Register Mapping

Specification

CAL207

Ver: 1.02

HERCROM400G2
CALYPSO

Department: European Wireless Terminal Chipset Business Unit

	Originator
Name	Michel Gac
Date	2 – May – 2000

HISTORY

Version	Date	Author	Approval manager	Approval date	Notes
0.1	02-May-2000	Michel Gac			1
0.2	27-Jun-2000	Michel Gac			2,3
0.3	10-Aug-2000	Michel Gac			4
0.4	05-Sep-2000	Michel Gac			5
0.5	23-Sep-2000	Michel Gac			6
0.6	11-Dec-2000	R. Servato	Michel Gac	12-Dec-2000	7
0.7	15-Dec-2000	R. Servato	Michel Gac	18-Dec-2000	8
0.8	15-Feb-2001	R. Servato	Michel Gac	16-Feb-2001	9
0.9	26-Mar-2001	R. Servato	Michel Gac	27-Mar-2001	10
0.91	1-June-2001	R. Servato	Michel Gac	1-June-2001	11
0.92	18-June-2001	R. Servato	Michel Gac	23-July-2001	12
0.93	20-Aug-2001	R. Servato	Michel Gac	22-Augt-2001	13
0.94	27-sept-2001	R. Servato	Michel Gac	11-Oct-2001	14
0.95	22-oct-2001	R. Servato	Michel Gac	22-Oct-2001	15
1.00	24-oct-2001	R. Servato	Michel Gac	14-Nov-2001	16
1.01	28-Nov-2001	R. Servato	Michel Gac	28-Nov-2001	17
1.02	28-Jun-2001	R. Servato	Michel Gac	12-July-2001	18

NOTES

1. Imported from SAM207.
2. Updated MCU memory mapping.
3. Updated MCU Rhea strobe 0,1 mapping.
4. Fix bug in MPU mapping table.
5. Update RTC registers.
6. Update MCU IRQ mapping
7. Preliminary update for Calypso
8. Update UART interrupt (Irq 7/18), UART registers (SSR, IIR, SFLSR, BLR)
9. Update DPLL register (PLL_MULT),RIF Register (SPCX), MCU memory map (Table 2 : API RAM)
10. Update Device version code, DSP configuration (Bit 8), Change active edge of serial clock for UWIRE register (cso_edge_rd, cso_edge_wr)
11. Update MPU (Ch 5.8, 5.10, 5.11).
12. Add Rhea peripheral latency (Ch 17.6). remove reference document Ch.
13. update Ch 17.2 (reset value for ACCESS_FACTOR1), update MCU peripheral mapping Table 4 (remove Osc32K), Add Ch 28.6.1 (Current resistance value for OSC32K), update Ch 8.3 (reset value for TIMEOUT), update Ch 21.3/21.4 (reset value), update Ch 33.10.2 (reset value for RX_FIFO_E).
14. Update MPU Chap, update MCU peripheral mapping table
15. Update GAUGING_CTRL_REG register (ch 29.4.1)
16. Update DieID code (Ch7.6, bit63:47), update Acess_Factor formula ch17.2, update Maximun latency for each peripherals table(Ch 17.6), add Pheripheral_Acces_Freq for peripherals in Ch Rhea_cntl_reg (Ch 17.2), , complete description of IO_CNTL_REG (Ch 24.4), update Ch 11.2.2 (register in can be accessed in supervisor mode only),

update ASIC_CONF_REG (Bit 6) and MCR reg bit0 (Ch 33.12) Add features for CALYPSO C035: Update Ch 1 (MCU Memory MAP using ADD(22), using CS4), Update Ch 4.5 (MPU description) update Ch 7.2/7.3 (ID code, version code), update Ch 7.9 (Asic Configuration), Update Ch19 (CNTL_ARM_DIV reg), update Ch 28.6 (RTC_RES_PROG_REG), modify nota Ch 26.4 (TPU sequencer internal address mapping)

17. Update API size to 16K in Chapter MCU memory map using ADD(22) (ch1.2)
18. Update reset value for DPLL control reg (PLL_MULT=0), update GEA programming scheduling chapter (Ch6.17), update UART mode LSR register (bitO = ‘0’ for reset value Ch 33.10.1)

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI seems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI-products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Not does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CAUTION

Naming convention adopted for register’s content definition is:
? : Means “undefined value” (could be 0 or 1 when read)
i : means “value dependent of an external input pin of the chip”

1. MCU MEMORY MAP

MCU memory space is shared between external Memory Interface and RHEA bus.
The Memory Interface is providing 6 chip-select signals.
All internal peripherals are mapped on ARM memory space with a range of 32Kbytes.

Note: *Setting bit 3 of ARM_CONF_REG register (see §5.9) does the addressing range and usage of CS4.*

1.1 MCU memory map using CS4 (default)

Important : nCS4 is only available from Calypso C035 – F751619

(*): External device

Device name	nIBOOT	Start address	Stop address	Size (byte)	Data
nCS0 (*)	1	0000:0000	003F:FFFF	4M	8/16/32
	0	0000:2000	003F:FFFF	4M – 8K	
nCS6	-	0080:0000	00BF:FFFF	512K	8/16/32
not allocated	-	*00C0:0000*	*00FF:FFFF*	-	-
nCS1 (*)	-	0100:0000	013F:FFFF	4M	8/16/32
nCS2 (*)	-	0180:0000	01BF:FFFF	4M	8/16/32
nCS3 (*)	-	0200:0000	023F:FFFF	4M	8/16/32
CS4 (*)	-	0280:0000	02BF:FFFF	4M	8/16/32
nCS4 (*)	-	0280:0000	02BF:FFFF	4M	8/16/32
nCS0 image	-	0300:0000	033F:FFFF	4M	8/16/32
nCS7	1	0380:0000	03BF:FFFF	4M	8/16/32
	0	0000:0000	0000:1FFF	8K	
Debug Unit (DU)	-	03C0:0000	03FF:FFFF	32	32
not allocated	-	*0400:0000*	*FFCF:FFFF*	-	-
API RAM	-	FFD0:0000	FFD0:3FFF	16K	16/32
API control register	-	FFE0:0000	FFE0:0001	2	16

Table 1: MCU memory mapping

1.2 MCU memory map using ADD(22)

Important : nCS4 is only available from Calypso C035 – F751619

(*): External device

Device name	nIBOOT	Start address	Stop address	Size (byte)	Data
nCS0 (*)	1	0000:0000	007F:FFFF	8M	8/16/32
	0	0000:2000	007F:FFFF	8M – 8K	
nCS6	-	0080:0000	00BF:FFFF	512K	8/16/32
not allocated	-	*00C0:0000*	*00FF:FFFF*	-	-
nCS1 (*)	-	0100:0000	017F:FFFF	8M	8/16/32
nCS2 (*)	-	0180:0000	01FF:FFFF	8M	8/16/32
nCS3 (*)	-	0200:0000	027F:FFFF	8M	8/16/32
nCS4 (*)	-	0280:0000	02BF:FFFF	8M	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*	-	-
API RAM	-	FFD0:0000	FFD0:1FFF	16K	16/32
API control register	-	FFE0:0000	FFE0:0001	2	16
Debug Unit	-				

Table 2: MCU memory mapping (cont.)

1.3 External Flash/ROM image

Whatever the value of nIBOOT, the external memory connected on nCS0 could be read or write (depending of the WE bit value) at the address range nCS0-image.
It is not possible to fetch and run code directly at this address due to the ARM long jump instruction set limitation.

1.4 MCU rhea peripherals mapped on Strobe 0

Device name		**Start address**	**Stop address**	**Size in 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*		
UART_IRDA	CS10	FFFF:5000	FFFF:57FF	2K	8
UART_MODEM	CS11	FFFF:5800	FFFF:5FFF	2K	8
UART_UIR	CS12	FFFF:6000	FFFF:67FF	2K	8
reserved	*CS13*	*FFFF:6800*	*FFFF:6FFF*		
RIF	CS14	FFFF:7000	FFFF:77FF	2K	16
reserved	CS15	*FFFF:7800*	*FFFF:7FFF*		
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
RHEA 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
ARM ID code		FFFF:FE00	FFFF:FE01	2	16
CDSP ID code		FFFF:FE02	FFFF:FE03	2	16
MPU		FFFF:FF00	FFFF:FFFF	256	16

Table 3: MCU peripheral mapping (strobe 0)

1.5 MCU rhea peripherals mapped on Strobe 1

Device name		Start address	Stop address	Size in bytes	Data
SIM	CS0	FFFE:0000	FFFE:07FF	2K	16
TSP	CS1	FFFE:0800	FFFE:0FFF	2K	16
reserved	---	*FFFE:1000*	*FFFE:17FF*		
RTC	CS3	FFFE:1800	FFFE:1FFF	2K	8
ULPD	CS4	FFFE:2000	FFFE:27FF	2K	16
I2C	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
ARMIO	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	---	*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*		
not allocated	*CS21*	*FFFE:A800*	*FFFE:AFFF*		
not allocated	*CS22*	*FFFE:B000*	*FFFE:B7FF*		
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*		
Config. JTAG ID	CS30	FFFE:F000	FFFE:F003	2	16
JTAG ver		FFFE:F002	FFFE:F005	2	16
DSP		FFFE:F004	FFFE:F005	2	16
MCU		FFFE:F006	FFFE:F007	2	16
ASIC		FFFE:F008	FFFE:F009	2	16
IOs		FFFE:F00A	FFFE:F00B	2	16
MCU emu		FFFE:F00E	FFFE:F00F	2	16
DIE ID		FFFE:F010	FFFE:F017	8	16
reserved	*CS31*	*FFFE:F800*	*FFFE:FFFF*	-	-

Table 4: MCU peripheral mapping (strobe 1)

1.6 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">API RAM</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td colspan="2">APIC</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td colspan="2">SIM</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td colspan="2">TSP</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td colspan="2">TPU_REG</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td colspan="2">TPU_RAM</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td>NOT MAPPED</td><td>RTC</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td colspan="2">ULPD</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td colspan="2">SPI</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td colspan="2">TIMER1</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td>NOT MAPPED</td><td>LPG</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td>NOT MAPPED</td><td>PWL</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td>NOT MAPPED</td><td>PWT</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td colspan="2">UWIRE</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td colspan="2">ARMIO</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td>NOT MAPPED</td><td>UART_IRDA</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td>NOT MAPPED</td><td>UART_MODEM</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td colspan="2">TIMER2</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td colspan="2">RHEA bridge</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td colspan="2">INTH</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td colspan="2">Memory Interface</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td colspan="2">DMA controller</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td colspan="2">CLKM</td></tr>
<tr><td colspan="2">NOT MAPPED</td><td colspan="2">JTAG ID code</td></tr>
</table>

2. DSP MEMORY MAP

2.1 Memory areas definitions

DARAM: dual access data RAM. It is always mapped in data space and can be overlaid in program space using the OVLY bit.

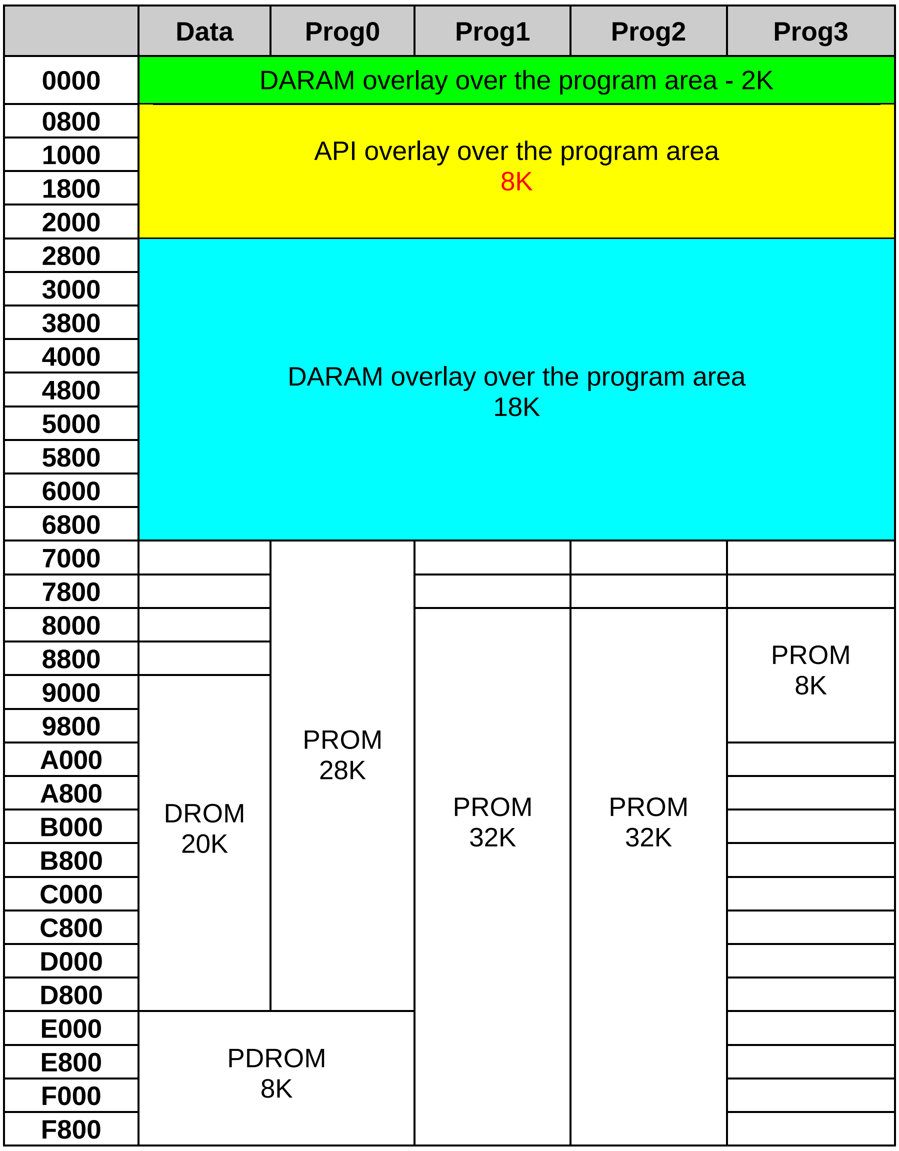
APIRAM: dual access data RAM. It is always mapped in data space and can be overlaid in program space using the OVLY bit. The ARM host processor via the API interface module can also access this memory. It behaves as a communication memory between the Lead CPU and the ARM host processor.

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.

2.2 Mapping diagram



	Data	Prog0	Prog1	Prog2	Prog3
0000	DARAM overlay over the program area - 2K				
0800	API overlay over the program area 8K				
1000					
1800					
2000					
2800	DARAM overlay over the program area 18K				
3000					
3800					
4000					
4800					
5000					
5800					
6000					
6800					
7000		PROM 28K			
7800					
8000			PROM 32K	PROM 32K	PROM 8K
8800					
9000	DROM 20K				
9800					
A000					
A800					
B000					
B800					
C000					
C800					
D000					
D800					
E000	PDROM 8K				
E800					
F000					
F800					

2.3 API

The API interface offers a dual access capability to 8 k-Words of 16 bits of mixed data program memory, it can be configured to manage data access of 8,16 or 32 bits through the API control registers and the memory interface configuration registers (See document [7]).
The API dual access capability is either enabled (SAM mode) or disabled (HOM mode) by the DSP. SAM mode is the default configuration when the DSP exits from a reset phase.
In SAM mode (Shared Access Mode), ARM (or DMA controller) and DSP can both access simultaneously to this shared memory space with ARM access resynchronized on DSP cycle-clock (3 times ratio required between ARM and DSP cycle clocks).
In HOM mode (Host Only Mode), the API RAM is dedicated to external access under the control of either the ARM or the DMA controller and therefore the access time is unconstrained.

2.4 XIO-Rhea

Internal and external peripherals are mapped on XIO or data memory spaces. These spaces are accessible through nXSTROBE [3:0] with a range of 2Kbytes for external peripherals allowing connecting up to:

- 6 external devices on program space
- 26 external devices on data space
- 31 external devices on I/O space
- internals peripherals are connected on I/O space or data memory space

2.4.1 External peripherals mapping - Program space

Device name	Strobe 0	Start address	Stop address	Size in bytes	Data access
not allocated	CS0	0000	07FF	2K	16
…	…	…	…	…	…
not allocated	CS5	3000	37FF	2K	16

2.4.2 External peripherals mapping - Data Space 1

Device name	Strobe 1	Start address	Stop address	Size in bytes	Data access
not allocated	CS6	3800	3FFF	2K	16
…	…	…	…	…	…
not allocated	CS15	7800	7FFF	2K	16

2.4.3 External peripherals mapping - Data Space 2

Device name	Strobe 2	Start address	Stop address	Size in bytes	Data access
UART_MODEM	CS16	8000	87FF	2K	8
not allocated	CS17	8800	8FFF	2K	16
…	…	…	…	…	…
not allocated	CS31	F800	FFFF	2K	16

2.4.4 External peripherals mapping – I/O Space

Device name	Strobe 3	Start address	Stop address	Size in bytes	Data access
RIF	CS0	0000	07FF	2K	16
MCSI	CS1	0800	0FFF	2K	16
not allocated	CS2	1000	17FF	2K	16
not allocated	CS3	1800	1FFF	2K	16
not allocated	CS4	2000	27FF	2K	16
CYPHER	CS5	2800	2FFF	2K	16
…	…	…	…	…	…
not allocated	CS30	F000	F7FF	2K	16
XI0-2-RHEA bridge	CS31	F800	F8FF	256	16
API Control		F900	F9FF	256	16
INTH		FA00	FAFF	256	16
not allocated		FB00	FBFF	256	
DMA controller		FC00	FCFF	256	16
not allocated		FD00	FDFF	256	
not allocated		FE00	FEFF	256	
not allocated		FF00	FFFF	256	

3. ARM MEMORY INTERFACE – FFFF:FB00

3.1 Memory interface register mapping

Register name	Address	Access	Reset value
nCS0 memory range	FFFF:FB00	12bits R/W	xxxx 1110 0011 1111
nCS1 memory range	FFFF:FB02	12bits R/W	xxxx 1110 0011 1111
nCS2 memory range	FFFF:FB04	12bits R/W	xxxx 1110 0011 1111
nCS3 memory range	FFFF:FB06	12bits R/W	xxxx 1110 0011 1111
nCS7 memory range	FFFF:FB08	12bits R/W	xxxx 0011 0000 0000
CS4 memory range	FFFF:FB0A	12bits R/W	xxxx 1110 0011 1111
nCS6 memory range	FFFF:FB0C	12bits R/W	xxxx 0011 0000 0000
API-Rhea control reg.	FFFF:FB0E	7bits R/W	xxxx xxxx x000 0000
Extra-control register	FFFF:FB10	9bits R/W	xxxx x?10 xx00 0000

Table 5: Memory interface registers

3.2 nCS0 memory range (Read / Write) – FFFF:FB00

Bit	Name	Function	Reset (dec)
4:0	WS	number of wait state for nCS0 memory access	31
6:5	DVS(1)	Device data size:(2) 00 = 8-bit, 01 = 16-bit, 10 = 32-bit	01
7	WE	0 = ABORT when write operation on a nCS0 1= Write enable	0
8	-	*Reserved*	0
9:11	DC	Dummy Cycles that should be inserted while bank switching from nCS0 to a faster memory bank.	7

(1) These values are read-only. They are sampled at system reset on CS5 for BIGEND and nCS4 and nCS3 for DVS

(2) CS0 device data size. This field is read-only. Its value is sampled at system reset on ROMSIZE pins

3.3 nCS1 memory range (Read / Write) – FFFF:FB02

Bit	Name	Function	Reset (dec)
4:0	WS	number of wait state for nCS1 memory access	31
6:5	DVS	Device data size:(2) 00 = 8-bit, 01 = 16-bit, 10 = 32-bit	01
7	WE	0 = ABORT when write operation on a nCS1 1= Write enable	1
8	-	*Reserved*	0
11:9	DC	Dummy cycles that should be inserted while bank switching from nCS1 to a faster memory bank.	7

3.4 nCS2 memory range (Read / Write) – FFFF:FB04

Bit	Name	Function	Reset (dec)
4:0	WS	number of wait state for nCS2 memory access	31
6:5	DVS	Device data size:(2) 00 = 8-bit, 01 = 16-bit, 10 = 32-bit	01
7	WE	0 = ABORT when write operation on a nCS2 1= Write enable	1
8	-	*Reserved*	0
11:9	DC	Dummy cycles that should be inserted while bank switching from nCS2 to a faster memory bank.	7

3.5 nCS3 memory range (Read / Write) – FFFF:FB06

Bit	Name	Function	Reset (dec)
4:0	WS	number of wait state for nCS3 memory access	31
6:5	DVS	Device data size:(2) 00 = 8-bit, 01 = 16-bit, 10 = 32-bit	01
7	WE	0 = ABORT when write operation on a nCS3 1= Write enable	1
8	-	*Reserved*	0
11:9	DC	Dummy cycles that should be inserted while bank switching from nCS3 to a faster memory bank.	7

3.6 CS4 memory range (Read / Write) – FFFF:FB0A

Bit	Name	Function	Reset (dec)
4:0	WS	number of wait state for CS4 memory access	31
6:5	DVS	Device data size:(2) 00 = 8-bit, 01 = 16-bit, 10 = 32-bit	01
7	WE	0 = ABORT when write operation on a CS4 1= Write enable	1
8	-	*Reserved*	0
11:9	DC	Dummy cycles that should be inserted while bank switching from CS4 to a faster memory bank.	7

3.7 nCS6 internal memory range (Read / Write) – FFFF:FB0C

Bit	Name	Function	Reset (dec)
4:0	WS	number of wait state for nCS6 memory access	0
6:5	DVS	Device data size:(2) 00 = 8-bit, 01 = 16-bit, 10 = 32-bit	10
7	WE	0 = ABORT when write operation on a nCS6 1= Write enable	1
8	-	*Reserved*	0
11:9	DC	Dummy cycles that should be inserted while bank switching from nCS6 to a faster memory bank.	0

3.8 nCS7 internal memory range (Read / Write) - FFFF:FB08

Bit	Name	Function	Reset (dec)
4:0	WS	number of wait state for nCS7 memory access	0
6:5	DVS	Device data size:(2) 00 = 8-bit, 01 = 16-bit, 10 = 32-bit	10
7	WE	0 = ABORT when write operation on a nCS7 1= Write enable	1
8	-	*Reserved*	0
11:9	DC	Dummy cycles that should be inserted while bank switching from nCS7 to a faster memory bank.	0

3.9 API-RHEA control register CTRL (Read / Write) – FFFF:FB0E

Bit	Name	Function	Reset
0	-	*Reserved*	0
1	ADAPT0	Rhea strobe 0 Access size adaptation: 0 = MCU access size using the target device size. 1 = MCU rhea access can be transformed in several Rhea transaction regarding the Arm access size and the Rhea target data width.	0
2	-	*Reserved*	0
3	ADAPT1	Rhea strobe 1 Access size adaptation: 0 = MCU access size using the target device size. 1 = MCU rhea access can be transformed in several Rhea transaction regarding the Arm access size and the Rhea target data width.	0
4	-	*Reserved*	0
5	ADAPTAPI	API access size adaptation: 1 = 32-bit access transformed into 2 16-bit API transaction.	0
6	DEBUG	Debug/visibility enable: 0 = ADD frozen when no external access performed. 1 = MCU address can be observed on the external memory ADD bus.	0

3.10 Extra control register CONF (Read / Write) – FFFF:FB10

Bit	Name	Function	Reset
0	1WSR	Write strobe control: 0 = RnW signal stays low for half a clock cycle during 0-WS write access. 1 = The 0-WS WRITE access to the arm memory are transformed into a 1 WS write access to increase the write access duration.	0
1	CTRL_WS0	0 = Accessing nCS0 with programmed wait-state. 1 = When accessing nCS0 the number of wait is set to 127, but access is ended when nready_mem goes low. If nready_mem stays at ‘1’ after 127 wait states an abort is generated.	0
2	CTRL_WS1	0 = Accessing nCS1 with programmed wait-state. 1 = When accessing nCS1 the number of wait is set to 127, but access is ended when nready_mem goes low. If nready_mem stays at ‘1’ after 127 wait states an abort is generated.	0
3	CTRL_WS2	0 = Accessing nCS2 with programmed wait-state. 1 = When accessing nCS2 the number of wait is set to 127, but access is ended when nready_mem goes low. If nready_mem stays at ‘1’ after 127 wait states an abort is generated.	0
4	CTRL_WS3	0 = Accessing nCS3 with programmed wait-state. 1 = When accessing nCS3 the number of wait is set to 127, but access is ended when nready_mem goes low. If nready_mem stays at ‘1’ after 127 wait states an abort is generated.	0
5	CTRL_WS4	0 = Accessing CS4 with programmed wait-state. 1 = When accessing CS4 the number of wait is set to 127, but access is ended when nready_mem goes low. If nready_mem stays at ‘1’ after 127 wait states an abort is generated.	0
7-6	*Reserved*		-
9:8	IBOOT_SRC	Boot memory selection: x0 = Defined by pin nIBOOT state (low = internal) 11 = External memory boot. 01 = Internal memory boot	10
10	nIBOOT_PIN	Value on the nIBOOT pin. (read only)	-
11	Disable_DU	Debug unit control: 0 = Debug-Unit is enabled and is recording Debug-data if processor is not running the abort mode 1 = Debug-Unit is disabled, it can be accessed as General Purpose RAM.	0
15-12	*Reserved*		-

4. MEMORY PROTECTION UNIT (MPU) – FFFF:FF00

4.1 Configuration 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
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

Table 6: MPU registers

Important:

MPU apply only to the INTERNAL Memory space.

