CLKX clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in XSR1, and thus DX. The transmitter then asynchronously detects the frame synchronization, FSX, going active high and immediately starts driving the first bit to be transmitted on the DX pin.

12.8.12.3 2-Bit Data Delay

A data delay of 2 bit-periods allows the serial port to interface to different types of T1 framing devices where the data stream is preceded by a framing bit. During reception of such a stream with data delay of 2 bits (framing bit appears after a 1-bit delay and data appears after a 2-bit delay), the serial port essentially discards the framing bit from the data stream, as shown in the following figure. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on.

Figure 12–58. 2-Bit Data Delay Used to Skip a Framing Bit



12.8.13 Set the Transmit DXENA Mode

Table 12–59. Register Bit Used to Set the Transmit DXENA (DX Delay Enabler) Mode

Register	Bit	Name	Function		Type	Reset Value
SPCR1	7	DXENA	DX delay enabler mode		R/W	0
			DXENA = 0	DX delay enabler is off.		
			DXENA = 1	DX delay enabler is on.		

12.8.13.1 DXENA Mode

The DXENA bit controls the delay enabler on the DX pin. Set DXENA to enable an extra delay for turn-on time (for the length of the delay, see the data sheet for your TMS320C55x DSP). This bit does not control the data itself, so only the first bit is delayed.

If you tie together the DX pins of multiple McBSPs, ensure that DXENA = 1, to avoid having more than one McBSP transmit on the data line at one time.

12.8.14 Set the Transmit Interrupt Mode

The transmitter interrupt (XINT) signals the CPU of changes to the serial port status. Four options exist for configuring this interrupt. The options are set by the transmit interrupt mode bits, XINTM, in SPCR2.

Table 12–60. Register Bits Used to Set the Transmit Interrupt Mode

Register	Bit	Name	Function		Type	Reset Value
SPCR2	5:4	XINTM	Transmit interrupt mode		R/W	00
			XINTM = 00	XINT generated when XRDY changes from 0 to 1.		
			XINTM = 01	XINT generated by an end-of-block or end-of-frame condition in a transmit multichannel selection mode. In any of the transmit multichannel selection modes, interrupt after every 16-channel block boundary has been crossed within a frame and at the end of the frame. For details, see Section 12.5.8, *Using Interrupts Between Block Transfers*. In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated.		
			XINTM = 10	XINT generated by a new transmit frame-synchronization pulse. Interrupt on detection of each transmit frame-synchronization pulse. This generates an interrupt even when the transmitter is in its reset state. This is done by synchronizing the incoming frame-synchronization pulse to the CPU clock and sending it to the CPU via XINT.		
			XINTM = 11	XINT generated when XSYNCERR is set. Interrupt on frame-synchronization error. Regardless of the value of XINTM, XSYNCERR can be read to detect this condition. For more information on using XSYNCERR, see Section 12.4.5, *Unexpected Transmit Frame-Synchronization Pulse*.		

12.8.15 Set the Transmit Frame-Synchronization Mode

Table 12–61. Register Bits Used to Set the Transmit Frame-Synchronization Mode

Register	Bit	Name	Function		Type	Reset Value
PCR	11	FSXM	Transmit frame-synchronization mode		R/W	0
			FSXM = 0	Transmit frame synchronization is supplied by an external source via the FSX pin.		
			FSXM = 1	Transmit frame synchronization is supplied by the McBSP, as determined by the FSGM bit of SRGR2.		
SRGR2	12	FSGM	Sample rate generator transmit frame-synchronization mode		R/W	0
			Used when FSXM = 1 in PCR.			
			FSGM = 0	The McBSP generates a transmit frame-synchronization pulse when the content of DXR[1,2] is copied to XSR[1,2].		
			FSGM = 1	The transmitter uses frame-synchronization pulses generated by the sample rate generator. Program the FWID bits to set the width of each pulse. Program the FPER bits to set the frame-synchronization period.		

12.8.15.1 Transmit Frame-Synchronization Modes

Table 12–62 shows how FSXM and FSGM select the source of transmit frame-synchronization pulses. The three choices are:

- External frame-synchronization input
- Sample-rate generator frame-synchronization signal (FSG)
- Internal signal that indicates a DXR-to-XSR copy has been made

Table 12–62 also shows the effect of each bit setting on the FSX pin. The polarity of the signal on the FSX pin is determined by the FSXP bit.

Table 12–62. How FSXM and FSGM Select the Source of Transmit Frame-Synchronization Pulses

FSXM	FSGM	Source of Transmit Frame Synchronization	FSX Pin Status
0	0 or 1	An external frame-synchronization signal enters the McBSP through the FSX pin. The signal is then inverted by FSXP before being used as internal FSX.	Input
1	1	Internal FSX is driven by the sample rate generator frame-synchronization signal (FSG).	Output. FSG is inverted by FSXP before being driven out on FSX pin.
1	0	A DXR-to-XSR copy causes the McBSP to generate a transmit frame-synchronization pulse that is 1 cycle wide.	Output. The generated frame-synchronization pulse is inverted as determined by FSXP before being driven out on FSX pin.

12.8.15.2 Other Considerations

If the sample rate generator creates a frame-synchronization signal (FSG) that is derived from an external input clock, the GSYNC bit determines whether FSG is kept synchronized with pulses on the FSR pin. For more details, see Section 12.3.3, *Synchronizing Sample Rate Generator Outputs to an External Clock*.

In the clock stop mode (CLKSTP = 10b or 11b), the McBSP can act as a master or as a slave in the SPI protocol. If the McBSP is a master and must provide a slave-enable signal (SS_) on the FSX pin, ensure that FSXM = 1 and FSGM = 0 so that FSX is an output and is driven active for the duration of each transmission. If the McBSP is a slave, ensure that FSXM = 0 so that the McBSP can receive the slave-enable signal on the FSX pin.

12.8.16 Set the Transmit Frame-Synchronization Polarity

Table 12–63. Register Bit Used to Set Transmit Frame-Synchronization Polarity

Register	Bit	Name	Function		Type	Reset Value
PCR	3	FSXP	Transmit frame-synchronization polarity		R/W	0
			FSXP = 0	Frame-synchronization pulse FSX is active high.		
			FSXP = 1	Frame-synchronization pulse FSX is active low.		

12.8.16.1 Frame Synchronization Pulses, Clock Signals, and Their Polarities

Transmit frame-synchronization pulses can be generated internally by the sample rate generator (see Section 12.3.2) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSXM, in PCR. FSX is also affected by the FSGM bit in SRGR2. For informa-

tion about the effects of FSXM and FSGM, see Section 12.8.15, *Set the Transmit Frame-Synchronization Mode*). Similarly, transmit clocks can be selected to be inputs or outputs by programming the mode bit, CLKXM, in the PCR (see Section 12.8.18, *Set the Transmit Clock Mode*).

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

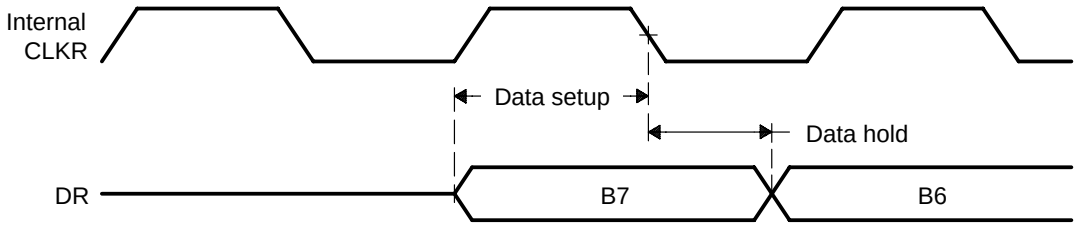
FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP) and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected and the polarity bit FS(R/X)P = 1, the internal active-high frame-synchronization signals are inverted before being sent to the FS(R/X) pin.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1, and internal clocking selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the CLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising-edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal CLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising-edge triggered input clock on CLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the CLKR pin.

CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge. Figure 12–59 shows how data clocked by an external serial device using a rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.

Figure 12–59. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



12.8.17 Set the SRG Frame-Synchronization Period and Pulse Width

Table 12–64. Register Bits Used to Set SRG Frame-Synchronization Period and Pulse Width

Register	Bit	Name	Function	Type	Reset Value
SRGR2	11:0	FPER	Sample-rate generator frame-synchronization period For the frame-synchronization signal FSG, (FPER + 1) determines the period from the start of a frame-synchronization pulse to the start of the next frame-synchronization pulse. Range for (FPER + 1): 1 to 4096 CLKG cycles.	R/W	0000 0000 0000
SRGR1	15:8	FWID	Sample-rate generator frame-synchronization pulse width This field plus 1 determines the width of each frame-synchronization pulse on FSG. Range for (FWID + 1): 1 to 256 CLKG cycles.	R/W	0000 0000

12.8.17.1 Frame-Synchronization Period and Frame-Synchronization Pulse Width

The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. If the sample rate generator is supplying receive or transmit frame synchronization, you must program the bit fields FPER and FWID.

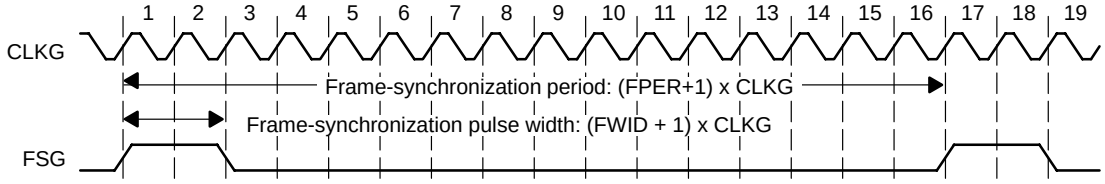
On FSG, the period from the start of a frame-synchronization pulse to the start of the next pulse is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-synchronization period of 1 to 4096 CLKG cycles, which allows up to 4096 data bits per frame. When GSYNC = 1, FPER is a don't care value.

Each pulse on FSG has a width of (FWID + 1) CLKG cycles. The 8 bits of FWID allow a pulse width of 1 to 256 CLKG cycles. It is recommended that FWID be programmed to a value less than the programmed word length.

The values in FPER and FWID are loaded into separate down-counters. The 12-bit FPER counter counts down the generated clock cycles from the programmed value (4095 maximum) to 0. The 8-bit FWID counter counts down from the programmed value (255 maximum) to 0.

Figure 12–60 shows a frame-synchronization period of 16 CLKG periods (FPER = 15 or 00001111b) and a frame-synchronization pulse with an active width of 2 CLKG periods (FWID = 1).

Figure 12–60. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods



When the sample rate generator comes out of reset, FSG is in its inactive state. Then, when $\overline{\text{GRST}}$ = 1 and FSGM = 1, a frame-synchronization pulse is generated. The frame width value (FWID + 1) is counted down on every CLKG cycle until it reaches 0, at which time FSG goes low. At the same time, the frame period value (FPER + 1) is also counting down. When this value reaches 0, FSG goes high, indicating a new frame.

12.8.18 Set the Transmit Clock Mode

Table 12–65. Register Bit Used to Set the Transmit Clock Mode

Register	Bit	Name	Function		Type	Reset Value
PCR	9	CLKXM	Transmit clock mode		R/W	0
			CLKXM = 0	The transmitter gets its clock signal from an external source via the CLKX pin.		
			CLKXM = 1	The CLKX pin is an output pin driven by the sample rate generator of the McBSP.		

12.8.18.1 Selecting a Source for the Transmit Clock and a Data Direction for the CLKX Pin

Table 12–66 shows how the CLKXM bit selects the transmit clock and the corresponding status of the CLKX pin. The polarity of the signal on the CLKX pin is determined by the CLKXP bit.

Table 12–66. How the CLKXM Bit Selects the Transmit Clock and the Corresponding Status of the CLKX Pin

CLKXM in PCR	Source of Transmit Clock	CLKX Pin Status
0	Internal CLKX is driven by an external clock on the CLKX pin. CLKX is inverted as determined by CLKXP before being used.	Input
1	Internal CLKX is driven by the sample rate generator clock, CLKG.	Output. CLKG, inverted as determined by CLKXP, is driven out on CLKX.

12.8.18.2 Other Considerations

If the sample rate generator creates a clock signal (CLKG) that is derived from an external input clock, the GSYNC bit determines whether CLKG is kept synchronized with pulses on the FSR pin. For more details, see Section 12.3.3, *Synchronizing Sample Rate Generator Outputs to an External Clock*.

In the clock stop mode (CLKSTP = 10b or 11b), the McBSP can act as a master or as a slave in the SPI protocol. If the McBSP is a master, ensure that CLKXM = 1 so that CLKX is an output to supply the master clock to any slave devices. If the McBSP is a slave, ensure that CLKXM = 0 so that CLKX is an input to accept the master clock signal.

12.8.19 Set the Transmit Clock Polarity

Table 12–67. Register Bit Used to Set Transmit Clock Polarity

Register	Bit	Name	Function		Type	Reset Value
PCR	1	CLKXP	Transmit clock polarity		R/W	0
			CLKXP = 0	Transmit data sampled on rising edge of CLKX.		
			CLKXP = 1	Transmit data sampled on falling edge of CLKX.		

12.8.19.1 Frame Synchronization Pulses, Clock Signals, and Their Polarities

Transmit frame-synchronization pulses can be either generated internally by the sample rate generator (see Section 12.3.2) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSXM, in PCR. FSX is also affected by the FSGM bit in SRGR2. For information about the effects of FSXM and FSGM, see Section 12.8.15, *Set the Transmit Frame-Synchronization Mode*). Similarly, transmit clocks can be selected to be inputs or outputs by programming the mode bit, CLKXM, in the PCR (see Section 12.8.18, *Set the Transmit Clock Mode*).

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-synchronization pulses), the McBSP detects them on the internal falling

edge of the clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from an external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transitioned to their active state) on the rising edge of the internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

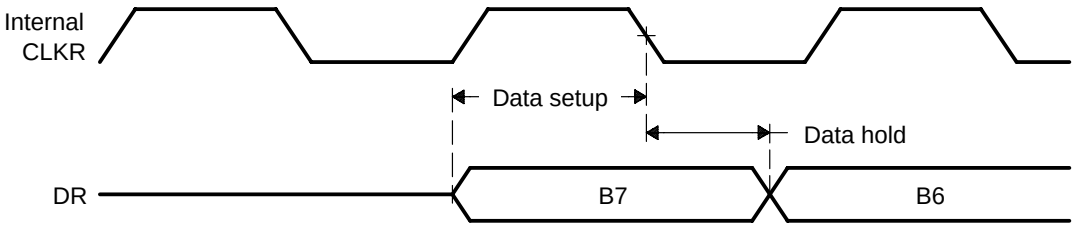
On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1 and internal clocking is selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the CLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising-edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal CLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising-edge triggered input clock on CLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the CLKR pin.

CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge (see Figure 12–59).

Figure 12–61 shows how data clocked by an external serial device using a rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.

Figure 12–61. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



12.8.20 Set the SRG Clock Divide-Down Value

Table 12–68. Register Bits Used to Set Sample Rate Generator (SRG) Clock Divide-Down Value

Register	Bit	Name	Function	Type	Reset Value
SRGR1	7:0	CLKGDV	Sample rate generator clock divide-down value The input clock of the sample rate generator is divided by (CLKGDV + 1) to generate the required sample rate generator clock frequency. The default value of CLKGDV is 1 (divide input clock by 2).	R/W	0000 0001

12.8.20.1 Sample Rate Generator Clock Divider

The first divider stage generates the serial data bit clock from the input clock. This divider stage uses a counter, preloaded by CLKGDV, that contains the divide ratio value.

The output of the first divider stage is the data bit clock, which is output as CLKG and which serves as the input for the second and third stages of the divider.

CLKG has a frequency equal to 1/(CLKGDV + 1) of the sample-rate generator input clock. Thus, the sample-generator input clock frequency is divided by a value between 1 and 256. When CLKGDV is odd or equal to 0, the CLKG duty cycle is 50%. When CLKGDV is an even value, 2p, representing an odd divide-down, the high-state duration is p+1 cycles and the low-state duration is p cycles.

12.8.21 Set the SRG Clock Synchronization Mode

Table 12–69. Register Bit Used to Set the SRG Clock Synchronization Mode

Register	Bit	Name	Function	Type	Reset Value
SRGR2	15	GSYNC	Sample-rate generator clock synchronization GSYNC is used only when the input clock source for the sample rate generator is external—on the CLKS, CLKR, or CLKX pin. GSYNC = 0: The sample-rate generator clock (CLKG) is free-running. CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles. GSYNC = 1: Clock synchronization is performed. When a pulse is detected on the FSR pin: ❑ CLKG is adjusted as necessary so that it is synchronized with the input clock on the CLKS, CLKR, or CLKX pin. ❑ FSG pulses. FSG only pulses in response to a pulse on the FSR pin. The frame-synchronization period defined in FPER is ignored.	R/W	0

For more details on using the clock synchronization feature, see Section 12.3.3, *Synchronizing Sample Rate Generator Outputs to an External Clock*.

12.8.22 Set the SRG Clock Mode (Choose an Input Clock)

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both, but CLKG is derived from an input clock. Table 12–70 shows the four possible sources of the input clock. For more details on generating CLKG, see Section 12.3.1, *Clock Generation in the Sample Rate Generator*.

Table 12–70. Register Bits Used to Set the SRG Clock Mode (Choose an Input Clock)

Register	Bit	Name	Function		Type	Reset Value
PCR	7	SCLKME	Sample-rate generator clock mode		R/W	0
SRGR2	13	CLKSM			R/W	1
			SCLKME = 0 CLKSM = 0	Sample-rate generator clock derived from CLKS pin		
			SCLKME = 0 CLKSM = 1	Sample-rate generator clock derived from CPU clock (This is the condition forced by an OMAP730 reset.)		
			SCLKME = 1 CLKSM = 0	Sample-rate generator clock derived from CLKR pin		
			SCLKME = 1 CLKSM = 1	Sample-rate generator clock derived from CLKX pin		

12.8.23 Set the SRG Input Clock Polarity

Table 12–71. Register Bits Used to Set the SRG Input Clock Polarity

Register	Bit	Name	Function		Type	Reset Value
SRGR2	14	CLKSP	CLKS pin polarity		R/W	0
			CLKSP determines the input clock polarity when the CLKS pin supplies the input clock (SCLKME = 0 and CLKSM = 0).			
			CLKSP = 0	Rising edge on CLKS pin generates CLKG and FSG.		
			CLKSP = 1	Falling edge on CLKS pin generates CLKG and FSG.		
PCR	1	CLKXP	CLKX pin polarity		R/W	0
			CLKXP determines the input clock polarity when the CLKX pin supplies the input clock (SCLKME = 1 and CLKSM = 1).			
			CLKXP = 0	Rising edge on CLKX pin generates transitions on CLKG and FSG.		
			CLKXP = 1	Falling edge on CLKX pin generates transitions on CLKG and FSG.		
PCR	0	CLKRP	CLKR pin polarity		R/W	0
			CLKRP determines the input clock polarity when the CLKR pin supplies the input clock (SCLKME = 1 and CLKSM = 0).			
			CLKRP = 0	Falling edge on CLKR pin generates transitions on CLKG and FSG.		
			CLKRP = 1	Rising edge on CLKR pin generates transitions on CLKG and FSG.		

12.8.23.1 Using CLKSP/CLKXP/CLKRP to Choose an Input Clock Polarity

The sample rate generator can produce a clock signal (CLKG) and a frame-synchronization signal (FSG) for use by the receiver, the transmitter, or both. To produce CLKG and FSG, the sample rate generator must be driven by an input clock signal derived from the CPU clock or from an external clock on the CLKS, CLKX, or CLKR pin. If you use a pin, choose a polarity for that pin by using the appropriate polarity bit (CLKSP for the CLKS pin, CLKXP for the CLKX pin, CLKRP for the CLKR pin). The polarity determines whether the rising or falling edge of the input clock generates transitions on CLKG and FSG.

12.9 General-Purpose I/O on the McBSP Pins

Table 12–72 summarizes how to use the McBSP pins as general-purpose I/O pins. All of the bits mentioned in the table except $\overline{\text{XRST}}$ and $\overline{\text{RRST}}$ are in the pin control register (PCR). $\overline{\text{XRST}}$ and $\overline{\text{RRST}}$ are in the serial-port control registers (SPCRs).

To use receiver pins CLKR, FSR, and DR as general-purpose I/O pins rather than as serial port pins, you must set two conditions:

- The receiver of the serial port is in reset ($\overline{\text{RRST}}$ = 0 in SPCR1).
- General-purpose I/O is enabled for the serial port receiver (RIOEN = 1 in PCR).

The CLKR and FSR pins can be individually configured as either input or output pins with the CLKRM and FSRM bits, respectively. The DR pin can only be an input pin. Table 12–72 shows which bits in PCR are used to read from/write to these pins.

For the transmitter pins CLKX, FSX, and DX, you must meet two conditions:

- The transmitter of the serial port is in reset ($\overline{\text{XRST}}$ = 0 in SPCR2).
- General-purpose I/O is enabled for the serial port transmitter (XIOEN = 1 in PCR).

The CLKX and FSX pins can be individually configured as input or output pins with the CLKXM and FSXM bits, respectively. The DX pin can only be an output pin. Table 12–72 shows which bits in PCR are used to read from/write to these pins.

For the CLKS pin, the reset and I/O enable conditions must be met:

- Both the receiver and transmitter of the serial port are in reset ($\overline{\text{RRST}}$ = 0 and $\overline{\text{XRST}}$ = 0).
- General-purpose I/O is enabled for both the receiver and the transmitter (RIOEN = 1 and XIOEN = 1).

The CLKS pin can only be an input pin. To read the status of the signal on the CLKS pin, read the CLKS_STAT bit in PCR.

Table 12–72. Using McBSP Pins for General-Purpose Input/Output

Pin	General-Purpose Use Enabled by This Bit Combination	Selected as Output When …	Output Value Driven From This Bit	Selected as Input When …	Input Value Read From This Bit
CLKRX	$\overline{\text{XRST}}$ = 0 XIOEN = 1	CLKXM = 1	CLKXP	CLKXM = 0	CLKXP
FSRX	$\overline{\text{XRST}}$ = 0 XIOEN = 1	FSXM = 1	FSXP	FSXM = 0	FSXP
DX	$\overline{\text{XRST}}$ = 0 XIOEN = 1	Always	DX_STAT	Never	Does not apply
CLKRX	$\overline{\text{RRST}}$ = 0 RIOEN = 1	CLKRM = 1	CLKRP	CLKRM = 0	CLKRP
FSRX	$\overline{\text{RRST}}$ = 0 RIOEN = 1	FSRM = 1	FSRP	FSRM = 0	FSRP
DR	$\overline{\text{RRST}}$ = 0 RIOEN = 1	Never	Does not apply	Always	DR_STAT
CLKS	$\overline{\text{RRST}}$ = $\overline{\text{XRST}}$ = 0 RIOEN = XIOEN = 1	Never	Does not apply	Always	CLKS_STAT

12.10 Emulation, Power, and Reset Considerations

This section covers the following topics:

- How to program McBSP response to a breakpoint in the high-level language debugger (see Section 12.10.1)
- How to conserve power in the OMAP730 by placing the McBSP into its idle mode (section 12.10.2)
- How to reset and initialize the various parts of the McBSP (see Section 12.10.3)

12.10.1 McBSP Emulation Mode

FREE and SOFT are special emulation bits in SPCR2 that determine the state of the McBSP when a breakpoint is encountered in the high-level language debugger. If FREE = 1, the clock continues to run on a software breakpoint and data is still shifted out. When FREE = 1, the SOFT bit is a don't care.

If FREE = 0, the SOFT bit takes effect. If SOFT = 0 when a breakpoint occurs, the clock stops immediately, aborting a transmission. If SOFT = 1 and a breakpoint occurs while transmission is in progress, the transmission continues until completion of the transfer and then the clock halts. These options are listed in Table 12–73.

The McBSP receiver functions in a similar fashion. If a mode other than the immediate stop mode (SOFT = FREE = 0) is chosen, the receiver continues to run and an overrun error is possible.

Table 12–73. McBSP Emulation Modes Selectable with FREE and SOFT Bits of SPCR2

FREE	SOFT	McBSP Emulation Mode
0	0	Immediate stop mode (reset condition) The transmitter or receiver stops immediately in response to a breakpoint.
0	1	Soft stop mode When a breakpoint occurs, the transmitter stops after completion of the current word. The receiver is not affected.
1	0 or 1	Free run mode The transmitter and receiver continue to run when a breakpoint occurs.

12.10.2 Reducing Power Consumed by McBSPs

The McBSP is placed into idle mode with reduced power consumption when the PERIPH idle domain is idle (PERIS = 1 in ISTR) and the McBSP idle enable bit is set (IDLE_EN = 1 in PCR).

In the McBSP idle mode:

- If the McBSP is configured to operate with internally generated clocking and frame synchronization, it is completely stopped.

- If the McBSP is configured to operate with externally generated clocking and frame synchronization (either directly or through the sample rate generator), the external interface portion of the McBSP continues to function during external clock activity periods. The McBSP sends a request to activate the PERIPH and DMA idle domains when it needs to be serviced. If the domains were idle, they are made idle again after the McBSP has been serviced.

When IDLE_EN = 0 in PCR, the McBSP keeps running, regardless of whether the PERIPH domain is idle.

12.10.3 Resetting and Initializing McBSPs

12.10.3.1 McBSP Pin States: OMAP730 Reset Versus Receiver/Transmitter Reset

Table 12–74 shows the state of McBSP pins when the serial port is reset due to direct receiver or transmitter reset on the OMAP730 device.

Table 12–74. Reset State of Each McBSP Pin

Pin	Possible State(s)	State Forced by OMAP730 Reset	State Forced by Receiver/Transmitter Reset
Receiver reset ($\overline{\text{RRST}}$ = 0 and $\overline{\text{GRST}}$ = 1)			
DR	I	Input	Input
CLKRX	I/O/Z	Input	Known state if input; CLKR running if output
FSRX	I/O/Z	Input	Known state if input; FSRP inactive state if output
CLKS	I/O/Z	Input	Input
Transmitter reset ($\overline{\text{XRST}}$ = 0 and $\overline{\text{GRST}}$ = 1)			
DX	O/Z	High impedance	High impedance
CLKRX	I/O/Z	Input	Known state if input; CLKX running if output
FSRX	I/O/Z	Input	Known state if input; FSXP inactive state if output

Note: In Possible State(s) column, I = input, O = output, Z = high impedance

12.10.3.2 OMAP730 Reset, McBSP Reset, and Sample Rate Generator Reset

When the McBSP is reset in either of the above two ways, the machine is reset to its initial state, including reset of all counters and status bits. The receive status bits include RFULL, RRDY, and RSYNCERR. The transmit status bits include XEMPTY_, XRDY, and XSYNCERR.

- OMAP730 reset. When the whole DSP is reset ($\overline{\text{RESET}}$ signal is driven low), the entire serial port, including the transmitter, receiver, and the sample rate generator, is reset. All input-only pins and 3-state pins must be in a known state. The output-only pin DX is in high-impedance state.

  The OMAP730 reset forces the sample-rate generator clock, CLKG, to behalf the frequency of the CPU clock. No pulses are generated on the sample-rate generator frame-synchronization signal, FSG.

  When the device is pulled out of reset, the serial port remains in reset state. In this state the DR and DX pins can be used as general-purpose I/O pins, as described in Section 12.9, *General-Purpose I/O on the McBSP Pins*.

- McBSP reset. When the receiver and transmitter reset bits, $\overline{\text{RRST}}$ and $\overline{\text{XRST}}$, are loaded with zeros, the respective portions of the McBSP are reset and activity in the corresponding section of the serial port stops. All input-only pins such as DR and CLKS, and all other pins that are configured as inputs, are in a known state. The FSR and FSX pins are driven to their inactive state if they are not outputs. If the CLKR and CLKX pins are programmed as outputs, they are driven by CLKG, provided that $\overline{\text{GRST}}$ = 1. Lastly, the DX pin is in high-impedance state when the transmitter and/or the device is reset.

  During usual operation, the sample rate generator is reset if the $\overline{\text{GRST}}$ bit is cleared. $\overline{\text{GRST}}$ must be 0 only when neither the transmitter nor the receiver is using the sample rate generator. In this case, the internal sample-rate generator clock (CLKG) and its frame-synchronization signal (FSG) are driven inactive low.

  When the sample rate generator is not in the reset state ($\overline{\text{GRST}}$ = 1), pins FSR and FSX are in an inactive state when $\overline{\text{RRST}}$ = 0 and $\overline{\text{XRST}}$ = 0, respectively, even if they are outputs driven by FSG. This ensures that when only one portion of the McBSP is in reset, the other portion can continue operation when $\overline{\text{GRST}}$ = 1 and its frame synchronization is driven by FSG.

- Sample-rate generator reset. The sample rate generator is reset when the DSP is reset or when $\overline{\text{GRST}}$ is loaded with 0. In the case of an OMAP730 reset, the sample-rate generator clock, CLKG, is driven by the CPU clock divided by 2 and the frame-synchronization signal, FSG, is driven inactive low.

  When neither the transmitter nor the receiver is fed by CLKG and FSG, you can reset the sample rate generator by clearing $\overline{\text{GRST}}$. In this case, CLKG and FSG are driven inactive low. If you then set $\overline{\text{GRST}}$, CLKG starts and runs as programmed. Later, if $\overline{\text{GRST}}$ = 1, FSG pulses active high after the programmed number of CLKG cycles has elapsed.

12.10.3.3 McBSP Initialization Procedure

The serial port initialization procedure is as follows:

1) Make $\overline{\text{XRST}}$ = $\overline{\text{RRST}}$ = $\overline{\text{GRST}}$ = 0 in SPCR[1,2]. If coming out of an OMAP730 reset, this step is not required.

2) While the serial port is in the reset state, program only the McBSP configuration registers (not the data registers) as required.

3) Wait for two clock cycles. This ensures proper internal synchronization.

4) Set up data acquisition as required (such as writing to DXR[1,2]).

5) Make $\overline{\text{XRST}}$ = $\overline{\text{RRST}}$ = 1 to enable the serial port. Ensure that as you set these reset bits, you do not modify any of the other bits in SPCR1 and SPCR2. Otherwise, you change the configuration you selected in step 2.

6) Set $\overline{\text{FRST}}$ = 1, if internally generated frame synchronization is required.

7) Wait two clock cycles for the receiver and transmitter to become active.

Alternatively, on either write (step 1 or 5), the transmitter and receiver can be placed in or taken out of reset individually by modifying the desired bit.

The above procedure for reset/initialization can be applied in general when the receiver or transmitter must be reset during its usual operation and when the sample rate generator is not used for either operation.

Notes:

- The necessary duration of the active-low period of $\overline{\text{XRST}}$ or $\overline{\text{RRST}}$ is at least two CLKR/CLKX cycles.
- The appropriate bits in serial port configuration registers SPCR[1,2], PCR, RCR[1,2], XCR[1,2], and SRGR[1,2] must only be modified when the affected portion of the serial port is in its reset state.
- In most cases, the data transmit registers (DXR[1,2]) must be loaded by the CPU or by the DMA controller only when the transmitter is enabled ($\overline{\text{XRST}}$ = 1). An exception to this rule is when these registers are used for companding internal data (see Section 12.2.2.2, *Capability to Compand Internal Data*).
- The bits of the channel control registers—MCR[1,2], RCER[A-H], XCER[A-H]—can be modified at any time as long as they are not being used by the current reception/transmission in a multichannel selection mode.

12.10.3.4 Resetting the Transmitter While the Receiver is Running

Example 12–1 shows values in the control registers that reset and configure the transmitter while the receiver is running.

Example 12–1. Resetting and Configuring McBSP Transmitter While McBSP Receiver Running

```
SPCR1 = 0001h      ; The receiver is running with the receive
SPCR2 = 0030h      ; interrupt (RINT) triggered by the
                   ; receiver ready bit (RRDY). The
                   ; transmitter is in its reset state. The
                   ; transmit interrupt (XINT) is
                   ; triggered by the transmit frame-sync
                   ; error bit (XSYNCERR).

PCR = 0900h        ; Transmit frame synchronization is
                   ; generated internally according to the
                   ; FSGM bit of SRGR2. The transmit clock
                   ; is driven by an external source. The
                   ; receive clock continues to be driven by
                   ; sample rate generator. The input clock
                   ; of the sample rate generator is supplied
                   ; by the CLKS pin or by the CPU clock
                   ; depending on the CLKSM bit of SRGR2.

SRGR1 = 0001h      ; The CPU clock is the input clock for
SRGR2 = 2000h      ; the sample rate generator. The sample
                   ; rate generator divides the CPU clock by
                   ; 2 to generate its output clock (CLKG).
                   ; Transmit frame synchronization is tied
                   ; to the automatic copying of data from
                   ; the DXR(s) to the XSR(s).

XCR1 = 0740h       ; The transmit frame has two phases.
XCR2 = 8321h       ; Phase 1 has eight 16-bit words. Phase 2
                   ; has four 12-bit words. There is 1-bit
                   ; data delay between the start of a
                   ; frame-sync pulse and the first data bit
                   ; transmitted.

SPCR2 = 0x0031     ; The transmitter is taken out of reset.
```

12.11 Data Packing Examples

This section shows two ways to implement data packing in the McBSP.

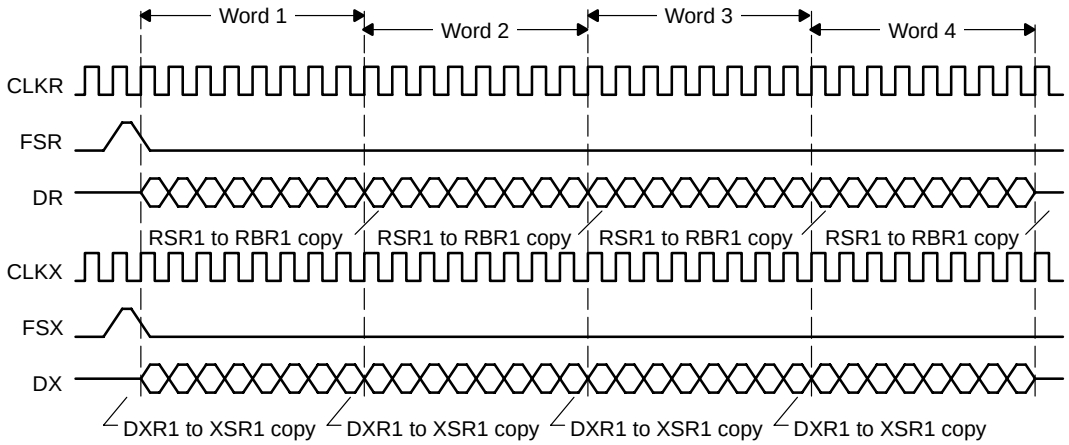
12.11.1 Data Packing Using Frame Length and Word Length

Frame length and word length can be manipulated to effectively pack data. For example, consider a situation where four 8-bit words are transferred in a single-phase frame, as shown in Figure 12–62. In this case:

- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLEN1 = 0000011b: 4-word frame
- (R/X)WDLEN1 = 000b: 8-bit words

Four 8-bit data words are transferred to and from the McBSP by the CPU or by the DMA controller. Thus, four reads from DRR1 and four writes to DXR1 are necessary for each frame.

Figure 12–62. Four 8-Bit Data Words Transferred To/From the McBSP



This data can also be treated as a single-phase frame consisting of one 32-bit data word, as shown in Figure 12–63. In this case:

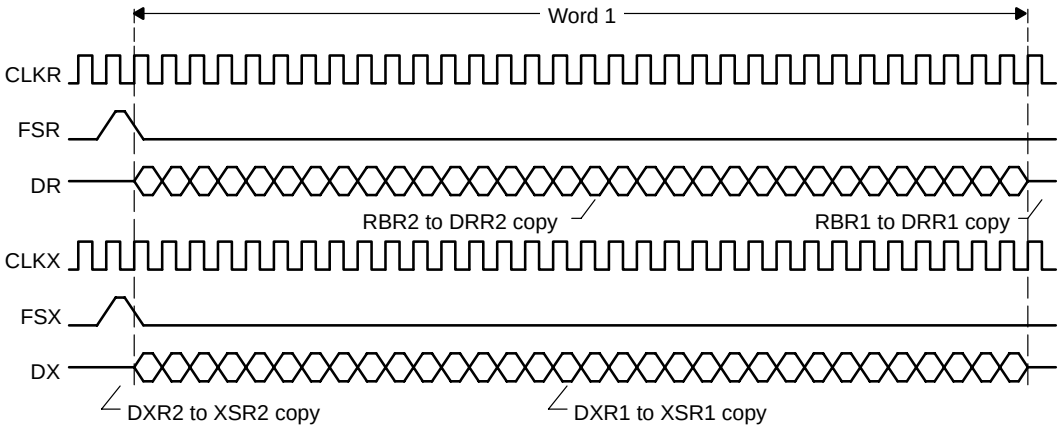
- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLEN1 = 0000000b: 1-word frame
- (R/X)WDLEN1 = 101b: 32-bit word

Two 16-bit data words are transferred to and from the McBSP by the CPU or DMA controller. Thus, two reads, from DRR2 and DRR1, and two writes, to DXR2 and DXR1, are necessary for each frame. This results in only half the number of transfers compared to the previous case. This manipulation reduces the percentage of bus time required for serial port data movement.

Note:

When the word length is larger than 16 bits, ensure that you access DRR2/DXR2 before you access DRR1/DXR1. McBSP activity is tied to accesses of DRR1/DXR1. During the reception of 24-bit or 32-bit words, read DRR2 and then read DRR1. Otherwise, the next RBR[1,2]-to-DRR[1,2] copy occurs before DRR2 is read. Similarly, during the transmission of 24-bit or 32-bit words, write to DXR2 and then write to DXR1. Otherwise, the next DXR[1,2]-to-XSR[1,2] copy occurs before DXR2 is loaded with new data.

Figure 12–63. One 32-Bit Data Word Transferred To/From the McBSP



12.11.2 Data Packing Using Word Length and the Frame-Synchronization Ignore Function

When there are multiple words per frame, you can implement data packing by increasing the word length (defining a serial word with more bits) and by ignoring frame-synchronization pulses. First, consider Figure 12–64, which shows the McBSP operating at the maximum packet frequency. Here, each frame has only a single 8-bit word. Notice the frame-synchronization pulse that initiates each frame transfer for reception and for transmission. For reception, this configuration requires one read operation for each word. For transmission, this configuration requires one write operation for each word.

Figure 12–64. 8-Bit Data Words Transferred at Maximum Packet Frequency

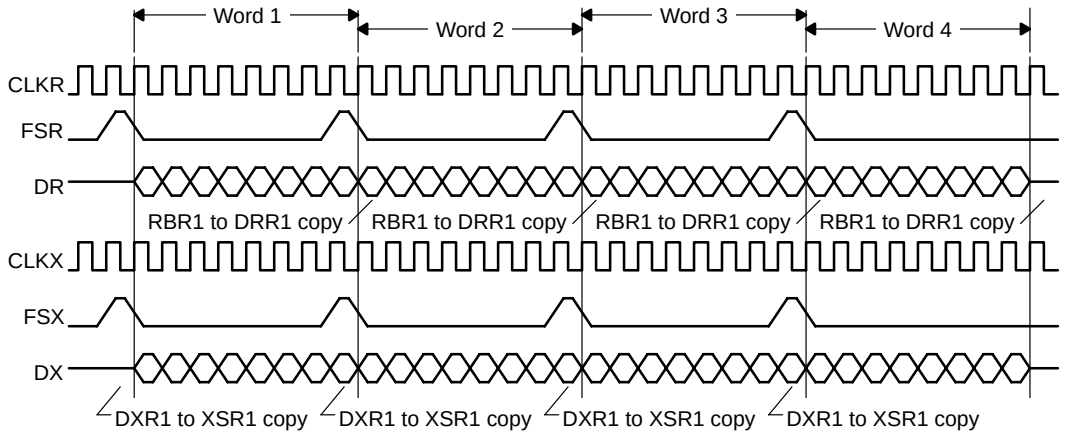
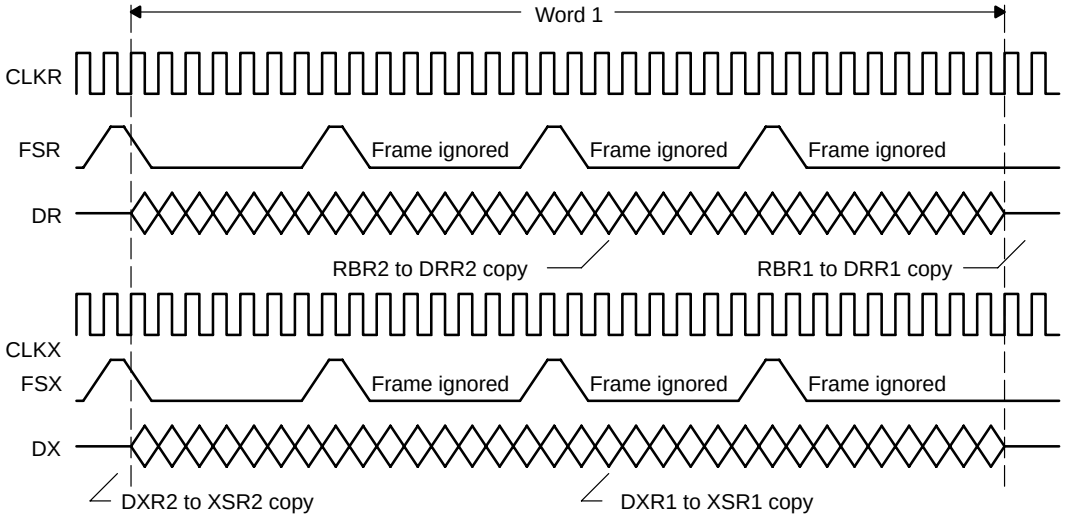


Figure 12–65 shows the McBSP configured to treat this data stream as a continuous 32-bit word. In this example, the McBSP responds to an initial frame-synchronization pulse. However, (R/X)FIG = 1 so that the McBSP ignores subsequent pulses. Only two read transfers or two write transfers are needed every 32 bits. This configuration effectively reduces the required bus bandwidth to half the bandwidth needed to transfer four 8-bit words.

Figure 12–65. Configuring the Data Stream of Figure 12–64 as a Continuous 32-Bit Word



12.12 McBSP on the OMAP730 Device—Applications

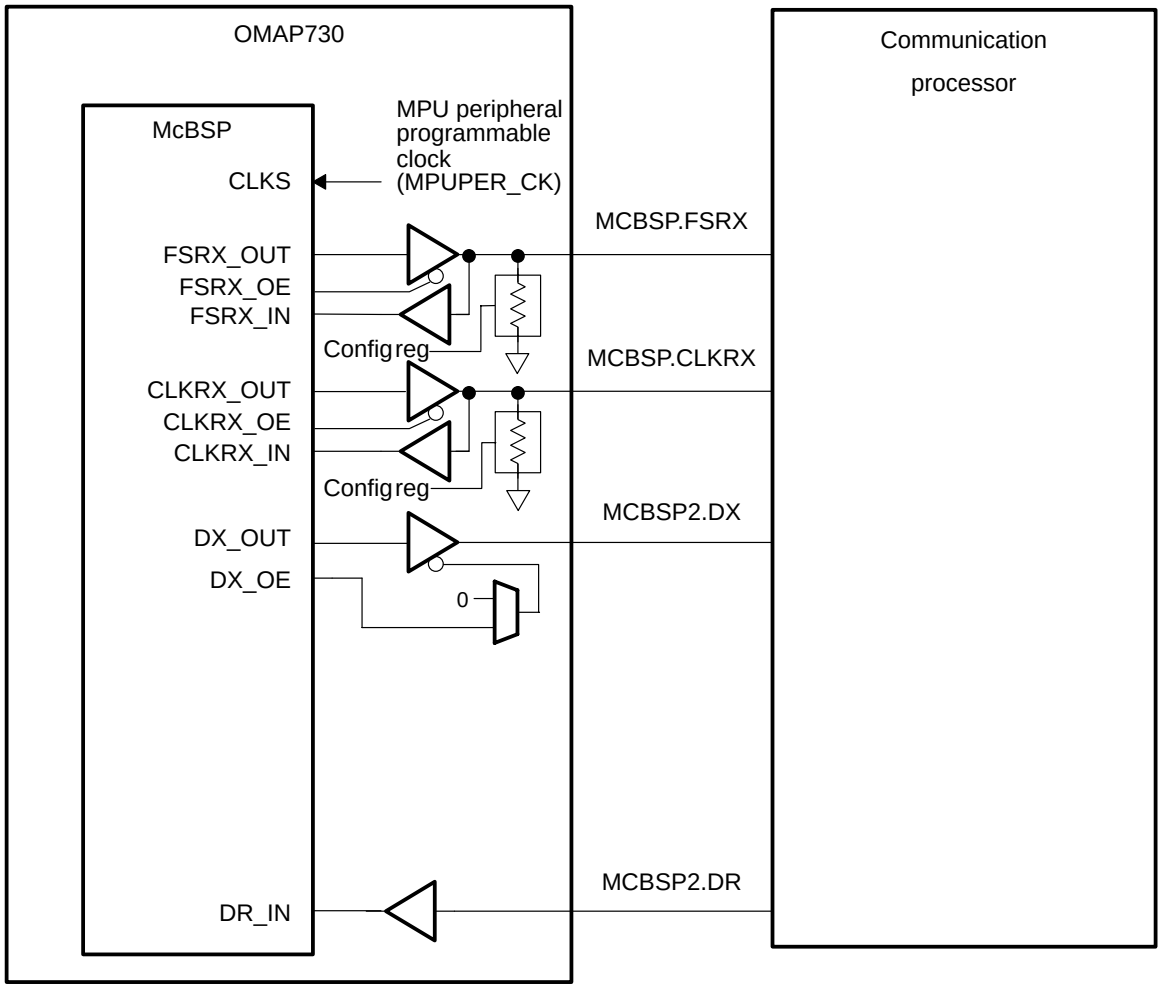
There are two McBSPs on the OMAP730 device.

This section provides configuration examples for common McBSP applications.

12.12.1 Communication McBSP Interface

Figure 12–66 illustrates the use of McBSP as a communication processor data interface that is the master of TX and slave for RX communications. The actual implementation is generic: FSX, CLKX, FSR, and CLKR are bidirectional. The direction of these signals is configured by registers in the McBSP module. The CLKS signal is the active input clock for the McBSP modem block. The active input clock can be changed in a McBSP register, but register activity on CLKS is required to perform the set up and write to the McBSP.

Figure 12–66. Communication Processor Data Interface



Section 12.12.1.1 through Section 12.12.1.9 explain how to set up the McBSP registers for TX master and RX slave mode with 16-bit transfers using interrupts.

12.12.1.1 Serial Port Control Register Configuration

1) ARM_Write(0x0000) => SPCR1; set up SPCR1 as initial configuration. This setup is not needed after reset.
2) ARM_Write(0x0000) => SPCR2; set up SPCR2 as initial configuration. This setup is not needed after reset.

12.12.1.2 Pin Control Register Configuration

ARM_Write(0x0a00) => PCR; set up PCR per Table 12–75.

Table 12–75. Pin Control Register Bit Description

Bit	Configuration Value	Description
15:14	00b	Reserved
13	0b	Set serial port mode for DX, FSX, and CLKX pins
12	0b	Set serial port mode for DR, FSR, and CLKR pins
11	1b	TX frame-synchronization signal driven by internal generator
10	0b	RX frame-synchronization signal derived by external source
9	1b	CLKX set output pin and driven by internal generator
8	0b	CLKR set input pin and derived by external source
7	0b	Sample-rate generator input-clock mode bit
6	0b	CLKS pin status (no meaning in the OMAP730 device)
5	0b	DX pin status
4	0b	DR pin status
3	0b	Set FSX polarity as active high
2	0b	Set FSR polarity as active high
1	0b	Set CLKX polarity as data driven on rising edge
0	0b	Set CLKR polarity as data sampled on falling edge

12.12.1.3 Receive Control Register Configuration

ARM_Write(0x0040) => RCR1; set up RCR1 per Table 12–76.

Table 12–76. Receive Control Register1 Bit Description (RCR1)

Bit	Configuration Value	Description
15	0b	Reserved
14:8	000 0000b	Set receive frame length as one word per frame
7:5	010b	Set receive word length as 16 bits per frame
4:0	0 0000b	Reserved

ARM_Write(0x0001) = > RCR2; set up RCR2 per Table 12–77.

Table 12–77. Receive Control Register2 Bit Description (RCR2)

Bit	Configuration Value	Description
15	0b	Set single-phase frame
14:8	000 0000b	Don't care for single-phase frame
7:5	000b	Don't care for single-phase frame
4:3	00b	Set no companding data and transfer start with MSB first
2	0b	Set FSR not ignore after the first resets the transfer
1:0	01b	Set data delay as 1 bit

12.12.1.4 Transmit Control Register Configuration

ARM_Write(0x0040) => XCR1; set up XCR1 per Table 12–78.

Table 12–78. Transmit Control Register1 Bit Description (XCR1)

Bit	Config Value	Description
15	0b	Reserved
14:8	000 0000b	Set transmit frame length as one word per frame
7:5	010b	Set transmit word length as 16s bit per frame
4:0	0 0000b	Reserved

ARM_Write(0x0001) => XCR2; set up XCR2 per Table 12–79.

Table 12–79. Transmit Control Register2 Bit Description (XCR2)

Bit	Configuration Value	Description
15	0b	Set single-phase frame
14:8	000 0000b	Don't care for single-phase frame
7:5	000b	Don't care for single-phase frame
4:3	00b	Set no companding data and transfer start with MSB first
2	0b	Set FSX not ignore after the first resets the transfer
1:0	01b	Set data delay as 1 bit

12.12.1.5 Sample Rate Generator Configuration

To configure the sample rate generator appropriately for CLKX and FSX:

1) Wait two CLKSRG clocks.
2) ARM_Write SPCR2 or (0x0000 0040)=>SPCR2; CLKG enable
3) Wait two CLKG clocks.

For details, see *TMS320C54x DSP Enhanced Peripherals Reference Set, vol. 5*, SPRA302.

12.12.1.6 Interrupt Flag Configuration and Clear (ILR, ITR, MIR)

To clear interrupt flag configuration:

1) ARM_Write => ILR; set ILR appropriately for the interrupt handling priority.
2) ARM_Write ITR and (0xFFFF FFCF)=> ITR; clear remaining TX and RX interrupt.

Note:

This setup is not needed after reset.

3) ARM_Write MIR and (0xFFFF FFCF) => MIR; enable SPI TX and RX interrupt

12.12.1.7 Take out of Reset for Transmit and Receive Starting (SPCR[1,2])

To enable transmit and receive:

1) ARM_write SPCR1 or (0x0001) => SPCR1; enabled receive port
2) ARM_write SPCR2 or (0x0001) => SPCR2; enabled transmit port

12.12.1.8 Transmit Data Loading (TX_INT Handling in Interrupt Survive Routine)

ARM_Write => DXR

Note:

Clear interrupts flag in ITR, when the interrupt handle is taken.

12.12.1.9 Received Data Loading (RX_INT Handling in Interrupt Survive Routine)

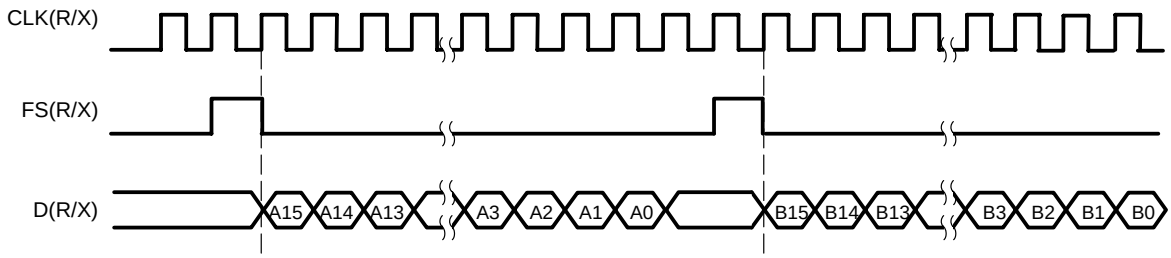
ARM_Read <= DRR

Note:

Clear interrupts flag in ITR, when the interrupt handle is taken.

Waveform Example

Figure 12–67. Waveform Example



Section 12.12.1.10 explains how to set up the McBSP registers for TX master and RX slave mode with 16-bit transfers using DMA support.

12.12.1.10 Serial Port Control Register Configuration

1) ARM_Write(0x0000) => SPCR1; set up SPCR1 as initial configuration.

   This setup is not needed after reset.

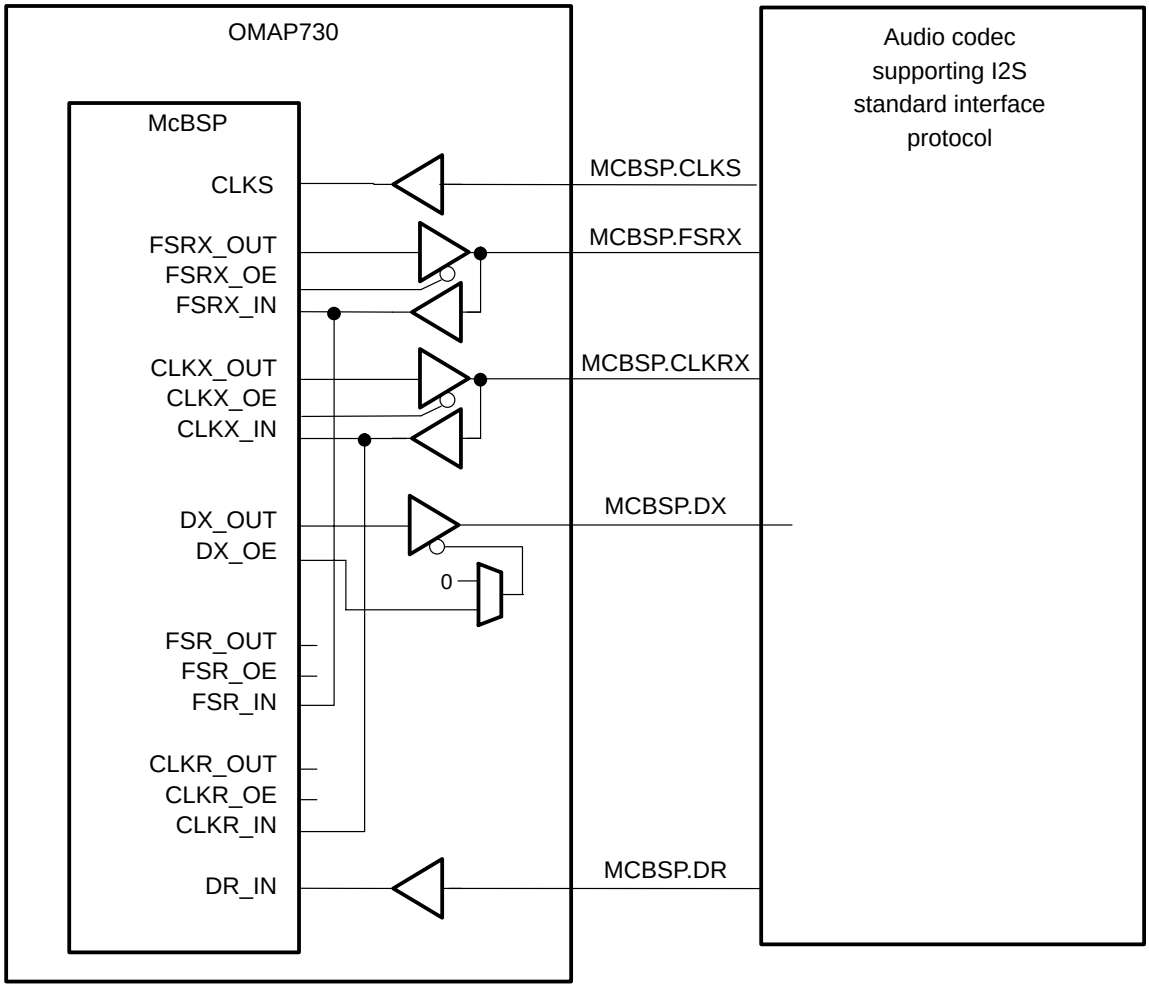
2) ARM_Write(0x0000) => SPCR2; set up SPCR2 as initial configuration.

   This setup is not needed after reset.

12.12.2 I2S Audio Codec McBSP Interface

This application uses the McBSP as an I2S audio codec interface (see Figure 12–68). The OMAP730 is intended to be either the master or slave device; that is, it either receives or provides the frame synchronization and bit clock.

Figure 12–68. I2S Audio Codec Interface



Section 12.12.2.1 through Section 12.12.2.9 explain how to set up the McBSP registers for I2S slave mode with 16-bit transfers using DMA support.

12.12.2.1 Serial Port Control Register Configuration

1) DSP_Write(0x0000) => SPCR1; set up SPCR1 as initial configuration. This setup is not needed after reset.
2) DSP_Write(0x0000) => SPCR2; set up SPCR2 as initial configuration. This set up is not needed after reset.

12.12.2.2 Pin Control Register Configuration

DSP_Write(0x0000) => PCR; set up PCR per Table 12–80.

Table 12–80. Pin Control Register Bit Description (PCR)

Bit	Config Value	Description
15:14	00b	Reserved
13	0b	Set serial port mode for DX, FSX and CLKX pins
12	0b	Set serial port mode for DR, FSR and CLKR pins
11	0b	TX frame-synchronization signal derived by external source
10	0b	RX frame-synchronization signal derived by external source
9	0b	CLKX set input pin and derived by external source
8	0b	CLKR set input pin and derived by external source
7	0b	Sample rate generator input clock mode bit
6	0b	CLKS pin status (no meaning in the OMAP730 device)
5	0b	DX pin status
4	0b	DR pin status
3	0b	Set FSX polarity as active high
2	0b	Set FSR polarity as active high
1	0b	Set CLKX polarity as data driven on rising edge
0	0b	Set CLKR polarity as data sampled on falling edge

12.12.2.3 Receive Control Register Configuration

DSP_Write(0x00a0) => RCR1; set up RCR1 per Table 12–81.

Table 12–81. Receive Control Register 1 Bit Description

Bit	Config Value	Description
15	0b	Reserved
14:8	000 0000b	Set receive frame length as one word per frame
7:5	101b	Set receive word length as 32 bits per frame
4:0	0 0000b	Reserved

DSP_Write(0x80a1) => RCR2; set up RCR2 per Table 12–82.

Table 12–82. Receive Control Register 2 Bit Description

Bit	Config Value	Description
15	1b	Set dual-phase frame
14:8	000 0000b	Set receive frame length as one word per frame
7:5	101b	Set receive word length as 32 bits per frame
4:3	00b	Don’t care for single-phase frame
2	0b	Set FSR not ignore after the first resets the transfer
1:0	01b	Set data delay as 1 bit

12.12.2.4 Transmit Control Register Configuration

DSP_Write(0x00a0) => XCR1; set up XCR1 per Table 12–83.

Table 12–83. Transmit Control Register 1 Bit Description (XCR1)

Bit	Config Value	Description
15	0b	Reserved
14:8	000 0000b	Set transmit frame length as one word per frame
7:5	101b	Set receive word length as 32 bits per frame
4:0	0 0000b	Reserved

DSP_Write(0x80a1) => XCR2; set up XCR2 per Table 12–84.

Table 12–84. Transmit Control Register 2 Bit Description (XCR2)

Bit	Config Value	Description
15	1b	Set dual-phase frame
14:8	000 0000b	Don’t care for single-phase frame
7:5	101b	Set receive word length as 32 bits per frame
4:3	00b	Set no companding data and transfer start with MSB first
2	0b	Set FSX not ignore after the first resets the transfer
1:0	01b	Set data delay as 1 bit

12.12.2.5 Sample Rate Generator Configuration (SRGR[1,2])

It is not necessary to configure the sample rate generator, because external clocks and frame synchronizations are provided appropriately for CLKX and FSX.

12.12.2.6 DMA Configuration

It is necessary to configure the REVT and XEVT bit for the DMA receive and transmit synchronized invent.

12.12.2.7 Interrupt Flag Configuration and Clear (ILR, MIR)

1) DSP_Write => ILR; set ILR appropriately for the interrupt handling priority.
2) DSP_Write MIR and (0x0000 0030) => MIR; disabled SPI TX and RX interrupt

Note:

Enable the appropriate DMA channel interrupts.

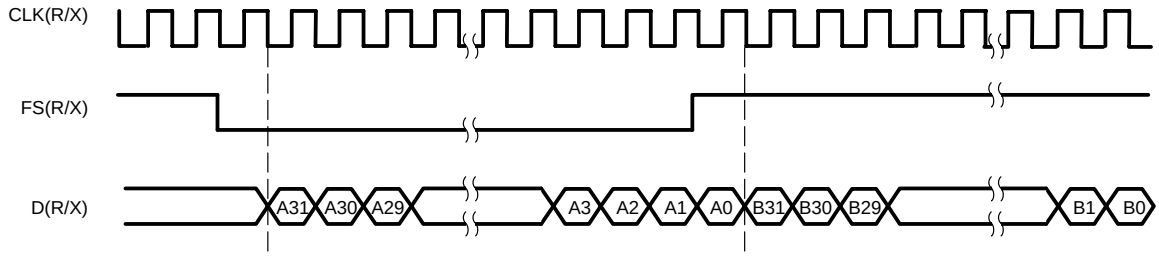
12.12.2.8 Take out of Reset for Transmit and Receive Starting (SPCR[1,2])

1) DSP_write SPCR1 or (0x0001) => SPCR1; enable receive port
2) DSP_write SPCR2 or (0x0001) => SPCR2; enable transmit port

12.12.2.9 Data Transfer (DMA Channel)

The DMA channel transfers the received data to the appropriate data buffer and transfers the new transmit data to the appropriate TX buffer. Clear the interrupt flag on ITR when the interrupt handle is taken.

Figure 12–69. Waveform Example



12.13 McBSP Registers

The base address for each McBSP register map is as follows:

- McBSP1: FFFB:1000
- McBSP2: FFFB:1800

Table 12–85 shows the registers accessible by a user on each McBSP. Table 12–86 through Table 12–104 describe register bits.

Table 12–85. McBSP Registers

Name	Description	Offset
DRR2(15:0)	Data receive register 2	0x00
DRR1(15:0)	Data receive register 1	0x02
DXR2(15:0)	Data transmit register 2	0x04
DXR1(15:0)	Data transmit register 1	0x06
SPCR2(15:0)	Serial port control register 2	0x08
SPCR1(15:0)	Serial port control register 1	0x0A
RCR2(15:0)	Receive control register 2	0x0C
RCR1(15:0)	Receive control register 1	0x0E
XCR2 (15:0)	Transmit control register 2	0x10
XCR1(15:0)	Transmit control register 1	0x12
SRGR2(15:0)	Sample rate generator register 2	0x14
SRGR1(15:0)	Sample rate generator register 1	0x16
MCR2(15:0)	Multichannel register 2	0x18
MCR1(15:0)	Multichannel register 1	0x1A
RCERA(15:0)	Receive channel enable register partition A	0x1C
RCERB(15:0)	Receive channel enable register partition B	0x1E
XCERA(15:0)	Transmit channel enable register partition A	0x20
XCERB(15:0)	Transmit channel enable register partition B	0x22
PCR0(15:0)	Pin control register	0x24
RCERC(15:0)	Receive channel enable register partition C	0x26
RCERD(15:0)	Receive channel enable register partition D	0x28
XCERC(15:0)	Transmit channel enable register partition C	0x2A
XCERD(15:0)	Transmit channel enable register partition D	0x2C
RCERE(15:0)	Receive channel enable register partition E	0x2E
RCERF(15:0)	Receive channel enable register partition F	0x30
XCERE(15:0)	Transmit channel enable register partition E	0x32
XCERF(15:0)	Transmit channel enable register partition F	0x34
RCERG(15:0)	Receive channel enable register partition G	0x36
RCERH(15:0)	Receive channel enable register partition H	0x38
XCERG(15:0)	Transmit channel enable register partition G	0x3A
XCERH(15:0)	Transmit channel enable register partition H	0x3C

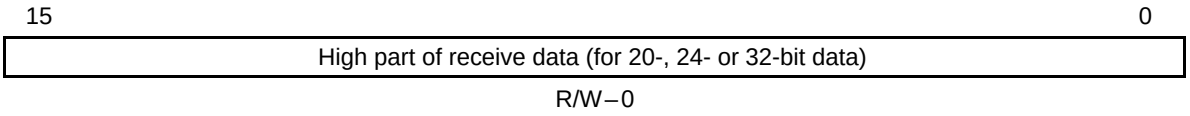
12.13.1 Data Receive Registers (DRR2 and DRR1)

The CPU or the DMA controller reads received data from one or both of the data receive registers (see Figure 12–70). If the serial word length is 16 bits or smaller, only DRR1 is used. If the serial length is larger than 16 bits, both DRR1 and DRR2 are used and DRR2 holds the most significant bits. Each frame of receive data in the McBSP can have one phase or two phases, each with its own serial word length.

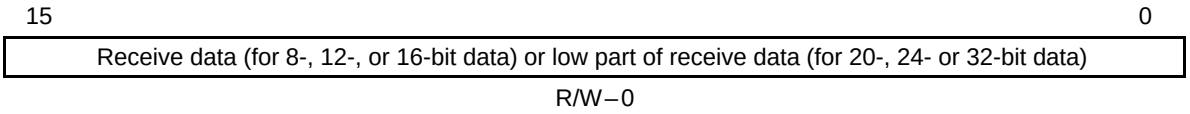
DRR1 and DRR2 are I/O mapped registers; they are accessible at addresses in I/O space.

Figure 12–70. Data Receive Registers (DRR2 and DRR1)

DRR2



DRR1



Legend: R = read; W = write; -*n* = value after reset

12.13.1.1 Data Travel From Data Receive Pins to the Registers

If the serial word length is 16 bits or smaller, receive data on the DR pin is shifted into receive shift register 1 (RSR1) and then copied into receive buffer register 1 (RBR1). The content of RBR1 is then copied to DRR1, which can be read by the CPU or by the DMA controller.

If the serial word length is larger than 16 bits, receive data on the DR pin is shifted into both of the receive shift registers (RSR2, RSR1) and then copied into both of the receive buffer registers (RBR2, RBR1). The content of the RBRs is then copied into both of the DRRs, which can be read by the CPU or by the DMA controller.

If companding is used during the copy from RBR1 to DRR1 (RCOMPAND = 10b or 11b), the 8-bit compressed data in RBR1 is expanded to a left-justified 16-bit value in DRR1. If companding is disabled, the data copied from RBR[1,2] to DRR[1,2] is justified and bit-filled according to the RJUST bits.

The RSRs and RBRs are not accessible. They are not mapped to I/O space as are the DRRs.

12.13.2 Data Transmit Registers (DXR2 and DXR1)

For transmission, the CPU or the DMA controller writes data to one or both of the data transmit registers (see Figure 12–71). If the serial word length is 16 bits or smaller, only DXR1 is used. If the word length is larger than 16 bits, both DXR1 and DXR2 are used and DXR2 holds the most significant bits. Each frame of transmit data in the McBSP can have one phase or two phases, each with its own serial word length.

DXR1 and DXR2 are I/O mapped registers; they are accessible at addresses in I/O space.

Figure 12–71. Data Transmit Registers (DXR2 and DXR1)

DXR2

15	0
High part of transmit data (for 20-, 24- or 32-bit data)	
R/W–0	

DXR1

15	0
Transmit data (for 8-, 12-, or 16-bit data) or low part of receive data (for 20-, 24- or 32-bit data)	
R/W–0	

Legend: R = read; W = write; -*n* = value after reset

12.13.2.1 Data Travel From Registers to Data Transmit (DX) Pins

If the serial word length is 16 bits or less, data written to DXR1 is copied to transmit shift register 1 (XSR1). From XSR1, the data is shifted onto the DX pin, one bit at a time.

If the serial word length is more than 16 bits, data written to DXR1 and DXR2 is copied to both transmit shift registers (XSR2, XSR1). From the XSRs, the data is shifted onto the DX pin, one bit at a time.

If companding is used during the transfer from DXR1 to XSR1 (XCOMPAND = 10b or 11b), the McBSP compresses the 16-bit data in DXR1 to 8-bit data in the μ-law or A-law format in XSR1. If companding is disabled, the McBSP passes data from the DXR(s) to the XSR(s) without modification.

The XSRs are not accessible. They are not mapped to I/O space as are the DXRs.

12.13.3 Serial Port Control Registers (SPCR1 and SPCR2)

Each McBSP has two serial port control registers. Table 12–86 and Table 12–88 describe the bits in SPCR1 and SPCR2, respectively. These I/O-mapped registers enable you to:

- Control various McBSP modes: digital loopback (DLB), sign-extension and justification for reception (RJUST), clock stop (CLKSTP), interrupt (RINTM and XINTM), and emulation (FREE and SOFT)
- Turn on and off the DX-pin delay enabler (DXENA)
- Check the status of receive and transmit operations (RSYNCERR, XSYNCERR, RFULL, XEMPTY, RRDY, and XRDY)
- Reset portions of the McBSP (RRST, XRST, FRST, and GRST)

Table 12–86. Serial Port Control 1 Register (SPCR1)

Bit	Name	Value	Description	Type	Reset Value
15	DLB		Digital-loopback mode bit. DLB disables or enables the digital loopback mode of the McBSP:	R/W	0
		0	Disabled Internal DR is supplied by the DR pin. Internal FSR and internal CLKR can be supplied by their respective pins or by the sample rate generator, depending on the mode bits FSRM and CLKRM.		
		1	Enabled Internal receive signals are supplied by internal transmit signals: DR connected to DX FSR connected to FSX CLKR connected to CLKX Internal DX is supplied by the DX pin. Internal FSX and internal CLKX are supplied by their respective pins or are generated internally, depending on the mode bits FSXM and CLKXM. This mode allows you to test serial port code with a single DSP. The McBSP transmitter directly supplies data, frame synchronization, and clocking to the McBSP receiver.		

Table 12–86. Serial Port Control 1 Register (SPCR1) (Continued)

Bit	Name	Value	Description	Type	Reset Value
14:13	RJUST		Receive sign-extension and justification mode bits. During reception, RJUST determines how data is justified and bit-filled before being passed to the data receive registers (DRR1, DRR2). RJUST is ignored if you enable a companding mode with the RCOMPAND bits. In a companding mode, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1. For more details about the effects of RJUST, see Section 12.7.13, *Set the Receive Sign-Extension and Justification Mode*.	R/W	00
		00	Right-justify the data and zero-fill the MSBs.		
		01	Right-justify the data and sign-extend the data into the MSBs.		
		10	Left-justify the data and zero fill the LSBs.		
		11	Reserved (do not use)		
12:11	CLKSTP		Clock stop mode bits. CLKSTP allows you to use the clock stop mode to support the SPI master-slave protocol. If you do not want to use the SPI protocol, you can clear CLKSTP to disable the clock stop mode. In the clock stop mode, the clock stops at the end of each data transfer. At the beginning of each data transfer, the clock starts immediately (CLKSTP = 10b) or after a half-cycle delay (CLKSTP = 11b). For more details, see Section 12.7.5, *Enable/ Disable the Clock Stop*.	R/W	00
		00/01	Clock stop mode is disabled.		
		10	Clock stop mode, without clock delay		
		11	Clock stop mode, with half-cycle clock delay		
10:8	Reserved		Reserved bits (not available for use). They are read-only bits and return 0s when read.		
7	DXENA		DX delay-enabler mode bit. DXENA controls the delay enabler for the DX pin. The enabler creates an extra delay for turn-on time (for the length of the delay.) See the data sheet for your TMS320C55x DSP for more information. For more details about the effects of DXENA, see Section 12.8.13, *Set the Transmit DXENA Mode*.	R/W	0

Table 12–86. Serial Port Control 1 Register (SPCR1) (Continued)

Bit	Name	Value	Description	Type	Reset Value
		0	DX delay enabler off		
		1	DX delay enabler on		
6	Reserved			R/W	0
5:4	RINTM		Receive interrupt mode bits. RINTM determines which event in the McBSP receiver generates a receive interrupt (RINT) request. If RINT is properly enabled inside the CPU, the CPU services the interrupt request; otherwise, the CPU ignores the request.	R/W	00
		00	The McBSP sends a receive interrupt (RINT) request to the CPU when the RRDY bit changes from 0 to 1, indicating that receive data is ready to be read (the content of RBR[1,2] has been copied to DRR[1,2]). Regardless of the value of RINTM, you can check RRDY to determine whether a word 1transfer is complete. The McBSP sends a RINT request to the CPU when 16 enabled bits have been received on the DR pin.		
		01	In the multichannel selection mode, the McBSP sends a RINT request to the CPU after every 16-channel block is received in a frame. Outside of the multichannel selection mode, no interrupt request is sent.		
		10	The McBSP sends a RINT request to the CPU when each receive frame-synchronization pulse is detected. The interrupt request is sent even if the receiver is in its reset state.		
		11	The McBSP sends a RINT request to the C PU when the RSYNCERR bit is set, indicating a receive frame-synchronization error. Regardless of the value of RINTM, you can check RSYNCERR to determine whether a receive frame-synchronization error occurred.		
3	RSYNCERR		Receive frame-sync error bit. RSYNCERR is set when a receive frame-sync error is detected by the McBSP. If RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU when RSYNCERR is set. The flag remains set until you write a 0 to it or reset the receiver. **Caution:** If RINTM = 11b, writing a 1 to RSYNCERR triggers a receive interrupt just as if a receive frame-synchronization error occurred.	R/W	0

Table 12–86. Serial Port Control 1 Register (SPCR1) (Continued)

Bit	Name	Value	Description	Type	Reset Value
		0	No error		
		1	Receive frame-synchronization error For more details about this error, see Section 12.4.2, *Unexpected Receive Frame-Synchronization Pulse*.		
2	RFULL		Receiver full bit. RFULL is set when the receiver is full with new data and the previously received data has not been read (receiver-full condition). For more details about this condition, see Section 12.4.1, *Overrun in the Receiver*.	R	0
		0	No receiver-full condition		
		1	Receiver-full condition: RSR[1,2] and RBR[1,2] are full with new data, but the previous data in DRR[1,2] has not been read.		
1	RRDY		Receiver ready bit. RRDY is set when data is ready to be read from DRR[1,2]. Specifically, RRDY is set in response to a copy from RBR1 to DRR1. If the receive interrupt mode is RINTM = 00b, the McBSP sends a receive interrupt request to the CPU when RRDY changes from 0 to 1. Also, when RRDY changes from 0 to 1, the McBSP sends a receive synchronization event (REVT) signal to the DMA controller.	R	0
		0	Receiver not ready When the content of DRR1 is read, RRDY is automatically cleared.		
		1	Receiver ready: New data can be read from DRR[1,2]. **Note:** If both DRRs are required (word length larger than 16 bits), the CPU or the DMA controller must read from DRR2 first and then from DRR1. As soon as DRR1 is read, the next RBR-to-DRR copy is initiated. If DRR2 is not read first, the data in DRR2 is lost.		
0	RRST		Receiver reset bit. You can use RRST to take the McBSP receiver into and out of its reset state. This bit has a negative polarity; RRST = 0 indicates the reset state. To read about the effects of a receiver reset, see Section 12.10.3, *Resetting and Initializing a McBSP*.	R/W	0

Table 12–86. Serial Port Control 1 Register (SPCR1) (Continued)

Bit	Name	Value	Description	Type	Reset Value
		0	If you read a 0, the receiver is in its reset state. If you write a 0, you reset the receiver.		
		1	If you read a 1, the receiver is enabled. If you write a 1, you enable the receiver by taking it out of its reset state.		

Table 12–87. Serial Port Control 2 Register (SPCR2)

Bit	Name	Value	Description	Type	Reset Value
15:10	Reserved		Reserved bits (not available for use). They are read-only bits and return 0s when read.		
9	FREE		Free run bit. When a breakpoint is encountered in the high-level language debugger, FREE determines whether the McBSP transmit and receive clocks continue to run or whether they are affected as determined by the SOFT bit. When one of the clocks stops, the corresponding data transfer (transmission or reception) stops.	R/W	0
		0	The McBSP transmit and receive clocks are affected as determined by the SOFT bit.		
		1	Free run. The McBSP transmit and receive clocks continue to run.		
8	SOFT		Soft stop bit. When FREE = 0, SOFT determines the response of the McBSP transmit and receive clocks when a breakpoint is encountered in the high-level language debugger. When one of the clocks stops, the corresponding data transfer (transmission or reception) stops.	R/W	0
		0	Hard stop. The McBSP transmit and receive clocks are stopped immediately.		
		1	Soft stop. The McBSP transmit clock stops after completion of the current serial word transfer. The McBSP receive clock is not affected.		
7	FRST		Frame-synchronization logic reset bit. The sample rate generator of the McBSP includes frame-synchronization logic to generate an internal frame-synchronization signal. You can use FRST to take the frame-synchronization logic into and out of its reset state. This bit has a negative polarity; FRST = 0 indicates the reset state.	R/W	0

Table 12–87. Serial Port Control 2 Register (SPCR2) (Continued)

Bit	Name	Value	Description	Type	Reset Value
		0	If you read a 0, the frame-synchronization logic is in its reset state. If you write a 0, you reset the frame-synchronization logic. In the reset state, the frame-synchronization logic does not generate a frame-synchronization signal (FSG).		
		1	If you read a 1, the frame-synchronization logic is enabled. If you write a 1, you enable the frame-synchronization logic by taking it out of its reset state. When the frame-synchronization logic is enabled (FRST = 1) and the sample rate generator as a whole is enabled (GRST = 1), the frame-synchronization logic generates the frame-synchronization signal FSG as programmed.		
6	GRST		Sample-rate generator reset bit. You can use GRST to take the McBSP sample rate generator into and out of its reset state. This bit has a negative polarity; GRST = 0 indicates the reset state. To read about the effects of a sample-rate generator reset, see Section 12.10.3, *Resetting and Initializing a McBSP*.	R/W	0
		0	If you read a 0, the sample rate generator is in its reset state. If you write a 0, you reset the sample rate generator. If GRST = 0 because of a reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven low (inactive). If GRST = 0 because of program code, CLKG and FSG are both driven low (inactive).		
		1	If you read a 1, the sample rate generator is enabled. If you write a 1, you enable the sample rate generator by taking it out of its reset state. When enabled, the sample rate generator generates the clock signal CLKG as programmed in the sample rate generator registers. If FRST = 1, the generator also generates the frame-synchronization signal FSG as programmed in the sample-rate generator registers.		
5:4	XINTM		Transmit-interrupt mode bits. XINTM determines which event in the McBSP transmitter generates a transmit interrupt (XINT) request. If XINT is properly enabled, the CPU services the interrupt request; otherwise, the CPU ignores the request.	R/W	00

Table 12–87. Serial Port Control 2 Register (SPCR2) (Continued)

Bit	Name	Value	Description	Type	Reset Value
		00b	The McBSP sends a transmit interrupt (XINT) request to the CPU when the XRDY bit changes from 0 to 1, indicating that transmitter is ready to accept new data (the content of DXR[1,2] has been copied to XSR[1,2]). Regardless of the value of XINTM, you can check XRDY to determine whether a word transfer is complete. The McBSP sends an XINT request to the CPU when 16 enabled bits have been transmitted on the DX pin.		
		01b	In the multichannel selection mode, the McBSP sends an XINT request to the CPU after every 16-channel block is transmitted in a frame. Outside of the multichannel selection mode, no interrupt request is sent.		
		10b	The McBSP sends an XINT request to the CPU when each transmit frame-synchronization pulse is detected. The interrupt request is sent even if the transmitter is in its reset state.		
		11b	The McBSP sends an XINT request to the CPU when the XSYNCERR bit is set, indicating a transmit frame-synchronization error. Regardless of the value of XINTM, you can check XSYNCERR to determine whether a transmit frame-synchronization error occurred.		
3	XSYNCERR		Transmit frame-synchronization error bit. XSYNCERR is set when a transmit frame-synchronization error is detected by the McBSP. If XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU when XSYNCERR is set. The flag remains set until you write a 0 to it or reset the transmitter. If XINTM = 11b, writing a 1 to XSYNCERR triggers a transmit interrupt just as if a transmit frame-synchronization error occurred. For details about this error see Section 12.4.5, *Unexpected Transmit Frame-Synchronization Pulse*.	R/W	0
		0	No error		
		1	Transmit frame-synchronization error		
2	XEMPTY		Transmitter empty bit. XEMPTY is cleared when the transmitter is ready to send new data but no new data is available (transmitter-empty condition). This bit has a negative polarity; a transmitter-empty condition is indicated by XEMPTY = 0.	R	0

Table 12–87. Serial Port Control 2 Register (SPCR2) (Continued)

Bit	Name	Value	Description	Type	Reset Value
		0	Transmitter-empty condition Typically, this indicates that all of the bits of the current word have been transmitted, but there is no new data in DXR1. XEMPTY is also cleared if the transmitter is reset and then restarted. For more details about this error condition, see Section 12.4.4, *Underflow in the Transmitter*.		
		1	No transmitter-empty condition		
1	XRDY		Transmitter ready bit. XRDY is set when the transmitter is ready to accept new data in DXR[1,2]. Specifically, XRDY is set in response to a copy from DXR1 to XSR1. If the transmit interrupt mode is XINTM = 00b, the McBSP sends a transmit interrupt (XINT) request to the CPU when XRDY changes from 0 to 1. Also, when XRDY changes from 0 to 1, the McBSP sends a transmit synchronization event (XEVT) signal to the DMA controller.	R	0
		0	Transmitter not ready When DXR1 is loaded, XRDY is automatically cleared.		
		1	Transmitter ready: DXR[1,2] is ready to accept new data. If both DXRs are needed (word length larger than 16 bits), the CPU or the DMA controller must load DXR2 first and then load DXR1. As soon as DXR1 is loaded, the contents of both DXRs are copied to the transmit shift registers (XSRs), as described in the next step. If DXR2 is not loaded first, the previous content of DXR2 is passed to the XSR2.		
0	XRST		Transmitter reset bit. You can use XRST to take the McBSP transmitter into and out of its reset state. This bit has a negative polarity; XRST = 0 indicates the reset state. To read about the effects of a transmitter reset, see Section 12.10.3, *Resetting and Initializing a McBSP*.	R/W	0
		0	If you read a 0, the transmitter is in its reset state. If you write a 0, you reset the transmitter.		
		1	If you read a 1, the transmitter is enabled. If you write a 1, you enable the transmitter by taking it out of its reset state.		

12.13.4 Receive Control Registers (RCR1 and RCR2)

Each McBSP has two receive control registers. Table 12–88 and Table 12–90 describe the bits of RCR1 and RCR2, respectively. These I/O-mapped registers enable you to:

- Specify one or two phases for each frame of receive data (RPHASE)
- Define two parameters for phase 1 and (if necessary) phase 2: the serial word length (RWDLEN1, RWDLEN2) and the number of words (RFRLEN1, RFRLEN2)
- Choose a receive companding mode, if any (RCOMPAND)
- Enable or disable the receive frame-synchronization ignore function (RFIG)
- Choose a receive data delay (RDATDLY)

Table 12–88. Receive Control 1 Register (RCR1)

Bit	Name	Value	Description	Type	Reset Value
15	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return zeros when read.	R	0
14:8	RFRLEN1	0–127	Receive frame length 1 (1 to 128 words). Each frame of receive data can have one or two phases, depending on the value that you load into the RPHASE bit. If a single-phase frame is selected, RFRLEN1 in RCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, RFRLEN1 determines the number of serial words in phase 1 of the frame, and RFRLEN2 in RCR2 determines the number of words in phase 2 of the frame. The 7-bit RFRLEN fields allow up to 128 words per phase. See Table 12–89 for a summary of how to determine the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization period. Program the RFRLEN fields with [*w minus 1*], where *w* represents the number of words per phase. For example, for a phase length of 128 words in phase 1, load 127 into RFRLEN1.	R/W	0
7:5	RWDLEN1		Receive word length 1. Each frame of receive data can have one or two phases, depending on the value loaded into the RPHASE bit. If a single-phase frame is selected, RWDLEN1 in RCR1 selects the length for every serial word received in the frame. If a dual-phase frame is selected, RWDLEN1 determines the length of the serial words in phase 1 of the frame, and RWDLEN2 in RCR2 determines the word length in phase 2 of the frame.	R/W	000
		000b	8 bits		
		001b	12 bits		
		010b	16 bits		
		011b	20 bits		
		100b	24 bits		
		101b	32 bits		
		other	Reserved (do not use)		
4:0	Reserved	0	Reserved bits (not available for use). They are read-only bits and return zeros when read.	R	0

Table 12–89. Frame Length Formula for Receive Control 1 Register (RCR1)

RPHASE	RFRLEN1	RFRLEN2	Frame Length
0	0 ≤ RFRLEN1 ≤ 127	Not used	(RFRLEN1 + 1) words
1	0 ≤ RFRLEN1 ≤ 127	0 ≤ RFRLEN2 ≤ 127	(RFRLEN1 + 1) + (RFRLEN2 + 1) words

Table 12–90. Receive Control 2 Register (RCR2)

Bit	Name	Value	Description	Type	Reset Value
15	RPHASE		Receive phase number bit. RPHASE determines whether the receive frame has one phase or two phases. For each phase you can define the serial word length and the number of serial words in the phase. To set up phase 1, program RWDLEN1 (word length) and RFRLEN1 (number of words). To set up phase 2 (if there are two phases), program RWDLEN2 and RFRLEN2.	R/W	0
		0	Single-phase frame The receive frame has only one phase, phase 1.		
		1	Dual-phase frame The receive frame has two phases, phase 1 and phase 2.		
14:8	RFRLEN2	0–127	Receive frame length 2 (1 to 128 words). Each frame of receive data can have one or two phases, depending on value loaded into the RPHASE bit. If a single-phase frame is selected, RFRLEN1 in RCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, RFRLEN1 determines the number of serial words in phase 1 of the frame, and RFRLEN2 in RCR2 determines the number of words in phase 2 of the frame. The 7-bit RFRLEN fields allow up to 128 words per phase. See Table 12–91 for a summary of how to determine the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization period. Program the RFRLEN fields with [*w minus 1*], where *w* represents the number of words per phase. For example, for a phase length of 128 words in phase 2, load 127 into RFRLEN2.	R/W	0
7:5	RWDLEN2		Receive word length 2. Each frame of receive data can have one or two phases, depending on the value loaded into the RPHASE bit. If a single-phase frame is selected, RWDLEN1 in RCR1 selects the length for every serial word received in the frame. If a dual-phase frame is selected, RWDLEN1 determines the length of the serial words in phase 1 of the frame, and RWDLEN2 in RCR2 determines the word length in phase 2 of the frame.	R/W	000

Table 12–90. Receive Control 2 Register (RCR2) (Continued)

Bit	Name	Value	Description	Type	Reset Value
		000b	8 bits		
		001b	12 bits		
		010b	16 bits		
		011b	20 bits		
		100b	24 bits		
		101b	32 bits		
		other	Reserved (do not use)		
4:3	RCOMPAND		Receive companding mode bits. Companding (COMpress and exPAND) hardware allows compression and expansion of data in either μ-law or A-law format. RCOMPAND allows you to choose one of the following companding modes for the McBSP receiver: For more details about these companding modes, see Section 12.2.2, *Companding (Compressing and Expanding) Data*.	R/W	00
		00b	No companding, any size data, MSB received first		
		01b	No companding, 8-bit data, LSB received first		
		10b	μ-law companding, 8-bit data, MSB received first		
		11b	A-law companding, 8-bit data, MSB received first		
2	RFIG		Receive frame-synchronization ignore bit. If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-synchronization pulse. For more details about the frame-synchronization error condition, see Section 12.4.2, *Unexpected Receive Frame-Synchronization Pulse*. Setting RFIG causes the serial port to ignore unexpected frame-synchronization signals during reception. For more details on the effects of RFIG, see Section 12.7.10.1, *Enable/Disable the Receive Frame-Synchronization Ignore Function*.	R/W	0
		0	Frame-synchronization detect. An unexpected FSR pulse causes the receiver to discard the contents of RSR[1,2] in favor of the new incoming data. The receiver: 1) Aborts the current data transfer 2) Sets RSYNCERR in SPCR1 3) Begins the transfer of a new data word		

Table 12–90. Receive Control 2 Register (RCR2) (Continued)

Bit	Name	Value	Description	Type	Reset Value
		1	Frame-synchronization ignore. An unexpected FSR pulse is ignored. Reception continues uninterrupted.		
1:0	RDATDLY		Receive data delay bits. RDATDLY specifies a data delay of 0, 1, or 2 receive clock cycles after frame-synchronization and before the reception of the first bit of the frame. For more details, see Section 12.7.12, *Set the Receive Data Delay*.	R/W	00
		00b	0-bit data delay		
		01b	1-bit data delay		
		10b	2-bit data delay		
		11b	Reserved (do not use)		

Table 12–91. Frame Length Formula for RCR2

RPHASE	RFRLEN1	RFRLEN2	Frame Length
0	0 ≤ RFRLEN1 ≤ 127	Not used	(RFRLEN1 + 1) words
1	0 ≤ RFRLEN1 ≤ 127	0 ≤ RFRLEN2 ≤ 127	(RFRLEN1 + 1) + (RFRLEN2 + 1) words

12.13.5 Transmit Control Registers (XCR1 and XCR2)

Each McBSP has two transmit control registers. Table 12–92 and Table 12–94 describe the bits of XCR1 and XCR2, respectively. These I/O-mapped registers enable you to:

- ❏ Specify one or two phases for each frame of transmit data (XPHASE)
- ❏ Define two parameters for phase 1 and (if necessary) phase 2: the serial word length (XWDLEN1, XWDLEN2) and the number of words (XFRLEN1, XFRLEN2)
- ❏ Choose a transmit companding mode, if any (XCOMPAND)
- ❏ Enable or disable the transmit frame-sync ignore function (XFIG)
- ❏ Choose a transmit data delay (XDATDLY)

Table 12–92. Transmit Control 1 Register (XCR1)

Bit	Name	Value	Description	Type	Reset Value
15	Reserved	0	Reserved bit. Read-only; returns 0 when read.	R	0
14:8	XFRLEN1	0–127	Transmit frame length 1 (1 to 128 words). Each frame of transmit data can have one or two phases, depending on value loaded into the XPHASE bit. If a single-phase frame is selected, XFRLEN1 in XCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, XFRLEN1 determines the number of serial words in phase 1 of the frame and XFRLEN2 in XCR2 determines the number of words in phase 2 of the frame. The 7-bit XFRLEN fields allow up to 128 words per phase. See Table 12–93 for a summary of how you determine the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization period. Program the XFRLEN fields with [*w minus 1*], where *w* represents the number of words per phase. For example, for a phase length of 128 words in phase 1, load 127 into XFRLEN1.	R/W	0
7:5	XWDLEN1		Transmit word length 1. Each frame of transmit data can have one or two phases, depending on the value loaded into the XPHASE bit. If a single-phase frame is selected, XWDLEN1 in XCR1 selects the length for every serial word transmitted in the frame. If a dual-phase frame is selected, XWDLEN1 determines the length of the serial words in phase 1 of the frame and XWDLEN2 in XCR2 determines the word length in phase 2 of the frame.	R/W	000
		000b	8 bits		
		001b	12 bits		
		010b	16 bits		
		011b	20 bits		
		100b	24 bits		
		101b	32 bits		
		other	Reserved (do not use)		
4:0	Reserved	0	Reserved bits. They are read-only bits and return zeros when read.	R	0

Table 12–93. Frame Length Formula for Transmit Control 1 Register (XCR1)

XPHASE	XFRLEN1	XFRLEN2	Frame Length
0	0 ≤ XFRLEN1 ≤ 127	Not used	(XFRLEN1 + 1) words
1	0 ≤ XFRLEN1 ≤ 127	0 ≤ XFRLEN2 ≤ 127	(XFRLEN1 + 1) + (XFRLEN2 + 1) words

Table 12–94. Transmit Control 2 Register (XCR2)

Bit	Name	Value	Description	Type	Reset Value
15	XPHASE		Transmit phase number bit. XPHASE determines whether the transmit frame has one phase or two phases. For each phase, you can define the serial word length and the number of serial words in the phase. To set up phase 1, program XWDLEN1 (word length) and XFRLEN1 (number of words). To set up phase 2 (if there are two phases), program XWDLEN2 and XFRLEN2.	R/W	0
		0	Single-phase frame The transmit frame has only one phase, phase 1.		
		1	Dual-phase frame The transmit frame has two phases, phase 1 and phase 2.		
14:8	XFRLEN2	0–127	Transmit frame length 2 (1 to 128 words). Each frame of transmit data can have one or two phases, depending on value loaded into the XPHASE bit. If a single-phase frame is selected, XFRLEN1 in XCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, XFRLEN1 determines the number of serial words in phase 1 of the frame and XFRLEN2 in XCR2 determines the number of words in phase 2 of the frame. The 7-bit XFRLEN fields allow up to 128 words per phase. See Table 12–95 for a summary of how to determine the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization period. Program the XFRLEN fields with [*w minus 1*], where *w* represents the number of words per phase. For example,for a phase length of 128 words in phase 1, load 127 into XFRLEN1.	R/W	0
7:5	XWDLEN2		Transmit word length 2. Each frame of transmit data can have one or two phases, depending on the value loaded into the XPHASE bit. If a single-phase frame is selected, XWDLEN1 in XCR1 selects the length for every serial word transmitted in the frame. If a dual-phase frame is selected, XWDLEN1 determines the length of the serial words in phase 1 of the frame and XWDLEN2 in XCR2 determines the word length in phase 2 of the frame.	R/W	000
		000b	8 bits		
		001b	12 bits		
		010b	16 bits		

Table 12–94. Transmit Control 2 Register (XCR2) (Continued)

Bit	Name	Value	Description	Type	Reset Value
		011b	20 bits		
		100b	24 bits		
		101b	32 bits		
		other	Reserved (do not use)		
4:3	XCOMPAND		Transmit companding mode bits. Companding (COMpress and exPAND) hardware allows compression and expansion of data in either μ-law or A-law format. For more details, see Section 12.2.2, *Companding Data*.	R/W	00
			XCOMPAND allows you to choose one of the following companding modes for the McBSP transmitter. For more details about these companding modes, see Section 12.2.2, *Companding (Compressing and Expanding) Data*.		
		00b	No companding, any size data, MSB transmitted first		
		01b	No companding, 8-bit data, LSB transmitted first		
		10b	μ-law companding, 8-bit data, MSB transmitted first		
		11b	A-law companding, 8-bit data, MSB transmitted first		

Table 12–94. Transmit Control 2 Register (XCR2) (Continued)

Bit	Name	Value	Description	Type	Reset Value
2	XFIG		Transmit frame-synchronization ignore bit. If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-synchronization pulse. For more details about the frame-synchronization error condition, see Section 12.4.5, *Unexpected Transmit Frame-Synchronization Pulse*.	R/W	0
			Setting XFIG causes the serial port to ignore unexpected frame-synchronization pulses during transmission. For more details on the effects of XFIG, see Section 12.8.10, *Enable/Disable the Transmit Frame-Synchronization Ignore Function*.		
		0	Frame-synchronization detect. An unexpected FSX pulse causes the transmitter to discard the content of XSR[1,2]. The transmitter: 1) Aborts the present transmission 2) Sets XSYNCERR in SPCR2 3) Begins a new transmission from DXR[1,2]. If new data was written to DXR[1,2] since the last DXR[1,2]-to-XSR[1,2] copy, the current value in XSR[1,2] is lost. Otherwise, the same data is transmitted.		
		1	Frame-synchronization ignore. An unexpected FSX pulse is ignored. Transmission continues uninterrupted.		
1:0	XDATDLY		Transmit data delay bits. XDATDLY specifies a data delay of 0, 1, or 2 transmit clock cycles after frame synchronization and before the transmission of the first bit of the frame. For more details, see Section 12.8.12, *Set the Transmit Data Delay*.	R/W	00
		00b	0-bit data delay		
		01b	1-bit data delay		
		10b	2-bit data delay		
		11b	Reserved (do not use)		

Table 12–95. Frame Length Formula for Transmit Control 2 Register (XCR2)

XPHASE	XFRLEN1	XFRLEN2	Frame Length
0	0 ≤ XFRLEN1 ≤ 127	Not used	(XFRLEN1 + 1) words
1	0 ≤ XFRLEN1 ≤ 127	0 ≤ XFRLEN2 ≤ 127	(XFRLEN1 + 1) + (XFRLEN2 + 1) words

12.13.6 Sample Rate Generator Registers (SRGR1 and SRGR2)

Each McBSP has two sample rate generator registers. Table 12–96 and Table 12–97 describe the bits of SRGR1 and SRGR2, respectively. The sample rate generator can generate a clock signal (CLKG) and a frame-synchronization signal (FSG). The I/O-mapped registers SRGR1 and SRGR2 enable you to:

- Select the input clock source for the sample rate generator (CLKSM, in conjunction with the SCLKME bit of PCR)
- Divide down the frequency of CLKG (CLKGDV)
- Select whether internally-generated transmit frame-synchronization pulses are driven by FSG or by activity in the transmitter (FSGM).
- Specify the width of frame-synchronization pulses on FSG (FWID) and specify the period between those pulses (FPER)

When an external source (via the CLKS, CLKR, or CLKX pin) provides the input clock source for the sample rate generator:

- If the CLKS pin provides the input clock, the CLKSP bit in SRGR2 allows you to select whether the rising edge or the falling edge of CLKS triggers CLKG and FSG. If the CLKX/CLKR pin is used instead of the CLKS pin, the polarity of the input clock is selected with CLKXP/CLKRP of PCR.
- The GSYNC bit of SRGR2 allows you to make CLKG synchronized to an external frame-synchronization signal on the FSR pin, so that CLKG is kept in phase with the input clock.

Table 12–96. Sample Rate Generator 1 Register (SRGR1)

Bit	Name	Value	Description	Type	Reset Value
15:8	FWID	0–255	Frame-synchronization pulse width bits for FSG The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. For frame-synchronization pulses on FSG, (FWID + 1) is the pulse width in CLKG cycles. The 8 bits of FWID allow a pulse width of 1 to 256 CLKG cycles: $0 \le FWID \le 255$ $1 \le (FWID + 1) \le 256$ CLKG cycles The period between the frame-synchronization pulses on FSG is defined by the FPER bits.	R/W	0
7:0	CLKGDV	0–255	Divide-down value for CLKG. The sample rate generator can accept an input clock signal and divide it down according to CLKGDV to produce an output clock signal, CLKG. The frequency of CLKG is: CLKG frequency = (Input clock frequency)/ (CLKGDV + 1) The input clock is selected by the SCLKME and CLKSM bits: **SCLKME** / **CLKSM** / **Input Clock For Sample Rate Generator** 0 / 0 / Signal on CLKS pin 0 / 1 / CPU clock 1 / 0 / Signal on CLKR pin 1 / 1 / Signal on CLKX pin A reset forces the CLKG frequency to 1/2 the input clock frequency (CLKGDV = 1), and the reset selects the CPU clock as the input clock.	R/W	1

Table 12–97. Sample Rate Generator 2 Register (SRGR2)

Bit	Name	Value	Description	Type	Reset Value
15	GSYNC		Clock synchronization mode bit for CLKG. GSYNC is used only when the input clock source for the sample rate generator is external—on the CLKS, CLKR, or CLKX pin. When GSYNC = 1, the clock signal (CLKG) and the frame-synchronization signal (FSG) generated by the sample rate generator are made dependent on pulses on the FSR pin.	R/W	0
		0	No clock synchronization CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles.		
		1	Clock synchronization ❑ CLKG is adjusted as necessary so that it is synchronized with the input clock on the CLKS, CLKR, or CLKX pin. ❑ FSG pulses. FSG only pulses in response to a pulse on the FSR pin. The frame-synchronization period defined in FPER is ignored. For more details, see Section 12.3.3, *Synchronizing Sample Rate Generator Outputs to an External Clock*.		
14	CLKSP		CLKS pin polarity bit. CLKSP is used only when the CLKS pin is the input clock source for the sample rate generator. The bit determines which edge of CLKS drives the clock signal (CLKG) and the frame-synchronization signal (FSG) that are generated by the sample rate generator:	R/W	0
		0	A rising edge on the CLKS pin		
		1	A falling edge on the CLKS pin		
13	CLKSM		Sample rate generator input clock mode bit. The sample rate generator can accept an input clock signal and divide it down according to CLKGDV to produce an output clock signal, CLKG. The frequency of CLKG is: CLKG frequency = (input clock frequency)/ (CLKGDV + 1) CLKSM is used in conjunction with the SCLKME bit to determine the source for the input clock. A reset selects the CPU clock as the input clock and forces the CLKG frequency to 1/2 the CPU clock frequency.	R/W	1

Table 12–97. Sample Rate Generator 2 Register (SRGR2) (Continued)

Bit	Name	Value	Description	Type	Reset Value
		0	The input clock for the sample rate generator is taken from the CLKS pin or from the CLKR pin, depending on the value of the SCLKME bit of PCR:		
			SCLKME / **CLKSM** / **Input Clock For Sample Rate Generator** 0 / 0 / Signal on CLKS pin 1 / 0 / Signal on CLKR pin		
		1	The input clock for the sample rate generator is taken from the CPU clock or from the CLKX pin, depending on the value of the SCLKME bit of PCR:		
			SCLKME / **CLKSM** / **Input Clock For Sample Rate Generator** 0 / 1 / CPU clock 1 / 1 / Signal on CLKX pin		
12	FSGM		Sample-rate generator transmit frame-synchronization mode bit. The transmitter can get frame synchronization from the FSX pin (FSXM = 0) or from inside the McBSP (FSXM = 1). When FSXM = 1, the FSGM bit determines how the McBSP supplies frame-synchronization pulses.	R/W	0
		0	If FSXM = 1, the McBSP generates a transmit frame-synchronization pulse when the content of DXR[1,2] is copied to XSR[1,2].		
		1	If FSXM = 1, the transmitter uses frame-synchronization pulses generated by the sample rate generator. Program the FWID bits to set the width of each pulse. Program the FPER bits to set the period between pulses.		
11:0	FPER	0–4095	Frame-synchronization period bits for FSG. The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. The period between frame-synchronization pulses on FSG is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-synchronization period of 1 to 4096 CLKG cycles: $0 \leq FPER \leq 4095$ $1 \leq (FPER + 1) \leq 4096$ CLKG cycles The width of each frame-synchronization pulse on FSG is defined by the FWID bits.	R/W	0

12.13.7 Multichannel Control Registers (MCR1 and MCR2)

Each McBSP has two multichannel control registers. MCR1 has control and status bits (with an R prefix) for multichannel selection operation in the receiver. MCR2 contains the same type of bits (bit with an X prefix) for the transmitter. The bits of MCR1 and MCR2 are described in Table 12–98 and Table 12–99, respectively. These I/O-mapped registers enable you to:

- ❏ Enable all channels or selected channels for reception (RMCM)
- ❏ Choose which channels are enabled/disabled and masked/unmasked for transmission (XMCM)
- ❏ Specify whether two partitions (32 channels at a time) or eight partitions (128 channels at a time) can be used (RMCME for reception, XMCME for transmission)
- ❏ Assign blocks of 16 channels to partitions A and B when the 2-partition mode is selected (RPABLK and RPBBLK for reception, XPABLK and XPBBLK for transmission)
- ❏ Determine which block of 16 channels is currently involved in a data transfer (RCBLK for reception, XCBLK for transmission)

Table 12–98. Multichannel Control 1 Register (MCR1)

Bit	Name	Value	Description	Type	Reset Value
15:10	Reserved	0	Reserved bits (not available for use). They are read-only bits and return zeros when read.	R	0
9	RMCME		Receive multichannel partition mode bit. RMCME is only applicable if channels can be individually enabled or disabled for reception (RMCM = 1). RMCME determines whether only 32 channels or all 128 channels are to be individually selectable.	R/W	0
		0	2-partition mode Only partitions A and B are used. You can control up to 32 channels in the receive multichannel selection mode (RMCM = 1). Assign 16 channels to partition A with the RPABLK bits. Assign 16 channels to partition B with the RPBBLK bits. You control the channels with the appropriate receive channel enable registers: RCERA: Channels in partition A RCERB: Channels in partition B		
		1	8-partition mode All partitions (A through H) are used. You can control up to 128 channels in the receive multichannel selection mode. You control the channels with the appropriate receive channel enable registers: RCERA: Channels 0 through 15 RCERB: Channels 16 through 31 RCERC: Channels 32 through 47 RCERD: Channels 48 through 63 RCERE: Channels 64 through 79 RCERF: Channels 80 through 95 RCERG: Channels 96 through 111 RCERH: Channels 112 through 127		

Table 12–98. Multichannel Control 1 Register (MCR1) (Continued)

Bit	Name	Value	Description	Type	Reset Value
8:7	RPBBLK		Receive partition B block bits	R/W	00
			RPBBLK is only applicable if channels can be individually enabled or disabled (RMCM = 1) and the 2-partition mode is selected (RMCME = 0). Under these conditions, the McBSP receiver can accept or ignore data in any of the 32 channels that are assigned to partitions A and B of the receiver.		
			The 128 receive channels of the McBSP are divided equally among 8 blocks (0 through 7). When RPBBLK is applicable, use RPBBLK to assign one of the odd-numbered blocks (1, 3, 5, or 7) to partition B. Use the RPABLK bits to assign one of the even-numbered blocks (0, 2, 4, or 6) to partition A.		
			If you want to use more than 32 channels, you can change block assignments dynamically. You can assign a new block to one partition while the receiver is handling activity in the other partition. For example, while the block in partition A is active, you can change which block is assigned to partition B. The RCBLK bits are regularly updated to indicate which block is active.		
			When XMCM = 11b (for symmetric transmission and reception), the transmitter uses the receive block bits (RPABLK and RPBBLK) rather than the transmit block bits (XPABLK and XPBBLK).		
		00b	Block 1: Channels 16 through 31		
		01b	Block 3: Channels 48 through 63		
		10b	Block 5: Channels 80 through 95		
		11b	Block 7: Channels 112 through 127		
6:5	RPABLK		Receive partition A block bits	R/W	00
			RPABLK is only applicable if channels can be individually enabled or disabled (RMCM = 1) and the 2-partition mode is selected (RMCME = 0). Under these conditions, the McBSP receiver can accept or ignore data in any of the 32 channels that are assigned to partitions A and B of the receiver. See the description for RPBBLK (bits 8–7) for more information about assigning blocks to partitions A and B.		
		00b	Block 0: Channels 0 through 15		
		01b	Block 2: Channels 32 through 47		
		10b	Block 4: Channels 64 through 79		
		11b	Block 6: Channels 96 through 111		

Table 12–98. Multichannel Control 1 Register (MCR1) (Continued)

Bit	Name	Value	Description	Type	Reset Value
4:2	RCBLK		Receive current block indicator. RCBLK indicates which block of 16 channels is involved in the current McBSP reception:	R	000
		000b	Block 0: Channels 0 through 15		
		001b	Block 1: Channels 16 through 31		
		010b	Block 2: Channels 32 through 47		
		011b	Block 3: Channels 48 through 63		
		100b	Block 4: Channels 64 through 79		
		101b	Block 5: Channels 80 through 95		
		110b	Block 6: Channels 96 through 111		
		111b	Block 7: Channels 112 through 127		
1	Reserved	0	Reserved bits (not available for use). They are read-only bits and return zeros when read.	R	0
0	RMCM		Receive multichannel selection mode bit. RMCM determines whether all channels or only selected channels are enabled for reception:	R/W	0
		0	All 128 channels are enabled.		
		1	Multichannel selection mode. Channels can be individually enabled or disabled. The only channels enabled are those selected in the appropriate receive channel enable registers (RCERs). The way channels are assigned to the RCERs depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit.		

Table 12–99. Multichannel Control 2 Register (MCR2)

Bit	Name	Value	Description	Type	Reset Value
15:10	Reserved	0	Reserved bits (not available for use). They are read-only bits and return zeros when read.	R	0
9	XMCME		Transmit multichannel partition mode bit. XMCME determines whether only 32 channels or all 128 channels are to be individually selectable. XMCME is only applicable if channels can be individually disabled/enabled or masked/unmasked for transmission (XMCM is nonzero).	R/W	0
		0	2-partition mode. Only partitions A and B are used. You can control up to 32 channels in the transmit multichannel selection mode selected with the XMCM bits. If XMCM = 01b or 10b, assign 16 channels to partition A with the XPABLK bits. Assign 16 channels to partition B with the XPBBLK bits. If XMCM = 11b (for symmetric transmission and reception), assign 16 channels to receive partition A with the RPABLK bits. Assign 16 channels to receive partition B with the RPBBLK bits. You control the channels with the appropriate transmit channel enable registers: XCERA: Channels in partition A XCERB: Channels in partition B		
		1	8-partition mode. All partitions (A through H) are used. You can control up to 128 channels in the transmit multichannel selection mode selected with the XMCM bits. You control the channels with the appropriate transmit channel enable registers: XCERA: Channels 0 through 15 XCERB: Channels 16 through 31 XCERC: Channels 32 through 47 XCERD: Channels 48 through 63 XCERE: Channels 64 through 79 XCERF: Channels 80 through 95 XCERG: Channels 96 through 111 XCERH: Channels 112 through 127		

Table 12–99. Multichannel Control 2 Register (MCR2) (Continued)

Bit	Name	Value	Description	Type	Reset Value
8:7	XPBBLK		Transmit partition B block bits	R/W	00
			XPBBLK is only applicable if channels can be individually disabled/enabled and masked/unmasked (XMCM is nonzero) and the 2-partition mode is selected (XMCME = 0). Under these conditions, the McBSP transmitter can transmit or withhold data in any of the 32 channels that are assigned to partitions A and B of the transmitter.		
			The 128 transmit channels of the McBSP are divided equally among 8 blocks (0 through 7). When XPBBLK is applicable, use XPBBLK to assign one of the odd-numbered blocks (1, 3, 5, or 7) to partition B, as shown in the following table. Use the XPABLK bit to assign one of the even-numbered blocks (0, 2, 4, or 6) to partition A.		
			If you want to use more than 32 channels, you can change block assignments dynamically. You can assign a new block to one partition while the transmitter is handling activity in the other partition. For example, while the block in partition A is active, you can change which block is assigned to partition B. The XCBLK bits are regularly updated to indicate which block is active.		
			When XMCM = 11b (for symmetric transmission and reception), the transmitter uses the receive block bits (RPABLK and RPBBLK) rather than the transmit block bits (XPABLK and XPBBLK).		
		00b	Block 1: Channels 16 through 31		
		01b	Block 3: Channels 48 through 63		
		10b	Block 5: Channels 80 through 95		
		11b	Block 7: Channels 112 through 127		
6:5	XPABLK		Transmit partition A block bits. XPABLK is only applicable if channels can be individually disabled/enabled and masked/unmasked (XMCM is nonzero) and the 2-partition mode is selected (XMCME = 0). Under these conditions, the McBSP transmitter can transmit or withhold data in any of the 32 channels that are assigned to partitions A and B of the transmitter. See the description for XPBBLK (bits 8–7) for more information about assigning blocks to partitions A and B.	R/W	00
		00b	Block 0: Channels 0 through 15		
		01b	Block 2: Channels 32 through 47		
		10b	Block 4: Channels 64 through 79		
		11b	Block 6: Channels 96 through 111		
4:2	XCBLK		Transmit current block indicator. XCBLK indicates which block of 16 channels is involved in the current McBSP transmission:	R	000

Table 12–99. Multichannel Control 2 Register (MCR2) (Continued)

Bit	Name	Value	Description	Type	Reset Value
		000b	Block 0: Channels 0 through 15		
		001b	Block 1: Channels 16 through 31		
		010b	Block 2: Channels 32 through 47		
		011b	Block 3: Channels 48 through 63		
		100b	Block 4: Channels 64 through 79		
		101b	Block 5: Channels 80 through 95		
		110b	Block 6: Channels 96 through 111		
		111b	Block 7: Channels 112 through 127		
1:0	XMCM		Transmit multichannel selection mode bits. XMCM determines whether all channels or only selected channels are enabled and unmasked for transmission. For more details on how the channels are affected, see Section 12.5.7 *Transmit Multichannel Selection Modes*.	R/W	00
		00b	No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.		
		01b	All channels are disabled unless they are selected in the appropriate transmit channel enable registers (XCERs). If enabled, a channel in this mode is also unmasked. The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERs.		
		10b	All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERs.		
		11b	This mode is used for symmetric transmission and reception. All channels are disabled for transmission unless they are enabled for reception in the appropriate receive channel enable registers (RCERs). Once enabled, they are masked unless they are also selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit determines whether 32 channels or 128 channels are selectable in RCERs and XCERs.		

12.13.8 Pin Control Register (PCR)

Each McBSP has one pin control register. Table 12–100 describes the bits of PCR. This I/O-mapped register enables you to:

- ❑ Allow the McBSP to enter a low-power mode when the idle instruction is executed (IDLEEN, in conjunction with the PERI bit of ICR)
- ❑ Specify whether McBSP pins can be used as general-purpose I/O pins when the transmitter and/or receiver is in its reset state (XIOEN and RIOEN)
- ❑ Choose a frame-synchronization mode for the transmitter (FSXM) and for the receiver (FSRM)
- ❑ Choose a clock mode for transmitter (CLKXM) and for the receiver (CLKRM)
- ❑ Select the input clock source for the sample rate generator (SCLKME, in conjunction with the CLKSM bit of SRGR2)
- ❑ Read or write data when the CLKS, DX, and DR pins are configured as general-purpose I/O pins (CLKSSTAT, DXSTAT, and DXSTAT)
- ❑ Choose whether frame-synchronization signals are active low or active high (FSXP for transmission, FSRP for reception)
- ❑ Specify whether data is sampled on the falling edge or the rising edge of the clock signals (CLKXP for transmission, CLKRP for reception)

Table 12–100. Pin Control Register (PCR)

Bit	Name	Value	Description	Type	Reset Value
15	Reserved	0	Reserved bit (not available for your use). It is a read-only bit and returns a 0 when read.	R	0
14	IDLEEN		Idle enable bit. If the PERIPH idle domain is configured to be idle and IDLEEN = 1, the McBSP stops and enters a low-power state.	R/W	0
		0	The McBSP is running.		
		1	If the PERIPH domain is idle (PERIS = 1 in the idle status register), the McBSP is stopped in a low-power state.		
13	XIOEN		Transmit I/O enable bit. When the transmitter is in reset (XRST = 0), XIOEN can configure certain McBSP pins as general-purpose I/O (GPIO) pins.	R/W	0
		0	The CLKX, FSX, DX, and CLKS pins are serial port pins.		
		1	If XRST = 0, the CLKX, FSX, and DX pins are GPIO pins. The CLKS is also a GPIO pin if RRST = 0 and RIOEN = 1.		
12	RIOEN		Receive I/O enable bit. When the receiver is in reset (RRST = 0), RIOEN can configure certain McBSP pins as general-purpose I/O (GPIO) pins. XRST and RRST are in the serial port control registers, but all other bits mentioned in this table are in the pin control register.	R/W	0

Table 12–100. Pin Control Register (PCR) (Continued)

Bit	Name	Value	Description	Type	Reset Value
		0	The CLKR, FSR, DR, and CLKS pins are serial port pins.		
		1	If RRST = 0, the CLKR, FSR, and DR pins are GPIO pins. The CLKS is also a GPIO pin if XRST = 0 and XIOEN = 1.		
11	FSXM		Transmit frame-synchronization mode bit. FSXM determines whether transmit frame-synchronization pulses are supplied externally or internally. The polarity of the signal on the FSX pin is determined by the FSXP bit.	R/W	0
		0	Transmit frame synchronization is supplied by an external source via the FSX pin.		
		1	Transmit frame synchronization is supplied by the McBSP, as determined by the FSGM bit of SRGR2.		
10	FSRM		Receive frame-synchronization mode bit. FSRM determines whether receive frame-synchronization pulses are supplied externally or internally. The polarity of the signal on the FSR pin is determined by the FSRP bit.	R/W	0
		0	Receive frame synchronization is supplied by an external source via the FSR pin.		
		1	Receive frame synchronization is supplied by the sample rate generator. FSR is an output pin reflecting internal FSR, except when GSYNC = 1 in SRGR2.		
9	CLKXM		Transmit clock mode bit. CLKXM determines whether the source for the transmit clock is external or internal and whether the CLKX pin is an input or an output. The polarity of the signal on the CLKX pin is determined by the CLKXP bit.	R/W	0
			In the clock stop mode (CLKSTP = 10b or 11b), the McBSP can act as a master or as a slave in the SPI protocol. If the McBSP is a master, ensure that CLKX is an output. If the McBSP is a slave, ensure that CLKX is an input.		
			Not in clock stop mode (CLKSTP = 00b or 01b):		
		0	The transmitter gets its clock signal from an external source via the CLKX pin.		
		1	Internal CLKX is driven by the sample rate generator of the McBSP. The CLKX pin is an output pin that reflects internal CLKX.		
			In clock stop mode (CLKSTP = 10b or 11b):		
		0	The McBSP is a slave in the SPI protocol. The internal transmit clock (CLKX) is driven by the SPI master via the CLKX pin. The internal receive clock (CLKR) is driven internally by CLKX so that both the transmitter and the receiver are controlled by the external master clock.		

Table 12–100. Pin Control Register (PCR) (Continued)

Bit	Name	Value	Description	Type	Reset Value
		1	The McBSP is a master in the SPI protocol. The sample rate generator drives the internal transmit clock (CLKX). Internal CLKX is reflected on the CLKX pin to drive the shift clock of the SPI-compliant slaves in the system. Internal CLKX also drives the internal receive clock (CLKR) so that both the transmitter and the receiver are controlled by the internal master clock.		
8	CLKRM		Receive clock mode bit. The role of CLKRM and the resulting effect on the CLKR pin depend on whether the McBSP is in the digital loopback mode (DLB = 1). The polarity of the signal on the CLKR pin is determined by the CLKRP bit. Not in digital loopback mode (DLB = 0):	R/W	0
		0	The CLKR pin is an input pin that supplies the internal receive clock (CLKR).		
		1	Internal CLKR is driven by the sample rate generator of the McBSP. The CLKR pin is an output pin that reflects internal CLKR. In digital loopback mode (DLB = 1):		
		0	The CLKR pin is in high impedance state. The internal receive clock (CLKR) is driven by the internal transmit clock (CLKX). CLKX is derived according to the CLKXM bit.		
		1	Internal CLKR is driven by internal CLKX. The CLKR pin is an output pin that reflects internal CLKR. CLKX is derived according to the CLKXM bit.		
7	SCLKME		Sample-rate generator input-clock mode bit. The sample rate generator can produce a clock signal, CLKG. The frequency of CLKG is: CLKG freq. = (Input clock frequency) / (CLKGDV + 1) SCLKME is used in conjunction with the CLKSM bit to select the input clock.	R/W	0
		0	The input clock for the sample rate generator is taken from the CLKS pin or from the CPU clock, depending on the value of the CLKSM bit of SRGR2:		
			SCLKME / **CLKSM** / **Input Clock For Sample Rate Generator** 0 / 0 / Signal on CLKS pin 0 / 1 / CPU clock		
		1	The input clock for the sample rate generator is taken from the CLKR pin or from the CLKX pin, depending on the value of the CLKSM bit of SRGR2:		

Table 12–100. Pin Control Register (PCR) (Continued)

Bit	Name	Value	Description	Type	Reset Value
			SCLKME **CLKSM** **Input Clock For Sample Rate Generator** 1 0 Signal on CLKR pin 1 1 Signal on CLKX pin		
6	CLKSSTAT		CLKS pin status bit. When CLKSSTAT is applicable, it reflects the level on the CLKS pin. CLKSSTAT is applicable only when the transmitter and the receiver are both in reset (XRST = RRST = 0), and CLKS is configured for use as a general-purpose input pin (XIOEN = RIOEN = 1).	R	0
		0	The signal on the CLKS pin is low.		
		1	The signal on the CLKS pin is high.		
5	DXSTAT		DX pin status bit. When DXSTAT is applicable, you can toggle the signal on DX by writing to DXSTAT. DXSTAT is applicable only when the transmitter is in reset (XRST = 0) and DX is configured for use as a general purpose output pin (XIOEN = 1).	R/W	0
		0	Drive the signal on the DX pin low.		
		1	Drive the signal on the DX pin high.		
4	DRSTAT		DR pin status bit. When DRSTAT is applicable, it reflects the level on the DR pin. DRSTAT is applicable only when the receiver is in reset (RRST = 0) and DR is configured for use as a general-purpose input pin (RIOEN = 1).	R	0
		0	The signal on DR pin is low.		
		1	The signal on DR pin is high.		
3	FSXP		Transmit frame-synchronization polarity bit. FSXP determines the polarity of FSX as seen on the FSX pin.	R/W	0
		0	Transmit frame-synchronization pulses are active high.		
		1	Transmit frame-synchronization pulses are active low.		
2	FSRP		Receive frame-synchronization polarity bit. FSRP determines the polarity of FSR as seen on the FSR pin.	R/W	0
		0	Receive frame-synchronization pulses are active high.		
		1	Receive frame-synchronization pulses are active low.		
1	CLKXP		Transmit clock polarity bit. CLKXP determines the polarity of CLKX as seen on the CLKX pin.	R/W	0

Table 12–100. Pin Control Register (PCR) (Continued)

Bit	Name	Value	Description	Type	Reset Value
		0	Transmit data is sampled on the rising edge of CLKX.		
		1	Transmit data is sampled on the falling edge of CLKX.		
0	CLKRP		Receive clock polarity bit. CLKRP determines the polarity of CLKR as seen on the CLKR pin.	R/W	0
		0	Receive data is sampled on the falling edge of CLKR.		
		1	Receive data is sampled on the rising edge of CLKR.		

Table 12–101. Bit Configuration for GPIOs

Pin	General-Purpose Use Enabled by This Bit Combination	Selected as Output When …	Output Value Driven From This Bit	Selected as Input When …	Input Value Read From This Bit
CLKX	XRST = 0 XIOEN = 1	CLKXM = 1	CLKXP	CLKXM = 0	CLKXP
FSX	XRST = 0 XIOEN = 1	FSXM = 1	FSXP	FSXM = 0	FSXP
DX	XRST = 0 XIOEN = 1	Always	DXSTAT	Never	Does not apply
CLKR	RRST = 0 RIOEN = 1	CLKRM = 1	CLKRP	CLKRM = 0	CLKRP
FSR	RRST = 0 RIOEN = 1	FSRM = 1	FSRP	FSRM = 0	FSRP
DR	RRST = 0 RIOEN = 1	Never	Does not apply	Always	DRSTAT
CLKS	RRST = XRST = 0 RIOEN = XIOEN = 1	Never	Does not apply	Always	CLKSSTAT

12.13.9 Receive Channel Enable Registers (RCERA, RCERB, RCERC, RCERD, RCERE, RCERF, RCERG, RCERH)

Each McBSP has eight receive channel enable registers of the format shown in Figure 12–72. There is one enable register for each of the receive partitions: A, B, C, D, E, F, G, and H. Table 12–102 provides a summary description that applies to any bit x of a receive channel enable register.

These I/O-mapped registers are used only when the receiver is configured to allow individual enabling and disabling of the channels (RMCM = 1).

Figure 12–72. Receive Channel Enable Registers (RCERA...RCERH)

15	14	13	12	11	10	9	8
RCE15	RCE14	RCE13	RCE12	RCE11	RCE10	RCE9	RCE8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
RCE7	RCE6	RCE5	RCE4	RCE3	RCE2	RCE1	RCE0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = read; W = write; *-n* = value after reset

Table 12–102. Receive Channel Enable Registers (RCERA...RCERH)

Bit	Name	Value	Description	
x	RCEx		Receive channel enable bit. The role of this bit depends on whether it is used to support the receive multichannel selection mode reception.	
			For receive multichannel selection mode (RMCM = 1):	
		0	Disable the channel that is mapped to RCEx.	
		1	Enable the channel that is mapped to RCEx.	
			RCEx	**Bit x of Incoming Value**
			RCE15	Bit 15: First bit to arrive
			RCE14	Bit 14: Second bit to arrive
			RCE13	Bit 13: Third bit to arrive
			:	:
			RCE0	Bit 0: Last bit to arrive

12.13.9.1 RCERs Used in the Receive Multichannel Selection Mode

For multichannel selection operation, the assignment of channels to the RCERs depends on whether 32 or 128 channels are individually selectable, as defined by the RMCME bit. For each of these two cases, Table 12–103 shows which block of channels is assigned to each of the RCERs used. For each RCER, the table shows which channel is assigned to each of the bits.

Table 12–103. Use of the Receive Channel Enable Registers

Number of Selectable Channels	Block Assignments		Channel Assignments	
	RCERx	**Block Assigned**	**Bit in RCERx**	**Channel Assigned**
32 (RMCME = 0)	RCERA	Channels n to (n + 15) The block of channels is chosen with the RPABLK bits.	RCE0 RCE1 RCE2 : RCE15	Channel n Channel (n + 1) Channel (n + 2) : Channel (n + 15)
	RCERB	Channels m to (m + 15) The block of channels is chosen with the RPBBLK bits.	RCE0 RCE1 RCE2 : RCE15	Channel m Channel (m + 1) Channel (m + 2) : Channel (m + 15)

Table 12–103. Use of the Receive Channel Enable Registers (Continued)

Number of Selectable Channels	Block Assignments		Channel Assignments	
	RCERx	**Block Assigned**	**Bit in RCERx**	**Channel Assigned**
128 (RMCME = 1)	RCERA	Block 0	RCE0	Channel 0
			RCE1	Channel 1
			RCE2	Channel 2
			:	:
			RCE15	Channel 15
	RCERB	Block 1	RCE0	Channel 16
			RCE1	Channel 17
			RCE2	Channel 18
			:	:
			RCE15	Channel 31
	RCERC	Block 2	RCE0	Channel 32
			RCE1	Channel 33
			RCE2	Channel 34
			:	:
			RCE15	Channel 47
	RCERD	Block 3	RCE0	Channel 48
			RCE1	Channel 49
			RCE2	Channel 50
			:	:
			RCE15	Channel 63
	RCERE	Block 4	RCE0	Channel 64
			RCE1	Channel 65
			RCE2	Channel 66
			:	:
			RCE15	Channel 79
	RCERF	Block 5	RCE0	Channel 80
			RCE1	Channel 81
			RCE2	Channel 82
			:	:
			RCE15	Channel 95
	RCERG	Block 6	RCE0	Channel 96
			RCE1	Channel 97
			RCE2	Channel 98
			:	:
			RCE15	Channel 111
	RCERH	Block 7	RCE0	Channel 112
			RCE1	Channel 113
			RCE2	Channel 114
			:	:
			RCE15	Channel 127

12.13.10 Transmit Channel Enable Registers (XCERA, XCERB, XCERC, XCERD, XCERE, XCERF, XCERG, XCERH)

Each McBSP has eight transmit channel enable registers of the form shown in Figure 12–73. There is one for each of the transmit partitions: A, B, C, D, E, F, G, and H. Table 12–104 provides a summary description that applies to each bit XCEx of a transmit channel enable register.

The I/O-mapped XCERs are used only when the transmitter is configured to allow individual disabling/enabling and masking/unmasking of the channels (XMCM is nonzero).

Figure 12–73. Transmit Channel Enable Registers (XCERA...XCERH)

15	14	13	12	11	10	9	8
XCE15	XCE14	XCE13	XCE12	XCE11	XCE10	XCE9	XCE8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
XCE7	XCE6	XCE5	XCE4	XCE3	XCE2	XCE1	XCE0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = read; W = write; -*n* = value after reset

Table 12–104. Transmit Channel Enable Registers (XCERA...XCERH)

Bit	Name	Value	Description
x	XCEx		Transmit channel enable bit. The role of this bit depends on which transmit multichannel selection mode is selected with the XMCM bits.
			For multichannel selection when XMCM = 01b (all channels disabled unless selected):
		0	Disable and mask the channel that is mapped to XCEx.
		1	Enable and unmask the channel that is mapped to XCEx.
			For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):
		0	Mask the channel that is mapped to XCEx.
		1	Unmask the channel that is mapped to XCEx.
			For multichannel selection when XMCM = 11b (all channels masked unless selected):
		0	Mask the channel that is mapped to XCEx. Even if the channel is enabled by the corresponding receive channel enable bit, this channel's data cannot appear on the DX pin.
		1	Unmask the channel that is mapped to XCEx. If the channel is also enabled by the corresponding receive channel enable bit, full transmission can occur.
			XCEx — **Bit x of DXR1 Value**
			XCE15 — Bit 15 (MSB) in DXR1
			XCE14 — Bit 14 in DXR1
			XCE13 — Bit 13 in DXR1
			: — :
			XCE0 — Bit 0 (LSB) in DXR1

12.13.10.1 XCERs Used in a Transmit Multichannel Selection Mode

For multichannel selection operation, the assignment of channels to the XCERs depends on whether 32 or 128 channels are individually selectable, as defined by the XMCME bit. These two cases are shown in Table 12–105. The table shows which block of channels is assigned to each XCER that is used. For each XCER, the table shows which channel is assigned to each of the bits.

Note:

When XMCM = 11b (for symmetric transmission and reception), the transmitter uses the receive channel enable registers (RCERs) to enable channels and uses the XCERs to unmask channels for transmission.

Table 12–105. Use of the Transmit Channel Enable Registers in Transmit Multichannel Selection Mode

Number of Selectable Channels	Block Assignments		Channel Assignments	
	XCERx	**Block Assigned**	**Bit in XCERx**	**Channel Assigned**
32 (XMCME = 0)	XCERA	Channels n to (n + 15) When XMCM = 01b or 10b, the block of channels is chosen with the XPABLK bits. When XMCM = 11b, the block is chosen with the RPABLK bits.	XCE0 XCE1 XCE2 : XCE15	Channel n Channel (n + 1) Channel (n + 2) : Channel (n + 15)
	XCERB	Channels m to (m + 15) When XMCM = 01b or 10b, the block of channels is chosen with the XPBBLK bits. When XMCM = 11b, the block is chosen with the RPBBLK bits.	XCE0 XCE1 XCE2 : XCE15	Channel m Channel (m + 1) Channel (m + 2) : Channel (m + 15)
128 (XMCME = 1)	XCERA	Block 0	XCE0 XCE1 XCE2 : XCE15	Channel 0 Channel 1 Channel 2 : Channel 15
	XCERB	Block 1	XCE0 XCE1 XCE2 : XCE15	Channel 16 Channel 17 Channel 18 : Channel 31
	XCERC	Block 2	XCE0 XCE1 XCE2 : XCE15	Channel 32 Channel 33 Channel 34 : Channel 47
	XCERD	Block 3	XCE0 XCE1 XCE2 : XCE15	Channel 48 Channel 49 Channel 50 : Channel 63
	XCERE	Block 4	XCE0 XCE1 XCE2 : XCE15	Channel 64 Channel 65 Channel 66 : Channel 79

Table 12–105. Use of the Transmit Channel Enable Registers in a Transmit Multichannel Selection Mode (Continued)

Number of Selectable Channels	Block Assignments		Channel Assignments	
	XCERx	**Block Assigned**	**Bit in XCERx**	**Channel Assigned**
	XCERF	Block 5	XCE0	Channel 80
			XCE1	Channel 81
			XCE2	Channel 82
			:	:
			XCE15	Channel 95
	XCERG	Block 6	XCE0	Channel 96
			XCE1	Channel 97
			XCE2	Channel 98
			:	:
			XCE15	Channel 111
	XCERH	Block 7	XCE0	Channel 112
			XCE1	Channel 113
			XCE2	Channel 114
			:	:
			XCE15	Channel 127

12.14 McBSP Register Worksheet

This register worksheet is meant to be printed and used as a guide for configuring the McBSP registers. Each figure on the worksheet provides space in every register field for entering the binary value that must be loaded into that field. For read-only fields, you can use zeros or ones. When all of the fields have been filled in, you can use the line above the register figure to record the corresponding hexadecimal value to load into the register during initialization.

12.14.1 General Control Registers

SPCR1 – Initialization Value: _____

15	14–13	12–11	10–8
DLB	RJUST	CLKSTP	Reserved
			Read-only

7	6	5–4	3	2	1	0
DXENA	Reserved	RINTM	RSYNCERR	RFULL	RRDY	$\overline{\text{RRST}}$
				Read-only	Read-only	

SPCR2 – Initialization Value: _____

15–10	9	8
Reserved	FREE	SOFT
Read-only		

7	6	5–4	3	2	1	0
$\overline{\text{GRST}}$	$\overline{\text{GRST}}$	XINTM	XSYNCERR	XEMPTY_	XRDY	$\overline{\text{XRST}}$
				Read-only	Read-only	

PCR – Initialization Value: _____

15	14	13	12	11	10	9	8
Reserved	IDLE_EN	XIOEN	RIOEN	FSXM	FSRM	CLKXM	CLKRM

Read-only

7	6	5	4	3	2	1	0
SCLKME	CLKS_STAT	DX_STAT	DR_STAT	FSXP	FSRP	CLKXP	CLKRP

Read-only Read-only

RCR1 – Initialization Value: _____

15	14–8	7–5	4–0
Reserved	RFRLEN1	RWDLEN1	Reserved

Read-only Read-only

RCR2 – Initialization Value: _____

15	14–8	7–5	4–3	2	1–0
RPHASE	RFRLEN2	RWDLEN2	RCOMPAND	RFIG	RDATDLY

XCR1 – Initialization Value: _____

15	14–8	7–5	4–0
Reserved	XFRLEN1	XWDLEN1	Reserved

Read-only Read-only

XCR2 – Initialization Value: _____

15	14–8	7–5	4–3	2	1–0
XPHASE	XFRLEN2	XWDLEN2	XCOMPAND	XFIG	XDATDLY

SRGR1 – Initialization Value: _____

15–8	7–0
FWID	CLKGDV

SRGR2 – Initialization Value: _____

15	14	13	12	11–0
GSYNC	CLKSP	CLKSM	FSGM	FPER

12.14.2 Multichannel Selection Control Registers

MCR1 – Initialization Value: _____

15–10	9	8–7	6–5	4–2	1	0
Reserved	RMCME	RPBBLK	RPABLK	RCBLK	Reserved	RMCM
Read-only				Read-only	Read-only	

MCR2 – Initialization Value: _____

15–10	9	8–7	6–5	4–2	1–0
Reserved	XMCME	XPBBLK	XPABLK	XCBLK	XMCM
Read-only				Read-only	

RCERA – Initialization Value: _____

15	14	13	12	11	10	9	8
RCEA15	RCEA14	RCEA13	RCEA12	RCEA11	RCEA10	RCEA9	RCEA8
Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____

7	6	5	4	3	2	1	0
RCEA7	RCEA6	RCEA5	RCEA4	RCEA3	RCEA2	RCEA1	RCEA0
Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____

RCERB – Initialization Value: _____

15	14	13	12	11	10	9	8
RCEB15	RCEB14	RCEB13	RCEB12	RCEB11	RCEB10	RCEB9	RCEB8
Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____

7	6	5	4	3	2	1	0
RCEB7	RCEB6	RCEB5	RCEB4	RCEB3	RCEB2	RCEB1	RCEB0
Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____

RCERC – Initialization Value: _____

15	14	13	12	11	10	9	8
RCEC15	RCEC14	RCEC13	RCEC12	RCEC11	RCEC10	RCEC9	RCEC8
Channel 47	Channel 46	Channel 45	Channel 44	Channel 43	Channel 42	Channel 41	Channel 40

7	6	5	4	3	2	1	0
RCEC7	RCEC6	RCEC5	RCEC4	RCEC3	RCEC2	RCEC1	RCEC0
Channel 39	Channel 38	Channel 37	Channel 36	Channel 35	Channel 34	Channel 33	Channel 32

RCERD – Initialization Value: _____

15	14	13	12	11	10	9	8
RCED15	RCED14	RCED13	RCED12	RCED11	RCED10	RCED9	RCED8
Channel 63	Channel 62	Channel 61	Channel 60	Channel 59	Channel 58	Channel 57	Channel 56

7	6	5	4	3	2	1	0
RCED7	RCED6	RCED5	RCED4	RCED3	RCED2	RCED1	RCED0
Channel 55	Channel 54	Channel 53	Channel 52	Channel 51	Channel 50	Channel 49	Channel 48

RCERE – Initialization Value: _____

15	14	13	12	11	10	9	8
RCEE15	RCEE14	RCEE13	RCEE12	RCEE11	RCEE10	RCEE9	RCEE8
Channel 79	Channel 78	Channel 77	Channel 76	Channel 75	Channel 74	Channel 73	Channel 72

7	6	5	4	3	2	1	0
RCEE7	RCEE6	RCEE5	RCEE4	RCEE3	RCEE2	RCEE1	RCEE0
Channel 71	Channel 70	Channel 69	Channel 68	Channel 67	Channel 66	Channel 65	Channel 64

RCERF – Initialization Value: _____

15	14	13	12	11	10	9	8
RCEF15	RCEF14	RCEF13	RCEF12	RCEF11	RCEF10	RCEF9	RCEF8
Channel 95	Channel 94	Channel 93	Channel 92	Channel 91	Channel 90	Channel 89	Channel 88

7	6	5	4	3	2	1	0
RCEF7	RCEF6	RCEF5	RCEF4	RCEF3	RCEF2	RCEF1	RCEF0
Channel 87	Channel 86	Channel 85	Channel 84	Channel 83	Channel 82	Channel 81	Channel 80

RCERG – Initialization Value: _____

15	14	13	12	11	10	9	8
RCEG15	RCEG14	RCEG13	RCEG12	RCEG11	RCEG10	RCEG9	RCEG8
Channel 111	Channel 110	Channel 109	Channel 108	Channel 107	Channel 106	Channel 105	Channel 104

7	6	5	4	3	2	1	0
RCEG7	RCEG6	RCEG5	RCEG4	RCEG3	RCEG2	RCEG1	RCEG0
Channel 103	Channel 102	Channel 101	Channel 100	Channel 99	Channel 98	Channel 97	Channel 96

RCERH – Initialization Value: _____

15	14	13	12	11	10	9	8
RCEH15	RCEH14	RCEH13	RCEH12	RCEH11	RCEH10	RCEH9	RCEH8
Channel 127	Channel 126	Channel 125	Channel 124	Channel 123	Channel 122	Channel 121	Channel 120

7	6	5	4	3	2	1	0
RCEH7	RCEH6	RCEH5	RCEH4	RCEH3	RCEH2	RCEH1	RCEH0
Channel 119	Channel 118	Channel 117	Channel 116	Channel 115	Channel 114	Channel 113	Channel 112

XCERA – Initialization Value: _____

15	14	13	12	11	10	9	8
XCEA15	XCEA14	XCEA13	XCEA12	XCEA11	XCEA10	XCEA9	XCEA8
Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____

7	6	5	4	3	2	1	0
XCEA7	XCEA6	XCEA5	XCEA4	XCEA3	XCEA2	XCEA1	XCEA0
Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____

XCERB – Initialization Value: _____

15	14	13	12	11	10	9	8
XCEB15	XCEB14	XCEB13	XCEB12	XCEB11	XCEB10	XCEB9	XCEB8
Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____

7	6	5	4	3	2	1	0
XCEB7	XCEB6	XCEB5	XCEB4	XCEB3	XCEB2	XCEB1	XCEB0
Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____	Channel _____

XCERC – Initialization Value: _____

15	14	13	12	11	10	9	8
XCEC15	XCEC14	XCEC13	XCEC12	XCEC11	XCEC10	XCEC9	XCEC8
Channel 47	Channel 46	Channel 45	Channel 44	Channel 43	Channel 42	Channel 41	Channel 40

7	6	5	4	3	2	1	0
XCEC7	XCEC6	XCEC5	XCEC4	XCEC3	XCEC2	XCEC1	XCEC0
Channel 39	Channel 38	Channel 37	Channel 36	Channel 35	Channel 34	Channel 33	Channel 32

XCERD – Initialization Value: _____

15	14	13	12	11	10	9	8
XCED15	XCED14	XCED13	XCED12	XCED11	XCED10	XCED9	XCED8
Channel 63	Channel 62	Channel 61	Channel 60	Channel 59	Channel 58	Channel 57	Channel 56

7	6	5	4	3	2	1	0
XCED7	XCED6	XCED5	XCED4	XCED3	XCED2	XCED1	XCED0
Channel 55	Channel 54	Channel 53	Channel 52	Channel 51	Channel 50	Channel 49	Channel 48

XCERE – Initialization Value: _____

15	14	13	12	11	10	9	8
XCEE15	XCEE14	XCEE13	XCEE12	XCEE11	XCEE10	XCEE9	XCEE8
Channel 79	Channel 78	Channel 77	Channel 76	Channel 75	Channel 74	Channel 73	Channel 72

7	6	5	4	3	2	1	0
XCEE7	XCEE6	XCEE5	XCEE4	XCEE3	XCEE2	XCEE1	XCEE0
Channel 71	Channel 70	Channel 69	Channel 68	Channel 67	Channel 66	Channel 65	Channel 64

XCERF – Initialization Value: _____

15	14	13	12	11	10	9	8
XCEF15	XCEF14	XCEF13	XCEF12	XCEF11	XCEF10	XCEF9	XCEF8
Channel 95	Channel 94	Channel 93	Channel 92	Channel 91	Channel 90	Channel 89	Channel 88

7	6	5	4	3	2	1	0
XCEF7	XCEF6	XCEF5	XCEF4	XCEF3	XCEF2	XCEF1	XCEF0
Channel 87	Channel 86	Channel 85	Channel 84	Channel 83	Channel 82	Channel 81	Channel 80

XCERG – Initialization Value: _____

15	14	13	12	11	10	9	8
XCEG15	XCEG14	XCEG13	XCEG12	XCEG11	XCEG10	XCEG9	XCEG8
Channel 111	Channel 110	Channel 109	Channel 108	Channel 107	Channel 106	Channel 105	Channel 104

7	6	5	4	3	2	1	0
XCEG7	XCEG6	XCEG5	XCEG4	XCEG3	XCEG2	XCEG1	XCEG0
Channel 103	Channel 102	Channel 101	Channel 100	Channel 99	Channel 98	Channel 97	Channel 96

XCERH – Initialization Value: _____

15	14	13	12	11	10	9	8
XCEH15	XCEH14	XCEH13	XCEH12	XCEH11	XCEH10	XCEH9	XCEH8
Channel 127	Channel 126	Channel 125	Channel 124	Channel 123	Channel 122	Channel 121	Channel 120

7	6	5	4	3	2	1	0
XCEH7	XCEH6	XCEH5	XCEH4	XCEH3	XCEH2	XCEH1	XCEH0
Channel 119	Channel 118	Channel 117	Channel 116	Channel 115	Channel 114	Channel 113	Channel 112

Chapter 13

NAND Flash

This chapter describes the NAND flash memory interface of the OMAP730 multimedia processor.

Topic **Page**

13.1 Hardware NAND Flash Controller

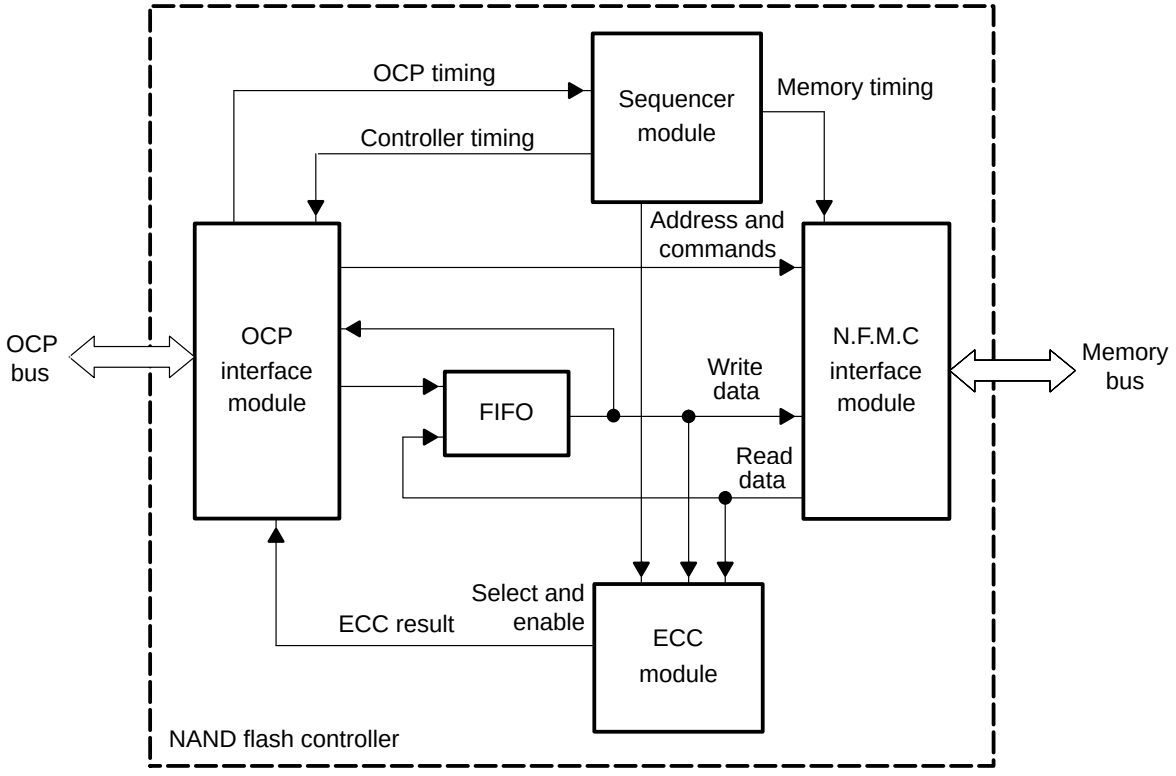
The NAND flash controller (NFC) is the interface between the host processor and the NAND flash memory core (NFMC). The NFC (see Figure 13–1) supports the following NFMC configurations:

- 1G-bit
- 512M-bit
- 256M-bit
- 128M-bit
- 64M-bit
- 32M-bit

The 1G-bit NFMC is a dual-die made with two 512M bits.

See Figure 13–1 for an overview of the NAND flash controller.

Figure 13–1. NAND Flash Controller Overview



The NFMC is an 8-bit interface (byte addressable). From the host processor, the NFC can access data in 8, 16, or 32 bits.

The NFC supports prefetch and postwrite modes implemented around a FIFO buffer for latency optimization.

The NFC supports DMA transfers on a page-in-read and program operations and has the following general features:

- ❏ Flexible architecture to support different sizes of NFMCs
- ❏ 8-bit interface to NAND flash
- ❏ 8-bit, 16-bit, or 32-bit interface from the host processor
- ❏ Error code correction (ECC) for maximum data integrity
- ❏ DMA support on one NAND flash memory page for read and program operations
- ❏ Ability to support from one to four NFMCs

Accessing an NFMC requires an 8-bit bus, which is multiplexed among data, address, and commands. Table 13–1 lists the command operations.

Table 13–1. Command Operations

Function	First Cycle	Second Cycle	Third Cycle
Read 1 (lower half-page or Area A)	0x00	-	-
Read 1 (upper half-page or Area B)	0x01	-	-
Read 2 (spare or Area C)	0x50	-	-
Read ID	0x90	-	-
Reset	0xFF	-	-
Page program	0x80	0x10	-
Multiple-page program	0x80	0x11	-
Copy-back program	0x00	0x8A	0x10
Multiple copy-back program	0x03	0x8A	0x10/0x11
Block erase	0x60	0xD0	-
Multiplane block erase	0x60–0x60	0xD0	-
Read status	0x70	-	-
Read multiplane status	0x71	-	-

Associated with this 8-bit bus are qualifiers: CLE (command latch enable) which characterizes that the data on the bus is a command, and ALE (address latch enable), which characterizes that the value on the bus is an address. When CLE is low and ALE is low, the value on the bus is data. The combination CLE high and ALE high is not permitted. Reading data requires a negative pulse on signal RE_. Writing command/address/data requires a negative pulse on signal WE_.

For program operation, data is latched into the NFMC on the rising edge of WE_. For read operation, data is latched on the falling edge of RE_. The NFMC signal Ready/Busy_ (R/B_) indicates the status of the ongoing operation.

When low, R/B_ indicates that a program read or erase is in process and returns to high state upon completion. When returning high, the NFC also asserts an interrupt so that the local host can be aware that the ongoing operation is completed.

The core-memory check page reference inside the core is organized by a page of 512 bytes and a spare area of 16 bytes for the most recent NFMC, as shown in Figure 13–18, *Single Page Program DMA in Postwrite Mode*.

The 528-byte page is itself divided into two main areas: A (from 0 to 255) and B (from 256 to 511), and a C, or spare, area (from 512 to 527). To access area A, the command 0x00 is sent to the flash core. To access area B, the command 0x01 is sent. To access the spare area, the command 0x50 is sent (see Table 13–2).

Table 13–2. Pointer Operation

Command	Accessed Address	Area
0x00	0-255	Area A (lower half-page)
0x01	256-511	Area B (upper half-page)
0x50	512-527	Area C (spare)

The erase operation performs on a block basis. For the most recent NFMCs, one block equals 32 pages, or 16K bytes. On 512M-bit and 1G-bit NFMCs, there is also a logical array division by plane:

- The 512M-bit has four planes, each with 1024 blocks of 32 pages per block.
- The 1G-bit has eight planes, each with 1024 blocks of 32 pages per block.

To be generic, most control must be done in software. Commands, for instance, are not hard-coded because they can change in different versions of NFMC, or new commands can be added.

There are two types of commands: one must be followed by an address (for instance, read, program, and erase); the other does not need to be followed by an address (for instance, read status, end of data (0x10 or 0x11) in the case of program operation). An address register is necessary to store the starting address (32 bits). An access to the first type of command register places the command on the 8-bit NFMC bus and the data located in the address register also is driven on the bus. Writing data in the second type of command register does not issue an access to the address register. For read operation, after the address is transmitted to the NFMC, there is a latency time to wait for the data to be ready (typically 12 μs). The wait can be done either by an interrupt or by polling the R/B_.

13.1.1 Read Operation

First, the command 0x00, 0x01 or 0x50 is driven on the bus with the qualifier CLE being high. Then the start address is driven byte-by-byte with the least significant byte first, with the qualifier ALE being high. The NFMC drives its Ready/Busy_ (R/B_) to low. When R/B_ goes back to high, an interrupt is asserted and data is ready to be sampled by negative pulse on RE_. An access on the NAND controller access register (NND_ACCESS) triggers the negative pulse on RE_ and data is read and stored in the NAND controller access register (NND_ACCESS). The interrupt bit must be cleared by software.

Another solution to waiting for data to be ready is to poll the ready bit in the NAND controller ready register (NND_READY). This bit is the sampled R/B_ pin of the NFMC. The read status command can also be sent and the NFMC status register tested. In that case, bit 6 of the NFMC status register reflects the R/B_ bit, as shown in Table 13–7 and Table 13–8. When R/B_ returns to 1, data is ready.

If the readiness of the NFMC was tested by accessing the NFMC status register, the command code 0x00 must be sent because the NFMC stays in read status mode when no new command is sent.

Reading 32 (16) bits is possible because it is decomposed on 4 (2) 8-bit access. Returned data is preserved in a 4 x 8-bit buffer until the current operation is completed. The data is then latched in a register in the NFC before being driven on the output sdata bus. When no operation is selected, the NFC drives 0x00000000 to the sdata bus (see Table 13–3).

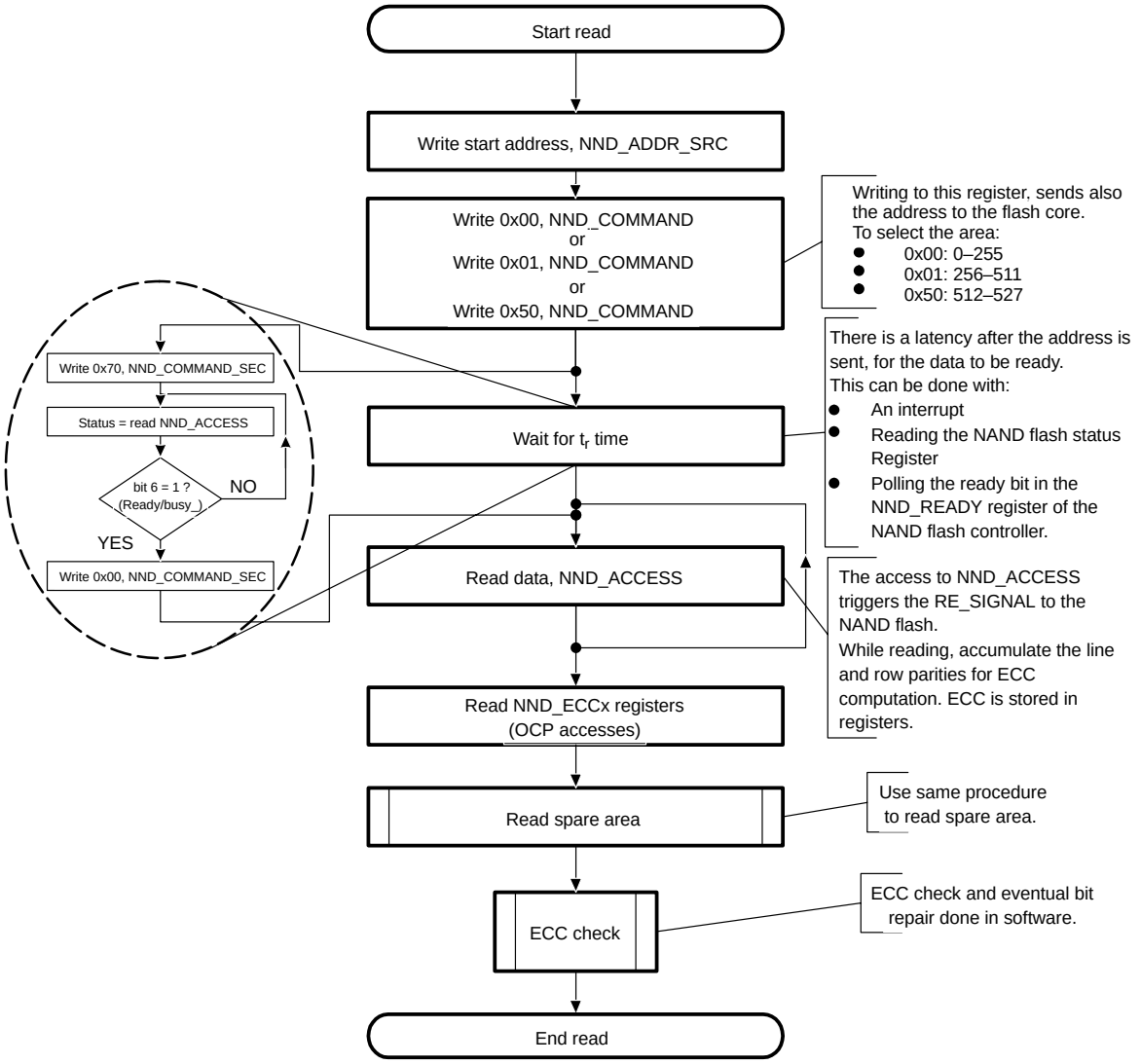
Table 13–3. Sent Address Function of Core Sizes

Size of Flash Core (Mega Bits)	Address[31-24]	Address[23-16]	Address[15-8]	Address[7-0]
32	00000000	000-A21-A17	A16-A9	A7-A0
64	00000000	00-A22-A17	A16-A9	A7-A0
128	00000000	0-A23-A17	A16-A9	A7-A0
256	00000000	A24-A17	A16-A9	A7-A0
512	0000000-A25	A24-A17	A16-A9	A7-A0
1024	000000-A26-A25	A24-A17	A16-A9	A7-A0

The pointer 0x00, 0x01, or 0x50 selects which area to access. For instance, to access data at address 256 in the 528-byte page of the NAND flash, the command 0x01 must be sent with an address of 0x00. When the bit A8 of the NND_CTRL of the NFC is set to 0, the NFC does not send the bit 8 of the NND_ADDR_SRC register to the NFMC The address that is written in the NND_ADDR_SRC must be formatted with a dummy bit at location 8 of the address because the NFC hardware does not send this bit to the NFMC.

The flow chart shown in Figure 13–2 shows how to read one or multiple bytes in a page of the NFMC.

Figure 13–2. Read Operation

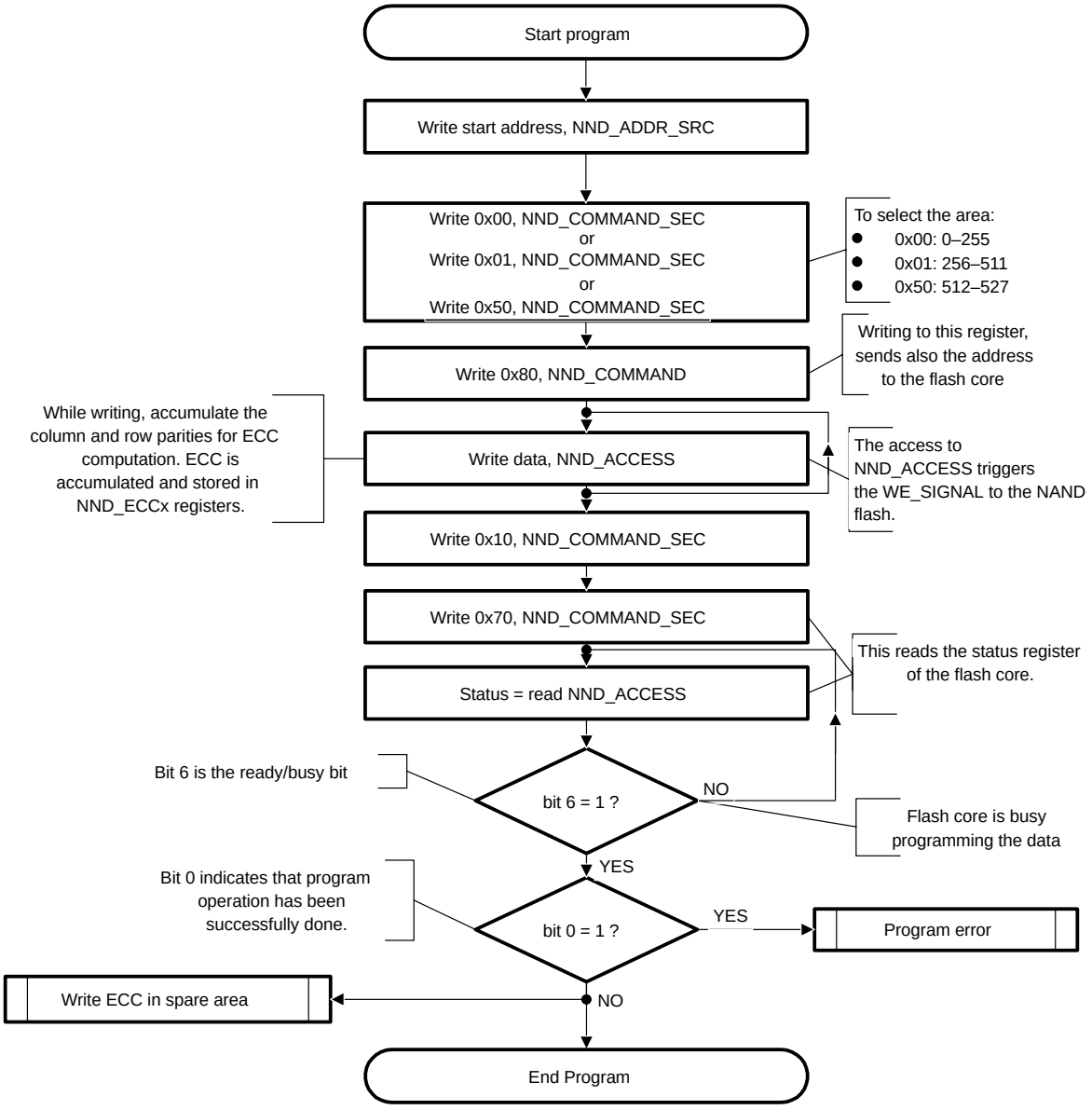


13.1.2 Write Operation

First, the command 0x00 or 0x01 or 0x50 is driven on the bus with qualifier CLE high. This is to select which area (A, B, or C) to program. Then, the command 0x80 is driven on the bus with the qualifier CLE being high. Last, the start address is driven byte-by-byte, least significant byte first, with the qualifier ALE being high (see Table 13–3).

Writing data in the NAND controller access register (NND_ACCESS) places the data on the bus and asserts a negative pulse on WE_ (CLE and ALE are low). After the last data has been driven, a command 0x10 is sent to terminate the flow of data. The NFMC drives R/B_ to low during the programming of NFMC (from its internal page register to memory cell) and R/B_ goes back to high, asserting an interrupt. This interrupt bit is cleared by software using the scheme shown in Table 13–30. The NFMC is now ready for the next operation. The result of the operation can be checked with a read status operation (sending the command 0x70 or 0x71) to the NFMC

Figure 13–3. Write Operation



13.1.3 Multiplane Page Program

The multiplane page program is an extension of the page program operation in which up to four pages (and for each page, up to 528 bytes) are programmed. This operation is available only on 512M-bit and 1G-bit NFMCs.

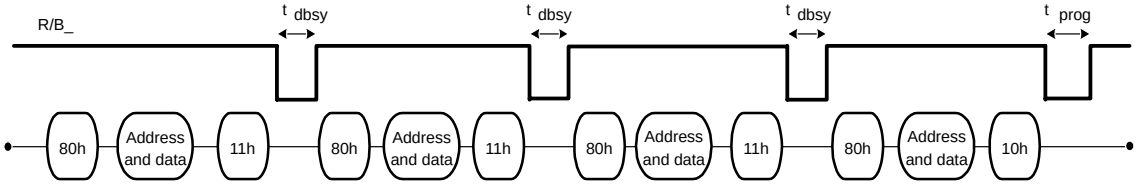
For the 1G-bit NFMC, multiplane page programming accesses pages in planes 0, 1, 2, and 3 or planes 4, 5, 6, and 7.

13.1.3.1 Restriction in Addressing With Multiplane Page Program

Although any block in each plane can be addressable for the multiplane page program, the page address in the selected block must be the same. This means that, given four addresses (one for each plane), bits 9 to 13 of those addresses must be the same because bits 9 to 13 select the address of the page in a block (see Figure 13–4).

It is impossible to start loading data in area B (upper half-page). The NFC does not check the validity of addresses.

Figure 13–4. Multi-Page Program Operation



The command to mark the end of data is 0x10 for the last operation and 0x11 for the first accesses. If the program operation fails, the software takes the responsibility to program the data in another page and mark the block (where the page belongs) as invalid. After each address and data sent, there is a latency time (t_{dbsy}). The actual programming of the flash occurs after the last address and data (t_{prog}). Typically, t_{dbsy} and t_{prog} are, respectively ,1 µs and 200 µs.

13.1.4 Erase Operation

The erase operation (see Figure 13–5) is possible only at block level. Command 0x60 is driven with CLE high, then the address of the block to erase is driven with ALE high, and the command 0xD0 is written with CLE high. During the physical erase, the flash core drives R/B_ to low. When the erase operation is completed, R/B_ returns to high. The command is repeatable up to four times (depending on the NFMC, because this operation is valid on 1G bit and 512M bits, as shown in Table 13–13, *Supported Operations on NFMCs.)* The status of the erase operation can be checked with a read status operation (command 0x70 or 0x71).

Table 13–4.Programming Address for Erase Operation

Size of Flash Core (MB)	Block Address (sent)	Address Page in Block (sent but ignored)	Start Address (not sent)	Erasable Block Size
32	A21-A13	A12-A9	A7-A0	8K
64	A22-A13	A12-A9	A7-A0	8K
128	A23-A14	A13-A9	A7-A0	16K
256	A24-A14	A13-A9	A7-A0	16K
512	A25-A14	A13-A9	A7-A0	16K
1024	A26-A14	A13-A9	A7-A0	16K

The address of the erase block must be properly formatted in the address register NN_ADDR_SRC. Bits that correspond to the address page in the selected block must be sent, but they are discarded by the internal logic of the NFMC (see Table 13–5).

Table 13–5. Formatting of Erase Address in Address Register

Size of Flash Core (MB)	Address[31-24]	Address[23-16]	Address[15-8]	Address[7-0]
32	00000000	00000000	000-A21-A17	A16-A9
64	00000000	00000000	00-A22-A17	A16-A9
128	00000000	00000000	0-A23-A17	A16-A9
256	00000000	00000000	A24-A17	A16-A9
512	00000000	0000000-A25	A24-A17	A16-A9
1024	00000000	000000-A26-A25	A24-A17	A16-A9

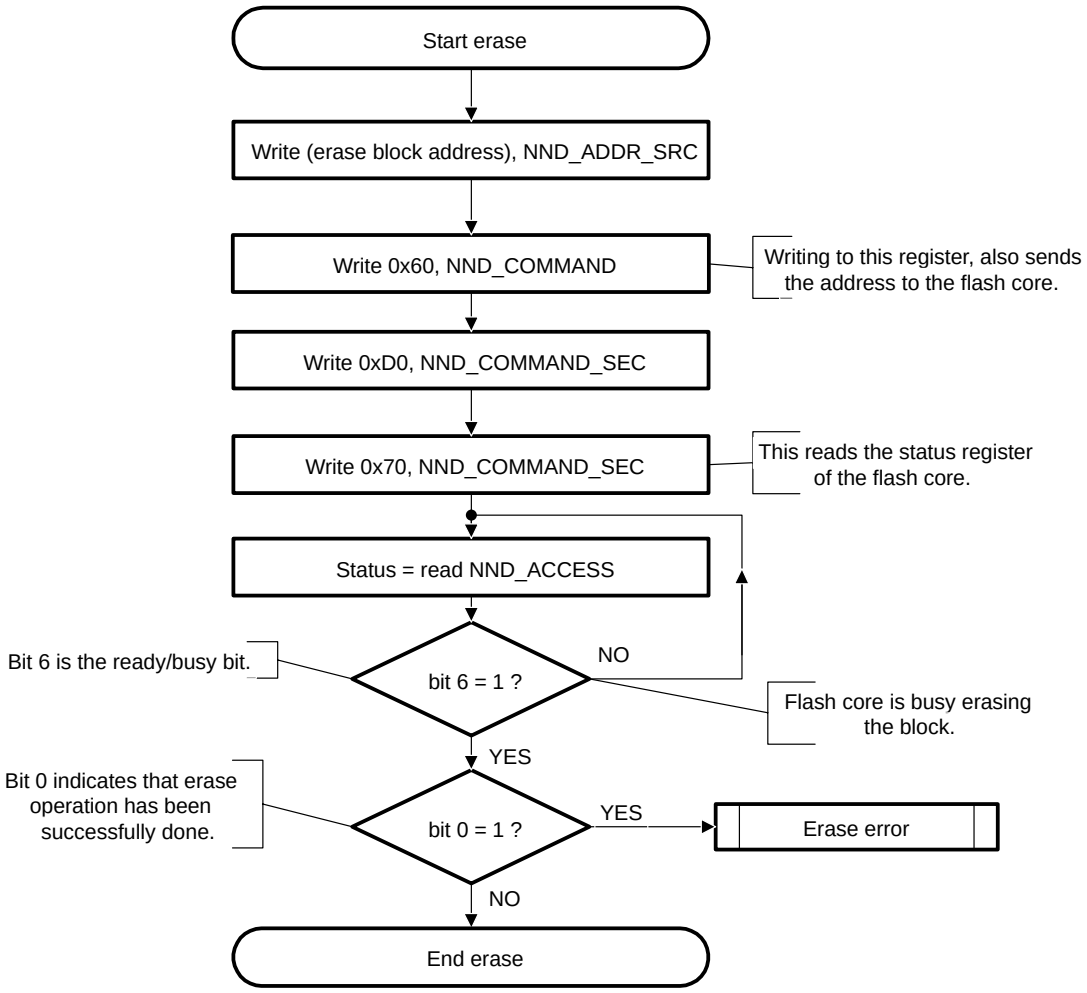
If the erase operation fails, software must mark the block as invalid by writing a value different from 0xFF in the invalid block location (6th byte) in the spare area.

If bit A8 of NND_CTRL is 0, then the erase address must be formatted with a dummy bit at location 8 of the erase address, because the NFC does not send the bit 8 to the NFMC (see Table 13–6).

Table 13–6. Formatting of Erase Address in Address Register With Bit A8 Not Sent

Size of Flash Core (MB)	Address[31-24]	Address[23-16]	Address[15-8]	Address[7-0]
32	00000000	00000000	00-A21-A17-**0**	A16-A9
64	00000000	00000000	0-A22-A17-**0**	A16-A9
128	00000000	00000000	A23-A17-**0**	A16-A9
256	00000000	0000000-A24	A23-A17-**0**	A16-A9
512	00000000	000000-A25-A24	A23-A17-**0**	A16-A9
1024	00000000	00000-A26-A24	A23-A17-**0**	A16-A9

Figure 13–5. Erase Operation

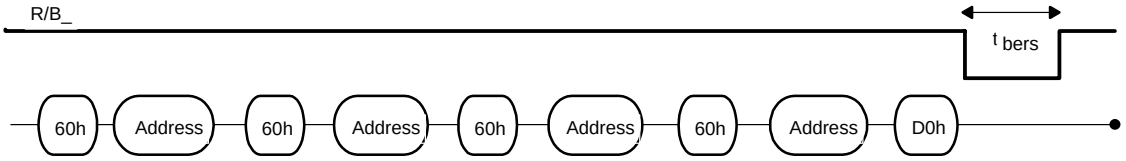


13.1.5 Multiplane Block Erase Operation

The multiplane block erase operation is identical to the multiplane page program. Up to four blocks, one from each plane, can be simultaneously erased. Standard block erase command sequences are repeated up to four times for erasing. Only one block must be selected from each plane. The NFC does not check the validity of these erase addresses (see Figure 13–6).

The erase confirm command (0xD0) initiates the erasing process and R/B_ goes low for t_{bers} time (typically 2 ms). The completion is detected by testing bit 6 (Ready/Busy_) of the NFMC status register. When the erase is completed, the pass/fail status of each block is examined by reading the extended pass/fail status of the NFMC status register (bit 1 to bit 4) using the read multiplane status command (0x71). If not masked, an interrupt is asserted when R/B_ returns to high state.

Figure 13–6. Multiplane Block Erase Operation



13.1.5.1 Copy-Back Program Operation

The copy-back program quickly and efficiently rewrites data stored in one page within the plane to another page within the same plane, without using an external memory. Because the time-consuming sequentially reading and reloading cycles are removed, system performance is improved. A normal read operation with 0x00 command and the address of the source page moves the whole 528-byte data into an NFMC internal buffer. As soon as the NFMC returns to the ready state, command 0x8A with the address cycles of the destination page is sent to the NFMC. The confirm command (0x10) is required to begin the programming operation.

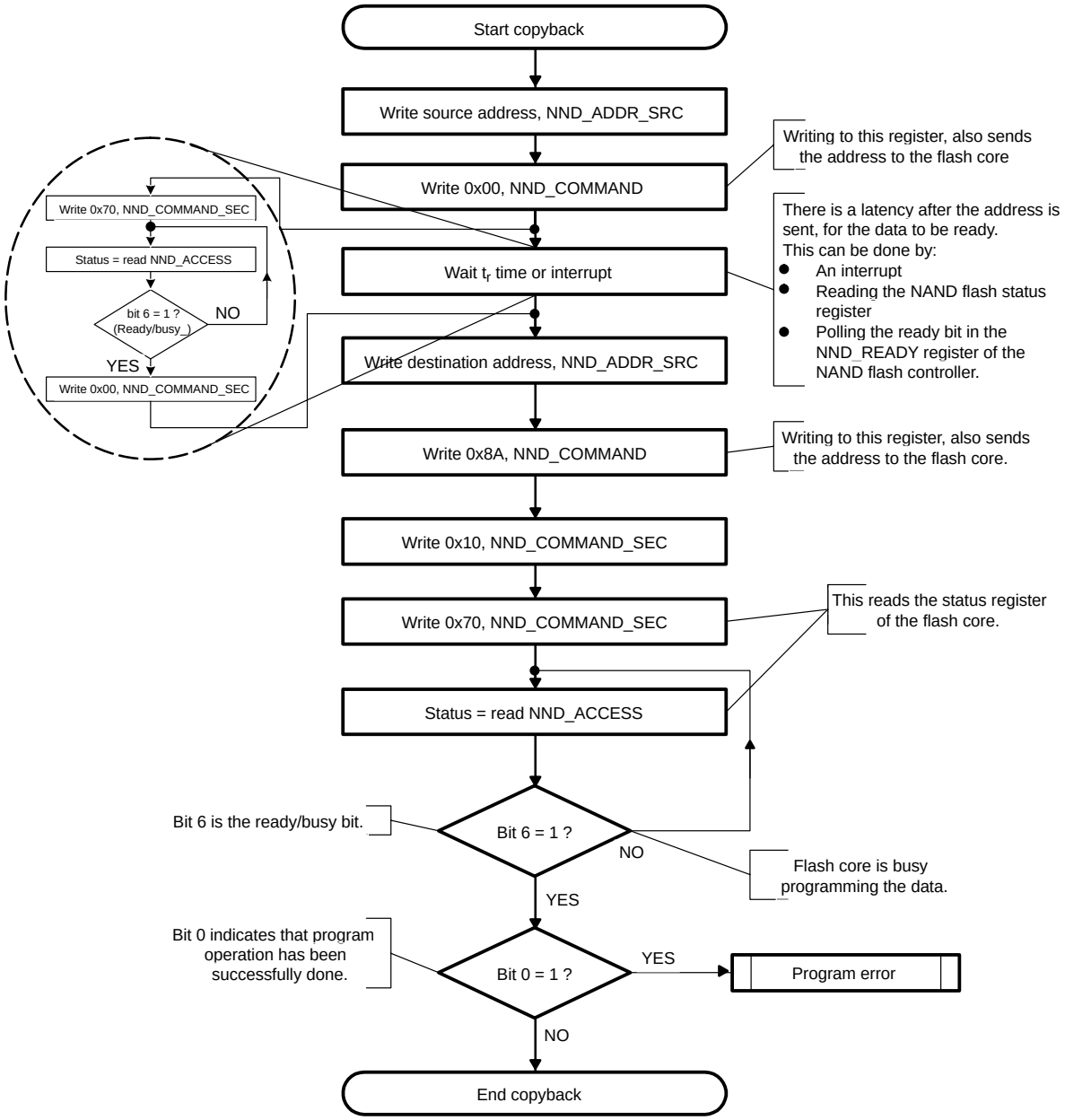
There is some restriction because the copy-back program operation is allowed only within the same memory plane. Although a full address is sent, bits 7 to 0 of the address are discarded. *The software must check the validity of page addresses.*

The source page and the destination page must be in the same plane:

- For 512M-bit, bit 14 and bit 15 of the address must be the same.
- For 1G-bit, bit 14, bit 15, and bit 26 of the address must be the same.

If the operation fails, software marks the block where the page belongs as invalid (see Figure 13–7).

Figure 13–7. Copy-Back Operation



13.1.6 Multiplane Copy-Back Program Operation

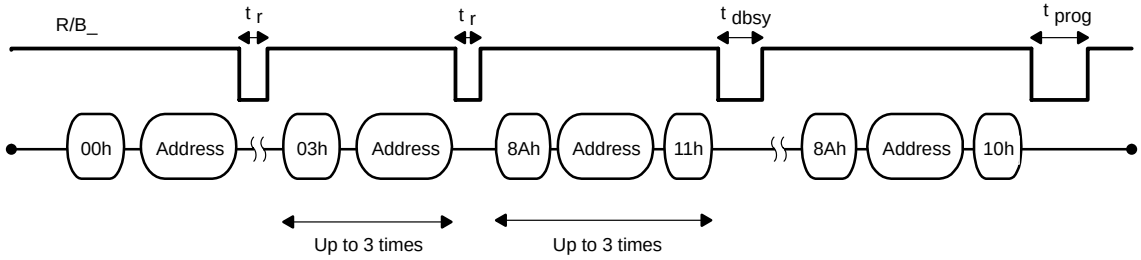
The multiplane copy-back program is an extension of the one-page copy-back program and is available on 512M-bit and 1G-bit NFMCs (see Figure 13–8).

First, the command for read operation (0x00) is sent, followed by the source address. Any further read operation (up to three) for transferring the addressed pages to the corresponding page register must be executed with command code 0x03 instead of a 0x00 command. Up to four pages can be addressed.

There is a latency time Tr (typically 12 μs), indicated by the R/B_ line going low, between each source address. Data is transferred from memory cells to the

NFMC internal buffer. After the input of commands for reading the source pages, the same procedure as multiplane page programming occurs, except that the command operation is 0x8A instead of 0x80. Because no programming process is involved during data loading at the destination plane address, R/B_ remains in a busy state for a short period of time. When the last command operation 0x10 is sent, the programming of the NFMC starts and the NFMC drives its R/B_ signal low. A read status operation (command 0x70 or 0x71) can be issued to find out when the NFMC returns to ready state by polling the READY/BUSY_ bit in the status register (bit 6). Once the multiplane copy-back program is finished, any additional partial-page programming into the copied pages is prohibited before an erase operation. If not masked, an interrupt is asserted each time R/B_ returns to high state. Typically, t_r, t_{dbsy}, and t_{prog} are 12µs, 1µs, and 200µs, respectively (see Figure 13–8).

Figure 13–8. Multiplane Copy-Back Operation



13.1.7 Read Status and Read Multiplane Status Operations

The NFMC has a status register, which can be accessed to determine whether a program or erase operation was completed, and whether the ongoing operation was completed successfully. After writing command 0x70 or 0x71 to the NND_COMMAND_SEC register, an access to the NAND controller access register (NND_ACCESS) drives the content of the NFMC status register on the multiplexed 8-bit bus.

The NFMC remains in read status mode until further commands are issued to it. Therefore, if the status register is read during a random read cycle, a read command (0x00) must be issued before a sequential-page read cycle.

For the read status of a multiplane program or multi-erase operations, the read multiplane status command (71h) must be used rather than 70h. (See Table 13–7 and Table 13–8).

Table 13–7. Status Register Mapping for 32, 64, 128, and 256 Megabits

Bit	Definition	Read Status (70h)
7	Write protect	0: Protected
		1: Not protected
6	Device operation	0: Busy
		1: Ready
5	Reserved	0
4	Reserved	0
3	Reserved	0
2	Reserved	0
1	Reserved	0
0	Program/erase	0: Pass
		1: Fail

Table 13–8. Status Register Mapping for 512, 1024 Megabits

Bit	Definition	Read Status (70h)	Read Multiplane Status (71h)
7	Write protect	0: Protected	0: Protected
		1: Not protected	1: Not protected
6	Device operation	0: Busy	0: Busy
		1: Ready	1: Ready
5	Reserved	0	0
4	Plane 3 pass/fail	0	0: Pass
			1: Fail
3	Plane 2 pass/fail	0	0: Pass
			1: Fail
2	Plane 1 pass/fail	0	0: Pass
			1: Fail
1	Plane 0 pass/fail	0	0: Pass
			1: Fail
0	Program/erase (all operations)	0: Pass	0: Pass
		1: Fail	1: Fail

13.1.8 Reset Operation

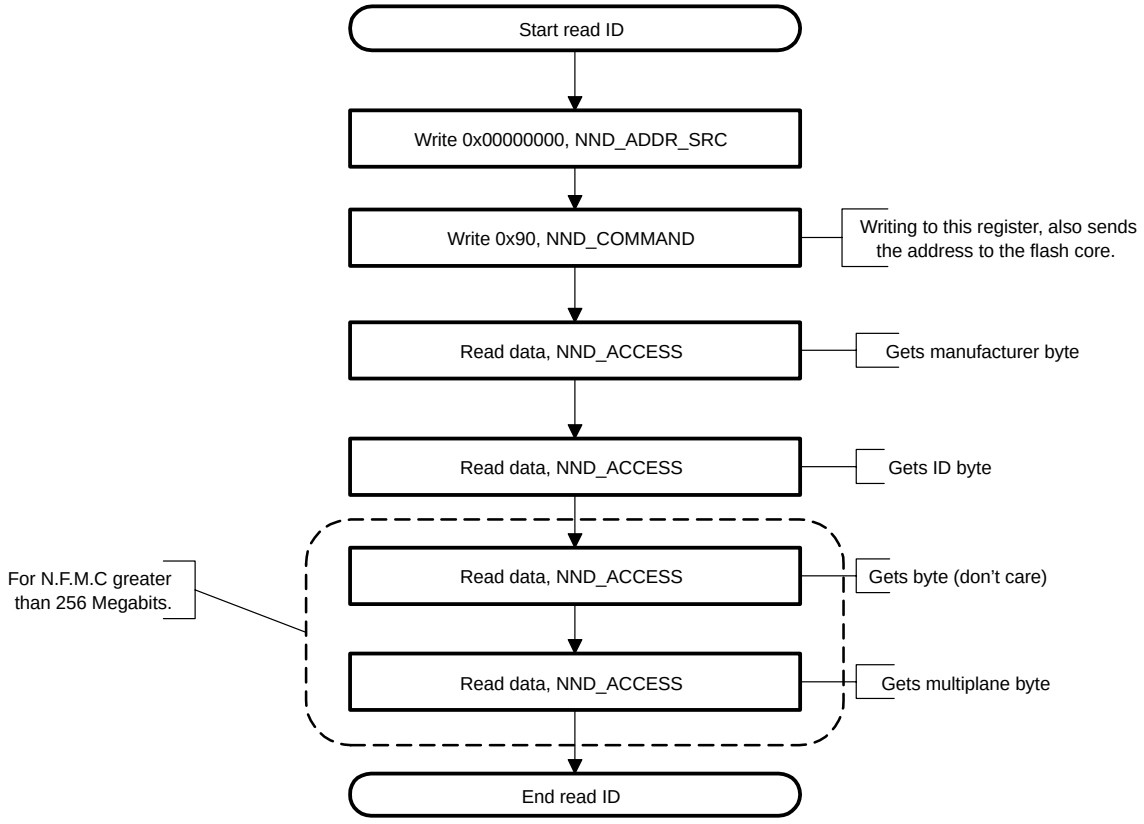
The NFMC has a reset feature executed by writing 0xFF to the command register (NND_COMMAND_SEC) because there is no address to be sent. When the NFMC is in a busy state during the random read program or erase mode, the reset operation aborts these operations. The contents of memory cells being altered are no longer valid, because the data is partially programmed or

erased and cannot be ensured. The command register of the NFMC is cleared to wait for the next command. The R/B_ pin goes to low for a certain time depending on the operation (typically 5, 10, or 500 µs for read/program/erase) after the reset command is written. If the NFMC is already in reset state, the flash-memory command register does not accept a new reset command. It is recommended, after a hardware reset, to start any access to the NFMC by sending a reset command.

13.1.9 Read ID Operation

The flash core has a product identification code. To read this code, the command 0x90 is used, followed by an address input of 00h. Depending on the size of the flash core, the code can be composed of either 2 or 4 bytes (for the 512M-bit and 1G-bit). Reading the NFMC sequentially outputs the NMFC manufacturer byte, the NFMC code or ID byte (a unique byte that represents the size of the NFMC), a reserved byte, and the multiplane byte, respectively. By reading the ID of the NFMC, the software determines which features are supported by the NFMC (see Table 13–12, *Characteristics of Supported NFMCs*). The memory core remains in read ID operation unless a new command code is sent.

Figure 13–9. Read ID Operation



13.1.10 Error Code Correction

To better protect data, the NFC provides an error code correction (ECC) logic. The algorithm for ECC is the one that Samsung recommends (see Figure 13–10). It can be used on 256 bytes or 512 bytes (selectable by a control bit in the NND_CTRL register), and can detect errors and correct one bit error. The ECC logic can be used either for read or program operation.

Bit checking and eventual bit repair is done by software after either a half-page or page read.

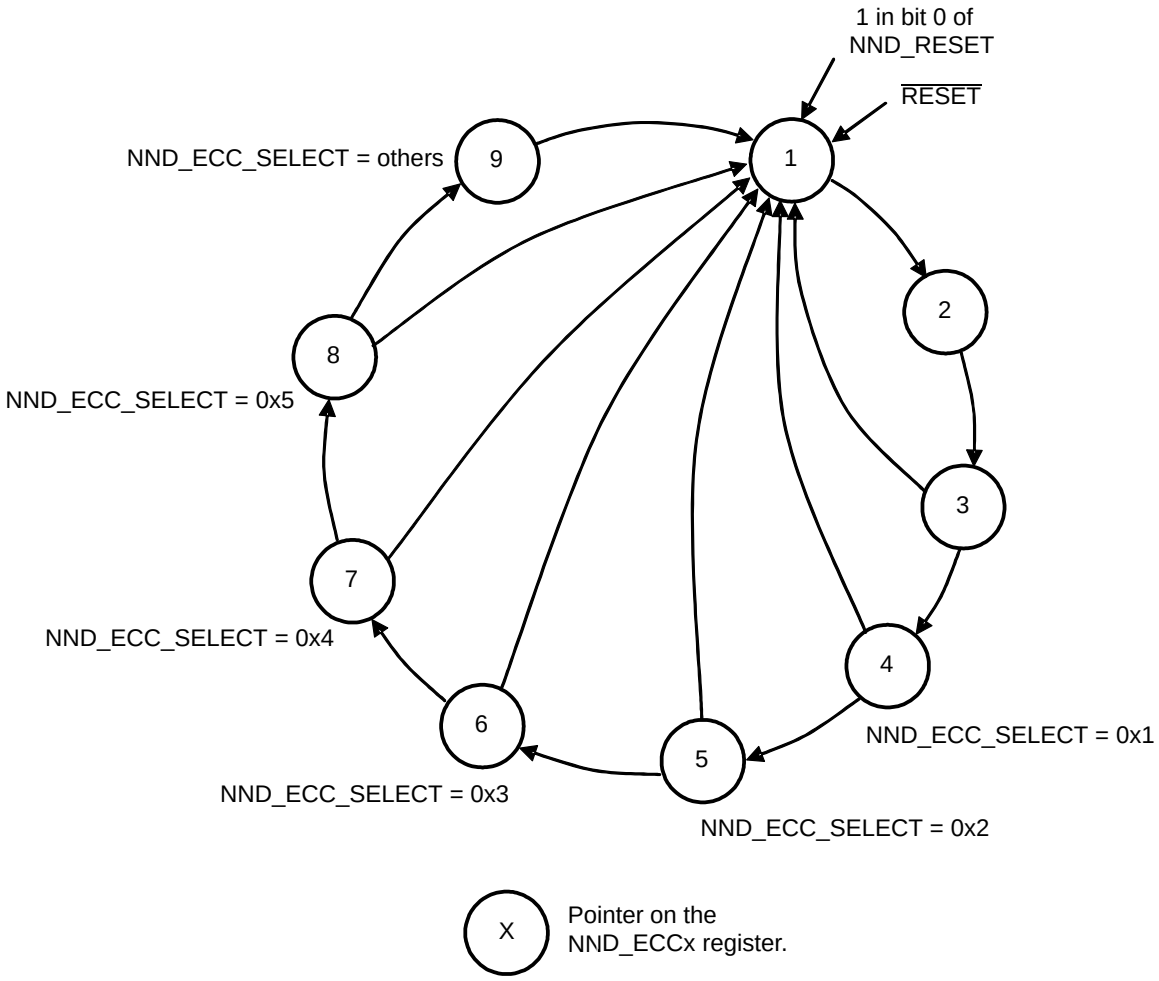
When program operation is used, ECC is usually accumulated. It is the responsibility of the software to read the ECC through dedicated registers (NND_ECCx) via host access and to write them back to the NFMC in the spare area.

For a read operation, the ECC is accumulated after each data read. At least a half-page needs to be accessed, and the starting address is the beginning of the half-page. The calculated ECC can be compared with the one that was previously stored in the spare area when the last program operation in that page was performed. The NFC does not assume the physical address of the ECC bytes in the spare byte area.

ECC can be disabled by resetting a bit in the control register (NND_CTRL) of the NFC.

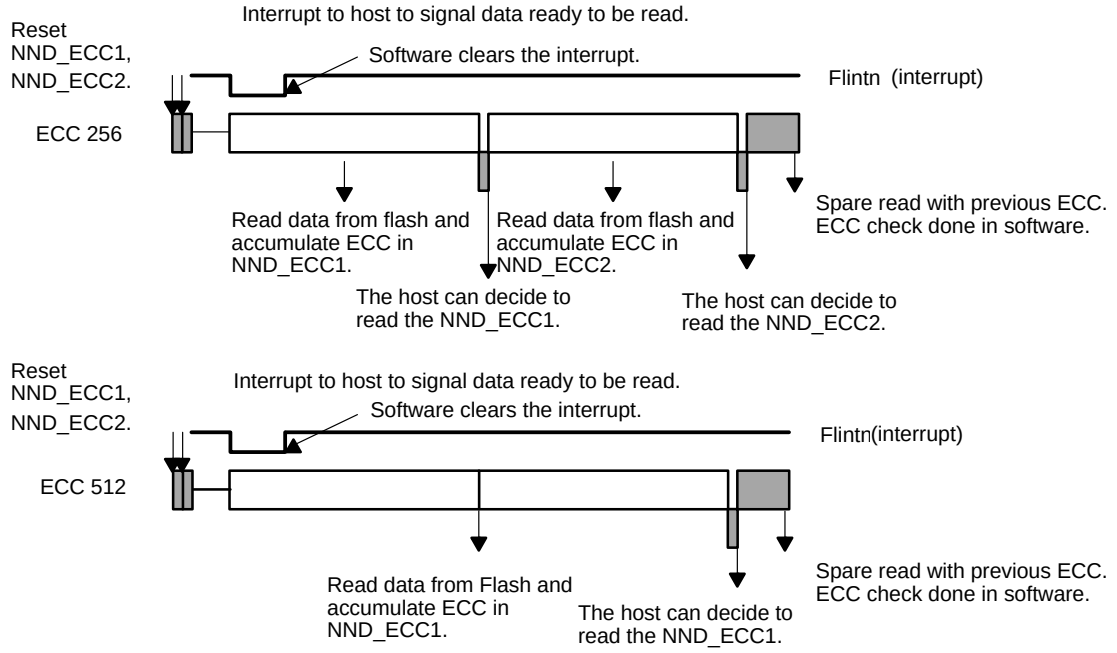
ECC registers are reset when the RESETN pin at the NFC boundary goes low or by writing a bit in the NND_RESET register. The management of the ECC pointer is done automatically by hardware. After a reset, the pointer is set to register 1 (NND_ECC1). Then, when the number of read/writes exceeds the ECC length coded by ECC_256 in the NND_CTRL register of the NFC, the pointer shifts to register 2 (NND_ECC2), next to register 3, and then moves back to the NND_ECC1 register. During ECC computation, if bit ECC_ON is reset, the ECC registers are not updated anymore. When ECC_ON is set to 1, the computation resumes. After a write in the NND_RESET register, the pointer for the ECC is set to NND_ECC1.

Figure 13–10. ECC Pointer Management



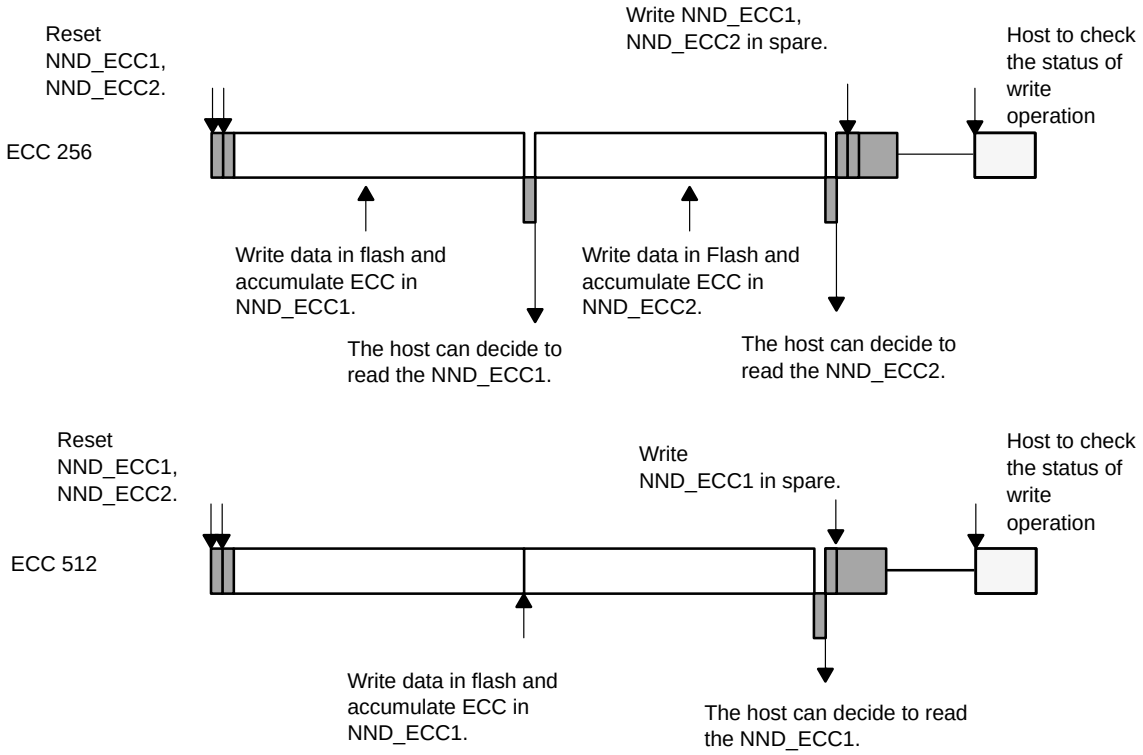
The pointer transition from one state to another is made after either 256 or 512 data has been read (control bit ECC_256 in the NND_STATUS register). After a reset **(**RESETN going low, soft reset, or writing 1 to bit 0 of NND_RESET), the pointer is set to NND_ECC1 register.

Figure 13–11. Single Page Read Host Mode



Note: In the case of ECC 256, the reading of ECC can also be performed when the full page has been read.

Figure 13–12. Single-Page Write Host Mode



Note: In the case of ECC 256, the reading of ECC can also be performed when the full page has been written, and before the ECC registers in the spare area.

It is the responsibility of the software, when program operation is in effect, to read back the ECC registers and to save them in the spare area. The software can also calculate the ECC on a chunk of data. The reading of the ECC registers can be done once a full page has been written.

Once all of the parities have been computed, they are stored in the ECC registers under the mapping shown in Table 13–9.

Table 13–9. Bit Mapping of ECC Registers

Register Name	7	6	5	4	3	2	1	0
NND_ECCx[31:24]	Reserved	Reserved	Reserved	Reserved	P2048o	P1024o	P512o	P256o
NND_ECCx[23:16]	P128o	P64o	P32o	P16o	P8o	P4o	P2o	P1o
NND_ECCx[15:8]	Reserved	Reserved	Reserved	Reserved	P2048e	P1024e	P512e	P256e
NND_ECCx[7:0]	P128e	P64e	P32e	P16e	P8e	P4e	P2e	P1e

13.1.11 Invalid Block Management

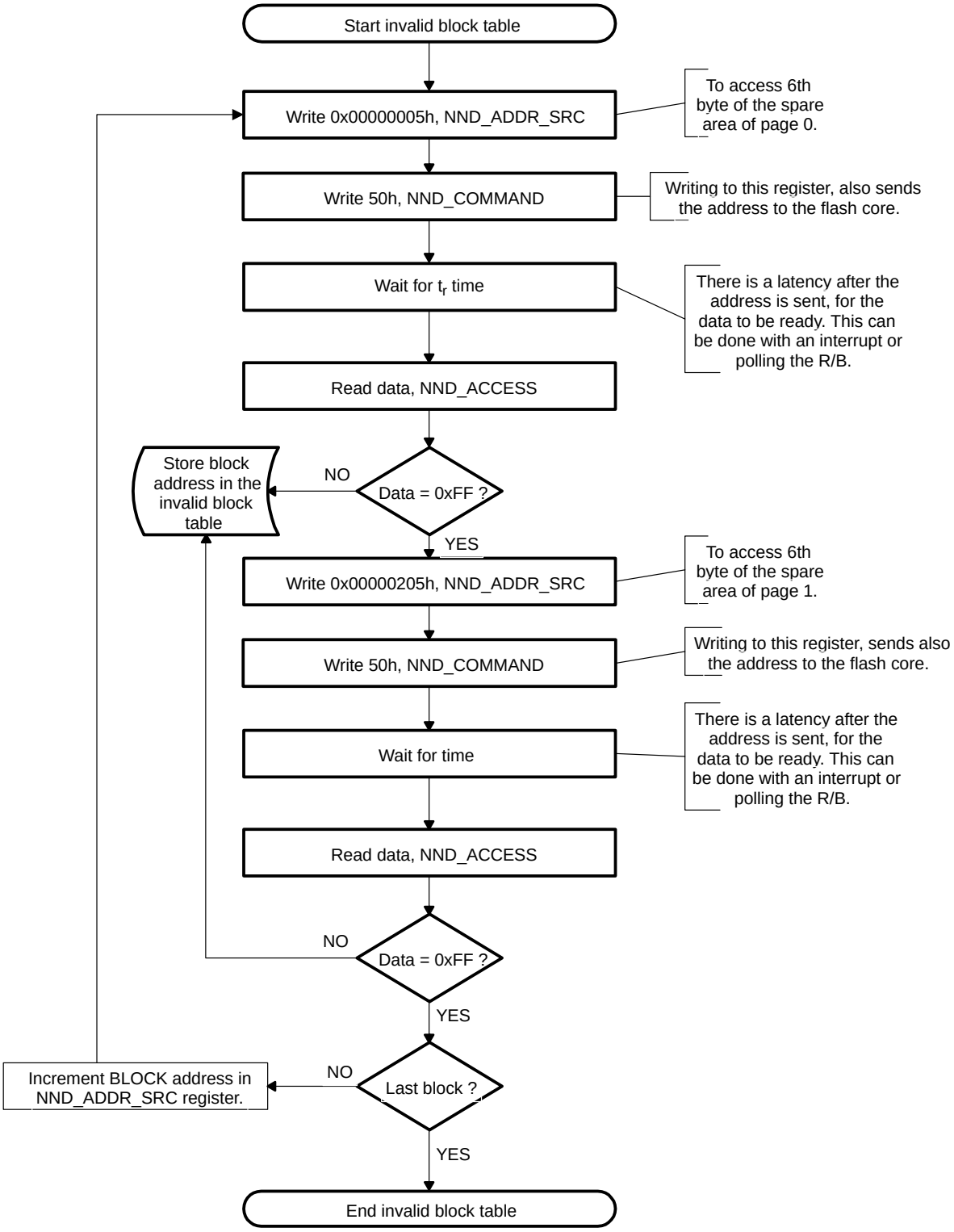
Over time, reading/programming/erasing NFMC can introduce errors in data. ECC can protect the data, but if there is more than one error, ECC cannot correct any errors. Invalid blocks are defined as blocks that contain one or more invalid bits whose reliability is not ensured by the NFMC manufacturer. All NFMC locations are erased (0xFF), except locations where the invalid block information is written before shipping. The invalid block status is defined by the 6^{th} byte in the spare area in page 0 and page 1 (the first 2 pages) of a block. Because the invalid block information is also erasable, it is impossible to recover the information once it has been erased. Therefore, the system must be able to recognize the invalid block based on the original invalid block information and create an invalid block table. If, after an operation, a block becomes corrupted, it must be added to the invalid block table. This table management is done by software. When a page or block becomes corrupted, a value other than 0xFF must be written in the invalid block byte in the spare area. *The invalid block location can depend on the NFMC. For instance, on the 1G-bit mono-die, the invalid block status is defined by the 1^{st} byte of the spare area* (see Figure 13–13).

If an operation fails, a decision must be made, as shown in Table 13–10.

Table 13–10. Decision Table When Operation Fails

	Failure	Decision
Write	Erase failure	Write data to a new page. Mark block as invalid.
	Program failure	Write data to a new page. Mark block as invalid.
Read	Single-bit failure	ECC correction
	Multiple-bit failure	Uncorrectable error Mark block as invalid.

Figure 13–13. Invalid Block Mapping



13.1.12 FIFO (Prefetch and Postwrite)

A FIFO with prefetch and postwrite functions prevents stalling the processor for too many cycles when reading or programming the NFMC. The host either writes or reads this FIFO directly; the NFC state machine handles the actual programming or reading of the NFMC.

Two bits in the NND_CTRL register of the NFC select prefetch or postwrite mode (see Table 13–11).

Table 13–11. Prefetch/Postwrite Mode

Prefetch-Postwrite	Selected Mode
0–0	Host mode
0–1	Postwrite mode
1–0	Prefetch mode
1–1	Host mode

The prefetch/postwrite can read/write in advance 1, 2, 4, 8, or 16 bytes. The FIFO_SIZE field of the NND_FIFOCTRL register holds one of these values.

The BLOCK_COUNT field of the NND_FIFOCTRL register also holds the number of (FIFO_SIZE) chunks of data to read/write from the NFMC. An internal counter is loaded from the NND_FIFOCTRL under certain conditions, but is not accessible from software.

The software must assure coherence between the FIFO_SIZE and BLOCK_COUNT values stored in the NND_FIFOCTRL.

For instance, if FIFO_SIZE equals 16, then the BLOCK_COUNT must be programmed with 32 or 33 (if spare bytes are included).

13.1.13 Prefetch

After the host programs the read address of the page, the read command is sent, and the host enables the prefetch bit. The internal counter is loaded with the BLOCK_COUNT value of the NND_FIFOCTRL register. Interrupt is asserted only if the different contributions of events (READY_EVENT, FIFO_FULL, and so on) are not masked (see Figure 13–14).

The NFMC enters its busy mode as indicated by the R/B_ pin. When the NFMC is ready, the NFC starts fetching one FIFO_SIZE byte(s) of data and begins to fill the NFC internal FIFO. While the FIFO is being loaded with data from NFMC, any access to data, address, and command registers is stalled.

When the FIFO is full with FIFO_SIZE byte(s), the NFC signals it by asserting low an interrupt (event FIFO_FULL is 1 and MSK_FULL is 1) and the counter is decremented. The host can read the FIFO through the register NND_FIFO access(es). Any read past the last byte of the FIFO returns the last byte of the FIFO.

For instance, if the FIFO_SIZE is 16 and the pointer is on the 14th position of the FIFO, a 32-bit read of the FIFO returns:

		[31:24]	[23:16]	[15:8]	[7:0]
Data read	=	B15	B15	B15	B14

where B14 was the byte at position 14 in the FIFO, and B15 was the byte at position 15 in the FIFO.

When the FIFO is fully read by the host, the NFC state machine can fill the FIFO again. Active events are cleared by software by writing to the NND_STATUS register. When the internal counter reaches 0, an interrupt is asserted (event COUNT_ZERO is 1 and MSK_COUNT is 1), and the prefetch is stopped. The host can also decide at any time to stop the prefetch by writing a 0 to the prefetch bit.

1) NFC fetches the data and fills the FIFO. (Access to FIFO, in that case, is stalled.)

2) When the FIFO is full, the NFC asserts the interrupt (FIFO_FULL event pending and MSK_FULL equals 1) and the internal counter is decremented.

3) By host accesses to NND_FIFO, read the FIFO. The FIFO_FULL event must be cleared by writing to NND_STATUS or a future FIFO_FULL event will not be seen.

4) When the FIFO is empty and the counter is not 0, the NFMC triggers a new prefetch.

5) If the counter is 0, the event COUNT_ZERO is set and interrupt is asserted if MSK_COUNT is set and the prefetch is stopped.

6) The host clears the pending event(s) by writing to NND_STATUS of the NFC.

When prefetch goes from 0 to 1:

7) The internal counter is loaded with the BLOCK_COUNT value of the NND_FIFOCTRL register.

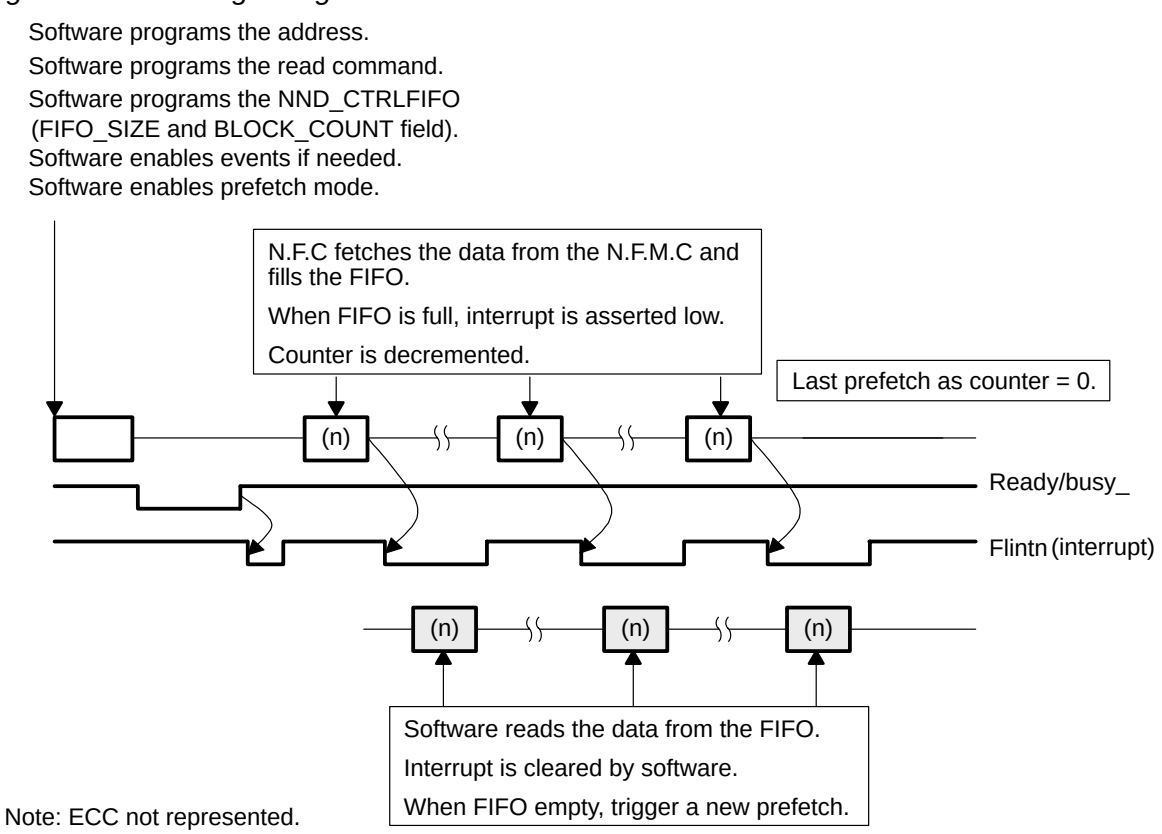
8) The FIFO is flushed. (FIFO_EMPTY event is set. Depending on the value of MSK_EMPTY, the interrupt is asserted or not.)

When prefetch goes from 1 to 0:

9) Prefetch is aborted.

Before accessing a new page, software must successively write 0 and then 1 in the prefetch bit. At reset, the prefetch bit is set to 0.

Figure 13–14. Single Page Read in Prefetch Mode



Note: ECC not represented.

13.1.14 Postwrite

After the host programs the address of the page to program, the command program is sent, and the host enables the postwrite bit. The internal counter is loaded with the BLOCK_COUNT value of the NND_FIFOCTRL register (see Figure 13–15).

The software can write a FIFO_SIZE byte(s) in the FIFO of the NFC by accessing NND_FIFO. When the FIFO is full, the NFC sends the data to the NFMC, asserts the interrupt (if event FIFO_FULL is unmasked), and the counter is decremented. When all of the data is sent, the FIFO is empty and the NFC asserts low the interrupt (if event FIFO_EMPTY is unmasked). The CPU refills the FIFO and clears the interrupt. When the counter reaches zero, the interrupt is also asserted (if event COUNT_ZERO is unmasked). To finish programming the NFMC, the command 0x10 must be sent.

1) The host writes the data in the FIFO of the NFC. Once the FIFO is full, all access is stalled.
2) When the FIFO is full, the event FIFO_FULL is set, the NFC sends the data from the FIFO to the NFMC, and the internal counter is decremented.
3) When the FIFO is empty, the event FIFO_EMPTY is set, and interrupt is asserted if unmasked (active low).
4) When the FIFO is empty and the counter is not 0, the host writes a new batch of data in the FIFO of the NFC.

5) If the counter is 0, the event COUNT_ZERO is set and the interrupt is asserted if unmasked, the NFC sends the data from the FIFO to the NFMC, and the postwrite mechanism is stopped.

6) The host clears the pending event(s) by writing to NND_STATUS of the NFC.

While the FIFO is being filled, access to command, data, and address registers is stalled.

When postwrite goes from 0 to 1:

7) The FIFO is flushed. (FIFO_EMPTY event is set. Depending on the value of MSK_EMPTY, the interrupt is asserted or not.)

8) The internal counter is loaded with the BLOCK_COUNT value of the NND_FIFOCTRL register.

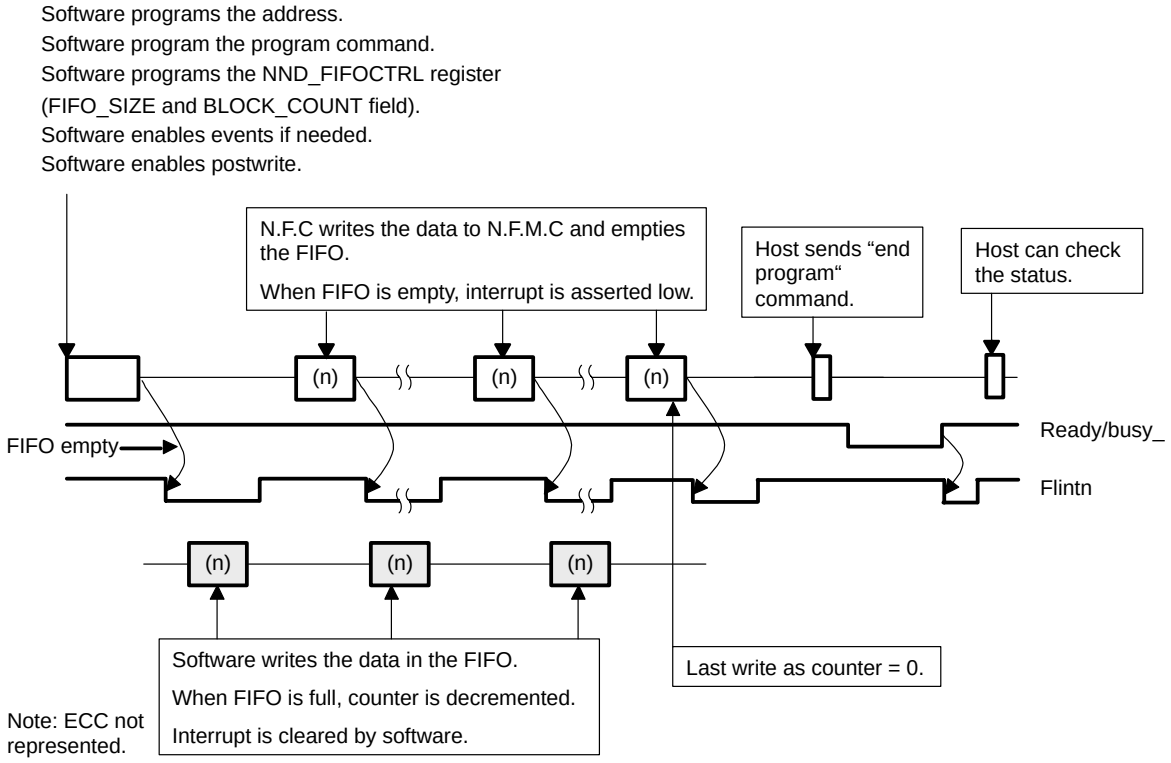
When the postwrite goes from 1 to 0:

9) Postwrite is aborted.

Any write past the last byte of FIFO is discarded. For instance, if the FIFO_SIZE is 16 and the pointer is on the 14th position of the FIFO, a 32-bit write of a data equal to X3X2X1X0 is placed on the FIFO :

	13th	**14th**	**15th**	**16th**
Be = 0	data	X0	X1	X2
Be = 1	data	X3	X2	X1

Figure 13–15. Single-Page Program in Postwrite Mode



13.1.15 DMA Support

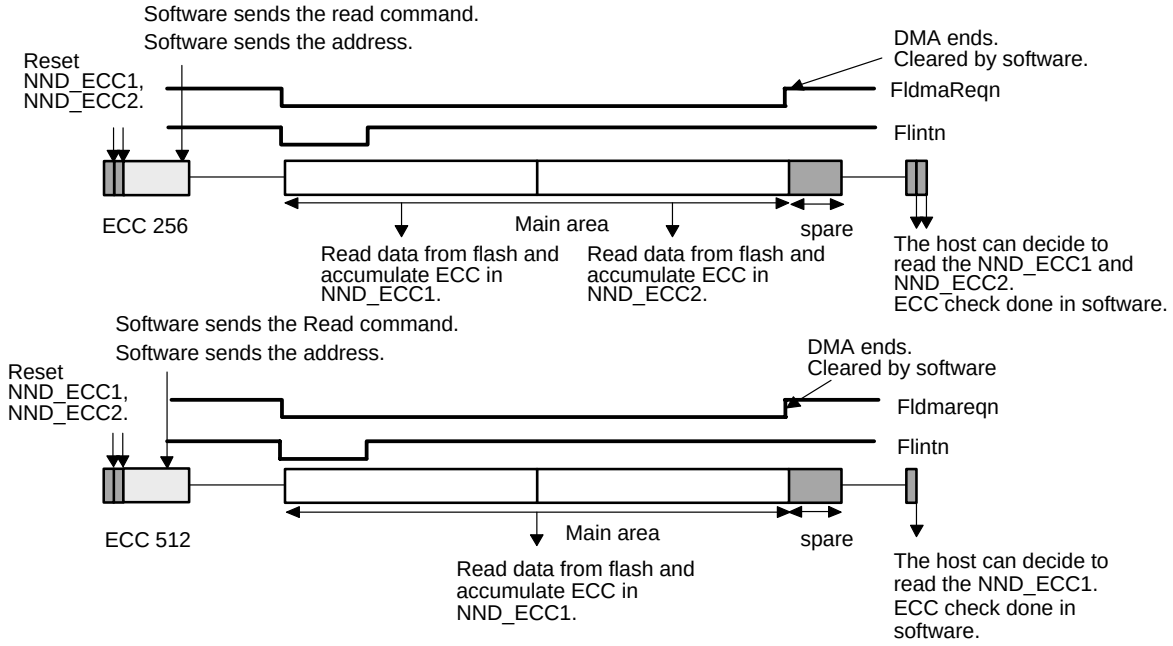
The NFC supports DMA read and write on one page in the host and FIFO (prefetch and postwrite) mode.

13.1.16 Host Mode

13.1.16.1 Read Operation

The DMA request is asserted low after the latency time necessary for the NFMC to transfer data from the memory cell to its internal register. READY/ BUSY_ goes low and transitions to high to indicate data ready. By writing to ResetDMASynchro of the NND_RESET register, the DMA request is asserted high (see Figure 13–16).

Figure 13–16. Single Page Read DMA



13.1.16.2 Write Operation

No DMA signal is generated in DMA program operation.

13.1.17 FIFO Mode

13.1.17.1 Prefetch

After the host programs the address of the page to read, and the command read is sent, the host enables the FIFO prefetch bit and enables DMA in system DMA. The internal counter is loaded with the BLOCK_COUNT value of the NND_FIFOCTRL register, and the DMA request is driven high (see Figure 13–17).

The NFMC enters its busy mode as indicated by the R/B_ terminal. When the NFMC is ready, the NFC starts fetching one FIFO_SIZE byte(s) of data and

begins to fill the NFC internal FIFO. While the FIFO is being loaded with data from NFMC, access to data, address, and command registers is stalled.

When the FIFO is full with FIFO_SIZE byte(s), the NFC signals it by asserting low the DMA request, and the internal counter is decremented by 1. The DMA can read the FIFO through access(es) to the NND_FIFO register. Any read past the last byte of the buffer returns the last byte of the FIFO. When the FIFO is fully read, the DMA request is asserted high and the NFC state machine can fill the FIFO again. When the internal counter reaches 0, the FIFO is filled a last time, and when the FIFO is full, the DMA request is asserted low. DMA can read the FIFO a last time, and when it is empty, the DMA request is asserted high. Because the counter has reached 0, however, a new prefetch is not triggered. The host can also decide at any time to stop the prefetch by writing a 0 to the prefetch bit.

1) The host enables the prefetch bit. DMA request is asserted high.
2) NFC fetches the data and fills the FIFO (access to the FIFO, in that case, is stalled)
3) When the FIFO is full, the NFC asserts low the DMA request and the internal counter is decremented.
4) The DMA reads the FIFO.
5) When the FIFO is read and is empty, the DMA request is asserted high. If the counter is not 0, a new prefetch is triggered.
6) If the counter is 0, the prefetch is stopped.
7) The prefetch bit is disabled if necessary.

When prefetch goes from 0 to 1:

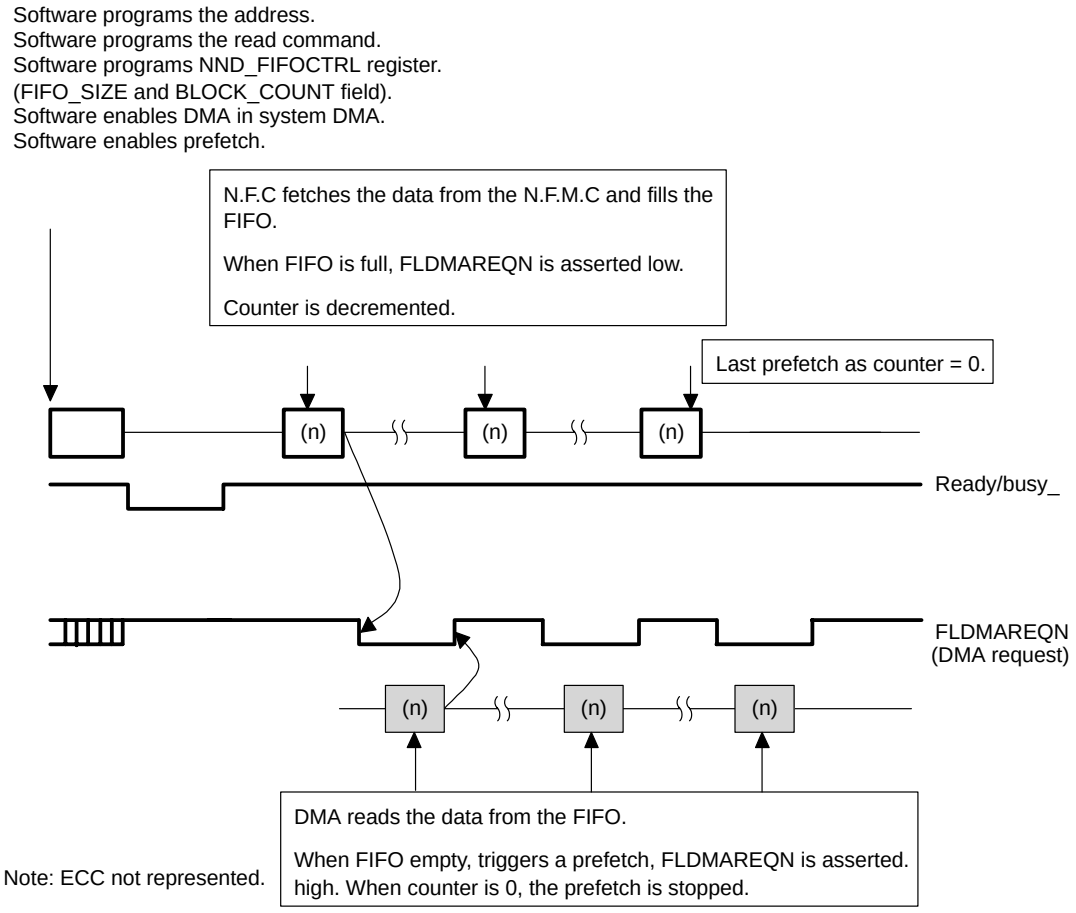
8) The internal counter is loaded with the BLOCK_COUNT field of the NND_FIFOCTRL.
9) The FIFO is flushed.
10) The DMA request is asserted to high.

When prefetch goes from 1 to 0:

11) Prefetch is aborted.

Before accessing a new page, software must successively write 0 and then 1 in the prefetch bit. At reset, the prefetch bit is set with 0.

Figure 13–17. Single-Page Read DMA in Prefetch Mode



13.1.17.2 Postwrite

After the host programs the address of the page to program, the program command is sent, and the host enables the FIFO postwrite mode and DMA in system DMA. The internal counter is loaded with the BLOCK_COUNT value of the NND_FIFOCTRL register and the DMA request is driven low (see Figure 13–18).

The DMA can write a FIFO_SIZE byte(s) in the FIFO of the NFC. When the FIFO is full, the DMA request is asserted high and the counter is decremented by 1. The NFC sends the data to the NFMC. When all of the data is sent to the NFMC, the FIFO is empty, and the NFC asserts low the DMA request. The DMA can refill the FIFO. When the counter is 0, the DMA request is not asserted any longer, and the NFC sends the data from the FIFO to the NFMC for the last time. To finish the programming of the NFMC, the command 0x10 must be sent.

1) DMA writes the data in the FIFO: Once the FIFO is full, any access is stalled.

2) When the FIFO is full, the internal counter is decremented, and the NFC sends the data from the FIFO to the NFMC.

3) When the FIFO is empty, the DMA request is asserted low.

4) When the FIFO is empty and the counter is not 0, the DMA writes a new batch of data in the FIFO.

5) If the counter is 0, the NFC sends the data from the FIFO to the NFMC for the last time and the postwrite mechanism is stopped. No DMA request is asserted.

While the FIFO is being filled, access to command, data, and address registers is stalled.

When postwrite goes from 0 to 1:

6) The DMA request is asserted low.

7) The FIFO is flushed.

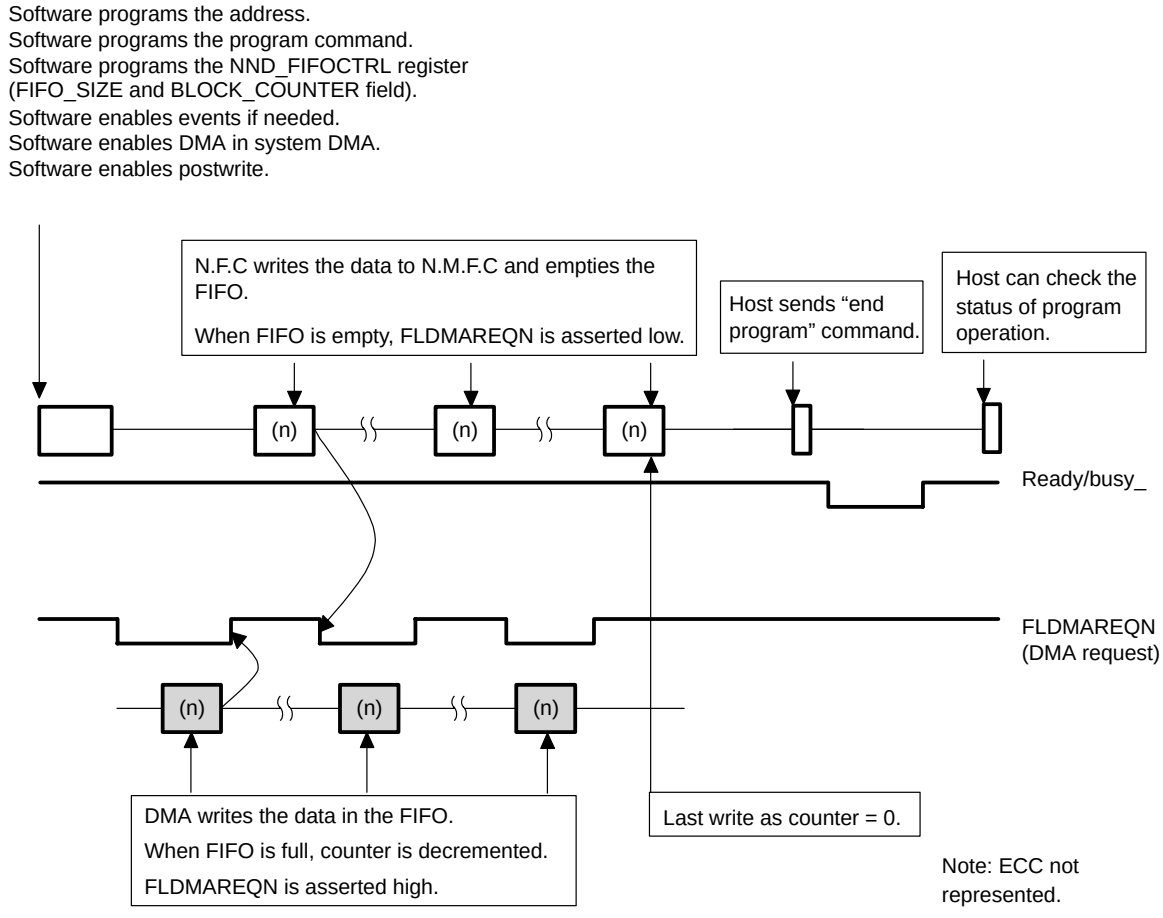
8) The internal counter is loaded with the BLOCK_COUNT value of the NND_FIFOCTRL register.

When the postwrite goes from 1 to 0:

9) The next postwrite is aborted.

10) The DMA request is asserted to high.

Figure 13–18. Single-Page Program DMA in Postwrite Mode



13.1.18 NAND Flash Memory Core Support

Table 13–12 summarizes the characteristics of supported NFMCs.

Table 13–12. Characteristics of Supported NFMCs

	1Gb/ 128 MB	**512Mb/ 64 MB**	**256Mb/ 32 MB**	**128Mb/ 16 MB**	**64Mb/ 8 MB**	**32Mb/ 4 MB**
Number of pages	262144	131072	65536	32768	16384	8192
Page size+ spare (bytes)	512+16	512+16	512+16	512+16	512+16	512+16
Number of blocks	8192	4096	2048	1024	1024	512
Number of pages per block	32	32	32	32	16	16
Block size (bytes)	16K+512	16K+512	16K+512	16K+512	8K+256	8K+256
Addresses	4 cycles	4 cycles	3 cycles	3 cycles	3 cycles	3 cycles
	a7-a0	a7-a0	a7-a0	a7-a0	a7-a0	a7-a0
	a16-a9	a16-a9	a16-a9	a16-a9	a16-a9	a16-a9
	a24-a17	a24-a17	a24-a17	a23-a17	a22-a17	a21-a17
	a25-a26	a25				
ID	79	76	75	73	E6	E3
Samsung code	EC	EC	EC	EC	EC	EC
Toshiba code	98	98	98	98	98	98
Fujitsu code	04	04	04	04	04	04

Table 13–13 summarizes the type of operations supported by the NFMCs.

Table 13–13. Supported Operations on NFMCs

Supported Operations	1Gb/ 128MB	512Mb/ 64MB	256Mb/ 32MB	128Mb/ 16MB	64Mb/ 8MB	32Mb/ 4MB
Read 1	P	P	P	P	P	P
Read 2	P	P	P	P	P	P
Read ID	P	P	P	P	P	P
Reset	P	P	P	P	P	P
Page program	P	P	P	P	P	P
Page program multiple	P	P	-	-	-	-
Copy-back program	P	P	P	-	-	-
Copy-back program multiple	P	P	-	-	-	-
Block erase	P	P	P	P	P	P
Block erase multiple	P	P	-	-	-	-
Read status	P	P	P	P	P	P
Read status multiple	P	P	-	-	-	-

Note: Note: This table summarizes the supported operations for the current NFMCs. These operations may change in the future when new NFMCs are introduced in the market. To program new NFMCs, refer to the NFMC specifications from vendors.

13.1.19 NAND Flash Registers

For the mapping of the register, see Table 13–14. Each register is 32 bits wide; for a register, the most significant word16 is above the least significant word16.

Table 13–14. Register Mapping

Addr	Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	Reg1	Reg1_value (most significant word16)															
0x00	Reg1	Reg1_value (least significant word16)															
0x04	Reg2	Reg2_value (most significant word16)															
0x04	Reg2	Reg2_value (least significant word16)															

Table 13–15 lists the NAND flash registers. Table 13–16 through Table 13–46 describe the register bits.

Table 13–15. NAND Flash Registers

Name	Description	Address Offset
NND_REVISION	NAND controller revision	0x00
NND_ACCESS	NAND controller access	0x04
NND_ADDR_SRC	NAND controller address	0x08
RESERVED	Reserved	0x0C
NND_CTRL	NAND controller control	0x10
NND_MASK	NAND controller mask	0x14
NND_STATUS	NAND controller status	0x18
NND_READY	NAND controller ready	0x1C
NND_COMMAND	NAND controller command	0x20
NND_COMMAND_SEC	NAND controller second command	0x24
NND_ECC_SELECT	NAND controller ECC bank selection	0x28
NND_ECC1 to NND_ECC9	NAND controller ECC1...NAND controller ECC9	0x2C...0x4C
NND_RESET	NAND controller reset	0x50
NND_FIFO	NAND controller FIFO access	0x54
NND_FIFOCTRL	NAND controller FIFO control	0x58
NND_PSC_CLK	NAND controller clock prescaler	0x5C
NND_SYSTEST	NAND controller system test	0x60
NND_SYSCFG	NAND controller system configuration	0x64
NND_SYSSTATUS	NAND controller system status	0x68
NND_FIFOTEST1	NAND controller FIFO test 1	0x6C

Note: All reserved bits must be written with 0.

Table 13–15. NAND Flash Registers (Continued)

Name	Description	Address Offset
NND_FIFOTEST2	NAND controller FIFO test 2	0x70
NND_FIFOTEST3	NAND controller FIFO test 3	0x74
NND_FIFOTEST4	NAND controller FIFO test 4	0x78

Note: All reserved bits must be written with 0.

Table 13–16. NAND Controller Revision Register (NND_REVISION)

Bit	Name	Description
31-8	Reserved	Reserved
7-0	NND_REVISION	Revision number

The NND_REVISION field indicates the current revision number of the NFC. This value is fixed by hardware.

- The 4 LSBs indicate a minor revision.
- The 4 MSBs indicate a major revision.
  - 0x01: Revision 0.1
  - 0x02: Revision 0.2
  - 0x21: Revision 2.1

Hardware reset and software reset do not act on this register.

Table 13–17. NAND Controller Access Register (NND_ACCESS)

Bit	Name	Description	R/W	Reset Value
31:0	Data	Data read/write access to external NAND flash	R/W	0

NND_ACCESS contains the data read/write access to the external NAND flash. An access to this location performs the read or program(write) operation, depending on the value of the MCMD signal. For program operation, data present on the MDATA bus is written in the NFMC.

For read operation, data is fetched from the NFMC and driven on the sdata bus. 8-bit, 16-bit, and 32-bit accesses are supported, but bytes are serialized from/to NFC to/from NFMC. A 4 x 8-bit buffer in the NFC packs the bytes coming from/to the NFMC.

When no operation is selected, the value 0x00000000 is driven on the SDATA bus.

Table 13–18. NAND Controller Address Register (NND_ADDR_SRC)

Bit	Name	Description	R/W	Reset Value
31:0	Address	External NAND flash start address	R/W	0

NND_ADDR_SRC contains the byte address of the location in the NFMC. From this location, the data is read or written by accessing the NAND controller

access register (NND_ACCESS). The address in this register represents the starting address, and thus is never incremented. For multiple reading and writing, the NFMC has a pointer that is incremented.

This register also holds the block erase address, which must be properly formatted before being written in this register (see Table 13–5). Because an access to NND_COMMAND also sends this address to the NFMC, the software first must write the new address in the NND_ADDR_SRC, and then write a command through NND_COMMAND. This address remains unchanged even if a read and a program operation are performed. The address formatting depends on whether bit A8 of NND_CTRL is set or not. The command codes 0x00, 0x01, or 0x50 select which area of the NFMC to access.

Table 13–19. Address Decomposition

Size of Flash Core (MB)	Block Address	Page Address in Block	Start Address in Page
32	A21-A13	A12-A9	A7-A0
64	A22-A13	A12-A9	A7-A0
128	A23-A14	A13-A9	A7-A0
256	A24-A14	A13-A9	A7-A0
512	A25-A14	A13-A9	A7-A0
1024	A26-A14	A13-A9	A7-A0

When a page is selected, sending the command 0x00, 0x01, or 0x50 distinguishes among areas A, B, or C (spare).

Table 13–20. NAND Controller Control Register (NND_CTRL)

Bit	Name	Description
31:18	Reserved	Reserved
17	Prefetch	When 1, prefetch mode is enabled. Prefetch bit. Writing 1 to this bit enables the prefetch mechanism.Access to the internal FIFO is possible. The NFC fetches data from the NFMC and fills the FIFO. The host or DMA reads data from the FIFO.
16	Postwrite	When 1, postwrite mode is enabled. Postwrite bit. Writing 1 to this bit enables the postwrite mechanism.Access to the internal FIFO is possible. The host or DMA writes data in the FIFO, and the NFC is responsible for unloading the FIFO to the NFMC.
15	WriteProt3	Write protect 3. When 0, the voltage generator is reset. The WriteProt3 bit provides inadvertent write/erase protection during power transitions. The NFMC internal high voltage generator is reset when the WriteProt3 signal is low. This signal is low during reset, and software must write a 1 in this location before any access to the NFMC.

Table 13–20. NAND Controller Control Register (NND_CTRL) (Continued)

Bit	Name	Description
14	ChipEn3	Chip enable 3. When 0, NFMC device is selected. The ChipEn3 directly controls the NFMC selection control. When ChipEn3 is high, access to the NFMC is impossible. When ChipEn3 goes high during a read operation, the NFMC returns to standby mode. However, when the NFMC is in a busy state during a program or an erase operation, the NFMC ignores ChipEn3 and does not return to standby mode. *Because software has direct access to the ChipEn3 signal, it must be set to low when access is performed and not set back to high before all operations to the NFMC are terminated.*
13	WriteProt2	Write protect 2. When 0, the voltage generator is reset. The WriteProt2 bit provides inadvertent write/erase protection during power transitions. The NFMC internal high voltage generator is reset when the WriteProt2 signal is low. This signal is low during reset, and software must write a 1 in this location before any access to the NFMC.
12	ChipEn2	Chip enable 2. When 0, NFMC device is selected. The ChipEn2 directly controls the NFMC selection control. When ChipEn2 is high, access to the NFMC is impossible. When ChipEn2 goes high during a read operation, the NFMC returns to standby mode. However, when the NFMC is in a busy state during a program or an erase operation, the NFMC ignores ChipEn2 and does not return to standby mode. *Because software has direct access to the ChipEn2 signal, it must be set to low when access is performed and not set back to high before all operations to the NFMC are terminated.*
11	WriteProt1	Write protect 1. When 0, the voltage generator is reset. The WriteProt1 bit provides inadvertent write/erase protection during power transitions. The NFMC internal high voltage generator is reset when the WriteProt1 signal is low. This signal is low during reset, and software must write a 1 in this location before any access to the NFMC.
10	CHIPEN1	Chip enable1. When 0, NFMC device is selected. The ChipEn1 directly controls the NFMC selection control. When ChipEn1 is high, access to the NFMC is impossible. When ChipEn1 goes high during a read operation, the NFMC returns to standby mode. However, when the NFMC is in a busy state during a program or an erase operation, the NFMC ignores ChipEn1 and does not return to standby mode. *Because software has direct access to the ChipEn1 signal, it must be set to low when access is performed and not set back to high before all operations to the NFMC are terminated.*
9	WRITEPROT0	Write protect 0. When 0, the voltage generator is reset. The WriteProt0 bit provides inadvertent write/erase protection during power transitions. The NFMC internal high voltage generator is reset when the WriteProt0 signal is low. This signal is low during reset and software must write a 1 in this location before any access to the NFMC.

Table 13–20. NAND Controller Control Register (NND_CTRL) (Continued)

Bit	Name	Description
8	CHIPEN0	Chip enable 0. When 0, NFMC device is selected. The ChipEn0 directly controls the NFMC selection control. When ChipEn0 is high, access to the NFMC is not allowed. When ChipEn0 goes high during a read operation, the NFMC returns to standby mode. However, when the NFMC is in a busy state during a program or an erase operation, the NFMC ignores ChipEn0 and does not return to standby mode. *Because software has direct access to the ChipEn0 signal, it must be set to low when access is performed and not set back to high before all operations to the NFMC are terminated.*
7	Reserved	Reserved
6:5	ADRCNT[1:0]	Address counter for sending bytes to NFMC. This counter sends bytes from the NND_ADDR_SRC to the NFMC. Extra address bytes are usually discarded by the NFMC logic, but this is a protection in case the NFMC logic does not do it. See Table 13–27.
4	A8	When 0, bit A8 of address register is not sent to NFMC. For the current NFMCs, bit 8 of the address is not sent to NFMC, because its functionality is replaced by a pointer set by command 0x00, 0x01, or 0x50. Future NFMCs have a different page addressing configuration and bit 8 is part of the address. When control bit A8 is 0, bit 8 of NND_ADDR_SRC is not sent to NFMC. When control bit A8 is 1, bit 8 of NND_ADDR_SRC is sent to NFMC. See Table 13–24 through Table 13–26.
3	BE	Switch big/little endian. 1: Big endian, 0: Little endian Depending on the access (32-/16-/8-bit) type and this bit, the NFC packs bytes differently. For example, assume that the NAND flash memory core contains: B0 at address n B1 at address n+1 B2 at address n+2 B3 at address n+3 Table 13–22 and Table 13–23 shows the resulting read data from the NAND flash memory core.
2	Reserved	Reserved
1	ECC_256	If 1, ECC is calculated on 256 bytes; else, 512 bytes. When set, ECC is accumulated on a chunk of 256 bytes or 512 bytes. The algorithm is the same for both; only two more parities (P2048 and P2048') are calculated. This bit must be set before enabling the ECC logic. Depending on the value of ECC_ON (bit 0), two accesses may be needed to change the size of the ECC calculation.
0	ECC_ON	If 1, ECC logic is enabled. When set, the ECC logic is enabled and for read or program operation, ECC is computed and stored in the NND_ECCx registers. When ECC_ON is set, it is impossible to write to the ECC_256 bit to avoid corrupting the ECC logic. Table 13–21 shows the effects of ECC bit settings.

Table 13–21. ECC Bits Operation

ECC_ON	ECC_256	Effect
1	1/0	No change (previous value)
0	1	ECC calculated on 256 bytes
0	0	ECC calculated on 512 bytes

Table 13–22. Byte Packing Function of MBYTEEN and Little/Big Endianism (NND_ACCESS/NND_FIFO)

MBYTEEN	BE = 0				BE = 1			
	[31:24]	[23:16	[15:8]	[7:0]	[31:24]	[23:16	[15:8]	[7:0]
1111	B3	B2	B1	B0	B0	B1	B2	B3
0011	0	0	B1	B0	0	0	B0	B1
1100	B1	B0	0	0	B0	B1	0	0
0001	0	0	0	B0	0	0	0	B0
0010	0	0	B0	0	0	0	B0	0
0100	0	B0	0	0	0	B0	0	0
1000	B0	0	0	0	B0	0	0	0

For the NND_ACCESS/NND_FIFO register (program operation):

- Other combinations of MBYTEEN are not permitted.

For all other registers:

- The BE bit has no effect on other NFC registers. Only MBYTEEN bits apply.

Table 13–23. Byte Packing Function of MBYTEEN for Registers (Except NND_ACCESS/NND_FIFO).

MBYTEEN	Registers			
	[31:24]	[23:16]	[15:8]	[7:0]
1111	DATA3	DATA2	DATA1	DATA0
0011	0	0	DATA1	DATA0
1100	DATA3	DATA2	0	0
0001	0	0	0	DATA0
0010	0	0	DATA1	0
0100	0	DATA2	0	0
1000	DATA3	0	0	0

Table 13–24. 1 Gigabit Dual-Die Layout of Bytes Sent

	I/O 7	I/O 6	I/O 5	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0
1st cycle	A7	A6	A5	A4	A3	A2	A1	A0
2nd cycle	A16	A15	A14	A13	A12	A11	A10	A9
3rd cycle	A24	A23	A22	A21	A20	A19	A18	A17
4th cycle	0	0	0	0	0	0	A26	A25

Table 13–25. 1 Gigabit Mono-Die Layout of Bytes Sent

	I/O 7	I/O 6	I/O 5	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0
1st cycle	A7	A6	A5	A4	A3	A2	A1	A0
2nd cycle	0	0	0	0	A11	A10	A9	A8
3rd cycle	A19	A18	A17	A16	A15	A14	A13	A12
4th cycle	A27	A26	A25	A24	A23	A22	A21	A20

Table 13–26. NND_ADDR_SRC Decomposition of Flash Bus Function of Control Bit A8

Control Bit A8	Cycle	NND_ADDR_SRC Decomposition on Flash Bus
0	1st cycle	NND_ADDR_SRC[7:0]
	2nd cycle	NND_ADDR_SRC[16:9]
	3rd cycle	NND_ADDR_SRC[24:17]
	4th cycle	0-NND_ADDR_SRC[31:25]
1	1st cycle	NND_ADDR_SRC[7:0]
	2nd cycle	NND_ADDR_SRC[15:8]
	3rd cycle	NND_ADDR_SRC[23:16]
	4th cycle	NND_ADDR_SRC[31:24]

Table 13–27. Address Counter for Sending Bytes

ADRCNT[1]-ADRCNT[0]	Number of Byte Address Sent
0-0	4 bytes sent (with least significant byte first)
0-1	3 bytes sent (with least significant byte first)
1-0	2 bytes sent (with least significant byte first)
1-1	1 byte sent (with least significant byte first)

Table 13–28. NAND Controller Mask Register (NND_MASK)

Bit	Name	Description
31:4	Reserved	Reserved
3	MSK_EMPTY	Mask. When 0, event for FIFO empty is masked.
2	MSK_FULL	Mask. When 0, event for FIFO full is masked.
1	MSK_COUNT	Mask. When 0, event for counter reaching 0 is masked.
0	MSK_READY	Mask. When 0, event for data ready is masked.

The NAND controller mask register masks event sources. At reset, all event sources are masked. To unmask an event, a 1 must be written in the bit of the event to be unmasked. These masks are ANDed with the corresponding bits in the NND_STATUS register. Then, all of these contributions are ORed again to form the global interrupt at the boundary of the NFC.

Table 13–29. NAND Controller Status Register (NND_STATUS)

Bit	Name	Description
31:4	Reserved	Reserved
3	FIFO_EMPTY	When FIFO is empty, event is pending and this bit is 1. When the FIFO is empty, the FIFO_EMPTY goes to 1 and an interrupt is asserted if unmasked. The interrupt is cleared by software according to the scheme shown in Table 13–30.
2	FIFO_FULL	When FIFO is full, event is pending and this bit is 1. When the FIFO is full, the FIFO_FULL event goes to 1 and an interrupt is asserted if unmasked. The interrupt is cleared by software according to the scheme shown in Table 13–30.
1	COUNT_ZERO	When internal counter reaches 0, event is pending and this bit is 1. When the NFC is either in prefetch or postwrite mode, an interrupt is generated when the internal counter reaches 0. This interrupt is cleared by software according to the scheme shown in Table 13–30. The interrupt is active if unmasked.
0	READY_ EVENT	When R/B_ goes from 0 to 1, event is pending and this bit is 1. The READY_EVENT bit can be masked with the corresponding MSK_EVENT bit in the NND_MASK register. This bit reflects the status of an active event in different cases: For a read operation after the address has been sent to the NFMC, there is a latency time before the data is ready. The READY_EVENT bit is set when the READY/BUSY pin goes back to high, indicating that data can be read. For a program operation, the READY_EVENT bit is set when the programming is finished.For an erase operation, the READY_EVENT bit is set when the NFMC is finished erasing the selected block. It is the responsibility of the software to clear this bit, as described in Table 13–30.

All pending events sources are active high. Before they can be active, events must be unmasked by writing a 1 in the corresponding mask in the NND_MASK. For example, if READY_EVENT is 1 (pending interrupt) and MSK_READY is 1, then the interrupt at the boundary of the NFC is active low. A reset (resetn going low) or a soft reset clears all interrupts. To clear a pending event, a 1 must be written in the corresponding event source. When there is

an interrupt, the software can read this register to determine which event is active.

Table 13–30. How to Clear Pending Event

Previous Event State	Write	Next Event State
0	1 or 0	0
1	1	0
1	0	1

Table 13–31. NAND Controller Ready Register (NND_READY)

Bit	Name	Description	R/W	Reset Value
31:1	Reserved	Reserved		
0	READY	Ready. When 1, NFMC is ready for next operation.		

Bit 0 in the NAND controller ready register is a copy of the R/B_ signal coming from the NFMC resynchronized inside the NFC. This bit polls the readiness of the NFMC. *It is impossible to write to this bit.* Its value at reset is the sampled value of the R/B_ pin.

Table 13–32. NAND Controller Command Register (NND_COMMAND)

Bit	Name	Description		
31:8	Reserved	Reserved		
7:0	COMMAND	Command operation code		
		Read 1 (lower)	00000000	00
		Read 1 (upper)	00000001	01
		Read 2 (spare)	01010000	50
		Read ID	10010000	90
		Page program/page program (multiple)	10000000	80
		Copy-back program (source address)	00000000	00
		Copy-back program (destination address)	10001010	8A
		Multiple copy-back program	00000011	03
		Block erase/multiplane block erase	01100000	60
		No action	others	

The NAND controller command register writes a specific command to the NFMC. Writing to this register also sends the address contained in NND_ADDR_SRC to the NFMC. There is another type of command register, which does not need to send an address. See the register NND_COM-

MAND_SEC in Table 13–33. The NFMC ignores bit combinations other than those stated in Table 13–32.

Table 13–33. NAND Controller Second Command Register (NND_COMMAND_SEC)

Bit	Name	Description			R/W	Reset Value
31:8	Reserved	Reserved				
7:0	COM-MAND_SEC	Command operation code				
		Program 1 (lower)	00000000	00		
		Program 1 (upper)	00000001	01		
		Program 2 (spare)	01010000	50		
		Reset	11111111	FF		
		Page program end	00010000	10		
		Multiple page program end	00010001	11		
		Copy-back program end	00010000	10		
		Block erase/multiplane block erase end	11010000	D0		
		Read status	01110000	70		
		Read multiplane status	01110001	71		
		No action	others			

Writing to the NAND controller second command register does not send any address to the NFMC. The command operation codes are primarily to indicate the end of the current operation. For instance, for program operation, after the last data is sent, a delimiter is needed, so command 0x10 (page program end) is sent through the NND_COMMAND_SEC register because there is no need to send the address again. The reset and read status operations also use this register. The NFMC ignores bit combinations other than those stated in Table 13–33.

Table 13–34. NAND Controller EEC Bank Selection Register (NND_ECC_SELECT)

Bit	Name	Description
31:3	Reserved	Reserved
2:0	ECC_SELECT	Defines how many NND_ECC registers to enable

NND_ECC_SELECT defines the number of NND_ECC registers to enable. Depending on the size of an NFMC and the size of ECC computation (256/512 bytes), ECC registers are necessary. For a page of 528 bytes and an ECC computation on 256 bytes, three 32-bit ECC registers are necessary:

- First register for the data from address 0 to 255
- Second register for the data from address 256 to 511
- Third register for the data from address 511 to 527

For a future NFMC, the page size can be up to 2048 bytes with a spare of 64 bytes. This means that for an ECC of 256 bytes, nine 32-bit ECC registers are necessary. The first three 32-bit ECC registers are always enabled. By writing to NND_ECC_SELECT, more ECC registers can be enabled.

Table 13–35. Legal Values for NND_ECC_SELECT Registers

NND_ECC_SELECT[2:0]	ECC Selected
000	NND_ECC1, NND_ECC2, NND_ECC3
001	All registers above + NND_ECC4
010	All registers above + NND_ECC5
011	All registers above + NND_ECC6
100	All registers above + NND_ECC7
101	All registers above + NND_ECC8
Any other value	All registers above + NND_ECC9

Table 13–36. NAND Controller ECC Registers (NND_ECC1...NND_ECC9)

Bit	Name	Description	R/W	Reset Value
31:28	Reserved	Reserved	R	0
27	P2048o	Holds ECC code parities accumulated on row	R	0
26	P1024o	Holds ECC code parities accumulated on row	R	0
25	P512o	Holds ECC code parities accumulated on row	R	0
24	P256o	Holds ECC code parities accumulated on row	R	0
23	P128o	Holds ECC code parities accumulated on row	R	0
22	P64o	Holds ECC code parities accumulated on row	R	0
21	P32o	Holds ECC code parities accumulated on row	R	0
20	P16o	Holds ECC code parities accumulated on row	R	0
19	P8o	Holds ECC code parities accumulated on row	R	0
18	P4o	Holds ECC code parities accumulated on column	R	0
17	P2o	Holds ECC code parities accumulated on column	R	0
16	P1o	Holds ECC code parities accumulated on column	R	0
15:12	Reserved	Reserved	R	0
11	P2048e	Holds ECC code parities accumulated on row	R	0
10	P1024e	Holds ECC code parities accumulated on row	R	0
9	P512e	Holds ECC code parities accumulated on row	R	0
8	P256e	Holds ECC code parities accumulated on row	R	0
7	P128e	Holds ECC code parities accumulated on row	R	0
6	P64e	Holds ECC code parities accumulated on row	R	0
5	P32e	Holds ECC code parities accumulated on row	R	0

Table 13–36. NAND Controller ECC Registers (NND_ECC1...NND_ECC9) (Continued)

Bit	Name	Description	R/W	Reset Value
4	P16e	Holds ECC code parities accumulated on row	R	0
3	P8e	Holds ECC code parities accumulated on row	R	0
2	P4e	Holds ECC code parities accumulated on column	R	0
1	P2e	Holds ECC code parities accumulated on column	R	0
0	P1e	Holds ECC code parities accumulated on column	R	0

The NAND controller ECC registers hold the ECC code calculated while reading/writing the NFMC. The Pxxxx{o,e (from 2048 to 8) are the parities accumulated on row, while P4{o,e}, P2{o,e}, and P1{o,e} are the parities columns. P2048e and P2048o are necessary only when ECC is computed on 512 bytes. For 256 bytes, those bit locations are left to 0.

Table 13–37. NAND Controller Reset Register (NND_RESET)

Bit	Name	Description
31:8	Reserved	Reserved
7	RESETDMASYNCHRO	Writing 1 asserts high the DMA request signal. When in host mode (no prefetch or postwrite enabled), writing a 1 in this bit in NND_RESET asserts high the DMA request. This bit is cleared automatically.
6:1	Reserved	Reserved
0	RESET_ECC	When 1, resets NND_ECCx (x from 0 to 9) registers. Writing a 1 in the RESET_ECC bit resets the NND_ECCx registers. After the reset is done, RESET_ECC is reset automatically to 0. The NND_ECCx registers are also reset when the RESETN pin at the NFC boundary goes low. After a RESET_ECC, the next ECC computation uses the NND_ECC1 register.

Table 13–38. NAND Controller FIFO Access Register (NND_FIFO)

Bit	Name	Description	R/W	Reset Value
31:28	Data	FIFO data in prefetch or postwrite mode	R/W	0

The NAND controller FIFO access register is used to access the FIFO when prefetch or postwrite mode is selected. Any access to this register is stalled when FIFO is busy. Access to this register in host mode has no effect; the return value in case of a read is 0x00000000.

Table 13–39. NAND Controller FIFO Control Register (NND_FIFOCTRL)

Bit	Name	Description	R/W	Reset Value
31:24	FIFO_SIZE	Holds the size in bytes of the FIFO	R/W	0
23:16	Reserved	Reserved	R/W	0
15:0	BLOCK_COUNT	Holds the block count of (FIFO_SIZE) bytes to read/write in advance	R/W	0

The NAND controller FIFO control register holds the size of the FIFO and how many blocks of (FIFO_SIZE) bytes to fetch/write to access a full page. For instance, if FIFO_SIZE is 4 bytes, then to read/write 512 bytes (528 bytes with spare), the BLOCK_COUNT field is loaded with 128 (132 with spare). This register is the seed for the internal NFC counter.

The permitted values for the FIFO_SIZE are 1, 2, 4, 8, or 16 bytes, as shown in Table 13–40.

Table 13–40. Permitted Values for FIFO_SIZE

Value	FIFO_SIZE (bytes)
00000001	1(reset value)
00000010	2
00000100	4
00001000	8
Others	16

Table 13–41. NAND Controller Clock Prescale Register (NND_PSC_CLK)

Bit	Name	Description	Reset Value	R/W
31:4	Reserved	Reserved	0	R/W
3:0	PSC_CLK	Prescale sampling clock divider value	0x1	R/W

The NFC uses this 4-bit value to divide the interface clock to adjust the timing for the signals on the NFMC, as shown in Table 13–42.

Table 13–42. Values for Divider

PSC Value	Division	PSC Value	Division
0x0	Divided by 1	0x8	Divided by 9
0x1	Divided by 2	0x9	Divided by 10
0x2	Divided by 3	0xA	Divided by 11
0x3	Divided by 4	0xB	Divided by 12
0x4	Divided by 5	0xC	Divided by 13

Table 13–42. Values for Divider (Continued)

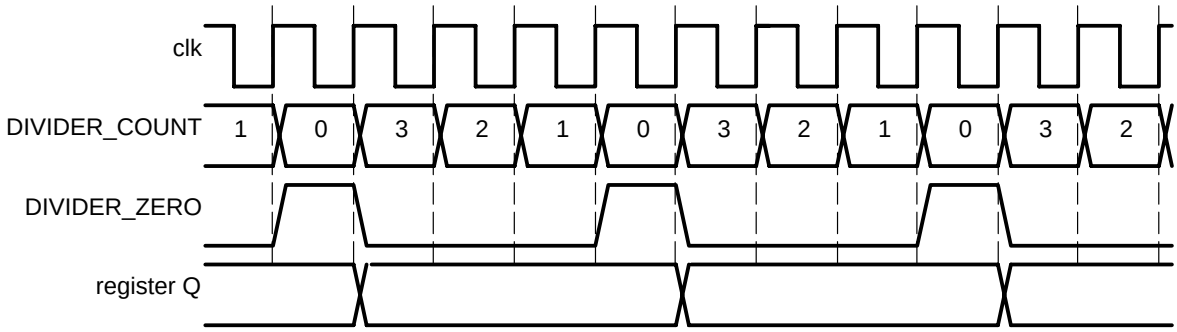
PSC Value	Division	PSC Value	Division
0x5	Divided by 6	0xD	Divided by 14
0x6	Divided by 7	0xE	Divided by 15
0x7	Divided by 8	0xF	Divided by 16

This prescale register is used to generate an enable pulse at the divided rate (see Table 13–42). This pulse is used to enable the NFC sequencer state machine registers.

When it counts down and reaches 0, it generates DIVIDER_ZERO, the enable pulse, and reloads with the value in the NND_PSC_CLK register. All registers that operate at the divided frequency use this pulse as an enable, allowing their outputs to change, as shown by the register Q signal (see Figure 13–19).

Figure 13–19 shows the generation of the divided clock for a divisor of 4: there is no actual clock at the divided rate.

Figure 13–19. Clock Divider Timing



When the counter reloads and it completes a period, truncated periods are not generated. The new value from the NND_PSC_CLK register is not loaded until the end of the current period.

Table 13–43. NAND Controller System Test Register (NND_SYSTEST)

Bit	Name	Description	R/W	Reset Value
31:16	Reserved	Reserved	R/W	0
15	TEST-UNLOCK	Unlocks test features. This bit must be set initially, and then for each other access be kept to 1 by the local host, to allow the local host to set/clear other register-bit positions in this test register on subsequent accesses. Hence, to perform a successful set in another bit position, two successive accesses are required: the 1st access to set the TEST_UNLOCK bit to 1, and the 2nd and subsequent accesses to set another bit position to 0/1 while keeping TEST_UNLOCK to 1. If TEST_UNLOCK is 0, the full sequence needs to take place again.	R/W	0
14:3	Reserved	Reserved	R/W	0

Table 13–43. NAND Controller System Test Register (NND_SYSTEST) (Continued)

Bit	Name	Description	R/W	Reset Value
2	MAP	When 1, the internal FIFO is mapped as registers. This bit, when 1, maps the FIFO to the addressable space of registers. The FIFO is mapped from address 0x64 to 0x70. Those registers are accessible only in test mode; in normal mode, accessing these registers returns an error.	R/W	0
1	ACCESS	If 1, unlock registers for read/write access. When set, all registers become accessible in read and in write (except the NND_REV register). All pins to NFMC are inactive. It is possible to write a value up to 32 bits in the buffer, thus generating an interrupt, and reading back the NAND controller access register (NND_ACCESS).	R/W	0
0	ALLOW_INT	If 1, can initiate an interrupt. Usually, the pending interrupt bits of the NND_STATUS register are set to 1 only when an interrupt is generated; the software cannot initiate an interrupt. By writing a 1 to the ALLOW_INT bit, the software can write a 1 in the interrupt bit of the NND_STATUS, thereby initiating an interrupt.	R/W	0

The NAND controller system test register tests some features of the NFC. There is a special scheme to access it.

Table 13–44. NAND Controller System Configuration Register (NND_SYSCFG)

Bit	Name	Description	R/W	Reset Value
31:2	Reserved	Reserved	R/W	0
1	SOFTRESET	When 1, software reset sequence starts.	R/W	0
0	AUTOIDLE	Controls clock activity	R/W	0

In the NAND controller system configuration register, bit 1 resets the NFC. This is a software reset because the resetn pin is not used for this soft reset. By writing a 1 to this bit, the soft reset sequence starts. This bit is automatically reset to 0 and reads always 0.

If bit 0 is set to 0, the normally enabled clock is free-running. When 1, the module is in power-saving mode. The local or internal clock runs only when the NFMC is accessed or an operation is ongoing.

Table 13–45. NAND Controller System Status Register (NND_SYSSTATUS)

Bit	Name	Description	R/W	Reset Value
31:1	Reserved	Reserved	R	0
0	ResetDone	Internal reset monitoring	R	?

The NAND controller system status register contains the internal reset monitoring. When bit 0 is 1, the NFC has finished its reset. When 0, the NFC is resetting.

Table 13–46. NAND Controller FIFO Test Register (NND_FIFOTEST)

Bit	Name	Description	R/W	Reset Value
31:0	DATA	FIFO data in test mode	R/W	0

When the NND_SYSTEST is used and the map bit is 1, the internal FIFO is mapped to addressable range. It is then possible to directly read or write to the FIFO, bypassing the NND_FIFO register. In normal mode, an access to these registers is denied and the returned value is 0x00000000.

13.2 Software NAND Flash Controller

This section describes two options for interfacing the OMAP730 device to commercially available NAND memories:

- The first option uses the existing memory interface of the EMIFS device.
- The second option outlines the use of a dedicated on-chip NAND flash controller peripheral. This controller (16-pin total interface) is used in conjunction with the multiplexed address/data EMIFS interface.

Some NAND flash devices require that CS be low during the read access time (t_R); hereafter, these devices are called *NAND CE Care.* Thus, a standard OMAP73X chip-select cannot be used for the NAND flash chip-select. However, some NAND flash devices do not require that CS be low during t_R; these devices are called *NAND CE Don't Care*. The OMAP73X chip-select can be used directly for them.

13.2.1 EMIFS Interface With NAND CE Care Flash Device Option

The interface of a NAND CE CARE flash device to the OMAP730 is possible by using $\overline{\text{FLASH.CS2UOE}}$ (output enable) and $\overline{\text{FLASH.CS2UWE}}$ (write enable). The only exception to this policy is that several signals that are used for the NAND flash interface are muxed on signals that are needed for support of synchronous burst flash memories**.**

If both NAND and synchronous burst flash memories are required in the system, two solutions are available:

- Generate the NAND flash interface signals by GPIOs at some loss of system performance
- Use a NAND flash device that is *NAND CE Don't Care*. Be careful to purchase an appropriate NAND flash device (most of the NAND flash devices manufactured are *NAND CE Don't Care*-compliant).

13.2.1.1 NAND Flash Interface Overview

The features of the system are as follows:

- The NAND flash device is mapped on one of the chip-selects of the EMIFS interface of the OMAP730 device.
- One 8-bit- or 16-bit-wide interface NAND flash device is supported on EMIFS.
- The processor manages the command sequence required for block erase, write, read, and block invalidation and management.
- To reduce processor overhead, the system DMA (GDMA) can be used to write or read blocks of data to/from the NAND flash device.

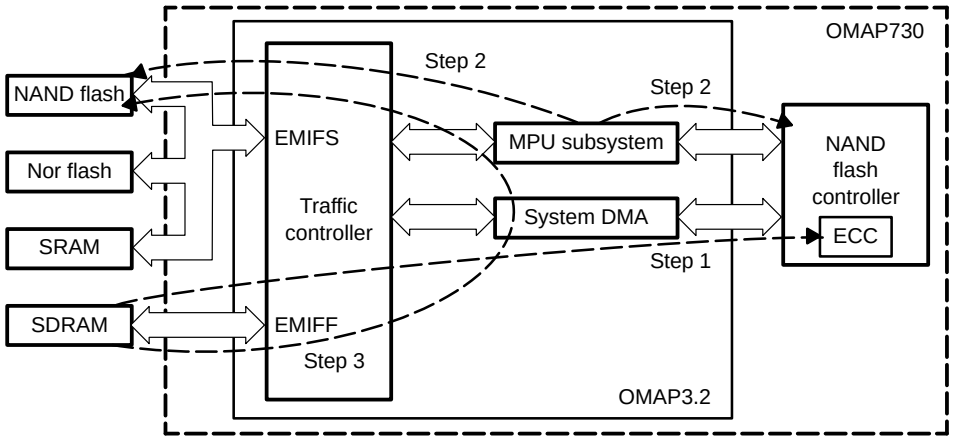
- There are two options for ECC calculation:
  - The MPU calculates the ECC by software.
  - To reduce processor overhead, the NAND flash controller peripheral module can calculate the ECC with the support of the DMA for moving the data. Using the NAND flash controller peripheral, an ECC can be calculated on up to nine blocks of 256 bytes at one time before it is required to read the ECC calculation results.

Section 13.2.2 explains the write sequence example that is required to use a NAND flash device with the OMAP730.

13.2.2 Write Data Sequence Example

The high-level sequence of operations that is needed to write data to the NAND flash device is shown in Figure 13–20. The assumption is that the source data to be written to the NAND flash device resides in external SDRAM memory, but this is not a requirement. The source data can reside in any memory space available in the system. The MPU is the processor controlling the write operations.

Figure 13–20. NAND Flash Write Sequence



Step 1: Calculate ECC

The MPU programs the system DMA as follows:

1) Transfers the data in 8-, 16-, or 32-bit word format from SDRAM into the NAND flash controller
2) Transfer mode (single mode= > channel stop when current transfer finishes)
3) The DMA creates an interrupt on completion of the transfer.
4) Transfer start (hardware start => on DMA request)

The processor MPU configures the NAND flash controller peripheral as follows:

5) Selects the number of blocks to calculate ECC on (1-9) NND_ECC_ SELECT register
6) Selects the type of ECC (256/512 byte blocks) by programming the NND_CTRL register
7) Enables the read and program operation: ECC logic-enable by programming bit 0 of the NND_CTRL register
8) Enables the clock in the NND_SYSCFG register bit 0 (power management)
9) Sets 1 in NND_PSC_CLK (the module runs full-speed for minimum delay in the ECC calculation)
10) Configures the NAND flash FIFO access
11) Resets the ECC calculation by writing to NND_RESET

Note:

To optimize the transfer from SDRAM to NAND flash controller, keep the RDY/BUSY signal of NAND flash controller at 1.

12) The processor initiates a write command to the NAND flash device (in the NAND flash controller) NND_COMMAND.
13) The NAND flash controller sends a DMA request (FIFO mode).
14) The data is written into the NAND flash controller peripheral by the DMA to the NAND flash FIFO.
15) ECC is calculated.
16) The DMA interrupts the processor when the transfer is complete.
17) The processor reads the ECC calculation from the NAND Flash controller peripheral.
18) The calculated ECC results are written in SDRAM.

Step 2: Configure EMIFS and NAND Flash for a Write

1) The associated EMIFS CS chip-select control register is programmed.

2) The configuration register multiplexes the chip-select with a GPIO. The GPIO module drives the GPIO active low when access to the NAND flash device is needed.

3) The configuration register programs the FLASH.RDY signal to be a GPIO input for the generation of interrupts on the rising edge of the NAND flash device R/B signal.

4) The processor programs and enables an interrupt on the rising edge of the GPIO multiplexed on the R/B signal to signify when the NAND flash has completed programming.

5) The processor initiates a write command (80h= serial data input) and an address write to the NAND flash device.

6) The NAND flash device is now ready to receive a block of data.

Step 3: Write Data to NAND Flash Device

1) The MPU programs the system DMA:

   - Transfer the data in 8-, 16-, or 32-bit word format from SDRAM to the NAND flash peripheral.
   - Transfer mode (single mode= > channel stop when current transfer finishes)
   - The DMA creates an interrupt on completion of the transfer.
   - Transfer start (software request)

2) Data are written into the NAND flash device.

3) When all of the data for a block are written into the device, the processor initiates an auto program command (10h = auto program).

4) The R/B signal goes low to signify that the NAND flash is busy with the internal programming operation (t_{PROG}).

5) The R/B signal goes high, creating an interrupt to the processor that signifies that the block write has been completed.

6) The processor initiates a status read command (70h= status read).

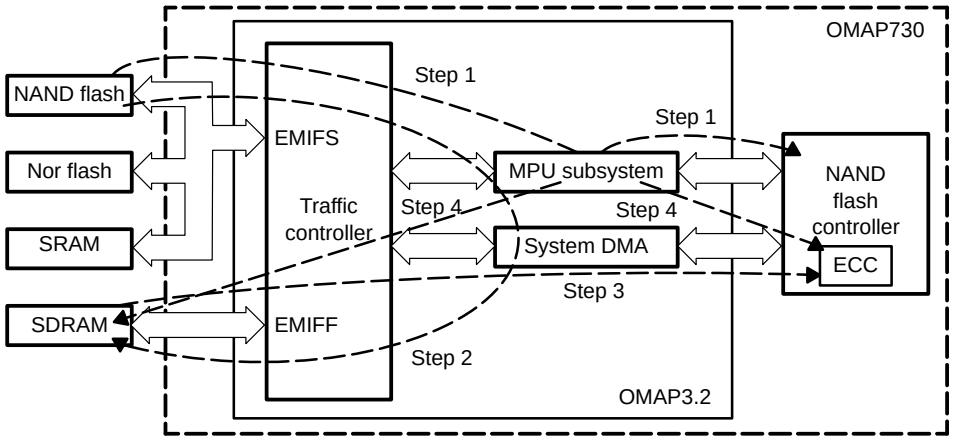
7) The processor reads the NAND flash status register (status read command is sent) and takes the necessary action based on the programming status (if there is an error, block management is required).

8) One block write is completed; further block writes can continue from step #2.

13.2.3 Read Data Sequence Example

The high-level sequence of operations that are needed to read data from the NAND flash device is shown in Figure 13–21. The assumption is that the destination data read from the NAND flash device is saved in external SDRAM memory, but this is not a requirement. Data can be saved in any memory space available in the system. The MPU is the processor controlling the read operations.

Figure 13–21. NAND Flash Read Sequence



Step 1: Configure EMIFS, ECC, and NAND Flash for a Read

1) The associated EMIFS chip-select control register is programmed.

2) The configuration register multiplexes the associated CS chip-select with a GPIO. The GPIO module drives the GPIO active low when an access is needed to the NAND flash device.

3) The configuration register programs the FLASH.RDY signal to be a GPIO input for the generation of interrupts on the rising edge of the NAND flash device R/B signal.

4) The MPU programs the system DMA:

   - Transfer the data in 8-,16-, or 32-bit word format from the NAND_FLASH device to SDRAM.
   - Transfer mode (single mode= > channel stop when current transfer finishes)
   - The DMA creates an interrupt on completion of the transfer.

5) The processor MPU configures the NAND flash controller peripheral as follows:

   - Selects the number of blocks to calculate ECC on (1-9) the NND_ECC_SELECT register
   - Selects the type of ECC (256/512 byte blocks) by programming the NND_CTRL register

- Enables the read and program operation: ECC logic-enable by programming bit 0 of the NND_CTRL register
- Enables the clock in the NND_CTRL register bit 18 (power management)
- Configures NAND flash FIFO access (NND_CTRL)
- Sets 1 in NND_DSC_CLK (the module runs full speed for minimum delay in the ECC calculation)
- Resets ECC calculation by writing to NND_RESET

Note:

To optimize the transfer from SDRAM to the NAND flash controller, keep the RDY/BUSY signal of the NAND flash controller at 1.

Step 2: Read Data From NAND Flash Device

1) The processor initiates a read command.
2) The R/B signal goes low to signify that the NAND flash is busy with the internal programming operation (t_{PROG}).
3) The R/B signal goes high, creating an interrupt to the processor that signifies that the NAND flash device is ready to send data.
4) The MPU programs the system DMA:
   - Transfer start (software request)
5) Data are transmitted from the NAND flash device to SDRAM.
6) The DMA interrupts the processor when the transfer is complete:
   - The calculation of ECC can begin.
7) The R/B signal goes low to signify that the NAND flash is busy with the internal operation.
8) The R/B signal goes high, creating an interrupt to the processor that signifies that the block read has been completed (new block can be read).

Step 3: Calculate ECC

1) The MPU programs the system DMA:
   - Transfer the data in 8-, 16-, or 32-bit word format from SDRAM into the NAND flash controller.
   - Transfer mode (single mode= > channel stop when current transfer finishes)
   - The DMA creates an interrupt on completion of the transfer.
   - Transfer start (hardware start => on DMA request)

2) The processor initiates a write command to the NAND flash device (in the NAND flash controller) NND_COMMAND.

3) The data is written into the NAND flash controller peripheral by the DMA in FIFO.

4) ECC is calculated.

5) The DMA interrupts the processor when the transfer is complete.

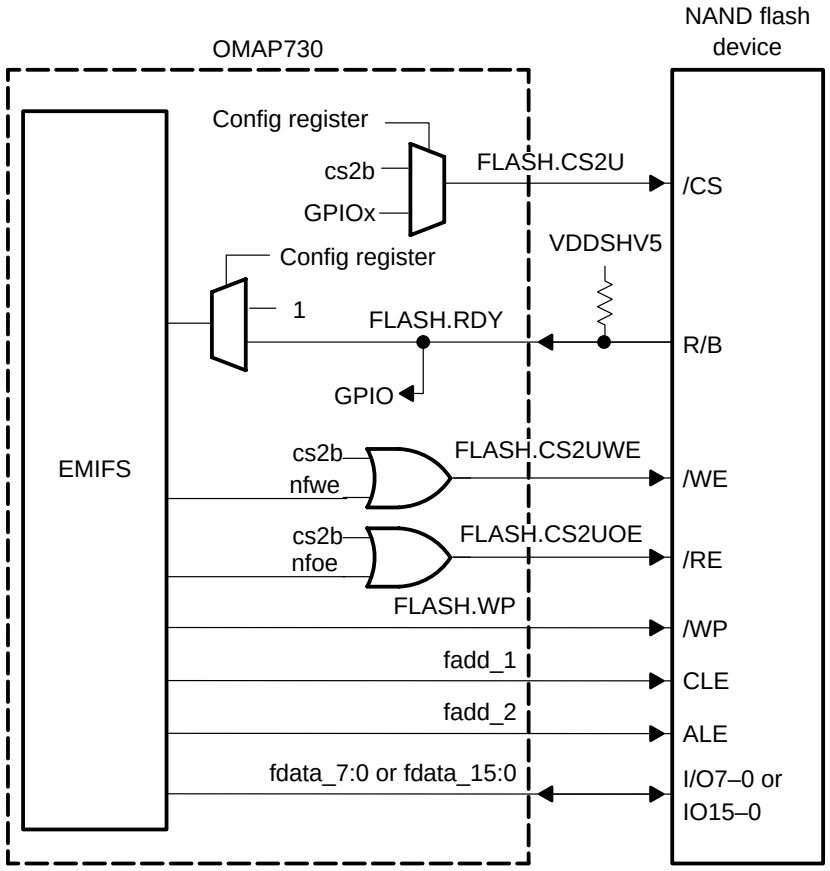
Step 4: Postprocessing

1) The processor reads the ECC calculation from the NAND flash controller peripheral.

2) The processor checks the ECC result with the one save in SDRAM (spare area)

Figure 13–22 shows the interface between the OMAP73X and a NAND flash device.

Figure 13–22. NAND Flash Device Interface Schematic



❑ CS: Most NAND flash devices require that CS be low during the read access time (t_R). For example, the Samsung K9K1G08U0M device requires

CS to be low during t_R. Thus, a standard OMAP73X chip-select cannot be used for the NAND flash chip-select. To resolve this issue, a GPIO is multiplexed on this pin so that the processor can directly control the state of the NAND flash chip-select during accesses. However, some NAND flash devices, such as the Samsung K9F1G08Q0M and K9F1G16U0M, do not require that CS be low during t_R. For these types of flash devices, the OMAP73X chip-select can be used directly.

- R/B: During read or write operations the NAND flash device R/B signal goes low to indicate that the device is busy and that other operations must wait until completion. To signal the OMAP73X that the operation has completed, the R/B signal of the NAND flash device can be connected to the FLASH.RDY that is multiplexed with a GPIO. The GPIO is programmed to create an interrupt on the rising edge of R/B.

Note: FLASH.RDY Signal

The FLASH.NFWAIT signal is primarily intended for use with synchronous burst flash devices. If the system does have a synchronous burst flash device and the FLASH.NFWAIT signal is required, then the NAND flash R/B must be connected on some other GPIO of the OMAP73X device. If another GPIO is used, the interface voltage range must be considered between the open drain output of the NAND flash and the OMAP73X. Otherwise, it is possible to remove this input requirement by the use of timers to create the delay and/or the use polling of the NAND flash device status register.

- $\overline{WE}$ and $\overline{RE}$: As explained before, during t_R some NAND flash devices require that $\overline{CS}$ be low and during this time $\overline{WE}$ and $\overline{RE}$ must not toggle. Thus, it is necessary to gate the OMAP73X chip-select, cs2b, with the NFWE and NFOE signals to create the $\overline{FLASH.CS2UWE}$ and $\overline{FLASH.CS2UOE}$, respectively.

  If the muxed signals are needed in the system, it is possible to generate $\overline{WE}$ and $\overline{RE}$ by GPIO with some performance impact and a more complicated programming model.

- CLE and ALE: The command latch enable (CLE) and address latch enable (ALE) signals are generated by the address pins of OMAP73X FADD_1 and FADD_2, respectively, as shown in Table 13–47.

Table 13–47. CLE and ALE

CLE: FADD_1	ALE: FADD_2	System Address (cs2b)	Function
L	L	0x0A00 0000	Data read or write access
H	L	0x0A00 0002	Command write access
L	H	0x0A00 0004	Address write access
H	H	0x0A00 0006	Non-valid access condition

- I/O7:0 or I/015:0 : NAND flash devices have either 8- or 16-bit-wide data buses. The FDATA_15:0 signals of the OMAP73X connect directly to the NAND flash I/O signals.

ac Specifications and Issues

A review of the Samsung K9K1G08U0M NAND flash device ac specifications versus the EMIFS interface has resulted in the following findings:

- All timings required for write accesses are acceptable. WP ($\overline{WE}$ pulse width) minimum is 25 ns, which implies a maximum rate of 40 MHz.
- All timings required for read accesses are acceptable for interface with EMIFS with the note that t_{AR2} minimum of 50 ns (ALE to $\overline{RE}$ delay-read cycle) must be observed. This can be controlled by the new OE control added to OMAP 3.2.

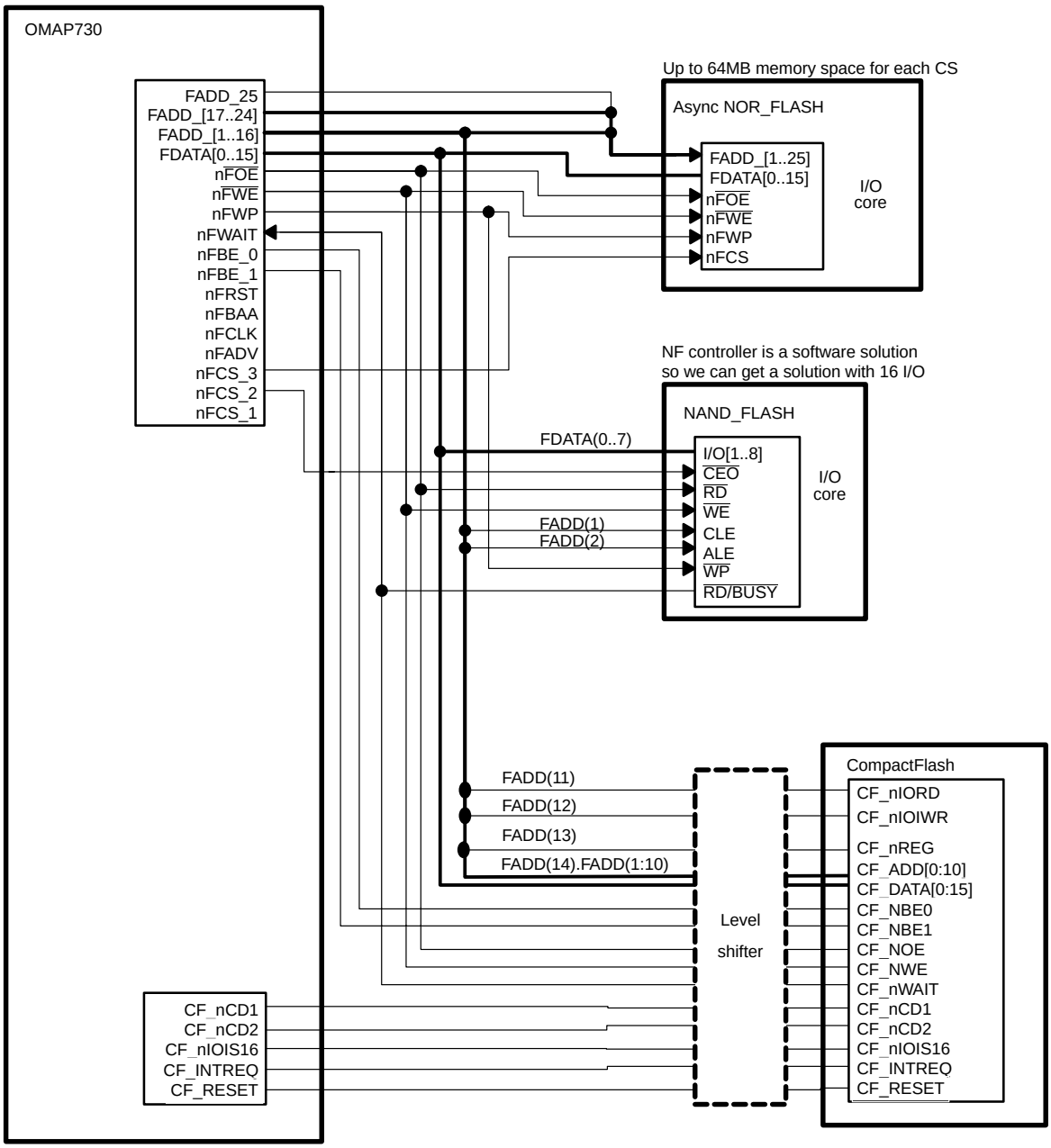
13.2.4 EMIFS Interface With NAND CE Don't Care Flash Device Option

In this case, there is no dedicated signal to interface the NAND flash device. Instead, the standard NOR flash interface is used. Burst flash and CompactFlash can be connected at the same time.

The procedure to follow is the same as that described for NAND CE Don't Care devices.

Figure 13–23 shows how CompactFlash, NAND flash, and an asynchronous NOR flash can be connected.

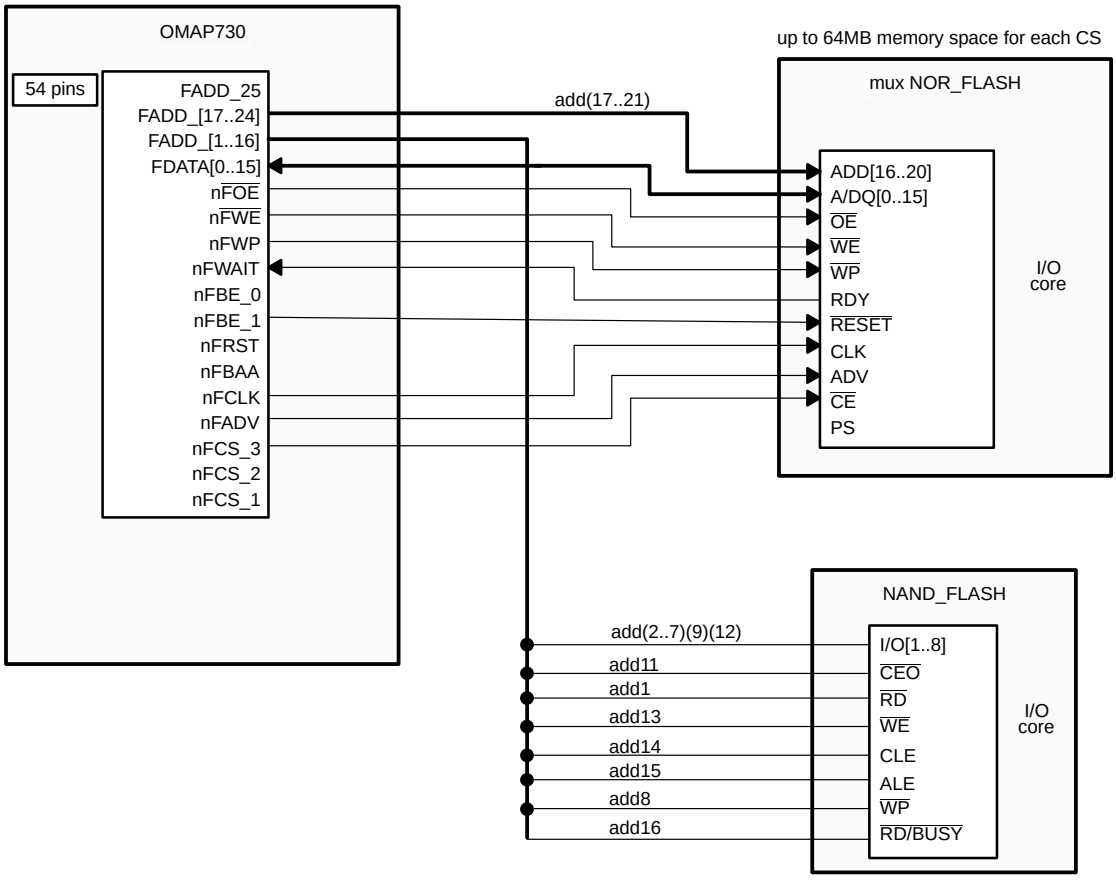
Figure 13–23. Flash Connections



13.2.5 NAND Flash Controller Peripheral/NOR Flash Add-On Option

If this solution is chosen, the NOR flash interface must be configured with ADDRESS and DATA multiplexed. This solution reuses ADD(1:16) for the hardware NAND flash controller.

Figure 13–24. Flash Add On Option



Chapter 14

MPU-S I²C Serial Interface

This chapter describes the I²C serial interface of the OMAP730 multimedia processor.

Topic **Page**

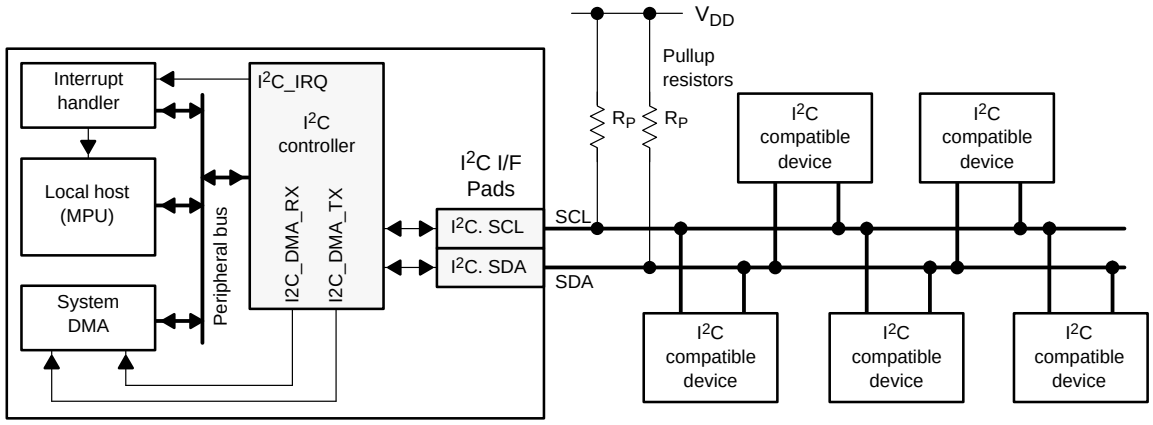
14.1 I²C Master/Slave

14.1.1 Overview

The multimaster I²C peripheral provides an interface between a local host (LH) such as an MPU, MIPS, or DSP processor and any I²C-bus-compatible device that connects via the I²C serial bus. External components attached to the I²C bus can serially transmit/receive up to 8-bit data to/from the LH device through the two-wire I²C interface.

This I²C peripheral supports any slave or master I²C-compatible device. Figure 14–1 shows the example of a system with multiple I²C compatible devices in which the I²C serial ports are connected together for a two-way transfer from one device to other devices.

Figure 14–1. I²C System Overview



14.1.2 Functional Overview

The I²C bus is a multimaster bus. The I²C controller supports the multimaster mode that allows more than one device capable of controlling the bus to be connected to it. Each I²C device is recognized by a unique address and can operate as either transmitter or receiver, according to the function of the device. In addition to being a transmitter or receiver, a device connected to the I²C bus can also be considered as master or slave when performing data transfers. A master device is a device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. During this transfer, any device addressed by this master is considered a slave.

14.1.3 I²C Controller Features

The main features of the I²C controller are:

- ❏ Compliance with Philips I²C specification version 2.1, January 2000
- ❏ Standard mode (up to 100K bits/s) and fast mode (up to 400K bits/s) support
- ❏ 7-bit and 10-bit device addressing modes

- ❏ General call
- ❏ Start/restart/stop
- ❏ Multimaster transmitter/slave receiver mode
- ❏ Multimaster receiver/slave transmitter mode
- ❏ Combined master transmit/receive and receive/transmit mode [1]
- ❏ Built-in FIFO for buffered read or write
- ❏ Module enable/disable capability
- ❏ Programmable clock generation
- ❏ 16-bit wide access to maximize bus throughput
- ❏ Low-power design
- ❏ Two DMA channels
- ❏ Wide-interrupt capability

The current I^2C does not support:

- ❏ High-speed (HS) mode for transfer up to 3.4M bits/s
- ❏ C-bus compatibility mode

14.1.4 I^2C Master/Slave Controller Signal Pads

Data are communicated to devices interfacing with the I^2C via the serial data line (SDA) and the serial clock line (SCL). These two wires carry information between the MPU device and other devices connected to the I^2C bus. Both SDA and SCL are bidirectional pins. They must be connected to a positive supply voltage via a pull-up resistor. When the bus is free, both pins are high. The driver of these two pins has an open drain to perform the required wired-AND function.

Table 14–1. Signal Pads

Name	Type	Reset Value	Description
I2C_SCL	In/ Out(OD)	Input	I^2C serial CLK line Open-drain output buffer. Requires external pull-up resistor (Rp).
I2C_SDA	In/ Out(OD)	Input	I^2C serial data line Open-drain output buffer. Requires external pull-up resistor (Rp).

14.1.5 I²C Reset

The I²C module can be reset in the following three ways:

- A system bus reset (RESET_ = 0). A device reset causes the system bus reset.
- A software reset by setting the SRST bit in the I2C_SYSC register. This bit has the same action on the module logic as the system bus reset.
- The I2C_EN bit in the I2C_CON register can also reset a part of the I²C module. When the system bus reset is released (RESET_ = 1), I2C_EN = 0 keeps the functional part of the I²C module in reset state and all configuration registers can be accessed.

Table 14–2. Reset State of I²C Signals

Pin	I/O/Z	System Reset	I²C Reset (I2C_EN = 0)
SDA	I/O/Z	High impedance	High impedance
SCL	I/O/Z	High impedance	High impedance

14.1.6 I²C Bit Transfer

The master device generates one clock pulse for each data bit transferred. Because of the variety of technology devices (CMOS, NMOS, bipolar) that can be connected to the I²C bus, the levels of logical 0 (low) and 1 (high) are not fixed and depend on the associated level of VDD. See Table 14–3 for electrical specifications.

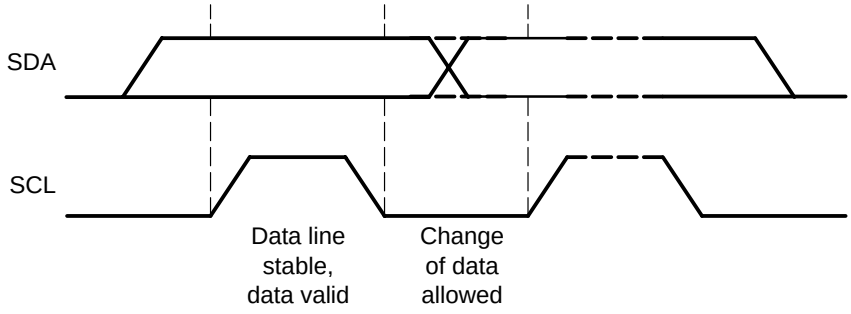
Table 14–3. Electrical Specification of the Input/Output

	Parameter	Standard Mode Devices		Fast-Mode Devices		Unit
		Min	Max	Min	Max	
VIL	Low-level input voltage:					
	Fixed input levels	−0.5	1.5	n/a	n/a	V
	VDD-related input levels	−0.5	0.3VDD	−0.5	0.3VDD	
VIH	High-level input voltage:					
	Fixed input levels	3.0	VDDmax+0.5	n/a	n/a	V
	VDD-related input levels	0.7VDD	VDDmax+0.5	0.7VDD	VDDmax+0.5	
	Low-level output voltage:					
	VDD>2V					
VOL1	At 3mA sink current	0	0.4	0	0.4	V
VOL2	At 6mA sink current	n/a	n/a	0	0.6	
	VDD<2V					
VOL3	At 3mA sink current	n/a	n/a	0	0.2VDD	

14.1.7 Data Validity

The data on the SDA line must be stable during the high period of the clock. The high and low states of the data line can change only when the clock signal on the SCL line is low.

Figure 14–2. Bit Transfer on the I²C Bus



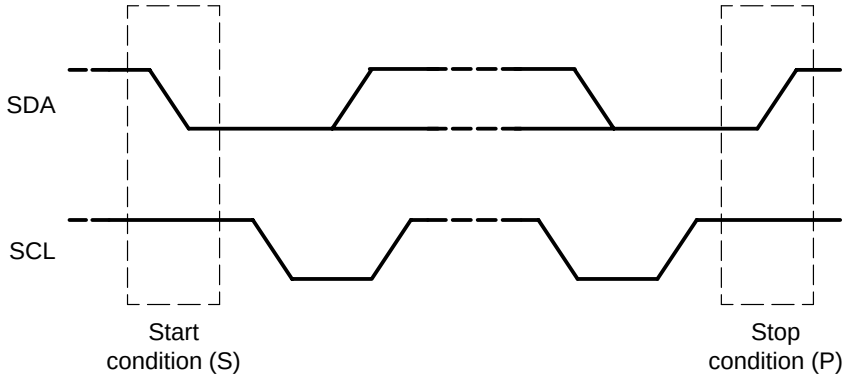
14.1.8 START and STOP Conditions

The I²C module generates start and stop conditions when it is configured as a master:

- START condition is a high-to-low transition on the SDA line while SCL is high.
- STOP condition is a low-to-high transition on the SDA line while SCL is high.

The bus is considered to be busy after the START condition (BB = 1) and free after the STOP condition (BB = 0).

Figure 14–3. Start and Stop Condition Events

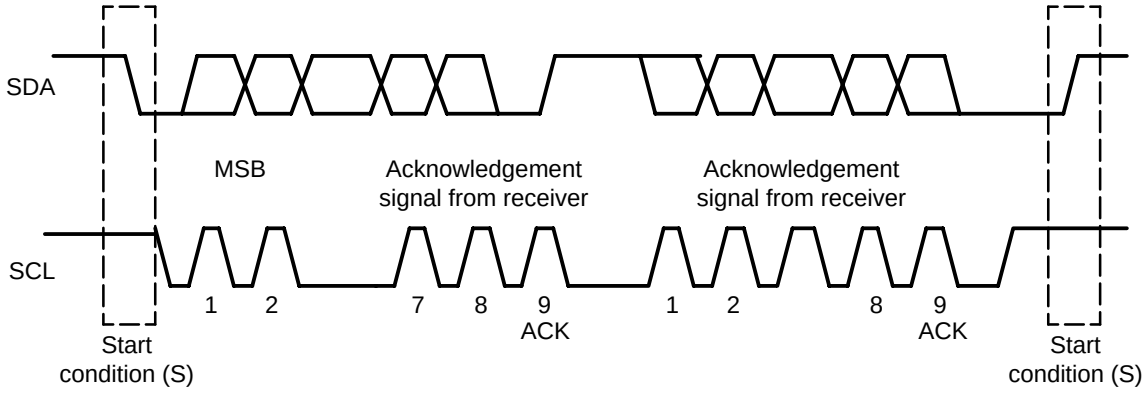


14.2 I²C Operation

14.2.1 Serial Data Formats

The I²C controller operates in 16-bit word data format (byte write access supported for the last access). Each byte put on the SDA line is 8 bits long. The number of bytes that can be transmitted or received is unrestricted. The data are transferred with the most-significant bit (MSB) first. Each byte is followed by an acknowledge bit from the I²C module, if it is in receiver mode. The I²C controller supports endianism.

Figure 14–4. I²C Data Transfer



The I²C module supports two data formats, as shown in Figure 14–5:

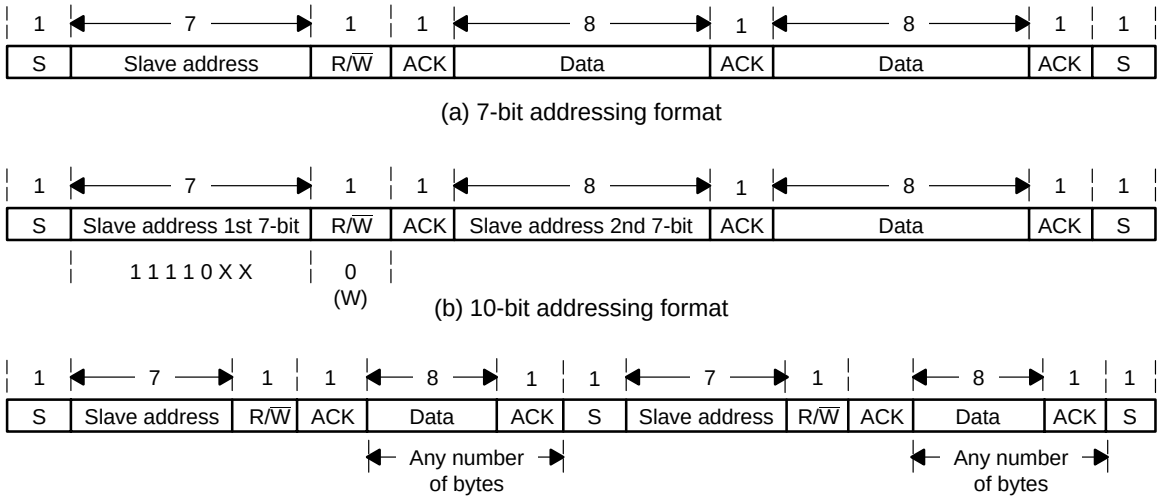
- 7-bit/10-bit addressing format
- 7-bit/10-bit addressing format with repeated start condition

The first byte after a start condition (S) always consists of 8 bits. In the acknowledge mode, an extra bit dedicated for acknowledgement is inserted after each byte.

In the addressing formats with 7-bit addresses, the first byte is composed of 7 MSB slave-address bits and 1 LSB R/W_ bit. In the addressing formats with 10-bit addresses, the first byte has 7 MSB slave address bits, such as 11110XX where XX is the two MSB of the 10-bit addresses and 1 LSB R/W_ bit, which is 0 in this case.

The least-significant R/W_ of the address byte indicates the direction of transmission of the following data bytes. If R/W_ is 0, the master writes data into the selected slave; if it is 1, the master reads data out of the slave.

Figure 14–5. I^2C Data Transfer Formats



14.2.1.1 Master Transmitter

In this mode, data assembled in one of the previously described data formats is shifted out on the serial data line SDA in synch with the self-generated clock pulses on the serial clock line SCL. The clock pulses are inhibited and SCL is held low when the intervention of the processor is required (XUDF) after a byte has been transmitted.

14.2.1.2 Master Receiver

This mode can be entered only from the master transmitter mode. With either of the address formats (Figure 14–5 (a), (b), and (c)), the master receiver is entered after the slave address byte and bit R/W_ have been transmitted, if R/W_ is high. Serial data bits received on bus line SDA are shifted in synch with the self-generated clock pulses on SCL. The clock pulses are inhibited and SCL held low when the intervention of the processor is required (ROVR) after a byte has been transmitted. At the end of a transfer, it generates the stop condition.

14.2.1.3 Slave Transmitter

This mode can be entered only from the slave receiver mode. With either of the address formats (Figure 14–5 (a), (b), and (c)), the slave transmitter is entered if the slave address byte is the same as its own address and bit R/W_ has been transmitted, if R/W_ is high. The slave transmitter shifts the serial data out on the data line SDA in synch with the clock pulses that are generated by the master device. It does not generate the clock, but it can hold clock line SCL low while the local host is required to intervene (XUDF).

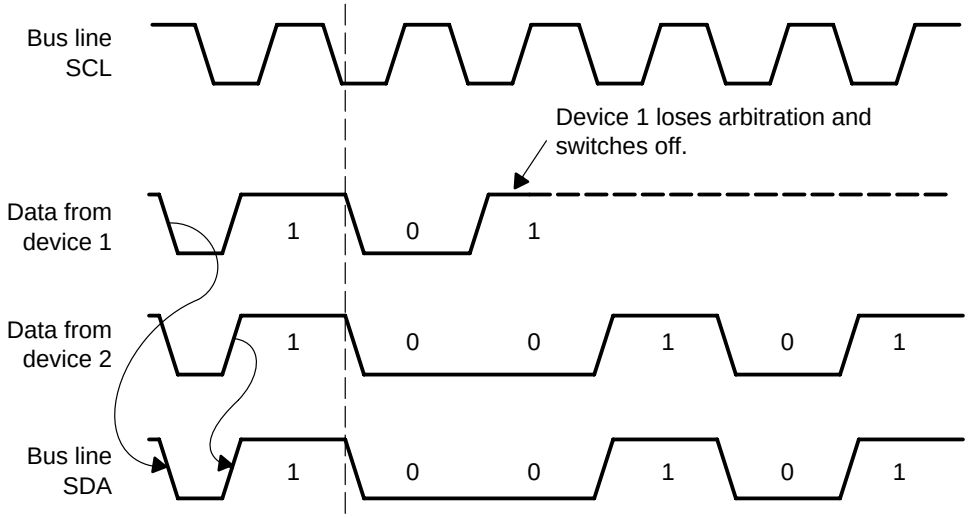
14.2.1.4 Slave Receiver

In this mode, serial data bits received on the bus line SDA are shifted-in in synch with the clock pulses on SCL that are generated by the master device. It does not generate the clock, but it can hold the clock line SCL low while the local host is required to intervene (ROVR) following the reception of a byte.

14.2.1.5 Arbitration

If two or more master transmitters start a transmission on the same bus almost simultaneously, an arbitration procedure is invoked. The arbitration procedure uses the data presented on the serial bus by the competing transmitters. When a transmitter senses that a high signal it has presented on the bus has been overruled by a low signal, it switches to the slave receiver mode, sets the arbitration lost (AL) flag, and generates the arbitration lost interrupt. Figure 14–6 shows the arbitration procedure between two devices. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. Should two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

Figure 14–6. Arbitration Procedure Between Two Master Transmitters

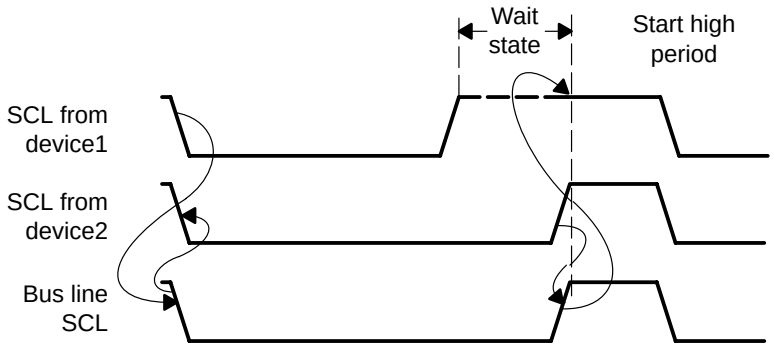


14.2.1.6 I^2C Clock Generation and I^2C Clock Synchronization

Under usual conditions, only one master device generates the clock signal, SCL. During the arbitration procedure, however, two or more master devices and the clock must be synchronized so that the data output can be compared. The wired-AND property of the clock line means that a device that first generates a low period of the clock line overrules the other devices. At this high/low transition, the clock generators of the other devices are forced to start generation of their own low period. The clock line is then held low by the device with the longest low period, while the other devices that finish their low periods must wait for the clock line to be released before starting their high periods. A synchronized signal on the clock line is thus achieved, where the slowest device determines the length of the low period and the fastest the length of the high period.

If a device pulls down the clock line for a longer time, the result is that all clock generators must enter the WAIT state. In this way, a slave can slow down a fast master, and the slow device can create enough time to store a received byte or prepare a byte to be transmitted. Figure 14–7 illustrates the clock synchronization.

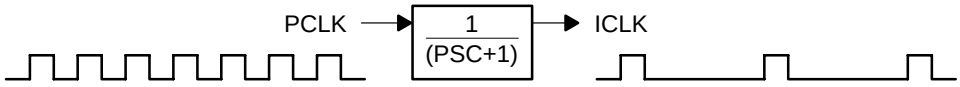
Figure 14–7. Synchronization of Two I²C Clock Generators



14.2.1.7 Prescaler (SCLK/ICLK)

The I²C module is operated with an internal, approximately12-MHz clock (ICLK). This clock is generated via the I²C prescaler block. The prescaler consists of an 8-bit register. I²C _PSC is used for dividing down the system clock (SCLK) to obtain an approximately12-MHz clock for the I²C module.

Figure 14–8. Synchronization of Two I²C Clock Generators



0x0:	Divide by 1
0x1:	Divide by 2
↓	↓
0xFF:	Divide by 256

Values after reset are low (All 8-bits).

14.2.1.8 Noise Filter

The noise filter suppresses any noise that is 50 ns or less. It is designed to suppress noise with 1 ICLK, assuming the lower and upper limits of ICLK are 8 MHz and 16 MHz, respectively.

14.2.2 I²C Interrupts

The I²C module generates six types of interrupt: arbitration-lost, no-acknowledge, general call, registers-ready-for-access, receive, and transmit. These six interrupts are accompanied by six interrupt masks and flags defined in the I2C_IE and I2C_STAT registers, respectively.

- Arbitration lost interrupt (AL) is generated when the I²C arbitration procedure is lost.
- No-acknowledge interrupt (NACK) is generated when the master I²C does not receive an acknowledge from the receiver.
- General call interrupt (GC) is generated when the device detects the address of all zeros (8 bits).
- Registers-ready-for-access interrupt (ARDY) is generated by the I²C when the previously programmed address, data, and command have been performed and the status bits have been updated. This interrupt is used to let the LH know that the I²C registers are ready for access.
- Receive interrupt/status (RRDY) is generated when there is received data ready to be read by the LH from the I2C_DATA register. The LH can poll this bit to read the received data from the I2C_DATA register.
- Transmit interrupt/status (XRDY) is generated when the LH needs to put more data in the I2C_DATA register after the transmitted data has been shifted out on the SDA pin. The LH can poll this bit to write the next transmitted data into the I2C_DATA register.

When the interrupt signal is activated, the local host must read the I2C_STAT register to define the type of interrupt, process the request, and then write into this register the correct value to clear the interrupt flag.

14.2.3 DMA Events

The I²C module can generate two DMA requests events, read (I2C_DMA_RX) and write (I2C_DMA_TX), that the DMA controller can use to synchronously read received data from the I2C_DATA and write transmitted data to the I2C_DATA register. The DMA read and write requests are generated in a similar manner as RRDY and XRDY, respectively.

The I²C DMA request signals (I2C_DMA_TX and I2C_DMA_RX) are activated for every new 16-bit word to be read or written in the FIFOs.

14.3 I²C Registers

Table 14–4 lists the I²C registers. Table 14–5 through Table 14–22 describe the register bits.

Start address: FFFB 3800

Table 14–4. I²C Registers

Name	Description	R/W	Offset
I2C_REV	I²C module revision		0x00
I2C_IE	I²C interrupt enable		0X0
I2C_STAT	I²C status		0X0
RESERVED	Reserved		0X0
I2C_SYSS	I²C system status		0X0
I2C_BUF	I²C buffer configuration		0X0
I2C_CNT	I²C data counter		0X0
I2C_DATA	I²C data access		0X0
I2C_SYSC	I²C system configuration		0X0
I2C_CON	I²C configuration		0X0
I2C_OA	I²C own address		0X0
I2C_SA	I²C slave address		0X0
I2C_PSC	I²C clock prescaler		0X0
I2C_SCLL	I²C SCL low time control		0X0
I2C_SCLH	I²C SCL high time control		0X0
I2C_SYSTEST	I²C system test		0X0

All bits defined as reserved must be written by software with zeros to preserve future compatibility. When read, any reserved bit returns 0. Also, note that it is good software practice to use complete mask patterns for setting or testing bit fields individually within a register.

Table 14–5. I²C Module Revision Register (I2C_REV)

Bit	Name	Description
15:8	–	Reserved
7:0	REV	Module version number This 8-bit field indicates the revision number of the current I²C controller module. Its value is fixed by hardware and corresponds to the RTL revision of this module. The four LSBs indicate a minor revision. The four MSBs indicate a major revision. For example: 0x20: Revision 2.0 0x21: Revision 2.1 A reset has no effect on the value returned.

I2C_REV (read-only) contains the hard-coded revision number of the module. A write to this register has no effect.

Note:

I²C controller with interrupt using interrupt vector register (I2C_IV) is revision 1.x

I²C controller with interrupt using status register bits (I2C_STAT) is revision 2.x

Table 14–6. I²C Interrupt Enable Register (I2C_IE)

Bit	Name	Description
15:6	–	Reserved
5	GC_IE	General call interrupt enable
4	XRDY_IE	Transmit data ready interrupt enable
3	RRDY_IE	Receive data ready interrupt enable
2	ARDY_IE	Register access ready interrupt enable
1	NACK_IE	No acknowledgment interrupt enable
0	AL_IE	Arbitration lost interrupt enable

The I²C interrupt enable register is a R/W register that controls the interrupt mask/unmask function.

The following are common to all bits:

When the local host sets a bit location to 1 , an interrupt is signaled to the local host if the corresponding bit location in I2C_STAT (status register) is asserted to 1 by the core of the I²C controller. If it is set to 0, the interrupt is masked and is not signaled to the local host.

- 0: Interrupt disabled
- 1: Interrupt enabled

Value after reset is low (all bits).

Table 14–7. I^2C Status Register (I2C_STAT)

Bit	Name	Description
15	SBD	Single byte data This read-only bit is set to 1 in slave-receive or master-receive modes when the last byte that was read from the I2C_DATA register contains a single valid byte. The core clears this bit to 0 when the local host clears the register access ready interrupt flag. **NOTE:** When SBD = 1, in little-endian data format (I2C_CON:BE = 0) the MSB reads as 0x00, and in big-endian format (I2C_CON:BE = 1) the LSB reads as 0x00. Whenever the number of bytes to be received is unknown (for example, slave receiver), the LH must poll this bit before clearing the register access ready interrupt flag. 0: No action 1: Single valid byte in last 16-bit data read Value after reset is low.
14:13	–	Reserved
12	BB	Bus busy This read-only bit indicates the state of the serial bus.In slave mode, on reception of a start condition, the device sets BB to 1. BB is cleared to 0 after reception of a stop condition. In the master mode, the software controls BB. To start a transmission with a start condition, MST, TRX, and STT must be set to 1 in the I2C_CON register. To end a transmission with a stop condition, STP must be set to 1 in the I2C_CON register. When BB = 1, and STT is set to a 1, a restart condition is generated. 0: Bus is free. 1: Bus is occupied. Value after reset is low.
11	ROVR	Receive overrun (receive mode only) This read-only bit indicates whether the receiver has experienced overrun. Overrun occurs when the shift register is full and the receive FIFO is full. An overrun condition does not result in a data loss; the peripheral is just holding the bus (low on SCL) to prevent other bytes from being received. ROVR is set to 1 when the I^2C recognizes an overrun. ROVR is clear when reading the I2C_DATA register or when resetting the I^2C (I2C_CON:I2C_EN = 0). 0: Normal operation 1: Receiver overrun Value after reset is low.

Table 14–7. I^2C Status Register (I2C_STAT) (Continued)

Bit	Name	Description
10	XUDF	Transmit underflow (transmit mode only) This read-only bit indicates whether the transmitter has experienced underflow. In the master transmit mode, underflow occurs when the shift register is empty, the transmit FIFO is empty, and there are still some bytes to transmit (DCOUNT ≠ 0). In the slave transmit mode, underflow occurs when the shift register is empty, the transmit FIFO is empty, and there are still some bytes to transmit (read request from external I^2C master). XUDF is set to 1 when the I^2C recognizes an underflow. The core holds the line until the underflow cause disappears. XUDF is clear when writing the I2C_DATA register or resetting the I^2C (I2C_CON:I2C_EN = 0). 0: Normal operation 1: Transmit underflow Value after reset is low.
9	AAS	Address as slave The device sets this bit to 1 when it recognizes its own slave address or an address of all zeros (8 bits). The AAS bit is reset to 0 by restart or stop. 0: No action 1: Address as slave Value after reset is low.
8:6	–	Reserved
5	GC	General call The device sets this read-/clear-only bit to1 if it detects the address of all zeros (8 bits), that is, general call. When the core sets this bit to 1, an interrupt is signaled to the local host if the interrupt was enabled. The LH is able to clear this bit only by writing a 1 into this register. Writing 0 has no effect. When this bit is set to 1, AAS also reads as 1. 0: No action 1: General call Value after reset is low.

Table 14–7. I^2C Status Register (I2C_STAT) (Continued)

Bit	Name	Description
4	XRDY	Transmit data ready (transmit mode only) This read-/clear-only bit (XRDY) is set to 1 when the I^2C peripheral is a master or slave transmitter, the LH is able to write new data into the I2C_DATA register, and the transmitter still requires new data. A master transmitter requests new data if DCOUNT ≠ 0 , and a slave transmitter requests new data if a read request is received from external master. **NOTE:** The transmitter requests two bytes to be written even if only a single byte is needed. In this case, the other byte needs to be filled with a dummy 0x00 value that is not transmitted over the I^2C line. When the core sets this bit to 1, an interrupt is signaled to the local host if the interrupt was enabled. The LH can also poll this bit to write new transmitted data into the I2C_DATA register. The LH is able to clear this bit only by writing a 1 into this register. Writing 0 has no effect. If the DMA transmit mode is enabled, this bit is not set. Instead, a DMA TX request to the main DMA controller of the system is generated. 0: Transmit buffer full (or receiver mode) 1: Transmit data ready (for write) and byte is needed Value after reset is low.
3	RRDY	Receive data ready This read-/clear-only RRDY is set to 1 when the LH is able to read new data from the I2C_DATA register. When the core sets this bit to 1, an interrupt is signaled to the local host if the interrupt was enabled. The LH can also poll this bit to read the received data in I2C_DATA register. The LH is able to clear this bit only by writing a 1 into this register. Writing 0 has no effect. In interrupt mode, the LH needs to poll this bit after each read to I2C_DATA to ensure that there is no other DATA on the FIFO waiting to be read. Indeed, the RRDY needs to be cleared to 0 in order to receive a new RRDY interrupt. If the DMA receive mode is enabled, this bit is not set. Instead, a DMA RX request to the main DMA controller of the system is generated. 0: Receive buffer empty 1: Receive data ready (for read) Value after reset is low.
2	ARDY	Register access ready. See Table 14–8. This read-/clear-only bit when set to 1 indicates that the previously programmed data and command (receive or transmit, master or slave) has been performed and the status bit has been updated. The LH uses this flag to let it know that the I^2C registers are ready to be accessed again. The LH is able to clear this bit only by writing a 1 into this register. Writing 0 has no effect. 0: No action 1: Access ready Value after reset is low.

Table 14–7. I²C Status Register (I2C_STAT) (Continued)

Bit	Name	Description
1	NACK	No acknowledgment interrupt enable The read-/clear-only NO_ACKNOWLEDGE flag bit is set when the hardware detects NO_ACKNOWLEDGE has been received. The LH is able to clear this bit only by writing a 1 into this register. Writing 0 has no effect. 0: Normal/no action required 1: NACK Value after reset is low.
0	AL	Arbitration lost interrupt enable The read-/clear-only ARBITRATION_LOST flag bit is set to 1 when the device in the master transmitter mode senses it has lost an arbitration. This occurs when two or more transmitters start a transmission almost simultaneously, or when the I²C attempts to start a transfer while BB (bus busy) is 1. When this is set to 1 because of arbitration lost, the core automatically clears the MST/STP bits in the I2C_CON register and the I²C becomes a slave receiver. The LH is able to clear this bit only by writing a 1 to this register. Writing 0 has no effect. 0: Normal/no action required 1: Arbitration lost Value after reset is low.

The I²C status register is composed of read-only and read-/clear-only registers. It provides core status information for interrupt handling and other I²C control management.

Table 14–8. ARDY Set Conditions

Mode	Others	ARDY Set Conditions
Master transmit	STP = 1	DCOUNT = 0
Master receive	STP = 1	DCOUNT = 0 and receiver FIFO empty
Master transmit or receive	STP = 0	DCOUNT passed 0
Slave transmit	–	Stop or restart condition received from master
Slave receive	–	Stop or restart condition and receiver FIFO empty

DCOUNT is the data count value in the I2C_CNT register.

Table 14–9. I²C System Status Register (I2C_SYSS)

Bit	Name	Description
15:1	–	Reserved
0	RDONE	Reset Done This read-only bit indicates the state of the reset in case of hardware reset, global software reset (I2C_SYSC.SRST), or partial software reset (I2C_CON.I2C_EN). The module must receive all of its clocks before it can grant a reset-completed status. 0: Internal module reset is ongoing or partially held in reset 1: Reset completed Value after reset is low.

Table 14–10. I²C Buffer Configuration Register (I2C_BUF)

Bit	Name	Description
15	RDMA_EN	Receive DMA channel enable When this bit is set to 1, the receive DMA channel is enabled and the core forces the receive data ready status bit (I2C_STAT:RRDY) to 0. 0: Receive DMA channel disabled 1: Receive DMA channel enabled Value after reset is low.
14:8	–	Reserved
7	XDMA_EN	Transmit DMA channel enable When this bit is set to 1, the transmit DMA channel is enabled and the core forces the transmit data ready status (I2C_STAT:XRDY) bit to 0. 0: Transmit DMA channel disabled 1: Transmit DMA channel enabled Value after reset is low.
6:0	–	Reserved

The I²C buffer configuration register is a R/W register that enables DMA transfers.

Table 14–11. I^2C Data Counter Register (I2C_CNT)

Bit	Name	Description
15:0	DCOUNT	Data count Master modes only (receive or transmit). This 16-bit countdown counter decrements by 1 for every byte received or sent. A write initializes DCOUNT to a saved initial value. A read returns the number of bytes that are yet to be received or sent. A read into DCOUNT returns the initial value only before a start condition and after a stop condition. When DCOUNT reaches 0, the core generates a stop condition if a stop condition was specified (I2C_CON:STP = 1), and the ARDY status flag is set to 1 in the I2C_STAT register. If I2C_CON:STP = 0, then the I^2C asserts SCL = 0 when DCOUNT reaches 0. The LH can then reprogram DCOUNT to a new value and resume sending or receiving data with a new start condition (restart). This process repeats until the LH sets the STP to 1 in the I2C_CON register. The ARDY flag is set each time DCOUNT reaches 0 and DCOUNT is reloaded to its initial value. In slave mode (receive or transmit), DCOUNT is not used. 0x0: Data counter = 65536 bytes (2^{16}) 0x1: Data counter = 1 bytes ↓ ↓ 0xFFFF: Data counter = 65535 bytes (2^{16} –1) DCOUNT value after reset is 0x0000.

The I^2C data counter register is a R/W register that controls the number of bytes in the I^2C data payload.

Table 14–12. I²C Data Access Register (I2C_DATA)

Bit	Name	Description
15:0	DATA	Transmit/Receive FIFO data When read, this register contains the received I²C data packet (1 or 2 bytes). This register must be accessed 16-bit-wise by the LH. In case of an odd number of bytes received to read, the upper byte (with I2C_CON.BE = 0) or the lower byte (with I2C_CON.BE = 1) of the last access always reads as 0x00. The LH must check the SBD status bit in I2C_STAT register in order to flush this null byte. When written, this register contains the byte(s) value(s) to transmit over the I²C data line (1or 2 bytes). This register must be accessed 16-bit-wise, except for the last byte in case of an odd number of bytes to transmit. The last byte of the data packet can be written using a byte write access or a 16-bit-wise access. In the 16-bit-wise access, the module transmits only the relevant byte, based on the byte counter (I2C_CNT). This feature is useful for DMA access, which supports only one word size per channel. In SYSTEST loopback mode (I2C_SYSTEST:TMODE = 11), this register is also the entry/receive point for the data. Values after reset is low (all 16-bits). A read access when the buffer is empty returns the previous read data value. A write access when the buffer is full is ignored. In both events, the FIFO pointers are not updated and a remote access error (hardware error) is generated (access qualifier). No remote error is generated if the local host performs a 16-bit access if the buffer contains a single byte.

The I²C data access register is the entry point for the local host to read data from or write data to the FIFO buffer. The FIFO size is 2 x 16 bits (4 bytes). Bytes within a word are stored and read in little-endian format (I2C_CON:BE = 0) or big-endian format (I2C_CON:BE = 1).

Table 14–13. I²C System Configuration Register (I2C_SYSC)

Bit	Name	Description
15:2	–	Reserved
1	SRST	Soft Reset When this bit is set to 1, all of the module is reset, as for the hardware reset. The core automatically sets this bit to 0, and it is only reset by the hardware reset. During reads, it always returns 0. 0: Normal mode 1: The module is reset Values after reset is low.
0	–	Reserved

Table 14–14. I²C Configuration Register (I2C_CON)

Bit	Name	Description
15	I2C_EN	I²C module enable When this bit is set to 0, the I²C controller is not enabled and reset. When 0, the receive and transmit FIFOs are cleared and all status bits are set to their default values. None of the configuration registers (I2C_IE, I2C_BUF, I2C_CNT, I2C_CON, I2C_OA, I2C_SA, I2C_PSC, I2C_SCLL, and I2C_SCLH) are reset, all keep their initial values, and all can be accessed. The LH must set this bit to 1 for normal operation. 0: I²C controller in reset 1: I²C module enabled Value after reset is low.
14	BE	Big-endian mode When this bit is 0 (default), the FIFO is accessed in little-endian format. In transmit mode, the LSB (I2C_DATA [7:0]) is transmitted first, and the MSB (I2C_DATA [15:8]) is transmitted in second position over the I²C line. Conversely, in receive mode, the first, or odd, byte received (1,3, 5…) is stored in the LSB position, and the second, or even, byte received is stored in the MSB position. When the LH sets this bit to a 1, the FIFO is accessed in big-endian format. In transmit mode, the MS byte (I2C_DATA [15:8]) is transmitted first and the LSB (I2C_DATA [7:0]) is transmitted in second position over the I²C line. Conversely, in receive mode, the first, or odd, byte received (1,3, 5…) is stored in the MS byte position, and the second, or even, byte received is stored in the LSB position. 0: Little-endian mode 1: Big-endian mode Value after reset is low.
13:12	–	Reserved
11	STB	Start byte mode (master mode only) The local host sets the start-byte mode bit to 1 to configure the I²C in start byte mode (I2C_SA = 00000001). See the Philips I²C specification Version 2.1 for more details. 0: Normal mode 1: Start byte mode Value after reset is low.
10	MST	Master/slave mode When this bit is cleared, the I²C controller is in slave mode and the serial clock (SCL) is received from the master device. When this bit is set, the I²C controller is in the master mode and it generates the serial clock. This bit is automatically cleared at the end of the transfer on a detected stop condition and in case of arbitration lost. 0: Slave mode 1: Master mode Value after reset is low.

Table 14–14. I^2C Configuration Register (I2C_CON) (Continued)

Bit	Name	Description
9	TRX	Transmitter/receiver mode (master mode only) When this bit is cleared, the I^2C controller is in receiver mode, and data on data line SDA is shifted into the receiver FIFO and can be read from the I2C_DATA register. When this bit is set, the I^2C controller is in transmitter mode, and the data written in the transmitter FIFO via I2C_DATA is shifted out on data line SDA. 0: Receiver mode 1: Transmitter mode Value after reset is low. The operating modes are defined in Table 14–15.
8	XA	Expand address When set, this bit expands the address to 10-bit. 0: 7-bit address mode 1: 10-bit address mode Value after reset is low.
7:2	–	Reserved
1	STP	Stop condition (master mode only). See Table 14–16. The local host is able to set this bit to a 1 to generate a stop condition. The hardware resets it to 0 after the stop condition has been generated. The stop condition is generated when DCOUNT passes 0. When this bit is not set to 1 before the end of the transfer (DCOUNT = 0), the stop condition is not generated and the SCL line is held to 0 by the master, which can restart a new transfer by setting the STT bit to 1. 0: No action or stop condition detected 1: Stop condition queried Value after reset is low.
0	STT	Start condition (master mode only). See Table 14–16. The local host is able to set this bit to a 1 to generate a start condition. The hardware resets it to 0 after the start condition has been generated. The start/stop bits can be configured to generate different transfer formats. 0: No action or start condition generated 1: Start Value after reset is low.

Active Transfer Phase

During an active transfer phase (STT has been set to 1), no modification must be done in this register. Changing it may result in an unpredictable behavior.

Table 14–15. Operating Modes

MST	TRX	Operating Modes
0	x	Slave receiver
0	x	Slave transmitter
1	0	Master receiver
1	1	Master transmitter

Table 14–16. Start/Stop Condition Settings

STT	STP	Conditions	Bus Activities
1	0	Start	S–A–D
0	1	Stop	P
1	1	Start–Stop (DCOUNT = n)	S–A–D..(n)..D–P
1	0	Start (DCOUNT = n)	S–A–D..(n)..D

DCOUNT is the data count value in the I2C_CNT register.

Table 14–17. I^2C Own Address Register (I2C_OA)

Bit	Name	Description
15:10	–	Reserved
9:0	OA	Own address This field specifies either: ❑ A 10-bit address coded on OA [9:0] when XA (Expand Address, I2C_CON [8]) is set to 1. ❑ A 7-bit address coded on OA [6:0] when XA (Expand Address, I2C_CON [8]) is set to 0. In this case, application software must set OA [9:7] bits to 000. Values after reset is low (all 10 bits).

The I^2C own address register specifies the module I^2C 7-bit or 10-bit address (own address).

Table 14–18. I^2C Slave Address Register (I2C_SA)

Bit	Name	Description
15:10	–	Reserved
9:0	SA	Slave address This field specifies either: ❑ A 10-bit address coded on SA [9:0] when XA (Expand Address, I2C_CON [8]) is set to 1. ❑ A 7-bit address coded on SA [6:0] when XA (Expand Address, I2C_CON [8]) is set to 0. In this case, application software must set SA [9:7] bits to 000. Values after reset is high (all 10 bits).

The I^{2}C slave address register specifies the addressed I^{2}C module 7-bit or 10-bit address (slave address).

Table 14–19. I^{2}C Clock Prescaler Register (I2C_PSC)

Bit	Name	Description
15:8	–	Reserved
7:0	PSC	Prescale sampling clock divider value I2C_PSC [7–0]: Prescale Sampling Clock Divider Value (PSC) The core uses this 8-bit value to divide the system clock (SCLK) and generate its own internal sampling clock (ICLK). The core logic is sampled at the clock rate of the system clock for the module, divided by (PSC+1). 0x0: Divide by 1 0x1: Divide by 2 ↓ ↓ 0xFF: Divide by 256 Values after reset is low (all 8 bits).

The I^{2}C clock prescaler register specifies the internal clocking of the I^{2}C peripheral core.

Table 14–20. I^{2}C SCL Low Time Control Register (I2C_SCLL)

Bit	Name	Description
15:8	–	Reserved
7:0	SCLL	SCL low time Master mode only. This 8-bit value generates the SCL low time value (t_{LOW}) when the peripheral is operated in master mode. The SCL low time depends on the I2C_PSC value and the ICLK time period (internal sampling clock rate): ❏ When I2C_PSC = 0, t_{LOW} = (SCLL+7) * ICLK time period ❏ When I2C_PSC = 1, t_{LOW} = (SCLL+6) * ICLK time period ❏ When I2C_PSC > 1, t_{LOW} = (SCLL+5) * ICLK time periodThe different values to compute the SCL low time are due to the synchronization stage and noise filter on the SCL line. Values after reset is low (all 8 bits).

The I^{2}C SCL low time control register determines the SCL low time value when master.

Table 14–21. I²C SCL High Time Control Register (I2C_SCLH)

Bit	Name	Description
15:8	–	Reserved
7:0	SCLH	SCL high time Master mode only. This 8-bit value generates the SCL high-time value (t_{HIGH}) when the peripheral is operated in master mode. The SCL high time depends on the I2C_PSC value and the ICLK time period (internal sampling clock rate): ❑ When I2C_PSC = 0, t_{HIGH} = (SCLH+7) * ICLK time period ❑ When I2C_PSC = 1, t_{HIGH} = (SCLH+6) * ICLK time period ❑ When I2C_PSC > 1, t_{HIGH} = (SCLH+5) * ICLK time periodThe different values to compute the SCL high time are due to the synchronization stage and noise filter on the SCL line. Values after reset is low (all 8 bits).

The I²C SCL high time control register determines the SCL high time value when master.

Table 14–22. I²C System Test Register (I2C_SYSTEST)

Bit	Name	Description
15	ST_EN	System test enable This bit must be set to 1 to permit other system test registers bits to be set. 0: Normal mode 1: System test enabled Value after reset is low.
14	FREE	Free-running mode (on breakpoint) This bit determines the state of the I²C controller when a breakpoint is encountered in the HLL debugger. This bit can be set independently of the ST_EN value. FREE = 0: If the I²C controller is a master, it stops immediately after the completion of the ongoing bit transfer. If the I²C controller is a slave, it stops during the data phase transfer when one byte is completely transmitted/received. FREE = 1: The I²C runs free. 0: Stop mode (on breakpoint condition) 1: Free-running mode Value after reset is low.

Table 14–22. I²C System Test Register (I2C_SYSTEST) (Continued)

Bit	Name	Description
13:12	TMODE	Test mode select. See Table 14–23. In usual functional mode (ST_EN = 0), these bits are *don't care*. They always read as 00 and a write is ignored. In system test mode (ST_EN = 1), these bits can be set according to the following table to permit various system tests. Values after reset is low (2 bits). ❑ SCL counter test mode: In this mode, the SCL pin is driven with a permanent clock as if mastered, with the parameters set in the I2C_PSC, I2C_SCLL, and I2C_SCLH registers. ❑ Loopback mode: In the master transmit mode only, data transmitted out of the I2C_DATA register (write action) is received in the same I2C_DATA register via an internal path through the 1-deep FIFO buffers. The DMA and interrupt requests are usually generated if enabled. ❑ SDA/SCL IO mode: In this mode, the SCL IO and SDA IO are controlled via the I2C_SYSTEST [3:0] register bits.
11	SSB	Set status bits Writing 1 into this bit also sets the six read/clear-only status bits contained in the I2C_STAT register (bits 5:0) to 1. Writing 0 into this bit does not clear status bits that are already set; only writing 1 into a set status bit can clear it. This bit must be cleared before attempting to clear a status bit. 0: No action 1: Set all six bits in I2C_STAT [5:0] to 1. Value after reset is low.
10:4	–	Reserved
3	SCL_I	SCL line sense input value In usual functional mode (ST_EN = 0), this read-only bit always reads 0. In system test mode (ST_EN = 1 & TMODE = 11), this read-only bit returns the logical state taken by the SCL line (either 1 or 0). Value after reset is low.
2	SCL_O	SCL line drive output value In usual functional mode (ST_EN = 0), this bit is don't care. It always reads 0 and a write is ignored. In system test mode (ST_EN = 1 & TMODE = 11), a 0 forces a low level on the SCL line, and a 1 puts the I²C output driver to a high impedance state. 0: Forces 0 on the SCL data line 1: SCL output driver in HiZ state Value after reset is low.

Table 14–22. I²C System Test Register (I2C_SYSTEST) (Continued)

Bit	Name	Description
1	SDA_I	SDA line sense input value In usual functional mode (ST_EN = 0), this read-only bit always reads 0. In system test mode (ST_EN = 1 & TMODE = 11), this read-only bit returns the logical state taken by the SDA line (either 1 or 0). Value after reset is low.
0	SDA_O	SDA line drive output value In usual functional mode (ST_EN = 0), this bit is don't care. It reads as 0 and a write is ignored. In system test mode (ST_EN = 1 & TMODE = 11), a 0 forces a low level on the SDA line and a 1 puts the I²C output driver to a high impedance state. 0: Forces 0 on the SDA data line 1: SDA output driver in Hi-Z state Value after reset is low.

System Test Register

Never set this register for normal I²C operation.

This register facilitates system-level tests by overriding some of the standard functional features of the peripheral. It can permit the test of SCL counters, control the signals that connect to I/O pins, or create digital loopback for self-test when the module is configured in system test (SYSTEST) mode. It also provides stop/no-stop functionality in debug mode.

Table 14–23. System Test Mode Settings

TMODE	Mode
00	Functional mode (default)
01	Reserved
10	Test of SCL counters (SCLL, SCLH, PSC)
11	Loopback mode select + SDA/SCL IO mode select

14.4 Programming Guidelines

14.4.1 Main Program

14.4.1.1 Module Configuration Before Enabling the Module

Step 1: Program the prescaler to obtain an approximately 12-MHz I^2C module clock (I2C_PSC = x; this value is to be calculated and is dependent on the system clock frequency).

Step 2: Program the I^2C clock to obtain 100K bps or 400K bps (I2C_SCLL = x and I2C_SCLH = x; these values are to be calculated and are dependent on the system clock frequency).

Step 3: Configure its own address (I2C_OA = x).

Step 4: Take the I^2C module out of reset (I2C_CON:I2C_EN = 1).

14.4.1.2 Initialization Procedure

Step 1: Configure the I^2C mode register (I2C_CON) bits.

Step 2: If using interrupt for transmit/receive data, enable interrupt masks (I2C_IE).

Step 3: If using DMA for transmit/receive data, enable the DMA (I2C_BUF) and program the DMA controller.

14.4.1.3 Configure Slave Address and Data Counter Registers

In master mode, configure the slave address (I2C_SA = x) and the number of bytes associated with the transfer (I2C_CNT = x).

14.4.1.4 Initiate a Transfer

Poll the bus busy (BB) bit in the I^2C status register (I2C_STAT). If it is cleared to 0 (bus not busy), configure START/STOP (I2C_CON:STT / I2C_CON:STP) condition to initiate a transfer.

14.4.1.5 Poll Receive Data

Poll the receive-data ready interrupt flag bit (RRDY) in the I^2C status register (I2C_STAT). Use the RRDY interrupt or the DMA to read the receive data in the data receive register (I2C_DATA).

14.4.1.6 Poll Transmit Data

Poll the transmit data ready interrupt flag bit (XRDY) in the I^2C status register (I2C_STAT). Use the XRDY interrupt or the DMA to write data into the data transmit register (I2C_DATA).

14.4.2 Interrupt Subroutines

Step 1: Test for arbitration lost and resolve accordingly.

Step 2: Test for no-acknowledge and resolve accordingly.

Step 3: Test for register access ready and resolve accordingly.

Step 4: Test for receive data and resolve accordingly.

Step 5: Test for transmit data and resolve accordingly.

14.4.3 Flow Diagrams

Figure 14–9. Setup Procedure

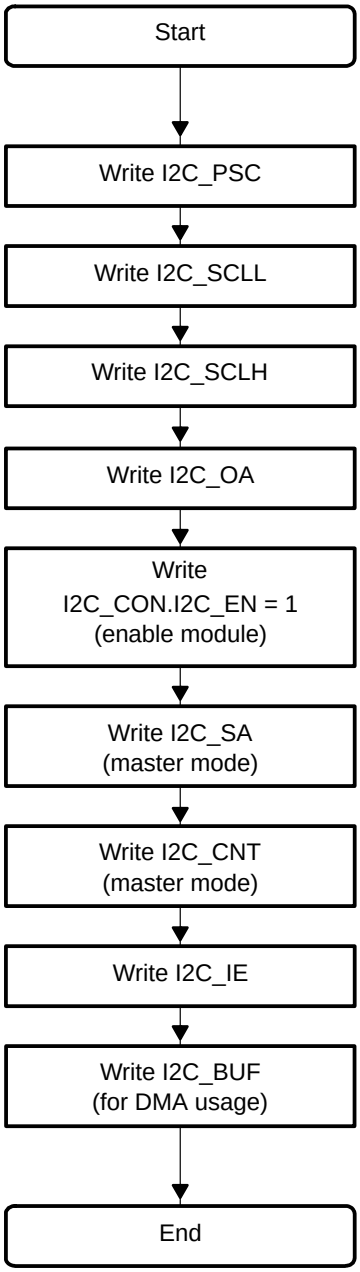


Figure 14–10. Master Transmitter Mode, Polling

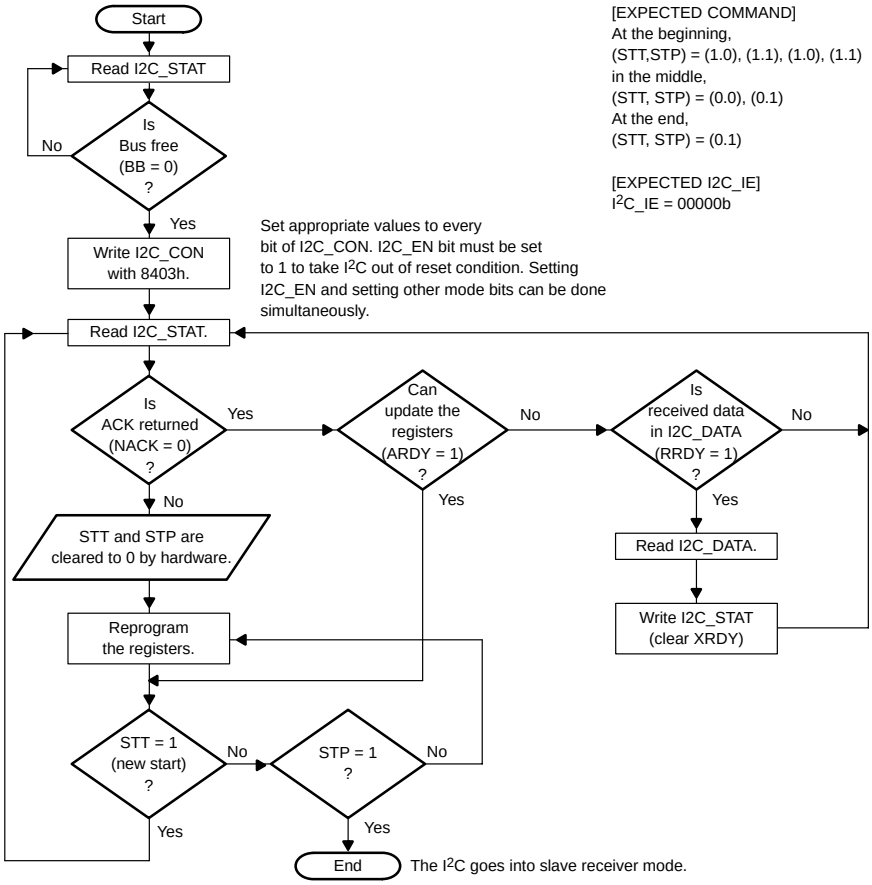


Figure 14–11. Master Receiver Mode, Polling

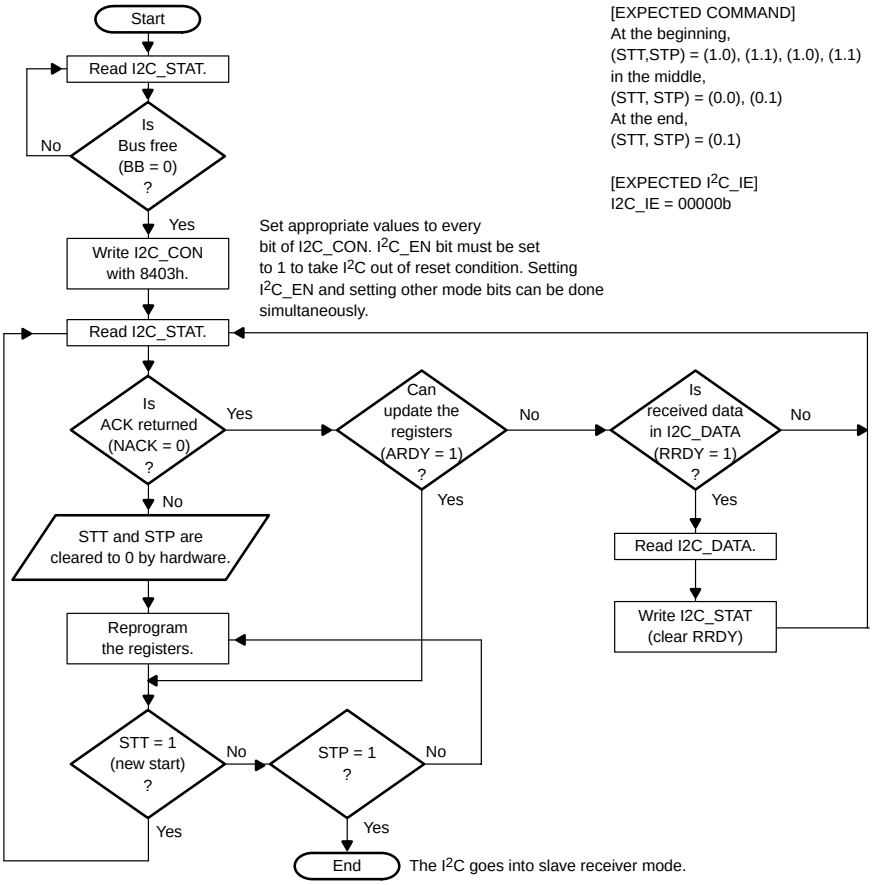


Figure 14–12. Master Transmitter Mode, Interrupt

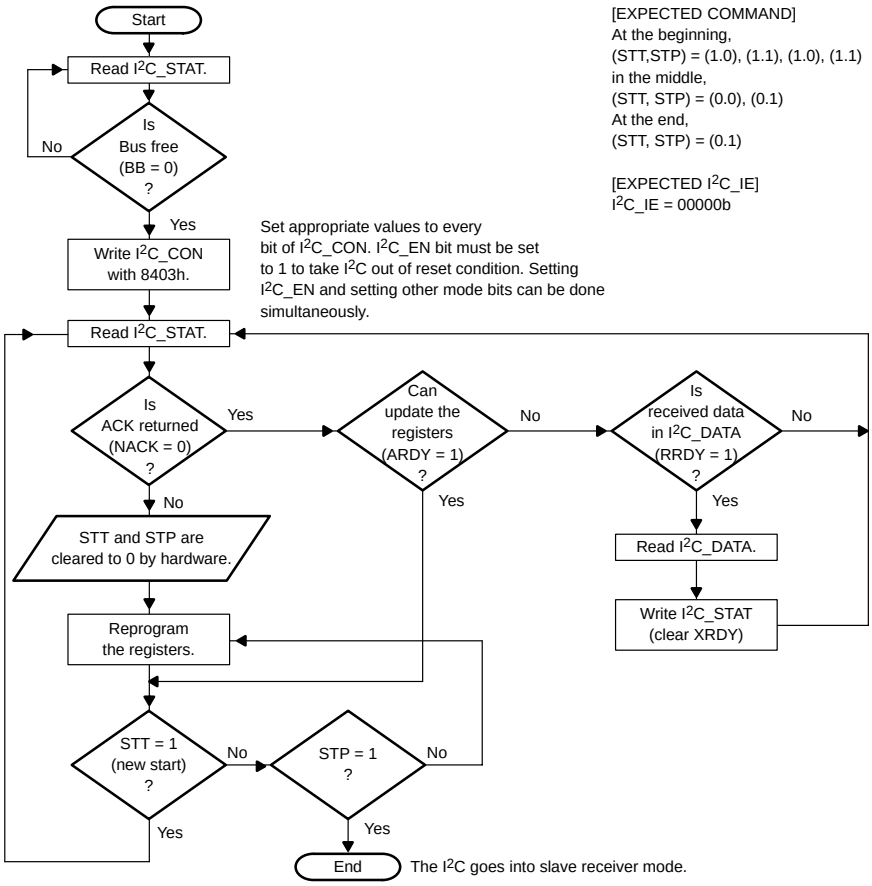


Figure 14–13. Master Receiver Mode, Interrupt

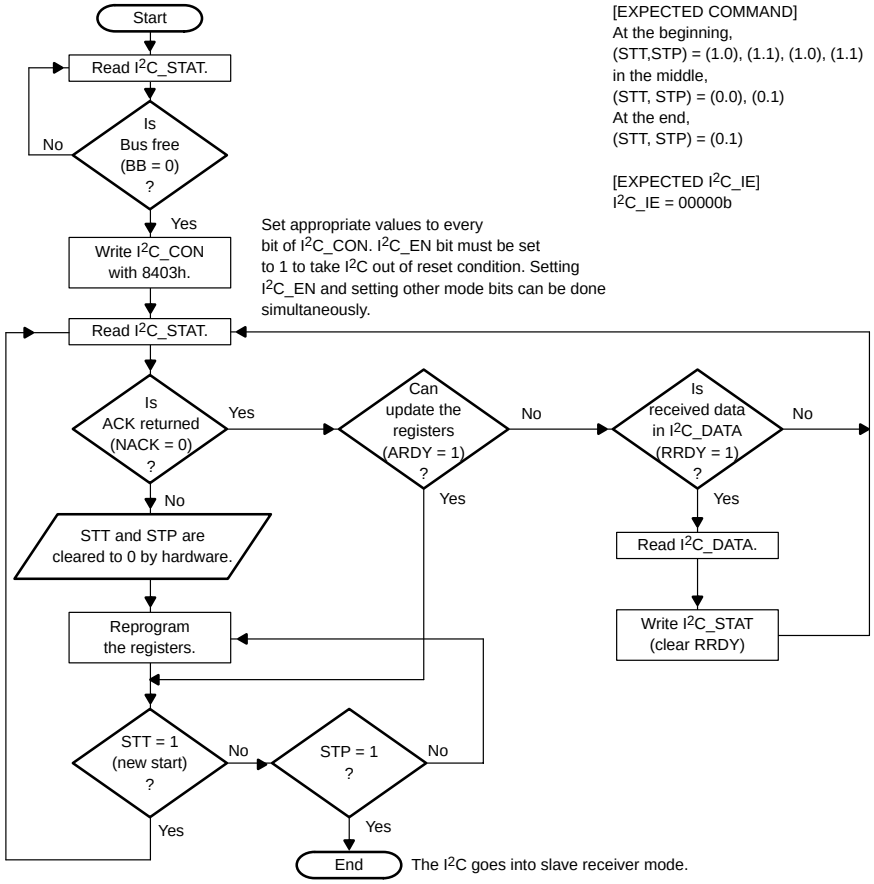


Figure 14–14. Master Transmitter Mode, DMA

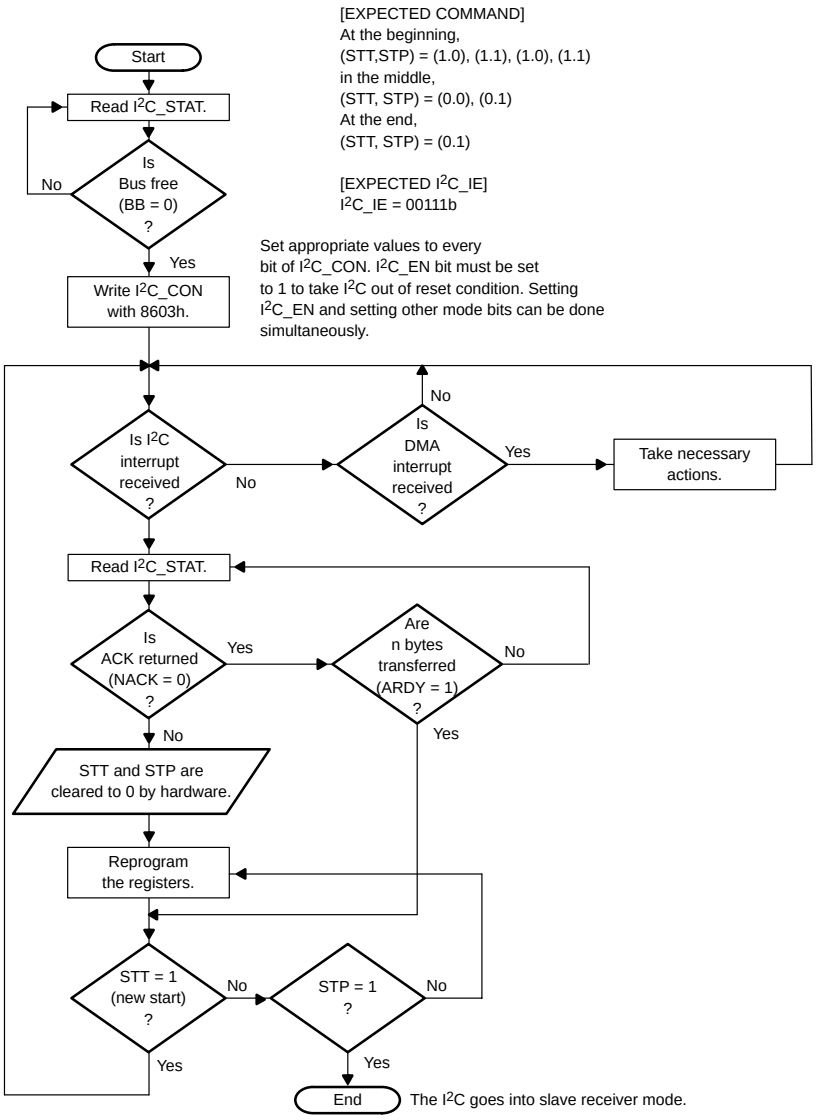


Figure 14–15. Master Receiver Mode, DMA

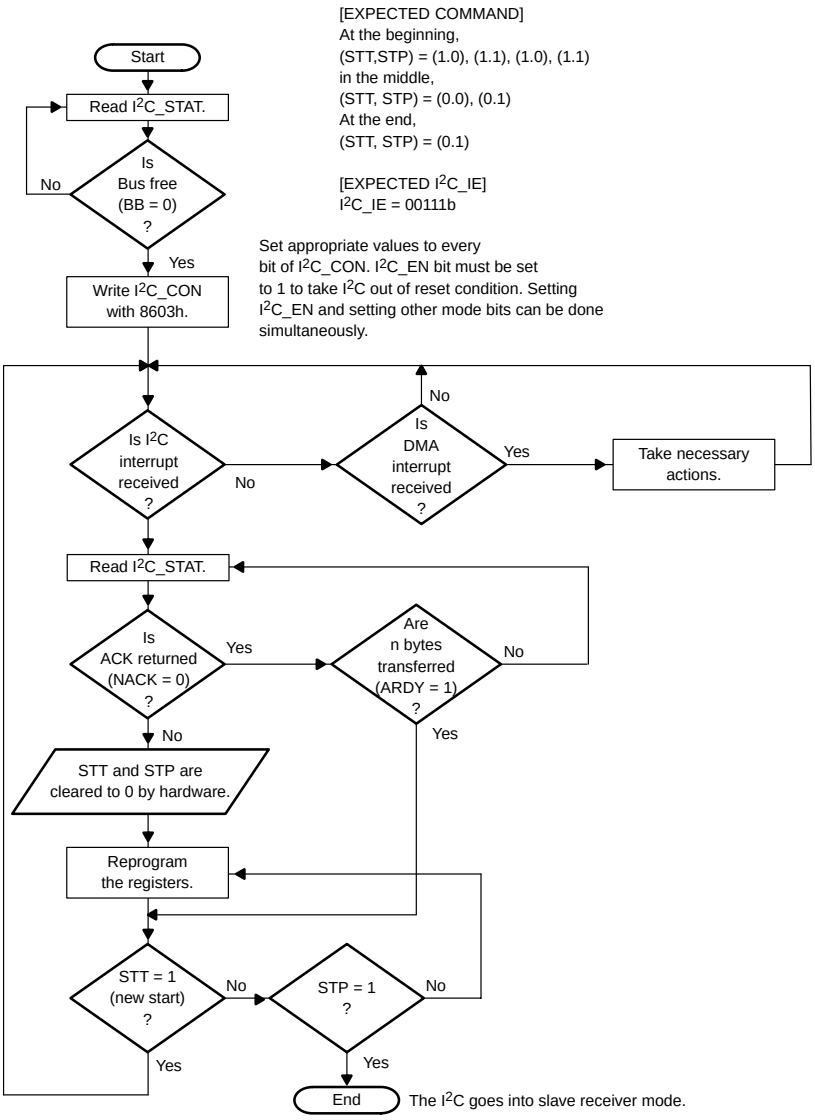


Figure 14–16. Slave Transmitter/Receiver Mode, Polling

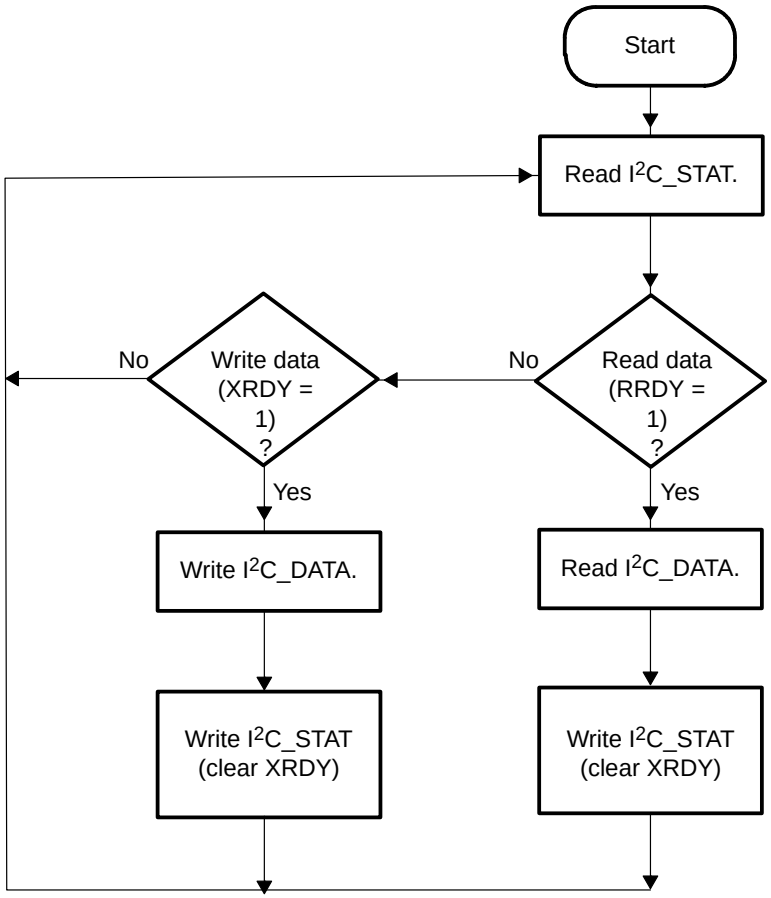
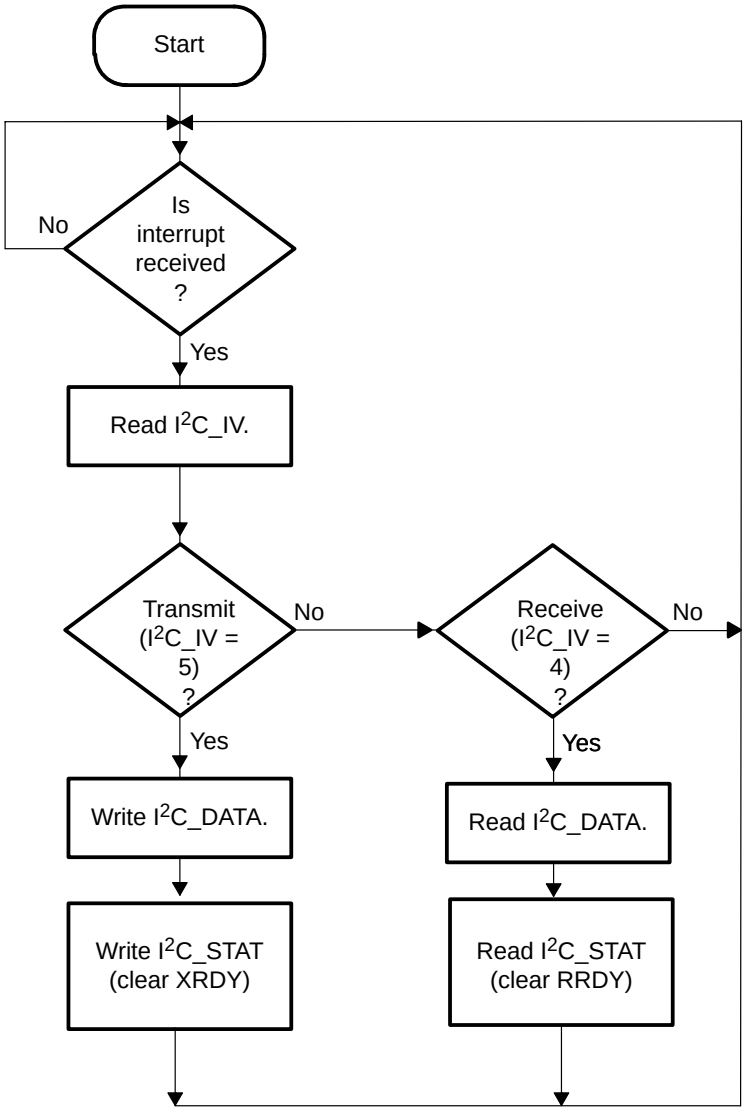


Figure 14–17. Slave Transmitter/Receiver Mode, Interrupt



Chapter 15

Enhanced Audio Controller (EAC)

This chapter describes the enhanced audio controller (EAC) and its registers. The first section describes the EAC and explains how to configure it. The second section describes the EAC applications.