4.2 MPU Overview

Within a memory space, the MPU allows defining memory sub-regions, each having a separate Read/Write protection attribute; this permits for partitioning the memory space into program instruction, system data, user data, stack …

The application program configures the MPU, which interfaces to the processor via the RHEA bus. The address bus directly issued from 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.

4.3 Block diagram

The figure below, shows the Memory Protection Unit within an TMS470R1 core and associated memory environment.

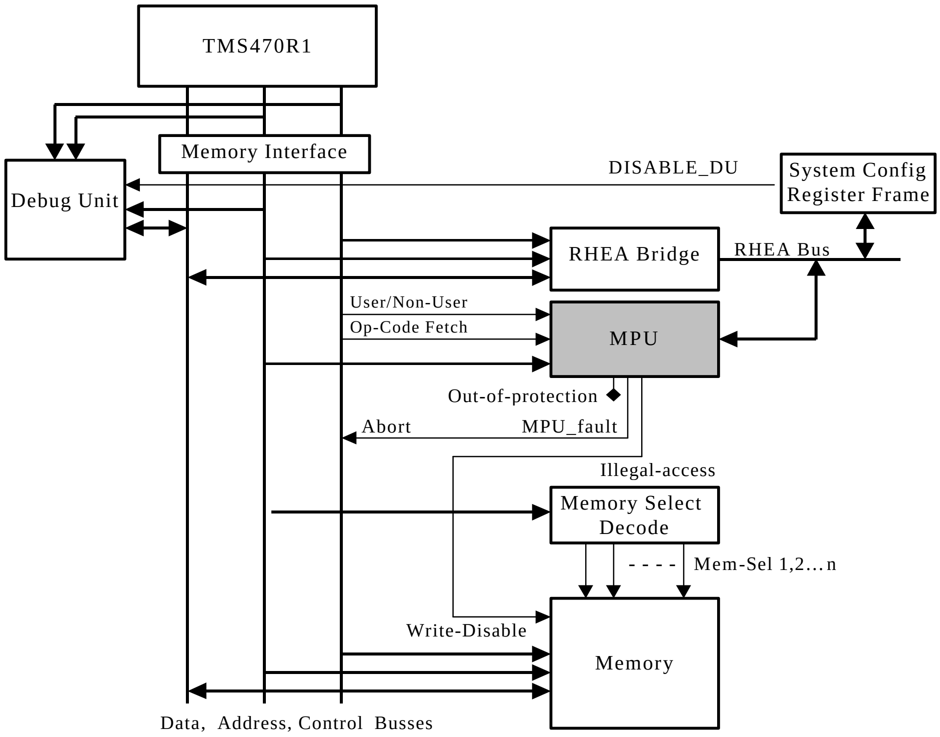


Figure 1: MPU within the Micro-Controller Environment

4.4 MPU Feature' s

The MPU allows for controlling:

- Up to four programmable protected regions within a maximum memory space of 512 k-byte
- A maximum protected size of 128 k-byte for each region (512 k-byte for 4 regions)
- A minimum granularity of 8 bytes
- A privileged-code memory region

For each region, memory mapped control registers define:

- Protected memory region base-address
- Starting/ending addresses within the protected memory region
- Protection mode (Non-User R/W, User Read-only, ROM, Privileged-region write…) applied to the memory sub-region 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 a non-authorized access to a memory region (i.e. 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 sub-region. (helpful for stack overflow monitoring)
- A MPU-fault signal which is the OR'ed 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 analyze.

4.5 MPU Description

The MPU hardware module is basically made of bus compares and logic decode associated to a programmable register frame that adjust the MPU configuration to application needs.

The control and status register-bits listed below are accessed through the RHEA bus:

- Four regions can be defined via the base-address register-bits.
- For each region, the start-address and the end-address register-bits define the lower and the upper bounds of the protected space.
- The control register-bits “Protection Mode” (PM 2,1&0) define the protection type associated to the protected space as listed, here after, in the table “Protection Mode Definition”.
- The bit Out-of-Protection (OPn_EN) enables the MCU to be informed of access above the upper bound of a protected space. This feature is useful for detection of stack under-flow/overflow.
- The Status register-bits save the condition (either illegal-access or out-of protection) that generated the fault, hence, providing useful information to the application program regarding the source which initiates the exception/interruption.

At **power-on-reset** All control & status register-bits are cleared to 0, disabling protection for all regions (user, non-user and privileged-region read/write access allowed). Then after the power-on reset is released, the application program can set the MPU register frame according to the application protection Scheme.

Important:

- ***Illegal-access signal is automatically and always enabled as soon as and for as long as a protection mode is selected (PMn[2-0] other than “000”) into the MPU Protection Mode register.***
- ***Out-of-protection signal can be enabled/disabled at any time from the OPn_EN bit into the MPU Protection Mode register.***
- ***MPU-Fault signal is active only when the “MPU_FAULT_EN” bit is set to 1 (MPU Control Register)***

This note below does not apply from CALYPSO C035-F751619 (Only for CALYPSO C05-F741979)

Important:

- ***A reset (warm start) does affect neither the MPU Protection Mode register nor the base, start and end-address registers. Only the MPU Status register is cleared to 0x00 after a warm reset.***

Once configured, the MPU, for every memory access, compares the incoming address values (bits 18 down to 3) from the MCU to the values stored into the base-address, the start-address, and the end-address registers. Then decoding the PMn 2,1&0 and OPn_EN bits, the MPU determines if the current access can be completed, if and illegal access must be generated or if an Out-of-Protection access must be indicated.

The memory protection mechanism does not impact the micro-controller speed performances since incoming addresses are compared in parallel while the memory selection is decoded.

When an access attempts to write a protected area, the illegal-access signal is asserted to:

- Disable the current write access.
- Flag the violation into the MPU Status register.
- Indicate the fault condition (MPU-FAULT-EN must be previously set to 1) to the processor (abort; see Note1).

When an Out-of-Protection address is detected (provided it was previously enabled; OPn_EN bit set to 1)

- The out-of protection access is flagged into the MPU Status register
- The MPU generates a fault condition (if enabled) to the processor (abort; see Note1).

Important:

> ***The illegal-access is monitored during write-accesses only. The Out-of-Protection is monitored during both read/ write accesses.***
>
> ***The illegal-access detection prevents from writing, whereas the Out-of-Protection does not.***

Note 1:
Typically, a fault condition (Illegal-access or Out-of-protection signals active***)*** initiates an MCU-abort operating mode, where the abort handler analyzes the fault using the MPU status register and additional available resources (e.g. Debug Unit) Then fixes the error prior to resume application program operations.

Note 2:
In emulation mode, when the “DBGACK” signal is active (see TMS470R1x User’s Guide), the protection mechanism is disabled

4.6 Protection Mode Definition

Each of the four possible memory regions has a separate protection attribute. The type of protection associated to a region is defined through the PMn[2-0] bits into the MPU Protection Mode register***.***

The MPU recognizes several operating modes and use this identification to build the protection scheme. Among the mode that can be detected are:

- User mode: This is the user mode as defined by the TMS470R1x user’s guide. (“most application program will run in user mode”)
- Non-User mode: It is also defined in the TMS470R1x user’s guide. (it’s “entered in order to service interrupts or exceptions or to access protected resources”)

- Privileged-region: This mode is specific to MPU architecture; It allows the application program to set memory regions (regions 1 & 2 as defined by the MPU register frame) to be a privileged memory space where the operational codes (OP-CODE) located in, benefits of write right to a given memory region.

The following table summarizes the protection mode that can be selected.

PMn 2	**PMn 1**	**PMn 0**	**Protection Mode**
0	0	0	***Protection Disabled***; User, Non-User & Privileged-region Read/Write allowed
0	0	1	Non-User Read/Write, User Read-Only (whatever fetched code location)
0	1	0	***Reserve***
0	1	1	***ROM***; Non-User Read-Only, User Read-Only (whatever fetched code location)
1	0	0	Writes are only authorized when performed from code fetched in protected region 1 (whatever MCU operating mode)
1	0	1	Writes are only authorized when performed from code fetched in protected region 1 or protected region 2 (whatever MCU operating mode).
1	1	0	***Reserved***
1	1	1	***Reserved***

Table 7: **Protection Mode Definition**

4.7 Memory protection example

4.7.1 Memory Map With Protected Region - Example 1

The 4 figures below show examples of memory protection configuration.

The Out-Of-Protection is indicated from the end limit of the protected area up to the end of the region defined by the base-address.

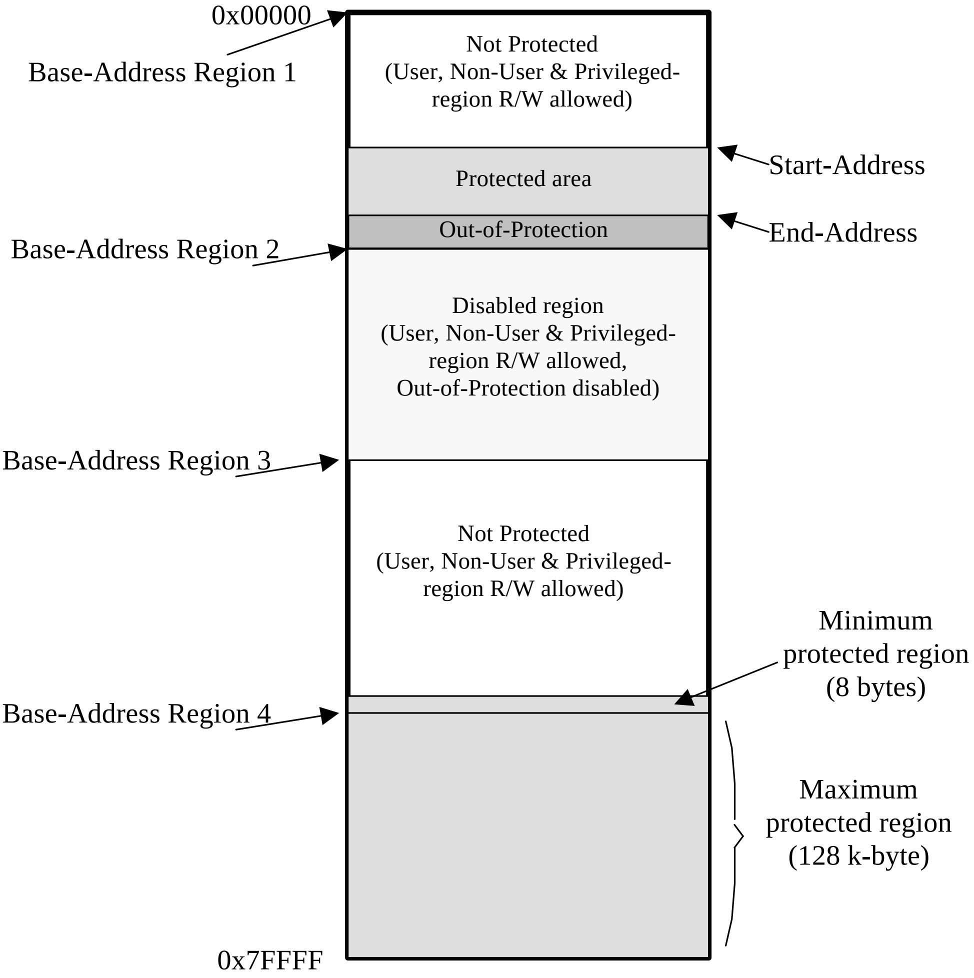


Figure 2: Memory Map With Protected Region (Example 1)

4.7.2 Memory Map with Protected Region - Example 2

Out-Of-Protection is not monitored when the program goes through a protected space. As shown in the figure below, if protected spaces are declared within the same base address, then Out-Of-Protection is only flagged outside the protected area, in addition the flags indicate all protected area that have been past

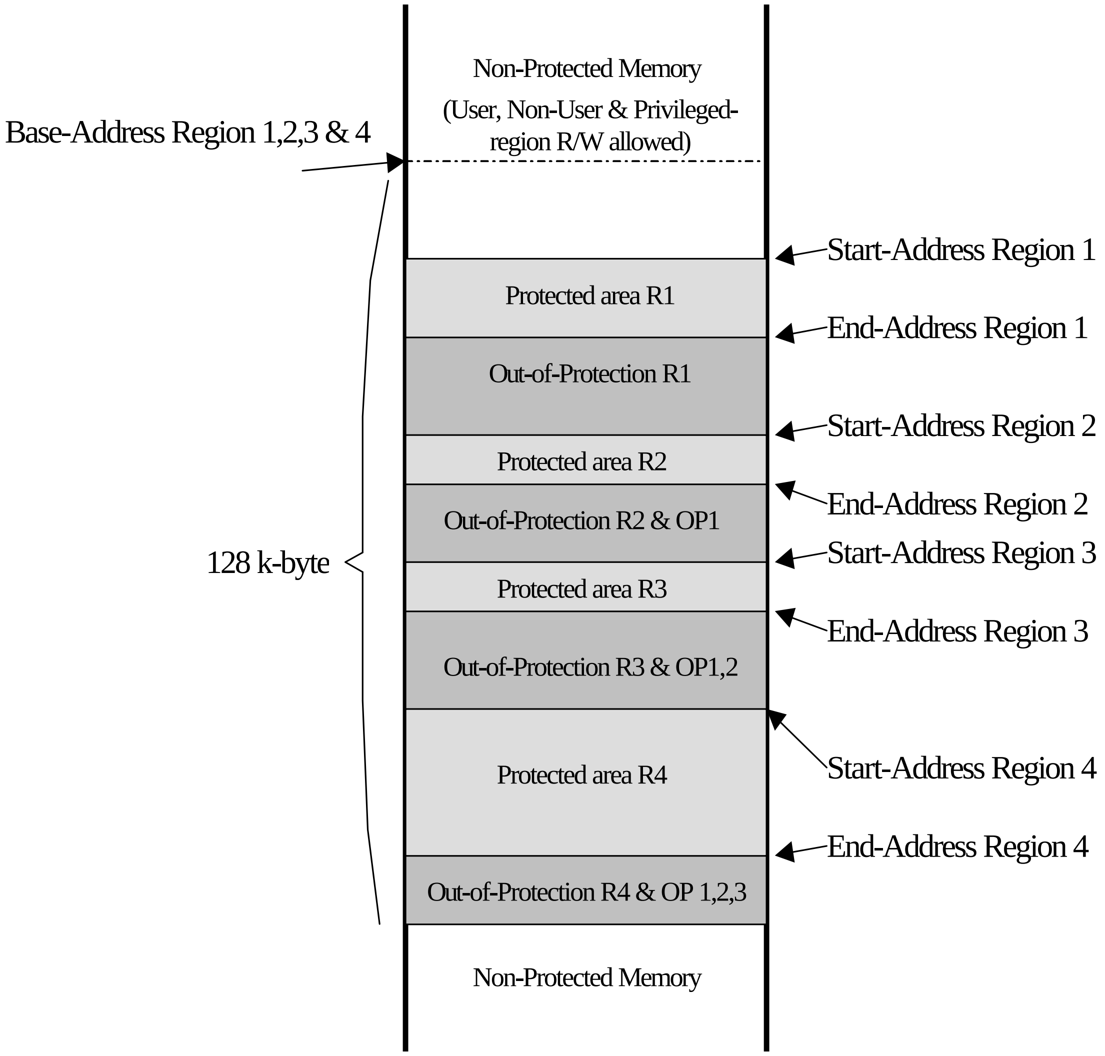


Figure 3: Memory Map with Protected Region (Example 2)

4.7.3 Memory Map With Stack Over/Under-Flow Definition - Example 3

The figure below shows an example of stack overflow/under-flow definition. Two 8-byte memory-areas (available minimum granularity) are reserved (non-usable memory) below and above the stack region. An access to the area “out-of-protection R1” will be recognized as a stack under-flow. An access to the area “out-of-protection R2” will be recognized as a stack overflow.

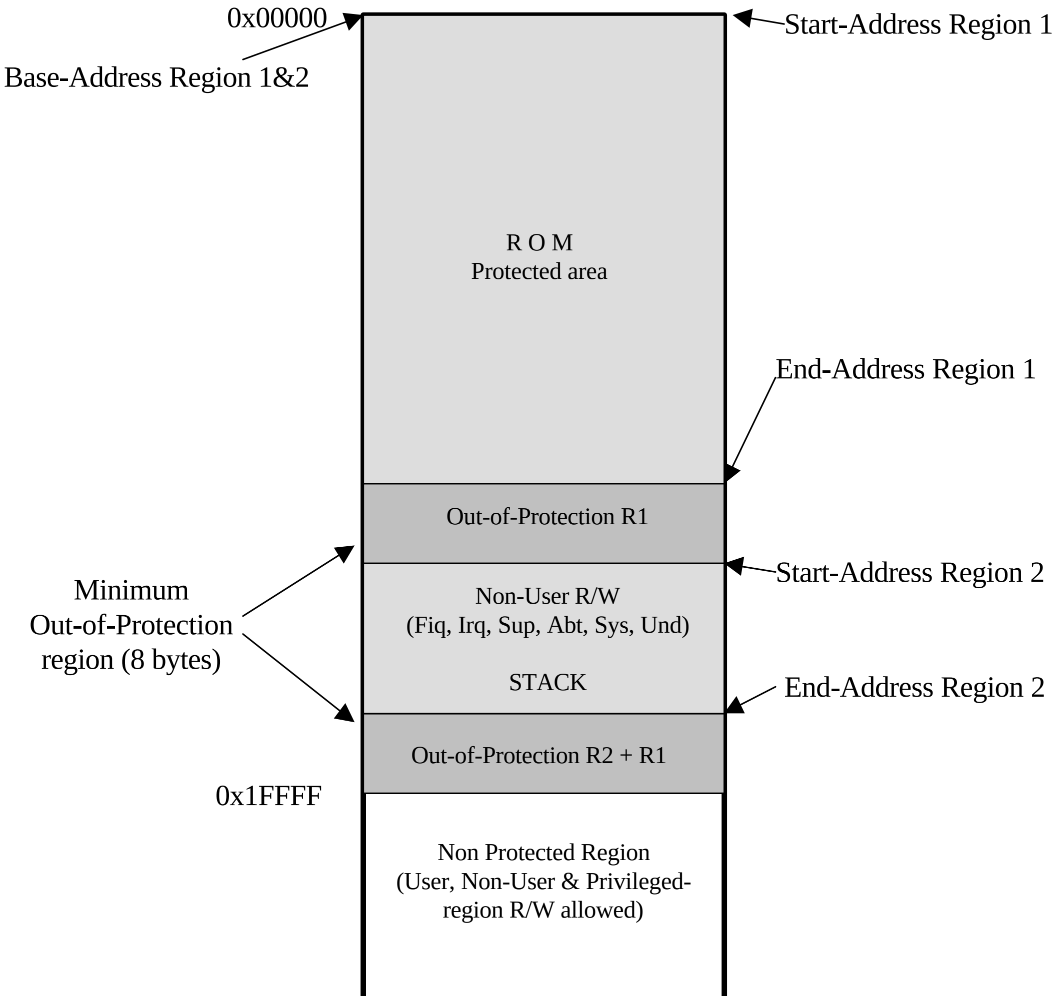


Figure 4: Memory Map With Stack Over/Under-Flow Definition (Example 3)

4.7.4 Privileged Memory - Example 4

The figure below illustrates a configuration that defines privileged and protected memory region. In the below example, only the write instructions fetched from the protected region1 or from Out-of-Protection region2 are authorized to write into the protected region2. If the write code is issued from an other memory area (e.g: region 3), then an illegal-access is asserted. The privileged region (protected region1 + Out-of-Protection region2) is also qualified as ROM access.