EAC-2 (EAC version 2) is an upgrade to EAC. There is no architectural change in EAC-2. See Section 15.4 for a comparison of EAC and EAC-2.

Topic **Page**

Topic **Page**

15.1 Introduction

The enhanced audio controller (EAC) provides an application with stand-alone audio control without any CPU (DSP or MPU) support. It allows a modem and/or Bluetooth subsystem to be connected to an AC97, a PCM, or an I2S codec while the MPU subsystem (PDA, WinCE MPU, etc.) is in power-down mode. It eliminates the need for two codecs (one for the modem and one for the MPU subsystems).

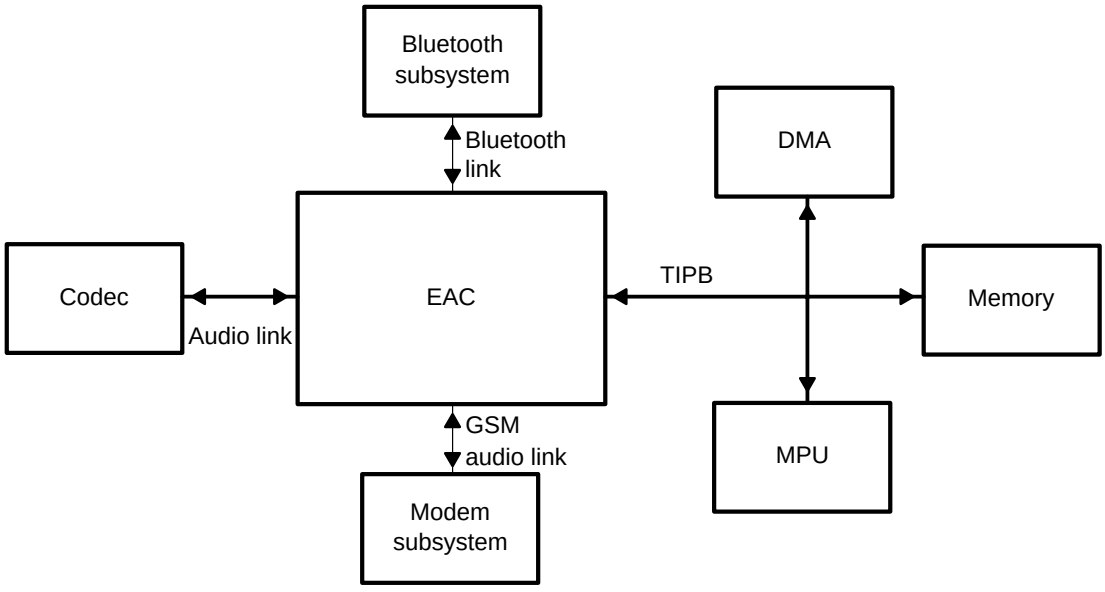
The EAC provides the ability to record/play PCM (wave) files with various sample frequencies and bit-length formats without any CPU processing. It also allows using the same microphone input line for the modem 8-/16-kHz sampling frequency or for high-quality voice recording.

High-quality audio files can be mixed with the voice input channel, down-sampled, and sent to the modem/Bluetooth uplink path (wave-file play during phone call). The EAC provides the CPU the ability to get the data samples coming from the microphone path at the codec sampling frequency, process the signal, and send it back to the modem uplink path. The audio signal coming from the modem/Bluetooth downlink path can be processed by the CPU at 8 or 16 kHz or at the codec sampling frequency before being sent to the codec and the loudspeakers.

Modem/Bluetooth input and/or output voice data samples can be automatically stored in the CPU memory through the DMA write channel (record the phone call).

The EAC provides the ability to loop the Bluetooth audio data samples back to the modem uplink path.

Figure 15–1. General Overview



15.2 General Description

The EAC is an audio-codec peripheral interface designed specifically for applications that require audio data streaming. Applications include digital audio processing, which requires the streaming of digital audio and/or voice data between:

- ❏ The audio codec and the modem uplink (and/or Bluetooth) via SPI connection
- ❏ Memories and audio codec
- ❏ Memories and AuSPI (audio SPI) ports
- ❏ Modem uplink and Bluetooth stream

Table 15–1 shows the possible combinations.

Table 15–1. Loopback Matrix

		EAC Inputs			
		Modem Port	**BT Port**	**Codec Port**	**DMA Read Op**
EAC Outputs	Modem port	–	√ (fixed)	√ (weighted)	√ (weighted)
	BT port	√ (fixed)	–	√ (weighted)	√ (weighted)
	Codec port	√ (weighted)	√ (weighted)	√ (weighted)†	√ (weighted)
	DMA write op	√ (weighted)	√ (weighted)	√ (weighted)	–

† With sidetone

Note: For each line (output), the output can be the sum of all possible inputs.

The codec port interface (C-port) can be configured to support several industry-standard serial interface protocols. These protocols include the AC97, Rev. 1.x without sample-rate converter, inter-IC sound (I2S) modes, and pulse code modulation (PCM) modes.

Nevertheless, sampling rate support on the codec port interface is limited to:

- ❏ 48 kHz in AC97 mode
- ❏ 44.1- or 48 kHz in I2S mode

The EAC peripheral includes two audio serial-port interfaces (AuSPI) to handle 8-kHz or 16-kHz audio data transfers between EAC and modem/Bluetooth. They can be configured to work as a master or a slave SPI.

AuSPI serial interfaces operating at 16-kHz sampling rate are defined as wide-band voice support. The wide-band mode selection applies to both AuSPIs simultaneously (that is, modem and Bluetooth ports always operate at the same sampling frequency).

Furthermore, the EAC includes two independent DMA channels for read or write operations into the system memory. Each channel has a FIFO for data-flow control.

Between EAC endpoints (AuSPI, codec, and DMA operations), the EAC performs sampling rate conversion (SRC) to obtain the selected sample frequency.

15.3 Features

The enhanced audio controller (EAC) features:

- Codec multiple format data interface
  - AC97-compatible (revision 1.x)
  - I2S compatible
  - 16-bit PCM compatible
    - LSB-justified format or standard format
- Sidetone for the codec port interface
    - Attenuation 0 down to 127 dB
    - Inserted delay less than four audio-sampling clock periods (< 100 μs)
- Audio SPI format data interface
  - SPI link for modem audio support
    - 1 to 32 bits using 8- or 16-kHz frame synchronization, main channel master/slave mode for modem device
    - 1 to 32 bits, standard SPI master mode for auxiliary channel
  - SPI link for Bluetooth audio support
    - 1 to 32 bits using 8- or 16-kHz frame synchronization, main channel master/slave mode for Bluetooth subsystem
    - 1 to 32 bits standard SPI, master mode for auxiliary channel
  - μ-law and A-law companding on main channel
- Sound file formats supported
  - PCM raw data format (wave PCM file) support
    - Monophonic/stereo
    - 8-bit/16-bit
    - PCM raw data little-endian or big-endian support
    - Twos complement management for 8-bit PCM file
- Digital processing
  - Generation of audio sampling clock (that is, 8-, 11.025-, 16-, 22.05-, 44.1-, and 48-kHz) from the module clock
  - External audio-sampling clock source allowed on serial interface
  - Independent DMA channels for audio input and output data to read and write from/to the CPU memory
  - Double-buffered data registers allow a continuous data stream for DMA operations
  - Sound-file support at DMA interface for play/record at 8-, 11.025-, 22.05-, 44.1-, and 48-kHz sampling frequency

- High-quality digital hardware up-sampling rate conversion from/to 8.0, 11.025, 22.050, 44.1, and 48 kHz
- Modem/Bluetooth audio signal mixer with input data
- Multipoint volume control
- Peak-value detection on codec port input and on all port outputs for volume monitoring

15.4 EAC and EAC-2 Comparison

This section identifies EAC and EAC-2 differences. EAC-2 is an upgrade to EAC. There are no changes in the general architecture, but new capabilities have been added to existing features.

15.4.1 EAC-2 New Features Description

EAC-2 supports a set of enhancements, which are summarized in Table 15–2.

Table 15–2. EAC-2 Enhancements

Description	EAC	EAC-2
Addition of DMA play/record file @ 48 kHz	Play/record is done at 8-,11.025-, 22.05-, or 44.1-kHz sample rate.	Play/record is done at 8-,11.025-, 22.05-44.1-, or 48-kHz sample rate
Addition of I2S/PCM 48 kHz on EAC codec port	Frame rate on codec port must be 44.1 kHz in I2S/PCM mode.	Frame rate on codec port must be 44.1 kHz, or 48 kHz in I2S/PCM
Addition of 16-kHz wide-band voice on AuSPIs	Frame rate on AuSPI ports must be 8 kHz.	Frame rate on AuSPI ports must be 8 kHz or 16 kHz (wideband)

15.4.2 EAC-2 Programming Model

Any software driver developed on EAC can be reused on EAC-2. The programming model from EAC is preserved (register mapping and register definitions), and control of EAC-2 new features is resumed in a single new register: AGCFR2. Table 15–3 presents a detailed description.

Table 15–3. EAC–2 AGCFR2 Register

Bit	Name	Function	R/W	Reset Value
15:9	RESERVED	Reserved	R	0x0
5	BT_MD_WIDEBAND	Bluetooth and modem ports wideband mode 0: Wideband mode disabled (8-kHz frame rate) 1: Wideband mode enabled (16-kHz frame rate)	R/W	0x0
4	MCLK_I2S_N11M_12M	MCLK frequency indicator for audio operations 0: Clock on MCLK is 11.289 MHz (I2S, 44.1 kHz) 1: Clock on MCLK is 12.288 MHz (I2S, 48 kHz)	R/W	0x0
3	I2S_N44K_48K	Frame sample frequency of I2S codec port 0: Codec port frame rate = 44.1 kHz 1: Codec port frame rate = 48 kHz	R/W	0x0
2:0	FSINT2	DMA intermediate sample frequency 111: AGCFR/FSINT are used (EAC rev1) 000: FSINT is 8 kHz 001: FSINT is 11.025 kHz 010: FSINT is 22.05 kHz 011: FSINt is 44.1 kHz 100: FSINT is 48 kHz (EAC rev 2 new) Others: Reserved	R/W	0x7

15.5 Architecture

15.5.1 Functional Block Diagram

Figure 15–2. EAC Functional Diagram

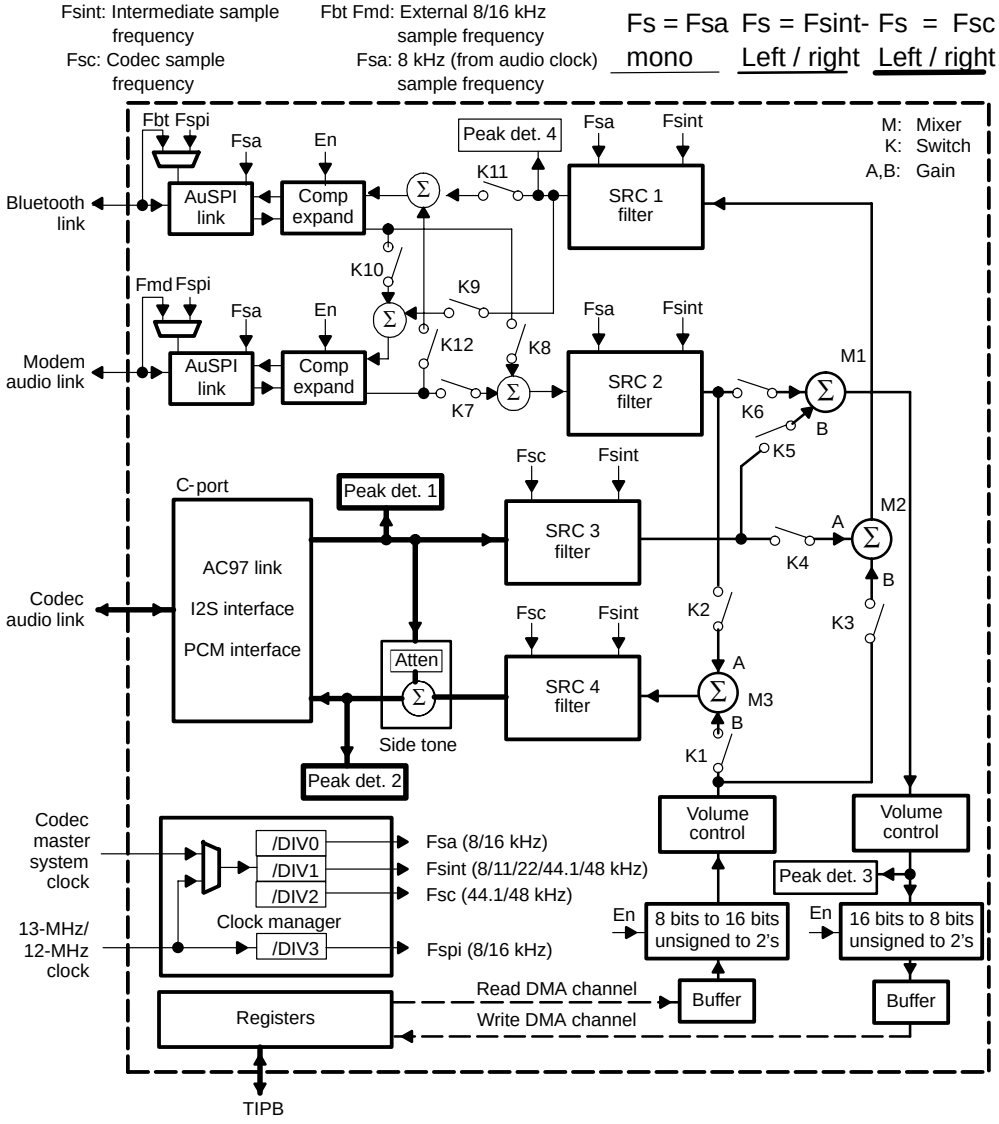


Table 15–4. EAC Signals Definition

Name	Function	I/O	Size
	Clocks Port Signals		
MCLK	Audio codec oscillator or square clock input	I	1
CLK13M	13-MHz square clock dedicated input	I	1
CLK12M	12-MHz square clock dedicated input	I	1
MCLK_OUT	Audio clock output for external codec	O	1
OSCMCLK_EN	Audio codec oscillator shutdown/running output	O	1

Table 15–4. EAC Signals Definition (Continued)

Name	Function	I/O	Size
	Codec Port Signals		
CSYNC	Codec port-interface frame synchro	I/O	1
CSCLK	Codec port-interface serial clock	I/O	1
CDI	Codec port-interface serial data input	I	1
CDO	Codec port-interface serial data output	O	1
CRESET	Codec port-interface reset output	O	1
	Bluetooth Port Signals		
BT SCLK	Bluetooth port-interface serial clock	I/O	1
BT FSYNC	Bluetooth port- interface frame-synchro device 0	I/O	1
BT SDI	Bluetooth port-interface serial data input	I	1
BT SDO	Bluetooth port-interface serial data output	O	1
BT SEN1	Bluetooth port-interface chip-select enable device 1	O	1
	Modem Port Signals		
MD SCLK	Modem port-interface serial clock	I/O	1
MD FSYNC	Modem port-interface frame-synchro device 0	I/O	1
MD SDI	Modem port-interface serial data input	I	1
MD SDO	Modem port-interface serial data output	O	1
MD SEN1	Modem port-interface chip-select enable device 1	O	1

15.5.2 Memory-Mapped Register

The EAC provides a set of control and status registers the MPU can use to control the overall operation of the controller. This section describes the memory-mapped registers.

Address of one register = Start address + Offset address

Start address depends of the memory mapping of the EAC in the chip.

Table 15–5. EAC Mapped Registers

Register	Offset Address	Description
		Codec Port
CPCFR1	0x00	Codec port configuration register 1
CPCFR2	0x02	Codec port configuration register 2
CPCFR3	0x04	Codec port configuration register 3

Table 15–5. EAC Mapped Registers (Continued)

Register	Offset Address	Description
		Codec Port (Continued)
CPCFR4	0x06	Codec port configuration register 4
CPTCTL	0x0A	Codec port interface and status register
CPTTADR	0x0C	Codec port-interface address register
CPTDATL	0x0E	Codec port-interface data register (low byte)
CPTDATH	0x10	Codec port-interface data register (high byte)
CPTVSLL	0x12	Codec port-interface valid time-slots register (low byte)
CPTVSLH	0x14	Codec port-interface valid time-slots register (high byte)
		Modem Port
MPCTR	0x20	Modem port control register
MPMCCFR	0x22	Modem port main channel configuration register
MPACCFR	0x24	Modem port auxiliary-channel control register
MPADLTR	0x26	Modem port auxiliary-data LSB transmit register
MPADMTR	0x28	Modem port auxiliary-data MSB transmit register
MPADLRR	0x2A	Modem port auxiliary-data LSB receive register
MPADMRR	0x2C	Modem port auxiliary-data MSB receive register
		Bluetooth Port
BPCTR	0x30	Bluetooth port control register
BPMCCFR	0x32	Bluetooth port main-channel configuration register
BPACCFR	0x34	Bluetooth port auxiliary-channel control register
BPADLTR	0x36	Bluetooth port auxiliary-data LSB transmit register
BPADMTR	0x38	Bluetooth port auxiliary-data MSB transmit register
BPADLRR	0x3A	Bluetooth port auxiliary-data LSB receive register
BPADMRR	0x3C	Bluetooth port auxiliary-data MSB receive register

Table 15–5. EAC Mapped Registers (Continued)

Register	Offset Address	Description
		Audio Mixer
AMSCFR	0x40	Audio mixer switches configuration register
		Audio Volume Control
AMVCTR	0x42	Audio master volume-control register
AM1VCTR	0x44	Audio mixer 1 volume-control register
AM2VCTR	0x46	Audio mixer 2 volume-control register
AM3VCTR	0x48	Audio mixer 3 volume-control register
		Audio Side Tone
ASTCTR	0x4A	Audio side-tone control register
		Audio Peak Detector
APD1LCR	0x4C	Audio peak-detector 1 left-channel register
APD1RCR	0x4E	Audio peak-detector 1 right-channel register
APD2LCR	0x50	Audio peak-detector 2 left-channel register
APD2RCR	0x52	Audio peak-detector 2 right-channel register
APD3LCR	0x54	Audio peak-detector 3 left-channel register
APD3RCR	0x56	Audio peak-detector 3 right-channel register
APD4R	0x58	Audio peak-detector 4 register
		Audio DMA
ADWDR	0x5A	Audio DMA-write data register
ADRDR	0x5C	Audio DMA-read data register
		Audio Global
AGCFR	0x5E	Audio global-configuration register
AGCTR	0x60	Audio global control register
AGCFR2	0x62	Audio global-configuration register rev2

See descriptions of memory-mapped registers in Section 15.29, *Memory-Mapped Register Descriptions*.

The peripheral internal clocks must be running for any read/write access (except for the audio global registers AGCTR, AGCFR, and AGCFR2):

- ❑ Internal audio clock for codec port registers, audio mixers, audio volume, audio side tone, audio peak detector, and audio DMA registers
- ❑ Bluetooth AuSPI clock for Bluetooth port registers
- ❑ Modem AuSPI clock for modem port registers

The clock source for each peripheral subsystem (modem port, Bluetooth port, and audio/codec port) must be selected or enabled before any register access (see Section 15.6, *Clock Manager*, for clock management details).

Table 15–6. EAC Configuration Summary

Name	Register	Function
		Clock Manager
MCLK	AGCFR	Audio master-clock input frequency
MCLK_OUT	AGCFR	Audio clock output frequency
AUD_CKSRC	AGCFR	Internal audio clock source (codec port and audio part)
		Codec Port
MODE	CPCFR1	Operation mode
MTSL	CPCFR1	Number of time slots per codec frame
TSLOL	CPCFR2	Number of serial clock cycles for time slot 0
BPTSL	CPCFR2	Number of data bits per audio time slot
TSLL	CPCFR2	Number of serial clock cycles for all other than slot 0
DDLY	CPCFR3	Data delay
TRSEN	CPCFR3	3-state data serial output
CLKBP	CPCFR3	Frame sync, serial data output, and serial data input signals
CSYNCP	CPCFR3	Active state of the frame sync output signal
CSYNCL	CPCFR3	Length of the frame sync output signal
CSCLKD	CPCFR3	Direction of the clock
CSYNCD	CPCFR3	Direction of the frame sync
ATSL	CPCFR4	Time slot format
I2S_N44K_48K	AGCFR2	Frame sampling frequency when in I2S mode
		Modem Port
M_CKSRC	AGCFR	Modem port AuSPI clock source
MD_BT_WIDEBAND	AGCFR2	Modem port wideband mode
PRE_MC	MPCTR	Main channel serial bit clock prescaler

Table 15–6. EAC Configuration Summary (Continued)

Name	Register	Function
		Modem Port (Continued)
WS_MC	MPMCCFR	Transmission bit number for main channel
CPB_MC	MPMCCFR	Clock polarity for main channel
FSBL_MC	MPMCCFR	Frame synchronization level for main channel
FSP_MC	MPMCCFR	Frame synchronization polarity for main channel
MCM	MPMCCFR	Main channel mode
EXPAND	MPMCCFR	Expand mode for main channel receive
COMPAND	MPMCCFR	Compand mode for main channel transmit
DJUST_MC	MPMCCFR	Data justify for main channel
DFILL_MC	MPMCCFR	Justified data filling value for main channel
PRE_AC	MPCTR	Auxiliary channel serial bit clock prescaler
WS_AC	MPACCFR	Transmission bit number for auxiliary channel
CPB_AC	MPACCFR	Clock polarity for auxiliary channel
CSBL_AC	MPACCFR	Chip-select level for auxiliary channel
CSP_AC	MPACCFR	Chip-select polarity for auxiliary channel
CST_AC	MPACCFR	Chip-select trigger for auxiliary channel
		Bluetooth Port
BT_CKSRC	AGCFR	Bluetooth port AuSPI clock source
MD_BT_WIDEBAND	AGCFR2	Bluetooth port wide-band mode
PRE_MC	BPCTR	Main channel serial bit clock prescaler
WS_MC	BPMCCFR	Transmission bit number for main channel
CPB_MC	BPMCCFR	Clock polarity for main channel
FSBL_MC	BPMCCFR	Frame synchronization level for main channel
FSP_MC	BPMCCFR	Frame synchronization polarity for main channel
MCM	BPMCCFR	Main channel mode
EXPAND	BPMCCFR	Expand mode for main channel receive
COMPAND	BPMCCFR	Compand mode for main channel transmit
DJUST_MC	BPMCCFR	Data justify for main channel
DFILL_MC	BPMCCFR	Justified data filling value for main channel
PRE_AC	BPCTR	Auxiliary channel serial-bit clock prescaler
WS_AC	BPACCFR	Transmission bit number for auxiliary channel

Table 15–6. EAC Configuration Summary (Continued)

Name	Register	Function
	Bluetooth Port (Continued)	
CPB_AC	BPACCFR	Clock polarity for auxiliary channel
CSBL_AC	BPACCFR	Chip-select level for auxiliary channel
CSP_AC	BPACCFR	Chip-select polarity for auxiliary channel
CST_AC	BPACCFR	Chip-select trigger for auxiliary channel
	DMA Channels	
FSINT	AGCFR	DMA sample frequency (intermediate sample frequency)
FSINT2	AGCFR2	
LI_BI	AGCFR	DMA sample endianism
8_16B	AGCFR	DMA sample size
MN_ST	AGCFR	DMA stereo
GWO	AMVCTR	Write (record) DMA channel volume
GRO	AMVCTR	Read (play) DMA channel volume
	Side Tone	
ATTEN	ASTCTR	Audio side tone
ATT	ASTCTR	Side tone attenuation
	Mixer	
MUTE	AM1VCTR	Audio mixer 1 mute
GINB	AM1VCTR	Audio mixer 1 gain on input B
GINA	AM1VCTR	Audio mixer 1 gain on input A
MUTE	AM2VCTR	Audio mixer 2 mute
GINB	AM2VCTR	Audio mixer 2 gain on input B
GINA	AM2VCTR	Audio mixer 2 gain on input A
MUTE	AM3VCTR	Audio mixer 3 mute
GINB	AM3VCTR	Audio mixer 3 gain on input B
GINA	AM3VCTR	Audio mixer 3 gain on input A

15.6 Clock Manager

15.6.1 Description

Three different clock sources can be provided to the EAC module:

- 13-MHz square clock on CLK13M input
- 12-MHz square clock on CLK12M input
- Codec master clock (square clock or oscillator clock) on MCLK input. This clock frequency is related to the type of codec connected to the codec port, that is, 11.2896 MHz or 22.5792 MHz in I2S/PCM mode and 11.2896 MHz or 24.576 MHz in AC97 mode.

Square-clock enabling/disabling is not controlled by EAC.

For the MCLK clock, optional running/shutdown control of an external oscillator on MCLK is provided through OSCMCLK_EN output. A software sequence is required for proper enabling/disabling of MCLK (see Section 15.5.2, *Memory-Mapped Registers*)**.**

The clock manager outputs the MCLK clock on MCLK_OUT with an optional division by 2. This allows clocking the EAC with a 22.5792-MHz (I2S/PCM codec) or 24.576-MHz (AC97 codec) clock source and still allows using a codec that can support only 11.2896-MHz input clock.

From these input clocks, the clock manager generates the clocks used by the different parts of the peripheral:

- Modem port clock from CLK13M/CLK12M
- Bluetooth port clock from CLK13M/CLK12M
- Codec port and audio-processing shared clock from MCLK/ CLK13M/CLK12M

Note:

CLK13M/CLK12M clocks are dedicated to the modem and Bluetooth AuSPI ports. The usual operation for the codec port and the audio processing section is to use MCLK (clock is shared with the external codec).

CLK12M/CLK13M selection is given for the codec/audio section running when MCLK is not provided (for example, continuous operation between modem, Bluetooth, and memory, even if the codec is not running). The codec port cannot generate the right sampling rates in this case. The external codec must provide frame synchronization and the serial clock and the codec port must be set in slave mode.

In EAC power-down mode, all internal clocks are shut off.

15.6.2 Configuration

15.6.2.1 MCLK Input Frequency Ratio

The MCLK clock frequency must be either:

- 11.2896 MHz or 22.5792 MHz in I2S/PCM 44.1-kHz mode
- 12.288 MHz or 24.576 MHz in AC97 mode or I2S/PCM 48-kHz mode

MCLK= 0x1 (reset value)

Table 15–7. Configuration of MCLK Clock Source Frequency

Register	Name	Bit 0	MCLK Frequency
AGCFR	MCLK	0	MCLK is 11.2896 MHz or 12.288 MHz.
		1	MCLK is 22.5792 MHz or 24.576 MHz.

Note: The internal MCLK is always 11.2896/12.288 MHz according to this bit setting.

15.6.2.2 MCLK_OUT Divider from MCLK

Table 15–8. Configuration of MCLK_OUT Frequency

Register	Name	Bit 1	MCLK_OUT Frequency
AGCFR	MCLK_OUT	0	MCLK_OUT = MCLK
		1	MCLK_OUT = MCLK/2

The clock on MCLK_OUT output is either the MCLK input clock or the MCLK clock divided by 2.

MCLK= 0x0 (reset value)

15.6.2.3 MCLK I2S Frequency

Table 15–9. Configuration of I2S Mode MCLK Reference Frequency

Register	Name	Bit 0	MCLK Frequency
AGCFR2	MCLK_I2S_N11M_12M	0	MCLK is 11.2896 MHz or 22.5792 MHz.
		1	MCLK is 12.288 MHz or 24.576 MHz.

Note: This selection is taken into account only if the codec port is in I2S mode.

The I2S frame rate support is either 44.1 kHz or 48 kHz. In this I2S mode, MCLK can be either 11.2896/22.5792 MHz (targeted for 44.1 kHz) or 12.288/24.576 MHz (for 48 kHz).

MCLK_I2S_N11M_12M = 0x0 (reset value)

15.6.2.4 Audio/Codec Port Clock Source

Table 15–10. Codec/Audio Clock Source Configuration

Register	Name	Bit 5	Bit 4	Clock Source
AGCFR	AUD_CKSRC	0	0	Codec master clock on MCLK
		0	1	12-MHz clock on CLK12M input
		1	0	13 MHz clock on CLK13M input
		1	1	Reserved

The codec/audio clock source is chosen from among three clock inputs (MCLK, CLK12M, and CLK13M).

AUD_CKSRC = 0x0 (reset value)

15.6.2.5 Modem Port Clock Source

Table 15–11. Configuration of Modem Interface Clock Source

Register	Name	Bit 3	Modem Interface Clock Source
AGCFR	M_CKSRC	0	12 MHz
		1	13 MHz

The modem interface clock source is chosen from either 12-MHz clock input or 13-MHz clock input.

M_CKSRC = 0x0 (reset value)

15.6.2.6 Bluetooth Port Clock Source

Table 15–12. Configuration of Bluetooth Interface Clock Source

Register	Name	Bit 2	Modem Interface Clock Source
AGCFR	BT_CKSRC	0	12 MHz
		1	13 MHz

The Bluetooth interface clock source is chosen from either 12-MHz clock input or 13-MHz clock input.

BT_CKSRC = 0x1 (reset value)

15.6.3 External Audio Oscillator Support for Codec Master Clock

The codec master clock delivered to the EAC can be an externally-controlled square clock or the output of an external oscillator. A software sequence is required in the latter case to enable/disable the internal clock (applies only if the MCLK clock is selected as the codec port and audio processing clock source: AUD_CKSRC = 00 in AGCFR).

At oscillator start time, a delay is required before output clock stabilization. The MCLK clock is internally gated for safe start until this stability is achieved. At

oscillator shutdown, the MCLK is internally gated before the oscillator is turned off.

The EAC provides support for these oscillator start-up/shutdown signals through separate enables for the external oscillator and internal clock.

OSCMCLK_EN = 0x0 (reset value)

Table 15–13. External MCLK Oscillator Control

Register	Name	Bit 2	Modem Interface Clock Source
AGCTR	OSCMCLK_EN	0	OSCMCLK_EN output low: Oscillator disabled
		1	OSCMCLK_EN output high: Oscillator enabled

MCLK_EN = 0x0 (reset value)

Table 15–14. MCLK Gating Control

Register	Name	Bit 2	Modem Interface Clock Source
AGCTR	MCLK_EN	0	MCLK clock is internally gated.
		1	MCLK is ungated.

15.6.3.1 Oscillator Enable Sequence

EAC-disabled state (or out-of-reset state):

- ❏ OSCMCLK_EN low: Oscillator shut off
- ❏ MCLK_EN low: MCLK clock input gated

Step 1: Turn oscillator on: OSCMCLK_EN = 1

Step 2: Add software delay (for oscillator output to stabilize)

Step 3: Enable clock internally: MCLK_EN = 1

15.6.3.2 Oscillator Disable Sequence

Clock-running state:

- ❏ OSCMCLK_EN = 1
- ❏ MCLK_EN = 1

Step 1: Disable clock internally: MCLK_EN = 0

Step 2: A few cycles delay required for clock to be effectively gated

Step 3: Shut down oscillator: OSCMCLK_EN = 0

15.6.3.3 Square Clock Enable/Disable

If the codec master clock on MCLK is not provided by an oscillator, the OSCMCLK_EN bit must be left low (out-of-reset state)

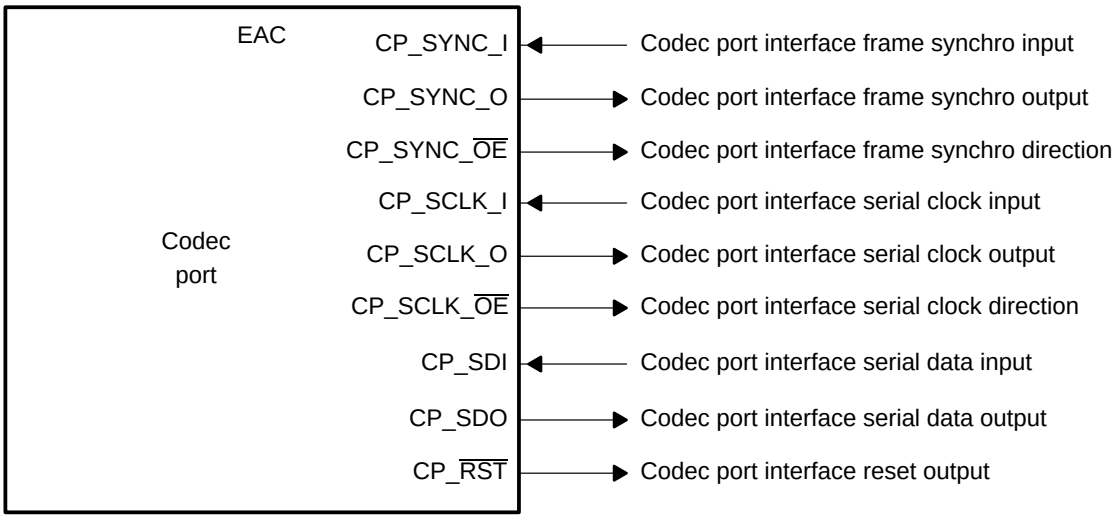
For MCLK internal usage, MCLK_EN must be set high for the clock to be internally released.

15.7 Codec Port Interface

15.7.1 Description

The codec port (C-port) interface is a serial interface used to transfer data between the EAC and a codec device. The serial protocol and formats supported include AC97 1.0 modes, PCM modes, and I2S modes.

Figure 15–3. EAC C-Port Pins Description



The following features of the codec port interface can be configured:

- ❏ Mode of operation
- ❏ Number of time slots per codec frame
- ❏ Number of serial clock cycles for slot 0
- ❏ Number of serial clock cycles for all slots other than slot 0
- ❏ Number of valid data bits per audio data time slot
- ❏ Time slots to be used for command/status address and data
- ❏ Serial clock (CSCLK) frequency in relation to the codec master clock (MCLK) frequency
- ❏ Serial clock signal source; internally generated or an input from the codec device
- ❏ Polarity, duration, and direction of the codec frame-sync signal
- ❏ Relationship between the codec frame-sync signal and the serial clock signal
- ❏ Relationship between the codec frame-sync signal and the serial data signals
- ❏ Relationship between the serial clock signal and the serial data signals

15.7.2 Configuration

The serial interface is a time-division multiplexed (TDM), time-slot-based scheme. The serial format is programmed by setting the number of time slots per codec frame and the number of serial clock cycles (or bits) per time slot. The interface in all modes is bidirectional and full duplex. For some modes, both audio data and command/status data are transferred via the serial interface.

In I2S mode, the serial data output and input signals are delayed one serial clock (SCLK) cycle in reference to the leading edge of the sync signal.

Codec port configuration:

- Operation mode: AC97, I2S, or PCM
- Number of time slots per codec frame: 1 to 32
- Number of serial clock cycles for time slot 0: 8, 16, 32, or others
- Number of data bits per audio time slot: 8, 16, 18, 20, 24, or 32
- Number of serial clock cycles for all slots other than slot 0: 8, 16, 18, 20, 24, or 32
- Data delay in reference to the leading edge of the SYNC signal: No delay or one cycle delay
- Data serial output state: High impedance or enabled
- The CSYNC and CDATO signals are generated with the negative/positive edge of the CSCLK signal, and the CDATI signal is sampled with the positive/negative edge of the CSCLK signal.
- Active state of the CSYNC signal: Low or high
- The CSYNC length may or may not be equal to the time slot 0 length.
- The codec port clock CSCLK is bidirectional.
- The codec port interface frame-sync CSYNC is bidirectional.
- Time slot format: Time slot 1 for address and time slot 2 for data or other time slot format

Note:

EAC does not support sample frequencies (or frame frequencies) other than 44.1 kHz/48 kHz in I2S/PCM mode and 48 kHz in AC97 mode.

CPCFR1, CPCFR2, CPCFR3, and CPCFR4 are used for operation mode configuration. Table 15–15 through Table 15–28 describe the configurations.

These registers can be updated only if the interface is not running (that is, bit CPEN of the codec port control register CPCTL is set to 0).

Table 15–15. Operation Mode Configuration for Codec Port

Register	Name	Bit 2	Bit 1	Bit 0	Mode
CPCFR1	MODE	0	0	0	General-purpose mode
		0	0	1	PCM mode
		0	1	0	AC97 mode
		0	1	1	Reserved
		1	0	0	I2S mode
		1	0	1	Reserved
		1	1	0	Reserved
		1	1	1	Reserved

Bits 2:0 of the codec port configuration register 1 (CPCFR1) are used to select the codec port interface mode: AC97 mode, I2S mode, or PCM mode.

Mode = 0x4 (reset value)

Table 15–16. Configuration of Number of Time Slots per Codec Frame

Register	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Number of Time Slots per Audio Frame
CPCFR1	MTSL	0	0	0	0	0	1
		0	0	0	0	1	2
		–	–	–	–	–	–
		1	1	1	1	0	31
		1	1	1	1	1	32

Bits 7:3 of the codec port configuration register 1 (CPCFR1) are used to program the number of time slots per audio frame: 1 to 32 time slots per audio frame.

MTSL = 0x1 (reset value)

15.7.2.1 Number of Serial Clock Cycles for Time Slot 0

Table 15–17. Configuration of Number of Serial Clock Cycles for Slot 0

Register	Name	Bit 7	Bit 6	Cycles for Time Slot 0
CPCFR2	TSLOL	0	0	Same as TSLL
		0	1	8 BIT_CLK
		1	0	16 BIT_CLK
		1	1	32 BIT_CLK

Bits 7:6 of the codec port configuration register 2 (CPCFR2) are used to program the number of serial clock (BIT_CLK) cycles for time slot 0: 8-, 16-, or

32-serial clock cycles for time slot 0, or the same number as the other time slots (set by TSLL; see Table 15–19).

TSLOL = 0X0 (reset value)

Table 15–18. Configuration of Number of Data Bits per Audio Time Slot

Register	Name	Bit 5	Bit 4	Bit 3	Data Bits per Time Slot
CPCFR2	BPTSL	0	0	0	8
		0	0	1	16
		0	1	0	18
		0	1	1	20
		1	0	0	24
		1	0	1	32
		1	1	0	Reserved
		1	1	1	Reserved

Bits 5:3 of the codec port configuration register 2 (CPCFR2) are used to program the number of data bits per audio time slot: 8-, 16-, 18-, 20-, 24-, or 32-data bits per audio time slot.

BPTSL = 0x1 (reset value)

Table 15–19. Configuration of Number of Serial Clock Cycles for Slots Other Than Slot 0

Register	Name	Bit 2	Bit 1	Bit 0	BIT_CLK Cycles for all Time Slots Except Time Slot 0
CPCFR2	TSLL	0	0	0	8
		0	0	1	16
		0	1	0	18
		0	1	1	20
		1	0	0	24
		1	0	1	32
		1	1	0	Reserved
		1	1	1	Reserved

Bits 2:0 of the codec port configuration register 2 (CPCFR2) are used to program the number of serial clock (BIT_CLK) cycles for all time slots except time slot 0: 8-, 16-, 18-, 20-, 24- or 32-serial clock cycles for all time slots except time slot 0.

TSLL = 0X1 (reset value)

Table 15–20. Configuration of Cycle Delay

Register	Name	Bit 7	Data Delay
CPCFR3	DDLY	0	No delay
		1	One cycle delay

Bit 7 of the codec port configuration register 3 (CPCFR3) is used to program a one BIT_CLK cycle delay of the serial data output and input signals in reference to the leading edge of the SYNC signal: One or no cycle delay.

DDLY = 0x1 (reset value)

Table 15–21. Configuration of Data Serial Output State

Register	Name	Bit 6	Data Serial Output
CPCFR3	TRSEN	0	3-state
		1	Enable

Bit 6 of the codec port configuration register 3 (CPCFR3) is used to program the hardware to 3-state data serial output while the audio frame is not valid.

TRSEN = 0x0 (reset value)

Table 15–22. Generation of CP_SYNC, CP_SDO, and CP_SDI Signals

Register	Name	Bit 5	Generation of CP_SYNC Signal at	Generation of CP_SDO Signal at	Sample of CP_SDI Signal at
CPCFR3	CLKBP	0	Positive edge of CP_SCLK signal	Positive edge of CP_SCLK signal	Negative edge of CP_SCLK signal
		1	Negative edge of CP_SCLK signal	Negative edge of CP_SCLK signal	Positive edge of CP_SCLK signal

Bit 5 of the codec port configuration register 3 (CPCFR3) is used to program the codec port interface frame sync (CP_SYNC) output signal, the codec port interface serial data output (CP_SDO) signal, and the codec port interface serial data input (CP_SDI) signal relative to the codec port clock CP_SCLK.

CLKBP = 0X1 (reset value)

Table 15–23. Configuration of CP_SYNC Signal, Active State

Register	Name	Bit 4	Active State of CP_SYNC
CPCFR3	CSYNCP	0	Low
		1	High

Bit 4 of the codec port configuration register 3 (CPCFR3) is used to program the active state of the codec port interface frame sync (CP_SYNC) output signal: Low or high.

CSYNCP = 0X1 (reset value)

Table 15–24. Configuration of CP_SYNC Signal Length

Register	Name	Bit 3	Length of CP_SYNC Active State
CPCFR3	CSYNCL	0	≠ Length of time slot 0
		1	= Length of time slot 0

Bit 3 of the codec port configuration register 3 (CPCFR3) is used to program the length of the codec port interface frame sync (CP_SYNC) output signal to be the same number of CP_SCLK cycles as time slot 0.

CSYNCL = 0X1 (reset value)

Table 15–25. Configuration of SP_SCLK Signal Direction

Register	Name	Bit 1	CP_SCLK
CPCFR3	CSCLKD	0	Output
		1	Input

Bit 1 of the codec port configuration register 3 (CPCFR3) is used to program the direction of the codec port clock CP_SCLK: Output or input.

CSCLKD = 0x1 (reset value)

Table 15–26. Configuration of CP_SYNC Signal Direction

Register	Name	Bit 0	CP_SYNC
CPCFR3	CSYNCD	0	Output
		1	Input

Bit 0 of the codec port configuration register 3 (CPCFR3) is used to program the direction of the codec port interface frame sync CP_SYNC: Output or input.

CSYNCD = 0x1 (reset value)

Table 15–27. Configuration of Time Slot Format

Register	Name	Bit 7	Bit 6	Bit 5	Bit 4	Time Slot Format
CPCFR4	ATSL	0	0	0	0	For I2S or PCM mode
		0	0	0	1	For AC97 mode (time slot 1 for address, time slot 2 for data)

Bits 7:4 of the codec port configuration register 4 (CPCFR4) are used to program the format of the time slot: Time slot 1 for address and time slot 2 for data or others.

In I2S and PCM modes, ATSL bits must be left to 0.

ATSL = 0x00 (reset value)

Table 15–28. Configuration of CSCLK Signal Divider

Register	Name	Bit 2	Bit 1	Bit 0	CSCLK Signal Divider
CPCFR4	DIVB	0	0	0	Disabled (no divider)
		0	0	1	Divide by 2
		0	1	0	Divide by 3
		0	1	1	Divide by 4
		1	0	0	Divide by 5
		1	0	1	Divide by 6
		1	1	0	Divide by 7
		1	1	1	Divide by 8

Bits 2:0 of the codec port configuration register 4 (CPCFR4) are used to program the CSCLK signal divider: Divide by 2 to 8, or no division.

DIVB = 0x00 (reset value)

15.7.3 Codec Port Programming Limits

For interfacing with the audio part of EAC, the codec port frame rate must be limited to the following possibilities:

- ❏ AC97: 48 kHz
- ❏ I2S/PCM: 44.1 kHz
- ❏ I2S/PCM: 48 kHz

Programming of all codec port configuration bits must obey these constraints.

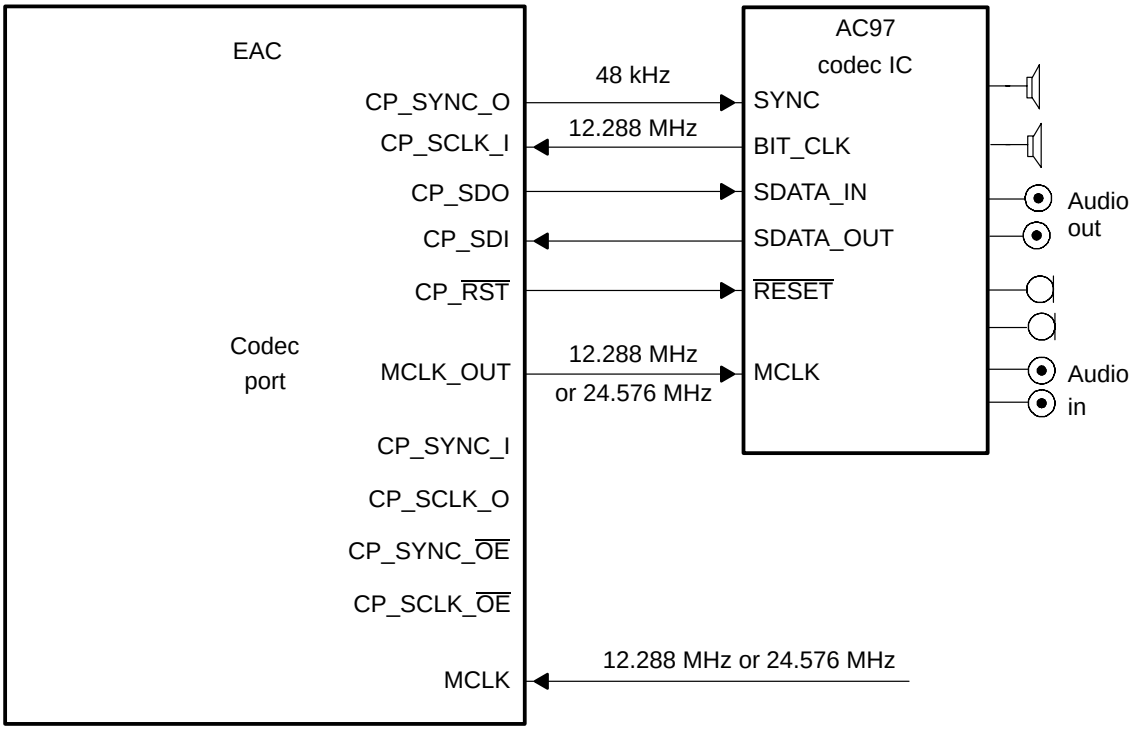
This means that:

- ❏ In master mode: The audio clock source must be selected so that the combination of audio clock frequency, divide by B, number of serial clock cycles per slot, and number of slots parameter results in the right frame frequency.
- ❏ In slave mode: The external codec must generate the frame synchronization at the right frequency. Even in slave mode, the configuration register content must match the actual serial signal waveforms as generated by the external codec.

15.7.4 AC97 (Audio Codec97) Mode

In AC97 mode, the codec port interface can be configured as an AC link serial interface to the AC97 codec device. The AC link serial interface is a time division multiplexed (TDM) slot-based serial interface which is used to transfer both audio data and command/status data to the codec device.

Figure 15–4. Connection of the EAC to an AC97 Codec IC



In this mode, the codec port interface is configured as a bidirectional full-duplex serial interface with a fixed rate of 48 kHz. Each 48-kHz frame is divided into 13 time slots, with the use of each time slot predefined by the Audio Codec ’97 specification:

Table 15–29. AC97 Audio Frame

Time Slot	Out	In
0	Tag	Tag
1	Command address port	Status address port
2	Command data port	Status data port
3	PCM playback left channel	PCM record left channel
4	PCM playback right channel	PCM record right channel
5	Optional modem line 1 DAC	Optional modem line 1 ADC
6	Optional PCM center	Optional dedicated microphone record data
7	Optional PCM left surround	Reserved
8	Optional PCM right surround	Reserved
9	Optional LFE DACs	Reserved
10	Optional modem line 2 DAC	Optional modem line 2 ADC
11	Optional modem headset DAC	Optional modem headset ADC
12	Optional modem GPIO control	Optional modem GPIO status

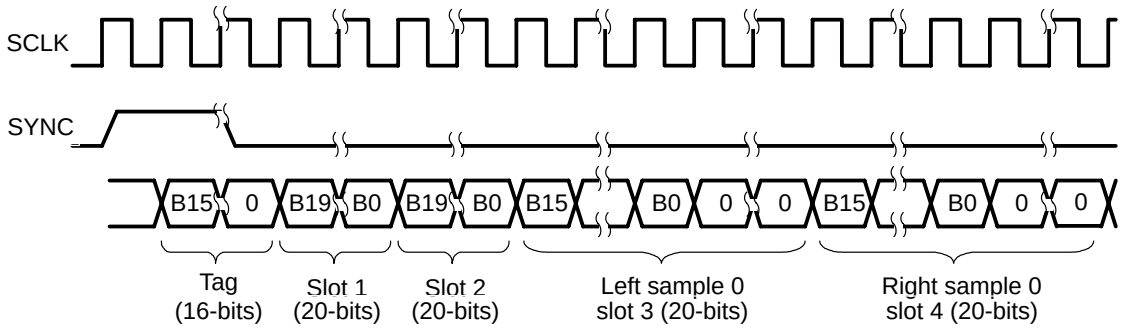
Each time slot is 20 serial clock cycles in length, except for time slot 0, which is only 16 serial clock cycles. The serial clock, which is referred to as the BIT_CLK for AC ’97 modes, is set to 12.288 MHz.

The BIT_CLK is derived from the codec master system clock (MCLK) input:

- If MCLK = 12.288 MHz, then BIT_CLK = MCLK = 12.288 MHz
- If MCLK = 24.576 MHz, then BIT_CLK = MCLK/2 = 12.288 MHz

The SYNC is generated by dividing the serial bit clock (BIT_CLK): 12.288 MHz/256 = 48 kHz.

Figure 15–5. AC97 Serial Interface Format



The beginning of all audio sample packets, or audio frames, transferred over the AC link is synchronized to the rising edge of the SYNC signal. Data transitions on the AC link on every rising edge of BIT_CLK and is subsequently sampled on the receiving side of the AC link on each falling edge immediately following the falling edge of BIT_CLK.

- Tag phase: Portion of audio frame where the SYNC signal is high, shifted one BIT_CLK forward. It is the duration of time slot 0. Time slot 0 has only 16 bits. Each bit of time slot 0 conveys a valid tag for its corresponding time slot within the current audio frame (1 → valid data). Tag bits are set/read through registers CPTVSLL and CPTVSLH (codec port interface valid time slots low and high byte) registers.
- Data phase: Portion of audio frame where the SYNC signal is low, shifted one BIT_CLK forward. Each audio frame has 12 outgoing and 12 incoming data streams with 20-bit sample resolution.

  Outgoing and ingoing data streams are MSB-justified (MSB first). Unused least significant bits are stuffed with 0s.

15.7.4.1 Command/Status Address and Data Slots Managing

The command/status address on slot 1 is sent to external code device/status through the CPTTADR (codec port interface address) register. Operation (read for status and write for command) is also set in this register.

The command/status data words on slot 2 are exchanged through CPTDATL and CPTDATH (codec port interface data register low and high bytes). Data written to these registers is sent to external code upon a command write opera-

tion. Status received from the external codec can be read in these registers after a status read operation.

Transmit and receive register status is provided by the TXE (transmit data register empty) and the RXF (receive data register full) flags in the CPCTL register.

Note:

IRQ generation upon transmit register empty and receive register full in CPCTL is not supported.

15.7.4.2 Audio Samples Exchange

The EAC codec port only supports stereo data: Only slot 3 (left sample) and slot 4 (right sample) contain valid audio samples.

The codec port is directly interfaced to the audio processing part with 16-bit resolution. Only the 16 MSBs of each audio sample are taken into account: Input sample LSBs are discarded, whereas output sample LSBs are filled with zeros.

Note:

In AC97 mode, the frame frequency must be 48 kHz for data exchanges with an audio processing part.

15.7.4.3 Configuration Registers Setting

Table 15–30 through Table 15–33 describe the CPCFR1, CPCFR2, CPCFR3, and CPCFR4 configuration register settings for AC97 mode.

Table 15–30. CPCFR1 Configuration for AC97 Mode (0X62)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Reserved		0
7:3	MTSL	13 time slots per audio frame		0x0C
2:0	MODE	AC97 mode		0x2

Table 15–31. CPCFR2 Configuration for AC97 Mode (0X8B)

Bit	Name	Function	Reset Value
15:8	RESERVED	Reserved	0
7:6	TSOL	16 serial-clock cycles for time slot 0	0x2
5:3	BPTSL	16 data bits per audio time slot. Only the 16 MSBs are used.	0x1
2:0	TSLL	20 serial-clock (BIT_CLK) cycles for all time slots except time slot 0.	0x3

Table 15–32. CPCFR3 Configuration for AC97 Mode (0XDA)

Bit	Name	Function	Reset Value
15:8	RESERVED	Reserved	0
7	DDLY	One BIT_CLK cycle delay of the serial data output and input signals in reference to the leading edge of the SYNC signal.	0x1
6	TRSEN	Data serial output enabled during the audio frame is not valid.	0x1
5	CLKBP	The CSYNC signal is generated with the positive edge of the CSCLK signal. Also, the CDATO signal is generated with the positive edge of the CSCLK signal, and the CDATI signal is sampled with the negative edge of the CSCLK signal.	0x0
4	CSYNCP	The polarity of the codec port interface frame sync (CSYNC) output signal is active high.	0x1
3	CSYNCL	The length of the codec port interface frame sync (CSYNC) output signal is the same number of CSCLK cycles as time slot 0.	0x1
2	RESERVED	Reserved	0
1	CSCLKD	The direction of the codec port interface serial clock (CP_SCLK) signal is an input of the device: CSCLKD = 0x1.	0x1
0	CSYNCD	The direction of the codec port interface frame sync (CP_SYNC) signal is an output from the device.	0

Table 15–33. CPCFR4 Configuration for AC97 Mode (0X10)

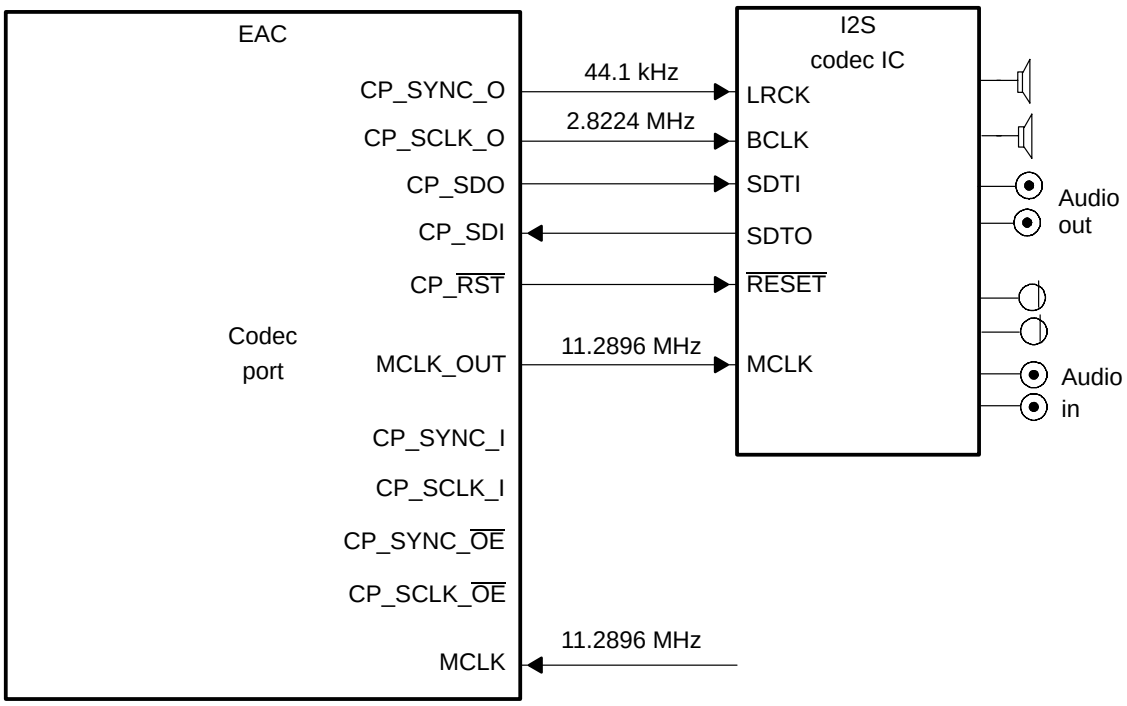
Bit	Name	Function	Reset Value
15:8	RESERVED	Reserved	0
7:4	ATSL	Time slot 1 is used for the address and time slot 2 is used for the data.	0x01
3	CLKS	The source of the clock signal to be used for the divide-by-B circuit is the codec master system clock: CLKS = 0 (deleted).	0
2:0	DIVB	N divider on MCLK to generate CSCLK (MCLK = 12.288 MHz, 256 bits per frame: Frame synchro frequency = 48 kHz)	0x0

15.7.5 Inter-IC Sound (I2S) Mode

In inter-IC sound (I2S) mode, the codec port interface can be configured as an I2S link serial interface to the I2S codec device. The I2S link serial interface

is a time division multiplexed (TDM) slot-based serial interface which is used to transfer both audio data and command/status data to the codec device. See Figure 15–6.

Figure 15–6. Connection of the EAC to an I2S Codec IC



In this mode, the codec port interface is configured as a bidirectional full-duplex serial interface with two time slots per frame: Time slot 0 is used for the left channel audio data and time slot 1 for the right channel audio data. Each time slot is 16 serial clock cycles in length, so the frame is $2 \times 16 = 32$ serial clock cycles in length.

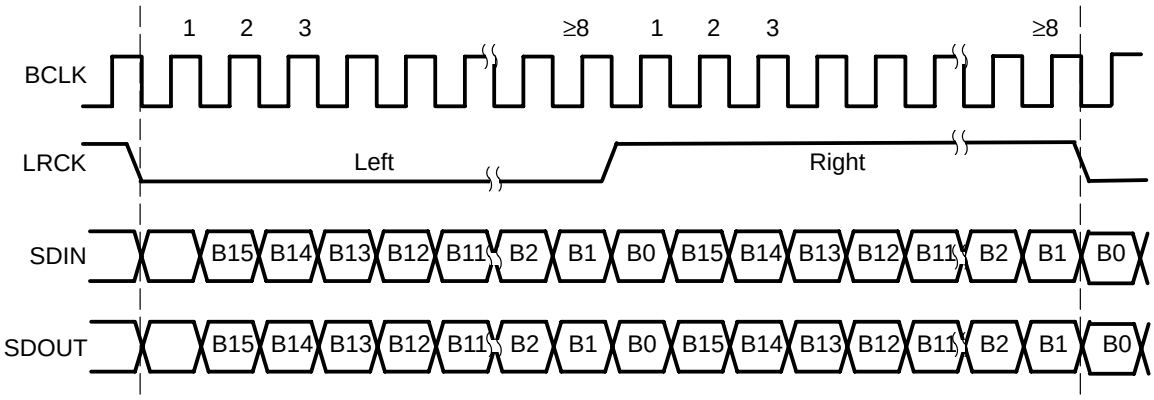
Table 15–34. FS, MCLK, and SCLK Frequency for I2S Mode

Frame Sync (FS)	Master Clock (MCLK) (256xFS)	Serial Clock (SCLK) (32xFS)
44.1 kHz	11.2896 MHz	1.4112 MHz
48 kHz	12.288 MHz	1.536 MHz

The LRCK signal has a 50% duty cycle. The LRCK signal is low for the left channel time slot and high for the right channel time slot. In addition, the LRCK signal is synchronous to the falling edge of the SCLK. Serial data is shifted out on the falling edge of SCLK and shifted in on the rising edge of SCLK. There is a one SCLK cycle delay from the edge of the LRCK before the most significant bit of the data is shifted out for both the left channel and the right channel.

For the I2S mode of the codec port interface, there is a 16-bit transmit and a 16-bit receive shift register for each SDOUT and SDIN signal. The interface automatically pads the unused bits with zeros. Serial data is transmitted in twos-complement with the MSB first.

Figure 15–7. I2S Serial Interface Format



The transmitter always sends the MSBs of the next word one clock period after LRCK changes.

The codec port is directly interfaced to the audio processing part with 16-bit resolution. Valid data bits and the time slot length setting must not be less than 16 bits. In case of higher values, only the 16 MSBs of each left and right sample are taken into account: Input sample LSBs are discarded, whereas output sample LSBs are filled with zeros.

Note:

In I2S mode, the frame frequency must be 44.1 kHz or 48 kHz for data exchanges with an audio processing part.

AC97-only registers are not used: Codec port valid time slot registers (CPTVSLL and CPTVSLH) must be left at their reset value. The codec port address and data register (CPTTADR, CPTDATL, and CPTATH) content is unknown.

Table 15–35 through Table 15–38 describe the CPCFR1, CPCFR2, CPCFR3, and CPCFR4 configuration registers for I2S mode.

Table 15–35. CPCFR1 Configuration for I2S Mode (0X0C)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Reserved		0
7:3	MTSL	2 time slots per audio frame		0x1
2:0	MODE	I2S mode		0x4

Table 15–36. CPCFR2 Configuration for I2S Mode (0X0D)

Bit	Name	Function	Reset Value
15:8	RESERVED	Reserved	0
7:6	TSOL	Serial clock cycles for time slot 0 same as other time slots.	0x0

Table 15–36. CPCFR2 Configuration for I2S Mode (0X0D) (Continued)

Bit	Name	Function	Reset Value
5:3	BPTSL	16 data bits per audio time slot. Only the 16 MSBs are used.	0x1
2:0	TSLL	32 serial clock (BCLK) cycles for all time slots except time slot 0.	0x1

Table 15–37. CPCFR3 Configuration for I2S Mode (0XE8)

Bit	Name	Function	Reset Value
15:8	RESERVED	Reserved	0
7	DDLY	One BIT_CLK cycle delay of the serial data output and input signals in reference to the leading edge of the SYNC signal.	0x1
6	TRSEN	Data serial outputs enabled during the audio frame are not valid.	0x1
5	CLKBP	The CSYNC signal is generated with the negative edge of the CSCLK signal. Also, the CDATO signal is generated with the negative edge of the CSCLK signal, and the CDATI signal is sampled with the positive edge of the CSCLK signal.	0x1
4	CSYNCP	The polarity of the codec port interface frame sync (CSYNC) output signal is active low during time slot 0. Time slot 0 = left sample.	0x0
3	CSYNCL	The length of the codec port interface frame sync (CSYNC) output signal is the same number of CSCLK cycles as time slot 0.	0x1
2	RESERVED	Reserved	0
1	CSCLKD	The direction of the codec port interface serial clock (CP_SCLK) signal is an output from the device.	0x0
0	CSYNCD	The direction of the codec port interface frame sync (CP_SYNC) signal is an output from the device.	0

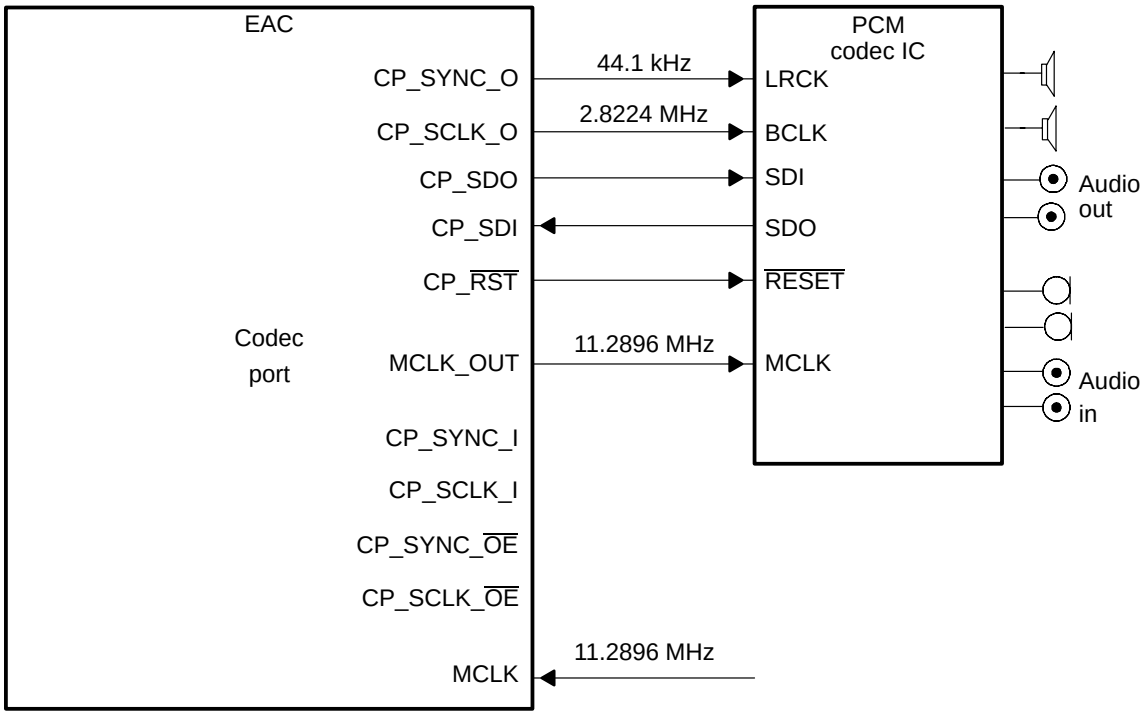
Table 15–38. CPCFR4 Configuration for I2S Mode (0X03)

Bit	Name	Function	Reset Value
15:8	RESERVED	Reserved	0
7:4	ATSL	No address/data time slot.	0x0
3	CLKS	The source of the clock signal to be used for the divide-by-B circuit is the codec master system clock (deleted).	0
2:0	DIVB	Divide by 8 (MCLK = 11.289 MHz, 32 CSCLK cycles per frame: MCLK must be divided by 8 to achieve a 44.1 kHz frame frequency)	0x7

15.7.6 Pulse Code Modulation (PCM) Mode

In pulse code modulation (PCM) mode, the codec port interface can be configured as a PCM link serial interface to the PCM codec device. The PCM link serial interface is a time division multiplexed (TDM) slot-based serial interface which is used to transfer both audio data and command/status data to the codec device.

Figure 15–8. Connection of the EAC to PCM Codec IC



In this mode, the codec port interface is configured as a bidirectional full-duplex serial interface with two time slots per frame: Time slot 0 is used for the left channel audio data and time slot 1 for the right channel audio data. Each time slot is 32 serial clock cycles in length, so the frame is $2 \times 32 = 64$ serial clock cycles in length.

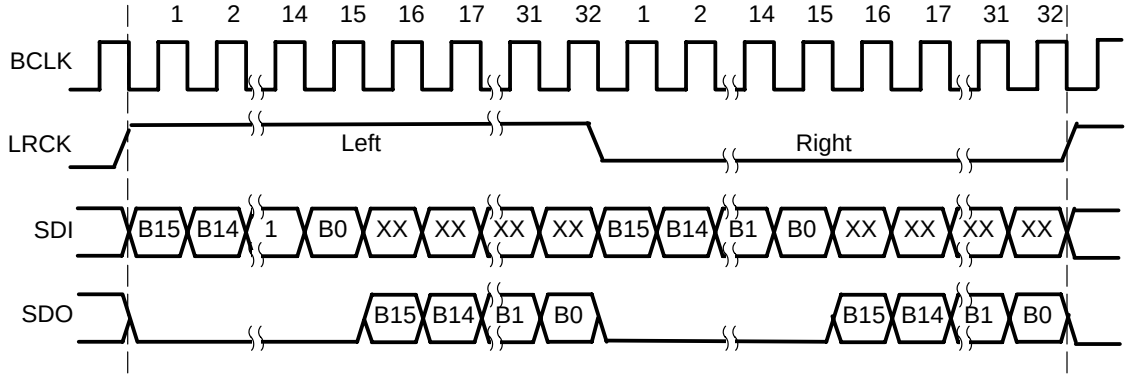
Table 15–39. FS, MCLK, and SCLK Frequency for PCM Mode

Frame Sync (FS)	Master Clock (MCLK) (256 × FS)	Serial Clock (SCLK) (64 × FS)
44.1 kHz	11.2896 MHz	2.8224 MHz
48 kHz	12.288 MHz	3.072 MHz

The LRCK signal has a 50% duty cycle. The LRCK signal is high for the left channel time slot and low for the right channel time slot. In addition, the LRCK signal is synchronous to the falling edge of the SCLK. Serial data is shifted out on the falling edge of SCLK and shifted in on the rising edge of SCLK. There is no SCLK cycle delay from the edge of the LRCK before the most significant bit of the data is shifted out for both the left channel and the right channel.

For the PCM mode of the codec port interface, there is a 16-bit transmit and a 16-bit receive shift registers for each of the SDOUT and SDIN signals.

Figure 15–9. PCM Serial Interface Format



In the I2S mode, the codec port is directly interfaced to the audio processing part with 16-bit resolution. Valid data bits and the time slot length setting must not be less than 16 bits. In case of PCM mode with higher valid serial data bits values, only the 16 MSBs of each left and right sample are taken into account: Input sample LSBs are discarded, whereas output sample LSBs are filled with zeros.

Note:

In PCM mode, the frame frequency must be 44.1 kHz or 48 kHz for data exchanges with an audio processing part.

AC97-only registers are not used: The codec port valid time slot registers (CPTVSLL and CPTVSLH) must be left at their reset value. The codec port address and data registers (CPTTADR, CPTDATL, and CPTATH) content is unknown.

Table 15–40 through Table 15–43 describe the configuration registers for PCM mode.

Table 15–40. CPCFR1 Configuration for PCM Mode (0X09)

Bit	Name	Function	Reset Value
15:8	RESERVED	Reserved	0
7:3	MTSL	2 time slots per audio frame	0x1
2:0	MODE	PCM mode	0x1

Table 15–41. CPCFR2 Configuration for PCM Mode (0X0C)

Bit	Name	Function	Reset Value
15:8	RESERVED	Reserved	0
7:6	TSOL	Serial clock cycles for time slot 0 are the same as for other time slots.	0x0

Table 15–41. CPCFR2 Configuration for PCM Mode (0X0C) (Continued)

Bit	Name	Function	Reset Value
5:3	BPTSL	16 data bits per audio time slot. Only the 16 MSBs are used.	0x1
2:0	TSLL	32 serial clock (BCLK) cycles for all time slots except time slot.	0x5

Table 15–42. CPCFR3 Configuration for PCM Mode (0X78)

Bit	Name	Function	Reset Value
15:8	RESERVED	Reserved	0
7	DDLY	One BIT_CLK cycle delay of the serial data output and input signals in reference to the leading edge of the SYNC signal.	0x0
6	TRSEN	Data serial outputs enabled during the audio frame are not valid.	0x1
5	CLKBP	The CSYNC signal is generated with the negative edge of the CSCLK signal. Also, the CDATO signal is generated with the negative edge of the CSCLK signal, and the CDATI signal is sampled with the positive edge of the CSCLK signal.	0x1
4	CSYNCP	The polarity of the codec port interface frame sync (CSYNC) output signal is active low during time slot 0. Time slot 0 = left sample.	0x1
3	CSYNCL	The length of the codec port interface frame sync (CSYNC) output signal is the same number of CSCLK cycles as time slot 0.	0x1
2	RESERVED	Reserved	0
1	CSCLKD	The direction of the codec port interface serial clock (CP_SCLK) signal is an output from the device.	0x0
0	CSYNCD	The direction of the codec port interface frame sync (CP_SYNC) signal is an output from the device.	0

Table 15–43. CPCFR4 Configuration for PCM Mode (0X03)

Bit	Name	Function	Reset Value
15:8	RESERVED	Reserved	0
7:4	ATSL	Not address/data time slot: ATSL = 0X0	0
3	CLKS	The source of the clock signal to be used for the divide-by-B circuit is the codec master system clock: CLKS = 0 (deleted).	0
2:0	DIVB	Divide by 4 (MCLK = 11.289 MHz, 64 CSCLK cycles per frame: MCLK must be divided by 4 to achieve a 44.1 kHz frame frequency): DIVB = 0x3	0x3

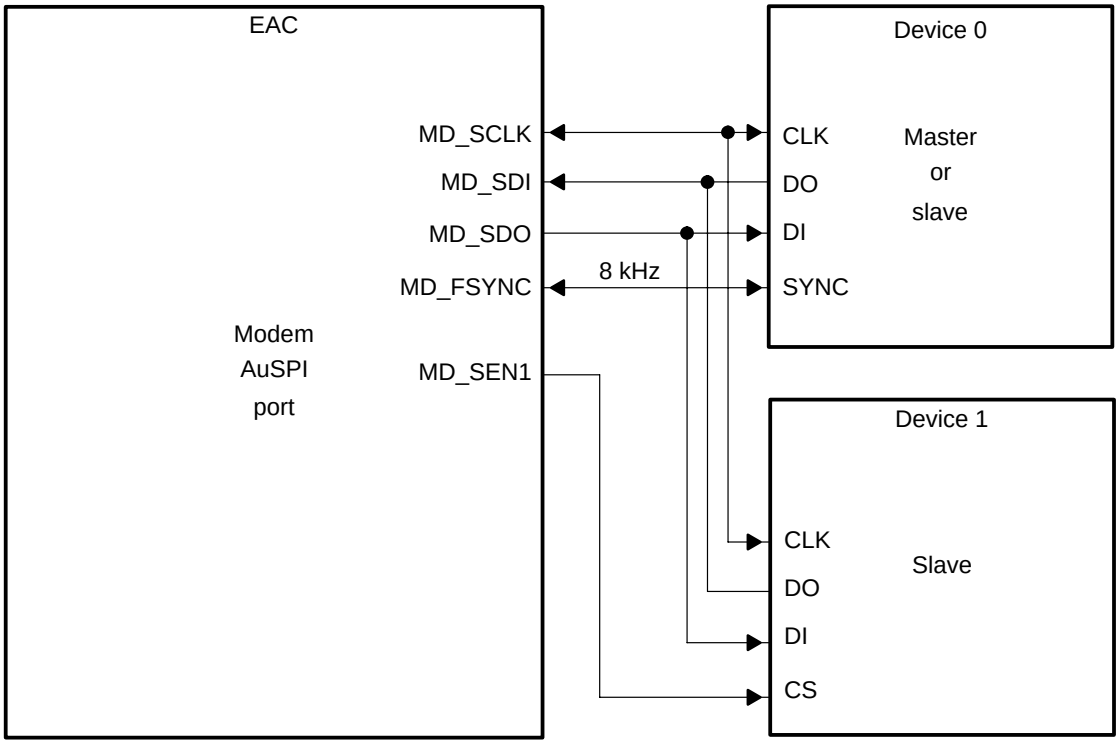
15.8 Modem Port Interface

15.8.1 Description

The modem interface is a bidirectional three-line interface dedicated to the transfer of data to and from external devices offering three-line serial interface. This serial port is based on a looped shift register, thus allowing both transmit (PISO) and receive (SIPO) modes. The modem serial-port interface is a time division multiplexed (TDM) time slot-based scheme.

The modem port interface can communicate with two different devices (from channel 0 or main channel and channel 1 or auxiliary channel) with the same lines (SCLK, SDI, SDO).

Figure 15–10. Connections of the EAC to AuSPI Devices



Communication on the main channel is handled with the frame synchronization signal (FSYNC). This signal is a one SCLK-wide pulse every 125 µs (8 kHz) or 62.5 µs (16 kHz). The communication between EAC modem port interface and the device on channel 0 can be in master or slave modes:

- Master mode: FSYNC and SCLK are generated by the modem port interface.
- Slave mode: FSYNC and SCLK are given by device 0, which is linked to the main channel.

Communication on the auxiliary channel is always done with the modem port interface in master mode. SPI Device1 on this channel is selected by the SEN1 chip-select pin and receives the serial clock.

In practice, the main channel is used as AuSPI (audio SPI) for voice/audio samples (serial data exchanged on this channel come from and go to the EAC audio processing units). The auxiliary channel is intended for use as a general-purpose SPI to configure, for example, the codec IC (serial data are exchanged through the memory-mapped registers).

The main channel and the auxiliary channel have independent serial settings.

The features of the modem port interface that can be configured are:

- Mode of operation of the main channel (master or slave)
- Frame rate of the main channel: 8 kHz or 16 kHz (wide-band) modes
- Frequency of the serial clock on the main channel
- Number of valid data bits on the main channel
- Relationship between the serial data bits and the serial clock edges on the main channel
- Use of μ-law and A-law compression for the main channel transmit path
- Use of μ-law and A-law compression for the main channel receive path
- Serial data bit alignment on the main channel
- Level of the frame synchro signal on the main channel
- Relationship between the frame synchro and the serial clock edges on the main channel
- Frequency of the serial clock on the auxiliary channel
- Number of valid data bits on the auxiliary channel
- Relationship between the serial data bits and the serial clock edges on the auxiliary channel
- Mode of operation of the chip-select (level or edge) on the auxiliary channel
- Active level of the chip-select on the auxiliary channel

The modem port interface channels can be enabled independently:

- Main channel enabled only: Samples are exchanged with the EAC audio processing part in full-duplex mode according to the frame synchronization signal. Audio samples are then automatically processed on an 8-/16-kHz rate. No more software action on the modem port is required after the channel enable.
- Auxiliary channel enabled only: data exchange occurs in full-duplex mode upon software request. The data transmit register content is serialized immediately. Data in the receive register can be read after the programmed number of data bits has been deserialized.
- Both main and auxiliary channels enabled: Main channel transfers still occur upon frame synchronization generation. Serial transfers on the

auxiliary channel still require software but take place only at the end of the next main channel transfer.

Note:

If both channels are enabled and the main channel is in slave mode, the device connected to the main channel must be able to drive its SDO and SCLK lines to 3-state level for the EAC to be able to communicate on the auxiliary channel. Moreover, if there are no exchanges on the main channel, communication with the auxiliary channel is broken.

15.8.2 Configuration

The main channel and the auxiliary channel on the modem port are programmed and enabled separately.

Modem port registers can be accessed only if the modem port clock source is running. The modem port interface clock source must be selected first (M_CKSRC in AGCFR register) if the default clock source is not provided to the EAC (see section 15.6.2.4, *Audio/Codec Port Clock Source*).

15.8.2.1 Main Channel Configuration

Main channel configuration capabilities:

- Operation mode: Master or slave
- Serial clock prescaling: 1, 2, 4, 8, 16
- Serial clock polarity: Serial data bits generated on SDO on negative/positive edge of SCLK; SDI sampled on the positive/negative edge of SCLK
- Frame synchronization level: FSYNC active high or low
- Frame synchronization polarity: FSYNC generated on negative/positive edge of SCLK
- Serial bit number to be transferred: 1 to 32
- Compand mode for receive: None, μ-law, A-law
- Expand mode for receive: None, μ-law, A-law
- Serial data justify: MSB or LSB alignment
- Justified data filling bits: Low or high bits

The MPMCCFR register is used for modem port main channel serial configuration. PRE_MC bits of the modem port control register (MPCTR) set the main channel serial clock prescaler.

These configuration fields can be updated only when the main channel is disabled (that is, MC_EN = 0 in the MPCTR register).

Table 15–44 through Table 15–49 describe the configuration of the modem main channel.

Table 15–44. Configuration of Modem Main Channel Mode

Register	Bit 8 (MCM)	Main Channel Mode	Reset Value
MPMCCFR	0	Slave	0x0
	1	Master	

In slave mode, the serial clock and the frame synchronization signals are generated by the device connected to main channel. MP_SCLK and MP_FSYNC are in input mode.

In master mode, the serial clock and frame synchronization are generated by the modem port. MP_SCLK and MP_FSYNC are in output mode.

Table 15–45. Configuration of Modem Serial Bit Clock Prescaler

	PRE_MC				
Register	Bit 3	Bit 2	Bit 1	Prescale Clock Divisor	Reset Value
MPCTR	0	0	0	1	0x0
	0	0	1	2	
	0	1	0	4	
	0	1	1	8	
	1	0	0	16	
	1	0	1	Reserved	
	1	1	0	Reserved	
	1	1	1	Reserved	

The serial bit clock prescaler sets the serial clock frequency when in master mode according to the modem port clock source frequency with a 50/50 duty-cycle: Serial clock frequency ranges from clock source frequency (that is, 13 or 12 MHz) with PRE_MC = 0x0 down to clock source frequency/16 (that is, 812 kHz and 750 kHz, respectively) when PRE_MC = 0x4.

Table 15–46. Configuration of Clock Polarity for Modem Main Channel

Register	Bit 5 (CPB_MC)	Clock Polarity for Main Channel	Reset Value
MPMCCFR	0	Falling	0x0
	1	Rising	

Clock polarity is set to select the active serial clock edge for serial bit exchanges: clock polarity falling (rising) means that serial data bits are updated on serial data output MP_SDO on the falling edge (rising edge) of the serial clock MP_SCLK. Serial data bits received on MP_SDI from the device connected to the main channel are sampled on the opposite edge.

Table 15–47. Configuration of Frame Synchronization Level for Modem Main Channel

Register	Bit 6 (FSBL_MC)	Frame Synchronization Level for Main Channel	Reset Value
MPMCCFR	0	Low	0x0
	1	High	

Frame synchronization level high configures the MP_FSYNC signal as a serial clock positive pulse to indicate the start of transfer on the main channel. When low, MP_FSYNC is a negative pulse.

Table 15–48. Configuration of Frame Synchronization Polarity for Modem Main Channel

Register	Bit 7 (FSP_MC)	Frame Synchronization Polarity for Main Channel	Reset Value
MPMCCFR	0	Falling	0x0
	1	Rising	

Frame synchronization with polarity falling (rising) changes the state on the serial clock falling edge (rising edge). Serial transfer starts on the first serial clock active edge that follows the clock edge where MP_FSYNC is activated.

Table 15–49. Configuration of Modem Compand Mode

	COMPAND			
Register	Bit 12	Bit 11	Compand Mode	Reset Value
MPMCCFR	0	0	Disable	0x0
	0	1	μ-law companding	
	1	0	A-law companding	
	1	1	Reserved	

Audio sample compression for transmit is provided though the COMPAND field:

- μ-law is a standard commonly used in the United States. It takes 14-bit data values and compresses them to 8-bit values.
- A-law is the standard used in Europe. It takes 13-bit data values and compresses them to 8-bit values.

When compression is enabled, 8 data bits are to be transmitted. When compression is disabled, audio samples from the audio processing have a 16-bit resolution.

Table 15–50. Configuration of Modem Expand Mode

	EXPAND			
Register	Bit 10	Bit 9	Expand Mode	Reset Value
MPMCCFR	0	0	Disable	0x0
	0	1	μ-law companding	

Table 15–50. Configuration of Modem Expand Mode (Continued)

Register	EXPAND Bit 10	EXPAND Bit 9	Expand Mode	Reset Value
	1	0	A-law companding	
	1	1	Reserved	

Expand modes are provided to receive compressed audio sample expansion to the 16-bit internal format of audio processing.

Compressed data have an 8-bit length, whereas uncompressed data have 16-bit resolution.

Table 15–51. Configuration of Transmission Bit Number for Modem Main Channel

Register	WS_MC Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Transmission Bit Number for Main Channel	Reset Value
MPMCCTR	0	0	0	0	0	1	0x00
	0	0	0	0	1	2	
	–	–	–	–	–	–	
	1	1	1	1	0	31	
	1	1	1	1	1	32	

Serial data bits are always transferred MSB first. 1 to 32 bits can be transmitted per frame.

Table 15–52. Configuration of Serial Data Alignment

Register	Bit 13 (DJUST_MC)	Serial Data Alignment	Reset Value
MPMCCFR	0	MSB alignment	0x0
	1	LSB alignment	

Serial data can be aligned to the serial data MSB or LSB, that is, the significant bits are the MSB or the LSB of the serial data bits. These bits are significant only if the serial length (in WS_MC) differs from the internal data length: 16 bits for uncompressed data, and 8 bits for μ-law/A-law compressed data.

Table 15–53. Configuration of Serial Data Filling

Register	Bit 14 (DFILL_MC)	Serial Data Filling Bits	Reset Value
MPMCCFR	0	Fill with zeros	0x0
	1	Fill with ones	

This bit is significant only if the serial length differs from the internal data length. Missing bits take the DFILL_MC value (examples in Section 15.7.3, *Codec Port Programming Limits*).

Table 15–54. Configuration of Serial Data Alignment

Register	Bit 5 (BT_MD_WIDE-BAND)	Serial Data Filling Bits	Reset Value
AGCFR2	0	WideBand OFF: 8 kHz	0x0
	1	WideBand ON: 16 kHz	

Note: Changes in the frame rate configuration for the modem port main channel simultaneously apply to the Bluetooth port main channel. The frame rate selection must be done before enabling any of the AuSPI's main channels.

Serial data are exchanged at a fixed frame rate that is either:

- ❏ 8 kHz: Standard voice signals sampling rate
- ❏ 16 kHz: Wide-band voice signals sampling rate

Selection between these two modes must be done before enabling the main channel.

15.8.2.2 Auxiliary Channel Configuration

Auxiliary channel SPI configuration capabilities:

- ❏ Serial clock prescaling: 1, 2, 4, 8, 16
- ❏ Serial clock polarity: Serial data bits generated on SDO on negative/ positive edge of SCLK, SDI sampled on the positive/negative edge of SCLK
- ❏ Device chip-select level: MD_SEN1 active high or low
- ❏ Device chip-select MD_SEN1 generated on negative/positive edge of SCLK
- ❏ Device chip-select mode: Active on level or on trigger
- ❏ Serial bit number to be transferred: 1 to 32

The MPACFR register is used for modem port auxiliary serial configuration. The PRE_AC bits of the modem port control register (MPCTR) set the auxiliary channel serial clock prescaler.

These configuration fields can be updated only when the auxiliary channel is disabled (that is, AC_EN = 0 in the MPCTR register).

Serial Bit Clock Prescaler

Table 15–55. Configuration of Modem Serial Bit Clock Prescaler

	PRE_AC				
Register	Bit 6	Bit 5	Bit 4	Prescale Clock Divisor	Reset Value
MPCTR	0	0	0	1	0x0
	0	0	1	2	

Table 15–55. Configuration of Modem Serial Bit Clock Prescaler (Continued)

Register	PRE_AC Bit 6	PRE_AC Bit 5	PRE_AC Bit 4	Prescale Clock Divisor	Reset Value
	0	1	0	4	
	0	1	1	8	
	1	0	0	16	
	1	0	1	Reserved	
	1	1	0	Reserved	
	1	1	1	Reserved	

The serial bit clock prescaler sets the serial clock frequency to the modem port clock source frequency with a 50/50 duty-cycle: The serial clock frequency ranges from the clock source frequency (that is, 13 or 12 MHz) with PRE_AC = 0x0 down to clock source frequency/16 (that is, 812 kHz and 750 kHz, respectively) when PRE_AC = 0x4.

Table 15–56. Configuration of Clock Polarity for Modem Main Channel

Register	Bit 5 (CPB_AC)	Clock Polarity for Main Channel	Reset Value
MPACCFR	0	Falling	0x0
	1	Rising	

Clock polarity is set to select the active serial clock edge for serial bit exchanges: The clock polarity falling (rising) means that serial data bits are updated on the serial data output MP_SDO on the falling edge (rising edge) of the serial clock MP_SCLK. Serial data bits received on MP_SDI from the device connected to the auxiliary channel are sampled on the opposite edge.

Table 15–57. Configuration of Chip-Select Level for Modem Auxiliary Channel

Register	Bit 6 (CSBL_AC)	Chip-Select Level for Auxiliary Channel	Reset Value
MPACCFR	0	Low	0x0
	1	High	

CSBL_AC defines the active level of MP_SEN1 that selects the device of the auxiliary channel for transmission.

Table 15–58. Configuration of Chip-Select Polarity for Modem Auxiliary Channel

Register	Bit 7 (CSP_AC)	Chip-Select Polarity for Auxiliary Channel	Reset Value
MPACCFR	0	Falling	0x0
	1	Rising	

When polarity is falling, the device chip-select is generated on a serial clock falling edge.

Device Chip Select Mode

Table 15–59. Configuration of Chip-Select Polarity for Modem Auxiliary Channel

Register	Bit 8 (CST_AC)	Chip-Select Trigger for Auxiliary Channel	Reset Value
MPACCFR	0	Level trigger	0x0
	1	Edge trigger	

When the device chip-select mode is on level, MP_SEN1 remains at its active level during the entire transfer. On trigger mode, the device chip-select is activated at the end of the transfer to store a word. Transfer ends after all the serial data bits have been exchanged.

Table 15–60. Configuration of Transmission Bit Number for Modem Auxiliary Channel

	WS_AC					Transmission Bit Number for Auxiliary Channel	Reset Value
Register	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0		
MPACCFR	0	0	0	0	0	1	0x00
	0	0	0	0	1	2	
	–	–	–	–	–	–	
	1	1	1	1	0	31	
	1	1	1	1	1	32	

Serial data bits are always transferred MSBs first. 1 to 32 bits can be transferred.

15.8.3 Main Channel Protocol Chronograms

15.8.3.1 Serial Signals Relationship

Serial protocol includes setting the frame synchronization MD_FSYNC signal and the serial data MD_SDI/MD_SDO signal relationships to the serial bit clock MD_SCLK.

Figure 15–11. Serial Signals Relationship—Example 1

Serial clock polarity falling: MPMCCFR.CPB_MC = 0
Frame synchro polarity falling: MPMCCFR.FSP_MC = 0
Frame synchro level high: MPMCCFR.FSBL_MC = 1

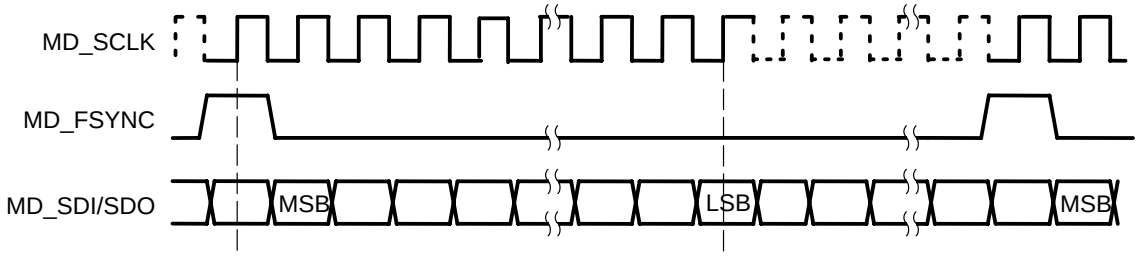


Figure 15–12. Serial Signals Relationship—Example 2

Serial clock polarity rising: MPMCCFR.CPB_MC = 1
Frame synchro polarity falling: MPMCCFR.FSP_MC = 0
Frame synchro level low: MPMCCFR.FSBL_MC = 0

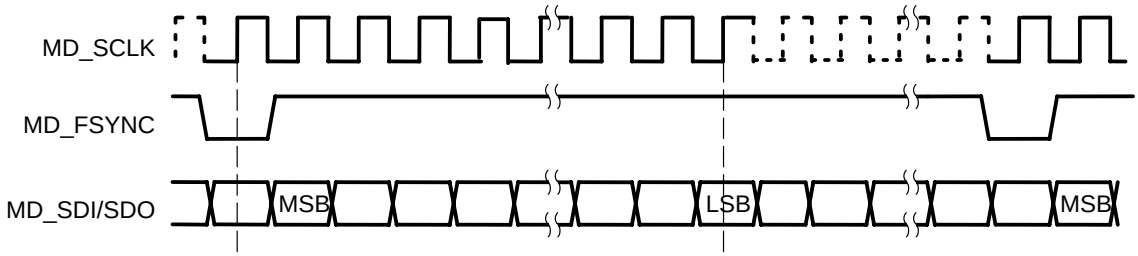


Figure 15–13. Serial Signals Relationship—Example 3

Serial clock polarity falling: MPMCCFR.CPB_MC = 0
Frame synchro polarity rising: MPMCCFR.FSP_MC = 1
Frame synchro level low: MPMCCFR.FSBL_MC = 1

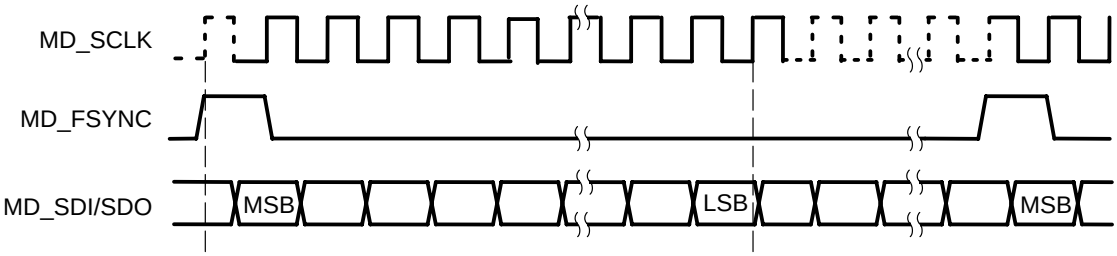
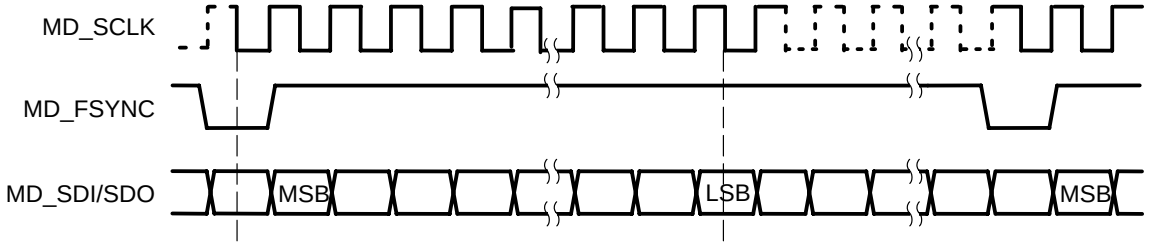


Figure 15–14. Serial Signals Relationship—Example 4

Serial clock polarity rising: MPMCCFR.CPB_MC = 1
Frame synchro polarity rising: MPMCCFR.FSP_MC = 1
Frame synchro level low: MPMCCFR.FSBL_MC = 0



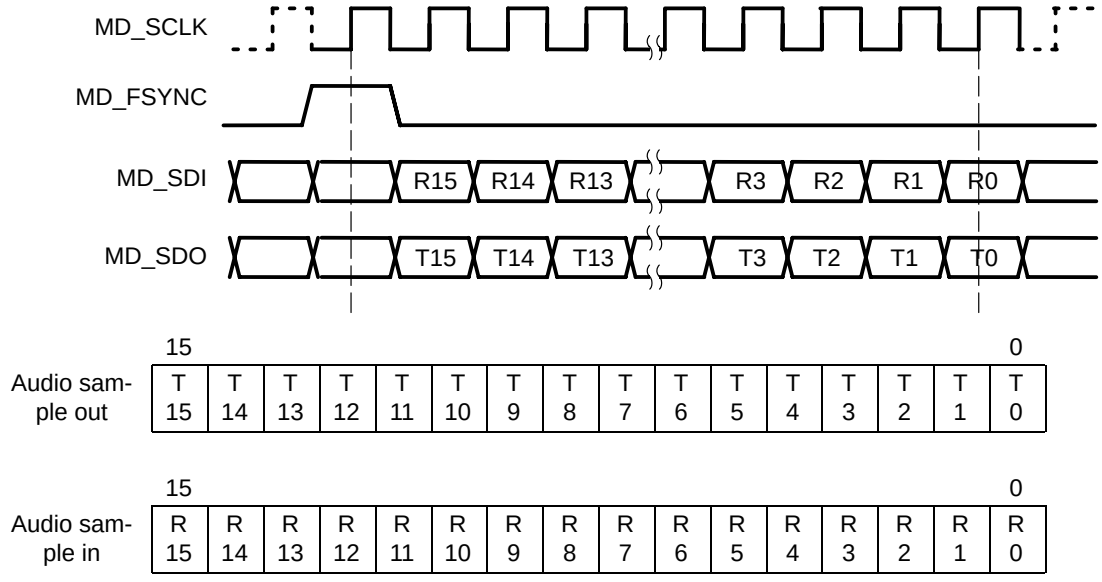
15.8.3.2 Serial Data Length to Internal Data Relationship

The main channel interfaces to the audio processing part with a fixed 16-bit resolution (uncompressed) or 8-bit, if compand/expand mode is selected. The serial part of the modem port adapts the serial stream bit number (WS_MC of MPMCCFR) to the internal sample width.

- ❑ Serial data length equal to internal parallel data length
  - ■ Serial data length: 16 bits, if compression is disabled
  - ■ Serial data length: 8 bits, if compand/expand compression is enabled

Figure 15–15. Serial Data Length Equal to Internal Parallel Data Length

16 serial data bits per transfer: MPMCCFR.WS_MC = 0x0F
Expand for data receive disabled: MPMCCFR.EXPAND = 0x0
Compand for data transmit disabled: MPMCCFR.COMPAND = 0x0



- ❑ Serial data length greater than internal parallel data length
  - ■ Serial data length: More than 16 bits, if compression is disabled
  - ■ Serial data length: More than 8 bits, if compand/expand compression is enabled

For receive path, extra LSBs (or MSBs) are discarded if data are aligned on MSBs (or LSBs).

For transmit path, missing LSBs (or MSBs) are padded with the DFILL_MC bit value for MSB alignment (or LSB alignment).

Figure 15–16. MSB Alignment—Example 1

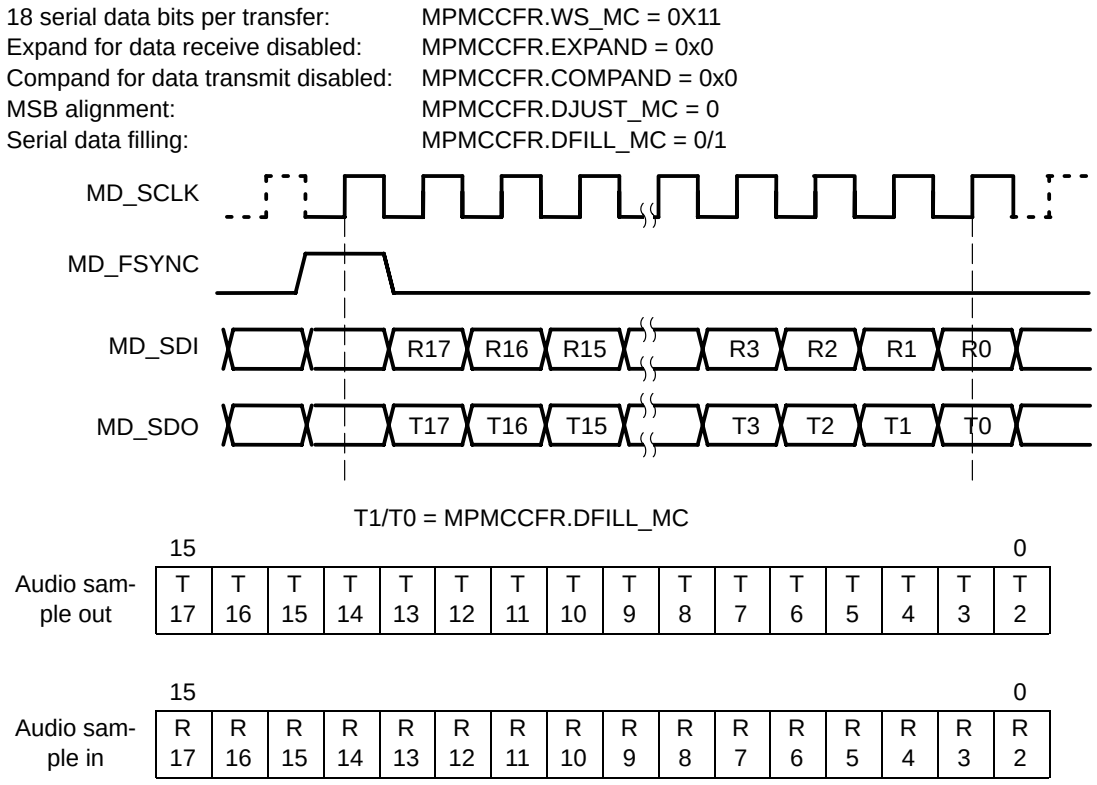
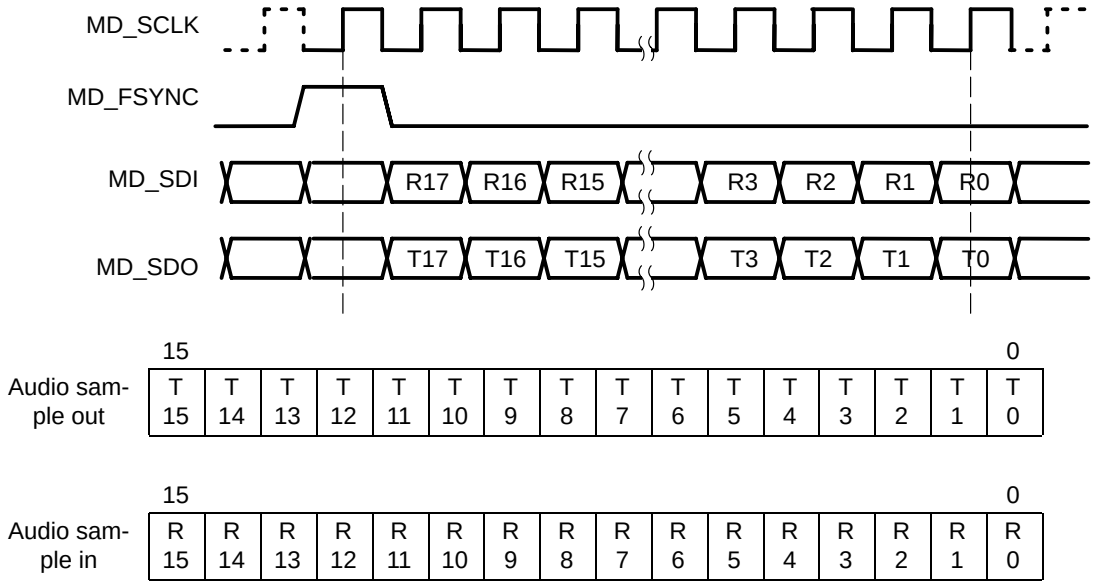


Figure 15–17. LSB Alignment—Example 2

18 serial data bits per transfer: MPMCCFR.WS_MC = 0X11
Expand for data receive disabled: MPMCCFR.EXPAND = 0x0
Compand for data transmit disabled: MPMCCFR.COMPAND = 0x0
LSB alignment: MPMCCFR.DJUST_MC = 1
Serial data filling: MPMCCFR.DFILL_MC = 0/1



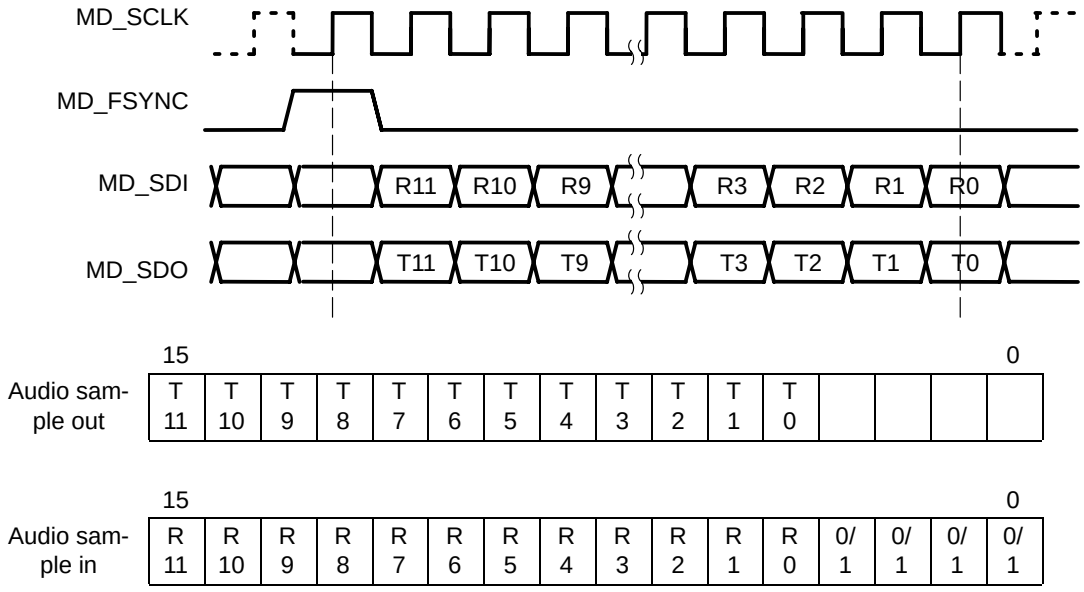
- ❑ Serial data length less than internal parallel data length
  - ■ Serial data length: Less than 16 bits, if compression is disabled
  - ■ Serial data length: Less than 8 bits, if compand/expand compression is enabled

For transmit path, extra LSBs (or MSBs) of audio sample that are not transmitted are discarded if data are aligned on MSBs (or LSBs).

For receive path, the received audio sample is rebuilt from the serial data bits received. Missing LSBs (or MSBs) are padded with the DFILL_MC bit value for MSB alignment (LSB alignment).

Figure 15–18. MSB Alignment—Example 1

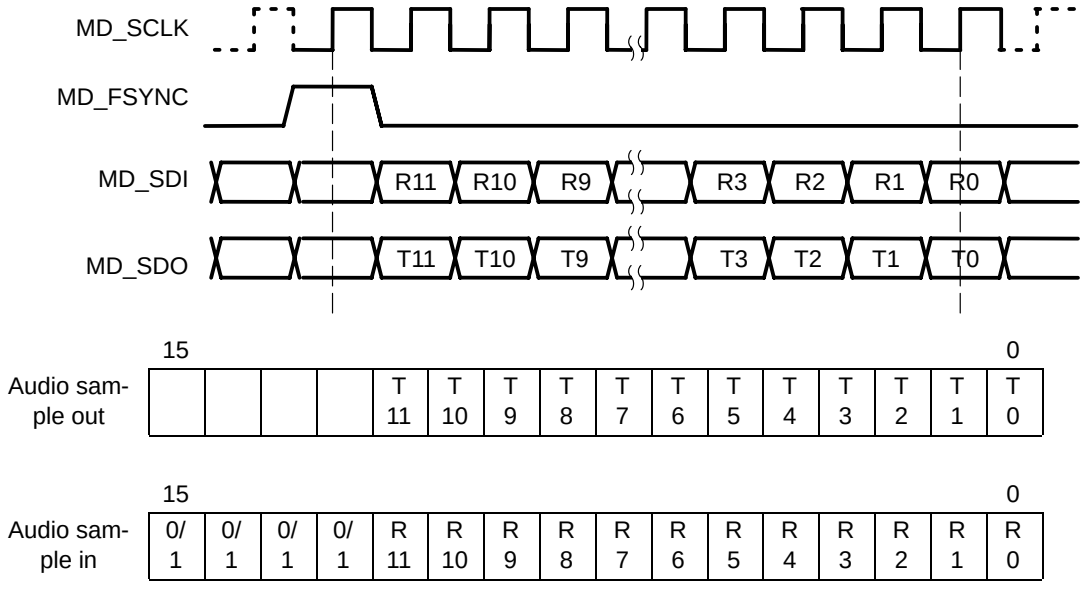
12 serial data bits per transfer: MPMCCFR.WS_MC = 0XB
Expand for data receive disabled: MPMCCFR.EXPAND = 0x0
Compand for data transmit disabled: MPMCCFR.COMPAND = 0x0
MSB alignment: MPMCCFR.DJUST_MC = 0
Serial data filling: MPMCCFR.DFILL_MC = 0/1



Sample in bits [3:0] is filled according to value of DFILL_MC.

Figure 15–19. LSB Alignment—Example 2

12 serial data bits per transfer: MPMCCFR.WS_MC = 0XB
Expand for data receive disabled: MPMCCFR.EXPAND = 0x0
Compand for data transmit disabled: MPMCCFR.COMPAND = 0x0
LSB alignment: MPMCCFR.DJUST_MC = 1
Serial data filling: MPMCCFR.DFILL_MC = 0/1

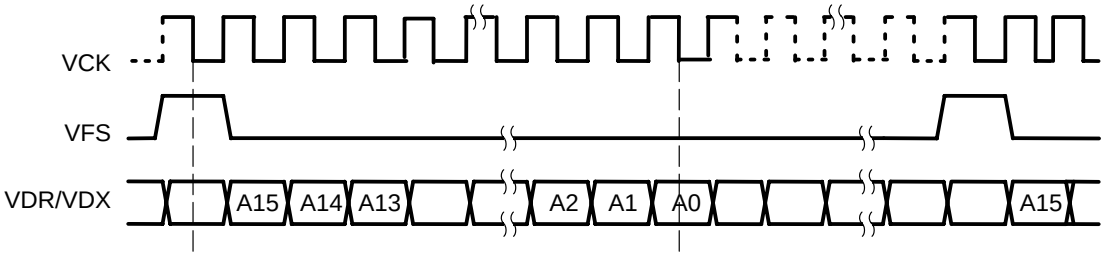


15.8.3.3 Programming Example: GSM DSP Voice Port Connection

An EAC modem port is provided to allow the connection of a GSM subsystem to a high-quality codec. In this application, the modem port replaces the voice band codec which is connected on the GSM DSP voice interface.

DSP Voice Serial Port Characteristics

Figure 15–20. DSP Voice Serial Port Characteristics



VCK frequency: 500 kHz
VCK duty cycle: 50/50
VCK and VFS is generated by analog baseband.

Modem Port Main Channel Configuration for Connection to DSP Voice Serial Port

Modem port in master mode: MPMCCFR.MCM = 1

Serial clock prescaler/16: MPCTR.PRE_MC = 0x4

16-bit serial transfer: MPMCCFR.WS_MC = 0x0F

Serial clock polarity rising: MPMCCFR.CPB_MC = 1

Frame synchro polarity rising: MPMCCFR.FSP_MC = 1

Frame synchro level high: MPMCCFR.FSBL_MC = 1

A-Law/μ-law compression disabled: MPMCCFR.EXPAND = 00
MPMCCFR.COMPAND = 00

Modem Port Configuration for Connection to Analog Baseband

In this case, the modem port is set in slave mode.

15.8.4 Auxiliary Channel Protocol Chronograms

The modem port is always master for transmissions on the auxiliary channel. The modem port interface generates the serial clock.

15.8.4.1 Serial Signals Relationship

Serial protocol includes setting the device chip-select MD_SEN1 and the serial data MD_SDI/MD_SDO signal relationship to the serial bit clock MD_SCLK.

Figure 15–21. Serial Signals Relationship—Example 1

Serial clock polarity falling:	MPACCFR.CPB_AC = 0
Device chip select polarity falling:	MPACCFR.CSP_AC = 0
Device chip select active on level:	MPACCFR.CST_AC = 0
Device chip select active low:	MPACCFR.CSBL_AC = 0

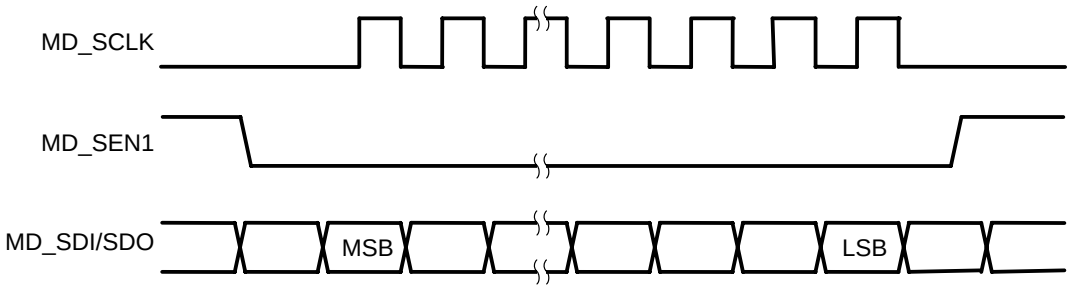


Figure 15–22. Serial Signals Relationship—Example 2

Serial clock polarity falling:	MPACCFR.CPB_AC = 1
Device chip select polarity falling:	MPACCFR.CSP_AC = 0
Device chip select active on level:	MPACCFR.CST_AC = 0
Device chip select active low:	MPACCFR.CSBL_AC = 1

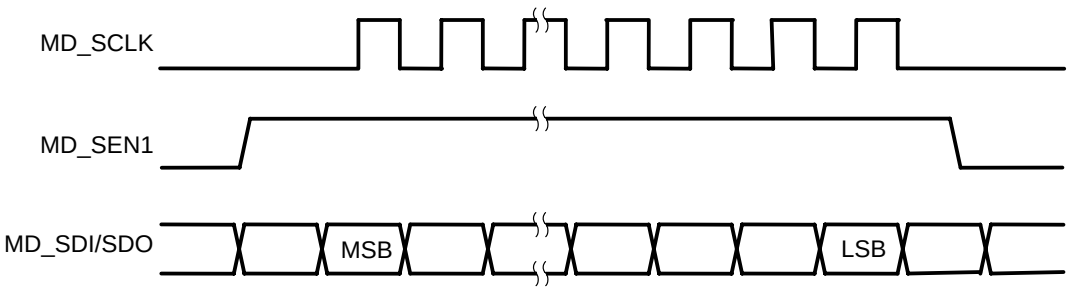
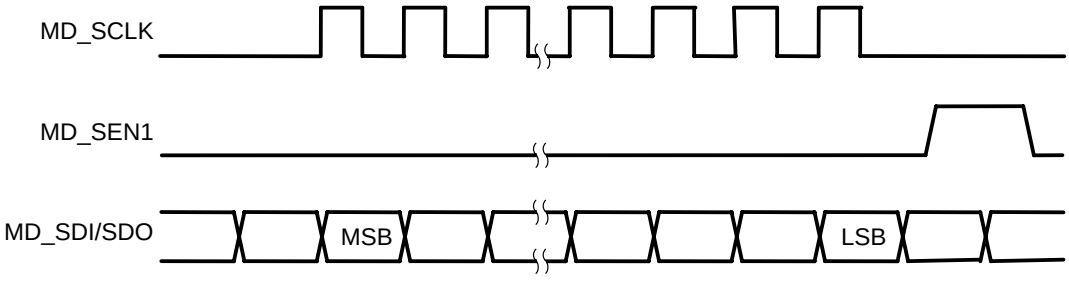


Figure 15–23. Serial Signals Relationship—Example 3

Serial clock polarity falling:	MPACCFR.CPB_AC = 1
Device chip select active on level:	MPACCFR.CST_AC = 0
Device chip select active low:	MPACCFR.CSBL_AC = 1



15.8.4.2 Serial Data Length

Data to be transmitted are written in the MPADMTR and MPADLTR registers (32-bit data high and low parts). Bits serialized on MP_SDO are taken from the MSBs.

Example:

Serial data length: 10 bits; MPADMTR[15:6] are transmitted

Serial data length: 20 bits: MPADMTR[15:0] and MPADLTR[15:12] are transmitted

Serial data sampled on MP_SDI are readable in the MPADMRR and MPADLRR registers (32-bit data high and low part). They are stored starting with LSBs.

Example:

Serial data length: 10 bits; data is on MPADLRR[9:0].

Serial data length: 20 bits; data is on MPADMRR[3:0] and MPADLRR[15:0].

15.9 Bluetooth Port Interface

15.9.1 Description

The serial interface is a bidirectional three-line interface dedicated to the transfer of data to and from external devices offering a three-line serial interface.

The Bluetooth port interface features are the same as the modem port interface.

Note:

The compand/expand data compression feature on the main channel is targeted for Bluetooth connection.

15.9.2 Configuration

Section 15.8.2 applies to the Bluetooth interface by replacing:

MPMCCFR with BPMCCFR

MPACCFR with BPACCFR

MPCTR with BPCTR

Note:

The frame-rate configuration (that is, 8 kHz or 16 kHz) for the Bluetooth port main channel also applies to the modem port main channel. The frame rate selection must be done before enabling any of the AuSPI main channels.

15.10 Sample-Rate Converter

15.10.1 Description

The EAC integrates four sample-rate converters (SRC) which make it possible to obtain the several audio data streams. Two SRCs are dedicated to the uplink path and two to the downlink path.

15.10.1.1 Uplink Data Stream

In the uplink path, the first SRC (SRC 2 in Figure 15–2) allows getting a data stream at the intermediate sample frequency. It integrates an interpolation filter and a 16-bit monophonic to a 16-bit left/right converter. Therefore, after the first SRC, the data can be used for the write DMA operation.

The second SRC (SRC 4 in Figure 15–2) allows getting the audio data at the codec sampling rate. This frequency is set automatically to 44.1 kHz or 48 kHz when a PCM or I2S codec is used; it is set to 48 kHz when an AC97 codec is used.

If the intermediate frequency is set to 44.1 kHz in PCM or I2S mode, the second SRC is bypassed. In this case, the audio codec frequency is equal to the intermediate frequency.

If the intermediate frequency is set to 8 kHz in all modes, only the 16-bit monophonic to 16-bit left/right converter is running on the first SRC. The up-sampler is bypassed.

15.10.1.2 Downlink Data Stream

In the downlink path, SRC 3 (in Figure 15–2) allows getting a data stream at the intermediate frequency. Therefore, after this SRC, the data can be used for the write DMA operation. The second SRC (SRC1 in Figure 15–2) adapts the audio rate to obtain an 8-kHz data flow. SRC 1 integrates a decimation filter and a 13-bit left/right to 16-bit monophonic converter.

15.10.2 Configuration

Table 15–61. Enable of Audio SRCs Processing

Register	Name	Bit 11	Audio Processing Enable
AGCTR	AUDEN	0	Disable
		1	Enable

There is no specific configuration for SRCs. The interpolation/decimation rates are given by the FSINT setting for DMA files play/record (see Section 15.11, *DMA Channels*).

Audio processing SRCs must be enabled for transfers to take place.

AUDEN = 0x0 (reset value)

15.10.3 Group Delay Inserted by SRCs

Sample-rate converters are based on FIR filters. A group delay is inserted on the audio data stream because of the necessity to store several extra samples before a calculation (interpolation or decimation) can occur.

Filter delay depends on:

- ❏ Input sampling frequency
- ❏ Output sampling frequency
- ❏ Mode (interpolation or decimation)

When the input sampling frequency equals the output sampling frequency, the operator is in bypass and group delay is reduced to a minimum.

The selection of an intermediate frequency for DMA channels then changes all path latencies.

Table 15–62 summarizes the internal path delays with all combinations of AuSPIs and codec port sampling frequency and intermediate frequency selections. Delays include the serialization/deserialization process at AuSPIs and codec interfaces.

Table 15–62. Interpolation Group Delays

AuSPI Port Sample Freq	DMA Freq	SRC2 Delay	Codec Port Sample Freq	SRC4 Delay	Total Delay
8 kHz	8 kHz	125 µs	44.1 kHz	695 µs	820 µs
	11.025 kHz	930 µs		525 µs	1455 µs
	22.05 kHz	840 µs		300 µs	1140 µs
	44.1 kHz	795 µs		25 µs	810 µs
	48 kHz	790 µs		170 µs	960 µs
	8 kHz	125 µs	48 kHz	690 µs	815 µs
	11.025 kHz	930 µs		620 µs	1550 µs
	22.05 kHz	840 µs		290 µs	1130 µs
	44.1 kHz	795 µs		180 µs	975 µs
	48 kHz	790 µs		20 µs	810 µs
16 kHz	8 kHz	825 µs	44.1 kHz	695 µs	1520 µs
	11.025 kHz	655 µs		525 µs	1180 µs
	22.05 kHz	470 µs		300 µs	770 µs
	44.1 kHz	410 µs		25 µs	435 µs
	48 kHz	410 µs		170 µs	580 µs
	8 kHz	825 µs	48 kHz	690 µs	1515 µs
	11.025 kHz	655 µs		620 µs	1275 µs
	22.05 kHz	470 µs		290 µs	760 µs
	44.1 kHz	410 µs		180 µs	590 µs
	48 kHz	410 µs		20 µs	430 µs

Note:

SRC2 delay is the delay due to AuSPI port deserialization and translation to DMA sampling frequency. SRC4 delay is the delay due to translation from DMA sampling frequency and codec port serialization.

Table 15–63. Decimation Group Delays

Codec Port Sample Freq	DMA Freq	SRC3 Delay	Codec Port Sample Freq	SRC1 Delay	Total Delay
44.1 kHz	8 kHz	785 µs	8 kHz	125 µs	910 µs
	11.025 kHz	570 µs		885 µs	1455 µs
	22.05 kHz	295 µs		885 µs	1180 µs
	44.1 kHz	25 µs		885 µs	910 µs
	48 kHz	180 µs		875 µs	1055 µs
48 kHz	8 kHz	770 µs		125 µs	795 µs
	11.025 kHz	570 µs		885 µs	1455 µs
	22.05 kHz	300 µs		885 µs	1185 µs
	44.1 kHz	170 µs		885 µs	1055 µs
	48 kHz	20 µs		875 µs	895 µs
44.1 kHz	8 kHz	785 µs	16 kHz	815 µs	1600 µs
	11.025 kHz	570 µs		640 µs	1210 µs
	22.05 kHz	295 µs		445 µs	740 us
	44.1 kHz	25 µs		445 µs	470 us
	48 kHz	180 µs		440 µs	620 us
48 kHz	8 kHz	770 µs		815 µs	1585 us
	11.025 kHz	570 µs		640 µs	1210 µs
	22.05 kHz	300 µs		445 µs	745 µs
	44.1 kHz	170 µs		445 µs	615 µs
	48 kHz	20 µs		440 µs	460 µs

Note:

SRC3 delay is the delay due to codec port deserialization and translation to DMA sampling frequency. SRC1 delay is the delay due to translation from DMA sampling frequency and AuSPI serialization.

15.11 DMA Channels

15.11.1 Description

The EAC provides two DMA channels for transferring data between all the combinations of endpoints and a memory: play DMA channel and record DMA channel. Therefore, audio data can be recorded in memory and the audio file can be played. Both channels can be enabled together or separately.

The audio data registers are double-buffered, which allows a continuous data stream.

The DMA event for the play operation is generated when the ADRDR register is empty. The DMA writes in this memory location. The DMA event for the record operation is generated when the ADWDR register is full. The DMA reads from this memory location.

EAC supports uncompressed audio files with 8- or 16-bit samples. The16-bit samples are in C2's format (–32768 to + 32767 centered on 0), whereas the 8-bit sample range is 0 to 512 centered on 256 (wave PCM files format).

EAC supports monophonic or stereo files. For stereo files, the left channel sample of a sample pair is written/read first (wave PCM files format).

15.11.2 Configuration

The intermediate sample frequency for DMA operations is: 8, 11.25, 22.05, 44.1, or 48 kHz).

Two sets of bits in two different registers are used to set the desired sampling frequency for DMA operations. One is kept for EAC-1 programming model backward compatibility.

Table 15–64. Configuration of Intermediate Sample Frequency Through AGCFR2

Register	Name	Bit 2		Bit 0	Intermediate Sample Frequency FSINT
AGCFR2	FSINT2	0	0	0	8 kHz
		0	0	1	11.025 kHz
		0	1	0	22.05 kHz
		0	1	1	44.1 kHz
		1	0	0	48 kHz
		Others			Reserved
		1	1	1	AGCFR FSINT used

FSINT2 = 0x7 (reset value)

Table 15–65. Configuration of Intermediate Sample Frequency Through AGCFR

Register	Name	Bit 7	Bit 6	Intermediate Sample Frequency FSINT
AGCFR	FSINT	0	0	8 kHz
		0	1	11.025 kHz

Table 15–65. Configuration of Intermediate Sample Frequency Through AGCFR (Continued)

Register	Name	Bit 7	Bit 6	Intermediate Sample Frequency FSINT
		1	0	22.05 kHz
		1	1	44.1 kHz

FSINT = 0x1 (reset value)

Table 15–66. Audio File Endianism Configuration

Register	Name	Bit 8	Audio File Endianism
AGCFR	LI_BI	0	Little endian
		1	Big endian

LI_BI = 0x0 (reset value)

Table 15–67. Audio Sample Size Configuration

Register	Name	Bit 9	Audio Sample Size
AGCFR	8/16B	0	8-bit samples
		1	16-bit samples

8/16B = 0x1 (reset value)

Table 15–68. Stereo Audio File Configuration

Register	Name	Bit 10	Audio File Stereo
AGCFR	MN_ST	0	Monophonic file
		1	Stereo file

MN_ST = 0x1 (reset value)

Table 15–69. Enable/Disable of Audio File Play Operation

Register	Name	Bit 12	Audio File Play Operation
AGCTR	DMAREN	0	Disable
		1	Enable

DMAREN = 0x0 (reset value)

Table 15–70. Enable/Disable of Audio File Record Operation

Register	Name	Bit 11	Audio File Record Operation
AGCTR	DMAWEN	0	Disable
		1	Enable

DMAWEN = 0x0 (reset value)

15.12 μ-Law/A-Law Companding

15.12.1 Description

Companding (COMpress and exPAND) hardware allows data to be compressed and expanded in either μ-law or A-law. μ-law and A-law allow14 bits and 13 bits of dynamic range, respectively. Any value outside this range is set to the most positive or most negative value. The μ-law and A-law formats encode data into 8-bit code words. Companded data is always 8 bits wide.

When companding is used, transmit data is encoded and receive data is decoded.

15.12.2 Configuration

Compand is configured on the modem port and the Bluetooth port via COMPAND and EXPAND of the modem port configuration register (MPCFR) and the Bluetooth port configuration register (BPCFR). See Section 15.8.2 for modem port compand configuration and Section 15.9.2 for Bluetooth port compand configuration.

Companding mode (μ-law companding or A-law companding) can be disabled.

15.13 Sidetone

15.13.1 Description

Close to C-port, the sidetone adds a part of the input of the codec audio link to the uplink audio stream. It is made up of a mixer and an attenuator. This block inserts a delay of less than four FS clock periods. In the worst case, the inserted delay is < 0.1 ms.

15.13.2 Configuration

Table 15–71. Configuration of Audio Sidetone—Disable/Enable

Register	Name	Bit 0	Audio Sidetone
ASTCTR	ATTEN	0	Disable
		1	Enable

ATTEN = 0x0 (reset value)

Table 15–72. Configuration of Sidetone Attenuation

Register	Name	Bit 7:1	Sidetone Attenuation
ASTCTR	ATT	X	Gain = 12 – 0.5 * (X – 127) dB

Attenuator level is given by:

- GAIN = 12 dB – 0.5 * (Data – 127) dB
- ATT = 0x67 (reset value) → Gain = 0 dB

15.14 Mixer

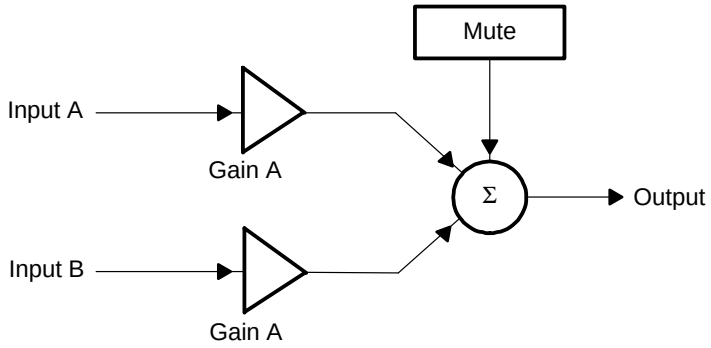
15.14.1 Description

There are three digital mixers on the EAC as illustrated in Figure 15–2. These mixers make different audio scenarios possible, including:

- Record the phone call in normal communication
- Play a system sound in normal communication
- Listen to music in normal communication

The digital mixers add the inputs together according to the settings in the mixer X volume control registers.

Figure 15–24. Mixer Functional Diagram



All volume controls implement controlled volume steps at nominally 0.5 dB per step.

Gain level is given by:

GAIN = 12 – 0.5 * (127 – DATA) dB

Table 15–73. Mixer Gain Level

Data	Gain
7Fh	+12 dB
7Eh	+11.5 dB
67h	0 dB (reset value)
E6h	–0.5 dB
01h	–51 dB
00h	–51.5 dB

15.14.2 Configuration

The audio mixer switch-configuration register (AMSCFR) allows different audio configurations.

Table 15–74. Switch Configurations

Scenario	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12
Normal communication (normal phone call)		■		■			■		■			
Normal communication and phone call recording		■		■	■	■	■		■			
Play a system sound in normal communication	■	■		■			■		■			
Listen to music in normal communication such that caller and person being called can both hear	■	■	■	■			■		■			
Communication with headset							■			■		■
Communication with headset and recording						■	■			■		■
Listen to music in communication with headset			■				■			■	■	■
Record a message					■							
Listen to music	■											
Record a message from headset						■		■				
Listen to music with headset			■									

■ Closed □ Open

The states of switches K1 to K12 are configured by bits 0 to 11 of AMSCFR, respectively: 0 for opened and 1 for closed.

Each mixer has a corresponding audio mixer volume control register. This register controls the volume of each mixer input:

Reset value: MUTE = 0x0 GINB = 0x67 GINA = 0x67

Table 15–75. Audio Mixer Volume Configuration

Mixer	Register	Name	Bits	Function
1	AM1VCTR	MUTE	15	Mute (1) or not mute (0) on mixer 1 output
		GINB	14:8	Gain on input B of mixer 1
		Reserved	7	Reserved
		GINA	6:0	Gain on input A of mixer 1
2	AM2VCTR	MUTE	15	Mute (1) or not mute (0) on mixer 2 output
		GINB	14:8	Gain on input B of mixer 2
		Reserved	7	Reserved

Table 15–75. Audio Mixer Volume Configuration (Continued)

Mixer	Register	Name	Bits	Function
		GINA	6:0	Gain on input A of mixer 2
3	AM3VCTR	MUTE	15	Mute (1) or not mute (0) on mixer 3 output
		GINB	14:8	Gain on input B of mixer 3
		Reserved	7	Reserved
		GINA	6:0	Gain on input A of mixer 3

15.15 DMA Volume Control

15.15.1 Description

For audio coming from or sending to DMA read or write channels, the left and right data volumes can be controlled.

All volume controls implement controlled volume steps at nominally 0.5-dB per step.

Gain level is given by:

Gain = 12 – 0.5 * (255 – DATA) dB

Table 15–76. Gain Level for DMA Volume Control

Data	Gain
FFh	+12 dB
FEh	+11.5 dB
E7h	0 dB (reset value)
E6	–0.5 dB
01h	–115.5 dB
00h	Mute

15.15.2 Configuration

The gain level on each DMA channel is configured using the audio master volume control register (AMVCTR):

Reset value: GWO = 0xE7 GRO = 0xE7

Table 15–77. DMA Channel Volume Configuration

Register	Name	Bits	Function
AMVCTR	GWO	15:8	Gain on write DMA channel: GAIN = 12 – 0.5 * (255 – GWO) dB
	GRO	7:0	Gain on read DMA channel: Gain = 12 – 0.5 * (255 – GRO) dB

15.16 Peak Detectors

15.16.1 Description

For volume control, signal levels are monitored by four peak detectors. The peak signal sample value is kept in registers and cleared upon reading.

- Peak detector 1 (codec port input):

  This peak detector monitors the signal level on the samples received from the codec (that is, the signal level of the external codec microphone input).

  Reading the APD1LCR register gives the signal level on the left channel; APD1RCR is used for the right channel.

- Peak detector 2 (codec port input):

  This peak detector monitors the signal level on the SRC1 output after the sidetone mixer (that is, the level of the signal samples sent to the external codec).

  Reading the APD2LCR register gives the signal level on the left channel; APD2RCR is used for the right channel.

- Peak detector 3 (DMA write output):

  This peak detector monitors the signal level of the audio file written into memory through the DMA write channel (that is, recording level).

  Reading the APD3LCR register gives the signal level on the left channel; APD3RCR is used for the right channel.

- Peak detector 4 (AuSPI port output):

  This peak detector monitors the signal level on the SRC3 output (the level of the signal samples sent through Bluetooth or modem output ports).

  Reading the APD4R register gives the signal level on this channel (monophonic).

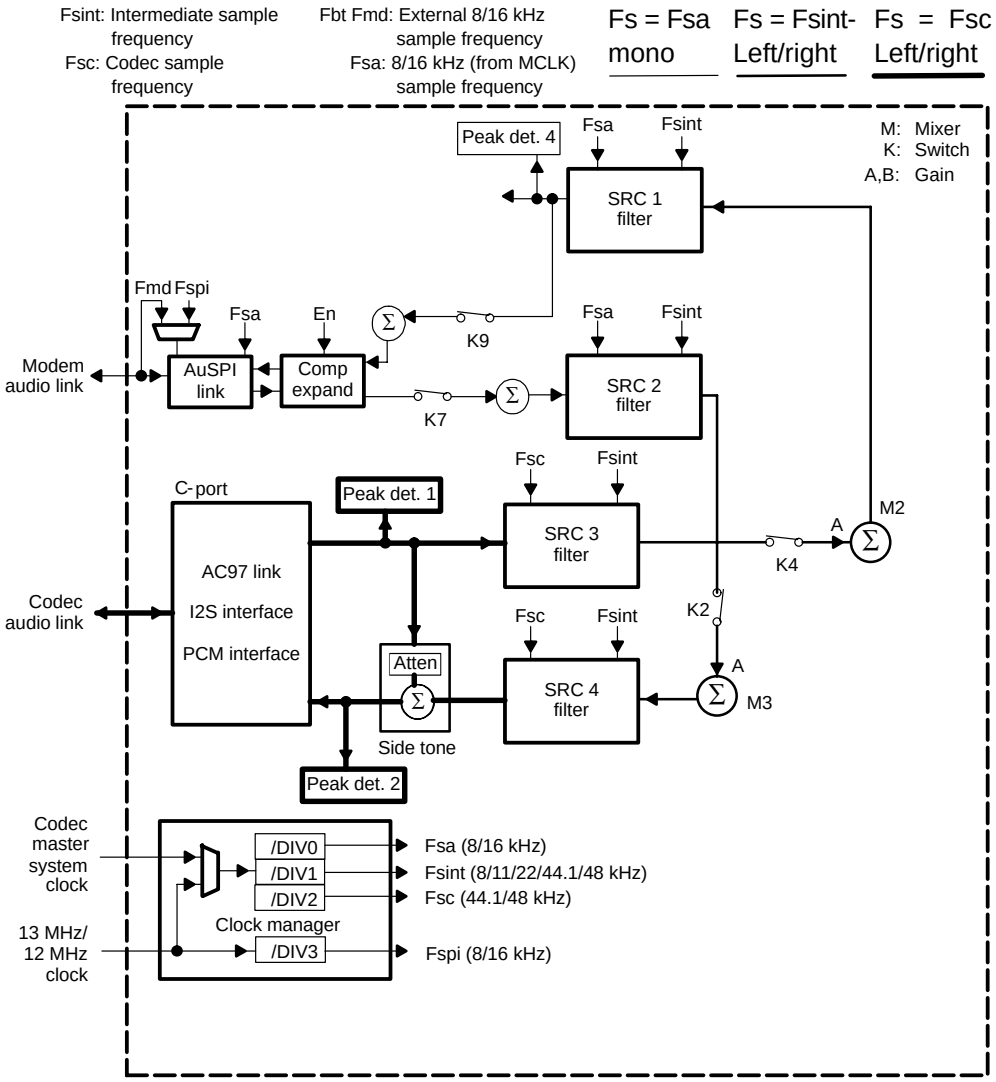
For these registers, reset value = 0x0.

Table 15–78. Audio Peak Detector Registers

Register	Name	Bits	Function
APD1LCR	PEAK	15:0	Peak value on the codec port input left channel
APD1RCR	PEAK	15:0	Peak value on the codec port input right channel
APD2LCR	PEAK	15:0	Peak value on the codec port output left channel
APD2RCR	PEAK	15:0	Peak value on the codec port output right channel
APD3LCR	PEAK	15:0	Peak value on the DMA write output left channel
APD3RCR	PEAK	15:0	Peak value on the DMA write output right channel
APD4LCR	PEAK	15:0	Peak value on the AuSPI port output left channel
APD4RCR	PEAK	15:0	Peak value on the AuSPI port output right channel

15.17 Application: Normal Phone Call

Figure 15–25. Normal Phone Call



This application uses the modem AuSPI and codec ports.

Voice coming from the modem AuSPI port is sent to the codec port. Voice coming from the codec device is sent to the modem AuSPI.

The Bluetooth AuSPI port input is not used.

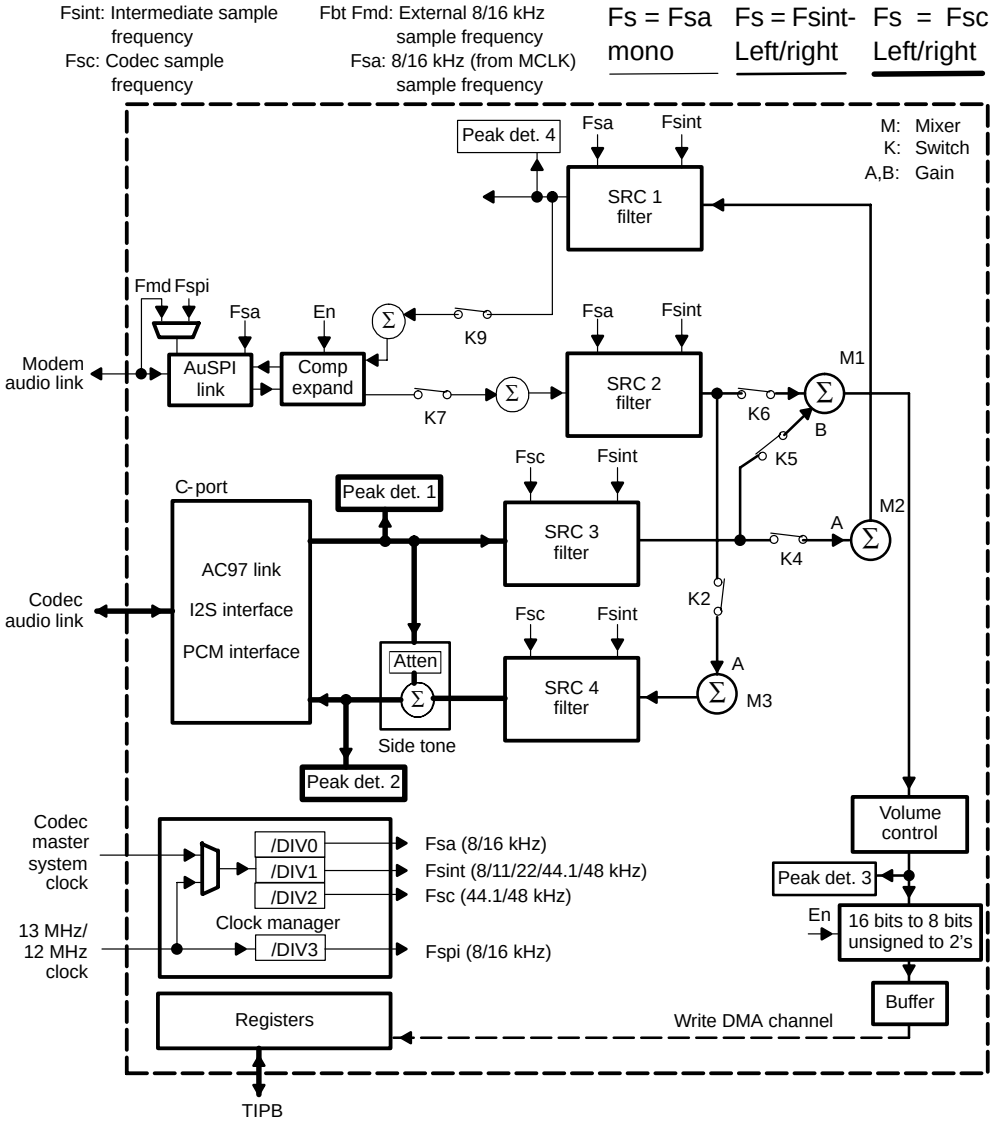
Audio mixer switches configuration: AMSCFR = 0x014C

Table 15–79. AMSCFR Configuration for Normal Phone Call

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				K12	K11	K10	K9	K8	K7	K6	K5	K4	K3	K2	K1
Value	Reserved				0	0	0	1	0	1	0	0	1	0	1	0

15.18 Application: Normal Phone Call With Record

Figure 15–26. Normal Phone Call With Record



This application uses the modem AuSPI port, the codec port, and the DMA.

The DMA records in memory the voice coming from the modem AuSPI port or the codec port.

The Bluetooth port is not used.

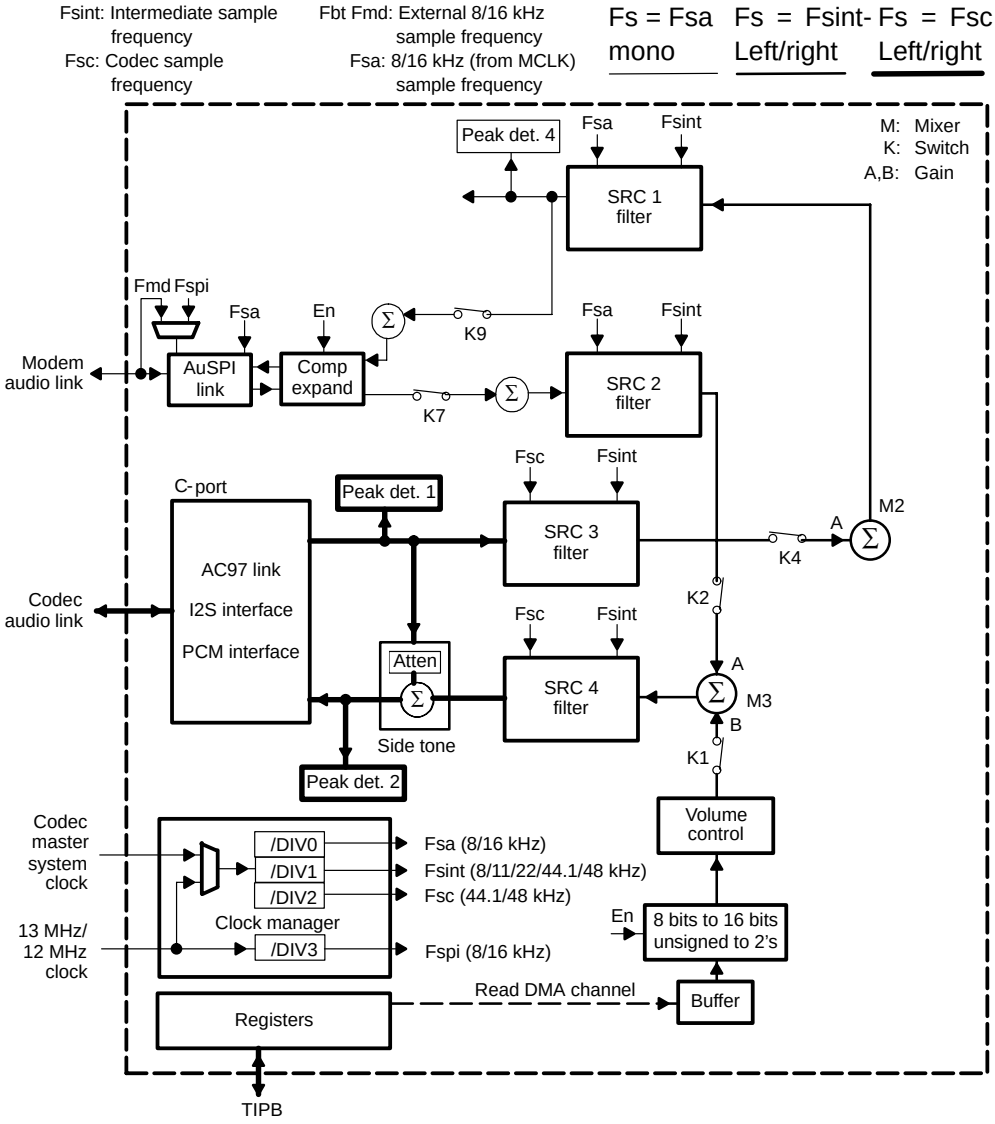
Audio mixer switches configuration: AMSCFR = 0x017C

Table 15–80. AMSCFR Configuration for Normal Phone Call With Record

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				K12	K11	K10	K9	K8	K7	K6	K5	K4	K3	K2	K1
Value	Reserved				0	0	0	1	0	1	1	1	1	0	1	0

15.19 Application: Normal Phone Call With Play

Figure 15–27. Normal Phone Call With Play



This application uses the modem AuSPI port, the codec port, and the DMA.

Play a system sound in normal communication. DMA sends the voice recorded in memory to the codec port only, and not to the modem AuSPI port.

The Bluetooth port is not used.

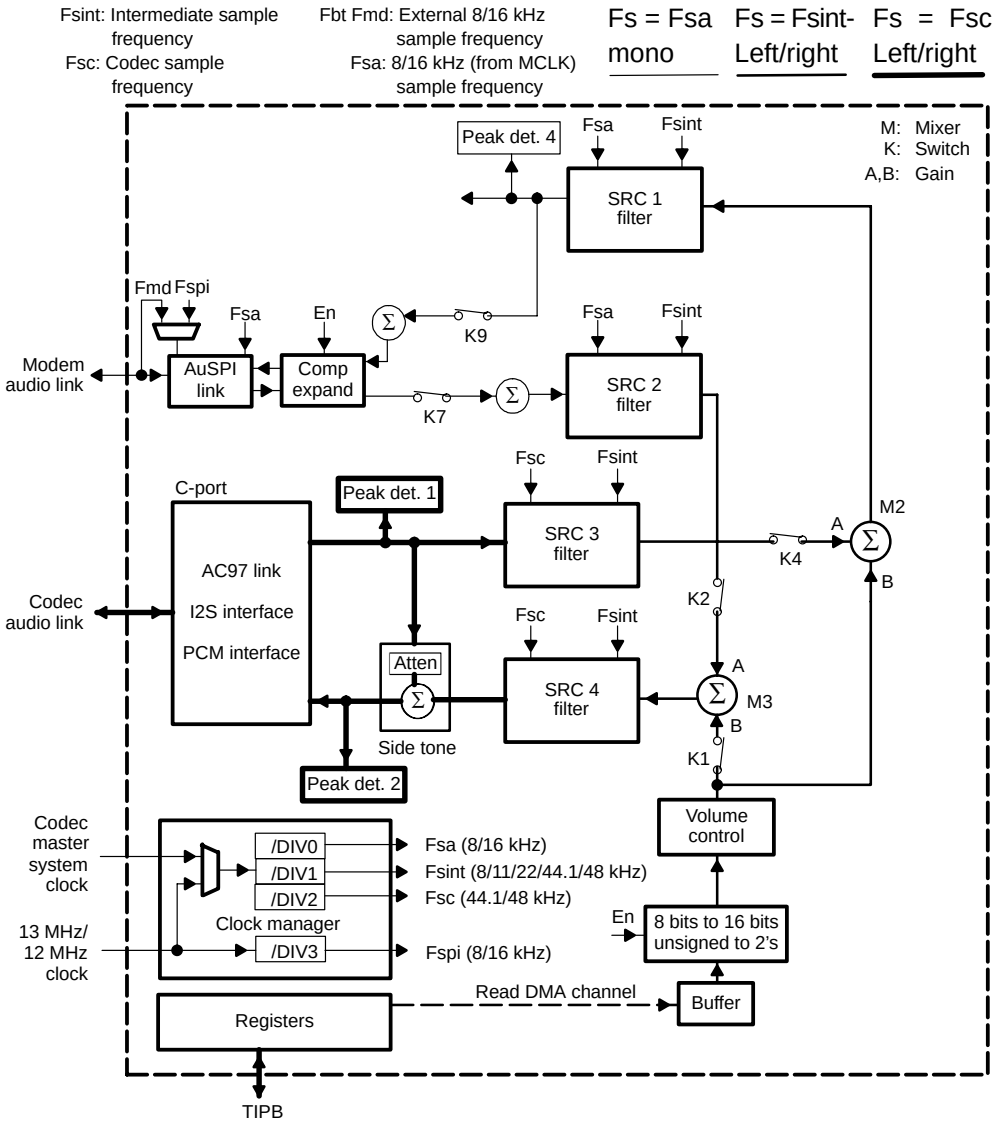
Audio mixer switches configuration: AMSCFR = 0x014D

Table 15–81. AMSCFR Configuration for Normal Phone Call With Play

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				K12	K11	K10	K9	K8	K7	K6	K5	K4	K3	K2	K1
Value	Reserved				0	0	0	1	0	1	0	0	1	0	1	1

15.20 Application: Normal Phone Call With Music

Figure 15–28. Normal Phone Call With Music Listening



This application uses the modem AuSPI port, the codec port, and the DMA.

Listen to music in normal communication such that the caller and the person being called can both hear. The DMA sends the voice recorded in memory to the codec port and the modem AuSPI port.

The Bluetooth AuSPI is not used.

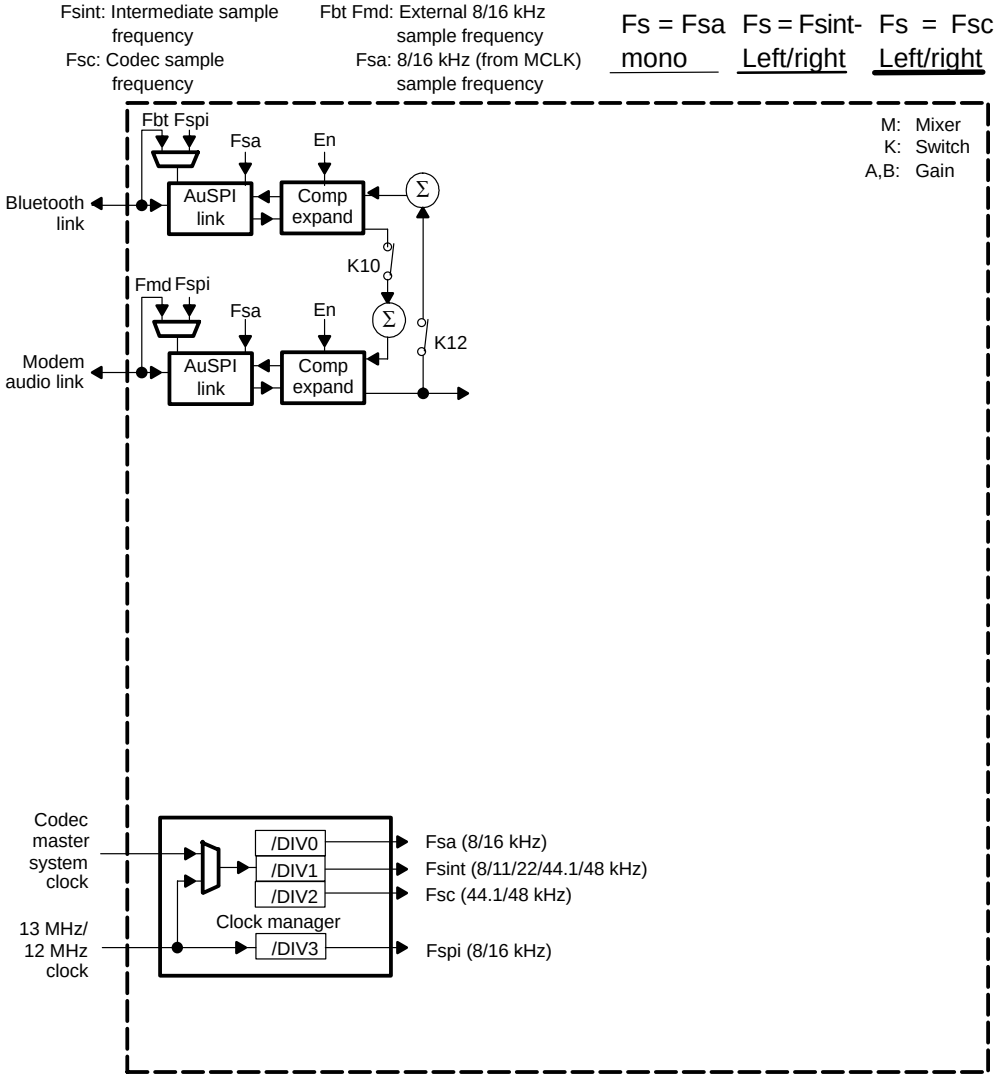
Audio mixer switches configuration: AMSCFR = 0x014F

Table 15–82. AMSCFR Configuration for Normal Phone Call With Music Listening

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				K12	K11	K10	K9	K8	K7	K6	K5	K4	K3	K2	K1
Value	Reserved				0	0	0	1	0	1	0	0	1	1	1	1

15.21 Application: Communication With Headset

Figure 15–29. Communication With Headset



This application uses the Bluetooth AuSPI port and the modem AuSPI port.

Communication with headset: The modem AuSPI port and the Bluetooth AuSPI port communicate bidirectionally.

The DMA and codec ports are not used.

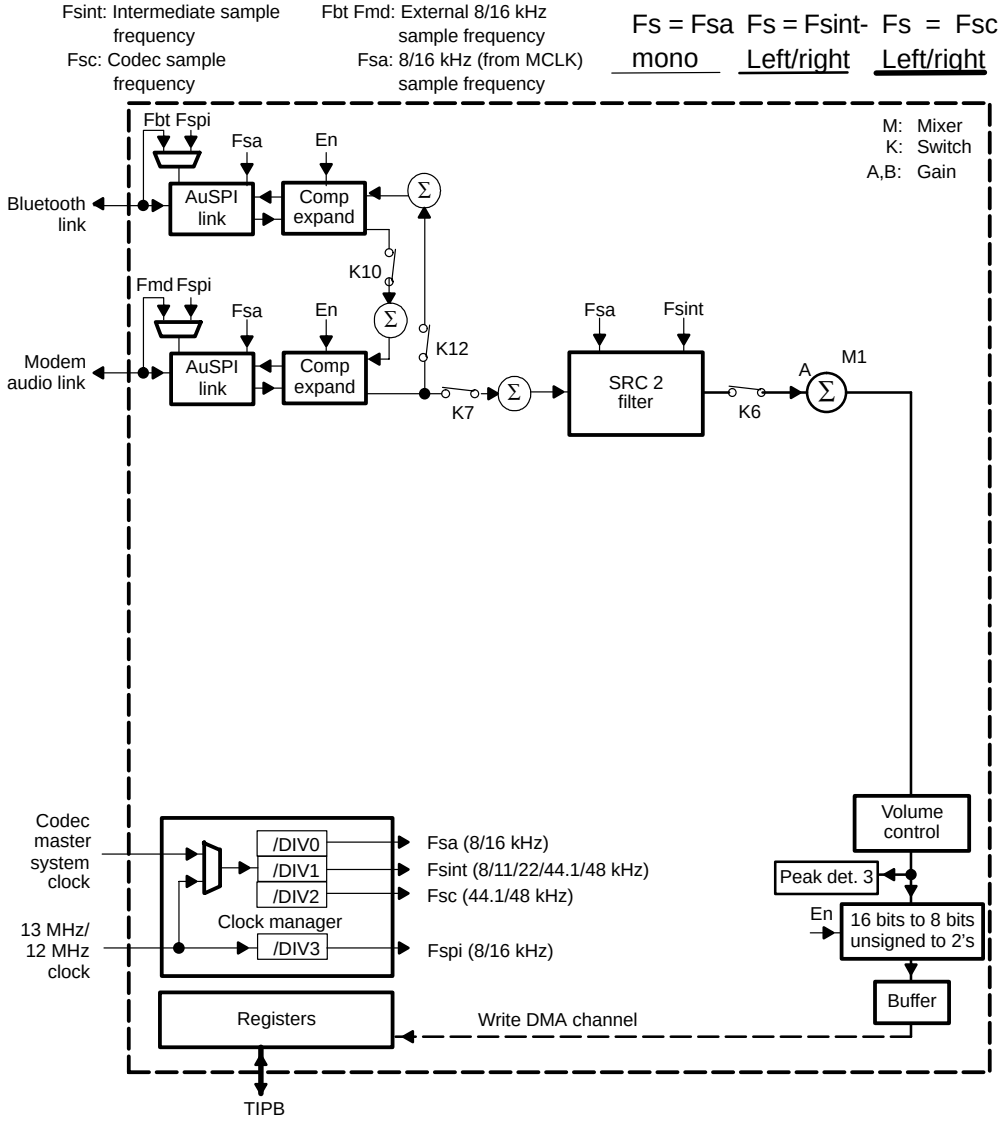
Audio mixer switches configuration: AMSCFR = 0x0A00

Table 15–83. AMSCFR Configuration for Communication With Headset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				K12	K11	K10	K9	K8	K7	K6	K5	K4	K3	K2	K1
Value	Reserved				1	0	1	0	0	0	0	0	0	0	0	0

15.22 Application: Communication With Headset and Record

Figure 15–30. Communication With Headset and Record



This application uses the Bluetooth AuSPI port, the modem AuSPI port, and the DMA.

Communication with headset and record. The modem AuSPI port and the Bluetooth AuSPI port communicate bidirectionally.

DMA records in memory the voice coming from the modem AuSPI port.

The codec port is not used.

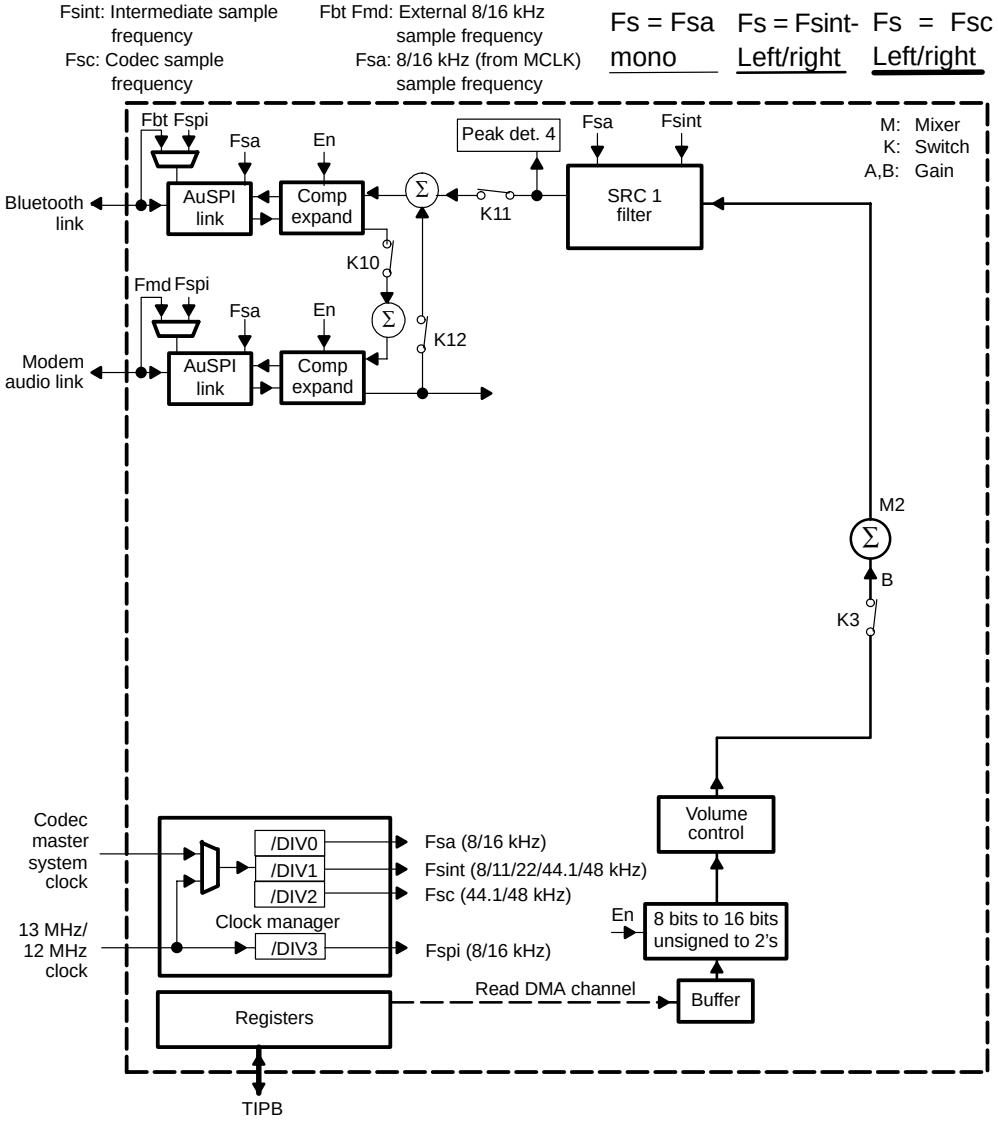
Audio mixer switches configuration: AMSCFR = 0x0A60

Table 15–84. AMSCFR Configuration for Communication With Headset and Record

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				K12	K11	K10	K9	K8	K7	K6	K5	K4	K3	K2	K1
Value	Reserved				1	0	1	0	0	1	1	0	0	0	0	0

15.23 Application: Communication With Headset and Music

Figure 15–31. Communication With Headset and Music Listening



This application uses the Bluetooth AuSPI port, the modem AuSPI port, and the DMA.

Play a system sound in communication with headset. The DMA sends the voice recorded in memory to the Bluetooth AuSPI port only, and not to the modem AuSPI port.

The codec port is not used.

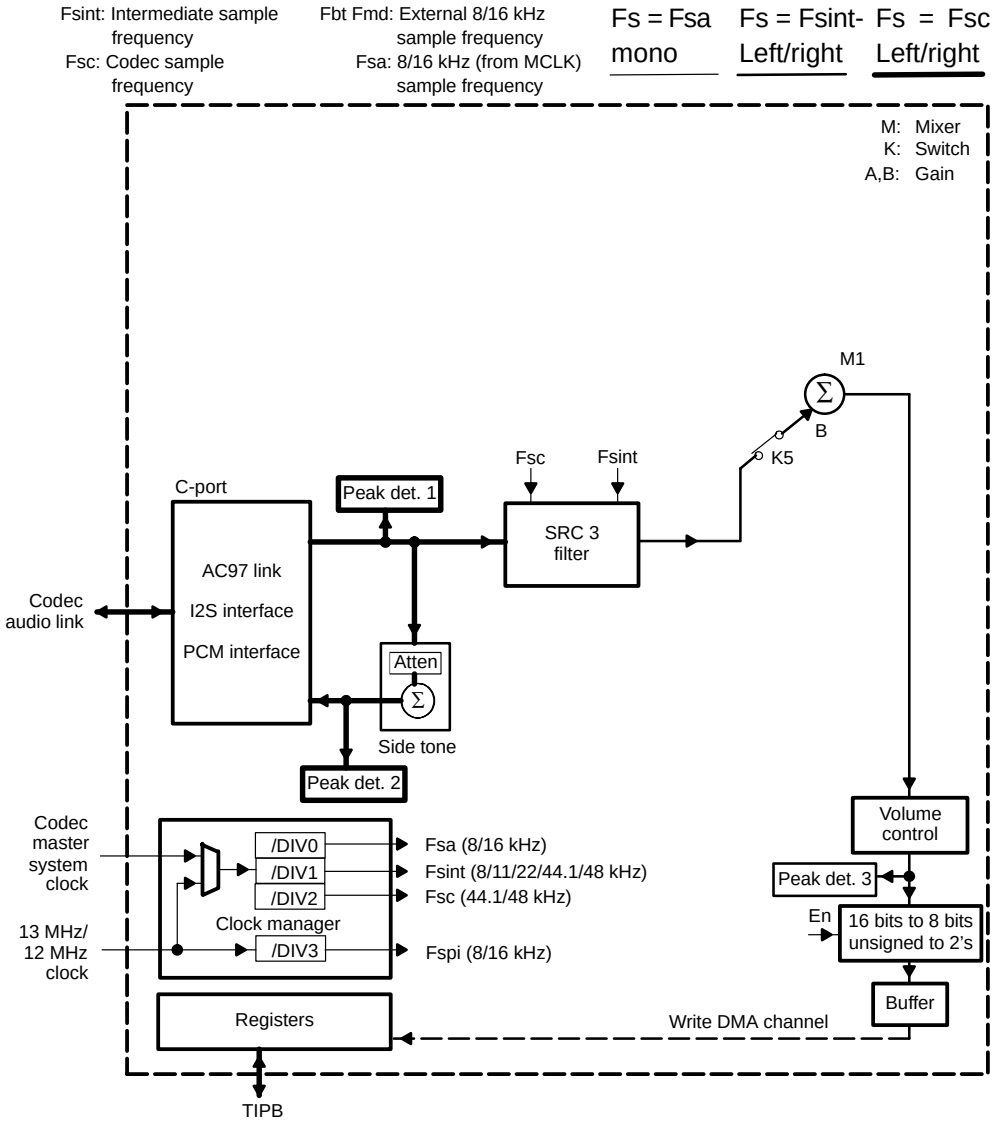
Audio mixer switches configuration: AMSCFR = 0x0E04

Table 15–85. AMSCFR configuration for Communication With Headset and Music

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				K12	K11	K10	K9	K8	K7	K6	K5	K4	K3	K2	K1
Value	Reserved				1	1	1	0	0	0	0	0	0	1	0	0

15.24 Application: Record a Message

Figure 15–32. Record a Message



This application uses the codec port and DMA.

The DMA records in memory the voice coming from the codec port.

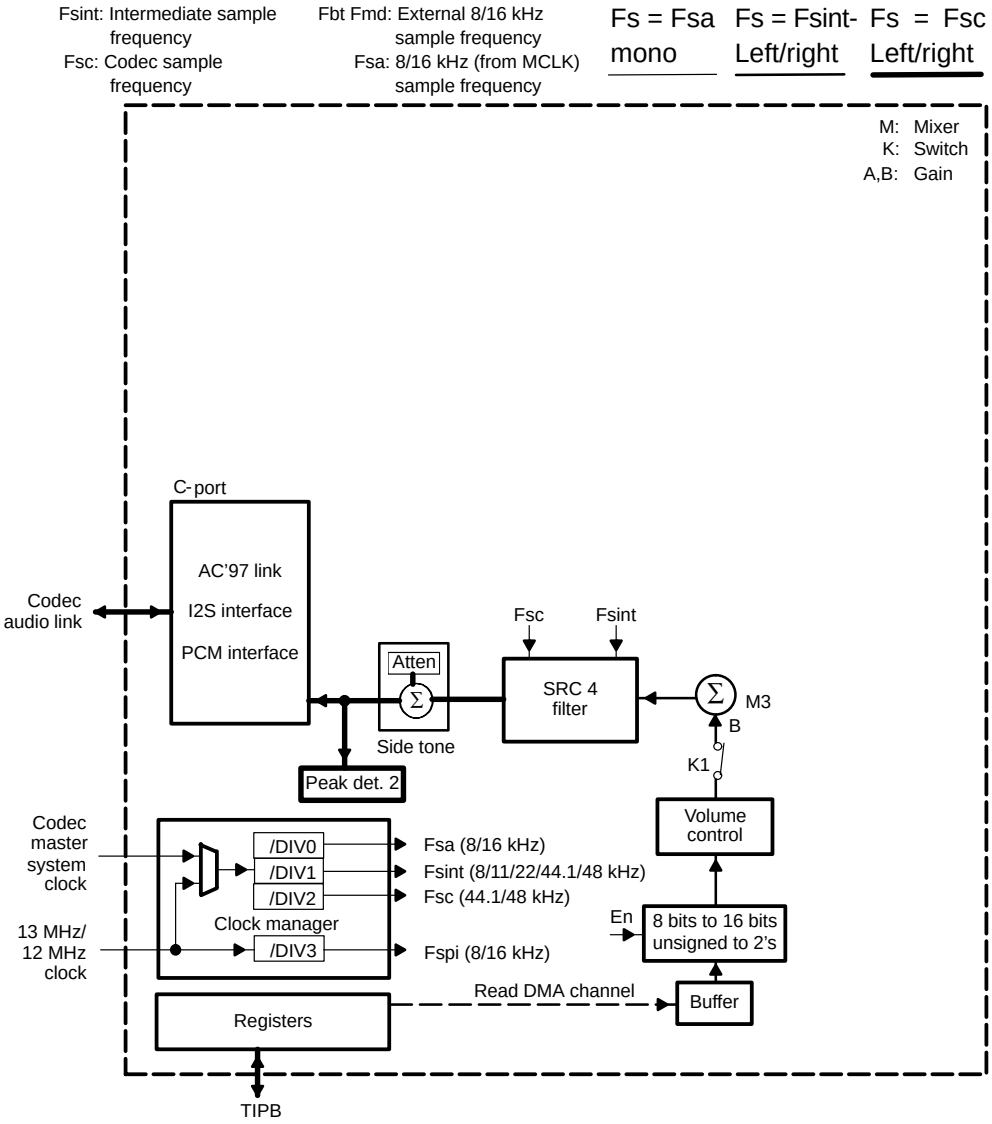
Audio mixer switches configuration: AMSCFR = 0x0010

Table 15–86. AMSCFR Configuration to Record a Message

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				K12	K11	K10	K9	K8	K7	K6	K5	K4	K3	K2	K1
Value	Reserved				0	0	0	0	0	0	0	1	0	0	0	0

15.25 Application: Listen to Music

Figure 15–33. Listening to Music



This application uses the codec port and the DMA.