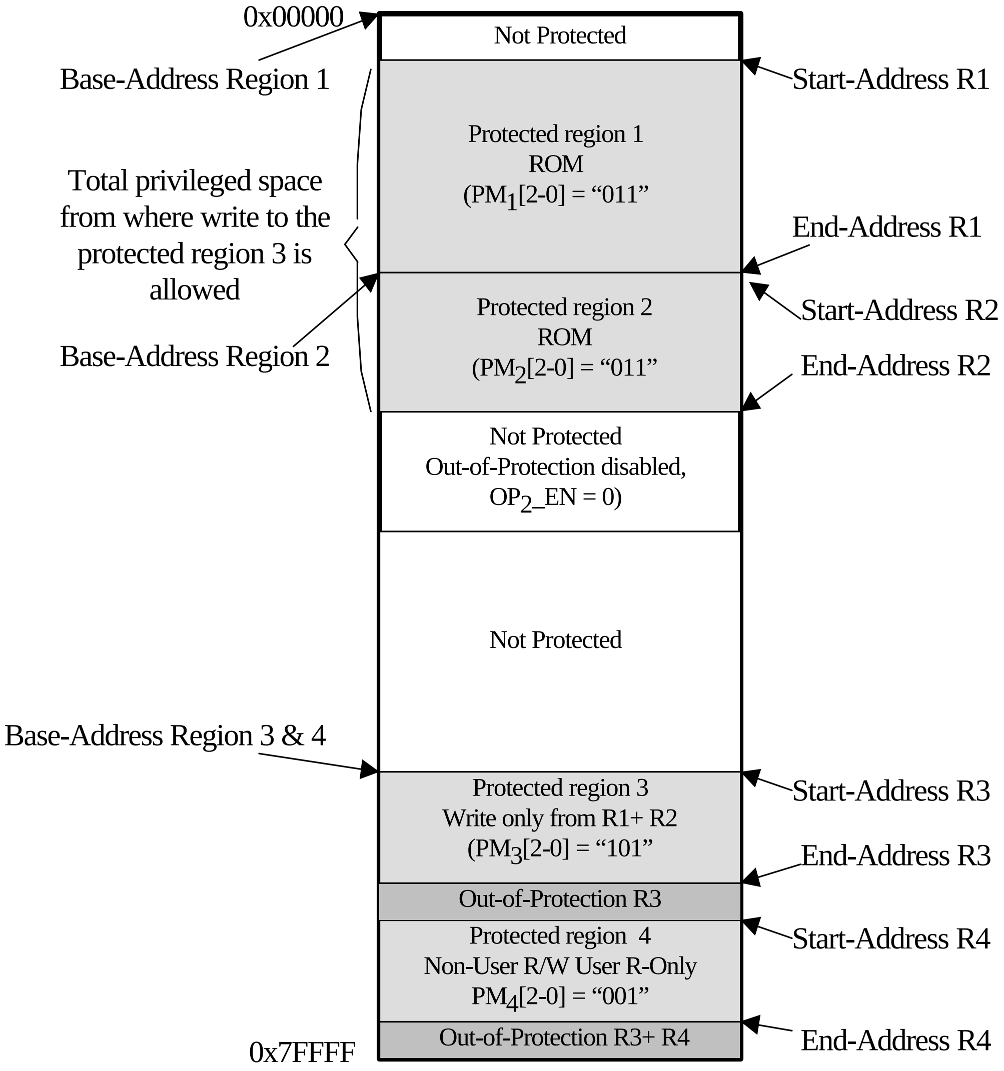


Figure 5: Privileged Memory (Example 4)

4.8 Mpu control register frame

4.8.1 Miscellaneous

The MPU module contains eleven 16-bit memory mapped registers that can be accessed through a RHEA bus.

- The MPU register’s mapping to the MCU’ s memory space is device dependent, hence defined at the chip/system level. Within the register frame, the physical address of registers is the Start address (defined by the system) + Offset address (given in the next tables).

- The status register saves illegal-accesses as well as out-of protection fault signatures.

- The controls 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 8 registers are partitioned by region (2 registers by region); each register’ s pair defining the Base & Start address and the End Address for its associated region.

- The registers can be read at any time without affecting the on going operations. All register can be written accordingly to the bit definition as discussed in the next section. The status register is a clear-only register (only write to 0)

4.8.2 Table summary

RegisterName	**Offset Addr**	**Bits**															
		15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MPU_ST	0x00	Reserved								OP4	OP3	OP2	OP1	IA_R4	IA_R3	IA_R2	IA_R1
MPU_PM	0x02	Reserved				PM4 2	PM4 1	PM4 0	PM3 2	PM3 1	PM3 0	PM2 2	PM2 1	PM2 0	PM1 2	PM1 1	PM1 0
- - -	0x04	Reserved															
- - -	0x06	Reserved															
MPU_CTL	0x08	Reserved											OP4_EN	OP3_EN	OP2_EN	OP1_EN	MPU_ FAULT EN
MPU_ B&ST1	0x0A	Base1 1	Base1 0	Start1 13	Start1 12	Start1 11	Start1 10	Start1 9	Start1 8	Start1 7	Start1 6	Start1 5	Start1 4	Start1 3	Start1 2	Start1 1	Start1 0
MPU_ END1	0x0C	Reserved		End1 13	End1 12	End1 11	End1 10	End1 9	End1 8	End1 7	End1 6	End1 5	End1 4	End1 3	End1 2	End1 1	End1 0
MPU_ B&ST2	0x0E	Base2 1	Base2 0	Start2 13	Start2 12	Start2 11	Start2 10	Start2 9	Start2 8	Start2 7	Start2 6	Start2 5	Start2 4	Start2 3	Start2 2	Start2 1	Start2 0
MPU_ END2	0x10	Reserved		End2 13	End2 12	End2 11	End2 10	End2 9	End2 8	End2 7	End2 6	End2 5	End2 4	End1 3	End2 2	End2 1	End2 0
MPU_ B&ST3	0x12	Base3 1	Base3 0	Start3 13	Start3 12	Start3 11	Start3 10	Start3 9	Start3 8	Start3 7	Start3 6	Start3 5	Start3 4	Start3 3	Start3 2	Start3 1	Start3 0
MPU_ END3	0x14	Reserved		End3 13	End3 12	End3 11	End3 10	End3 9	End3 8	End3 7	End3 6	End3 5	End3 4	End3 3	End3 2	End3 1	End3 0
MPU_ B&ST4	0x16	Base4 1	Base4 0	Start4 13	Start4 12	Start4 11	Start4 10	Start4 9	Start4 8	Start4 7	Start4 6	Start4 5	Start4 4	Start4 3	Start4 2	Start4 1	Start4 0
MPU_ END4	0x18	Reserved		End4 13	End4 12	End4 11	End4 10	End4 9	End4 8	End4 7	End4 6	End4 5	End4 4	End4 3	End4 2	End4 1	End4 0

Table 8: MPU Control & Status Register Frame

4.9 MPU_ST: Status register (Read) – FFFF:FF00

The MPU_ST register indicates which event has initiated the fault.

Bit	Name	Function	Reset
0	IA_R1	1 = Illegal-access to the protected region 1	0
1	IA_R2	1 = Illegal-access to the protected region 2	0
2	IA_R3	1 = Illegal-access to the protected region 3	0
3	IA_R4	1 = Illegal-access to the protected region 4	0
4	OP1	1 = Out-of-Protection access within region 1	0
5	OP2	1 = Out-of-Protection access within region 2	0
6	OP3	1 = Out-of-Protection access within region 3	0
7	OP4	1 = Out-of-Protection access within region 4	0
15:8	*Reserved*	*No effect*	-

4.10 MPU_CTL: Control register (Read / Write) – FFFF:FF08

The MPU_CTL register controls the MPU_FAULT signal assertion and for each region enables the Out-of-Protection monitoring.

Bit	Name	Function	Reset
0	MPU_FAULT_EN	This read/write bit enables the MPU_FAULT signal. 0 = Memory protection as well as the out-of-protection supervision is active according to the value of the Opn_EN and PMn[2-0] bits. The status register is still recording the MPU' s events and remains available for reading, however the signal MPU_FAULT is locked to a low level not passing the fault indication to the processor (e.g. abort not generated).. 1 = Any fault flagged into the status register initiates a MPU_FAULT signal transition (high level) indicating the fault occurrence to the processor (e.g. abort generation).	0
1	OP1_EN	Out-of-Protection supervision within the protected region 1. 0 = disable 1 = enable	0
2	OP2_EN	Out-of-Protection supervision within the protected region 2. 0 = disable 1 = enable	0
3	OP3_EN	Out-of-Protection supervision within the protected region 3. 0 = disable 1 = enable	0
4	OP4_EN	Out-of-Protection supervision within the protected region 4. 0 = disable 1 = enable	0
15:5	*Reserved*	*No effect*	-

4.11 MPU_PM: Protection Mode Register (Read / Write) – FFFF:FF02

4.11.1 Register bit mapping

The MPU_PM register defines the protection associated to four possible protected regions.

Bit	Name	Function	Reset
2:0	PM1 (2:0)	Protection mode associated to region 1	0
5:3	PM2 (2:0)	Protection mode associated to region 2	0
8:6	PM3 (2:0)	Protection mode associated to region 3	0
11:9	PM4 (2:0)	Protection mode associated to region 4	0
15:12	*Reserved*		-

4.11.2 Protection mode summary

The following table summarizes the protection mode that can be selected.

PM2	PM1	PM0	Protection Mode
0	0	0	***Protection Disabled***: User, Non-User & Privileged-region Read/Write allowed
0	0	1	Non-User Read/Write, User Read-Only (whatever fetched code location)
0	1	0	***Reserved***
0	1	1	***ROM***: Non-User read-only, User read-only (whatever fetched code location)
1	0	0	Writes are only authorized when performed from code fetched in protected region 1 (whatever MCU operating mode)
1	0	1	Writes are only authorized when performed from code fetched in protected region 1 or protected region 2 (whatever MCU operating mode). See example figure 6.
1	1	0	***Reserved***
1	1	1	***Reserved***

Table 9: Protection Mode Definition

4.12 MPU_B&STn: Base & Start Address – region n – (Read / Write)

The MPU_B&ST(n) registers define Base & Start address of the protected memory region (n).

Bit	Name	Function	Reset
13:0	STARTn (13:0)	Start address for the corresponding protected memory region[n]. The fourteen bits (13:0) start-address is compared to the MCU address bus (16:3).	0
15:14	BASEn (1:0)	Base address for the corresponding protected memory region[n]. The two bits (15:14) base address is compared to MCU address bus (18:17).	0

4.13 MPU_ENDn: End Address Definition – region n - (Read / Write)

The MPU_End(n) registers define the End address of the protected memory region (n).

Bit	Name	Function	Reset
13:0	ENDn (13:0)	End address for the corresponding protected memory region[n]. The fourteen bits (13:0) end-address is compared to the MCU address bus (16:3).	0
15:14	*Reserved*	No effect	-

5. DEBUG UNIT (DU) – 03C0:0000

5.1 Debug Unit Overview

The Debug Unit is a hardware resource intended to provide additional support to a software abort-handler. The DU provides ***64*** stages deep history table of the last memory accesses prior entering the abort mode, then permitting analysis of previous bus transaction' s.

The DU is an autonomous function that does not need to be configured, hence, does not interfaces to a control bus such as RHEA. The DU is connected directly to the processor busses (Address & Control) from where it collects the data, and to the memory interface system where the saved history table can be read.

The figure, below, shows the Debug Unit connected to a TMS470R1 core.

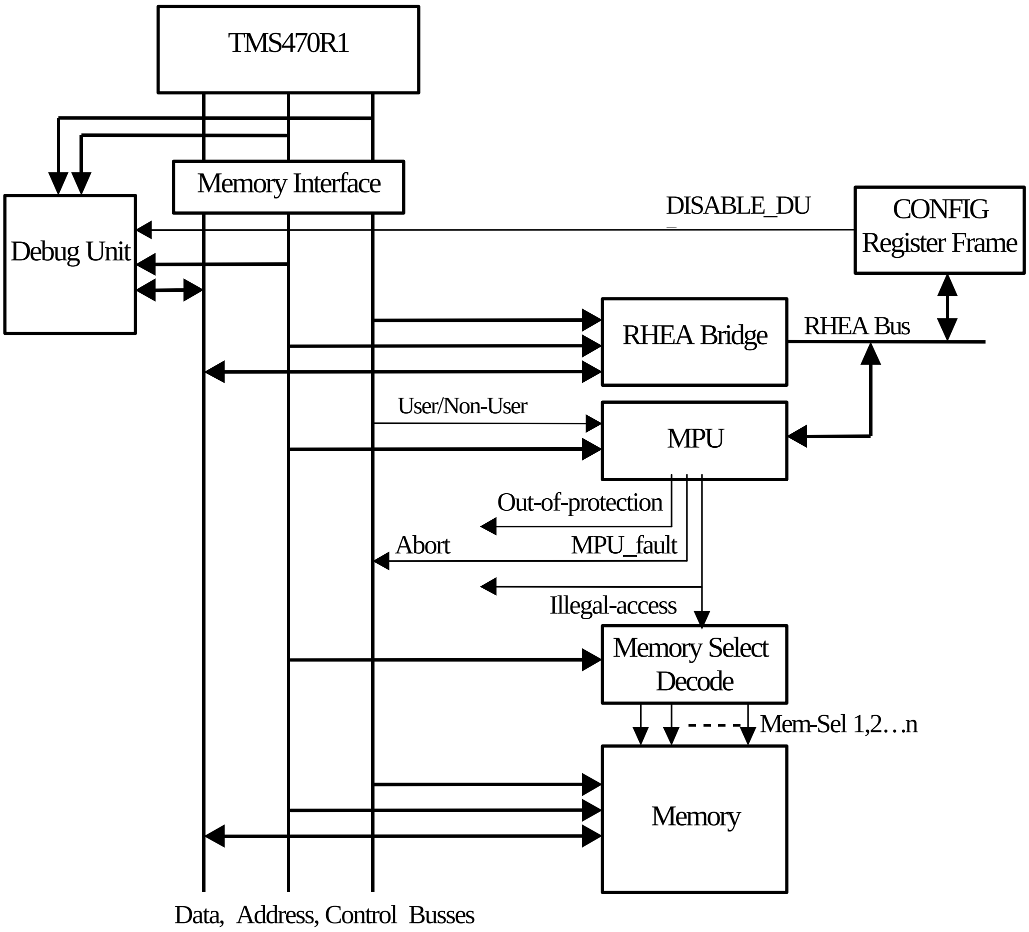


Figure 6: Debug Unit connection to the Micro-Controller.

5.2 Debug Unit Feature' s

The Debug Unit offers the following:

- Sixty-four 32-bit words deep FIFO register file
    - 26-bit Processor Address and 8 Processor control signals recorded (nM[1:0], MAS[1:0], nEXEC, nOPC, nMREQ, nRW)
    - Continuous storage for every processor fetch (either instruction or data)
    - Data record automatically frozen upon switch 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 debug function is disabled

5.3 Debug Unit Description

Basically the Debug Unit hardware is a 64 words dual port (separate read/write bus) memory with additional control that make the last written address being the read base-address.

Each input word is 32 bits wide allowing for storing 26 address-bits and 6 additional bits that inform on the processor operating state. (See: Debug-Data Organization). The read-data bus is output in a 32-bit format.

The Debug Unit does not impact the micro-controller speed performances; it is implemented as a stand-alone module spying the MCU busses and not participating to the data processing.

- There is no configuration to setup, the only available control is an input signal “DISABLE_DU”, which allows to enable/disable the debug-Unit function.
- The enable/disable signal can controlled from a register bit (typical use) located into the system configuration frame (§ 4.10)
- When the DU is enabled and for as long as the abort mode is not detected, the data input onto the “debug-data bus is partially encoded then synchronously stacked into the 64-word register file
- A 64-state rollover counter generates the write addresses; for every access to the system-memory, the counter decrements pointing the debug-memory location where the bus transaction data is written.
- The input signal nCLK_DU controls the counter decrement (nCLK_DU /rising edge) and the debug-memory synchronous write (nCLK_DU /falling edge). Typically nCLK_DU connects the TMS470R1 processor clock “ECLK”; note that in this case the Wait-State’s are not recorded.
- When the counter rollovers, debug-memory locations are sequentially updated overriding (old data lost) the previous debug-memory content.

The Debug Unit then continuously collects data until:

- Either the processor ABORT mode is entered; this is decoded when the TMS470R1 processor output signals nM[3] and nM[2] are at the value 1 and 0 respectively.
- or the DISABLE_DU signal is asserted high (see Debug Unit Disable/Enable Control)
- or the TMS470R1 debug state is activated (DBGACK_DU signal high; see emulation literature).

When the abort mode is entered, the automatic-write is locked and the Debug-memory content is frozen. In this state, the debug unit acts as a standard static RAM block where the data previously collected is available to the software abort-handler for working out the abort cause.

In abort mode the counter value is frozen pointing the last write-addressed location.

- The counter value is used as offset. It is added to the address value supplied by the MCU to generate the read or write addresses, in such a way that a read at the location 0x00 (address from MCU) returns the last data written (0x00 + write-add-pointer). A read to the location 0x01 returns the data before the last written, and so on… (See: Debug-Data Organization).
- The debug-memory data-input bus is switched to the memory-interface output-data bus
- The write function (write-enable, write-byte, and write-clock) is controlled from the MCU.
- The debug-memory data-output bus is valid and can be read from the MCU.

Important:
The debug- memory should be read only when DU is disabled or if abort mode is active. Reading the DU while collecting data returns an undefined value. The write from the processor is disabled and has no effect when the DU is collecting data.

When the abort mode is released (once abort-handler has processed), the DU re-starts recording data; the 64-state rollover counter starts counting from where it has been stopped.

Note 1:
The 64-state rollover counter value is initialized at “000000” upon reset (nRST_DU low). There is no other means of setting the counter value.

5.4 Debug Unit Enable/Disable Control

The debug unit should be enabled at reset by setting the “DISABLE_DU” bit, which is the bit 11 of the “Extra control register CONF” (§ 4.10) of memory interface (FFFF:FB10) to a low level (logic 0). The Debug Unit then starts collecting data until the abort mode is entered (see behavior description above).

If the application does not require such a debug facility or in order to save power consumption during critical application phases, the debug unit can be disabled by asserting the “DISABLE_DU” signal to a high level (bit 11, Extra-control-register CONF, § 4.10).

When the debug unit is disabled, the debug-memory can be used as general-purpose RAM.

Disabling the Debug Unit

- Locks the automatic-write mechanism and sets the debug-memory write control to the memory-interface (data, address, write-enable, write-byte and write-clock signals).
- Freezes the 64-state rollover for as long as the DISABLE_DU signal is high. This nullifies the offset pointer-value effect and lets the MCU address bus to directly control the read/write location.

Re/enabling the DU (DISABLE_DU signal set to a low level) re/starts the automatic-write mechanism, and make the DU collecting data again.

5.5 Debug-Data Organization

The debug-data is collected from the processor address and control buses. It is partially encoded before being recorded into the debug-memory.

The **input signals** and the coded debug-data (recorded-data) are listed below:

- **MAS[1:0]:** Memory Access Size; these 2 signals are coded to provide a single bit (Byte/Non-Byte) information.

TMS470R1 “**MAS[1:0]**” definition	Debug-Data “Byte” definition
00 : Byte 01 : Half-Word 10 : Word 11 : Reserved	0 : Non-Byte 1 : Byte

- **nOPC**: OP-Code fetch indicator, **nRW**; not Read/Write and , **nMREQ**; not Memory Request; these 3 signals are coded and give a 2 bits MCU Access-Type data.

TMS470R1 “**nOPC**” definition	Debug-Data “Access-Type” definition
0 : Fetching instruction 1 : Not fetching instruction	00 : Read Instruction 01 : CPU Internal Cycle 10 : Read Data 11 : Write Data
TMS470R1 “**nRW**” definition	
0 : Read 1 : Write	
TMS470R1 “**nMREQ**” definition	
0 : Memory Request 1 : Not Memory Request	

- **nEXEC**; EXECuted instruction indicator; this signal is directly recorded

TMS470R1 “**nEXEC**” definition	
0 : Instruction being executed 1 : Instruction not being executed	0 : Instruction being executed 1 : Instruction not being executed

- **nM[1:0]**; TMS470R1 processor operating Mode; these 2 signals are directly recorded

TMS470R1 “**nM[1:0]**” definition	
00 : Supervisor / Abort / Undef./ System 01 : IRQ 10 : FIQ 11 : User	00 : Supervisor / Abort / Undef./ System 01 : IRQ 10 : FIQ 11 : User

- ***A[25:0]*** ; TMS470R1 processor Address; Among the 26-bit address bus, the **25-MSB signals** are directly recorded whereas the LSBIT “***A[0]***“ Go through a special treatment as described below.

  When the Non-Byte is flagged (bit Byte=0), the address LSBIT “***A [0]***“ is ignored, the “***MAS [1]***” signal is recorded instead to indicate if the current access is an half-word (16-bit) or word (32) type.

5.6 Recorded data examples

No	Byte	Access Type	nEXEC	nM[1:0]	25-MSBIT Address	LSBIT-Address MAS[1]	
1	1	11	0	11	0101010101010101010101010	1	A[0]
2	0	00	0	10	1101010101010101010101010	1	MAS[1]
3	0	10	1	01	0001010101010101010101011	0	MAS[1]

1. Indicates a **BYTE DATA WRITE** access at address **0x1555555** while the processor is in **USER** mode and effectively **EXECUTE** the instruction in the execute pipe stage.
2. Illustrates a **WORD(32-bit) INSTRUCTION FETCH (32-BIS)** access at address **0x3555554** while the processor is in **FIQ** mode and effectively **EXECUTE** the instruction in the execute pipe stage.
3. Shows a **HALF-WORD(16-bit) DATA READ** access at address **0x0555556** while the processor is in **IRQ** mode and does **NOT EXECUTE** the instruction in the execute pipe stage.

5.7 Debug Unit memory map (03C0:0000)

Generated Write addresses	**Bits**						**Read address *[location]***
	31 Byte	30:29 Acc-Typ	28 nEXEC	27 nM[1:0]	26:1 A[25-1]	0 A[0] / MAS[1]	
Counter	transaction /time n						03C0:0000 *[0x00]*
Counter+1	transaction /time n-1						03C0:0001 *[0x01]*
Counter+2	transaction /time n-2						03C0:0002 *[0x02]*
Counter+3	transaction /time n-3						03C0:0003 *[0x03]*
Counter+4	transaction /time n-4						03C0:0004 *[0x04]*
. . .	. . .						. . .
Counter+62	transaction /time n-62						03C0:002D *[0x2D]*
Counter+63	transaction /time n-63						03C0:002E *[0x2E]*
Counter+64	transaction /time n-64						03C0:002F *[0x2F]*

Table 10: DU memory map

6. GPRS ENCRYPTION ALGORTHYM (GEA1 & 2) – FFFF:C000

6.1 GEA Miscellaneous

In GPRS mode, the data confidentiality is performed by a ciphering function (GPRS Encryption Algorithm). The ciphering is executed within the LLC upper layer. According to the option negotiated with the network the GEA mode 1 or mode 2 may be selected.

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 shall support both acknowledge and unacknowledged mode and implement a FCS according to the mode used.

The HW block provided support the computation of the FCS according to the LLC frame to send/received as well as the ciphering/deciphering GEA mode 1 and 2. These procedures are described in 01.61, 04.64 GSM recommendations and detailed in GSM MoU documents.

6.2 LLC overview

LLC is considered as a sub-layer of layer 2 in the ISO 7-layer model. The purpose of LLC is to convey information between layer-3 entities in the MS and SGSN.

The frame formats defined for LLC are based on those defined for LAPD and RLP, and the LLC procedures are modeled upon the concepts of HDLC.

LLC shall support 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, LLC provides service to the GPRS Mobility Management (GMM) protocol, and to the SMS protocol.

LLC shall support both acknowledged and unacknowledged data transfers:

6.3 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:

1. **Protected mode** in which the FCS field protects the frame header and information field; and
2. **Unprotected mode** in which the FCS field protects the frame header and only the first N202 octets of the information field.

6.4 Acknowledged operation

With this type of operation, layer-3 information is transmitted in order 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 the case of errors that cannot be corrected by the logical link control layer, a report to GPRS mobility management shall be made.

6.5 Frame format

Each LLC frame consists in a header, an information field and a FCS.

- Header has a size of a minimum of 2 bytes up to a maximum of 37 bytes. It is split into an address field (1 byte) and a control field (up to 36 bytes).
- Information field has a variable length with a maximum size of N201[1] bytes (maximum size is negotiated).
- FCS has a fixed size of 3 bytes.

LLC Frame

H	information field	FCS

Identification of the type of the frame and processing applied on it, are based on the frame header. It is used by the MCU in order to know the FCS and ciphering configurations and to split the header and the information fileds.

6.5.1 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 & 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 that is performed over the frame header and the first N202 bytes of the information field only. The information over which the CRC is calculated is referred to as the dividend. First bit of the dividend is the highest-order term in the calculation.

CRC calculation shall be done before ciphering at the transmitting side, and after deciphering at the receiving side.

Note:
The CRC shall be the ones complement of the sum (modulo 2) of:

- the remainder of $x^k (x^{23} + x^{22} + x^{21} + \dots + x^2 + x + 1)$ divided (modulo 2) by the generator polynomial, where k is the number of bits of the dividend; and
- 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:

$$G(x) = X^{24} + X^{23} + X^{21} + X^{20} + X^{19} + X^{17} + X^{16} + X^{15} + X^{13} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X^{1}$$

The result of the CRC calculation is placed within the FCS field defined in chapter 6.5, with highest order terms transmitted first.

---

[1] N201 size depends on the SAPI chosen. Max size is 1520 bytes and is defined in [2]

6.5.2 Ciphering

LLC shall provide user data confidentiality by means of a ciphering function. This ciphering is applied on the information field and the FCS but **not on the frame header**. There are currently two ciphering algorithm defined by the ETSI (see [4] and [5]). Both of them are supported by the module defined below (see chapter 6.11.1 and 6.12 for algorithm selection).

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 message key.

DIRECTION: This defines the direction (1bit) of the data transmission (uplink/downlink).
Set to 0 if the direction of LLC frame transmission is from the MS to the SGSN.
Set to 1 if the direction of 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 are using the same input variables, only their internal processing differs.

6.5.3 UI frames

There are two modes for the FCS computation of the UI frames.

1. The first mode is the classical one where FCS is applied on the header plus all the information bits (protected mode)
2. The second mode consist in applying the FCS only on the header and the N202 information bits (unprotected mode). If the length of the information field is less than N202 octets then the FCS shall 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 non-protected mode and no ciphering are selected (second mode), the MCU has two possibilities:

1. MCU writes the LLC-PDU header and the following N202 information bytes into the input buffer. The LLC-PDU size 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 for data transfer.
2. Or the MCU writes the LLC-PDU header and the full LLC-PDU information field into the input buffer. The LLC-PDU size given to the module is equal to LLC-PDU header + LLC-PDU information field. However, the MCU specifies to the module that the non-protected mode is used. Therefore, the module computes its FCS only on LLC-PDU header+N202[2] bytes. This method is optimal if the MCU is using the same data flow for protected mode or not.

In both cases, same buffer formats are used and first byte shifting is still applied if specified.

---

[2] The maximum number of octets in the layer-3 unit data PDU header (N202) is an LLC layer parameter. The N202 maximum value shall be 5 for LLC version number 0. See [2]

6.5.4 GEA Processing

The MCU transfers, through the Rhea Bus, part or the entire LLC frame to the module's memory. The module can then start its processing.

It is possible for the MCU to 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 (i.e.: X, Y, Z registers needed for the encryption algorithm and the number of bytes processed) in order to restart the processing from where it was stopped, whenever wanted. INTERFACE DESCRIPTION

6.6 MEMORY MANAGEMENT

All external access to the memory inside the GEA module are done through the RHEA bus. From the outside the memory is seen as a register (see address in chapter 6.16), the address increment is managed by the GEA module.

Management is done by the use of three address pointers. Each time a word is written into the data register the "write pointer" is incremented. Once all the words are written and the START bit is activated (see chapter 6.8) the data is processed by the GEA module and the "processed pointer" is incremented. When all the data are processed the WORKING signal (see chapter 6.9) goes down and a maskable interrupt is sent. The data can then be read by the MCU, and the "read pointer" is incremented each time a word is read. It is not mandatory to read the data. The MCU can start writing again and the "read pointer" will take the value of the "processed value". The encryption module directly manages these pointers.

8 and 16 bits accesses can be done to the memory through two different registers DATA16 and DATA8 register (see chapters 6.16 & 6.16.4).

Up to 1600 bytes can be written into the memory, therefore a whole frame can be processed in one shot. **Warning:** if the OS bit is set (see chapter 6.11) only 1599 bytes can be written.

6.7 GEA register mapping

Register	Address	access	reset value
CNTL_REG	FFFF:C000	6 bits R/W	???? ???? ??00 0011
STATUS_REG	FFFF:C002	2 bits R	???? ???? ???? ??00
STATUS_irq_REG	FFFF:C004	1 bit R	???? ???? ???? ???0
CONF_UL_REG1	FFFF:C006	8 bits R/W	???? ???? 0000 0?00
CONF_UL_REG2	FFFF:C008	16 bits R/W	0000 0000 0000 0000
CONF_UL_REG3	FFFF:C00A	16 bits R/W	0000 0000 0000 0000
CONF_UL_REG4	FFFF:C00C	16 bits R/W	0000 0000 0000 0000
CONF_UL_REG5	FFFF:C00E	16 bits R/W	0000 0000 0000 0000
CONF_DL_REG1	FFFF:C010	8 bits R/W	???? ???? 0000 0000
CONF_DL_REG2	FFFF:C012	16 bits R/W	0000 0000 0000 0000
CONF_DL_REG3	FFFF:C014	16 bits R/W	0000 0000 0000 0000
CONF_DL_REG4	FFFF:C016	16 bits R/W	0000 0000 0000 0000
CONF_DL_REG5	FFFF:C018	16 bits R/W	0000 0000 0000 0000
KC_REG1	FFFF:C01A	16 bits R/W	0000 0000 0000 0000
KC_REG2	FFFF:C01C	16 bits R/W	0000 0000 0000 0000
KC_REG3	FFFF:C01E	16 bits R/W	0000 0000 0000 0000
KC_REG4	FFFF:C020	16 bits R/W	0000 0000 0000 0000
FCS_UL_REG1	FFFF:C022	16 bits R/W	0000 0000 0000 0000
FCS_UL_REG2	FFFF:C024	16 bits R/W	0000 0000 0000 0000
FCS_DL_REG1	FFFF:C026	16 bits R/W	0000 0000 0000 0000
FCS_DL_REG2	FFFF:C028	16 bits R/W	0000 0000 0000 0000
DATA16_REG	FFFF:C030	16 bits R/W	???? ???? ???? ????
DATA8_REG	FFFF:C032	16 bits R/W	???? ???? ???? ????

Table 11: GEA registers

6.8 Control Register: CNTL_REG (Read / Write) – FFFF:C000

Bit	Name	Function	Reset
0	NRESET_UL	Reset uplink-module (ciphering + FCS). 0 = Abort current processing and reset internal variables. *This bit is set by internal logic.*	1
1	NRESET_DL	Reset downlink-module (deciphering + FCS). 0 = Abort current processing and reset internal variables. *This bit is set by internal logic.*	1
2	START	Start the module. 0 = The uplink/downlink state machine that is started is selected with the *b_uldl* bit. *This bit is cleared by internal logic.*	0
3	CLOCK_EN	Enable the internal clock 0 = Clock stopped 1 = Clock enabled	0
4	UL_DL	Select processing: 0 = uplink (ciphering+FCS) 1 = downlink (deciphering+FCS check)	0
5	IT_EN	Enable interrupt: 0 = No interrupt 1 = Interrupt generated when processing completed.	0
15:6	Unused	-	-

Notes:

- NRESET_UL and NRESET_DL can be used to reset the module at any time. Setting them to 0 will reset the module. NRESET_UL and NRESET_DL are resynchronized, hence they are fully taken in account when the clock is enabled.
- 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, then the WORKING bit of STATUS_REG is set and START bit is automatically reset.
- 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 data buffer
- UL_DL bit is used to select the part of the module that is going to be enabled. It can be considered that the GEA module is split into two internal modules, i.e. a module for ciphering+FCS and a module for deciphering_FCS checking. This bit is different than the D bit of the CONF_xx_REG1
- IT_EN bit allows disabling the ciphering interrupt. If disabled, the MCU has to make polling on the WORKING bit of the STATUS_REG and wait a return to 0. The interrupt can be masked by INTH too.

6.9 Status Register: STATUS_REG (Read) – FFFF:C002

Bit	Name	Function	Reset
0	WORKING	Module activity: 0 = not working 1 = working.	0
1	FCS_STATUS	FCS status (downlink only): 0 = FCS good 1 = FCS error	0
15:2	Unused		-

Notes:

- WORKING bit indicates that the module is doing processing. This bit is automatically enabled when the START bit of CNTL_REG has been set.
- FCS_STATUS bit is valid in **downlink only** and on the very last frame of the processed LLC_PDU only.

6.10 Interrupt Status Register: STATUS_irq_REG (Read) – FFFF:C004

Bit	Name	Function	Reset
0	LLC_IT	Processing status: 0 = On going. 1 = Finished. *(until read).* *When read, the interrupt is an acknowledge*	0
15:2	Unused		-

Notes:

- LLC_IT bit is used only when the IT_EN bit of CNTL_REG register has been enabled. When a MCU interrupt occurs, this register is checked by the MCU in order to know if the interrupt comes from the GEA module. It is automatically acknowledged when read.
- Interrupt is mapped in the MCU as defined in the top level specification

6.11 Uplink configuration registers: CONF_UL_REG(1:5)

6.11.1 Uplink registers 1: CONF_UL_REG1 (Read / Write) – FFFF:C006

Bit	Name	Function	Reset
0	IS	Input buffer shift 0 = No action 1 = byte 0 of the LLC frame is ignored	0
1	OS	Output buffer shift 0 = No action 1 = Output frame is shifted by one byte	0
2	reserved	-	-
3	E	Encryption: 0 = No encryption 1 = Encryption	0
4	PM	Protection mode: 0 = FCS computed on frame header + N202 bytes 1 = FCS computed on frame header + info field	0
5	F	FCS computation: 0 = No FCS computed 1 = FCS computed	0
6	D	Direction bit: See chapter 6.5.2 for direction bit values	0
7	AS	GEA algorithm selection: 0 = First GEA algorithm [4] 1 = Second GEA algorithm [5]	0
15:8	unused	-	-

<u>Notes</u>:

- IS bit: Input buffer Shift. When set to 1, byte 0 is ignored. First byte sent to ciphering/FCS computation is byte 1.
- 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.
- E bit: Encryption mode bit. When set to 1, the information and FCS fields of the frame shall be encrypted (ciphered). When set to 0, no encryption is applied on the frame.
- PM bit: Protection Mode bit. When set to 1, the FCS shall be calculated using both the frame header and information fields (LLC-PDU size). When set to 0, the FCS shall be calculated using only the frame header and the first N202 octets of the information field.
- F bit: FCS computation bit. When set to 1, the module has to compute the FCS. When set to 0, the FCS is included into the LLC-PDU data and the module must not compute it. PM bit and N202 field is ignored when F bit is set to 0.
- D bit: Direction bit. This bit is used as an input of the ciphering algorithm. Its value depends on which part of the network the user is located (MS or SGSN). It has not the same use than UL_DL bit of CNTL_REG.

6.11.2 Uplink registers 2: CONF_UL_REG2 (Read / Write) – FFFF:C008

Bit	Name	Function	Reset
15:0	PDU_SIZE	LLC-PDU size in bytes	0x00

Notes:

- LLC-PDU size is the number of bytes of the LLC-PDU contained into the buffer (IS bit has no influence on this size).
- 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).
- 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).

6.11.3 Uplink registers 3: CONF_UL_REG3 (Read / Write) – FFFF:C00A

Bit	Name	Function	Reset
7:0	N202	N202 variable size. *It is used only when PM bit is set to 1*	0x00
15:8	HEADER_SIZE	LLC-PDU header size	0x00

Notes:

- Header size is the number of bytes in the LLC frame header. This is used to know where the information field is starting for the ciphering or for unprotected mode in order to know, with N202, the size of the data to be protected. It is not mandatory to fill it if no ciphering and protected mode is used.
- N202 indicates the number of bytes of the information field that shall 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, then the FCS shall cover the complete information field. When the PM bit is set to 1, the module shall ignore this field. N202 is defined in [2].

6.11.4 Uplink registers 4: CONF_UL_REG4 (Read / Write) – FFFF:C00C

Bit	Name	Function	Reset
15:0	CIPH_IN_LSB	LSB part of the Ciph_in register (“message key”)	0x0000

6.11.5 Uplink registers 5: CONF_UL_REG5 (Read / Write) – FFFF:C00E

Bit	Name	Function	Reset
15:0	CIPH_IN_MSB	MSB part of the Ciph_in register (“message key”)	0x0000

6.11.6 CONF_UL_REG(4:5) bit mapping

	15															0
CONF_REG4	b15	…													…	b0
CONF_REG5	b31	…													…	b16

Notes:

- 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: Ciph_in = 0x12345678

CONF_UL_REG4 = 0x5678

and

CONF_UL_REG5 = 0x1234

6.12 Downlink Configuration Registers: CONF_DL_REG(1:5)

6.12.1 Downlink register 1: CONF_DL_REG1 (Read / Write) – FFFF:C010

Bit	Name	Function	Reset
0	IS	Input buffer shift: 0 = No action 1 = byte 0 of the LLC frame is ignored	0
1	OS	Output buffer shift: 0 = No action 1 = Output frame is shifted by one byte	0
2	reserved	-	-
3	E	Encryption: 0 = No encryption 1 = Encryption	0
4	PM	Protection mode: 0 = FCS computed on frame header + N202 bytes 1 = FCS computed on frame header + info field	0
5	F	FCS computation: 0 = No FCS computed 1 = FCS computed	0
6	D	Direction bit See chapter 6.5.2 for direction bit values	0
7	AS	GEA algorithm selection: 0 = First GEA algorithm [4] 1 = Second GEA algorithm [5]	0
15:8	unused	-	-

Notes: - Same as for uplink.

6.12.2 Downlink register 2: CONF_DL_REG2 (Read / Write) – FFFF:C012

Bit	Name	Function	Reset
15:0	PDU_SIZE	LLC-PDU size in bytes with FCS	0x0000

Notes:

- LLC-PDU size is the number of bytes of the LLC-PDU contained into the buffer (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 end of frame has been reached, all the internal variables must be reset in order to be ready to process a new frame.

  This value is not known in advance for downlink because it is only known on the last RLC/MAC block received. This is the reason why the MCU 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 than 0, then this is the last part of the LLC-PDU.

  PDU size must have been written before all the PDU bytes have been processed.

The GEA module never modifies this register. The MCU has to set/reset it with the right values.

6.12.3 Downlink register 3: CONF_DL_REG3 (Read / Write) – FFFF:C014

Bit	Name	Function	Reset
7:0	N202	N202 variable size. *It is used only when PM bit is set to 1*	0x00
15:8	HEADER_SIZE	LLC-PDU header size	0x00

Notes: Same as for uplink.

6.12.4 Downlink register 4: CONF_DL_REG4 (Read / Write) – FFFF:C016

Bit	Name	Function	Reset
15:0	CIPH_IN_LSB	LSB part of the Ciph_in register (“message key”)	0x0000

6.12.5 Downlink register 5: CONF_DL_REG5 (Read / Write) – FFFF:C018

Bit	Name	Function	Reset
15:0	CIPH_IN_MSB	MSB part of the Ciph_in register (“message key”)	0x0000

6.12.6 CONF_DL_REG(4:5) bit mapping:

	15															0
CONF_REG4	b15	…													…	b0
CONF_REG5	b31	…													…	b16

Notes: Same as for uplink.

6.13 Ciphering Key registers: KC_REG(1:4) (Read / Write) – FFFF:C01A

The 64-bits ciphering key Kc. It is split in four 16-bits registers with the following bit order:

	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 MCU and is taken into account by the module at the beginning of the 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 will be used only on the next LLC-PDU.

6.14 FCS Uplink registers: FCS_UL_REG(1:2) – FFFF:C022 / C024

This is the 24-bits FCS computed by the LLC module on the uplink LLC-PDU. It is split on two 16-bits registers. Highest order terms of the FCS are transmitted first, so they are located into the LSB of the register. These registers contain the plain or the ciphered FCS depending on the E bit of CONF_UL_REG1.

	15															0
FCS_UL_REG1	b8	…													…	b23
FCS_UL_REG2	0	0	0	0	0	0	0	0	b0	…					…	b7

6.15 FCS Downlink registers: FCS_DL_REG(1:2) – FFFF:C026 / C028

This is the 24-bits FCS computed by the LLC module on the downlink LLC-PDU. It is split on two 16-bits registers. Highest order terms of the FCS are received first, so the computed FCS is displayed in the same order. These registers always contained the plain FCS.

	15															0
FCS_DL_REG1	b8	…													…	b23
FCS_DL_REG2	0	0	0	0	0	0	0	0	b0	…					…	b7

Notes:

- The FCS must be computed on the LLC header and information field but not on the received FCS.

6.16 DATA registers

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 chapter 6.6 for details).

The size of the data given to the module are automatically determined by an internal counter that counts the number of write access to the register before START bit enabled.

These data are in 16-bits word format with the following bit order, with bit 0 processed first:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_REG	DATA_MSB								DATA_LSB							

6.16.1 DATA16_REG (Read / Write) – FFFF:C030

Bit	Name	Function	Reset
7:0	DATA_LSB	LSB part of the 16-bits word	U
15:8	DATA_MSB	MSB part of the 16-bits word	U

Note:

- In order to manage efficiently this interface, the MCU 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.

6.16.2 Frame Bit order

A LLC frame contains a header, information field and checksum (FCS) (see §7.5),.
In uplink, depending on the F bit, the FCS can be given or not to the module.
In downlink, this FCS must always be given to the module.

This frame is split into 16-bits words that are sent to the GEA module for processing.

The bits are processed in the order b0…b15b16…b32… with b0 the first bit of the header of the LLC frame (see schematic above). However, FCS (in downlink) is given with highest order terms first.

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

[*] bf are FCS bits

6.16.3 Frame splitting

The LLC frames have a dynamic size (two LLC frame do not have necessarily 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, i.e. 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 done as soon as new data are received, so the processing is done on a RLC/MAC frame basis (every 20ms).

Uplink and downlink LLC frames are multiplexed, i.e. GPRS can start ciphering a part of an uplink LLC frame then start to process a part of a received LLC-frame and so on.

Therefore, the GEA module keeps in memory all the static variables used for uplink or downlink.

Note:

- If ciphering is disabled but not FCS, the MCU has to write the frame into the buffer, but it is not mandatory to read them back (no modification has been done on these data).

6.16.4 DATA8_REG (Read / Write) – FFFF:C032

Bit	Name	Function	Reset
7:0	DATA	8 bits of data	U

Note:

- If an 8 bits write access is followed by 16 bits write access. The data will be continuous in the memory. Example : 8 bits write 0x11, followed by 16 bits write 0x3322. Data will be stored as follows in the memory :

MSB Byte	LSB byte
22	11
	33

6.17 GEA Programming scheduling

The module has to be programmed in the following order by the MCU:

1. Set CLOCK bit from CNTL_REG to 1. Then wait return to 0
   ⇒ Clock is ON.
2. Set RESET_UL and RESET_DL bits from CNTL_REG to 0. Then wait return to 1
   ⇒ Uplink and downlink GEA are reseted.

<u>In uplink</u>

3. Fill the configuration registers:
   CONF_UL_REG1-3
4. Fill the ciphering registers:
   KC_REG1-4 (not mandatory to set it every frame. can be done once at the beginning)
   CONF_UL_REG4-5
5. Fill DATA_REG n times with the data to be processed (if 6 words, n =6)
6. Set in one shot, START bit to 1, UL_DL to 0 and IT_EN to 1 of CNTL_REG.
7. Wait for interrupt [3]
8. When interrupt occurs, read LLC_IT bit from STATUS_irq_REG in order to check that the GEA module has generated the IT.
9. If yes, read n times the processed data from DATA_REG
10. If this is the last part of the LLC-PDU, then read LLC_UL_REG1-2 registers that contain the FCS result.

    Clock can be disabled here or if another frame must be processed, the module is ready to restart from 3/
11. If this is not the last part of the LLC-PDU, go to 5/

<u>In downlink</u>

3. Fill the configuration registers:
   CONF_DL_REG1&3
4. Fill the ciphering registers:
   KC_REG1-4 (not mandatory to set it every frame. can be done once at the beginning)
   CONF_DL_REG4-5
5. If this last part of the LLC_PDU , CONF_DL_REG2 must have been filled with the downlink PDU size.
6. Fill DATA_REG n times with the data to be processed (if 6 words, n =6)
7. Set in one shot, START bit to 1, UL_DL to 0 and IT_EN to 1 of CNTL_REG.
8. Wait for interrupt[4]
9. When interrupt occurs, read LLC_IT bit from STATUS_irq_REG in order to check that the GEA module has generated the IT.
10. If yes, read n times the processed data from DATA_REG
11. If this is the last part of the LLC-PDU, then read LLC_UL_REG1-2 registers that contain the FCS result.

    Clock can be disabled here or if another frame must be processed, the module is ready to restart from 3/
12. If this is not the last part of the LLC-PDU, go to 5/.

---

[3] If IT_EN set to 0, no interrupt occurs. The MCU has to make polling on the WORKING bit of STATUS_REG. This bit is automatically set to 0 when the module has finished its processing.

[4] If IT_EN set to 0, no interrupt occurs. The MCU has to make polling on the WORKING bit of STATUS_REG. This bit is automatically set to 0 when the module has finished its processing.

7. CONFIGURATION REGISTERS – FFFE:F000

7.1 Configuration register mapping

register	address	access
DEVICE ID code	FFFE:F000	16 bits R
DEVICE version code	FFFE:F002	4 bits R
ARM version code	FFFF:FE00	4 bits R
cDSP ID code	FFFF:FE02	16 bits R
DSP configuration	FFFE:F004	11 bits R/W
Extended MCU configuration	FFFE:F006	8 bits R/W
Asic configuration	FFFE:F008	16 bits R/W
I/O selection	FFFE:F00A	12 bits R/W
-	FFFE:F00C	-
MCU software trace	FFFE:F00E	6 bits R/W
Device ID code	FFFE:F010	64 bits R

Table 12: Configuration registers

7.2 Device ID code (Read only) – FFFE:F000

Bit	Name	Function	Reset C05 (version A)	Reset C05 (version B)	Reset C035
15:0	PART	Device ID code part	B2AC	B396	B496

7.3 Device version code (Read only) – FFFE:F002

Bit	Name	Function	Reset (version A)	Reset (version B)	Reset C035
3:0	VERSION	Device code version	0000	0002	0000
15:4	*Unused*	-	?		

7.4 cDSP ID code (Read only) – FFFF:FE02

Bit	Name	Function	Reset (version A)	Reset (version B)
15:0	CDSP_ID	CDSP ID code (F7...)	B396	0128

7.5 ARM ID code (Read only) – FFFF:FE00

Bit	Name	Function	Reset (version A)	Reset (version B)
3:0	ARM_ID	ARM_ID code	0003	0003
15:4	*Unused*	-	?	

Note*: In this register, only the first four bits are used, the other ones are always cleared to 0.*

7.6 Die Identification code (Read only) – FFFE:F010 .. F016

The Die ID is a 64 bits fuse based register.