Listen to music: DMA sends the voice recorded in memory to the codec port.

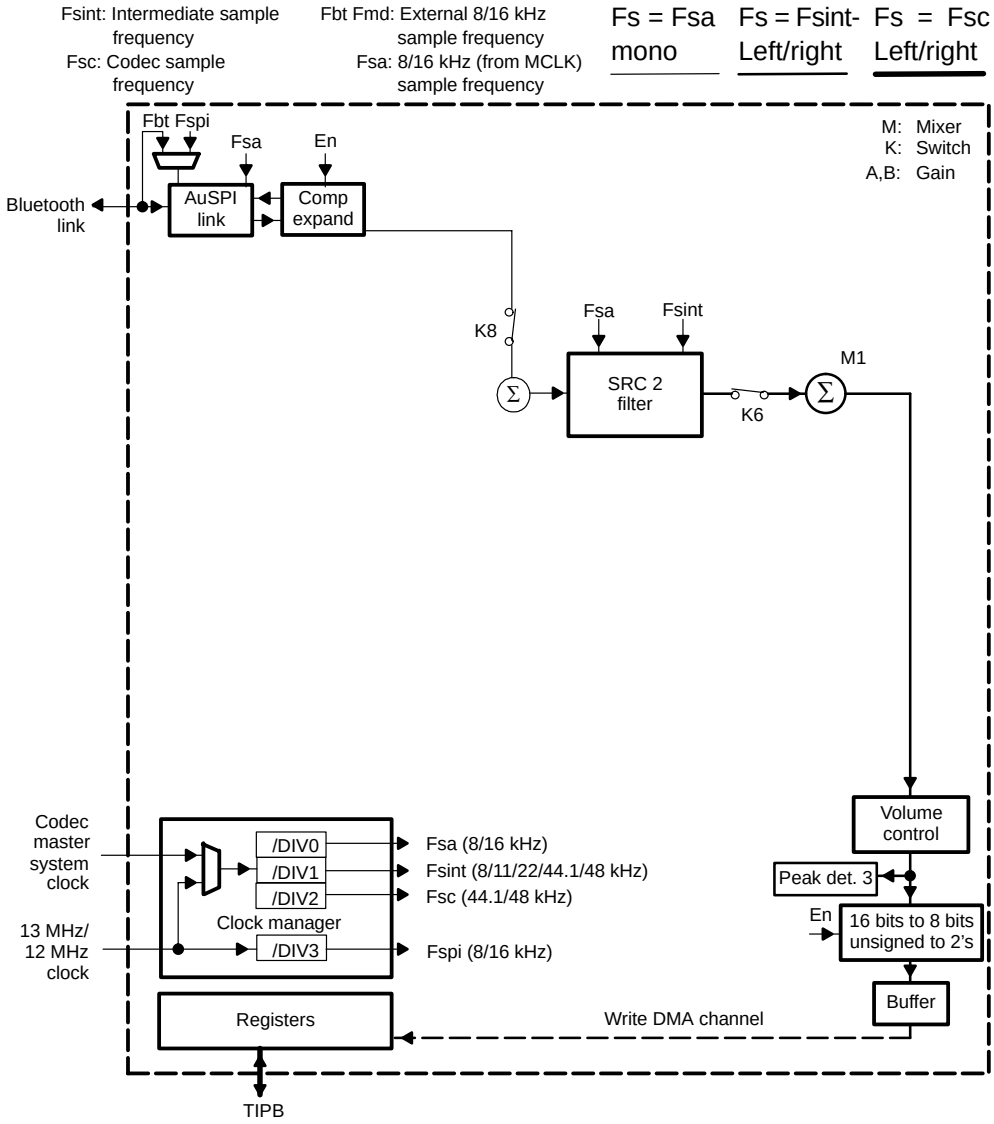
Audio mixer switches configuration: AMSCFR = 0x0001

Table 15–87. AMSCFR Configuration for Music Listening

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				K12	K11	K10	K9	K8	K7	K6	K5	K4	K3	K2	K1
Value	Reserved				0	0	0	0	0	0	0	0	0	0	0	1

15.26 Application: Record a Message From Headset

Figure 15–34. Record a Message From Headset



This application uses the Bluetooth AuSPI port and the write DMA channel.

Record a message from headset: The write DMA channel sends to memory the voice coming from the Bluetooth AuSPI port.

The codec port is not used.

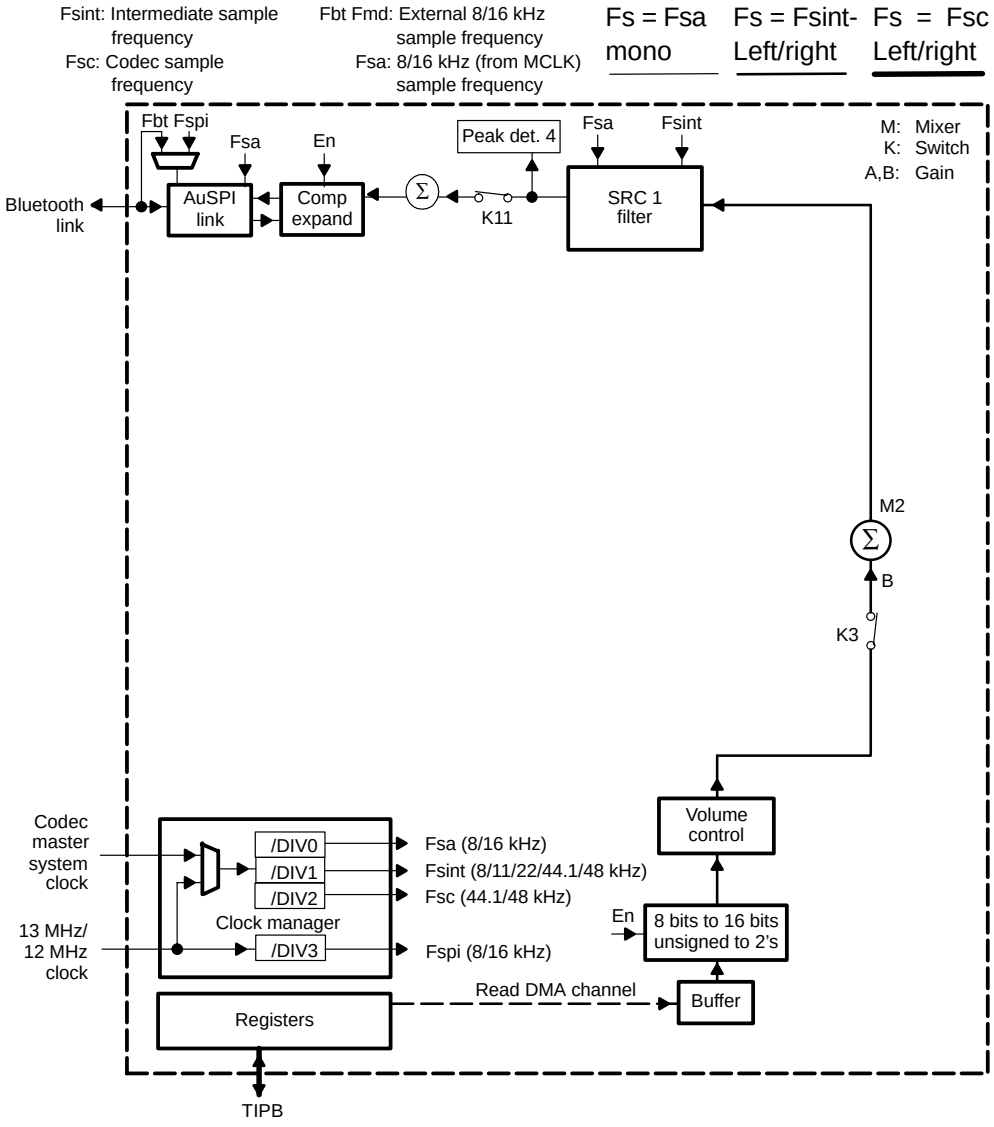
Audio mixer switches configuration: AMSCFR = 0x00C0

Table 15–88. AMSCFR Configuration to Record a Message From Headset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				K12	K11	K10	K9	K8	K7	K6	K5	K4	K3	K2	K1
Value	Reserved				0	0	0	0	1	0	1	0	0	0	0	0

15.27 Application: Listen to Music With Headset

Figure 15–35. Application: Listening to Music With Headset



This application uses the Bluetooth AuSPI port and the read DMA channel.

Play a system sound to headset: The DMA sends the voice recorded in memory only to the Bluetooth AuSPI port.

The codec port is not used.

Audio mixer switches configuration: AMSCFR = 0x0404

Table 15–89. AMSCFR Configuration to Listen to Music With Headset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				K12	K11	K10	K9	K8	K7	K6	K5	K4	K3	K2	K1
Value	Reserved				0	1	0	0	0	0	0	0	0	1	0	0

15.28 Application: Display Wave File Stored in Flash Memory

The wave file is stored at address 0x0400:0000 in flash memory and its size is 10M bytes (→ 0x049F:FFFF).

The wave file is read in flash memory by EAC DMA and sent to the I2S device for display.

15.28.1 Description

This application requires a single transfer of 4K bytes to a 16-bit TIPB EAC in synchronized mode (by DMA_REQ)**.** The TIPB peripheral (EAC) has an internal FIFO and works by DMA_REQ. When the FIFO is empty, the TIPB peripheral performs a DMA_REQ. The DMA controller fills the peripheral TIPB FIFO; once the FIFO is full, the peripheral stops the transfer (temporarily) by asserting the NEND_DMA signal. When the FIFO is again empty, the peripheral performs another DMA_REQ.

A write to the enable bit starts the read from the source (flash memory). When a DMA_REQ occurs, the write to the destination (EAC audio DMA read data register) can be done.

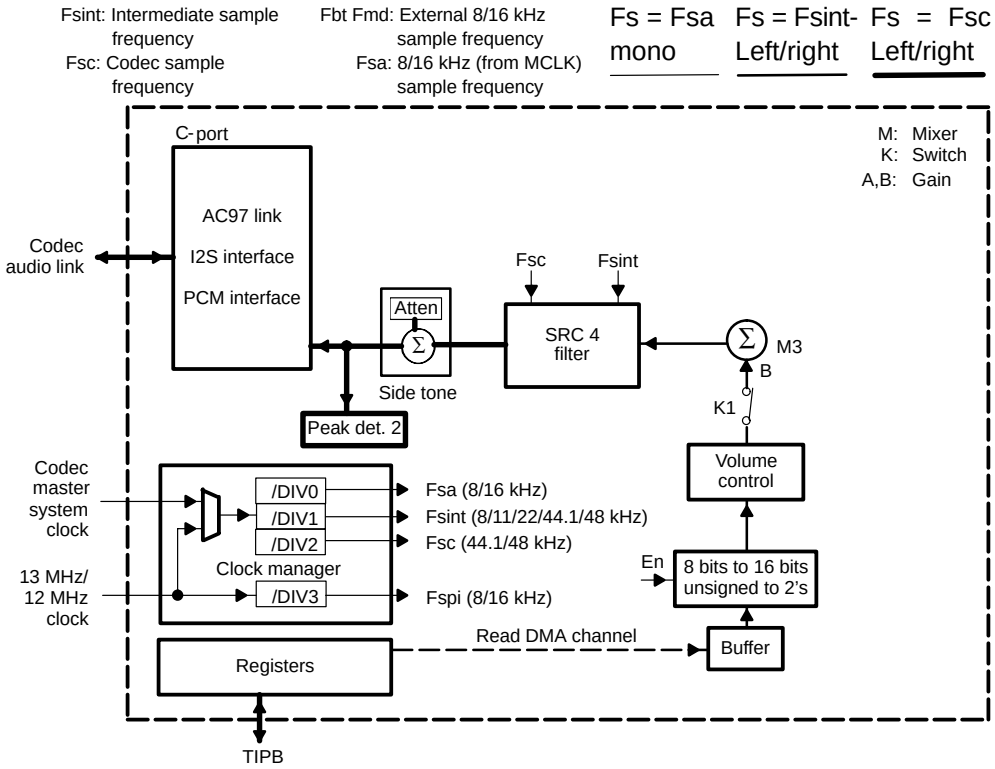
The transfer is performed and an interruption is generated after all bytes are transferred.

Read the DMA_CCR register to know the source of the interruption. If DMA_CCR(17) = 1, an end block interrupt occurs.

Reset DMA_CCR(17) for the next transfer.

15.28.2 EAC Configuration

Figure 15–36. Application: Display Wave File Stored in Flash Memory



15.28.2.1 EAC Audio Mixer Switches Configuration

Audio mixer switches configuration: AMSCFR = 0x0001 (only K1 closed).

15.28.2.2 EAC Codec Port Configuration

- CPCFR1 (codec port configuration register 1): 0X0C
  - I2S mode: MODE = 0X4
  - Two time slots per audio frame: MTSL = 0X1
- CPCFR2 (codec port configuration register 2): 0XED
  - 32 serial clock cycles for time slot 0: TSOL = 0X3
  - 32 data bits per audio time slot: BPTSL = 0X5
  - 32 serial clock (BCLK) cycles for all time slots, except time slot 0: TSLL = 0X5
- CPCFR3 (codec port configuration register 3): 0XE8
  - One BIT_CLK cycle delay of the serial data output and input signals in reference to the leading edge of the SYNC signal: DDLY = 0x1
  - Data serial output enabled during the audio frame is not valid: TRSEN = 0X1.
  - The CSYNC signal is generated with the negative edge of the CSCLK signal. Also, the CDATO signal is generated with the negative edge of

the CSCLK signal, and the CDATI signal is sampled with the positive edge of the CSCLK signal: CLKBP = 0X1.

- The polarity of the codec port interface frame sync (CSYNC) output signal is active low: CSYNCP = 0x0.
- The length of the codec port interface frame sync (CSYNC) output signal is the same number of CSCLK cycles as time slot 0: CSYNCL = 0X1.
- The direction of the codec port interface serial clock (CP_SCLK) signal is an output of the device: CSCLKD = 0x0.
- The direction of the codec port interface frame sync (CP_SYNC) signal is an output from the device: CSYNCD = 0.

❑ CPCFR4 (codec port configuration register 4): 0x00

- No address/data time slot: ATSL = 0X0
- The source of the clock signal to be used for the divide-by-B circuit is the codec master system clock: CLKS = 0 (deleted).
- No divide: DIVB = 0

15.28.2.3 EAC DMA Configuration

Audio file play operation: Enable

DMAREN = 0x1

15.28.2.4 EAC Audio Mixer Volume Configuration

Table 15–90. Audio Mixer 3 Volume Configuration

Mixer	Register	Name	Bits	Function	Reset Value
3	AM3VCTR	MUTE	15	Mute (1) or not (0) on mixer 1 output	0x0
		GINB	14:8	Gain on input B of mixer 1	0x67
		Reserved	7	Reserved	
		GINA	6:0	Gain on input A of mixer 1	0x67

Each mixer has a correspondent audio mixer volume control register. This register controls the volume of each mixer input:

For audio coming from or going to DMA read or write channels, the left and right data volumes are controlled by the gain configured in AM3VCTR register (GINA and GINB).

All volume control is implemented in 0.5-dB steps by all the controls.

Gain level is given by:

GAIN = 12 dB – 0.5 * (DATA – 255) dB

15.28.2.5 EAC Audio Master Volume Configuration

Table 15–91. DMA Channel Volume Configuration

Register	Name	Bits	Function	Reset Value
AMVCTR	GWO	15:8	Gain on write DMA channel GAIN = 12 – 0.5 * (GWO – 255) dB	0xE7
	GRO	7:0	Gain on read DMA channel GAIN = 12 – 0.5 * (GRO – 255) dB	0xE7

Gain level on each DMA channel is configured in the audio master volume control register (AMVCTR):

15.28.3 DMA Controller Configuration

To define a transfer, it is necessary to configure:

- A source port
- A destination port
- A source start address
- A destination start address
- A byte count (the amount of data that must be transferred)
- The type of transfer (transfer width, interrupt generation mode, autoinitialization mode, etc.)

Note:

The following DMA configuration is for the OMAP710. It is also valid for the OMAP730 if the bit OMAP_31_MAPPING_DISABLE = DMA_GSCR[3] = 0. This bit enables/disables the DMA register mapping of the OMAP710.

15.28.3.1 DMA Source Address Register (Offset Address 0x10 * n + 04)

The DMA source is flash memory in EMIF. The wave file is stored at address 0x0400:0000 in flash memory and its size is 10M bytes (→ 0x049F:FFFF).

The DMA source address register is a 31-bit register that contains the source address for the next DMA transfer.

Memory space is 31-bit addressable words (2G bytes), but only the 16 LSBs are incremented during the transfer. The 15 MSBs of the address remain constant during the transfer.

Table 15–92. DMA_SRC_ADDn = 0x0400 0000

Bit 30:0	SRC_ADD = 0x400 0000
Bit 31	Unused

15.28.3.2 DMA Destination Address Register (Offset Address 0x10 * n + 08)

The DMA destination is EAC ADRDR on the TIPB bus.

The DMA destination address register is a 31-bit register that contains the destination address for the next DMA transfer.

Memory space is 31-bit addressable words (2G bytes), but only the 16 LSBs are incremented during the transfer. The 15 MSBs of the address remain constant during the transfer.

Table 15–93. DMA_DST_ADDn = 0x0016 0050

EAC	CS = 22	Address FFFB:B000→ FFFB:B7FF 2K bytes
Bit 30:0	DST_ADD = 0x016 0050	DST_ADD[20:16] = CS = 0x16 DST_ADD[10:0] = 0x50 (ADRDR address)
Bit 31	Unused	

15.28.3.3 DMA Block Count Register (Offset Address 0x10 * n + 0xC)

The DMA block count register is a 32-bit register that contains the number of words to transfer (the first 16 bits). The 16 MSBs indicate the next position in the memory mapping where the next data can be written.

All the DMA transfers are performed within pages. For a transfer, the maximum size allowed is 65535 bytes (64K bytes).

For EMIF, the page number is coded on 11 bits.

Table 15–94. DMA_BKCn

Bit 15:0	BLK_COUNT = 0x07FF	Block count, transfer length in bytes
Bit 31:16	NEXT_WRITE_ADD	Next write address following last write

15.28.3.4 DMA Channel Control Register (Offset Address 0x10 * n + 00)

The DMA channel control register is a 25-bit register that controls the DMA operation.

Bits 0 to 11 are general channel configuration parameters.

Bits 12 to 19 are used to define interruption parameters.

Table 15–95. DMA_CCRn

Bit	Setting	Description
Bit 0	PRIO = 0	Channel has low priority
Bit 2:1	SRC_PORT = 01	EMIF
Bit 4:3	DST_PORT = 11	TIPB
Bit 6:5	TRANSFER_WIDTH = 01	16 bits
Bit 7	SRC_ADD_MODE = 1	Incremental
Bit 8	DST_ADD_MODE = 0	Constant

Table 15–95. DMA_CCRn (Continued)

Bit	Setting	Description
Bit 9	AUTOINIT = 0	Autoinit disable
Bit 10	SYNCHRO = 1	Synchronization active
Bit 11	ENABLE = 1	DMA channel enable
Bit 12	BUS_ERROR_IE = 1	Bus error interrupt
Bit 13	BLOCK_TR_IE = 1	End block interrupt
Bit 14	HALF_BK_IE = 0	No half block interrupt
Bit 15	DROP_IE = 1	Event drop interrupt
Bit 16	BUS_ERROR_COND	
Bit 17	BLOCK_TR_COND	
Bit 18	HALF_BK_COND	
Bit 19	DROP_COND	
Bit 23:20	FIFO_BYTES_LEFT	
Bit 24	FIFO_STATUS	

15.29 Memory-Mapped Register Descriptions

Register	Offset	Register	Offset
CPCFR1	0x00	CPCFR2	0x02
CPCFR1	0x04	CPCFR4	0x06
CPTCTL	0x08	CPTTADR	0x0A
CPTDATL	0x0C	CPTDATH	0x0E
CPTVSLL	0x10	CPTVSLH	0x12
MPCTR	0x20	MPPCCFR	0X22
MPACCFR	0x24	MPADLTR	0x26
MPADMTR	0x28	MPADLRR	0x2A
MPADMRR	0x2C	BPCTR	0x30
BPMCCFR	0X32	BPACCFR	0x34
BPADLTR	0x36	BPADMTR	0x38
BPADLRR	0x3A	BPADMRR	0x3C
AMSCFR	0x40	AMVCTR	0x42
AM1VCTR	0x44	AM3VCTR	0x46
AM3VCTR	0x48	ASTCTR	0x4A
APD1LCR	0x4C	APD1RCR	0x4E
APD2LCR	0x50	APD2RCT	0x52
APD3LCR	0x54	APD3RCR	0x56
APD4R	0x58	ADWDR	0x5A
ADRDR	0x5C	AGCFR	0x5E
AGCTR	0x60	AGCFR2	0x62

Table 15–96. Codec Port Configuration Register 1 (CPCFR1)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Reserved	R	0X00
7:3	MTSL	Number of time slots per audio frame 00000b: 1 time slot per frame 00001b: 2 time slots per frame (default: I2S) 11111b: 32 time slots per frame	R/W	0X1
2:0	MODE	Codec-port interface mode 000 b: General-purpose mode 001 b: PCM mode 010 b: AC 97 mode 100 b: I2S mode (Default) Others reserved	R/W	0X4

*Table 15–97. Codec Port Configuration Register 2 (CPCFR2) **

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Reserved	R	0X00
7:6	TSLOL	Time slots 0 length in number of serial clock (CLK_BIT) cycles 00b: Same as TSLL (default: I2S) 01b: 8 CLK_BIT cycles 10b: 16 CLK_BIT cycles 11b: 32 CLK_BIT cycles	R/W	0X0
5:3	BPTSL	Number of data bits per audio time slot Note: This value in not used for the secondary communication address and data time slots. 000: 8 data bits per time slot 001: 16 data bits per time slot (Default: I2S) 010: 18 data bits per time slot 011: 20 data bits per time slot 100: 24 data bits per time slot 101: 32 data bits per time slot 110: Reserved 111: Reserved	R/W	0X01
2:0	TSLL	Time slot length (all except time slot 0) in number of serial clock (CLK_BIT) cycles 000b: 8 CLK_BIT cycles 001b: 16 CLK_BIT cycles (Default: I2S) 010b: 18 CLK_BIT cycles 011b: 20 CLK_BIT cycles 100b: 24 CLK_BIT cycles 101b: 32 CLK_BIT cycles 110b: Reserved 111b: Reserved	R/W	0X1

Table 15–98. Codec Port Interface Configuration Register 3 (CPCFR3)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Reserved	R	0X00
7	DDLY	Data delay: Data bits start according to SYNC signal leading edge. 0: No delay 1: One cycle delay (default: I2S)	R/W	0X1
6	TRSEN	3-state enable: Data serial output state during invalid audio frames 0: 3-state 1: Data serial output is enabled.	R/W	0X0
5	CLKBP	CLOCK POLARITY 0: Rising edge (CP_SYNC and serial data are generated on serial clock rising edge; serial data input is sampled on falling edge) 1: Falling edge (default: I2S)	R/W	0X1

Table 15–98. Codec Port Interface Configuration Register 3 (CPCFR3) (Continued)

Bit	Name	Function	R/W	Reset Value
4	CSYNCP	CP_SYNC (synchro) polarity 0: Active low (default: I2S, time slot 0 is left time slot) 1 = Active high	R/W	0X0
3	CSYNCL	CSYNC length 0: Synchro is a one serial-clock cycle pulse. 1: Synchro length is equal to time slot 0 length (default: I2S)	R/W	0X1
2	RESERVED2	Reserved (byte order)	R/W	0X0
1	CSCLKD	CP_SCLK port (serial clock) direction 0: CP_SCLK is an output. 1: CP_SCLK is an input.	R/W	0X1
0	CSYNCD	CP_SYNC (synchro) direction 0: CP_SYNC is an output. 1: CP_SYNC is an input.	R/W	0X1

Table 15–99. Codec Port Interface Configuration Register 4 (CPCFR4)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Reserved	R	0X00
7:4	ATSL	Audio time slots for secondary communication address and data value 0000: I2S or PCM mode 0001: AC97 mode (time slot 1 for address, time slot 2 for data)	R/W	0X00
3	CLKS	Clock source (reserved) C-port input clock is selected according to AUD_CKSRC programming of AGCFR register.	R/W	0X0
2:0	DIVB	CP_SCLK divider value Input clock division ratio to generate CP_SCLK 000: No division 001: Divide by 2 010: Divide by 3 011: Divide by 4 (default: I2S) 100: Divide by 5 101: Divide by 6 110: Divide by 7 111: Divide by 8 (Example: I2S 11.289MHz input clock, 44.1 kHz frame synchro and 32 bits per frame → DIVB = 11.289 MHz/(32*44.1 kHz) = 4)	R/W	0X3

Table 15–100. Codec Port Interface Control and Status Register (CPTCTL)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Reserved	R	0x00
7	RXF	Receive data register full Set to a 1 when a new data value has been received into the receive data register from the codec device. Cleared by writing a 1.	R/W	0X0
6	RXIE	Receive interrupt enable 0: Interrupt generation disabled 1: Interrupt generated upon receive data register full.	R/W	0X0
5	TXE	Transmit data register empty Set to 1 when data value in the transmit data register has been sent to the codec device. Cleared by writing a 1.	R/W	0x1
4	TXIE	Transmit interrupt enable 0: Interrupt generation disabled 1: Interrupt generated upon transmit data register empty.	R/W	0X0
3	CPEN	C_PORT ENABLE The C-port enable bit is used to control the C-port interface. 0: Disabled 1: Enabled (access to config. registers is disabled.)	R/W	0X0
2:1	CID	AC97 only: codec ID Selection between primary and secondary codec devices 0: Primary codec	R/W	0X00
0	CRST	Codec reset 0: CP_NRST output is low. 1: CP_NRST output is high.	R/W	0X0

Table 15–101. Codec Port Interface Address Register (CPTTADR)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Reserved	R	0X00
7	RW	AC97 only: 0: Command/status write operation 1: Command/status read operation	R/W	0X0
6:0	A	AC97 only Command/status address value for codec device control/status register to be accessed	R/W	0X00

Table 15–102. Codec Port Interface Data Register (Low Byte) (CPTDATL)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Reserved	R	0X00
7:0	D	AC97 only: Write command data value (LSB part) to be transmitted to codec device Read status data value (LSB part) received from codec device	R/W	0X00

Table 15–103. Codec Port Interface Data Register (High Byte) (CPTDATH)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Reserved	R	0X00
7:0	D	AC97 only: Write command data value (MSB part) to be transmitted to codec device Read status data value (MSB part) received from codec device	R/W	0X00

Table 15–104. Codec Port Interface Valid Time Slots Register (Low Byte) (CPTVSLL)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Reserved	R	0X00
7:3	VTSL	AC97 only: Valid time slots Bits 7 to 3 correspond to time slots 8 to 12 in slot 0 tag. Set to 1 to indicate audio valid time slot.	R/W	0X00
2:0	RESERVED2	Reserved	R	0x0

Table 15–105. Codec Port Interface Valid Time Slots Register (High Byte) (CPTVSLH)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Reserved	R	0X00
7	VF	AC97 only: Valid frame Valid frame bit for slot 0 tag 1: Indicates that the audio frame contains at least one valid time slot	R/W	0X0
6	CRDY1	AC97 only: Primary codec ready 1: Codec ready	R	0X00
5	CRDY2	AC97 only: Secondary codec ready 1: Codec ready	R	0x0
4:0	VTSL	AC97 only: Valid time slots Bits 4 to 0 correspond to time slots 3 to 7 in Slot 0 tag. Set to 1 to indicate audio valid time slot.	R/W	0X00

Table 15–106. Modem Port Control Register (MPCTR)

Bit	Name	Function	R/W	Reset Value
15	ACTRIS	Auxiliary channel transmit register interrupt status 1: Interrupt generated Cleared by writing a one	R/W	0x0
14	ACTRIEN	Auxiliary channel transmit register interrupt 0: Interrupt generation disabled 1: Interrupt generated upon auxiliary channel transmit buffer empty condition	R/W	0x0
13	ACRRIS	Auxiliary channel receive register interrupt status 1: Interrupt generated Cleared by writing a one	R/W	0x0
12	ACRRIEN	Auxiliary channel receive register interrupt 0: Interrupt generation disabled 1: Interrupt generated upon auxiliary channel receive buffer-full condition	R/W	0x0
11	ACRRS	Auxiliary channel receive register status 0: Register empty 1: Register full (data waiting to be read)	R	0x0
10	ACTRF	Auxiliary channel transmit register status 0: Register empty 1: Register full (data waiting for transmit)	R	0x0
9	AC_DTS	Data start for auxiliary channel Set to 1 to request data transmit on auxiliary channel Cleared by hardware upon data transmit	R/W	0X0
8	AC_EN	Auxiliary channel enable 0: Disabled 1: Enabled	R/W	0X0
7	MC_EN	Main channel enable 0: Disabled 1: Enabled	R/W	0X0
6:4	PRE_AC	Prescale clock divisor for main channel 000b: 1,001b = 2, 010b: 4,011b = 8, 100b: 16.	R/W	0X0
3:1	PRE_MC	Prescale clock divisor for main channel 000b: 1,001b = 2, 010b: 4,011b = 8, 100b: 16.	R/W	0X0
0	CKEN	AuSPI clock enable 0: Clocks are shut off. 1: Clocks are running.	R/W	0X0

Table 15–107. Modem Port Main Channel Configuration Register (MPMCCFR)

Bit	Name	Function	R/W	Reset Value
15:13	RESERVED	Reserved	R	0x0
14	DFILL_MC	Justified data filling value Missing data bits take this value (for TX when WS_MC > internal data length, for RX when WS_MC < internal data length) 0: Filled with zeros 1: Filled with ones	R/W	0x0
13	DJUST_MC	Data justification (accurate only if WS_MC is different from internal audio data length) 0: Data justify left (MSB alignment) 1: Data justify right (LSB alignment)	R/W	0x0
12:11	COMPAND	Compand mode for main channel transmit register 00: Disabled 01: μ-law 10: A-law 11: Reserved	R/W	0x0
10:9	EXPAND	Expand mode for main channel receive register 00: Disabled 01: μ-law 10: A-law 11: Reserved	R/W	0x0
8	MCM	Main channel mode 0: Slave mode 1: Master mode	R/W	0X00
7	FSP_MC	Frame synchro polarity for main channel 0: Falling 1: Rising	R/W	0X0
6	FSBL_MC	Frame synchro level for main channel 0: Low level 1: High level	R/W	0X0
5	CPB_MC	Clock polarity for main channel 0: Falling 1:Rising	R/W	0X0
4:0	WS_MC	The Word size bits are set to define the bit number of the transmission for main channel. 00000b: 1 bit 00001b: 2 bits ... ... 11111b: 32 bits	R/W	0X0

Table 15–108. Modem Port Auxiliary Channel Configuration Register (MPACCFR)

Bit	Name	Function	R/W	Reset Value
15:9	RESERVED	Reserved	R	0x0
8	CST_AC	Chip-select trigger for main channel 0: Level trigger 1: Edge trigger	R/W	0x0
7	CSP_AC	Chip-select polarity for auxiliary channel 0: Falling edge 1: Rising edge	R/W	0x0
6	CSBL_AC	Chip-select level for auxiliary channel 0: Low level 1: High level	R/W	0x0
5	CPB_AC	Clock polarity for auxiliary channel 0: Falling 1: Rising	R/W	0x0
4:0	WS_AC	Word size: Number of transmission bits for auxiliary channel 00000b: 1 bit 00001b: 2 bits ... ... 11111b: 32 bits	R/W	0x0

Table 15–109. Modem Port Auxiliary Data LSB Trasnsmit Register (MPADLTR)

Bit	Name	Function	R/W	Reset Value
15:0	DX_LSB	LSB part of data to be transmitted on auxiliary channel	R/W	0X0000

Table 15–110. Modem Port Auxiliary Data MSB Trasnsmit Register (MPADMTR)

Bit	Name	Function	R/W	Reset Value
15:0	DX_MSB	MSB part of data to be transmitted on auxiliary channel	R/W	0x0

Table 15–111. Modem Port Auxiliary Data LSB Receive Register (MPADLRR)

Bit	Name	Function	R/W	Reset Value
15:0	DR_LSB	LSB part of data received from auxiliary channel	R/W	0X0000

Table 15–112. Modem Port Auxiliary Data MSB Receive Register (MPADMRR)

Bit	Name	Function	R/W	Reset Value
15:0	RX_MSB	MSB part of data received from auxiliary channel	R	0x0

Table 15–113. Bluetooth Port Control Register (BPCTR)

Bit	Name	Function	R/W	Reset Value
15:12	RESERVED	Reserved	R	0x0
11	ACRRS	Auxiliary channel receive register status 0: Register empty 1: Register full (data waiting to be read)	R	0x0
10	ACTRSV	Auxiliary channel transmit register status 0: Register empty 1: Register full (data waiting for transmit)	R	0x0
9	AC_DTS	Data start for auxiliary channel Set to 1 to request data transmit on auxiliary channel Cleared by hardware upon data transmit	R/W	0X0
8	AC_EN	Auxiliary channel enable 0: Disabled 1: Enabled	R/W	0X0
7	MC_EN	Main channel enable 0: Disabled 1: Enabled	R/W	0X0
6:4	PRE_AC	Prescale clock divisor for main channel 000b: 1 001b: 2 010b: 4 011b: 8 100b: 16	R/W	0X0
3:1	PRE_MC	Prescale clock divisor for main channel 000b: 1, 001b: 2 010b: 4 011b: 8 100b: 16	R/W	0X0
0	CKEN	AuSPI clock enable 0: Clocks are shut off. 1: Clocks are running.	R/W	0X0

Table 15–114. Bluetooth Port Main Channel Configuration Register (BPMCCFR)

Bit	Name	Function	R/W	Reset Value
15:13	RESERVED	Reserved	R	0x0
14	DFILL_MC	Justified data filling value Missing data bits take this value (for TX when WS_MC > internal data length for RX when WS_MC < internal data length) 0: Filled with zeros 1: Filled with ones	R/W	0x0

Table 15–114. Bluetooth Port Main Channel Configuration Register (BPMCCFR) (Continued)

Bit	Name	Function	R/W	Reset Value
13	DJUST_MC	Data justification (accurate only if WS_MC is different from internal audio data length) 0: Data justified left (MSB alignment) 1: Data justified right (LSB alignment)	R/W	0x0
12:11	COMPAND	Compand mode for main channel transmit register 00: Disabled 01: μ-law 10: A-law 11: Reserved	R/W	0x0
10:9	EXPAND	Expand mode for main channel receive register 00: Disabled 01: μ-law 10: A-law 11: Reserved	R/W	0x0
8	MCM	Main channel mode 0: Slave mode 1: Master mode	R/W	0X00
7	FSP_MC	Frame synchro polarity for main channel 0: Falling 1: Rising	R/W	0X0
6	FSBL_MC	Frame synchro level for main channel 0: Low level 1: High level	R/W	0X0
5	CPB_MC	Clock polarity for main channel 0: Falling 1: Rising	R/W	0X0
4:0	WS_MC	The word size (in bits) is set to define the number of bits for transmission by the main channel. 00000b: 1 bit 00001b: 2 bits ... ... 11111b: 32 bits	R/W	0X0

Table 15–115. Bluetooth Port Auxiliary Channel Configuration Register (BPACCFR)

Bit	Name	Function	R/W	Reset Value
15:9	RESERVED	Reserved	R	0x0
8	CST_AC	Chip-select trigger for main channel 0: Level trigger 1: Edge trigger	R/W	0x0

Table 15–115. Bluetooth Port Auxiliary Channel Configuration Register (BPACCFR) (Continued)

Bit	Name	Function	R/W	Reset Value
7	CSP_AC	Chip-select polarity for auxiliary channel 0: Falling edge 1: Rising edge	R/W	0x0
6	CSBL_AC	Chip-select level for auxiliary channel 0: Low level 1: High level	R/W	0x0
5	CPB_AC	Clock polarity for auxiliary channel 0: Falling 1: Rising	R/W	0x0
4:0	WS_AC	Word size bits number of transmission for auxiliary channel 00000b: 1 bit 00001b: 2 bits ... ... 11111b: 32 bits	R/W	0x0

Table 15–116. Bluetooth Port Auxiliary Data LSB Transmit Register (BPADLTR)

Bit	Name	Function	R/W	Reset Value
15:0	DX_LSB	LSB part of data to be transmitted on auxiliary channel	R/W	0X0000

Table 15–117. Bluetooth Port Auxiliary Data MSB Transmit Register (BPADMTR)

Bit	Name	Function	R/W	Reset Value
15:0	DX_MSB	MSB part of data to be transmitted on auxiliary channel	R/W	0x0

Table 15–118. Bluetooth Port Auxiliary Data LSB Receive Register (BPADLRR)

Bit	Name	Function	R/W	Reset Value
15:0	DR_LSB	LSB part of data received from auxiliary channel	R/W	0X0000

Table 15–119. Bluetooth Port Auxiliary Data MSB Receive Register (BPADMRR)

Bit	Name	Function	R/W	Reset Value
15:0	RX_MSB	MSB part of data received from auxiliary channel	R	0x0

Table 15–120. Audio Mixer Switches Configuration Register (AMSCFR)

Bit	Name	Function	R/W	Reset Value
15:12	RESERVED	Reserved	R	0x0
11	K12	0: K12 switch is open. 1: K12 switch is closed.	R/W	0x0

Table 15–120. Audio Mixer Switches Configuration Register (AMSCFR) (Continued)

Bit	Name	Function	R/W	Reset Value
10	K11	0: K11 switch is open. 1: K11 switch is closed.	R/W	0x0
9	K10	0: K10 switch is open. 1: K10 switch is closed.	R/W	0x0
8	K9	0: K9 switch is open. 1: K9 switch is closed.	R/W	0x0
7	K8	0: K8 switch is open. 1: K8 switch is closed.	R/W	0x0
6	K7	0: K7 switch is open. 1: K7 switch is closed.	R/W	0x0
5	K6	0: K6 switch is open. 1: K6 switch is closed.	R/W	0x0
4	K5	0: K5 switch is open. 1: K5 switch is closed.	R/W	0x0
3	K4	0: K4 switch is open. 1: K4 switch is closed.	R/W	0x0
2	K3	0: K3 switch is open. 1: K3 switch is closed.	R/W	0x0
1	K2	0: K2 switch is open. 1: K2 switch is closed.	R/W	0x0
0	K1	0: K1 switch is open. 1: K1 switch is closed.	R/W	0x0

Table 15–121. Audio Master Volume Control Register (AMVCTR)

Bit	Name	Function	R/W	Reset Value
15:8	GWO	Gain on write DMA operation Gain: 12 – 0.5*(255 – GWO) dB	R/W	0xE7
7:0	GRO	Gain on read DMA operation Gain:12 – 0.5*(255 – GRO) dB	R/W	0xE7

Table 15–122. Audio Mixer 1 Volume Control Register (AM1VCTR)

Bit	Name	Function	R/W	Reset Value
15	MUTE	0: No mute on mixer output 1: Mute on mixer output	R/W	0x0
14:8	GINB	Gain on input B Gain: 12 – 0.5*(127 – GINB) dB	R/W	0x67

Table 15–122. Audio Mixer 1 Volume Control Register (AM1VCTR) (Continued)

Bit	Name	Function	R/W	Reset Value
7	RESERVED	Reserved	R	0x0
6:0	GINA	Gain on input A Gain:12 – 0.5*(127 – GINA) dB	R/W	0x67

Table 15–123. Audio Mixer 2 Volume Control Register (AM2VCTR)

Bit	Name	Function	R/W	Reset Value
15	MUTE	0: No mute on mixer output 1: Mute on mixer output	R/W	0x0
14:8	GINB	Gain on input B Gain: 12 – 0.5*(127 – GINB) dB	R/W	0x67
7	RESERVED	Reserved	R	0x0
6:0	GINA	Gain on input A Gain: 12 – 0.5*(127 – GINA) dB	R/W	0x67

Table 15–124. Audio Mixer 3 Volume Control Register (AM3VCTR)

Bit	Name	Function	R/W	Reset Value
15	MUTE	0: No mute on mixer output 1: Mute on mixer output	R/W	0x0
14:8	GINB	Gain on input B GAIN: 12 – 0.5*(127 – GINB) dB	R/W	0x67
7	RESERVED	Reserved	R	0x0
6:0	GINA	Gain on input A Gain: 12 – 0.5*(127 – GINA) dB	R/W	0x67

Table 15–125. Audio Sidetone Control Register (ASTCTR)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Reserved	R/W	0x0
7:1	ATT	Attenuation of the sidetone Gain = 0.5*(127 – ATT) dB	R/W	0x67
0	ATTEN	0: Sidetone disabled 1: Sidetone enabled	R/W	0x0

Table 15–126. Audio Peak Detector 1 Left Channel Register (APD1LCR)

Bit	Name	Function	R/W	Reset Value
15:0	PEAK	Peak value on the channel Cleared upon reading	R	0x0

Table 15–127. Audio Peak Detector 1 Right Channel Register (APD1RCR)

Bit	Name	Function	R/W	Reset Value
15:0	PEAK	Peak value on the channel Cleared upon reading	R	0x0

Table 15–128. Audio Peak Detector 2 Left Channel Register (APD2LCR)

Bit	Name	Function	R/W	Reset Value
15:0	PEAK	Peak value on the channel Cleared upon reading	R	0x0

Table 15–129. Audio Peak Detector 2 Right Channel Register (APD2RCR)

Bit	Name	Function	R/W	Reset Value
15:0	PEAK	Peak value on the channel Cleared upon reading	R	0x0

Table 15–130. Audio Peak Detector 3 Left Channel Register (APD3LCR)

Bit	Name	Function	R/W	Reset Value
15:0	PEAK	Peak value on the channel Cleared upon reading	R	0x0

Table 15–131. Audio Peak Detector 3 Right Channel Register (APD3RCR)

Bit	Name	Function	R/W	Reset Value
15:0	PEAK	Peak value on the channel Cleared upon reading	R	0x0

Table 15–132. Audio Peak Detector 4 Register (APD4R)

Bit	Name	Function	R/W	Reset Value
15:0	PEAK	Peak value on the channel Cleared upon reading	R	0x0

Table 15–133. Audio DMA Write Data Register (ADWDR)

Bit	Name	Function	R/W	Reset Value
15:0	WR_DATA	DMA write operation data set by hardware	R	0x0

Table 15–134. Audio DMA Read Data Register (ADRDR)

Bit	Name	Function	R/W	Reset Value
15:0	DATA_RD	DMA read operation data. Data in this register is read by hardware.	W	0x0

Table 15–135. Audio Global Configuration Register (AGCFR)

Bit	Name	Function	R/W	Reset Value
15:11	RESERVED	Reserved	R/W	0x0
10	MN_ST	Mono/stereo audio file	R/W	0x1
9	B8_16	8-bit/16-bit audio file 0: 8 bits 1: 16 bits	R/W	0x1
8	LI_BI	Audio file endianism 0: Little endian 1: Big endian	R/W	0x0
7:6	FSINT	Intermediate sample frequency for DMA read and write operations 00: FSINT is 8 kHz. 01: FSINT is 11.025 kHz. 10: FSINT is 22.05 kHz. 11: FSINT is 44.1 kHz.	R/W	0x1
5:4	AUD_CKSRC	Audio processing clock source 00: Codec master clock 01: 12-MHz clock on CLK_12M input 10: 13-MHz clock on CLK_13M input 11: Reserved	R/W	0x0
3	M_CKSRC	Modem Interface clock source 0: 12-MHz clock on CLK_12M input 1: 13-MHz clock on CLK_13M input	R/W	0x1
2	BT_CKSRC	Bluetooth interface clock source 0: 12-MHz clock on CLK_12M input 1: 13-MHz clock on CLK_13M input	R/W	0x0
1	MCLK_OUT	0: MCLK_OUT = MCLK 1: MCLK_OUT = MCLK/2	R/W	0x0
0	MCLK	0: Clock on MCLK input is 11.2896/12.288 MHz. 1: Clock on MCLK input is 22.5792/24.576 MHz (internally used audio clock is MCLK/2).	R/W	0x1

Table 15–136. Audio Global Control Register 2 (AGCTR)

Bit	Name	Function	R/W	Reset Value
15	AUDRD	Audio ready 1: Audio is ready for processing. 0: Audio not ready (either disabled by AUEN or disabled parameters setting changes)	R	0x0
14	AUDRDI	Audio ready interrupt status 0: No interrupt generated 1: Interrupt generated (cleared by writing a 1)	R/W	0x0

Table 15–136. Audio Global Control Register 2 (AGCTR) (Continued)

Bit	Name	Function	R/W	Reset Value
13	AUDRDIEN	Audio ready interrupt enable 0: Interrupt generation disabled 1: Interrupt generated upon audio-ready condition	R/W	0x0
12	DMAREN	Audio files play operation 0: DMA read operation disabled 1: DMA read operation enabled	R/W	0x0
11	DMAWEN	Audio file record operation 0: DMA write operation disabled 1: DMA write operation enabled	R/W	0x0
10:4	RESERVED	Reserved	R	0x0
3	MCLK_EN	Internal MCLK enable 0: MCLK gated internally 1: MCLK enabled	R/W	0x0
2	OSCMCLK_EN	OSCMCLK_EN output for MCLK oscillator control 0: OSCMCLK_EN low (oscillator disabled) 1: OSCMCLK_EN high (oscillator enabled)	R/W	0x0
1	AUDEN	0: Audio processing disabled 1: Audio processing enable Audio processing must be disabled while configuring module	R/W	0x0
0	EACPWD	0: Normal EAC operation 1: EAC in power-down mode	R/W	0x0

Table 15–137. Audio Global Configuration Register 2 (AGCFR2)

Bit	Name	Function	R/W	Reset Value
15:9	RESERVED	Reserved	R	0x0
5	BT_MD_WIDEBAND	Bluetooth and modem AuSPI wide-band mode 0: Wide-band mode disabled (reset value for EAC rev. 1 Backward compatibility: Only 8-kHz frame frequency support) 1: Wide-band mode enabled (both BT and MD AuSPI Operates on 16-kHz frame frequency)	R/W	0x0
4	MCLK_I2S_N11M_12M	MCLK frequency indicator for audio operations This indicator is accurate only when: – MCLK is selected for audio operations. (AGCFR/ AUD_CKSRC = 00) – Codec port is not set for AC97. 0: Clock on MCLK is 11.289 MHz (reset value for EAC rev. 1 backward compatibility: I2S requires 11.289 MHz) 1: Clock on MCLK is 12.288 MHz for I2S 48M master mode compatibility.	R/W	0x0

Table 15–137. Audio Global Configuration Register 2 (AGCFR2) (Continued)

Bit	Name	Function	R/W	Reset Value
3	I2S_N44K_48K	Frame sample frequency of I2S codec port 0: Codec frequency = 44.1 kHz (reset value for EAC rev. 1 backward compatibility: Only 44.1 kHz support) 1: Codec frequency = 48 kHz	R/W	0x0
2:0	FSINT2	EAC rev. 2 intermediate sample frequency for DMA read and write operations 111: AGCFR/FSINT are used (reset value. EAC rev. 1 backward compatibility). 000: FSINT is 8 kHz. 001: FSINT is 11.025 kHz . 010: FSINT is 22.05 kHz . 011: FSINT is 44.1 kHz . 100: FSINT is 48 kHz (EAC rev2 new). 101: Reserved 110: Reserved	R/W	0x7

Note: This register exists for EAC version 2 only.

Chapter 16

Multichannel Serial Interface (MCSI)

This chapter describes the multichannel serial interface for the OMAP730 multimedia processor.

Topic **Page**

16.1 Multichannel Serial Interface

The multichannel serial interface (MCSI) has multichannel transmission capability. The MCSI expands the parallel interface of ARM926EJS to connect to external devices such as codecs, DSPs, and GSM system simulators.

The MCSI on the OMAP730 device provides full-duplex transmission and master or slave clock control. All transmission parameters are configurable to cover the maximum number of operating conditions:

- Master or slave clock control (transmission clock and frame-synchronization pulse)
- Programmable transmission clock frequency
- Single-channel or multichannel (x16) frame structure
- Programmable word length: 3 to 16 bits
- Full-duplex transmission
- Programmable frame configuration
  - Continuous or burst transmission
  - Normal or alternate framing
  - Normal or inverted frame polarity
  - Short- or long frame pulse
  - Programmable oversize frame length
  - Programmable frame length
- Programmable interrupt occurrence time (TX and RX)
- Error detection with interrupt generation on wrong frame length
- DMA support for both TX and RX data transfers

16.1.1 Communication Protocol

16.1.1.1 Configuration Parameters

The configuration parameters can be modified only if the MCSI is disabled (CONTROL_REG[0] = 0).

Slave/Master Control

Using the control bit, the interface can be configured in one of two ways:

- In master mode with the transmission clock and the frame-synchronization pulse generated by the interface
- In slave mode with the transmission clock and the frame-synchronization pulse generated from an external device

Control bit:
MAIN_ PARAMETERS _REG(6) = MCSI_MODE
1: Master
0: Slave

Single-Channel/Multichannel

The frame structure can be either single-channel-based (one channel per frame) or multichannel-based with the number of channels fixed at 16.

Control bit:
MAIN_ PARAMETERS _REG(7) = MULTI
1: Multichannel
0: Single-channel

Short/Long Framing

The frame-synchronization pulse duration can be either short, with a pulse duration equal to the bit duration, or long, with a pulse duration equal to the channel duration.

The long frame is active only during transmission on channel 0.

Control bit:
MAIN_ PARAMETERS _REG(8) = FRAME_SIZE
1: Long
0: Short

Normal/Alternate Frame Synchronization

The frame-synchronization pulse position is either normal, with the frame-synchronization pulse starting one bit before channel 0, or it alternates with the frame-synchronization pulse, starting with the first bit of channel 0.

Control bit:
MAIN_ PARAMETERS _REG(9) = FRAME_POSITION
1: Alternate
0: Normal

Continuous/Burst Mode

The frame mode is either continuous, with one frame-synchronization pulse at the first frame, or it bursts with one frame-synchronization pulse at each frame.

Control bit:
MAIN_ PARAMETERS _REG(5) = FRAME_MODE
1: Continuous
0: Burst

Normal/Inverted Clock

The polarity of the clock can be either normal, with writing on positive edge clock and reading on negative edge clock, or inverted, with writing on negative edge clock and reading on positive edge clock.

Control bit:
MAIN_ PARAMETERS _REG(4) = CLOCK_POLARITY
1: Inverted
0: Normal

Normal/Inverted Frame Synchronization

The polarity of the frame-synchronization pulse can be either normal, with a positive pulse, or inverted, with a negative pulse.

Control bit:
MAIN_ PARAMETERS _REG(10) = FRAME_POLARITY
1: Inverted
0: Normal

Channel Used

To enable a channel in multimode, set bit n for the desired channel n.

Word Size

To choose the size of the word, set its size minus one in the main parameters registers.

Control bit:
MAIN_PARAMETERS_REG(3:0) = WORD_ SIZE
(2 <= WORD_ SIZE <= 15)

The MCSI transmits and receives the most significant bit first. For example, if WORD_SIZE equals 11, the upper 12 bits of the TX registers are transmitted. The upper 12 bits of the RX registers contain the received data, and the lower 4 bits are zeros.

Frame Size

To add any overhead bits at the end of each frame, set the number of desired overhead bits in OVER_SIZE_REG.

Control bit:
OVER_CLOCK_REG(9:0) = OVER_CLK (0<=OVER_CLK <=1023)

Transmission Clock Frequency

In master mode, the clock frequency is derived from the 13-MHz master clock and can be programmed from 5.8 kHz to 6 MHz in increments of 83 ns.

Control bit:
CLOCK_FREQUENCY_REG(10:0) = CLK_FREQ
(2<=CLK_FREQ <= 2047)

with
($t_{CLK} = t_{13MHz} * CLK\_FREQ$)

16.1.1.2 Sample Setup for Communication μ-Law Interface Using Interrupts

MCSI Configuration

An example of communication μ-law interface setup using interrupts follows.

- ❏ CPU_Write(0x0000) = CONTROL_REG (disable MCSI for setup)
- ❏ CPU_Write(0x0007) = MAIN_PARAMETERS_REG (set up MCSI per configuration below)
  - Bit 15-14 (00b): No DMA
  - Bit 10 (0b): Positive polarity for frame
  - Bit 9 (0b): Normal synchronization mode
  - Bit 8 (0b): Short framing
  - Bit 7 (0b): Single channel
  - Bit 6 (0b): Slave mode
  - Bit 5 (0b): Burst mode
  - Bit 4 (0b): Positive edge for clock
  - Bit 3-0 (0111b): 8-bit data
- ❏ CPU_Write(0x0700) = INTERRUPTS_REG (all interrupts are enabled)
- ❏ CPU_Write(0x0000) = OVER_CLOCK_REG
- ❏ CPU_Write(0x0001) = CONTROL_REG (start MCSI)

Transmit Data Loading (TX_INT ISR)

- ❏ CPU_Write = TX_REG

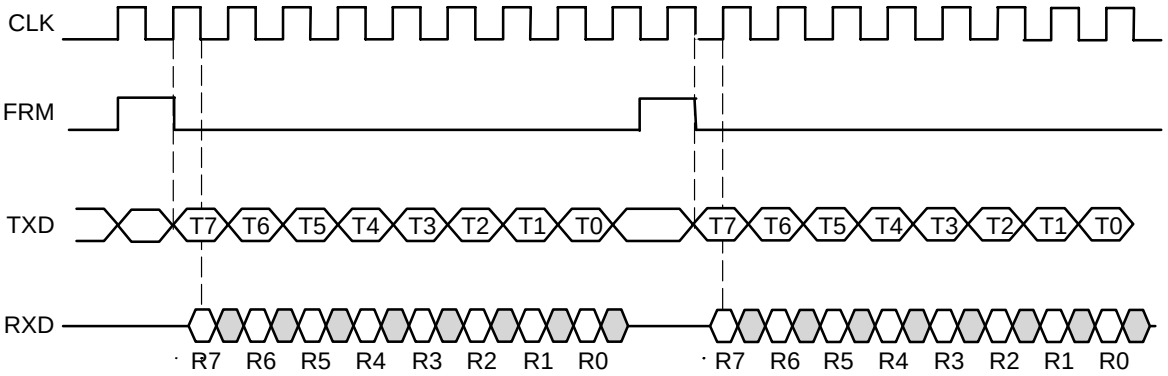
Received Data Loading (RX_INT ISR)

- ❏ CPU_Read= RX_REG

Stop MCSI

- CPU_Write(0x0000) = CONTROL_REG (disable MCSI clock)
- CPU_Write(0x0002) = CONTROL_REG (reset MCSI registers)

Figure 16–1. Communication μ-Law Interface Interrupts Waveform Example



16.1.1.3 Interface Management

Interrupt Generation

Three physical interrupts are available for real-time management of the MCSI by the CPU:

- RX_INT (data receive interrupt)
- TX_INT (data transmit interrupt)
- FERR_INT (frame-duration error interrupt)

RX_INT, TX_INT, and FERR_INT are maskable with dedicated programmable control bits of the interrupt register INTERRUPTS_REG.

- RX_INT is masked when MASK_IT_RX = 0.
- TX_INT is masked when MASK_IT_TX = 0.
- FERR_INT is masked when MASK_IT_RX = 0.

Each interrupt is associated with a flag bit in STATUS_REG that is set to 1 when the interrupt is generated. To acknowledge the interrupt and release the corresponding physical signal, the CPU must write a 1 at the bit location in the status register. The following list provides interrupt/flag bit associations:

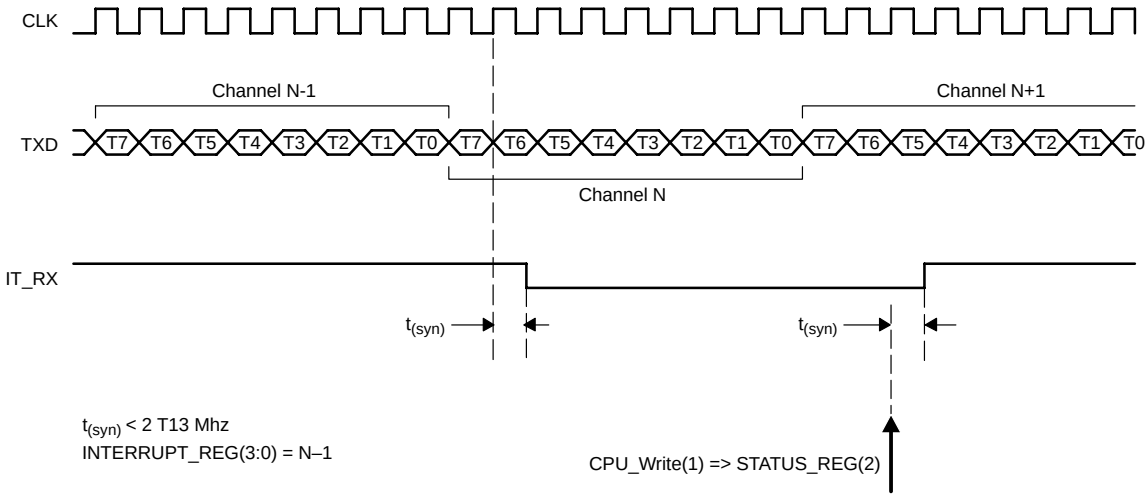
- RX_INT (RX_READY flag and acknowledge bit)
- TX_INT (TX_READY flag and acknowledge bit)
- FERR_INT (FRAME _ERROR flag and acknowledge bit)

Receive Interrupt

The receive interrupt is generated every frame after the completion of the reception of a data word:

- In single-channel mode, the interrupt is generated one clock period after the reception of the word.
- In multichannel mode, the interrupt is generated one clock period after the reception of the word of the channel whose number is defined by the NB_CHAN_IT_RX parameter of INTERRUPTS_REG register.

Figure 16–2. Receive Interrupt Timing Diagram

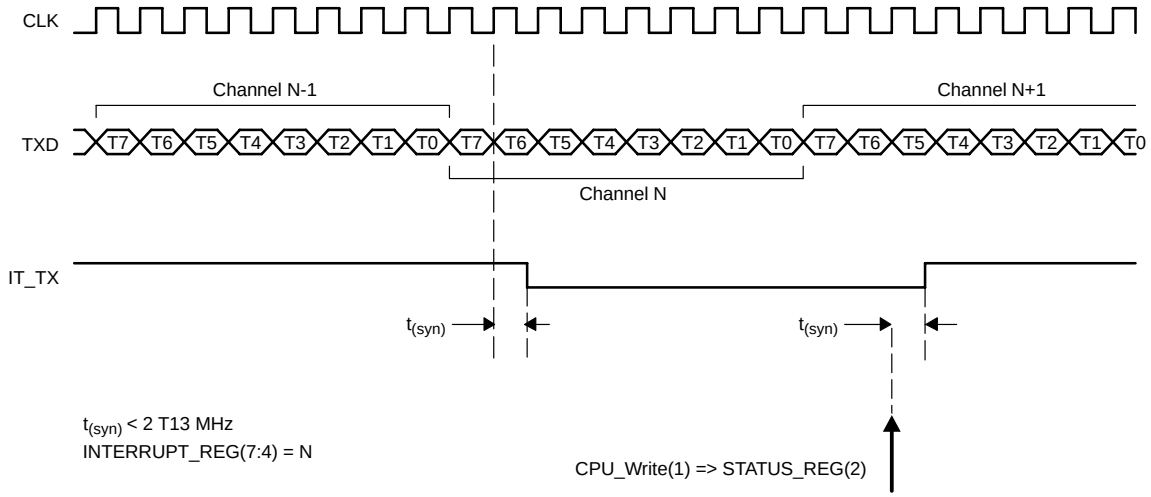


Transmit Interrupt

The transmit interrupt is generated every frame after the start of the transmission of a data word.

- In single-channel mode, the interrupt is generated one clock period after the beginning of the transmission of the word.
- In multichannel mode, the interrupt is generated one clock period after the transmission of the word of the channel whose number is defined by the NB_CHAN_IT_RX parameter of INTERRUPTS_REG.

Figure 16–3. Transmit Interrupt Timing Diagram



Frame Duration Error Interrupt

The frame duration error interrupt is generated only when:

- ❏ The interface is configured in burst mode (CONTINUOUS = 0).
- ❏ The frame duration is smaller or longer than the expected value.

Namely, expected frame duration = [(channels number) * (word size)] + (oversize number) in clock periods units with the oversize number defined in OVER_SIZE_REG.

If the frame duration is longer than the expected value, the interrupt is generated one clock period after the number of the OVER_SIZE clock periods, as defined in OVER_CLOCK parameter.

If the frame duration is smaller than the expected value, the interrupt is generated one clock period after the occurrence of the next frame pulse (first active edge).

Figure 16–4. Frame Duration Error—Too Many (Long)

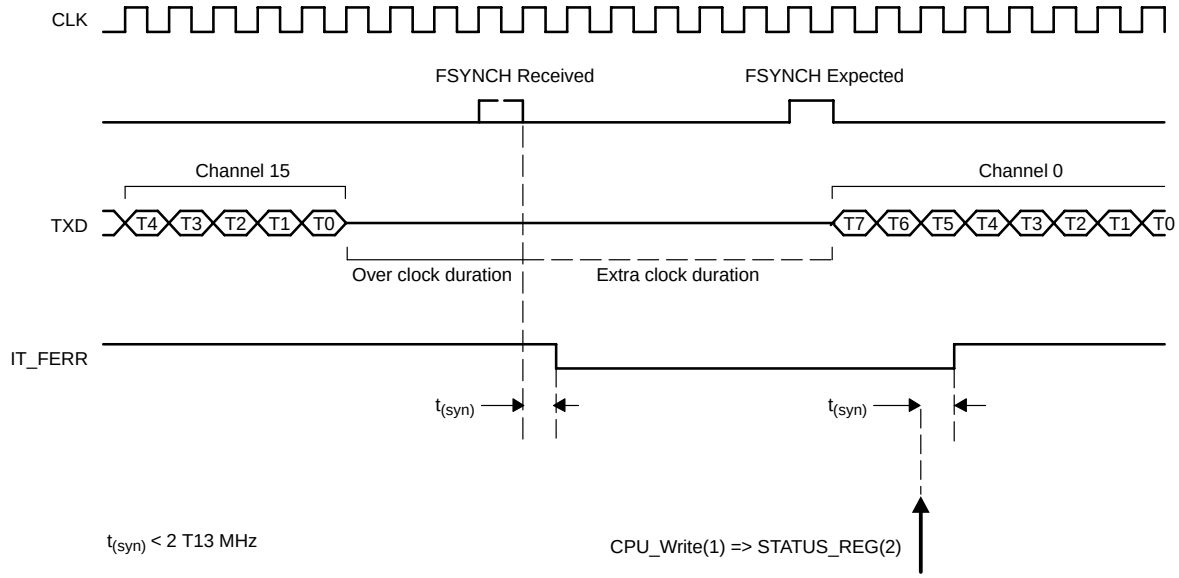
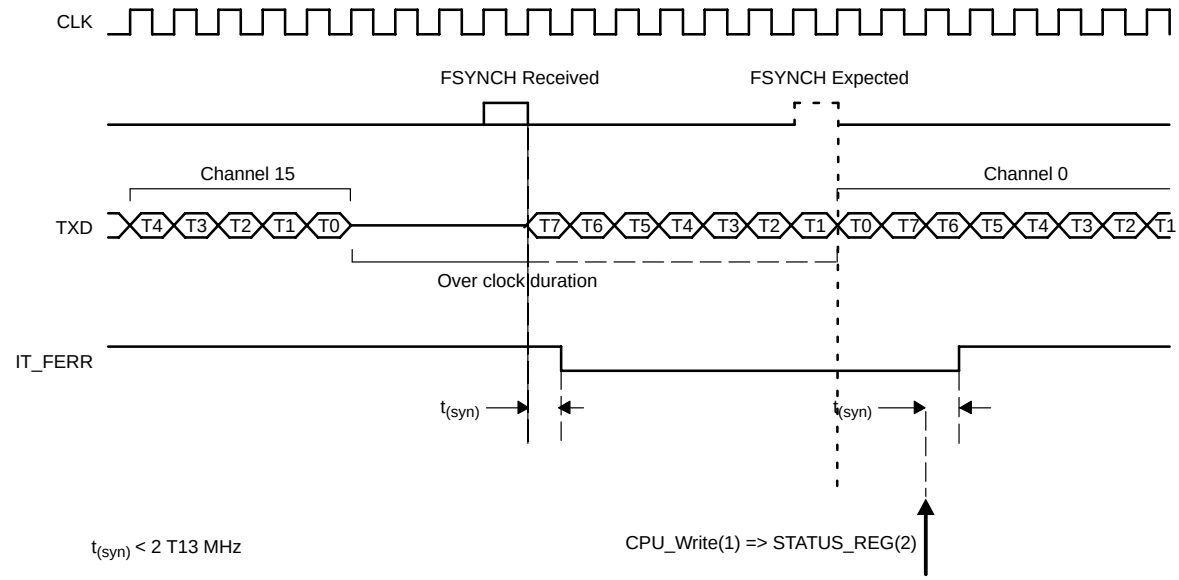


Figure 16–5. Frame Duration Error—Too Few (Short)



16.1.1.4 Interrupt Programming

At module reset, RX_INT, TX_INT, and FERR_INT are masked.

To validate an interrupt:

If in multichannel mode, the RX and TX interrupts can be configured to occur in a dedicated channel of the frame [1-16].

- CPU_WRITE(channel_nb) = INTERRUPTS_REG(3:0) for RX_INT
- INTERRUPTS_REG(7:4) for TX_INT

To unmask the interrupt:

- CPU_WRITE(1) =
  - INTERRUPTS_REG(8) for RX_INT
  - INTERRUPTS_REG(9) for TX_INT
  - INTERRUPTS_REG(10) for FERR_INT

On interrupt occurrence:

- CPU_READ=
  - STATUS_REG(1) for FERR_INT occurrence
  - STATUS_REG(2) for RX_INT occurrence
  - STATUS_REG(3) for RX character overflow
  - STATUS_REG(4) for TX_INT occurrence
  - STATUS_REG(5) for TX character underflow

Then, to release the interrupt signal and reset the corresponding status bits:

- CPU_WRITE(1) =
  - STATUS_REG(1) for FERR_INT release
  - STATUS_REG(2) for RX_INT release
  - STATUS_REG(4) for TX_INT release

16.1.1.5 DMA Channel Operation

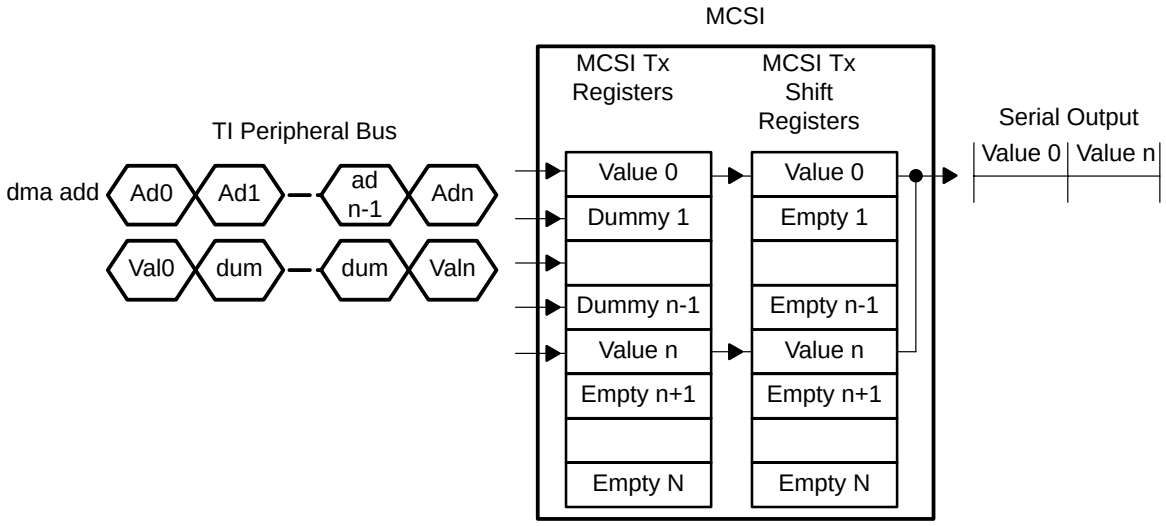
Both transmit and receive operations can be supported by DMA. DMA support is enabled by control bits in the MAIN_PARAMETERS_REG:

- MAIN_PARAMETERS_REG(15:14) = DMA_ENABLE(1:0)
  - TX_DMA_REQ enabled when DMA_ENABLE(0) = 1
  - TX_DMA_REQ disabled when DMA_ENABLE(0) = 0
  - RX_DMA_REQ enabled when DMA_ENABLE(1) = 1
  - RX_DMA_REQ disabled when DMA_ENABLE(1) = 0

Transmit DMA Transfers

A new transmit DMA transfer is initiated during the transmission of the last channel of a frame, at which time all data in the transmit registers (TX_REGs) has been moved to shift registers; the TX_REGs are now ready to be rewritten. If N channels are used, the DMA controller successively accesses all consecutive registers between TX_REG(0) and TX_REG(N-1). If some channels between TX_REG(0) and TX_REG(N-1) are not used, the DMA controller writes dummy values when addressing these unused registers (see Figure 16–6).

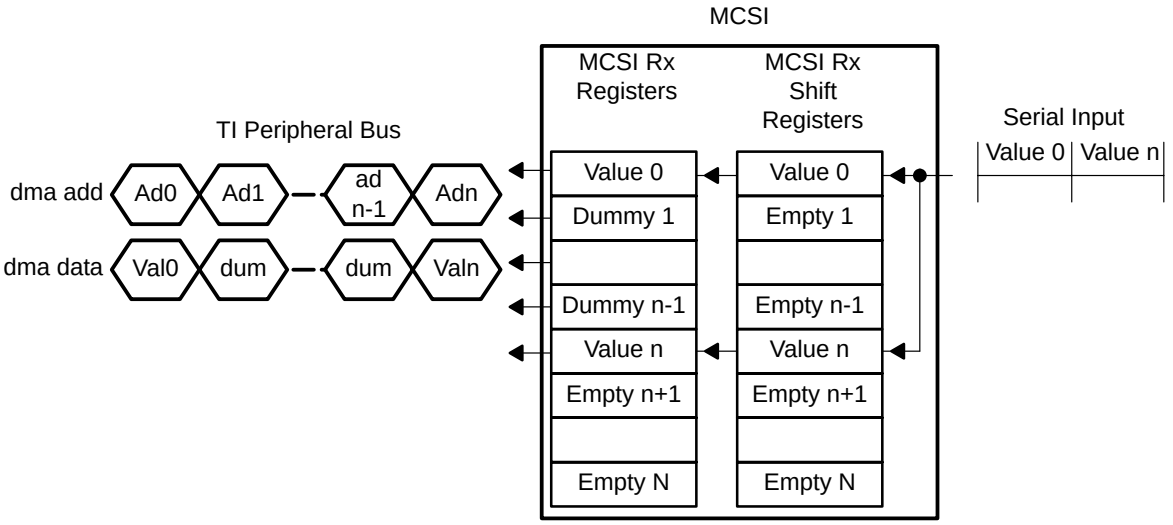
Figure 16–6. Transmit DMA Transfers



Receive DMA Transfers

A receive DMA transfer is initiated after the reception of the last channel of a frame, at which time all receive registers RX_REG have been updated and are ready to be read. If N channels are used, the DMA controller successively accesses all consecutive registers between RX_REG(0) and RX_REG(N-1). If some channels between RX_REG(0) and RX_REG(N-1) are not used, the DMA controller reads dummy values when addressing these unused registers (see Figure 16–7).

Figure 16–7. Receive DMA Transfers



A multichannel application cannot use DMA for some channels and interrupt servicing for others. RX/TX interrupts are not generated when DMA RX/TX transfers are enabled.

16.1.1.6 Interface Activation

Start Sequence

A typical sequence to start the interface is:

1) MCSI configuration:
   a) CPU_WRITE(0x0000)= CONTROL_REG in order to remove the write protection on the control registers
   b) CPU_WRITE(0x….)= MAIN_PARAMETERS_REG
   c) CPU_WRITE(0x….)= INTERRUPTS_REG
   d) CPU_WRITE(0x….)= CHANNEL_USED_REG
   e) CPU_WRITE(0x….)= CLOCK_FREQUENCY_REG
   f) CPU_WRITE(0x….)= OVER_CLOCK_REG
2) Transmit data loading for selected channels:
   a) CPU_WRITE(0x….)= TX_REG[channel index]
3) Enable MCSI clock:
   a) CPU_WRITE(0x0001)= CONTROL_REG

Stop Sequence

A typical sequence to stop the interface is:

1) Disable MCSI clock: CPU_WRITE(0x0000) = CONTROL_REG

   The status register keeps its content even after the stop of the transmission. The control registers can now be modified.

2) Software reset: CPU_WRITE(0x0002) = CONTROL_REG

   The software reset initializes the status register.

Software Reset

The MCSI software reset is activated with the SW_RESET bit of the control register (CONTROL_REG) (see Table 16–2, *Activity Control Register*).

This reset is limited to the control and status registers, the internal state machine, and the PISO and SIPO logic. The parameters registers are not affected by this software reset.

On the software reset, the MCSI reference clock is disabled, thus halting the execution of any current operating mode.

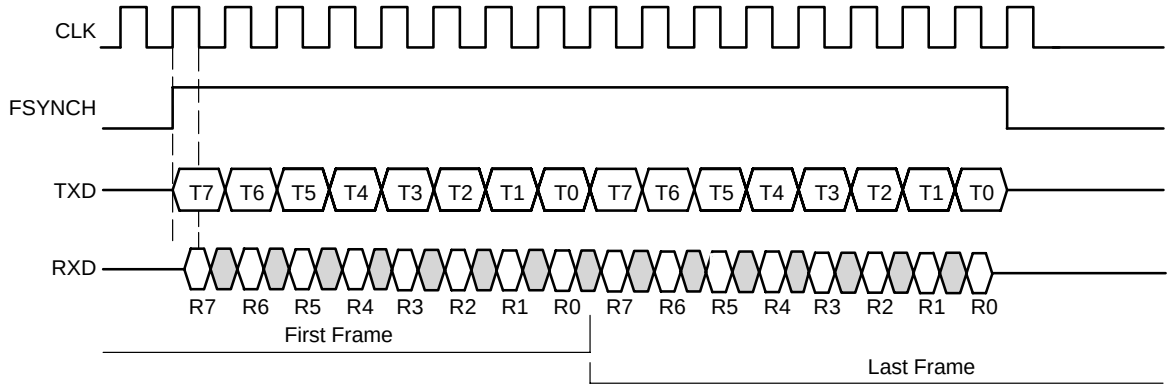
16.1.1.7 Functional Mode Chronograms

All chronograms below are based on a positive clock polarity with parameter CLOCK_POL = 0.

(Transmit on rising edge/receive on falling edge)

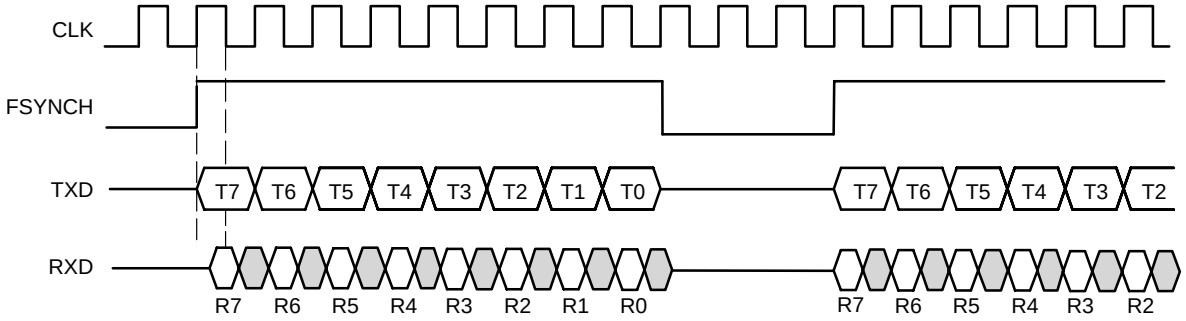
Single-Channel/Alternate Long Framing

Figure 16–8. Single-Channel/Alternate Long Framing



Single-Channel/Alternate Long Framing/Burst

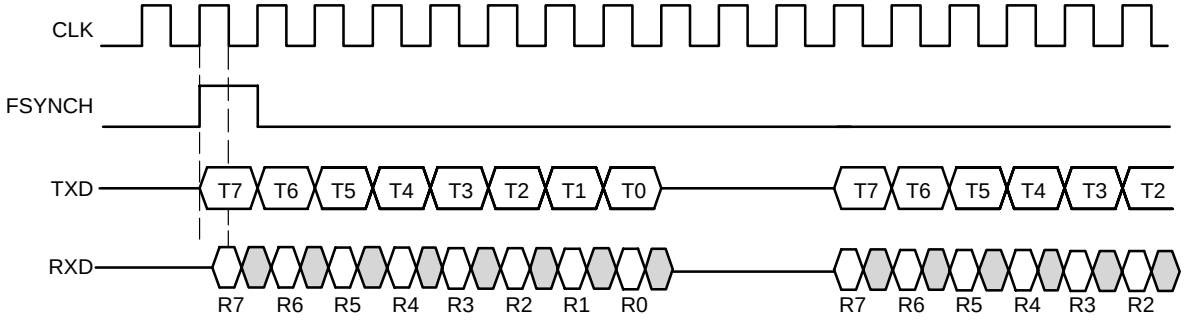
Figure 16–9. Single-Channel/Alternate Long Framing/Burst



OVER_CLOCK_REG = 0x0003

Single-Channel/Alternate Short Framing/Continuous/Burst

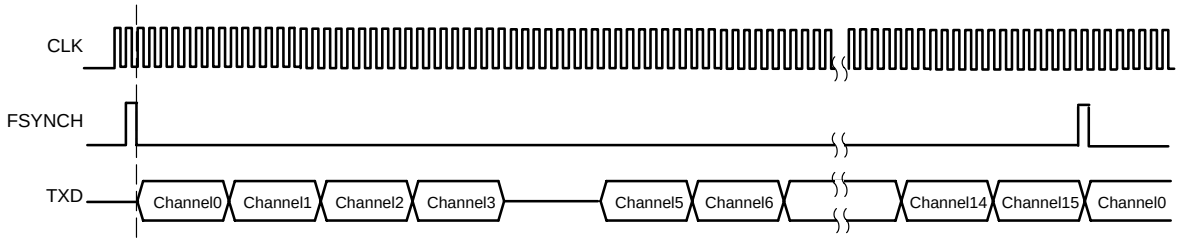
Figure 16–10. Single-Channel/Alternate Short Framing/Continuous/Burst



OVER_CLOCK_REG = 0x0003

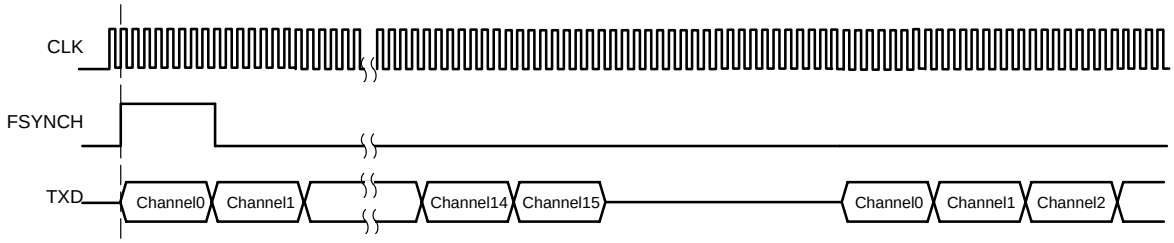
Multichannel/Normal Short Framing/Channel4 Disable

Figure 16–11. Multichannel/Normal Short Framing/Channel4 Disable



Multichannel/Alternate Long Framing/Continuous/Burst

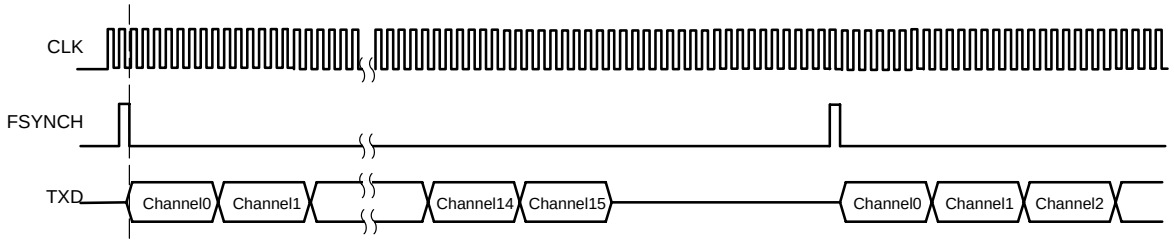
Figure 16–12. Multichannel/Alternate Long Framing/Continuous/Burst



OVER_CLOCK_REG = 0x0013

Multichannel/Normal Short Framing/Burst

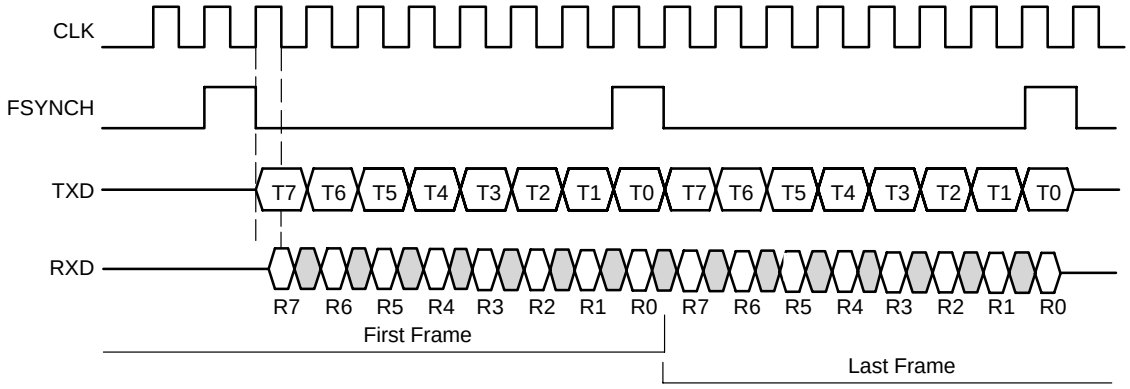
Figure 16–13. Multichannel/Normal Short Framing/Burst



OVER_CLOCK_REG = 0x0013

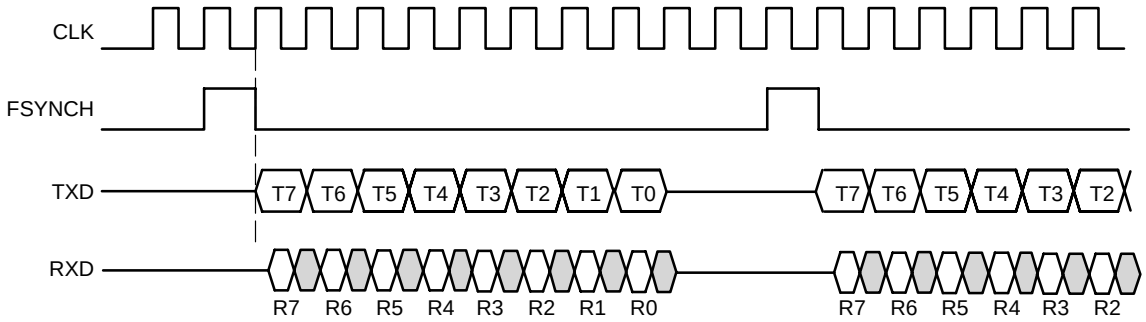
Single-Channel/Normal Short Framing

Figure 16–14. Single-Channel/Normal Short Framing



Single-Channel/Normal Short Framing/Burst

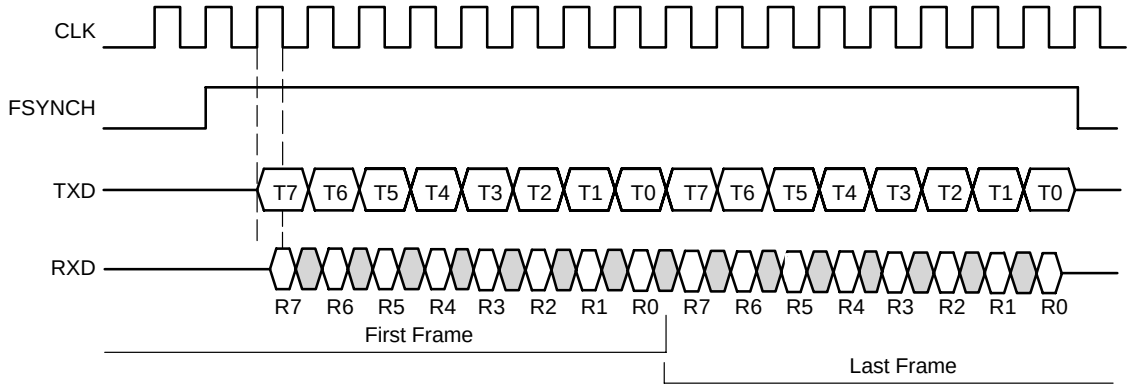
Figure 16–15. Single-Channel/Normal Short Framing/Burst



OVER_CLOCK_REG = 0x0003

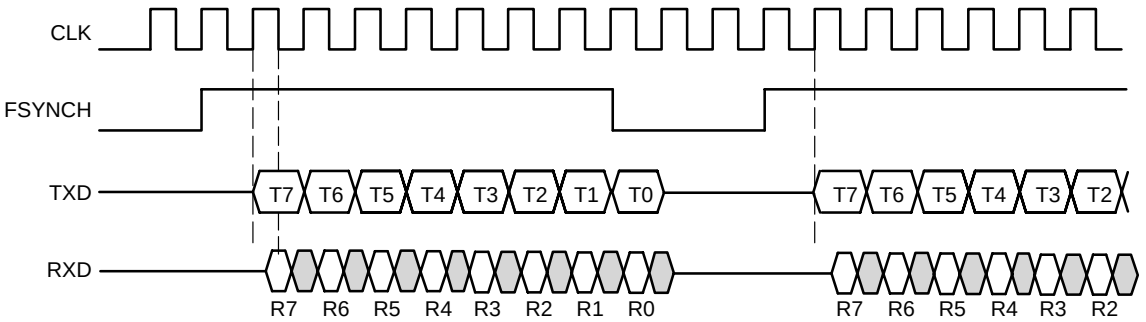
Single-Channel/Normal Long Framing

Figure 16–16. Single-Channel/Normal Long Framing



Single-Channel/Normal Long Framing/Burst

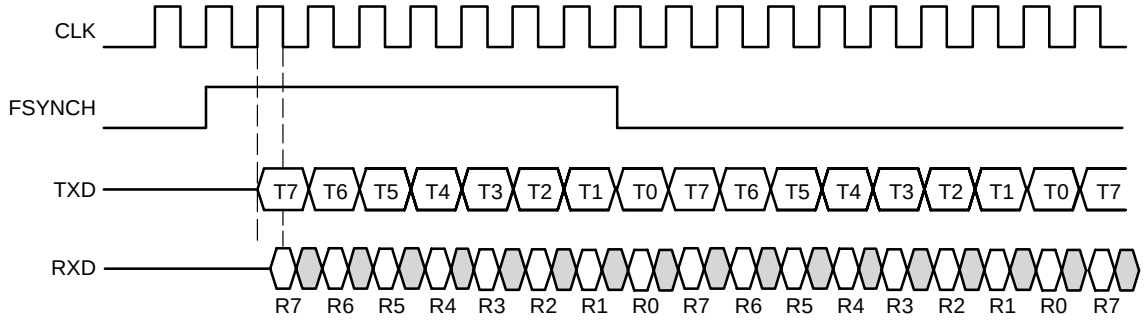
Figure 16–17. Single-Channel/Normal Long Framing/Burst



OVER_CLOCK_REG = 0x0003

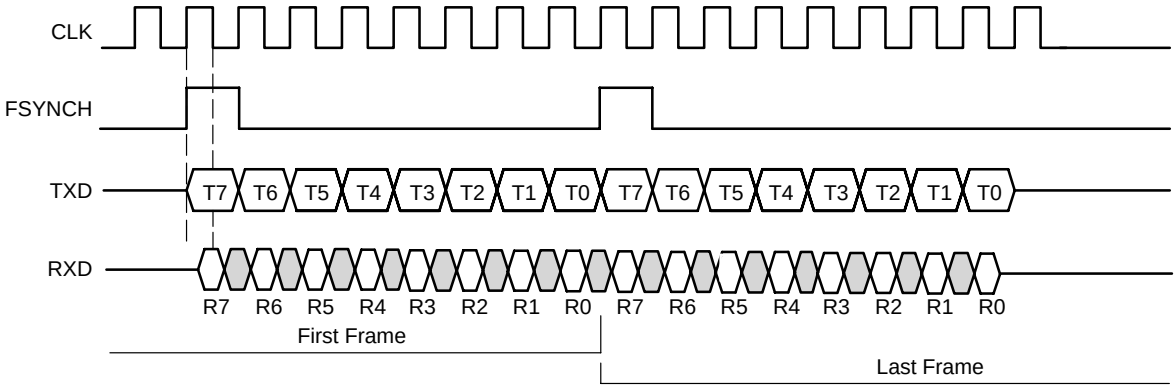
Single-Channel/Normal Long Framing/Continuous

Figure 16–18. Single-Channel/Normal Long/Continuous



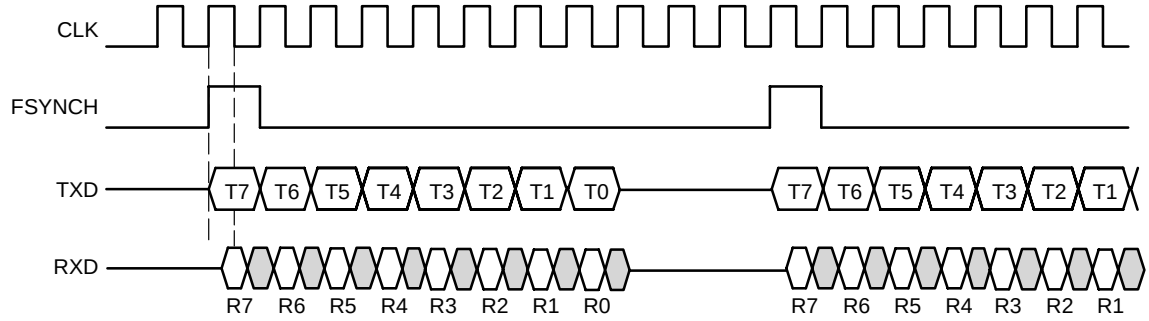
Single-Channel/Alternate Short Framing

Figure 16–19. Single-Channel/Alternate Short Framing



Single-Channel/Alternate Short Framing/Burst

Figure 16–20. Single-Channel/Alternate Short Framing/Burst



OVER_CLOCK_REG = 0x0003

16.1.2 MCSI Registers

Table 16–1 lists the 16-bit MCSI registers Table 16–2 through Table 16–10 describe the register bits.

Table 16–1. MCSI Registers

Name	Description	R/W	Offset
CONTROL_REG	Activity control	R/W	0800
MAIN_PARAMETERS_REG†	Main parameters	R/W	0801
INTERRUPTS_REG†	Interrupts mask	R/W	0802
CHANNEL_USED_REG†	Channel selection	R/W	0803
OVER_CLOCK_REG†	Oversized frame dimension	R/W	0804
CLOCK_FREQUENCY_REG†	MCSI clock frequency	R/W	0805
STATUS_REG	MCSI status	R/W	0806
TX_REG0...TX_REG15	16 transmit word registers	R/W	0820...082F
RX_REG0...RX_REG15	16 receive word registers	R	0831...083F

† Write protected if MCSI is enabled

Table 16–2. Activity Control Register (CONTROL_REG)

Bit	Name	Description	Access	Hardware Reset	Software Reset
15:3	RESERVED		R	0000 0000 0000 0	0000 0000 0000 0
2	DAI clk enable	Not used	R/W	0	0
1	MCSI software reset	Asynchronous reset of MCSI module (0 disable/1 enable)	R/W	0	1
0	MCSI clk enable	Enable clock of MCSI module (0 disable/1 enable)	R/W	0	0

Note:

The software reset is applied as long as the MCSI software reset bit is set to 1. A software reset disables the MSCI (the MCSI clk enable bit is cleared) and initializes the status register. It does not modify the other registers.

To clear an interrupt on the MCSI, the processor must write to the MCSI status register with the bit corresponding to the interrupt set to 1. The MCSI status register has a two-cycle latency when writing into it, so the interrupt line is cleared two cycles after a write. To prevent clearing the interrupt handler before the interrupt line is cleared, the interrupt routine must be at least two cycles long.

Table 16–3. Main Parameters Register (MAIN_PARAMETERS__REG)

Bit	Name	Description	Access	Hardware Reset
15:14	DMA enable	Enable bits for DMA: 00: Normal mode (No DMA) 01: DMA transmit mode, normal receive mode 10: Normal transmit mode, DMA receive mode 11: DMA transmit and receive mode	R/W	00
13:12	DAI config	Not used	R/W	00
11	DAI_SYN_REQ	Not used	R/W	0
10	FSYNCH_ POLARITY	Frame-synchronization pulse polarity (0 positive, 1 negative)	R/W	0
9	FSYNCH_MODE	Frame-synchronization pulse position (0 normal, 1 alternate)	R/W	0
8	FSYNCH_SIZE	Frame-synchronization pulse shape (0 short/, 1 long)	R/W	0
7	Multi/single	Frame structure (0 single, 1 multi)	R/W	0
6	MCSI mode	Interface transmission mode (0 slave, 1 master)	R/W	0
5	Continuous/burst	Frame mode (0 burst, 1 continuous)	R/W	0
4	CLOCK_ POLARITY	Clock edge selection (0 positive, 1 negative)	R/W	0
3:0	Word size	Word size in bits number (2 <size < 15) with 2 for 3 bits and 15 for 16 bits.	R/W	0000

Table 16–4. Interrupt Masks Register (INTERRUPTS_REG)

Bit	Name	Description	Access	Hardware Reset
15:11	RESERVED		R	0000 0
10	MASK_IT_ERROR	Mask of frame duration error interrupt (active at 0)	R/W	0
9	MASK_IT_TX	Mask of transmit interrupt (active at 0)	R/W	0
8	MASK_IT_RX	Mask of receive interrupt (active at 0)	R/W	0
7:4	Number channel for IT_TX	Channel number for transmit interrupt generation (0 <NB_CHAN < 15)	R/W	0000
3:0	Number channel for IT_RX	Channel number for receive interrupt generation (0 < NB_CHAN <15)	R/W	0000

Table 16–5. Channel Selection Register (CHANNEL_USED_REG)

Bit	Name	Access	Hardware Reset
15	USE_CH15	R/W	0
14	USE_CH14	R/W	0
13	USE_CH13	R/W	0
12	USE_CH12	R/W	0
11	USE_CH11	R/W	0
10	USE_CH10	R/W	0
9	USE_CH9	R/W	0
8	USE_CH8	R/W	0
7	USE_CH7	R/W	0
6	USE_CH6	R/W	0
5	USE_CH5	R/W	0
4	USE_CH4	R/W	0
3	USE_CH3	R/W	0
2	USE_CH2	R/W	0
1	USE_CH1	R/W	0
0	USE_CH0	R/W	0

The channel selection register is only used in multichannel mode.

USE_CHL[i] select channel [i] for data transmission (active 1).

Table 16–6. Oversized Frame Dimension Register (OVER_CLOCK_REG)

Bit	Name	Description	Access	Hardware Reset
15:10	RESERVED		R	0000 00
9:0	OVER_CLOCK	Overhead clock periods in frame duration (0 = OVER_CLOCK = 1023)	R/W	00 0000 0000

Table 16–7. Clock Frequency Register (CLOCK_FREQUENCY_REG)

Bit	Name	Description	Access	Hardware Reset
15:11	RESERVED		R	0000 0
10:0	CLK_FREQ	Division factor of 13-MHz reference clock (2<CLK_FREQ< 2047) In master mode, this register defines the transmission baud rate from a frequency ratio based on a 13-MHz reference clock. The transmission clock frequency can be programmed from 5.8 kHz to 6 MHz in steps or increments of 77 ns. Clock frequency = 13 MHz /CLK_FREQ with 2 <CLK_FREQ <2047.	R/W	000 0000 0000

The clock frequency register is used in master mode only when the interface generates the serial clock (see Table 16–7).

CLK_FREQ: Division factor of 13-MHz reference clock (2<CLK_FREQ< 2047)

In master mode, the clock frequency register defines the transmission baud rate from a frequency ratio based on a 13-MHz reference clock. The transmission clock frequency can be programmed from 6.3 kHz to 6.5 MHz in steps or increments of 77 ns.

Clock frequency = 13 MHz /CLK_FREQ with 2 < CLK_FREQ < 2047.

Table 16–8. Interface Status Register (STATUS_REG)

Bit	Name	Description	Access	Hardware Reset	Software Reset
15:7	RESERVED		R	0000 0000 0	0000 0000 0
6	DAI ready	Not used	R/W	0	0
5	TX underflow	Transmit underflow (0 no underflow, 1 underflow)	R	0	0
4	TX ready	Flag for transmit interrupt occurrence (0 no interrupt, 1 interrupt)	R/W	0	0
3	RX overflow	Receive overflow (0 no overflow, 1 overflow)	R	0	0
2	RX ready	Flag for receive interrupt occurrence (0 no interrupt, 1 interrupt)	R/W	0	0
1	Error type few/ many	Too short (few) or too long frame (many) status (0 short, 1 long)	R	0	0
0	Frame error	Error flag when wrong frame duration (0 correct, 1 bad)	R/W	0	0

The interface status register is cleared by a software reset.

Table 16–9. Transmit Word Registers (TX_REG15...TX_REG0)

Bit	Name	Access	Hardware Reset
15	b15	R/W	U
14	b14	R/W	U
13	b13	R/W	U
12	b12	R/W	U
11	b11	R/W	U
10	b10	R/W	U
9	b9	R/W	U
8	b8	R/W	U
7	b7	R/W	U
6	b6	R/W	U
5	b5	R/W	U
4	b4	R/W	U
3	b3	R/W	U
2	b2	R/W	U
1	b1	R/W	U
0	b0	R/W	U

Note: The MCSI transmits the most significant bit first. For example, if the WORD_SIZE equals 11, the upper 12 bits of the TX registers are transmitted.

Table 16–10. Receive Word Registers (RX_REG15...RX_REG0)

Bit	Name	Access	Hardware Reset
15	b15	R	U
14	b14	R	U
13	b13	R	U
12	b12	R	U
11	b11	R	U
10	b10	R	U
9	b9	R	U
8	b8	R	U
7	b7	R	U
6	b6	R	U
5	b5	R	U
4	b4	R	U

Note: The MCSI receives the most significant bit first. For example, if the WORD_SIZE equals 11, the upper 12 bits of the RX registers contain the received data, and the lower 4 bits are zeros.

Table 16–10. Receive Word Registers (RX_REG15...RX_REG0) (Continued)

Bit	Name	Access	Hardware Reset
3	b3	R	U
2	b2	R	U
1	b1	R	U
0	b0	R	U

Note: The MCSI receives the most significant bit first. For example, if the WORD_SIZE equals 11, the upper 12 bits of the RX registers contain the received data, and the lower 4 bits are zeros.

16.2 MPU MCSI

This section provides information on the MPU MCSI in OMAP730.

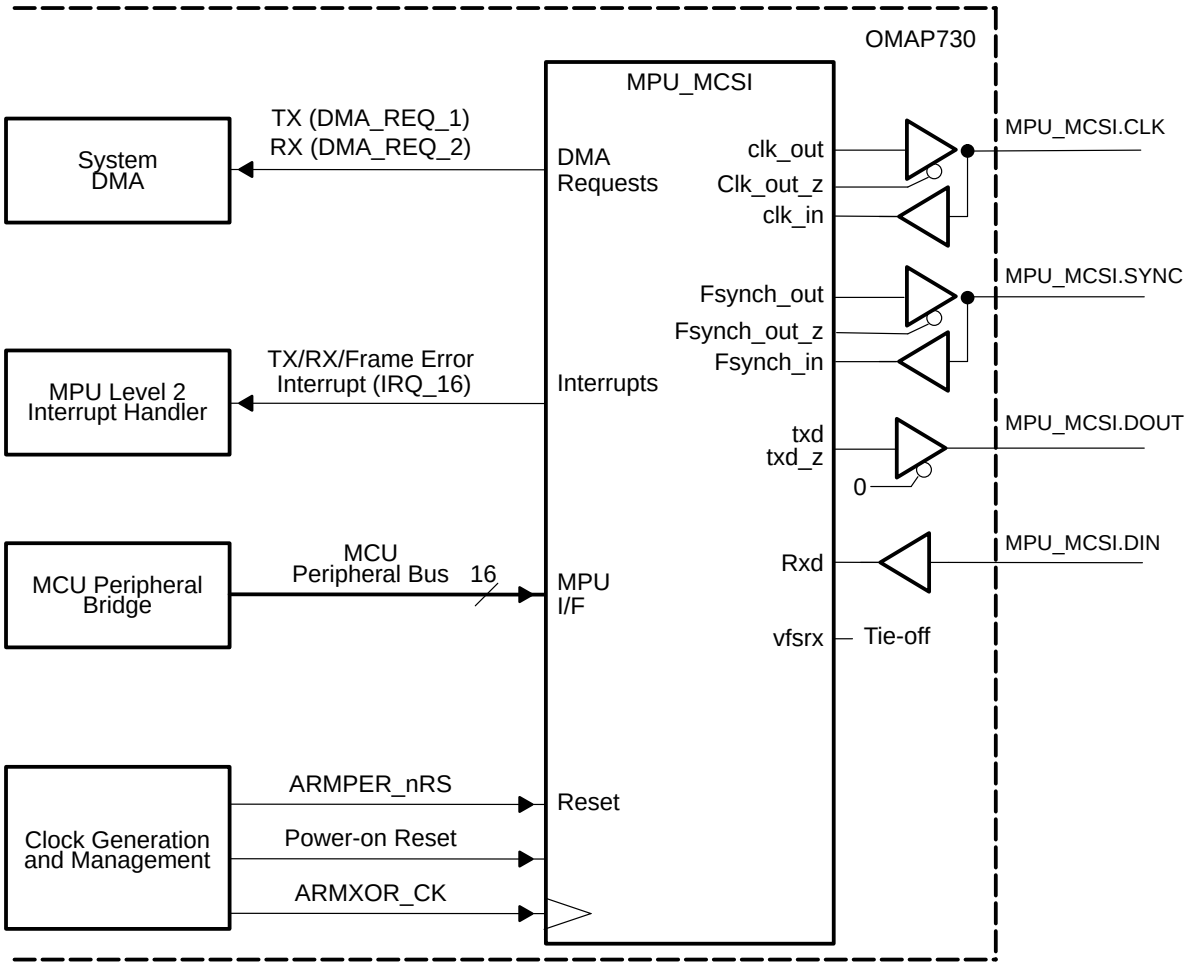
16.2.1 MPU MCSI Pin Description

Table 16–11 identifies the MPU MCSI I/O pins.

Table 16–11. MPU MCSI Pin Descriptions

Pin	I/O Direction	Description
MPU_MCSI.DIN	In	Data input
MPU_MCSI.DOUT	Out	Data output
MPU_MCSI.CLK	In/out	Bit clock
MPU_MCSI.SYNC	In/out	Frame synchronization

Figure 16–21. MPU MCSI Interface Diagram



16.2.2 MPU MCSI Interrupt Mapping

Table 16–12 identifies the MPU MCSI interrupt mappings. MPU MCSI generates a level 2 interrupt for the MPU. Only one MPU MCSI interrupt covers TX, RX, and frame error conditions; software must check the MPU MCSI status register to determine the interrupt source.

Table 16–12. MPU MCSI Interrupt Mapping

Incoming Interrupts	Level 2 MPU Interrupt
MPU_MCSI TX interrupt	IRQ_16
MPU_MCSI RX interrupt	IRQ_16
MPU_MCSI Frame Error	IRQ_16

16.2.3 MPU MCSI DMA Request Mapping

Table 16–13 identifies MPU MCSI DMA request lines.

Table 16–13. TDMA Request Mapping—MPU MCSI

DMA Request Source	DMA Request Line—MPU
MPU_MCSI TX	DMA_REQ_01
MPU_MCSI RX	DMA_REQ_02

16.2.4 MCSI Addresses and Mapping

The base address for each MCSI register map is:

- MPU MCSI (Bluetooth MCSI):
  - FFFB:2000 (MPU memory map)

Table 16–14 shows the MCSI registers and their offset addresses.

Table 16–14. MCSI Register Mapping

Register Name	Offset Address
RX15	0x7E
RX14	0x7C
RX13	0x7A
RX12	0x78
RX11	0x76
RX10	0x74
RX9	0x72
RX8	0x70

Table 16–14. MCSI Register Mapping (Continued)

Register Name	Offset Address
RX7	0x6E
RX6	0x6C
RX5	0x6A
RX4	0x68
RX3	0x66
RX2	0x64
RX1	0x62
RX0	0x60
TX15	0x5E
TX14	0x5C
TX13	0x5A
TX12	0x58
TX11	0x56
TX10	0x54
TX9	0x52
TX8	0x50
TX7	0x4E
TX6	0x4C
TX5	0x4A
TX4	0x48
TX3	0x46
TX2	0x44
TX1	0x42
TX0	0x40
RESERVED	0x3F
/	/
RESERVED	0x0E
Status	0x0C
Clock frequency	0x0A
Over-clock	0x08
Channel used	0x06
Interrupts	0x04

Table 16–14. MCSI Register Mapping (Continued)

Register Name	Offset Address
Main parameters	0x02
Control	0x00

Chapter 17

Power and Control Clock

This chapter describes the power and control clock (PCC) module, outlines its architecture, and provides hardware and software information to designers.

Topic **Page**

17.1 General Description

The power and clock control (PCC) module generates and manages clocks and reset signals for OMAP3.2, digital baseband (DBB), and some peripherals. It controls chip-level power-down modes and handles chip-level wake-up events. In deep sleep mode, this module is still active to monitor wake-up events.

17.1.1 Power and Clock Control Features

The power and clock control module has the following features:

- ❏ Performs the transitions between the power modes (awake, big sleep, and deep sleep)
- ❏ Handles the idle/wake-up handshake of OMAP and DBB
- ❏ Monitors wake-up events
- ❏ Controls system clock input sources (slicer, APLL)
- ❏ Performs the calibration of the 32-kHz oscillator (gauging)
- ❏ Manages the clocks and resets distributed to OMAP, DBB, and some peripherals
- ❏ Manages the security reset and violation
- ❏ Handles the power-up sequence
- ❏ Manages the APLL 13-MHz-to-slicer and slicer-to-APLL 13-MHz clock switch
- ❏ Controls an on-chip PLL that generates a 96-MHz clock
- ❏ Handles the embedded low-dropout voltage (LDO) regulator with bypass capability

17.1.2 Functional Description

The PCC module runs at 32 kHz and provides three main system power modes:

- ❏ Awake
- ❏ Big sleep
- ❏ Deep sleep

The functional state machine (FSM) handles the idle/wakeup handshake with OMAP and monitors the wake-up events. The FSM also controls the wake-up of the input system clock, generates the clock and resets to OMAP and some peripherals, and manages the power-up sequence. For more detail on these modes and the transitions among them, see Section 17.2.1, *ULPD Subsystem*.

The PCC module defines two functional modes, depending on the system clock source:

- Low-accuracy mode: Uses the APLL 13-MHz clock as the system clock. This mode is selected when the DBB is not awake and no peripherals need the accurate clock.
- High-accuracy mode: Uses the slicer output as a system clock.

The PCC module manages switching between these two modes. For more details on the switch condition, see Section 17.2.9, *Clock Switching Conditions*.

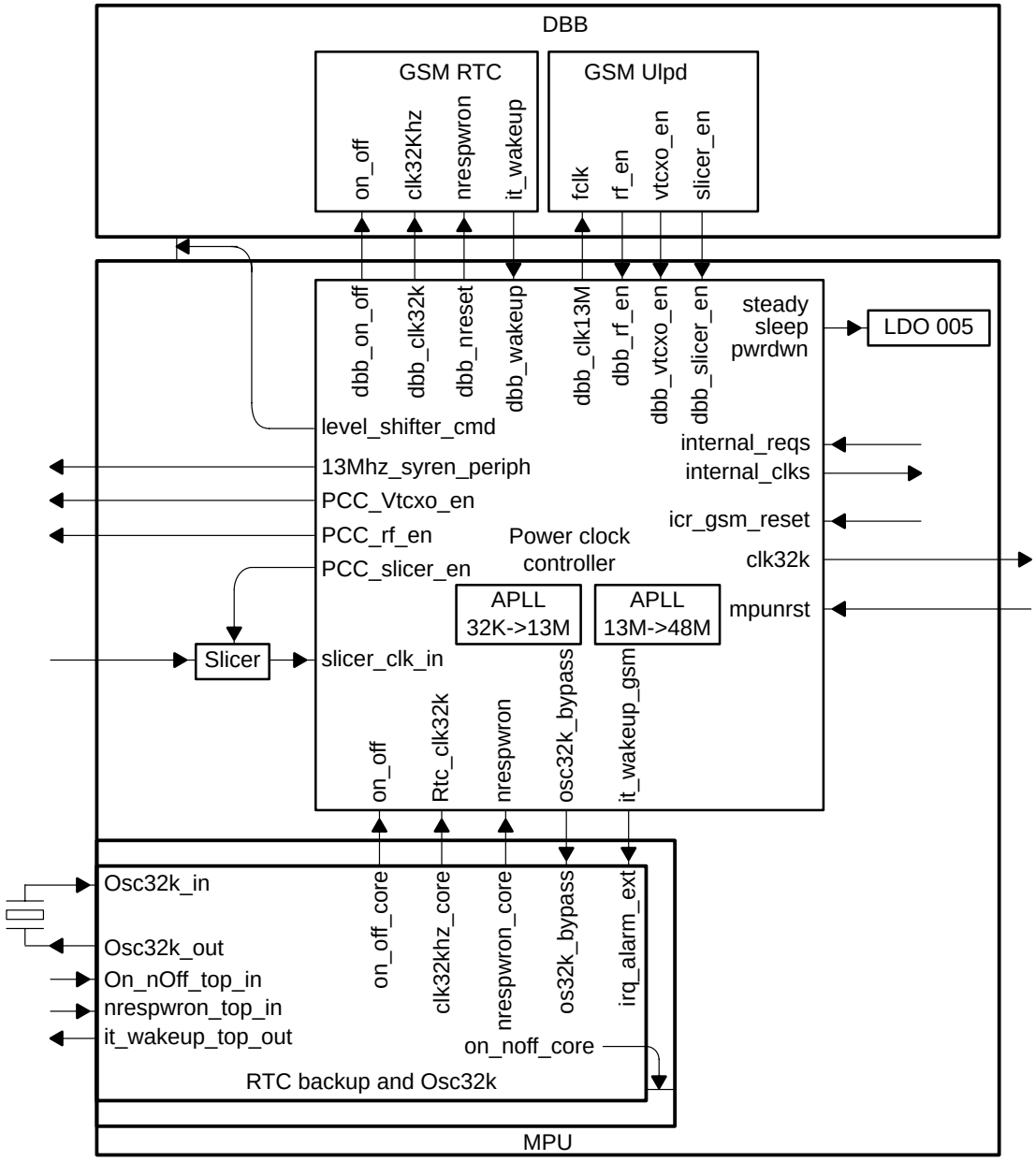
You can use the control register file, an MPU peripheral connected to the MPU private peripheral bus, to set/configure the ULPD features (see Section 17.6, *PCC ULPD Registers*.

The PCC module manages the low-dropout voltage regulator according to the OMAP and DBB states. If these submodules are in sleep mode, the LDO is off; otherwise, it is on.

The PCC also includes a 96-MHz APLL for peripherals that need a clock greater than 13-MHz.

Figure 17–1 shows the PCC module connections.

Figure 17–1. PCC Module Connections



❑ OMAP interface

The OMAP interface includes the clock and reset signals and the wake-up signals.

The ULPD manages all of the OMAP protocols, reset, sleep sequence, and awake sequence.

❑ RTC interface

The RTC provides the 32-kHz clock for OMAP730. It also provides the ON_nOFF and NRESPWRON reset signals. It receives the IT_WKUP_GSM that is the power-ramping order for TWL3016 when the GSM is in sleep mode.

The OSC_BYPASS is used to put the oscillator in power down and to measure its power consumption.

❑ DBB interface

The PCC drives the input clock and the resets of the DBB. The DBB drives the slicer in the PCC according to the ULPD PCC subsystem scheme.

The reset signals have the following feature:

- DBB_NRESET is ANDed between ICR_GSM_RESET and GSM_PROTECT_RESET.
  - The ICR_GSM_RESET is a register output from the ICR module. It allows to reset the DBB by the application processor.
  - The GSM_PROTECT_RESET is asserted when the application processor tries to access the memory space reserved to the DBB.
- DBB_NRSPWRON is the NRESPWRON from RTC backup.
- DBB_ON_nOFF is ANDed between RTC_ON_nOFF and ICR_GSM_RESET and GSM_PROTECT_RESET.

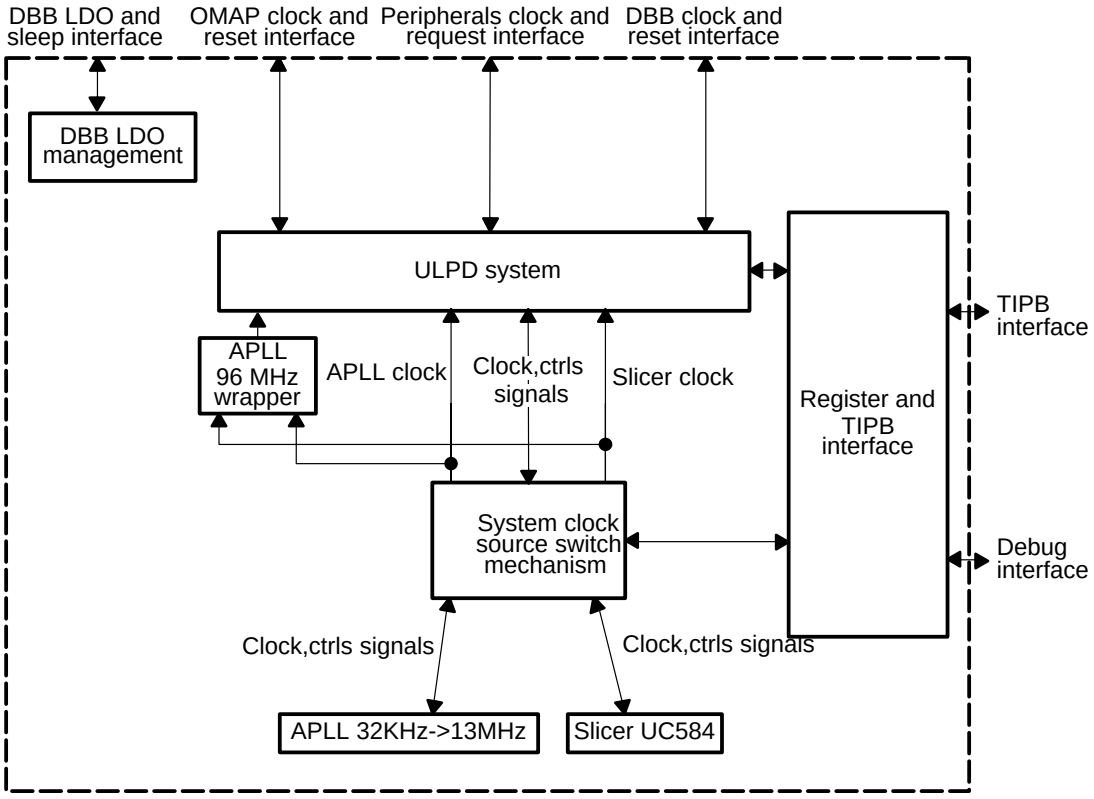
The NIRQ_DEEP_SLEEP_WKUP signal allows to ramp up the power via the TWL3016 when DBB is coming out from deep sleep state.

17.2 Functional Description

The PCC module consists of the following blocks:

- Ultralow-power device (ULPD) system: manages the OMAP clock and reset interface, the peripheral clocks and request interface, the OMAP digital phase-locked loop (DPLL) LDO, the DBB clock and reset interface, and the MPU digital phase-locked loop LDO.
- System clock switch mechanism: switches between the slicer output and the APLL 13-MHz output. The switching decision is made directly by the software via the TIPB interface or by the hardware, depending on the configuration.
- DBB DPLL LDO block: manages the low dropout regulator (LDO) of the DBB digital phase-locked loop (DPLL).
- Register and TIPB interface: allows control and configuration of the PCC module.

Figure 17–2. PCC Module Functional Block Diagram



17.2.1 ULPD Subsystem

The ULPD has one state machine, a clock management module, and a control register file.

- The FSM manages the global power modes transitions. It handles the idle/wakeup handshake with OMAP and monitors the wake up events. The FSM uses setup timer to manage the sequencing of the wake up procedure. Each setup timer is associated with a specific analog cell.
- The clock management module is composed of clock gating logic, multiplexers, and clock dividers. It generates and manages clocks to OMAP and to some peripherals. It also manages the 96-MHz clocks. This clock is generated by an on-chip PLL referred to as ULPD_PLL, which is not effectively part of the ULPD module.
- From the user's point of view, the control register file is an MPU peripheral connected to the MPU private peripheral bus and is combined with PCC registers to set/configure the PCC/ULPD features.

17.2.1.1 ULPD Setup Counters

As stated earlier, the ULPD can sequence properly the wake up of the system from deep sleep by cascading three consecutive wait-state counters and wait stability of the analog cells of the device, such as RF, VTCXO, and SLICER.

Setup counters are cascaded and must be programmed with the stabilization time of the associated analog cell (see the default value in the register description). Whenever a counter underflows, it enables the next analog cell and triggers the associated counter.

The RF enable is asserted as soon as the FSM leaves the deep sleep state. Then the cascaded flow of the setup_analog_cell starts with setup_analog_cell1, enables VTCXO, then setup_analog_cell2, enables SLICER, and terminates the wake-up sequence with setup_analog_cell3. When setup_analog_cell3 underflows, all the analog cells are designed to be stable and the input system clock is released internally in ULPD. The ULPD FSM can then move in big sleep or awake mode

17.2.1.2 Power Modes

The ULPD handles three global power modes, and the ULPD state machine, which is in charge of the wakeup/idle handshake with OMAP3.2, manages the states of the system in each mode and performs transitions between the modes.

- In deep sleep mode, all internal clocks are inactive except the CLK32K clock, which is the ULPD state machine clock. In this mode, ULPD_PLL is always inactive. In oscillator mode, the oscillator is disabled; therefore, the system input clock is off, except when POWER_CTRL_REG[9] = 0. In external mode, the input system clock is allowed to be on or off; this is not controlled by ULPD.

Note: Deep Sleep

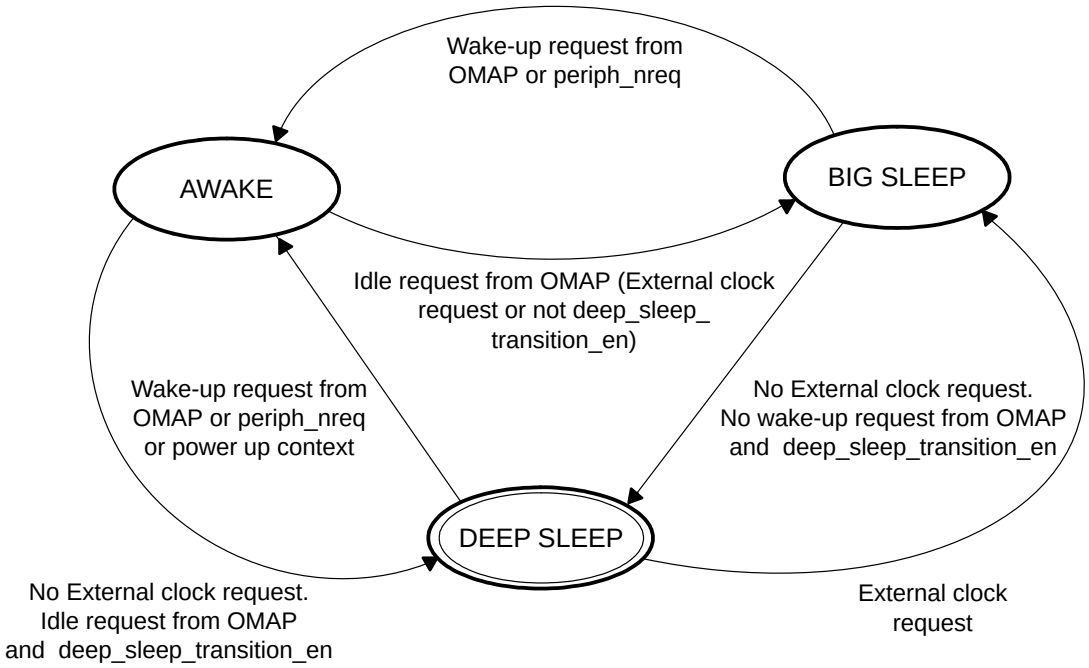
The OSC12M_STOP output of ULPD is active (high) every time the state machine is in deep sleep state. EXT_CLK_REQ is the same signal but with inverted polarity.

- In big sleep mode, the OMAP input clock is inactive, the CLK32K clock is active, and the system input clock is active in both oscillator and external modes. This state is characterized by active external clock requests or POWER_CTRL_REG[4] = 0. This mode has a shorter wake-up latency. It also allows providing clocks (system frequency clocks and/or ULPD_PLL clock) to peripherals whenever requested and while the OMAP3.2 input clock is stopped.
- In awake mode, the OMAP input clock is active, as are any requested peripheral clocks. In this mode, the CLK32K clock is also active.

17.2.1.3 Power Mode Transitions

Figure 17–3 shows the basic FSM scheme.

Figure 17–3. ULPD FSM



- Power-on transition to deep sleep mode

  At power up, when the power-up input signal NPORE is asserted (low), the ULPD FSM enters asynchronously in deep sleep mode. When NPORE is released, the FSM automatically switches from deep sleep to awake synchronously with the 32-kHz clock.

- Transition from deep sleep mode

  In deep sleep mode, the FSM monitors OMAP wake-up request and external clock requests. OMAP propagates asynchronously unmasked periph-

eral interrupts to generate a wake-up request. External clock requests and OMAP wake-up request are, respectively, initiators of deep sleep to big sleep and deep sleep to awake transitions.

The power-up reset forces a transition from deep sleep to awake, whatever the status of any wake-up requests.

- Transition from deep sleep mode to big sleep mode

  Transition to big sleep state occurs when there is at least one specific external clock request (see Table 17–1).

Table 17–1. Deep Sleep to Big Sleep Transition

Source	Signals
Camera	CAM_DPLL_MCLK_REQ
UART 1,3	UART1_DPLL_REQ UART3_DPLL_REQ
USB host	USB_HOST_DPLL_REQ
USB client 48 MHz	USB_DPLL_MCLK_REQ
MMC	MMCSDIO_DPLL_REQ PCONF_MMC_DPLL_REQ
MCBSP1_REQ	MCBSP1_MCLK_REQ
MCBSP2_REQ	MCBSP2_MCLK_REQ
Software requests	SOFT_REQ_REG REGISTER†
DEEP_SLEEP_TRANSITION_EN (software)	DEEP_TRANSITION_ENABLE‡
DBB_RF_EN	DBB_RF_EN

† Software requests prevent the transition to deep sleep when leaving awake state but are not initiators of deep sleep to big sleep transition.

‡ DEEP_SLEEP_TRANSITION_EN is not initiator of deep sleep to big sleep transition. Its purpose is to keep FSM in big sleep mode and prevent transition to deep sleep mode when leaving awake state. This feature can be used to prepare a fast CPU wake-up.

- Transition from deep sleep to awake mode occurs whenever a wake-up event occurs:
  - OMAP asserts the wake-up request (WAKEUP_NREQ active low). A wake-up request is initiated by peripherals unmasked interrupts.
  - The input signal PERIPH_NREQ is asserted.
  - Power on reset (NPORE)
  - Warm reset (MPUNRST)
  - External reset sources

Table 17–2. Initiators of Deep Sleep to Awake Transition

Sources	Wake-up Signals
Power-on reset	NPORE
System reset pin	MPUNRST
OMAP	WAKEUP_NREQ
Security Violations	
Secure watchdog	SECURE_WDT_RESET
OMAP	POSECVIOLATION
GSM protect	MPU_RESET

- Transition from big sleep to deep sleep mode

  There are three necessary conditions leading to deep sleep mode:

  - Wake-up request from OMAP is not active.
  - Peripheral clock requests and software clock requests are inactive. Initiators to deep sleep –> big sleep transition are not active.
  - POWER_CTRL_REG[4] =1 (enable deep-sleep transition).

- Transition from big sleep to awake mode

  Transition to awake mode occurs upon an OMAP wake-up request (WAKEUP_NREQ active low) or when PERIPH_NREQ is asserted (low). An OMAP wake-up request is initiated by peripherals unmasked interrupts.

- Transition from awake to deep sleep mode

There are three necessary conditions leading to deep sleep mode.

- OMAP asserts CHIP_IDLE signal (active high) when no clocks are needed in OMAP or by any peripheral using OMAP output clocks.
- No peripheral clock requests or software requests. Initiators to deep sleep –> big sleep transition are inactive.
- POWER_CTRL_REG[4]=1 (enable deep sleep transition).

- Transition from awake to big sleep mode

  There are two necessary conditions leading to big sleep mode.

  - OMAP asserts the CHIP_IDLE (active high) when no clocks are needed in OMAP or by any peripherals using OMAP output clocks.
  - At least one specific peripheral clock request is active (see Table 17–1: Initiators to deep sleep –> big sleep transition) or POWER_CTRL_REG[4] = 0 (disable deep sleep transition).

17.2.1.4 Voltage Supply Control (Low Power)

The low power (LOW_PWR) signal is used in oscillator clock mode to control the external core voltage supply directly. When active high, the signal

LOW_PWR drives the external core voltage supply in low voltage (1.1v) operations.

The behavior of the LOW_PWR signal is set by software and can be configured as follows:

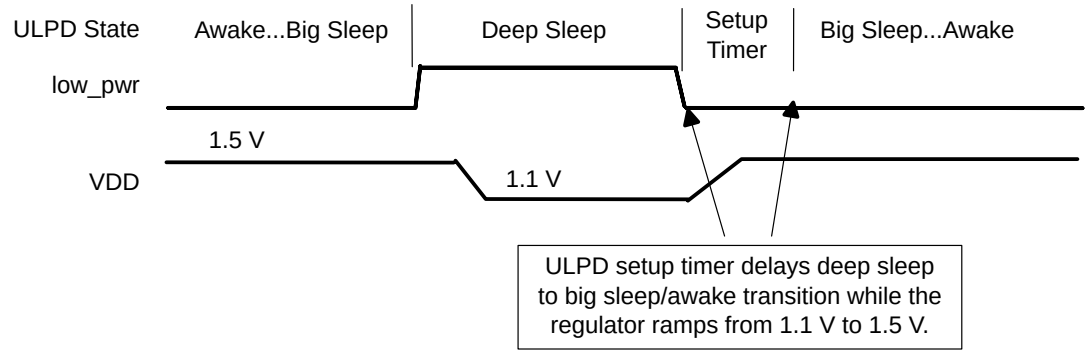
- ❏ POWER_CTRL_REG[0] set to 1: Enable LOW_PWR feature
- ❏ POWER_CTRL_REG[4] set to 1: Enable transition to deep sleep mode

The signal LOW_PWR switches to active high whenever the ULPD enters in deep sleep state. This way, the external core voltage supply can be driven in low-voltage operation in deep sleep mode, reducing the leakage current consumption in this mode.

When the ULPD exits deep sleep mode, the signal LOW_PWR switches back to inactive low, and the external core voltage supply ramps up to nominal 1.5 V.

At reset, the LOW_PWR feature is disabled, and POWER_CTRL_REG[0] is set to 0. The LOW_PWR signal is inactive low, which indicates nominal voltage requirement.

Figure 17–4. Low Power Signal Behavior

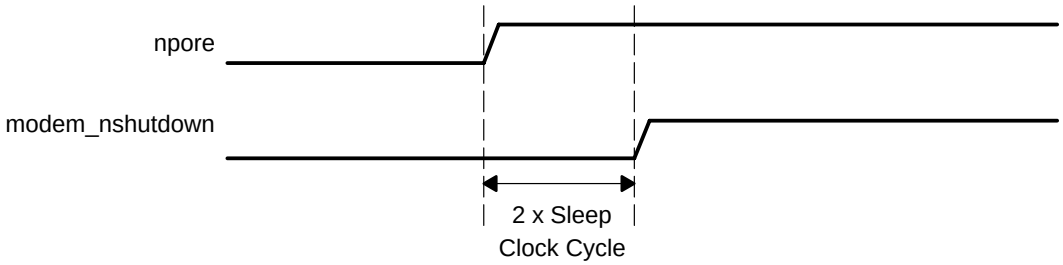


17.2.1.5 Battery Failed Interrupt

A battery failed event is indicated on the ULPD external fast-interrupt request signal (EXT_NFIQ active low). This signal is connected both on OMAP3.2 and ULPD in order to perform a power-down action. Upon EXT_NFIQ, the ULPD starts a programmable counter. When the counter underflows, ULPD generates the modem shut down signal (MODEM_NSHUTDOWN active low).

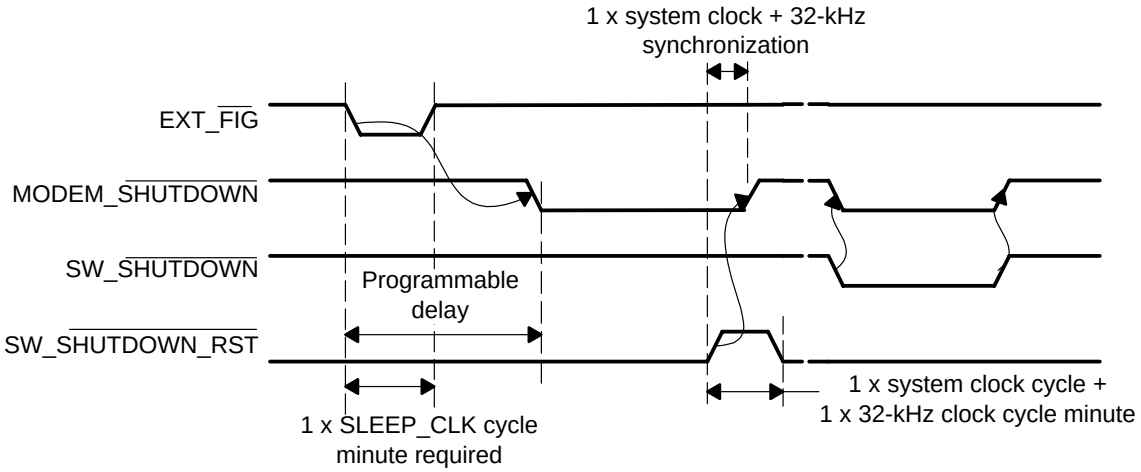
One of the three situations to initiate modem shutdown is during power-on reset. After releasing NPORE, MODEM_NSHUTDOWN is kept low for two CLK32K cycles and then goes inactive high.

Figure 17–5. Modem Shutdown Activation on NPORE



The second situation is when receiving an external FIQ. The delay from the falling edge of EXT_NFIQ to the falling edge of MODEM_NSHUTDOWN is software programmable (see COUNTER_32_FIQ_REG). The default value is one CLK32K cycle. The release of the shutdown signal is also controlled by software (see POWER_CTRL_REG[2]).

Figure 17–6. Modem Shutdown Activation on EXT_NFIQ and SW_NS

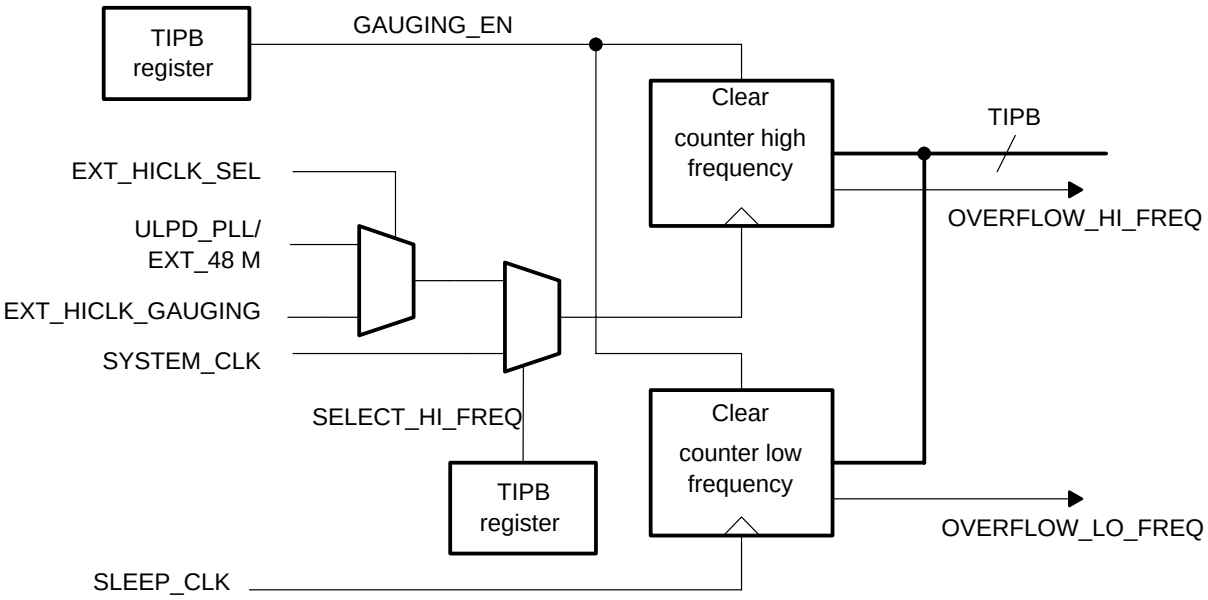


The third situation is to use SW_NSHUTDOWN from the register bit to toggle the MODEM_NSHUTDOWN state to low.

Figure 17–6 shows MODEM_NSHUTDOWN activation on EXT_NFIQ and SW_NSHUTDOWN as the programmable delay corresponds to the counter start time (3 x 32-kHz cycles) plus the counter decrement to 0. The counter initial value corresponds to COUNTER_32_FIQ_REG.

17.2.1.6 32-kHz Oscillator Calibration

Figure 17–7. 32-kHz Gauging



Because the exact frequency of the CLK32K clock is unknown, it is necessary to gauge it by comparing the CLK32K clock with a higher frequency clock (system clock, PLL clock out, or EXT_HICLK_GAUGING) during any active period.

Software limitation: The counter is not resynchronized on the TIPB strobe. Therefore, the value is not readable while the counter is running (when gauging is enabled). The correct procedure is first to disable the gauging (GAUGIG_CTRLREG[0] to 0) and then read the counter high frequency and 32-kHz counter value.

17.2.1.7 Gauging Versus High Frequency Clock

To gauge the 32-kHz clock, two counters clocked on the 32-kHz clock and a high-frequency clock, respectively, run concurrently during the gauging period. The high-frequency clock is selected among the 12-MHz clock, DPLL, and external clock. At the end of the gauging period, the numbers of the 32-kHz clock and the numbers of the high-frequency clock are calculated by:

Nb_32kHz = counter_32_msb * 65536 + counter_32_lsb

with : counter_32_msb is the MSB value of 32kHz counter

counter_32_lsb is the LSB value of 32kHz counter

Nb_hi_freq = counter_hi_freq_msb * 65536 + counter_hi_freq_lsb

with: counter_hi_freq_msb is the MSB value of high freq counter

counter_hi_freq_lsb is the LSB value of high freq counter

To gauge the 32-kHz clock versus the high-frequency clock, perform the following sequence:

1) Select gauging versus the high-frequency clock.

   Write GAUGING_CTRL[0] = 0 to select gauging versus the 12-MHz clock
   or
   Write GAUGING_CTRL[0] = 1 to select gauging versus the external or DPLL clock.

2) Start gauging.

   Write GAUGING_CTRL[1] = 0 to start gauging.

   Wait for a time.

3) Stop gauging.

   Write GAUGING_CTRL[1] = 0 to stop gauging.

   On reception of the gauging interrupt (low level sensitive interrupt):

4) Check the overflow of the 32-kHz counter.

   Read IT_STATUS[2].

5) Check the overflow of the high-frequency counter.

   Read IT_STATUS[1].

6) If an overflow occurred during the process, then return to step 3 to restart the gauging and reduce the waiting time. Else, continue to the next step.

7) Read the 32-kHz counter value.

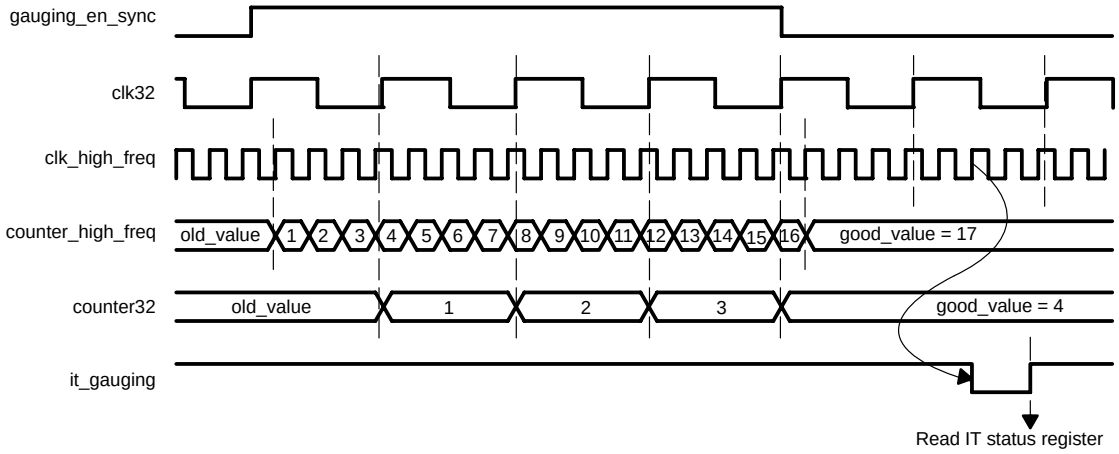
   Read the COUNTER_32_MSB and COUNTER_32_LSB registers.

8) Read the HI_FREQ counter value.

   Read the COUNTER_HI_FREQ_MSB and COUNTER_HI_FREQ_LSB registers.

9) Compare the values of the counter and proceed with calibration of the 32-kHz clock.

Figure 17–8. 32-kHz Clock Gauging



17.2.2 Power-Up Mechanism

The power-up sequence is as follows:

1) The real time clock (RTC) releases ON_nOFF and NRESPWRON_CORE resets and turns on CLK32K_CORE.

2) Two CLK32K cycles after the reset releases the PCC module, the wake-up sequence of the MPU subsystem starts:

   a) Assert RF_EN.
   b) Wait for a setup time.
   c) Assert VTCXO_EN.
   d) Wait for a setup time.
   e) Assert SLICER_EN.
   f) Wait for a setup time.
   g) OMAP input clock starts an OMAP reset sequence.

When the MPU starts its boot sequence, it can program the PCC module to use the APLL 13-MHz clock instead of the slicer. In that case, it enables the APLL 13-MHz clock with the PCC_CTRL_REG[2] and commands the switching with the PCC_CTRL_REG[0]. Switching is done when the APLL 13-MHz clock is locked.

The power-up latency is between 30 ms and 255 ms, depending on the reset value of the different setups.

The reset value latency is split as follows:

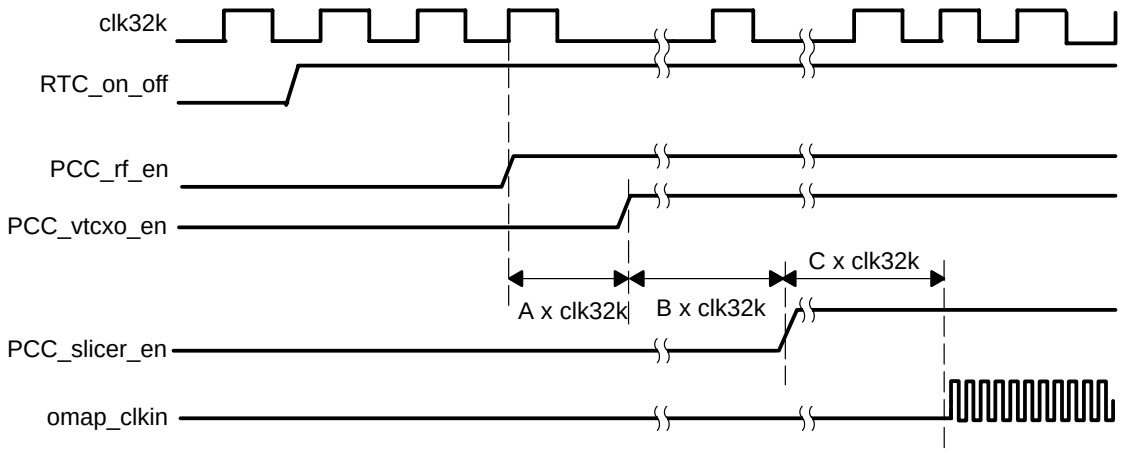
- 5 ms for RF setup
- 20 ms for VTCXO setup
- 5 ms for slicer setup

The maximum latency is split as follows:

- 125 ms for RF setup
- 125 ms for VTCXO setup
- 5 ms for slicer setup

Figure 17–9 shows the PCC module power-up scheme.

Figure 17–9. PCC Module Power-Up Scheme



17.2.3 Wake-Up Mechanism

The PCC module manages two wake-up mechanisms:

- The first wakeup mechanism behaves the same as the power-up scheme (see Section 17.2.2).
- The second is the fast-wake-up mechanism. It is implemented to allow the deep-sleep-to-awake transition with only the APLL 13-MHz clock on. To use this mechanism, the software programs the configuration register PCC_CTRL_REG[2] of the power and clock control module to keep the APLL13-MHz clock on throughout the deep sleep state. This configuration allows setting all of the setup analog cell registers to their minimum values.

For the wake-up mechanism, the MPU RF_EN assertion can be disabled with the PCC_POWER_CTRL_REG[7]. In that case, the RF_SETUP value can be programmed to 0.

The wake-up latency depends on the configuration.

Wake-up with the slicer resets the value for all of the setups:

- 5 ms latency for RF setup
- 20 ms latency for VTCXO setup
- 5 ms latency for slicer setup

Wake-up with the APLL13-MHz clock off during sleep:

- 50 ms latency for APLL13-MHz stabilization time

Wake-up with APLL on during sleep:

- Twelve CLK32K cycles for deep-sleep-to-awake ULPD transition with all the setup counters programmed to 0. The DEEP_SLEEP_TRANSITION_ENABLE bit 4 of the POWER_CTRL_REG can also be used to stay in big sleep state. In that case, the wake-up time is four CLKC32K cycles.

The APLL13-MHz clock consumes 250 µA in this mode.

17.2.4 GSM-MPU Wake-Up Interaction

The objective is to minimize the wake-up time of the MPU when the DBB is awake for traffic controller access. The DBB constraint is that the traffic controller must be accessible 500 µs after a DBB traffic controller request.

This constraint requires that the MPU ultralow-power device be placed in big sleep state, if it was in deep sleep state before the DBB wake-up. The DBB_RF_EN signal is used for the transition from deep sleep to big sleep.

If the DBB requires the traffic controller:

1) It asserts the traffic controller request.
2) OMAP asserts its wake-up request to the MPU ultralow-power device. This assertion allows the big-sleep-to-awake transition of the ULPD and

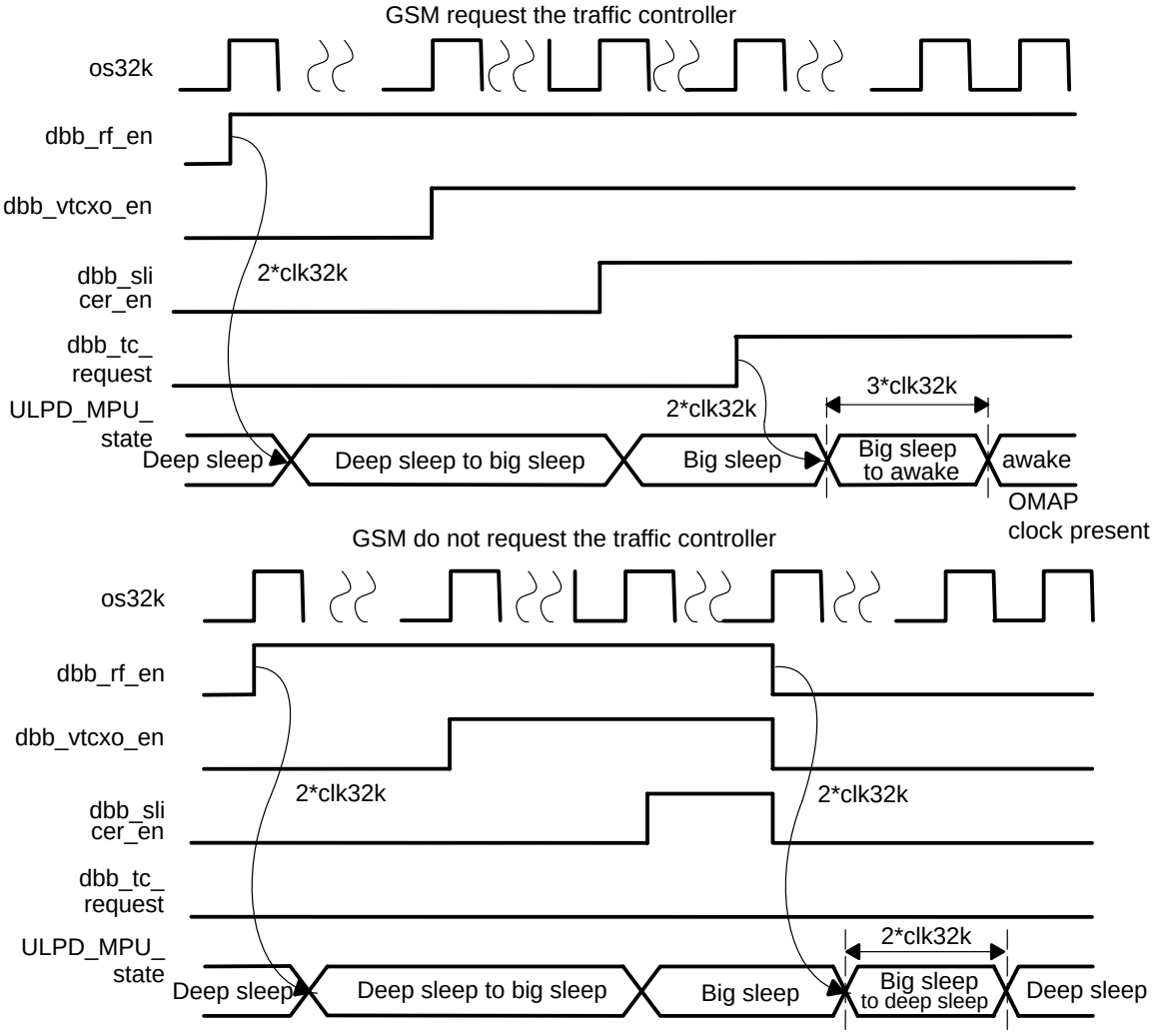
provides the OMAP clock a few CLK32K cycles after the OMAP wake-up assertion.

If the MPU uses the slicer during its wake up, the software programs the same setup value in GSM and MPU ultralow-power devices.

If the MPU uses the APLL 13-MHz clock, two configurations can result:

- APLL is always on. In this case, all the setup registers of the MPU ultralow-power device can be programmed to 0 to reach the big sleep state before the end of the GSM wake-up.
- APLL mode with APLL in power down when the MPU ultralow-power device is in deep sleep mode. In this case, the GSM wake-up sequence must be changed according to the APLL lock time to have the APLL locked when GSM enters the awake state. This means that if the APLL lock time is 40 ms, the GSM wake-up sequence must be equal to 40 ms.

Figure 17–10. GSM State and ULPD MPU Interaction



17.2.5 MPU and DBB Sleep Sequence

The sleep sequence takes place when OMAP asserts the chip-idle signal for cutting the entire MPU clock domain. The MPU domain sleep sequence is completely managed by the ULPD.

The power and clock control module does not affect the DBB system sleep sequence. The DBB sequence used is via the DBB ultralow-power device.

17.2.6 Low-Dropout Voltage (LDO) Management

17.2.6.1 OMAP LDO

The OMAP low-dropout voltage regulator is managed as follows:

- The LDO is off when the FSM is in deep sleep state; otherwise, it is on.
- PWRDN must be high while NRESPWRON is low.
- The OMAP730 PCC drives only the LDO sleep signal. The LDO PWRDN signal is driven by the PCC_POWER_CTRL_REG [0]. The reset value of this register is 0; setting it to 1 allows bypassing the LDO.

17.2.6.2 DBB LDO

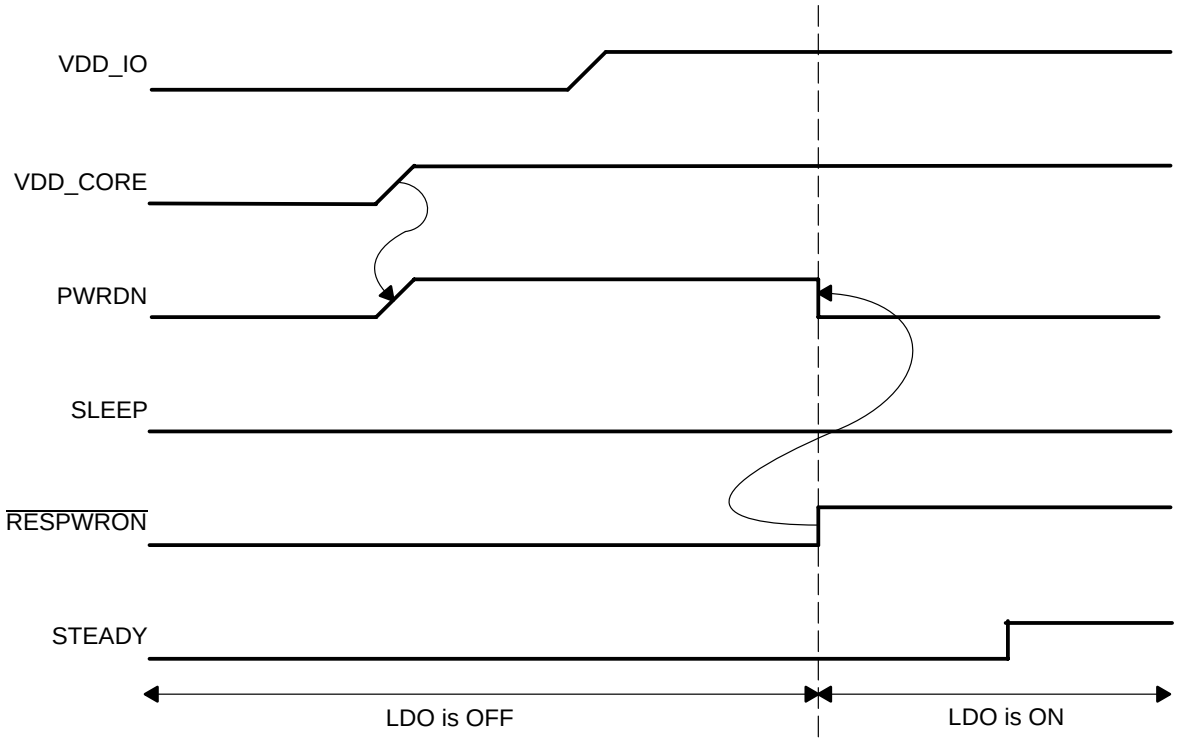
The DBB LDO is managed by the power and clock control module. The PWRDN signal is driven by the PCC_POWER_CTRL_REG [1]. The reset value of this register is 0. Setting this register to 1 allows bypassing the LDO.

The sleep signal is managed as follows:

- When the DBB is off, the LDO is in sleep mode; when the DBB is on, the LDO is in active mode. In this mode, DBB_VTCXO_EN is used to drive the LDO sleep signal.
- Software can also manage the sleep to optimize the LDO management.
- The PWRDN must be set to high during NRESPWRON low.

17.2.7 Timing Diagram

Figure 17–11. VDD_CORE Ramps First



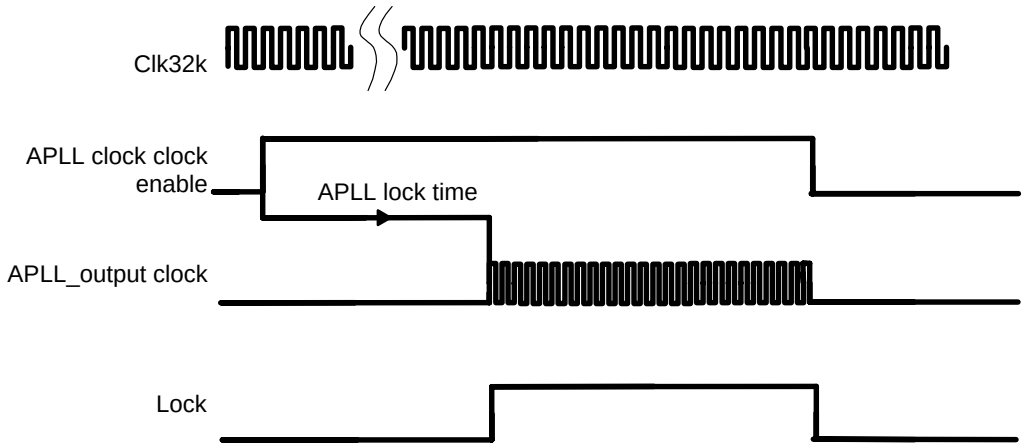
17.2.8 APLL 32-kHz to 13-MHz Wrapper Overview

This APLL 13-MHz **(**UC647**)** does not have a lock signal to indicate that the output clock is stable. Therefore**,** the PCC module generates a lock signal with a configurable time. The PCC_LOCK_TIME_REG allows the lock time configuration. It represents the number of CLK32K periods for the lock generation after the enable of the APLL 13-MHz input clock. This register allows programming the lock time from 31.25 µs to 2.048 s. The default value is 50 ms.

When the APLL 13-MHz clock is not selected, the wrapper automatically puts it in power-down mode.

The wrapper also manages the use of an external clock source to replace the APLL13-MHz clock. When the external clock source is selected, the APLL 13-MHz clock is put in power-down mode and the lock signal is set high, because it is assumed that the external clock is always on.

Figure 17–12. APLL 13-MHz Lock-Emulation Scheme



17.2.9 Clock Switching Conditions

17.2.9.1 PCC Module Input Clock Sources

The PCC module has two main clock sources: 32-kHz and 13-MHz clocks. The PCC module has two different sources for the 13-MHz clock: the slicer output and the APLL 13-MHz output. The switch condition depends on software configuration. By default, the MPU subsystem uses only the slicer output.

The slicer output may or may not be divided by 2, depending on the PCC_CTRL_REG[3] configuration bit: if 1, the output is divided; otherwise, it is not. This division is used only for the MPU subsystem because the DBB already implements this divider.

17.2.9.2 OMAP Input Clock Switch

The OMAP input clock source depends on the PCC_CTRL_REG [0]. When it is equal to 0 (reset value), OMAP receives the slicer output clocks; otherwise, it receives the APLL 13-MHz output clock.

17.2.10 APLL 96-MHz Input Clock Switch

Depending on the OMAP input clock sources, there can be two different behaviors:

- When OMAP uses the slicer output:

  There are no switching problems. All clock requests use the slicer output. The APLL 13-MHz clock is not turned on, except for the LCD low-power clock (LLPC) when OMAP is required to go into sleep mode.

- When OMAP uses the APLL 13-MHz clock for the input clock, two situations can occur**:**

  - First, an inaccurate clock request is asserted. This means that the APLL 96-MHz clock uses the APLL 13-MHz output for the input clock. The slicer remains off.

- Second, an accurate clock request is asserted. This means that the PCC module automatically wakes up the slicer and switches the APLL 96-MHz input clock from APLL 13 MHz to slicer output without switching the OMAP input clock.

In the second situation, the wake-up time depends on the DBB state. If the DBB is on, it means that the slicer is already on and the PCC module must switch the clock from the APLL 13-MHz clock to the slicer. Otherwise, the PCC module wakes up the slicer and the latency is16 ms.

Table 17–3 describes the latency for accurate and inaccurate peripheral clock requests.

Table 17–3. Peripheral Clock Request Latency

Latency for MPU in Inaccurate Mode	Accurate Clock Request	Inaccurate Clock Request
DBB is on.	APLL 96-MHz lock time + glitch-free switch	APLL 96-MHz lock time + glitch-free switch
DBB is off.	Slicer wake-up time+APLL 96-MHz lock time + glitch-free switch	APLL 96-MHz lock time + glitch-free switch
Latency for MPU in accurate mode	Accurate clock request	Inaccurate clock request
DBB state has no effect.	APLL 96-MHz lock time + glitch-free switch	APLL 96-MHz lock time + glitch-free switch

† APLL 96-MHz lock time: 200 µs
‡ Slicer wake-up time: 16 ms
§ Glitch-free switch: 2 clk13 m cycles

17.2.11 NIRQ Generation

The negative interrupt request (NIRQ) output of PCC is generated when one of the following unmasked events has been detected:

- ❑ APLL13 MHz is locked. When the PCC activates the APLL 13 MHz, when the lock is rising, the interruption is activated**.** This interrupt can be used instead of polling for the APLL13-MHz status.
- ❑ Slicer is re-enabled when in APLL mode.
- ❑ An overflow occurred on the 32-kHz counter during gauging. This is the same event as the one that triggers the IT_STATUS_REG[2] bit.
- ❑ An overflow occurred on the HI_FREQ counter during gauging versus high frequency. This is the same event as the one that triggers the IT_STATUS_REG[1] bit.
- ❑ Gauging has stopped. This is the same event as the one that triggers the IT_STATUS_REG[0] bit.
- ❑ The following two conditions have been met:
  - The USB_MCLK_REQ clock request has been asserted (high) or SOFT_REQ_REG[3] has been active (high) for more than two 32-kHz clock cycles (there is a synchronization mechanism on the 32-kHz clock).
  - CLOCK_CTRL_REG[5] is inactive (low).

Note:

When a USB_MCLK_REQ (or SOFT_REQ_REG[3]) is asserted, it wakes up the FSM to big sleep for fast wakeup (to awake) as soon as the wake-up request from OMAP is asserted. In the mean time, it generates a NIRQ pulse (active low during one 32-kHz cycle) to OMAP.

17.3 Peripherals Interface

17.3.1 PCC Peripheral Clock Constraints

17.3.1.1 USB_OTG Clock

The USB OTG subchip requires two different clocks:

- The open-core protocol (OCP) free-running clock, provided by OMAP via the ARMPERCLK clock
- The system clock, provided by the PCC via the ULPD

When the USB OTG requests the clock, the PCC module automatically switches the APLL 96-MHz input clock to the slicer output clock, because the USB requires a jitter-free clock. But if OMAP uses the APLL 13-MHz clock, it is not mandatory to switch this clock, unless requested by the software (PCC_clock_sel[0] = 0: slicer;1:APLL13-MHz).

The OCP clock must have a frequency within the 11-MHz to 60-MHz range. It is autogated inside the subchip when no OCP access is pending.

The frequency of the system clock must be 48 MHz.

The USB OTG system clock is mapped to the dedicated port of the ULPD.

17.3.1.2 MMC/SDIO Clock

The MMC/SDIO subchip requires two different clocks:

- The OCP free-running clock, provided by OMAP via the ARMPERCLK clock
- The system clock, provided by the PCC module via the ULPD

The frequency of the OCP clock must be in the 0-MHz to 100-MHz range. It is autogated inside the subchip when no OCP access is pending.

The system clock has a frequency of 48 MHz. The maximum frequency depends on the card type. For an MMC card, the maximum frequency is 20 MHz, but when using the internal prescaler of the MMC, the target is 16 MHz. In MMC mode, the subchip does not provide a hardware request for the system clock; therefore, the MMC mode requires a software request.

For an SDIO card, the maximum frequency of the system clock is 25 MHz. However, when using the internal prescaler of the subchip, the target is 24 MHz. In SDIO mode, the subchip provides a hardware request for the system clock.

Because the internal prescaler is used to change the frequency, the PCC merges the two requests into a single ULPD request.

Even if the MMC/SDIO does not need an accurate clock, the PCC implements a backup mechanism that allows software to select the input source of the APLL 96-MHz clock with the PCC_CLOCK_SEL_REG[1] (0: slicer; 1: APLL 13 MHz). This is the default hardware selection.

The MMC-SDIO clock is mapped to the MMC_DPLL port of the ULPD.

17.3.1.3 UART Clocks

The UART subchip requires two clocks:

- The OCP clock, provided by OMAP via the ARMPERCLK
- The system clock, provided by the PCC module

The PCC module uses the dedicated UART port to provide the system clock to each UART.

The OMAP730 has two UARTs:

- UART1 (UART modem)
- UART3 (UART modem/IrDA)

The request associated with each UART is issued from a register. When UART3_DPLL_REQ is activated, the PCC module automatically switches the input clock of the APLL 96 MHz to the slicer output, because UART3 is an IRDA UART and it requires an accurate clock. The APLL 13-MHz output clock can be used for the other UART as the input clock to APLL96MHz.

For the other UART, the software can select the APLL 96-MHz input clock source with PCC_CLOCK_SEL_REG[2] (0: slicer; 1: APLL 13-MHz output) for UART1. This must be the default hardware configuration.

For the UART IrDA (UART3), the PCC also allows selection of the APLL 96-MHz input clock via configuration register PCC_CLOCK_SEL_REG[3] (0: slicer; 1: APLL 13-MHz output).

UART1 and UART3 are mapped to their dedicated ULPD ports.

17.3.1.4 Enhanced Audio Controller (EAC) Clock

The EAC can have one of two clock frequencies: 12 MHz or 13 MHz. Both clocks require an accurate clock. The PCC module implements a selection mechanism between the slicer and the APLL 13 MHz to allow the selection. By default, the slicer output is selected.

The behavior is as follows:

- Default mode (slicer selected):

1) Software activates a request.

2) If required, the PCC module enables the slicer and waits for the slicer clock-out stable.

3) The PCC module switches the input clock of the APLL 96 MHz from the APLL 13 MHz to the slicer output if required and degates the PCC_EAC_13M_CLK, if EAC_13M_REQ is active.

4) When the APLL 96 MHz is locked, the PCC module provides the 12-MHz PCC_EAC_12M_CLK via the divide-by-4 module, if the request is activated.

- Inaccurate mode:

1) Software activates a request.

2) If required, the PCC enables the APLL 13 MHz clock and waits for its clock output to become stable (lock high).

3) The PCC switches the input clock of the APLL 96 MHz from slicer output to APLL 13 MHz, if required, and degates PCC_EAC_13M_CLK if EAC_13M_REQ is active.

4) When the APLL 96 MHz is locked, the PCC module provides the 12-MHz PCC_EAC_12M_CLK via the divide-by-4 module, if the request is activated.

The selection between the different modes is done with the PCC_CLOCK_SEL_REG[4] (0: slicer; 1: APLL 13-MHz output).

17.3.1.5 Camera Interface Clock

The camera interface requires one clock:

- The PCC_CAM_SENSOR_CLK, which has a frequency of 48 MHz

The clock is generated on the software request PCC_CAM_REQ.

By default, the clock uses the inaccurate mode, but software can select the clock source (slicer output or APLL 13-MHz output). The selection can be done in the PCC_CLOCK_SEL_REG[5] (0: slicer; 1: APLL 13-MHz output)

The camera clock request can be enabled/disabled with the SOFT_REQ_REG[7] bit.

17.3.1.6 McBSP Clock

The McBSP subchip requires two clocks:

- The OCP clock, with a maximum frequency of 100 MHz
- The system clock, with a maximum frequency of 200 MHz. One constraint is to have the system clock after the prescaler < OCP_CLK/2.

MCBSP1 is mapped to the COM_MCKO clock of the ULPD. This clock allows either 13 MHz or 48-MHz/N, with N being a programmable value in the COM_CLK_DIV_CTRL_SEL register. The source clock could be either the slicer out or the APLL 13-MHz out. The selection is done in the PCC_CLOCK_SEL_REG[6] (0: slicer; 1: APLL 13-MHz output).

MCBSP2 is mapped to the SDW_MCKO clock of the ULPD OMAP1610. This clock allows either 13 MHz or 48 MHz/N, with N being a programmable value in the SDW_CLK_DIV_CTRL_SEL register. The source clock could be either the slicer out or the APLL 13-MHz out. The selection is done in the PCC_CLOCK_SEL_REG[7] (0: slicer; 1: APLL 13-MHz output).

17.3.1.7 TWL3016 Peripheral Clock

The TWL3016 clock is not mapped to an OMAP730 ULPD port, but it is managed by the PCC module.

When the DBB is on:

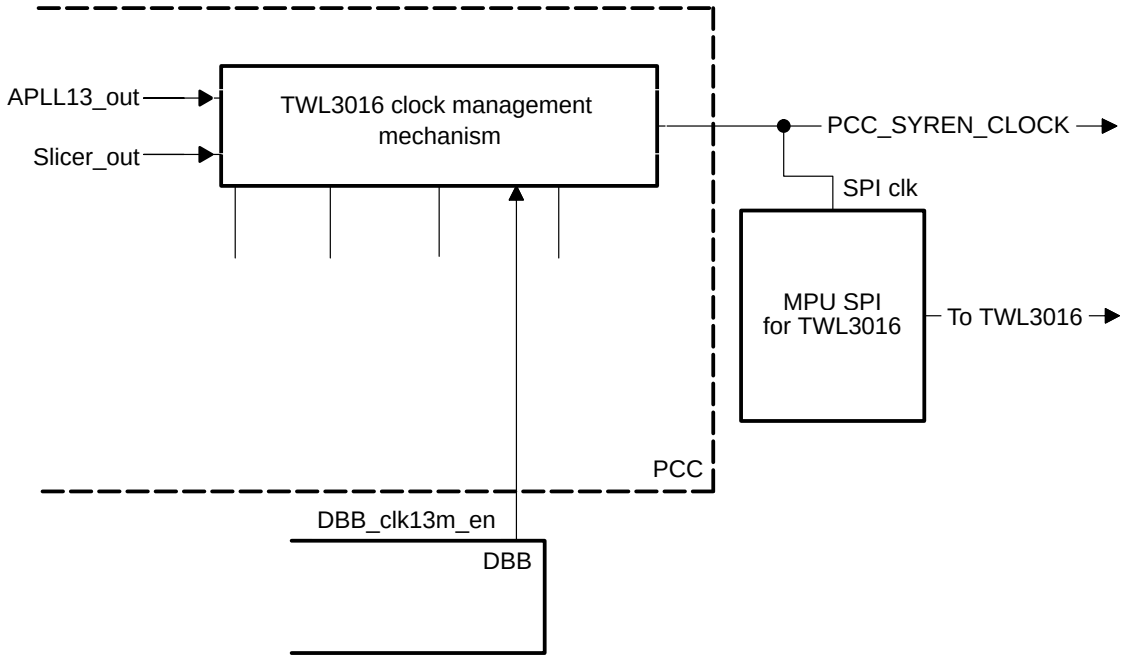
1) TWL3016 or external peripherals receive the slicer clock according to the DBB_CLK13M_EN, independently of the TAS value.
2) The TAS informs whether the MPU wants to communicate with TWL3016 via its own SPI.
3) Before communicating with TWL3016, the software must check the TAS status in the PCC status register.

When the DBB is off:

1) TWL3016 or external peripherals receive a clock if the MPU requests it by software using the TAS control bit from ICR, or if the external clock request is activated.
2) When the slicer is selected, the TWL3016 clock scheme stays the same as when DBB is on.
3) When the MPU selects the APLL, the clock is switched. The MPU can also select the source clock by software using the PCC_PERIPH_CLOCK_SRC_SEL[9] bit.

Figure 17–13 illustrates the system clock scheme.

Figure 17–13. TWL3016 Clock Scheme



The TAS activity is as follows: TAS = 0: GSM SPI is selected; TAS = 1: MPU SPI is selected.

- DBB awake to DBB sleep transition:
  - If an EXT_CLK_REQ or the TAS is on and the clock source is the slicer, the TWL3016 clock stays on the same clock scheme.
  - If an EXT_CLK_REQ or the TAS is on and the clock source is the APLL 13 MHz, the clock is switched when the DBB internally gates its own clock.
  - If no TAS or EXT_CLK_REQ is on, the clock is turned off.
- DBB sleep to DBB awake transition:
  - If an EXT_CLK_REQ or the TAS is on and the clock source is the slicer, the TWL3016 clock stays on the same clock scheme.
  - If an EXT_CLK_REQ or the TAS is on and the clock source is APLL 13 MHz, the clock is switched when the DBB internally activates its own clock.
  - If no TAS or EXT_CLK_REQ is on, the clock is turned on according to the DBB CLK13M_EN.

17.3.1.8 LCD Low-Power Clock (LLPC)

The LCD low-power block requires a 13-MHz clock (active or not) and a 32-kHz free-running clock. Because this clock can be provided when the MPU is in sleep state, software must set the APLL 13-MHz lock time (maximum 50 ms) before entering the sleep state. If the MPU enters the sleep state after activating the LCD low-power clock request, and if the APLL 13-MHz clock is off before this request, the LLPC can be without a clock for 50 ms (maximum APLL lock time).

The software sequence for activating the LLPC clock is:

1) Activate the LLPC mode3 to activate the hardware request.
2) Wait for the APLL 13-MHz lock by polling the PCC_APLL_LOCK_REGISTER[15] or by waiting an PCC interrupt for a lock rising event.
3) Go into sleep mode.

Current consumption when using the 13-MHz clock is a maximum of 250 µA.

17.3.1.9 MPU Shared SPI Clock

The MPU shared SPI functional clock is the TWL3016 clock.

17.4 PCC to ULPD Look-Up Tables

17.4.1 Registers

Table 17–4 shows only those registers that need clarification.

Table 17–4. Register Look-Up Table

ULPD Name	Address Offset	PCC Function
SETUP_ANALOG_CELL3_ULPD1_REG	0x24	RF_EN setup
SETUP_ANALOG_CELL2_ULPD1_REG	0x28	VTCXO setup
SETUP_ANALOG_CELL1_ULPD1_REG	0x2C	Slicer setup
ULPD_PLL_CTRL_STATUS	0x4C	APLL96MHZ_CTRL_REG
SDW_CLK_DIV_CTRL_SEL	0x74	MCBSP1 clock control register
COM_CLK_DIV_CTRL_SEL	0x78	MCBSP2 clock control register

17.4.2 Software-Request

Table 17–5 lists the software requests and their respective register bits.

Table 17–5. Software Request Look-Up Table

PCC Software Request Name	Frequency	ULPD Register and Bit to Use
EAC_12M_REQ	12 MHz	SOFT_REQ_REG[14]
EAC_13M_REQ	13 MHz	CAM_CLK_CTRL[2]
MMC_SDIO_REQ	48 MHz	SOFT_REQ_REG[12]
UART3_SOFT_REQ	48 MHz	SOFT_REQ_REG[11]
UART1_SOFT_REQ	48 MHz	SOFT_REQ_REG[9]
USB_OTG_SOFT_REQ	48 MHz	SOFT_REQ_REG[8]
CAM_IF_SOFT_REQ	48 MHz	SOFT_REQ_REG[7]
MCBSP1_SOFT_REQ	13 MHz	SOFT_REQ_REG[2] and put 1 in SDW_CLK_DIV_CTRL_SEL[0]
MCBSP1_SOFT_REQ	48 MHz	SDW_CLK_DIV_CTRL_SEL[1] and put 0 in SDW_CLK_DIV_CTRL_SEL[0]
MCBSP2_SOFT_REQ	13 MHz	SOFT_REQ_REG[6] and put 1 in COM_CLK_DIV_CTRL_SEL[0]
MCBSP2_SOFT_REQ	48 MHz	COM_CLK_DIV_CTRL_SEL[1] and put 0 in COM_CLK_DIV_CTRL_SEL[0]

17.5 Generated PCC Clocks

Table 17–6 lists the distributed PCC clocks.