Bit	Name	Description	Reset

15:0	Die-ID-LSB	***UNIQUE BY DIE***	-
31:16	Die-ID-MID		-
46:32	Die-ID-MSB		-
63:47	Reserved		-

Note: *The die ID value is all ‘1s’ when not laser fuse blown.*

7.7 DSP configuration (Read / Write) – FFFE:F004

DSP_CONF_REG

Bit		Reset
0	0 = keyboard interface (KBR(4..0), KBC(4..2)) 1 = XDI_O(7..0) DSP data bus	0
1	0 = functional mode (TXIR_IRDA, nFOE, nSCS1, RXIR_IRDA, nFWE) 1 = DSP address bus (DSP_A(4..0))	0
2	0 = enable LMM power 1 = DSP strobe signal (DSP_IOSTRBN)	0
3	0 = RTS_MODEM 1 = timer output (TOUT)	0
4	0 = CTS_MODEM 1 = external flag (XF)	0
5	0 = uwire data out (SDO) 1 = DMA interrupt (INT10n)	0
6	0 = uwire serial clock (SCLK) 1 = RIF transmit interrupt (INT1n)	0
7	0 : uwire data in (SDI) 1 : TPU frame interrupt (INT8n)	0
8	0 = Chip select 3 (nCS3) 1 = MCSI Txint (DSP-INT4n)	0
9	0 = flash deep low power (FDP) 1 = Interrupt acknowledge (nIACK)	0
10	0 = IRDA transceiver shut down mode (SD_IRDA) 1 = DSP clock out (CLKOUT_DSP)	0
15:11	*unused* Fixed to 0	0

***Note**: When configured in DSP mode, signal CTS_MODEM will be internally forced to high level in order from preventing any unwanted signal level change detection in UART module when externally observing XF.*

7.8 Extended MCU configuration (Read / Write) – FFFE:F006

ARM_CONF_REG

Bit	Description	Reset
0	*unused*	0
1	0 = Keyboard interface (KBC(0)) 1 = ARM fast interrupt (NFIQ)	0
2	0 = keyboard interface (KBC(1)) 1 = ARM normal interrupt (NIRQ)	0
3	0 = CS4 1 = ADD(22)	0
4	0 = Serial clock (BCLKR) 1 = ARMCLK	0
5	0 = TSPACT6 1 = Internal memory chip select nCS6	0
6	0 = I/O(2) 1 = IRQ4	0
7	0 = TSPACT4 1 = nRDYMEM ready signal for external slow device.	0
15:8	*unused*	0

***Note**: When configured in ARM extended mode, in order to maintain the RIF receive path, signal BCLKR must be internally generated from the RIF transmit clock with selecting either the digital loop-back mode or the clock loop mode in the RIF register SPCR_REG.*

7.9 Asic configuration (Read / Write) – FFFE:F008

ASIC_CONF_REG

Bit	Description	Reset
0	0 = I/O(0) 1 = TPU wait mode (TPU_WAIT)	0
1	0 = I/O(1) 1 = TPU idle mode (TPU_IDLE)	0
2	0 = reset of external peripheral (CLK13M_OUT) 1 = start bit detection (START_BIT)	0
3	*Unused* *must be fixed to 0.*	0
4	light output 0 = LT 1 = PWL	0
5	buzzer output 0 = BU 1 = PWT	0
6	0 = DSR_MODEM 1 = LPG	0
7	0 = nSCS0 1 = clock of I2C (SCL)	0
8	0 = ADD(21) 1 = clock of uart IRDA (CLK_16X_IRDA)	0
9	0 = TSPACT7 1 = clock of spi (CLKX_SPI)	0
10	0 = I/O(3) 1 = Sim RnW signal (SIM_RnW)	0
11	0 = TSPEN3 1 = uwire chip select (nSCS2)	0
12	0 = uwire data in (SDI) 1 = data of I2C (SDA)	0
13	0 = RIF config clock RX (falling edge) 1 = RIF config clock RX (rising edge)	0
14	0 = SPI config clock RX (falling edge) 1 = SPI config clock RX (rising edge)	0
15	0 = TSPACT5 1 = DPLL clock output.	0

Important : bit 3 definition is only available from Calypso C035 – F751619

3	0 : Normal mode 1 : DmaDbg mode : ASIC_CONF_REG bits

5	4	3	Pin Function's
0	0	0	(Reset) LT & BU
0	0	1	LT & BU
0	1	0	PWL & BU
0	1	1	nEndDma & BU
1	0	0	LT & PWT
1	0	1	LT & nDmaReq
1	1	0	PWL & PWT
1	1	1	nEndDma & nDmaReq

7.10 IO selection (Read / Write) – FFFE:F00A

IO_CONF_REG

Bit	Description	Reset
0	0 = TSPDI 1 = I/O(4)	0
1	0 = SIM_PWCTRL 1 = I/O(5)	0
2	0 = BCLKX 1 = I/O(6)	0
3	0 = nRESET_OUT 1 = I/O(7)	0
4	0 = MCUEN(1) 1 = I/O(8)	0
5	0 = MCSI_TXD 1 = I/O(9)	0
6	0 = MCSI_RXD 1 = I/O(10)	0
7	0 = MCSI_CLK 1 = I/O(11)	0
8	0 = MCSI_FSYNCH 1 = I/O(12)	0
9	0 = MCUEN(2) 1 = I/O(13)	0
10	0 = nBHE 1 = I/O(14)	0
11	0 = nBLE 1 = I/O(15)	0
15:12	*Unused*	0

7.11 MCU software trace (Read / Write) – FFFE:F00E

MCU_SW_TRACE

Bit	Description	Reset
0	0 = SIM_CD 1 = MAS(0)	0
1	0 = TSPACT9 1 = MAS(1)	0
2	0 = TSPACT11 1 = MCLK	0
3	0 = TSPACT10 1 = nWAIT	0
4	0 = RFEN 1 = nOPC	0
5	0 = TSPACT8 1 = nMREQ	0
15:6	*unused*	0

8. DSP XIO to RHEA - XIO:F800

There are two 16-bit control registers: a transfer_rate register and a bridge_cntl register. Both registers are mapped directly into DSP I/O space. The transfer_rate register is mapped at I/O space address 0xF800, and the bridge_cntl register is mapped at I/O space address 0xF801.

8.1 XIO-Rhea register mapping

Register	Address	Access	HW Reset value
TRANSFER_RATE_REG	XIO:F800	16bits R/W	0110 0110 0110 0110
BRIDGE-CTRL_REG	XIO:F801	10bits R/W	???? ?001 0111 1111

Table 13: DSP Rhea registers

8.2 Transfer_rate (Read / Write) - XIO:F800

Programmer must define two registers to allow correct DSP accesses to external peripheral connected on the RHEA bus:

1. SWWSR internal register of the DSP to set I/O waits states.
2. TRANSFER_RATE_REG XIO register to fit DSP access to RHEA bus.

Note that:

- Sequence of programmatic must be SWWSR first then, TRANSFER_RATE_REG.

- When the number of wait states in the SWWSR register is equal to 0 or 1, then the DSP peripherals 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 RHEA bus. The following relations must be always satisfied:

$$Strobe\_Length \geq \frac{1}{2} * RHEA\_CLK\_Period$$

$$Strobe\_Length = \frac{1}{2} LEAD\_CLKout * (TRANS\_RATE + 1)$$

$$TRANS\_RATE \geq Number\_of\_WS\_in\_SWWSR$$

All bits of this register can be written to and read from by DSP.

Bit	Name	Function	Reset
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 duration of half a DSP clock cycle - down to slowest speed (5) - duration of 8 DSP clock cycles .	0F
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 duration of half a DSP clock cycle - down to slowest speed (5) - duration of 8 DSP clock cycles .	0F
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 duration of half a DSP clock cycle - down to slowest speed (5) - duration of 8 DSP clock cycles .	0F
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 duration of half a DSP clock cycle - down to slowest speed (5) - duration of 8 DSP clock cycles .	0F

8.3 Bridge_cntl (Read / Write) - XIO:F801

All bits of this register can be read and written by the DSP.

Bit	Name	Function	Reset
7:0	TIMEOUT	RHEA buses access time out. Limits, by counting nXSTROBE cycles, the maximum time a peripheral can stall the processor When starting a transaction on the RHEA bus, the time out counter is at a count of zero. If the current cycle is not finished when the counter reaches a count of TIMEOUT + 1, the transaction is aborted (by sending nXABORT to the peripheral and a NMI interrupt if TIMEOUT_ENABLE is set).	FF
8	TIMEOUT_ENABLE	When set to logic '1', enable the RHEA bus TIMEOUT watchdog to send a NMI interrupt when the maximal RHEA access time is reached.	0
9	NSUPV	RHEA buses supervisor flag. This flag is used to control the access to privileged peripheral registers. If NSUPV is set to 0, Access can be done on privileged registers. If NSUPV is set to 1, access is forbidden.	0
15:10	-	*Reserved*	-

9. API REGISTERS - XIO:F900 – FFE0:0000

In order to optimize the API access from the host side, DSP software should provide to the ARM7-to-RHEA bridge (in charge of managing access to API memory) the following information thanks to register API_CONF.

9.1 DSP API configuration register mapping

Register	Address	Access	HW Reset value
API_CONF	XIO:F900	3bits R/W	???? ???? ???? ?010

Table 14: DSP API configuration register

Bit	Name	Function	Value at HW reset
0	-	***Reserved. Must be '0'***	0
1	API_HOM	0 = API is configured in SAM mode 1 = API is configured in HOM mode	1
2	BRIDGE_CLK_EN	0 = ARM-RHEA bridge and DMA controller clock runs according to the ARM sleep mode and the DMA channels activity. 1 = Force ARM-RHEA bridge and DMA controller clock to run whatever the ARM sleep mode is.	0

The DSP software during the ARM sleep mode must use this command when a DMA channel is controller by the DSP. Indeed, The control and status register of each DMA controller channel (DMA1_CTRL and DMA2_CTRL) is only writable and readable when the DMA controller clock runs. It means that each time the DSP software want to access to these registers, it must set the BRIDGE_CLK_EN bit to '1', then access to these registers and then set back BRIDGE_CLK_EN bit to '0' in order to conserve power.

9.2 APIC CONTROL REGISTER

9.2.1 MCU reads from APIC.

Register	Address	Access	HW Reset value
APIC	FFE0:0000	2bits R	0000 0000 0000 0010

Table 15: APIC control register (MCU reads)

The APIC contains the status and control bits for the API. The APIC is not a memory-mapped register, but accessed by the DSP through the bank-switching control register (BSCR). The three bits of the APIC are:

- ❒ SMODE – enables the SAM or HOM (sent from DSP).
- ❒ HINT – interrupt sent by the DSP to MCU.
- ❒ DSPINT – interrupt sent by MCU to DSP.

The MCU accesses the APIC 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 MCU access before being sent on the data bus. All other bits are tied to 0.

APIC control register (MCU reads)

Bit	Name	Function	MCU	DSP	Reset
0	-	***This bit is read as 0.***	R	R	0
1	SMODE	Shared access mode (SAM): 0 = API is configured in SAM mode 1 = API 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	1
2	-	***This bit is read as 0.***	R	R	0
3	HINT	Host processor interrupt bit: 0 = No effect. 1 = The MCU acknowledges and clears the interrupt. *This bit enables/disables an interrupt written by the DSP to the MCU. 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 MCU receives an interrupt pulse. Duration of pulse is controlled by DSP software.*	R	W	0
15:4	-	***These bits are read as 0.***	R	R	0

9.2.2 MCU writes to APIC.

The DSPINT is a write only bit. Only the MCU can write to this bit. The MCU writes using the address FFE0:000 mapped in the memory interface. The other bits are not modified by the MCU.

Register	Address	Access	HW Reset value
APIC	FFE0:0000	1bits W	???? ???? ???? ?0??

Table 16: APIC control register (MCU writes)

Bit	Name	Function	MCU	DSP	Reset
1:0	-	***These bits are not modified***	R	R	x
2	DSPINT	DSP interrupt bit: 0 = No interrupt is generated. 1 = The MCU writes a 1 to generate a DSP interrupt (INT9). *This bit enables/disables an interrupt from the MCU to the DSP. The DSPINT bit is written by the MCU; an DSP write has no effect on the DSPINT bit, the write operation is locally latched and re-synchronized on DSP clocks. DSPINT is automatically reset in API logic.* *This interrupt can be used to wake up the DSP from an IDLE state or to generate an interrupt.*	W	R	0
15:3	-	***These bits are not modified***	R	R	x

9.2.3 DSP accesses from/to APIC.

The DSP cannot directly read from or write to the APIC, since the APIC 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 APIC. The comparison between BSCR and APIC are:

BSCR

15 – 12	11	10 – 4	3	2	1	0
BNKCMP	PS-DS	Reserved	HINT	SMODE	APIBN	EXIO

APIC

15 – 4	3	2	1	0
	HINT	DSPINT	SMODE	

10. DMA MAPPING

The DMA controller is managing the access to the DSP API 6K-word shared memory.

1. The MCU ARM7.
2. The Radio InterFace (RIF).
3. The MODEM UART.
4. The IRDA UART.

The RIF-RX and RIF-TX have a dedicated channel each. For UART the following combination are possible:

- ⇨ UART-MODEM have 2 channels:
  - ch#2: TX
  - ch#3:RX
- ⇨ UART-IRDA have 2 channels:
  - ch#2: RX
  - ch#3: TX
- ⇨ UART-MODEM and UART-IRDA have one channel each
  - ch#2: UART-MODEM
  - ch#3: UART-IRDA

  RX and/or TX are independently selectable for each module.
- ⇨ UART-MODEM have one channel
  - ch#2: RX or TX
- ⇨ UART-IRDA have one channel
  - ch#3: RX or TX
- ⇨ UARTs have no DMA.

After reset, all modules have DMA functions disabled.

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				✓
nDMA_REQ_ARM(0) IRDA				✓
nDMA_REQ_ARM(1) IRDA			✓	

Table 17: DMA channels allocation

<u>Note</u>: 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.

11. DMA CONTROLLER - FFFF:FC00 - XIO:FC00

The DMA registers are connected to a Rhea bus. The DMA controller configuration registers is connected to the ARM Rhea bus, the DMA transfer configuration registers can be accessed by the ARM o the DSP Rhea bus regarding the DMA_ALLOC register value.

11.1 DMA register mapping.

Register	MCU Address	DSP Address	Access	Reset value
CONTROLLER_CONF	FFFF:FC00	XIO:FC00	6bits R/W	11 111?
ALLOC_CONFIG	FFFF:FC02	XIO:FC02	4bits R/W	1111
DMA1_RAD	FFFF:FC10	XIO:FC10	16bits R/W	0000 0000 0000 0000
DMA1_RDPTH	FFFF:FC12	XIO:FC12	11bits R/W	000 0000 0000
DMA1_AAD	FFFF:FC14	XIO:FC14	12bits R/W	0000 0000 0000
DMA1_ALGTH	FFFF:FC16	XIO:FC16	12bits R/W	0000 0000 0000
DMA1_CTRL	FFFF:FC18	XIO:FC18	13	0 0100 1?10 0010
DMA1_CUR_OFFSET_API	FFFF:FC1A	XIO:FC1A	12bits R	0000 0000 0000
DMA2_RAD	FFFF:FC20	XIO:FC20	16bits R/W	0000 0000 0000 0000
DMA2_RDPTH	FFFF:FC22	XIO:FC22	11bits R/W	000 0000 0000
DMA2_AAD	FFFF:FC24	XIO:FC24	12bits R/W	0000 0000 0000
DMA2_ALGTH	FFFF:FC26	XIO:FC26	12bits R/W	0000 0000 0000
DMA2_CTRL	FFFF:FC28	XIO:FC28	13bits R/W	0 0100 1?10 0010
DMA2_CUR_OFFSET_API	FFFF:FC2A	XIO:FC2A	12bits R	0000 0000 0000
DMA3_RAD	FFFF:FC30	XIO:FC30	16bits R/W	0000 0000 0000 0000
DMA3_RDPTH	FFFF:FC32	XIO:FC32	11bits R/W	000 0000 0000
DMA3_AAD	FFFF:FC34	XIO:FC34	12bits R/W	0000 0000 0000
DMA3_ALGTH	FFFF:FC36	XIO:FC36	12bits R/W	0000 0000 0000
DMA3_CTRL	FFFF:FC38	XIO:FC38	31bits R/W	0 0100 1?10 0010
DMA3_CUR_OFFSET_API	FFFF:FC3A	XIO:FC3A	12bits R	0000 0000 0000
DMA4_RAD	FFFF:FC40	XIO:FC40	16bits R/W	0000 0000 0000 0000
DMA4_RDPTH	FFFF:FC42	XIO:FC42	11bits R/W	000 0000 0000
DMA4_AAD	FFFF:FC44	XIO:FC44	12bits R/W	0000 0000 0000
DMA4_ALGTH	FFFF:FC46	XIO:FC46	12bits R/W	0000 0000 0000
DMA4_CTRL	FFFF:FC48	XIO:FC48	13bits R/W	0 0100 1?10 0010
DMA4_CUR_OFFSET_API	FFFF:FC4A	XIO:FC4A	12bits R	0000 0000 0000

Table 18: DMA register mapping

11.2 MCU registers

The DMA registers are connected to a Rhea bus. The DMA controller configuration registers is connected to the ARM Rhea bus, the DMA transfer configuration registers can be accessed by the ARM o the DSP Rhea bus regarding the DMA_ALLOC register value.

11.2.1 CONTROLLER_CONFIG* (Read / Write) – FFFF:FC00 – XIO:FC00

This register can only be accessed if the MCU is in SUPERVISOR mode. The bits can be read and written by the ARM.

Bit	Name	Function	Reset
0:1	Reserved	-	-
4-2	DMA_BURST	Defines the length of the DMA transfer burst. The ARM can not access the RHEA bus during the DMA burst.	0x1
5	PRIORITY_ENABLE	0 = The Rhea bus allocation is done using the DMA_BURST factor 1 = ARM has the same priority than the DMA transfers regarding Rhea bus allocation when it is in Exception mode (IRQ & FIQ).	1
15-6	Reserved	-	-

* Accessible only from the ARM (not the DSP)

11.2.2 ALLOC_CONFIG* (Read / Write) – FFFF:FC02 – XIO:FC02

This register can only be accessed if the MCU is in SUPERVISOR mode. The bits can be read and written by the ARM.

Bit	Name	Function	Reset
0	DMA_ALLOC_1	DMA channel 1 control: 0 = DMA channel 1 is controlled by DSP. 1 = DMA channel 1 is controlled by ARM	1
1	DMA_ALLOC_2	DMA channel 2 control: 0 = DMA channel 2 is controlled by DSP. 1 = DMA channel 2 is controlled by ARM	1
2	DMA_ALLOC_3	DMA channel 3 control: 0 = DMA channel 3 is controlled by DSP. 1 = DMA channel 3 is controlled by ARM	1
3	DMA_ALLOC_4	DMA channel 4 control: 0 = DMA channel 4 is controlled by DSP. 1 = DMA channel 4 is controlled by ARM	1
15-4	Unused	-	-

Accessible only from the ARM (not the DSP)

11.3 DMA Channel 1 configuration registers – FFFF:FC10

The DMA transfer configuration registers are connected to either the ARM Rhea bus or to the DSP Rhea bus regarding the corresponding DMA_ALLOC flag.
These registers define the channel 1 transfer parameters.

11.3.1 DMA1_RAD (Read / Write) – FFFF:FC10 – XIO:FC10

DMA1 Rhea Address

Bit	Name	Function	Reset
10:0	RHEA_START	Rhea buffer start address	00
15:11	RHEA_CS	Peripheral chip select	00

11.3.2 DMA1_RDPTH (Read / Write) – FFFF:FC12 – XIO:FC12

DMA1 Rhea depth

Bit	Name	Function	Reset
10:0	RHEA_DEPTH	Rhea buffer depth in BYTE.	00
15:11	*Unused*		

11.3.3 DMA1_AAD (Read / Write) – FFFF:FC14 – XIO:FC14

DMA1 Api Address

Bit	Name	Function	Reset
11:0	API_START	Start address of the reception buffer in the API memory. *The address is always expressed in Bytes*	00
15:12	*Unused*		-

11.3.4 DMA1_ALGTH (Read / Write) – FFFF:FC16 – XIO:FC16

DMA1 Api length

Bit	Name	Function	Reset
11:0	API_LENGTH	API page length in Bytes.	00
15:12	*Unused*		-

11.3.5 DMA1_CTRL* (Read / Write) – FFFF:FC18 – XIO:FC18

DMA1 control

Bit	Name	Function	Reset	Access
0	ENABLE	0 = DMA channel 1 is disabled 1 = DMA channel 1 is enabled ***When channel is DISABLE hard or soft requests are ignored***.	0	R/W
1	IDLE	0 = DMA transfer is running 1 = DMA channel is idle	1	R
2	ONE_SHOT	0 = *No effect* 1 = ENABLE flag is cleared at the end of the API page transfer.	0	R/W
3	FIFO_MODE	1= RHEA_START address is used for all the Rhea access and the peripheral can stop the transfer only by nEND_DMA signal. ***RHEA_DEPTH isn't used in this mode.***	0	R/W
4	CURRENT_PAGE	0 = API access are done in the first API page 1 = API access are done in the 2nd API page ***This bit is automatic updated during transfer***.	0	R/W
5	MAS	0 = DMA transactions are done on 8-bit API and Rhea data 1 = DMA transactions are done on 16-bit API and Rhea data	1	R/W
6	DMA_START	0 = *No Effect* 1 = (Write) initiates the DMA transfer. ***Reading of this bit is always equal to zero.***	-	R/W
7	IRQ_MODE	0 = Interrupt is requested at the end of Rhea buffer transfer or if there is an nEND_DMA 1 = an interrupt has to be launched at the end of API page transfer,	1	R/W
8	IRQ_STATE	0 = cleared after being read 1 = an IRQ requested by DMA channel 1	0	R
9	RHEA_ERROR	0 = cleared after being read 1 = DMA RHEA access is aborted, an IRQ is also generated.	0	R
10	DIRECTION	1 = Transactions are done on Rhea -> API 0 = Transactions are done on API -> Rhea	1	R/W
12:11	PRIORITY	number of additional reading on the bus: 00 = zero read more, one reading 01 = one read more, 2 successive reading 10 = two read more, 3 successive reading 11 = three read more, 4 successive reading	00	R/W
15:13	-	*Reserved*		

*DMA1_CTRL is only writable and readable when the DMA controller clock runs. It means that each time the DSP software want to access to these registers, if ARM is in sleep mode, it must set the BRIDGE_CLK_EN bit of DSP API configuration register to '1',

11.3.6 DMA1_CUR_OFFSET_API (Read) – FFFF:FC1A – XIO:FC1A

DMA1 Current OFFSET API

Bit	Name	Function	Reset
11:0	API_OFFSET	This register indicated offset API of the next reading or writing API memory.	00
15:12	*Unused*		-

You must read this register only if the DMA controller isn't running.

11.4 DMA Channel 2 configuration registers – FFFF:FC20

The DMA transfer configuration registers are connected to either the ARM Rhea bus or to the DSP Rhea bus regarding the corresponding DMA_ALLOC flag.
These registers define the channel 1 transfer parameters.

11.4.1 DMA2_RAD (Read / Write) – FFFF:FC20 – XIO:FC20

DMA2 Rhea Address

Bit	Name	Function	Reset
10:0	RHEA_START	Rhea buffer start address	00
15:11	RHEA_CS	Peripheral chip select	00

11.4.2 DMA2_RDPTH (Read / Write) – FFFF:FC22 – XIO:FC22

DMA2 Rhea depth

Bit	Name	Function	Reset
10:0	RHEA_DEPTH	Rhea buffer depth in BYTE.	00
15:11	*Unused*		

11.4.3 DMA2_AAD (Read / Write) – FFFF:FC24 – XIO:FC24

DMA2 Api Address

Bit	Name	Function	Reset
11:0	API_START	Start address of the reception buffer in the API memory. *The address is always expressed in Bytes*	00
15:12	*Unused*		-

11.4.4 DMA2_ALGTH (Read / Write) – FFFF:FC26 – XIO:FC26

DMA2 Api length

Bit	Name	Function	Reset
11:0	API_LENGTH	API page length in Bytes.	00
15:12	*Unused*		-

11.4.5 DMA2_CTRL* (Read / Write) – FFFF:FC28 – XIO:FC28

DMA1 control

Bit	Name	Function	Reset	Access
0	ENABLE	0 = DMA channel 2 is disabled 1 = DMA channel 2 is enabled ***When channel is DISABLE hard or soft requests are ignored*.**	0	R/W
1	IDLE	0 = DMA transfer is running 1 = DMA channel is idle	1	R
2	ONE_SHOT	0 = *No effect* 1 = ENABLE flag is cleared at the end of the API page transfer.	0	R/W
3	FIFO_MODE	1= RHEA_START address is used for all the Rhea access and the peripheral can stop the transfer only by nEND_DMA signal. ***RHEA_DEPTH isn't used in this mode.***	0	R/W
4	CURRENT_PAGE	0 = API access are done in the first API page 1 = API access are done in the 2nd API page ***This bit is automatic updated during transfer*.**	0	R/W
5	MAS	0 = DMA transactions are done on 8-bit API and Rhea data 1 = DMA transactions are done on 16-bit API and Rhea data	1	R/W
6	DMA_START	0 = *No Effect* 1 = (Write) initiates the DMA transfer. ***Reading of this bit is always equal to zero.***	-	R/W
7	IRQ_MODE	0 = Interrupt is requested at the end of Rhea buffer transfer or if there is an nEND_DMA 1 = an interrupt has to be launched at the end of API page transfer,	1	R/W
8	IRQ_STATE	0 = cleared after being read 1 = an IRQ requested by DMA channel 2	0	R
9	RHEA_ERROR	0 = cleared after being read 1 = DMA RHEA access is aborted, an IRQ is also generated.	0	R
10	DIRECTION	1 = Transactions are done on Rhea -> API 0 = Transactions are done on API -> Rhea	1	R/W
12:11	PRIORITY	number of additional reading on the bus: 00 = zero read more, one reading 01 = one read more, 2 successive reading 10 = two read more, 3 successive reading 11 = three read more, 4 successive reading	00	R/W
15:13	-	*Reserved*		

*DMA2_CTRL is only writable and readable when the DMA controller clock runs. It means that each time the DSP software want to access to these registers, if ARM is in sleep mode, it must set the BRIDGE_CLK_EN bit of DSP API configuration register to '1',

11.4.6 DMA2_CUR_OFFSET_API (Read) – FFFF:FC2A – XIO:FC2A

DMA2 Current OFFSET API

Bit	Name	Function	Reset
11:0	API_OFFSET	This register indicated offset API of the next reading or writing API memory.	00
15:12	*Unused*		-

You must read this register only if the DMA controller isn't running.

11.5 DMA Channel 3 configuration registers – FFFF:FC30

The DMA transfer configuration registers are connected to either the ARM Rhea bus or to the DSP Rhea bus regarding the corresponding DMA_ALLOC flag.
These registers define the channel 1 transfer parameters.

11.5.1 DMA3_RAD (Read / Write) – FFFF:FC30 – XIO:FC30

DMA3 Rhea Address

Bit	Name	Function	Reset
10:0	RHEA_START	Rhea buffer start address	00
15:11	RHEA_CS	Peripheral chip select	00

11.5.2 DMA3_RDPTH (Read / Write) – FFFF:FC32 – XIO:FC32

DMA3 Rhea depth

Bit	Name	Function	Reset
10:0	RHEA_DEPTH	Rhea buffer depth in BYTE.	00
15:11	*Unused*		

11.5.3 DMA3_AAD (Read / Write) – FFFF:FC34 – XIO:FC34

DMA3 Api Address

Bit	Name	Function	Reset
11:0	API_START	Start address of the reception buffer in the API memory. *The address is always expressed in Bytes*	00
15:12	*Unused*		-

11.5.4 DMA3_ALGTH (Read / Write) – FFFF:FC36 – XIO:FC36

DMA3 Api length

Bit	Name	Function	Reset
11:0	API_LENGTH	API page length in Bytes.	00
15:12	*Unused*		-

11.5.5 DMA3_CTRL (Read / Write) – FFFF:FC38 – XIO:FC38

DMA3 control

Bit	Name	Function	Reset	Access
0	ENABLE	0 = DMA channel 3 is disabled 1 = DMA channel 3 is enabled ***When channel is DISABLE hard or soft requests are ignored*.**	0	R/W
1	IDLE	0 = DMA transfer is running 1 = DMA channel is idle	1	R
2	ONE_SHOT	0 = *No effect* 1 = ENABLE flag is cleared at the end of the API page transfer.	0	R/W
3	FIFO_MODE	1= RHEA_START address is used for all the Rhea access and the peripheral can stop the transfer only by nEND_DMA signal. ***RHEA_DEPTH isn't used in this mode.***	0	R/W
4	CURRENT_PAGE	0 = API access are done in the first API page 1 = API access are done in the 2nd API page ***This bit is automatic updated during transfer*.**	0	R/W
5	MAS	0 = DMA transactions are done on 8-bit API and Rhea data 1 = DMA transactions are done on 16-bit API and Rhea data	1	R/W
6	DMA_START	0 = *No Effect* 1 = (Write) initiates the DMA transfer. ***Reading of this bit is always equal to zero.***	-	R/W
7	IRQ_MODE	0 = Interrupt is requested at the end of Rhea buffer transfer or if there is an nEND_DMA 1 = an interrupt has to be launched at the end of API page transfer,	1	R/W
8	IRQ_STATE	0 = cleared after being read 1 = an IRQ requested by DMA channel 3	0	R
9	RHEA_ERROR	0 = cleared after being read 1 = DMA RHEA access is aborted, an IRQ is also generated.	0	R
10	DIRECTION	1 = Transactions are done on Rhea -> API 0 = Transactions are done on API -> Rhea	1	R/W
12:11	PRIORITY	number of additional reading on the bus: 00 = zero read more, one reading 01 = one read more, 2 successive reading 10 = two read more, 3 successive reading 11 = three read more, 4 successive reading	00	R/W
15:13	-	*Reserved*		

11.5.6 DMA3_CUR_OFFSET_API (Read) – FFFF:FC3A – XIO:FC3A

DMA1 Current OFFSET API

Bit	Name	Function	Reset
11:0	API_OFFSET	This register indicated offset API of the next reading or writing API memory.	00
15:12	*Unused*		-

You must read this register only if the DMA controller isn't running.

11.6 DMA Channel 4 configuration registers – FFFF:FC40

The DMA transfer configuration registers are connected to either the ARM Rhea bus or to the DSP Rhea bus regarding the corresponding DMA_ALLOC flag.
These registers define the channel 1 transfer parameters.

11.6.1 DMA4_RAD (Read / Write) – FFFF:FC40 – XIO:FC40

DMA4 Rhea Address

Bit	Name	Function	Reset
10:0	RHEA_START	Rhea buffer start address	00
15:11	RHEA_CS	Peripheral chip select	00

11.6.2 DMA4_RDPTH (Read / Write) – FFFF:FC42 – XIO:FC42

DMA4 Rhea depth

Bit	Name	Function	Reset
10:0	RHEA_DEPTH	Rhea buffer depth in BYTE.	00
15:11	*Unused*		

11.6.3 DMA4_AAD (Read / Write) – FFFF:FC44 – XIO:FC44

DMA4 Api address

Bit	Name	Function	Reset
11:0	API_START	Start address of the reception buffer in the API memory. *The address is always expressed in Bytes*	00
15:12	*Unused*		-

11.6.4 DMA4_ALGTH (Read / Write) – FFFF:FC46 – XIO:FC46

DMA4 Api length

Bit	Name	Function	Reset
11:0	API_LENGTH	API page length in Bytes.	00
15:12	*Unused*		-

11.6.5 DMA4_CTRL (Read / Write) – FFFF:FC48 – XIO:FC48

DMA4 control

Bit	Name	Function	Reset	Access
0	ENABLE	0 = DMA channel 4 is disabled 1 = DMA channel 4 is enabled ***When channel is DISABLE hard or soft requests are ignored***.	0	R/W
1	IDLE	0 = DMA transfer is running 1 = DMA channel is idle	1	R
2	ONE_SHOT	0 = *No effect* 1 = ENABLE flag is cleared at the end of the API page transfer.	0	R/W
3	FIFO_MODE	1= RHEA_START address is used for all the Rhea access and the peripheral can stop the transfer only by nEND_DMA signal. ***RHEA_DEPTH isn't used in this mode.***	0	R/W
4	CURRENT_PAGE	0 = API access are done in the first API page 1 = API access are done in the 2nd API page ***This bit is automatic updated during transfer***.	0	R/W
5	MAS	0 = DMA transactions are done on 8-bit API and Rhea data 1 = DMA transactions are done on 16-bit API and Rhea data	1	R/W
6	DMA_START	0 = *No Effect* 1 = (Write) initiates the DMA transfer. ***Reading of this bit is always equal to zero.***	-	R/W
7	IRQ_MODE	0 = Interrupt is requested at the end of Rhea buffer transfer or if there is an nEND_DMA 1 = an interrupt has to be launched at the end of API page transfer,	1	R/W
8	IRQ_STATE	0 = cleared after being read 1 = an IRQ requested by DMA channel 4	0	R
9	RHEA_ERROR	0 = cleared after being read 1 = DMA RHEA access is aborted, an IRQ is also generated.	0	R
10	DIRECTION	1 = Transactions are done on Rhea -> API 0 = Transactions are done on API -> Rhea	1	R/W
12:11	PRIORITY	number of additional reading on the bus: 00 = zero read more, one reading 01 = one read more, 2 successive reading 10 = two read more, 3 successive reading 11 = three read more, 4 successive reading	00	R/W
15:13	-	*Reserved*		

11.6.6 DMA4_CUR_OFFSET_API (Read) – FFFF:FC4A – XIO:FC4A

DMA4 current OFFSET API

Bit	Name	Function	Reset
11:0	API_OFFSET	This register indicated offset API of the next reading or writing API memory.	00
15:12	*Unused*		-

You must read this register only if the DMA controller isn't running.

12. RIF REGISTERS - FFFF:7000 - XIO:0000

12.1 RIF register mapping

Register	Address	Address ARM	Access	Reset value
DXR	XIO:0000	FFFF:7000	16b R/W	Undefined
DRR	XIO:0001	FFFF:7002	16b R	Undefined
SPCX	XIO:0002	*n.a*	15b	000 0101 1001 1110
SPCR	XIO:0003	*n.a*	15b	011 1100 1010 0010

Table 19: RIF register

12.2 Transmit Data Register (DXR) – FFFF:7000 - XIO:0000

DXR is accessible via the Rhea DSP interface.

Bit	Name	Access	Function	Reset
15:0	DXR	R/W	Data to transmit	Undefined

12.3 Receive Data Register (DRR) – FFFF:7002 - XIO:0001

DRR is accessible via the Rhea DSP and ARM interfaces.

Bit	Name	Access	Function	Reset
15:0	DRR	R	Received Data DRR is updated when RSR is ready to be read and RRDY=1. *DRR cannot be written via the RHEA interface*.	Undefined

12.4 Shift Data Registers (XSR and RSR)

XSR and RSR are not directly accessible.
XSR is driven by CLKX.
RSR is driven by CLKR.

Note that the size of data is always 16 bits.

12.5 Control Register (SPCX) – XIO:0002

Bit	Name	Acc	Function	Reset
0	MCM	R/W	Clock source for rif_clkx. rif_clkx is taken from the clkx pin. rif_clkx is the internal ASIC clock.	0
1	XRST	W	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 made into the SPCX register. 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 TMS320C5X RIF. Clocks shut off must be performed externally of the RIF.	1
2	XRDY	R	*Unused*	1
3	CLKX_AUTO	R/W	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 shift of the lsb.	1
4	TXM	R/W	FSX configuration: 0 = input 1 = output	1
5	NCLK_EN	R/W	Clock enable for the internal transmission clock.	0
6	NCK13_EN	R/W	Clock enable for the reference clock (rif_clk13).	0
7	ALMOST_EMPTY	R	1 = transmit FIFO is almost empty.	1
8	FIFO_EMPTY	R	0 = the transmit FIFO is not empty.	1
9	FIFO_FULL	R	1 = the transmit FIFO is full.	0
11:10	THRESHOLD	R/W	Threshold level for FIFO-almost-empty flag. The maximum value is 2.	01
14:12	DIV_CLK	R/W	Frequency of transmission clock (ck13m). 000 = f13 001 = f13/2 010 = f13/4 011 = f13/8 100 = f13/16 others unused	000

12.6 Control Register (SPCR) – XIO:0003

Bit	Name	Acc	Function	Reset
0	DLB	R/W	Digital Loop Back Mode. 0 = Normal mode 1 = DR and FSR are connected in the RIF to, respectively, DX and FSX. The transmit clock is internally looped back to the receive clock.	0
1	RRST	R/W	Receiver reset: If the SPCR register 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 configuration and a one to RRST. ***Note:*** If RRST = 0, internal clocks of the serial port are not shut off as in the TMS320C5X RIF. Clocks shut off must be performed externally of the RIF. Writing a zero to RRST clears the RSRFULL bit and RRDY bit.	1
2	RRDY	R	*Unused*	0
3	RSRFULL*	R	Receive Shift Register Full: 0 = upon device reset and SPI receiver reset (RRST) 1 = a word has been shifted in the RSR register 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.	0
4	ALMOST_FULL	R	1 = receive FIFO is not-empty.	0
5	FIFO_EMPTY	R	0 = receive FIFO is empty.	1
6	FIFO_FULL	R	1 = receive FIFO is full.	0
9:7	THRESHOLD	R/W	Threshold of receive FIFO-not-empty-flag. The maximum value is 4.	001
10	XINT_MASK	R/W	1 = RIF transmit-interrupt masked	1
11	RINT_MASK	R/W	1 = RIF receive-interrupt masked	1
12	XDMA_MASK	R/W	1 = RIF transmit DMA-request masked	1
13	RDMA_MASK	R/W	1 = RIF receive DMA-request masked	1
14	CLKLB	R/W	Clock Loop Back Mode bit. 1 = Receive & Transmit clocks are generated from the same internal source.	0

Note: The RSRFULL bit is not reset by a DRR read event. This feature differs from the TMS320C4X RIF functional specification. This allows keeping trace of a reception problem. The RV_WAIT state of the receive state machine matches with the TMS320C4X 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 resets.

13. CYPHER REGISTERS - XIO:2800

13.1 CYPHER register mapping

register	address	access	reset value
CNTL_REG	2800 (00)	6 bits R/W	00 0000
STATUS_IRQ_REG	2801 (01)	1 bit R	0
STATUS_WORK_REG	2802 (02)	1 bit R	0
KC_REG_1	2803 (03)	16 bits R/W	0000 0000 0000 0000
KC_REG_2	2804 (04)	16 bits R/W	0000 0000 0000 0000
KC_REG_3	2805 (05)	16 bits R/W	0000 0000 0000 0000
KC_REG_4	2806 (06)	16 bits R/W	0000 0000 0000 0000
COUNT_REG_1	2807 (07)	11 bits R/W	000 0000 0000
COUNT_REG_2	2808 (08)	11 bits R/W	000 0000 0000
DECI_REG_1	2809 (09)	16 bits R	0000 0000 0000 0000
DECI_REG_2	280A (0A)	16 bits R	0000 0000 0000 0000
DECI_REG_3	280B (0B)	16 bits R	0000 0000 0000 0000
DECI_REG_4	280C (0C)	16 bits R	0000 0000 0000 0000
DECI_REG_5	280D (0D)	16 bits R	0000 0000 0000 0000
DECI_REG_6	280E (0E)	16 bits R	0000 0000 0000 0000
DECI_REG_7	280F (0F)	16 bits R	0000 0000 0000 0000
DECI_REG_8	2810 (10)	2 bits R	00
ENCI_REG_1	2811 (11)	16 bits R	0000 0000 0000 0000
ENCI_REG_2	2812 (12)	16 bits R	0000 0000 0000 0000
ENCI_REG_3	2813 (13)	16 bits R	0000 0000 0000 0000
ENCI_REG_4	2814 (14)	16 bits R	0000 0000 0000 0000
ENCI_REG_5	2815 (15)	16 bits R	0000 0000 0000 0000
ENCI_REG_6	2816 (16)	16 bits R	0000 0000 0000 0000
ENCI_REG_7	2817 (17)	16 bits R	0000 0000 0000 0000
ENCI_REG_8	2818 (18)	2 bits R	00

Table 20: CYPHER registers

13.2 Control Register (CNTL_REG) – XIO:2800

Bit	Name	Function	HW Reset	Acc
0	START	Start ciphering: 0 = *No Effect* 1 = starts the process (rising edge only) Toggle bit (always at 0 when read)	0	R/W
1	RESET_SW	0 = Module reset 1 = *No Effect*	0	R/W
3:2	MODE	Defines which algorithm is used 00 = no algorithm, outputs are forced to zero, inputs do not matter 01 = algorithm A51 10 = algorithm A52 11 = forbidden	0	R/W
4	CLK_EN	Internal clock 0 = Disable 1 = Enable	0	R/W
5	cypher_only	0 = both decipher and encipher data are performed 1 = Only the decipher data are performed	0	R/W
15:6	*Unused*		*Undef.*	

13.3 Interrupt status register (STATUS_irq_REG) – XIO:2801

Bit	Name	Function	Reset	Acc
0	It_fin	1= ciphering is completed. Remain 1 until read	0	R
15:2	Unused		*Undef*	

It_fin is reset when on read access.

13.4 Working status register (STATUS_work_REG) – XIO:2802

Bit	Name	Function	HW Reset	Acc
0	working	1 = Ciphering on-going	0	R
15:1	Unused		*Undef*	

13.5 Kc registers 1 to 4 (KC_REG#) – XIO:2803 .. 2806

Bit	Name	Function	HW Reset	Acc
15:0	KC_REG#	Contains 16 bits of Kc	0	R/W

The key is written into the Kc registers as it‘s described in the following table.

	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

Bit 1 is lsb of Kc, bit 54 is msb of Kc.

13.6 COUNT registers 1 and 2 (COUNT_REG#) – XIO:2807.. 2808

Bit 1 is lsb of COUNT, bit 22 is msb of COUNT.Then, bits should be scaled correctly for COUNT before the process starts.

.Bit	Name	Function	HW Reset	Acc
10:0	COUNT_REG#	Contains 11 bits of COUNT	0	R/W
15:11	*unused*		0	

This registers contain the frame number COUNT as specified in the following table

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

13.7 Decipher data registers 1 to 8 (DECI_REG_#) XIO 2809 .. 2810

These registers contain the 114 bits of BLOCK1.

.Bit	Name	Function	Reset	Acc
15:0	DECI_REG#	Contains 16 DECI_REG bits	0000	R

The less significant bit of BLOCK 1 is in DECI_REG_1[15] and the most significant bit is in DECI_REG_8[14].

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

13.8 Encipher data register 1 to 8 (ENCI_REG_#) – XIO:2811 .. 2818

These registers contain the 114 bits of BLOCK 2.

.Bit	Name	Function	Reset	Acc
15:0	ENCI_REG#	Contains 16 ENCI_REG bits	0000	R

The less significant bit of BLOCK2 in ENCI_REG_1[15] and the most significant bit in ENCI_REG_8[14].

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

14. MCSI REGISTERS – XIO:0800

14.1 MCSI register mapping

The mcsi is on the chip select number 1

register	Address	access	reset value
Rx15	083F (3F)	16 bits R	???? ???? ???? ????
Rx14	083E (3E)	16 bits R	???? ???? ???? ????
Rx13	083D (3D)	16 bits R	???? ???? ???? ????
Rx12	083C (3C)	16 bits R	???? ???? ???? ????
Rx11	083B (3B)	16 bits R	???? ???? ???? ????
Rx10	083A (3A)	16 bits R	???? ???? ???? ????
Rx9	0839 (39)	16 bits R	???? ???? ???? ????
Rx8	0838 (38)	16 bits R	???? ???? ???? ????
Rx7	0837 (37)	16 bits R	???? ???? ???? ????
Rx6	0836 (36)	16 bits R	???? ???? ???? ????
Rx5	0835 (35)	16 bits R	???? ???? ???? ????
Rx4	0834 (34)	16 bits R	???? ???? ???? ????
Rx3	0833 (33)	16 bits R	???? ???? ???? ????
Rx2	0832 (32)	16 bits R	???? ???? ???? ????
Rx1	0831 (31)	16 bits R	???? ???? ???? ????
Rx0	0830 (30)	16 bits R	???? ???? ???? ????
Tx15	082F (2F)	16 bits R	???? ???? ???? ????
Tx14	082E (2E)	16 bits R	???? ???? ???? ????
Tx13	082D (2D)	16 bits R	???? ???? ???? ????
Tx12	082C (2C)	16 bits R	???? ???? ???? ????
Tx11	082B (2B)	16 bits R	???? ???? ???? ????
Tx10	082A (2A)	16 bits R	???? ???? ???? ????
Tx9	0829 (29)	16 bits R	???? ???? ???? ????
Tx8	0828 (28)	16 bits R	???? ???? ???? ????
Tx7	0827 (27)	16 bits R	???? ???? ???? ????
Tx6	0826 (26)	16 bits R	???? ???? ???? ????
Tx5	0825 (25)	16 bits R	???? ???? ???? ????
Tx4	0824 (24)	16 bits R	???? ???? ???? ????
Tx3	0823 (23)	16 bits R	???? ???? ???? ????
Tx2	0822 (22)	16 bits R	???? ???? ???? ????
Tx1	0821 (21)	16 bits R	???? ???? ???? ????
Tx0	0820 (20)	16 bits R	???? ???? ???? ????
unused			
/	/		
unused			
STATUS_REG	0806 (06)	7 bits	0000 0000 0000 0000
CLOCK FREQUENCY_REG	0805 (05)	11 bits R/W	0000 0000 0000 0000
OVER-CLOCK_REG	0804 (04)	10 bits R/W	0000 0000 0000 0000
CHANNEL USED_REG	0803 (03)	16 bits R/W	0000 0000 0000 0000
INTERRUPTS_REG	0802 (02)	11 bits R/W	0000 0000 0000 0000
MAIN PARAMETERS_REG	0801 (01)	14 bits R/W	0000 0000 0000 0000
CONTROL_REG	0800 (00)	3 bits R/W	0000 0000 0000 0000

Table 21: MCSI registers

14.2 Control registers - XIO:0803 .. 0805

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)

14.2.1 CHANNEL_USED_REG register - XIO:0803

Configuration for channels selection. This register is only used in Multi-Channels mode.