Table 17–6. Distributed PCC Clocks

Name	Source	Clock Source Selection	Gated	Enable Signal	Wake-up Request	Clock State (for each FSM1 State)		
						Deep Sleep	Big Sleep	Awake
CLK32K (32 kHz)	Osc 32 kHz from RTC	None	No			32 kHz	32 kHz	32 kHz
OMAP_CLKIN (13 MHz)	SYSTEM_CLOCK		Yes	CHIP_NWAKEUP (internal signal telling that the FSM is in AWAKE state)	NPORE MPUNRST EXTERNAL_RESET_SOURCE(1,2,3) WAKEUP_NREQ PERIPH_NREQ (can be disabled by SOFT_DISABLE_REQ_REG[3])	0	0	SYSTEM_CLOCK
MCBPS1_PLL_CLK (13 MHz – 48 MHz)	SYSTEM_CLOCK APLL	SDW_SYSCLK_PLLCLK_SEL (SDW_CLK_DIV_CTRL_SEL[0]) selects between (when 0) clock derived from APLL_96M_CLK and (when 1) SYSTEM_CLOCK inverted or not (see below) SDW_MCLK_INV (CLOCK_CTRL_REG[2]) selects between system clock and inverted system clock	Yes	SDW hard request (SDW_MCLK_REQ) (this request is related to system clock) SDW system clock soft request (SOFT_REQ_REG[2]) SDW APLL_96M_CLK clock soft request (SDW_CLK_DIV_CTRL_SEL[1] is a request related to APLL_96M_CLK divided clock) SDW soft invert (CLOCK_CTRL_REG[2])	SDW_MCLK_REQ (can be disabled by SOFT_DISABLE_REQ_REG[1]) SOFT_REQ_REG[2] (this is the request for the system clock) SDW_CLK_DIV_CTRL_SEL[1] (this is the request for the clock derived from APLL_96M_CLK)	0 or 1 (when SDW_SYSCLK_PLLCLK_SEL =1, it depends on CLOCK_CTRL_REG[2]) When SDW_SYSCLK_PLLCLK_SEL = 0, state is 0	Clock not requested => same as in deep sleep	

Table 17–6. Distributed PCC Clocks (Continued)

Name	Source	Clock Source Selection	Gated	Enable Signal	Wake-up Request	Clock State (for each FSM1 State)		
						Deep Sleep	Big Sleep	Awake
							Clock requested =>SYSTEM_CLOCK (inverted or not) or APLL_96M_CLK/2 (divided by 1,1.5,2,2.5,3,3.5,4,5,6…47 or 48) Division factor is software programmable by SDW_CLK_DIV_CTRL_SEL[7:2])	
MCBSP2_PLL_CLK (13 MHz – 48 MHz)	SYSTEM_CLOCK APLL EXT_48M	COM_SYSCLK_PLLCLK_SEL (COM_CLK_DIV_CTRL_SEL[0] selects between (when 0) clock derived from APLL_96M_CLK and (when 1) system clock inverted or not or 48 MHz (see below). COM_MCKO_SEL (when 1, 48-MHz clock (EXT_48M when PLL_NCLKEXT_SEL= 0 or APLL_96M_CLK/2 when PLL_NCLKEXT_SEL=1), when 0 SYSTEM_CLOCK (inverted or not, see below)) CLOCK_CTRL_REG[1] (selects between SYSTEM_CLOCK and inverted system clock)	Yes	COM hard request (COM_MCLK_REQ) (this is the request for the system clock) COM system/48-MHz soft request (SOFT_REQ_REG[1]) COM APLL clock soft request (COM_CLK_DIV_CTRL_SEL[1]; this request is related to APLL divided clock) COM soft invert (CLOCK_CTRL_REG[1]) COM_MCKO_SEL	COM_MCLK_REQ (can be disabled by SOFT_DISABLE_REQ_REG[0]) SOFT_REQ_REG[6] (this request is for the system clock or 48 MHz) COM_CLK_DIV_CTRL_SEL[1] (this request is related to APLL divided clock) COM_MCKO_SEL (PLL clock request) can be disabled by SOFT_DISABLE_REQ_REG[0]	0 or 1 (when COM_MCKO_SEL = 0, it depends on CLOCK_CTRL_REG[1]) When COM_MCKO_SEL = 1, state is 0.	Clock not requested => same as in deep sleep	

Table 17–6. Distributed PCC Clocks (Continued)

Name	Source	Clock Source Selection	Gated	Enable Signal	Wake-up Request	Clock State (for each FSM1 State)		
						Deep Sleep	Big Sleep	Awake
							Clock requested => SYSTEM_CLOCK (inverted or not) or 48 MHz (APLL_96M_CLK/2 or EXT_48M) or APLL_96M_CLK/2 (divided by 1,1.5,2,2.5,3,3.5,4,5,6…47 or 48) Division factor is software programmable by COM_CLK_DIV_CTRL_SEL[7:2])	
							Clock requested =>APLL_96M_ CLK/2 div by 8 or EXT_48M Div by 8	Clock requested => APLL_96M_CLK/2 div by 8 or EXT_48M Div by 8
							Clock requested =>SYSTEM_ CLOCK	Clock requested =>SYSTEM_ CLOCK
PLL_48M (48 MHz)	APLL EXT_48M	PLL_NCLKEXT_SEL (selects EXT_48M when 0 or APLL_96M_ CLK/2 when 1)	No		SOFT_DPLL_REQ (SOFT_REQ_REG[0]) or any other hard or soft clock request related to APLL (listed in this table and not disabled) When PLL_NCLKEXT_ SEL = 0, DPLL_48M has the same behavior as EXT_48M.	0 (if PLL_ NCLKEXT_SEL = 1)	Clock not requested (and PLL_NCLKEXT_SEL = 1) =>0	
						EXT_48M (if PLL_ NCLKEXT_SEL = 0)	Clock requested (or PLL_NCLKEXT_ SEL = 0) =>APLL_96M_CLK/2 or EXT_48M	

Table 17–6. Distributed PCC Clocks (Continued)

Name	Source	Clock Source Selection	Gated	Enable Signal	Wake-up Request	Clock State (for each FSM1 State)		
						Deep Sleep	Big Sleep	Awake
UART1_PLL_CLK (48 MHz)	APLL EXT_48M	PLL_NCLKEXT_SEL (selects EXT_48M when 0 or APLL_96M_CLK/2 when 1)	Yes	UART1 PLL hard request (UART1_DPLL_REQ) SOFT_UART1_DPLL_REQ (SOFT_REQ_REG[9])	UART1_DPLL_REQ (can be disabled by SOFT_DISABLE_REQ_REG[7]) SOFT_REQ_REG[9]	0	Clock not requested => 0	Clock not requested =>0
							Clock requested =>APLL_96M_CLK/2 or EXT_48M	Clock requested =>APLL_96M_CLK/2 or EXT_48M
							Clock requested => APLL_96M_CLK/2 or EXT_48M	Clock requested =>APLL_96M_CLK/2 or EXT_48M
UART3_PLL_CLK (48 MHz)	APLL EXT_48M	PLL_NCLKEXT_SEL (selects EXT_48M when 0 or APLL_96M_CLK/2 when 1)	Yes	UART3 PLL request (UART3_DPLL_REQ) SOFT_UART3_DPLL_REQ (SOFT_REQ_REG[11])	UART3_DPLL_REQ (can be disabled by SOFT_DISABLE_REQ_REG[9]) SOFT_UART3_DPLL_REQ (SOFT_REQ_REG[11])	0	Clock not requested =>0	Clock not requested =>0
							Clock requested => APLL_96M_CLK/2 or EXT_48M	Clock requested => APLL_96M_CLK/2 or EXT_48M
EAC12M_CLK (12 MHz)	APLL EXT_48M SYSTEM_CLK	PLL_NCLKEXT_SEL (selects EXT_48M/4 when 0 or APLL_96M_CLK/8 when 1)	Yes	ClOCK1 PLL hard request (CLOCK1_DPLL_REQ) SOFT_CLOCK1_DPLL_REQ (SOFT_REQ_REG[14])	CLOCK1_DPLL_REQ (can be disabled by SOFT_DISABLE_REQ_REG[12]) SOFT_REQ_REG[14]	0	Clock not requested =>0 APLL_96M_CLK/8 or EXT_48M/4	Clock not requested =>0 APLL_96M_CLK/8 or EXT_48M

Table 17–6. Distributed PCC Clocks (Continued)

Name	Source	Clock Source Selection	Gated	Enable Signal	Wake-up Request	Clock State (for each FSM1 State)		
						Deep Sleep	Big Sleep	Awake
							Clock requested =>APLL_96M_CLK/8 or EXT_48M/4	Clock requested =>APLL_96M_CLK/8 or EXT_48M/4
							Clock requested =>APLL_96M_CLK/2 or EXT_48M	Clock requested =>APLL_96M_CLK/2 or EXT_48M
							Clock requested =>APLL_96M_CLK/2 or EXT_48M	Clock requested =>APLL_96M_CLK/2 or EXT_48M
CAM_MCKO (48 MHz)	APLL EXT_48M	PLL_NCLKEXT_SEL (selects EXT_48M when 0 or APLL_96M_CLK/2 when 1)	Yes	Camera PLL request (CAM_DPLL_MCLK_REQ) SOFT_CAM_DPLL_MCKO_REQ (SOFT_REQ_REG[7])	CAM_DPLL_MCLK_REQ (can be disabled by SOFT_DISABLE_REQ_REG[5]) SOFT_REQ_REG[7]	0	Clock not requested	Clock not requested
							Clock requested =>APLL_96M_CLK/2 or EXT_48M	Clock requested =>APLL_96M_CLK/2 or EXT_48M
							Clock requested =>MPU_PER_CLK (is not present in this mode)	Clock requested =>MPU_PER_CLK

Table 17–6. Distributed PCC Clocks (Continued)

Name	Source	Clock Source Selection	Gated	Enable Signal	Wake-up Request	Clock State (for each FSM1 State)		
						Deep Sleep	Big Sleep	Awake
USB_OTG_ MCKO (48 MHz)	APLL EXT_48M	PLL_NCLKEXT_SEL (selects EXT_48M when 0 or APLL_96M_CLK/2 when 1)	Yes	USB PLL request (USB_DPLL_MCLK_ REQ) USB_HOST_DPLL_REQ SOFT_USB_OTG_ DPLL_REQ (SOFT_REQ_REG[8])	USB_DPLL_MCLK_REQ (can be disabled by SOFT_REQ_REG[4]) USB_HOST_DPLL_REQ (can be disabled by SOFT_ DISABLE_REQ_REG[6]) SOFT_REQ_REG[8]	0	Clock not requested or disabled by software =>0	
							Clock requested => APLL_96M_ CLK/2 or EXT_48M	Clock requested =>APLL_96M_ CLK/2 or EXT_48M
MMC_DPLL_ CLK (48 MHz)	APLL EXT_48M	PLL_NCLKEXT_SEL (selects EXT_48M when 0 or APLL_96M_CLK/2 when 1)	Yes	MMC_DPLL_REQ SOFT_MMC_DPLL_ REQ (SOFT_REQ_REG[12])	MMC_DPLL_REQ (can be disabled by SOFT_ DISABLE_REQ_REG[10]) SOFT_REQ_REG[12]	0	Clock not re-quested =>0	Clock not requested =>0
							Clock requested => APLL_96M_ CLK/2 or EXT_48M	Clock requested => APLL_96M_CLK/2 or EXT_48M

Table 17–6. Distributed PCC Clocks (Continued)

Name	Source	Clock Source Selection	Gated	Enable Signal	Wake-up Request	Clock State (for each FSM1 State)		
						Deep Sleep	Big Sleep	Awake
							Clock requested => APLL_96M_ CLK/2 or EXT_48M	Clock requested => APLL_96M_CLK/2 or EXT_48M
EAC_13M_ CLK (13 MHz)	SYSTEM_ CLOCK		Yes	Software enable (CAM_CLK_CTRL[2])	WAKEUP_NREQ PERIPH_NREQ (can be disabled by SOFT DISABLE_REQ_REG[3]) NPORE MPUNRST EXTERNAL_RESET_ SOURCE(1,2,3)	0	0	System clock

17.6 PCC ULPD Registers

Table 17–7 lists the 16-bit PCC ULPD registers. Table 17–8 through Table 17–47 describe the register bits.

Table 17–7. PCC ULPD Registers

Register	Description	R/W	Offset
COUNTER_32_LSB_REG	32-kHz clock low	R	0x000
COUNTER_32_MSB_REG	32-kHz clock high	R	0x004
COUNTER_HIGH_FREQ_LSB_REG	High-frequency clock LSB	R	0x008
COUNTER_HIGH_FREQ_MSB_REG	High-frequency clock MSB	R	0x00C
GAUGING_CTRL_REG	Gauging control	R/W	0x010
IT_STATUS_REG	Interrupt status	R	0x014
RESERVED	Reserved	R	0x018
RESERVED1	Reserved	R	0x01C
RESERVED2	Reserved	R	0x020
SLICER_SETUP	Slicer setup	R/W	0x024
VCTXO_SETUP	VCTXO setup	R/W	0x028
RF_SETUP	RF setup	R/W	0x02C
CLOCK_CTRL_REG	Clock control	R/W	0x030
SOFT_REQ_REG	Software request	R/W	0x034
COUNTER_32_FIQ_REG	32-kHz counter FIQ	R/W	0x038
RESERVED3	Reserved	R	0x03C
STATUS_REQ_REG	Status request	R	0x040
PLL_DIV_REG	PLL divisor	R/W	0x044
RESERVED4	Reserved	R	0x048
ULPD_PLL_CTRL_STATUS	ULPD PLL control status	R/W	0x04C
POWER_CTRL_REG	Power control	R/W	0x050
STATUS_REQ_REG2	Status request 2	R	0x054
SLEEP_STATUS	Sleep status	R	0x058
RESERVED5	Reserved	R	0x05C
RESERVED6	Reserved	R	0x060
RESERVED7	Reserved	R	0x064
SOFT_DISABLE_REQ_REG	Software disable request	R/W	0x068
RESET_STATUS	Reset status	R/W	0x06C

Table 17–7. PCC ULPD Registers (Continued)

Register	Description	R/W	Offset
REVISION_NUMBER	Revision number	R	0x070
MCBSP2_CLK_DIV_CTRL_SEL	McBSP2 clock divisor control	R/W	0x074
MCBSP1_CLK_DIV_CTRL_SEL	McBSP1 clock divisor control	R/W	0x078
CAM_CLK_CTRL	CAM clock control	R/W	0x07C
RESERVED8	Reserved	R	0x080
PCC_CTRL_REG	PCC control	R/W	0x100
PCC_POWER_CTRL_REG	PCC power control	R/W	0x104
PCC_PERIPH_CLOCK_SOURCE_SEL	PCC peripheral clock source	R/W	0x108
PCC_APLL_LOCK_REGISTER	PCC APLL lock	R/W	0x10C
PCC_DBB_STATUS	PCC DBB status	R	0x120
PCC_IT_STATUS	PCC interrupt status	R/W	0x124
PCC_MASK_IT	PCC mask interrupt	R/W	0x128

Table 17–8. 32-kHz Clock Low Register (COUNTER_32_LSB_REG)

Bit	Name	Function	R/W	Reset Value
15:0	COUNTER_SLEEP_CLK_LSB	Lower value of the number of sleep clocks during gauging time	R	0x1

This register contains the lower value of the number of 32-kHz clocks during gauging time.

Table 17–9. 32-kHz Clock High Register (COUNTER_32_MSB_REG)

Bit	Name	Function	R/W	Reset Value
15:0	COUNTER_32_MSB	Upper value of the number of 32-kHz clocks during gauging time	R	0x0

This register contains the upper value of the number of 32-kHz clocks during gauging time.

Table 17–10. High Frequency Clock LSB Register (COUNTER_HIGH_FREQ_LSB_REG)

Bit	Name	Function	R/W	Reset Value
15:0	COUNTER_HIGH_FREQ_LSB	Lower value of the number of high-frequency clocks during gauging time	R	0x1

This register contains the lower value of high-frequency clocks during gauging time.

Table 17–11. High Frequency Clock MSB Register (COUNTER_HIGH_FREQ_MSB_REG)

Bit	Name	Function	R/W	Reset Value
15:6	UNUSED	Unused	R	0x0
5:0	COUNTER_HIGH_FREQ_MSB	Upper value of the number of high-frequency clocks during gauging time	R	0x0

This register contains the upper value of the number of high-frequency clocks during gauging time.

Table 17–12. Gauging Control Register (GAUGING_CTRL_REG)

Bit	Name	Function	R/W	Reset Value
15:2	UNUSED	Unused	R/W	0x0
1	SELECT_HI_FREQ_CLOCK	0: High-frequency clock = 12-MHz clock 1: High-frequency clock = auxiliary gauging clock	R/W	0x0
0	GAUGING_EN	0: Gauging is stopped. 1: Gauging is running.	R/W	0x0

GAUGING_CTRL_REG drives the gauging function.

Table 17–13. Interrupt Status Register (IT_STATUS_REG)

Bit	Name	Function	R/W	Reset Value
15:4	UNUSED	Unused	R	0x0
3	IT_WAKEUP_USB	Wake-up interrupt from USB W2FC	R	0x0
2	OVERFLOW_32	An overflow occurred on the 32-kHz counter during gauging.	R	0x0
1	OVERFLOW_HI_FREQ	An overflow occurred on the high-frequency counter versus the high-frequency clock during gauging.	R	0x0
0	IT_GAUGING	Informs CPU that gauging is stopped and high- and low-frequency registers can be read.	R	0x0

IT_STATUS_REG contains the interrupt status.

Table 17–14. Reserved Register (RESERVED)

Bit	Name	Function	R/W	Reset Value
15:0	RESERVED	Reserved	R	0x3FF

Table 17–15. Reserved Register 1 (RESERVED1)

Bit	Name	Function	R/W	Reset Value
15:0	RESERVED	Reserved	R	0x3FF

Table 17–16. Reserved Register 2 (RESERVED2)

Bit	Name	Function	R/W	Reset Value
15:0	RESERVED	Reserved	R	0x3FF

Table 17–17. Slicer Setup Register (SLICER_SETUP)

Bit	Name	Function	R/W	Reset Value
15:0	SLICER_SETUP	Setup time of internal slicer in number of sleep clock cycles	R/W	0xA0

Note: This setup stage is for the FSM1 of the ULPD.

The slicer setup register contains the setup time of internal slicer in number of sleep clock cycles.

Table 17–18. VTCXO Setup Register (VTCXO_SETUP_REG)

Bit	Name	Function	R/W	Reset Value
15:0	VTCXO_SETUP	Setup time of external VTCXO in number of sleep clock cycles	R/W	0x280

Note: This setup stage is for the FSM1 of the ULPD.

The VTCXO setup register contains the setup time of external VTCXO in number of sleep clock cycles.

Table 17–19. RF Setup Register (RF_SETUP)

Bit	Name	Function	R/W	Reset Value
15:0	RF_EN_SETUP	Setup time of RF_EN in number of sleep clock cycles	R/W	0xA0

Note: This setup stage is for the FSM1 of the ULPD.

The RF setup register contains the setup time of RF_EN in number of sleep clock cycle.

Table 17–20. Clock Control Register (CLOCK_CTRL_REG)

Bit	Name	Function	R/W	Reset Value
15:7	UNUSED	Unused	R/W	0x0
6	SLICER_BYPASS	Slicer control enable/bypass. The slicer is enabled by default. 0: Slicer enabled 1: Slicer bypassed	R/W	0x0
5	Reserved	–	R/W	0x0
4	Reserved	–	R/W	0x0

Table 17–20. Clock Control Register (CLOCK_CTRL_REG) (Continued)

Bit	Name	Function	R/W	Reset Value
3	Reserved	–	R/W	0x0
2	MCBSP2_MCLK_INV	0: MCBSP2 clock is low when inactive. 1: Clock is high when inactive.	R/W	0x0
1	MCBSP1_MCLK_INV	0: MCBSP1 clock is low when inactive. 1: Clock is high when inactive.	R/W	0x0
0	Reserved	–	R/W	0x0

The clock control register manages the clock output and inactive values.

Table 17–21. Software Request Register (SOFT_REQ_REG)

Bit	Name	Function	R/W	Reset Value
15	Reserved	–	R/W	0X0
14	EAC12M_DPLL_REQ	EAC12M software request. 0: Request inactive 1: Request active	R/W	0x0
13	Reserved	–	R/W	0x0
12	SOFT_MMC_DPLL_REQ	Software ULPD_PLL req for MMC 0: Request inactive 1: Request active	R/W	0x0
11	SOFT_UART3_DPLL_REQ	Software UART3 ULPD_PLL request 0: Request inactive 1: Request active	R/W	0x0
10	Reserved	–	R/W	0x0
9	SOFT_UART1_DPLL_REQ	Software UART1 ULPD_PLL request 0: Request inactive 1: Request active	R/W	0x0
8	SOFT_USB_OTG_DPLL_REQ	Software request for USB OTG for ULPD_PLL clock. 0: Request inactive 1: Request active	R/W	0x0
7	SOFT_CAM_DPLL_MCKO_REQ	Software camera ULPD_PLL request 0: Request inactive 1: Request active	R/W	0x0
6	SOFT_MCBSP1_REQ	Software request for MCBSP1 clock 0: Request inactive 1: Request active	R/W	0x0
5	Reserved	–	R/W	0x0

Table 17–21. Software Request Register (SOFT_REQ_REG) (Continued)

Bit	Name	Function	R/W	Reset Value
4	USB_REQ_EN	USB client hardware DPLL request 0: Disable 1: Enable	R/W	0x1
3	SOFT_USB_REQ	Software system clock request for USB host 0: Request inactive 1: Request active	R/W	0x0
2	SOFT_MCBSP2_REQ	Software system clock request for MCBSP2 0: Request inactive 1: Request active	R/W	0x0
1	Reserved	–	R/W	0x0
0	SOFT_DPLL_REQ	Software ULPD_PLL clock request 0: Request inactive 1: Request active	R/W	0x0

SOFT_REQ_REG manages software clock requests. All requests are active high.

Table 17–22. 32-kHz Counter FIQ Register (COUNTER_32_FIQ_REG)

Bit	Name	Function	R/W	Reset Value
15:8	UNUSED	Unused	R/W	0x0
7:0	COUNTER_32_FIQ	Number of 32-kHz clocks to delay active modem shut-down signal after receiving FIQ	R/W	0x01

The 32-kHz counter FIQ register contains the delay before modem shut down.

Table 17–23. Reserved Register 3 (RESERVED3)

Bit	Name	Function	R/W	Reset Value
15:0	RESERVED	Reserved	R	0x0

Table 17–24. Status Request Register (STATUS_REQ_REG)

Bit	Name	Function	R/W	Reset Value
15	CLOCK3_DPLL_REQ	ULPD_PLL clock request from request RESERVED3 1: Active 0: Inactive	R	Unknown
14	CLOCK2_DPLL_REQ	ULPD_PLL clock request from request RESERVED2 1: Active 0: Inactive	R	Unknown
13	CLOCK1_DPLL_REQ	ULPD_PLL clock request from request RESERVED1 1: Active 0: Inactive	R	Unknown

Table 17–24. Status Request Register (STATUS_REQ_REG) (Continued)

Bit	Name	Function	R/W	Reset Value
12	MMC_DPLL_REQ	ULPD_PLL clock request from MMC 1: Request active 0: Request inactive	R	Unknown
11	UART3_DPLL_REQ	ULPD_PLL clock request from UART3 1: Request active 0: Request inactive	R	Unknown
10	UART2_DPLL_REQ	ULPD_PLL clock request from UART2 1: Request active 0: Request inactive	R	Unknown
9	UART1_DPLL_REQ	ULPD_PLL clock request from UART1 1: Request active 0: Request inactive	R	Unknown
8	USB_HOST_DPLL_REQ	ULPD_PLL clock request from USB host. 1: Request active 0: Request inactive	R	Unknown
7	CAM_DPLL_MCLK_REQ	ULPD_PLL clock request from camera interface 1: Request active 0: Request inactive	R	Unknown
6	USB_DPLL_MCLK_REQ	ULPD_PLL clock request from USB client 1: Request active 0: Request inactive	R	Unknown
5	USB_MCLK_REQ	Hardware system clock request by USB client 1: Request active 0: Request inactive	R	Unknown
4	MCBSP2_MCLK_REQ	Hardware system clock request from MCBSP2 1: Request active 0: Request inactive	R	Unknown
3	MCBSP1_MCLK_REQ	Hardware system clock request from MCBSP1 1: Request active 0: Request inactive	R	Unknown
2	PERIPH_NREQ	System clock request from UART modem 0: Request active 1: Request inactive	R	Unknown
1	WAKEUP_NREQ	System clock request from OMAP 0: Request active 1: Request inactive	R	Unknown
0	CHIP_IDLE	Sleep request received from OMAP 1: Request active 0: Request inactive	R	Unknown

The status request register contains the status of hardware requests.

Table 17–25. PLL Divisor Register (PLL_DIV_REG)

Bit	Name	Function	R/W	Reset Value
15:0	PLL_DIV_FACTOR	PLL clock out division factor. Bypassed if programmed value is 0; else: 0001h: divide by 2 0002h: divide by 4 0004h: divide by 8 .... 8000h: divide by 65536	R/W	0x0

PLL_DIV_REG contains the PLL clock out division factor for monitoring. It allows choosing the division factor for the PMT_CLKO output clock. This divider is implemented for testing purposes.

Table 17–26. Reserved Register (RESERVED)

Bit	Name	Function	R/W	Reset Value
15:0	RESERVED	For software compatibility	R	0x960

Table 17–27. ULPD PLL Control Status Register (ULPD_PLL_CTRL_STATUS)

Bit	Name	Function	R/W	Reset Value
15	LOCK_STATUS	Gives the lock status of the module that provides the high-frequency clock. 1: PLL locked 0: PLL unlocked	R	Unknown
14:5	PLL_CTRL_RES	This register allows managing the ULPD_PLL. If the PLL needs more than three control bits, these bits can be used.	R/W	0x0
4	PWRDNZ	Control bit for the PWRDNZ input of the APLL 96-MHz UC684	R/W	0x1
3	TEST	Control bit for the TEST input of the APLL 96-MHz UC684	R/W	0x0
2	SEL2	Control bit for the SEL2 input of the APLL 96-MHz UC684	R/W	0x0
1	SEL1	Control bit for the SEL1 input of the APLL 96-MHz UC684	R/W	0x1
0	SEL0	Control bit for the SEL0 input of the APLL 96-MHz UC684	R/W	0x0

Note: The reset value depends on the project and the external module used. The 16 bits of this register are at the ULPD entity.

ULPD_PLL_CTRL_STATUS allows control of the external module that generates the high-frequency clock.

Table 17–28. Power Control Register (POWER_CTRL_REG)

Bit	Name	Function	R/W	Reset Value
13:15	UNUSED	Unused	R	0x0
12	Reserved	–	R/W	0x0
11	MIN_MAX_REG	0: Operation at minimum voltage 1: Operation at nominal voltage	R/W	0x1
10	DVS_ENABLE	0: DVS feature disabled 1: DVS feature enabled	R/W	0x0
9	OSC_STOP_EN	0: The oscillator is not stopped in deep-sleep mode. 1: The oscillator is stopped in deep-sleep mode.	R/W	0x1
8	LDO_PWRDOWN	Controls the power-up/down of the e-LDO that supplies the DPLL 1: LDO power down is forced to active state. 0: LDO is powered up (except in deep-sleep mode if LDO_CTRL_EN=1).	R/W	0x0
7	LDO_CTRL_EN	0: The activation of the LDO is fully controlled by LDO_PWRDOWN bit. 1: The LDO is powered down in deep-sleep mode or if LDO_PWRDOWN=1.	R/W	0x0
6	LDO_STEADY	Stability of LDO output voltage status 0: LDO output voltage not stable 1: LDO output voltage stable	R	Unknown
5	UNUSED	Unused	R	0x0
4	DEEP_SLEEP_TRANSITION_EN	1: Transition to deep-sleep mode allowed 0: Transition to deep-sleep mode not allowed	R/W	0x1
3	SW_NSHUTDOWN	Software generation of MODEM_NSHUTDOWN. 0: MODEM_NSHUTDOWN active low 1: MODEM_NSHUTDOWN active high	R/W	0x1
2	SW_NSHUTDOWN_RST	Allows deactivation of the hardware MODEM_NSHUTDOWN when = 1	R/W	0x0

Table 17–28. Power Control Register (POWER_CTRL_REG) (Continued)

Bit	Name	Function	R/W	Reset Value
1	LOW_PWR_REQ	Low-power software request 0: Does not force the LOW_PWR signal to active state. 1: Forces LOW_PWR signal to active state.	R/W	0x0
0	LOW_PWR_EN	0: Low-power feature is disabled. 1: Low-power feature is available.	R/W	0x0

POWER_CTRL_REG is used for power control.

Table 17–29. Status Request Register 2 (STATUS_REQ_REG2)

Bit	Name	Function	R/W	Reset Value
15:1	UNUSED	Unused	R	0x0
0	MMC2_DPLL_REQ	Status of the MMC2_PLL_REQ. 0: Inactive 1: Active	R	Unknown

STATUS_REQ_REG2 provides the status of the hardware request.

Table 17–30. Sleep Status Register (SLEEP_STATUS)

Bit	Name	Function	R/W	Reset Value
15:2	UNUSED	Unused	R	0x0
1	BIG_SLEEP	This bit is asserted (1) when waking up from big-sleep. This bit is cleared (0) when the FSM goes out of awake mode.	R	Unknown
0	DEEP_SLEEP	This bit is asserted (1) when waking up from deep-sleep. This bit is cleared (0) when the FSM goes out of awake mode.	R	Unknown

Note: Bits 0 and 1 can be read as 1 in case the ULPD goes from deep-sleep to big-sleep before transitioning back to awake.

The sleep status register provides a way to track the ULPD state transitions.

Table 17–31. Reserved Register (RESERVED)

Bit	Name	Function	R/W	Reset Value
15:0	RESERVED	For software compatibility	R	0x960

Table 17–32. Reserved Register (RESERVED)

Bit	Name	Function	R/W	Reset Value
15:0	RESERVED	For software compatibility	R	0x960

Table 17–33. Reserved Register (RESERVED)

Bit	Name	Function	R/W	Reset Value
15:0	RESERVED	For software compatibility	R	0x960

Table 17–34. Software Disable Request Register (SOFT_DISABLE_REQ_REG)

Bit	Name	Function	R/W	Reset Value
15	UNUSED	Unused	R	0x0
14	RESERVED	–	R/W	0x0
13	RESERVED	–	R/W	0x0
12	DIS_EAC12M_DPLL_REQ	Disable for the EAC12M PLL hardware request. 0: Not disabled 1: Disable	R/W	0x0
11	RESERVED	–	R/W	0x0
10	DIS_MMC_DPLL_REQ	Disables the current hardware request if active 0: Not disabled 1: Disabled	R/W	0x0
9	DIS_UART3_DPLL_REQ	Disables hardware request for UART3 0: Not disabled 1: Disabled	R/W	0x0
8	RESERVED	–	R/W	0x0
7	DIS_UART1_DPLL_REQ	Disables UART1 PLL hardware request 0: Not disabled 1: Disabled	R/W	0x0
6	DIS_USB_HOST_DPLL_REQ	Disables the USB host system clock hardware 0: Not disabled 1: Disabled	R/W	0x0
5	DIS_CAM_DPLL_MCLK_REQ	Disables hardware CAM_PLL_MCLK_REQ 0: Not disabled 1: Disabled	R/W	0x0
4	UNUSED	Unused	R/W	0x0
3	RESERVED	–	R/W	0x0
2	UNUSED	Unused	R/W	0x0
1	DIS_MCBSP2_MCLK_REQ	Disables MCBSP2_MCLK_REQ if active 0: Not disabled 1: Disabled	R/W	0x0
0	DIS_MCBSP1_MCLK_REQ	Disables MCBSP1_MCLK_REQ if active 0: Not disabled 1: Disabled	R/W	0x0

The soft disable request register disables the hardware request, if required.

Table 17–35. Reset Status Register (RESET_STATUS)

Bit	Name	Function	R/W	Reset Value
15:4	UNUSED	Unused	R	0x0
3	EXTERNAL_RESET_SOURCE_3	This bit is asserted whenever an EXTERNAL_RESET_SOURCE3 (from GSM protect) event occurs. It is the users responsibility to clear this bit by writing a 0 to it. TIPB reset has no effect on this bit. 0: EXTERNAL_RESET_SOURCE3 has not occurred since last NPORE (or user has cleared this bit). 1: EXTERNAL_RESET_SOURCE3 has occurred since last NPORE (and since last time user cleared this bit).	R/W	0x0
2	EXTERNAL_RESET_SOURCE_2	This bit is asserted whenever an EXTERNAL_RESET_SOURCE2 (from OMAP security violation) event occurs. It is the users responsibility to clear this bit by writing a 0 to it. TIPB reset has no effect on this bit. 0: EXTERNAL_RESET_SOURCE2 has not occurred since last NPORE (or user has cleared this bit). 1: EXTERNAL_RESET_SOURCE2 has occurred since last NPORE (and since last time user cleared this bit).	R/W	0x0
1	EXTERNAL_RESET_SOURCE_1	This bit is asserted whenever an EXTERNAL_RESET_SOURCE1 (from secure watchdog timer in the secure block) event occurs. It is the users responsibility to clear this bit by writing a 0 to it. TIPB reset has no effect on this bit. 0: EXTERNAL_RESET_SOURCE1 has not occurred since last NPORE (or user has cleared this bit). 1: EXTERNAL_RESET_SOURCE1 has occurred since last NPORE (and since last time user has cleared this bit).	R/W	0x0
0	POWER_ON_RESET	This bit is asserted whenever an NPORE reset occurs. It is the users responsibility to clear this bit by writing a 0 to it. TIPB reset has no effect on this bit. 0: NPORE reset has not occurred (or user has cleared this bit). 1: NPORE reset occurred.	R/W	0x1

Note: This register is reset by the NPORE reset only and not by the TIPB reset (nbrst) as with the other registers.

The reset status register provides the status for the reset source of the ULPD.

Table 17–36. Revision Number Register (REVISION_NUMBER)

Bit	Name	Function	R/W	Reset Value
15:8	UNUSED	Unused	R	0x0
7:0	REVISION_NUMBER	Indicates the ULPD revision number. The revision number must be read as follows: [7:4].[3:0]. For example, 0x14 has to be read as revision 1.4.	R	0x11

Note: The 4 LSBs indicate a minor revision, and the 4 MSBs indicate a major revision.

REVISION_NUMBER is a read-only register that contains the revision number of the module.

Table 17–37. McBSP2 Clock Divisor Control Register (MCBSP2_CLK_DIV_CTRL_SEL)

Bit	Name	Function	R/W	Reset Value
15:8	UNUSED	Unused	R	0x0
7:2	MCBSP2_RATIO_SEL	Select the divider ratio to apply to ULPD_PLL_CLK to generate MCBSP2_MCKO 000000=>1; 000001=>1.5; 000010=>2; 000011=>2.5 000100=>3; 000101=>3.5; 000110=>4; 000111=>5; 001000=>6; 001001=>7; .... 010010=>16 ... 110010=>48. All the others must use 1 for the division factor.	R/W	0x0
1	MCBSP2_ULPD_PLL_CLK_REQ	MCBSP2_MCKO clock software request 0: Request inactive 1: Request active	R/W	0x0
0	MCBSP2_SYSCLK_PLLCLK_SEL	0: Selects the divided version of ULPD_PLL_CLK for MCBSP2 1: Selects SYSTEM_CLOCK for MCBSP2	R/W	0x1

The McBSP2 clock divisor control register allows selection of the division ratio for the MCBSP2_MCKO. It also allows selection of the system clock or ULPD PLL clock. The divider is applied only to the ULPD PLL clock.

Table 17–38. McBSP1 Clock Divisor Control Register (MCBSP1_CLK_DIV_CTRL_SEL)

Bit	Name	Function	R/W	Reset Value
15:8	UNUSED	Unused	R	0x0
7:2	MCBSP1_RATIO_SEL	Select the divider ratio to apply to the ULPD_PLL_CLK to generate MCBSP1 000000=>1; 000001=>1.5; 000010=>2; 000011=>2.5 000100=>3; 000101=>3.5; 000110=>4; 000111=>5; 001000=>6; 001001=>7; .... 010010=>16 ... 110010=>48. All of the others must use 1 for the division factor.	R/W	0x0
1	MCBSP1_ULPD_PLL_CLK_REQ	MCBSP1 clock software request 0: Request inactive 1: Request active	R/W	0x0
0	MCBSP1_SYSCLK_PLLCLK_SEL	0: Selects the divided version of ULPD_PLL_CLK for MCBSP1 1: Selects SYSTEM_CLOCK for MCBSP1	R/W	0x1

The McBSP1 clock divisor control register allows selection of the division ratio for the MCBSP1. It also allows selection of the system clock or ULPD PLL clock. The divider is applied only to the ULPD PLL clock.

Table 17–39. CAM Clock Control Register (CAM_CLK_CTRL)

Bit	Name	Function	R/W	Reset Value
15:3	UNUSED	Unused	R	0x0
2	SYSTEM_CLK_EN	Clock enable of the PO_CLK_SYSTEM_CLK output clock of ULPD. 0: Clock disabled 1: Clock enabled	R/W	0x0
1	CAM_CLK_DIV	0: The CAM_CLK is the system clock. 1: The CAM_CLK is the SYSTEM_CLOCK/2.	R/W	0x0
0	CAM_CLOCK_EN	Clock enable of the CAM_CLK. 0: CAM_CLK is off. 1: CAM_CLK is on.	R/W	0x0

The CAM clock control register allows enabling of the CAM_CLK and selection of the division factor.

Table 17–40. Reserved Register (RESERVED)

Bit	Name	Function	R/W	Reset Value
15:0	RESERVED	For software compatibility	R	0x960

Table 17–41. PCC Control Register (PCC_CTRL_REG)

Bit	Name	Function	R/W	Reset Value
15:5	RESERVED	Reserved	R	0x0
4	APLL96_INPUT_SWITCH_STS	Effective switch status for the APLL 96-MHz input clock	R	0x0
3	ULPD_INPUT_SWITCH_STS	Effective ULPD input clock switch status determines whether the clocks have been switched or not.	R	0x0
2	SLC_OUT_DIV	0: Slicer output is not divided by 2. 1: Slicer output is divided by 2.	R/W	0x1
1	APLL_ALWAYS_ON	0: APLL is managed according to the hardware state. 1: APLL is turned on even if it is not used. This bit must be used for fast wake-up.	R/W	0x0
0	CLK_SWITCH_CMD	0: PCC uses the slicer output as OMAP input clock. 1: PCC uses the APLL_32kHz 13-MHz output as the OMAP input clock.	R/W	0x0

The PCC control register allows control of the OMAP input clock.

Table 17–42. PCC Power Control Register (PCC_POWER_CTRL_REG)

Bit	Name	Function	R/W	Reset Value
15:10	RESERVED	Reserved	R	0x0
9	DIS_DBB32K_CLK	Allows turning the DBB 32-kHz clock off 0: The clock is off. 1: The clock is on.	R/W	0x1
8	OSC_32K_BYPASS	0: No bypass for OSC32K 1: Bypass for OSC32K	R/W	0x0
7	DIS_RF_EN	0: RF_EN is managed by hardware according to chip state. 1: RF_EN is always equal to 0 for the MPU.	R/W	0x0
6	ENABLE_SOFT_DRIVE	Allows enabling the software drive for the sleep signal 0: Sleep signal driven by hardware 1: Sleep signal driven by software	R/W	0x0
5	MPU_LDO_PWRDN	0: Drives the MPU LDO power-down signal to 0 1: Drives the MPU LDO power-down signal to 1	R/W	0x0
4	MPU_LDO_SLEEP	Status of the sleep signal. See bits 7 and 8 of POWER_CTRL_REG (offset 0x50) for driving it.	R	0x0
3	MPU_LDO_STEADY	Status of the MPU LDO	R	0x0
2	DBB_LDO_PWRDN	0: Drives the DBB LDO power-down signal to 0 1: Drives the DBB LDO power-down signal to 1	R/W	0x0

Table 17–42. PCC Power Control Register (PCC_POWER_CTRL_REG) (Continued)

Bit	Name	Function	R/W	Reset Value
1	DBB_LDO_SLEEP	0: Drives the DBB LDO sleep signal to 0 1: Drives the DBB LDO sleep signal to 1	R/W	0x0
0	DBB_LDO_STEADY	Status for the DBB LDO steady signal	R	0x0

The PCC power control register allows control of the LDO by software.

Table 17–43. PCC Peripheral Clock Source Register (PCC_PERIPH_CLOCK_SOURCE_SEL)

Bit	Name	Function	R/W	Reset Value
15:10	RESERVED	Reserved	R	0x0
9	SYREN_PERIPH_CLK	When the DBB is off, this bit allows selection of the clock source of the TWL3016 peripheral clock. 0: Slicer output clock 1: APLL13-MHz output clock	R/W	0x0
8	RESERVED2	Reserved	R/W	0x0
7	MCBSP2_CLK_SOURCE	0: Use slicer clock out 1: Use APLL clock out	R/W	0x1
6	MCBSP1_CLK_SOURCE	0: Use slicer clock out 1: Use APLL clock out	R/W	0x1
5	CAM_IF_CLK_SOURCE	0: Use slicer clock out 1: Use APLL clock out	R/W	0x1
4	EAC_CLK_SOURCE	0: Use slicer clock out 1: Use APLL clock out	R/W	0x0
3	UART3_CLK_SOURCE	0: Use slicer clock out 1: Use APLL clock out	R/W	0x0
2	UART1_CLOCK_SOURCE	0: Use slicer clock out 1: Use APLL clock out	R/W	0x1
1	MMC_SDIO_CLK	0: Use slicer clock 1: Use APLL clock	R/W	0x1
0	USB_OTG_CLK	0: Use slicer output 1: Use APLL output	R/W	0x0

Note: All selections are used only if OMAP receives the APLL clock output. If the slicer is on and used by OMAP, all the peripherals receive the slicer output clock.

The PCC peripheral clock source register allows software selection of the peripheral clock sources: APLL or slicer.

Table 17–44. PCC APLL Lock Register (PCC_APLL_LOCK)

Bit	Name	Function	R/W	Reset Value
15	APLL_LOCK	Lock status of APLL13 MHz, not resynchronized. Generated on the 32-kHz clock	R	0x0
14:0	LOCK_TIME_REG	Number of input clock periods before lock goes high when the APLL_32kHz 13 MHz is selected	R/W	0x640

The PCC APLL lock register generates the lock signal for the APLL13-MHz clock according to the input clock period number.

Table 17–45. PCC DBB Status Register (PCC_DBB_STATUS)

Bit	Name	Function	R/W	Reset Value
15:4	RESERVED	Reserved	R	0x0
3	DBB_CLK13M_EN	DBB CLK13M_EN status	R	0x0
2	DBB_SLICER_EN	DBB slicer enable status	R	0x0
1	DBB_VTCXO_EN	DBB VTCXO enable status	R	0x0
0	DBB_RF_EN	RF_EN DBB status without resynchronization	R	0x0

PCC_DBB_STATUS stores the DBB status.

Table 17–46. PCC Interrupt Status Register (PCC_IT_STATUS)

Bit	Name	Function	R/W	Reset Value
15:2	RESERVED	Reserved	R	0x0
1	APLL_13_LOCK_IT	Allows generation of an interrupt each time the APLL13 MHz is locked	R/W	0x0
0	SLICER_ENABLE_IT	Allows generation of an interrupt each time the slicer is reenabled and stable when in APLL mode	R/W	0x0

Note: Reset on write access.

PCC_IT_STATUS provides the status of the PCC interrupt. All status is cleared on a write access to this value, whatever the value written.

Table 17–47. PCC Mask Interrupt Register (PCC_MASK_IT)

Bit	Name	Function	R/W	HW Reset
15:2	RESERVED	Reserved	R	0x0
1	MASK_APLL_13_LOCK_IT	Mask for the APLL13MHz lock interrupt Masked when 1, unmasked when 0	R/W	0x1
0	MASK_SLICER_EN_IT	Mask for the SLICER_EN interrupt. Masked when 1, unmasked when 0	R/W	0x1

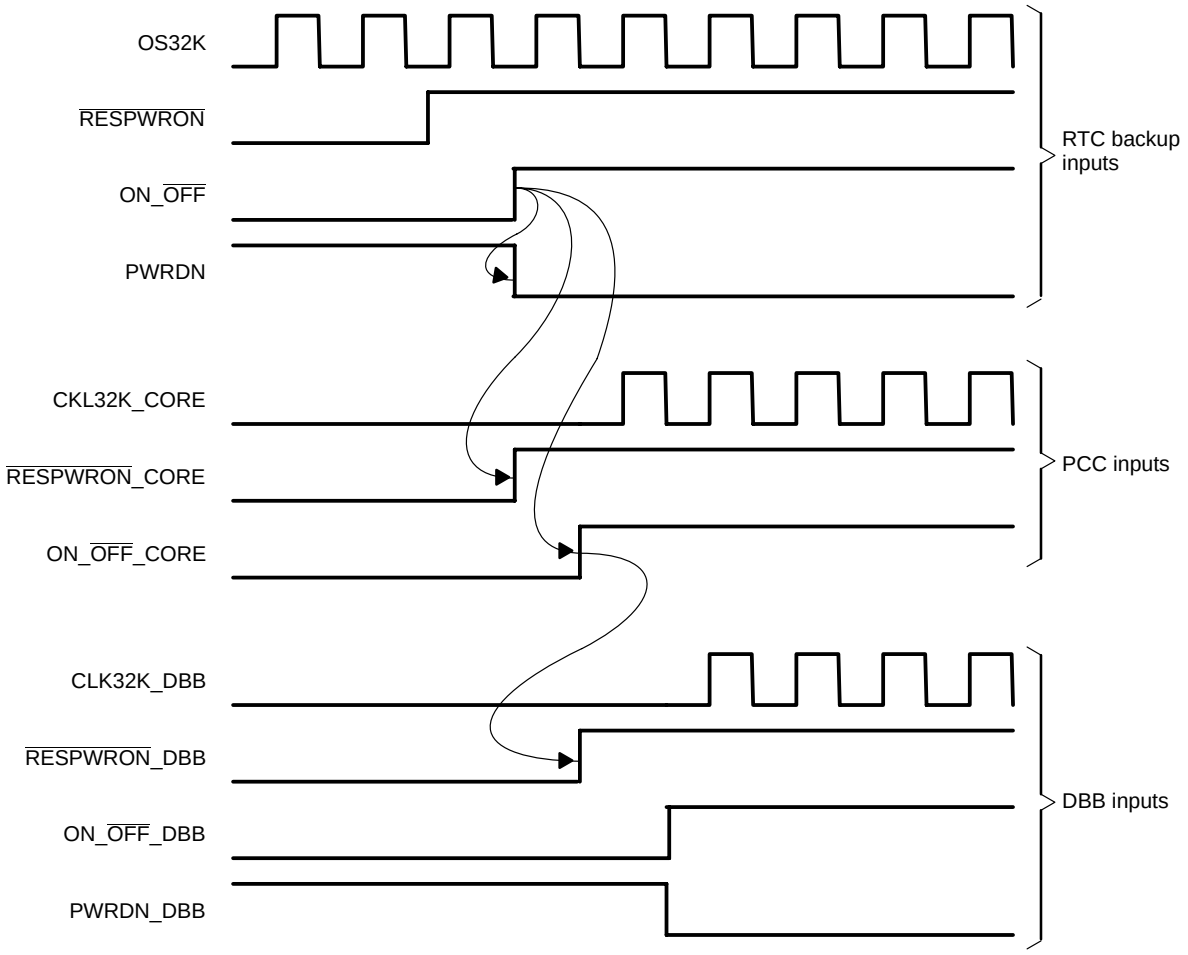
Note: All interrupts are masked after reset.

PCC_MASK_IT allows the masking of the PCC interrupt.

17.7 Timing Diagrams

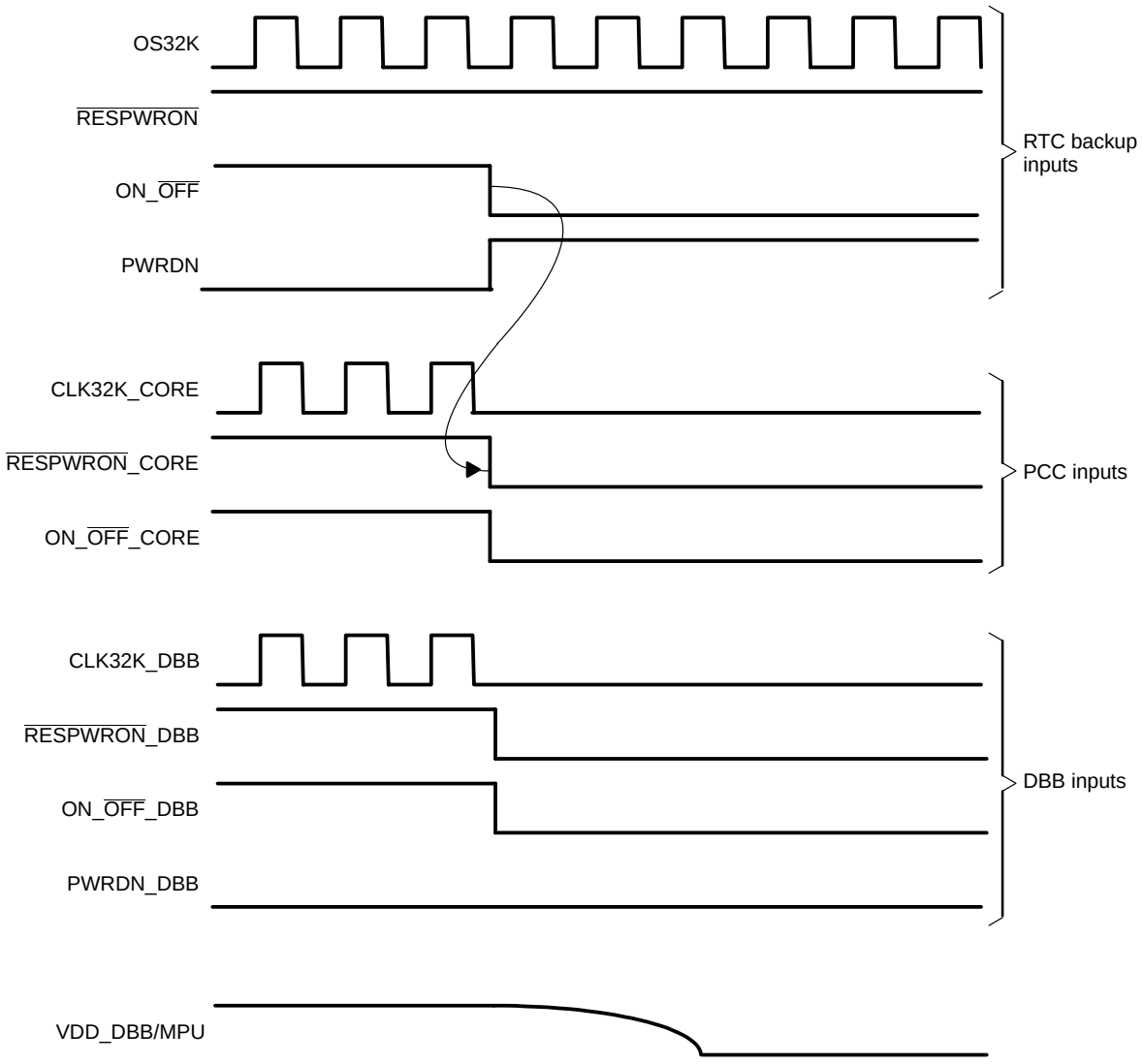
17.7.1 First Power-On Waveform

Figure 17–14. Power-Up Sequence



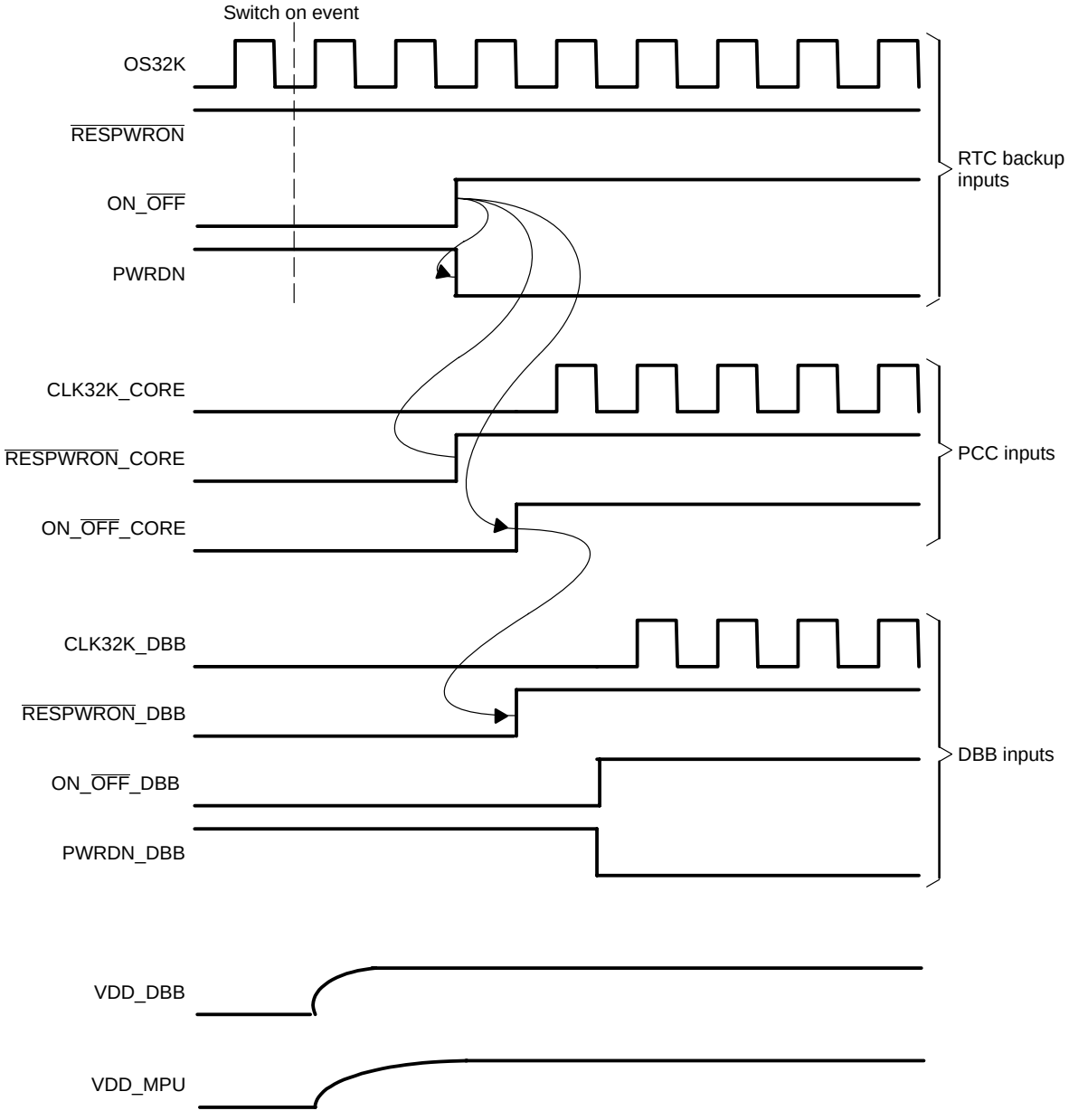
17.7.2 On-to-Off Waveform

Figure 17–15. On-to-Off Waveform



17.7.3 Off-to-On Waveform

Figure 17–16. Off-to-On Waveform



17.8 Embedded LDO Management

17.8.1 Requirements

The embedded low-dropout (LDO) voltage regulator must have the following characteristics:

- No dependency upon external supply ramping sequence. VDD_CORE or VDD_IO can ramp first.
- No effect on boot time
- Possibility of disabling the e-LDO by software
- Possibility of connecting an external supply for the PLL and starting it before or after releasing NRESPWRON (the power on reset signal)

17.8.2 LDO Management Scheme

The e-LDO has two low-power modes:

- Power-down mode (PWRDN=1): Vout = 0 V or Vout = VDD_OUT (external supply)
- Sleep mode (SLEEP=1): Vout = VDD_CORE

The PWRDN signal takes precedence over the SLEEP signal.

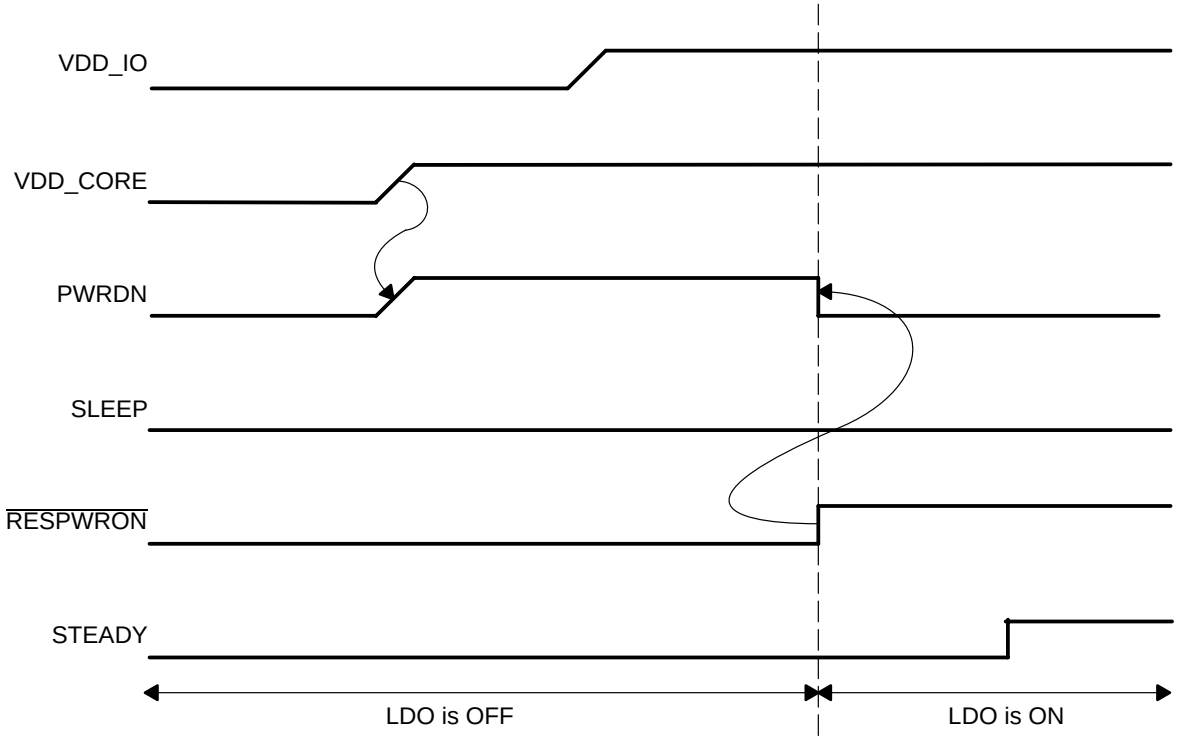
The SLEEP signal is driven by the ULPD module. At reset, SLEEP = 0, meaning that LDO is on.

The PWRDN signal is driven by the OMAP730 configuration module. While the power-on reset is active (NRESPWRON=0), the PWRDN signal is driven high (LDO in power down). As soon as NRESPWON is released, the PWRDN signal switches to low (LDO is on).

In addition, the e-LDO has a built-in, power-on-reset scheme. At power up, this scheme maintains active the internal PWRDN signal of the e-LDO (LDO in power-down mode) until VDD_CORE reaches a steady level.

17.8.3 Timing Diagrams

Figure 17–17. VDD_CORE Ramps First



17.8.4 Using an External Supply for the PLL

An external supply can be used for the PLL. It can be the same as the one supplying VDD_CORE, or it can be a different one.

17.8.4.1 Case 1

The external supply is off at reset and is switched on by software. In this case, the sequence of the operation is as follows:

- ❏ NRESPWRON is released.
- ❏ MCU boots up.
- ❏ Software disables the e-LDO by programming the appropriate OMAP730 configuration register.
- ❏ Software enables the external regulator.

17.8.4.2 Case 2

The external supply is on at reset. In that case, the sequence of the operation is as follows:

- ❏ NRESPWRON is released
- ❏ MCU boots up

- Software disables the e-LDO by programming the appropriate OMAP730 configuration register.

This case imposes specific constrains on the external supply:

- It must be stable when NRESPOWRON is released.
- Its output voltage must be greater than or equal to 1.5 V (a safety margin of 30 to 50 mV is recommended).

Chapter 18

VLYNQ Serial Communications Interface

This chapter discusses the VLYNQ serial communications interface.

Topic **Page**

18.1 Introduction

VLYNQ is a serial (low pin count) communications interface that enables the extension of an internal CBA bus segment to one or more external physical devices. VLYNQ accomplishes this function by serializing bus transactions in one device, transferring the serialized transaction between devices via a VLYNQ port, and deserializing the transaction in the external device. The general features of VLYNQ are:

- Low pin count (as few as 3 signals)
- No 3-state signals
  - All signals dedicated and driven by only 1 device
  - Significant reduction in I/O timing analysis complexity
  - Support of high speed PHYs required
- Scalable performance/support for different PHY technologies
  - TTL @ 150 MHz and 1, 2, 3, 4, …, 8 bits for TX and RX data
  - LVDS @ 622 MHz and 1, 2, 3, 4, … , 8 bits for TX and RX data
  - Gandalf @ 3.5 GHz and 1 bit for TX and RX data
  - Performance with any given PHY increases linearly as the data port width is increased
- Simple packet-based transfer protocol for memory-mapped access
  - Write request/data packet
  - Read request packet
  - Read response data packet
  - Interrupt request packet
- Symmetric operation
  - TX pins on first device connect to RX pins on second device and vice versa
  - Request packets, response packets, and flow control information are all multiplexed and sent across the same physical pins.
  - Supports both host/peripheral and peer-to-peer communication models
  - Able to emulate all currently-used peripheral-interface mechanisms
- Simple block code packet formatting (8b/10b)
- Support for in-band flow control
  - Needs no extra pins
  - Allows receiver to momentarily throttle back transmitter when overflow is about to occur
  - Uses built-in special code capability of block code to seamlessly interleave flow control information with user data
  - Allows system designers to balance cost of data buffering versus performance

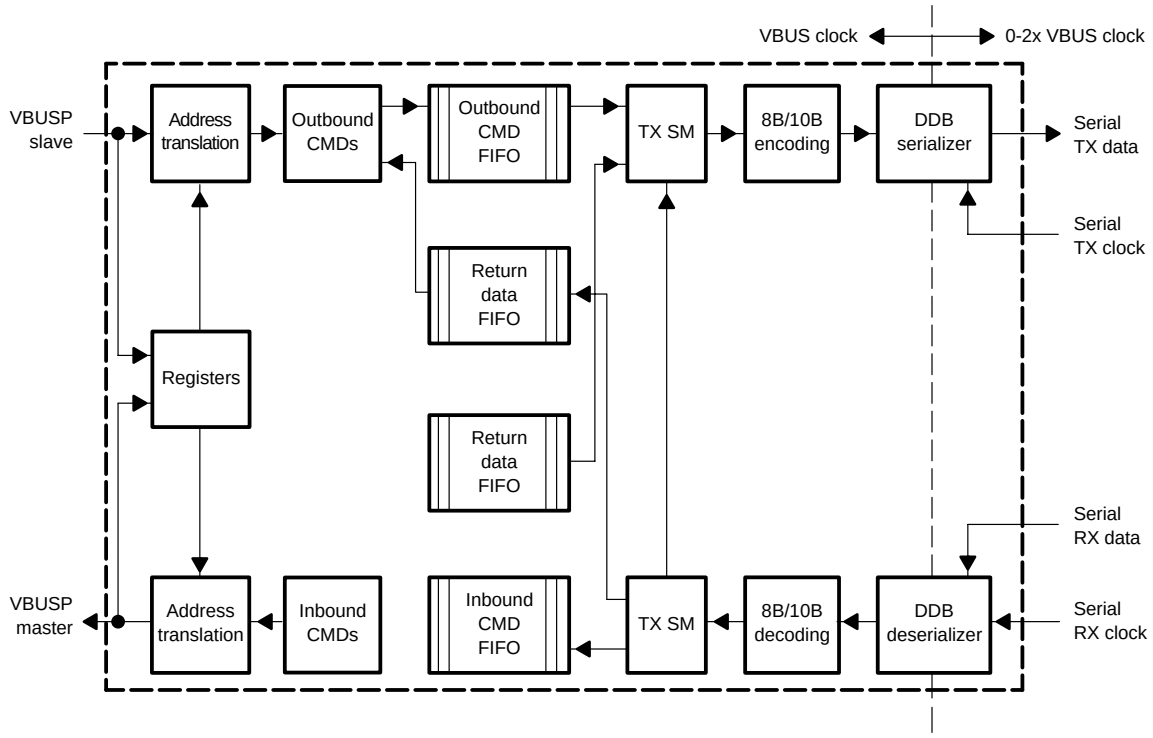
- Supports multiple outstanding transactions
  - Automatic packet formatting optimizations
  - Internal loopback mode

18.2 Operation

18.2.1 Overview

VLYNQ supports both host-to-peripheral and peer-to-peer communication models and is symmetrical. The following block diagram shows a typical structure of a VLYNQ module:

Figure 18–1. Block Diagram



The VLYNQ module implements two 32-bit CBA 2.0 (VBUSP) interfaces. The slave interface is required for transmit and control register access while one master interface is required for receive. Serializer and deserializer blocks are required to convert to/from the 32-bit bus to the external serial interface.

Data on the serial interface is encoded with 8b/10b block coding. Special overhead code groups are used for frame delineation, initialization, and flow control.

For maximum performance, FIFOs are required to buffer entire bursts on the bus, thus minimizing bus latency.

The dribble-down buffer (DDB) is required to align asynchronous serial data to the internal VBUS clock.

Multiple VLYNQ modules may be included on a single device such that VLYNQ devices are effectively daisy chained. Each VLYNQ module can be remotely configured for appropriate address translation and operation.

18.2.2 Write Operation

Write requests from the VBUS are written into the outbound command FIFO. Data is subsequently read from the FIFO and encapsulated in a write request packet. The format of the packet is shown in Figure 18–5. The address is translated, and the packet is encoded and serialized before being transmitted to the remote device. The remote device subsequently deserializes and decodes the receive data and writes it into the inbound command FIFO. A VBUS write operation is then initiated on the bus after reading the address and data from the FIFO.

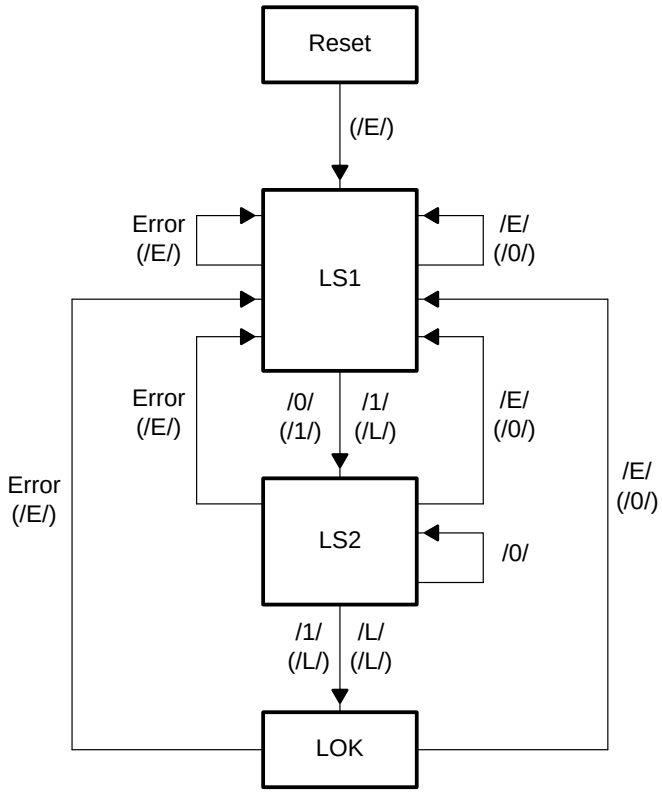
18.2.3 Read Operation

Read requests from the VBUS are written into the outbound command FIFO similar to write requests. Data is subsequently read from the FIFO and encapsulated in a read request packet. The packet is encoded and serialized before being transmitted to the remote device. The remote device subsequently deserializes and decodes the receive data and writes it into the Inbound command FIFO. A VBUS read operation is then initiated on the bus after reading the address from the FIFO. When the remote VBUS receives the read data, the data is written to its return data FIFO before being encoded and serialized. When the receive data reaches the local VLYNQ module, it is deserialized, decoded, and written to the return data FIFO. Finally, the ready signal is asserted, and the read data is transferred on the VBUS. This process is repeated for all data phases within the read burst. Read data is driven onto the VBUS as each 32-bit word is received from the serial interface (read data is not held until the entire burst is stored in the return data FIFO).

18.2.4 Initialization

Because VLYNQ devices can be controlled solely over the serial interface (that is, no local CPU exists), a reliable initialization sequence is necessary to establish a connection between two VLYNQ devices. The same sequence is used to recover from error conditions. The state machine illustrated in Figure 18–2 represents the VLYNQ initialization sequence.

Figure 18–2. Initialization Sequence



A link timer is used to generate a periodic link code, /L/, every 2048 clocks. Transitions from the LOK state occur when this timer expires and no link code has been detected during a period of 4096 clocks.

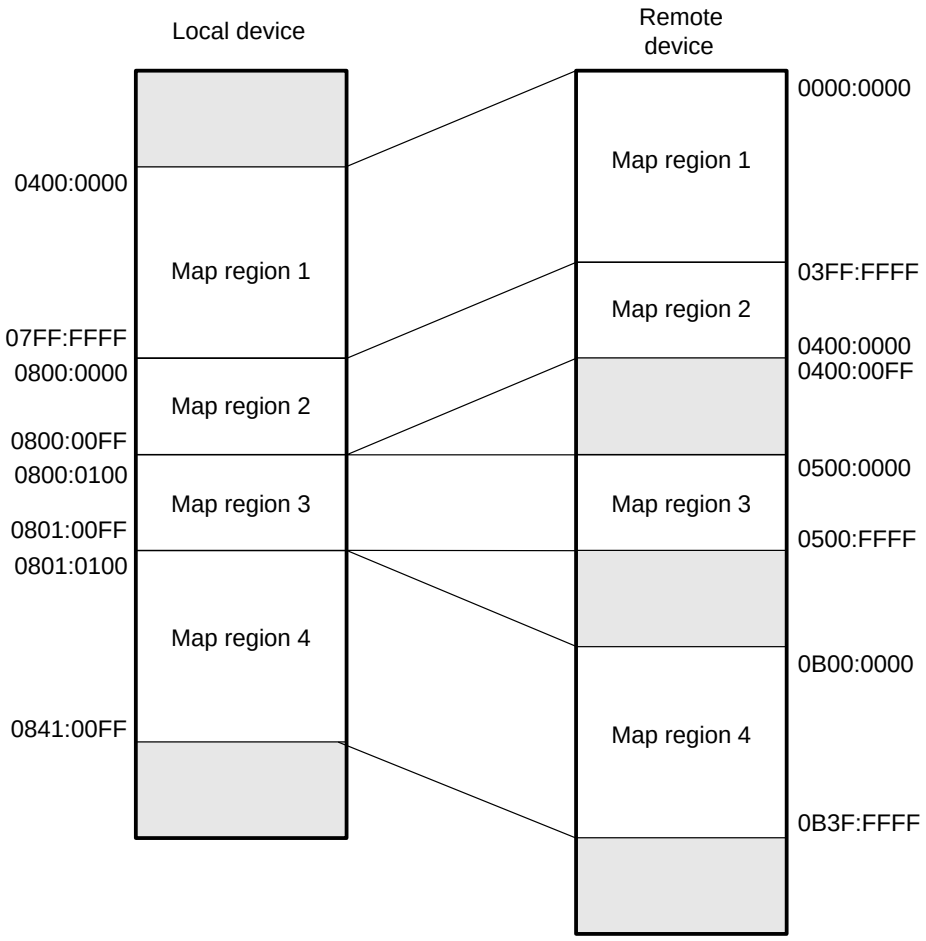
18.2.4.1 Serial Interface Width Configuration

The width of the serial interface is determined from the state of the VLYNQ_TXD_I pins following the assertion of the VBUSP_RST_N pin. The link timer is used to ensure a internal reset period of 2000 clocks, which allows time for the VLYNQ_TXD pins to reach their intended state based on external pullup or pulldown resistors. The actual latching of the VLYNQ_TXD_I pins is done 2000 clocks after VBUSP_RST_N is deasserted. The width of the serial interface is set to the value on the VLYNQ_TXD_I pins plus one.

18.2.5 Address Translation

VLYNQ allows each receive packet address to be translated into one of four mapped regions. No restriction is placed on the size or offset of each mapped region except that each must be aligned to 32-bit words. This is accomplished via register pairs that define the size and offset of each mapped region. Figure 18–3 illustrates one possible memory map.

Figure 18–3. Address Memory Map Example



The following example illustrates the address translation used in each VLYNQ module:

Table 18–1. Address Translation Example (Single Mapped Region)

Register	Local VLYNQ Module	Remote VLYNQ Module
TX address map	0x0400:0000	Don't care
RX address map size 1	Don't care	0x0000:00FF
RX address map offset 1	Don't care	0x0000:0000

❑ Local VLYNQ Module

	0x0400:0054	Initial address from VBUSP slave interface
Subtract	0x0400:0000	TX address map register
	0x0000:0054	Translated address to remote device via serial interface

❑ Remote VLYNQ Module

	0x0000:0054	Initial address from RX serial interface
Compare	0x0000:00FF	RX address map size 1 register

	0x0000:0054	
Add	0x0000:0000	RX address map offset 1 register
	0x0000:0054	Translated address to remote device

In this example, the local address 0x0400:0054 was translated to 0x0000:0054 on the remote VLYNQ device.

The translated address for packets received on the serial interface is determined as follows:

```
If (RX Packet Address) < (RX Address Map Size 1 Register) {

Translated Address = (RX Packet Address) + (RX Address Map Offset 1 Register)

} Else {

RX Packet Address –= (RX Address Map Size 1 Register)

     if (RX Packet Address) < (RX Address Map Size 2 Register)

Translated Address = (RX Packet Address) + (RX Address Map Offset 2 Register)

     } Else {

          RX Packet Address –= (RX Address Map Size 2 Register)

          if (RX Packet Address) < (RX Address Map Size 3 Register) {

Translated Address = (RX Packet Address) + (RX Address Map Offset 3 Register)

          } Else {

               RX Packet Address –= (RX Address Map Size 3 Register)

               if (RX Packet Address) < (RX Address Map Size 4 Register) {

Translated Address = (RX Packet Address) + (RX Address Map Offset 4 Register)

               } Else {

                    RX Packet Address = 0x0

                    Generate address translation interrupt

              }

          }

     }

}
```

Multiple VLYNQ modules can be included on any device such that VLYNQ serial interfaces are effectively daisy chained. The only requirement is that the address translation registers are configured to include the outbound VLYNQ module of the remote device.

18.2.6 Clocking

The VLYNQ module serial clock direction and frequency are software configurable by writing the CLKDIR and CLKDIV fields of the control register. The VLYNQ serial clock can be sourced by the internal VBUS clock (CLKDIR = 1) or by an external clock (CLKDIR = 0). When the internal VBUS clock is selected as the source, the CLKDIV field determines the rate of the serial clock.

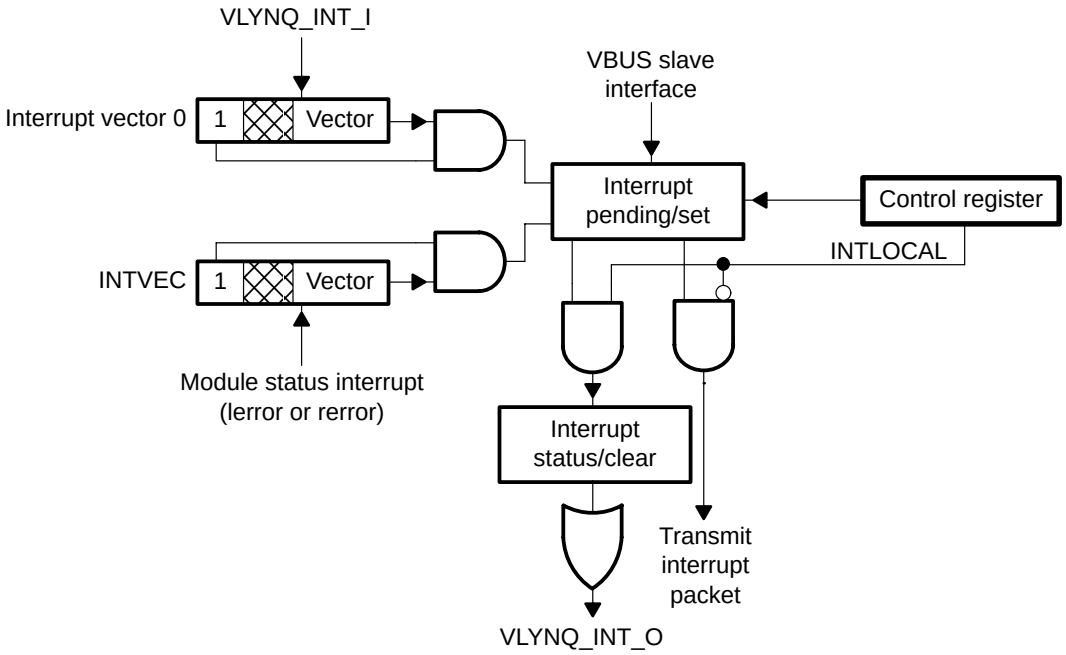
18.2.7 Interrupts

Each VLYNQ module is designed to provide maximum flexibility for generation, detection, and forwarding of interrupt information. Interrupts are generated when the interrupt pending/set register is written. This register can be written by the local VBUS slave interface or via the serial interface using configuration packets.

18.2.7.1 Interrupt Generation

Interrupts are generated by setting bits in the interrupt pending/set register. Bits in the interrupt pending/set register can be set by either writing the interrupt pending/set register directly via the VBUSP slave interface or by asserting one of the VLYNQ_INT_I pins. Whenever the interrupt pending/set register is nonzero, the interrupt status is forwarded in one of two ways depending on the state of the INTLOCAL bit in the control register. If INTLOCAL = 0, the contents of the interrupt pending/set register are inserted in an interrupt packet that is sent over the serial interface. When packet transmission has completed, the associated bits are cleared in the interrupt pending/set register. If INTLOCAL = 1, the bits in the interrupt pending/set register are transferred to the interrupt status/clear register and the ORing of all of the bits in the interrupt status/clear register is driven onto the VLYNQ_INTLVL_O pin. If the VLYNQ_INTLVL_O value was inactive, a pulse on the VLYNQ_INTLVL_O pin is created. One of these pins is connected to the device interrupt controller and causes an interrupt. When software clears all bits in the interrupt status/clear register, the VLYNQ_INTLVL_O pin is deasserted. When the system writes to the interrupt status/clear register while interrupts are still pending, a new pulse is generated on the VLYNQ_INTPLS_O pin. Figure 18–4 is a conceptual illustration.

Figure 18–4. Interrupt Detection and Generation



18.2.7.2 Interrupt Detection

When one of the VLYNQ_INT_I pins is detected high and the corresponding INTEN bit in the interrupt vector register is set, the interrupt is written to the interrupt pending/set register. The actual bit that is set in the interrupt pending/set register is determined by the interrupt vector registers. Once the interrupt pending/set register is written, interrupt generation occurs as previously described.

If an external interrupt is configured as a pulse-type source, for each pulse on the VLYNQ_INT_I pin, the associated bit in the interrupt pending/set register is set. If an external interrupt is configured as a level, for each edge detected on the VLYNQ_INT_I pin, the associated bit in the interrupt pending/set register is set. Until the level interrupt source is cleared, no more interrupts are set in the interrupt pending/set register. An end-of-interrupt (EOI) function exists to reenable the detection of active level interrupts. If the system attempts to clear an interrupt by writing the interrupt status/clear register, any level-sensitive interrupts still active are set in the interrupt pending/set register.

When an interrupt packet is received on the serial interface, the interrupt status is extracted from the packet and written to the register indicated by the interrupt pointer register. This register can be anywhere in memory mapped space. For example, the interrupt pointer register can contain the memory-mapped location of an interrupt status/set register in the device's interrupt controller. Alternatively, the interrupt pointer register can point to the memory-mapped location of its own interrupt pending/set register. Once the interrupt pending/set register is written, interrupt generation occurs as described above.

18.2.8 Flow Control

The VLYNQ module automatically generates flow control enable requests, /P/, when the RX FIFO resources are nearly consumed. The remote device begins transmitting /I/s starting on the first byte boundary following reception of the request. When enough RX FIFO resources have been made available, a flow control disable request, /C/, is transmitted to the remote device. In response, the remote device resumes transmission of data.

18.3 Packet Structure

VLYNQ relies on 8b/10b block coding to minimize the number of serial pins and to allow for inband packet delineation and control.

18.3.1 Special Code Groups

Table 18–2 presents the special code groups.

Table 18–2. Special Code Groups

Code Group Name	Octet Value	Octet Bits	Current RD –	Current RD +
K28.0	1C	000 11100	001111 0100	110000 1011
K28.1	3C	001 11100	001111 1001	110000 0110
K28.2	5C	010 11100	001111 0101	110000 1010
K28.3	7C	011 11100	001111 0011	110000 1100
K28.4	9C	100 11100	001111 0010	110000 1101
K28.5	BC	101 11100	001111 1010	110000 0101
K28.6	DC	110 11100	001111 0110	110000 1001
K28..7	FC	111 11100	001111 1000	110000 0111
K23.7	F7	111 10111	111010 1000	000101 0111
K27.7	FB	111 11011	110110 1000	001001 0111
K29.7	FD	111 11101	101110 1000	010001 0111
K30.7	FE	111 11110	011110 1000	100001 0111

18.3.2 Supported Ordered Sets

Each VLYNQ module must support a limited number of ordered sets. Ordered sets provide for the delineation of packets and synchronization between VLYNQ modules at opposite ends of the serial connection.

Table 18–3. Supported Ordered Sets

Code	Ordered Set	Encoding
/I/	Idle	/K28.5/
/S/	Start-of-packet	/K27.7/
/T/	End-of-packet	/K29.7/
/M/	Byte disable	/K23.7/
/P/	Flow control enable	/K28.0/
/C/	Flow control disable	/K28.2/
/F/	Flowed	/K28.3/
/E/	Error indication	/K28.1/
/0/	Init0	/K28.4/

Table 18–3. Supported Ordered Sets (Continued)

Code	Ordered Set	Encoding
/1/	Init1	/K28.6/
/L/	Link	/K30.7/

18.3.2.1 Idle (/I/)

The IDLE ordered sets are transmitted continuously and repetitively whenever the serial interface is idle.

18.3.2.2 Start-of-Packet (/S/)

A start-of-packet delimiter is used to delineate the starting boundary of a packet.

18.3.2.3 End-of-Packet (/T/)

An end-of-packet delimiter is used to delineate the ending boundary of a packet.

18.3.2.4 Flow Control Enable (/P/)

A flow control enable request is transmitted when a VLYNQ module's receive FIFO is full or nearly full. This code causes the remote VLYNQ device to cease transmission of data.

18.3.2.5 Flow Control Disable (/C/)

The flow control disable request is transmitted by a VLYNQ module when RX FIFO resources are available to accommodate additional data.

18.3.2.6 Flowed (/F/)

The flowed code is transmitted by a VLYNQ module following the reception of a flow control enable request. This code is continually transmitted until a flow control disable is received.

18.3.2.7 Error Indication (/E/)

The error indication is transmitted when errors are detected within a packet. Examples of such errors include illegal packet types and code groups.

18.3.2.8 Init0 (/0/)

The Init0 code group is used during the link initialization sequence.

18.3.2.9 Init1 (/1/)

The Init1code group is used during the link initialization sequence.

18.3.2.10 Link (/L/)

The link code group is used during the link initialization sequence. A link code group is also transmitted each time the internal link timer expires.

18.3.3 Packet Format

The VLYNQ packet format is shown in Figure 18–5, where 0<N<65. Multibyte fields are transferred least-significant-byte first.

Figure 18–5. Packet Format (10-Bit Symbol Representation)

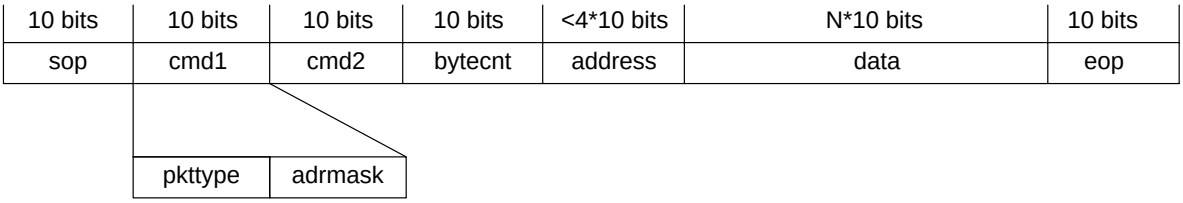


Table 18–4. Packet Format Description

Field	Description	
SOP	Start-of-packet indicator, /S/.	
PKTTYPE[3:0]	This field indicates the packet type.	Mode=0
	0000: Write	
	0001: Write with address increment	3
	0010: Reserved	
	0011: Write 32-bit word with address increment	3
	0100: Configuration write	
	0101: Configuration write with address increment	3
	0110: Reserved for extended command indicator (CMD2)	
	0111: Interrupt	3
	1000: Read	
	1001: Read with address increment	3
	1010: Reserved	
	1011: Read 32-bit word with address increment	3
	1100: Configuration read	
	1101: Configuration read with address increment	3
	1110: Read response	3
	1111: Configuration read response	
ADRMASK[3:0]	Indicates which byte of address has been included in the packet. Only address bytes that have changed since the previous address are included. Each bit corresponds to one byte of address.	
BYTECNT[7:0]	Byte count. This field indicates the total number of bytes in the packet. This field is only included for write, read, and configuration packet types. All other packet types have fixed lengths and do not require this field.	
ADDRESS[7:0]	Address byte 0. This byte is included only if ADRMASK[0] is set to one. If ADRMASK[0] is set to zero, this byte must be assumed to be equal to bits 7:0 of the previous address. Read response packets do not include this field.	

Table 18–4. Packet Format Description (Continued)

Field	Description
ADDRESS[15:8]	Address byte 1. This byte is included only if ADRMASK[1] is set to one. If ADRMASK[1] is set to zero, this byte must be assumed to be equal to bits 15:8 of the previous address. Read response packets do not include this field.
ADDRESS[23:16]	Address byte 2. This byte is included only if ADRMASK[2] is set to one. If ADRMASK[2] is set to zero, this byte must be assumed to be equal to bits 23:16 of the previous address. Read response packets do not include this field.
ADDRESS[31:24]	Address byte 3. This byte is included only if ADRMASK[3] is set to one. If ADRMASK[3] is set to zero, this byte must be assumed to be equal to bits 31:24 of the previous address. Read response packets do not include this field.
DATA	Data payload. The maximum data payload size is limited to 16 32-bit words to allow it to fit in the RX FIFO.
EOP	End-of-packet indicator, /T/.

The CMD2 field is included in the packet only if the packet type indicates extended command (PKTTYPE = 0110). This provides a mechanism to add new features at some point in the future.

Configuration packet types are used to access VLYNQ module registers remotely and do not depend on control register bit settings such as big endian.

18.4 VLYNQ Registers

Table 18–5. Register Mapping

Address Offset	Register	Access
0x00	Revision/ID register	R
0x04	Control register	R/W
0x08	Status register	R/W
0x0c	Reserved	
0x10	Interrupt status/clear register	R/W
0x14	Interrupt pending/set register	R/W
0x18	Interrupt pointer register	R/W
0x1c	TX address map	R/W
0x20	RX address map size 1	R/W
0x24	RX address map offset 1	R/W
0x28	RX address map size 2	R/W
0x2c	RX address map offset 2	R/W
0x30	RX address map size 3	R/W
0x34	RX address map offset 3	R/W
0x38	RX address map size 4	R/W
0x3c	RX address map offset 4	R/W
0x40	Chip version register	R
0x44–0x5c	Reserved	
0x60	Interrupt vector 3–0	R/W
0x64	Interrupt vector 7–4	R/W

Table 18–6. Revision Register (Base Address + 0x00)

Bits	Name	Function	Reset Value
31:16	ID	Unique module ID	0x1
15:8	REVMAJ	Major revision	0x1
7:0	REVMIN	Minor revision	0xb

The revision register contains the major and minor revisions for the VLYNQ module.

Table 18–7. Control Register (Base Address + 0x04)

Bits	Name	Function	Access	Reset Value
31:24	Reserved	Always read as 0. Writes have no effect.	R	0
23:21	MODE	Mode. This field corresponds to a particular set of VLYNQ features. Additional modes will be supported as new features are added. Currently, only mode 0 is supported.	R	0
20:19	Reserved	Always read as 0. Writes have no effect.	R	0
18:16	CLKDIV	Serial clock output divider. This field defines the division factor of the VLYNQ_CLK pin when this pin is sourced from the VBUSP_CLK (CLKDIR set to one). The output clock frequency is equal to VBUSP_CLK/(1+CLKDIV).	R/W	0
15	CLKDIR	Serial clock direction. This bit determines whether the VLYNQ_CLK pin is an input or an output. When set to zero, the VLYNQ_CLK pin is sourced externally. When set to one, the VLYNQ_CLK pin is sourced from a divided-down version of VBUSP_CLK.	R/W	0
14	INTLOCAL	Interrupt local. This bit determines whether interrupts are posted in the interrupt status/clear register or forwarded via the serial interface. When set, interrupts are posted in the interrupt status/clear register and the VLYNQ_INT_O pin is asserted. When clear, interrupts are forwarded out the serial interface to the remote device.	R/W	0
13	INTENABLE	Interrupt enable. This bit causes VLYNQ module status interrupts to be posted to the interrupt pending/set register.	R/W	0
12:8	INTVEC	Interrupt vector. This field indicates which bit in the interrupt pending/set register is set for VLYNQ module interrupts.	R/W	0
7	INT2CFG	Interrupt to configuration register. When this bit is set, the interrupt pending/set register is written directly with the status contained in interrupt packets. When this bit is set, the least-significant 8 bits of the interrupt pointer register are used to point to a local configuration register (typically the interrupt pending/set register).	R/W	0
6:2	Reserved	Always read as 0. Writes have no effect.	R	0

Table 18–7. Control Register (Base Address + 0x04) (Continued)

Bits	Name	Function	Access	Reset Value
1	ILOOP	Internal loopback. When set, this bit causes the serial transmit data to be wrapped back to the serial receive data. No external serial clock is required.	R/W	0
0	Reserved	Always read as 0. Writes have no effect.	R	0

The control register determines the operation of the VLYNQ module.

Table 18–8. Status Register (Base Address + 0x08)

Bits	Name	Function	Access	Reset Value
31:29	DEBUG	Debug. These bits are reserved for manufacturing test.	R	0
28:27	Reserved	Always read as 0. Writes have no effect.	R	0
26:24	SWIDTH	Size of the interface minus 1. This field is set to the value sampled on the VLYNQ_TXD_I pins when exiting reset.	R	X
23:21	MODESUP	Highest supported mode. Indicates the highest mode supported.	R	0
20:11	Reserved	Always read as 0. Writes have no effect.	R	0
10	IFLOW	Inbound flow control. Indicates that a flow control enable request has been received and has stalled transmit until a flow control disable request is received.	R	0
9	OFLOW	Outbound flow control. Indicates that the internal flow control threshold has been reached and a flow control enable request has been sent to the remote device.	R	0
8	RERROR	Remote error. This bit indicates that a downstream VLYNQ module has detected a packet error. This bit is set when a /E/ is received from the serial interface. This bit is cleared by writing a 1 to it.	W	0
7	LERROR	Local error. This bit indicates that an inbound packet contains an error. This bit is cleared by writing a 1 to it.	W	0
6	NFEMPTY3	FIFO 3 not empty. This bit indicates that the slave command is not empty.	R	0
5	NFEMPTY2	FIFO 2 not empty. This bit indicates that the slave data FIFO is not empty.	R	0

Table 18–8. Status Register (Base Address + 0x08) (Continued)

Bits	Name	Function	Access	Reset Value
4	NFEMPTY1	FIFO 1 not empty. This bit indicates that the master command FIFO is not empty.	R	0
3	NFEMPTY0	FIFO 0 not empty. This bit indicates that the master data FIFO is not empty.	R	0
2	SPEND	Pending slave requests. Indicates that a request has been detected on the TX VBUS slave interface.	R	0
1	MPEND	Pending master requests. Indicates that a request has been asserted on the RX VBUS master interface.	R	0
0	LINK	Link. Indicates that the serial interface initialization sequence has completed successfully.	R	0

The status register is used to detect conditions that are of interest to the system designer.

Table 18–9. Interrupt Status/Clear Register (Base Address + 0x10)

Bits	Name	Function	Reset Value
31:0	INTCLR	This field indicates the unmasked status of each interrupt. Writing 1 to any bit in this field clears the corresponding interrupt. When the INTLOCAL bit in the control register is set, the VLYNQ_INT_O pin is driven high when any bit in this register is set.	0

The interrupt status/clear register indicates the unmasked interrupt status. Writing 1 to any bit in this register clears the corresponding interrupt. Any write to this register clears EOI level sensitive interrupts attached directly to the VLYNQ module.

Table 18–10. Interrupt Pending/Set Register (Base Address + 0x14)

Bits	Name	Function	Reset Value
31:0	INTSET	This field indicates the unmasked status of each pending interrupt. Writing to 1 to any bit in this field causes an interrupt packet to be sent on the serial interface if the INTLOCAL control bit is not set.	0

The interrupt pending/set register indicates the pending interrupt status when the INTLOCAL bit in the control register is not set. These bits are cleared when the interrupt packet is forwarded on the serial interface.

Table 18–11. Interrupt Pointer Register (Base Address + 0x18)

Bits	Name	Function	Access	Reset Value
31:2	INTPTR	Interrupt pointer. This register contains the address of the interrupt set register. It can point to any memory-mapped address including the VLYNQ module itself.	R/W	0
1:0	Reserved	Always read as 0. Writes have no effect.	R/O	0

The interrupt pointer register must be written with the address of the interrupt set register for the device. This register must contain the address of either the interrupt set register in the interrupt controller module of the device or the interrupt pending/set register within the VLYNQ module.

Table 18–12. TX Address Map Register (Base Address + 0x1c)

Bits	Name	Function	Access	Reset Value
31:2	TXADRMAP	This field is subtracted from the VBUS address to obtain the zero relative transmit packet address.	R/W	0
1:0	Reserved	Always read as 0. Writes have no effect.	R/O	0

The TX address map register is used to translate transmit packet addresses to remote device VBUS addresses.

Table 18–13. RX Address Map Size 1 Register (Base Address + 0x20)

Bits	Name	Function	Access	Reset Value
31:2	RXADRSIZE1	This field is used to determine if receive packets are destined for the first of four mapped address regions. This field is compared with the address contained in the receive packet. If the received packet address is less than the value in this field, the packet address is added to the RX address map offset 1 register to obtain the translated address.	R/W	0
1:0	Reserved	Always read as 0. Writes have no effect.	R	0

The RX address map size 1 register is used to identify the intended destination of inbound serial packets.

Table 18–14. RX Address Map Offset 1 Register (Base Address + 0x24)

Bits	Name	Function	Access	Reset Value
31:2	RXADROFFSET1	This field is used with the RX address map size 1 register to determine the translated address for serial data. If the received packet address is less than the value in the RX address map size 1 register, the packet address is added to the contents of this register to obtain the translated address.	R/W	0
1:0	Reserved	Always read as 0. Writes have no effect.	R	0

The RX address map offset 1 register is used with the RX address map size 1 register to translate receive packet addresses to local device VBUS addresses.

Table 18–15. RX Address Map Size 2 Register (Base Address + 0x28)

Bits	Name	Function	Access	Reset Value
31:2	RXADRSIZE2	This field is used to determine if receive packets are destined for the second of four mapped address regions. This field is compared with the address contained in the receive packet. If the received packet address is less than the value in this field, the packet address is added to the RX address map offset 2 register to obtain the translated address.	R/W	0
1:0	Reserved	Always read as 0. Writes have no effect.	R	0

The RX address map size 2 register is used to identify the intended destination of inbound serial packets.

Table 18–16. RX Address Map Offset 2 Register (Base Address + 0x2c)

Bits	Name	Function	Access	Reset Value
31:2	RXADROFFSET2	This field is used with the RX address map size 2 register to determine the translated address for serial data. If the received packet address is less than the value in the RX address map size 2 register, the packet address is added to the contents of this register to obtain the translated address.	R/W	0
1:0	Reserved	Always read as 0. Writes have no effect.	R	0

The RX address map offset 2 register is used with the RX address map size 2 register to translate receive packet addresses to local device VBUS addresses.

Table 18–17. RX Address Map Size 3 Register (Base Address + 0x30)

Bits	Name	Function	Access	Reset Value
31:2	RXADRSIZE3	This field is used to determine if receive packets are destined for the third of four mapped address regions. This field is compared with the address contained in the receive packet. If the received packet address is less than the value in this field, the packet address is added to the RX address map offset 3 register to obtain the translated address.	R/W	0
1:0	Reserved	Always read as 0. Writes have no effect.	R	0

The RX address map size 3 register is used to identify the intended destination of inbound serial packets.

Table 18–18. RX Address Map Offset 3 Register (Base Address + 0x34)

Bits	Name	Function	Access	Reset Value
31:2	RXADROFFSET3	This field is used with the RX address map size 3 register to determine the translated address for serial data. If the received packet address is less than the value in the RX address map size 3 register, the packet address is added to the contents of this register to obtain the translated address.	R/W	0
1:0	Reserved	Always read as 0. Writes have no effect.	R	0

The RX address map offset 3 register is used with the RX address map size 3 register to translate receive packet addresses to local device VBUS addresses.

Table 18–19. RX Address Map Size 4 Register (Base Address + 0x38)

Bits	Name	Function	Reset Value
1:0	Reserved	Always read as 0. Writes have no effect.	0

The RX address map size 4 register is used to identify the intended destination of inbound serial packets.

Table 18–20. RX Address Map Offset 4 Register (Base Address + 0x3c)

Bits	Name	Function	Access	Reset
31:2	RXADROFFSET4	This field is used with the RX address map size 4 register to determine the translated address for serial data. If the received packet address is less than the value in the RX address map size 4 register, the packet address is added to the contents of this register to obtain the translated address.	R/W	0
1:0	Reserved	Always read as 0. Writes have no effect.	R	0

The RX address map offset 4 register is used with the RX address map size 4 register to translate receive packet addresses to local device VBUS addresses.

Table 18–21. Chip Version Register (Base Address + 0x40)

Bits	Name	Function	Reset Value
31:16	DEVREV	Device revision. This field reflects the value of the DEVICE_REV pins.	0x0
15:0	DEVID	Device ID. This field reflects the value of the DEVICE_ID pins.	0x0E

The chip version register reflects the value on the DEVICE_ID and DEVICE_REV pins. This register provides a mechanism for software to determine the type and version of VLYNQ devices. The value of DEVICE_ID and DEVICE_REV field must be specified in the device specification.

Table 18–22. Interrupt Vector 3–0 Register (Base Address + 0x60)

Bits	Name	Function	Reset Value
31	INTEN3	Interrupt enable 3. When set, this bit indicates that interrupts detected on the VLYNQ_INT_I[3] pin must be written to the interrupt pending/set register, which subsequently generates an interrupt depending on the status of the INTLOCAL bit in the control register.	0
30	INTTYPE3	Interrupt type 3. When set, this bit indicates that the VLYNQ_INT_I[3] interrupt is pulsed. When clear, this bit indicates that VLYNQ_INT_I[3] is level sensitive.	0
29	INTPOL3	Interrupt polarity 3. When set, this bit indicates that the VLYNQ_INT_I[3] interrupt is active low. When clear, this bit indicates that VLYNQ_INT_I[3] is active high.	0
28:24	INTVEC3	Interrupt vector 3. This field maps the VLYNQ_INT_I[3] pin to a bit in the interrupt pending/set register.	0
23	INTEN2	Interrupt enable 2. When set, this bit indicates that interrupts detected on the VLYNQ_INT_I[2] pin must be written to the interrupt pending/set register, which subsequently generates an interrupt depending on the status of the INTLOCAL bit in the control register.	0
22	INTTYPE2	Interrupt type 2. When set, this bit indicates that the VLYNQ_INT_I[2] interrupt is pulsed. When clear, this bit indicates that VLYNQ_INT_I[2] is level sensitive.	0
21	INTPOL2	Interrupt polarity 2. When set, this bit indicates that the VLYNQ_INT_I[2] interrupt is active low. When clear, this bit indicates that VLYNQ_INT_I[2] is active high.	0

Table 18–22. Interrupt Vector 3–0 Register (Base Address + 0x60) (Continued)

Bits	Name	Function	Reset Value
20:16	INTVEC2	Interrupt vector 2. This field maps the VLYNQ_INT_I[2] pin to a bit in the interrupt pending/set register.	0
15	INTEN1	Interrupt enable 1. When set, this bit indicates that interrupts detected on the VLYNQ_INT_I[1] pin must be written to the interrupt pending/set register, which subsequently generates an interrupt depending on the status of the INTLOCAL bit in the control register.	0
14	INTTYPE1	Interrupt type 1. When set, this bit indicates that the VLYNQ_INT_I[1] interrupt is pulsed. When clear, this bit indicates that VLYNQ_INT_I[1] is level sensitive.	0
13	INTPOL1	Interrupt polarity 1. When set, this bit indicates that the VLYNQ_INT_I[1] interrupt is active low. When clear, this bit indicates that VLYNQ_INT_I[1] is active high.	0
12:8	INTVEC1	Interrupt vector 1. This field maps the VLYNQ_INT_I[1] pin to a bit in the interrupt pending/set register.	0
7	INTEN0	Interrupt enable 0. When set, this bit indicates that interrupts detected on the VLYNQ_INT_I[0] pin must be written to the interrupt pending/set register, which subsequently generates an interrupt depending on the status of the INTLOCAL bit in the control register.	0
6	INTTYPE0	Interrupt type 0. When set, this bit indicates that the VLYNQ_INT_I[0] interrupt is pulsed. When clear, this bit indicates that VLYNQ_INT_I[0] is level sensitive.	0
5	INTPOL0	Interrupt polarity 0. When set, this bit indicates that the VLYNQ_INT_I[0] interrupt is active low. When clear, this bit indicates that VLYNQ_INT_I[0] is active high.	0
4:0	INTVEC0	Interrupt vector 0. This field maps the VLYNQ_INT_I[0] pin to a bit in the interrupt pending/set register.	0

The interrupt vector 3–0 register enables and maps interrupts sourced from the VLYNQ_INT_I[3:0] pins.

Table 18–23. Interrupt Vector 7–4 Register (Base Address + 0x64)

Bits	Name	Function	Reset Value
31	INTEN7	Interrupt enable 7. When set, this bit indicates that interrupts detected on the VLYNQ_INT_I[7] pin must be written to the interrupt pending/set register, which subsequently generates an interrupt depending on the status of the intlocal bit in the control register.	0
30	INTTYPE7	Interrupt type 7. When set, this bit indicates that the VLYNQ_INT_I[7] interrupt is pulsed. When clear, this bit indicates that VLYNQ_INT_I[7] is level sensitive.	0
29	INTPOL7	Interrupt polarity 7. When set, this bit indicates that the VLYNQ_INT_I[7] interrupt is active low. When clear, this bit indicates that VLYNQ_INT_I[7] is active high.	0
28:24	INTVEC7	Interrupt vector 7. This field maps the VLYNQ_INT_I[7] pin to a bit in the interrupt pending/set register.	0
23	INTEN6	Interrupt enable 6. When set, this bit indicates that interrupts detected on the VLYNQ_INT_I[6] pin must be written to the interrupt pending/set register, which subsequently generates an interrupt depending on the status of the intlocal bit in the control register.	0
22	INTTYPE6	Interrupt type 6. When set, this bit indicates that the VLYNQ_INT_I[6] interrupt is pulsed. When clear, this bit indicates that VLYNQ_INT_I[6] is level sensitive.	0
21	INTPOL6	Interrupt polarity 6. When set, this bit indicates that the VLYNQ_INT_I[6] interrupt is active low. When clear, this bit indicates that VLYNQ_INT_I[6] is active high.	0
20:16	INTVEC6	Interrupt vector 6. This field maps the VLYNQ_INT_I[6] pin to a bit in the interrupt pending/set register.	0
15	INTEN5	Interrupt enable 5. When set, this bit indicates that interrupts detected on the VLYNQ_INT_I[5] pin must be written to the interrupt pending/set register, which subsequently generates an interrupt depending on the status of the intlocal bit in the control register.	0
14	INTTYPE5	Interrupt type 5. When set, this bit indicates that the VLYNQ_INT_I[5] interrupt is pulsed. When clear, this bit indicates that VLYNQ_INT_I[5] is level sensitive.	0

Table 18–23. Interrupt Vector 7–4 Register (Base Address + 0x64) (Continued)

Bits	Name	Function	Reset Value
13	INTPOL5	Interrupt polarity 5. When set, this bit indicates that the VLYNQ_INT_I[5] interrupt is active low. When clear, this bit indicates that VLYNQ_INT_I[5] is active high.	0
12:8	INTVEC5	Interrupt vector 5. This field maps the VLYNQ_INT_I[5] pin to a bit in the interrupt pending/set register.	0
7	INTEN4	Interrupt enable 4. When set, this bit indicates that interrupts detected on the VLYNQ_INT_I[4] pin must be written to the interrupt pending/set register, which subsequently generates an interrupt depending on the status of the intlocal bit in the control register.	0
6	INTTYPE4	Interrupt type 4. When set, this bit indicates that the VLYNQ_INT_I[4] interrupt is pulsed. When clear, this bit indicates that VLYNQ_INT_I[4] is level sensitive.	0
5	INTPOL4	Interrupt polarity 4. When set, this bit indicates that the VLYNQ_INT_I[4] interrupt is active low. When clear, this bit indicates that VLYNQ_INT_I[4] is active high.	0
4:0	INTVEC4	Interrupt vector 4. This field maps the VLYNQ_INT_I[4] pin to a bit in the interrupt pending/set register.	0

The interrupt vector 7–4 register enables and maps interrupts sourced from the VLYNQ_INT_I[7:4] pins.

18.4.1 Remote Configuration Registers (Base Address 0x80–0xfc)

The remote configuration registers are the same registers described above, but for the remote VLYNQ device. Configuration accesses are used to access these registers and do not require configuration of the address translation registers.

18.5 Pins

Table 18–24 through Table 18–32 describe the VLYNQ interface pins.

Table 18–24. Clock Pins

Pin Name	Type	Function
VBUSP_CLK	In	Clock

Table 18–25. Reset Pins

Pin Name	Type	Function
VBUSP_RST_N	In	Reset

Table 18–26. VBUSP Slave Interface Pins

Pin Name	Type	Function
S_VBUSP_REQ_REG	In	Request for register access
S_VBUSP_REQ_MAP	In	Request for serial interface access. Indicates that address must be translated.
S_VBUSP_ ADDRESS[31:0]	In	Word aligned address
S_VBUSP_AMODE[1:0]	In	Address mode indicator
S_VBUSP_CLSIZE[1:0]	In	Cacheline size
S_VBUSP_DIR	In	Direction (read or write)
S_VBUSP_FIRST	In	First data phase indicator
S_VBUSP_LAST	In	Last data phase indicator
S_VBUSP_BYTECNT[9:0]	In	Burst length in bytes
S_VBUSP_BYTEN[3:0]	In	Byte enables
S_VBUSP_WDATA[31:0]	In	Write data
S_VBUSP_RDATA[31:0]	Out	Read data
S_VBUSP_RREADY	Out	Read ready
S_VBUSP_WREADY	Out	Write ready

Table 18–27. VBUSP Master Interface Pins

Pin Name	Type	Function
M_VBUSP_REQ	Out	Request
M_VBUSP_ ADDRESS[31:0]	Out	Word aligned address
M_VBUSP_AMODE[1:0]	Out	Address mode indicator
M_VBUSP_CLSIZE[1:0]	Out	Cacheline size

Table 18–27. VBUSP Master Interface Pins (Continued)

Pin Name	Type	Function
M_VBUSP_DIR	Out	Direction (read or write)
M_VBUSP_FIRST	Out	First data phase indicator
M_VBUSP_LAST	Out	Last data phase indicator
M_VBUSP_BYTECNT[9:0]	Out	Burst length in bytes
M_VBUSP_BYTEN[3:0]	Out	Byte enables
M_VBUSP_WDATA[31:0]	Out	Write data
M_VBUSP_RDATA[31:0]	In	Read data
M_VBUSP_RREADY	In	Read ready
M_VBUSP_WREADY	In	Write ready

Table 18–28. Interrupt Interface Pins

Pin Name	Type	Function
VLYNQ_INT_I[7:0]	In	Interrupt input. Transitions on these pins cause an interrupt packet type to be transmitted on the serial interface.
VLYNQ_INTPLS_O	Out	Interrupt output (pulse). Pulses high when an interrupt is detected.
VLYNQ_INTLVL_O	Out	Interrupt output (level). Driven high when an interrupt is detected.

Table 18–29. Serial interface Pins

Pin Name	Type	Function
VLYNQ_RCLK_I	In	Serial receive clock input. Internal pullups or pulldowns are not recommended at the chip level because they can interfere with the width detection of a remote VLYNQ device. If a VLYNQ interface is not used, external resistors must exist on the board to prevent a latch-up condition.
VLYNQ_TCLK_I	In	Serial transmit clock input. Internal pullups or pulldowns are not recommended at the chip level because they can interfere with the width detection of a remote VLYNQ device. If a VLYNQ interface is not used, external resistors must exist on the board to prevent a latch-up condition.
VLYNQ_CLK_O	Out	Serial clock output
VLYNQ_CLK_OE_N	In	Serial clock output enable. Connects to the 3-state enable of VLYNQ_CLK I/O buffer.
VLYNQ_RXD_I[7:0]	In	Serial receive data. Internal pullups or pulldowns are not recommended at the chip level because they can interfere with the width detection of a remote VLYNQ device. If a VLYNQ interface is not used, external resistors must exist on the board to prevent a latch-up condition.

Table 18–29. Serial interface Pins (Continued)

Pin Name	Type	Function
VLYNQ_TXD_I[2:0]	In	Serial transmit data input. Used to determine width of serial interface. These pins are sampled 2000 VBUSP_CLK cycles after reset is deasserted. Pullups or pulldowns must be connected on the board so that the value on these pins is 1 less than the desired serial interface width. If a VLYNQ interface is not used, external resistors must exist on the board to prevent a latch-up condition.
VLYNQ_TXD_O[7:0]	Out	Serial transmit data output
VLYNQ_TXD_OE_N	Out	Serial transmit output enable. Connects to the 3-state enable of all VLYNQ_TXD output buffers.

Table 18–30. Device ID/Revision Pins

Pin Name	Type	Function
DEVICE_ID[15:0]	In	Device ID
DEVICE_REV[15:0]	Out	Device revision

Table 18–31. Test Pins

Pin Name	Type	Function
SCAN_IN	In	Scan input
SCAN_OUT	Out	Scan output
SCAN_EN	In	Scan enable
SCAN_MODE	In	Scan mode
IDDQ_MODE	In	IDDQ mode

Table 18–32. Miscellaneous Pins

Pin Name	Type	Function
DEFAULT_CLKDIR	In	Default serial clock direction. A 0 specifies the clock as an input. This pin is driven to the VLYNQ_CLK_OE_N during reset; it is latched when reset goes inactive.
SBUSY	Out	Serial clock busy. When this pin is asserted, the serial clock is required; it is used in power management functions to control the external serial clock.

18.6 Electrical Information

VLYNQ devices are connected such that adequate setup and hold timing is easily achieved. Figure 18–6 and Figure 18–7 illustrate the proper connection of VLYNQ devices.

Figure 18–6. External Clock Source (CLKDIR = 0)

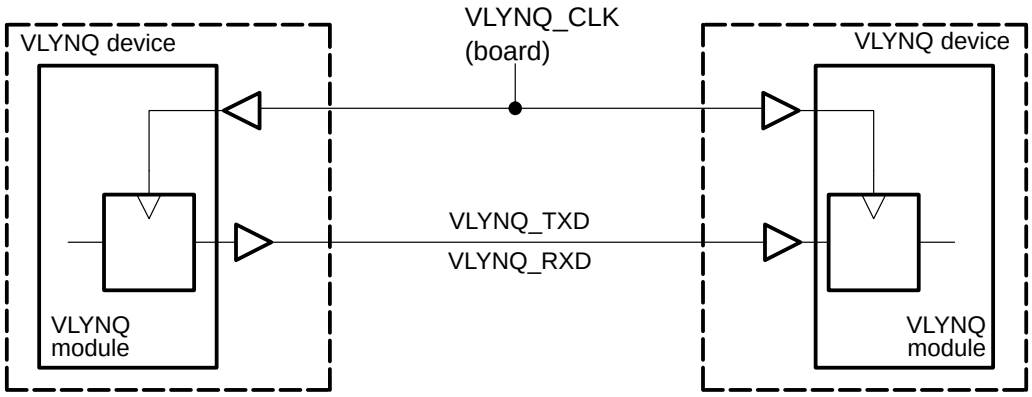


Figure 18–7. Internal Clock Source (CLKDIR = 1)

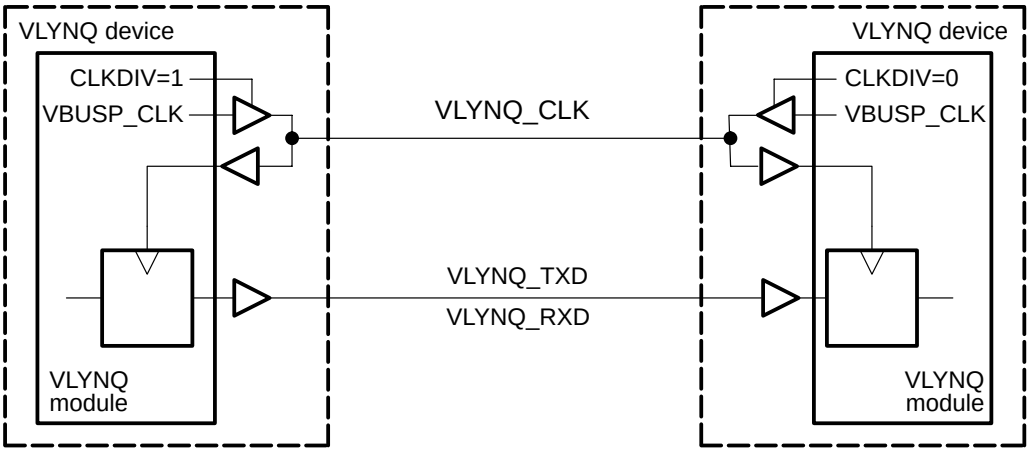


Table 18–33. Timing Requirements

No.	Parameter	Min	Max	Unit
1	Clock period (Tperiod), VLYNQ_CLK	0.5*period (VBUSP_CLK)	No limit	ns
2	Pulse duration, VLYNQ_CLK high	tbd		ns
3	Pulse duration, VLYNQ_CLK low	tbd		ns
4	Delay time, VLYNQ_CLKC to VLYNQ_TXD invalid	2.25		ns
5	Delay time, VLYNQ_CLKC to VLYNQ_TXD valid		tperiod – 0.25	ns
6	Setup time, VLYNQ_RXD before VLYNQ_CLKC	–0.75		ns
7	Hold time, VLYNQ_RXD after VLYNQ_CLKC	2.0		ns

Figure 18–8. VLYNQ Timing Diagram

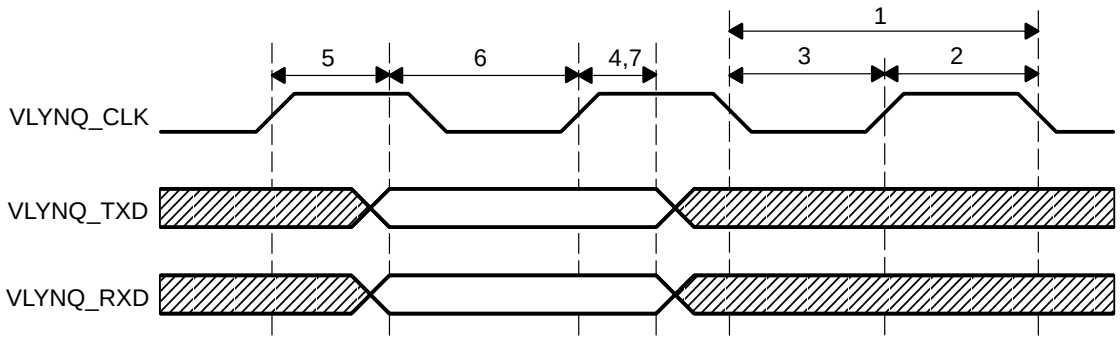


Table 18–33 assumes that the difference in the board delay for the VLYNQ_CLK and any VLYNQ_TXD pin is less than 0.25 ns (used for RX hold time margin). It also assumes that the total board delay for the VLYNQ_CLK plus any VLYNQ_TXD signal is less than 1.00 ns (used for maximum VLYNQ_CLK to VLYNQ_TXD valid delay time).

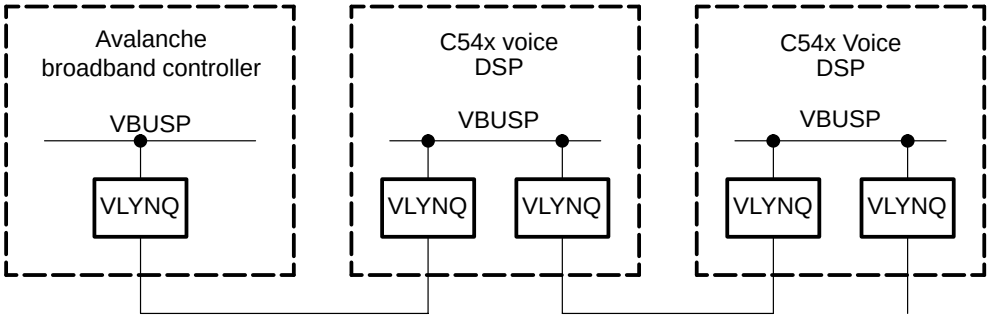
Maximum pin loading is assumed to be 15 pF (used for maximum VLYNQ_CLK to VLYNQ_TXD valid delay time). Minimum capacitance is assumed to be 7 pF (used for minimum VLYNQ_CLK to VLYNQ_TXD delay time).

18.7 Application Examples

18.7.1 Integrated Access Device

VLYNQ enables low-cost derived voice applications by allowing one or more C54x™ DSP devices to connect to an avalanche broadband controller device over 3-pin serial interfaces.

Figure 18–9. Integrated Access Device



18.8 General Description

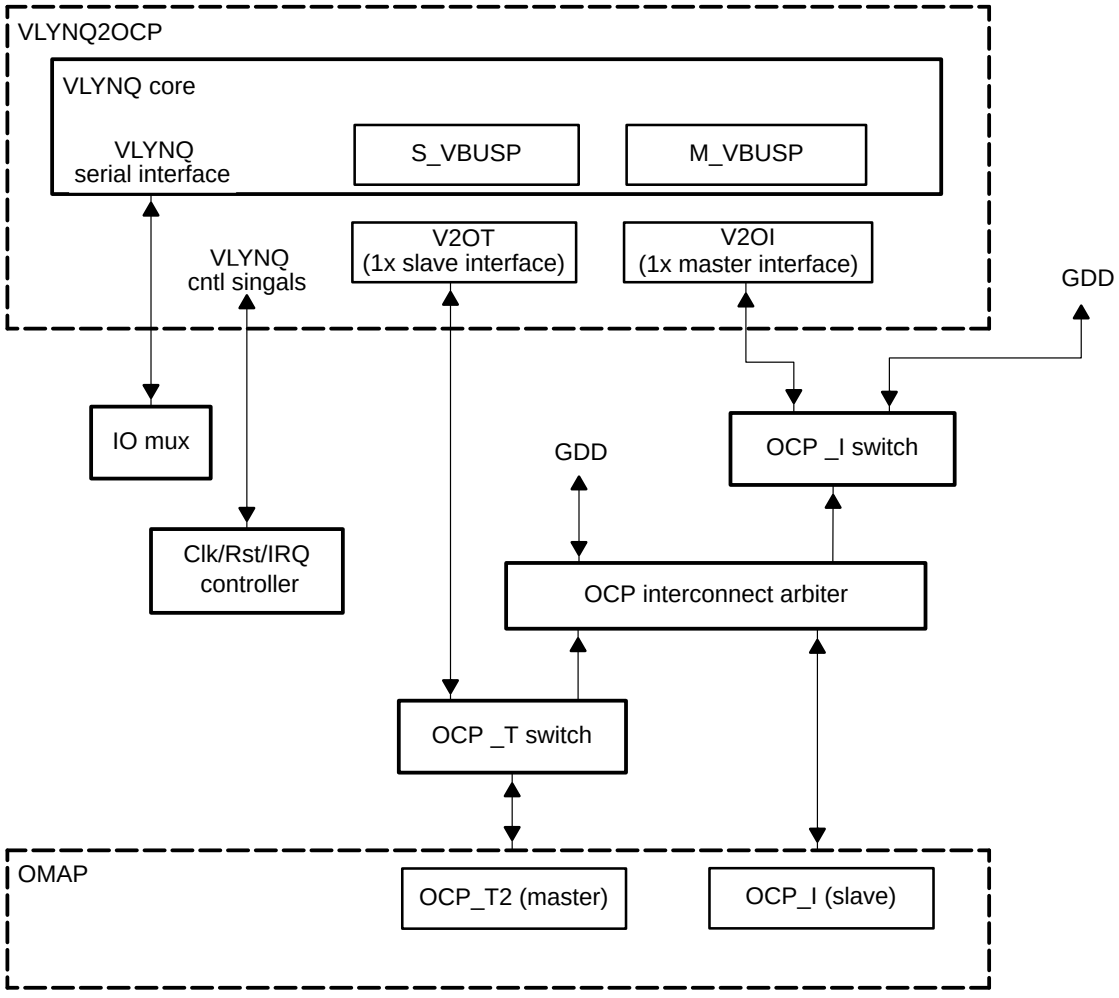
18.8.1 VLYNQ2OCP Module Overview

The VLYNQ2OCP contains a VLYNQ serial communication IP module and an OCP interface wrapper. The OCP interface wrapper enables the integration of a VBUS-based IP module in an OCP-compliant OMAP1611/730 system.

18.8.2 Block Diagram

The following block diagram shows the interfacing between the VLYNQ2OCP module and the target system.

Figure 18–10. VLYNQ2OCP in OMAP System



18.9 Submodule Functional Description

18.9.1 VLYNQ Module

The VLYNQ is a serial (that is, low pin count) communications interface that enables the extension of an internal CBA bus segment to one or more external physical devices. The VLYNQ serializes bus transactions in one device, transfers the serialized transactions between devices via a VLYNQ port, and deserializes the transaction in the external device. The general features of the VLYNQ are as follows:

- Low pin count (5-pin option chosen for VLYNQ2OCP)
- Simple packet-based transfer protocol for memory mapped access
- Symmetric operation
  - TX pins on the first device connect to RX pins on the second device, and vice versa.
  - Request packets, response packets, and flow control information are multiplexed and sent across the same physical pins.
  - Both host/peripheral and peer-to-peer communication models are supported.
  - All currently used peripheral interface mechanisms can be emulated.
- Simple block-code packet formatting (8b/10b)
  - Allows receiver to momentarily throttle back transmitter when overflow is about to occur
  - Uses built-in special code capability of block code to seamlessly interleave flow control information with user data
- Supports multiple outstanding transactions
- Automatic packet formatting optimizations
  - Internal loopback mode on the serial interface to test VLYNQ module via VBUS slave interface

The VLYNQ module has two 32-bit VBUSP 2.0 interfaces. The slave interface is used for data transfer and control register access by a local host processor or a DMA controller. The master interface is used to process VBUS requests passed through the VLYNQ serial interface from a remote host processor or a DMA controller.

Each VLYNQ module has pins that connect to a unique device ID. The value of these pins is then mirrored in a memory mapped register within the VLYNQ module, which is easily accessed by software. The local VLYNQ device register map also contains a copy of the remote device registers. This allows the software to determine the remote chip ID and, therefore, the remote memory map without trying to access random remote addresses to find a device ID register. Each device with a VLYNQ module must be registered to obtain a unique device ID; the OMAP730 chip ID is 0x000E.

18.9.2 OCP Wrapper

Two OCP wrappers are implemented to support master and slave VBUS-to-OCP transactions. The VBUS2OCP wrapper handles the master VBUS

transfer requests from VLYNQ and interfaces to the OCP interconnect arbiter via an OCP_I switch module. The OCP2VBUS wrapper handles the slave VBUS transfer requests from OMAP and interfaces to the OCP_T2 (L3 OCP target) via an OCP_T switch.

These wrappers support OCP transfer protocols supported by the interfacing modules and VBUS transactions that can be translated to equivalent OCP transactions. VBUS functions that are not easily translatable to OCP transfer functions may be prevented from occurring in the system**.** See Section18.11, *OCP-VBUS Interface Wrapper,* for more details.

18.10 Signal Description

Table 18–34. OCP Master Interface Signals

Signal Name	Width	I/O	Signal Description	Reset and Idle Mode
V2OI_SDATAI	31:0	In	OCP data bus input from slave	–
V2OI_ SCMDACCEPTI	1	In	Slave accepts transfer. 0: The slave waits before accepting the master's transfer request. 1: The slave accepts the master's transfer request.	–
V2OI_SRESPI	1:0	In	Response field generated by the slave as a response to a transfer request from the master Any response field is accepted (NULL, DVA, ERR).	–
V2OI_SERRORI	1	In	Slave error indication	–
V2OI_MADDRO	31:0	Out	OCP address bus of the master (size of the system bus)	0x00000000
V2OI_MCMDO	2:0	Out	Output transfer command (only idle/read/write modes are generated)	000
V2OI_MDATAO	31:0	Out	OCP data bus for slave	0x00000000
V2OI_MBYTEENO	3:0	Out	OCP byte enable for slave	0000
V2OI_MBURSTO	2:0	Out	Burst type. This 3-bit wide signal allows linking related transfers into a burst transaction.	000

Table 18–35. OCP Slave Interface Signals

Signal Name	Width	I/O	Signal Description	Reset and Idle Mode
V2OT_SDATAO	31:0	Out	OCP data bus slave output	0x00000000
V2OT_ SCMDACCEPTO	1	Out	Slave accepts transfer. 0: The slave waits before accepting the master's transfer request. 1: The slave accepts the master's transfer request.	0
V2OT_SRESPO	1:0	Out	Response field generated by the slave as a response to a transfer request from the master Any response field is accepted (NULL, DVA, ERR).	00
V2OT_SERRORO	1	Out	Slave error response	0
V2OT_MADDRI	31:0	In	OCP address bus from the master (size of the system bus)	–
V2OT_MCMDI	2:0	In	Input transfer command (only idle/read/write modes are generated)	–
V2OT_MDATAI	31:0	In	OCP data bus from the master	–

Table 18–35. OCP Slave Interface Signals (Continued)

Signal Name	Width	I/O	Signal Description	Reset and Idle Mode
V2OT_MBYTEENI	3:0	In	OCP byte enable from the master	–
V2OT_MBURSTI	2:0	In	Burst type. This 3-bit wide signal allows linking related transfers into a burst transaction.	–
V2OT_ SDMAREQUEST	1	Out	Write DMA request	0

Table 18–36. VLYNQ Serial Interface Signals

Signal Name	Width	I/O	Signal Description	Reset and CFG Mode
VLYNQ_RXD_I	7:0	In	VLYNQ serial receive data input	–
VLYNQ_TXD_O	7:0	Out	VLYNQ serial transmit data output	0x00
VLYNQ_RCLK_I	1	In	VLYNQ serial RX clock	–
VLYNQ_TCLK_I	1	In	VLYNQ serial TX clock	–
VLYNQ_CLK_O	1	Out	VLYNQ internal generated clock output	V2O_OCP_ CLK
VLYNQ_CLK_OE_N	1	Out	VLYNQ internal clock output enable (ref clock: V2O_OCP_CLK)	(See note.)
VLYNQ_TXD_OE_N	1	Out	VLYNQ serial transmit data output enable (This always drives out 1.)	0
VLYNQ_DEFAULT_ CLKDIR	1	In	VLYNQ default clock direction (ref clock: V2O_OCP_CLK)	–

Note: The reset value of VLYNQ_CLK_OE_N is the inverse of VLYNQ_DEFAULT_CLKDIR.

Table 18–37. Control Signals

Signal Name	Width	I/O	Signal Description	Reset Mode
V2O_OCP_CLK	1	In	VLYNQ OCP/VBUS clock input – 100 MHz	–
V2O_RST_N	1	In	System reset	–
V2O_CLK_EN_I	1	In	V2O system clock enable input (active low enable)	–
VLYNQ_CLKOUT_ EN_I	1	In	VLYNQ output clock enable input (active low enable) (ref clock: V2O_OCP_CLK)	–
V2O_INTPLS_O	1	Out	VLYNQ interrupt output (active low pulse)	1

Table 18–38. Test Signals

Signal Name	Width	I/O	Signal Description	Reset Mode
V2O_SCAN_MODE	1	In	Scan enable (for memory DFT collar and reset bypass enable)	–
V2O_SCAN_CLK	1	In	Scan mode clock	–
V2O_SCAN_IN	4	In	Scan inputs	–
V2O_SCAN_OUT	4	Out	Scan outputs	TBD
V2O_SCAN_ENABLE	1	In	Scan enable	–
V2O_IDDQ_MODE	1	In	IDDQ enable (to disable FIFO memory write during scan)	–

Four scan chains are planned for VLYNQ2OCP.

Table 18–39. VLYNQ2OCP Module Tie-off Signals

Signal Name	Width	I/O	Signal Description	Reset Mode
V2O_DEVICE_ID	16	In	Chip ID	–
V2O_DEVICE_REV	16	In	Device revision information	–
V2O_REG_START_ADDR	32	In	Local VLYNQ/wrapper register space start address	–
V2O_MEM_START_ADDR	32	In	Start address of mapped VLYNQ memory space	–
V2O_MEM_END_ADDR	32	In	End address of mapped VLYNQ memory space	–

Note: These inputs must be tied off at the top with correct constant values.

All signals with the prefix VLYNQ_ are referenced to VLYNQ_CLK, unless otherwise specified in the signal description. All other signals are referenced to V2O_OCP_CLK.

18.11 OCP-VBUS Interface Wrapper

18.11.1 OCP2VBUS (OCP Slave)

18.11.1.1 OCP to VBUS Protocol Translation

The OCP_T2 target module requires the following features of an OCP slave module:

- ❏ Supported commands: idle, read, and write (post-write)
- ❏ Data width: 32 bits
- ❏ Support for single accesses of read and write with data width of 8/16/32 bits
- ❏ Support for incrementing burst accesses only. The burst size is always four 32-bit words with a burst sequence of four-two-two-last. While a burst read or write command is active, no idle commands are issued by the OCP target as long as the SCMDACCEPT from the slave is pipelined.
- ❏ Support for abort management (bus error and time-out error)
- ❏ Support for the OMAP pipeline/nonpipeline read mode. When the pipeline mode is enabled in OMAP, the OCP target can send out a second read command as soon as the request phase of the first read command is completed. The OCP target supports only one deep-pipeline read.

The OCP2VBUS wrapper supports the above features with the error management scheme described in Section 18.11.1.2.

18.11.1.2 OCP Slave Error Handling

The OCP2VBUS slave interface generates an error response to the master if it receives an illegal command, out of bound address, or illegal burst code sequence. During a write command, SERRORO is set to 1 when SCMDACCEPTI is asserted to indicate a write error condition (SRESPO is null). During a read command, SRESPO is set to ERR when SCMDACCEPTI is asserted to indicate a read error condition (SERRORO is not asserted).

Furthermore, if an error is found during a write command, whether it is single or a burst command, the command is not translated into a VBUS transaction. During a read command, if an error condition is found in the single- or first cycle of a burst cycle, the entire transaction is not translated, and SDATAO returns 0x0 with SRESPO set to ERR. If an error is found after the first cycle of a burst command, only the transaction with the error is not translated to VBUS read, and again SDATA returns 0x0.

The time-out error feature of the L3 OCP target module is not supported by VLYNQ2OCP. Therefore, the time-out feature in the L3 master must be disabled when accessing the VLYNQ interface.

When an access does not finish within a prescribed time-out cycle count, the time-out error handling scheme is for the OCP target to generate an MABORT

signal to the requestor (MPU, DMA, or OCP_I) to retry the access and for the OCP slave device to terminate any unfinished access immediately. Although this is a rare occurrence for an OMAP internal access, a VLYNQ access can easily exceed this if a slow device is connected on the other end and/or there is heavy traffic on the arbitration logic on the remote side.

18.11.1.3 Address Decoding

The address space for the VLYNQ register and memory are module-pin programmed, as shown in Table 18–40.

Table 18–40. VLYNQ2OCP Address Space Configuration

Block Name	Start Address	End Address	Size
VLYNQ registers	V2O_REG_START_ADDR	V2O_REG_START_ADDR + 0xFF	Fixed 256 bytes
VLYNQ2OCP registers	V2O_REG_START_ADDR + 0x100	V2O_REG_START_ADDR + 0x10B	Currently only three registers are defined
(Reserved)	V2O_REG_START_ADDR + 0x10C	V2O_REG_START_ADDR + 0x1FF	
VLYNQ memory	V2O_MEM_START_ADDR	V2O_MEM_END_ADDR	Programmable, typically 64M bytes

Any access outside of the valid register regions and the VLYNQ memory-mapped space is considered an out-of-bounds access, and a slave error response is issued. This includes an access to the reserved space shown in Table 18–40.

18.11.1.4 Byte-Enable and Byte-Count Support

All OCP burst commands are translated to VBUS transactions with BYTECNT16. By definition of the OCP_T2 design, byte enables for these transfers are always 0xF. Single OCP commands are translated into VBUS transactions with BYTECNT 4. During a single access only the following byte enables are supported:

- 32-bit access: 1111
- 16-bit access: 0011 or 1100
- 8-bit access: 1000 or 0100 or 0010 or 0001

The VBUS address [1:0] is always set to 00, and byte address generation is not supported.

18.11.1.5 DMA Request Support

When a DMA transfer is performed out to the remote VLYNQ memory space (write), the VLYNQ uses 16-word deep FIFO to buffer OCP burst writes.

Because the DMA is much faster than the serial VLYNQ interface (1- or 2-bit wide bus), data backup occurs if the DMA is not synchronized to the FIFO fullness status, and subsequent OCP_T2 transactions to VLYNQ2OCP are not serviced until the FIFO has room. To avoid this situation, the VLYNQ2OCP pro-

vides a DMA request signal (V2OT_SDMAREQUEST) to allow DMA transfers to be synchronized by hardware.

The V2OT_SDMAREQUEST (an active low signal—low for two OCP clock cycles) is asserted on the following conditions:

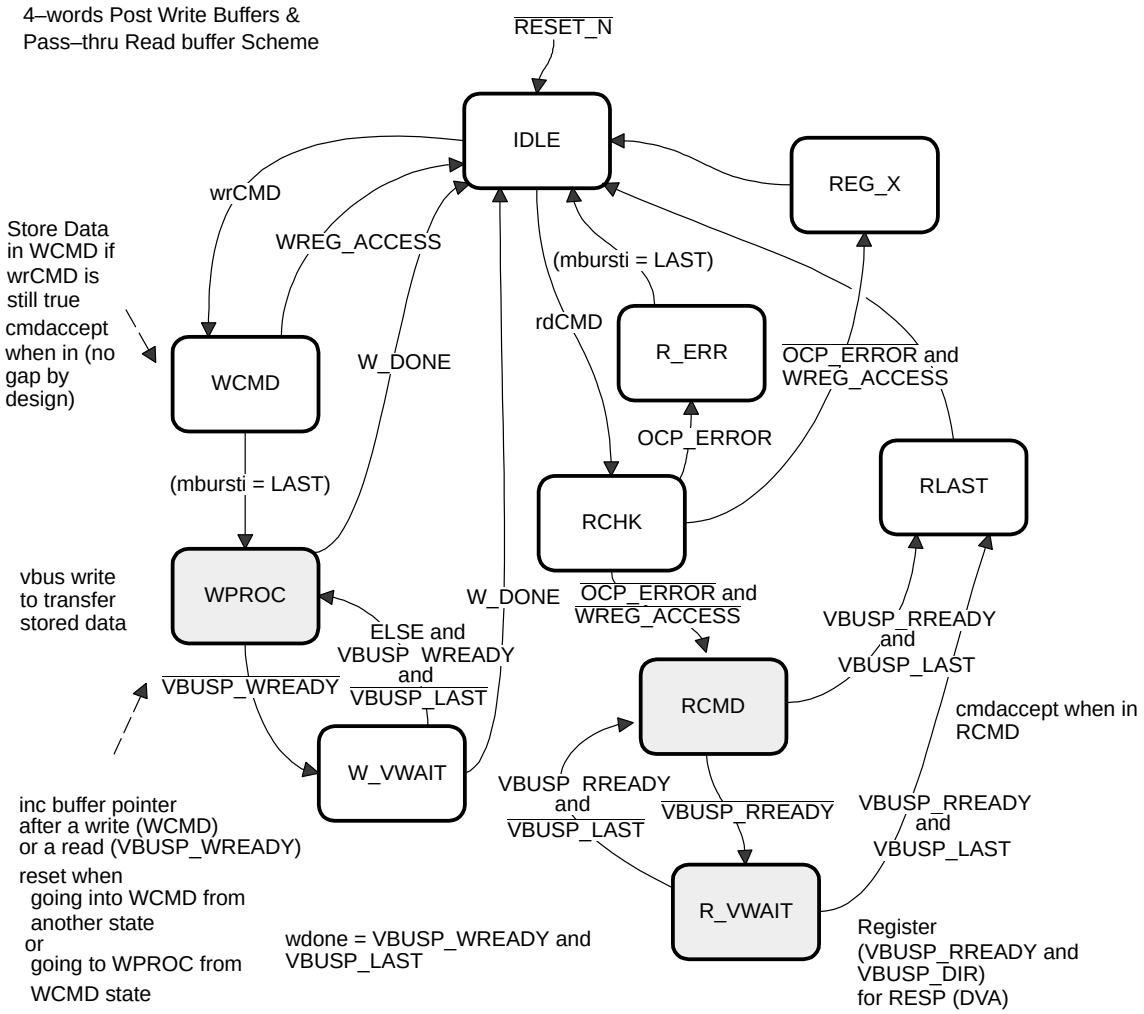
- When the MPU writes 0x1 to V2O_DMAREQ_EN register (offset 0x10C)
- When a burst write transfer is finished (the resulting VBUS transfer to VLYNQ from the wrapper has completed)

The VLYNQ2OCP always generates one more DMA request than needed, because it does not know the size of the transfer. The system DMA controller ignores this. Also, the V2O_DMAREQ_EN must be cleared and rewritten before each programmed DMA transfer to create the initial DMA generation condition.

18.11.1.6 Implementation

The interface runs at 1 x OCP clock mode and is clocked by a balanced 100-MHz clock input. The diagram in Figure 18–11 shows the FSM for handling OCP slave transactions. The write handling in the diagram comprehends any idle inserted by the OCP target during a burst access using a 4-word-deep write buffer. The read data transfer is throttled by the VBUSP_READY signal (which trickles in), which enables SCMDACCEPT and SRESP valid. The read side does not need additional buffering.

Figure 18–11. OCP Slave Interface FSM



18.11.2 VBUS2OCP (OCP Master)

18.11.2.1 VBUS to OCP Protocol Translation

The following subset of the VBUS master interface is supported by the VLYNQ master interface.

Table 18–41. VBUS Master Interface Signals

Signal Name	Direction	Class	Function
VBUSP_REQ	Output	Required	Request
VBUSP_ADDRESS	Output	Required	Address
VBUSP_AMODE	Output	Required	Addressing mode indicator
VBUSP_CLSIZE	Output	Optional	cache line size
VBUSP_DIR	Output	Required	Direction
VBUSP_FIRST	Output	Required	First data phase indicator

Table 18–41. VBUS Master Interface Signals (Continued)

Signal Name	Direction	Class	Function
VBUSP_LAST	Output	Required	Last data phase indicator
VBUSP_BYTECNT	Output	Required	Byte count
VBUSP_BYTEN	Output	Required	Byte enables
VBUSP_WDATA	Output	Required	Write data
VBUSP_RDATA	Input	Required	Read data
VBUSP_WRREADY	Input	Required	Write ready
VBUSP_RREADY	Input	Required	Read ready

Command Support

The VLYNQ2OC OCP master issues only idle, read, and write commands. READEX is not used, although it is one of the valid commands OCPI accepts.

BYTECNT Support Based on VLYNQ VBUS Limitation

Although the VLYNQ supports the VBUS 2.0 compatible interface (that is, odd byte counts and non-word-aligned byte addresses), its internal logic still performs the transfers in VBUS1.0 mode. This means that all requests made on the VBUS slave interface of the remote VLYNQ device are converted to transfers based on word count and word address, and the local VLYNQ VBUS master reflects this. Therefore, in VLYNQ2OCP, word-aligned transfers only need to be supported with a multiple-of-four BYTECNT. The *byteen*, on the other hand, is passed as is from the source.

All VBUSP transactions with BYTECNT less than 16 are translated to single OCP accesses. All other VBUS transactions with BYTECNT equal to or greater than 16 are translated to burst transactions of four words, plus single transactions for any remaining non-four-word transfers.

The condition for a burst transaction is either:

- Command is a read access.
- Byte enable for all four words is 0xF.

Byte Enable Support

The VBUS allows any combination of byte enables, including all zeros. The wrapper passes all OMAP legal byte enables (1111, 1100, 0011, 1000, 0100, 0010, 0001) and converts illegal byte enables to repeated OCP single commands with a single one-byte enable bit, one at a time. For example, a write transfer with a byte enable equal to 0101 is converted to two OCP write commands to the same address with byte enables 0100 and 0001. This is done only for write commands. During a read, byte enable is ignored by the remote VLYNQ device, and VLYNQ2OCP only sees 0xF as a byte enabled during a

read. Also, the OMAP only allows burst transfers in 32-bit words (which always has byte enable equal to 1111). Therefore, only four consecutive writes with all byte enables equal to 1111 are sent as a burst transfer. Any transfer with byte enable equal to 0000 is ignored and is not passed to the OMAP side.

AMODE Support

Although the VBUS supports four addressing modes (linear incrementing, linear decrementing, constant, and cache line wrap), the first generation VLYNQ IP core can only handle linear incrementing bursts. Therefore, the VBUS2OCP wrapper only sees linear incrementing-mode transfer requests from the VLYNQ core, and thus only needs to support burst mode transfers in incrementing mode.

CLSIZE Support

CLSIZE is not supported.

Wait-Insertion Support

Because the VBUS master must provide data in write mode and accept data in read mode without any wait inserted, slave response (from OCP-I/OCP-IA) can be used directly for the flow control.

Error Management

The VBUS master interface on the VLYNQ does not support error handling. If a SERRORI or ERR on SRESPI is detected during a write or read command, an interrupt is generated and the address is saved to the V2O_IRQ_STATUS register. The generated interrupt is a pulse that is ORed with VLYNQ_INTPLS_O and sent out as V2O_INTPLS_O. This interrupt is disabled after a reset and must be enabled by setting bit[4] of V2O_CFG_REG.

18.11.2.2 Implementation

The interface runs at 1 × OCP clock mode and is clocked by a balanced 100-MHz clock input.

Figure 18–12 through Figure 18–14 show the FSMs for handling VBUS master transactions. VBUS single accesses are translated directly to OCP single accesses. VBUS burst accesses are converted to four-two-two-last OCP burst transactions. If the VBUS transfer word count is not divisible by four, the last non-four-word burst is done as OCP single accesses.

Figure 18–12. OCP Master Interface FSM

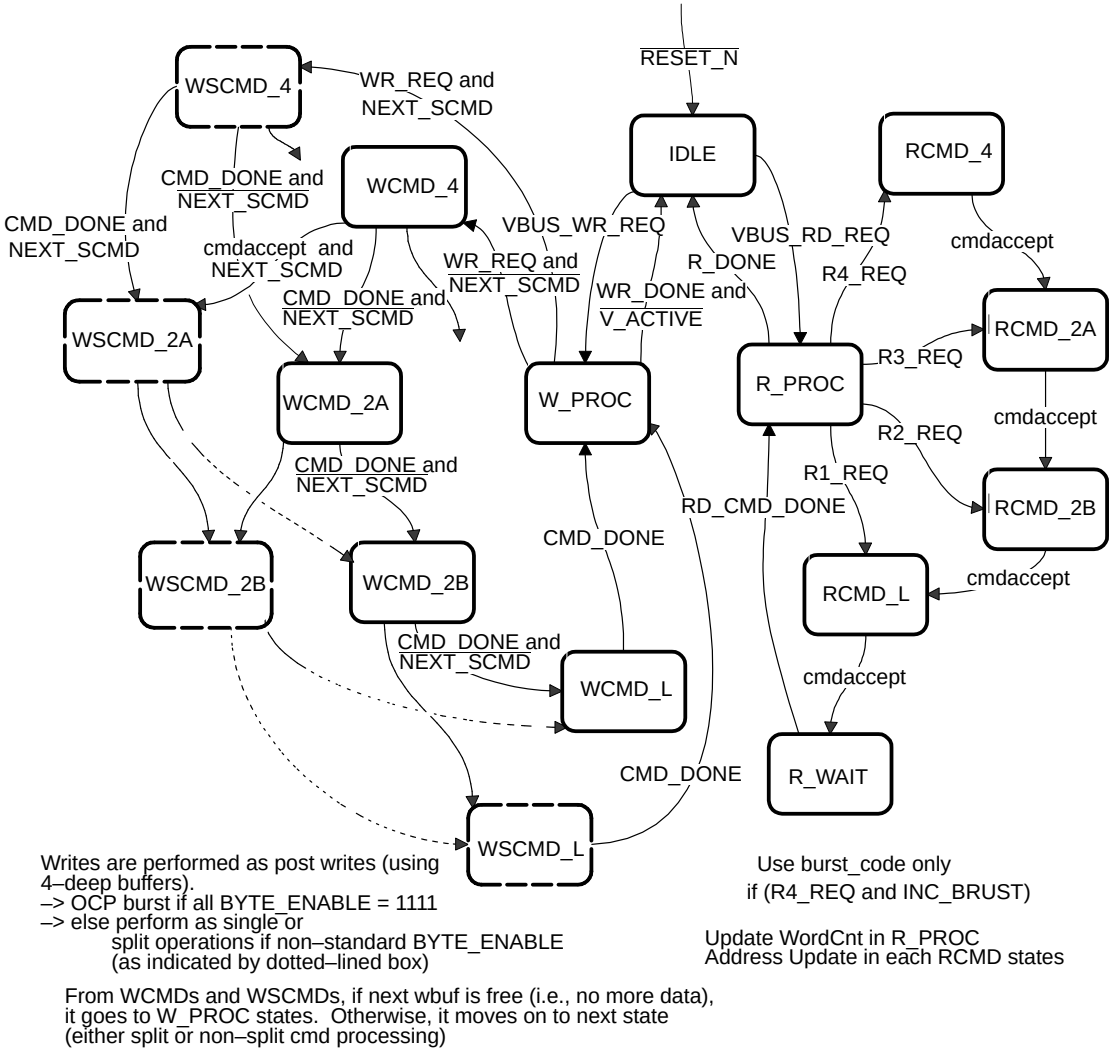


Figure 18–13. OCP Master Interface – Split Transfer Control FSM

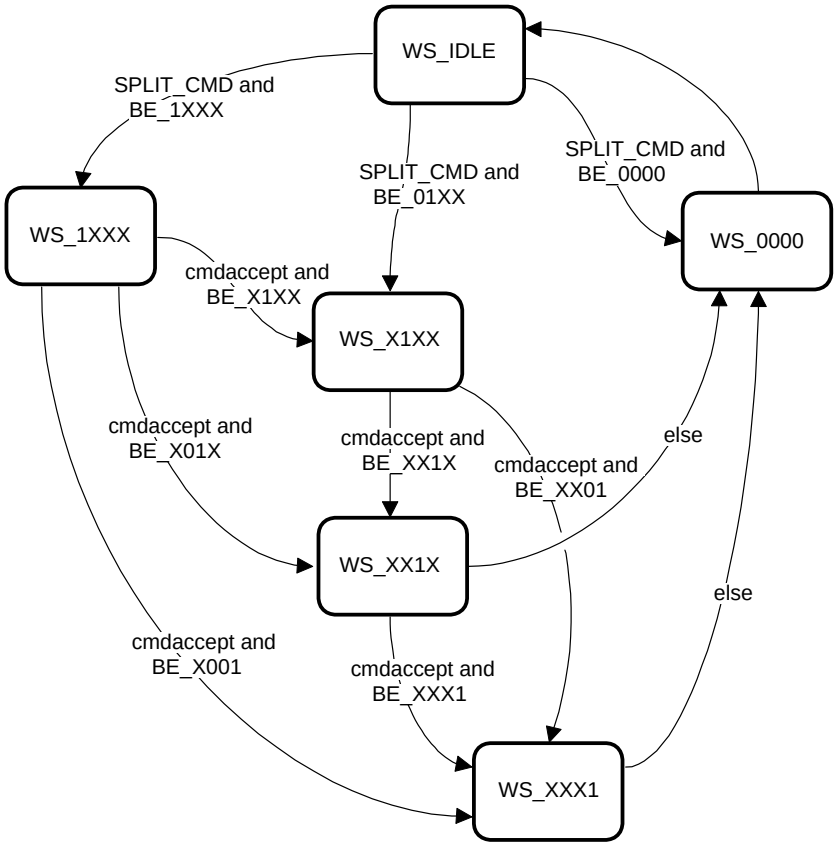
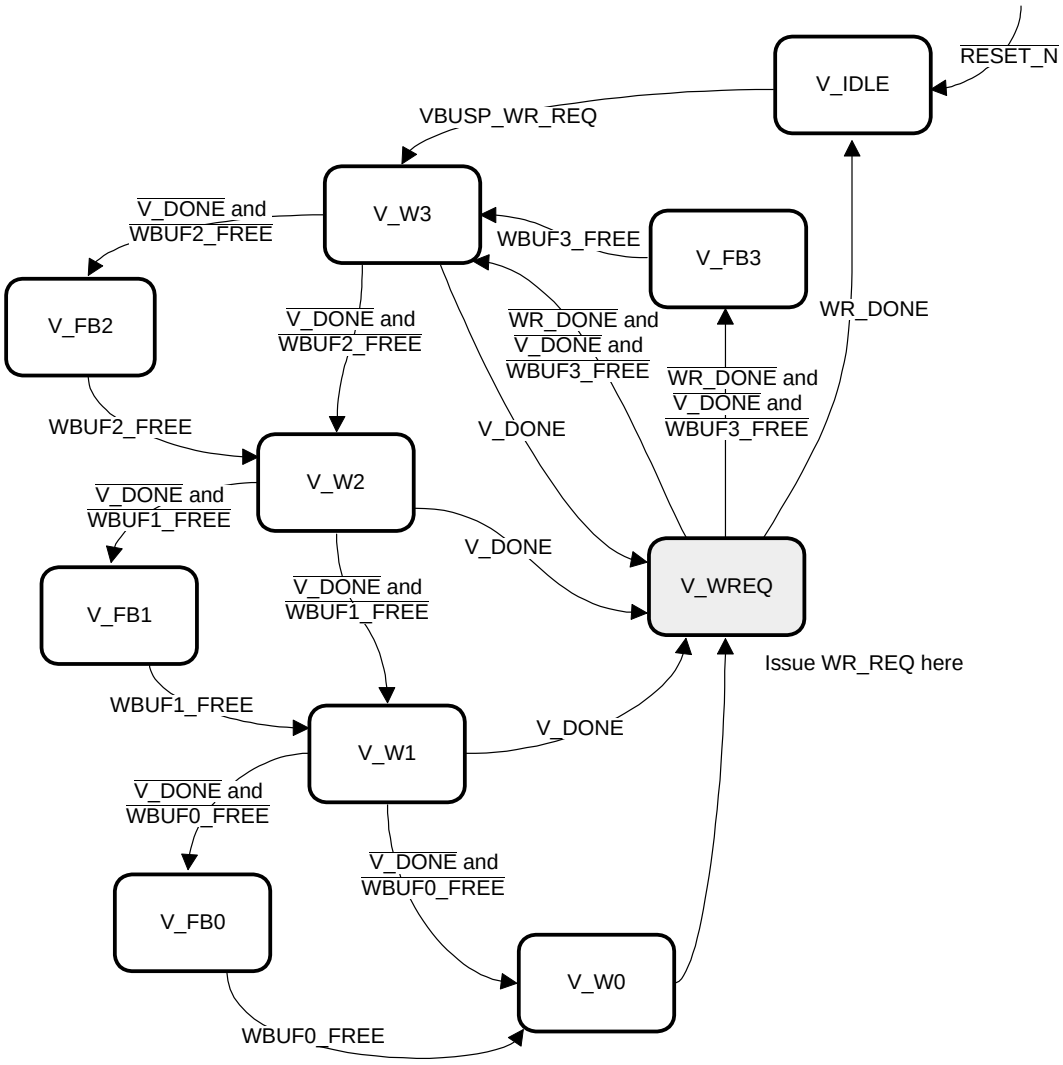


Figure 18–14. OCP Master Interface – VBUS Write Buffer Control FSM



18.12 VLYNQ Submodule

18.12.1 VLYNQ Module Register Memory Map

The VLYNQ module has 256 bytes allocated for register space. The first 128 bytes map to the VLYNQ configuration registers maintained by the local VLYNQ register control module, and the second 128 bytes map to the remote configuration registers physically located in the remote device linked by the VLYNQ serial interface. Any access to the second set of registers causes VLYNQ to issue a read or write VLYNQ packet to be transmitted and completes only if LINK is established between two devices.

BASE ADDRESS: V2O_REG_START_ADDR

Table 18–42. VLYNQ Register Mapping – Local

Address Offset	Register
0x00	Revision/ID register
0x04	Control register
0x08	Status register
0x0C	Reserved
0x10	Interrupt status/clear register
0x14	Interrupt pending/set register
0x18	Interrupt pointer register
0x1c	TX address map
0x20	RX address map size 1
0x24	RX address map offset 1
0x28	RX address map size 2
0x2c	RX address map offset 2
0x30	RX address map size 3
0x34	RX address map offset 3
0x38	RX address map size 4
0x3c	RX address map offset 4
0x40	Chip version register
0x44–0x5c	Reserved
0x60	Interrupt vector 3–0
0x64	Interrupt vector 7–4
0x68–0x7c	Reserved for interrupt vectors 8–31

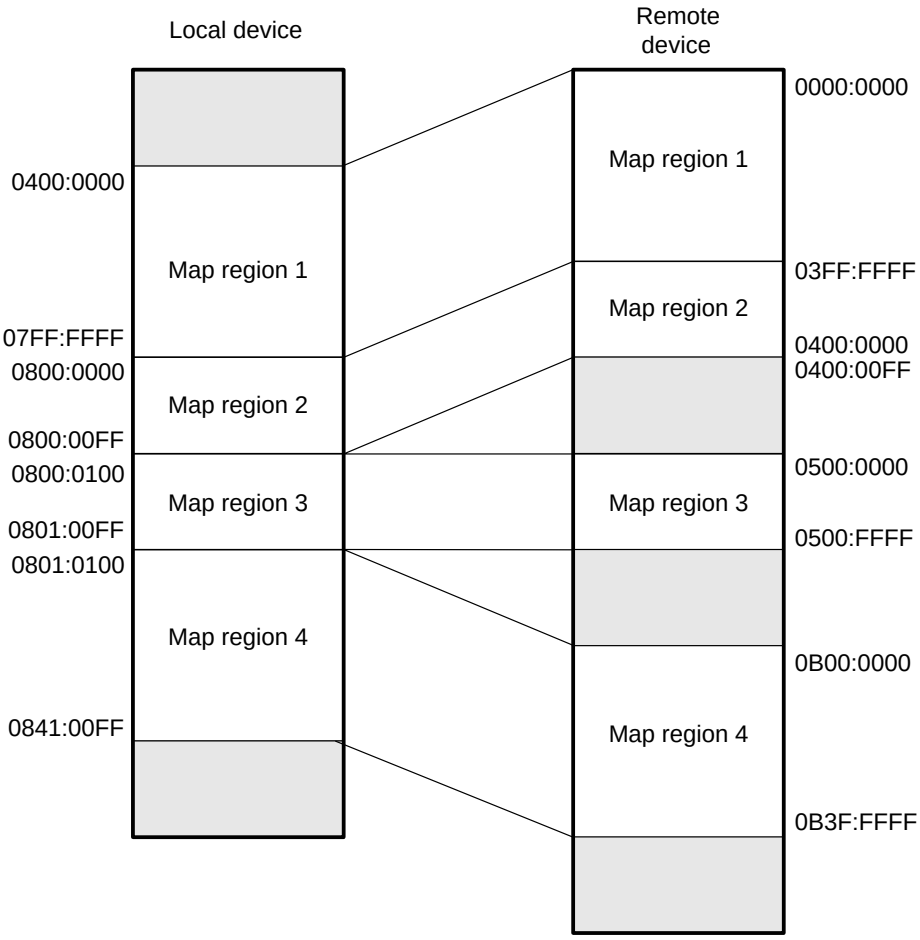
Table 18–43. VLYNQ Register Mapping – Remote

Address Offset	Register
0x80	Remote revision register
0x84	Remote control register
0x88	Remote status register
0x8c	Reserved
0x90	Remote interrupt status/clear register
0x94	Remote interrupt pending/set register
0x98	Remote interrupt pointer register
0x9c	Remote TX address map
0xa0	Remote RX address map size 1
0xa4	Remote RX address map offset 1
0xa8	Remote RX address map size 2
0xac	Remote RX address map offset 2
0xb0	Remote RX address map size 3
0xb4	Remote RX address map offset 3
0xb8	Remote RX address map size 4
0xbc	Remote RX address map offset 4
0xc0	Remote chip version register
0xc4–0xdc	Reserved
0xe0	Remote interrupt vector 3–0
0xe4	Remote interrupt vector 7–4
0xe8–0xfc	Reserved for remote interrupt vector 8–31

18.12.2 VLYNQ Module Address Translation

VLYNQ allows each receive packet address to be translated into one of four mapped regions. No restrictions are placed on the size or offset of each mapped region except that they must be aligned on 32-bit words. This is accomplished via register pairs that define the size and offset of each mapped region. Figure 18–15 illustrates one possible memory map.

Figure 18–15. Address Translation Example (Single-Mapped)



The translated address for packets received on the serial interface is determined as follows:

```
If (RX Packet Address) < (RX Address Map Size 1 Regis-
ter) {
    Translated Address = (RX Packet Address) + (RX Ad-
dress Map Offset 1 Register)
  } Else {
       RX Packet Address –= (RX Address Map Size 1 Reg-
ister)
       if (RX Packet Address) < (RX Address Map Size 2
Register)
          Translated Address = (RX Packet Address) +
(RX Address Map Offset 2 Register)
       } Else {
            RX Packet Address –= (RX Address Map Size 2
Register)
            if (RX Packet Address) < (RX Address Map
Size 3 Register) {
              Translated Address = (RX Packet Address)
```

```
+ (RX Address Map Offset 3 Register)
            } Else {
                  RX Packet Address –= (RX Address Map
Size 3 Register)
                  if (RX Packet Address) < (RX Address
Map Size 4 Register) {
                    Translated Address = (RX Packet Ad-
dress) + (RX Address Map Offset 4 Register)
                  } Else {
                       RX Packet Address = 0x0
                       Generated address translation in-
terrupt
                  }
            }
    }
  }
```

18.12.3 VLYNQ ASIC Memories

The VLYNQ IP core uses the following two-port compiler memories as command and data FIFOs:

- MG00024032020: Quantity 2
- MG00024036020: Quantity 2

There is no MEMBIST for these memories in the core. Instead, there is a DFT memory collar around each memory to add controllability on the Q output of the memory during test mode. In normal mode, a dedicated FIFO test program is used to verify the functionality of the memories.

18.12.4 VLYNQ Initialization

18.12.4.1 Serial Bus Width Configuration

The width of the serial interface is determined from the state of the VLYNQ_TXD_I signals during the device reset condition. The VLYNQ disables the VLYNQ_TXD output during the V2O_RST_N low period and 2000 cycles afterward to allow the VLYNQ_TXD pins to reach their intended state, based on external pullup or pulldown resistors. The VLYNQ_TXD_I is then latched at the end of this period and the width of the serial interface is set according to the value on the VLYNQ_TXD_I pins + 1, as shown in Table 18–44.

Table 18–44. VLYNQ Serial Interface Width Configuration

VLYNQ_TXD_I	Serial Bus Width (Total VLYNQ Pins)
0	1 (3-pin VLYNQ)
1	2 (5-pin VLYNQ)

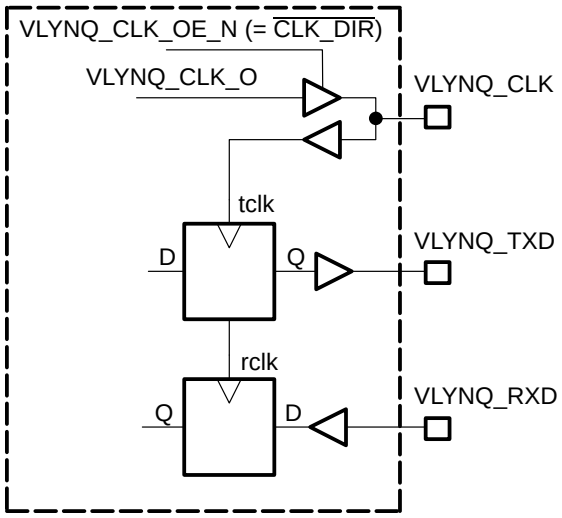
For VLYNQ2OCP, the VLYNQ_TXD_I signals are driven by an internal configuration register. The VLYNQ module is held at reset by a software reset regis-

ter after a pin reset. The software takes the VLYNQ out of reset during device initialization by clearing the software reset register and setting the SWIDTH configuration register. Then, the VLYNQ module latches the width value as previously described. This software configuration scheme removes the need for any external pullup/pulldown resistors and allows the configuration to take place beyond the 2000 cycles following the pin reset.

18.12.4.2 Clock Direction Initialization

The VLYNQ module serial clock direction is software configurable by writing to CLKDIR of the control register. The serial clock is sourced by the internal VBUS clock (CLKDI =1) or by an external clock (CLLKDIR= 0). The default value of the CLKDIR bit is set during the reset with the value on the VLYNQ_DEFAULT_CLKDIR input signal. In either case, the VLYNQ serial clocks (VLYNQ_RCLK_I and VLYNQ_TCLK_I) always come from the VLYNQ_CLK input pin, as shown in Figure 18–16, in order to constrain the VLYNQ timing with respect to a device clock pin.

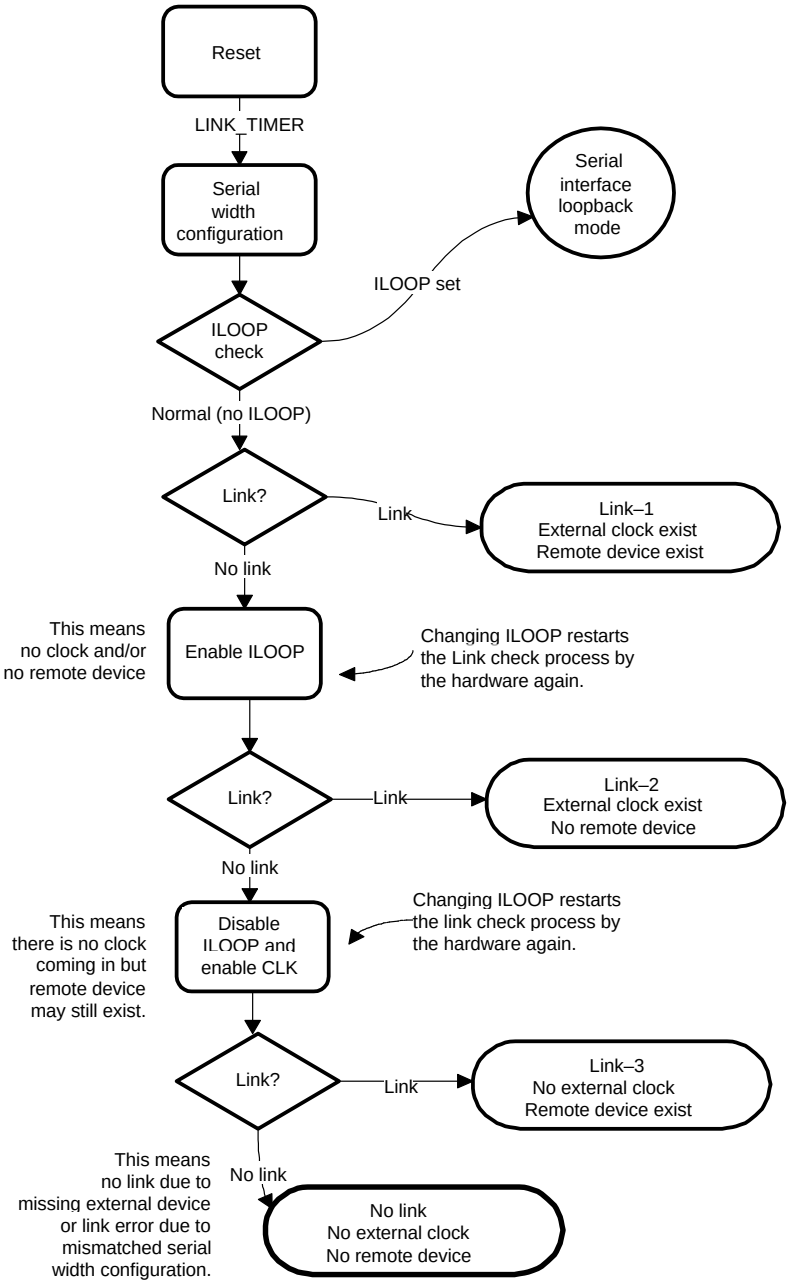
Figure 18–16. VLYNQ Clock Direction Configuration



18.12.4.3 Linking Process Overview

Unless the clock direction is fixed and set by the default CLK direction signal, the link process must follow the algorithm depicted in the flow diagram, as shown in Figure 18–17, to avoid clock contention. After establishing a successful link, the VLYNQ device enumeration process can continue configuring the VLYNQ connection.

Figure 18–17. VLYNQ – LINK Process Flowchart



18.13 VLYNQ2OCP Configuration

Table 18–45 lists the VLYNQ2OCP configuration registers. Table 18–46 through Table 18–49 describe the register bits.

Table 18–45. VLYNQ2OCP Configuration Registers

Name	Function	R/W	Offset
V2O_CFG_REG	VLYNQ2OCP configuration register	R/W	0x100
V2O_ADDR_FAULT	V2O wrapper error IRQ status register (for debug)	R/W	0x104
VLYNQ_BUSY	VLYNQ busy status register (for idle-mode configuration)	R	0x108
V2O_DMAREQ_EN	VLYNQ2OCP DMA request generation enable Write 1 to trigger the initial DMA request generation. Write 0 to disable the DMA request generation or to retrigger the initial DMA request.	R/W	0x10C

Table 18–46. VLYNQ2OCP Configuration Register (V2O_CFG_REG)

Bit	Name	Function	R/W	Reset Value
31:0	RESERVED	Reserved	R	0x0
4	OCPM_EIRQEN	V2O master error interrupt enable (write 1 to enable)	R/W	0x0
3:1	VLYNQ_SWIDTH	Serial bus width value + 1 becomes the width of TXD/RXD bus	R/W	0x0
0	VLYNQ_SWRESET	VLYNQ module software reset (active high)	R/W	0x1

Table 18–47. V2O Wrapper Error IRQ Status Register (V2O_ADDR_FAULT)

Bit	Name	Function	R/W	Reset Value
31:2	ERR_ADDR	Word address of error transfer	R/W	0x0
1	ERR_RWN	0: Write error 1: Read error	R/W	0x0
0	ERR_FLAG	1: Error	R/W	0x0

Table 18–48. VLYNQ Busy Status Register (VLYNQ_BUSY)

Bit	Name	Function	R/W	Reset Value
31:1	Reserved	Always read as 0	R	0x0
0	SBUSY	0: Indicates VLYNQ has no data in the pipeline 1: Indicates VLYNQ has data in the pipeline	R	0x0

Table 18–49. VLYNQ2OCP DMA Request Generation Enabale Register (V2O_DMA_REQ_EN)

Bit	Name	Function	R/W	Reset Value
3	Reserved	Always reads 0.	R	0x0
1:2	V2O_RESETDONE	VLYNQ2OCP reset done indicator. System reset and software reset not asserted. Reads 0x1 if VLYNQ software reset is cleared.	R	0x0
0	DMA_EN	Write 1 to trigger the initial DMA request generation. Write 0 to disable the DMA request generation or to retrigger the initial DMA request.	R/W	0x0

The VLYNQ configuration registers are located in the OCP2VBUS wrapper module and are typically accessed only by an OCP host. An external host can access these registers via an OCP_I/OCP_T loopback path if necessary.

18.13.1 Clock Management

The VLYNQ2OCP has one system clock (V2O_OCP_CLK) and two VLYNQ serial clocks that are used in functional mode. The system clock V2O_OCP_CLK can be disabled by an V2O_CLK_EN input signal when the module is not used or the interface is in idle mode. The ONE_GATED_ENABLE_SYNCHRO and ONE_GATED_CLK macros are used to synchronize the enable and multiplex the clock, as shown in Figure 18–18. The VLYNQ serial clocks are not gated but are multiplexed with a test scan clock using the same ONE_GATED_CLK macros (to keep the clock paths equal).

The VLYNQ2OCP generates a serial VLYNQ clock (VLYNQ_CLK_O) from V2O_OCP_CLK. This output clock can be disabled by the VLYNQ_CKOUT_EN input signal. This signal is also synchronized to VLYNQ_CLK_O out of VLYNQ IP using the ONE_GATE_ENABLE_SYNCHRO macro, as shown in Figure 18–19.

Users must always disable VLYNQ_CKOUT_EN before disabling V2O_CLK_EN and enable V2O_CLK_EN before enabling VLYNQ_CKOUT_EN.

Figure 18–18. VLYNQ2OCP Clock Enable/Reset Synchronization Diagram

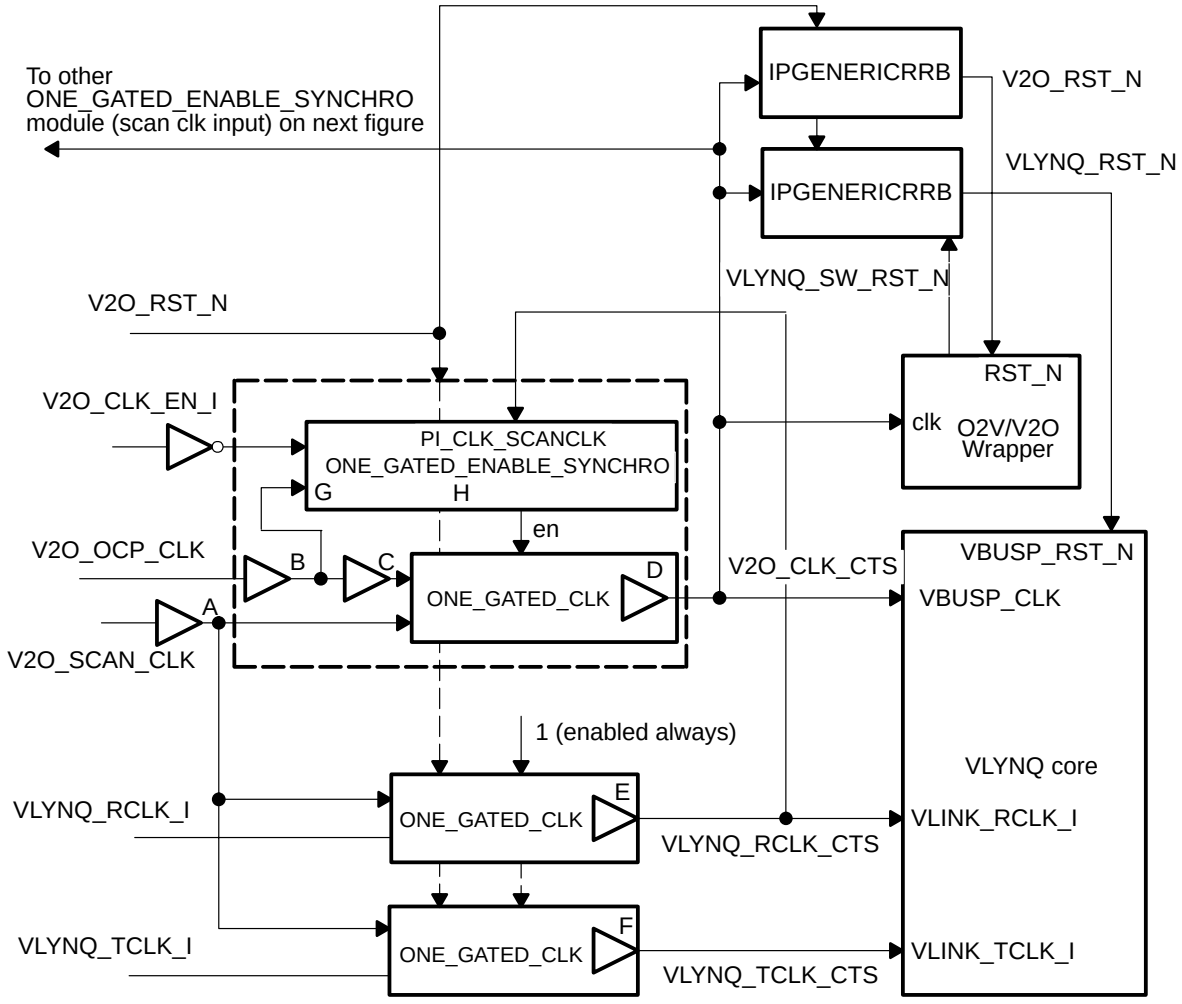
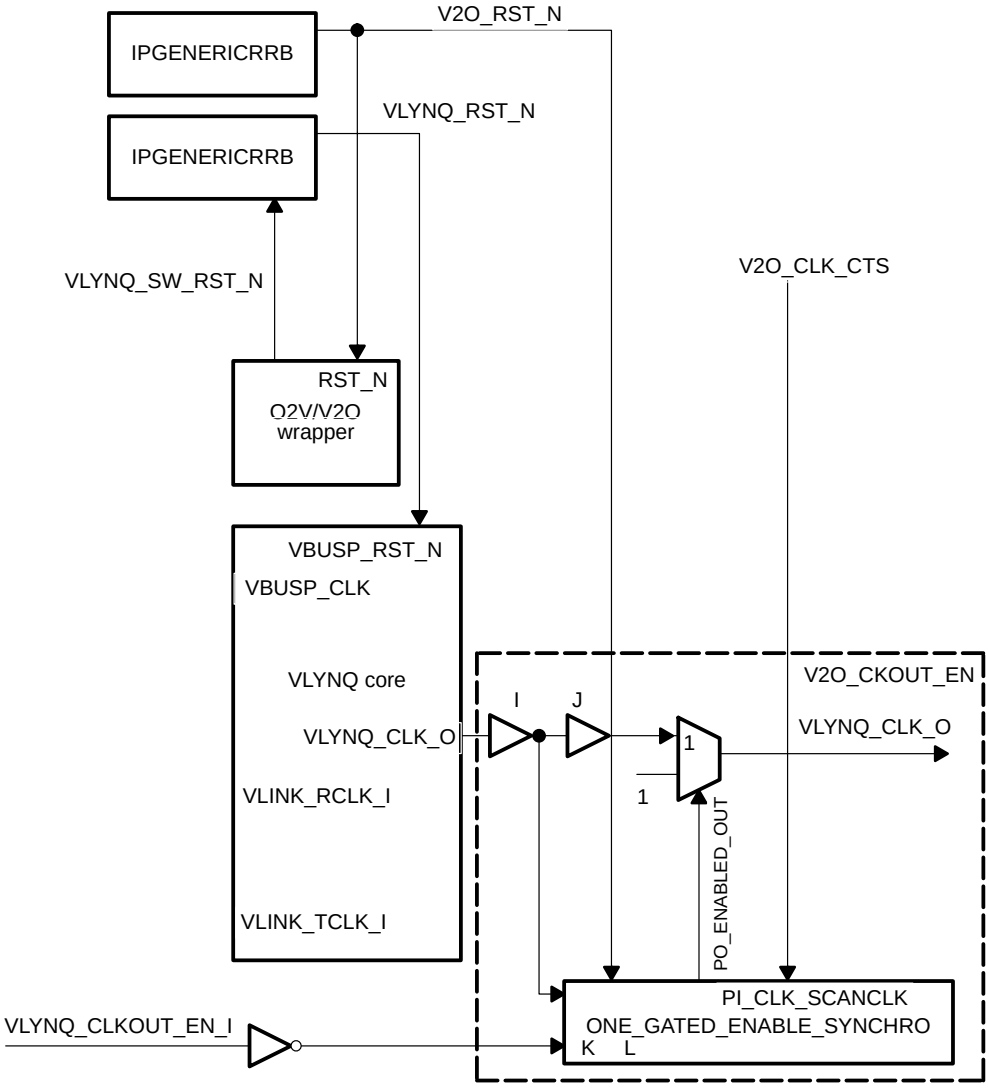


Figure 18–19. VLYNQ Output Clock Enable Synchronization Diagram



18.13.2 Reset Management

System Reset

The VLYNQ2OCP receives a single system-reset signal (V2O_RST_N). This signal is used directly to reset the VLYNQ2OCP input clock enable and input the clock buffer macros, as shown in Figure 18–18. A synchronized reset is used to reset the rest of the VLYNQ2OCP logic. Inside the VLYNQ core, the reset goes through an additional synchronizer for resetting registers in the VLYNQ serial clock domain.