Bit	Name	Function	Acc	Reset
0	use_ch0	Data-transmission on channel 0 0 = unselected 1 = selected	R/W	0
1	use_ch1	Data-transmission on channel 1 0 = unselected 1 = selected	R/W	0
2	use_ch2	Data-transmission on channel 2 0 = unselected 1 = selected	R/W	0
3	use_ch3	Data-transmission on channel 3 0 = unselected 1 = selected	R/W	0
4	use_ch4	Data-transmission on channel 4 0 = unselected 1 = selected	R/W	0
5	use_ch5	Data-transmission on channel 5 0 = unselected 1 = selected	R/W	0
6	use_ch6	Data-transmission on channel 6 0 = unselected 1 = selected	R/W	0
7	use_ch7	Data-transmission on channel 7 0 = unselected 1 = selected	R/W	0
8	use_ch8	Data-transmission on channel 8 0 = unselected 1 = selected	R/W	0
9	use_ch9	Data-transmission on channel 9 0 = unselected 1 = selected	R/W	0
10	use_ch10	Data-transmission on channel 10 0 = unselected 1 = selected	R/W	0
11	use_ch11	Data-transmission on channel 11 0 = unselected 1 = selected	R/W	0
12	use_ch12	Data-transmission on channel 12 0 = unselected 1 = selected	R/W	0
13	use_ch13	Data-transmission on channel 13 0 = unselected 1 = selected	R/W	0
14	use_ch14	Data-transmission on channel 14 0 = unselected 1 = selected	R/W	0
15	use_ch15	Data-transmission on channel 15 0 = unselected 1 = selected	R/W	0

14.2.2 CLOCK_FREQUENCY_REG register - XIO:0805

In Master mode, this register defines the transmission baud-rate from a frequency ratio based on a 13MHz-reference clock.
This register is used only in MASTER mode when the interface generates the serial clock.

Bit	Name	Function	Acc	Reset
10:0	CLK_FREQ	Division factor of 13MHz reference clock: range: 2 to 2047	R/W	0
15:11	*Unused*	-	R	0

Note: *The transmission clock frequency can be programmed from 6.3KHz to 6.5MHz in step of 76ns:*
clock frequency = 13MHz / clk_freq with $2 \le clk\_freq \le 2047$.

14.2.3 OVER_CLOCK_REG register XIO:0804

Over-sized frame dimension

Bit	Name	Function	Acc	Reset
9:0	OVER_CLOCK	Over clock periods in frame duration: range: 0 to 1023	R/W	0
15:10	*Unused*	-	R	0

14.2.4 INTERRUPTS_REG register XIO:0802

Interrupts masks

Bit	Name	Function	Acc	Reset
3:0	NB_CHAN_IT_RX	channel number for receive interrupt range: 0 to 15	R/W	0
7:4	NB_CHAN_IT_TX	channel number for transmit interrupt range: 0 to 15	R/W	0
8	MASK_IT_RX	receive interrupt 0 = mask 1 = unmask	R/W	0
9	MASK_IT_TX	Transmit interrupt 0 = mask 1 = unmask	R/W	0
10	MASK_IT_ERROR	frame duration error interrupt 0 = mask 1 = unmask	R/W	0
15:11	*Unused*	-	R	0

14.2.5 MAIN_PARAMETERS_REG register XIO:0801

Bit	Name	Function	Acc	Reset
3:0	WORD_SIZE	Word size in bits number: range: 2 to 15 (*with 2 for 3 bits and 15 for 16 bits*)	R/W	0
4	CLOCK_POL	Clock edge selection: 0 = rising 1 = falling	R/W	0
5	CONTINUOUS	Frame mode: 0 = burst 1 =continuous	R/W	0
6	MCSI_MODE	Interface transmission mode: 0 = slave 1= master	R/W	0
7	MULTI	Frame structure:	R/W	0
8	FSYNCH_SIZE	frame synchronization pulse shape: 0 = short 1 = long	R/W	0
9	FSYNCH_MODE	frame synchronization pulse position: 0 = normal 1 = alternate	R/W	0
10	FSYNCH_POL	frame synchronization pulse polarity: 0 = positive 1 = negative	R/W	0
11	DAI_SYN_REQ	synchronization with audio frame: 0 = no sync 1 = synchronized	R/W	0
13:12	DAI_CONF	DAI mode selection: 00 = normal (no DAI) 01 = RADIO down-link 10 = RADIO up-link 11 = acoustic	R/W	0
15:14	*Unused*	-	R	0

14.2.6 CONTROL_REG register XIO:0800

Bit	Name	Function	Acc	SW Reset	HW Reset
0	CLK_ENABLE	Clock of MCSI module: 0 = disable 1 = enable	R/W	0	0
1	SW_RESET	Asynchronous reset of module: 0 = disable 1 = enable	R/W	1	0
2	DAICLKEN	DAI interface activity: 0 = disable 1 = enable	R/W	0	0
15:3	*Unused*	-	R	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 mcsi (the MCSI clk enable bit is cleared), initialized the status register and don’t modified the others registers).*

14.2.7 STATUS_REG register - XIO:0806

Bit	Name	Function	Acc	SW Reset	HW Reset
0	FRAME_ERROR	Frame duration error: 0 = no error 1 = wrong frame duration	R/W	0	0
1	ERROR_TYPE	Error type: 0 = too short frame 1 = too long frame	R	0	0
2	RX_READY	Receive interrupt: 0 = none 1 = pending	R/W	0	0
3	RX_OVFLOW	Receive overflow error: 0 = no error 1 = overflow	R	0	0
4	TX_READY	Transmit interrupt: 0 = none 1 = pending	R/W	0	0
5	TX_UNFLOW	Transmit underflow: 0 = no error 1 = underflow error	R	0	0
6	DAI_READY	System-Simulator reset: 0 = not detected 1 = detected	R/W	0	0
15:7	*Unused*	-	R	0	0

14.3 Data registers - XIO:0820 .. 083F

14.3.1 RX_REG[15:0] XIO:0830 .. 083F

Receive word register

	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 bit of the Rx registers contains the received data and the lower 4 bits being zeroes*.

14.3.2 TX_REG[15:0] XIO:0820 .. 082F

Transmit word register

	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 bit of the Tx registers is transmitted*.

15. DSP INTERRUPTS - XIO:FA00

15.1 DSP interrupts Mapping

The DSP subchip owns 17 interrupt lines with 11 of which INT0n to INT10n are dedicated for external peripherals. 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 follows:

Name	Sense	Location (hex)	Function
RSN			reset (HW or SW)
nMIN		4	Abort on Rhea bus OR INT4n redirection
TINT		4C	Timer interrupts
RINT		50	SPI receive interrupt
XINT		54	SPI transmit interrupt
AINT		64	API 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 timeout. 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. error of processing
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 ¼-GSM bit accuracy. The interrupt is set in a scenario by using a time-stamped instruction*.

15.2 Internal registers XIO:FA00 .. FA01

The XIO Interrupt Processor has one 16-bit control register and one "non-implemented" command register.

The control register is used exclusively for assigning edge-triggered / level-sensitive status to each of the 12 interrupts channels.

The "non-implemented" command register is a block of decoding logic that issues "clear" commands to the level-sensitive logic in each interrupt channel upon detecting a RHEA Bus *write* transaction to an address that falls within the required address range.

Register	Address	Access	Reset value
CNTRL_REG	XIO:FA00	13bits	???0 0000 0000 0000
CLEAR_REG	XIO:FA01	12bits W	

Table 22: DSP interrupts registers

15.2.1 Edge-Triggered / Level-Sensitive Control Register - XIO:FA00

The bit-alignment of interrupt channel assignments within the control register, the definition of the assignments, the default values at power turn-on, the address used to write to the control register, and the address used to read the content of the register are all presented in Figure 5.

This register is mapped in DSP IO space on nXSTROBE(3)
Write address 1111 101X ???? ??X0
Read address 1111 101X ???? ????

12	11	10	9	8	7	6	5	4	3	2	1	0
int4 switch	Ch11 Assg	Ch10 Assg	Ch9 Assg	Ch8 Assg	Ch7 Assg	Ch6 Assg	Ch5 Assg	Ch4 Assg	Ch3 Assg	Ch2 Assg	Ch1 Assg	Ch0 Assg

Bit	Name	Function	Reset
11:0	CHx Trig/Level	Channel CHx sense: 1 = CHx is edge sensitive 0 = CHx is level sensitive	0
12	INT4 switch	Channel 4 connection: 0 = Channel 4 is connected to DSP INT4N (0x58) 1 = Channel 4 is connected to DSP nNMI (0x4)	0
15:13	*Reserved*	-	-

15.2.2 Level-Sensitive "Clear" Commands - XIO:FA01

A C54x write transaction in I/O Space (nXSTROBE(3)) at an odd-valued address in the range of to 0xFA01 through 0xFBFF will result in a clear being issued to those interrupt channels whose assigned bit in the 16-bit word being written is a logic '1'.

Commands to clear interrupt channels are necessary for those channel assigned as level-sensitive interrupt channels and designated as shared channels. Figure 6 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.

15.2.3 NMI interrupt

Bit 12 of the control register (int4_switch) permit 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,and NMI='1'
- int4_switch ='1' => nxirq(4) connected to the NMI of the DSP, and INT4N='1'

16. MCU INTERRUPTS – FFFF:FA00

16.1 MCU interrupts mapping

The ARM7 owns 2 interrupt lines nIRQ and nFIQ. The ABB fast interrupt is mapped on nFIQ
All peripheral interrupts are mapped as follows:

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 timeout 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*** I2C data transfer error / completion
IRQ11	edge	✓		ULPD end of gauging interrupt
IRQ12	level	✓		External interrupt
IRQ13	edge	✓		SPI interrupt 1. received data 2. data to transmit
IRQ14	level	✓		DMA interrupt
IRQ15	edge	✓		API interrupts (nHINT)
IRQ16			✓	SIM card-detect fast interrupt
IRQ17			✓	Fast external interrupt
IRQ18	level	✓		UART_IRDA interrupts 1. error on receiver line 2. receive timeout 3. received character 4. character to transmit 5. modem status change 6. Received XOFF / special character detected 7. CTS/RTS deactivation 8. RxD/CTS activity detection (OFF mode only)
IRQ19	level	✓		ULPD GSM timer
IRQ20	level	✓		GEA interrupt

16.2 Interrupt sequence

As IRQ and FIQ treatment are exactly identical, the following sequence is described only for IRQ interrupt.

- One or several incoming interrupts go down, setting the corresponding ITR bits.
- At this time, there are two possible cases :
  - There is only one incoming interrupts, which is active. If IRQ is not already active Interrupt handler sends an IRQ.
  - There are several incoming interrupts, which are active. In this case, Interrupt Handler must determine which is the new interrupt 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 Interrupt handler sends an IRQ.
- When an IRQ is sent, SIR_IRQ register is updated (indicating the interrupt contained in N_IRQ) and the priority resolver is reset (and restart if necessary)
- To know which is the incoming interrupt having requested a MCU action, this one must read SIR_IRQ_CODE register. After that, it runs the corresponding sub-routine.
- To finish this sequence **MCU software must set a dedicated bit (NEW_IRQ_AGR of Control register) in order to reset IRQ output and SIR_IRQ register, and thus, to allow a new IRQ generation.**

16.3 INTH register - FFFF:FA00 .. FA46

All these registers are controlled directly by the internal RHEA bus. The Interrupt handler is selected when CS(2:0) is equal to 2.

Register	Address	Access	HW Reset value
IT_REG1	FFFF:FA00	16 bits R	0000 0000 0000 0000
IT_REG2	FFFF:FA02	4 bits R	???? ???? ???0 0000
MASK_IT_REG1	FFFF:FA08	16 bits R/W	1111 1111 1111 1111
MASK_IT_REG2	FFFF:FA0A	4 bits R/W	1111 1111 1111 1111
SRC_IRQ_BIN_REG	FFFF:FA10	5bits R	???? ???? ???0 0000
SRC_FIQ_BIN_REG	FFFF:FA12	5bits R	???? ???? ???0 0000
INT_CTRL_REG	FFFF:FA14	2bits R/W	???? ???? ???? ??00
ILR_IRQ0_REG	FFFF:FA20	7 bits R/W	???? ???? ?000 0000
ILR_IRQ1_REG	FFFF:FA22	7 bits R/W	???? ???? ?000 0000
ILR_IRQ2_REG	FFFF:FA24	7 bits R/W	???? ???? ?000 0000
ILR_IRQ3_REG	FFFF:FA26	7 bits R/W	???? ???? ?000 0000
ILR_IRQ4_REG	FFFF:FA28	7 bits R/W	???? ???? ?000 0000
ILR_IRQ5_REG	FFFF:FA2A	7 bits R/W	???? ???? ?000 0000
ILR_IRQ6_REG	FFFF:FA2C	7 bits R/W	???? ???? ?000 0000
ILR_IRQ7_REG	FFFF:FA2E	7 bits R/W	???? ???? ?000 0000
ILR_IRQ8_REG	FFFF:FA30	7 bits R/W	???? ???? ?000 0000
ILR_IRQ9_REG	FFFF:FA32	7 bits R/W	???? ???? ?000 0000
ILR_IRQ10_REG	FFFF:FA34	7 bits R/W	???? ???? ?000 0000
ILR_IRQ11_REG	FFFF:FA36	7 bits R/W	???? ???? ?000 0000
ILR_IRQ12_REG	FFFF:FA38	7 bits R/W	???? ???? ?000 0000
ILR_IRQ13_REG	FFFF:FA3A	7 bits R/W	???? ???? ?000 0000
ILR_IRQ14_REG	FFFF:FA3C	7 bits R/W	???? ???? ?000 0000
ILR_IRQ15_REG	FFFF:FA3E	7 bits R/W	???? ???? ?000 0000
ILR_IRQ16_REG	FFFF:FA40	7 bits R/W	???? ???? ?000 0000
ILR_IRQ17_REG	FFFF:FA42	7 bits R/W	???? ???? ?000 0000
ILR_IRQ18_REG	FFFF:FA44	7 bits R/W	???? ???? ?000 0000
ILR_IRQ19_REG	FFFF:FA46	7 bits R/W	???? ???? ?000 0000
ILR_IRQ20_REG	FFFF:FA48	7 bits R/W	???? ???? ?000 0000

Table 23: INTH registers

16.4 IT register (Read only) – FFFF:FA00 .. FA02

In case of edge sensitive interrupt, it stores an incoming interrupt. When MCU accesses SIR_IRQ_CODE or SIR_FIQ_CODE register the bit corresponding to the interrupt which have requested MCU action is reset

MCU can also clear individually each bit. To do this, MCU must write a 0 to the corresponding bits (at ITR address) (Others bits will keep their previous value). This possibility could be used just before MCU unmasks some interrupts and thus, allows to “forget” some interrupt occurrences.

MCU can read this register. If incoming interrupt is edge sensitive, the read value corresponds to the value held in the storage element. In the other case, the read value corresponds to the inverted value of the incoming interrupt.

IT_REG1 = FFFF:FA00

	15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	irq 15	irq 14	irq 13	irq 12	irq 11	irq 10	irq 9	irq 8	irq 7	irq 6	irq 5	irq 4	irq 3	irq 2	irq 1	irq 0
Acc.	r	r	r	r	r	R	r	r	r	r	r	r	r	r	r	r
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IT_REG2 = FFFF:FA02

	15:5	**4**	**3**	**2**	**1**	**0**
Name	*reserved*	irq 20	irq 19	irq 18	irq 17	irq 16
Acc.	R	r	r	r	r	r
Reset	0	0	0	0	0	0

16.5 Mask Interrupt Register (Read / Write) - FFFF:FA08 .. FA0A

Each incoming interrupt can be masked individually by this register. MIR operates after ITR, that means that occurrences of incoming interrupt are always stored in ITR.

MASK_IT_REG1 = FFFF:FA08

Bit	Name	Function	Interrupt Source (§15.1)	Reset
0	IRQ_0_MSK	Disable IRQ_0 interrupt	Watchdog Timer	1
1	IRQ_1_MSK	Disable IRQ_1 interrupt	Timer1	1
2	IRQ_2_MSK	Disable IRQ_2 interrupt	Timer2	1
3	IRQ_3_MSK	Disable IRQ_3 interrupt	TSP receive	1
4	IRQ_4_MSK	Disable IRQ_4 interrupt	TPU frame	1
5	IRQ_5_MSK	Disable IRQ_5 interrupt	TPU page	1
6	IRQ_6_MSK	Disable IRQ_6 interrupt	SIM	1
7	IRQ_7_MSK	Disable IRQ_7 interrupt	UART Modem	1
8	IRQ_8_MSK	Disable IRQ_8 interrupt	Keyboard	1
9	IRQ_9_MSK	Disable IRQ_9 interrupt	RTC periodical timer	1
10	IRQ_10_MSK	Disable IRQ_10 interrupt	RTC alarm or I2C	1
11	IRQ_11_MSK	Disable IRQ_11 interrupt	ULPD end of gauging	1
12	IRQ_12_MSK	Disable IRQ_12 interrupt	External	1
13	IRQ_13_MSK	Disable IRQ_13 interrupt	SPI	1
14	IRQ_14_MSK	Disable IRQ_14 interrupt	DMA	1
15	IRQ_15_MSK	Disable IRQ_15 interrupt	API	1

MASK_IT_REG2 = FFFF:FA0A

Bit	Name	Function	Interrupt Source (§15.1)	Reset
0	IRQ_16_MSK	Disable IRQ_16 interrupt	SIM card detect	1
1	IRQ_17_MSK	Disable IRQ_17 interrupt	Fast external	1
2	IRQ_18_MSK	Disable IRQ_18 interrupt	UART IrDa	1
3	IRQ_19_MSK	Disable IRQ_19 interrupt	ULPD GSM-timer	1
4	IRQ_20_MSK	Disable IRQ_20 interrupt	GEA module	1
15:5	*Reserved*	*Reserved*	*None*	1

16.6 Source IRQ binary coded Register (Read only) - FFFF:FA10

Indicates the active interrupt In order to save software processing time, that register indicates the interrupt number having requested a MCU action.

Bit	Name	Reset
4:0	IRQ_NUM	0

16.7 Source FIQ binary coded Register (Read only) - FFFF:FA12

Indicates the active interrupt. In order to save software processing time, that register indicates the interrupt number having requested a MCU action.

Bit	Name	Reset
4:0	FIQ_NUM	0

16.8 Control register (Read / Write) - FFFF:FA14

Bit	Name	Function	Reset
0	NEW_IRQ_AGR	New IRQ Agreement. Reset IRQ output Clear Source IRQ Register Enables a new IRQ generation Active at level 1 Reset by internal logic	0
1	NEW_FIQ_AGR	New FIQ Agreement. Reset FIQ output Clear Source FIQ Register Enables a new FIQ generation Active at level 1 Reset by internal logic	0

Warning **:** *IRQ (FIQ) output and 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 MCU action is already cleared or masked.*

The time where 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's reset when the corresponding interrupt becomes inactive.

16.9 Interrupt Level Registers (Read / Write) - FFFF:FA20 .. FA46

There is one ILR per incoming interrupt.

Offset Address (hex)	Name	Corresponding Interrupt	Interrupt Source (§15.1)
20	ILR_IRQ_0	IRQ_0	Watchdog Timer
22	ILR_IRQ_1	IRQ_1	Timer1
24	ILR_IRQ_2	IRQ_2	Timer2
26	ILR_IRQ_3	IRQ_3	TSP receive
28	ILR_IRQ_4	IRQ_4	TPU frame
2A	ILR_IRQ_5	IRQ_5	TPU page
2C	ILR_IRQ_6	IRQ_6	SIM
2E	ILR_IRQ_7	IRQ_7	UART Modem
30	ILR_IRQ_8	IRQ_8	Keyboard
32	ILR_IRQ_9	IRQ_9	RTC periodical timer
34	ILR_IRQ_10	IRQ_10	RTC alarm or I2C
36	ILR_IRQ_1	IRQ_11	ULPD end of gauging
38	ILR_IRQ_12	IRQ_12	External
3A	ILR_IRQ_13	IRQ_13	SPI
3C	ILR_IRQ_14	IRQ_14	DMA
3E	ILR_IRQ_15	IRQ_15	API
40	ILR_IRQ_16	IRQ_16	SIM card detect
42	ILR_IRQ_17	IRQ_17	Fast external
44	ILR_IRQ_18	IRQ_18	UART IrDa
46	ILR_IRQ_19	IRQ_19	ULPD GSM-timer
48	ILR_IRQ_20	IRQ_20	GEA module

Bit	Name	Function	Reset
0	FIQ	1 = The corresponding interrupt is routed to FIQ 0 = The corresponding interrupt is routed to IRQ	0
1	SENS_EDGE	1 = Corresponding interrupt is falling edge sensitive 0 = The corresponding interrupt is low level sensitive	0
6:2	PRIORITY	Priority level when the corresponding interrupt is routed to IRQ. 0 = is the highest priority level 31 = is the lowest priority level	0

16.10 Predefined order in case of identical priority level

Assuming that all interrupts have the same priority level and they are active at the same at the same moment, the order of servicing will be this one: IRQ_N-1 , IRQ_N-2, …. IRQ_0

17. ARM7 TO RHEA – FFFF:F900

17.1 ARM7 Rhea register mapping

All these registers are controlled directly by the internal RHEA bus. The ARM-RHEA bridge is selected when CS(2:0) is equal to 1.

Register	Address	Access	HW Reset value
RHEA_CNTL_REG	FFFF:F900	16bits R/W	1111 1111 0010 0000
API_WS_REG	FFFF:F902	10bits R/W	???? ??11 1110 0000
ARM-RHEA_CNTL_REG	FFFF:F904	2bits R/W	???? ???? ???? ??11
ENHANCED_RHEA_CTL	FFFF:F906	1bit R/W	???? ???? ???? ???1

Table 24: ARM7 to RHEA registers

17.2 Rhea_cntl_reg (Read / Write) - FFFF:F900

Once ARM clock frequency has been selected, programmer must adapt ARM accesses duration to the timing of peripherals connected on the RHEA bus.
For this purpose the RHEA_CNTL_REG defines an access factor which allows to adapt the RHEA access duration to slow peripheral.

1 ACCESS_FACTOR0 (3:0) allows to match access duration to slow peripheral on strobe 0
2 ACCESS_FACTOR1 (7:4) allows to match access duration to slow peripheral on strobe 1

Value for the access factors is obtained using the following formula:

$$Access\_Factor \geq \frac{ARM\_Access\_Freq}{Pheripheral\_Accces\_Freq} - 1$$

<u>Note:</u> **Pheripheral_Acces_Freq=**

For revA: 39 MHz for all Pheripherals except GEA=36MHz and DMA=34MHz

For revB: 39 MHz for all Pheripherals

For Calypso C035-F751619 : 52 MHz for all Pheripherals

Bit	Name	Function	Reset
3:0	ACCESS_FACTOR0	Division factor of nASTROBE(0). Allows accessing slow peripherals by reducing the access frequency.	0
7:4	ACCESS_FACTOR1	Division factor of nASTROBE(1). Allows accessing slow peripherals by reducing the access frequency.	0010
15:8	TIMEOUT*	RHEA buses access time out. Allows to limit the maximum time a peripheral can stall the processor When starting a cycle on RHEA bus, the time out counter is loaded with this value. If the current cycle is not finished when the counter reach 0, cycle is aborted.(by sending ARM_ABORT to the ARM (or DMA_ABORT to the DMA) and AABORT to the peripheral)	FF

(*) <u>nASTROBE low level pulse duration:</u>

- *access_factor = 0 → bridge_clk low level.*
- *access_factor != 0 → access_factor ⇔ number of bridge_clk periods to use.*
- *Max value is 256 which allows to have a time out of 6.56 us if bridge_clk is equal to 39MHz*

(*) see Ch 17.6

17.3 Api_ws_reg (Read / Write) - FFFF:F902

The API memory is a dual access memory which can be configured in 2 states :

- HOM (Host Only Mode) with the memory dedicated to the ARM (no access possible from DSP)
- SAM (Shared Access Mode) with the memory access shared between ARM and DSP

In HOM mode, the ARM access to the memory is fully asynchronous and does not imply any time constraints on the ARM access frequency, nevertheless the ARM access frequency must take care of the access time of the API memory (technology dependant).