Software Reset

In addition to the system reset, the VLYNQ2OCP supports a software reset for resetting the VLYNQ IP submodule. Bit[0] of V2O_CFG_REG is used for software reset for the VLYNQ IP module. After a system reset, this bit is set high, holding the VLYNQ IP module in reset. The software must write 0 to this bit to take the VLYNQ IP module out of reset and start the VLYNQ initialization process.

The VLYNQ2OCP does not support the software reset triggered by an OCP write transaction.

18.14 VLYNQ2OCP Power-Down Mode

18.14.1 VLYNQ2OCP Implementation

The VLYNQ2OCP provides two types of power-saving mode operations on the VLYNQ interface. The first one disables the entire VLYNQ interface module when the host devices are in one of the idle modes that do not require the VLYNQ interface. The second one slows down the VLYNQ serial clock output during normal mode when there is either no data flow or low data flow in the VLYNQ pipeline.

18.14.2 Idle Mode Implementation

The idle mode is always initiated by the host (OMAP) but can be awakened by either the host or a peripheral device (for example, ACX111). Two GPIOs are used to communicate an idle/wake mode change request/acknowledge between two devices, as described below in the system application examples (OMAP730-ACX111). VLYNQ2OCP implements clock gating as described in Section 18.13.1, *Clock Management***.**

Putting Into IDLE Mode (Host Initiated Only)

1) The host (OMAP) puts the ACX111 in a power-down mode based on the application and/or based on the current traffic (or lack thereof).
2) The OMAP sends an idle request packet (writes to a register or sends an interrupt) to ACX111.
3) ACX111 acknowledges the idle request by setting GPIO_TO_OMAP low (GPIO_TO_OMAP is usually set high). If the ACX111 VLYNQ is sourcing the VLYNQ clock, ACX111 stops the VLYNQ clock.
4) The OMAP waits for GPIO_TO_OMAP low and shuts down the VLYNQ clock if it is sourcing the clock. After that, the OMAP sets GPIO_FROM_OMAP low and shuts down the VBUS clock to the VLYNQ module.
5) ACX111 monitors GPIO_FROM_OMAP low level, pulls GPIO_TO_OMAP high (to create a pulse), shuts down the VBUS clock to the VLYNQ, and goes into idle mode.
6) Result: Both VLYNQ modules are in idle mode.

Waking Up From IDLE (Host Initiated)

1) The host (OMAP) decides to wake the VLYNQ interface and ACX111.
2) The OMAP enables the VBUS clock to VLYNQ.
3) The OMAP pulls GPIO_FROM_OMAP high indicating wake-up.
4) ACX111 samples GPIO_FROM_OMAP going high, enables the VBUS clock to its VLYNQ, and pulls GPIO_TO_OMAP low indicating it is awake.

At the same time, if ACX111 is sourcing the clock, it enables the VLYNQ clock.

5) The OMAP monitors GPIO_TO_OMAP low level and starts the VLYNQ serial clock, if it is sourcing the clock.

6) ACX111 pulls GPIO_TO_OMAP high (to create a pulse) after a timer timout (two 32-kHz clock periods).

7) Result: Communication resumes between the two devices.

Waking Up From Idle (ACX111 Initiated)

1) ACX111 is awakened by activities in the physical layer (PHY) and proceeds to wake up the host.

2) ACX starts its VBUS clock and pulls up GPIO_TO_OMAP low indicating a host wake-up.

3) The OMAP samples GPIO_TO_OMAP going low and enables the VBUS clock to its VLYNQ and starts the VLYNQ serial clock if it is to source the clock. The OMAP also pulls up GPIO_FROM_OMAP high to indicate the host is enabled.

4) ACX111 samples GPIO_FROM_OMAP going high and enables the VLYNQ clock if it is to source the clock. It also pulls GPIO_TO_OMAP high (for pulsing).

5) Result: Communication resumes between the two devices.

Summary of OMAP Operation

The following sequence describes the operation of each device in the above process.

- ❏ Going to idle:
  - ■ Writes to the ACX111 mode register
  - ■ Waits for GPIO_TO_OMAP to go low
  - ■ Disables the VLYNQ clock out if CLKDIR is 1 in VLYNQ
  - ■ Disables the VBUS clock
  - ■ Sets GPIO_FROM_OMAP to low
- ❏ Wake up (OMAP initiated)
  - ■ Enables the VBUS clock
  - ■ Sets GPIO_FROM_OMAP to high
  - ■ Waits for GPIO_TO_OMAP to go low
  - ■ Enables the VLYNQ clock if CLKDIR is 1 in VLYNQ
- ❏ Wake up (ACX111 initiated)
  - ■ Detects GPIO_TO_OMAP going low
  - ■ Enables the VBUS clock
  - ■ Enables the VLYNQ clock if CLKDIR is 1 in VLYNQ
  - ■ Sets GPIO_FROM_OMAP high

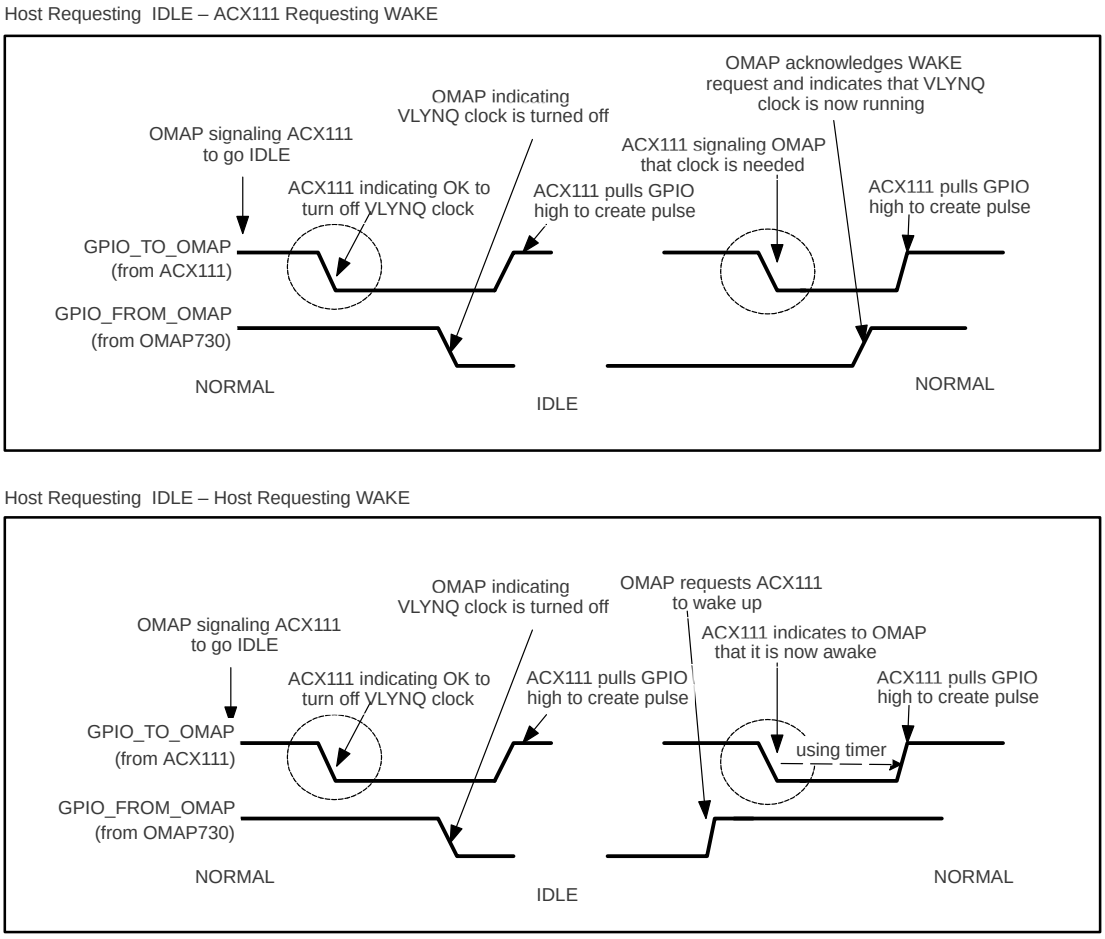
Summary of ACX111 Operation

- Going to IDLE:
  - Detects OMAP writing to its mode register
  - If *ok to go idle*, disables the VLYNQ clock if CLKDIR is 1 in VLYNQ
  - Sets GPIO_TO_OMAP low
  - Waits for GPIO_FROM_OMAP to go low
  - Pulls GPIO_TO_OMAP high (pulsing)
  - Disables the VBUS clock
- Wake up (OMAP initiated)
  - Detects GPIO_FROM_OMAP going high
  - Enables the VBUS clock
  - Enables the VLYNQ clock out if CLKDIR is 1 in VLYNQ
  - Pulls GPIO_TO_OMAP low
  - Pulls GPIO_TO_OMAP high (pulsing)
- Wake up (ACX111 initiated)
  - Enables the VBUS clock
  - Pulls GPIO_TO_OMAP low
  - Waits for GPIO_FROM_OMAP to go high
  - Enables the VLYNQ clock if CLKDIR is 1 in VLYNQ
  - Pulls GPIO_TO_OMAP high (pulsing)

Ok to Go Idle

- The host (OMAP) ensures no further VBUS transaction to its VLYNQ module prior to requesting ACX111 to go idle (no OMAP to ACX111 request).
- ACX111 monitors the local VLYNQ_BUSY signal for *ok to go idle* or relies on current DMA status to determine that there are no pending data transfers.

Figure 18–20. GPIO Timing Diagram for Idle/Wake Request/Acknowledge



The OMAP GPIO input logic relies on single edge detection and requires pulses to be at least two 32-kHz clock periods long. If needed, OMAP can send an interrupt to ACX111 to pull its GPIO high during the wake-up process.

18.14.3 VLYNQ Serial Clock Frequency Slow Down

If the VLYNQ_CLK is sourced internally, the CLK_DIV parameter in the VLYNQ IP core can be reprogrammed to send a divided-down clock (1/1 – 1/8) out to VLYNQ_CLK_O to save normal mode power at the expense of reduced performance. To avoid any glitches while changing frequency, use VLYNQ_CLKOUT_EN_I to disable VLYNQ_CLK_O while changing the value of CLK_DIV.

18.15 VLYNQ2OCP Timing

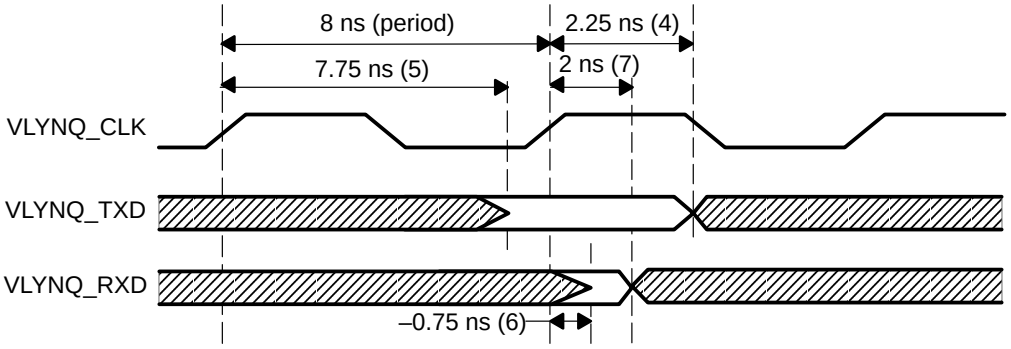
Table 18–50 lists the VLYNQ I/O timing requirements. Figure 18–21 shows the VLYNQ timing.

Table 18–50. VLYNQ2OCP – VLYNQ I/O Timing Requirements

VLYNQ2OCP – VLYNQ Timing (CLK at125 MHz – 25% over 100-MHz Target)

No.	Parameter	Min	Max	Unit
1	Clock period (Tperiod), VLYNQ_CLK	8		ns
2	Pulse duration, VLYNQ_CLK high	TBD		ns
3	Pulse duration, VLYNQ_CLK low	TBD		ns
4	Delay time, VLYNQ_CLKC to VLYNQ_TXD invalid	2.25		ns
5	Delay time, VLYNQ_CLKC to VLYNQ_TXD valid		7.75	ns
6	Setup time, VLYNQ_RXD before VLYNQ_CLKC	–0.75		ns
7	Hold time, VLYNQ_RXD after VLYNQ_CLKC	2.0		ns

Figure 18–21. VLYNQ Timing Diagram



Chapter 19

Security Features

This chapter describes the security features of the OMAP730 multimedia processor.

Topic **Page**

19.1 Security Features

The OMAP730 security scheme relies on the OMAP3.2 secure mode. The distributed security on the OMAP3.2 platform is a Texas Instruments solution to address m-commerce and security issues within a mobile phone environment.

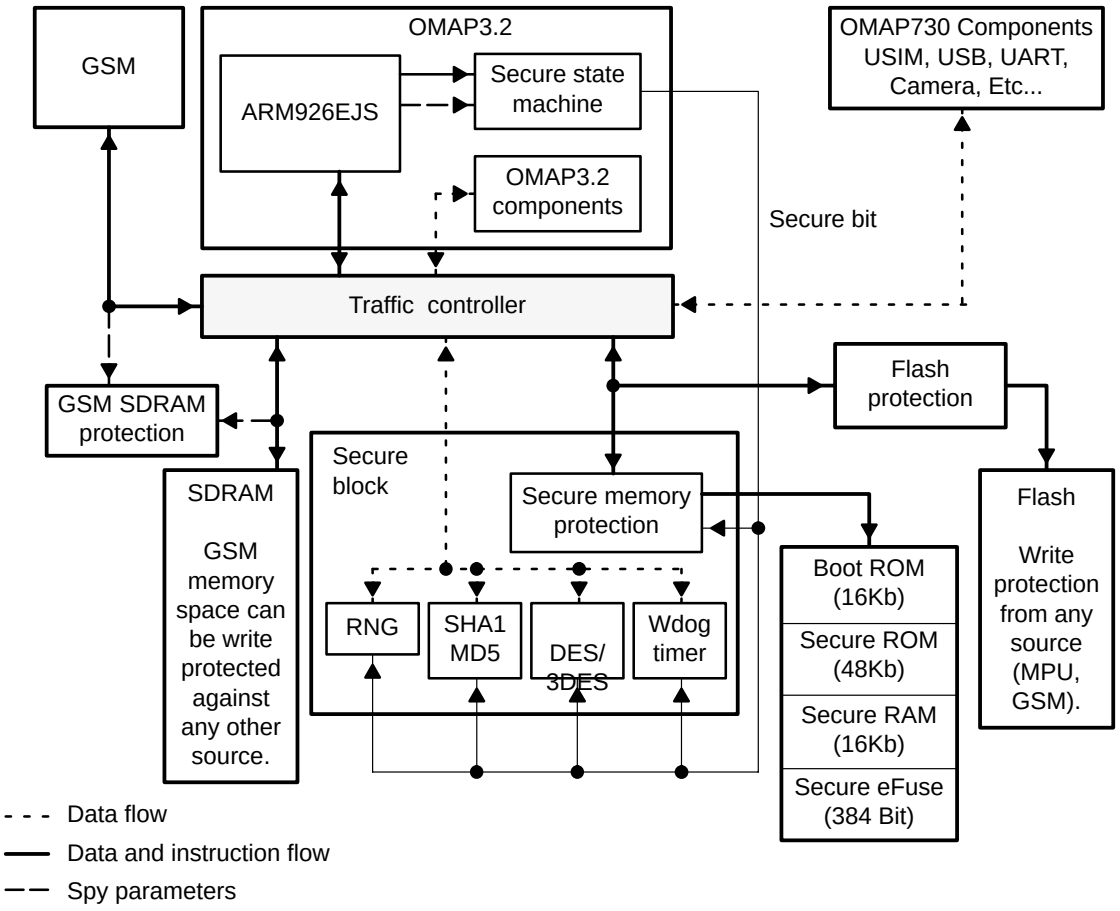
A few operating systems have been certified as secure for specific financial or safety-critical applications. Some general-purpose operating systems claim to have security built in, but their fragility is well publicized. For some critical security applications, the OS cannot be trusted. Only a well combined hardware and software solution can provide adequate protection.

The OMAP3.2 secure mode was developed to bring hardware robustness to the overall OMAP730 security scheme.

For more specific details on OMAP730 security, please contact your TI representative.

Figure 19–1 shows the OMAP730 processor with the security area highlighted.

Figure 19–1. OMAP730 Security Area



19.1.1 Hardware Security

The OMAP73x secure environment is a hardware-based security feature available on the MPU side, where critical code and data can be executed securely and sensitive information can be hidden from the outside world with the help of the following security hardware:

- Secure mode (secure execution): the ability to execute trusted code from secure ROM or secure RAM
- Security state machine
- Hardware-based entry and exit sequence
- Security control register
- Security hardware accelerators
- DES/3DES, SHA-1/MD5 hardware blocks for performing cryptographic operations
  - A true hardware random number generator
- Secure watchdog timer
- Secure eFuses programmed at the factory with keys used for hardware-based key management (authentication and encryption)
- Secure RAM
- Secure ROM, containing security services such as secure storage and cryptographic libraries

19.1.2 Secure Mode

The secure mode can be viewed as a third privilege level on the ARM926EJS subsystem. Its activation relies on the presence of special-purpose hardware for protecting sensitive information from access by nontrusted software. The secure mode is set with the assertion of a dedicated security signal. It propagates across the system and creates a boundary between resources that trusted software can access and those available to any software.

The security relies on the fact that in secure mode, only trusted code stored in secure memories can be executed on the platform.

The robustness of the secure mode relies heavily on the secure code itself. There can be no possible flaws by which nontrusted code can make secure code perform tasks it must not, nor fool the hardware into entering secure mode. If the boundary is created properly, there is no way to use the normal operation of the processor to move information from inside the boundary to outside, except through controlled operations.

19.1.3 Security State Machine

There are two kinds of security hardware:

- Logic that controls the security signal: the security state machine (SSM) briefly discussed in this section.
- Hardware resources restricted to secure mode:
  - Security control register
  - Security hardware accelerators
  - Secure watchdog
  - Secure eFuse
  - Secure ROM
  - Secure RAM

The SSM monitors the conditions of entering in secure mode, setting/unsetting the security signal, and detecting secure mode hardware violations.

The security state machine is tightly coupled to the low-level assembly code of the entry sequence. It reacts on events generated by the entry sequence on various probed signals.

When the secure signal is set, it propagates in the system and enables access to the secure resources.

Note:

Access to the ROM, SRAM, and the keys in secure mode is restricted to ARM926EJS by design (tie-off configuration bit on OMAP3.2 interface).

19.1.4 Secure Mode Options

The secure mode on OMAP730 is implemented with the following options with respect to the OMAP3.2 secure mode:

- ARM926EJS MMU is enabled in secure mode.
- ARM926EJS data and instruction cache are enabled in secure mode.
- Interrupt is enabled in secure mode.
- Interrupt is remapped by software in secure mode using the MMU.
- SWI does not initiate exit from secure mode.
- Abort does not initiate exit from secure mode.

Security Layer

The security layer module generates the CS for the secure memory (ROM, SRAM, and eFuse), independently of the security signal that propagates directly to the secure memories.

The security layer also isolates the EMIFS address and data bus from the external device I/O pin to prevent secure data/address from being probed on the external flash interface.

19.1.5 ULPD

On security violation, the ULPD generates a global system reset of the OMAP730.

The ULPD also performs the initialization sequence of the eFuses in OMAP730 at power up.

19.2 Security Control Register

The security control register controls various security settings, such as:

- Secure watchdog
- Access to secure hardware accelerator
- Debug/emulation capabilities

The security control register is accessed (R/W) only in secure mode, namely when the security signal is asserted. Otherwise, access to the security control register is denied and returns dummy data: all zeros.

The security control register is reset at power-on reset ($\overline{\text{PWRON}}$_$\overline{\text{RESET}}$), and its reset value depends on the SECURITY_DEVICE_TYPE [1:0] signals the configuration eFuse generates.

Several bits in the security control register are one-time programmable. Therefore, after the first write access, these bits are locked in a read-only mode and cannot be written again until the next power-up reset occurs.

For details on the security control register (SECCTRL), see Section 4.7, *OMAP730 Configuration Registers*.

19.3 Security eFuse

Two bit fields of electrical fuses (eFuses) are part of this secure environment: field B (128 bits) and field C (256 bits). They are accessible through the external memory interface slow (EMIFS). Field B can be read when the SECURE bit is active. Field C can be read when the SECURE and CKEY bits are active.

19.3.1 Security Keys

The security keys B (128 bits) and C (256 bits) are made of eFuses. The B key is a 128-bit number, which is device-specific and generated randomly at fuse-burning time during probe test. The C key is a 256-bit number, which is common to all devices.

After the security eFuses are programmed, the access path through JTAG is permanently disabled using an additional eFuse that sits in the production eFuse module.

The B and C keys can be accessed only in secure mode. In addition, access to the C key depends on the value of the CKEY bit in the security control register.

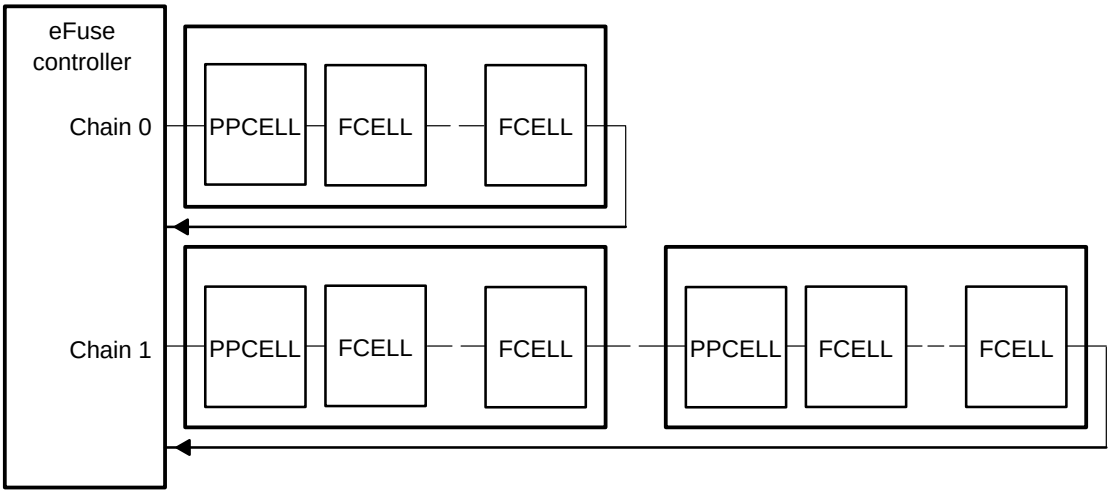
When access to security keys is denied, it returns dummy data: all zeros.

19.3.2 Electrical Fuse

Figure 19–2 shows the electrical fuse implementation, which has three building blocks:

- eFuse controller
- eFuse cell (FCELL)
- Program protection cell (PPCELL)

Figure 19–2. eFuse Implementation



Each eFuse cell contains an electrically alterable fuse, supporting circuitry, and an output. It can be permanently set to logic 1 instead of its natural unprogrammed state, logic 0. A program protection cell is similar to a fuse cell except that once programmed, it prohibits programming of subsequent fuses.

The fuse chains consist of one or more fuse cells and a program protection cell (PPCELL).

19.4 Boot ROM

The MPU boots from the boot ROM at EMIFS CS0. The boot ROM is a 32-bit wide, 64K-byte memory (although its address space is reserved for 256K bytes). Access is allowed to the secure parts of the boot ROM in secure mode only.

19.4.1 Access Control Management

The ROM sections have different security levels:

- Public ROM, or boot ROM (16K bytes + 128 bytes), which is always accessible
- Secure ROM (48K bytes - 128 bytes), which is accessible only in secure mode
- Secret ROM (128 bytes), part of secure ROM, is accessible only in secure mode, depending on the value of the PKEY bit in the security control register

Access to the ROM is restricted to ARM926EJS by design in OMAP3.2. When access to secure/secret ROM is denied, the ROM returns dummy data (all zeros).

19.5 Secure RAM

This secure environment includes 16K bytes of secure SRAM.

Secure RAM supports byte addressing and is accessible through the external memory interface slow (EMIFS) only when the secure bit is set.

19.5.1 Secure SRAM

The secure SRAM is a 16K-byte program and data SRAM that is connected to OMAP3.2 EMIFS CS0.

The secure SRAM can be accessed (R/W) only in secure mode. In addition, access to the secure SRAM is restricted to ARM926EJS only. When access to secure SRAM is denied, the SRAM returns dummy data: all zeros.

Two 32-bit registers are mapped into the secure SRAM address ranges and shadow the corresponding SRAM locations. These registers are all reset to 1 at power-on reset.

Whenever a BIST operation is started on the secure SRAM, the event is logged in one of these registers.

19.5.2 Access Management

Access to secure RAM must be carefully controlled. Its contents are qualified as secured data, which means that access is allowed in secure mode only.

19.5.2.1 Timing

The secure RAM works at 100 MHz. This device can perform read or write access in 20 ns.

19.6 Secure Watchdog

Secure protection is activated on read/write access to the watchdog OCP registers. To access the secure watchdog, the user must be in secure mode. In other words, the user must enter secure mode at least once every 45 seconds to reload the counter (assuming a 192-MHz timer clock).

To grant any access (read/write) on any OCP register, secure mode must be enabled and the CONF_WD_REG_EN must be set to 1. CONF_WD_REG_EN is configurable in SECCTRL (Bit [1] in the OMAP730 configuration).

Other combinations close access to the watchdog registers and, in that case, an error is sent through the OCP error line of the OCP interface.

When access to the watchdog register is denied, a read returns dummy data (all zeros) and writes are not effective.

To prevent an external user from hacking the code execution, a watchdog counter is implemented. Within a window the watchdog time-out program has defined, the code must reenter the secure mode to reload the counter. Otherwise, a reset is asserted.

The reset event is logged into the ULPD reset status register (Bit [3]).

Reload of the watchdog counter can be restricted to secure-mode-only by setting the CONF_WD_REG_EN in the secure mode control register (SECCTRL).

Table 19–1 describes the OCP register accesses. Table 19–2 describes the reset mode.

Table 19–1. Secure Mode-OCP Register Accesses

Secure Mode	CONF_WD_REG_EN	Access to Watchdog
1	1	Granted/OCP access done
X	X	Rejected/OCP error generated

In addition, if the CONF_WD_OP_DIS is set to 1, the watchdog is in reset mode. CONF_WD_OP_DIS is configurable in SECCTRL(Bit[0] in the OMAP730 configuration.

Table 19–2. Secure Mode-Reset Mode

CONF_WD_OP_DIS	Watchdog Mode
1	Module under reset
0	Run (counting)

Security features are activated/deactivated depending on module integration. The 32-bit watchdog module can be used as a secure watchdog or a nonsecure watchdog.

19.7 Secure Hardware Accelerators

The secure peripherals, RNG, SHA-1/MD5, and DES/3DES, are connected to the MPU TIPB. Depending on the bits set inside the security control register, access to the secure peripherals is restricted to secure mode or not.

The security access restriction is directly managed in the L4 interface.

As described in the secure-mode specifications, one of the secure-mode entry conditions requires that DMA access to MCU TIPB private be disabled. However, the security software is entitled to enable the DMA access on MPU TIPB private in secure mode. It is therefore possible to perform DMA transactions with the secure peripherals in secure mode.

It is also possible to perform DMA transactions with the secure peripherals outside the secure mode, if the proper configuration is set in the security control register.

Nevertheless, it is never possible to perform DMA transactions with the secure peripherals during the execution of the secure-mode entry sequence, whatever the setting in the security control register.

19.7.1 DES/3DES Modules

The DES/3DES security module provides hardware-accelerated data encryption/decryption functions. It can run either the single DES algorithm or the triple DES (TDES or 3DES) algorithm in compliance with the FIPS 46-3 standard. It supports the electronic codebook (ECB) and the cipher block chaining (CBC) modes of operation. It does not support the cipher feedback (CFB) nor the output feedback (OFB) modes of operation in hardware.

This section describes the algorithm basics, architecture overview, operational mode description, register mapping and content, and programming flowchart for the DES/3DES module.

19.7.1.1 Algorithm Basics

The DES algorithm encrypts (enciphers) or decrypts (deciphers) binary-coded information. Encrypting data converts it to an unintelligible form called cipher text. Decrypting cipher text converts the data back to its original form, called plain text. Both enciphering and deciphering operations are based on a binary key. DES is a symmetric algorithm, which means that the encryption and decryption keys are identical. Each triple DES encryption/decryption operation is a compound of DES encryption and decryption operations.

19.7.1.2 DES Algorithm

The DES algorithm generates block ciphers. Block ciphers, as opposed to stream ciphers, operate on blocks of plain text and cipher text. The DES block size is 8 bytes. The DES key consists of 64 binary digits, of which 56 bits are randomly generated and used directly by the algorithm. The other 8 bits can be used for error detection. The 64-bit block of input data to be enciphered is initially permuted, then passed through 16 iterations of a calculation that uses

a cipher function, and finally permutated to the inverse of the initial permutation. At each of the 16 iterations, a 48-bit key computed from the 64-bit input key is applied to one of the 32-bit sub-blocks of the 64-bit input block using the cipher function. The 48-bit key differs at each iteration. The result of the cipher function is a 32-bit sub-block, which is concatenated with the second 32-bit input sub-block. The resulting 64-bit output block of each iteration feeds back the input of the next iteration. See the National Institute of Standards and Technology (NIST), *Federal Information Processing Standard (FIPS)* 46-3, Data Encryption Standard (DES) document for more details on the cipher function.

To decipher, it is only necessary to apply the same algorithm to the enciphered message block, taking care that each iteration of the computation uses the same 48-bit key that was used during enciphering.

19.7.1.3 Triple DES Algorithm

The triple DES algorithm uses the same 64-bit block for ciphers and input data. Let $E_K(I)$ and $D_K(I)$ represent the DES encryption and decryption of block I using DES key K.

For triple DES, the encryption of input block I into a 64-bit output block O is defined as follows:

$$O = E_{K3}(D_{K2}(E_{K1}(I)))$$

For triple DES, the decryption of input block I into a 64-bit output block O is defined as follows:

$$O = D_{K3}(E_{K2}(D_{K1}(I)))$$

19.7.2 Random Number Generator

The random number generator (RNG) module provides a true, nondeterministic noise source for generating keys, initialization vectors, and other random number requirements. It is FIPS 140-1 compliant, which facilitates system certification to this security standard. It also includes built-in-self-test (BIST) logic that allows for testing the randomness of the module output and its compliance with the FIPS 140-1 standard.

An ANSI X9.17, annex C post-processor is available to meet the NIST requirements of FIPS 140-1.

The RNG module has a hardware-based nondeterministic RNG core and a wrapper that provides a bus interface, clock, reset, and test features.

19.7.3 SHA-1/MD5 Module

The SHA-1/ MD5 (secure hash algorithm version 1/message digest algorithm version) security module provides hardware-accelerated hash functions. It can run either the SHA-1 algorithm in compliance with the National Institute of Standards and Technology, *Federal Information Processing Standards* (FIPS) 180-1, Secure Hash Standard, or the MD5 message-digest algorithm

developed by Rivest in 1991. Up to 2^{27} bytes (128M bytes) of data can be hashed in a single operation to produce a 160-bit signature in the case of SHA-1, and a 128-bit signature in the case of MD5.

This section discusses the algorithm basics, architecture overview, operational mode description, register mapping and content, and programming flowchart.

19.7.3.1 Algorithm Basic

Algorithms produce a condensed representation of a message or a data file. This condensed representation, also called a digest or signature, can be used to verify the message integrity, provided the verifier and the creator used the same digital signature algorithm (DSA). The message itself, however, cannot be reconstructed from the signature, and it is not possible to find two different messages that produce the same signature. Algorithms are thus called secure.

SHA-1/MD5 Algorithm

The SHA-1/MD5 algorithm can take an input message of up to 2^{64} bits. It sequentially processes blocks of 512 bits to compute a 160-bit signature. Thus, a message-padding step is required to make the message a multiple of 512 bits (64 bytes). This message padding is done in the hardware.

19.8 Debug Support Versus Security

The OMAP730 device can be configured via eFuse as an emulator device or as a normal device.

The debug is supported by control bits located in the secure control register (SECCTRL) in the OMAP730 configuration module (see Table 19–3).

Table 19–3. Control Bits for Debug in Secure Control Register (SECCTRL)

Bit	Name	Function	R/W	Reset Value
7	NORMAL_EMU_MODE_R	This bit has information about security type: 0: Security type is normal public debug. 1: Security type is normal secure debug. This bit is one-time programmable for normal device and read-only for emulation device. Its reset value is 0 for both normal and emulation device type.	R/ W1	0x0
6	CONF_JTAG_EN_R	MPU JTAG enable control register. 1: MPU JTAG is enabled (the functionality of the MPU JTAG is not altered). 0: MPU JTAG is disabled. Reset: 0: Reset condition for a normal device 1: Reset condition for an emulation device This bit can be configured one-time-only for a normal device. It cannot be configured at all for an emulation device.	R/ W1	0X0 or 0x1
5	CONF_ETM_EN_R	ETM enable control register. 1: Trace is not affected. 0: Trace is disabled. Reset: 0: Reset condition for a normal device 1: Reset condition for an emulation device This bit can be configured one-time-only for a normal device. It cannot be configured at all for an emulation device.	R/ W1	0X0 or 0X1
2	CONF_ICE_EN_R	MPU emulation enable control register. 1: MPU debug is not affected. 0: MPU debug is disabled. Reset: 0: Reset condition for a normal device 1: Reset condition for an emulation device This bit can be configured one-time-only for a normal device. It cannot be configured at all for an emulation device.	R/ W1	0X0 or 0X1

19.8.1 Production eFuse

The production eFuse holds four bits, EMULATOR[1:0] and NORMAL[1:0], that the security scheme uses. A decoding logic generates the signals SECURITY_DEVICE_TYPE[0:1] out of these four fuses.

The JTAG can read the EMULATORx and NORMALx eFuse values. In addition, the value of SECURITY_DEVICE_TYPE[1:0] can be learned from the OMAP730 configuration register *(CONF_STATUS)*.

Depending on the value of the EMULATORx and NORMALx bit, a device is one of three types:

- Normal
- Emulator
- Bad

A bad device keeps OMAP3.2 in a global-reset state. Three output pins on the external STI interface are driven to 101 to reflect this situation.

Normal and emulator devices have different properties, such as the reset value of the security control register (with the behavior difference it implies) and the ability to boot from external flash.

In addition to the 4-bit EMULATORx/NORMALx, the production eFuse also holds10 eFuses programmed to the same known value (PATTERN) and used in the eFuse initialization scheme. When the ULPD generates the INITZ pulse, it triggers a comparison between the eFuse value and the hard-coded PATTERN value. If the two match, the INITZ sequence has been correctly performed. In the case of a mismatch, the SECURITY_DEVICE_TYPE is forced to bad value.

At power-up reset, SECURITY_DEVICE_TYPE is set to bad until the INITZ sequence is performed and EMULATORx/NORMALx forces the correct value.

This scheme is necessary because the ULPD relies on the external input clock to generate the INITZ pulse, and this clock frequency cannot be trusted from a security point of view.

19.8.2 STI

On normal devices, the secure mode affects the behavior of the STI. In secure mode, the window tracer in STI for peripherals access trace is disabled.

The STI and window-tracer operations are always allowed on emulator devices, independently of the secure mode.

Note:

In secure mode/normal and emulation:

- The STI transmit path is still active. The TX FIFO content (captured before security signal assertion) is transmitted.
- The STI receive path is still active.
- The DSP/MCU write command is still available.
- The STI FIFO for memory dump is still available.

OMAP730 switching to secure mode on normal device disrupts any ongoing DSP/DMA STI operation that uses the window tracers.

19.8.3 Emulator Device

An emulator device provides full debug support for public and secure code execution. Any security violation detected by the security state machine attached to the ARM926EJS generates a debug request instead of a global reset.

This allows the programmer to dump the system state and investigate the security violation root cause.

The MPU is allowed to boot from external memory for emulator devices.

19.8.4 Normal Device

The debug support is disabled by default at power-up. The debug capabilities are blocked at two levels:

- Debug disabled at ARM926EJS core boundary (controlled by CONF_ICE_EN)
- ARM926EJS JTAG state machine forced into reset (controlled by CONF_JTAG_EN)

The MCU and DSP JTAG domains are daisy-chained. Blocking the ARM926EJS JTAG port also prevents an external host from taking control of the DSP code execution through the JTAG interface.

The application code can allow debug support for a normal device. This requires a secure transaction that sets the appropriate bits in the OMAP730 security control register. The debug support can be enabled for:

- Public code only
- Public and secure code

These security control register bits are one-time programmable. This means that once debug support is active, a power-on reset is required to disable it again.

19.8.4.1 Enabling Public Code Debug Only

The application code must call a secure procedure that programs the security control register with:

- CONF_JTAG_EN = 1
- CONF_ICE_EN = 1

Once these bits are set a host to target emulation connection is possible and debug support for public code is provided.

In this configuration the security state machine still monitors the ICE and JTAG control bits. If the application code to be debugged includes a secure-entry sequence, the security state machine prevents the application from entering secure mode.

19.9 Security Registers

19.9.1 DES/3DES Registers

Table 19–4. DES/3DES Registers (FFFE:4000)

Register	Offset	Register	Offset
DES_KEY3_L	0x00	DES_KEY3_H	0x04
DES_KEY2_L	0x08	DES_KEY2_H	0x0C
DES_KEY1_L	0x10	DES_KEY1_H	0x14
DES_IV_L	0x18	DES_IV_H	0x1c
DES_CTRL	0x20	DES_DATA_L	0x24
DES_DATA_H	0x28	DES_REV	0x2C
DES_MASK	0x30	DES_SYSSTATUS	0x34

Table 19–5. DES_KEY3_L

Bit	Name	Function	R/W	Reset Value
31:0	DES_KEY3_L	Key	R/W	0x0

Table 19–6. DES_KEY3_H

Bit	Name	Function	R/W	Reset Value
31:0	DES_KEY3_H	Key 3 MSB	R/W	0x0

Table 19–7. DES_KEY2_L

Bit	Name	Function	R/W	Reset Value
31:0	DES_KEY2_L	Key 2 LSB	R/W	0x0

Table 19–8. DES_KEY2_H

Bit	Name	Function	R/W	Reset Value
31:0	DES_KEY2_H	Key 2 MSB	R/W	0x0

Table 19–9. DES_KEY1_L

Bit	Name	Function	R/W	Reset Value
31:0	DES_KEY1_L	Key 1 LSB	R/W	0x0

Table 19–10. DES_KEY1_H

Bit	Name	Function	R/W	Reset Value
31:0	DES_KEY1_H	Key 1 MSB	R/W	0x0

Table 19–11. DES_IV_L

Bit	Name	Function	R/W	Reset Value
31:0	DES_IV_L	Initialization vector LSB	R/W	0x0

Table 19–12. DES_IV_H

Bit	Name	Function	R/W	Reset Value
31:0	DES_IV_H	Initialization vector MSB	R/W	0x0

Table 19–13. DES_CTRL

Bit	Name	Function	R/W	Reset Value
31:5	RESERVED	Reserved	R/W	0x0
4	CBC	When 1, cypher block chaining is used. When 0, electronic codebook is used.	R/W	0x0
3	TDES	When 1, triple DES is selected.	R/W	0x0
2	DIRECTION	When 1, an encrypt operation is performed. When 0, a decrypt operation is performed.	R/W	0x0
1	INPUT_READY	When 1, the 8-byte input buffer is empty and the next data can be written.	R	0x1
0	OUTPUT_READY	When 1, the 8-byte output buffer can be read. This bit is read-only	R	0x0

Table 19–14. DES_DATA_L

Bit	Name	Function	R/W	Reset Value
31:0	DES_DATA_L	Register to hold data (LSB) for encryption/decryption	R/W	0x0

Table 19–15. DES_DATA_H

Bit	Name	Function	R/W	Reset Value
31:0	DES_DATA_H	Register to hold data (MSB) for encryption/decryption	R/W	0x0

Table 19–16. DES_REV

Bit	Name	Function	R/W	Reset Value
31:8	RESERVED	Reserved	R/W	0x0
7:0	REV_NB	Revision number	R	none

Table 19–17. DES_MASK

Bit	Name	Function	R/W	Reset Value
31:6	RESERVED	Reserved	R/W	0x0
5	START	When 1, input DMA request is sent.	W	0x0
4	DIRECT_BUS_EN	When 1, Key 1 is loaded through the direct bus interface.	R/W	0x0
3	DMA_REQ_OUT_EN	When 1, input DMA is not masked.	R/W	0x0
2	DMA_REQ_IN_EN	When 1, output DMA is not masked.	R/W	0x0
1	SOFTRESET	When 1, a soft-reset sequence starts.	R/W	0x0
0	AUTOIDLE	When 0, clock is free-running. When 1, module is in power-saving mode.	R/W	0x0

Table 19–18. DES_SYSSTATUS

Bit	Name	Function	R/W	Reset Value
31:1	RESERVED	Reserved	R	0x0
0	RESETDONE	Internal reset monitoring	R	Unknown

19.9.2 RNG Registers

Table 19–19. RNG Registers (FFFE:5000)

Register	Offset	Register	Offset
RNG_OUT	0x00	RNG_STAT	0x04
RNG_CTRL	0x08	RNG_ENTA	0x0C
RNG_ENTB	0x10	RNG_X0	0x14
RNG_X1	0x18	RNG_X2	0x1C
RNG_COUNT	0x20	RNG_ALARM	0x24
RNG_CONFIG	0x28	RNG_LFSR1_0	0x2C
RNG_LFSR1_1	0x30	RNG_LFSR2_0	0x34
RNG_LFSR2_1	0x38	RNG_REV	0x3C
RNG_MASK	0x40	RNG_SYSSTATUS	0x44

Table 19–20. RNG_OUT

Bit	Name	Function	R/W	Reset Value
31:0	RNG_OUT	Output random number	R	0x00

Table 19–21. RNG_STAT

Bit	Name	Function	R/W	Reset Value
31:1	RESERVED	Reserved	R	0x0
0	BUSY	When 1, module is busy generating a new random number.	R	0x1

Table 19–22. RNG_CTRL

Bit	Name	Function	R/W	Reset Value
31:11	RESERVED	Reserved	R	0x0
10	RESET_LFSRS	If set to 0, LFSRs are reset	R/W	0x1
9	TEST_LFSR	If set to 1, system clock replaces ring oscillators.	R/W	0x0
8	TEST_ALARM	If set to 1, alarm counter increments on each clock cycle.	R/W	0x0
7	SHORT_CYCLE	If set to 1, rapid sate changes are used on the FSM test.	R/W	0x0
6	TEST_COUNTER	If set to 1, the RNG FSM state timer increments on each system clock.	R/W	0x0
5	DISABLE_ALARM	If set to 1, the LFSRs does not reset when the alarm is triggered.	R/W	0x0
4	TEST_RING2	If set to 1, the entropy ring oscillator #2 self-test is enabled.	R/W	0x0
3	TEST_RING1	If set to 1, the entropy ring oscillator #1 self-test is enabled.	R/W	0x0
2	TEST_RUN	If set to 1, all cycles of the RNG FSM are tested.	R/W	0x0
1	TEST_MODE	If set to 1, the RNG finite state machine (FSM) is disabled and write access to entropy and counter registers is enabled.	R/W	0x0
0	TEST_RING_OUTPUT	If set to 1, the RNG finite state machine (FSM) is disabled and write access to entropy and counter registers is enabled.	R/W	0x0

Table 19–23. RNG_ENTA

Bit	Name	Function	R/W	Reset Value
31:16	RESERVED	Reserved	R	0x0
15:0	RNG_ENTA	Value of entropy	R/W	0x0000

Table 19–24. RNG_ENTB

Bit	Name	Function	R/W	Reset Value
31:16	RESERVED	Reserved	R	0x0
15:0	RNG_ENTB	Entropy value	R/W	0x0

Table 19–25. RNG_X0

Bit	Name	Function	R/W	Reset Value
31:0	X	Seed	R/W	0x7841707C

Table 19–26. RNG_X1

Bit	Name	Function	R/W	Reset Value
31:0	X	Seed	R/W	0x75CD6092

Table 19–27. RNG_X2

Bit	Name	Function	R/W	Reset Value
31:17	RESERVED	Reserved	R	0x0
16:0	X	Seed	R/W	0x1CFFA

Table 19–28. RNG_COUNT

Bit	Name	Function	R/W	Reset Value
31:24	RESERVED	Reserved	R	0x0
23:0	FSM_COUNT	Counter	R/W	0x0

Table 19–29. RNG_ALARM

Bit	Name	Function	R/W	Reset Value
31:8	RESERVED	Reserved	R	0x0
7:0	ALARM_COUNTER	Counter for alarm	R	0x0

Table 19–30. RNG_CONFIG

Bit	Name	Function	R/W	Reset Value
31:12	RESERVED	Reserved	R	0x0
11:6	RESET_COUNT	Threshold for unchanged consecutive output bits	R/W	0x20
5:3	RING2_DELAY	Delay for RING2 oscillator	R/W	0x3
2:0	RING1_DELAY	Delay for RING1 oscillator	R/W	0x4

Table 19–31. RNG_LFSR1_0

Bit	Name	Function	R/W	Reset Value
31:0	LFSR1	LFSR1 value	R	0x00000000

Table 19–32. RNG_LFSR1_1

Bit	Name	Function	R/W	Reset Value
31:17	RESERVED	Reserved	R	0x0
16:0	LFSR1	LFSR1 value	R	0x0

Table 19–33. RNG_LFSR2_0

Bit	Name	Function	R/W	Reset Value
31:0	LFSR2	LFSR2 value	R	0x00000000

Table 19–34. RNG_LFSR2_1

Bit	Name	Function	R/W	Reset Value
31:16	RESERVED	Reserved	R	0x0
15:0	LFSR2	Value	R	0x0

Table 19–35. RNG_REV

Bit	Name	Function	R/W	Reset Value
31:8	RESERVED	Reserved	R	0x0
7:0	REV_NB	Revision number	R	0x20

Table 19–36. RNG_MASK

Bit	Name	Function	R/W	Reset Value
31:3	RESERVED	Reserved	R	0x0
2	IT_EN	Mask for interrupt (when 1, interrupt is not masked)	R/W	0x0
1	SOFTRESET	1: A soft-reset sequence starts.	R/W	0x0
0	AUTOIDLE	0: Clock is free-running. 1: Module is in power-saving mode.	R/W	0x0

Table 19–37. RNG_SYSSTATUS

Bit	Name	Function	R/W	Reset Value
31:1	RESERVED	Reserved	R	0x0
0	RESETDONE	Reset monitor	R	Unknown

19.9.3 Swatchdog Registers

Table 19–38. SWATCHDOG Registers (FFFE:A800)

Register	Offset	Register	Offset
WIDR	0x00	WD_SYSCONFIG	0x10
WD_SYSSTATUS	0x14	WCLR	0x24
WCRR	0x28	WLDR	0x2C
WTGR	0x30	WWPS	0x34
WSPR	0x48		

Table 19–39. WIDR

Bit	Name	Function	R/W	Reset Value
31:8	RESERVED	Reserved	R	0x000000
7:0	WD_REV	Module HW revision number	R	0x10

Table 19–40. WD_SYSCONFIG

Bit	Name	Function	R/W	Reset Value
31:6	RESERVED_0	Reserved	R/W	0x0
5	EMUFREE	Enable sensitivity to suspend (emulation) 0: The module freezes its internal logic upon suspend assertion. 1: The module ignores the suspend input.	R/W	0x0
4:2	RESERVED_1	Reserved	R/W	0x0
1	SOFTRESET	Software reset. Set this bit to 1 to trigger a module reset. This bit is automatically reset by hardware. Read returns 0. 0: Normal mode 1: The module is under reset.	R/W	0x0
0	AUTOIDLE	Internal OCP gating strategy: 0: OCP clock is free-running. 1: OCO clock-gating is active.	R/W	0x0

Table 19–41. WD_SYSSTATUS

Bit	Name	Function	R/W	Reset Value
31:8	RESERVED_0	Reserved for module-specific status information Reads return 0.	R	0x000000
7:1	RESERVED_1	Reserved for OCP socket status information Reads returns 0.	R	0x00
0	RESETDONE	Internal reset monitoring 0: Internal module reset is ongoing. 1: Reset completed	R	0x1

Table 19–42. WCLR

Bit	Name	Function	R/W	Reset Value
31:6	RESERVED_0	Reserved	R/W	0x00000
5	PRE	Prescaler stage enabled/disabled 0: Prescaler disabled 1: Prescaler enabled	R/W	0x1
4:2	PTV	Prescaler ratio values	R/W	0x7
1:0	RESERVED_1	Reserved	R/W	0x0

Table 19–43. WCRR

Bit	Name	Function	R/W	Reset Value
31:0	TIME_COUNTER	Value of timer counter	R/W	0x00000000

Table 19–44. WLDR

Bit	Name	Function	R/W	Reset Value
31:0	TIME_LOAD	Timer counter value loaded when overflow or when trigger context (WTGR)	R/W	0x00000000

Table 19–45. WTGR

Bit	Name	Function	R/W	Reset Value
31:0	TTGR_VALUE	Trigger value. Used to trigger a reload by writing value on WTGR different than the previous one.	R/W	0x00000000

Table 19–46. WWPS

Bit	Name	Function	R/W	Reset Value
31:5	RESERVED	Reserved	R	0x000000
4	W_PEND_WSPR	When equal to1, a write operation is pending to the WSPR register.	R	0x0
3	W_PEND_WTGR	When equal to 1, a write operation is pending to the WTGR register.	R	0x0
2	W_PEND_WLDR	When equal to 1, a write operation is pending to the WLDR register.	R	0x0
1	W_PEND_WCRR	When equal to 1, a write operation is pending to the WCRR register.	R	0x0
0	W_PEND_WCLR	When equal to 1, a write operation is pending to the WCLR register.	R	0x0

Table 19–47. WSPR

Bit	Name	Function	R/W	Reset Value
31:0	WSPR_VALUE	WSPR value used to switch off the watchdog with specific protocol	R/W	0x00000000

19.9.4 SHA-1/MD5 Registers

Table 19–48. SHA-1/MD5 Registers (FFFE:4800)

Register	Offset	Register	Offset
SHA_DIGEST_A	0x00	SHA_DIGEST_B	0x04
SHA_DIGEST_C	0x08	SHA_DIGEST_D	0x0C
SHA_DIGEST_E	0x10	SHA_DIGCNT	0x14
SHA_CTRL	0x18	SHA_DIN_0	0x1C
SHA_DIN_1	0x20	SHA_DIN_2	0x24
SHA_DIN_3	0x28	SHA_DIN_4	0x2C
SHA_DIN_5	0x30	SHA_DIN_6	0x34
SHA_DIN_7	0x38	SHA_DIN_8	0x3C
SHA_DIN_9	0x40	SHA_DIN_10	0x44
SHA_DIN_11	0x48	SHA_DIN_12	0x4C
SHA_DIN_13	0x50	SHA_DIN_14	0x54
SHA_DIN_15	0x58	SHA_REV	0x5C
SHA_MASK	0x60	SHA_SYSSTATUS	0x64

Table 19–49. SHA_DIGEST_A

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIGEST_A	Initial digest	R/W	0x0

Table 19–50. SHA_DIGEST_B

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIGEST_B	Initial digest	R/W	0x0

Table 19–51. SHA_DIGEST_C

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIGEST_C	Initial digest	R/W	0x0

Table 19–52. SHA_DIGEST_D

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIGEST_D	Initial digest	R/W	0x0

Table 19–53. SHA_DIGEST_E

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIGEST_E	Initial digest	R/W	0x0

Table 19–54. SHA_DIGCNT

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIGCNT	Initial digest count	R/W	0x0

Table 19–55. SHA_CTRL

Bit	Name	Function	R/W	Reset Value
31:25	RESERVED	Reserved bits	None	0x0
24:5	LENGTH	Code the length of message to hash	R/W	0x0
4	CLOSE_HASH	When 1, the final hash is performed.	R/W	0x0
3	ALGO_CONSTANT	When 1, the default values are used.	R/W	0x0
2	ALGO	When 1, hash is a SHA-1. When 0, hash is an MD5.	R/W	0x0
1	INPUT_READY	If 1, the SHA_DIN_X registers are available to input new 64-byte data.	R/W	0x1
0	OUTPUT_READY	If 1, hash operation is complete. To clear this bit, write 1. This serves as a pending interrupt.	R/W1C	0x0

Table 19–56. SHA_DIN_0

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIN_0	Holds message to be hashed	R/W	0x0

Table 19–57. SHA_DIN_1

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIN_1	Holds message to be hashed	R/W	0x0

Table 19–58. SHA_DIN_2

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIN_2	Holds message to be hashed	R/W	0x0

Table 19–59. SHA_DIN_3

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIN_3	Holds message to be hashed	R/W	0x0

Table 19–60. SHA_DIN_4

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIN_4	Holds message to be hashed	R/W	0x0

Table 19–61. SHA_DIN_5

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIN_5	Holds message to be hashed	R/W	0x0

Table 19–62. SHA_DIN_6

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIN_6	Holds message to be hashed	R/W	0x0

Table 19–63. SHA_DIN_7

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIN_7	Holds message to be hashed	R/W	0x0

Table 19–64. SHA_DIN_8

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIN_8	Holds message to be hashed	R/W	0x0

Table 19–65. SHA_DIN_9

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIN_9	Holds message to be hashed	R/W	0x0

Table 19–66. SHA_DIN_10

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIN_10	Holds message to be hashed	R/W	0x0

Table 19–67. SHA_DIN_11

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIN_11	Holds message to be hashed	R/W	0x0

Table 19–68. SHA_DIN_12

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIN_12	Holds message to be hashed	R/W	0x0

Table 19–69. SHA_DIN_13

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIN_13	Holds message to be hashed.	R/W	0x0

Table 19–70. SHA_DIN_14

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIN_14	Holds message to be hashed	R/W	0x0

Table 19–71. SHA_DIN_15

Bit	Name	Function	R/W	Reset Value
31:0	SHA_DIN_15	Holds message to be hashed	R/W	0x0

Table 19–72. SHA_REV

Bit	Name	Function	R/W	Reset Value
31:8	RESERVED	Reserved	R	0x00
7:0	REV_NB	Module revision number	R	0x04

Table 19–73. SHA_MASK

Bit	Name	Function	R/W	Reset Value
31:4	RESERVED	Reserved	R/W	0x0
3	DMA_EN	Enables DMA when 1	R/W	0x0
2	IT_EN	Enables interrupt when 1	R/W	0x0
1	SOFTRESET	When a 1 is written to this bit, it initiates a soft-reset sequence.	W/C	0x0
0	AUTOIDLE	0: Clocks are on. 1: Module is in power-saving mode.	R/W	0x0

Table 19–74. SHA_SYSSTATUS

Bit	Name	Function	R/W	Reset Value
31:1	RESERVED	Reserved	R/W	0x0
0	RESETDONE	Status of reset done	R	0x0

Chapter 20

Smart Card Controller

This chapter discusses the OMAP730 smart card controller (SMC).

Topic **Page**

20.1 Smart Card Controller (SMC)

The new features offered by the third generation mobile communication systems lead to a system migration of the existing single-application processor subscriber identity module (SIM) to a multiapplication platform for the future universal subscriber identity module (USIM).

In order to support numerous applications for value-added services (VAS), the smart card integrates a true card operating system (COS) with increased memory capacity and computing capability. This card copes with the requirements of any remote application management in terms of dynamic allocation and adaptation of data. Moreover, the smart card implements security features (crypto-processors, key generation, access control) as an element of the security frameworks for mobile commerce.

This smart card is a migration from the worldwide-acknowledged GSM SIM CARD.

This document is specifying the functional features to be implemented in the physical interface of the mobile equipment (ME) to the IC card. The protocol features are described only if supported partially or globally by hardware.

The smart card interface module implements the hardware interfaces to the smart card and a sequencer that manages the transmission protocol T = 1, in addition to T = 0, defined in ISO7816.3, thus freeing the M-controller of the real time constraints.

This module handles:

- The activation sequence (including hardware interpretation of coding convention and timing management for ATR sequence)
- T = 0 transmit protocol with parity error and timers management
- T = 0 receive protocol with parity error and timers management
- T = 1 transmit protocol with error (parity/EDC) and timers management
- T = 1 receive protocol with error (parity/EDC) and timers management
- The warm reset sequence
- The clock stop sequence
- The deactivation sequence

It interfaces with:

- The smart card through the ISO smart card interface
- The M-controller through a TIPB bus compliant parallel data interface.

The operation of the smart card interface is under control of the M-controller. Figure 20–1 and Figure 20–2 show the smart card interconnection.

20.1.1 Characteristics of Smart Cards

The smart card interface module is designed to support SIM (GSM), USIM (3GPP) and UICC (EMV) cards. Figure 20–1 and Figure 20–2 show different ways to connect smart cards to the TI interface.

Figure 20–1. Dual-Card Connection

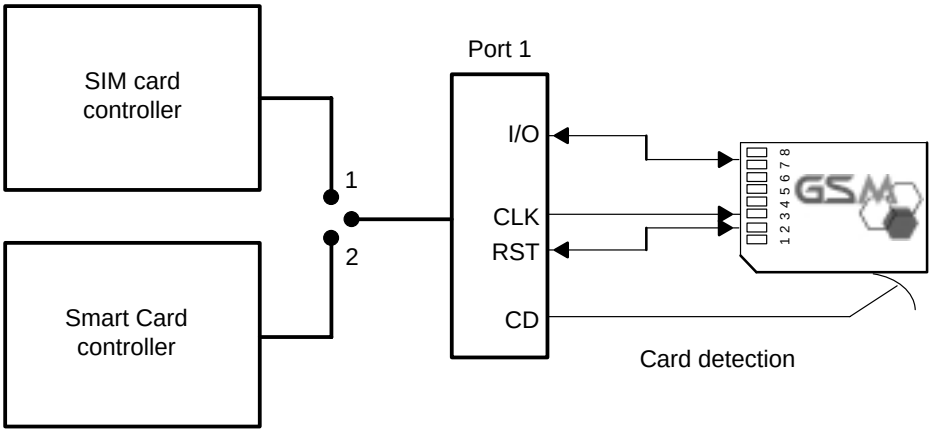
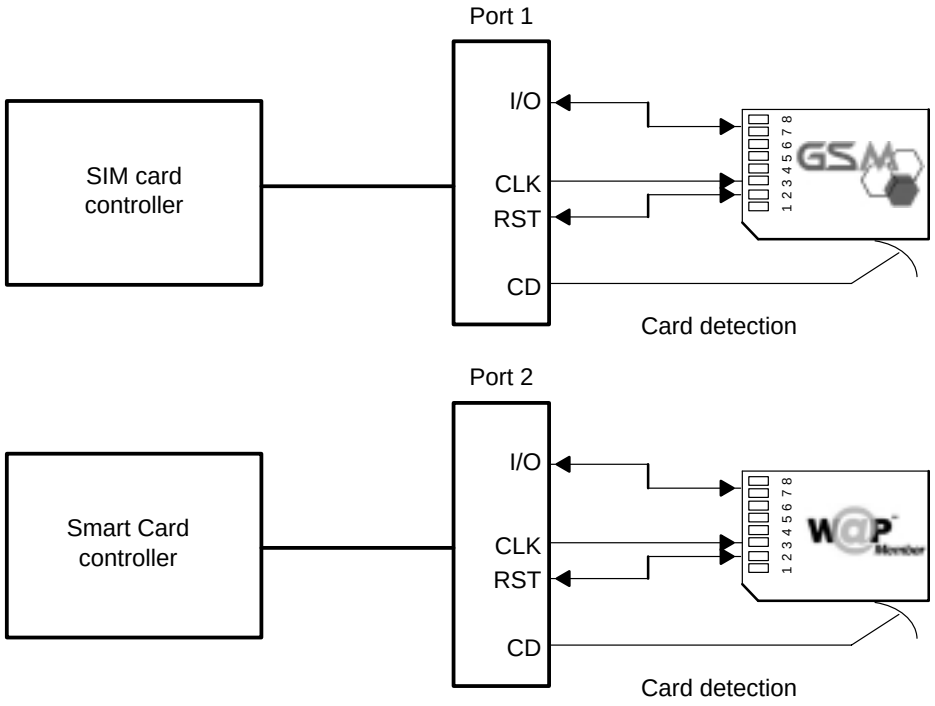


Figure 20–2. Single-Card Connection



The main smart card features are:

- Power supply V_{CC} equal to either:
  - 5 V±10%
  - 3 V±10%
  - 1.8 V±10%
- I_{CC} current supply:
  - Max 10 mA at V_{CC} = 5 V (Classes A and B)
  - Max 6 mA at V_{CC} = 3 V (Classes B and C)
  - Max 4 mA at V_{CC} = 1.8 V (Class C)
    (features relaxed by comparison to ISO: max 60 mA at 5 V in Class A)

- ❑ Programming voltage V_{PP} connected to V_{CC} (power supply)
- ❑ External clock only
- ❑ Smart card clock frequency in the 1 MHz−5 MHz range for Class A and 1 MHz−4 MHz range for Class B/C
- ❑ Clock duty-cycle of 45%−55%
  (feature strengthened by comparison to ISO: 40%−60%)
- ❑ Extra guard-time between characters equal to 0 (for GSM card only)
- ❑ Transmit protocol T = 0 and T = 1 must be supported.

20.1.2 Smart Card Interface Features

20.1.2.1 Features for Compliance With ISO, ETSI/GSM, 3GPP

General

- ❑ Vcc voltage supply:
  - ■ Class C (Vcc = 1.8 V and Vcc = 3 V) mandatory
- ❑ External clock frequency equal to 13/4 MHz or 13/8 MHz during ATR sequence.
- ❑ Mandatory transmission factor values: Di/Fi= 1/372, 8/512, and 16/512.
- ❑ Start bit detection logic activated at 11 ETU (T = 0) or at 10 ETU (T = 1)
  (feature strengthened by comparison to ISO: min 12 ETU for T = 0)
- ❑ Automated or manual card activation and deactivation sequencing management
- ❑ Over-sampling frequency in reception equal to $F_{ETU} \times 8$
- ❑ Card presence detection signal with 32-kHz-based filtering logic
- ❑ Warm reset generation mandatory (with timers)
- ❑ External clock stop mode
  - ■ External clock autodisabling at least 1860 clock cycles after last RX character
  - ■ TX character at least 744 clock cycles after reenabling external clock

ATR Procedure

- ❑ Hardware error handling of character parity check
- ❑ Hardware error handling of ATR time-out
- ❑ Hardware identification of character coding convention (direct/inverse)

T = Protocol

- ❑ Hardware error handling of WWT timer time-out
- ❑ Delayed release of I/O data line after successful completion of last byte transmission

T = 1 Protocol

- ❏ Transmit protocol (T) of type 1 (TD1 character equal to xH01) (asynchronous half-duplex block transmission protocol)
- ❏ Character waiting time (CWT) management (hardware timer based)
- ❏ Block waiting time (BWT) management (hardware timer based)
- ❏ Block guard time (BGT) management (hardware timer based)
- ❏ Must keep parity checking at character level without acknowledgement (i.e., no repeat)

20.1.2.2 Functional Features for Improved Performance

The increase of the card role in the third generation VAS leads to a corresponding increase in data flow between the card and the ME. Hardware management of the transmission protocol is the answer to the reduction of the consequent impact on the host CPU loading.

General

External clock frequency (F_{SCK}) equal to 13/N MHz (with N= 4 or 8)

ATR Procedure

- ❏ Hardware identification of character coding convention (direct/inverse) from TS character with update for processing of subsequent characters (T0, TAi, etc.)—See section 20.2.1.1, *Coding Convention*.
- ❏ Software processing of PPS/PTS procedure
- ❏ Software identification of procedure bytes of PTS0 and PTS1 with update of protocol type T and transmission factor value Fi/Di

T = 1 Protocol

- ❏ Hardware management of block length (interpretation of LEN character)
- ❏ Hardware checking of EDC error code if LRC is used
- ❏ Hardware error handling of:
  - ■ LEN block length error (in RX mode)
  - ■ Character parity check in complement to EDC error.
  - ■ CWT, BWT and BGT timers time-out (with IT generation)

Note:

If CRC is used, EDC error code checking should be done by software.

Handling of PCB erroneous encoding should be done by software.

20.1.3 Functional Description

20.1.3.1 Module Overview

This module realizes all the functions required for information exchange between the terminal and smart cards according to the corresponding protocol standard. A complete card session consists of the following steps:

- ❏ Insertion of the card and activation sequence
- ❏ Data transmit and receive sequence
- ❏ Deactivation of the contacts and extraction of the smart card

The operation of the smart card interface is under control of the M-controller.

In automatic mode, the card activation and deactivation of START and STOP commands from the MCU, respectively, are monitored by the smart card interface itself, as accurate timing must be maintained to fulfill the ISO/GSM/3GPP standards.

The M-controller leads:

- ❏ The start of a transmission sequence
- ❏ The writing and reading of characters in the USIM interface
- ❏ The contacts deactivation on card reject decision
- ❏ The SIM clock de/activation for power management in idle period
- ❏ The reset of the card (deactivation + activation) on consecutive error detections.

In manual mode, the MCU directly controls the interface signals (SIM reset, SIM clock, SIM power control, data-line direction, data-line level) during the card activation and deactivation sequences.

20.1.4 Functional Operating Modes

20.1.4.1 IDLE State

The IDLE state characterizes the state of the smart card interface when the card contacts the are deactivated.

This means:

- ❏ sim_clk = 0
- ❏ sim_rst = 0
- ❏ sim_pwrctrl = 0
- ❏ sim_io = 0 with the line in transmission configuration

20.1.4.2 Activation Sequence

In order to initiate an interaction with a mechanically connected card, the terminal should activate the electrical circuits first (in response to start command from MCU).

The activation of the card must be subsequent to:

- The module hardware reset to initialize the logic at power-up, or the software reset of smart card interface module
- The enabling of the internal clocks of the module
- The detection of the card insertion event after debouncing of the card detect signal (SIM_SINEX)

The activation sequence includes the following sequence of steps:

Step 1: Connection of the contacts

Step 2: Activation of the contacts (card power-up)

Step 3: Reset of the card (cold reset)

Step 4: Answer to reset (ATR) detection

The whole sequence can be executed automatically or manually (bypass mode) based on setup of appropriate configuration register (bit CONFBYPASS of the USIMCONF1 register):

- If the automatic execution mode is selected, the whole sequence is executed by the smart card interface as soon as the M-controller has initiated the start procedure (CMDSTART bit of the command register USIMCMD).
- If the bypass mode is selected (that is, manual mode), the MCU can directly control the activation of signals such as SIM_RST, SIM_PWCTRL, SIM_CLK, SIM_IO level, and SC_IO direction. It can do so via the setup of various bits in appropriate configuration register (SVCCLEV, SRSTLEV, SCLKLEV, CONFSIOLOW of the USIMCONF1 register, and TXNRX of the USIMCONF2 register).

20.1.4.3 Connection of the Contacts

The contacts of the smart card are connected as soon as the card is inserted in the driver. This operation is controlled by the external smart card driver, which sets the SC_CD signal on the insertion detection.

Smart card insertion => sim_sinex = 1

The SIM_SINEX signal state moving from low to high activates the card presence detection circuit in USIM module. When the filtering logic toggles, the status bit STATNOCARD of USIMSTAT register is set (card presence detection) or reset (no card).

20.1.4.4 Activation of the Contacts (Card Power-up)

After card insertion is detected, the contacts must be activated in the following sequence:

- Card reset is maintained in low state (SIM_RST=0) during contact activation.
- Card power-supply is switched on (signal SIM_PWRCTRL=1)

- ❏ Bidirectional data line is set in reception mode and driven to high-Z
- ❏ (SIM_IO)
- ❏ Card external clock is activated (signal SC_CLK toggling)

Each of these steps is executed with a scheduling based on a predefined and configurable time period T_{actv} (TDUSIM in USIM_CONF3 register).

20.1.4.5 Reset of the Card (Cold Reset)

After activation of the contacts, the terminal should initiate a cold reset sequence to receive expected answer-to-reset from the card. The card-reset process differs for different card types (SIM, USIM, etc.).

SIM (GSM) and USIM (3GPP) Cards:

Two types of GSM SIM cards exist at this time:

- ❏ Cards active on an internal reset (synchronous cards)
- ❏ Cards active on an external low-level reset (asynchronous cards)

Thus, a specific scheduling must be adopted.

At a notional time T0:

1) Reset maintain low (SIM_RST = 0) and smart card clock on (SIM_CLK = /\/\)

T0+400Tclk < *tc* < T0+40,000Tclk

2) If synchronous card, then normally ATR sequence begins (see Section 20.1.4.6, *ATR Sequence Reception*. SIM_RST must be maintained low.

If during *tc*, ATR begins,

3) SIM_RST must be maintained low, and the following 4), 5), 6) steps are not activated.

If terminal does not receive an ATR received, then:

AT time *T1* = T0+40,000Tclk

4) Switch reset high (SIM_RST=1)

T1+400Tclk < *tc* < T1+40,000Tclk

5) If asynchronous card, then normally ATR sequence begins. Reset must be maintained in high state.

If no ATR received during *tc*, then:

At time *T2* = T1+40,000Tclk:

6) Reset switched to low (SIM_RST=0)

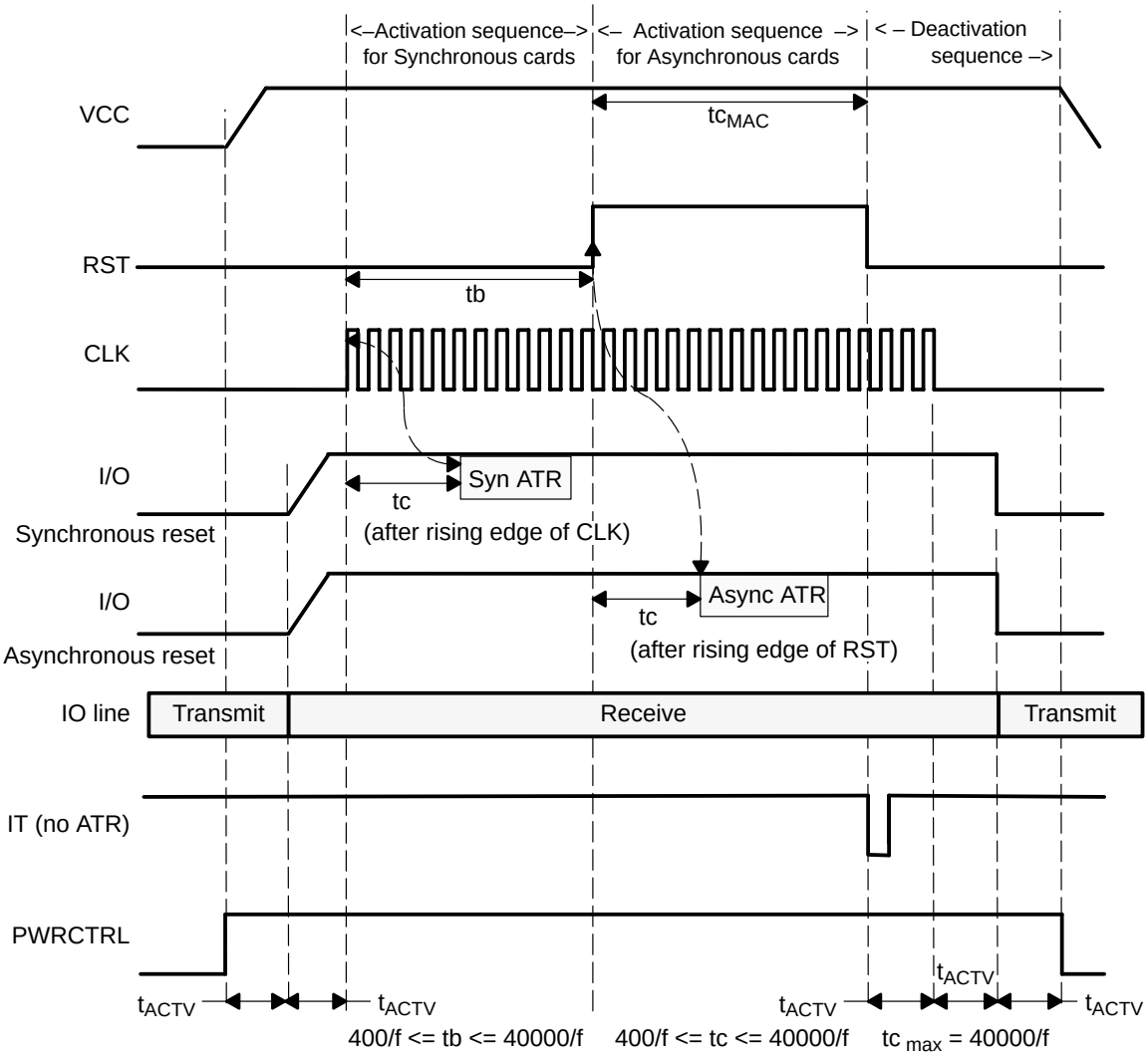
7) After that, deactivation of the contacts is initiated (see Section 20.1.7, *Deactivation Sequence*).

If no ATR is detected, the last phase (7) is automatically run by the USIM interface (after activation of the contacts phase).

The GSM standard requires that an activation sequence for asynchronous card always follow an activation sequence for synchronous card. However, since synchronous cards are not often used, the smart card interface allows bypassing of the activation sequence for synchronous cards. This can be done either by hardware implementation or software programming. This allows to save 40,000 clock cycles in case of asynchronous cards (mostly frequently used).

See Figure 20–3 concerning the timing diagram of the activation and cold reset sequences.

Figure 20–3. Contact Activation and Cold Reset Sequence



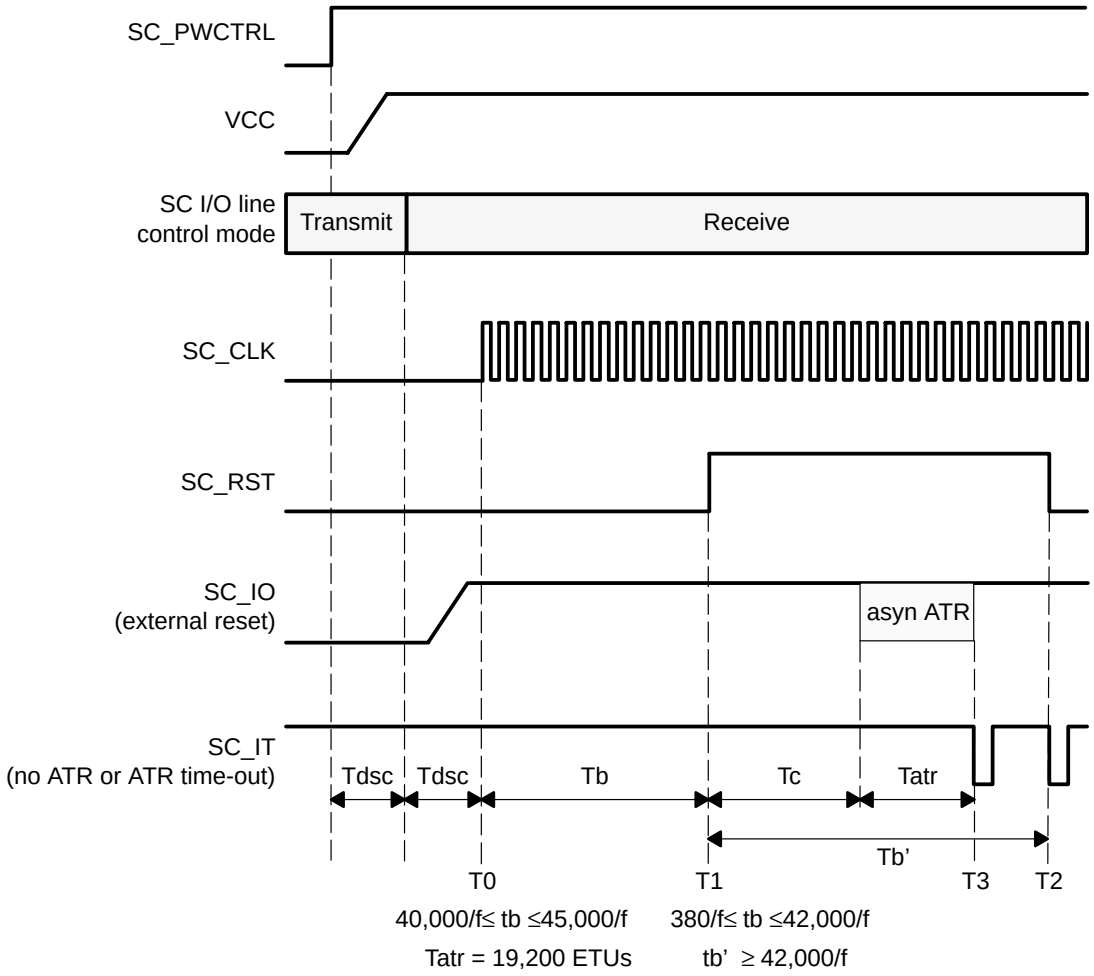
EMV Cards

The following are the main differences between EMV and GSM specifications:

- For EMV specification, all the ICC answer to reset asynchronously using active low reset.
- *tb* is in the *40,000/f – 45,000/f* range
- The terminal must be able to receive an ATR that begins within 380 to 42,000 clock cycles (*380/f ≤ tc ≤ 42,000/f*)
- Answer-to-reset must have a duration less than or equal to 19,200 ETUs. If time-out, terminal must initiate deactivation sequence within 24,000 ETUs following TS character.

The timing sequence for EMV card is shown in Figure 20–4.

Figure 20–4. Contact Activation and Cold Reset for EMV Cards



20.1.4.6 ATR Sequence Reception

The processing of the ATR sequence is similar to the processing of any receive sequence.

The hardware of the USIM module is designed to interpret the first character received (that is, TS) of ATR. After reception of TS, coding convention (direct or inverse) is ascertained and the corresponding configuration register is set accordingly (CONFCODCONV bit of USIMSTAT). And then, hardware of the USIM may do some settings according to the coding convention for automatic interpretation of subsequent characters.

ATR sequence reception includes:

- ❏ Initial configuration of the USIM with direct convention (default setup)
- ❏ Hardware interpretation of the protocol convention after reception of TS
- ❏ Set configuration register accordingly (direct or inverse convention). The direct convention is set by default.
- ❏ Hardware settings based on the coding convention
- ❏ Automatic reception of the other ATR characters following TS

20.1.4.7 Data Receive and Transmit Procedures

All data exchanges between the terminal and the smart card are initiated by the MCU.

Several transmission protocols are part of the standards:

ATR	(Answer-to-reset)	Receive procedure only
PPS	(Protocol and parameter selection)	Receive and transmit procedure
T	(Transmit Protocol—T = 0 or T = 1)	Receive and transmit procedure

Data Transmit Procedure

In transmit mode, the characters are sent from the mobile equipment card interface (transmitter) to the smart card (receiver). Different data transmit procedures apply for T = 0 and T = 1.

T = 0 Protocol

The transmit procedure is based on a parity test by the receiver with confirmation of the checking to the transmitter. The parity of the information byte is computed by the smart card interface and concatenated to the byte for the transmission.

The transmitter sends characters sequentially so as to meet minimum character guard time (CGT). A timer is reactivated at each new transmitted character to control this minimum delay.

The WWT timer is also used to flag the potential overflow of the waiting time between a transmitted character and the next character (sent by the card to ME card interface). The timer is reactivated at each new transmitted character, and an interrupt is generated in case of overflow (wait timeout). This is the maximum card waiting time to answer after switching from transmit to receive mode.

The consecutive steps of the transmit procedure are:

1) Set bidirectional data line I/O to transmit mode

2) Transmit the start bit active at low-level (0). All consecutive time delays are referenced to the falling edge of this start bit.

3) At 1 ETU, the information byte is serialized and transferred followed by the parity bit at 9 ETU.

4) Release the I/O line and behave as a receiver at 10.5 ETU to allow the receiver to force on the line the state value corresponding to the result of the checking of the parity bit:

   - State value = 0 => Parity false
   - State value = 1 => Parity good

The receiver maintains the minimum value of 1 ETU (whatever parity check result) or 2 ETU max (if wrong parity is detected) before releasing the line.

Note:

In the hardware implementation of the USIM module, the duration of the reverse of the IO line is fixed to two ETUs (regardless of the parity check result) so that the next character in TX does not happen before 13 ETUs (in compliance with ISO7816-3), and after the minimum guard time required (CGT value).

5) The transmitter checks on the fly the parity confirm bit at 11 ETUs.

Depending on the value of this parity confirm bit, the transmitter can repeat or not the transmission of the current character (this should be handled as a software decision). The next character can only be transmitted after guard time *t*.

The sequences for transmit mode without or with parity error, are shown in Figure 20–5 and Figure 20–6, respectively.

Figure 20–5. Transmit Mode Without Parity Error (T = 0)

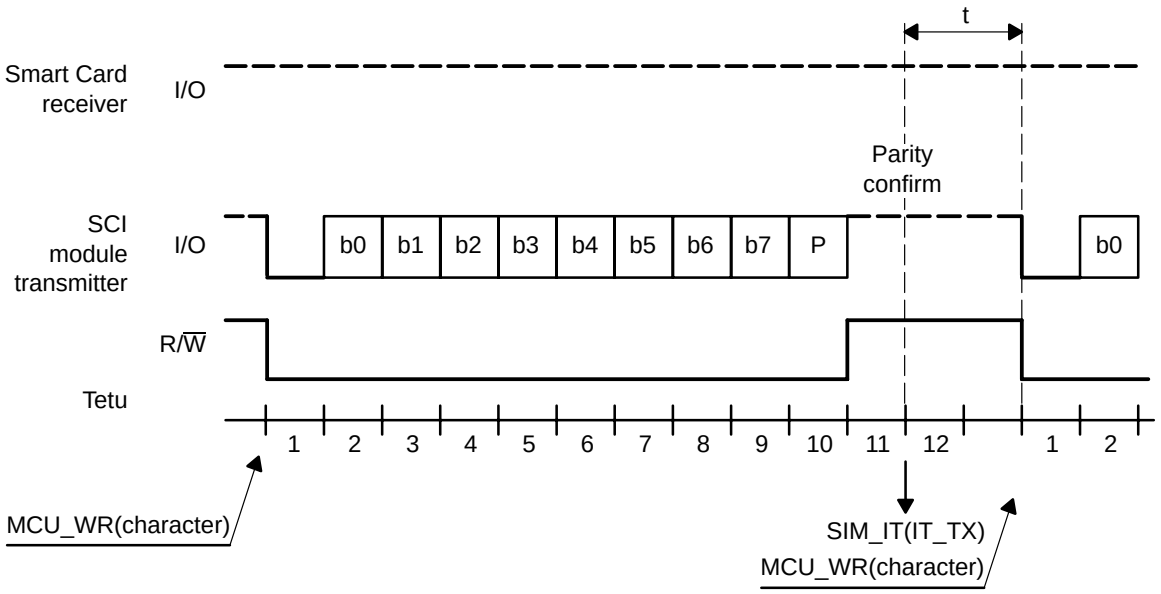
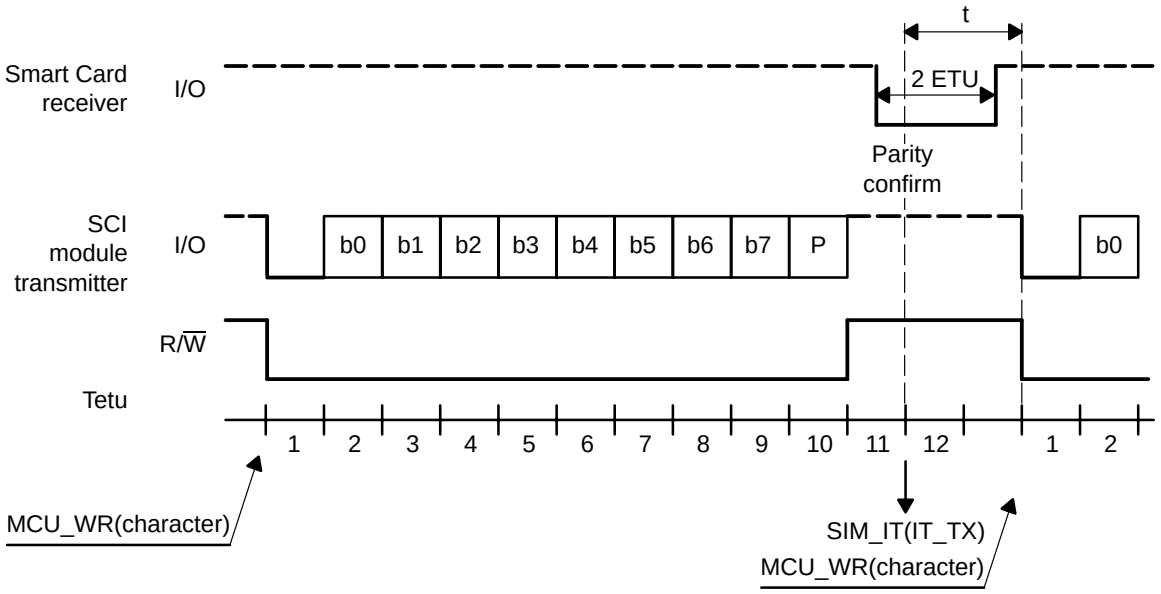


Figure 20–6. Transmit Mode With Parity Error (T = 0)



T = 1 Protocol

The transmit procedure of T = 1 protocol is more complex than that of T = 0 protocol. A block is the smallest data unit that can be transferred. The transmitter sends block-size characters sequentially so as to meet minimum character guard time (CGT) and maximum character waiting time (CWT); the CGT timer is reactivated at each new transmitted character to control this minimum guard time between transmitted characters. The receiver (smart card) stays in reception mode during the whole block transmission.

Notice that the BWT timer is also used to flag the potential overflow of the waiting time between the last transmitted character of a block and the first charac-

ter of next block (sent by the card to ME card interface). The timer is reactivated at each new transmitted character and an interrupt is generated in case of overflow (wait time-out). This is the maximum waiting time for answer of the card after switching from transmit to receive mode.

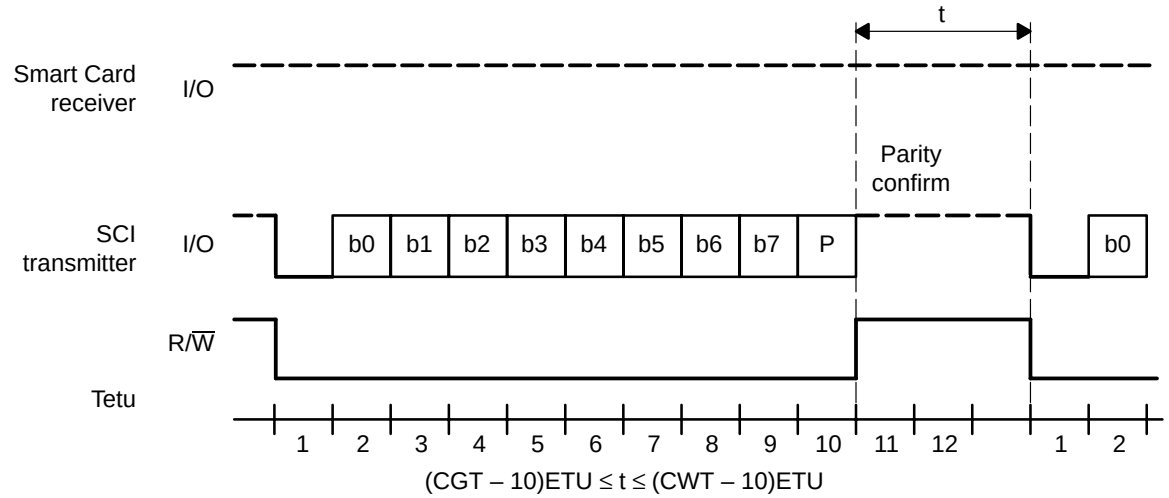
For T = 1 protocol, the parity error management as used in T = 0 protocol is not applicable. The error handling includes block error detection code (EDC) and the parity checking will be used as a complement to EDC. The EDC uses either CRC or LRC.

The consecutive steps of transmit procedure are:

- Set bidirectional data line I/O in transmit mode
- Transmit the start bit active at low level (0). All the consecutive time delays are referenced to the falling edge of this start bit.
- At 1 ETU, the first block is serialized and transferred, followed by the parity bit (at 9 ETU). After the transmission of a character, the receiver checks the parity but neither issues an error signal on the I/O line, nor asks for character repeat (even if a parity error occurs). The parity checking is used as a complement to EDC (block error detection code).
- Then
  - If one or more characters of the same block have not been transferred, after time t, the transmitter can transfer the next character of the block;

  Or
  - If the transfer of the whole block is finished, release the I/O line and give the right of sending acknowledgement to the other side (that is, the card).

Figure 20–7. Transmit Mode Without Parity Check



Data Receive Procedure

In receive mode, characters are sent from the smart card (transmitter) to the ME card interface (receiver). Also, a different process is applied according to the corresponding protocol T = 0 or T = 1.

T = 0 Protocol

As for the transmit procedure, the receive procedure is based on a parity test by the receiver and with a confirmation of the checking to the transmitter. The parity of the information byte is computed by the smart card interface from the received byte, and compared to the parity sent by card.

A timer (WWT) flags the potential overflow of the waiting time between two consecutive received characters. The timer is reactivated at each new character and an interrupt is generated in case of overflow (wait timeout).

The consecutive steps of receive procedure are:

- ❏ Setting of the serial line SIM_IO in reception mode
- ❏ Detection of the start-bit with over sampling clock at the rate of $8 \times F_{ETU}$ (*Fsam1)*. On the point where the start bit is confirmed, the midstart-bit sampling clock (*Fsam2*) is activated and set from low to high state (see Figure 20–20 for reference). All the time delays in the following text are referenced to the time origin, which is the falling edge of the start bit.
- ❏ On the start-bit occurrence, the smart card interface initializes its receiver submodule, and reinitializes the WWT timer.
- ❏ Using the midstart-bit sampling clock, the bits of the received character (including the parity bit) can be sampled at each ETU with a reference time based on the middle of the start bit.
- ❏ At 9 ETUs, the information byte is paralleled and its parity is computed. Parity checking can be either enabled or disabled by appropriate setting of one of the configuration registers.
- ❏ If parity check is active, the calculated parity is then compared to the transmitted parity sampled at time 9.5 ETUs.If parity check is inactive, the parity is assumed to be correct whatever its value.
- ❏ At 10.5 ETUs the interface forces on the data line, released by the smart card, the confirm bit whose value results of the parity comparison:
  - SIM_IO value = 0 => *Parity false*
  - SIM_IO value = 1 => *Parity true*

  The smart card interface maintains this value for 1 ETU only (regardless of good or wrong parity detected) before releasing the line at 11.5 ETUs. The smart card interface can detect the start bit of a new character either at 11 ETUs or at 12 ETUs (after the start bit of the previous character) depending on the parity checking (either OK or FALSE respectively). The transmitter (smart card) checks on the fly the parity confirm bit at 11 ETUs. Depending of the value of this parity confirm bit, the transmitter repeats or not the transmission of the current character.

The sequences for receive mode without or with parity error, are shown in Figure 20–8 and Figure 20–9, respectively.

Figure 20–8. Receive Mode Without Parity Error (T = 0)

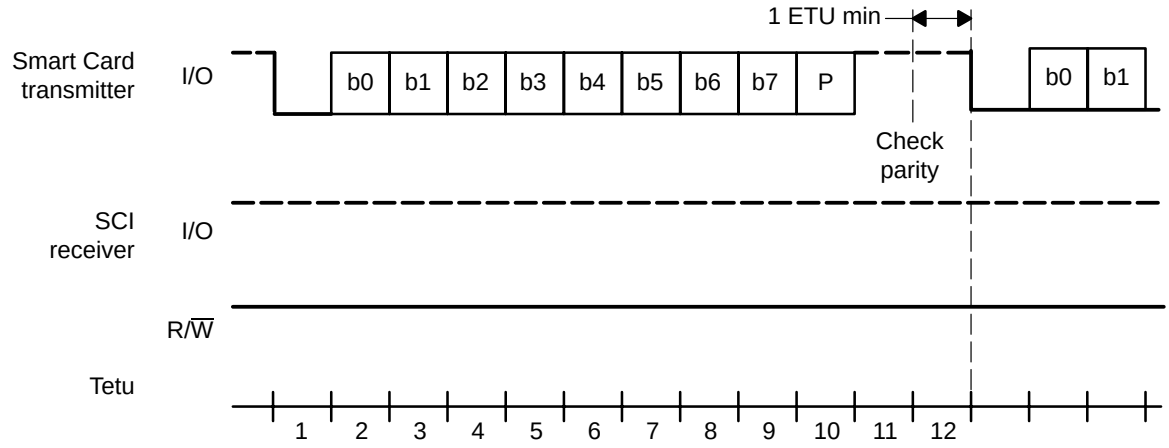
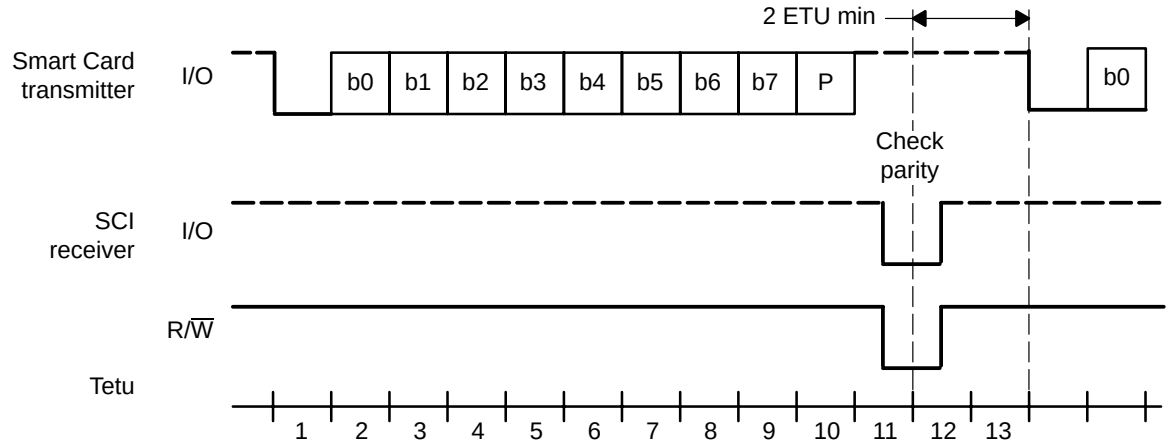


Figure 20–9. Receive Mode With Parity Error (T = 0)



T = 1 Protocol

For T = 1 protocol, the receiver receives the characters of a block sequentially, without giving any confirmation to the transmitter of the parity checking result.

As a receiver, the smart card interface processes blocks in order to flag potential overflow of the waiting time between two consecutive received characters (CWT), and counts the number of incoming characters. The CWT timer is reactivated each time a new character is received and an interrupt is generated if an overflow occurs (wait timeout). In T = 1 protocol, block error detection code (EDC) code uses either CRC or LRC.

A block is the smallest data unit that can be transferred.

The consecutive steps of the receive procedure are:

- ❏ Setting of the serial line SIM_IO to reception mode
- ❏ Detection of the start-bit with over sampling clock at the rate of $8 \times F_{ETU}$ (*Fsam1)*. On the point where the start bit is confirmed, the midstart-bit sampling clock (*Fsam2*) is activated and set from low to high state (see

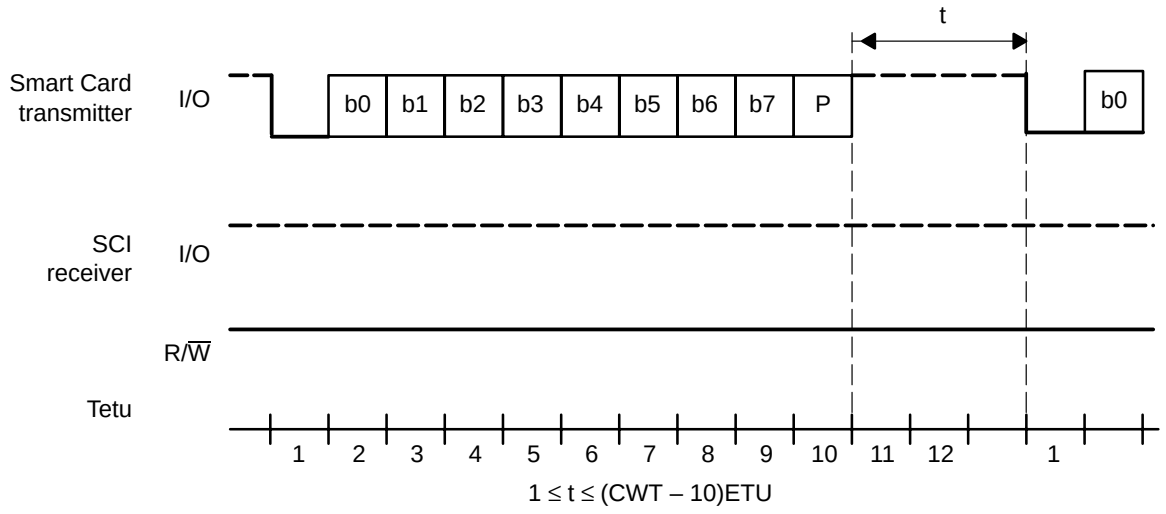
Figure 20–20 for reference). All the time delays in the following text are referenced to the time origin, which is the falling edge of the start bit.

- On the start-bit occurrence, the smart card interface initializes its receiver submodule and reinitializes the CWT timer.
- Using the midstart-bit sampling clock, the bits of the received character (including the parity bit) can be sampled at each ETU with a reference time based on the middle of the start bit.
- At 9 ETUs, the information byte is paralleled and its parity is computed. The counter used to log the number of incoming characters is incremented.
- After the reception of a character, the smart card interface checks the parity but neither issues an error signal on the I/O line, nor asks for character repeat (even if a parity error occurs). The parity checking is used as a complement to EDC (block error detection code).
- When the counter of incoming characters is equal to three, the received character is interpreted to extract the LEN value. When the total number of received characters (including prologue and epilogue) is equal to either [LEN+4] if EDC is based on LRC, or [LEN+5] if EDC is based on CRC, the end of block is reached. The CWT timer is deactivated and the counter of incoming characters is reset.

  So:

  - If the block has not been received completely, the smart card interface can detect the start bit of a new character after the guard time *t* and the receiver can process the next character of the block.
  - If the block reception is finished, the smart card interface checks the EDC and releases the I/O line and can send an acknowledgement to the other side (that is, the card).

Figure 20–10. Receive Mode Without Parity Check Echo



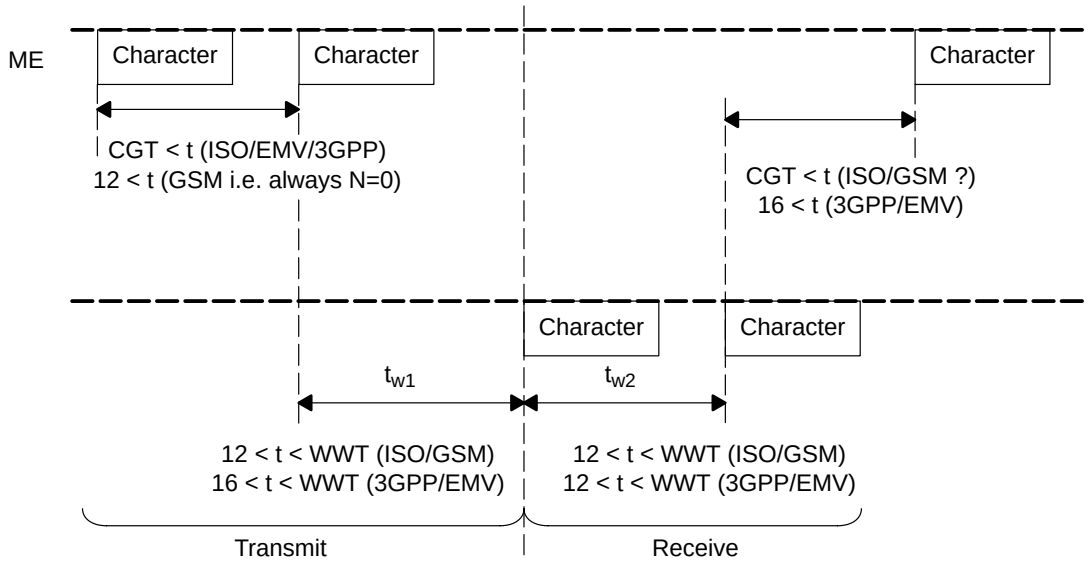
20.1.4.8 Switching Sequence

The switching from transmit mode to the receive mode is programmed with one bit in the appropriate configuration register (bit TXNRX in USIMCONF2 register), which controls the data line direction (1→TX/0→RX). To allow immediate switching of the data-line after the last character of the block has been transmitted, whatever the inertial time of the MCU to react to an interrupt, the MCU can program the data-line configuration in receive mode during the transmission of a character. The physical switching of the line is delayed until the end of the transmission.

The configuration register update can be done immediately after writing the last character of the block to transmit.

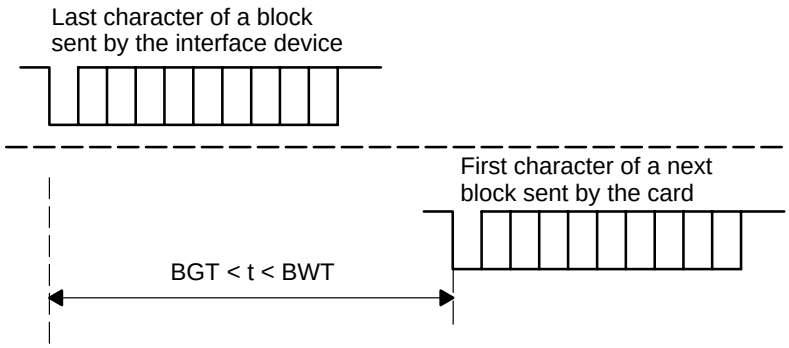
For t = 0, the WWT timer can be used to flag the potential time out (t_{w1}) between the last character transmitted by the ME card interface, and the next character sent by the card to the card interface ($t_{w1} \leq$ WWT). This is the maximum waiting time for answer from the card after the switching from transmit to receive mode.

Figure 20–11. Switching From Transmit to Receive Mode (t = 0)



For T=1, the BGT and BWT timers can be used to flag the potential time-out between the last character of a block sent by the terminal (ME) and the first character of the next block sent from the card. The time interval *t* is such that BGT < *t* < BWT.

Figure 20–12. Switching From Transmit to Receive Mode (t = 1)



20.1.5 Warm Reset Procedure

The controller can decide to reset the smart card when there are no ongoing data exchanges. For all kinds of cards supported (GSM, 3GPP), use the following sequence:

At notional time t0:

1) Card reset is asserted low (SIM_RST = 0)
   Maintain V_{CC} and CLK stable (signal SIM_CLK toggling)
   Keep I/O line in reception mode and driven high-Z (SIM_IO = Z)

At time t1:

2) Card reset is released high (SIM_RST=1)
   The interval between time t0 and t1 must be *te* (that is, *te* = T1−T0):
   For GSM and 3GPP: $400Tclk \leq te$

Figure 20–13. ISO 7816-3 Compliant Warm Reset Sequence

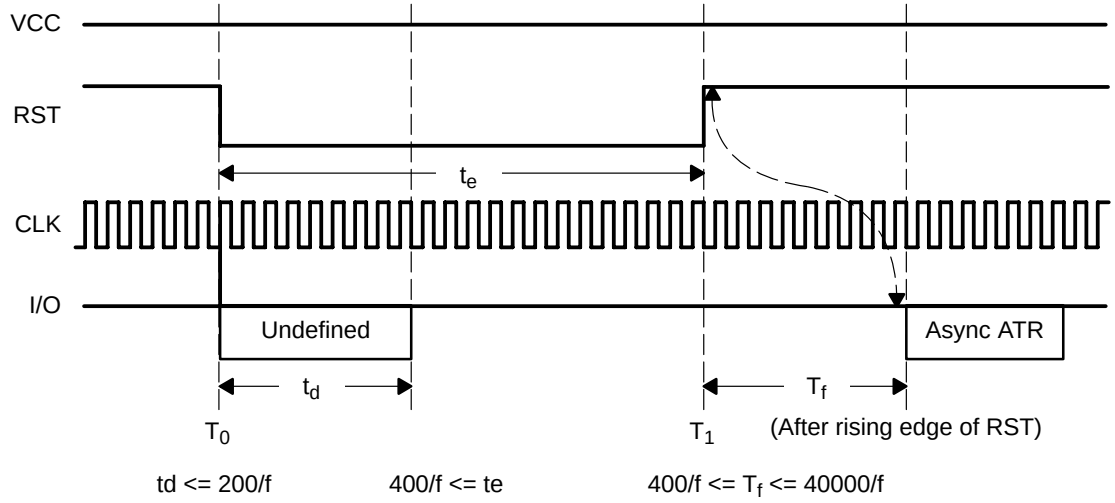
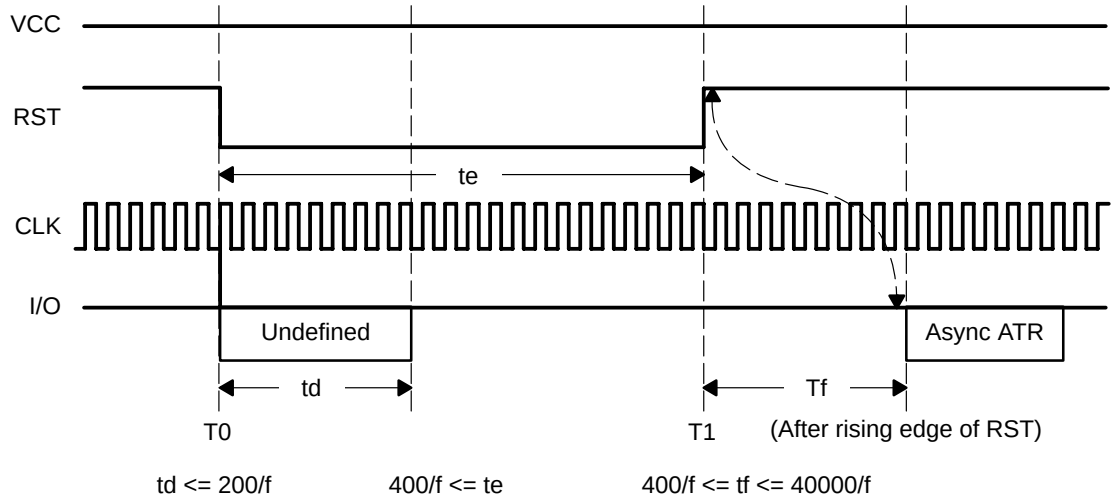


Figure 20–14. EMV Compliant Warm Reset Sequence



Considering the differences between card types, the default value of t_e is set according to GSM, EMV, and 3GPP cards with different configurations.

20.1.6 Sleep Mode (Clock Stop Mode)

In transmit mode, the MCU can decide to stop the smart clock to save power if, and only if, the card can support clock stop mode and there is no data transmission processing.

The clock stop sequence includes:

Init

Configure idle level of external card clock (SIM_CLK) in corresponding configuration register (SCLKLEV bit of USIMCONF1).

Time t0

No more transmission (SIM_IO=Z, SIM_CLK active), timer t_g = 1860Tclk started (the timer t_g is reactivated after each new character).

Time t1 ≥ ***t0+1860Tclk***

Smart card clock is stopped.

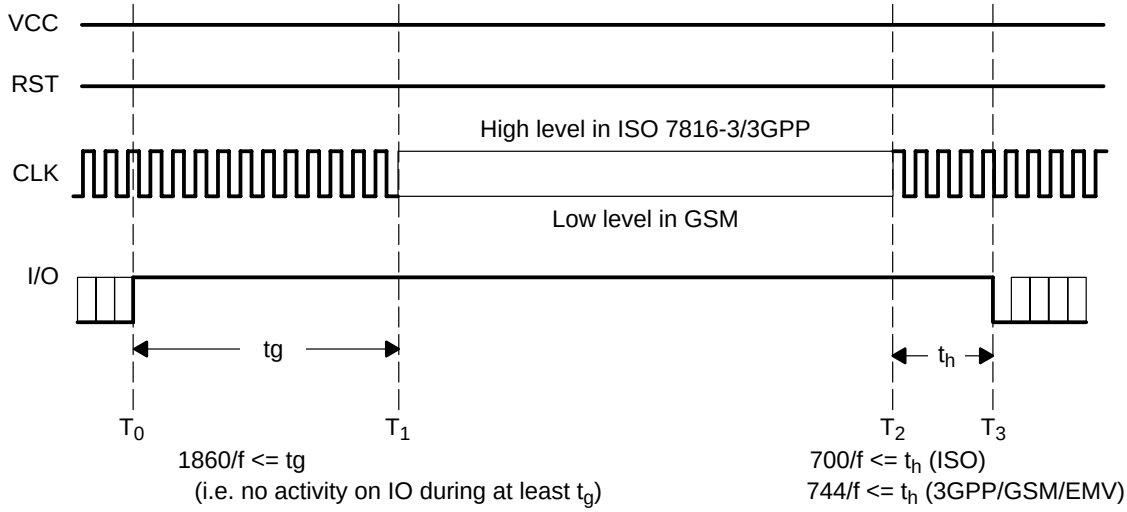
Time t2 ≥ ***t1***

On new data transmission request, restart smart card clock (SIM_CLK active) and start timer t_h = 744Tclk.

Time t3 ≥ ***t2+744Tclk***

Restart transmission.

Figure 20–15. Clock Stop Sequence



20.1.7 Deactivation Sequence

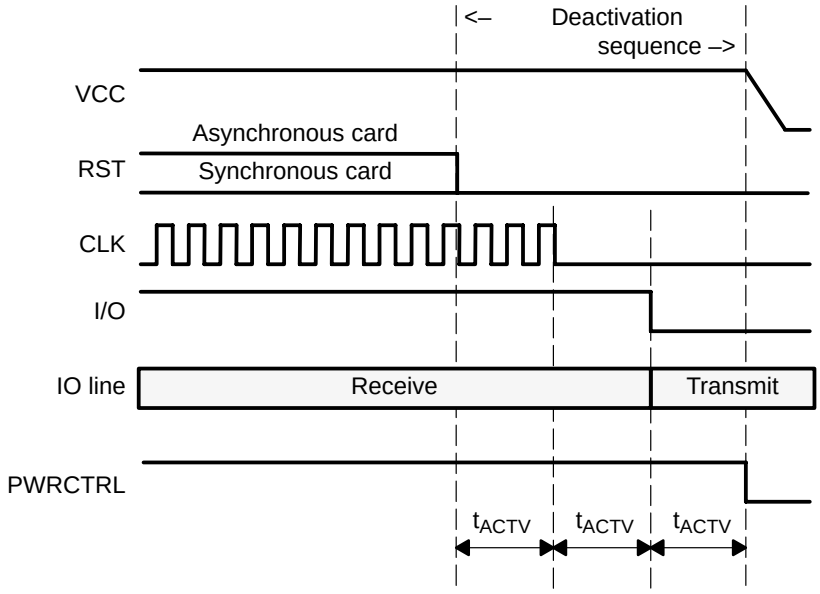
When information exchange is concluded or aborted (e.g., unresponsive card, detection of card extraction), the USIM must deactivate the electrical circuits (in response to stop command from MCU).

Deactivation sequence includes the following consecutive steps:

- ❏ Card reset is asserted low (SIM_RST = 0)
- ❏ Card clock is disabled (signal SIM_CLK inactive low)
- ❏ Bidirectional data line is set in transmit mode and driven low
- ❏ Card power-supply is switched off (signal SIM_PWRCTRL = 0)

Each of these steps is executed with a scheduling based on the configurable time period (T_{actv}) as used for the scheduling of the contacts activation phase.

Figure 20–16. Deactivation Sequence



20.1.7.1 Disconnection of the Contacts

The card can be extracted from the smart card driver if and only if the deactivation sequence is finished.

As the card can be extracted without any preliminary warning from the user, the duration of the deactivation sequence must be shorter than the time of disconnection. That means that the parameter t_{actv} must be calculated from the parameters of speed of extraction and length of the contacts.

20.2 SMC Protocol

20.2.1 Bit Duration

The smart card external clock frequency (F_{SCK}) can be equal to $1/N \times F_{CLK}$ with N = 4 or 8.

The nominal duration of a transmitted/received bit on the SIO link is called the elementary time unit (ETU). The ETU period depends of the two parameters F (conversion factor of the SIM clock rate) and D (adjusting factor of the binary baud rate) which are defined in character TA1 of answer-to-reset (ATR) sequence:

$$T_{ETU} = F/D \times T_{SCK}$$

Supported transmission factors (Fi/Di) are described in Section 20.2.1.3.

The accuracy for bit positioning in receive mode must be better than $0.2 \times$ ETU. So, an over-sampling frequency of $8 \times F_{ETU}$ is adopted for the detection of the start bit and thus allows to sample the bits with an accuracy 0.125 ETU (thus better than $0.2 \times$ ETU).

The switching of the data line from receive to transmit and vice-versa for the confirmation of the parity is realized in the time range of [$-0.2 \times$ ETU, $+0.2 \times$ ETU] as requested by ISO.

20.2.1.1 Coding Convention

The coding convention is based on the value of the first byte (TS) transmitted in the ATR of the smart card. Two conventions are proposed:

Direct

- ❏ LSB first
- ❏ Logic 1 coded on state Z (high level) and logic 0 coded on state A (low level)

Inverse

- ❏ MSB first
- ❏ Logic 1 coded on state A (low level) and logic 0 coded on state Z (high level)

Hardware identification of convention mode selection from TS character is adopted. Upon reset, the default coding convention is set as direct convention mode; hardware is then set up based on decoding of character TS.

When TS character is interpreted (based on the direct convention set by default), there are only two correct possibilities:

- ❏ TS= 3B H (that is, convention mode is direct)
- ❏ TS= 03 H (that is, convention mode is inverse)

If TS is not equal to 3B H or 03 H, then a coding convention error occurs, and an interrupt is generated to the M-controller (TS_ERROR in USIM_IT). However, direct convention (that is, default setup) is maintained.

20.2.1.2 Parity Check

The parity bit is calculated on the basis of the even parity of logic one rule as defined by coding convention (see Section 20.2.1.1). So:

- For the direct convention, the parity is based on an even number of state Z, that is,
  P = EXOR (b_0,b_1,b_2,b_3,b_4,b_5,b_6,b_7).
- For the inverse convention, the parity is based on an even number of state A, that is,
  P = EXNOR(b_0,b_1,b_2,b_3,b_4,b_5,b_6,b_7).

20.2.1.3 Transmission Factors (F/D)

As described in Section 20.2.1, the ETU period depends of the two parameters: F (conversion factor of the SIM clock rate) and D (adjusting factor of the binary baud rate), that is, $T_{ETU} = F/D^* T_{SCK}$.

During the ATR sequence, the initial ETU period is the one corresponding to the F and D default values (Fd, Dd). The default values are equal to (372, 1), respectively. Then the M-controller can either keep this period, or select other (F, D) parameters as defined during the ATR and/or the PPS procedures.

To ensure transmission and/or reception of characters, two clocks are used: The ETU clock and the over-sampling clock. Both clocks must be generated based on F and D factors.

In order to support all (F, D) pairs as mentioned in ISO/IEC 7816-3, the clock module must allow generation of all corresponding ETU and sampling clocks. It is adopted to implement clock generation using a hardware prescaler divider instead of using PLL. Instead of generating synchronously F_{ETU} from F_{SAM} by a 1:8 clock divider, both clocks are generated independently from the 13-MHz reference (F_{CLK}) using the following formulas:

$$T_{ETU\ approximated} = \text{Round}\ [\ (F / D) \times N\ ] \times T_{13MHz}$$

$$T_{SAM\ approximated} = \text{Round}\ [\ (F / D) \times (N / 8)\ ] \times T_{13MHz}$$

The corresponding values of ETU clock divider (that is, round[(F/D) $\times$ N]) and sampling clock divider (that is, round[(F/D) $\times$ (N/8)]) for a certain (F,D) pair is decided by hardware based on the combinatorial decoding shown in Section 20.6.3 to Section 20.6.6.

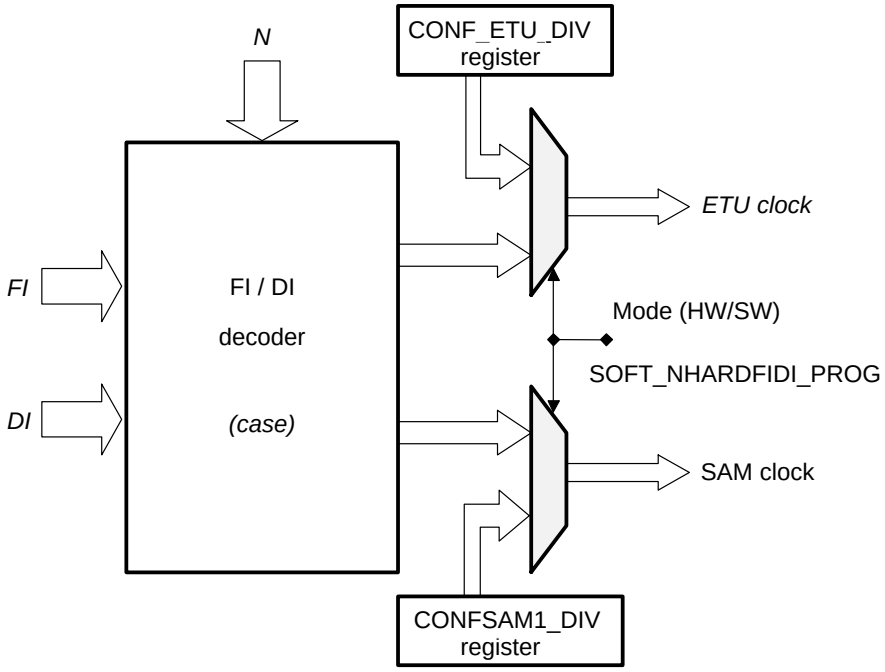
$$F_{SAM} = F_{CLK}/\text{CONFSAM1\_DIV};$$

$$F_{ETU} = F_{CLK}/\text{CONF\_ETU\_DIV}$$

CONF_ETU_DIV and CONFSAM1 factors are generated from the binary values FI and DI given in TA(1) character of ATR, and the binary coding of the clock divider for smart card clock (factor N). Optionally, in order to increase

flexibility for debugging purposes, for instance, and/or software programming, CONF_ETU_DIV and CONFSAM1 factors can also be stored in separate registers. The CONFETU_DIV value is stored in CONF_ETU_DIV register and the CONFSAM1_DIV value is stored in the CONFSAM1_DIV register.

Figure 20–17. Clock Generation Prescaler Divider



20.2.2 Timer Descriptions

A summary of main parameters for the activation and the character/block timers used in USIM, is given in ANNEXE B.

20.2.2.1 Work Waiting Time (WWT)

Work waiting time (WWT) is the maximum delay of the interval between the leading edge of any character sent by the card and the leading edge of the previous character (sent either by the card or by the card interface module).

The WWT timer flags the potential overflow of the waiting time between a received character (sent by the card to the ME card interface) and the previous character sent by the card. The WWT timer is also used to flag the potential overflow of the waiting time between a transmitted character and the next character (sent by the card to ME card interface). This is the maximum waiting time for answer of the card, after the switching from transmit to receive mode (see).

The work waiting time value is defined by the ATR sequence. WWT is based on the WI parameter from the TC2 character of ATR:

- For SIM and USIM cards, the work waiting time is given by the following formula:

$WWT = 960 \times D \times WI \times ETU = CONFWAITI \times 480 \times D \times ETU$
$(CONFWAITI = 2 \times WI)$

So,

- For SIM, USIM card:

  $WWT = CONFWAITI \times 480 \times D \times ETU$ (with $CONFWAITI = 2 \times WI$)

CONFWAITI is defined in CONF4_REG register. This timer is used for T = 0 protocol only, and the timer is reactivated each time a new character is received (start-bit detection).

20.2.2.2 Character Guard Time (CGT)

Character guard time (CGT) is the minimum guard time between a character sent from the ME card interface to the card, and the previous character (either sent by the card or the ME card interface). See Figure 20–11.

$CGT = (12+N) \times ETU$ for $0 \leq N \leq 254$

CGT = 12 for N = 255 and T = 0
CGT = 11 for N = 255 and T = 1

with:

Parameter N (additional guard time) defined by TC1 character of the ATR. The default value is N = 0. There is no additional guard time to send characters from the card to the ME card interface.

and

ETU based on:

F/D, that is, the values used for computing the ETU, if T = 15 is absent in the ATR

or

Fi/Di, if T = 15 is present in the ATR, Fi/Di are the values indicated by the card in TA(1)
if TA(1) is absent, then Fi and Di are set at default values.

The parameter of CGT timer is character guard time = CGT, with CGT being defined in USIM_CGT(7:0) register.

The CGT timer is applicable for both T = 0 and T = 1 protocols, and is activated each time a new character is transmitted; it is also activated for the switching from the receive mode to the transmit mode (that is, minimum delay is required before card is ready to accept a new character).

Note:

According to 3GPP, T = 15 parameters shall be returned by the UICC.

In the 3GPP specification, there is a minimum interval restriction between characters sent in opposite directions. This minimum interval is fixed (16 ETUs). See Figure 20–11.

20.2.2.3 Character Waiting Time (CWT)

Character waiting time (CWT) is defined as the maximum delay between the leading edges of two consecutive characters in a block.

The character waiting time value is defined by the ATR sequence; CWT is based on the CWI parameter from the first TBi character (the 4 LSBs) of ATR, after first occurrence of T = 1 in TD_{i-1} for i > 2.

The character waiting time is given by following formula:

$$CWT = (11 + 2^{CWI}) \text{ ETU}$$

For both 3GPPs, CWI is in the range 00 to 05.

Character waiting timer can be defined as:

$$CWT = CWT\_value \times Tetu$$

with:

$CWT\_value = 11+2^{CWI}$ (for USIM card) <=> CWT_value in the range [12,43]

$CWT\_value=15+2^{CWI}$ (for EMV card) <=> CWT_value in the range [16,47]

The CWT_value is defined in USIM_CWT(15:0) register.

CWT is used for T = 1 protocol only. When there is a potential error in the length, CWT may be used to detect the end of a block. This timer is reactivated each time a new character is received (start_bit detection).

20.2.2.4 Block Waiting Time (BWT)

Block waiting time (BWT) is the maximum delay between the last character of the block received by the card and the first character of the next block sent by the card to the ME.

The block waiting time value is defined by the ATR sequence; BWT is based on the BWI parameter from the first TBi character (the 4 MSBs) of ATR, after first occurrence of T = 1 in TD_{i-1} for i > 2.

The block waiting time is given by the following formula:

$$BWT = (11+2^{CONFBWI} \times 960 \times 372 \times D/F) \text{ ETU}$$

CONFBWI is defined in the USIM_BWT registers.

For 3GPP, BWI is in the range 00 to 09, with a default value of 4.

BWT is used for T = 1 only. The timer is reactivated each time a block is transmitted to the card (start bit of the last character of the block), and it is used to detect an unresponsive card.

20.2.2.5 Block Guard Time (BGT)

Block guard time (BGT) is the minimum guard time between two consecutive blocks sent in opposite directions.

The block guard time has a fixed value:

BGT = 22 ETU

Since BGT is constant, a fixed timer is adopted.

BGT is used for T = 1 only. This timer is reactivated each time a new block is received or transmitted.

20.2.2.6 ATR Response Timer

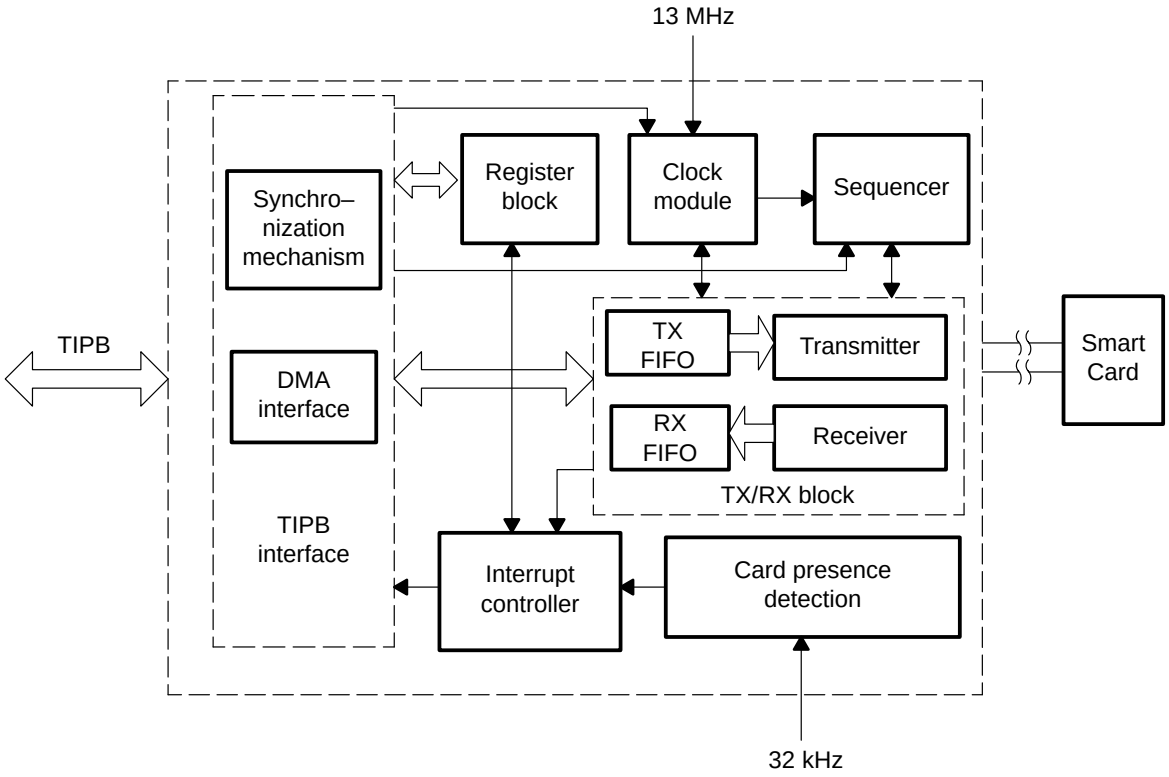
According to the EMV specification, the terminal must be able to receive an ATR with a duration of less than or equal to 20,160 ETUs. If ATR is not completed, then the terminal should initiate a deactivation sequence within 24,000 initial ETUs following the TS character. So, a fixed timer of 20,160 ETUs is used and reactivated when the TS character is received (start-bit detection).

This timer is only used with EMV card.

20.3 SMC Architecture

The whole module function is realized in several submodules, that is: TIPB interface submodule, clock management module, sequencer, interrupt management, register blocks, transmitter/receiver block, card presence detection submodule, etc. See Figure 20–14.

Figure 20–18. USIM Module Architecture



20.3.1 TIPB Interface Submodule

This submodule includes synchronization mechanism, DMA management interface, register address decoder, and other related circuits.

20.3.1.1 Synchronization Mechanism

Since TIPB strobe and SMC module clock are asynchronous, the synchronization mechanism is used to guarantee that M-controller can correctly exchange data with SMC module.

20.3.1.2 DMA Manager Interface

Data transfer can be executed by two methods: using DMA, or using interrupt (TIPB interrupts). Both methods use the FIFO. The corresponding control register bit is DMA_MODE in USIM_FIFOS register. With DMA_MODE set to 1, DMA transfer mode is adopted.

20.3.2 Clock Submodule

The clock submodule generates reference clocks for the main sequencer, the transmit/receiver blocks and associated timers, as well as for other submodules:

- Provides SIM_CLK to smart card clock contact, that is, 13/N MHz with N = 1, 2, 4, or 8
- Provides RX and TX ETU clocks (Fetu) to both receiver and transmitter blocks
- Provides sampling clocks: $F_{sam1} = 8 \times$ Fetu and F_{sam2} = Fetu to receiver block

The smart card clock (external clock F_{SCK}) and all submodules clocks (internal clocks) are derived from the 13-MHz reference clock signal.The smart card external clock frequency (F_{SCK}) can be equal to $1/N \times F_{CLK}$ with N = 1, 2, 4, or 8, and F_{CLK} being the free running clock of TIPB (13 MHz). However, the maximum frequency allowed on smart card clock contact (F_{SCK}) depends on the selected conversion factor for the clock (FI). Also, the default clock frequency applied to the smart card during the answer-to-reset (ATR) sequence (following a cold and warm reset) must be 13/4 MHz or 13/8 MHz in order to fulfill ISO 7816-3 requirements. The following clocks are generated:

- The ETU clock (F_{ETU}) which schedules the bit/character transmission on the SIO line.
- The oversampling clocks (F_{SAM1}/F_{SAM2}), which schedule the start bit detection and the bit/character reception on SIO line.

ETU Clock

The nominal duration of a transmitted bit on the SIO link is called the elementary time unit (ETU). The ETU period depends on two parameters: F (conversion factor of the SIM clock rate) and D (adjusting factor of the binary baud rate) which are defined in character TA1 of answer-to-reset (ATR) sequence:

$$T_{ETU} = F/D \times T_{SCLK} \leftrightarrow T_{ETU} = (F/D) \times N \times T_{13\ MHz}$$

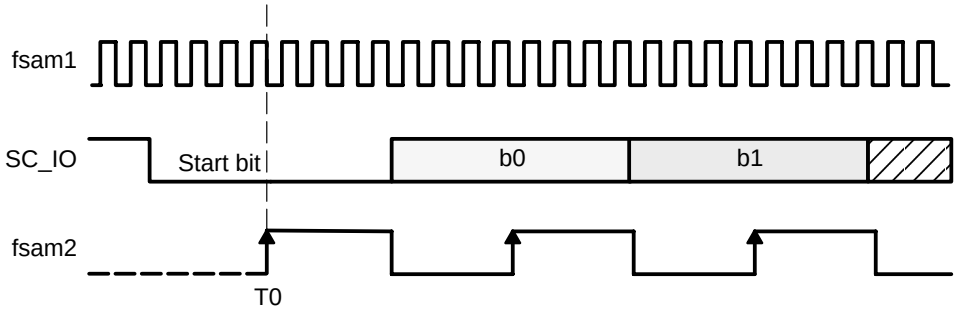
Oversampling Clocks

The accuracy for bit positioning in receive mode must be better than $0.2 \times$ ETU. So, an over-sampling frequency of $8 \times F_{ETU}$ is adopted for the detection of the start bit and thus allows to sample the bits with an accuracy 0.125 ETU (which is better than $0.2 \times$ ETU):

$$T_{SAM} = 1/8 \times T_{ETU} \leftrightarrow T_{SAM} = (F/D) \times (N/8) \times T_{13\ MHz}$$

To ensure correct reception of characters, two sampling clocks are used. One is used for the start bit detection ($F_{Sam1} = 8 \times F_{ETU}$), and the other is used for in-the-middle sampling of the bits of a character following the start bit detection ($F_{Sam2} = F_{ETU}$).

Figure 20–19. Oversampling Clocks



20.3.3 Sequencer Submodule

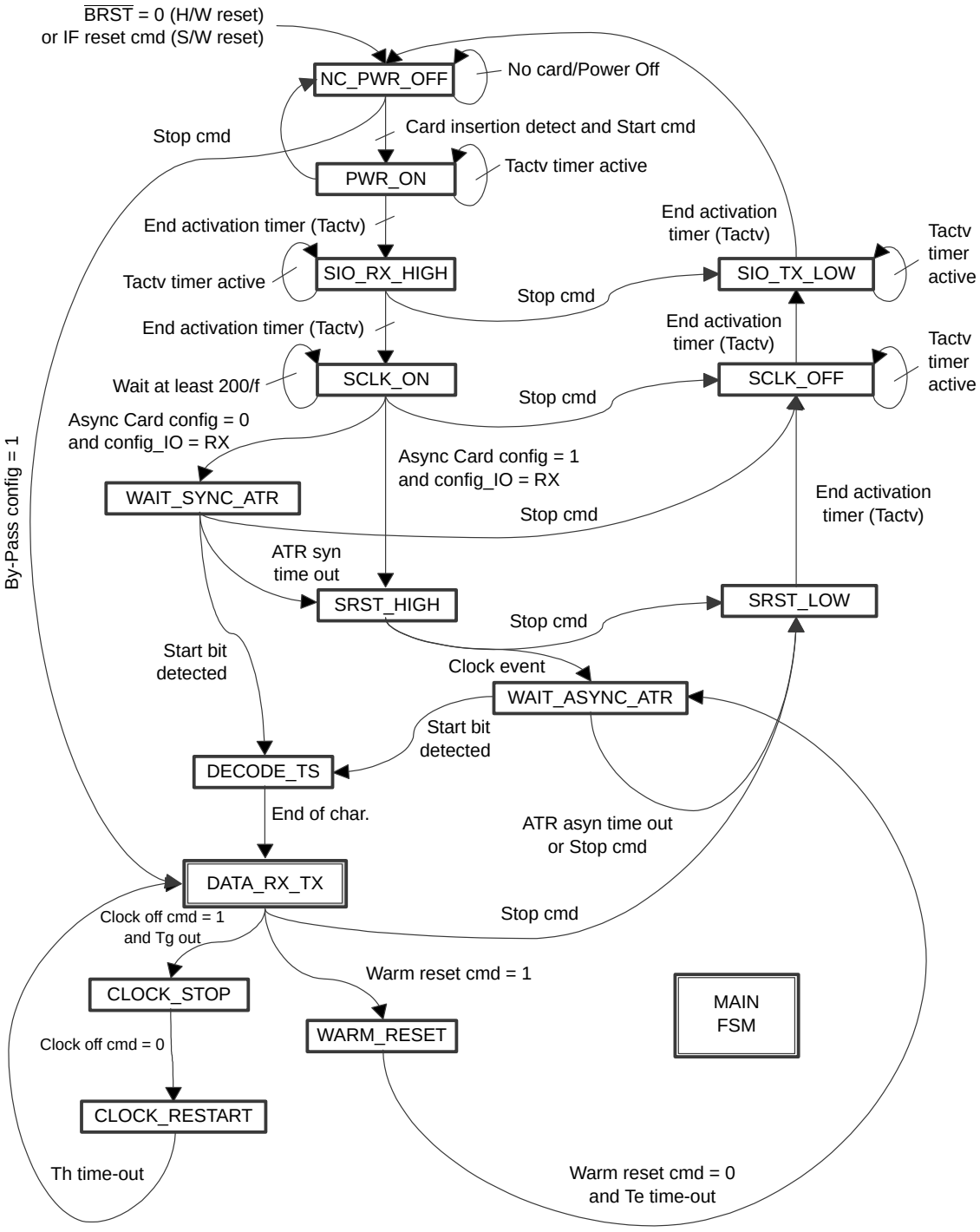
The sequencer of the module manages all the functional modes of the SIM interface with a possibility to bypass some automatic sequences if the manual mode is selected. It handles all the waiting times as defined by the ISO with, once again, the possibility to remove the corresponding hardware timers to give the full control to the M-controller.

The main sequencer is based on a finite state-machine scheduled on the ETU clock timing. The reception and transmission state machines are initiated by main sequencer (when in data communication mode) and scheduled on the ETU clock timing.

The states of the main FSM are:

NC_PWR_OFF	No card inserted and no card power supply
PWR_ON	Card power supply activated (signal SIM_PWRCTRL high)
SIO_RX_HIGH	IO line set in receive (RX) mode and driven high-Z
SCLK_ON	Card clock activated (signal SIM_CLK toggling)
WAIT_SYNC_ATR	Wait for ATR sequence of synchronous card
SRST_HIGH	Card reset released (signal SIM_RST inactive high)
WAIT_ASYNC_ATR	Wait for ATR sequence of asynchronous card
SRST_LOW	Card reset asserted (signal SIM_RST active low)
SCLK_OFF	Card clock disabled (signal SIM_CLK inactive low)
SIO_TX_LOW	IO line set in transmit (TX) mode and driven Low
DECODE_TS	Coding convention interpreted (direct/inverse)
DATA_RX_TX	Interface set in data receive (RX) or transmit (TX) mode
WARM_RESET	Card reset asserted for time te (signal SIM_RST active low)
CLOCK_STOP	Card clock stopped (signal SIM_CLK inactive low or high)
CLOCK_RESTART	Card clock restarted (signal SIM_CLK toggling)

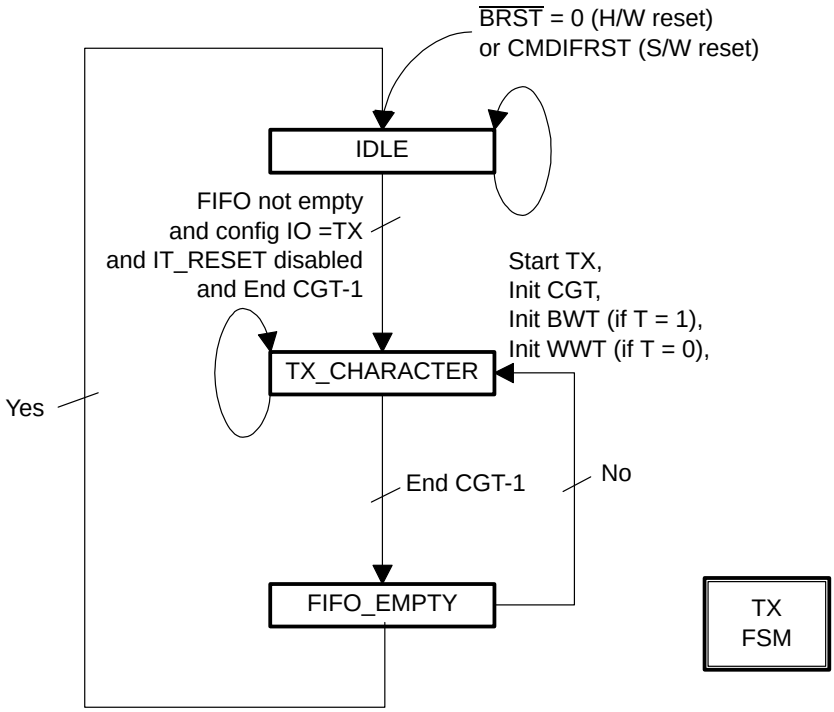
Figure 20–20. Main State Machine



The states of the RX-FSM are:

IDLE	Idle mode (no transmission with card clock active)
TX_CHARACTER	Data transmission (TX)
FIFO_EMPTY	FIFO management

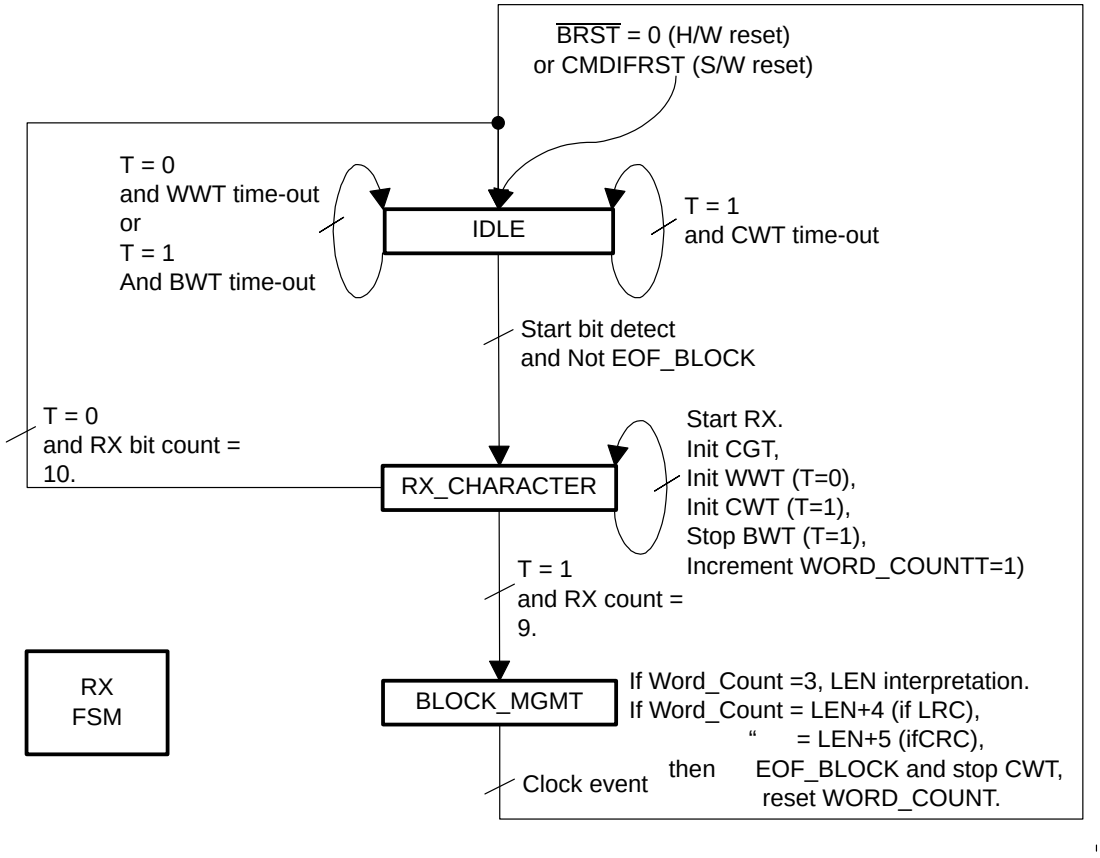
Figure 20–21. Reception (RX) State Machine



The states of the TX-FSM are:

IDLE	Idle mode (no transmission with card clock active)
RX_CHARACTER	Data reception (RX)
BLOCK_MGMT	Block counter management

Figure 20–22. Transmission (TX) State Machine



20.3.4 Transmit/Receive Submodule

This submodule is composed of the transmit block (TX), the receive block (RX), the timing control block, and the FIFO block.

20.3.4.1 Receiver

This block implements all the features defined in ISO/GSM/3GPP standards for T = 0 and T = 1 protocols.

It is scheduled on the receive (ETU/sampling) clocks with baud rates as defined by F and D parameters. The two clocks are used to interpret characters/blocks sent by the card. Sampling clock1 (Fsam1 = 8 × Fetu) is used to oversample the SIO line to detect the start-bit of a new character/block. When start-bit is detected, sampling clock 2 is activated (Fsam2 = Fetu) and each bit of a character is sampled in the middle of its period within ±2 ETUs (see Figure 20–19).

At time T0, after start bit detection, Fsam2 is activated (that is, in the middle of start bit period).Subsequent bits of a character are sampled at successive times t0+N × Tetu, with N in the 1–9 range.

The main features of the receiver block are:

- ❏ Interpretation of convention mode (direct or inverse)
- ❏ Detection of start bit (with Fsam1 = 8 × Fetu)
- ❏ Sampling of character byte plus parity bit (with Fsam2 = Fetu)
- ❏ Check between computed and received parity, and set parity flag STATRXPAR (USIM_DRX register)
- ❏ For T = 0 only, automatic request for character repetition if error detected For T = 1 only, check EDC and set parity flag STATRXEDC (see note)
- ❏ Ready to read interrupt generated to M-controller
- ❏ Write signal generated to RX FIFO
- ❏ Hardware management of block length and automatic reception of a block

Note:

The EDC is checked by hardware, if based on LRC. Otherwise (if CRC used) EDC checking is done by software.

20.3.4.2 Transmitter

This block implements all the features defined in ISO/GSM/3GPP/EMV standards for T = 0 and T = 1 protocols.

It is scheduled on the transmit ETU clock with baud rates as defined by F and D parameters (see section 20.2.1.3**,** *Transmission Factors*).

The main features are:

- ❏ Parity computation based on inverse or direct convention
- ❏ EDC computation (T = 1 only)
- ❏ Ready to transmit interrupt generated to M-controller
- ❏ Read signal generated to TX FIFO
- ❏ Hardware management of block length and automatic transmission of a block

When the smart card interface is configured in transmission mode, the transfer of a character starts as soon as a byte is available in the FIFO (that is, FIFO not empty).

20.3.4.3 FIFO

Although the transmission between the USIM and smart card is asynchronous and half-duplex, both receive path and transmit path integrate a FIFO (two pages, FIFO_RX and FIFO_TX), considering that the M-controller does not need to reload the last block in FIFO, when an error occurs and terminal needs to retransmit the block.

The main features of the FIFO are:

- ❏ Each page of FIFO has a 16-byte storage capacity.
- ❏ Dynamically programmable trigger level: 1 to 16 (for both FIFO_RX and FIFO_TX)
- ❏ Automatic flow control
- ❏ Flag FIFO full, FIFO empty, and FIFO trigger reached

The generation of the receive interrupt USIM_RX is related to the trigger level of the FIFO.

In case of a FIFO full detection, a dedicated interrupt USIM_OV is generated. The four status bits FIFOTX_FULL, FIFOTX_EMPTY, FIFORX_FULL, and FIFORX_EMPTY of the USIM_FIFOS register represent the state of FIFO_TX and FIFO_RX.

The FIFO trigger level can be modified on the flow during an ongoing reception phase. This allows to fit the FIFO size to the expected number of characters to be received, removing the need to wait for the work waiting time interrupt when the transmission is completed.

20.3.5 Interrupt Controller

The interrupts for smart card protocol management are mapped on the MPU IRQ line, and the smart card detect (insertion/extraction) is mapped on the MPU FIQ. The interrupt controller is associated with IT, MASK_IT, FIFO, and STATUS registers.

The main features include:

- ❏ Sets IT bits in interrupt register: USIM_IT
- ❏ Sets flag bits in status register: USIM_STAT
- ❏ Sets flag bits in FIFOs register: USIM_FIFOS
- ❏ Generates two interrupt signals (nirq and nirq_cd) to M-controller.

20.3.6 Registers Block

The registers block gathers the command and status registers writable and/or readable by the M-controller.

These registers are:

- ❏ The command register (CMD_START, CMD_STOP…)
- ❏ The status registers (FIFO states, Parity in TX/RX…)
- ❏ The status of card type register (SIM, USIM, EMV…)
- ❏ The interrupt status register (RX, TX, FIFO full…)

- ❏ The interrupt mask register
- ❏ The configuration registers (coding convention, parity check, timers ranges, FIFO trigger level, the control on external signals, etc)

20.3.7 Card Presence Detection Submodule (Card Detect Debouncing)

Because the terminal must be capable of sustaining a short circuit of any duration between any or all contacts without suffering damage or malfunction, detection of the smart card insertion or extraction is necessary.

The detection principle relies on the physical presence of the card in the card reader. To prevent signal oscillations (spring effect on contact), the USIM_CD signal can be debounced with an AND filter.

Based on the reference clock of the interface (32-kHz), the debouncing logic enables the smart card detect event as soon as a given number of consecutive samples of USIM_CD input have been detected at the high level. The same principle is applied to disable the SIM card detect-event as soon as a given number of consecutive samples of USIM_CD input have been detected at the low level.

The filtering duration is based on the TFUSIM parameter of the USIM_CONF3 register. Two different time units are adopted depending on whether the card is inserted or extracted, as the filtering duration must be consistent with the minimum contacts deactivation time to prevent from any card destruction in case of card extraction.

The filtering delay TFUSIM (T_{filter}) can be defined in the range:

$[1 \times \text{Time-unit}, 16 \times \text{Time-unit}] \leftrightarrow [500\ \mu s, 8\ ms]$ for card insertion

or

$[1 \times \text{Time-unit}, 16 \times \text{Time-unit}] \leftrightarrow [62.5\ \mu s, 1\ ms]$ for card extraction

with:

Time-unit = 16/f For card insertion (f = 32 kHz)

Time-unit = 2/f For card extraction (f = 32 kHz)

The filtering time can be defined by the following formula: TFUSIM = (CONFTFUSIM+1) $\times$ Time-unit.

The main features of the card detect submodule are:

- ❏ Based on the RTC clock of 32 kHz
- ❏ Fast interrupt generated to M-controller
- ❏ Flags the STATNOCARD bit in USIM_STAT register

20.4 SMC Registers

Table 20–1 lists the 16-bit SMC registers. Table 20–2 through Table 20–21 describe the register bits.

Table 20–1. SMC REGISTERS

Name	Description	R/W	Offset
USIMCMD	USIM control and command	R/W	0x00
USIMSTAT	USIM status	R	0x02
USIMCONF1	USIM configuration 1	R/W	0x04
USIMCONF2	USIM configuration 2	R/W	0x06
USIM_CONF3	USIM configuration 3	R/W	0x08
USIM_IT	USIM interrupts	R/W	0x0A
USIM_DRX	USIM received data	R	0x0C
USIM_DTX	USIM transmitted data	R/W	0x0E
USIM_MASK_IT	USIM mask interrupt	R/W	0x10
USIM_FIFOS	USIM RX/TX FIFO management	R/W	0x12
USIM_CGT	USIM character guard time	R/W	0x14
USIM_CWT	USIM character waiting time	R/W	0x16
USIM_BWT_LSB	USIM block waiting time low	R/W	0x18
USIM_BWT_MSB	USIM block waiting time high	R/W	0x1A
DEBUG_REG	SMC debug	R	0x1C
CONF_SAM1_DIV	SAM clock configuration 1	R/W	0x1E
CONF4_REG	SMC configuration 4	R/W	0x20
ATR_CLK_PRD_NBS	ATR clock period number	R	0x22
CONF_ETU_DIV	ETC clock configuration	R/W	0x24
CONF5_REG	SMC configuration 5	R/W	0x26

Table 20–2. USIM Control and Command (USIMCMD)

Bit	Name	Function	R/W	Reset Value
15:7	RESERVED	Reserved	R	0x0
6	STOP_EMV_ATR_LENGTH_TIMER	Toggle bit at 1 It allows to stop the EMV_ATR_LENGTH timer when the last word of the ATR have been received.	R/W	0x0
5	CLOCK_STOP_CMD	If 1, the clock_stop sequence is activated. Must be set to 0 to stop the clock stop mode.	R/W	0x0

Table 20–2. USIM Control and Command (USIMCMD) (Continued)

Bit	Name	Function	R/W	Reset Value
4	WARM_RESET_CMD	If 1, the warm reset sequence is activated. Must be set to 0 to stop the warm reset.	R/W	0x0
3	MODULE_CLK_EN	Activation of the clock of the module (active on level 1)	R/W	0x0
2	CMDSTART	Activation of start procedure for the USIM card. Active at 1	R/W	0x0
1	CMDSTOP	Activation of stop procedure for the USIM card. Active at 1	R/W	0x0
0	CMDIFRST	Soft reset of the module. Active at 1	R/W	0x0

Table 20–3.USIM Status Register (USIMSTAT)

Bit	Name	Function	R/W	Reset Value
15:5	RESERVED	Reserved	R	0x0
4	X_MODE	Effective direction of the IO line 0: Reception 1: Transmission	R	0x1
3	CONFCODCONV	Status of the coding convention used by the card. 0: Direct convention 1: Inverse convention	R	0x0
2	STATLRC	LRC check on receive block 0 if LRC OK 1 if LRC error Only when T = 1 protocol and edc_type = LRC	R	0x0
1	STATTXPAR	Parity check for transmitted byte 0: Parity error 1: Parity OK	R	0x1
0	STATNOCARD	Status on card presence (1 if present)	R	0x0

Table 20–4. USIM Configuration Register 1 (USIMCONF1)

Bit	Name	Function	R/W	Reset Value
15:7	RESERVED	Reserved	R	0x0
6	EMV_CONF	1 EMV card 0 not EMV card	R/W	0x0
5	CONF_SCLK_EN	Enables the sim clock. Used if CONFBYPASS = 1	R/W	0x0
4	SRSTLEV	Logic level on SRST (used if CONFBYPASS = 1)	R/W	0x0

Table 20–4. USIM Configuration Register 1 (USIMCONF1) (Continued)

Bit	Name	Function	R/W	Reset Value
3	SVCCLEV	Logic level on Svcc Used if CONFBYPASS = 1	R/W	0x0
2	CONFBYPASS	Bypasses hardware timers (tdusim) and start and stop sequence	R/W	0x0
1	CONFSIOLOW	Force SIO to low level Used if CONFBYPASS = 1 (Active at 1)	R/W	0x0
0	SCLKLEV	USIM clock idle level 0: Low 1: High	R/W	0x0

Table 20–5.USIM Configuration Register 2 (USIMCONF2)

Bit	Name	Function	R/W	Reset Value
15:11	RESERVED	Reserved	R	0x0
10:8	CONFRESENT	This value indicates the number of automatic resents on parity error in T = 0 protocol.	R/W	0x0
7	CONFLRCCHECK	0: No check 1: Check LRC Only when T = 1 protocol	R/W	0x0
6	CONFEDC	If 0 LRC, else CRC Only when T = 1 protocol	R/W	0x0
5	CONFPROTOCOL	0: T = 0 protocol 1: T = 1 protocol	R/W	0x0
4	ATR_ASYN_BYPASS	0: Checks for synchronous and asynchronous ATR 1: Bypass ATR waiting sequence for synchronous cards.	R/W	0x0
3:2	CONFSCLKDIV	Gives the frequency of the simcard clock 00: 13 MHz/2 01: 13 MHz/4 10: 13 MHz/8 11:13 MHz	R/W	0x01
1	TXNRX	SIO line direction (control bit) 0: Receive mode 1: Transmit mode	R/W	0x1
0	CONFCHKPAR	Enables parity check on reception	R/W	0x0

Table 20–6. USIM Configuration Register 3 (USIM_CONF3)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Reserved	R	0x00
7:4	TDUSIM	Coefficient for contact activation/deactivation duration time-unit = 8 × Tetu	R/W	0x4
3:0	TFUSIM	Coefficient for filtering depth of USIM_CD time-unit = 1024 × T13M (card extraction) or 8192 × T13M (card insertion)	R/W	0x0

Note: Debouncing/filtering time: TFUSIM = (CONFTFUSIM +1) × [1024 or 8192] × T13M Activation/deactivation time step: TDUSIM = (CONFTDUSIM +1) × 8 × Tetu work waiting timer (TA1): WWT = (CONFWAITI +1) × 960 × [1 or 8] × Tetu

Table 20–7. USIM Interrupts Register (USIM_IT)

Bit	Name	Function	R/W	Reset Value
15:12	RESERVED	Reserved	R	0x0
11	IT_EMV_ATR_LENGTH_TIME_OUT	Ensures that ATR sequence in EMV mode is less than 19200 SIM clock cycles.	R/W	0x0
10	TS_ERROR (see Note)	Interrupt flag for TS_decode error. (Active at 1) Only for T = 1.	R/W	0x0
9	USIM_RESENT (see Note)	Interrupt flag for CONF_RESENT(USIM_CONF2 reg) Only reach when T = 0 protocol. Once the same word has been sent (USIM_CONF2) times, the connection between the card and the usim is considered bad. Thus, the card must be reset before any other try. (Active at 1)	R/W	0x0
8	USIM_TOB (see Note)	Interrupt flag for time out block (BWT) (T = 1) (Active at 1)	R/W	0x0
7	USIM_TOC (see Note)	Interrupt flag for time out character (CWT) (T = 1) (Active at 1)	R/W	0x0
6	USIM_EOB (see Note)	Interrupt flag for end of block receive (T = 1) (Active at 1)	R/W	0x0
5	USIM_CD (see Note)	Interrupt flag for USIM card insertion/extraction (Active at 1)	R/W	0x0
4	USIM_RX (see Note)	Interrupt flag for maximum number of characters waiting to be read. RX_FIFO threshold is reached. (Active at 1)	R/W	0x0

Table 20–7. USIM Interrupts Register (USIM_IT) (Continued)

Bit	Name	Function	R/W	Reset Value
3	USIM_TX (see Note)	Interrupt flag for minimum number of characters waiting to be transmitted. TX_FIFO threshold is reached. (Active at 1)	R/W	0x0
2	USIM_OV (see Note)	Interrupt flag for receive overflow. RX_FIFO is full (Active at 1)	R/W	0x0
1	USIM_WT (see Note)	Interrupt flag for character underflow. (Active at 1)	R/W	0x0
0	USIM_NATR (see Note)	Interrupt flag for no ATR (Active at 1)	R/W	0x0

Note: Reset on write access to interrupt register REG_USIM_IT.

Table 20–8.USIM Received Data Register (USIM_DRX)

Bit	Name	Function	R/W	Reset Value
15:9	RESERVED	Reserved	R	0x0
8	STATRXPAR	Parity check for received byte 0: Not Ok 1: Ok	R	0x0
7:0	USIMDRX	Next data byte in the FIFO is ready to be read	R	0x0

Table 20–9.USIM Transmitted Data Register (USIM_DTX)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Reserved	R	0x0
7:0	USIM_DTX	Next data byte to be transmitted	W	Unknown

Table 20–10. USIM Mask Interrupt Register (USIM_MASK_IT)

Bit	Name	Function	R/W	Reset Value
15:12	RESERVED	Reserved	R/W	0x0
11	MASK_IT_EMV_ATR_LENGTH_TIME_OUT	Mask for ATR length timeout irq Only in EMV mode Active at 1	R/W	0x1
10	MASK_USIM_TS_ERROR	Mask interrupts for TS_DECODE error Active at 1	R/W	0x1
9	MASK_USIM_RESENT	Mask interrupts for CONF_RESENT reached (active at 1)	R/W	0x1

Table 20–10. USIM Mask Interrupt Register (USIM_MASK_IT) (Continued)

Bit	Name	Function	R/W	Reset Value
8	MASK_USIM_TOB	Mask interrupts for Time Out Block (BWT) (Active at 1) Only when T = 1	R/W	0x1
7	MASK_USIM_TOC	Mask interrupts for Time Out Character (Active at 1) T = 1 only	R/W	0x1
6	MASK_USIM_EOB	Mask interrupts for end of block receive (Active at 1) T = 1 only	R/W	0x1
5	MASK_USIM_CD	Mask interrupts for USIM card insertion/extraction (Active at 1)	R/W	0x1
4	MASK_USIM_RX	Mask interrupts for characters waiting to be read (Active at 1)	R/W	0x1
3	MASK_USIM_TX	Mask interrupts for characters waiting to be transmitted (Active at 1)	R/W	0x1
2	MASK_USIM_OV	Mask interrupts for receive overflow (Active at 1)	R/W	0x1
1	MASK_USIM_WT	Mask interrupts for character wait-time underflow (Active at 1)	R/W	0x1
0	MASK_USIM_NATR	Mask interrupts for no ATR (Active at 1)	R/W	0x1

Table 20–11. USIM RX/TX FIFO Management Register (USIM_FIFOS)

Bit	Name	Function	R/W	Reset Value
15	FIFORX_FULL	1: FIFO is full	R	0x0
14	FIFORX_EMPTY	1: FIFO is empty	R	0x1
13	FIFORX_RESET	1: Reset all the FIFO_RX pointers. Must be set to 0 to remove the reset.	R/W	0x0
12:9	FIFORX_TRIGGER	Value of the FIFO_RX trigger (0 to 15)	R/W	0x0
8	FIFOTX_FULL	1: FIFO_TX is full.	R	0x0
7	FIFOTX_EMPTY	1: FIFO_TX is empty	R	0x1
6	FIFOTX_RESET	1: Reset all the FIFO_TX pointers. Must be set to 0 to remove the reset .	R/W	0x0
5:2	FIFO_TX_TRIGGER	Value of the FIFO_TX trigger (0 to 15)	R/W	0xF

Table 20–11. USIM RX/TX FIFO Management Register (USIM_FIFOS) (Continued)

Bit	Name	Function	R/W	Reset Value
1	FIFO_ENABLE	0: R/W to the FIFOs is forbidden. 1: R/W to the FIFOs is allowed.	R/W	0x0
0	DMA_MODE	0: Not in DMA mode 1: DMA mode for FIFO_TX and FIFO_RX	R/W	0x0

Table 20–12. USIM Character Guard Time Register (USIM_CGT)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Reserved	R	0x0
7:0	CGT	The character guard time is in ETU. Default value is 13 ETUs.	R/W	0xD

Table 20–13. USIM Character Waiting Time Register (USIM_CWT)

Bit	Name	Function	R/W	Reset Value
15:0	CWT	The character waiting time is in ETUs. Default value is 8203 ETUs (ISO 7816-3). The value written in this register must be the 2^{CWI} of the following formula: $CWT = 11 + 2^{CWI}$. The module automatically adds 11 to the value written in the register.	R/W	0x200B

This register is only used for T = 1 protocol.

Table 20–14. USIM Block Waiting Time LSB Register (USIM_BWT_LSB)

Bit	Name	Function	R/W	Reset Value
15:0	BWT_LSB	Least-significant byte of BWT	R/W	0x3C0B

The BWT is specified in ETU and its default value is 15371 (ISO 7816-3). Card clock frequency (Fsclk) is equal to 13/4 MHz. ETU clock period is 372/Fsclk.

Table 20–15. USIM Block Waiting Time MSB Register (USIM_BWT_MSB)

Bit	Name	Function	R/W	Reset Value
15:7	RESERVED	Reserved	R	0x0000
6:0	BWT_MSB	Most-significant byte of the BWT	R/W	0x0

Table 20–16. SMC Debug Register (DEBUG_REG)

Bit	Name	Function	R/W	Reset Value
15:8	RESERVED	Reserved	R	0x0
7:6	RX_STATE_MACHINE	Current state of the reception state machine 00: Idle 01: Receive character on going 10: Block length management	R	0x0
5:4	TX_STATE_MACHINE	Current state of the transmission state machine 00: Idle 01: Transmit character on going 10: FIFO empty	R	0x0
3:0	MAIN_STATE_DEBUG	Current state of the main state machine: 0000: Not connected 0001: SVCC_on 0010: SIO_RX 0011: SCLK_on 0100: Wait_ATR_int 0101: SRST_on 0110: Wait_ATR_ext 0111: Clock_stop 1000: Clock_restart 1001: Warm_reset 1010: Decode_ts 1011: Communication 1100: SRST_off 1101: SCLK_off 1110: SIO_TX 1111: tdsim	R	0x0

Note: This register is not resynchronized.

Table 20–17. SAM Clock Configuration 1 Register (CONF_SAM1_DIV)

Bit	Name	Function	R/W	Reset Value
15:12	RESERVED	Reserved	R	0x0
11:0	SAM1_DIV	Number of 13-MHz periods – 1 Used for the ratio F/D of the SAM_clock when the software mode is programmed. See register CONF5 for the selection of the soft/hard mode programming. 186d at reset.	R/W	0xB9

Table 20–18. SMC Configuration 4 Register (CONF4_REG)

Bit	Name	Function	R/W	Reset Value
15:13	RESERVED	Reserved	R	0x0
12:0	CONFWAITI	CONFWAITI[7:0] Coefficient for character underflow wait time in RX_time unit = 960 × D × Tetu with D = 1 Only for T = 0	R/W	0xA

Table 20–19. ATR Clock Period Number Register (ATR_CLK_PRD_NBS)

Bit	Name	Function	R/W	Reset Value
15:0	CLOCK_NUMBER_BEFORE_ATR	Number of SIM clock periods before receipt of the start_bit of the ATR. The value is updated for each ATR.	R	0x0

Table 20–20. ETU Clock Configuration Register (CONF_ETU_DIV)

Bit	Name	Function	R/W	Reset Value
15:0	CONF_ETU_DIV	Number of 13 MHz periods –1 to calculate the ETU_clock period. Only used in software mode.	R/W	0x5CF

Note: This register must be set to the correct value (Fi = 372 and Di = 1) when a warm reset is done in software mode (see CONF5_REGISTER). In hardware mode the setting is automatic.

Table 20–21. SMC Configuration Register 5 (CONF5_REG)

Bit	Name	Function	R/W	Reset Value
15:9	RESERVED	Reserved	R	0x0
8	SOFT_NHARD_FIDI_PROG	0: USIM uses hardware coding: – FI and DI factors for the ratio F/D – CONFSCLKDIV for the sim_clock 1: CONF_SAM1 is used for the sampling clock period, and CONF_ETU_DIV for the ETU period.	R/W	0x1

Table 20–21. SMC Configuration Register 5 (CONF5_REG) (Continued)

Bit	Name	Function	R/W	Reset Value
7:4	CONFFI	Decoding of the FI value according to the TA(1) character of the smart card ATR. 0000: 372 0001: 372 0010: 558 0011: 744 0100: 1116 0101: 1488 0110: 1860 1001: 512 1010: 768 1011: 1024 1100: 1536 1101: 2048 Others: 372	R/W	0x0
3:0	DI	Decoding of the DI value according to the TA(1) character of the smart card ATR. 0001: 1 0010: 2 0010: 4 0100: 8 0101: 16 0110: 32 1000: 12 1001: 20 Others: 1	R/W	0x1

This register is a copy of the TA(1) word of the ATR. It has to be written by the software. It represents the FI and the DI of the smart card.

20.5 SMC Interface

20.5.1 General Description

The exchange of data between the smart card and the interface is based on a half-duplex asynchronous serial line.

The interface provides the card the reference clock SIM_CLK from which the baud rate for character transmission is derived. Two serial lines (SIM_SIORX or SIM_SIOTX) support the exchange of data. SIM_SIOEN controls these lines when they need to be in high impedance.

The activity of the smart card is under control of the interface that can switch on and off the clock SIM_CLK and the power-supply of the card through control signal SIM_PWCTRL. The interface drives the reset of the card through the dedicated pin SIM_RST.

An external device must be provided which uses a dedicated signal to indicate to the USIM any insertion or extraction of the card with SIM_SINEX. The SIM-LEN signal is the command of the 1-mA pull-down included in the SIM card I/O tactical pad. It forces the output of the buffer to 0 when SIM_PWCTRL is deactivated.

The M-controller monitors the USIM through a parallel memory interface supporting a TIPB protocol.

20.5.2 USIM Interface

20.5.2.1 Logical Description

The USIM is a bidirectional asynchronous half-duplex interface. Two serial line (SIM_SIORX or SIM_SIOTX) supports the exchange of data. The interface provides the card a reference clock (SIM_CLK) and a reset signal (SIM_RST). Moreover, the interface can control the power switching of the smart card through a control signal (SIM_PWRCTRL). The presence of a smart card is indicated by a dedicated input signal (SIM_SINEX).

Table 20–22. USIM Signals List

Signal Name	Definition	Type	Active Level	Reset Level	Note
SIM_CLK	Smart card clock	OUT	/ \	0	
SIM_SIORX	Smart card interface data in line	IN	0–1	HZ	
SIM_SIOTX	Smart card interface data out line	OUT	0–1	HZ	
SIM_SIOEN	Smart card data line enable	OUT	0–1	0	
SIM_RST	Smart card reset	OUT	0	0	
SIM_PWRCTRL	Smart card power supply ctrl	OUT	1	0	

Table 20–22. USIM Signals List (Continued)

Signal Name	Definition	Type	Active Level	Reset Level	Note
SIMLEN	SIM card I/O output level	OUT	1	1	
SIM_SINEX	Smart card presence	IN	0–1	—	1

Notes: 1) Programmable active level to match any external smart card detection logic.

20.5.2.2 Local Host Interface (TIPB/IT)

Table 20–23. TIPB Interface Signal List

Signal Name	Definition	Type	Size
NBRST†	Reset from the TIPB	IN	1
NSTROBE	Data strobe from the TIPB	IN	1
CS	Chip select from the TIPB	IN	5
A	Address bus from the TIPB	IN	5
RNW	Read/write from the TIPB	IN	1
DO	Data bus from the TIPB	IN	16
NREADY	Ready to the TIPB	OUT	1
PERHMAS	Peripheral word access size	OUT	2
DI	Data bus to the TIPB	OUT	16
nIRQ	SC interrupt to the TIPB	OUT	1
nIRQ_CD	Fast interrupt to the TIPB	OUT	1
nOE_CTRL	Output enable control to GSM 3-state	OUT	1
nOE_DI	Output enable data to GSM 3-state	OUT	1

† In addition to the software reset CMDIFRST, a hardware-reset nbrst is provided to initialize the logic at power-up.

20.5.2.3 Debug Signals

Table 20–24. TIPB Interface Signals List

Signal Name	Definition	Type	Size
DEBUG_MAIN_STATE	Current state of the main state machine	Out	4
DEBUG_TX_FSM	Current state of the TX state machine	Out	2
DEBUG_RX_FSM	Current state of the RX state machine	Out	2

20.5.2.4 Test

Scan Test

The module is configured in scan mode for test through the configuration signal SCAN_MODE. The USIM implements N (N ≤ 8) scan-chains in order to reduce the test time. The scan clock. SCAN_CLK is multiplexed with all the internal functional clocks.

Table 20–25. Test Signals List

Signal Name	Definition	Type	Size
SCAN_IN(N–1:0)	Scan serial data in	In	N
SCAN_OUT(N–1:0)	Scan serial data out	OUT	N
SCAN_CLK	Scan clock	In	1
SCAN_MODE	Scan configuration	In	1
SCAN_SHIFT	Scan shift enable	In	1

Other Signals

Reference clock:The USIM module receives the reference clock FCLK which is provided by the external VTCXO at 13 MHz.

Clock @ 32 kHz: For card presence detection, 32 kHz based filtering logic is adopted:CLK32.

P_CS(4:0): Hardware address

Table 20–26. Other Signals

Signal Name	Definition	Type	Size
FCLK	Module reference clock (13 MHz)	IN	1
CLK32	Module reference clock (32 kHz)	IN	1
P_CS(4:0)	Hardware address	IN	5

20.6 Baud Rates Supported by SMC

20.6.1 General Description

Several clocks are used in SCI module:

- F_{CLK} = 13 MHz, provided by FCLK of TIPB
- F_{SCK} =13/N MHz (N = 1, 2, 4, 8), clock given to the smart card
- F_{ETU} = D/F $\times$ F_{SCK}, ETU clock
- F_{SAM} = 8$\times$ F_{ETU}, Oversampling clock for start bit detection in receive mode with:
  - F = Conversion factor for ETU clock
  - D = Adjustment factor for ETU clock

In order to support as many (F,D) pairs as mentioned in ISO/IEC 7816–3, F_{ETU} and F_{SAM1} is directly generated from 13-MHz clock, but arithmetical precision cannot be achieved for all (F, D) pairs. So a rule is defined to pick out the (F,D) pairs supported by SCI:

- $t_{ETU\ approximated}$ = Round [(F/D) $\times$ N] $\times$ t_{13MHz}, approximated ETU clock
- $t_{SAM\ approximated}$ =Round [(F/D) $\times$ (N/8)] $\times$ t_{13MHz}, approximated SAM clock

20.6.1.1 Transmission Clock Definition in ISO7816-3

The baud rate period of the transmission clock of the data bit between the smart card and the physical interface device is called the elementary time unit (ETU). The ETU is defined by both the:

- Clock rate conversion factor F
- Bit rate adjustment factor D

From the system clock provided to the smart card f

$$1\ \text{etu} = \frac{F}{D} \times \frac{1}{f}$$

In order to compute the ETU, the pair of (F/D) factors can be equal to the following values:

- Fd and Dd default values equal to (372/1)
- Fi and Di specific values as defined in TA(1)
- Fn and Dn negotiated values after a PPS

The possible (F/D) pair values are defined in the ISO7816-3 standard:

Table 20–27. Fi Clock Rate Conversion Factor

FI	0000	0001	0010	0011	0100
Fi	372	372	558	744	1116
f (max) MHz	4	5	6	8	12

FI	0101	0110	0111	1000	1001
Fi	1488	1860	RUF	RUF	512
f (max) MHz	16	20	——	——	5

FI	1010	1011	1100	1101	111x
Fi	768	1024	1536	2048	RUF
f (max) MHz	7,5	10	15	20	——

Table 20–28. Di Bit Rate Adjustment Factor

DI	0000	0001	0010	0011	0100
Di	RUF	1	2	4	8

DI	0101	0110	0111	1000	1001
Di	16	32	RUF	12	20

DI	1010	1011	1100	1101	111x
Di	RUF	RUF	RUF	RUF	RUF

The default pair value (Fd/Dd) is used during the answer-to-reset. If TA(1) byte is part of the ATR, according to the bit5 value of the TA(2) byte, either the specific pair value (Fi/Di) is used for the subsequent communication or an implicit pair value preprogrammed in the physical interface device.

If TA(2) byte is missing in the ATR, the interface device can initiate a protocol parameter selection and propose to the card a (Fn/Dn) pair value with Fn and Dn respectively in the range Fd to Fi and Dd to Di. If the card accepts the proposed (Fn/Dn) pair value, then this pair is used to define the ETU for the subsequent data exchanges.

20.6.2 Supported (F,D) Pairs

Table 20–29 through Table 20–36 show which (F,D) pairs can be used. The grayed values indicate invalid configurations for which F_{SCIK} = F13MHZ × (1/N) exceed the f(max) value specified in ISO 7816-3 standard for a given Fi. The remaining values indicate the (F,D) pairs that can be supported with the selected card clock frequency.

20.6.3 F_{SCLK} = 13/1 MHz

The divider factor (CONFETU_DIV) for the ETU clock generation from 13 MHz is shown in Table 20–29.

Table 20–29. $t_{ETU\ Approximated}/t_{13MHz}$

D →	1	2	4	8	16	32	12	20
F	N =							
372	372	186	93	47	23	12	31	19
558	558	279	140	70	35	17	47	28
744	744	372	186	93	47	23	62	37
1116	1116	558	279	140	70	35	93	56
1488	1488	744	372	186	93	47	124	74
1860	1860	930	465	233	116	58	155	93
512	512	256	128	64	32	16	43	26
768	768	384	192	96	48	24	64	38
1024	1024	512	256	128	64	32	85	51
1536	1536	768	384	192	96	48	128	77
2048	2048	1024	512	256	128	64	171	102

The divider factor (CONFSAM_DIV) for the start bit sample from 13 MHz is shown in Table 20–30.

Table 20–30. $t_{SAM\ Approximated}/t_{13MHz}$

D →	1	2	4	8	16	32	12	20
F	N =							
372	47	23	12	6	3	1	4	2
558	70	35	17	9	4	2	6	3
744	93	47	23	12	6	3	8	5
1116	140	70	35	17	9	4	12	7
1488	186	93	47	23	12	6	16	9
1860	233	116	58	29	15	7	19	12
512	64	32	16	8	4	2	5	3
768	96	46	24	12	6	3	8	5
1024	126	64	32	16	8	4	11	6
1536	192	96	48	24	12	6	16	10
2048	256	128	64	32	16	8	21	13

20.6.4 F_{SCLK} = 13/2 MHz

The divider factor (CONF_ETU_DIV) for the ETU clock generation from 13 MHz is shown Table 20–31.

Table 20–31. $t_{ETU\ Approximated}/t_{13MHz}$

D →	1	2	4	8	16	32	12	20
F	N =							
372	744	372	186	93	47	23	62	37
558	1116	558	279	140	70	35	93	56
744	1488	744	372	186	93	47	124	74
1116	2232	1116	558	279	140	70	186	112
1488	2976	1488	744	372	186	93	248	149
1860	3720	1860	930	465	233	116	310	186
512	1024	512	256	128	64	32	85	51
768	1536	768	384	192	96	48	128	77
1024	2048	1024	512	256	128	64	171	102
1536	3072	1536	768	384	192	96	256	154
2048	4096	2048	1024	512	256	128	341	205

The divider factor (CONFSAM1_DIV) for start bit sample clock generation from 13 MHz is shown in Table 20–32.

Table 20–32. $t_{SAM\ Approximated}/t_{13MHz}$

D →	1	2	4	8	16	32	12	20
F	N =							
372	93	47	23	12	6	3	8	5
558	140	70	35	17	9	4	12	7
744	186	93	47	23	12	6	16	9
1116	279	140	70	35	17	9	23	14
1488	372	186	93	47	23	12	31	19
1860	465	233	116	58	29	15	39	23
512	128	64	32	16	8	4	11	6
768	192	96	48	24	12	6	16	10
1024	256	128	64	32	16	8	21	13
1536	384	192	96	48	24	12	32	19
2048	512	256	128	64	32	16	43	26

20.6.5 F_{SCLK} = 13/4 MHz

The divider factor (CONF_ETU_DIV) for the ETU clock generation from 13 MHz is shown Table 20–33.

Table 20–33. $t_{ETU\ Approximated}/t_{13MHz}$

D →	1	2	4	8	16	32	12	20
F	N =							
372	1488	744	372	186	93	47	124	74
558	2232	1116	558	279	140	70	186	112
744	2976	1488	744	372	186	93	248	149
1116	4464	2232	1116	558	279	140	372	223
1488	5952	2976	1488	744	372	186	496	298
1860	7440	3720	1860	930	465	233	620	372
512	2048	1024	512	256	128	64	171	102
768	3072	1536	768	384	192	96	256	154
1024	4096	2048	1024	512	256	128	341	205
1536	6144	3072	1536	768	384	192	512	307
2048	8192	4096	2048	1024	512	256	683	410

The divider factor (CONFSAM1_DIV) for start bit sample clock generation from 13 MHz is shown in Table 20–34.

Table 20–34. $t_{SAM\ Approximated}/t_{13MHz}$

D →	1	2	4	8	16	32	12	20
F	N =							
372	186	93	47	23	12	6	16	9
558	279	140	70	35	17	9	23	14
744	372	186	93	47	23	12	31	19
1116	558	279	140	70	35	17	47	28
1488	744	372	186	93	47	23	62	37
1860	930	465	233	116	58	29	78	47
512	256	128	64	32	16	8	21	13
768	384	192	96	48	24	12	32	19
1024	512	256	128	64	32	16	48	26
1536	768	384	192	96	48	24	64	38
2048	1024	512	256	128	64	32	85	51

20.6.6 F_{SCLK} = 13/8 MHz

The divider factor (CONF_ETU_DIV) for the ETU clock generation from 13 MHz is shown Table 20–35.

Table 20–35. $t_{ETU\ Approximated}/t_{13MHz}$

D →	1	2	4	8	16	32	12	20
F	N =							
372	2976	1488	744	372	186	93	248	149
558	4464	2232	1116	558	279	140	372	223
744	5952	2976	1488	744	372	186	496	298
1116	8928	4464	2232	1116	558	279	744	446
1488	11904	5952	2976	1488	744	372	992	595
1860	14880	7440	3720	1860	930	465	1240	744
512	4096	2048	1024	512	256	128	341	205
768	6144	3072	1536	768	384	192	512	307
1024	8192	4096	2048	1024	512	256	683	410
1536	12288	6144	3072	1536	768	384	1024	614
2048	16384	8192	4096	2048	1024	512	1365	819

The divider factor (CONFSAM_DIV) for start bit sample clock generation from 13 MHz is shown in Table 20–36.

Table 20–36. $t_{SAM\ Approximated}/t_{13MHz}$

D →	1	2	4	8	16	32	12	20
F	N =							
372	372	186	93	47	23	12	31	19
558	558	279	140	70	35	17	47	28
744	744	372	186	93	47	23	62	37
1116	1116	558	279	140	70	35	93	56
1488	1488	744	372	186	93	47	124	74
1860	1860	930	465	233	116	58	155	93
512	512	256	128	64	32	16	43	26
768	768	384	192	96	48	24	64	38
1024	1024	512	256	128	64	32	85	51
1536	1536	768	384	192	96	48	128	77
2048	2048	1024	512	256	128	64	171	102

Table 20–37. Character and Block Timers

						Range					
Timer	**Parameter**	**Delay Type**	**Applicable Standard**	**Conditions**	**Delay Formula**	**Default Value**	**MAX Value**	**MIN Value**	**Step**	**Unit**	**Reference**
Guard time between a character and previous character	CGT	MIN	ISO, GSM, EMV, 3GPP	For T = 0 & T = 1; Consecutive characteristics in same direction sent from ME to Card.	If 0 < N < 254, then CGT = (12+N) etu; If N = 255 & T = 0, then CGT = 12 etu; If N = 255 & T = 1, then CGT = 11 etu	12 (N=0)	12 (T=0) 11 (T=1)	266	1	ETU	ISO 7816-3/ Clause 6.5.3
		MIN	ISO, GSM, EMV, 3GPP	For T = 0 & T = 1; Consecutive characteristics in same direction sent from Card to ME.	Fixed value	12	12		1	ETU	ISO 7816-3/ Clause 6.3.2; EMV Book1/ Clause 5.2.2.1; 3GTS 31.101/ Clause 7.2.2.1
		MIN	ISO, GSM	For T = 0 & T = 1; Characters sent in opposite direction.	See CGT above formulas	12 (N=0)	12 (T=0) 11 (T=1)	266	1	ETU	ISO 7816-3/ Clause 6.3.2
		MIN	3GPP, EMV		Fixed value	16	16		1	ETU	EMV Book1/ Clause 5.2.2.1; 3GTS 31.101/ Clause 7.2.2.1
Waiting time between a character and next character (send from Card to ME)	WWT	MAX	ISO, GSM, 3GPP, EMV	For T = 0 only; Characters sent from Card to ME (characters in same or in opposite direction)	WWT = 960*WI*D etu with W1 = [1,255]	9600*D (WI=10 & D=1)	960*1	960*8160	960	ETU	ISO 7816-3/Clause 8.2
	WWT	MAX	EMV	For T = 0 only; Interpretation by ME of character sent by Card to ME (characters in same or in opposite direction)	WWT = 480*(2*WI+1)*D etu with WI = [1,255]	10060*D (WI=10 & D=1)	960*3	960*16352	960	ETU	EMV Book1/ Clause 5.2.2.1
Waiting time between two consecutive characters in a block	CWT	MAX	ISO, GSM	For T = 1 only; Characters in same direction	CWT = $(11+2^{CWI})$ etu with CWI = [0,15]	8203 (CWI=13)	32779	12	2	ETU	ISO 7816-3/ Clause 9.5.3.1
	CWT	MAX	3GPP		CWT = $(11+2^{CWI})$ etu with CWI = [0,5]	TBD See Note 2	43	12	2	ETU	3GTS 31.101/ Clauses 7.2.3.1.2 & 7.2.3.1.3
	CWT	MAX	EMV		CWT = CWT+4 ETUs with CWI = [0,5]; CWI = $(15+2^{CWI})$ etu	TBD See Note 3	47	16	2	ETU	EMV Book1/ Clause 5.2.4.22

Table 20–37. Character and Block Timers (Continued)

Waiting time between last characters of the received block and the first character of the next block sent by the Card to the ME	BWT	MAX	ISO, GSM, 3GPP (see Note 1)	For T = 1 only; Characters (blocks) sent in opposite direction (transition ME = Tx to Card = Tx)	BWT = $(11+2^{BWI}*960*372*D/F)$ etu with BWI = [0,9]	11+2*960 (BWI=4 & D/F=1/372)	11+2*960*372 *D/F	11+960*372* D/F	960*372*D/F	ETU	ISO 7816-3/ Clause 9.5.3.2
	BWT	MAX	EMV		BWT = $(11+2^{BWI}*960*372*D/F)$ etu with BWI = [0,4]	TBD See Note 3	11+2*960*372 *D/F	11+960*372* D/F	960*372*D/F	ETU	EMV Book1/ Clause 5.2.4.2.2
	BWT	MAX	EMV	For T = 1 only; Interpretation by ME of character sent by Card to ME, after transition ME = Tx to Card = Tx.	BWT = $[(11+2^{BWI}*960*372*D/F) + D*960]$ etu with BWI = [0,4]	TBD See Note 3	11+2*960*372 D/F+D*960	11+960*372* D/F+D*960	960*372*D/F	ETU	EMV Book1/ Clause 5.2.4.2.2
Guard time between two consecutive blocks sent in opposite directions	BGT	MIN	ISO, GSM, 3GPP, EMV	For T = 1 only; Characters (blocks) sent in opposite direction.	Fixed value	22	22	22	N/A	ETU	ISO 7816-3/ Clause 9.5.3.3; 3GTS 31.101/ Clause 7.2.2.1; EMV Book1/ Clause 5.2.4.2.2

Notes: 1) In 3GPP (3GTS 31.101/Clause 7.2.3.1.4), it just define the BWT, but not defined its equation or BWI range.

2) 3GPP standard do not define the default value in the reference documents.

3) EMV standard do not define the default value in the reference documents.

20.6.6.1 WWT & BWT Timers (Min/Max) Range

*Table 20–38. WWT = 960*WI*D ETU, with WI = (1,255)*

Default value = 9600 (i.e., WI = 10 and D = 1)

D =	1		2		4		8		16		32		12		20	
F =	Min WI=1	Max WI=256	Min WI=0	Max WI=255	Min WI=0	Max WI=255	Min WI=0	Max WI=255	Min WI=0	Max WI=255	Min WI=0	Max WI=255	Min WI=0	Max WI=255	Min WI=0	Max WI=255
Don't Care	960	244,800	1,920	489,600	3,840	979,200	7,680	1,958,400	15,360	3,916,800	30,720	7,833,600	11,520	2,937,600	19,200	4,896,000

*Table 20–39. WWT = 960*WI*D ETU, with WI = (1,255)*

Default value = 15371 (i.e., BWI = 10 and D/F = 1/372)

D =	1		2		4		8		16		32		12		20	
F =	Min BWI=0	Max BWI=9	Min BWI=0	Max BWI=9	Min BWI=0	Max BWI=9	Min BWI=0	Max BWI=9	Min BWI=0	Max BWI=9	Min BWI=0	Max BWI=9	Min BWI=0	Max BWI=9	Min BWI=0	Max BWI=9
372	971	491,531	1,931	983,051	3,851	1,966,091	7,691	3,932,171	15,371	7,864,331	30,731	15,728,651	11,531	5,898,251	19,211	9,830,411
558	651	327,691	1,291	655,371	2,571	1,310,731	5,131	2,62,451	10,251	5,242,891	20,491	10,485,771	7,961	3,932,171	12,811	6,553,611
744	491	245,771	971	491,531	1,931	983,051	3,851	1,966,091	7,691	3,932,171	15,371	7,864,331	5,771	2,949,131	9,611	4,915,211
1116	331	163,851	651	327,691	1,291	655,371	2,571	1,310,731	5,131	2,621,451	10,251	5,242,391	3,851	1,966,091	6,411	3,276,811
1488	251	122,891	491	245,771	971	491,531	1,931	983,051	3,851	1,966,091	7,691	3,932,171	2,891	1,474,571	4,811	2,457,611
1060	203	98,315	395	196,619	779	393,227	1,547	786,443	3,083	1,572,875	6,155	3,145,739	2,315	1,179,659	3,851	1,966,091
512	709	357,131	1406	714,251	2801	1,428,491	5,591	2,856,971	11,171	5,713,931	22,331	11,427,851	8,381	4,285,451	13,961	7,142,411
768	476	238,091	941	476,171	1871	952,331	3,731	1,904,651	7,451	3,809,291	14,891	7,618,571	5,591	2,856,971	9,311	4,761,611
1024	360	178,571	709	357,131	1406	714,251	2,801	1,428,491	5,591	2,856,971	11,171	5,713,931	4,196	2,142,731	6,986	3,571,211
1536	244	119,051	476	238,091	941	476,171	1,871	952,331	3,731	1,904,651	7,451	3,809,291	2,801	1,428,491	4,661	2,380,811
2048	185	89,291	360	178,571	709	357,131	1,406	714,251	2,801	1,428,491	5,591	2,856,971	2,104	1,071,371	3,499	1,785,611

Chapter 21

Dual-Mode Timer

This chapter discusses the dual-mode timer of the OMAP730 multimedia processor.

Topic **Page**

21.1 Introduction

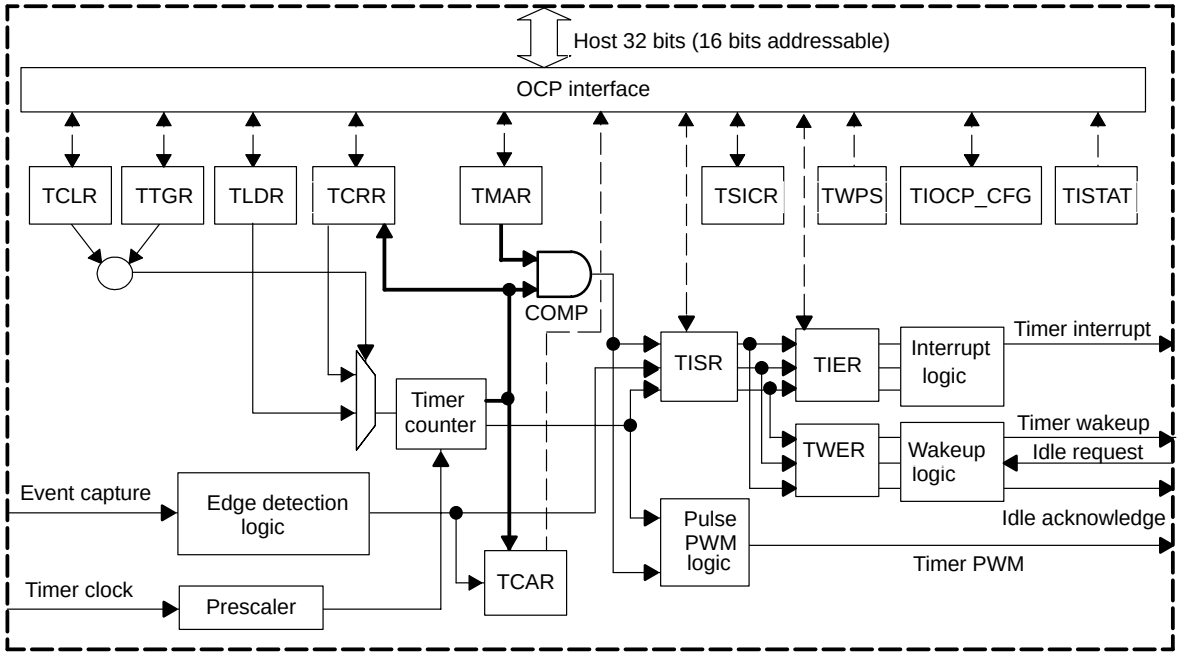
This OMAP730 peripheral is a 32-bit timer with the following features:

- Counter timer with compare and capture modes
- Autoreload mode
- Start-stop mode
- Programmable divider clock source
- 16-/32-bit addressing
- On the fly read/write registers
- Interrupts generated on overflow, compare and capture
- Interrupt enable
- Wake-up enable
- Write posted mode
- Dedicated input trigger for capture mode and dedicated output trigger/ PWM signal

The timer module contains a free-running upward counter with autoreload capability on overflow. The timer counter can be read and written on the fly (while counting). The timer module includes compare logic to allow interrupt event on programmable counter matching value. A dedicated output signal can be pulsed or toggled on overflow and match event. This offers timing stamp trigger signal or PWM (pulse width modulation) signal sources. A dedicated input signal can be used to trigger automatic timer counter capture and interrupt event, on programmable input signal transition type. A programmable clock divider (prescaler) allows reduction of the timer input clock frequency. All internal timer interrupt sources are merged into one module interrupt line and one wake-up line. Each internal interrupt sources can be independently enabled/ disabled with a dedicated bit of the TIER register for the interrupt features and a dedicated bit of TWER for the wake-up.

Figure 21–1 shows the dual-mode timer.

Figure 21–1. Dual-Mode Timer Block Diagram



21.2 Timer Functionality

The general-purpose timer is an upward counter. It supports three functional modes:

- ❏ Timer mode
- ❏ Capture mode
- ❏ Compare mode

By default, after core reset, the capture and compare modes are disabled.

21.2.1 One-Shot and Autoreload Mode Functionality

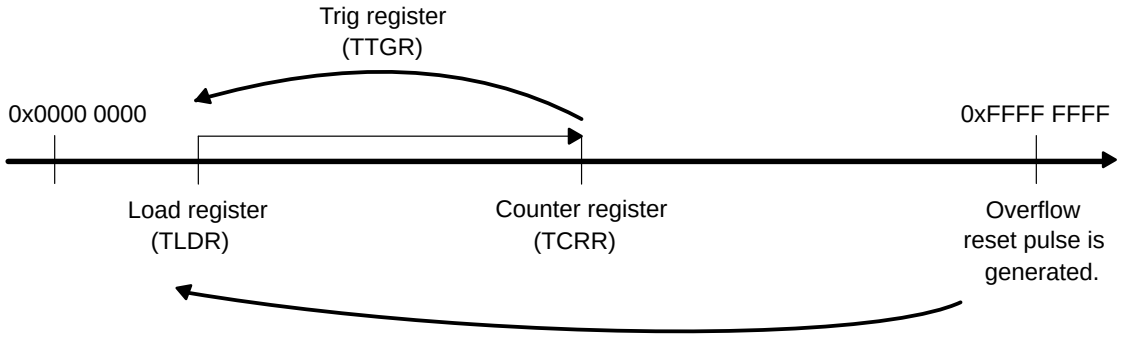
The timer is an upward counter that can be started and stopped at any time through the timer control register (TCLR ST bit). The timer counter register (TCRR) can be loaded when stopped or on the fly (while counting). TCRR can be loaded directly by a TCRR write access with the new timer value. TCRR can also be loaded with the value held in the timer load register TLDR by a trig register (TTGR) write access. The TCRR loading is done regardless the TTGR written value. The timer counter register TCRR value can be read when stopped or captured on the fly by a TCRR read access. The timer is stopped and the counter value set to 0 when the modules reset is asserted. The timer is maintained in stop after reset is released. When the timer is stopped TCRR is frozen and it can be restarted from the frozen value unless TCRR has been reloaded with a new value.

In one-shot mode (TCLR AR bit =0), the counter is stopped after counting overflow (counter value remains at zero).

When autoreload mode is enabled (TCLR AR bit =1), the TCRR is reloaded with the timer load register (TLDR) value after a counting overflow. An interrupt can be issued on overflow if the overflow interrupt enable bit is set in the timer interrupt enable register (TIER OVF_IT_ENA bit =1). A dedicated output pin (timer PWM) can be programmed through TCLR (TRG and PT bits) to generate one positive pulse (prescaler duration) or to invert the current value (toggle mode) when an overflow occurs.

Figure 21–2 shows the TCRR timing value.

Figure 21–2. TCRR Timing Value



21.2.2 Capture Mode Functionality

The timer value in TCRR can be captured and saved in TCAR when a transition is detected on the module input pin (event capture). The edge detection circuitry monitors transitions on the input pin (event capture).

Rising transition, falling transition, or both can be selected in TCLR (TCM bit) to trigger the timer counter capture. The module sets the TISR (TCAR_IT_FLAG bit) when an active transition is detected, and at the same time the counter value TCRR is stored in the timer capture register TCAR. If there are several consecutive capture events, the first one is saved in TCAR.

The edge detection logic is reset when the TCAR_IT_FLAG is cleared by writing a 1 to it, or when edge detection mode bits TCLR (TCM bit) passed from the no-capture mode detection to any other modes. The timer functional clock (input to prescaler) is used to sample the input pin (event capture). Input negative or positive pulse can be detected when pulse time is above functional clock period. An interrupt is issued on transition detection if the capture interrupt enable bit is set in the timer interrupt enable register TIER (TCAR_IT_ENA bit).

21.2.3 Compare Mode Functionality

When the compare enable TCLR (CE bit) is set to 1, the timer value (TCRR) is permanently compared to the value held in the timer match register (TMAR). TMAR value can be loaded at any time (timer counting or stop). When the TCRR and the TMAR values match, an interrupt can be issued if the TIER (MAT_IT_ENA bit) is set.

The dedicated output pin (timer PWM) can be programmed through TCLR (TRG and PT bits) to generate one positive pulse (timer clock duration) or to invert the current value (toggle mode) when an overflow and a match occur.

21.2.4 Prescaler Functionality

A prescaler counter can be used to divide the timer counter input clock frequency. The prescaler is enabled when TCLR bit 5 is set (PRE). The 2^n division ratio value (PTV) can be configured in the TCLR register.

The prescaler counter is reset when the timer counter is stopped or reloaded on the fly.

Table 21–1 lists the prescaler/timer reload values versus contexts.

Table 21–1. Prescaler/Timer Reload Values Versus Contexts

Contexts	Prescaler Counter	Timer Counter
Overflow (when autoreload on)	Reset	TLDR
TCRR write	Reset	TCRR
TTGR write	Reset	TLDR
Stop	Reset	Frozen

21.2.5 Pulse-Width Modulation

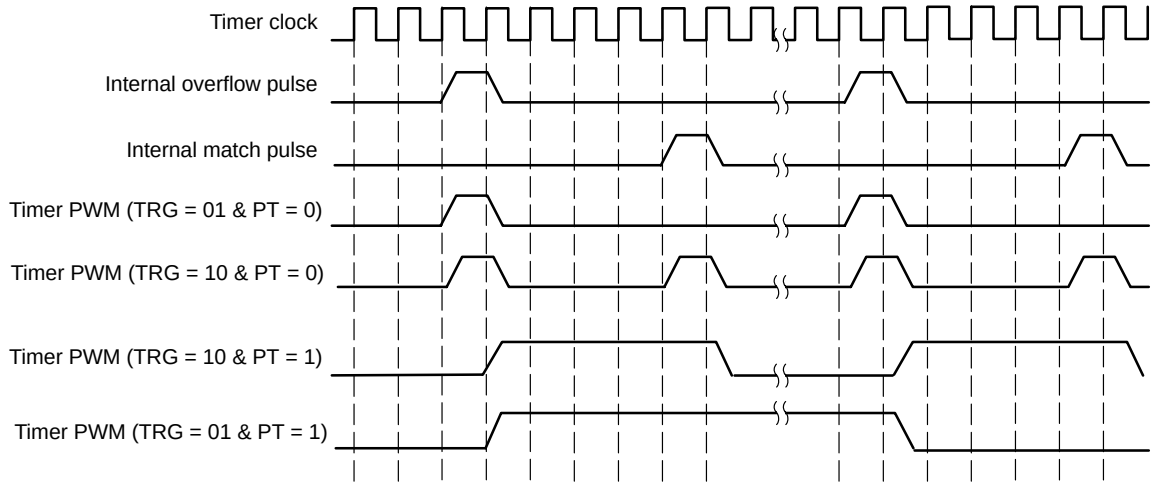
The timer can be configured to provide a programmable pulse-width modulation (timer PWM) output. The timer PWM output pin can be configured to toggle on specified event. TCLR (TRG bits) determine on which register value the timer PWM pin toggles. Either overflow or match can be used to toggle the timer PWM pin, when a compare condition occurs. The TCLR (SCPWM bit) can only be programmed to set or clear the timer PWM output signal while the counter is stopped or the trigger is off. This allows fixing a deterministic state of the output pin before modulation starts. The modulation is synchronously stopped when TRG bit is cleared and overflow occurred. This allows fixing a deterministic state of the output pin when modulation is stopped.

In Figure 21–3, the internal overflow pulse is set each time (0xFFFF FFFFF – TLDR +1) value is reached, and the internal match pulse is set when the counter reaches TMAR register value. According to TCLR (TRG and PT bits) value, the timer provides pulse or PWM on the output pin (timer PWM).

In Figure 21–3 TCLR (SCPWM bit) is set to 0.

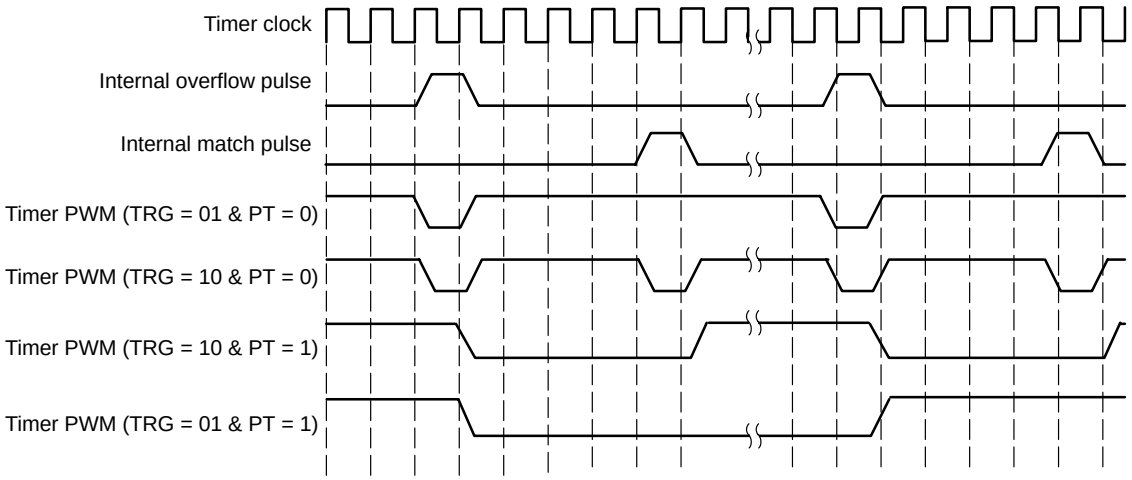
The TLDR and TMAR registers must keep values smaller than the overflow value (0xFFFFFFFF) with at least two units. In case the PWM trigger events are both overflow and match, the difference between the values kept in TMAR register and the value in TLDR must be at least two units. When match event is used, the compare mode TCLR (CE) must be set.

Figure 21–3. Timing Diagram of Pulse-Width Modulation With SCPWM Bit = 0



In Figure 21–4 TCLR (SCPWM bit) is set to 1.

Figure 21–4. Timing Diagram of Pulse-Width Modulation With SCPWM Bit = 1



21.2.6 Timer Interrupt Control

The timer can issue an overflow interrupt, a timer match interrupt, and a timer capture interrupt. Each internal interrupt source can be independently enabled/disabled in the interrupt enable register TIER. When the interrupt event has been issued, the associated interrupt status bit is set in the timer status register (TISR). The pending interrupt event is reset when the set status bit is overwritten by a 1 value. Reading the interrupt status register and writing the value back allows fast interrupt acknowledge.

21.2.7 Sleep Mode Request and Acknowledge

Upon a sleep mode request issued by the host processor, the timer module goes into sleep mode depending on the IDLEMODE field of the system configuration register (see TIOCP_CFG).

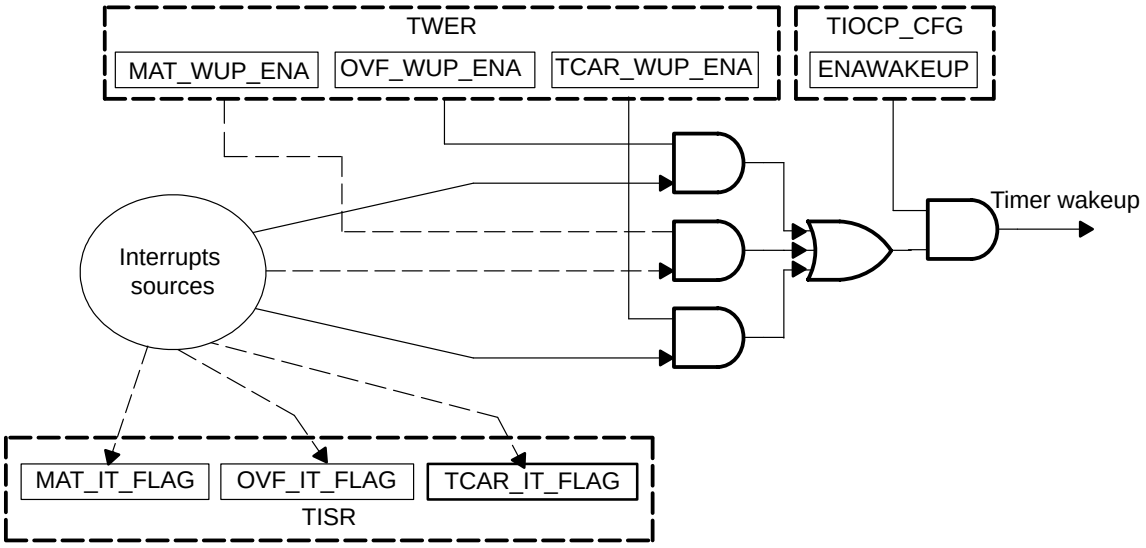
If the IDLEMODE field sets no-idle mode, the timer does not go into sleep mode and the idle acknowledge signal is never asserted.

If the IDLEMODE field sets force-idle mode, the timer goes into sleep mode independently of the internal module state, and the idle acknowledge signal is unconditionally asserted.

If the IDLEMODE field sets smart-idle mode, the timer module evaluates its internal capability to have the interface clock switched off. Once there is no more internal activity (no pending interrupt sources: match, overflow, or timer capture events), the idle acknowledge signal is asserted and the timer enters into sleep mode, ready to issue a wake-up request. This wake-up request is sent only if the field ENAWAKEUP of TIOCP_CFG enables the timer wake-up capability.

Figure 21–5 shows the wake-up request generation.

Figure 21–5. Wake-Up Request Generation



21.2.7.1 Wake-up Line Release

When the host processor receives a wake-up request issued by the timer peripheral, the interface clock is reactivated, the host processor deactivates the idle request signal, the timer deactivates the idle acknowledge signal, and the host then can read the corresponding bit in TISR to find out which interrupt source has triggered the wake-up request. After acknowledging the wake-up request, the processor resets the status bit and releases the interrupt line by writing a 1 in the corresponding bit of the TISR register.

21.2.8 Timer Counting Rate

The dual-mode timer is composed of a prescaler stage and a timer counter. The prescaler stage is clocked with the timer clock and acts as a clock divider for the timer counter stage.

Ratio can be managed by accessing the ratio definition field of the control register (PTV and PRE of TCLR).

The timer rate is defined by:

- ❏ Value of the prescaler fields (PRE and PTV of TCLR register)
- ❏ Value loaded into the timer load register (TLDR)

Table 21–2 lists prescaler clock ratios values.

Table 21–2. Prescaler Clock Ratios Values

PRE	PTV	Divisor (PS)
0	X	1
1	0	2
1	1	4
1	2	8
1	3	16
1	4	32
1	5	64
1	6	128
1	7	256

The timer rate equation is as follows:

(0xFFFF FFFF – TLDR + 1) x timer clock period x clock divider (PS)

With timer clock period = 1/ timer clock frequency and PS = $2^{(PTV + 1)}$.

For example, with a timer clock input of 32 kHz and a PRE field equal to 0, the timer output period is as shown in Table 21–3.

Table 21–3. Value and Corresponding Interrupt Period

TLDR	Interrupt Period
0x0000 0000	37 h
0xFFFF 0000	2 s
0xFFFF FFF0	500 µs
0xFFFF FFFF	31.25 µs

21.2.9 Dual-Mode Timer Under Emulation

During emulation mode, the dual-mode timer can/cannot continue to run according to the value of the EMUFREE bit of the timer OCP configuration register (TIOCP_CFG).

If EMUFREE is 1, timer execution is not stopped in emulation mode and the interrupt assertion is still generated when overflow is reached.

If EMUFREE is 0, counters (prescaler/timer) are frozen and an increment start occurs again on exit from emulation mode. The asynchronous input pin is internally synchronized on two timer-clock rising edges.

21.2.10 Accessing Registers

All registers are 32 bits wide, accessible via OCP interface with 16-bit or 32-bit OCP access (read/write).

21.2.11 Programming Timer Registers

The host uses the OCP bus protocol to write the TLDR, TCRR, TIER, TISR, TCLR, TIOCP_CFG, TWER, TTGR, TSICR, and TMAR registers synchronously with the timer interface clock.

The 32-bit register write update in 16-bit access mode must write the least significant 16 bits (LBS16) before writing the most-significant 16 bits (MSB16).

21.2.12 Reading Timer Registers

The counter register is a 32-bit atomic datum, and this 16-bit capture is performed on the 16-bit LSB first to allow atomic LSB16 + MSB16 capture.

21.2.13 Writing Timer Registers

21.2.13.1 Software Write Posting Synchronization Mode

This mode is used if TSCR (POSTED bit) is set to 1 in the timer control register.

This mode uses a posted write scheme for updating any internal register. This means that the write transaction is immediately acknowledged on the OCP interface, although the effective write operation occurs later, because of a resynchronization in the timer clock domain. The advantage is that neither the interconnect nor the CPU that requested the write transaction is stalled. For each register, a status bit is provided that is set if there is a pending write access to the register.

In this mode, it is mandatory that the CPU check this status bit prior to any write access. In case a write is attempted to a register with a previous access pending, the previous access is discarded without notice.

There is one status bit per register, accessible in the timer write-posted status register.

When the timer module operates in this mode, there is an automatic sampling of the current timer counter value, in an OCP-synchronized capture register. Consequently, any read access to the timer counter register does not add any resynchronization latency; the current value is always available.

A register read following a write post register on the same register is not ensured to read the previous write value if the write-posting process is not completed. Software synchronization must be used to avoid noncoherent read.

The drawback of this automatic update mechanism is that it assumes a given relationship between the timer interface frequency and the timer clock frequency:

Functional frequency range: freq(timer clock) ≤ freq(host peripheral clock)/4

21.2.13.2 Software Non-Write Posting Synchronization Mode

This mode is used if TSCR (POSTED bit) is set to 0 in the timer control register.

This mode uses a non-posted write scheme for updating any internal register. This means that the write transaction is not acknowledged on the OCP interface until the effective write operation occurs after the resynchronization in the timer clock domain. The drawback is that both the interconnect and the CPU are stalled during this period.

- The CPU cannot serve interrupts, the latency of which increases.
- An interconnect including time-out logic to detect erroneous transactions can generate an unwanted system abort event.

This stall period can be quantified:

- t (stall) = 3 MPU peripheral clocks + 5 timer clocks

The same full resynchronization scheme is used for a read transaction. The same stall period applies.

A register read following a write to the same register is always coherent.

This mode is functional whatever the ratio between the OCP interface frequency and the functional clock frequency.

21.2.13.3 Write Mode Selection

The choice between the synchronization modes must consider the frequency ratio and the stall periods that can be supported by the system without affecting the global performance.

21.3 Dual-Mode Timer Registers

Address of register: Start address + Offset address

Bit width: 32 bits (can be 32 bits or 2 × 16 bits)

Table 21–4 lists the 32-bit dual-mode timer registers. Table 21–5 through Table 21–18 describe the individual register bits. The registers are accessible in 16-bit mode and use little-endian addressing.

Table 21–4. Dual-Mode Timer Registers

Register	Description	Access	Offset LSB	Offset MSB
TIDR	Timer identification	Read	0x00	0x02
Reserved		-	0x04	0x06
Reserved		-	0x08	0x0A
Reserved		-	0x0C	0x0E
TIOCP_CFG	Timer OCP configuration	Read/write	0x10	0x12
TISTAT	Timer system status	Read	0x14	0x16
TISR	Timer status	Read/write	0x18	0x1A
TIER	Timer interrupt enable	Read/write	0x1C	0x1E
TWER	Timer wakeup enable	Read/write	0x20	0x22
TCLR	Timer control	Read/write	0x24	0x26
TCRR	Timer counter	Read/write	0x28	0x2A
TLDR	Timer load	Read/write	0x2C	0x2E
TTGR	Timer trig	Read/write	0x30	0x32
TWPS	Timer write posted status	Read	0x34	0x36
TMAR	Timer match	Read/write	0x38	0x3A
TCAR	Timer capture	Read	0x3C	0x3E
TSICR	Timer synchronization interface control	Read/write	0x40	0x42

Table 21–5.Timer Identification Register (TIDR)

Bit	Name	Function	Reset Value
31:8	RESERVED		0x000000
7:0	TID_REV	Module HW revision number	HW ID revision

The timer identification register is a read-only register that contains the revision number of the module. A write to this register has no effect.

This TID_REV field indicates the revision number of the current timer module. This value is fixed by hardware.

The four LSBs of TID_REV indicate a minor revision. The four MSBs of TID_REV indicate a major revision.

For example: 0x10 => Version 1.0

A reset has no effect on the value returned.

Table 21–6. Timer OCP Configuration Register (TIOCP_CFG)

Bit	Name	Function	Reset Value
31:6	RESERVED		0x0000000
5	EMUFREE	0: The timer is frozen in emulation mode. 1: The timer runs free.	0
4:3	IDLEMODE	Power management, request/acknowledge control 00: Force idle. An idle request is acknowledged unconditionally. 01: No idle. An idle request is never acknowledged. 10: Smart idle. Acknowledgement to an idle request is given based on the internal activity of the timer (see Section 21.2.7, *Sleep Mode Request and Acknowledge*). 11: Reserved	0x00
2	ENAWAKEUP	Wake-up feature control 0: Wake-up is disabled. 1: Wake-up is enabled.	0
1	SOFTRESET	Software reset. Set this bit to 1 to trigger a module reset. The bit is automatically reset by hardware. During reads, it always returns 0. 0: Normal mode 1: Reset the OCP and the functional domain	0
0	AUTOIDLE	Interface clocks gating strategy 0: Interface clock is free-running. 1: Automatic interface gating strategy is applied, based on the interface activity.	0

The timer OCP configuration register allows controlling various parameters of the OCP interface.

Table 21–7. TImer System Status (TISTAT)

Bit	Name	Function	Reset Value
31:1	RESERVED		0x0000000
0	RESETDONE	Internal global reset monitoring 1: Reset completed 0: Internal module reset is ongoing	-

The timer system status register monitors the internal global reset status. The status bit is set to one when all clock domains have been reset. This status can be monitored by the software to check if the module is ready-to-use following a reset (either hardware or software reset).

Table 21–8. Timer Status Register (TISR)

Bit	Name	Function	Reset Value
31:3	RESERVED		0x0000000
2	TCAR_IT_FLAG	0: No capture interrupt request 1: Capture interrupt request	0
1	OVF_IT_FLAG	0: No overflow interrupt request 1: Overflow interrupt pending	0
0	MAT_IT_FLAG	0: No compare interrupt request 1: Compare interrupt pending	0

The timer status register is used to determine which of the timer events requested an interrupt. Bit 0 corresponds to the compare result of TCRR and TMAR and is set when the compare register matches the counter value; bit 1 corresponds to the TCRR overflow; and bit 2 indicates that an external pulse transition of the correct polarity is detected on the external event capture pin. If the value is 1, then that timer event is requesting the interrupt. If the user wants to reset the status bit, then a 1 must be written to the appropriate bit. Writing a 1 to the bit TCAR_IT_FLAG resets the edge detection logic. However, the user cannot generate an interrupt by writing a 1 to the timer status register bits. If the user writes a 0 to a bit in the timer status register bits, the value remains unchanged.

Table 21–9. Timer Interrupt Enable Register (TIER)

Bit	Name	Function	Reset Value
31:3	RESERVED		0x00000000
2	TCAR_IT_ENA	Capture interrupt enable 0: Interrupt disabled 1: Interrupt enabled	0
1	OVF_IT_ENA	Overflow interrupt enable 0: Interrupt disabled 1: Interrupt enabled	0
0	MAT_IT_ENA	Match interrupt enable 0: Interrupt disabled 1: Interrupt enabled	0

The timer interrupt enable register allows the user to enable certain timer events for generating an interrupt request. The bit 0 determines whether or not the compare register flag MAT_IT_FLAG can generate an interrupt. Bit 1 determines whether or not the overflow counter flag OVF_IT_FLAG can gener-

ate an interrupt. Bit 2 determines whether the edge detection flag TCAR_IT_FLAG can generate an interrupt.

Table 21–10. Timer Wake-Up Enable Register (TWER)

Bit	Name	Function	Reset Value
31:3	RESERVED		0x00000000
2	TCAR_WUP_ENA	0: Wake-up generation is disabled. 1: Wake-up generation is enabled.	0x0
1	OVF_WUP_ENA	0: Wake-up generation is disabled. 1: Wake-up generation is enabled.	0x0
0	MAT_WUP_ENA	0: Wake-up generation is disabled. 1: Wake-up generation is enabled.	0x0

The timer wake-up enable register (TWER) allows the user to mask the expected source of a wake-up event that generates a wake-up request. The TWER is programmed synchronously with the interface clock before any idle mode request coming from the host processor.

Table 21–11. Timer Control Register (TCLR)

Bit	Name	Function	Reset Value
31:13	RESERVED		0x00000
12	PT	Pulse or toggle mode on timer PWM output pin 0: Pulse 1: Toggle	0x0
11:10	TRG	Trigger output mode on timer PWM output pin 00: No trigger 01: Trigger on overflow 10: Trigger on overflow and match 11: Reserved	0x0
9:8	TCM	Transition capture mode on event capture input pin 00: No capture 01: Capture on low to high transition 10: Capture on high to low transition 11: Capture on both edge transition	0x0
7	SCPWM	This bit must be set or cleared while the timer is stopped or the trigger is off. 1: Set the timer PWM output pin and select negative pulse for pulse mode. 0: Clear the timer PWM output pin and select positive pulse for pulse mode.	0

Table 21–11. Timer Control Register (TCLR) (Continued)

Bit	Name	Function	Reset Value
6	CE	1: Compare mode is enabled. 0: Compare mode is disabled.	0
5	PRE	Prescaler enable 0: The timer clock input pin clocks the counter. 1: The divided input pin clocks the counter.	0
4:2	PTV	Prescale clock timer value	0x0
1	AR	1: Autoreload timer 0: One-shot timer	0
0	ST	1: Start timer 0: Stop timer: Only the counter is frozen In case of one-shot mode selected (AR = 0), this bit is automatically reset by internal logic when the counter overflows.	0

Table 21–12. Timer Counter Register (TCRR)

Bit	Name	Function	Reset Value
31:0	TIME_COUNTER	Value of timer counter	0x0000000

The timer counter register is a 32-bit register (16-bit addressable). Therefore, the MPU can perform one 32-bit access or two 16-bit accesses to the register while the DSP performs two consecutive 16-bit transactions. Because the timer interface clock is completely asynchronous with the timer clock, some synchronization is done to ensure that the TCRR value is not read while it is being incremented.

In 16-bit mode, the following sequence must be followed to read the TCRR register properly:

1) Read the lower 16-bits of the TCRR register (offset = 0x28). When the TCRR is read, the lower 16-bit LSB are read, and the upper 16-bits of the TCRR MSB register are stored in a temporary register.

2) Read the upper 16 bits of the TCRR register (offset = 0x2A). During this read, the value of the upper 16-bit MSB that has been stored in the temporary register is read.

Therefore, to read the value of TCRR correctly, the first OCP read access must be to the lower 16-bits (that is, offset = 0x28), followed by OCP read access to the upper 16-bits (that is, offset= 0x2A).

Table 21–13. Timer Load Register (TLDR)

Bit	Name	Function	Reset Value
31:0	TIME_VALUE	Timer counter value loaded on overflow in autoreload mode or on TTGR write access	0x00000000

Table 21–14. Timer Trigger Register (TTGR)

Bit	Name	Function	Reset Value
31:0	TTGR_VALUE	Writing in the TTGR register, TCRR is loaded from TLDR and prescaler counter is cleared. Reload is done regardless the AR field value of TCLR register.	0xFFFF FFFF

The read value of the timer trigger register is always 0xFFFF FFFF.

Table 21–15. Timer Write Posted Status Register (TWPS)

Bit	Name	Function	Reset Value
31:5	RESERVED		0x0000000
4	W_PEND_TMAR	When equal to one, a write is pending to the TMAR register.	0
3	W_PEND_TTGR	When equal to one, a write is pending to the TTGR register.	0
2	W_PEND_TLDR	When equal to one, a write is pending to the TLDR register.	0
1	W_PEND_TCRR	When equal to one, a write is pending to the TCRR register.	0
0	W_PEND_TCLR	When equal to one, a write is pending to the TCLR register.	0

In posting mode, the software must read the pending write status bits (timer write posted status register bits [4:0]) to ensure that following write access is not discarded because of an ongoing write synchronization process. These bits are automatically cleared by internal logic when the write to the corresponding register is acknowledged. The TISR and TIER registers interface only with the timer interface clock, so they do not need a write pending status bit.

Table 21–16. Timer Match Register (TMAR)

Bit	Name	Function	Reset Value
31:0	COMPARE VALUE	Value to be compared to the timer counter	0x0000000

The compare logic consists of a 32-bit wide, read/write data, timer match register, and logic to compare the counter current value with the value stored in the TMAR.

Table 21–17. Timer Capture Register (TCAR)

Bit	Name	Function	Reset Value
31:0	CAPTURED VALUE	Timer counter value captured on an external event trigger	0x00000000

When the appropriate transition (rising, falling, or both) is detected in the edge detection logic, the current counter value is stored to the time capture register.

Table 21–18. Timer Synchronization Interface Control Register (TSICR)

Bit	Name	Function	Reset Value
31:2	RESERVED		0x00000000
2	POSTED	1: Posted mode active (clocks ratio needs to fit the MPU peripheral clock > 4 timer clock frequency requirement) 0: Posted mode inactive	1
1	SFT	This bit resets the all of the functional parts of the module. During reads, it always returns 0. 1: Software reset is enabled. 0: Software reset is disabled.	0
0	RESERVED		0

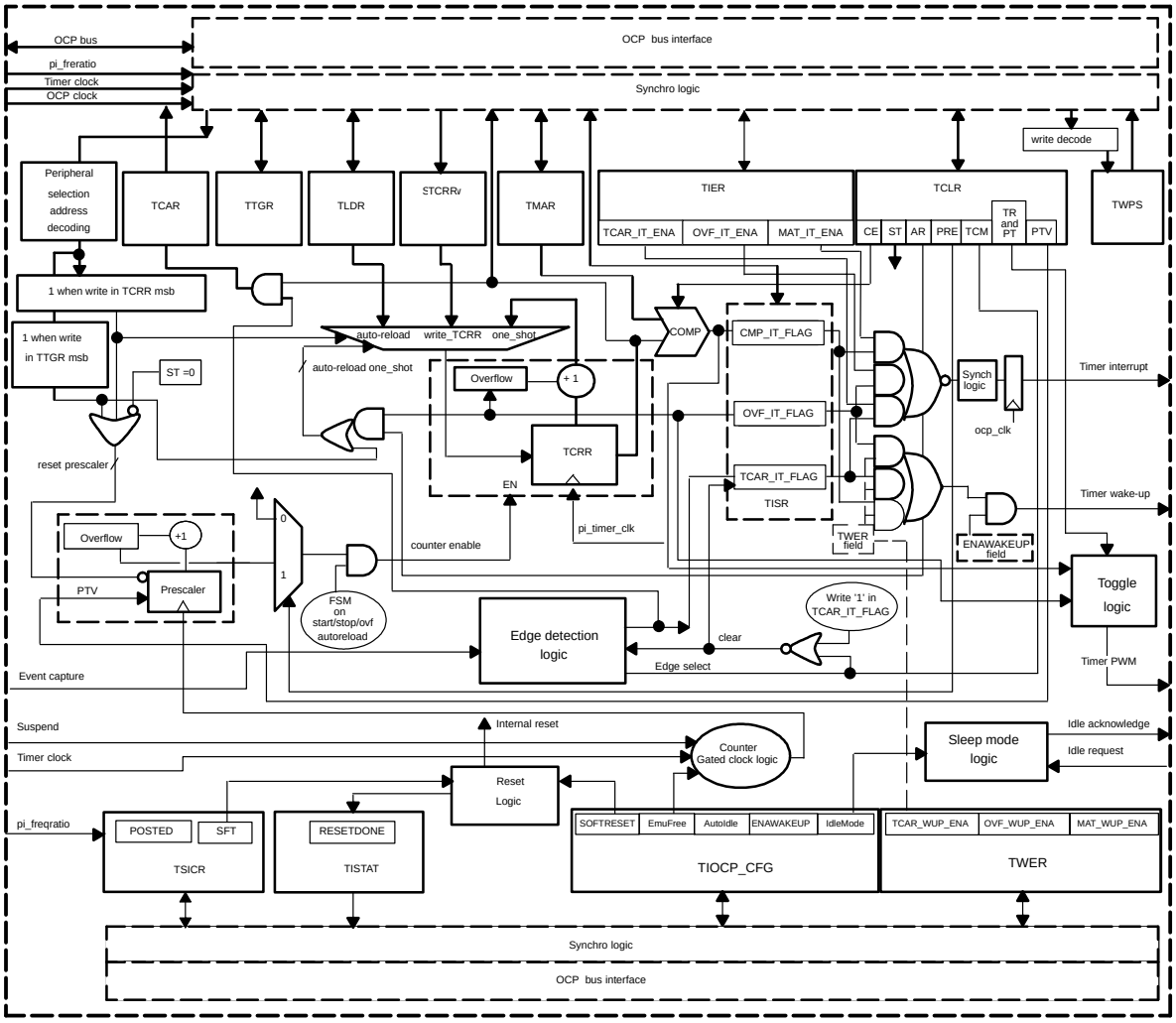
The timer clock has different sources. Based on the timer clock source selection, the user can override the reset value for POSTED to perform nonposted read/write accesses.

Typically, if the timer clock source is 20 MHz and the timer interface clock is 50 MHz, it is recommended to clear the POSTED bit.

21.4 Implementation

Figure 21–6 is an overview of the implementation and interaction of the registers according to the functional description and the input/output pins.

Figure 21–6. Dual-Mode Timer Design Implementation Overview



Chapter 22

Camera Interface

This chapter describes the camera interface implemented in the OMAP730 multimedia processor.

Topic **Page**

22.1 Camera Parallel Interface Overview

22.1.1 Introduction

A 32-bit camera interface connects a camera module to the MPU peripheral bus of the OMAP730 device. The interface handles multiple image formats synchronized on vertical and horizontal synchronization signals. Data transfer between the camera and the interface can be done synchronously or asynchronously. The data is stored in a buffer to be sent over the peripheral bus, using the DMA mode or the CPU mode (bypass mode).

The interface supports 8-bit parallel image data ports and horizontal/vertical signal ports separately (stand-alone synchronous method). The camera interface has a DMA port.

22.1.2 Functional Architecture

The camera architecture consists of four functional blocks:

- Buffer: Stores the data word received from the camera module and transfers it to the MPU peripheral bridge, using the DMA mode or the CPU mode. It contains a 128-word FIFO.

  The 8-bit data received from the camera module is latched and mixed to comply with the 32-bit data format of the MPU TIPB. A 128-bit-deep FIFO is implemented to provide local buffering of the data and to control the DMA request when the camera interface is enabled in DMA mode. The main goals of this mode are:

  - To discharge the CPU of the data transfer
  - To reduce the real-time constraints of the DMA read (FIFO buffering part)
  - To group the xDMA accesses in only one time slot (FIFO block part)

  The user can, however, forward a direct transfer to the CPU in bypass mode by disabling the DMA request line.

- Clock divider: Manages the clock division and handles the external clock generation for synchronous/asynchronous mode gating

- Interrupt generator: Generates an interrupt to indicate the start and end of frame, start and end of image, and FIFO overflow

- TIPB registers: Connect status, control, and data 32-bit registers

22.1.2.1 Camera Data Validation

The incoming byte on CAM_D can be latched on the rising or falling edge of CAM.LCLK generated by the camera itself. The POLCLK bit in the clock control register selects the polarity of CAM.LCLK.

The camera interface must be programmed so that data is always captured opposite the launch edge. For example, if data is latched by the sensor on the rising edge of CAM.LCLK, the interface must be configured to catch the data on the falling edge of CAM.LCLK.

The high level of the vertical synchronous and horizontal synchronous signals indicates that the data is valid on CAM_D. This level is registered in VSTATUS and HSTATUS, which are updated at edge detection of vertical and horizontal synchronous signals.

Figure 22–1 shows the image data transfer, and Figure 22–2 shows the timing chart.

Figure 22–1. Image Data Transfer

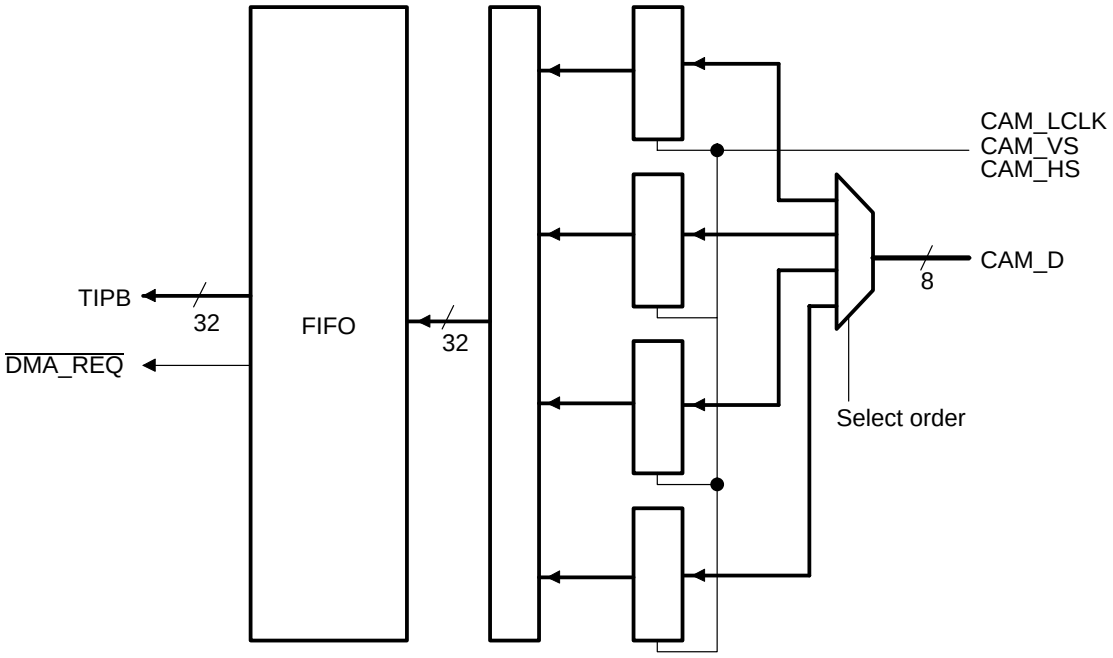
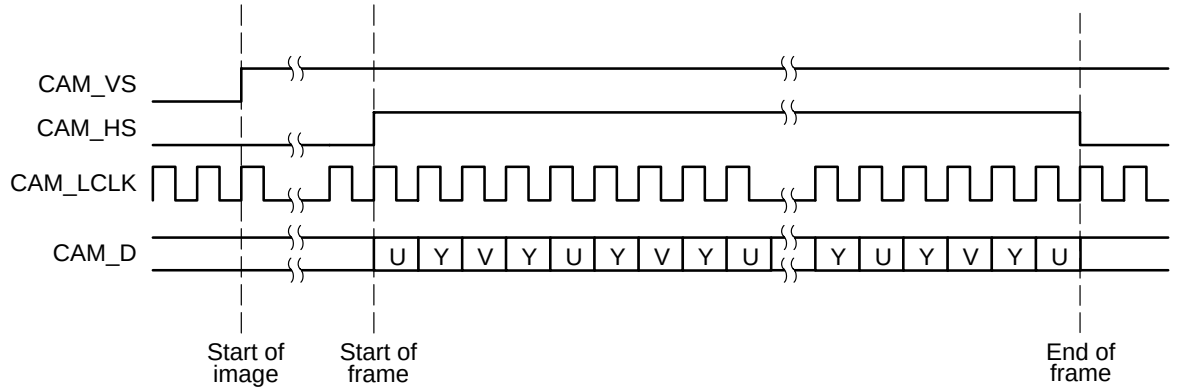


Figure 22–2. Timing Chart of Image Data Transfer (POLCLK = 1)



The clock can be gated during the VSYNC and/or HSYNC blanking periods, but it must be let to run because a process based on LCLK clears all internal resynchronization registers while VSYNC or HSYNC is low, before starting a new frame or new image. This mechanism prevents the FIFO from retaining any remaining data left over from a previous frame, which could corrupt the data of a new frame.

If either CAM_VS or CAM_HS goes inactive before receiving all four bytes, the data in buffers is cleared by the active CAM.LCLK edge and is not written into FIFO.

22.1.2.2 Autostart

Autostart is a protective function that prevents a start of capture during an image transfer. Autostart is launched after enabling the LCLK and waits for the next inactive level of CAM_VS to enable the data capture so that the transfer starts at the beginning of the image.

Note:

If a reset FIFO occurs (see Section 22.1.2.3) while the interface is latching data, the capture is automatically disabled and the autostart function is enabled.

22.1.2.3 Reset FIFO

An active-high reset FIFO is implemented at the RAZ_FIFO bit (18) of the camera mode register. This feature clears any remaining data in the FIFO before starting a new transfer. It also resets all status and control signals around the FIFO, such as the read and write pointers, the FIFO full interrupt, the FIFO peak counter, and the 32-bit resynchronization registers.

22.1.2.4 Set of Order

Each 4 bytes received from the camera must be packed and can be swapped to follow the order YUV specified in the camera mode register by ORDERCAMD. Figure 22–3 and Figure 22–4 show the camera data order not swapped and swapped, respectively.

Figure 22–3. Order of Camera Data on TIPB (Not Swapped)

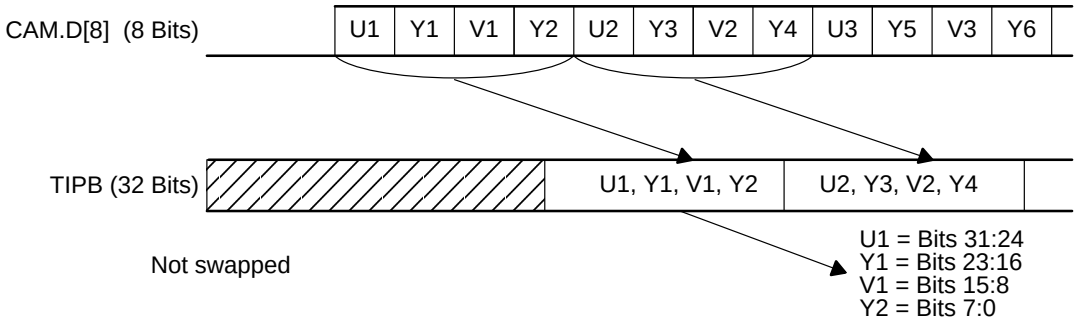
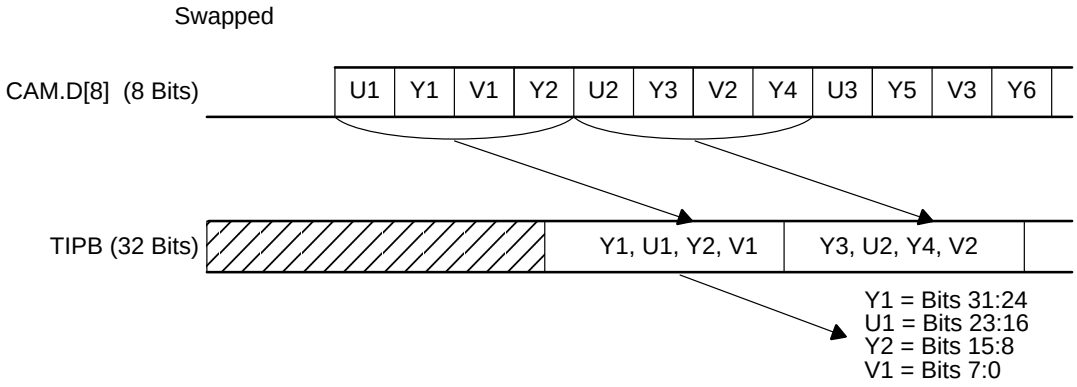


Figure 22–4. Order of Camera Data on TIPB (Swapped)



22.1.2.5 FIFO Buffer (128 x 32)

A write access is applied to the FIFO for each 32-bit word received. When the write FIFO counter reaches the trigger level, an interrupt request can be generated. The trigger level is programmable.

In DMA mode, the threshold can be programmed between 1 and 128, but the DMA must be set up to read the threshold amount out of FIFO per the DMA request issued by the camera interface. Otherwise, the locking mechanism is never rearmed, and it prevents DMA requests from being issued after every read.

A pulse on the DMA request (see Figure 22–5 and Figure 22–6) occurs when the number of words in the FIFO is above the threshold. The DMA request occurs if the number of remaining words is above the threshold and the system DMA has completed the transfer (number of words read by the DMA = threshold).

The camera FIFO continues to fill (up to its maximum 128 values) when an interrupt or DMA request has been generated but not yet responded to. When a data value is read from the camera FIFO, another IRQ or DMA request is immediately generated, as long as the amount of data present in the FIFO is above the trigger level.

Figure 22–5. DMA Request

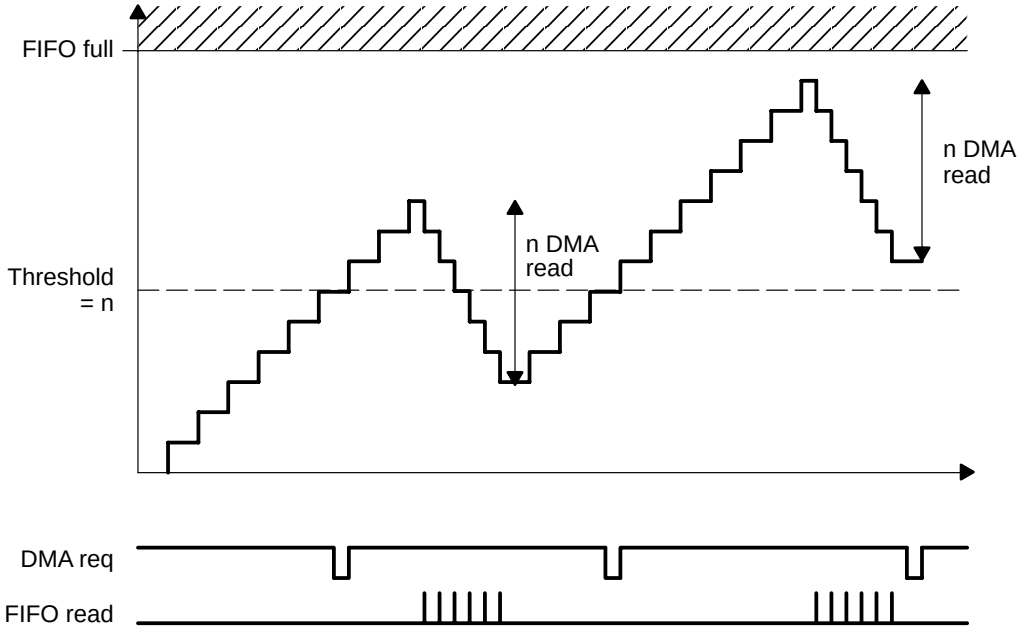
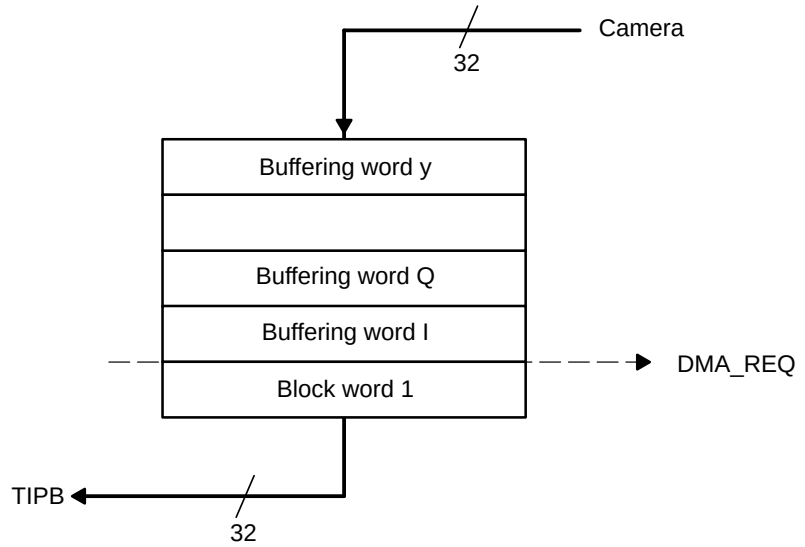


Figure 22–6. FIFO Buffer Parts



22.1.2.6 Clock Divider

The clock divider takes the internal 12-MHz clock source or the 48-MHz source from APLL to generate the external clock CAM.EXCLK. The division factor is programmable in the clock control register through FOSCMOD (see Table 22–1).

Table 22–1. Clock Ratios

Ratio	CAM.EXCLK	
	From 12 MHz	From 48 MHz
1	12 MHz	-
1/2	6 MHz	24 MHz
1/5	-	9.6 MHz
1/6	-	8 MHz

A request is automatically generated to wake up the APLL when 48 MHz is needed. The switch is performed when the 48-MHz signal is stable.

It is assumed that the switch is made when CAM.EXCLK is disabled (glitch protection).

The clock divider also allows disabling the external clock by setting the CAMEXCLK_EN bit.

22.1.2.7 Interrupt Generator

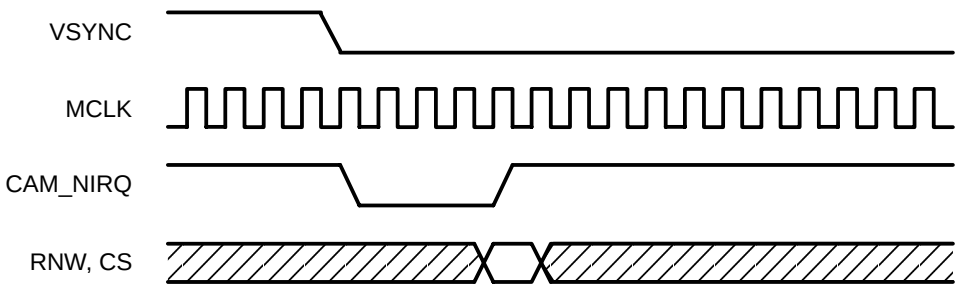
The interrupt generator handles six cases of interrupts:

- Data transfer interrupt. One IRQ is generated per word received.
- HSYNC rising edge (start of frame)
- HSYNC falling edge (end of frame). See Figure 22–7.
- VSYNC rising edge (start of image)
- VSYNC falling edge (end of image)
- FIFO overflow

Each case is registered by activating (high) one of the six interrupt register bits to indicate the origin of the interrupt. However, the interrupt mask register can disable the source of the interruption.

Only one line of interrupt is used to ask for a read of the interrupt register. When the read occurs, the register is automatically reset and the interrupt signal is released.

Figure 22–7. IRQ Generated on VSYNC Falling Edge



22.1.2.8 DMA Procedure

A typical procedure to perform the data transfer by DMA is as follows:

1) Rising edge of VSYNC sends an interrupt to ARM926EJS to alert the system DMA that a start of image has occurred. The system DMA is programmed to move one complete image of data and then give an interrupt when complete.

2) High level of HSYNC and proper clock edge start the first data transfer from the camera to the OMAP730 camera interface. After the first two pixels of data are received (8 bits x 4 transfers = 32 bits), a DMA request is made. The system DMA moves the 32-bit data to a predefined SDRAM location.

3) The camera, the OMAP730 device camera interface, and the system DMA continue the transfer of data. That is, 352/2 * 288 = 50688 transfers for a camera interface image format. After the full image is transferred, the DMA sends an interrupt to the ARM926EJS to signal that the end of frame occurred.

The camera interface and system DMA can be configured in many ways to move the data, but in this sequence the interrupt load on the ARM926EJS is minimal.

22.1.2.9 TIPB Registers

The camera interface contains seven registers for communication between the TIPB and camera module. These registers mainly control clock generation, interrupt request, and status (see Section 22.1.2.10).

The address of each register is equal to START_ADDRESS + OFFSET_ADDRESS when START_ADDRESS is defined by the chip-select allocated to the camera in the TIPB address space.

Table 22–2 shows the default configuration at reset.

Table 22–2. Default Configuration at Reset

Item	Function
ORDERCAMD	Not swapped
MASK	Interrupts on VSYNC and HSYNC disabled
FOSCMOD	Division rate for CAM.EXCLK = 1 (12 MHz)
POLCLK	Data latched on rising edge of CAM.LCLK
CAMEXCLK_EN	CAM.EXCLK disabled
MCLK_EN	Internal clock disabled
APLL_EN	APLL clock source disabled
THRESHOLD	Trigger level = 1 word

22.1.2.10 Camera Interface Registers

Table 22–3 lists the camera interface registers. Table 22–4 through Table 22–10 describe the register bits.

Table 22–3. Camera Interface Registers

Register	Description	R/W	Size	Offset Address
CTRLCLOCK	Clock control	R/W	32 bits	0x00
IT_STATUS	Interrupt source status	R	32 bits	0x04
MODE	Camera interface mode configuration	R/W	32 bits	0x08
STATUS	Status	R	32 bits	0x0C
CAMDATA	Image data	R/W	32 bits	0x10
GPIO	Camera interface GPIO (general-purpose input/ output)	R	32 bits	0x14
PEAK_COUNTER	FIFO peak counter	R/W	32 bits	0x18

Table 22–4. Clock Control Register (CTRLCLOCK)

Bit	Name	Function	R/W	Reset Value
31:8	RESERVED	Reserved (unknown value after reset)	R/W	0xX
7	LCLK_EN	0: Disables 1: Enables incoming CAM.LCLK	R/W	0x0
6	APLL_EN	0: Disables 1: Enables APLL source (48 MHz)	R/W	0x0
5	MCLK_EN	0: Disables 1: Enables internal clock of interface	R/W	0x0
4	CAMEXCLK_EN	0: Disables 1: Enables CAM.EXCLK	R/W	0x0
3	POLCLK	Sets polarity of CAM.LCLK: 0: Data latched on rising edge 1: Data latched on falling edge	R/W	0x0
2:0	FOSCMOD	Sets the frequency of the CAM.EXCLK clock: 000: 12 MHz 010: 6 MHz 100: 9.6 MHz (48 MHz/5) 101: 24 MHz (48 MHz/2) 110: 8 MHz (48 MHz/6)	R/W	0x00

The MCLK_EN bit in the clock control register gates the 12-MHz master clock of the camera interface to disable the clock when switching between two clock domains or to save power consumption when the camera module is not used. To clear PEAK_COUNTER, read all data in FIFO and then write PEAK_COUNTER with 0.

Table 22–5. Interrupt Source Status Register (IT_STATUS)

Bit	Name	Function	R/W	Reset Value
31:6	RESERVED	Reserved bits	R	0xX
5	DATA_TRANSFER	Data transfer status. Set to 1 when trigger is reached. Reset by reading IT_STATUS if no event in the meantime.	R	0x0
4	FIFO_FULL	Detect rising edge on FIFO full flag. Reset by reading IT_STATUS if no event in the meantime.	R	0x0
3	H_DOWN	Flag for horizontal synchronous falling edge occurred. Reset by reading IT_STATUS if no event in the meantime.	R	0x0
2	H_UP	Flag for horizontal synchronous rising edge occurred. Reset by reading IT_STATUS if no event in the meantime.	R	0x0
1	V_DOWN	Flag for vertical synchronous falling edge occurred. Reset by reading IT_STATUS if no event in the meantime.	R	0x0
0	V_UP	Flag for vertical synchronous rising edge occurred. Reset by reading IT_STATUS if no event in the meantime.	R	0x0

Table 22–6. Camera Interface Mode Configuration Register (MODE)

Bit	Name	Function	R/W	Reset Value
31:19	RESERVED	Reserved bits	R/W	0xX
18	RAZ_FIFO	When 1: Clears data in the FIFO; reinitializes read and write pointers; clears FIFO full interrupt, FIFO peak counter; and resynchronizes.	R/W	0x0
17	EN_FIFO_FULL	0: Disables 1: Enables interrupt on FIFO_FULL	R/W	0x0
16	EN_NIRQ	0: Disables 1: Enables data transfer interrupt (bypass DMA mode)	R/W	0x0
15:9	THRESHOLD	Programmable DMA request trigger value. DMA request is made when FIFO counter is equal to the threshold value. Currently, set this field to 1 in DMA mode.	R/W	0x0000001
8	DMA	Enables DMA mode when 1	R/W	0x0
7	EN_H_DOWN	Enables interrupt on HSYNC falling edge. Active when 1.	R/W	0x0
6	EN_H_UP	Enables interrupt on HSYNC rising edge. Active when 1.	R/W	0x0
5	EN_V_DOWN	Enables interrupt on VSYNC falling edge. Active when 1.	R/W	0x0

Table 22–6. Camera Interface Mode Configuration Register (MODE) (Continued)

Bit	Name	Function	R/W	Reset Value
4	EN_V_UP	Enables interrupt on VSYNC rising edge. Active when 1.	R/W	0x0
3	ORDERCAMD	Sets order of 2 consecutive bytes received from camera (YUV format). Not swapped when 0, swapped when 1.	R/W	0x0
2:1	IMGSIZE	Sets image size: 00: CIF 01: QCIF 10: VGA 11: QVGA Currently, these bits have no effect on the operation of the camera interface.	R/W	0x00
0	CAMOSC	0: Set synchronous mode 1: Set asynchronous mode Currently, this has no effect on the camera interface.	R/W	0x0

Table 22–7. Status Register (STATUS)

Bit	Name	Function	R/W	Reset Value
31:2	RESERVED	Reserved bits	R	0xX
1	HSTATUS	CAM_HS status (edge detection)	R	0x0
0	VSTATUS	CAM_VS status (edge detection)	R	0x0

Table 22–8. Camera Interface GPIO Register (GPIO)

Bit	Name	Function	R/W	Reset Value
31:1	RESERVED	Reserved bits	R/W	0xX
0	CAM_RST	Reset for camera module	R/W	0x0

Table 22–9. Image Data Register (CAMDATA)

Bit	Name	Function	R/W	Reset Value
31:0	CAMDATA	Image data from FIFO	R	0x0

Table 22–10. FIFO Peak Counter Register (PEAK_COUNTER)

Bit	Name	Function	R/W	Reset Value
31:7	RESERVED	Reserved	R/W	Unknown
6:0	PEAK_COUNTER	Maximum number of words written to FIFO during the transfer since the last clear to zero	R/W	0x0000000

22.1.3 Clock Switching Procedures

22.1.3.1 CAM.EXCLK Switch Protocol

The CAM.EXCLK switch protocol is required for any change of the CAM.EXCLK frequency value to first disable both the12-MHz clock source and the APLL clock source in clock control registers:

1) Disable MCLK and APLL_CLK (MCLK_EN = 0, DPLL_EN = 0, FOSCMOD = FOSCMOD).

2) Change CAM.EXCLK value (FOSCMOD = new FOSCMOD).

3) Enable MCLK and APLL_CLK (MCLK_EN = 1, DPLL_EN = 1, FOSCMOD = FOSCMOD).

22.1.3.2 CAM.LCLK Switch Protocol

Bit 3 of the clock control register (POLCLK) sets the polarity of CAM.LCLK. CAM.LCLK must be disabled before selecting the rising or falling edge.

1) Disable CAM.LCLK (LCLK_EN = 0).
2) Set the new polarity (POLCLK = 1 or 0).
3) Enable CAM.LCLK (LCLK_EN = 1).

22.2 Camera Interface Bandwidth

Table 22–11 lists the camera interface bandwidth.

Table 22–11. Camera Interface Bandwidth

Image Format	Lines	Pixels	Pixel Clock Frequency	Frame/s
CIF	288	352	24 MHz	78.91414
QCIF	144	176	24 MHz	315.6566
VGA	640	480	24 MHz	26.04167
SVGA	800	600	24 MHz	16.66667

Chapter 23

Real-Time Clock

This chapter discusses the real-time clock of the OMAP730 multimedia processor.

Topic **Page**

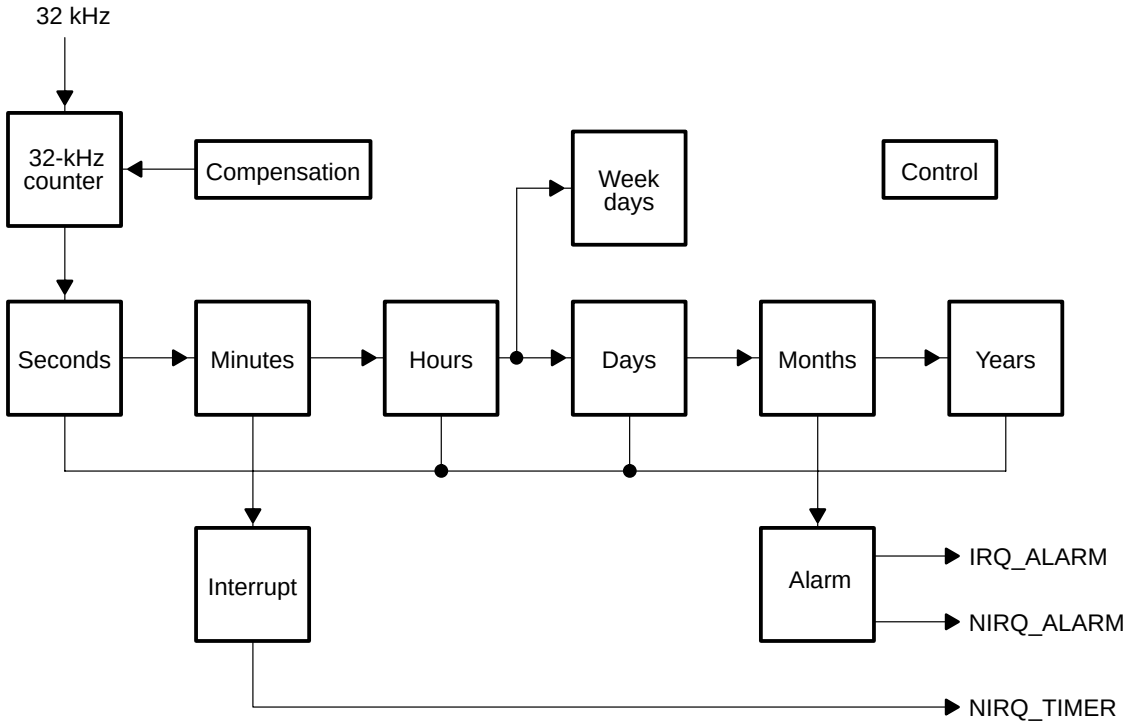
23.1 Real-Time Clock

The real-time clock (RTC) block is an embedded real-time clock module directly accessible from the TIPB bus interface. The basic functions of RTC block are:

- Time information (seconds/minutes/hours) directly in binary coded decimal (BCD) code
- Calendar information (day/month/year/day of the week) directly in BCD code up to the year 2099
- Interrupt generation, periodically (1s/1m/1h/1d period) or at a precise time of the day (alarm function)
- 30-s time correction
- Oscillator frequency calibration

Figure 23–1 shows the real-time clock block.

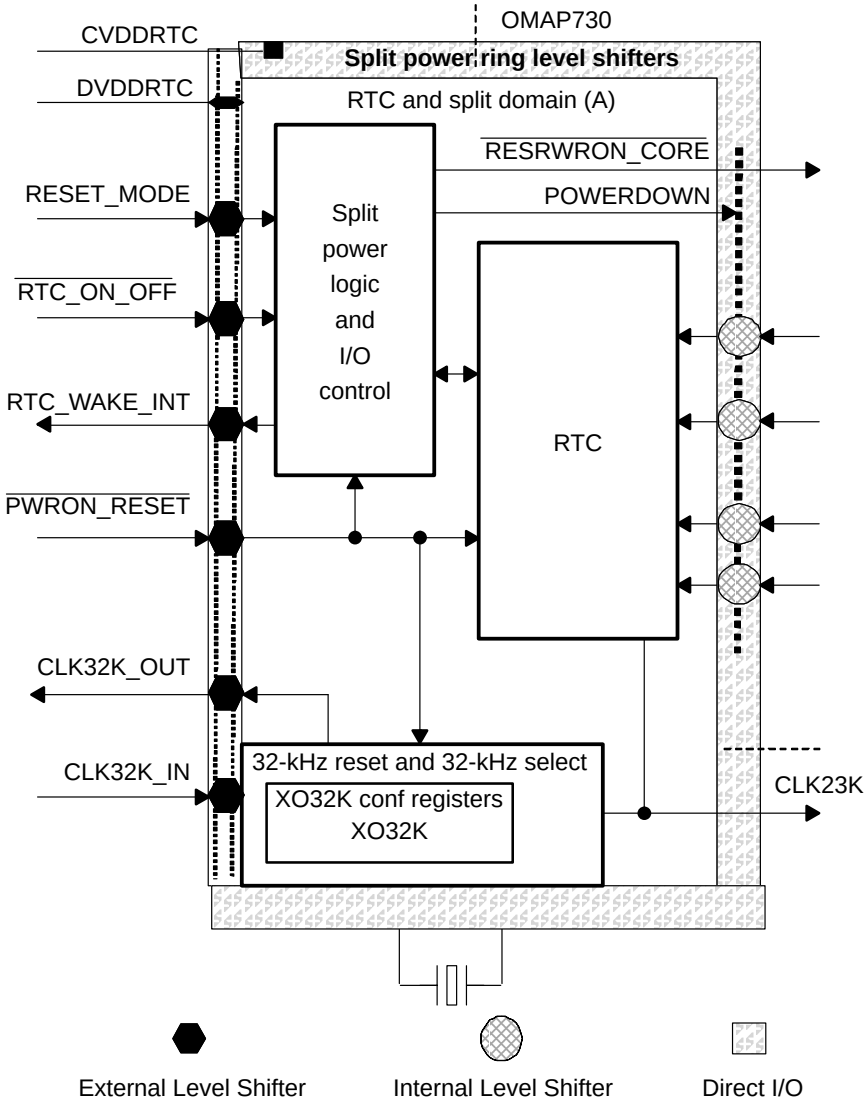
Figure 23–1. Real-Time Clock



23.1.1 Split Power Overview

To achieve minimum consumption in the OFF state in device equipment, some active logic elements are supplied. Those elements are the real-time clock (RTC) and the 32-kHz oscillator (OSC32K) in the digital baseband (DBB), and the power-on reset (POR) and the dedicated regulator in the analog baseband (ABB). This approach is possible when using split power, which splits the core power domain into two subdomains powered with different voltage supplies. Internal level shifters handle the separation between the core and the active domain.

Figure 23–2. Split Power System in OMAP730



23.1.2 Internal Level Shifters

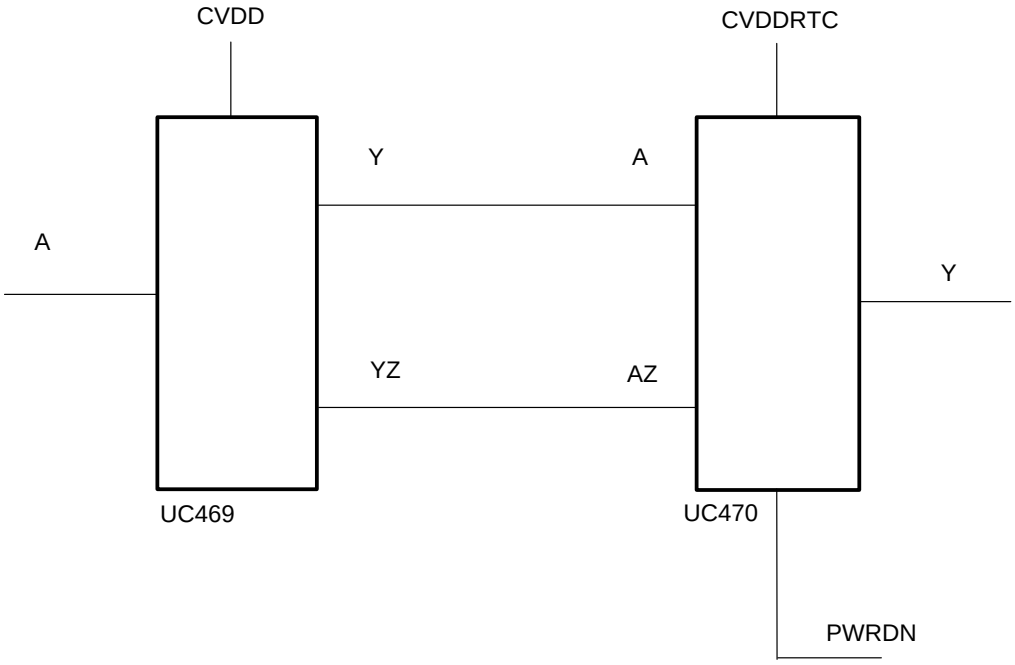
Internal level shifters are library-standard macrocells that interface two core domains powered with two different voltage supplies.

The cell can be seen as a means to isolate one power domain from an other. In the case where one domain is shut down, the level shifters ensure a known state at the boundary of the two domains.

The level shifters are divided into two parts:

- UC469: Powered by the primary power supply. Because of the input signal A, it creates Y and YZ, which are A buffered and A inverted, respectively.
- UC470: Powered by the secondary power supply. Because of the differential signals, A and AZ (given by UC469), it creates the signal Y that is equivalent to the input of UC469 but in the second core supply domain. This cell has a PWRDN signal so that when the power supply of UC469 does not exist (input signals A and AZ are ambiguous), this cell does not have any through-current and the output is set to 0.

Figure 23–3. Internal Level Shifter



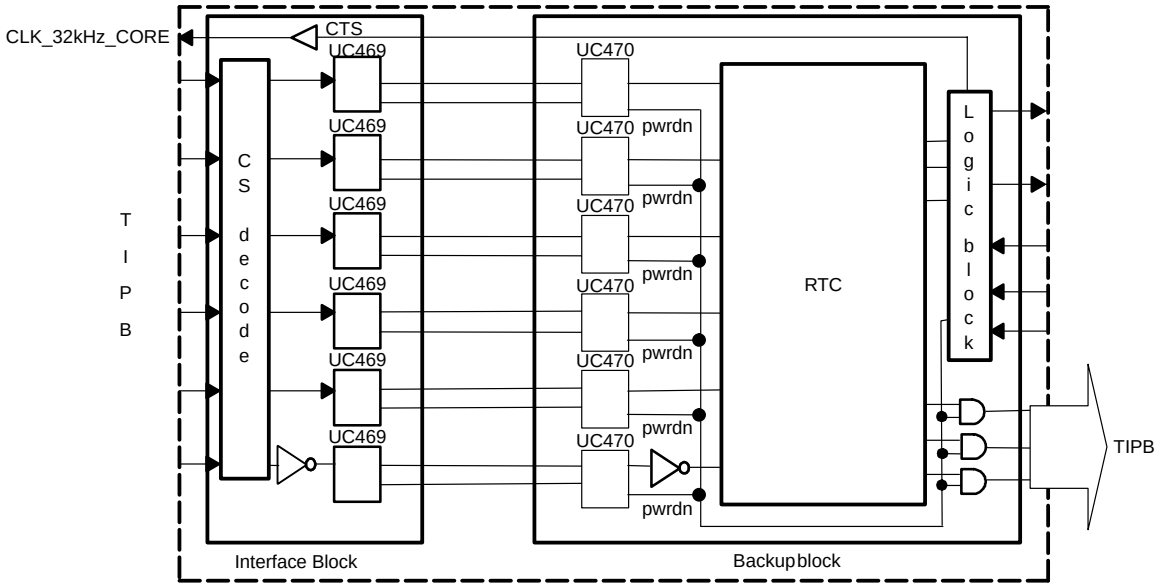
PWRDN = 0 => Y=A

PWRDN =1 => Y= 0

23.1.3 Split Power Module

The split power module contains two blocks: the interface block and the back-up block. The interface block, between the core and the backup elements, is supplied by the core power supply CVDD. The backup block contains the RTC and some logic.

Figure 23–4. Split Power Block Diagram



23.1.3.1 Interface Block

This block separates the backup elements and the ASIC core. It also contains the UC469 and some logic. The logic allows masking of the TIPB bus signals to avoid consumption of internal level shifters, except in RTC accesses.

23.1.3.2 Backup Block

This block contains the elements kept in OFF mode in the DBB, the RTC, the 32-kHz clock, the UC470, and the logic to force the DBB to OFF mode. The module input/outputs are RTC_ON_$\overline{\text{OFF}}$, $\overline{\text{PWRON\_RESET}}$, RTC_WAKE_INT, RESET_MODE, and POWERDOWN.

23.1.4 Output Control

To keep them from supplying the ASIC core, the split-power block outputs are forced to logical zero.

23.1.4.1 Backup Signal Management

In OFF mode, only the split power module is supplied. The $\overline{\text{ON\_OFF}}$, RTC_WAKE_INT, and $\overline{\text{PWRON\_RESET}}$ signals, which control the activity management of the DBB, are also active.

23.1.5 On-Chip Reset Generation

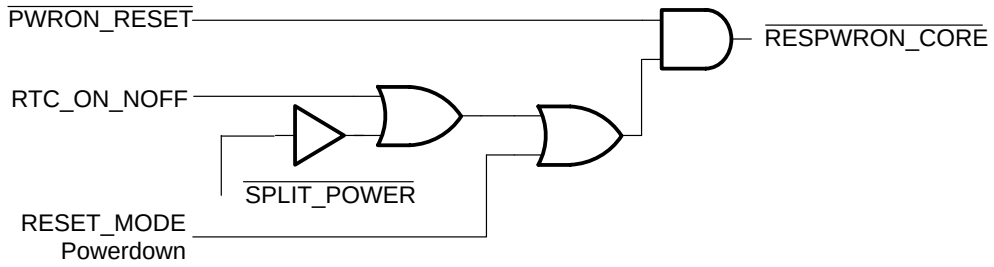
- If $\overline{\text{SPLIT\_POWER}}$ or RESET_MODE = 1:

  $\overline{\text{PWRON_RESET}}$_CORE = $\overline{\text{PWRON\_RESET}}$

- If RESET_MODE = 0 and $\overline{\text{SPLIT\_POWER}}$ = 1:

  $\overline{\text{PWRON_RESET}}$_CORE = 0 if $\overline{\text{PWRON\_RESET}}$ = 0 OR $\overline{\text{ON\_OFF}}$ = 0

Figure 23–5. ASIC Reset Scheme



23.1.5.1 Resets for OMAP730 Device With Split Power Feature Enabled

The RTC module and its internal real time counter are reset by $\overline{\text{PWRON\_RESET}}$ during power up. OMAP730 ASIC gates are reset by $\overline{\text{PWRON_RESET}}$_CORE.

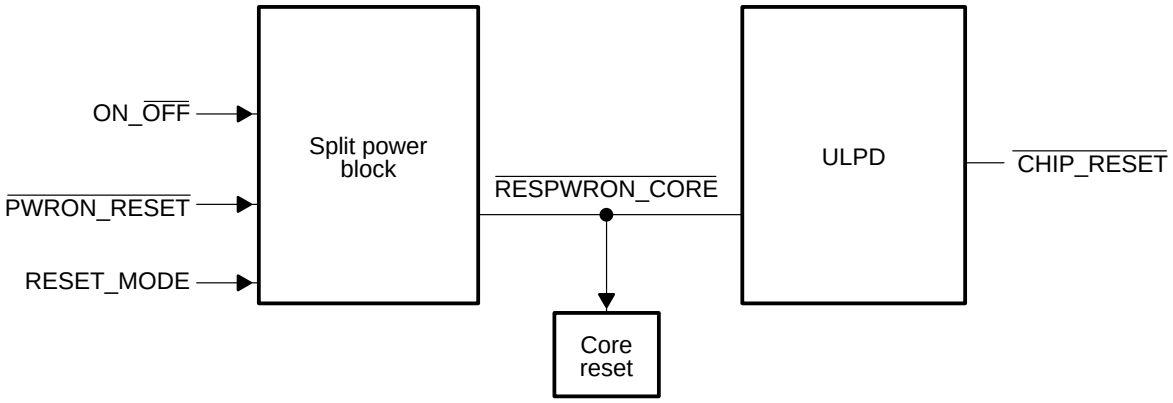
Subsequent resets are asserted with ON_$\overline{\text{OFF}}$. The RTC module is not reset by the assertion of ON_$\overline{\text{OFF}}$. While ON_$\overline{\text{OFF}}$ is asserted low, the system powers down OMAP730 ASIC gates.

23.1.5.2 Resets for OMAP730 Device With Split Power Feature Not Used

For OMAP730 devices that are forced during reset in reset mode 1, $\overline{\text{PWRON_RESET}}$_CORE is logically equal to $\overline{\text{PWRON\_RESET}}$ and only to $\overline{\text{PWRON\_RESET}}$.

For OMAP730 devices that are forced during reset in reset mode 0 (OMAP1510 legacy) and no split power**,** $\overline{\text{PWRON_RESET}}$_CORE is logically equal to $\overline{\text{PWRON\_RESET}}$ and only to $\overline{\text{PWRON\_RESET}}$.

Figure 23–6. $\overline{\text{PWRON\_RESET}}$ Connection



RTC_WAKE_INT

The RTC_WAKE_INT pin now collects the interrupt sources used to awaken the ULPD from deep sleep. It is gated with $\overline{\text{SPLIT\_POWER}}$ and the RTC alarm. In OFF mode the IRQ_SET is set to 0, and only the IRQ_ALARM_EXT can generate an interrupt on the pin RTC_WAKE_INT.

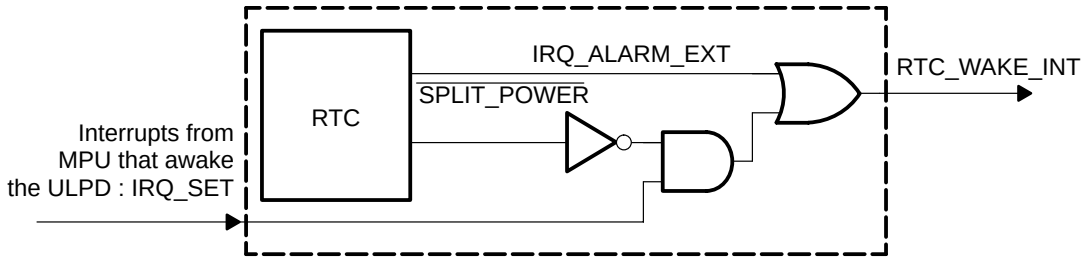
With a device in ON state, both IRQ_ALARM_EXT and IRQ_SET can generate an RTC_WAKE_INT.

The $\overline{\text{SPLIT\_POWER}}$ signal generates RTC_WAKE_INT to avoid an RTC_WAKE_INT in ON mode for the ABB.

Figure 23–7 shows the generation of the RTC_WAKE_INT.

In the OMAP730 core, IRQ_SET is not used and is set to 0. RTC_WAKE_INT is used only for the ABB.

Figure 23–7. RTC_WAKE_INT Generation

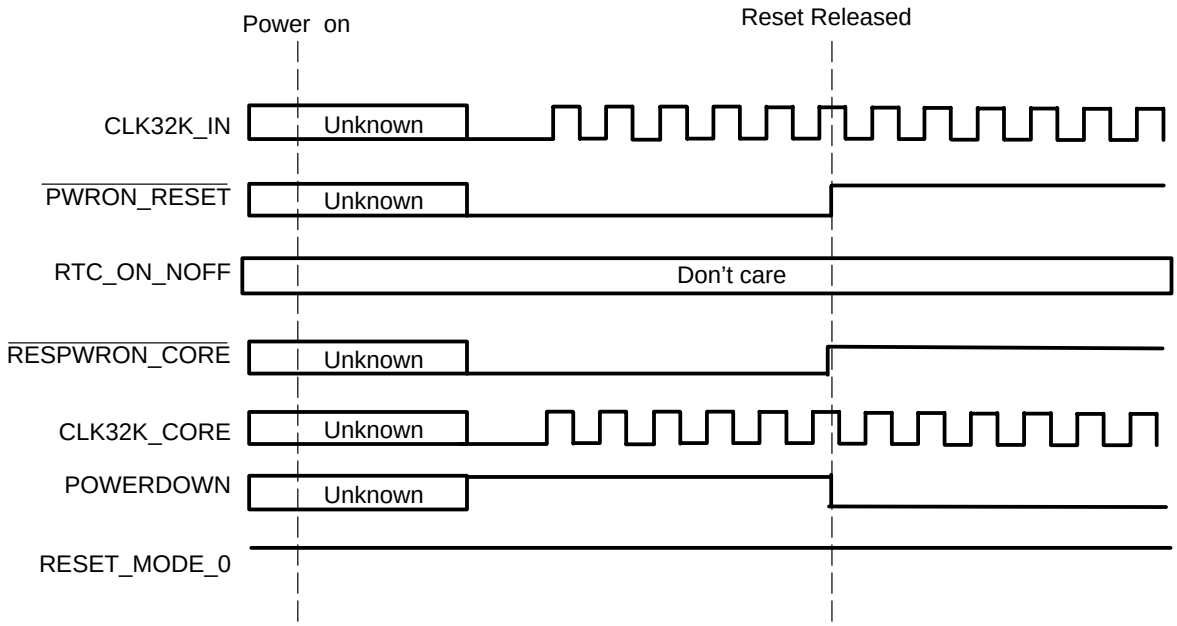


23.1.6 Using Split Power

23.1.6.1 ABB Functions Incompatible With Split Power

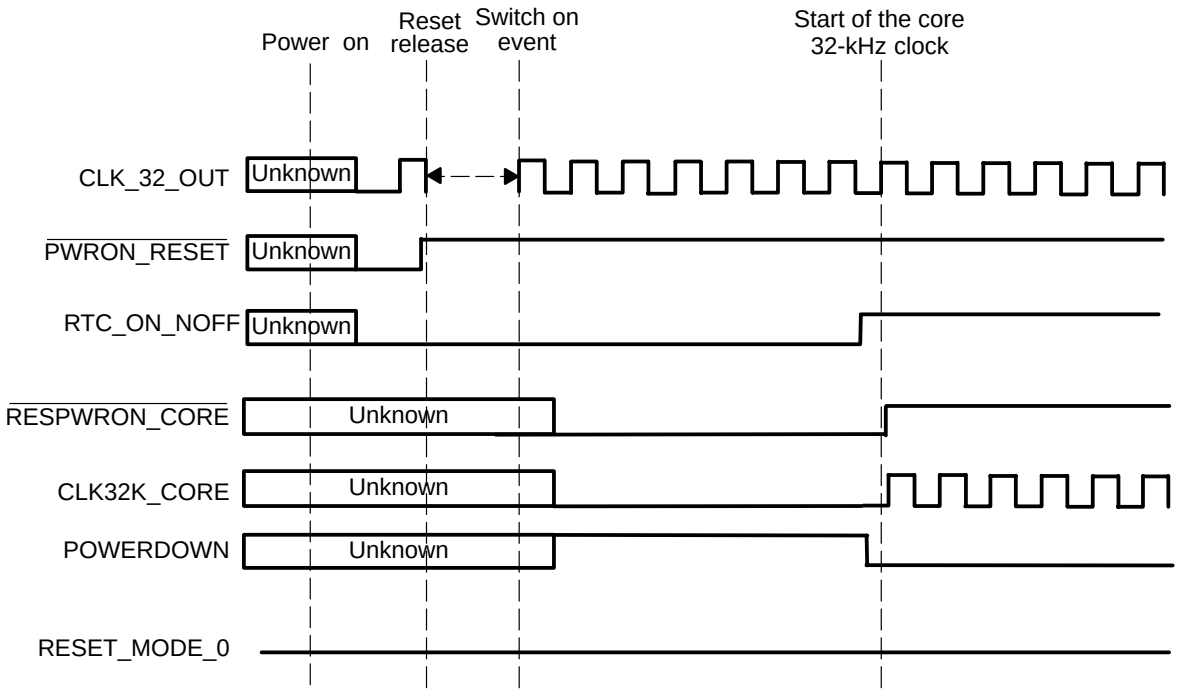
Do not use split power. The power cannot be cut in the core.

Figure 23–8. OMAP730 in RESET_MODE = 1 or SPLIT_POWER = 1



23.1.6.2 ABB Functions Compatible With Split Power

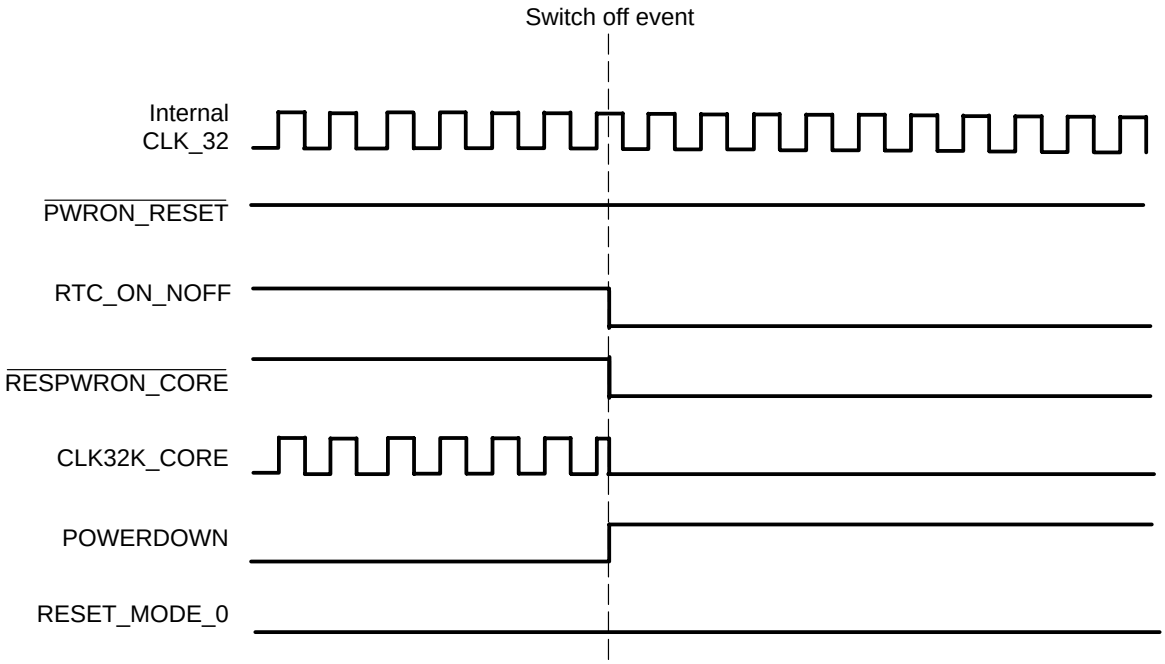
Figure 23–9. Startup With SPLIT_POWER = 1 for OMAP730 RESET_MODE = 0



On the plug of the backup battery, the CVDDRTC LDO regulator is active and supplies the RTC and the oscillator. The RTC is isolated from the core by the POWERDOWN signal. The reset signal for the backup module stays at 0 until the backup regulator reaches its value. The 32-kHz clock is sent to the ABB circuit by the CLK32K_OUT pin. On a switch-on event the ASIC LDO regulators are enabled and supply the DBB core. The core domain is reset by $\overline{\text{PWRON_RESET}}$_CORE until $\overline{\text{ON\_OFF}}$ becomes high; the isolation mode also becomes inactive. The ULPD state machine can start.

23.1.6.3 ON to OFF Description

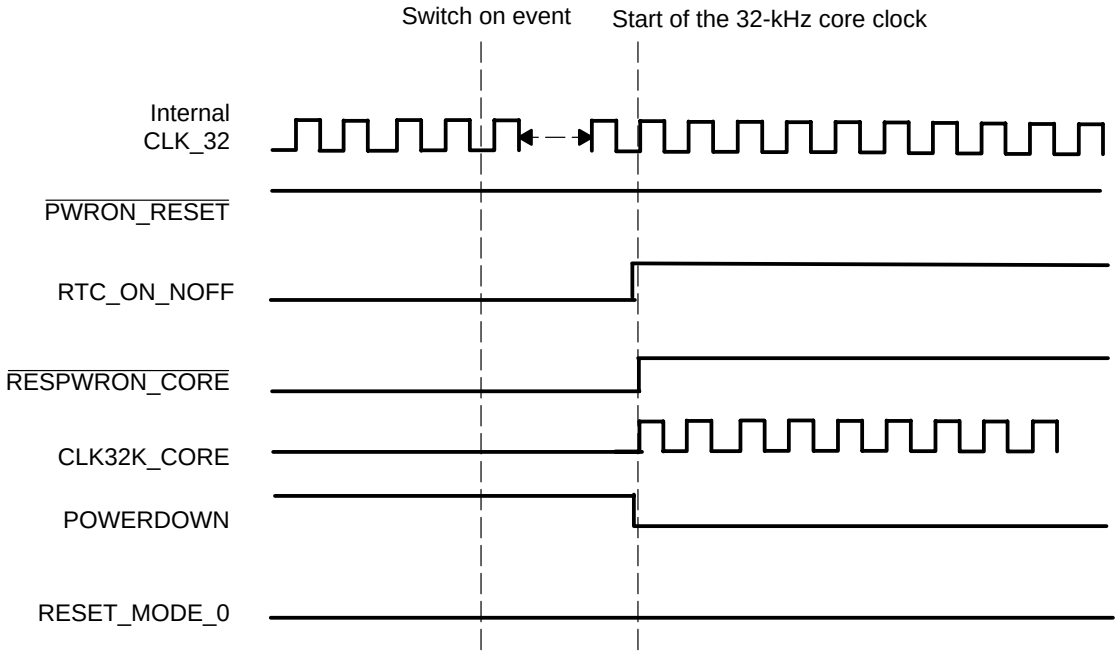
Figure 23–10. ON to OFF With SPLIT POWER = 1 for OMAP730 RESET_MODE = 0



On a switch-off event, ON_$\overline{\text{OFF}}$ is set to 0. The 32-kHz clock is stopped, and $\overline{\text{PWRON_RESET}}$_CORE is set to 0 to indicate that the device goes into OFF state. $\overline{\text{PWRON_RESET}}$_CORE becomes active, and the DBB core is reset. POWERDOWN becomes active and the backup module is isolated. The ABB circuit disables the CORE LDO regulators, and thus the DBB core is not supplied.

23.1.6.4 OFF to ON Description

Figure 23–11. OFF to ON With SPLIT_POWER = 1 for OMAP730 RESET_MODE = 0



On a switch-on event, the ASIC LDO regulators are enabled. The DBB core is also supplied and reset by PWRON_RESET_CORE until ON_OFF is set to 1. Then, the isolation mode is inactive, the ULPD state machine receives PWRON_RESET_CORE, and the clock starts its state machine.

23.1.7 Interrupt Management

RTC can generate three interrupts:

- A timer interrupt (IRQ_TIMER)
- An alarm interrupt (IRQ_ALARM_CHIP)
- An alarm interrupt external (IRQ_ALARM_EXT or IRQ_ALARM_CHIP inverted)

23.1.7.1 Timer Interrupt

IRQ_TIMER interrupt can be generated periodically: every second, every minute, every hour, or everyday (RTC_INTERRUPTS_REG[1:0]).

The IT_TIMER bit of the interrupt register enables this interrupt.

It is a negative edge-sensitive interrupt (low-level pulse duration = 15 µs).

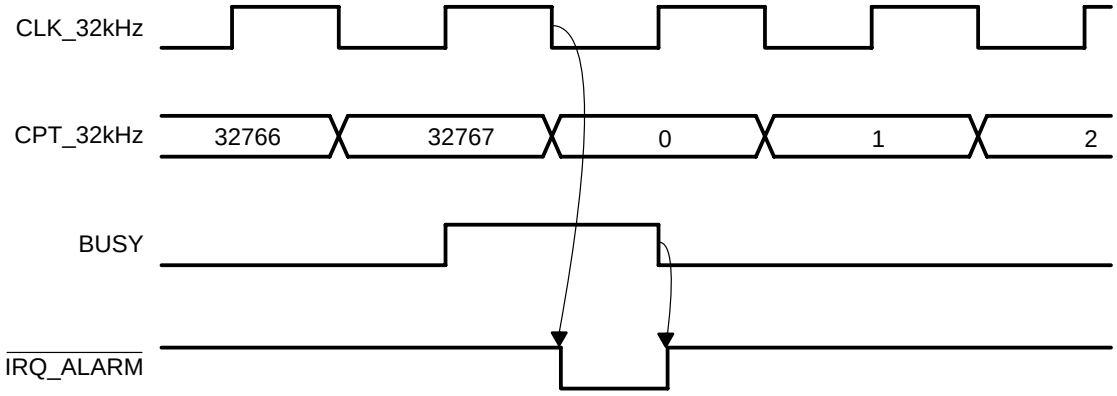
The RTC_STATUS_REG[5:2] are only updated at each new interrupt and show the events that have occurred (see Table 23–1).

Table 23–1. Timer Interrupt Events

RTC_INTERRUPTS_REG[1:0]	11	10	01	00
RTC_STATUS_REG[5] (DAY)	1	0/1†	0/1†	0/1†
RTC_STATUS_REG[4] (HOUR)	1	1	0/1†	0/1†
RTC_STATUS_REG[3] (MIN)	1	1	1	0/1†
RTC_STATUS_REG[2] (SEC)	1	1	1	1

† When this event is concurrent with programmed periodical period.

Figure 23–12. Periodic Interrupt



23.1.7.2 Alarm Interrupt

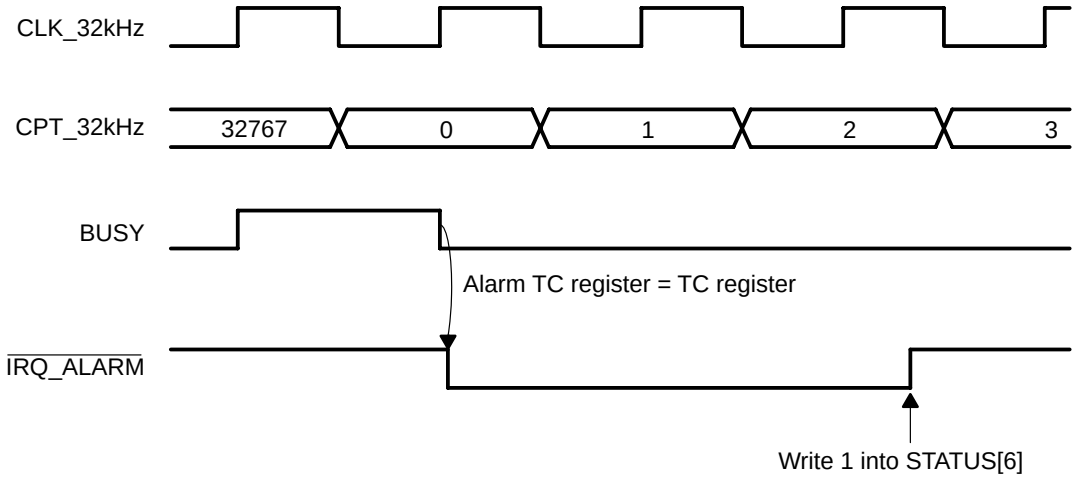
IRQ_ALARM_CHIP interrupt can be generated when the time set in the time and calendar ALARM registers is identical in the time and calendar registers.

This interrupt is then generated if the IT_ALARM bit of the interrupt register is set.

This interrupt is low-level sensitive. RTC_STATUS_REG[6] indicates that IRQ_ALARM_CHIP occurred.

This interrupt is disabled by writing 1 into the RTC_STATUS_REG[6].

Figure 23–13. Alarm Interrupt



23.1.8 Oscillator Drift Compensation

To compensate for any inaccuracy of the 32-kHz oscillator, the MPU can perform a calibration of the oscillator frequency, calculate the drift compensation versus one-hour period, and load the compensation registers with the drift compensation value.

Autocompensation is enabled by the AUTO_COMP_EN bit in the RTC_CTRL register.

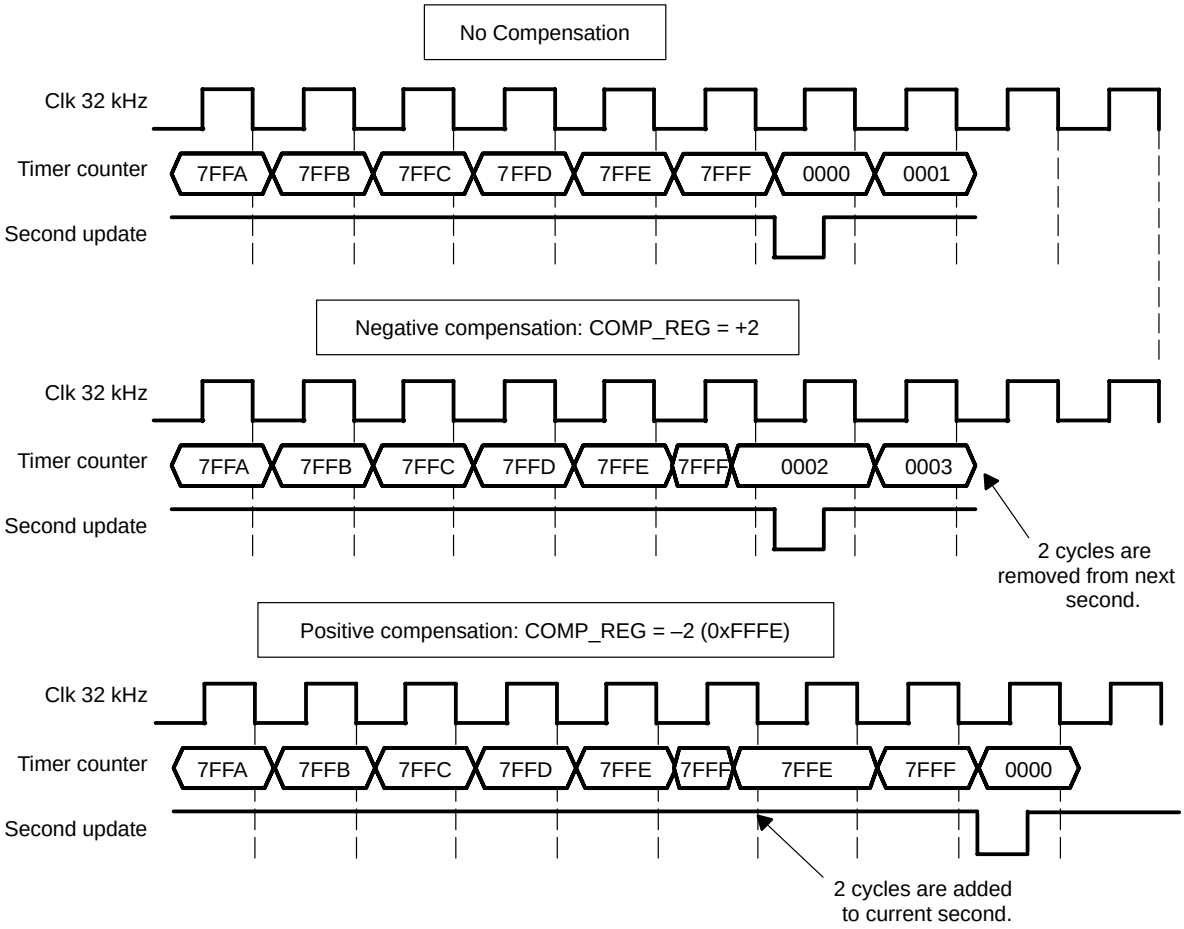
If the COMP_REG value is positive, compensation occurs *after* the second change event. COMP_REG cycles are removed from the next second.

If the COMP_REG value is negative, compensation occurs *before* the second change event. The COMP_REG cycles are added to the current second.

This compensation enables a 32-kHz period accuracy each hour.

The waveform in Figure 23–14 summarizes the positive and negative compensation effect.

Figure 23–14. Oscillator Drift Compensation



23.1.9 Split Power Compatibility

The RTC and the 32-kHz oscillator are the only elements that must be active in the device in OFF state. Therefore, the RTC has been modified to use split power. One register has been modified (RTC_CTRL_REG), and one register has been added (RTC_RES_PROG_REG).

In the RTC_CTRL_REG, the $\overline{\text{SPLIT\_POWER}}$ bit has been added so the user can choose whether to use the power split mode.

The RTC_RES_PROG_REG has been added to back up the resistance value of the oscillator in OFF state.

A power-down signal has been added to set the outputs to 0 in OFF. See Table 23–2 through Table 23–8 for the RTC registers.

23.1.10 RTC Registers

There are three types of registers:

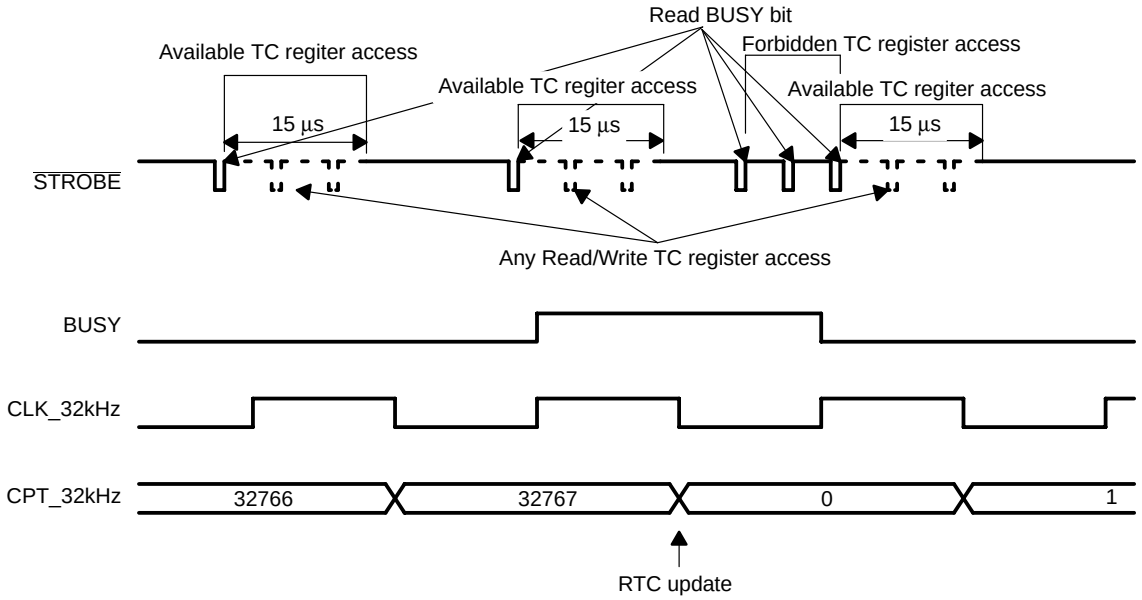
- Time and calendar, and time and calendar alarm
- General
- Compensation

These three types have their own access constraints.

23.1.10.1 Time and Calendar Registers and Time and Calendar Alarm Registers

To read or write correct data from/to the time and calendar registers and time and calendar alarm registers, the MPU must first read the BUSY bit of the status register until BUSY is equal to zero. From this time, and for a time of 15 µs (the available access period), the MPU can perform several accesses into the time and calendar registers and time and calendar alarm registers with ensured read/write data. At the end of one available access period, the MPU must restart the previous sequence. If the MPU accesses the time and calendar registers during an unavailable access period, access is not ensured.

Figure 23–15. Time and Calendar Register and Time and Calendar Alarm Register Access



To remove any possibility of interrupting the register read process, thus introducing a potential risk of violating the authorized 15–µs access period, it is strongly recommended that the user disable all incoming interrupts during the register read process.

23.1.10.2 General Registers

The MPU can access the STATUS_REG and the CTRL_REG at any time (with the exception of the CTRL_REG[5] bit, which must be changed only when the RTC is stopped).

For the INTERRUPTS_REG, the MPU must respect the available access period to prevent spurious interrupt.

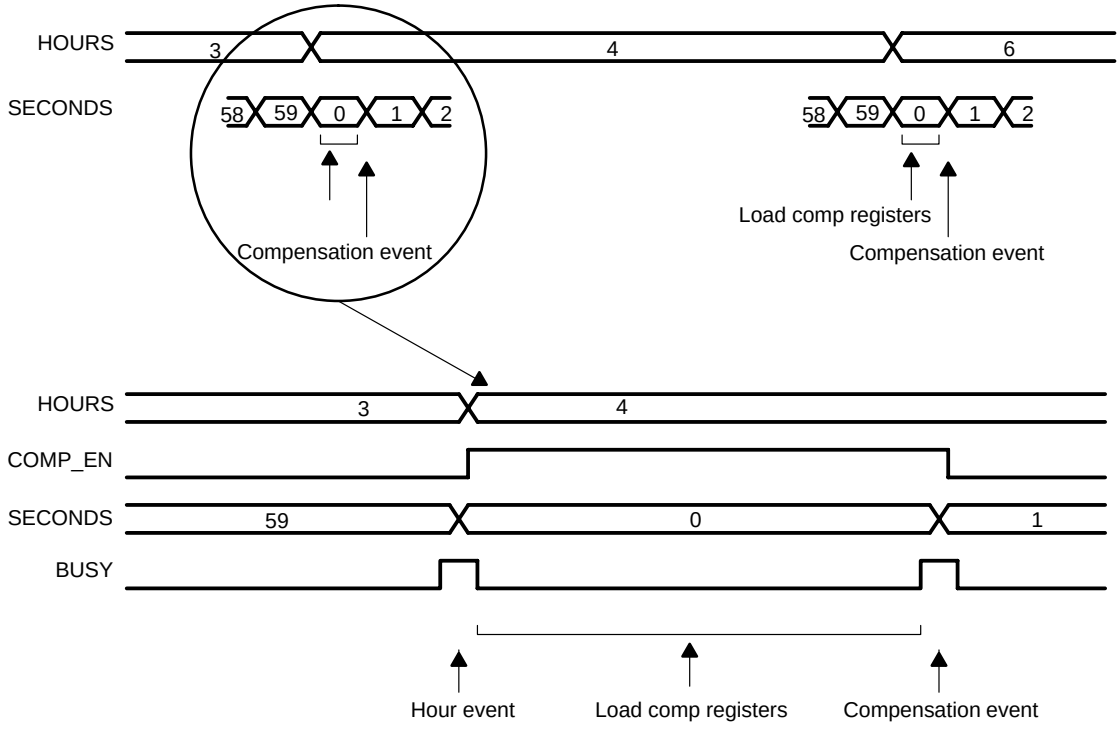
The RTC_DISABLE bit of the CTRL register must be used only to completely disable the RTC function. When this bit is set, the 32-kHz clock is gated, and the RTC is frozen. From this point, resetting this bit to zero can lead to unexpected behavior. This bit must only be used if the RTC function is unwanted in the application, to save power.

23.1.10.3 Compensation Registers

Access to the COMP_MSB_REG and COMP_LSB_REG registers must respect the available access period. These registers must not be updated during compensation (first second of each hour), but they can be updated during the second access period preceding a compensation event.

For example, the MPU can load the compensation value into these registers after each hour event during an available access period.

Figure 23–16. Compensation Scheduling



23.1.10.4 Setting Time and Calendar Information

Modify Time and Calendar Registers

To modify the current time, MPU writes the new time into time and calendar registers to fix the time/calendar information. MPU can write into time and calendar registers without stopping the RTC, but in this case MPU must read the status register to be sure that the RTC updating takes place in more than 15 µs (bit BUSY must be at 0). MPU must perform all changes in less than 15 µs to prevent partial updating between the beginning and the end of the writing sequence into time and calendar registers.

Also, MPU can stop the RTC by clearing the STOP_RTC bit of the control register (owing to internal resynchronization, the RUN bit of the status must be checked to ensure that the RTC is frozen), updating the time and calendar values, and restarting the RTC by resetting the STOP_RTC bit.

Rounding Seconds

Time can be rounded to the closest minute by setting the ROUND_30S bit of the control register. When this bit is set, time and calendar values are set to the closest minute value at the next second. The ROUND_30S bit is automatically cleared when rounding time is performed.

Example:

If current time is 10H59M45S, round operation changes time to 11H00M00S.

If current time is 10H59M29S, round operation changes time to 10H59M00S.

Table 23–2 lists the RTC registers. Table 23–3 through Table 23–21 describe the register bits. The RTC register types are grouped as follows:

- General registers (Table 23–3 and Table 23–6 through Table 23–8)
- Compensation registers (Table 23–4 and Table 23–5)
- Time and calendar alarm registers (Table 23–9 through Table 23–14)
- Time and calendar registers (Table 23–15 through Table 23–21)

Table 23–2. RTC Registers

Name	Description	Address Offset
RTC_OSC_REG	RTC oscillator	15
RTC_COMP_MSB_REG	RTC compensation MSB	14
RTC_COMP_LSB_REG	RTC compensation LSB	13
RTC_INTERRUPTS_REG	RTC interrupt	12
RTC_STATUS_REG	RTC status	11
RTC_CTRL_REG	RTC control	10

Table 23–3. RTC Oscillator Register (RTC_OSC_REG)

Bit	Name	Function	R/W	Reset Value
4	OSC32K_PWRDN_R	Control of 32-kHz oscillator power down (function mode)	R/W	0
3:0	SW_RES_PROG	Value of the oscillator resistance	R/W	

The oscillator receives the register value when TST_OSC32K_MUX_CTRL = 0 (functional mode); otherwise, the value resistance and the power down are from the JTAG register.

Table 23–4. RTC Compensation MSB Register (RTC_COMP_MSB_REG)

Bit	Name	Function	R/W	Reset Value
7:0	RTC_COMP_MSB	Indicates number of 32-kHz periods to be added into the 32-kHz counter every hour	R/W	0x00

Table 23–5. RTC Compensation LSB Register (RTC_COMP_LSB_REG)

Bit	Name	Function	R/W	Reset Value
7:0	RTC_COMP_LSB	Indicates number of 32-kHz periods to be added into the 32-kHz counter every hour	R/W	0x00

Note:

The RTC compensation MSB/LSB registers must be written in twos complement. This means that to add one 32-kHz oscillator period every hour, the MPU must write FFFF into RTC_COMP_MSB_REG and RTC_COMP_LSB_REG. To remove one 32-kHz oscillator period every hour, the MPU must write 0001 into RTC_COMP_MSB_REG and RTC_COMP_LSB_REG. The 7FFF value is not allowed.

Table 23–6. RTC Interrupts Register (RTC_INTERRUPTS_REG)

Bit	Name	Function	R/W	Reset Value
3	IT_ALARM	Enables one interrupt when the alarm value is reached (time and calendar alarm registers) by the time and calendar registers	R/W	0
2	IT_TIMER	Enable periodic interrupt 0: Interrupt disabled 1: Interrupt enabled	R/W	0
1:0	EVERY	Interrupt period 0: Every second 1: Every minute 2: Every hour 3: Every day	R/W	00

Note:

The MPU must respect the BUSY period to prevent spurious interrupt.

Table 23–7. RTC Status Register (RTC_STATUS_REG)

Bit	Name	Function	R/W	Reset Value
7	POWER_UP	Indicates that a reset occurred	R/W	1
6	ALARM	Indicates that an alarm interrupt has been generated	R/W	0
5	1D_EVENT	One day has occurred.	R	0
4	1H_EVENT	One hour has occurred.	R	0
3	1M_EVENT	One minute has occurred.	R	0
2	1S_EVENT	One second has occurred.	R	0
1	RUN	0: RTC is frozen. 1: RTC is running.	R	0
0	BUSY	0: Updating event in more than 15 µs 1: Updating event	R	0

The alarm interrupt keeps its low level until the MPU writes 1 in the ALARM bit of the RTC status register.

The timer interrupt is a low-level pulse (15-µs duration).

The RUN bit shows the real state of the RTC. Because the STOP_RTC signal is resynchronized on the 32-kHz clock, the action of this bit is delayed.

POWER_UP is set by a reset and is cleared by writing 1 in this bit.

Table 23–8. RTC Control Register (RTC_CTRL_REG)

Bit	Name	Function	R/W	Reset Value
7	$\overline{\text{SPLIT\_POWER}}$	0: Cannot use split power	R/W	0
		1: Can use split power		
6	RTC_disable	0: RTC enabled	R/W	0
		1: RTC disabled (no 32-kHz clock)		
5	SET_32_COUNTER	0: No action	R/W	0
		1: Set the 32-kHz counter with COMP_REG value.		
4	TEST_MODE	0: Functional mode	R/W	0
		1: Test mode (autocompensation is enabled when the 32-kHz counter reaches its end)		
3	MODE_12_24	0: 24-hour mode	R/W	0
		1: 12-hour mode (PM/AM mode)		
2	AUTO_COMP	0: No autocompensation	R/W	0
		1: Autocompensation enabled		
1	ROUND_30S	0: No update	R/W	0
		1: When a 1 is written, the time is rounded to the closest minute.		
0	STOP_RTC	0: RTC is frozen.	R/W	0
		1: RTC is running.		

The SET_32_COUNTER in the RTC control register must be used only when the RTC is frozen.

ROUND_30S is a toggle bit. MPU can only write 1, and RTC clears it. If the MPU sets the ROUND_30S bit and then reads it, the MPU reads 1 until the round-to-the-closest-minute is performed at the next second.

MODE_12_24: It is possible to switch between the two modes at any time without disturbing the RTC. Read or write is always performed with the current mode.

23.1.10.5 Time and Calendar Registers

The time and calendar information is available in dedicated registers called time and calendar registers. These register values are written in binary coded decimal (BCD) code. See Table 23–9 through Table 23–14 for the time and calendar alarm registers, and Table 23–15 through Table 23–21 for the time and calendar registers.

Time Unit	Range	Remarks
Year	00 to 99	Leap year: Year divisible by four
		Common year: Other years
Month	01 to 12	
Day	01 to 31	01 to 31 for months 1, 3, 5, 7, 8, 10, 12
		01 to 30 for months 4, 6, 9 ,11
		01 to 29 for month 2 (leap year)
		01 to 28 for month 2 (common year)
Week	00 to 06	Weekday
Hour	00 to 23	00 to 23 in 24 hours mode
		01 to 12 in AM/PM mode
Minutes	00 to 59	
Seconds	00 to 59	

Table 23–9.Alarm Years Register (ALARM_YEARS_REG)

Bit	Name	Function	R/W	Reset Value
7:4	ALARM_YEAR1	2nd digit of years	R/W	0000
		Range is 0 to 9.		
3:0	ALARM_YEAR0	1st digit of years	R/W	0000
		Range is 0 to 9.		

Table 23–10. Alarm Months Register (ALARM_MONTHS_REG)

Bit	Name	Function	R/W	Reset Value
7:4	ALARM_MONTH1	2nd digit of months	R/W	0000
		Range is 0 to 1.		
3:0	ALARM_MONTH0	1st digit of months	R/W	0001
		Range is 0 to 9.		

Table 23–11. Alarm Days Register (ALARM_DAYS_REG)

Bit	Name	Function	R/W	Reset Value
7:4	ALARM_DAY1	2nd digit for days Range is 0 to 3.	R/W	0000
3:0	ALARM_DAY0	1st digit for days Range is 0 to 9.	R/W	0001

Table 23–12. Alarm Hours Register (ALARM_HOURS_REG)

Bit	Name	Function	R/W	Reset Value
7	ALARM_PM_$\overline{AM}$	Only used in PM_AM mode (otherwise, 0) 0: AM 1: PM	R/W	0
6:4	ALARM_HOUR1	2nd digit of hours Range is 0 to 2.	R/W	000
3:0	ALARM_HOUR0	1st digit of hours Range is 0 to 9.	R/W	0000

Table 23–13. Alarm Minutes Register (ALARM_MINUTES_REG)

Bit	Name	Function	R/W	Reset Value
7:4	ALARM_MIN1	2nd digit of minutes Range is 0 to 5.	R/W	0000
3:0	ALARM_MIN0	1st digit of minutes Range is 0 to 9.	R/W	0000

Table 23–14. Alarm Seconds Register (ALARM_SECONDS_REG)

Bit	Name	Function	R/W	Reset Value
7:4	ALARM_SEC1	2nd digit of seconds Range is 0 to 5.	R/W	0000
3:0	ALARM_SEC0	1st digit of seconds Range is 0 to 9.	R/W	0000

Table 23–15. Weeks Register (WEEKS_REG)

Bit	Name	Function	R/W	Reset Value
3:0	WEEK	1st digit of days in a week Range is 0 to 6.	R/W	0000

Table 23–16. Years Register (YEARS_REG)

Bit	Name	Function	R/W	Reset Value
7:4	YEAR1	2nd digit of years Range is 0 to 9.	R/W	0000
3:0	YEAR0	1st digit of years Range is 0 to 9.	R/W	0000

Table 23–17. Months Register (MONTHS_REG)

Bit	Name	Function	R/W	Reset Value
7:4	MONTH1	2nd digit of months Range is 0 to 1.	R/W	0000
3:0	MONTH0	1st digit of months Range is 0 to 9.	R/W	0001

Usual notation taken for the month value:

01: January
02: February
....
12: December

Table 23–18. Days Register (DAYS_REG)

Bit	Name	Function	R/W	Reset Value
7:4	DAY1	2nd digit of days Range is 0 to 3	R/W	0000
3:0	DAY0	1st digit of days Range is 0 to 9	R/W	0001

Table 23–19. Hours Register (HOURS_REG)

Bit	Name	Function	R/W	Reset Value
7	PM_$\overline{AM}$	Only used in PM_AM mode (otherwise, 0) 0: AM 1: PM	R/W	0
6:4	HOUR1	2nd digit of hours Range is 0 to 2.	R/W	000
3:0	HOUR0	1st digit of hours Range is 0 to 9.	R/W	0000

Table 23–20. Minutes Register (MINUTES_REG)

Bit	Name	Function	R/W	Reset Value
7:4	MIN1	2nd digit of minutes Range is 0 to 5.	R/W	0000
3:0	MIN0	1st digit of minutes Range is 0 to 9.	R/W	0000

Table 23–21. Seconds Register (SECONDS_REG)

Bit	Name	Function	R/W	Reset Value
7:4	SEC1	2nd digit of seconds Range is 0 to 5.	R/W	0000
3:0	SEC0	1st digit of seconds Range is 0 to 9.	R/W	0000

Chapter 24

Memory Mapping

This chapter describes the shared memory and memory mapping for the MPU-S and the GSM-S.

The GSM-S and the MPU-S share the same memories through the use of the traffic controller (TC).

Topic **Page**

24.1 MPU-S Memory Mapping

The external memory space is shared by the ARM926EJS and the DMA controller. Shared data can be stored in this data space. Because the cache coherency feature is not supported by the cache controller of ARM926EJS, the shared data space must be used as noncacheable space.

24.1.1 MPU Memory Space

Seven chip-selects are provided for the external memory devices and the internal memory. A 64M-byte address range is available (except for the external SDRAM bus and the local bus) for each chip-select. To minimize the design complexity and optimize memory access time, some chip-selects are dedicated to devices plugged into specific buses. Figure 24–1 shows the MPU memory map and Table 24–1 describes the chip-selects associated with the memory bus.

Figure 24–1. MPU Memory Maps

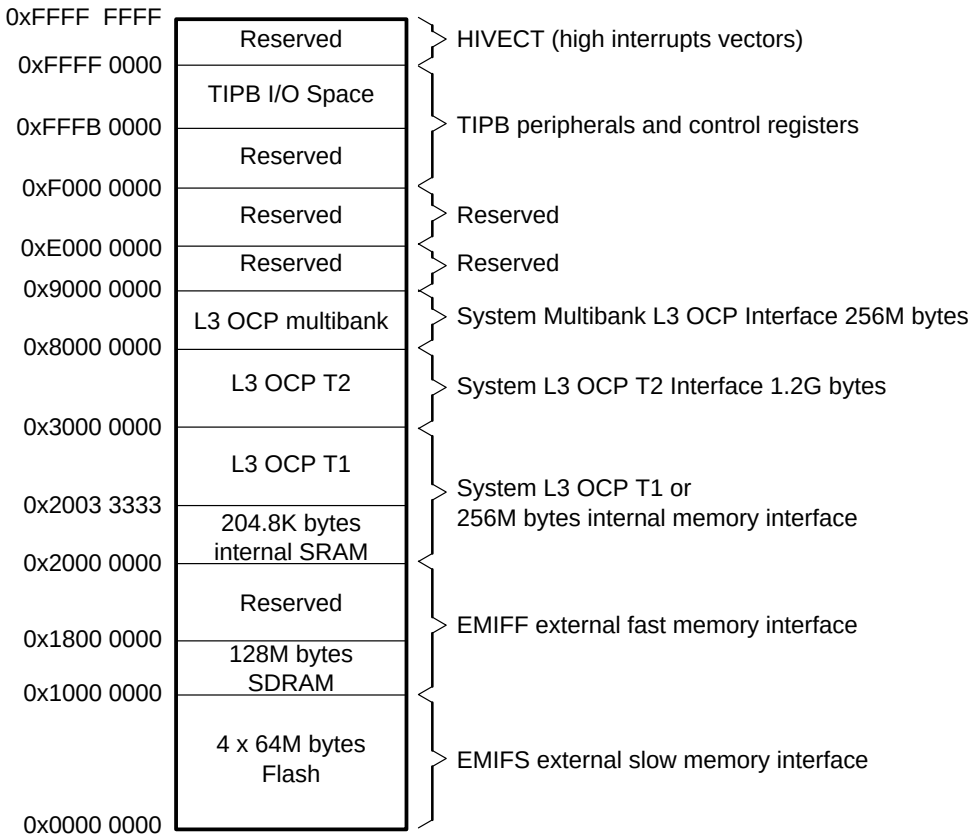


Table 24–1. Memory Bus Associated With Chip-Selects

Device Name		Start Address	Stop Address	Size	Data Access
Flash, SDRAM and OCP Space					
External Slow Memory Interface (EMIFS)					
nCS0	Flash	0000:0000	03FF:FFFF	64M bytes	8/16/32
nCS1	Flash	0400:0000	07FF:FFFF	64M bytes	8/16/32
nCS2	Flash	0800:0000	0BFF:FFFF	64M bytes	8/16/32
nCS3	Flash	0C00:0000	0FFF:FFFF	64M bytes	8/16/32
External Fast Memory Interface (EMIFF)					
nCS4	SDRAM	1000:0000	17FF:FFFF	128M bytes	8/16/32
Reserved		1800:0000	1FFF:FFFF		
OCP Targets					
OCP-T1		2000:0000	2FFF:FFFF	256M bytes	8/16/32
OCP-T2		3000:0000	7FFF:FFFF	1.2G bytes	8/16/32
OCP-MB		8000:0000	8FFF:FFFF	256M bytes	8/16/32

Internal and external peripherals are mapped to the ARM926EJS memory space in two different sections. These spaces are accessible through STROBE 1 and STROBE 0 with a range of 2K bytes for each external peripheral, allowing connection with up to 122 external devices (two MPU TIPB bridges).

Table 24–2 and Table 24–3 describe the MPU memory space and the TIPB peripherals address space, respectively.

Table 24–2. MPU Memory Address Space

Device Name		Start Address	Stop Address	Size	Data Access
External Slow Memory Interface (EMIFS)					
nCS0		0000:0000	03FF:FFFF	64M bytes	16/32 R/W
	Boot ROM	0000:0000	0000:FFFF	64K bytes	32-bit execute only
	Reserved boot ROM	0001:0000	0003:FFFF	192K bytes	32-bit execute only
	Reserved	0004:0000	001F:FFFF		
	Secure RAM	0020:0000	0020:3FFF	16K bytes	16-bit secure R/W
	Reserved	0020:4000	0020:FFFF		
	Secure eFuse chain1	0021:0000	0021:000F	128 bits	Read only
	Secure eFuse chain2	0021:0010	0021:002F	256 bits	Read only
	Reserved	0021:0030	01FF:FFFF		
	NOR flash	0200:0000	03FF:FFFF	32M bytes	16/32 R/W

Table 24–2. MPU Memory Address Space (Continued)

Device Name		Start Address	Stop Address	Size	Data Access
External Slow Memory Interface (EMIFS) (Continued)					
nCS1	NOR flash	0400:0000	07FF:FFFF	64M bytes	16/32 R/W
nCS2	NOR flash	0800:0000	0BFF:FFFF	64M bytes	16/32 R/W
nCS3	NOR flash	0C00:0000	0FFF:FFFF	64M bytes	16/32 R/W
External Fast Memory Interface (EMIFF)					
nCS4	SDRAM	1000:0000	17FF:FFFF	128M bytes	16 R/W
Reserved		1800:0000	1FFF:FFFF		
L3 OCP T1					
SRAM		2000:0000	2003:3332	1.6M bits	32 R/W
Reserved		2003:3333	2FFF:FFFF		
L3 OCP T2					
Reserved		3000:0000	3000:0FFF		
Reserved		3000:1000	3000:1FFF		
VLYNQ registers		3000:2000	3000:21FF	512 bytes	32 R/W
Reserved		3000:2200	30FF:FFFF		
VLYNQ TX		3100:0000	34FF:FFFF	64M bytes	
Reserved		3500:0000	7FFF:FFFF		

Table 24–3. TIPB Peripherals Address Space

Device Name		Start Address	End Address	Size (Bytes)	Data Access	Address Alignment	Address Compatibility
Reserved		F0000:0000	FFFA:FFFF				
Public TIPB Bridge							
STROBE0, FFFB:0000 -> FFFB:FFFF							
CS = 0		FFFB:0000	FFFB:07FF	2K			
	UART_MODEM 1	FFFB:0000	FFFB:03FF	1K	8	8	P1, H1, H2
	USB OTG	FFFB:0400	FFFB:07FF	1K	32	32	H2
CS = 1	UART_MODEM_IRDA 2	FFFB:0800	FFFB:0FFF	2K	8	8	P1, H1, H2
CS = 2	McBSP1	FFFB:1000	FFFB:17FF	2K	16	16	P1, H1, H2
CS = 3	McBSP2	FFFB:1800	FFFB:1FFF	2K	16	16	H1
CS = 4	MCSI	FFFB:2000	FFFB:27FF	2K	16	16	None

Table 24–3. TIPB Peripherals Address Space (Continued)

Device Name	Start Address	End Address	Size (Bytes)	Data Access	Address Alignment	Address Compatibility
Public TIPB Bridge (Continued)						
STROBE0, FFFB:0000 -> FFFB:FFFF (Continued)						
CS = 5 NAND_FLASH	FFFB:2800	FFFB:2FFF	2K	32	32	None
CS = 6 µWire	FFFB:3000	FFFB:37FF	2K	16	16	P1, H1
CS = 7 I^2C	FFFB:3800	FFFB:3FFF	2K	16	16	P1, H1, H2
CS = 8	FFFB:4000	FFFB:47FF	2K			
USB client	FFFB:4000	FFFB:43FF	1K	32	32	P1, H1, H2
Reserved	FFFB:4400	FFFB:47FF	1K			
CS = 9 RTC	FFFB:4800	FFFB:4FFF	2K	8	8	P1, H1, H2
CS = 10 MPUIO	FFFB:5000	FFFB:57FF	2K	16	16	H1, H2
CS = 11 PWL	FFFB:5800	FFFB:5FFF	2K	8	8	H1, H2
CS = 12 PWT	FFFB:6000	FFFB:67FF	2K	8	8	H1, H2
CS = 13 CAMERA_IF	FFFB:6800	FFFB:6FFF	2K	32	32	H1, H2
CS = 14 HDQ_1WIRE	FFFB:7000	FFFB:77FF	2K	32	32	None
CS = 15 MMC_SDIO	FFFB:7800	FFFB:7FFF	2K	16	16	P1, H1, H2
CS = 16 Reserved	FFFB:8000	FFFB:87FF	2K			
CS = 17 SMC	FFFB:8800	FFFB:8FFF	2K	16	16	None
CS = 18 TIMER32K	FFFB:9000	FFFB:97FF	2K	32	32	P1, H1, H2
CS = 19 DUAL_MODE_TIMER	FFFB:9800	FFFB:9FFF	2K	32	32	None
CS = 20	FFFB:A000	FFFB:A7FF	2K			
USB host	FFFB:A000	FFFB:A3FF	1K	32	32	H1, H2
Reserved	FFFB:4000	FFFB:47FF	1K			
CS = 21 LPG	FFFB:A800	FFFB:AFFF	2K	8	8	P1
CS = 22 EAC	FFFB:B000	FFFB:B7FF	2K	16	16	P1
CS = 23 ICR	FFFB:B800	FFFB:BFFF	2K	16	16	P1
CS = 24 MPUIO_1	FFFB:C000	FFFB:C7FF	2K	32	32	None
CS = 25 MPUIO_2	FFFB:C800	FFFB:CFFF	2K	32	32	P1
CS = 26 MPUIO_3	FFFB:D000	FFFB:D7FF	2K	32	32	P1
CS = 27 MPUIO_4	FFFB:D800	FFFB:DFFF	2K	32	32	P1

Table 24–3. TIPB Peripherals Address Space (Continued)

Device Name	Start Address	End Address	Size (Bytes)	Data Access	Address Alignment	Address Compatibility
Public TIPB Bridge (Continued)						
STROBE0, FFFB:0000 -> FFFB:FFFF (Continued)						
CS = 28 MPUIO_5	FFFB:E000	FFFB:E7FF	2K	32	32	None
CS = 29 MPUIO_6	FFFB:E800	FFFB:EFFF	2K	32	32	None
CS = 30 SPGPIO_WR	FFFB:F000	FFFB:F7FF	2K			None
CS = 31 Reserved	FFFB:F800	FFFB:FFFF	2K			
STROBE1, FFFC:0000 -> FFFC:FFFF						
CS = 0 LLPC	FFFC:0000	FFFC:07FF	2K	16	16	None
CS = 1 SPI_100K_1	FFFC:0800	FFFC:0FFF	2K	16	16	None
CS = 2 SPI_100K_2	FFFC:1000	FFFC:17FF	2K	16	16	None
CS = 3 SYREN_SPI	FFFC:1800	FFFC:1FFF	2K	16	16	None
CS = 4 to 31 Reserved	FFFC:2000	FFFC:FFFF	56K			
Private TIPB Bridge						
STROBE0, FFFD:0000 -> FFFD:FFFF						
CS = 0 to 31 Reserved	FFFD:0000	FFFD:FFFF	64K			
STROBE1, FFFE:0000 -> FFFE:FFFF						
CS = 0 Level2_INTH	FFFE:0000	FFFE:07FF	2K	32	32	P1, H1, H2
CS = 1 PCC_ULPD	FFFE:0800	FFFE:0FFF	2K	16	32	None
CS = 2 PERSEUS_CONF	FFFE:1000	FFFE:17FF	2K	16	16	P1
CS = 3 GSM_PROTECT	FFFE:1800	FFFE:1FFF	2K	32	32	None
CS = 4 Reserved	FFFE:2000	FFFE:27FF	2K			
CS = 5 COMPACT_FLASH	FFFE:2800	FFFE:2FFF	2K	16	16	H2
CS = 6 Reserved	FFFE:3000	FFFE:37FF	2K			
CS = 7 Reserved	FFFE:3800	FFFE:3FFF	2K			
CS = 8 DES/3DES	FFFE:4000	FFFE:47FF	2K	32	32	H2
CS = 9 SHA1/MD5	FFFE:4800	FFFE:4FFF	2K	32	32	H2
CS = 10 RNG	FFFE:5000	FFFE:57FF	2K	32	32	H2
CS = 11 to 20 Unused	FFFE:5800	FFFE:A7FF	20K			
CS = 21 SWATCHDOG	FFFE:A800	FFFE:AFFF	2K	32	32	H2

Table 24–3. TIPB Peripherals Address Space (Continued)

Device Name	Start Address	End Address	Size (Bytes)	Data Access	Address Alignment	Address Compatibility
Private TIPB Bridge (Continued)						
STROBE1, FFFE:0000 -> FFFE:FFFF (Continued)						
CS = 22 to 23 Unused	FFFE:B000	FFFE:BFFF	4K			
CS = 24	FFFE:C000	FFFE:C7FF	2K			
LCD	FFFE:C000	FFFE:C0FF	256	32	32	P1, H1, H2
L3 OCP T1	FFFE:C100	FFFE:C1FF	256	32	32	H2
L3 OCP T2	FFFE:C200	FFFE:C2FF	256	32	32	None
L3 OCP initiator	FFFE:C320	FFFE:C3FF	224	32	32	H2
Reserved	FFFE:C400	FFFE:C4FF	256			
Timer1	FFFE:C500	FFFE:C5FF	256	32	32	P1, H1, H2
Timer2	FFFE:C600	FFFE:C6FF	256	32	32	P1, H1, H2
Timer3	FFFE:C700	FFFE:C7FF	256	32	32	P1, H1, H2
CS = 25	FFFE:C800	FFFE:CFFF	2K			
Watchdog timer	FFFE:C800	FFFE:C8FF	256	32	32	P1, H1, H2
Reserved	FFFE:C900	FFFE:C9FF	256			
TIPB bridge1 (private)	FFFE:CA00	FFFE:CAFF	256	16	32	P1, H1, H2
MPU interrupt handler	FFFE:CB00	FFFE:CBFF	256	32	32	P1, H1, H2
Traffic controller	FFFE:CC00	FFFE:CCFF	256	32	32	P1, H1, H2
Reserved	FFFE:CD00	FFFE:CDFF	256			
CLKM	FFFE:CE00	FFFE:CEFF	256	32	32	P1, H1, H2
DPLL1	FFFE:CF00	FFFE:CFFF	256	32	32	P1, H1, H2
CS = 26	FFFE:D000	FFFE:D7FF	2K			
Reserved	FFFE:D000	FFFE:D0FF	256			
Reserved	FFFE:D100	FFFE:D1FF	256			
DSP MMU	FFFE:D200	FFFE:D2FF	256	32	32	H1, H2
TIPB bridge2 (public)	FFFE:D300	FFFE:D3FF	256	16	32	P1, H1, H2
Test block (PSA)	FFFE:D400	FFFE:D4FF	256	32	32	H2

Table 24–3. TIPB Peripherals Address Space (Continued)

Device Name	Start Address	End Address	Size (Bytes)	Data Access	Address Alignment	Address Compatibility
Private TIPB Bridge (Continued)						
STROBE1, FFFE:0000 -> FFFE:FFFF (Continued)						
Reserved	FFFE:D500	FFFE:D7FF				
CS = 27 and 28	FFFE:D800	FFFE:E7FF	4K			
DMA controller	FFFE:D800	FFFE:E7FF	4K	32	32	P1, H1, H2
CS = 29 Reserved	FFFE:E800	FFFE:EFFF	2K			
CS = 30 Reserved	FFFE:F000	FFFE:F7FF	2K			
CS = 31 Reserved	FFFE:F800	FFFE:FFFF	2K			

24.2 GSM-MPU Memory Mapping

The GSM-MPU memory space is shared between the external memory interface and the TIPB. The memory interface provides six chip-select signals. All internal peripherals are mapped on GSM-MPU memory space with a range of 32K bytes.

The 8K bytes of internal RAM (0380:0000h to 0380:1000h) can overlay the first 8K-byte region 0000:0000h-0000:1000h of the GSM-MPU address space. In this case, the first 8K-byte of the external memory is not accessible by the GSM-MPU. This overlay is controlled by GSM-MPU using a register of GSM-MPU memory interface.

24.2.1 GSM-MPU Memory Mapping

Table 24–4 lists the GSM-MPU memory map.

Table 24–4. GSM-MPU Memory Map

Device Name	nIBOOT	Start Address	Stop Address	Size (Bytes)	Data
nCS0 program†	1	0000:0000	007F:FFFF	8M	8/16/32
	0	0000:2000	007F:FFFF	8M – 8K	
nCS6	-	0080:0000	0084:FFFF	320K	8/16/32
nCS6 DSP-shared	-	0085:0000	0085:FFFF	64K	8/16/32
Not allocated	-	00C0:0000	00FF:FFFF	-	-
nCS1: data†	-	0100:0000	017F:FFFF	8M	8/16/32
nCS2: random†	-	0180:0000	01FF:FFFF	8M	8/16/32
Not allocated	-	0200:0000	02BF:FFFF	-	8/16/32
nCS0 image	-	0300:0000	037F:FFFF	8M	8/16/32
nCS7	1	0380:0000	03FF:FFFF	8M	8/16/32
	0	0000:0000	0000:1FFF	8K	
Debug unit (DU)	-	03C0:0000	03FF:FFFF	32	32
Not allocated	-	0400:0000	FFCF:FFFF	-	-
MPUI RAM	-	FFD0:0000	FFD0:3FFF	16K	16/32
MPUI control register	-	FFE0:0000	FFE0:0001	2	16

† External device

24.3 External Flash ROM Image

Whatever the value of nIBOOT, it is possible to write to or read from (depending on the value of the WE bit) the external memory connected to nCS0 at address range nCS0 image.

Table 24–5 presents the data format.

Table 24–5. Data Format

<table>
<tr><th>D32---------->D24</th><th>D23---------->D16</th><th>D15---------->D8</th><th>D7---------->D0</th></tr>
<tr><td colspan="4">nCS0</td></tr>
<tr><td colspan="4">nCS1</td></tr>
<tr><td colspan="4">nCS2</td></tr>
<tr><td colspan="4">nCS3</td></tr>
<tr><td colspan="4">CS4</td></tr>
<tr><td colspan="4">nCS6</td></tr>
<tr><td colspan="4">nCS7</td></tr>
<tr><td colspan="4">MPUI RAM</td></tr>
<tr><td colspan="2"></td><td colspan="2">MPU</td></tr>
<tr><td colspan="2"></td><td colspan="2">GEA</td></tr>
<tr><td colspan="2" rowspan="13">Not mapped</td><td colspan="2">APIC</td></tr>
<tr><td colspan="2">SIM</td></tr>
<tr><td colspan="2">TSP</td></tr>
<tr><td colspan="2">TPU_REG</td></tr>
<tr><td colspan="2">TPU_RAM</td></tr>
<tr><td>Not mapped</td><td>RTC</td></tr>
<tr><td colspan="2">ULPD</td></tr>
<tr><td>Not mapped</td><td>I2C</td></tr>
<tr><td colspan="2">SPI</td></tr>
<tr><td colspan="2">TIMER1</td></tr>
<tr><td rowspan="4">Not mapped</td><td>LPG</td></tr>
<tr><td>PWL</td></tr>
<tr><td>Reserved</td></tr>
<tr><td>PWT</td></tr>
</table>

Table 24–5. Data Format (Continued)

D32----------->D24	D23----------->D16	D15----------->D8	D7----------->D0
		µWIRE	
		MPUIO	
		Not mapped	Reserved
			UART_MODEM
		TIMER2	
		TIPB bridge	
		INTH	
		Memory interface	
		DMA controller	
		CLKM	
		JTAG ID code	
		Die ID code	

Table 24–6 presents the GSM-MPU peripheral mapping for Strobe 0, and Table 24–7 presents it for Strobe 1.

Table 24–6. GSM-MPU Peripheral Mapping (Strobe 0)

Device Name		Start Address	Stop Address	Size (Bytes)	Data
Reserved	CS0	FFFF:0000	FFFF:07FF		
Reserved	CS1	FFFF:0800	FFFF:0FFF		
TPU registers	CS2	FFFF:1000	FFFF:13FF	1K	16
Reserved	CS3	FFFF:1800	FFFF:1FFF		
Reserved	CS4	FFFF:2000	FFFF:27FF		
Reserved	CS5	FFFF:2800	FFFF:2FFF		
Reserved	CS6	FFFF:3000	FFFF:37FF		
Reserved	CS7	FFFF:3800	FFFF:3FFF		
Reserved	CS8	FFFF:4000	FFFF:47FF		
Reserved	CS9	FFFF:4800	FFFF:4FFF		
Reserved	CS10	FFFF:5000	FFFF:57FF		
UART_MODEM	CS11	FFFF:5800	FFFF:5FFF	2K	8
Reserved	CS12	FFFF:6000	FFFF:67FF		
Reserved	CS13	FFFF:6800	FFFF:6FFF		
RIF	CS14	FFFF:7000	FFFF:77FF	2K	16
Reserved	CS15	FFFF:7800	FFFF:7FFF		

Table 24–6. GSM-MPU Peripheral Mapping (Strobe 0) (Continued)

Device Name		Start Address	Stop Address	Size (Bytes)	Data
Reserved	CS16	FFFF:8000	FFFF:87FF		
Reserved	CS17	FFFF:8800	FFFF:8FFF		
TPU RAM	CS18	FFFF:9000	FFFF:97FF	2K	16
DPLL configuration	CS19	FFFF:9800	FFFF:9801	2	16
Not allocated	CS20	FFFF:A000	FFFF:A7FF		
Not allocated	CS21	FFFF:A800	FFFF:AFFF		
Not allocated	CS22	FFFF:B000	FFFF:B7FF		
Not allocated	CS23	FFFF:B800	FFFF:BFFF		
GEA	CS24	FFFF:C000	FFFF:C7FF	2K	8/16
Not allocated	CS25	FFFF:C800	FFFF:CFFF		
Not allocated	CS26	FFFF:D000	FFFF:D7FF		
Not allocated	CS27	FFFF:D800	FFFF:DFFF		
Not allocated	CS28	FFFF:E000	FFFF:E7FF		
Not allocated	CS29	FFFF:E800	FFFF:EFFF		
Reserved	CS30	FFFF:F000	FFFF:F7FF		
Watchdog timer	CS31	FFFF:F800	FFFF:F8FF	256	16
TIPB bridge		FFFF:F900	FFFF:F9FF	256	16
INTH		FFFF:FA00	FFFF:FAFF	256	16
Memory interface		FFFF:FB00	FFFF:FBFF	256	16
DMA controller		FFFF:FC00	FFFF:FCFF	256	16
CLKM		FFFF:FD00	FFFF:FDFF	256	16
JTAG ID code		FFFF:FE00	FFFF:FE03	4	16
MPU		FFFF:FF00	FFFF:FFFF	256	16

Table 24–7. GSM-MPU Peripheral Mapping (Strobe 1)

Device Name		Start Address	Stop Address	Size (Bytes)	Data
SIM	CS0	FFFE:0000	FFFE:07FF	2K	16
TSP	CS1	FFFE:0800	FFFE:0FFF	2K	16
Reserved	CS2	FFFE:1000	FFFE:17FF		
RTC	CS3	FFFE:1800	FFFE:1FFF	2K	8
ULPD	CS4	FFFE:2000	FFFE:27FF	2K	16

Table 24–7. GSM-MPU Peripheral Mapping (Strobe 1) (Continued)

Device Name		Start Address	Stop Address	Size (Bytes)	Data
I^2C	CS5	FFFE:2800	FFFE:2FFF	2K	8
SPI	CS6	FFFE:3000	FFFE:37FF	2K	16
TIMER1	CS7	FFFE:3800	FFFE:3FFF	2K	16
UWIRE	CS8	FFFE:4000	FFFE:47FF	2K	16
MPUIO	CS9	FFFE:4800	FFFE:4FFF	2K	16
Reserved	CS10	FFFE:5000	FFFE: 57FF		
Reserved	CS11	FFFE:5800	FFFE: 5FFF		
Reserved	CS12	FFFE:6000	FFFE: 67FF		
TIMER2	CS13	FFFE:6800	FFFE:6FFF	2K	16
Reserved	CS14	FFFE:7000	FFFE:77FF		
LPG	CS15	FFFE:7800	FFFE:7FFF	2K	8
PWL	CS16	FFFE:8000	FFFE:87FF	2K	8
PWT	CS17	FFFE:8800	FFFE:8FFF	2K	8
Reserved	CS18	FFFE:9000	FFFE:97FF		
Reserved	CS19	FFFE:9800	FFFE:9FFF		
Not allocated	CS20	FFFE:A000	FFFE:A7FF		
TCIF	CS21	FFFE:A800	FFFE:AFFF	2K	16
ICR	CS22	FFFE:B000	FFFE:B7FF	2K	16
Not allocated	CS23	FFFE:B800	FFFE:BFFF		
Reserved	CS24	FFFE:C000	FFFE:C7FF		
Not allocated	CS25	FFFE:C800	FFFE:CFFF		
Not allocated	CS26	FFFE:D000	FFFE:D7FF		
Not allocated	CS27	FFFE:D800	FFFE:DFFF		
Not allocated	CS28	FFFE:E000	FFFE:E7FF		
Not allocated	CS29	FFFE:E800	FFFE:EFFF		
PERSEUS2_CONF	CS30	FFFE:F000	FFFE:F7FF	2K	16

24.4 GSM-S DSP Memory Space

The DSP core is embedded with 28K words (16 bits) of random access memory and 128K words (16 bits) of read only memory.

The GSM-S DSP memory space consists of the following kinds of memory:

- DARAM: Dual-access data RAM. It is always mapped in data space and can be overlaid in program space using the OVLY bit.
- MPUIRAM: Dual-access data RAM. It is always mapped in data space and can be overlaid in program space using the OVLY bit. The GSM-MPU host processor can also access this memory via the MPUI interface module. It behaves as a communication memory between the lead CPU and the GSM-MPU host processor.
- PROM: Program ROM, always in program space
- DROM: Data ROM, always in data space
- PDROM: Program or data ROM. This ROM is always mapped in program space and can also be mapped in data space by setting the DROM control bit.
- Shared PDRAM: Program/data RAM mapped on both the data space and the program space of the DSP XIO interface

For this S28C128 configuration the memory mapping is as follows:

- 28K words of data memory (RAM based) mapped in both data space 0 and 1.
  - 2K words of dual access memory (DARAM)
  - 8K words of dual access memory (MPUI DARAM) shared between DSP and MPU/DMA
  - 18K words of dual access memory (DARAM)
- 128K words of program memory (ROM based)
  - 100K words of program memory (PROM) mapped in program space 0.
  - 20K words of data memory (DROM) mapped in data space 1.

8K words of mixed program/data memory (PDROM) are mapped in both program space 0 and data space 1.

Table 24–8 shows how the DSP memory is mapped.

Table 24–8. DSP Memory Mapping

	Data	Prog0	Prog1	Prog2	Prog3	Prog4	Prog5	Prog6
0000	DARAM overlay over the program area. 2K							
0800	MPUI overlay over the program area 8K							

Table 24–8. DSP Memory Mapping (Continued)

<table>
<tr><th></th><th colspan="2">Data</th><th>Prog0</th><th>Prog1</th><th>Prog2</th><th>Prog3</th><th>Prog4</th><th>Prog5</th><th>Prog6</th></tr>
<tr><td>1000</td><td colspan="9" rowspan="3"></td></tr>
<tr><td>1800</td></tr>
<tr><td>2000</td></tr>
<tr><td>2800</td><td colspan="9" rowspan="9">DARAM overlay over the program area
18K</td></tr>
<tr><td>3000</td></tr>
<tr><td>3800</td></tr>
<tr><td>4000</td></tr>
<tr><td>4800</td></tr>
<tr><td>5000</td></tr>
<tr><td>5800</td></tr>
<tr><td>6000</td></tr>
<tr><td>6800</td></tr>
<tr><td>7000</td><td colspan="2" rowspan="2"></td><td rowspan="12"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>7800</td></tr>
<tr><td>8000</td><td></td><td rowspan="4"></td><td rowspan="11">PROM
32K</td><td rowspan="11">PROM
32K</td><td rowspan="11">PROM
32K</td><td rowspan="4">PROM
8K</td><td rowspan="11">PDRAM
32K</td><td rowspan="11"></td><td rowspan="11"></td></tr>
<tr><td>8800</td><td rowspan="10">PDRAM
32K</td></tr>
<tr><td>9000</td><td rowspan="9">DROM
20K</td><td rowspan="9"></td></tr>
<tr><td>9800</td></tr>
<tr><td>A000</td></tr>
<tr><td>A800</td></tr>
<tr><td>B000</td></tr>
<tr><td>B800</td></tr>
</table>

Table 24–8. DSP Memory Mapping (Continued)

<table>
<tr><th></th><th colspan="2">Data</th><th>Prog0</th><th>Prog1</th><th>Prog2</th><th>Prog3</th><th>Prog4</th><th>Prog5</th><th>Prog6</th></tr>
<tr><td>C000</td><td rowspan="8">DROM M=0</td><td rowspan="4">DROM M=1</td><td rowspan="4"></td><td rowspan="8"></td><td rowspan="8"></td><td rowspan="8"></td><td rowspan="8"></td><td rowspan="8"></td><td rowspan="8"></td></tr>
<tr><td>C800</td></tr>
<tr><td>D000</td></tr>
<tr><td>D800</td></tr>
<tr><td>E000</td><td colspan="2" rowspan="4">PDROM 8K</td></tr>
<tr><td>E800</td></tr>
<tr><td>F000</td></tr>
<tr><td>F800</td></tr>
</table>

24.4.1 MPUI Shared Memory

The MPUI offers dual-access capability to 8K words of 16 bits of mixed data program memory.

The MPUI can be configured to manage data access of 8, 16, or 32 bits through the MPUI control registers and the memory interface configuration registers.

The MPUI dual-access capability is either enabled (SAM) or disabled (HOM) by the DSP. SAM is the default configuration when the DSP exits from a reset phase.

In shared access mode (SAM), the GSM-MPU (or DMA controller) and DSP can access the shared memory space simultaneously, with GSM-MPU access resynchronized on the DSP cycle clock (three+E times ratio required between GSM-MPU and DSP cycle clocks).

In host only mode (HOM), the MPUI RAM is dedicated to the sole external access under the control of either the GSM-MPU or the DMA controller; therefore, the access time is limited by the maximum access time of the DARAM used.

24.4.2 XIO Memory Mapping

All of the data space is mapped on page 0 from address 0x8000 to 0xFFFF.

To avoid overlaying the DROM memory space, the DROM bit is used to select either internal DROM (DROM = 1) or external PDRAM (DROM = 0).

The program space is mapped in the extended page 4, one-half-pages of 32K words of 16 bits. This configuration prevents the DSP memory extension from overlaying the lower half page of each extended page allocated to the existing DARAM and MPUIRAM.

Depending on the amount of memory targeted for program execution, the program space is mapped from 0x48000 to 0x4FFFF in page 4.

The sharing of this memory capacity between the DSP and the GSM-MPU can be statically configured through a dedicated register:

- Case 1 => 0M bits for DSP/0.5M bits for GSM-MPU
- Case 2 => 0.5M bits for DSP/0M bits for GSM-MPU

24.4.3 XIO-TIPB

Internal and external peripherals are mapped on XIO or data memory spaces. These spaces are accessible through nXSTROBE[3:0] with a range of 2K bytes for external peripherals, allowing connection up to:

- 6 external devices on program space
- 26 external devices on data space
- 31 external devices on I/O space

The 32-bit internal peripherals are directly connected on the internal memory interface.

Table 24–9 describes the DSP XIO memory space.

Table 24–9. DSP XIO Memory Space

Device Name		Start Address	Stop Address	Size in Bytes	Data
External Peripherals Mapping - Program Space					
		Strobe 0			
Not allocated	CS0	0000	07FF	2K	16
…	…	…	…	…	…
Not allocated	CS5	3000	37FF	2K	16
External Peripherals Mapping - Data Space 1					
		Strobe 1			
Not allocated	CS6	3800	3FFF	2K	16
UART_MODEM	CS14	7000	77FF	2K	8
MCSI (map1)	CS15	7800	7FFF	2K	16
External Peripherals Mapping - Data Space 2					
		Strobe 2			
Not allocated	CS16	8000	87FF	2K	16
…	…	…	…	…	…
Not allocated	CS31	F800	FFFF	2K	16

Table 24–9. DSP XIO Memory Space (Continued)

Device Name		Start Address	Stop Address	Size in Bytes	Data
External Peripherals Mapping -I/O Space					
		Strobe 3			
RIF	CS0	0000	07FF	2K	16
MCSI-1 (DAI)	CS1	0800	0FFF	2K	16
Not allocated	CS2	1000	17FF	2K	…
Not allocated	CS3	1800	1FFF	2K	…
Not allocated	CS4	2000	27FF	2K	…
A51/2	CS5	2800	2FFF	2K	16
Not allocated		…	…		…
Not allocated	CS14	7000	77FF	2K	16
Not allocated	CS15	7800	7FFF	2K	16
Not allocated	…	…	…	…	…
DMA controller	CS29	E800	EFFF	2K	16
Not allocated	CS30	F000	F7FF	2K	16
XI0-TIPB bridge	CS31	F800	F8FF	256	16
MPUI control		F900	F9FF	256	16
INTH		FA00	FAFF	256	16
NMI_ST_REG		FB00	FBFF	256	16
Not allocated		FC00	FCFF	256	
Not allocated		FD00	FDFF	256	
Not allocated		FE00	FEFF	256	
Not allocated		FF00	FFFF	256	

Chapter 25

Interrupt Mapping

This chapter describes MPU-S and GSM-S interrupt mapping.

Topic **Page**

25.1 MPU-S Interrupt Mapping

The interrupt controller handles 32 interrupt lines. This module handles edge-triggered or level-sensitive interrupts. The interrupts are enabled or disabled with an internal register and can be routed on one of the two MPU processor interrupts according to a programmable bit.

The mapping of incoming interrupts is shown in Table 25–1.

Table 25–1. MPU-S Incoming Interrupts

Incoming Interrupts	Default Sensitivity Configuration	Interrupt Line on Level1	Interrupt Line on Level2	Compatibility
Level2 INTH FIQ	Level	IRQ_0		P1
Level2 INTH IRQ	Level	IRQ_1		P1
USB Non-ISO	Level	IRQ_2		P1
USB ISO	Level	IRQ_3		P1
ICR	Edge	IRQ_4		P1
EAC	Level	IRQ_5		P1
MPUIO_1	Edge	IRQ_6		P1
MPUIO_2	Edge	IRQ_7		P1
MPUIO_3	Edge	IRQ_8		P1
IRQ_ABORT (TIPB)	Level	IRQ_9		P1, H1, H2
McBSP2 TX	Edge	IRQ_10		None
McBSP2 RX	Edge	IRQ_11		None
McBSP2 RX overflow	Edge	IRQ_12		None
IRQ_RHEA_BRIDGE_PRIVATE	Level	IRQ_13		P1, H1
IRQ_LCD_LINE	Level	IRQ_14		None
GSM_PROTECT	Level	IRQ_15		None
IRQ_TIMER3	Edge	IRQ_16		P1, H1, H2
MPUIO_5	Edge	IRQ_17		None
MPUIO_6	Edge	IRQ_18		None
IRQ_DMA_CH0	Level	IRQ_19		P1, H1, H2
IRQ_DMA_CH1	Level	IRQ_20		P1, H1, H2

- P1: OMAP710
- P2: OMAP730
- H1: OMAP1510
- H2: OMAP1610

Table 25–1. MPU-S Incoming Interrupts (Continued)

Incoming Interrupts	Default Sensitivity Configuration	Interrupt Line on Level1	Interrupt Line on Level2	Compatibility
IRQ_DMA_CH2	Level	IRQ_21		P1, H1, H2
IRQ_DMA_CH3	Level	IRQ_22		P1, H1, H2
IRQ_DMA_CH4	Level	IRQ_23		P1, H1, H2
IRQ_DMA_CH5	Level	IRQ_24		P1, H1, H2
IRQ_DMA_CH_LCD	Level	IRQ_25		P1, H1, H2
IRQ_TIMER1	Edge	IRQ_26		P1, H1, H2
IRQ_WD_TIMER	Edge	IRQ_27		P1, H1, H2
IRQ_RHEA_BRIDGE_PUBLIC	Level	IRQ_28		P1, H1
SPGIO_WR	Level	IRQ_29		
IRQ_TIMER2	Edge	IRQ_30		P1, H1, H2
IRQ_LCD_CTRL	Level	IRQ_31		P1, H1, H2
HW_errors (TCIF)		IRQ0 or 1	IRQ_00	P1
Fast external power fail interrupt (NFIQ_PWR_FAIL pin)	Level	IRQ0 or 1	IRQ_01	P1
CompactFlash (CFCD)	Edge	IRQ0 or 1	IRQ_02	P1
CompactFlash controller (CFIREQ)	Edge	IRQ0 or 1	IRQ_03	P1
I^2C	Level	IRQ0 or 1	IRQ_04	H1, H2
PCC	Level	IRQ0 or 1	IRQ_05	None
External interrupt (MPU_EXT_NIRQ pin)	Level	IRQ0 or 1	IRQ_06	P1
SPI_100K_1	Level	IRQ0 or 1	IRQ_07	P1
SYREN_SPI	Level	IRQ0 or 1	IRQ_08	None
VLYNQ	Edge	IRQ0 or 1	IRQ_09	None
MPUIO_4	Edge	IRQ0 or 1	IRQ_10	None
McBSP1 TX	Edge	IRQ0 or 1	IRQ_11	None
McBSP1 RX	Edge	IRQ0 or 1	IRQ_12	None

- P1: OMAP710
- P2: OMAP730
- H1: OMAP1510
- H2: OMAP1610

Table 25–1. MPU-S Incoming Interrupts (Continued)

Incoming Interrupts	Default Sensitivity Configuration	Interrupt Line on Level1	Interrupt Line on Level2	Compatibility
McBSP1 RX overflow	Edge	IRQ0 or 1	IRQ_13	None
UART_MODEM_IRDA 2	Level	IRQ0 or 1	IRQ_14	P1, H2
UART_MODEM 1	Level	IRQ0 or 1	IRQ_15	P1, H2
MCSI	Level	IRQ0 or 1	IRQ_16	H1, H2
µWIRE TX	Edge	IRQ0 or 1	IRQ_17	None
µWIRE RX	Edge	IRQ0 or 1	IRQ_18	None
SMC CD	Level	IRQ0 or 1	IRQ_19	None
SMC IREQ	Level	IRQ0 or 1	IRQ_20	None
HDQ_1WIRE	Level	IRQ0 or 1	IRQ_21	H2
TIMER32K	Edge	IRQ0 or 1	IRQ_22	P1, H2
MMC_SDIO	Level	IRQ0 or 1	IRQ_23	P1, H1, H2
ULPD	Level	IRQ0 or 1	IRQ_24	P1, H2
RTC synchronous timer	Edge	IRQ0 or 1	IRQ_25	P1, H2
RTC alarm	Level	IRQ0 or 1	IRQ_26	P1, H2
USB HHC 1	Level	IRQ0 or 1	IRQ_27	None
USB HHC 2	Level	IRQ0 or 1	IRQ_28	None
USB geni	Level	IRQ0 or 1	IRQ_29	P1
USB OTG	Level	IRQ0 or 1	IRQ_30	None
Camera IF	Level	IRQ0 or 1	IRQ_31	None
RNG	Level	IRQ0 or 1	IRQ_32	None
DUAL_MODE_TIMER	Level	IRQ0 or 1	IRQ_33	None
DBB_RF_EN	Edge	IRQ0 or 1	IRQ_34	None
ARMIO_Keypad	Edge	IRQ0 or 1	IRQ_35	None
SHA-1/MD5	Level	IRQ0 or 1	IRQ_36	None
SPI_100K_2	Level	IRQ0 or 1	IRQ_37	None
RNG idle mode	Level	IRQ0 or 1	IRQ_38	None

- P1: OMAP710
- P2: OMAP730
- H1: OMAP1510
- H2: OMAP1610

Table 25–1. MPU-S Incoming Interrupts (Continued)

Incoming Interrupts	Default Sensitivity Configuration	Interrupt Line on Level1	Interrupt Line on Level2	Compatibility
ARMIO_GPIO	Level	IRQ0 or 1	IRQ_39	None
LLPC_LCD_CTRL_CAN_BE_OFF	Edge	IRQ0 or 1	IRQ_40	None
LLPC_OE_FALLING_EDGE	Edge	IRQ0 or 1	IRQ_41	None
LLPC_OE_RISING_EDGE	Edge	IRQ0 or 1	IRQ_42	None
LLPC_VSYNC	Edge	IRQ0 or 1	IRQ_43	None
Reserved		IRQ0 or 1	IRQ_40 to IRQ_45	
WAKE_UP_REQ	Level	IRQ0 or 1	IRQ_46	H2
Reserved		IRQ0 or 1	IRQ_47 to IRQ_52	
IRQ_DMA_CH6	Edge	IRQ0 or 1	IRQ_53	H2
IRQ_DMA_CH7	Edge	IRQ0 or 1	IRQ_54	H2
IRQ_DMA_CH8	Edge	IRQ0 or 1	IRQ_55	H2
IRQ_DMA_CH9	Edge	IRQ0 or 1	IRQ_56	H2
IRQ_DMA_CH10	Edge	IRQ0 or 1	IRQ_57	H2
IRQ_DMA_CH11	Edge	IRQ0 or 1	IRQ_58	H2
IRQ_DMA_CH12	Edge	IRQ0 or 1	IRQ_59	H2
IRQ_DMA_CH13	Edge	IRQ0 or 1	IRQ_60	H2
IRQ_DMA_CH14	Edge	IRQ0 or 1	IRQ_61	H2
IRQ_DMA_CH15	Edge	IRQ0 or 1	IRQ_62	H2
NAND flash	Level	IRQ0 or 1	IRQ_63	H2

- ❏ P1: OMAP710
- ❏ P2: OMAP730
- ❏ H1: OMAP1510
- ❏ H2: OMAP1610

25.2 GSM-S Interrupt Mapping

The GSM-MPU has two interrupt lines: nIRQ and nFIQ. The ABB fast interrupt is mapped on nFIQ. All peripheral interrupts are mapped as shown in Table 25–2.

Table 25–2. GSM-MPU Peripheral Interrupts Mapping

Name	Sense	IRQ	FIQ	Function
IRQ0	Edge	✓		Watchdog timer interrupts
IRQ1	Edge	✓		TIMER1 interrupt
IRQ2	Edge	✓		TIMER2 interrupt
IRQ3	Edge		✓	TSP receives interrupt
IRQ4	Edge	✓		TPU frame interrupt
IRQ5	Edge	✓		TPU page interrupt
IRQ6	Edge	✓		SIM interrupt
				1) No answer to reset
				2) Character underflow
				3) Character overflow
				4) Character to transmit
				5) Received character
				6) SIM card insertion/extraction
IRQ7	Level	✓		UART_MODEM interrupts
				1) Error on receiver line
				2) Receive time-out
				3) Received character
				4) Character to transmit
				5) Modem status change
				6) Received XOFF/special character detected
				7) CTS/RTS deactivation
				8) DSR/RxD activity detection (off mode only)
IRQ8	Level	✓		Keyboard or GPIO interrupt
IRQ9	Edge	✓		RTC periodical timer interrupt
IRQ10	Level	✓		RTC ALARM or I^2C data transfer error/completion
IRQ11	Edge	✓		ULPD end of gauging interrupt
IRQ12	Level	✓		External interrupt (GSM_EXT_NIRQ pin)†
IRQ13	Edge	✓		SPI interrupt
				1) Received data
				2) Data to transmit

† The GSM_EXT_NIRQ signal is sent to both the GSM-S interrupt handler (IRQ12) and the MPU-S GPIN (5).

Table 25–2. GSM-MPU Peripheral Interrupts Mapping (Continued)

Name	Sense	IRQ	FIQ	Function
IRQ14	Level	✓		DMA interrupt
IRQ15	Edge	✓		MPUI interrupts (nHINT)
IRQ16	Edge		✓	SIM card-detect fast interrupt
IRQ17	Edge		✓	Fast external interrupt
IRQ18		✓		Reserved
IRQ19	Level	✓		ULPD GSM timer
IRQ20	Edge	✓		GEA interrupt
IRQ21-22		✓		Reserved
IRQ23	Level or Edge	✓		GSM edge external MPU
IRQ24	Level		✓	GSM protect
IRQ25				Reserved
IRQ26	Edge			XWC2090 processor ICR interrupt
IRQ27	Edge			TCIF GSM-MPU memory access error

† The GSM_EXT_NIRQ signal is sent to both the GSM-S interrupt handler (IRQ12) and the MPU-S GPIN (5).

25.2.1 DSP Interrupts Mapping

The DSP subchip owns 17 interrupt lines, 11 of which are dedicated for external peripherals (INT0n to INT10n). The mapping of these interrupts is described in Table 25–3.

Table 25–3. DSP Interrupt Mapping

Name	Sense	DSP INT	Function
RSN	Level	RSN	DSP subsystem reset (HW or SW)
nMIN		nMIN	Abort on TIPB bus
	Level	INT0n	RIF receive interrupt
	Level	INT1n	RIF transmit interrupt
		INT2n	UART interrupt ❑ Error on receiver line ❑ Receive timeout ❑ Received character ❑ Character to transmit ❑ Modem status change ❑ Received XOFF/special character detected ❑ CTS/RTS deactivation
TINT		TINT	Timer interrupt

† The TPU programmable interrupt is a facility offered to the DSP programmer to allow generation of a DSP interrupt at a dedicated time with a quarter of GSM bit accuracy. The interrupt is set in a scenario by using a time-stamped instruction.

Table 25–3. DSP Interrupt Mapping (Continued)

Name	Sense	DSP INT	Function
RINT		RINT	SPI receive interrupt
XINT		XINT	SPI transmit interrupt
	Level	INT3n	MCSI receive interrupt
	Level	INT4n	MCSI transmit interrupt
	Level	INT5n	MCSI frame duration error interrupt
	Level	INT6n	MCSI DAI interrupt
	Edge	INT7n	CYPHER interrupt ❑ End of ciphering process ❑ Processing error
	Edge	INT8n	TPU frame interrupt
AINT		AINT	MPUI interrupts
	Edge	INT9n	TPU programmable interrupt†
	Level	INT10n	DMA interrupt
	Edge	INT11n	External DSP interrupt

† The TPU programmable interrupt is a facility offered to the DSP programmer to allow generation of a DSP interrupt at a dedicated time with a quarter of GSM bit accuracy. The interrupt is set in a scenario by using a time-stamped instruction.

Chapter 26

DMA Requests

This chapter discusses the DMA interrupt requests.

Topic **Page**

26.1 MPU-S DMA Requests

Table 26–1 lists the MPU-S DMA requests.

Table 26–1. MPU-S DMA Requests

DMA Requests	MPU System DMA	Compatibility
MCSI TX	DMA_REQ_01	H1, H2
MCSI RX	DMA_REQ_02	H1, H2
I2C RX	DMA_REQ_03	H1, H2
I2C TX	DMA_REQ_04	H1, H2
VLYNQ	DMA_REQ_05	None
SHA1/MD5	DMA_REQ_06	None
µWIRE TX	DMA_REQ_07	H1, H2
McBSP1 TX	DMA_REQ_08	P1, H1, H2
McBSP1 RX	DMA_REQ_09	P1, H1, H2
McBSP2 TX	DMA_REQ_10	H1, H2
McBSP2 RX	DMA_REQ_11	H1, H2
UART_MODEM 1 TX	DMA_REQ_12	P1, H1, H2
UART_MODEM 1 RX	DMA_REQ_13	P1, H1, H2
UART_MODEM_IRDA 2 TX	DMA_REQ_14	P1, H1, H2
UART_MODEM_IRDA 2 RX	DMA_REQ_15	P1, H1, H2
DES/3DES IN	DMA_REQ_16	None
DES/3DES OUT	DMA_REQ_17	None
SMC TX	DMA_REQ_18	None
SMC RX	DMA_REQ_19	None
CAMERA_IF RX	DMA_REQ_20	H1, H2
MMC_SDIO TX	DMA_REQ_21	H1, H2
MMC_SDIO RX	DMA_REQ_22	H1, H2
NAND flash end of burst	DMA_REQ_23	H2
EAC REC	DMA_REQ_24	None
EAC PLAY	DMA_REQ_25	None
USB_OTG RX0 (client)	DMA_REQ_26	H1, H2
USB_OTG RX1 (client)	DMA_REQ_27	H1, H2
USB_OTG RX2 (client)	DMA_REQ_28	H1, H2
USB_OTG TX0 (client)	DMA_REQ_29	H1, H2

Table 26–1. MPU-S DMA Requests (Continued)

DMA Requests	MPU System DMA	Compatibility
USB_OTG TX1 (client)	DMA_REQ_30	H1, H2
USB_OTG TX2 (client)	DMA_REQ_31	H1, H2

- P1: OMAP710
- H1: OMAP1510
- H2: OMAP1610

26.2 GSM-S DMA Requests

The DMA controller manages the access of the following modules to the DSP MPUI 6K-word shared memory:

- GSM-MPU
  - Radio interface (RIF)
  - UART modem
  - Reserved

The RIF RX and RIF TX have a dedicated channel each. The following combinations are possible for UART:

- The UART modem has two channels:
  - Channel 2: TX
  - Channel 3: RX
- The UART modem has one channel:
  - Channel 2: RX or TX
- The UARTs have no DMA.

After reset, all modules have DMA functions disabled.

Table 26–2. DMA Channel Allocation

DMA Request	Channel			
	0	1	2	3
RIF_DMA_REQ_X	✓			
RIF_DMA_REQ_R		✓		
nDMA_REQ_ARM(0) modem			✓	
nDMA_REQ_ARM(1) modem				✓

Note:

Allocate only one UART at a time to each DMA channel (2 or 3). The potential conflicts between concurrent DMA requests must be solved at system level with only one peripheral configured in DMA mode.

Appendix A

MPU-S Registers

This chapter discusses the MPU subsystem (MPU-S) registers for the OMAP730 platform.

Topic **Page**

Topic **Page**

A.1 32-kHz Timer Registers

A.2 Camera Interface Registers

A.3 Clock Generation and Reset Registers

A.4 CompactFlash Controller Registers

A.5 DES/3DES Registers

A.6 DMA Logical Channel Configuration Registers

A.7 DPLL1 Register

A.8 DSP MMU Registers

A.9 Dual-Mode Timer Registers

A.10 EAC Registers

A.11 EMIFF Registers

A.12 EMIFS Registers

A.13 GPIO Registers

A.14 HDQ/1-Wire Registers

A.15 I^2C Registers

A.16 ICR Registers

A.17 Interrupt Registers

A.18 LCD Controller Registers

A.19 LLPC Registers

A.20 LPG Registers

A.21 McBSP Registers

A.22 MCSI Registers

A.23 MicroWire Registers

A.24 MMC Registers

A.25 MPU OS Timer Registers

A.26 MPUIO Registers

A.27 NAND Flash Registers

A.28 OCP Registers

A.29 OCP-T1/OCP-T2 Registers

A.30 OMAP730 Configuration Registers

A.31 OTG Controller Registers

A.32 PCC ULPD Registers

A.33 PWL Registers

A.34 PWT Registers

A.35 RNG Registers

A.36 RTC Registers

A.37 SHA-1/MD5 Registers

A.38 SMC Registers

A.39 SPI Registers

A.40 Swatchdog Registers

A.41 System DMA Registers

A.42 TCIF Functional Registers

A.43 TIPB Registers

A.44 UART Modem/IrDA Registers

A.45 USB Device Controller Registers

A.46 USB Host Controller Registers

A.47 VLYNQ2OCP Configuration Registers

Appendix B

GSM Subsystem Registers

This chapter discusses the GSM subsystem (GSM-S) registers for the OMAP730 platform.

Topic **Page**

Topic **Page**

B.1 Cipher Registers (XIO:2800)

B.2 CLKM Registers(FFFF:FD00)

B.3 Configuration Registers(FFFE:F000)

B.4 DMA Registers (FFFF:FC00 – XIO:FC00)

B.5 DPLL Register (FFFF:9800)

B.6 DSP Interrupts Registers (XIO:FA00 .. XIO:FA01)

B.7 DSP MPUI Configuration Register

B.8 DSP TIPB Registers(XIO:F800)

B.9 GEA Registers

B.10 GSM Protect Registers

B.11 GSM-MPU Interrupt Registers

B.12 GSM-MPU to TIPB Registers

B.13 I^2C Registers (FFFE:2800)

B.14 LPG Registers (FFFE:7800)

B.15 MCSI Registers(XIO:0800)

B.16 Memory Interface Registers

B.17 MPUIO Registers (FFFE:4800)

B.18 μWire Registers (FFFE:4000)

B.19 RIF Registers (FFFF:7000 – XIO:0000)

B.20 RTC Registers (FFFE:1800)

B.21 SIM Registers (FFFE:0000)

B.22 SPI Registers (FFFE:3000)

B.23 Timer Registers (FFFE:3800/FFFE:6800)

B.24 TPU Registers (FFFF:1000)

B.25 TSP Receive and Transmit Registers (FFFE:0800)

B.26 UART Modem Registers (FFFF:5000/6000)

B.27 ULPD Registers (FFFE:2000)

B.28 Watchdog Registers (FFFF:F800)

Appendix C

Pin Descriptions

This appendix presents the OMAP732/733 platform pin descriptions.

Topic **Page**

C.1 Pin Description by Module

Configuration Register = n: PERSEUS_IO_CONFn with n = 0–13

Register Field = n: Bits [4 x n: 3 + 4 x n] with n = 0–7

Table C–1. TPU/TSP (Mode 0 and Mode 2)

Signal	Designation	Type	Mode 0					Mode 2				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
TSPACT_0	Synchronous activation signal	O	W12	tspact_0	PS0201	0	1					
TSPACT_1	Synchronous activation signal	O	R12	tspact_1	PS0201	0	2					
TSPACT_2	Synchronous activation signal	O	P12	tspact_2	PS0201	0	3					
TSPACT_3	Synchronous activation signal	O	W11	tspact_3	PS0201	0	4					
TSPACT_4	Synchronous activation signal	O	V11	tspact_4	PS0201	0	5					
TSPACT_5	Synchronous activation signal	O						AA20	smc_io	PS0201	9	2
TSPACT_6	Synchronous activation signal	O						V17	smc_clk	PS0201	9	3
TSPACT_7	Synchronous activation signal	O						W19	smc_rst	PE0201	9	4
TSPACT_8	Synchronous activation signal	O						V18	smc_cd	PE0201	9	5
TSPACT_9	Synchronous activation signal	O						Y19	smc_ pwctrl	PE0201	9	6
TSPACT_10	Synchronous activation signal	O										
TSPACT_11	Synchronous activation signal	O										
TSPCLKX	Serial clock	O	V10	tspclkx	PS0201	0	0					
TSPDO	Output serial data	O	R11	tspdo	PE0201	0	5					

Table C–1. TPU/TSP (Mode 0 and Mode 2) (Continued)

Signal	Designation	Type	Mode 0					Mode 2				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
TSPDI	Input serial data	I										
TSPEN_0	Configurable enable triggers	O	AA9	tspen_0	PE0201	1	0					
TSPEN_1	Configurable enable triggers	O	P11	tspen_1	PE0201	0	7					
TSPEN_2	Configurable enable triggers	O	W10	tspen_2	PE0201	0	6					
TSPEN_3	Configurable enable triggers	O										

Table C–2. TPU/TSP (Mode3 and Mode 5)

Signal	Designation	Type	Mode 3					Mode 5				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
TSPACT_0	Synchronous activation signal	O										
TSPACT_1	Synchronous activation signal	O										
TSPACT_2	Synchronous activation signal	O										
TSPACT_3	Synchronous activation signal	O										
TSPACT_4	Synchronous activation signal	O										
TSPACT_5	Synchronous activation signal	O						R8	mpu_uart_tx1	PE0201	3	5
TSPACT_6	Synchronous activation signal	O						Y3	mpu_uart_rx1	PE0201	3	5
TSPACT_7	Synchronous activation signal	O						AA1	mpu_uart_cts1	PE0201	3	6

Table C–2. TPU/TSP (Mode3 and Mode 5) (Continued)

Signal	Designation	Type	Mode 3					Mode 5				
			Ball	Pad Name	PU/ PD	Conf Reg	Reg Field	Ball	Pad Name	PU/ PD	Conf Reg	Reg Field
TSPACT_8	Synchro-nous activa-tion signal	O						AA2	mpu_ uart_rts1	PE0201	3	6
TSPACT_9	Synchro-nous activa-tion signal	O	V4	cdi	PE0201	4	6					
TSPACT_10	Synchro-nous activa-tion signal	O	W2	mclk	PE0201	4	7					
TSPACT_11	Synchro-nous activa-tion signal	O	V18	smc_cd	PE0201	9	5					
TSPCLKX	Serial clock	O										
TSPDO	Output serial data	O										
TSPDI	Input serial data	I	F2	kbr_4	PS1001	13	1					
TSPEN_0	Configurable enable triggers	O										
TSPEN_1	Configurable enable triggers	O										
TSPEN_2	Configurable enable triggers	O										
TSPEN_3	Configurable enable triggers	O	T3 F3	nemu0 kbr_3	PS0201 PS1001	10 13	3 0					

Table C–3. GSM-S Voice

Signal	Designation	Type	Mode 1				
			Ball	Pad Name	PU/PD	Conf. Reg.	Reg. Field
VCLKRX	Transmit/receive clock	I	W6	sclk	PE0201	2	1
VDX	Transmit data	O	R9	sdo	PS0201	2	1
VDR	Receive data	I	Y6	sdi	PE0201	2	1
VFSRX	Transmit/receive synchro	I	Y5	fsync	PE0201	2	1

Table C–4. GSM-S RIF BB

Signal	Designation	Type	Mode 0					Mode 2				
			Ball	Pad Name	PU/ PD	Conf Reg	Reg. Field	Ball	Pad Name	PU/ PD	Con- fReg	Reg Field
BFSR	Receive synchro	I	W8	bfsr	PE0201	1	1					
BFSX	Transmit synchro	O	AA7	bfsx	PS0201	1	1					
BDR	Receive data	I	Y8	bdr	PE0201	1	1					
BDX	Transmit data	O	V8	bdx	PS0201	1	1					
BCLKR	Receive clock	I						K15	nfbe_0	UPS290	8	2
BCLKX	Transmit clock	O						K19	nfbe_1	UPS290	8	1

Table C–5. GSM-S SIM

Signal	Designation	Type	Mode 0				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
SIM_RST	SIM reset	O	Y20	sim_rst	PE0201	1	4
SIM_CD	Card detect	I	P15	sim_cd	PE0201	1	4
SIM_IO	Input output signal	I/O	N14	sim_io	PS0201	1	2
SIM_CLK	Output clock	O	W20	sim_clk	PS0201	1	2
SIM_PWRCTRL	Power control	O	V19	sim_pwrctrl	PS0201	1	3

Table C–6. GSM-S MCSI (Mode 1)

Signal	Designation	Type	Mode 1				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
MCSI_TXD	Transmit serial data	O	L1	lcd_pixel_13	PE0201	3	7
			C15	cam_exclk	PS0201	11	1
MCSI_RXD	Receive serial data	I	L3	lcd_pixel_12	PE0201	3	7
			C16	cam_hs	PS0201	11	2
MCSI_CLK	Bit synchronization clock	I/O	M8	lcd_pixel_14	PE0201	3	7
			D17	cam_lclk	PS0201	11	0
MCSI_FSYNCH	Frame synchronization clock or SS reset	I/O	M7	lcd_pixel_15	PE0201	3	7
			D15	cam_vs	PS0201	11	3

Table C–7. GSM-S MCSI (Modes 4 and 5)

Signal	Designation	Type	Mode 4					Mode 5				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
MCSI_TXD	Transmit serial data	O	W17	usb_vbusi	PE0201	2	7	V2	mpu_spi1_sdo	PS0201	8	5
MCSI_RXD	Receive serial data	I	V16	mclk_out	PS0201	3	0	T4	mpu_spi1_sdi	PE0201	8	6
MCSI_CLK	Bit synchronization clock	I/O	W16	usb_pu_en	PS0201	2	6	U3	mpu_spi1_sclk	PE0201	8	4
MCSI_FSYNCH	Frame synchronization clock or SS reset	I/O	W18	creset	PE0201	3	1	V3	mpu_spi1_sen0	PE0201	8	7

Table C–8. GSM-S UART

Signal	Designation	Type	Mode 2					Mode 4				
			Ball	Pad Name	PU/PD	Conf. Reg.	Reg. Field	Ball	Pad Name	PU/PD	Conf. Reg.	Reg. Field
UART_TX	Transmit data	O	V16	mclk_out	PS0201	3	0	Y4	mpu_uart_tx_ir2	PS0201	3	2
UART_RX	Receive data	I	W18	creset	PE0201	3	1	V5	mpu_uart_rx_ir2	PE0201	3	3
UART_CTS	Clear to send	I	W16	usb_pu_en	PS0201	2	6					
UART_RTS	Request to send	O	W17	usb_vbusi	PE0201	2	7					

Table C–9. GSM-S µ-Wire (Mode 1)

Signal	Designation	Type	Mode 1				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
SDI	Data in	I	F4	kbc_3	PE0201	13	5
SDO	Data out	O	E4	kbc_2	PE0201	13	4
SCLK	Serial clock	O	E3	kbc_4	PE0201	13	6
nSCS1	Chip select	O	F2	kbr_4	PS1001	13	1
nSCS2	Chip select	O					

Table C–10. GSM-S μ-Wire (Modes 3 and 4)

Signal	Designation	Type	Mode 3					Mode 4				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
SDI	Data in	I	Y3	mpu_uart_rx1	PE0201	3	5	AA20	smc_io	PS0201	9	2
			T4	mpu_spi1_sdi	PE0201	8	6	A21	cam_rstz	PE0201	11	4
SDO	Data out	O	R8	mpu_uart_tx1	PE0201	3	5	W19	smc_rst	PE0201	9	4
			V2	mpu_spi1_sdo	PS0201	8	5	A20	cam_data_1	PE0201	11	6
SCLK	Serial clock	O	AA1	mpu_uart_cts1	PE0201	3	6	V17	smc_clk	PS0201	9	3
			U3	mpu_spi1_sclk	PE0201	8	4	G14	cam_data_0	PS0201	11	5
nSCS1	Chip select	O	AA2	mpu_uart_rts1	PE0201	3	6	V18	smc_cd	PE0201	9	5
			V3	mpu_spi1_sen0	PE0201	8	7	B19	cam_data_2	PE0201	11	7
nSCS2	Chip select	O						Y19	smc_pwctrl	PE0201	9	6
			U4	mpu_spi1_sen1	PS0201	9	0	D15	cam_vs	PS0201	11	3

Table C–11. GSM-S LPG

Signal	Designation	Type	Mode 1				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
LPG1	LED1 control signal	O	N2	mux_mode_mlpg1	PE0201	10	1
LPG2	LED2 control signal	O	N7	arm_boot_mlpg2	PE0201	10	2

Table C–12. GSM-S I^2C

Signal	Designation	Type	Mode 2					Mode 3				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
GSM_I2C_SDA	I^2C data line	I/O	V6	mpu_i2c_sda	PS0201	5	0	C2	kbc_0	PE0201	13	2
GSM_I2C_SCK	I^2C clock line	I/O	W5	mpu_i2c_sck	PS0201	5	1	E2	kbr_0	PS1001	12	5

Table C–13. GSM-S GPIO (Modes 1 and 2)

Signal	Designation	Type	Mode 1					Mode 2				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
IO_GSM_0	Generic GSM IO signals	I/O	W11	tspact_3	PS0201	0	4	A20	cam_data_1	PE0201	11	6
IO_GSM_1	Generic GSM IO signals	I/O	V11	tspact_4	PS0201	0	5	B19	cam_data_2	PE0201	11	7
IO_GSM_2	Generic GSM IO signals	I/O	M4	rfen	PE0201	1	7	C18	cam_data_3	PE0201	12	0
								P15	sim_cd	PE0201	1	4
IO_GSM_3	Generic GSM IO signals	I/O	P3	clk_13m_req	PE0201	10	7					
IO_GSM_9	Generic GSM IO signals	I/O	W19	smc_rst	PE0201	9	4					

Table C–14. GSM-S GPIO (Modes 4 and 5)

Signal	Designation	Type	Mode 4					Mode 5				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
IO_GSM_0	Generic GSM IO signals	I/O	V4	cdi	PE0201	4	6					
IO_GSM_1	Generic GSM IO signals	I/O	W2	mclk	PE0201	4	7					
IO_GSM_2	Generic GSM IO signals	I/O						U4	mpu_spi1_sen1	PS0201	9	0
IO_GSM_3	Generic GSM IO signals	I/O						W1	mpu_spi1_sen2	PS0201	9	1
IO_GSM_9	Generic GSM IO signals	I/O										

Table C–15. MPU-S LCD (Mode 0)

Signal	Designation	Type	Mode 0				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
LCD_PXL_0	Pixel data out	O	G3	lcd_pixel_0	PE0201	4	3
LCD_PXL_1	Pixel data out	O	G2	lcd_pixel_1	PE0201	4	2
LCD_PXL_2	Pixel data out	O	K8	lcd_pixel_2	PS0201	4	2
LCD_PXL_3	Pixel data out	O	H4	lcd_pixel_3	PS0201	4	2
LCD_PXL_4	Pixel data out	O	G1	lcd_pixel_4	PS0201	4	2
LCD_PXL_5	Pixel data out	O	H2	lcd_pixel_5	PS0201	4	2
LCD_PXL_6	Pixel data out	O	H3	lcd_pixel_6	PS0201	4	2
LCD_PXL_7	Pixel data out	O	K7	lcd_pixel_7	PS0201	4	2
LCD_PXL_8	Pixel data out	O	J2	lcd_pixel_8	PS0201	4	2
LCD_PXL_9	Pixel data out	O	K3	lcd_pixel_9	PS0201	4	2
LCD_PXL_10	Pixel data out	O	L7	lcd_pixel_10	PS0201	4	1
LCD_PXL_11	Pixel data out	O	L4	lcd_pixel_11	PE0201	4	0
LCD_PXL_12	Pixel data out	O	L3	lcd_pixel_12	PE0201	3	7
LCD_PXL_13	Pixel data out	O	L1	lcd_pixel_13	PE0201	3	7
LCD_PXL_14	Pixel data out	O	M8	lcd_pixel_14	PE0201	3	7
LCD_PXL_15	Pixel data out	O	M7	lcd_pixel_15	PE0201	3	7
LCD_PXL_16	Pixel data out	O					
LCD_PXL_17	Pixel data out	O					
LCD_PCLK	Pixel clock	O	J4	lcd_pclk	PE0201	4	3
LCD_HSYNC	Horizontal synchro	O	J3	lcd_hsync	PE0201	4	3
LCD_VSYNC	Vertical synchro	O	K4	lcd_vsync	PE0201	4	4
LCD_AC	LCD bias	O	G4	lcd_ac	PE0201	4	4

Table C–16. MPU-S LCD (Modes 1 and 3)

Signal	Designation	Type	Mode 1					Mode 3				
			Ball	Pad Name	PU/ PD	Conf Reg	Reg. Field	Ball	Pad Name	PU/ PD	Conf Reg	Reg. Field
LCD_PXL_0	Pixel data out	O										
LCD_PXL_1	Pixel data out	O										
LCD_PXL_2	Pixel data out	O										
LCD_PXL_3	Pixel data out	O										
LCD_PXL_4	Pixel data out	O										
LCD_PXL_5	Pixel data out	O										
LCD_PXL_6	Pixel data out	O										
LCD_PXL_7	Pixel data out	O										
LCD_PXL_8	Pixel data out	O										
LCD_PXL_9	Pixel data out	O										
LCD_PXL_10	Pixel data out	O										
LCD_PXL_11	Pixel data out	O										
LCD_PXL_12	Pixel data out	O										
LCD_PXL_13	Pixel data out	O										
LCD_PXL_14	Pixel data out	O										
LCD_PXL_15	Pixel data out	O										
LCD_PXL_16	Pixel data out	O	Y4	mpu_ UART_tx_ ir2	PS0201	3	2					
LCD_PXL_17	Pixel data out	O	V5	mpu_ UART_ rx_ir2	PE0201	3	3					
LCD_PCLK	Pixel clock	O										
LCD_HSYNC	Horizontal synchro	O										
LCD_VSYNC	Vertical synchro	O										
LCD_AC	LCD bias	O										

Table C–17. MPU-S UART Modem–IrDA

Signal	Designation	Type	Mode 0					Mode 2				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
UART_TX2_IR	Infrared transmit pulse	O	Y4	mpu_uart_tx_ir2	PS0201	3	2	AA17	usb_dm		2	5
UART_RX2_IR	Infrared receive pulse	I	V5	mpu_uart_rx_ir2	PE0201	3	3	Y17	usb_dp		2	5
UART_SD2	IrDA transceiver shutdown mode	O	W4	mpu_uart_sd2	PS0201	3	4					
UART_TX2	Transmit data	O						R8	mpu_uart_tx1	PE0201	3	5
UART_RX2	Receive data	I						Y3	mpu_uart_rx1	PE0201	3	5
UART_CTS2	Clear to send	I						AA1	mpu_uart_cts1	PE0201	3	6
UART_RTS2	Request to send	O						AA2	mpu_uart_rts1	PE0201	3	6

Table C–18. MPU-S UART Modem (Mode 0)

Signal	Designation	Type	Mode 0				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
UART_TX1	Transmit data	O	R8	mpu_uart_tx1	PE0201	3	5
UART_RX1	Receive data	I	Y3	mpu_uart_rx1	PE0201	3	5
UART_CTS1	Clear to send	I	AA1	mpu_uart_cts1	PE0201	3	6
UART_RTS1	Request to send	O	AA2	mpu_uart_rts1	PE0201	3	6
UART_DCD1	Data carrier detect	I					
UART_DSR1	Data set ready	I					
UART_DTR1	Data transmit ready	O					

Table C–19. MPU-S UART Modem (Modes 1 and 3)

Signal	Designation	Type	Mode 1					Mode 3				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
UART_TX1	Transmit data	O	AA17	usb_dm		2	5					
UART_RX1	Receive data	I	Y17	usb_dp		2	5					
UART_CTS1	Clear to send	I	W16	usb_pu_en	PS0201	2	6					
UART_RTS1	Request to send	O	W17	usb_vbusi	PE0201	2	7					
UART_DCD1	Data carrier detect	I	V16	mclk_out	PS0201	3	0					
UART_DSR1	Data set ready	I	W18	creset	PE0201	3	1					
UART_DTR1	Data transmit ready	O						W1	mpu_spi1_sen2	PS0201	9	1

Table C–20. MPU-S SPI_100K 1 (Mode 0)

Signal	Designation	Type	Mode 0				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
MPU_SPI1_SCLK	Serial clock	O	U3	mpu_spi1_sclk	PE0201	8	4
MPU_SPI1_SDO	Output serial data	O	V2	mpu_spi1_sdo	PS0201	8	5
MPU_SPI1_SDI	Input serial data	I	T4	mpu_spi1_sdi	PE0201	8	6
MPU_SPI1_SEN0	Chip select enable 0	O	V3	mpu_spi1_sen0	PE0201	8	7
MPU_SPI1_SEN1	Chip select enable 1	O	U4	mpu_spi1_sen1	PS0201	9	0
MPU_SPI1_SEN2	Chip select enable 2	O	W1	mpu_spi1_sen2	PS0201	9	1

Table C–21. MPU-S SPI_100K 1 (Modes 1 and 4)

Signal	Designation	Type	Mode 1					Mode 4				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
MPU_SPI1_SCLK	Serial clock	O	W13	sdmc_clk	PE0201	2	2	R8	mpu_uart_tx1	PE0201	3	5
MPU_SPI1_SDO	Output serial data	O	R13	sdmc_cmd	PE0201	2	2	Y3	mpu_uart_rx1	PE0201	3	5
MPU_SPI1_SDI	Input serial data	I	V12	sdmc_dat_0	PE0201	2	2	AA1	mpu_uart_cts1	PE0201	3	6
MPU_SPI1_SEN0	Chip select enable 0	O	Y12	sdmc_dat_1	PE0201	2	2	AA2	mpu_uart_rts1	PE0201	3	6
MPU_SPI1_SEN1	Chip select enable 1	O	V13	sdmc_dat_2	PE0201	2	3					
MPU_SPI1_SEN2	Chip select enable 2	O	Y13	sdmc_dat_3	PE0201	2	4					

Table C–22. MPU-S SPI_100K 2

Signal	Designation	Type	Mode 2					Mode 4				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
MPU_SPI2_SCLK	Serial clock	O	M7	lcd_pixel_15	PE0201	3	7	N19	fadd_20	PS0201	6	1
								F2	kbr_4	PS1001	13	1
MPU_SPI2_SDO	Output serial data	O	M8	lcd_pixel_14	PE0201	3	7	N18	fadd_19	PS0201	6	2
								E4	kbc_2	PE0201	13	4
MPU_SPI2_SDI	Input serial data	I	L1	lcd_pixel_13	PE0201	3	7	N20	fadd_18	PS0201	6	3
								F4	kbc_3	PE0201	13	5
MPU_SPI2_SEN0	Chip select enable 0	O	L3	lcd_pixel_12	PE0201	3	7	M15	fadd_17	PS0201	6	4
								E3	kbc_4	PE0201	13	6
MPU_SPI2_SEN1	Chip select enable 1	O	L4	lcd_pixel_11	PE0201	4	0	P19	fadd_16	PS0201	6	5
								C2	kbc_0	PE0201	13	2
MPU_SPI2_SEN2	Chip select enable 2	O	L7	lcd_pixel_10	PS0201	4	1	D3	kbc_1	PE0201	13	3

Table C–23. MPU-S MMC/SDIO

Signal	Designation	Type	Mode 0				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
SDMC_CLK	MMC clock line	O	W13	sdmc_clk	PE0201	2	2
SDMC_CMD	MMC command line	I/O	R13	sdmc_cmd	PE0201	2	2
SDMC_DAT0	MMC data line	I/O	V12	sdmc_dat_0	PE0201	2	2
SDMC_DAT1	MMC data line	I/O	Y12	sdmc_dat_1	PE0201	2	2
SDMC_DAT2	MMC data line	I/O	V13	sdmc_dat_2	PE0201	2	3
SDMC_DAT3	MMC data line	I/O	Y13	sdmc_dat_3	PE0201	2	4

Table C–24. MPU-S I^2C (Mode 0)

Signal	Designation	Type	Mode 0				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
MPU_I2C_SDA	I^2C clock line	I/O	V6	mpu_i2c_sda	PS0201	5	0
MPU_I2C_SCK	I^2C data line	I/O	W5	mpu_i2c_sck	PS0201	5	1

Table C–25. MPU-S I^2C (Modes 1 and 3)

Signal	Designation	Type	Mode 1					Mode 3				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
MPU_I2C_SDA	I^2C clock line	I/O	E1	kbr_2	PS1001	12	7	V19	sim_pwrctrl	PS0201	1	3
MPU_I2C_SCK	I^2C data line	I/O	F3	kbr_3	PS1001	13	0	P15	sim_cd	PE0201	1	4

Table C–26. MPU-S HDQ 1-Wire

Signal	Designation	Type	Mode 1					Mode 2				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
HDQ1W		I/O	W4	mpu_uart_sd2	PS0201	3	4	T2	nemu1	PS0201	10	4

Table C–27. MPU-S μWire

Signal	Designation	Type	Mode 1					Mode 5				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
UW_SCLK	Transmit clock	O	J4	lcd_pclk	PE0201	4	3	V17	smc_clk	PS0201	9	3
UW_SDO	Serial output data	O	J3	lcd_ hsync	PE0201	4	3	W19	smc_rst	PE0201	9	4
UW_SDI	Serial receive data	I	G4	lcd_ac	PE0201	4	4	AA20	smc_io	PS0201	9	2
UW_nSCS1	Chip select 1	O	G3	lcd_ pixel_0	PE0201	4	3	V18	smc_cd	PE0201	9	5
UW_nSCS2	Chip select 2	O	K4	lcd_ vsync	PE0201	4	4	Y19	smc_ pwctrl	PE0201	9	6

Table C–28. MPU-S PWL

Signal	Designation	Type	Mode 1				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
PWL	Pulse width light modulator	O	T2	nemu1	PS0201	10	4

Table C–29. MPU-S PWT

Signal	Designation	Type	Mode 3				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
PWT	Pulse width tone modulator	O	T2	nemu1	PS0201	10	4

Table C–30. MPU-S LPG

Signal	Designation	Type	Mode 0				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
MLPG1	LED1 control signal	O	N2	mux_ mode_ mlpg1	PE0201	10	1
MLPG2	LED2 control signal	O	N7	arm_ boot_ mlpg2	PE0201	10	2

Table C–31. MPU-S Extended GPIO

Signal	Designation	Type	Mode 1					Mode 3				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
EXT_IO_0	Extended generic input/output	I/O	E2	kbr_0	PS1001	12	5	N2	mux_ mode_ mlpg1	PE0201	10	1
EXT_IO_1	Extended generic input/output	I/O	J7	kbr_1	PS1001	12	6	N7	arm_ boot_ mlpg2	PE0201	10	2
EXT_IO_2	Extended generic input/output	I/O	V6	mpu_ i2c_sda		5	0					
			C2	kbc_0	PE0201	13	2					
EXT_IO_3	Extended generic input/output	I/O	W5	mpu_ i2c_sck		5	1					
			D3	kbc_1	PE0201	13	3					

Table C–32. MPU-S EAC BT AuSPI (Mode 0)

Signal	Designation	Type	Mode 0				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
SCLK	Bluetooth port interface serial clock	I/O	W6	sclk	PE0201	2	1
SDO	Bluetooth port interface serial data output	O	R9	sdo	PS0201	2	1
SDI	Bluetooth port interface serial data input	I	Y6	sdi	PE0201	2	1
FSYNC	Bluetooth port interface frame synchro	I/O	Y5	fsync	PE0201	2	1
SEN1	Bluetooth port interface chip select enable	O					

Table C–33. MPU-S EAC BT AuSPI (Modes 1 and 4)

Signal	Designation	Type	Mode 1					Mode 4				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
SCLK	Bluetooth port interface serial clock	I/O	U3	mpu_ spi1_sclk	PE0201	8	4	C18	cam_ data_3	PE0201	12	0
SDO	Bluetooth port interface serial data output	O	V2	mpu_ spi1_sdo	PS0201	8	5	D16	cam_ data_4	PS0201	12	1
SDI	Bluetooth port interface serial data input	I	T4	mpu_ spi1_sdi	PE0201	8	6	C17	cam_ data_5	PE0201	12	2

Table C–33. MPU-S EAC BT AuSPI (Modes 1 and 4) (Continued)

Signal	Designation	Type	Mode 1					Mode 4				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
FSYNC	Bluetooth port interface frame synchro	I/O	V3	mpu_ spi1_ sen0	PE0201	8	7	B17	cam_ data_6	PE0201	12	3
SEN1	Bluetooth port interface chip select enable	O	U4	mpu_ spi1_ sen1	PS0201	9	0	A17	cam_ data_7	PS0201	12	4

Table C–34. MPU-S EAC Audio Codec

Signal	Designation	Type	Mode 0					Mode 2				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Con-fReg	Reg Field
CSCLK	Codec port interface serial clock	I/O	W3	csclk	PS0201	4	5					
CDO	Codec port interface serial data output	O	Y1	cdo	PS0201	4	5					
CDI	Codec port interface serial data input	I	V4	cdi	PE0201	4	6					
CSYNC	Codec port interface frame synchro	I/O	Y2	csync	PS0201	4	5					
CRESET	Asynchronous EAC reset. Active low	I	W18	creset	PE0201	3	1	W1	mpu_ spi1_ sen2	PS0201	9	1
MCLK	Codec master clock input	I	W2	mclk	PE0201	4	7					
MCLK_OUT	Codec master clock output	O	V16	mclk_out	PS0201	3	0					

Table C–35. MPU-S USB–OTG (Modes 0 and 1)

Signal	Designation	Type	Mode 0					Mode 1				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
USB_DP	USB differential (+) line	I/O	Y17	usb_dp		2	5					
USB_DM	USB differential (–) line	I/O	AA17	usb_dm		2	5					

Table C–35. MPU-S USB–OTG (Modes 0 and 1) (Continued)

Signal	Designation	Type	Mode 0					Mode 1				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
USB_TXD_VP	External USB transceiver data(+) signal	I/O										
USB_SEO_VM	External USB transceiver data(–) signal	I/O										
USB_TXD	External USB transceiver transmit signal	O										
USB_SEO	Drive single ended zero	O										
USB_VP	External USB transceiver VP function	I										
USB_VM	External USB transceiver VM function	I										
USB_VBUSI	Power supply of the USB bus	I	W17	usb_vbusi	PE0201	2	7	P4	mpu_ext_nirq	PS1001	9	7
USB_TXEN	External USB transceiver output enable	O										
USB_PU_EN	Pull_up enable	O	W16	usb_pu_en	PS0201	2	6					
USB_RCV	External USB transceiver received signal	I										
USB_SUSPEND	External receiver suspend signal	O										
USB_SPEED	External receiver speed signal	O										

Table C–36. MPU-S USB–OTG (Mode 3)

Signal	Designation	Type	Mode 3				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
USB_DP	USB differential (+) line	I/O					
USB_DM	USB differential (–) line	I/O					
USB_TXD_VP	External USB transceiver data(+) signal	I/O	Y17	usb_dp		2	5
USB_SEO_VM	External USB transceiver data(–) signal	I/O	AA17	usb_dm		2	5
USB_TXD	External USB transceiver transmit signal	O					
USB_SEO	Drive single ended zero	O					
USB_VP	External USB transceiver VP function	I	W18	creset	PE0201	3	1
USB_VM	External USB transceiver VM function	I	V16	mclk_out	PS0201	3	0
USB_VBUSI	Power supply of the USB bus	I					
USB_TXEN	External USB transceiver output enable	O	W17	usb_vbusi	PE0201	2	7
USB_PU_EN	Pull_up enable	O					
USB_RCV	External USB transceiver received signal	I	W16	usb_pu_en	PS0201	2	6
USB_SUSPEND	External receiver suspend signal	O					
USB_SPEED	External receiver speed signal	O					

Table C–37. MPU-S USB–OTG (Modes 4 and 5)

Signal	Designation	Type	Mode 4					Mode 5				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
USB_DP	USB differential (+) line	I/O										
USB_DM	USB differential (–) line	I/O										
USB_TXD_VP	External USB transceiver data(+) signal	I/O										

Table C–37. MPU-S USB–OTG (Modes 4 and 5) (Continued)

Signal	Designation	Type	Mode 4					Mode 5				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
USB_SEO_VM	External USB transceiver data(–) signal	I/O										
USB_TXD	External USB transceiver transmit signal	O	Y17	usb_dp		2	5					
USB_SEO	Drive single ended zero	O	AA17	usb_dm		2	5					
USB_VP	External USB transceiver VP function	I										
USB_VM	External USB transceiver VM function	I										
USB_VBUSI	Power supply of the USB bus	I										
USB_TXEN	External USB transceiver output enable	O										
USB_PU_EN	Pull_up enable	O										
USB_RCV	External USB transceiver received signal	I										
USB_SUSPEND	External receiver suspend signal	O						V6	mpu_i2c_sda		5	0
USB_SPEED	External receiver speed signal	O						W5	mpu_i2c_sck		5	1

Table C–38. MPU-S McBSP1 (Mode 1)

Signal	Designation	Type	Mode 1				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
CLKRX1	McBSP Rx/Tx clock	I/O	A21	cam_ rstz	PE0201	11	4
DX1	McBSP data transmit	O	G14	cam_ data_0	PS0201	11	5
DR1	McBSP data receive	I	A20	cam_ data_1	PE0201	11	6
FSRX1	McBSP frame synchro Rx/Tx	I/O	B19	cam_ data_2	PE0201	11	7
CLKS1	McBSP auxiliary clock	I	C18	cam_ data_3	PE0201	12	0

Table C–39. MPU-S McBSP1 (Modes 2 and 4)

Signal	Designation	Type	Mode 2					Mode 4				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
CLKRX1	McBSP Rx/ Tx clock	I/O	W3	csclk	PS0201	4	5	U3	mpu_ spi1_sclk	PE0201	8	4
DX1	McBSP data transmit	O	Y1	cdo	PS0201	4	5	V2	mpu_ spi1_sdo	PS0201	8	5
DR1	McBSP data receive	I	V4	cdi	PE0201	4	6	T4	mpu_spi 1_sdi	PE0201	8	6
FSRX1	McBSP frame synchro Rx/Tx	I/O	Y2	csync	PS0201	4	5	V3	mpu_ spi1_ sen0	PE0201	8	7
CLKS1	McBSP auxiliary clock	I	W2	mclk	PE0201	4	7	U4	mpu_ spi1_ sen1	PS0201	9	0

Table C–40. MPU-S McBSP2

Signal	Designation	Type	Mode 2				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
CLKRX2	McBSP Rx/Tx clock	I/O	J4	lcd_pclk	PE0201	4	3
DX2	McBSP data transmit	O	J3	lcd_hsync	PE0201	4	3
DR2	McBSP data receive	I	K4	lcd_vsync	PE0201	4	4
FSRX2	McBSP frame synchro Rx/Tx	I/O	G3	lcd_ pixel_0	PE0201	4	3
CLKS2	McBSP auxiliary clock	I	G4	lcd_ac	PE0201	4	4

Table C–41. MPU-S MCSI (Modes 1 and 2)

Signal	Designation	Type	Mode 1					Mode 2				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
MCSI_CLK	Bit synchronization clock	I/O	D16	cam_data_4	PS0201	12	1	W6	sclk	PE0201	2	1
								U3	mpu_spi1_sclk	PE0201	8	4
MCSI_FSYNCH	Frame synchronization clock or SS reset	I/O	A17	cam_data_7	PS0201	12	4	Y5	fsync	PE0201	2	1
								V3	mpu_spi1_sen0	PE0201	8	7
MCSI_RXD	Receive serial data	I	B17	cam_data_6	PE0201	12	3	Y6	sdi	PE0201	2	1
								T4	mpu_spi1_sdi	PE0201	8	6
MCSI_TXD	Transmit serial data	O	C17	cam_data_5	PE0201	12	2	R9	sdo	PS0201	2	1
								V2	mpu_spi1_sdo	PS0201	8	5

Table C–42. MPU-S MCSI (Mode 5)

Signal	Designation	Type	Mode 5				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
MCSI_CLK	Bit synchronization clock	I/O	W16	usb_pu_en	PS0201	2	6
MCSI_FSYNCH	Frame synchronization clock or SS reset	I/O	W18	creset	PE0201	3	1
MCSI_RXD	Receive serial data	I	V16	mclk_out	PS0201	3	0
MCSI_TXD	Transmit serial data	O	W17	usb_vbusi	PE0201	2	7

Table C–43. MPU-S SDRAM

Signal	Designation	Type	Mode 0				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
SADD_13	SDRAM address bus out	O	H11	sadd_13			
SADD_12	SDRAM address bus out	O	H9	sadd_12			

Table C–43. MPU-S SDRAM (Continued)

Signal	Designation	Type	Mode 0				
			Ball	Pad Name	PU/ PD	Conf Reg	Reg Field
SADD_11	SDRAM address bus out	O	H10	sadd_11			
SADD_10	SDRAM address bus out	O	B8	sadd_10			
SADD_9	SDRAM address bus out	O	B12	sadd_9			
SADD_8	SDRAM address bus out	O	G9	sadd_8			
SADD_7	SDRAM address bus out	O	G11	sadd_7			
SADD_6	SDRAM address bus out	O	G12	sadd_6			
SADD_5	SDRAM address bus out	O	B9	sadd_5			
SADD_4	SDRAM address bus out	O	G10	sadd_4			
SADD_3	SDRAM address bus out	O	A1	sadd_3			
SADD_2	SDRAM address bus out	O	B6	sadd_2			
SADD_1	SDRAM address bus out	O	B2	sadd_1			
SADD_0	SDRAM bank select	O	A2	sadd_0			
SBANK_1	SDRAM bank select	O	C3	sbank_1		5	3
SBANK_0	SDRAM data bus	O	B3	sbank_0			
SDATA_15	SDRAM data bus	I/O	C12	sdata_15			
SDATA_14	SDRAM data bus	I/O	D12	sdata_14			
SDATA_13	SDRAM data bus	I/O	D13	sdata_13			
SDATA_12	SDRAM data bus	I/O	C11	sdata_12			
SDATA_11	SDRAM data bus	I/O	C13	sdata_11			
SDATA_10	SDRAM data bus	I/O	D11	sdata_10			
SDATA_9	SDRAM data bus	I/O	D14	sdata_9			
SDATA_8	SDRAM data bus	I/O	C10	sdata_8			
SDATA_7	SDRAM data bus	I/O	D8	sdata_7			
SDATA_6	SDRAM data bus	I/O	C4	sdata_6			
SDATA_5	SDRAM data bus	I/O	C7	sdata_5			
SDATA_4	SDRAM data bus	I/O	D5	sdata_4			
SDATA_3	SDRAM data bus	I/O	D7	sdata_3			
SDATA_2	SDRAM data bus	I/O	C5	sdata_2			
SDATA_1	SDRAM data bus	I/O	C6	sdata_1			

Table C–43. MPU-S SDRAM (Continued)

Signal	Designation	Type	Mode 0				
			Ball	Pad Name	PU/ PD	Conf Reg	Reg Field
SDATA_0*	SDRAM data bus	I/O	D6	sdata_0			
SDCLK	SDRAM clock	O	C9	sdclk			
NSRAS	SDRAM row address strobe	O	H7	nsras			
NSCAS	SDRAM column address strobe	O	B4	nscas			
NSWE	SDRAM write enable	O	H8	nswe			
NSDQML	SDRAM low data byte mask	O	C8	nsdqml			
NSDQMU	SDRAM upper data byte mask	O	D10	nsdqmu			
nSDCS	SDRAM chip-select	O	G8	nsdcs		5	2
SDCLK_EN	SDRAM power-down control signal	O	H12	sdclk_en			

Table C–44. MPU-S DDR

Signal	Designation	Type	Mode 0				
			Ball	Pad Name	PU/PD	Conf. Reg.	Reg. Field
DQSH	Data strobe high	I/O	C14	dqsh		5	3
DQSL	Data Strobe Low	I/O	D4	dqsl		5	3
SDCLKX	Clock	O	D9	sdclkx		5	3

Table C–45. MPU-S EMIFS (Modes 0 and 1)

Signal	Designa-tion	Type	Mode 0					Mode 1				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
FDATA_0	Data bus	I/O	J18	fdata_0		7	0					
FDATA_1	Data bus	I/O	J20	fdata_1		7	0					
FDATA_2	Data bus	I/O	J15	fdata_2		7	0					
FDATA_3	Data bus	I/O	H19	fdata_3		7	0					
FDATA_4	Data bus	I/O	H18	fdata_4		7	0					
FDATA_5	Data bus	I/O	H15	fdata_5		7	0					
FDATA_6	Data bus	I/O	G20	fdata_6		7	0					
FDATA_7	Data bus	I/O	G19	fdata_7		7	0					
FDATA_8	Data bus	I/O	G18	fdata_8		7	0					
FDATA_9	Data bus	I/O	F20	fdata_9		7	0					

Table C–45. MPU-S EMIFS (Modes 0 and 1) (Continued)

Signal	Designation	Type	Mode 0					Mode 1				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
FDATA_10	Data bus	I/O	F19	fdata_10		7	0					
FDATA_11	Data bus	I/O	E20	fdata_11		7	0					
FDATA_12	Data bus	I/O	E19	fdata_12		7	0					
FDATA_13	Data bus	I/O	F18	fdata_13		7	0					
FDATA_14	Data bus	I/O	D19	fdata_14		7	0					
FDATA_15	Data bus	I/O	E18	fdata_15		7	0					
FADD_1	Address bus out	O	W21	fadd_1	UPS290	6	7					
FADD_2	Address bus out	O	U18	fadd_2	UPS290	6	7					
FADD_3	Address bus out	O	V20	fadd_3	UPS290	6	7					
FADD_4	Address bus out	O	U19	fadd_4	UPS290	6	7					
FADD_5	Address bus out	O	T18	fadd_5	UPS290	6	7					
FADD_6	Address bus out	O	U20	fadd_6	UPS290	6	7					
FADD_7	Address bus out	O	N15	fadd_7	UPS290	6	7					
FADD_8	Address bus out	O	T19	fadd_8	UPS290	6	7					
FADD_9	Address bus out	O	T20	fadd_9	UPS290	6	7					
FADD_10	Address bus out	O	R18	fadd_10	PS0201	6	7					
FADD_11	Address bus out	O	R19	fadd_11	PS0201	6	6					
FADD_12	Address bus out	O	R20	fadd_12	UPS290	6	6					
FADD_13	Address bus out	O	P18	fadd_13	PS0201	6	6					
FADD_14	Address bus out	O	R21	fadd_14	PS0201	6	6					
FADD_15	Address bus out	O	P20	fadd_15	PS0201	6	6					
FADD_16	Address bus out	O	P19	fadd_16	PS0201	6	5					

Table C–45. MPU-S EMIFS (Modes 0 and 1) (Continued)

Signal	Designation	Type	Mode 0					Mode 1				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
FADD_17	Address bus out	O	M15	fadd_17	PS0201	6	4					
FADD_18	Address bus out	O	N20	fadd_18	PS0201	6	3					
FADD_19	Address bus out	O	N18	fadd_19	PS0201	6	2					
FADD_20	Address bus out	O	N19	fadd_20	PS0201	6	1					
FADD_21	Address bus out	O	L14	fadd_21	PS0201	6	0					
FADD_22	Address bus out	O	M18	fadd_22	PS0201	5	7					
FADD_23	Address bus out	O	M19	fadd_23	PS0201	5	6					
FADD_24	Address bus out	O	L15	fadd_24	PS0201	5	5					
FADD_25	Address bus out	O	L18	fadd_25	PS0201	5	4	K14	nfcs_2	UPS290	7	1
NFCS_0	Chip selects active low	O										
NFCS_1	Chip selects active low	O	K18	nfcs_1	UPS290	7	2					
NFCS_2	Chip selects active low	O	K14	nfcs_2	UPS290	7	1					
NFCS_3	Chip selects active low	O	J19	nfcs_3	UPS290	7	3					
NFCS3_H	Chip selects active low	O										
NFCS3_L	Chip selects active low	O										
NFWE	Memory write enable	O	C19	nfwe	UPS290	7	4					
NFOE	Data output enable	O	D18	nfoe	UPS290	7	4					
NFWP	Flash write protect	O	B20	nfwp	UPS290	7	5					
NFBAA	Burst address advance for burst flash	O	C21	nfbaa	UPS290	7	5					

Table C–45. MPU-S EMIFS (Modes 0 and 1) (Continued)

Signal	Designation	Type	Mode 0					Mode 1				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
NFRST	Power down for TI/Reset for Intel flash	O	C20	nfrst	UPS290	8	3					
NFWAIT	Ready for TI/wait for Intel flash	I	B21	nfwait	PS0201	7	6					
NFADV	Address valid	O	L19	nfadv	UPS290	8	0					
FCLK	Clock for burst flash	O	J21	fclk		7	7					
NFBE_0	16-LSB byte low/high accesses enable	O	K15	nfbe_0	UPS290	8	2					
NFBE_1	16-LSB Byte low/ high accesses enable	O	K19	nfbe_1	UPS290	8	1					

Table C–46. MPU-S EMIFS (Modes 2 and 3)

Signal	Designation	Type	Mode 2					Mode 3				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
FDATA_0	Data bus	I/O										
FDATA_1	Data bus	I/O										
FDATA_2	Data bus	I/O										
FDATA_3	Data bus	I/O										
FDATA_4	Data bus	I/O										
FDATA_5	Data bus	I/O										
FDATA_6	Data bus	I/O										
FDATA_7	Data bus	I/O										
FDATA_8	Data bus	I/O										
FDATA_9	Data bus	I/O										
FDATA_10	Data bus	I/O										
FDATA_11	Data bus	I/O										
FDATA_12	Data bus	I/O										
FDATA_13	Data bus	I/O										

Table C–46. MPU-S EMIFS (Modes 2 and 3) (Continued)

Signal	Designation	Type	Mode 2					Mode 3				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
FDATA_14	Data bus	I/O										
FDATA_15	Data bus	I/O										
FADD_1	Address bus out	O										
FADD_2	Address bus out	O										
FADD_3	Address bus out	O										
FADD_4	Address bus out	O										
FADD_5	Address bus out	O										
FADD_6	Address bus out	O										
FADD_7	Address bus out	O										
FADD_8	Address bus out	O										
FADD_9	Address bus out	O										
FADD_10	Address bus out	O										
FADD_11	Address bus out	O										
FADD_12	Address bus out	O										
FADD_13	Address bus out	O										
FADD_14	Address bus out	O										
FADD_15	Address bus out	O										
FADD_16	Address bus out	O										
FADD_17	Address bus out	O										
FADD_18	Address bus out	O										
FADD_19	Address bus out	O										

Table C–46. MPU-S EMIFS (Modes 2 and 3) (Continued)

Signal	Designation	Type	Mode 2					Mode 3				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
FADD_20	Address bus out	O										
FADD_21	Address bus out	O										
FADD_22	Address bus out	O										
FADD_23	Address bus out	O										
FADD_24	Address bus out	O										
FADD_25	Address bus out	O										
NFCS_0	Chip selects active low	O	L19 L18	nfadv fadd_25	UPS290 PS0201	8 5	0 4	K19	nfbe_1	UPS290	8	1
NFCS_1	Chip selects active low	O										
NFCS_2	Chip selects active low	O										
NFCS_3	Chip selects active low	O										
NFCS3_H	Chip selects active low	O	K18	nfcs_1	UPS290	7	2					
NFCS3_L	Chip selects active low	O	J19	nfcs_3	UPS290	7	3					
NFWE	Memory write enable	O										
NFOE	Data output enable	O										
NFWP	Flash write protect	O										
NFBAA	Burst address advance for burst flash	O										
NFRST	Power down for TI/Reset for Intel flash	O										
NFWAIT	Ready for TI/wait for Intel flash	I										

Table C–46. MPU-S EMIFS (Modes 2 and 3) (Continued)

Signal	Designation	Type	Mode 2					Mode 3				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
NFADV	Address valid	O										
FCLK	Clock for burst flash	O										
NFBE_0	16-LSB byte low/high accesses enable	O										
NFBE_1	16-LSB Byte low/ high accesses enable	O										

Table C–47. MPU-S Compact Flash

Signal	Designation	Type	Mode 2				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
CF_nCD1	Card detect pin	I	W13	sdmc_ clk	PE0201	2	2
CF_nCD2	Card detect pin	I	R13	sdmc_ cmd	PE0201	2	2
CF_nIOIS16	I/O operation	I	V12	sdmc_ dat_0	PE0201	2	2
CF_INTREQ	Compact flash interrupt request	I	Y12	sdmc_ dat_1	PE0201	2	2
CF_RESET	Compact flash reset	O	V13	sdmc_ dat_2	PE0201	2	3

Table C–48. MPU-S NAND Flash

Signal	Designation	Type	Mode 1				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
I/O_0	In/out	I/O	T20	fadd_9	UPS290	6	7
I/O_1	In/out	I/O	U18	fadd_2	UPS290	6	7
I/O_2	In/out	I/O	V20	fadd_3	UPS290	6	7
I/O_3	In/out	I/O	U19	fadd_4	UPS290	6	7
I/O_4	In/out	I/O	T18	fadd_5	UPS290	6	7
I/O_5	In/out	I/O	U20	fadd_6	UPS290	6	7

Table C–48. MPU-S NAND Flash (Continued)

Signal	Designation	Type	Mode 1				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
I/O_6	In/out	I/O	N15	fadd_7	UPS290	6	7
I/O_7	In/out	I/O	R20	fadd_12	UPS290	6	6
WE	Write enable	O	P18	fadd_13	PS0201	6	6
RE	Read enable	O	W21	fadd_1	UPS290	6	7
WP	Write protect	O	T19	fadd_8	UPS290	6	7
CLE	Command latch enable	O	R21	fadd_14	PS0201	6	6
ALE	Address latch enable	O	P20	fadd_15	PS0201	6	6
CE1	Chip enable 1	O	R19	fadd_11	PS0201	6	6
CE2	Chip enable 2	O	R18	fadd_10	PS0201	6	7
RDY	Ready	I	P19	fadd_16	PS0201	6	5
			B21	nfwait	PS0201	7	6

Table C–49. MPU-S Camera

Signal	Designation	Type	Mode 0				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
CAM_EXCLK	Clock output for camera module	O	C15	cam_ exclk	PS0201	11	1
CAM_LCLK	Image data latch clock	O	D17	cam_lclk	PS0201	11	0
CAM_HS	Horizontal sync signal	O	C16	cam_hs	PS0201	11	2
CAM_VS	Vertical sync signal	O	D15	cam_vs	PS0201	11	3
CAM_RSTZ	Exclusive power supply for serial I/O ports	O	A21	cam_ rstz	PE0201	11	4
CAM_DATA_0	Digital image data	O	G14	cam_ data_0	PS0201	11	5
CAM_DATA_1	Digital image data	O	A20	cam_ data_1	PE0201	11	6
CAM_DATA_2	Digital image data	O	B19	cam_ data_2	PE0201	11	7
CAM_DATA_3	Digital image data	O	C18	cam_ data_3	PE0201	12	0
CAM_DATA_4	Digital image data	O	D16	cam_ data_4	PS0201	12	1

Table C–49. MPU-S Camera (Continued)

Signal	Designation	Type	Mode 0				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
CAM_DATA_5	Digital image data	O	C17	cam_ data_5	PE0201	12	2
CAM_DATA_6	Digital image data	O	B17	cam_ data_6	PE0201	12	3
CAM_DATA_7	Digital image data	O	A17	cam_ data_7	PS0201	12	4

Table C–50. MPU-S MPUIO

Signal	Designation	Type	Mode 2					Mode 4				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
ARMIO_0	Keypad row	I/O						D17	cam_lclk	PS0201	11	0
ARMIO_1	Keypad row	I/O						C15	cam_ exclk	PS0201	11	1
ARMIO_2	Keypad row	I/O						C16	cam_hs	PS0201	11	2
ARMIO_3	Keypad row	I/O	J7	kbr_1	PS1001	12	6					
ARMIO_4	Keypad row	I/O	E1	kbr_2	PS1001	12	7					

Table C–51. MPU-S Keypad (Modes 0 and 1)

Signal	Designation	Type	Mode 0					Mode 1				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
KBC_0	Keypad column	I	C2	kbc_0	PE0201	13	2					
KBC_1	Keypad column	I	D3	kbc_1	PE0201	13	3					
KBC_2	Keypad column	I	E4	kbc_2	PE0201	13	4					
KBC_3	Keypad column	I	F4	kbc_3	PE0201	13	5					
KBC_4	Keypad column	I	E3	kbc_4	PE0201	13	6					
KBC_5	Keypad column	I						Y1	cdo	PS0201	4	5
KBC_6	Keypad column	I						V4	cdi	PE0201	4	6
KBC_7	Keypad column	I										
KBR_0	Keypad row	I	E2	kbr_0	PS1001	12	5					

Table C–51. MPU-S Keypad (Modes 0 and 1) (Continued)

Signal	Designation	Type	Mode 0					Mode 1				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
KBR_1	Keypad row	I	J7	kbr_1	PS1001	12	6					
KBR_2	Keypad row	I	E1	kbr_2	PS1001	12	7					
KBR_3	Keypad row	I	F3	kbr_3	PS1001	13	0					
KBR_4	Keypad row	I	F2	kbr_4	PS1001	13	1					
KBR_5	Keypad row	I						Y2	csync	PS0201	4	5
KBR_6	Keypad row	I						W3	csclk	PS0201	4	5
KBR_7	Keypad row	I										

Table C–52. MPU-S Keypad (Modes 2 and 3)

Signal	Designation	Type	Mode 2					Mode 3				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
KBC_0	Keypad column	I										
KBC_1	Keypad column	I										
KBC_2	Keypad column	I										
KBC_3	Keypad column	I										
KBC_4	Keypad column	I										
KBC_5	Keypad column	I	D15	cam_vs	PS0201	11	3	W19	smc_rst	PE0201	9	4
KBC_6	Keypad column	I	A21	cam_rstz	PE0201	11	4	Y19	smc_pwctrl	PE0201	9	6
KBC_7	Keypad column	I	G14	cam_data_0	PS0201	11	5					
KBR_0	Keypad row	I										
KBR_1	Keypad row	I										
KBR_2	Keypad row	I										
KBR_3	Keypad row	I										
KBR_4	Keypad row	I										
KBR_5	Keypad row	I	D17	cam_lclk	PS0201	11	0	AA20	smc_io	PS0201	9	2

Table C–52. MPU-S Keypad (Modes 2 and 3) (Continued)

Signal	Designation	Type	Mode 2					Mode 3				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
KBR_6	Keypad row	I	C15	cam_ exclk	PS0201	11	1	V17	smc_clk	PS0201	9	3
KBR_7	Keypad row	I	C16	cam_ hs	PS0201	11	2					

Table C–53. MPU-S SMC

Signal	Designation	Type	Mode 0				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
SMC_IO	I/O	I/O	AA20	smc_io	PS0201	9	2
SMC_CLK	Clock	O	V17	smc_clk	PS0201	9	3
SMC_RST	Reset	O	W19	smc_rst	PE0201	9	4
SMC_CD	Card detect	I	V18	smc_cd	PE0201	9	5
SMC_PWCTRL	Power control	O	Y19	smc_ pwctrl	PE0201	9	6

Table C–54. MPU-S ETM9

Signal	Designation	Type	Mode 3					Mode 4				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
TRACE-SYNC	Trace sync signal		C15	cam_ exclk	PS0201	11	1					
TRACE-SYNCB	Trace sync signal							E1	kbr_2	PS1001	12	7
TRACECLK	Trace clock		D17	cam_lclk	PS0201	11	0					
PIPESTAT_0	Pipeline status		A21	cam_rstz	PE0201	11	4					
PIPESTAT_1	Pipeline status		D15	cam_vs	PS0201	11	3					
PIPESTAT_2	Pipeline status		C16	cam_hs	PS0201	11	2					
PIPESTAT_3	Pipeline status		P3	clk_ 13m_req	PE0201	10	7					
PIPESTAT_4	Pipeline status							J7	kbr_1	PS1001	12	6
PIPESTAT_5	Pipeline status							E2	kbr_0	PS1001	12	5

Table C–54. MPU-S ETM9 (Continued)

Signal	Designation	Type	Mode 3					Mode 4				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
TRACEPKT0	Data trace packet		G14	cam_data_0	PS0201	11	5					
TRACEPKT1	Data trace packet		A20	cam_data_1	PE0201	11	6					
TRACEPKT2	Data trace packet		B19	cam_data_2	PE0201	11	7					
TRACEPKT3	Data trace packet		C18	cam_data_3	PE0201	12	0					
TRACEPKT4	Data trace packet		D16	cam_data_4	PS0201	12	1					
TRACEPKT5	Data trace packet		C17	cam_data_5	PE0201	12	2					
TRACEPKT6	Data trace packet		B17	cam_data_6	PE0201	12	3					
TRACEPKT7	Data trace packet		A17	cam_data_7	PS0201	12	4					

Table C–55. MPU-S VLYNQ (Modes 1 and 2)

Signal	Designation	Type	Mode 1					Mode 2				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
VLYNQ_TXD0	Serial data output	O	R8	mpu_uart_tx1	PE0201	3	5	C17	cam_data_5	PE0201	12	2
								F3	kbr_3	PS1001	13	0
VLYNQ_RXD0	Serial data input	I	Y3	mpu_uart_rx1	PE0201	3	5	D16	cam_data_4	PS0201	12	1
								F2	kbr_4	PS1001	13	1
VLYNQ_RXD1	Serial data output	O	AA1	mpu_uart_cts1	PE0201	3	6	B17	cam_data_6	PE0201	12	3
								F4	kbc_3	PE0201	13	5
VLYNQ_TXD1	Serial data input	I	AA2	mpu_uart_rts1	PE0201	3	6	A17	cam_data_7	PS0201	12	4
								E3	kbc_4	PE0201	13	6
VLYNQ_CLK	Serial clock output	O	W2	mclk	PE0201	4	7					

Table C–56. MPU-S VLYNQ (Mode 5)

Signal	Designation	Type	Mode 5				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
VLYNQ_TXD0	Serial data output	O					
VLYNQ_RXD0	Serial data input	I					
VLYNQ_RXD1	Serial data output	O					
VLYNQ_TXD1	Serial data input	I					
VLYNQ_CLK	Serial clock output	O					

Table C–57. MPU-S TWL3016 Shared Port

Signal	Designation	Type	Mode 0				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
MCUDI	Serial data input	I	Y7	mcudi	PE0201	2	0
MCUDO	Serial data output	O	W7	mcudo	PS0201	2	0
MCUEN	Device enable	O	V7	mcuen	PS0201	2	0

Table C–58. MPU-S TAP (Mode 0)

Signal	Designation	Type	Mode 0				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
TDI	JTAG data input	I	R4	tdi	PE0201		
TDO	JTAG data output	O	R2	tdo			
TMS	JTAG mode select	I	R3	tms	PE0201		
$\overline{\text{TRST}}$	JTAG test reset (active low)	I	U2	ntrst	PE0201		
TCK	JTAG clock	I	P7	tck	PE0201		
RTCK	Return JTAG clock	O					

Table C–59. MPU-S TAP (Modes 2 and 4)

Signal	Designation	Type	Mode 2					Mode 4				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
TDI	JTAG data input	I										
TDO	JTAG data output	O										
TMS	JTAG mode select	I										
$\overline{\text{TRST}}$	JTAG test reset (active low)	I										
TCK	JTAG clock	I										
RTCK	Return JTAG clock	O	N7	arm_ boot_ mlpg2	PE0201	10	2					

Table C–60. MPU-S Test and Emulation

Signal	Designation	Type	Mode 0					Mode 5				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field	Ball	Pad Name	PU/PD	Conf Reg	Reg Field
NBSCAN	Boundary-scan selection	I	N4	nbscan	PS0201							
NEMU0	Test emulation pin 0	I/O	T3	nemu0	PS0201	10	3					
NEMU1	Test emulation pin 1	I/O	T2	nemu1	PS0201	10	4					
EXTERN0_ MPU	ARM9TDMI extern0 HW breakpoint	I						F3	kbr_3	PS1001	13	0
EXTERN1_ MPU	ARM9TDMI extern1 HW breakpoint	I						F2	kbr_4	PS1001	13	1
TEST_ MODE	Test mode	I	N3	test_ mode	PE0201	10	5					

Table C–61. MPU-S Debug

Signal	Designation	Type	Mode 5				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
DEBUG_0	Debug	I/O	A17	cam_data_7	PE0201	12	4
DEBUG_1	Debug	I/O	B17	cam_data_6	PE0201	12	3
DEBUG_2	Debug	I/O	C17	cam_data_5	PE0201	12	2
DEBUG_3	Debug	I/O	D16	cam_data_4	PS0201	12	1

Table C–61. MPU-S Debug (Continued)

Signal	Designation	Type	Mode 5				
			Ball	Pad Name	PU/PD	Conf Reg	Reg Field
DEBUG_4	Debug	I/O	C18	cam_data_3	PE0201	12	0
DEBUG_5	Debug	I/O	B19	cam_data_2	PE0201	11	7
DEBUG_6	Debug	I/O	A20	cam_data_1	PE0201	11	6
DEBUG_7	Debug	I/O	G14	cam_data_0	PS0201	11	5
DEBUG_8	Debug	I/O	A21	cam_rstz	PE0201	11	4
DEBUG_9	Debug	I/O	D15	cam_vs	PS0201	11	3
DEBUG_10	Debug	I/O	C16	cam_hs	PS0201	11	2
DEBUG_11	Debug	I/O	C15	cam_exclk	PS0201	11	1
DEBUG_12	Debug	I/O	D17	cam_lclk	PS0201	11	0

Table C–62. Miscellaneous (Modes 0 and 1)

Signal	Designation	Type	Mode 0					Mode 1				
			Ball	Pad Name	PU/PD	Conf. Reg.	Reg. Field	Ball	Pad Name	PU/PD	Conf. Reg.	Reg. Field
CLKTCXO	VCTXO input clock (13 MHz)	I	Y9	clktcxo								
CLK13M_OUT	CLKM output clock (13 MHz)	O	N1	clk13m_out								
OSC32K_IN	Input component signal of 32 kHz quartz	I	AA15	osc32k_in								
OSC32K_OUT	Output component signal of 32 kHz quartz	O	W15	osc32k_out								
ON_nOFF	Regulators activity	I	R14	on_noff								
NRESPWRON	Chip power-on reset	I	V15	nrespwron								
NRST_MPU	MPU reset	I	P2	mpu_nrst	PS1001							
IT_WAKEUP	Wake-up interrupt of real time clock	O	W14	it_wakeup		1	5					
TCXOEN	TCXO enable	O	R10	tcxoen	PE0201	1	6					
RFEN	External RF IC enable	O	M4	rfen	PE0201	1	7					
MPU_EXT_NIRQ	External MPU IRQ	I	P4	mpu_ext_nirq	PS1001	9	7					
GSM_EXT_NIRQ	External GSM IRQ	I	M3	gsm_ext_nirq	PS1001	10	0					
MUX_MODE_MLPG1		I	N2	mux_mode_mlpg1	PE0201	10	1					
ARM_BOOT_MLPG2		I	N7	arm_boot_mlpg2	PE0201	10	2					
EXT_ARM_NIRQ		I						AA20	smc_io	PS0201	9	2
EXT_DSP_NIRQ		I						V17	smc_clk	PS0201	9	3
IT_FRAME								V18	smc_cd	PE0201	9	5
XF								Y19	smc_pwctrl	PE0201	9	6
LOW_POWER								T3	nemu0	PS0201	10	3
NFIQ_PWRFAIL												
CLK32K	Digital 32-kHz output	O	V14	clk32k		10	6					
CLK48M_IN	CLKM input clock (48 MHz).	I										
CLK13M_IN	CLKM input clock (13 MHz).	I						W1	mpu_spi1_sen2	PS0201	9	1
CLK_13M_REQ	Clock 13-MHz request	I	P3	clk_13m_req	PE0201	10	7	L18	fadd_25	PS0201	5	4
CLKTCXO	VCTXO input clock (13 MHz)	I										
CLK13M_OUT	CLKM output clock (13 MHz)	O										
OSC32K_IN	Input component signal of 32 kHz quartz	I										

Table C–62. Miscellaneous (Modes 0 and 1) (Continued)

Signal	Designation	Type	Mode 0					Mode 1				
			Ball	Pad Name	PU/PD	Conf. Reg.	Reg. Field	Ball	Pad Name	PU/PD	Conf. Reg.	Reg. Field
OSC32K_OUT	Output component signal of 32 kHz quartz	O										
ON_nOFF	Regulators activity	I										
NRESPWRON	Chip power-on reset	I										
NRST_MPU	MPU reset	I										
IT_WAKEUP	Wake-up interrupt of real time clock	O										
TCXOEN	TCXO enable	O										
RFEN	External RF IC enable	O										
MPU_EXT_NIRQ	External MPU IRQ	I										
GSM_EXT_NIRQ	External GSM IRQ	I										
MUX_MODE_MLPG1		I										
ARM_BOOT_MLPG2		I										
EXT_ARM_NIRQ		I										
EXT_DSP_NIRQ		I										
IT_FRAME												
XF												
LOW_POWER												
NFIQ_PWRFAIL								N2	mux_mode_ mlpg1	PE0201	10	1
CLK32K	Digital 32-kHz output	O										
CLK48M_IN	CLKM input clock (48 MHz)	I	T3	nemu0	PS0201	10	3					
CLK13M_IN	CLKM input clock (13 MHz)	I										
CLK_13M_REQ	Clock 13-MHz request	I										

Table C–63. Miscellaneous (Mode 5)

Signal	Designation	Type	Mode 5				
			Ball	Pad Name	PU/PD	Conf. Reg.	Reg. Field
CLKTCXO	VCTXO input clock (13 MHz)	I					
CLK13M_OUT	CLKM output clock (13 MHz)	O					
OSC32K_IN	Input component signal of 32 kHz quartz	I					
OSC32K_OUT	Output component signal of 32 kHz quartz	O					
ON_nOFF	Regulators activity	I					
NRESPWRON	Chip power-on reset	I					
NRST_MPU	MPU reset	I					
IT_WAKEUP	Wake-up interrupt of real time clock	O					
TCXOEN	TCXO enable	O					
RFEN	External RF IC enable	O					
MPU_EXT_NIRQ	External MPU IRQ	I					
GSM_EXT_NIRQ	External GSM IRQ	I					
MUX_MODE_MLPG1		I					
ARM_BOOT_MLPG2		I					
EXT_ARM_NIRQ		I					
EXT_DSP_NIRQ		I					
IT_FRAME							
XF							
LOW_POWER							
NFIQ_PWRFAIL							
CLK32K	Digital 32-kHz output	O					
CLK48M_IN	CLKM input clock (48 MHz)	I					
CLK13M_IN	CLKM input clock (13 MHz)	I	W2	mclk	PE0201	4	7
CLK_13M_REQ	Clock 13-MHz request	I					

Table C–64. Power

Pin	Description	Balls
VDDOMAP	OMAP logic	A9 M2 B1
VDD	OMAP730 logic	A3 B15 B18 H20 N21 AA19 AA11 AA3 U1 D2
VDDDLL	DDR DLL	A15
VDDLMM	DBB DSP	B16 D20
VDDGSM	DBB logic	A13 G21 L21
VDDRTC	RTC logic	Y14
VDDLDO1	Output of embedded LDO for OMAP DPLLs	K2
VDDLDO2	Output of embedded LDO for DBB DPLL	K20
VPP	eFuse supply	AA21
VDDA	Slicer and APLLs	W9
VDDSHV1	SDRAM interface	A5 A7 B10 B14
VDDSHV2	Camera interface	A19
VDDSHV3	Flash interface	E21 M20 U21
VDDSHV4	SIM/SMC interface	Y21
VDDSHV5	RTC backup I/O	Y16
VDDSHV6	MMC interface	AA13
VDDSHV7	USB interface	Y18
VDDSHV8	Global I/Os	AA5 R1
VDDSHV9	LCD interface	J1 C1
VDDSHV10	Tspact interface	Y10
VSS	Common ground	B5 B7 A11 B13 G13 H13 H14 J14 M14 P14 P13 P10 P9 P8 N8 L8 J8
VSSA	Analog ground	V9
VSS32K	Ground for Osc32k	Y15

C.2 Pin Description by Pad Name

Table C–65. Pin Description by Pad Name

Ball	Type	Pin not Muxed or Mode 0 (Primary Function)	Input Pad Type	Output Pad Type	Pull Down
N7	I/O	ARM_boot_MLPG2	IDI091	OUK431	PE0201
Y8	I	BDR	IDI041		PE0201
V8	I/O	BDX	IDI041	OUK431	PS0201
W8	I/O	BFSR	IDI041	OUK431	PE0201
AA7	I/O	BFSX	IDI041	OUK431	PS0201
G14	I/O	CAM_DATA_0	IDI041	OUK431	PS0201
A20	I/O	CAM_DATA_1	IDI041	OUI831	PE0201
B19	I/O	CAM_DATA_2	IDI041	OUI831	PE0201
C18	I/O	CAM_DATA_3	IDI041	OUI831	PE0201
D16	I/O	CAM_DATA_4	IDI041	OUK431	PS0201
C17	I/O	CAM_DATA_5	IDI041	OUI831	PE0201
B17	I/O	CAM_DATA_6	IDI041	OUK431	PE0201
A17	I/O	CAM_DATA_7	IDI041	OUI431	PS0201
C15	I/O	CAM_EXCLK	IDI041	OUI431	PS0201
C16	I/O	CAM_HS	IDI041	OUI831	PS0201
D17	I/O	CAM_LCLK	IDI091	OUI831	PS0201
A21	I/O	CAM_RSTZ	IDI041	OUI831	PE0201
D15	I/O	CAM_VS	IDI041	OUI831	PS0201
V4	I/O	CDI	IDI041	OUK431	PE0201
Y1	I/O	CDO	IDI041	OUK431	PS0201
P3	I/O	CLK_13M_REQ	IDI091	OUK431	PE0201
N1	O	CLK13M_OUT		OUI831	
V14	O	CLK32K		OUK431	
Y9	I	CLKTCXO	UC584		
W18	I/O	CRESET	IDI091	OUK431	PE0201
W3	I/O	CSCLK	IDI091	OUK431	PS0201
Y2	I/O	CSYNC	IDI041	OUK431	PS0201
C14	I	DQSH	UIS512	UIS512	
D4	O	DQSL	UOS376	U0S376	
W21	I/O	FADD_1	UIS512	UOS334	UPS290
R18	I/O	FADD_10	UIS512	UOS334	PS0201
R19	I/O	FADD_11	UIS512	UOS334	PS0201
R20	I/O	FADD_12	UIS512	UOS334	UPS290
P18	I/O	FADD_13	UIS512	UOS334	PS0201
R21	I/O	FADD_14	UIS512	UOS334	PS0201
P20	I/O	FADD_15	UIS512	UOS334	PS0201
P19	I/O	FADD_16	UIS512	UOS334	PS0201

Table C–65. Pin Description by Pad Name (Continued)

Ball	Type	Pin not Muxed or Mode 0 (Primary Function)	Input Pad Type	Output Pad Type	Pull Down
M15	I/O	FADD_17	UIS512	UOS334	PS0201
N20	I/O	FADD_18	UIS512	UOS334	PS0201
N18	I/O	FADD_19	UIS512	UOS334	PS0201
U18	I/O	FADD_2	UIS512	UOS334	UPS290
N19	I/O	FADD_20	UIS512	UOS334	PS0201
L14	I/O	FADD_21	UIS512	UOS334	PS0201
M18	I/O	FADD_22	UIS512	UOS334	PS0201
M19	I/O	FADD_23	UIS512	UOS334	PS0201
L15	I/O	FADD_24	UIS512	UOS334	PS0201
L18	I/O	FADD_25	UIS512	UOS334	PS0201
V20	I/O	FADD_3	UIS512	UOS334	UPS290
U19	I/O	FADD_4	UIS512	UOS334	UPS290
T18	I/O	FADD_5	UIS512	UOS334	UPS290
U20	I/O	FADD_6	UIS512	UOS334	UPS290
N15	I/O	FADD_7	UIS512	UOS334	UPS290
T19	I/O	FADD_8	UIS512	UOS334	UPS290
T20	I/O	FADD_9	UIS512	UOS334	UPS290
J21	I/O	FCLK	UIS512	UOS373	UPS290
J18	I/O	FDATA_0	UIS512	UOS373	UPS290
J20	I/O	FDATA_1	UIS512	UOS373	UPS290
F19	I/O	FDATA_10	UIS512	UOS373	UPS290
E20	I/O	FDATA_11	UIS512	UOS373	UPS290
E19	I/O	FDATA_12	UIS512	UOS373	UPS290
F18	I/O	FDATA_13	UIS512	UOS373	UPS290
D19	I/O	FDATA_14	UIS512	UOS373	UPS290
E18	I/O	FDATA_15	UIS512	UOS373	UPS290
J15	I/O	FDATA_2	UIS512	UOS373	UPS290
H19	I/O	FDATA_3	UIS512	UOS373	UPS290
H18	I/O	FDATA_4	UIS512	UOS373	UPS290
H15	I/O	FDATA_5	UIS512	UOS373	UPS290
G20	I/O	FDATA_6	UIS512	UOS373	UPS290
G19	I/O	FDATA_7	UIS512	UOS373	UPS290
G18	I/O	FDATA_8	UIS512	UOS373	UPS290
F20	I/O	FDATA_9	UIS512	UOS373	UPS290
Y5	I/O	FSYNC	IDI041	OUK431	PE0201
M3	I	GSM_EXT_NIRQ	IDI091		PS1001
W14	O	IT_WAKEUP		OUK431	
C2	I/O	KBC_0	IDI091	OUK431	PE0201

Table C–65. Pin Description by Pad Name (Continued)

Ball	Type	Pin not Muxed or Mode 0 (Primary Function)	Input Pad Type	Output Pad Type	Pull Down
D3	I/O	KBC_1	IDI091	OUK431	PE0201
E4	I/O	KBC_2	IDI091	OUK431	PE0201
F4	I/O	KBC_3	IDI091	OUK431	PE0201
E3	I/O	KBC_4	IDI091	OUI431	PE0201
E2	I/O	KBR_0	IDI091	OUI431	PS1001
J7	I/O	KBR_1	IDI091	OUI431	PS1001
E1	I/O	KBR_2	IDI091	OUK431	PS1001
F3	I/O	KBR_3	IDI091	OUI431	PS1001
F2	I/O	KBR_4	IDI091	OUK431	PS1001
G4	I/O	LCD_AC	IDI041	OUK431	PE0201
J3	I/O	LCD_HSYNC	IDI041	OUK431	PE0201
J4	I/O	LCD_PCLK	IDI041	OUK431	PE0201
G3	I/O	LCD_PIXEL_0	IDI041	OUK431	PE0201
G2	I/O	LCD_PIXEL_1	IDI041	OUK431	PE0201
L7	I/O	LCD_PIXEL_10	IDI041	OUK431	PS0201
L4	I/O	LCD_PIXEL_11	IDI041	OUK431	PE0201
L3	I/O	LCD_PIXEL_12	IDI041	OUK431	PE0201
L1	I/O	LCD_PIXEL_13	IDI041	OUK431	PE0201
M8	I/O	LCD_PIXEL_14	IDI041	OUK431	PE0201
M7	I/O	LCD_PIXEL_15	IDI041	OUI831	PE0201
K8	I/O	LCD_PIXEL_2	IDI041	OUK431	PS0201
H4	I/O	LCD_PIXEL_3	IDI041	OUK431	PS0201
G1	I/O	LCD_PIXEL_4	IDI041	OUK431	PS0201
H2	I/O	LCD_PIXEL_5	IDI041	OUK431	PS0201
H3	I/O	LCD_PIXEL_6	IDI041	OUK431	PS0201
K7	I/O	LCD_PIXEL_7	IDI041	OUK431	PS0201
J2	I/O	LCD_PIXEL_8	IDI041	OUK431	PS0201
K3	I/O	LCD_PIXEL_9	IDI041	OUK431	PS0201
K4	I/O	LCD_VSYNC	IDI041	OUK431	PE0201
W2	I/O	MCLK	IDI091	OUI831	PE0201
V16	I/O	MCLK_OUT	IDI091	OUK431	PS0201
Y7	I	MCUDI	IDI041		PE0201
W7	I/O	MCUDO	IDI041	OUK431	PS0201
V7	I/O	MCUEN	IDI041	OUK431	PS0201
P4	I	MPU_EXT_NIRQ	IDI091		PS1001
W5	I/O	MPU_I2C_SCK	UIS513	UOS381	PS0201
V6	I/O	MPU_I2C_SDA	UIS513	UOS381	PS0201
P2	I	MPU_NRST	IDI091		PS1001

Table C–65. Pin Description by Pad Name (Continued)

Ball	Type	Pin not Muxed or Mode 0 (Primary Function)	Input Pad Type	Output Pad Type	Pull Down
U3	I/O	MPU_SPI1_SCLK	IDI041	OUK431	PE0201
T4	I/O	MPU_SPI1_SDI	IDI041	OUK431	PE0201
V2	I/O	MPU_SPI1_SDO	IDI041	OUK431	PS0201
V3	I/O	MPU_SPI1_SEN0	IDI041	OUK431	PE0201
U4	I/O	MPU_SPI1_SEN1	IDI041	OUK431	PS0201
W1	I/O	MPU_SPI1_SEN2	IDI091	OUK431	PS0201
AA1	I/O	MPU_UART_CTS1	IDG091	OUM431	PE0201
AA2	I/O	MPU_UART_RTS1	IDG041	OUM831	PE0201
V5	I/O	MPU_UART_RX_IR2	IDI091	OUK431	PE0201
Y3	I/O	MPU_UART_RX1	IDG041	OUM431	PE0201
W4	I/O	MPU_UART_SD2	IDI091	OUK431	PS0201
Y4	I/O	MPU_UART_TX_IR2	IDI091	OUK431	PS0201
R8	I/O	MPU_UART_TX1	IDG091	OUM831	PE0201
N2	I/O	Mux_mode_MLPG1	IDI091	OUK431	PE0201
N4	I	NBSCAN	IDI091		PS0201
T3	I/O	NEMU0	IDI091	OUK431	PS0201
T2	I/O	NEMU1	IDI091	OUK431	PS0201
L19	I/O	NFADV	UIS512	UOS334	UPS290
C21	I/O	NFBAA	UIS512	UOS334	UPS290
K15	I/O	NFBE_0	UIS512	UOS334	UPS290
K19	I/O	NFBE_1	UIS512	UOS334	UPS290
K18	I/O	NFCS_1	UIS512	UOS334	UPS290
K14	I/O	NFCS_2	UIS512	UOS334	UPS290
J19	I/O	NFCS_3	UIS512	UOS334	UPS290
D18	I/O	NFOE	UIS512	UOS334	UPS290
C20	I/O	NFRST	UIS512	UOS334	UPS290
B21	I/O	NFWAIT	UIS512	UOS334	PS0201
C19	I/O	NFWE	UIS512	UOS334	UPS290
B20	I/O	NFWP	UIS512	UOS334	UPS290
V15	I	NRESPWRON	IDI091		
B4	O	NSCAS		UOS375	
G8	I/O	NSDCS	UIS512	UOS375	
C8	O	NSDQML		UOS375	
D10	O	DSDQMU		UOS375	
H7	O	NSRAS		UOS375	
H8	P	NSWE		UOS375	
U2	I	nTRST	IDG091		PE0201
R14	I	ON_nOFF	IDI091		

Table C–65. Pin Description by Pad Name (Continued)

Ball	Type	Pin not Muxed or Mode 0 (Primary Function)	Input Pad Type	Output Pad Type	Pull Down
AA15	I	OSC32K_IN	OS11D1		
W15	O	OSC32K_OUT		OS11V1	
M4	I/O	RFEN	IDI041	OUI831	PE0201
A2	O	SADD_0		UOS376	
B2	O	SADD_1		UOS376	
B8	O	SADD_10		UOS376	
H10	O	SADD_11		UOS376	
H9	O	SADD_12		UOS376	
H11	O	SADD_13		UOS376	
B6	O	SADD_2		UOS376	
A1	O	SADD_3		UOS376	
G10	O	SADD_4		UOS376	
B9	O	SADD_5		UOS376	
G12	O	SADD_6		UOS376	
G11	O	SADD_7		UOS376	
G9	O	SADD_8		UOS376	
B12	O	SADD_9		UOS376	
B3	O	SBANK_0		UOS376	
C3	O	SBANK_1		UOS376	
W6	I/O	SCLK	IDI041	OUK431	PE0201
D6	I/O	SDATA_0	UIS512	UOS373	
C6	I/O	SDATA_0	UIS512	UOS373	
D11	I/O	SDATA_0	UIS512	UOS373	
C13	I/O	SDATA_0	UIS512	UOS373	
C11	I/O	SDATA_0	UIS512	UOS373	
D13	I/O	SDATA_0	UIS512	UOS373	
D12	I/O	SDATA_0	UIS512	UOS373	
C12	I/O	SDATA_0	UIS512	UOS373	
C5	I/O	SDATA_0	UIS512	UOS373	
D7	I/O	SDATA_0	UIS512	UOS373	
D5	I/O	SDATA_0	UIS512	UOS373	
C7	I/O	SDATA_0	UIS512	UOS373	
C4	I/O	SDATA_0	UIS512	UOS373	
D8	I/O	SDATA_0	UIS512	UOS373	
C10	O	SDATA_0	UIS512	UOS373	
D14	O	SDATA_0	UIS512	UOS373	
C9	I/O	SDCLK	UIS512	UOS373	
H12	O	SDCLK_EN		UOS375	

Table C–65. Pin Description by Pad Name (Continued)

Ball	Type	Pin not Muxed or Mode 0 (Primary Function)	Input Pad Type	Output Pad Type	Pull Down
D9	IO	SDCLKX	UIS512	UOS376	
Y6	I	SDI	IDI041		PE0201
W13	IO	SDMC_CLK	IDI091	OUK431	PE0201
R13	IO	SDMC_CMD	IDI091	OUK431	PE0201
V12	IO	SDMC_DAT_0	IDI091	OUK431	PE0201
Y12	IO	SDMC_DAT_1	IDI091	OUK431	PE0201
V13	IO	SDMC_DAT_2	IDI091	OUK431	PE0201
Y13	IO	SDMC_DAT_3	IDI091	OUK431	PE0201
R9	IO	SDO	IDI041	OUK431	PS0201
P15	IO	SIM_CD	IDI041	OUI831	PE0201
W20	IO	SIM_CLK	IDI041	OUI831	PS0201
N14	IO	SIM_IO	IDI041	UOS372	PS0201
V19	IO	SIM_PWRCTRL	IDI041	OUI831	PS0201
Y20	IO	SIM_RST	IDI041	OUI831	PE0201
V18	IO	SMC_CD	IDI041	OUI831	PE0201
V17	IO	SMC_CLK	IDI041	OUI831	PS0201
AA20	IO	SMC_IO	IDI041	UOS372	PS0201
Y19	IO	SMC_PWCTRL	IDI041	OUI831	PE0201
W19	IO	SMC_RST	IDI041	OUI831	PE0201
P7	I	TCK	IDG091		PE0201
R10	IO	TCXOEN	IDI091	OUI831	PE0201
R4	I	TDI	IDG091		PE0201
R2	O	TDO		OUM831	
N3	IO	TEST_MODE	IDI091	OUK431	PE0201
R3	I	TMS	IDG091		PE0201
W12	IO	TSPACT_0	IDI041	OUO431	PS0201
R12	IO	TSPACT_1	IDI041	OUO431	PS0201
P12	IO	TSPACT_2	IDI041	OUO431	PS0201
W11	IO	TSPACT_3	IDI041	OUO431	PS0201
V11	IO	TSPACT_4	IDI041	OUO431	PS0201
V10	IO	TSPCLKX	IDI041	OUI831	PS0201
R11	IO	TSPDO	IDI041	OUK431	PE0201
AA9	IO	TSPEN_0	IDI041	OUK431	PE0201
P11	IO	TSPEN_1	IDI041	OUK431	PE0201
W10	IO	TSPEN_2	IDI041	OUK431	PE0201
AA17	IO	USB_DM	UOS387	UOS387	
Y17	IO	USB_DP	UOS387	UOS387	
W16	IO	USB_PU_EN	IDI041	OUI831	PS0201

Table C–65. Pin Description by Pad Name (Continued)

Ball	Type	Pin not Muxed or Mode 0 (Primary Function)	Input Pad Type	Output Pad Type	Pull Down
W17	I/O	USB_VBUSI	IDI041	OUI831	PE0201
A9	PWR	VCCP			
B1	PWR	VDD			
B15	PWR	VDD			
B18	PWR	VDD			
H20	PWR	VDD			
N21	PWR	VDD			
AA19	PWR	VDD			
AA11	PWR	VDD			
AA3	PWR	VDD			
U1	PWR	VDD			
M2	PWR	VDD			
W9	PWR	VDDA			
A15	PWR	VDDDLL			
A13	PWR	VDDGSM			
G21	PWR	VDDGSM			
L21	PWR	VDDGSM			
K2	PWR	VDDLDOOMAP			
K20	PWR	VDDLDOGSM			
B16	PWR	VDDLMM			
D20	PWR	VDDLMM			
A3	PWR	VDDOMAP			
F2	PWR	VDDOMAP			
Y14	PWR	VDDRTC			
A5	PWR	VDDSHV1			
A7	PWR	VDDSHV1			
B10	PWR	VDDSHV1			
B14	PWR	VDDSHV1			
A19	PWR	VDDSHV2			
E21	PWR	VDDSHV3			
M20	PWR	VDDSHV3			
U21	PWR	VDDSHV3			
Y21	PWR	VDDSHV4			
Y16	PWR	VDDSHV5			
AA13	PWR	VDDSHV6			
Y18	PWR	VDDSHV7			
AA5	PWR	VDDSHV8			
R1	PWR	VDDSHV8			

Table C–65. Pin Description by Pad Name (Continued)

Ball	Type	Pin not Muxed or Mode 0 (Primary Function)	Input Pad Type	Output Pad Type	Pull Down
J1	PWR	VDDSHV9			
C1	PWR	VDDSHV9			
Y10	PWR	VDDSHV10			
AA21	PWR	VPP			
B5	PWR	VSS			
B7	PWR	VSS			
A11	PWR	VSS			
B13	PWR	VSS			
G13	PWR	VSS			
H13	PWR	VSS			
H14	PWR	VSS			
J14	PWR	VSS			
M14	PWR	VSS			
P14	PWR	VSS			
P13	PWR	VSS			
P10	PWR	VSS			
P9	PWR	VSS			
P8	PWR	VSS			
N8	PWR	VSS			
L8	PWR	VSS			
J8	PWR	VSS			
Y15	PWR	VSS32K			
V9	PWR	VSSA			

C.3 Pin Multiplexing

Table C–66. Pin Multiplexing

Ball	Type	Pin not muxed or Mode 0 (Primary function)		Mode 1 function		Mode 2 function		Mode 3 function		Mode 4 function		Mode 5 function		Mode 6 function (MPU GPIO)		Reset
V10	IO	TPU/TSP PORT	TSPCLKX											MPU GPIO	GPIO_0	mode6
W12	IO	TPU/TSP PORT	TSPACT_0											MPU GPIO	GPIO_1	mode6
R12	IO	TPU/TSP PORT	TSPACT_1											MPU GPIO	GPIO_2	mode6
P12	IO	TPU/TSP PORT	TSPACT_2											MPU GPIO	GPIO_3	mode6
W11	IO	TPU/TSP PORT	TSPACT_3	GSM GPIO	IO_ GSM_0									MPU GPIO	GPIO_4	mode6
V11	IO	TPU/TSP PORT	TSPACT_4	GSM GPIO	IO_ GSM_1									MPU GPIO	GPIO_5	mode6
R11	IO	TPU/TSP PORT	TSPDO											MPU GPIO	GPIO_6	mode6
W10	IO	TPU/TSP PORT	TSPEN_2											MPU GPIO	GPIO_7	mode6
P11	IO	TPU/TSP PORT	TSPEN_1											MPU GPIO	GPIO_8	mode6
AA9	IO	TPU/TSP PORT	TSPEN_0											MPU GPIO	GPIO_9	mode6
W8	IO	GSM BB I/F	BFSR											MPU GPIO	GPIO_10	mode6
Y8	I	GSM BB I/F	BDR											MPU GPIO	GPIN_1	mode6
AA7	IO	GSM BB I/F	BFSX											MPU GPIO	GPIO_11	mode6
V8	IO	GSM BB I/F	BDX											MPU GPIO	GPIO_12	mode6
N14	IO	GSM SIM	SIM_IO											MPU GPIO	GPIO_13	mode6
W20	IO	GSM SIM	SIM_CLK											MPU GPIO	GPIO_14	mode6
V19	IO	GSM SIM	SIM_ PWRCTRL					MPU I2C	MPU_ I2C_SDA					MPU GPIO	GPIO_15	mode6
P15	IO	GSM SIM	SIM_CD			GSM GPIO	IO_ GSM_2	MPU I2C	MPU_ I2C_SCK					MPU GPIO	GPIO_16	mode6
Y20	IO	GSM SIM	SIM_RST											MPU GPIO	GPIO_17	mode6
W14	O	Power mngt	IT_WAKEUP													mode0
R10	IO	CLK mngt	TCXOEN											MPU GPIO	GPIO_18	mode0
M4	IO	CLK mngt	RFEN	GSM GPIO	IO_ GSM_2									MPU GPIO	GPIO_19	mode0
R14	I	Power mngt	ON_nOFF													mode0
Y7	I	Syren SPI	MCUDI											MPU GPIO	GPIN_2	mode6
W7	IO	Syren SPI	MCUDO											MPU GPIO	GPIO_20	mode6
V7	IO	Syren SPI	MCUEN											MPU GPIO	GPIO_21	mode6

Table C–66. Pin Multiplexing (Continued)

Ball	Type	Pin not muxed or Mode 0 (Primary function)		Mode 1 function		Mode 2 function		Mode 3 function		Mode 4 function		Mode 5 function		Mode 6 function (MPU GPIO)		Reset
W6	IO	EAC BT auSPI port	SCLK	GSM Voice I/F	VCLKRX	MPU MCSI	MPU_ MCSI_ CLK							MPU GPIO	GPIO_22	mode6
R9	IO	EAC BT auSPI port	SDO	GSM Voice I/F	VDX	MPU MCSI	MPU_ MCSI_ TXD							MPU GPIO	GPIO_23	mode6
Y6	I	EAC BT auSPI port	SDI	GSM Voice I/F	VDR	MPU MCSI	MPU_ MCSI_ RXD							MPU GPIO	GPIN_3	mode6
Y5	IO	EAC BT auSPI port	FSYNC	GSM Voice I/F	VFSRX	MPU MCSI	MPU_ MCSI_ FSYNCH							MPU GPIO	GPIO_24	mode6
W13	IO	MMC/SDIO	SDMC_CLK	MPU SPI_ 100K_1	MPU_ SPI1_ SCLK	Compact Flash	CF_nCD1							MPU GPIO	GPIO_25	mode6
R13	IO	MMC/SDIO	SDMC_CMD	MPU SPI_ 100K_1	MPU_ SPI1_ SDO	Compact Flash	CF_nCD2							MPU GPIO	GPIO_26	mode6
V12	IO	MMC/SDIO	SDMC_DAT_0	MPU SPI_ 100K_1	MPU_ SPI1_SDI	Compact Flash	CF_ nIOIS16							MPU GPIO	GPIO_27	mode6
Y12	IO	MMC/SDIO	SDMC_DAT_1	MPU SPI_ 100K_1	MPU_ SPI1_ SEN0	Compact Flash	CF_ INTREQ							MPU GPIO	GPIO_28	mode6
V13	IO	MMC/SDIO	SDMC_DAT_2	MPU SPI_ 100K_1	MPU_ SPI1_ SEN1	Compact Flash	CF_ RESET							MPU GPIO	GPIO_29	mode6
Y13	IO	MMC/SDIO	SDMC_DAT_3	MPU SPI_ 100K_1	MPU_ SPI1_ SEN2									MPU GPIO	GPIO_30	mode6
AA17	IO	USB	USB_DM	MPU UART_ MODEM	MPU_ UART_ TX1	MPU UART_ MODEM_ IRDA	MPU_ UART_ TX_IR2	USB OTG	USB_ SEO_VM	USB	USB_ SEO			MPU GPIO	GPIO_31	mode6
Y17	IO	USB	USB_DP	MPU UART_ MODEM	MPU_ UART_ RX1	MPU UART_ IRDA	MPU_ UART_ RX_IR2	USB OTG	USB_ TXD_VP	USB	USB_TXD			MPU GPIO	GPIO_32	mode6
W16	IO	USB	USB_PU_EN	MPU UART_ MODEM	MPU_ UART_ CTS1	GSM UART	GSM_ UART_ CTS	USB OTG	USB_ RCV	GSM MCSI	GSM_ MCSI_ CLK	MPU MCSI	MPU_ CLK	MPU GPIO	GPIO_33	mode6
W17	IO	USB	USB_VBUSI	MPU UART_ MODEM	MPU_ UART_ RTS1	GSM UART	GSM_ UART_ RTS	USB OTG	USB_ TXEN	GSM MCSI	GSM_ MCSI_ TXD	MPU MCSI	MPU_ MCSI_ TXD	MPU GPIO	GPIO_34	mode6
V16	IO	EAC	MCLK_OUT	MPU UART_ MODEM	MPU_ UART_ DCD1	GSM UART	GSM_ UART_TX	USB	USB_VM	GSM MCSI	GSM_ MCSI_ RXD	MPU MCSI	MPU_ MCSI_ RXD	MPU GPIO	GPIO_35	mode6

Table C–66. Pin Multiplexing (Continued)

Ball	Type	Pin not muxed or Mode 0 (Primary function)		Mode 1 function		Mode 2 function		Mode 3 function		Mode 4 function		Mode 5 function		Mode 6 function (MPU GPIO)		Reset
W18	IO	EAC Audio Codec	CRESET	MPU UART_ MODEM	MPU_ UART_ DSR1	GSM UART	GSM_ UART_ RX	USB	USB_VP	GSM MCSI	GSM_ MCSI_ FSYNCH	MPU MCSI	MPU_ MCSI_ FSYNCH	MPU GPIO	GPIO_36	mode6
Y4	IO	MPU UART_ MODEM_IRDA	MPU_UART_ TX_IR2	LCD I/F	LCD_ PIXEL_16					GSM UART	GSM_ UART_ TX			MPU GPIO	GPIO_37	mode6
V5	IO	MPU UART_ MODEM_IRDA	MPU_UART_ RX_IR2	LCD I/F	LCD_ PIXEL_17					GSM UART	GSM_ UART_ RX			MPU GPIO	GPIO_38	mode6
W4	IO	MPU UART_ MODEM_IRDA	MPU_UART_ SD2	HDQ 1wire	HDQ1W									MPU GPIO	GPIO_39	mode6
R8	IO	MPU UART_MODEM	MPU_UART_ TX1	VLYNQ	VLYNQ_ TXD0	MPU UART_ MODEM_ IRDA	MPU_ UART_ TX2	GSM UWIRE	GSM_ UW_SDO	MPU SPI_ 100K_1	MPU_ SPI1_ SCLK	TPU/TSP PORT	TSPACT_ 5	MPU GPIO	GPIO_40	mode6
Y3	IO	MPU UART_MODEM	MPU_UART_ RX1	VLYNQ	VLYNQ_ RXD0	MPU UART_ MODEM_ IRDA	MPU_ UART_ RX2	GSM UWIRE	GSM_ UW_SDI	MPU SPI_ 100K_1	MPU_ SPI1_ SDO	TPU/TSP PORT	TSPACT_ 6	MPU GPIO	GPIO_41	mode6
AA1	IO	MPU UART_MODEM	MPU_UART_ CTS1	VLYNQ	VLYNQ_ RXD1	MPU UART_ MODEM_ IRDA	MPU_ UART_ CTS2	GSM UWIRE	GSM_ UW_ SCLK	MPU SPI_ 100K_1	MPU_ SPI1_SDI	TPU/TSP PORT	TSPACT_ 7	MPU GPIO	GPIO_42	mode6
AA2	IO	MPU UART_MODEM	MPU_UART_ RTS1	VLYNQ	VLYNQ_ TXD1	MPU UART_ MODEM_ IRDA	MPU_ UART_ RTS2	GSM UWIRE	GSM_ UW_ nSCS1	MPU SPI_ 100K_1	MPU_ SPI1_ SEN0	TPU/TSP PORT	TSPACT_ 8	MPU GPIO	GPIO_43	mode6
M7	IO	LCD I/F	LCD_ PIXEL_15	GSM MCSI	GSM_ MCSI_ FSYNCH	MPU SPI_ 100K_2	MPU_ SPI2_ SCLK							MPU GPIO	GPIO_44	mode6
M8	IO	LCD I/F	LCD_ PIXEL_14	GSM MCSI	GSM_ MCSI_ CLK	MPU SPI_ 100K_2	MPU_ SPI2_ SDO							MPU GPIO	GPIO_45	mode6
L1	IO	LCD I/F	LCD_ PIXEL_13	GSM MCSI	GSM_ MCSI_ TXD	MPU SPI_ 100K_2	MPU_ SPI2_ SDI							MPU GPIO	GPIO_46	mode6
L3	IO	LCD I/F	LCD_ PIXEL_12	GSM MCSI	GSM_ MCSI_ RXD	MPU SPI_ 100K_2	MPU_ SPI2_ SEN0							MPU GPIO	GPIO_47	mode6
L4	IO	LCD I/F	LCD_PIXEL_11			MPU SPI_ 100K_2	MPU_ SPI2_ SEN1							MPU GPIO	GPIO_48	mode6
L7	IO	LCD I/F	LCD_ PIXEL_10			MPU SPI_ 100K_2	MPU_ SPI2_ SEN2							MPU GPIO	GPIO_49	mode6
K3	IO	LCD I/F	LCD_PIXEL_9											MPU GPIO	GPIO_50	mode6

Table C–66. Pin Multiplexing (Continued)

Ball	Type	Pin not muxed or Mode 0 (Primary function)		Mode 1 function		Mode 2 function		Mode 3 function		Mode 4 function		Mode 5 function		Mode 6 function (MPU GPIO)		Reset
J2	IO	LCD I/F	LCD_PIXEL_8											MPU GPIO	GPIO_51	mode6
K7	IO	LCD I/F	LCD_PIXEL_7											MPU GPIO	GPIO_52	mode6
H3	IO	LCD I/F	LCD_PIXEL_6											MPU GPIO	GPIO_53	mode6
H2	IO	LCD I/F	LCD_PIXEL_5											MPU GPIO	GPIO_54	mode6
G1	IO	LCD I/F	LCD_PIXEL_4											MPU GPIO	GPIO_55	mode6
H4	IO	LCD I/F	LCD_PIXEL_3											MPU GPIO	GPIO_56	mode6
K8	IO	LCD I/F	LCD_PIXEL_2											MPU GPIO	GPIO_57	mode6
G2	IO	LCD I/F	LCD_PIXEL_1											MPU GPIO	GPIO_58	mode6
G3	IO	LCD I/F	LCD_PIXEL_0	MPU UWIRE	MPU_ UW_ nSCS1	MCBSP2	FSRX2							MPU GPIO	GPIO_59	mode6
J4	IO	LCD I/F	LCD_PCLK	MPU UWIRE	MPU_ UW_ SCLK	MCBSP2	CLKRX2							MPU GPIO	GPIO_60	mode6
J3	IO	LCD I/F	LCD_HSYNC	MPU UWIRE	MPU_ UW_SDO	MCBSP2	DX2							MPU GPIO	GPIO_61	mode6
K4	IO	LCD I/F	LCD_VSYNC	MPU UWIRE	MPU_ UW_ nSCS2	MCBSP2	DR2							MPU GPIO	GPIO_62	mode6
G4	IO	LCD I/F	LCD_AC	MPU UWIRE	MPU_ UW_SDI	MCBSP2	CLKS2							MPU GPIO	GPIO_63	mode6
Y2	IO	EAC Audio Codec	CSYNC	Keypad	KBR_5	MCBSP1	FSRX1							MPU GPIO	GPIO_64	mode6
W3	IO	EAC Audio Codec	CSCLK	Keypad	KBR_6	MCBSP1	CLKRX1							MPU GPIO	GPIO_65	mode6
Y1	IO	EAC Audio Codec	CDO	Keypad	KBC_5	MCBSP1	DX1							MPU GPIO	GPIO_66	mode6
V4	IO	EAC Audio Codec	CDI	Keypad	KBC_6	MCBSP1	DR1	TPU/TSP PORT	TSPACT_ 9	GSM GPIO	IO_ GSM_0			MPU GPIO	GPIO_67	mode6
W2	IO	EAC Audio Codec	MCLK	VLYNQ	VLYNQ_ CLK	MCBSP1	CLKS1	TPU/TSP PORT	TSPACT_ 10	GSM GPIO	IO_ GSM_1	Backup	CLK13M_ IN	MPU GPIO	GPIO_68	mode6
V6	IO	MPU I2C	MPU_I2C_SDA	Ex- tended GPIO	EXT_ IO_2	GSM I2C	GSM_ I2C_SDA	LCD I/F	LCD_ PIXEL_16			USB	USB_ SUS- PEND	MPU GPIO	GPIO_69	mode6
W5	IO	MPU I2C	MPU_I2C_SCK	Ex- tended GPIO	EXT_ IO_3	GSM I2C	GSM_ I2C_SCK	LCD I/F	LCD_ PIXEL_17			USB	USB_ SPEED	MPU GPIO	GPIO_70	mode6
H11	O	SDRAM	SADD_13													mode0
H9	O	SDRAM	SADD_12													mode0
H10	O	SDRAM	SADD_11													mode0
B8	O	SDRAM	SADD_10													mode0

Table C–66. Pin Multiplexing (Continued)

Ball	Type	Pin not muxed or Mode 0 (Primary function)		Mode 1 function		Mode 2 function		Mode 3 function		Mode 4 function		Mode 5 function		Mode 6 function (MPU GPIO)		Reset
B12	O	SDRAM	SADD_9													mode0
G9	O	SDRAM	SADD_8													mode0
G11	O	SDRAM	SADD_7													mode0
G12	O	SDRAM	SADD_6													mode0
B9	O	SDRAM	SADD_5													mode0
G10	O	SDRAM	SADD_4													mode0
A1	O	SDRAM	SADD_3													mode0
B6	O	SDRAM	SADD_2													mode0
B2	O	SDRAM	SADD_1													mode0
A2	O	SDRAM	SADD_0													mode0
C3	O	SDRAM	SBANK_1													mode0
B3	O	SDRAM	SBANK_0													mode0
C12	IO	SDRAM	SDATA_15													mode0
D12	IO	SDRAM	SDATA_14													mode0
D13	IO	SDRAM	SDATA_13													mode0
C11	IO	SDRAM	SDATA_12													mode0
C13	IO	SDRAM	SDATA_11													mode0
D11	IO	SDRAM	SDATA_10													mode0
D14	IO	SDRAM	SDATA_9													mode0
C10	IO	SDRAM	SDATA_8													mode0
D8	IO	SDRAM	SDATA_7													mode0
C4	IO	SDRAM	SDATA_6													mode0
C7	IO	SDRAM	SDATA_5													mode0
D5	IO	SDRAM	SDATA_4													mode0
D7	IO	SDRAM	SDATA_3													mode0
C5	IO	SDRAM	SDATA_2													mode0
C6	IO	SDRAM	SDATA_1													mode0
D6	IO	SDRAM	SDATA_0													mode0
C9	IO	SDRAM	SDCLK													mode0
H7	O	SDRAM	NSRAS													mode0
B4	O	SDRAM	NSCAS													mode0

Table C–66. Pin Multiplexing (Continued)

Ball	Type	Pin not muxed or Mode 0 (Primary function)		Mode 1 function		Mode 2 function		Mode 3 function		Mode 4 function		Mode 5 function		Mode 6 function (MPU GPIO)		Reset
H8	O	SDRAM	NSWE													mode0
C8	O	SDRAM	NSDQML													mode0
D10	O	SDRAM	NSDQMU													mode0
G8	IO	SDRAM	NSDCS													mode0
H12	O	SDRAM	SDCLK_EN													mode0
C14	IO	DDR	DQSH													mode6
D4	IO	DDR	DQSL													mode6
D9	IO	DDR	SDCLKX													mode6
L18	IO	EMIF slow	FADD_25	Clocks	clk_13m_ req	Slow memory	NFCS_0							MPU GPIO	GPIO_75	mode0
L15	IO	EMIF slow	FADD_24											MPU GPIO	GPIO_76	mode0
M19	IO	EMIF slow	FADD_23											MPU GPIO	GPIO_77	mode0
M18	IO	EMIF slow	FADD_22											MPU GPIO	GPIO_78	mode0
L14	IO	EMIF slow	FADD_21											MPU GPIO	GPIO_79	mode0
N19	IO	EMIF slow	FADD_20							MPU SPI_ 100K_2	MPU_ SPI2_ SCLK			MPU GPIO	GPIO_80	mode0
N18	IO	EMIF slow	FADD_19							MPU SPI_ 100K_2	MPU_ SPI2_ SDO			MPU GPIO	GPIO_81	mode0
N20	IO	EMIF slow	FADD_18							MPU SPI_ 100K_2	MPU_ SPI2_SDI			MPU GPIO	GPIO_82	mode0
M15	IO	EMIF slow	FADD_17							MPU SPI_ 100K_2	MPU_ SPI2_ SEN0			MPU GPIO	GPIO_83	mode0
P19	IO	EMIF slow	FADD_16	Hard-ware_ NFC	RDY					MPU SPI_ 100K_2	MPU_ SPI2_ SEN1			MPU GPIO	GPIO_84	mode0
P20	IO	EMIF slow	FADD_15	Hard-ware_ NFC	ALE									MPU GPIO	GPIO_85	mode0
R21	IO	EMIF slow	FADD_14	Hard-ware_ NFC	CLE									MPU GPIO	GPIO_86	mode0
P18	IO	EMIF slow	FADD_13	Hard-ware_ NFC	WE									MPU GPIO	GPIO_87	mode0
R20	IO	EMIF slow	FADD_12		I/O_7									MPU GPIO	GPIO_88	mode0

Table C–66. Pin Multiplexing (Continued)

Ball	Type	Pin not muxed or Mode 0 (Primary function)		Mode 1 function		Mode 2 function		Mode 3 function		Mode 4 function		Mode 5 function		Mode 6 function (MPU GPIO)		Reset
R19	IO	EMIF slow	FADD_11	Hard-ware_ NFC	CE_1									MPU GPIO	GPIO_89	mode0
R18	IO	EMIF slow	FADD_10	Hard-ware_ NFC	CE_2									MPU GPIO	GPIO_90	mode0
T20	IO	EMIF slow	FADD_9		I/O_0									MPU GPIO	GPIO_91	mode0
T19	IO	EMIF slow	FADD_8	Hard-ware_ NFC	WP									MPU GPIO	GPIO_92	mode0
N15	IO	EMIF slow	FADD_7		I/O_6									MPU GPIO	GPIO_93	mode0
U20	IO	EMIF slow	FADD_6		I/O_5									MPU GPIO	GPIO_94	mode0
T18	IO	EMIF slow	FADD_5		I/O_4									MPU GPIO	GPIO_95	mode0
U19	IO	EMIF slow	FADD_4		I/O_3									MPU GPIO	GPIO_96	mode0
V20	IO	EMIF slow	FADD_3		I/O_2									MPU GPIO	GPIO_97	mode0
U18	IO	EMIF slow	FADD_2		I/O_1									MPU GPIO	GPIO_98	mode0
W21	IO	EMIF slow	FADD_1	Hard-ware_ NFC	RE									MPU GPIO	GPIO_99	mode0
E18	IO	EMIF slow	FDATA_15											MPU GPIO	GPIO_ 100	mode0
D19	IO	EMIF slow	FDATA_14											MPU GPIO	GPIO_ 101	mode0
F18	IO	EMIF slow	FDATA_13											MPU GPIO	GPIO_ 102	mode0
E19	IO	EMIF slow	FDATA_12											MPU GPIO	GPIO_ 103	mode0
E20	IO	EMIF slow	FDATA_11											MPU GPIO	GPIO_ 104	mode0
F19	IO	EMIF slow	FDATA_10											MPU GPIO	GPIO_ 105	mode0
F20	IO	EMIF slow	FDATA_9											MPU GPIO	GPIO_ 106	mode0
G18	IO	EMIF slow	FDATA_8											MPU GPIO	GPIO_ 107	mode0
G19	IO	EMIF slow	FDATA_7											MPU GPIO	GPIO_ 108	mode0
G20	IO	EMIF slow	FDATA_6											MPU GPIO	GPIO_ 109	mode0

Table C–66. Pin Multiplexing (Continued)

Ball	Type	Pin not muxed or Mode 0 (Primary function)		Mode 1 function		Mode 2 function		Mode 3 function		Mode 4 function		Mode 5 function		Mode 6 function (MPU GPIO)		Reset
H15	IO	EMIF slow	FDATA_5											MPU GPIO	GPIO_110	mode0
H18	IO	EMIF slow	FDATA_4											MPU GPIO	GPIO_111	mode0
H19	IO	EMIF slow	FDATA_3											MPU GPIO	GPIO_112	mode0
J15	IO	EMIF slow	FDATA_2											MPU GPIO	GPIO_113	mode0
J20	IO	EMIF slow	FDATA_1											MPU GPIO	GPIO_114	mode0
J18	IO	EMIF slow	FDATA_0											MPU GPIO	GPIO_115	mode0
K14	IO	EMIF slow	NFCS_2	EMIF slow	FADD_25									MPU GPIO	GPIO_116	mode0
K18	IO	EMIF slow	NFCS_1			EMIF slow	NFCS3H							MPU GPIO	GPIO_117	mode0
J19	IO	EMIF slow	NFCS_3			EMIF slow	NFCS3L							MPU GPIO	GPIO_118	mode0
C19	IO	EMIF slow	NFWE											MPU GPIO	GPIO_119	mode0
D18	IO	EMIF slow	NFOE											MPU GPIO	GPIO_120	mode0
C21	IO	EMIF slow	NFBAA											MPU GPIO	GPIO_121	mode0
B20	IO	EMIF slow	NFWP											MPU GPIO	GPIO_122	mode0
B21	IO	EMIF slow	NFWAIT	Hard-ware_NFC	RDY									MPU GPIO	GPIO_123	mode0
J21	IO	EMIF slow	FCLK											MPU GPIO	GPIO_124	mode0
L19	IO	EMIF slow	NFADV			Slow memory	NFCS_0							MPU GPIO	GPIO_125	mode0
K19	IO	EMIF slow	NFBE_1			GSM RIF BB I/F	BCLKX	Slow memory	NFCS_0					MPU GPIO	GPIO_126	mode0
K15	IO	EMIF slow	NFBE_0			GSM RIF BB I/F	BCLKR							MPU GPIO	GPIO_127	mode0
C20	IO	EMIF slow	NFRST											MPU GPIO	GPIO_128	mode0
U3	IO	MPU SPI_100K_1	MPU_SPI1_SCLK	EAC BT auSPI port	SCLK	MPU MCSI	MPU_MCSI_CLK	GSM UWIRE	GSM_UW_SCLK	MCBSP1	CLKRX1	GSM MCSI	GSM_MCSI_CLK	MPU GPIO	GPIO_129	mode6

Table C–66. Pin Multiplexing (Continued)

Ball	Type	Pin not muxed or Mode 0 (Primary function)		Mode 1 function		Mode 2 function		Mode 3 function		Mode 4 function		Mode 5 function		Mode 6 function (MPU GPIO)		Reset
V2	IO	MPU SPI_100K_1	MPU_SPI1_ SDO	EAC BT auSPI port	SDO	MPU MCSI	MPU_ MCSI_ TXD	GSM UWIRE	GSM_ UW_SDO	MCBSP1	DX1	GSM MCSI	GSM_ MCSI_ TXD	MPU GPIO	GPIO_ 130	mode6
T4	IO	MPU SPI_100K_1	MPU_SPI1_ SDI	EAC BT auSPI port	SDI	MPU MCSI	MPU_ MCSI_ RXD	GSM UWIRE	GSM_ UW_SDI	MCBSP1	DR1	GSM MCSI	GSM_ MCSI_ RXD	MPU GPIO	GPIO_ 131	mode6
V3	IO	MPU SPI_100K_1	MPU_SPI1_ SEN0	EAC BT auSPI port	FSYNC	MPU MCSI	MPU_ MCSI_ FSYNCH	GSM UWIRE	GSM_ UW_ nSCS1	MCBSP1	FSRX1	GSM MCSI	GSM_ MCSI_ FSYNCH	MPU GPIO	GPIO_ 132	mode6
U4	IO	MPU SPI_100K_1	MPU_SPI1_ SEN1	EAC BT auSPI port	SEN1			GSM UWIRE	GSM_ UW_ nSCS2	MCBSP1	CLKS1	GSM GPIO	IO_GSM_ 2	MPU GPIO	GPIO_ 133	mode6
W1	IO	MPU SPI_100K_1	MPU_SPI1_ SEN2	Backup	CLK13M_ IN	EAC Au- dio Codec	CRESET	MPU UART	MPU_ UART_ DTR1			GSM GPIO	IO_GSM_ 3	MPU GPIO	GPIO_ 134	mode6
AA20	IO	SMC	SMC_IO	External MCU IRQ (IN)	EXT_ ARM_ NIRQ	TPU/TSP PORT	TSPACT_ 5	Keypad	KBR_5	GSM UWIRE	GSM_ UW_SDI	MPU UWIRE	MPU_ UW_SDI	MPU GPIO	GPIO_ 135	mode6
V17	IO	SMC	SMC_CLK	External DSP IRQ (IN)	EXT_ DSP_ NIRQ	TPU/TSP PORT	TSPACT_ 6	Keypad	KBR_6	GSM UWIRE	GSM_ UW_ SCLK	MPU UWIRE	MPU_ UW_ SCLK	MPU GPIO	GPIO_ 136	mode6
W19	IO	SMC	SMC_RST	GSM GPIO	IO_GSM_ 9	TPU/TSP PORT	TSPACT_ 7	Keypad	KBC_5	GSM UWIRE	GSM_ UW_SDO	MPU UWIRE	MPU_ UW_SDO	MPU GPIO	GPIO_ 137	mode6
V18	IO	SMC	SMC_CD	TPU IT TDMA (OUT)	IT_FRA- ME	TPU/TSP PORT	TSPACT_ 8	TPU/TSP PORT	TSPACT_ 11	GSM UWIRE	GSM_ UW_ nSCS1	MPU UWIRE	MPU_ UW_ nSCS1	MPU GPIO	GPIO_ 138	mode6
Y19	IO	SMC	SMC_PWCTRL	DSP GPO	XF	TPU/TSP PORT	TSPACT_ 9	Keypad	KBC_6	GSM UWIRE	GSM_ UW_ nSCS2	MPU UWIRE	MPU_ UW_ nSCS2	MPU GPIO	GPIO_ 139	mode6
V15	I	System	NRESPWRON													mode0
P2	I	System	MPU_NRST													mode0
P4	I	System	MPU_EXT_ NIRQ	USB	USB_ VBUSI									MPU GPIO	GPIN_4	mode6
M3	I	System	GSM_EXT_ NIRQ											MPU GPIO	GPIN_5	mode6
N2	IO	Config boot/MPU LPG	Mux_mode_ MLPG1	GSM LPG	GSM_ LPG1			Extended GPIO	EXT_ IO_0	System	NFIQ_ PWRFAIL			MPU GPIO	GPIO_ 140	mode0
N7	IO	Config boot / MPU LPG	ARM_boot_ MLPG2	GSM LPG	GSM_ LPG2	TAP	RTCK	Extended GPIO	EXT_ IO_1					MPU GPIO	GPIO_ 141	mode2
R4	I	TAP	TDI													mode0
U2	I	TAP	nTRST													mode0
R2	O	TAP	TDO													mode0

Table C–66. Pin Multiplexing (Continued)

Ball	Type	Pin not muxed or Mode 0 (Primary function)		Mode 1 function		Mode 2 function		Mode 3 function		Mode 4 function		Mode 5 function		Mode 6 function (MPU GPIO)		Reset
R3	I	TAP	TMS													mode0
P7	I	TAP	TCK													mode0
T3	IO	Test & Emulation	NEMU0	System	Low_ power	CLOCKS	CLK48M_ IN	TPU/TSP PORT	TSPEN_3					MPU GPIO	GPIO_ 142	mode0
T2	IO	Test & Emulation	NEMU1	Light lev- el control	PWL	HDQ1wire	HDQ1W	Buzzer	PWT					MPU GPIO	GPIO_ 143	mode0
N4	I	Test & Emulation	NBSCAN													mode0
N3	IO	Test & Emulation	TEST_MODE											MPU GPIO	GPIO_ 144	mode0
Y9	I	Clocks	CLKTCXO													mode0
N1	O	Clocks	CLK13M_OUT													mode0
AA15	I	Clocks	OSC32K_IN													mode0
W15	O	Clocks	OSC32K_OUT													mode0
V14	O	Clocks	CLK32K													mode0
P3	IO	Clocks	CLK_13M_ REQ	GSM GPIO	IO_GSM_ 3			ETM	PIPES- TAT_3					MPU GPIO	GPIO_ 145	mode6
D17	IO	CAMERA IF	CAM_LCLK	GSM MCSI	GSM_ MCSI_ CLK	Keypad	KBR_5	ETM	TRA- CECLK	ARMIO	armio_0	DEBUG	DEBUG_ 12	MPU GPIO	GPIO_ 146	mode6
C15	IO	CAMERA IF	CAM_EXCLK	GSM MCSI	GSM_ MCSI_ TXD	Keypad	KBR_6	ETM	TRACE- SYNC	ARMIO	armio_1	DEBUG	DEBUG_ 11	MPU GPIO	GPIO_ 147	mode6
C16	IO	CAMERA IF	CAM_HS	GSM MCSI	GSM_ MCSI_ RXD	Keypad	KBR_7	ETM	PIPES- TAT_2	ARMIO	armio_2	DEBUG	DEBUG_ 10	MPU GPIO	GPIO_ 148	mode6
D15	IO	CAMERA IF	CAM_VS	GSM MCSI	GSM_ MCSI_ FSYNCH	Keypad	KBC_5	ETM	PIPES- TAT_1	GSM UWIRE	GSM_ UW_ nSCS2	DEBUG	DEBUG_ 9	MPU GPIO	GPIO_ 149	mode6
A21	IO	CAMERA IF	CAM_RSTZ	MCBSP1	CLKRX1	Keypad	KBC_6	ETM	PIPES- TAT_0	GSM UWIRE	GSM_ UW_SDI	DEBUG	DEBUG_ 8	MPU GPIO	GPIO_ 150	mode6
G14	IO	CAMERA IF	CAM_DATA_0	MCBSP1	DX1	Keypad	KBC_7	ETM	TRA- CEPKT_0	GSM UWIRE	GSM_ UW_ SCLK	DEBUG	DEBUG_ 7	MPU GPIO	GPIO_ 151	mode6
A20	IO	CAMERA IF	CAM_DATA_1	MCBSP1	DR1	GSM GPIO	IO_GSM_ 0	ETM	TRA- CEPKT_1	GSM UWIRE	GSM_ UW_SDO	DEBUG	DEBUG_ 6	MPU GPIO	GPIO_ 152	mode6
B19	IO	CAMERA IF	CAM_DATA_2	MCBSP1	FSRX1	GSM GPIO	IO_GSM_ 1	ETM	TRA- CEPKT_2	GSM UWIRE	GSM_ UW_ nSCS1	DEBUG	DEBUG_ 5	MPU GPIO	GPIO_ 153	mode6
C18	IO	CAMERA IF	CAM_DATA_3	MCBSP1	CLKS1	GSM GPIO	IO_GSM_ 2	ETM	TRA- CEPKT_3	EAC BT auSPI port	SCLK	DEBUG	DE- BUG_4	MPU GPIO	GPIO_ 154	mode6

Table C–66. Pin Multiplexing (Continued)

Ball	Type	Pin not muxed or Mode 0 (Primary function)		Mode 1 function		Mode 2 function		Mode 3 function		Mode 4 function		Mode 5 function		Mode 6 function (MPU GPIO)		Reset
D16	IO	CAMERA IF	CAM_DATA_4	MPU MCSI	MPU_ MCSI_ CLK	VLYNQ	VLYNQ_ RXD0	ETM	TRA-CEPKT_4	EAC BT auSPI port	SDO	DEBUG	DE-BUG_3	MPU GPIO	GPIO_ 155	mode6
C17	IO	CAMERA IF	CAM_DATA_5	MPU MCSI	MPU_ MCSI_ TXD	VLYNQ	VLYNQ_ TXD0	ETM	TRA-CEPKT_5	EAC BT auSPI port	SDI	DEBUG	DE-BUG_2	MPU GPIO	GPIO_ 156	mode6
B17	IO	CAMERA IF	CAM_DATA_6	MPU MCSI	MPU_ MCSI_ RXD	VLYNQ	VLYNQ_ RXD1	ETM	TRA-CEPKT_6	EAC BT auSPI port	FSYNC	DEBUG	DE-BUG_1	MPU GPIO	GPIO_ 157	mode6
A17	IO	CAMERA IF	CAM_DATA_7	MPU MCSI	MPU_ MCSI_ FSYNCH	VLYNQ	VLYNQ_ TXD1	ETM	TRA-CEPKT_7	EAC BT auSPI port	SEN1	DEBUG	DE-BUG_0	MPU GPIO	GPIO_ 158	mode6
E2	IO	Keypad	KBR_0	Ex-tended GPIO	EXT_ IO_0			GSM I^2C	GSM_ I2C_SCK	ETM	PIPES-TAT_5			MPU GPIO	GPIO_ 159	mode6
J7	IO	Keypad	KBR_1	Ex-tended GPIO	EXT_ IO_1	ARMIO	armio_3			ETM	PIPES-TAT_4			MPU GPIO	GPIO_ 160	mode6
E1	IO	Keypad	KBR_2	MPU I^2C	MPU_ I2C_SDA	ARMIO	armio_4			ETM	TRACE-SYNCB			MPU GPIO	GPIO_ 161	mode6
F3	IO	Keypad	KBR_3	MPU I^2C	MPU_ I2C_SCK	VLYNQ	VLYNQ_ TXD0	TPU/TSP PORT	TSPEN_3			Test & Emulation	EX-TERN0_ MPU	MPU GPIO	GPIO_ 162	mode6
D2	IO	Keypad	KBR_4	GSM UWIRE	GSM_ UW_ nSCS1	VLYNQ	VLYNQ_ RXD0	TPU/TSP PORT	TSPDI	MPU SPI_ 100K_2	MPU_ SPI2_ SCLK	Test & Emulation	EX-TERN1_ MPU	MPU GPIO	GPIO_ 163	mode6
C2	IO	Keypad	KBC_0	Ex-tended GPIO	EXT_ IO_2			GSM I^2C	GSM_ I2C_SDA	MPU SPI_ 100K_2	MPU_ SPI2_ SEN1			MPU GPIO	GPIO_ 164	mode6
D3	IO	Keypad	KBC_1	Ex-tended GPIO	EXT_ IO_3					MPU SPI_ 100K_2	MPU_ SPI2_ SEN2			MPU GPIO	GPIO_ 165	mode6
E4	IO	Keypad	KBC_2	GSM UWIRE	GSM_ UW_SDO					MPU SPI_ 100K_2	MPU_ SPI2_ SDO			MPU GPIO	GPIO_ 166	mode6
F4	IO	Keypad	KBC_3	GSM UWIRE	GSM_ UW_SDI	VLYNQ	VLYNQ_ RXD1			MPU SPI_ 100K_2	MPU_ SPI2_SDI			MPU GPIO	GPIO_ 167	mode6
E3	IO	Keypad	KBC_4	GSM UWIRE	GSM_ UW_ SCLK	VLYNQ	VLYNQ_ TXD1			MPU SPI_ 100K_2	MPU_ SPI2_ SEN0			MPU GPIO	GPIO_ 168	mode6

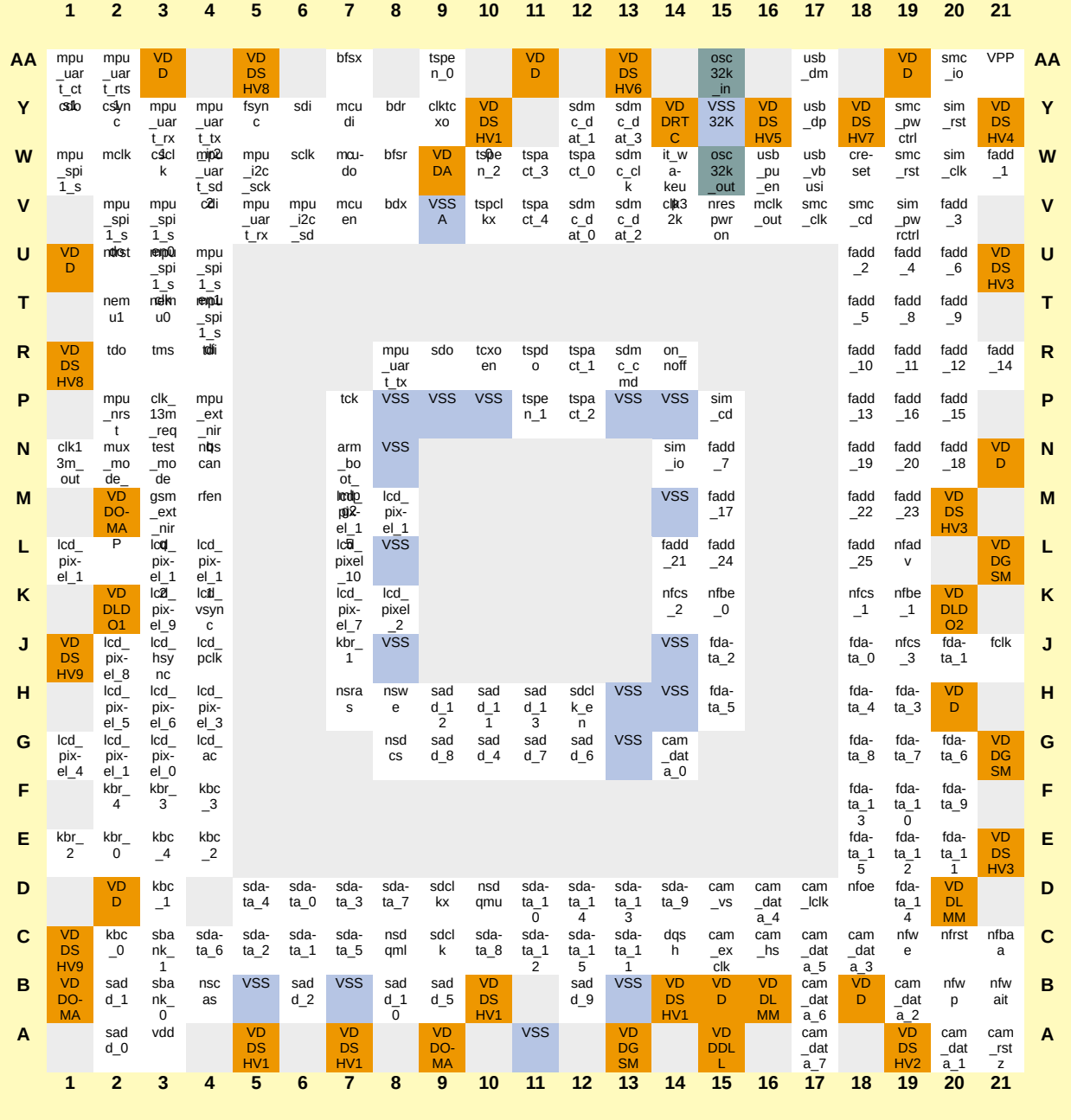
Appendix D

Packaging

This appendix presents the OMAP730 package information and mechanical data.

Topic **Page**

D.1 Package Pin Location



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
AA	mpu_uart_ctsi	mpu_uart_rtso_n	VDD		VDDSHV8		bfsx		tspen_0		VDD		VDDSHV6		osc32k_in		usb_dm		VDD	smc_io	VPP	AA
Y	csdo	csync	mpu_uart_rxi	mpu_uart_txo	fsync	sdi	mcudi	bdr	clktcxo	VDDSHV1		sdmc_dat_1	sdmc_dat_3	VDDRTC	VSS32K	VDDSHV5	usb_dp	VDDSHV7	smc_pwctrl	sim_rst	VDDSHV4	Y
W	mpu_spi1_s	mclk	csclk	mpu_uart_sdi2	mpu_i2c_sck	sclk	mcudo	bfsr	VDDA	tspen_2	tspact_3	tspact_0	sdmc_clk	it_wake_u	osc32k_out	usb_pu_en	usb_vbusi	cre-set	smc_rst	sim_clk	fadd_1	W
V		mpu_spi1_sdo	mpu_spi1_sclk	cdi	mpu_uart_rx	mpu_i2c_sd	mcuen	bdx	VSSA	tspclkx	tspact_4	sdmc_dat_0	sdmc_dat_2	clk32k	nreswpron	mclk_out	smc_clk	smc_cd	sim_pwrctrl	fadd_3		V
U	VDD	nmst	mpu_spi1_clk0	mpu_spi1_sdi0														fadd_2	fadd_4	fadd_6	VDDSHV3	U
T		nemu1	mpu_spi1_clk1	mpu_spi1_sdi1														fadd_5	fadd_8	fadd_9		T
R	VDDSHV8	tdo	tms	tdi				mpu_uart_tx	sdo	tcxoen	tspdo	tspact_1	sdmc_cmd	on_noff				fadd_10	fadd_11	fadd_12	fadd_14	R
P		mpu_nrst	clk_13m_req	mpu_ext_nirq			tck	VSS	VSS	VSS	tspen_1	tspact_2	VSS	VSS	sim_cd			fadd_13	fadd_16	fadd_15		P
N	clk13m_out	mux_mode_	test_mode	nbscan			arm_boot_	VSS						sim_io	fadd_7			fadd_19	fadd_20	fadd_18	VDD	N
M		VDDO-MAP	gsm_ext_nirq	rfen			lcdp-el_1	lcd_pix-el_1						VSS	fadd_17			fadd_22	fadd_23	VDDSHV3		M
L	lcd_pix-el_1		lcd_pix-el_1	lcd_pix-el_1			lcd_pixel_10	VSS						fadd_21	fadd_24			fadd_25	nfadv		VDDGSM	L
K		VDDLDO1	lcd_pix-el_9	lcd_vsync			lcd_pix-el_7	lcd_pixel_2						nfcs_2	nfbe_0			nfcs_1	nfbe_1	VDDLDO2		K
J	VDDSHV9	lcd_pix-el_8	lcd_hsync	lcd_pclk			kbr_1	VSS						VSS	fda-ta_2			fda-ta_0	nfcs_3	fda-ta_1	fclk	J
H		lcd_pix-el_5	lcd_pix-el_6	lcd_pix-el_3			nsras	nswe	sadd_12	sadd_11	sadd_13	sdclk_en	VSS	VSS	fda-ta_5			fda-ta_4	fda-ta_3	VDD		H
G	lcd_pix-el_4	lcd_pix-el_1	lcd_pix-el_0	lcd_ac				nsdcs	sadd_8	sadd_4	sadd_7	sadd_6	VSS	cam_data_0				fda-ta_8	fda-ta_7	fda-ta_6	VDDGSM	G
F		kbr_4	kbr_3	kbc_3														fda-ta_13	fda-ta_10	fda-ta_9		F
E	kbr_2	kbr_0	kbc_4	kbc_2														fda-ta_15	fda-ta_12	fda-ta_11	VDDSHV3	E
D		VDD	kbc_1		sdata_4	sdata_0	sdata_3	sdata_7	sdclkx	nsdqmu	sdata_10	sdata_14	sdata_13	sdata_9	cam_vs	cam_data_4	cam_lclk	nfoe	fdata_14	VDDLMM		D
C	VDDSHV9	kbc_0	sbank_1	sdata_6	sdata_2	sdata_1	sdata_5	nsdqml	sdclk	sdata_8	sdata_12	sdata_15	sdata_11	dqsh	cam_exclk	cam_hs	cam_data_5	cam_data_3	nfwe	nfrst	nfbaa	C
B	VDDO-MAP	sadd_1	sbank_0	nscas	VSS	sadd_2	VSS	sadd_10	sadd_5	VDDSHV1		sadd_9	VSS	VDDSHV1	VDD	VDDLMM	cam_data_6	VDD	cam_data_2	nfwp	nfwait	B
A		sadd_0	vdd		VDDSHV1		VDDSHV1		VDDO-MAP		VSS		VDDGSM		VDDDLL		cam_data_7		VDDSHV2	cam_data_1	cam_rstz	A
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	

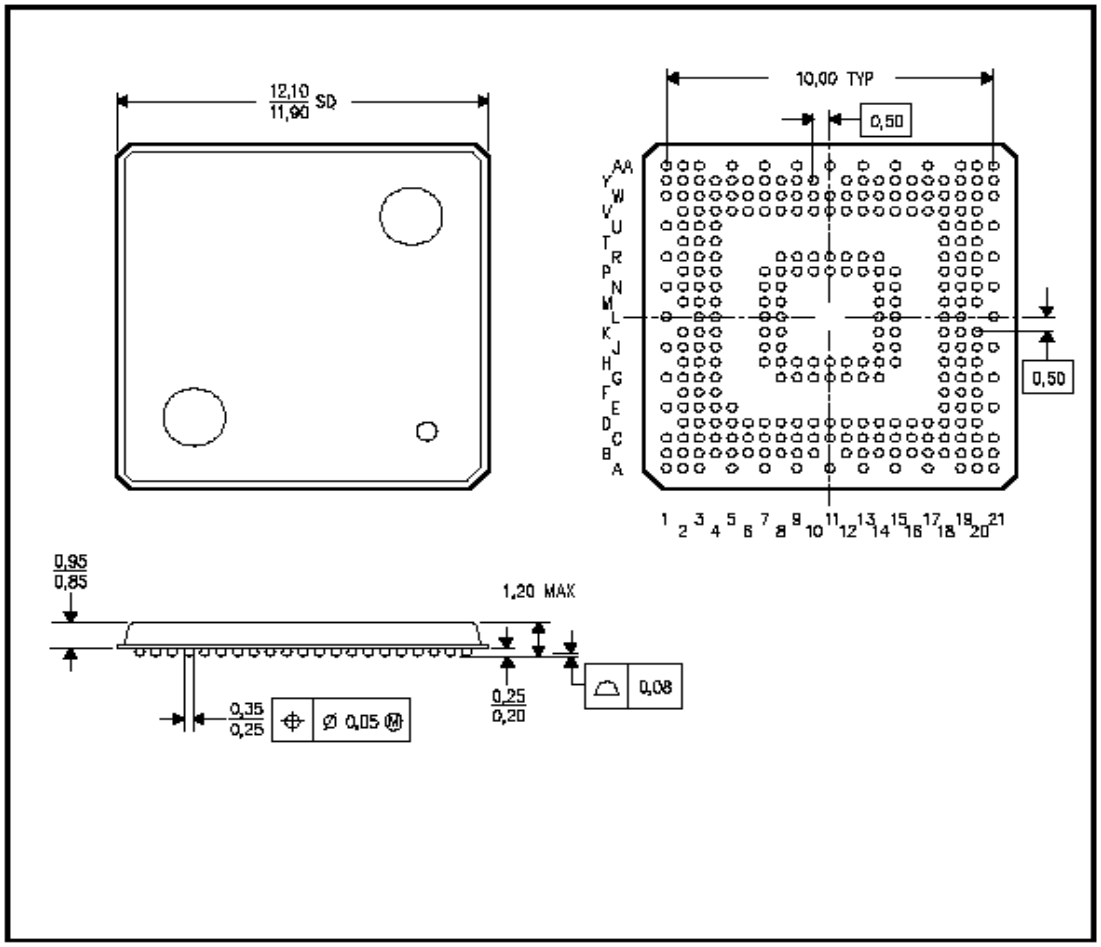
Table 1: OMAP730 Package Pin Location (bottom view)

D.2 Mechanical Data

OMAP730 package is 289–pin S–PBGA (GZG289).

Package code	**289–GZG μ*BGA**
Signal balls	**288**
Power balls	**0**
Leadframe	
Package size:	12 X 12 mm
Ball pitch	0.5 mm

GZG (S–PBGA–N289) PLASTIC BALL GRID ARRAY PACKAGE



NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Micro Star BGA configuration

Appendix E

Peripherals Revision Number

This appendix present peripheral revision number information.

Topic **Page**

E.1 Peripherals Revision Number

Table E–1.Peripherals Revision Number

Peripheral	Revision Number
DES/3DES	1.2
I^2C	1.6
McBSP	1.5
MMC/SDIO	1.7
NAND FLASH	1.3
RNG	2.1
SHA1/MD5	1.4
UART Modem – SIR/MIR/FIR IrDA	1.6A
USB OTG	1.4

Appendix F

Glossary

A

AAC: *Advanced Audio Coding*

ABU: *Autobuffering Unit*

AC97: *Audio Codec 1997.* A standard audio interface that defines a high-quality 16-bit audio architecture.

ACE: *ASIC Compiler Environment.* The graphical user interface delivery mechanism for submicron gate array memory compiler elements.

ADC: *Analog-to-Digital Converter/Conversion.* A converter with an internal sample-and-hold circuitry used to translate an analog signal to a digital signal. Also used to describe the process of conversion.

AFC: *Automatic Frequency Control/*

AIC: *Analog Interface Chip.* An integrated circuit that performs serial analog-to-digital (A/D) and digital-to-analog (D/A) conversions.

AMR: *Audio Modem Riser.* An Intel specification that defines a new architecture for the design of motherboards.

APC: *Automatic Power Control*

AR: *Automatic Reload*

ARM: *Advanced RISC Machine*

ARMIO: *Advanced RISC Machine Input/Output.* This module is connected to the peripheral bus and provides direct I/O communication between a processor controlling the peripheral bus and an external device.

ASIC: *Application Specific Integrated Circuit.* A chip built for a particular application. In the context of this document, this refers to the FPGA that resides on the EVM board.

ASP: *Application-Specific Peripheral*

ASSP: *Application-Specific Silicon Product* or *Application Specific Standard Product*

AuSPI: *Audio Serial Port Interface*

B

B_MMU: *Bufferable Memory Management Unit.* See MMU

BCD: *Binary-Coded Decimal.* A representation of decimal digits (0–9) using a nibble that uses a certain number of bits. For example, by using 4 bits, two BCDs can be packed into one byte.

BE: *Big Endian.* An addressing protocol in which bytes are numbered from left to right within a word. More significant bytes in a word have lower numbered addresses. Endian ordering is specific to hardware and is determined at reset. See also little endian (LE).

BIOS: *Built-In Operating System*

BGA: *Ball Grid Array*

BIST: *Built-In Self Test*

block: A group of interconnected cells. May contain instances of other blocks.

Bluetooth: *(Do not abbreviate.)* A short-range radio technology aimed at simplifying communications among network devices and between devices and the Internet. It also aims to simplify data synchronization between network devices and other computers.

BNC: *British Naval Connector, Bayonet Nut Connector, or Bayonet Neill Concelman.* A type of connector used with coaxial cables.

BSP: *Buffered Serial Port.* An on-chip module that consists of a full-duplex, double-buffered serial port interface and an autobuffering unit (ABU). The double-buffered serial port of the BSP is an enhanced version of the standard serial port interface. The double-buffered serial port allows transfer of a continuous communication stream (8-,10-,12- or 16-bit data packets).

C

C_MMU: *Cacheable Memory Management Unit.* See MMU.

C-Port: *Codec Port*

CASP: *Customer Application Specific Peripheral*

CB: *Copy Back*

CDMA: *Code Division Multiple Access*

CDO: *Coprocessor Data Operation*

CIF: *Common Intermediate Format.* A video format used in videoconferencing systems that easily supports both NTSC and PAL signals. CIF is part of the ITU H.261 videoconferencing standard. It specifies a data rate of 30 frames per second (fps), with each frame containing 288 lines and 352 pixels per line.

CIO: *Channel I/O*

CLK: *Clock*

CLKM: *Clock and Reset Management*

CLKSTP: *Clock Stop*

clock gating: Removing the ac frequency of the clock, usually through a logic (AND, OR) gate

clock throttling: Reducing the frequency of the clock

CO: *Controlled Oscillator*

codec: *Coder Decoder or Compression Decompression.* A device that codes in one direction of transmission and decodes in another direction of transmission.

COFF: *Common Object File Format*

COM: *Communication*

CompactFlash: *(Do not abbreviate).* Very small mass storage device designed with flash memory. This card is the size of a matchbook, weighs a half ounce, and is more rugged and reliable than a disk drive.

companding: Compressing and expanding. A quantization scheme for audio signals in which the input signal is compressed and then, after processing, is reconstructed at the output by expansion. There are two distinct companding schemes—A-law, used in Europe, and μ-law, used in the United States.

CP15: *Coprocessor 15.* This coprocessor controls the operation and configuration of the TI925T.

CPLD: *Complex Programmable Logic Device.* An integrated circuit that can be programmed to perform complex functions.

CPU: *Central Processing Unit.* The CPU is the portion of the processor involved in arithmetic, shifting, and Boolean logic operations, as well as the generation of data and program memory addresses. The CPU includes the central arithmetic logic unit (CALU), the multiplier, and the auxiliary register arithmetic unit (ARAU).

CPR: *Clock, Power, Reset*

CTRL: *Control*

CTS: *Clear to Send*

CS: *Chip-Select*

CSMI: *Coprocessor-Shared Memory Interface*

CSMM: *Coprocessor-Shared Memory Model*

D

DABORT: *Data Abort*

DAI: *Digital Audio Interface.* A GSM test interface that is used to determine the routing of speech data for the devices being tested.

DARAM: *Dual Access Random Access Memory.* RAM that can be accessed twice in a single CPU clock cycle. For example, your code can read from and write to DARAM in the same clock cycle.

DBB: *Digital Baseband*

D-cache: *Data Cache*

DCD: *Data Carrier Detect*

DCT: *Discrete Cosine Transform.* A fast Fourier transform used in manipulating compressed still and moving picture data.

DI: *Data In*

DIMM: *Dual Inline Memory Module*

DIN: *Data In*

DIP: *Dual Inline Package*

DLB: *Data Loopback.* A synchronous serial port test mode in which the receive pins are connected internally to the transmit pins on the same device. This mode, enabled or disabled by the DLB bit, allows you to test whether the port is operating correctly.

DLL: *Divisor Latch LSB*

DMA: *Direct Memory Access.* A mechanism whereby a device other than the host processor contends for, and receives, mastery of the memory bus so that data transfers can take place independent of the host.

DMA Controller: *Direct Memory Access Controller*. Controls data block transfers between memories, peripherals, and processors in the OMAP710 device.

DO: *Data Out*

DPLL: *Digital Phase-Locked Loop.* Digital implementation of PLL.

DPRAM: *Dual-Port RAM*

DRAM: *Dynamic Random Access Memory*. Single access data RAM generally used in RAM based modules to emulate data ROM in future ROM megamodules.

DSP: *Digital Signal Processor.* A semiconductor that manipulates discrete or discontinuous electrical impulses in a manner that implements a desired algorithm.

DSP/BIOS: *Digital Signal Processor/Basic Input/Output System*

DSR: *Data Set Ready*

D-TLB: *Data Transition Lookaside Buffer.* See TLB.

DTR: *Data Terminal Ready*

:DU: *Debug Unit*

E

E^{2}ICE: *Enhanced Embedded ICE Module*

EAC: *Enhanced Audio Controller*

EDGE: *Enhanced Data for GSM Evolution*

EGA: *Enhanced Graphics Adapter*

EEPROM: *Electrically Erasable Programmable Read Only Memory*

EMIF: *Extended Memory Interface.* Consists of the EMIFS and EMIFF.

EMIFS: *Extended Memory Interface Slow.* This 16-bit wide bus (32-bit in the 600-pin package) can interface with and handles all transactions to flash memory, ROM, asynchronous memories, and synchronous burst flash.

EMIFF: *Extended Memory Interface Fast.* The 16-bit bus can interface with either one synchronous burst SRAM (SBSRAM) or one synchronous DRAM (SDRAM).

EPLD: *Erasable Programmable Logic Device*

EPROM: *Erasable Programmable Read-Only Memory*

ES: *Erase Status*

ESC: *Escape*

ESK: *Emulation Software Kit*

ESS: *Erase Suspend Status*

ETK: *Embedded Toolkit*

ETM: *Embedded Trace Macrocell*

EVM: *Evaluation Module*

F

FAR: *Fault Address Register.* The FAR holds the virtual address of the access, which was attempted when a fault occurred.

FC: *Flow Control*

FCS: *Field and Checksum*

FE: *Framing Error.* An error that occurs when the asynchronous serial port receives a data character that does not have a valid stop bit.

FIFO: *First In First Out.* A queue; a data structure or hardware buffer from which items are taken out in the same order they were put in. A FIFO is useful for buffering a stream of data between a sender and receiver which are not synchronized; that is, the sender and receiver are not sending and receiving at exactly the same rate. If the rates differ by too much in one direction for too long, the FIFO becomes either full (blocking the sender) or empty (blocking the receiver).

FIQ: *Fast Interrupt Request.* See ISR.

FPGA: *Field Programmable Gate Array.* A type of logic chip that can be programmed. An FPGA is similar to a PLD; whereas PLDs are generally limited to hundreds of gates, FPGAs support thousands of gates.

FS: *Frame Synchronization*

FSM: *Finite State Machine.* A model of computation consisting of a set of states, a start state, an input alphabet, and a transition function which maps input symbols and current states to a next state.

FSR: *Fault Status Register*

G

GAL: *Generic Array Logic*

GEA: *GPRS Encryption Algorithm*

GDI: *Graphics Driver Interface*

GMM: *GPRS Mobility Management*

GND: *Ground*

GPE: *General-Purpose Event*

GPP: *General-Purpose Processor*

GPIO: *General-Purpose Input/Output.* Pins that can be used to accept input signals and/or send output signals but are not linked to specific uses.

GPS: *Global Positioning System*

GSM: *Global System for Mobile Communications*

GSM-S: *GSM Subsystem*

H

H/W: *Hardware*

HARP: *Hardware Application Reference Platform*

HIO: *Host I/O*

HIVECT: *High Interrupts Vector*

HOM: *Host-Only Mode.* A mode that allows only the host to access host port interface (HPI) memory. The CPU has no access to the HPI memory block during HOM.

HPI: *Host Port Interface*

HSAB: *High-Speed Access Bus.* This external high-speed access bus host is connected to the traffic controller via the high-speed access bus, and can access memories connected to IMIF, EMIFF, and EMIFS.

HSYNC: *Horizontal Synchronization.* A bidirectional horizontal timing signal occurring once per line with a pulse width defined as an integral number of frame clock (FCLK) periods. Synchronization signals can be used to enable retrace of the electron beam of a display screen.

HWA: *Hardware Accelerators.* In the context of this document, this refers to the DCT/IDCT, motion estimation and half-pixel interpolation accelerators in the OMAP710.

I

I^2C: *Inter-Integrated Circuit.* A multimaster bus where multiple chips can be connected. Each chip can act as a master by initiating a data transfer.

I2S: *Inter-IC Sound.* A digital audio interface standard.

I/F: *Interface*

I/O: *Input/Output*

IA: *Identifier Address*

IABORT: *Instruction Abort*

I-cache: *Instruction cache*

IC: *Integrated Circuit*

ICE: *In-Circuit Emulation*

ICR: *Intersystem Communication Registers*

ID: *Identifier Data*

IDCT: *Inverse Discrete Cosine Transform.* See DCT.

IDE: *Integrated Development Environment.* A programming environment integrated into an application.

IHV: *Independent Hardware Vendor*

IMIF: *Internal Memory Interface.*

INT: *Interrupt .* A signal sent by hardware or software to a processor requesting attention. An interrupt tells the processor to suspend its current operation, save the current task status, and perform a particular set of instructions. Interrupts communicate with the operating system and prioritize tasks to be performed.

INTC: *Interrupt Controller*

INTH: *Interrupt Handler*

IOM-2: *ISDN Oriented Modular Interface Revision 2*

IrDA: *Infrared Data Association.* Represents the group of device manufacturers that developed a standard for transmitting data via infrared light waves.

IR: *Infrared*

IRQ: *(Low Priority) Interrupt Request.* IRQs are hardware lines over which devices can send interrupt signals to the microprocessor.

ISO: *Isochronous*. This refers to processes where data must be delivered within certain time constraints. For example, multimedia streams require an isochronous transport mechanism to ensure that data is delivered as fast as it is displayed and to ensure that the audio is synchronized with the video. Also, International Standards Organization.

ISR: *Interrupt Service Routine.* A function or set of functions that are called when an interrupt is encountered.

IST: *Interrupt Service Thread.* A thread that provides additional processing necessary for an interrupt, but which may be on the order of several microseconds.

ISV: *Independent Software Vendor*

I-TLB: *Instruction Translation Lookaside Buffer.* See TLB.

J

JPEG: *Joint Photographics Experts Group.*

JTAG: *Joint Test Action Group.* The Joint Test Action Group was formed in 1985 to develop economical test methodologies for systems designed around complex integrated circuits and assembled with surface-mount technologies. The group drafted a standard that was subsequently adopted by IEEE as IEEE Standard 1149.1–1990, *IEEE Standard Test Access Port and Boundary-Scan Architecture*.

K

KB: *Kilobyte*

KBC: *Keyboard Column*

KBD: *Keyboard*

KBR: *Keyboard Row*

L

LAN: *Local Area Network*

LCD: *Liquid Crystal Display.* A display that uses two sheets of polarizing material with a liquid crystal solution between them.

LDC: *Load (from memory) to Coprocessor*

LDM: *Load Multiple*

LE: *Little Endian.* An addressing protocol in which bytes are numbered from right to left within a word. More significant bytes in a word have higher numbered addresses. Endian ordering is specific to hardware and is determined at reset. See also big endian (BE).

LED: *Light Emitting Diode.* An electronic device that lights up when electricity is passed through it.

LH: *Local Host.* In the context of this document, it refers to the TI925T.

:LLPC: *LCD Low-Power Controller*

LPG: *LED Pulse Generator*

LRU: *Least Recently Used*

LSB: *Least Significant Bit*

M

µWire™: *MicroWire.* This module provides a serial synchronous interface that can drive four serial external components.

MB: *Megabyte*

McBSP: *Multichannel Buffered Serial Port.* An enhanced buffered serial port that includes the following standard features: buffered data registers, full duplex communication, and independent clocking and framing for receive and transmit. In addition, the McBSP includes the following enhanced features: internal programmable clock and frame generation, multichannel mode, and general-purpose I/O.

MCSI: *Multichannel Serial Interface.* A serial Interface with multichannel transmission capability.

MCSI1: *Multichannel Serial Interface 1*

MCSI2: *Multichannel Serial Interface 2*

MCSI: *Multichannel Serial Interface*

MCU: *Microcontroller Unit.* Refers to the MPU.

MEMIF: *Memory Interface*

MicroWire: This module provides a serial synchronous interface that can drive four serial external components. Also µWire.

MIPS: *Million Instructions per Second*

MMA: *Multimedia Applications*

MMC: *Multimedia Controller*

MMIO: *Memory-Mapped I/O*

MMU: *Memory Management Unit.* The MMU performs virtual to physical address translations, performs access permission checks for access to the system memory, and provides the flexibility and security required for the OS to manage a shared physical memory space between the two processors.

MP3: *MPEG Layer 3.* An audio compression format.

MPEG: *Moving Pictures Expert Group.* A compression scheme for full motion video.

MPEG1: The first MPEG compression scheme specification.

MPEG4: The most current MPEG compression scheme specification, intended for very narrow bandwidths.

MPU: *Microprocessor Unit.* In the context of this document, this refers to the TI925T core processor. Also *Memory Protection Unit.*

MPUI: *Microprocessor Unit Interface.* Bus that allows the TI925T and the system DMA controller to communicate with the DSP and its peripherals via the MPUI port (part of DSP).

MPUIF: *Microprocessing Unit Interface Fast.* See also MPUI.

MPUIO: *Microprocessing Unit Input/Output*

MPU-S: *MPU-Subsystem*

MSB: *Most Significant Bit.* The highest order bit in a word. The plural form (MSBs) refers to a specified number of high-order bits, beginning with the highest order bit and counting to the right. For example, the eight MSBs of a 16-bit value are bits 15 through 8.

MUX: *Multiplex/Multiplexer*

MVIP: *Multi-Vendor Integration Protocol*

N

NC: *Not Connected*

NCB: *Non-Cacheable and Buffered*

NCNB: *Non-Cacheable and Non-Buffered*

NIRQ: *Negative (logic) Interrupt Request.* See IRQ.

NMI: *Nonmaskable Interrupt.* An interrupt that can be neither masked nor disabled.

NRT: *Non-Real Time*

NSC: *National Semiconductor Corporation*

O

OAL: *OEM Adaptation Layer*

:OCP: *Open Core Protocol*

:OCP-I: *Open Core Protocol Initiator*

:OCP-T: *Open Core Protocol Target*

OE: *Output Enable*

OEM: *Original Equipment Manufacturers*

OHCI: *Open Host Controller Interface*

OMAP: An open software and hardware platform targeted at second/third generation cellular phones with multimedia capabilities.

OS: *Operating System*

:OTG: *On-The-Go (USB)*

P

PA: *Program Address*

PAL™: *Programmable Array Logic*

PB: *Peripheral Bus. Refers to the TIPB.*

PCB: *Printed Circuit Board*

:PCC: *Power and Clock Controller*

PCI: *Peripheral Component Interconnect.* A local bus standard that is 64 bits wide, though it is usually implemented as a 32-bit bus. It can run at clock speeds of 33 or 66 MHz. At 32 bits and 33 MHz, it yields a throughput rate of 133 MBps.

PCM: *Pulse Code Modulation.* A technique for digitizing speech by sampling the sound waves and converting each sample into a binary number.

PCS: *Personal Communication System.* The U.S. Federal Communications Commission (FCC) term used to describe a set of digital cellular technologies being deployed in the U.S. PCS works over CDMA (also called IS-95), GSM, and North American TDMA (also called IS-136) air interfaces.

PD: *Program Data*

PDA: *Personal Digital Assistant*

PDRAM: *Programmable Dynamic Random Access Memory* (Single access) program data RAM, which is generally used in RAM-based modules to emulate program/data ROM in future ROM megamodules. This RAM is always mapped in program space and can also be mapped in data space using the DROM control bit.

PDROM: *Programmable Dynamic Read-Only Memory*

PE: *Parity Error*

PGA: *Pin Grid Array*

PHY: *Physical Layer Controller*

PID: *Protocol Identifier.* The PID register is used in Windows CE mode only.

PISO: *Parallel In Serial Out*

PLD: *Programmable Logic Devices*

PLL: *Phase-Locked Loop.* A closed loop frequency control system whose function is based on the phase-sensitive detection of the phase difference between the input signal and the output signal of the controlled oscillator (CO).

PMT: *Parallel Module Test.* One of the test configuration modes of the OMAP710 processor.

PRAM: Single-access program RAM that is generally used in RAM based modules to emulate program ROM in future ROM megamodules.

PRBS: *Pseudorandom Bit Sequence*

PROM: *Programmable Read-Only Memory.* A memory chip on which data can be written only once.

PSC: *Prescaler Counter*

PSS: *Program Suspend Status*

PTV: *Prescale Clock Timer Value.* Sets the value of the divisor used in scaling the clock.

PWL: *(pseudonoise) Pulse Width Light (modulator).* A 4096-bit random sequence generator that provides control of the LCD backlighting and keypad.

PWM: *Pulse Width Modulation*

PWT: *Pulse Width Tone.* Creates the output tone signal for a buzzer, programmable both in frequency as well as volume.

Q

QCIF: *Quarter Common Intermediate Format.* A videoconferencing format that specifies data rates of 30 frames per second (fps), with each frame containing 144 lines and 176 pixels per line. This is one fourth the resolution of CIF. QCIF support is required by the ITU H.261 videoconferencing standard.

QVGA: *Quarter Video Graphics Array.* One fourth the resolution of VGA.

R

RAM: *Random Access Memory.* A memory element that can be written to, as well as read.

RDRY: *Receive Data Ready*

RF: *Radio Frequency*

RIF: *Radio Interface*

RISC: *Reduced Instruction Set Computer.* A computer whose instruction set and related decode mechanism are much simpler than those of micro-programmed complex instruction set computers. The result is a higher instruction throughput and a faster real-time interrupt service response from a smaller, cost-effective chip.

RNG: *Random Number Generator*

ROM: *Read Only Memory.* A semiconductor storage element containing permanent data that cannot be changed.

RT: *Real Time*

RTC: *Real Time Clock.* A clock that keeps track of the time even when the device is turned off.

RTOS: *Real Time Operating System*

RTS: *Request to Send*

RX: *Receive/Receiver*

RXC: *Bidirectional Serial Receive Clock*

RXD: *Receive Data*

S

S/W: *Software*

SAM: *Shared Access Mode.* The mode that allows both the DSP and the host to access host port interface (HPI) memory. In this mode, asynchronous host accesses are synchronized internally, and, in case of conflict, the host has access priority and the DSP waits one cycle.

SAP: *Service Access Point*

SARAM: *Single Access Random Access Memory.* Memory space that only can be read from or written to in a single clock cycle; RAM that can accessed (read from or written to) once in a single CPU cycle.

SBFLASH: *Synchronous Burst Flash Memory*

SBZ: *Should Be Zero*

SCL: *Serial Clock.* Programmable serial clock used in the I^2C interface.

SCSA: *Signal Computing System Architecture.* An open, modular architecture for computer telephony that leverages applicable standards and evolves solutions to fill the gaps.

SD: *Starting Delimiter* or *Serial Data*

SDA: *Serial Data.* Serial data bus in the I^2C interface.

SDB: *Standard Development Board*

SDCard: *Serial Data Card*

SDF: *Standard Delay Format*

SDK: *Software Development Kit*

SDRAM: *Synchronous Dynamic Random Access Memory*

SDW: *Short Distance Wireless*

SIM: *Subscriber Identity Module*

SIPO: *Serial In Parallel Out*

:SMC: *Smart Card*

SO-DIMM: *Small-Outline Dual-Inline Memory Module*

SP: *Serial Port* or *Small Page*

SPC: *Serial Port Control*

SPI: *Serial Port Interface*. A signaling protocol for exchanging serial data.

SRAM: *Static Random Access Memory.* Fast memory that does not require refreshing, as DRAM does. It is more expensive than DRAM, though, and is not available in as high a density as DRAM.

SRC: *Sample Rate Conversion*

ST: *Start Timer*

ST-BUS: *Serial Telecom™ Bus*

STC: *Store from Coprocessor* (to memory) or *System Time Clock,* which is the master clock in an MPEG2 encoder or decoder system.

STM: *Synchronous Transfer Mode or Store Multiple.*

STN: *Super-Twist Nematic.* A technique for improving LCD display screens by twisting light rays.

T

TAS: *Test and Set*

TC: *Traffic Controller.* Allows asynchronous operation among the external memory interface, the MPU, and the DSP.

TCIF: *TC Interface*

TCK: *Test Clock*

TDDR: *Timer Divide-Down Register*

TDI: *Test Data Input*

TDMA: *Time Division Multiple Access.*

TDM: *Time Division Multiplex/Multiplexing.* The process by which a single serial bus is shared by multiple devices with each device taking turns to communicate on the bus. The total number of time slots (channels) depends on the number of devices connected. During a time slot, a given device may talk to any combination of devices on the bus.

TDO: *Test Data Output*

TDRY: *Transmit Data Ready*

TDT: *Trace Debug Tool*

TFT: *Thin Film Transistor.* A type of LCD flat panel display screen, in which each pixel is controlled by one to four transistors.

TI: *Texas Instruments*

TIM: *Timer.* Main count register

TINT: *Timer Interrupt*

TIPB: *Texas Instruments Peripheral Bus.* Consists of two buses (private and public) that connects the TI925T to the external and internal peripherals.

TLB: *Translation Lookaside Buffer.* A cache that contains entries for virtual-to-physical address translation and access permission checking.

TMS: *Test Mode Select*

TP: *Tiny Page*

TPA: *Trace Port Analyzer*

TPU: *Time Processing Unit*

TRST: *Test Reset*

TSP: *Time Serial Port*

TSS: *Timer Stop Status*

TTB: *Translation Table Base.* It points to the base of a table in physical memory, which contains section and page table descriptors.

TTL: *Transistor Transistor Logic*

TX: *Transmit/Transmission/Transmitter*

TXC: *Bidirectional Serial Transmit Clock*

TXD: *Transmit Data*

U

UART: *Universal Asynchronous Receiver/Transmitter.* Another name for the asynchronous serial port.

UI: *Unconfirmed Information*

ULPD: *Ultralow-Power Downconverter.* A state machine that can stop the oscillator, and restart it on a wake-up signal.

UND: *Undefined*

UNP: *Unpredictable*

USAR: *Universal Synchronous/Asynchronous Receiver*

USART: *Universal Synchronous/Asynchronous Receiver/Transmitter*

USB: *Universal Serial Bus*. An external bus standard that supports data transfer rates of 12 Mbps (12 million bits per second). A single USB port can be used to connect up to 127 peripheral devices.

V

VA: *Volt-Amps.* A form of power management. A VA rating is the volts rating multiplied by the amps (current) rating), used to indicate the output capacity of an uninterruptible power supply (UPS) or other power source.

VGA: *Video Graphics Array.* An industry standard for video cards.

VCO: *Voltage Controlled Oscillator*

VIVT: *Virtual Index Virtual Tag*

VSYNC: *Vertical Synchronization.* A bidirectional vertical timing signal occurring once per frame with a pulse width defined as an integral number of lines (halflines for interlaced mode).

W

WB: *Write Buffer*

WCDMA: *Wideband Code Division Multiple Access.* A third generation digital cellular technology that uses spread-spectrum techniques.

WD: *Watchdog.* A timer that requires that the user program or OS periodically write to the count register before the counter underflows.

WMA: *Windows Media Audio*

WSMS: *Write State Machine Status*

WS: *Wait State.* A period of time that the CPU must wait for external program, data, or I/O memory to respond when reading from or writing to that external memory. The CPU waits one extra cycle for every wait state.

WT: *Write Through*

X

XIO: *External Memory Interface (Input/Output)* of the lead processor

Index

Nu

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U