In SAM mode, the ARM access is resynchronized on the DSP cycle clock and the duration of the ARM access must respect the following rule :

$$LEAD\_CLK\_freq \geq 4 \times ARM\_ACCESS\_freq$$

If the ARM cycle frequency is not compliant with this rule, then wait states should be inserted during the ARM access to increase the access time duration.

The register API_WS_REG defines this number of wait states inserted during an ARM access to API memory depending on DSP mode:

- API_WS_H defines the wait states inserted when DSP is in HOME mode,
- API_WS_S defines the wait states inserted when DSP is in SAM mode.

Number of wait states can be calculated using the following formulas:

$$\left(4 \times \frac{MCU\_frq}{DSP\_frq}\right) - 1 = WS\_S \qquad \left(\frac{MCU\_frq}{Mem\_frq}\right) - 1 = WS\_H$$

Where: Mem_frq = 40 MHz

The HOM / SAM mode is selected by the DSP itself. Information is given to the ARM interface-to-API by setting / clearing the 1 bit in the API_CONF register (§7.1), and the selected wait state register also depends of the value of bit 5 in CNTL_CLK register (§17.3).

The number of wait-state is selected according to the following table:

CNTL_RST bit(1)	DSP in IDLE3	API_CONF bit(1)	Selected register
1 (DSP reset)	x	x	API_WS_H
0 (DSP run)	yes		
	no	1 (HOM mode)	API_WS_H
		0 (SAM mode)	API_WS_S

Bit	Name	Function	Reset
4:0	API_WS_H	Indicates the number of wait states inserted for each API access when DSP is in HOME mode	00
9:5	API_WS_S	Indicates the number of wait states inserted for each API access when DSP is in SAM mode.	1F

17.4 Arm_rhea_cntl_reg (Read / Write) - FFFF:F904

Bit	Name	Function	Reset
0	W_BUF_EN_0	0 = Write buffer is bypassed. 1 = Write buffer is enabled for strobe domain 0	1
1	W_BUF_EN_1	0 = Write buffer is bypassed 1 = Write buffer is enabled for strobe domain 1	1

17.5 Enhanced_rhea_cntl (Read / Write) - FFFF:F906

Bit	Name	Function	Reset
0	TIMEOUT_EN	0 = Timeout disable 1 = Timeout enable	1

17.6 Maximum latency for each peripherals

There are two formulas for latency calculation depending of the usage/or-not of synchro-block in the design of the peripheral.
If peripheral use synchro-read &/or synchro-write block the formula is

$$MAxLatency = \left(\frac{2\times\frac{Fmcu}{Fperi}}{AF+1} \right) + 3$$

Else the formula is:

$$MAxLatency = (AF+1)$$

- The latency unit is MCU clock period
- *Fmcu* is the MCU input frequency programmed thru "DPLL control register" and "CNTL_ARM_CLK" register
- AF is the Rhea access factor programmed thru "Rhea_cntl_reg" register.
- *Fperi* is the peripheral usage frequency (described in following table)

Following table describe the peripheral frequency usage and the use/no-use of synchro block.

Module Name	***Synchro block***		***Peripheral frequency Fperiph (MHz)***	***Maximum Latency (MCU cycle)***		
	read	***write***		***AF=0***	***AF=1***	***AF=2***
DMA	✓		13	9	6	5
I2C	✓		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 IrDA	✓	✓	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
PWT		✓	13	9	6	5
RIF		✓	13	9	6	5
TPU			13	1	2	3
DPLL			13	1	2	3
WATCHDOG			13	1	2	3
RHEA 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
UWIRE			13	1	2	3
ARMIO			13	1	2	3
RTC			13	1	2	3
LPG			13	1	2	3
PWL			13	1	2	3

18. DPLL REGISTER – FFFF:9800

18.1 DPLL register mapping

Register	Address	Access	Reset value
DPLL_CTRL	FFFF:9800	16bits R/W	

Table 25: DPLL register

18.2 DPLL functionality

The DPLL has two modes of operation – the bypass mode and the lock mode.

18.2.1 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.

$$Clk_{out} = \frac{F_{in}}{k} \qquad k = 1, 2 \text{ or } 4$$

18.2.2 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:

$$Clk_{out} = F_{in} \times \frac{PLL\_MULT}{PLL\_DIV + 1} \qquad 1 < \text{PLL\_MULT} < 31 \qquad \text{PLL\_DIV} = 0, 1 \text{ or } 2$$

18.2.3 Lock times

The lock times depend on the values of PLL_MULT and PLL_DIV and the clkout frequency as given below: (in number clkref cycles)

$$Lt = \frac{[4 \times (PLL\_DIV + 1) \times 11D] + 20}{F_{in}}$$

where: $$D = 1 + \log 2\left(\frac{PLL\_DIV + 1}{PLL\_MULT \times Clk_{out} \times \aleph \min} \right)$$

where: $$\aleph \min = 5 \times 10^{-9}$$
$$Fin = 13\,\mathrm{MHz}$$

18.2.4 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 Rhea address space. If the DPLL was operating in the synthesized mode, it will switch automatically to the bypass mode. Depending on the new control content, the DPLL may either initiates a new lock sequence or continues to remain in the bypass mode.

18.3 DPLL control register (Read / Write) – FFFF:9800

Writing to the control register will cause the DPLL to immediately switch to the BYPASS mode if not in idle state. If the PLL_ENABLE signal is set, it will begin its sequence to enter the locked mode. This avoids being able to change the multiple or divide values without re-entering the DPLL lock sequence. The DPLL control register bit positions are given below. Note that the register bit positions are after the big-to-little-endian conversion in the Rhea interface. Externally the position of Bits 15-8, Bits 7-0 must be swapped and the internal Rhea interface converts it to the below format.

Bit	**Name**	**Function**	**Acc.**	**Reset**	**Reset C035**
0	LOCK	PLL mode: 0 = bypass 1 = locked	R	0	
1	BREAKLN	Lock status: 0 = broken lock 1 = when lock restored or write to ctrl register occurs.	R	1	
3:2	BYPASS_DIV	Clock out frequency in BYPASS mode: 00 = clkref 01 = clkref/2 1x = clkref/4	R/W	01	
4	PLL_ENABLE	PLL enable: 0 = disable (switchback bypass mode) 1 = enable *Requests the DPLL to enter the lock mode. It will enter the lock mode only after it has synthesized the desired frequency.*	R/W	0	
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 clockout is not synthesized by the PLL but simply divided down version of clkref.*	R/W	00	
11:7	PLL_MULT	DPLL multiply value: range 0 to 31	R/W	10000	0000
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	0	
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 will be active low. 1 = switch to bypass mode and start a new locking sequence if the DPLL core ever indicates that is lost the lock	R/W	1	
14	IAI	DPLL initialize after idle: 0 = try to attempt to lock 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	0	
15	*Reserved*		R	0	

19. CLKM REGISTERS - FFFF:FD00

19.1 CLKM registers mapping

All these registers are controlled directly by the internal RHEA bus. The Clock generator is selected when CS(2:0) is equal to 5.

Important: from Calypso C035 – F751619: CNTL_ARM_DIV reg is read-only register (always 001 which is the current reset value)

Register	Address	Access	Reset value
CNTL_ARM_CLK	FFFF:FD00	13bits R/W	1 0000 1000 0001
CNTL_CLK	FFFF:FD02	8bits R/W	0001 0001
CNTL_RST	FFFF:FD04	4bits R/W	011?
Unused	*FFFF:FD06*	-	-
CNTL_ARM_DIV	FFFF:FD08	3bits R/W	001

Table 26: CLKM registers

Note: *In addition to these control registers, the clock configuration is set through the DPLL module, which includes its own RHEA control register. The DPLL is selected when CS (4:0) is equal to 19 (decimal). See the "DPLL SPECIFICATIONS ver 2.2" for a detailed DPLL control bit's definition.*

19.2 Clock bit-switching schematic

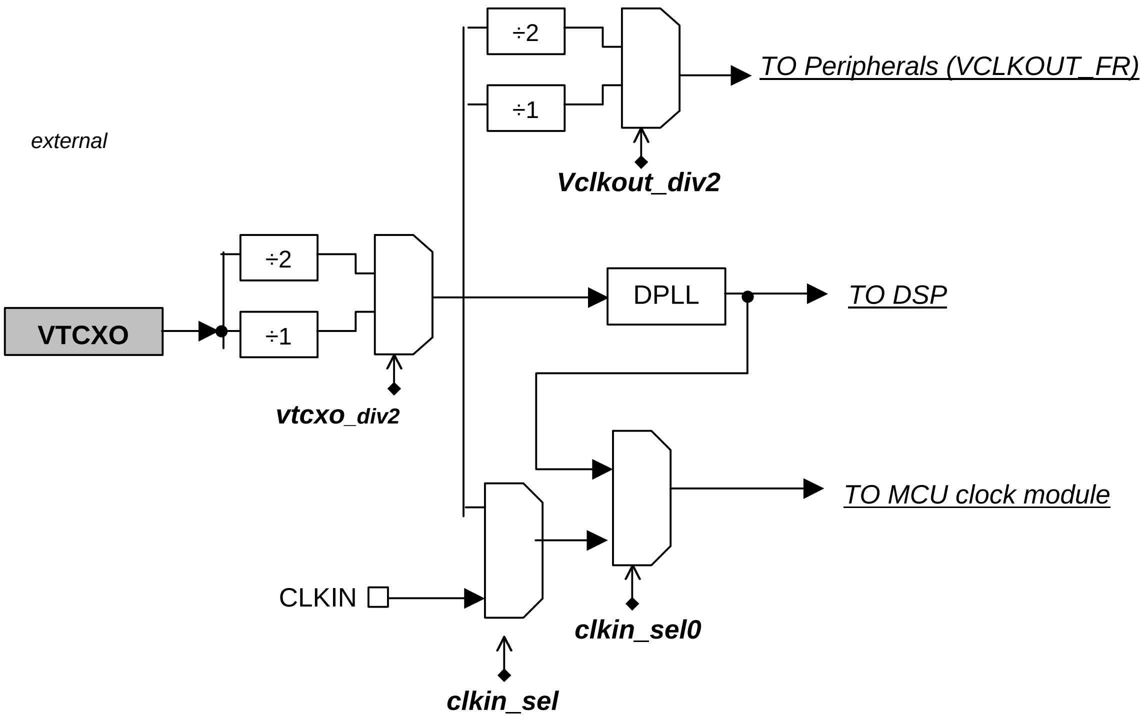


Figure 7: Clock schematic

19.3 Control MCU clock (CNTL_ARM_CLK) (Read / Write) - FFFF:FD00

<table>
<tr><th>Bit</th><th>Name</th><th colspan="3">Function</th><th>Reset</th></tr>
<tr><td>0</td><td>BIG_SLEEP</td><td colspan="3">MCU master clock (MCLK):
0 = Disable
1 = Enable
ARM can set this bit to 0 in order to cut off its clock. Only IRQ_SET can set it back to 1.</td><td>1</td></tr>
<tr><td>1</td><td>CLKIN_SEL0</td><td colspan="3">Source clock for MCU system:
0 = DPLL output clock
1 = VTCXO or CLKIN (*see CLKIN_SEL*)</td><td>0</td></tr>
<tr><td>2</td><td>CLKIN_SEL</td><td colspan="3">Clock-in selection:
0 = VTCXO source (typ. 13 MHz)
1 = CLKIN input (external up-to 40 MHz)</td><td>0</td></tr>
<tr><td>3</td><td>ARM_MCLK_XP5</td><td colspan="3">Enable the 1.5 or 2.5 division factor (2/3 duty cycle: _|¯¯|_|¯¯|_|¯) for generating the ARM system clock (from either CLKIN, VTCXO or DPLL_CLKOUT)
: division factor is 1,2,3,4,5,6 or 7, defined by the ARM_MCLK_DIV(6:4) vector (see below)
: /1.5 or /2.5 ratio is enabled and defined by the ARM_MCLK_DIV bit 5 :
ARM_MCLK_DIV(5) = 0 => div fact is 1.5
ARM_MCLK_DIV(5) = 1 => div fact is 2.5</td><td>0</td></tr>
<tr><td rowspan="4">6:4</td><td rowspan="4">ARM_MCLK_DIV</td><td colspan="3">Define the division factor applied to clock source (CLKIN, VTCXO or DPLL_CLKOUT)
to generate the ARM system clock.</td><td rowspan="4">000</td></tr>
<tr><td rowspan="2">**6:4**</td><td colspan="2">**ARM_MCLK_XP5**</td></tr>
<tr><td>**0**</td><td>**1**</td></tr>
<tr><td>000
001
010
011
100
101
110
111</td><td>/1
/1
/2
/3
/4
/5
/6
/7</td><td>/1.5
/1.5
/2.5
/2.5
/1.5
/1.5
/2.5
/2.5</td></tr>
<tr><td>7</td><td></td><td colspan="3">*Reserved*</td><td>1</td></tr>
<tr><td>11:8</td><td>DEEP_POWER</td><td colspan="3">tw delay (RESPWR high to ARM_CLK restart)</td><td>0000</td></tr>
<tr><td>12</td><td>DEEP_SLEEP</td><td colspan="3">Deep Sleep mode state</td><td>1</td></tr>
</table>

19.4 Control source clock (CNTL_CLK) (Read / Write) - FFFF:FD02

Bit	Name	Function	Reset
0	IRQ_CLK_DIS	IRQ clock control: 1 = *IRQ_CLK* is disabled and 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
1	BRIDGE_CLK_DIS	Bridge clock control: 1 = *BRIDGE_CLK* is disabled and enabled according to the sleep command (*ARM_MCLK_EN* bit) 0 = *BRIDGE_CLK* is always running	0
2	TIMER_CLK_DIS	Timer clock control: 1 = *TIMER_CLK* is disabled and 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
3	DPLL_DIS	DPLL control: 0 = DPLL **is not stopped** and continue providing an output clock (according to its control register content) when both DSP and ARM system are in IDLE3 and SLEEP mode respectively. 1 = DPLL is set in IDLE mode when both DSP and ARM system are in IDLE3 and SLEEP mode respectively. DPLL restart automatically (according to its control register content) as soon as one processor wake-up. See DPLL specification for a detailed description of the DPLL idle/wake-up sequences.	0
4	CLKOUT_EN	CLKOUT clock control: 0 = Disable 1 = Enable CLKOUT (2:0) output clocks.	1
5	EN_IDLE3_FLG	DSP idle flag control (to API): 0 = SAM / HOM wait-state register unchanged. 1 = SAM / HOM wait-state register force to HOM when DSP is in IDLE3.	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 clock used by the peripheral when using an external VTCXO at 26MHz instead of 13MHz.*	0
7	VTCXO_DIV2	VTCXO divider: 0 = VTCXO is divided by 1. 1 = VTCXO is divided by 2. *This bit is used to divide by 2 The VTCXO input clock before to supply the DPLL, the peripheral clock generator (see additional control above) and the ARM clock(when selected).* ***This bit is typically used to switch from a 13Mh to a 26Mhz VTCXO application without changing the DPLL programming model*** *.*	0

19.5 Reset control register (CNTL_RST) (Read / Write) - FFFF:FD04

Bit	Name	Function	Reset
0	*Reserved*	-	
1	DSP_RESET	DSP reset command: 0 = None 1 = Reset DSP	1
2	EXT_RESET	External peripherals reset command: 0 = nReset out is high 1 = nReset out is low	1
3	WATCHDOG_RESET	Watchdog reset: 0 = None 1 = Occured	0

19.6 MCU divider register (CNTL_ARM_DIV) (Read / Write) - FFFF:FD08

Important: from Calypso C035 – F751619: CNTL_ARM_DIV reg is read-only register (always 001 which is the current reset value)

Control directly the division factor between *ARM_MCLK* and *BRIDGDE_CLK*.

Bit	Name	Function	Reset
2:0	DIV_FACTOR	Defines the ARM_MCLK Division factor. 000 = /1 100 = /4 001 = /1 101 = /5 010 = /2 110 = /6 011 = /3 111 = /7	001

20. TIMER REGISTERS - FFFE:3800 / FFFE:6800

20.1 *Timer1/2 register mapping*

register	Timer1 address	Timer2 address	access	reset value
CNTL_TIMER	FFFE:3800	FFFE:6800	8bits R/W	0000 0000
LOAD_TIM	FFFE:3802	FFFE:6802	16bits R/W	???? ???? ???? ????
READ_TIM	FFFE:3804	FFFE:6804	16bits R	???? ???? ???? ????

Table 27: TIMER registers

20.2 *Control Timer1 (CNTL_TIMER1) (Read / Write) - FFFE:3800*

Bit	Name	Function	Reset
0	ST*	Start Timer1: 1 = Start timer1 0 = Stop timer1	0
1	AR	Reload Timer1: 1 = Auto-reload timer1 0 = One shot timer1	0
4:2	PTV	Pre-scale clock Timer Value	0
5	CLOCK_ENABLE	External Timer clock enable	0
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	0
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 will halt immediately, either retaining or discarding the current state 1 = the peripheral will stop after completion of the current task	0
15:8	*Unused*		

(*): In case of One shot mode selected (AR=0), the ST bit is automatically reset by internal logic when timer is equal to 0

20.3 *Load timer1 (LOAD_TIM1) (Read / Write) - FFFE:3802*

Bit	Name	Function	Reset
15:0	LOAD_TIM1	This value is loaded when timer1 passes through 0 or when it starts	Undef.

20.4 *Read timer1 (READ_TIM1) (Read) - FFFE:3804*

Bit	Name	Function	Reset
15:0	VALUE_TIM1	Value of TIMER1.	Undef.

20.5 Control Timer2 (CNTL_TIMER2) (Read / Write) - FFFE:6800

Bit	Name	Function	Reset
0	ST*	Start Timer2: 1 = Start timer2 0 = Stop timer2	0
1	AR	Reload Timer2: 1 = Auto-reload timer2 0 = One shot timer2	0
4:2	PTV	Pre-scale clock Timer2 Value	0
5	CLOCK_ENABLE	External Timer2 clock enable	0
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	0
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 will halt immediately, either retaining or discarding the current state 1 = the peripheral will stop after completion of the current task	0
15:8	*Unused*		

(*): In case of One shot mode selected (AR=0), the ST bit is automatically reset by internal logic when timer is equal to 0

20.6 Load timer2 (LOAD_TIM2) (Read / Write) - FFFE:6802

Bit	Name	Function	Reset
15:0	LOAD_TIM2	This value is loaded when timer2 passes through 0 or when it starts	Undef.

20.7 Read timer2 (READ_TIM2) (Read) - FFFE:6804

Bit	Name	Function	Reset
15:0	VALUE_TIM2	Value of TIMER2	Undef.

21. WATCHDOG TIMER REGISTERS - FFFF:F800

21.1 *Watchdog registers mapping*

Register	Address	Access	HW Reset value
WATCHDOG_CNTL_TIM	FFFF:F800	6bits R/W	???? 0000 0??? ??1?
WATCHDOG_LOAD_TIM	FFFF:F802	16bits W	1111 1111 1111 1101
WATCHDOG_READ_TIM	FFFF:F802	16bits R	1111 1111 1111 1101
WATCHDOG_TIM_MODE	FFFF:F804	9bits W	1??? ???? ???? ????

Table 28: Watchdog registers

21.2 *Control (WATCHDOG_CNTL_TIM) (Read / Write) - FFFF:F800*

Bit	Name	Function	Reset
6:0	*Reserved*		
7	ST*	Timer start: 1 = Start timer 0 = Stop timer	0
8	AR	Timer reload: 1 = Auto-reload timer 0 = One shot timer	0
11:9	PTV	Pre-scale clock Timer Value	0
15:12	*Reserved*	-	

(*): In case of One shot mode selected (AR=0), the ST bit is automatically reset by internal logic when timer is equal to 0

21.3 *Load timer (WATCHDOG_LOAD_TIM)* (Write) - FFFF:F802

Bit	Name	Function	Reset
15:0	LOAD_TIM	*General purpose timer :* This value is loaded when timer passes through 0 or when it starts *watchdog timer :* Reload TIMER with this value.	FFFF

21.4 *Read timer (WATCHDOG_READ_TIM)* (Read) - FFFF:F802

Bit	Name	Function	Reset
15:0	VALUE_TIM	Value of TIMER	FFFF

21.5 *Timer mode (WATCHDOG_TIM_MODE)* - FFFF:F804

Bit	Name	Function	Reset
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, ARM core is reset (thanks to rst_cmd output)	
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.	1

22. SPI REGISTERS - FFFE:3000

22.1 SPI register mapping

8 registers are mapped in the I/O port space of the MCU :

register	address	access	reset value
REG_SET1	FFFE:3000	6bits R/W	1111 1111 ??11 0000
REG_SET2	FFFE:3002	15bits R/W	1000 0000 0000 0000
REG_CTRL	FFFE:3004	10bits R/W	1111 1100 0000 0000
REG_STATUS	FFFE:3006	2bits R	1111 1111 1111 1100
REG_TX_LSB	FFFE:3008	16bits R/W	0000 0000 0000 0000
REG_TX_MSB	FFFE:300A	16bits R/W	0000 0000 0000 0000
REG_RX_LSB	FFFE:300C	16bits R	0000 0000 0000 0000
REG_RX_MSB	FFFE:300E	16bits R	0000 0000 0000 0000

Table 29: SPI registers

REG_SET1	setup serial port register.
REG_SET2	setup serial port register.
REG_CTRL	control serial port register.
REG_STATUS	status register.
REG_TX_LSB	transmit register (lower bits).
REG_TX_MSB	transmit register (upper bits).
REG_RX_LSB	receive register (lower bits).
REG_RX_MSB	receive register (upper bits).

The serial port offers input and output registers for, respectively, the loading of the data to serialize (TRANSMIT) or the reading of the data paralleled (RECEIVE).

22.2 Set up SPI 1 (REG_SET1) (Read / Write) - FFFE:3000

REG_SPI_SET1 is dedicated to the configuration of the serial port.

Bit	Name	Function	Reset
0	EN_CLK	Clock enable 0 = clock is shut off 1 = clock is running	0
3:1	PTV	Pre-scale clock divisor 000 = 1 001 = 2 010 = 4 011 = 8 100 = 16	000
4	MSK0	Enable interrupt for Write cycle 0 = interrupt active 1 = interrupt disable	1
5	MSK1	Enable interrupt for Read/Write cycle 0 = interrupt active 1 = interrupt disable	1
15:6	*unused*		

22.3 Set up SPI 2 (REG_SET2) (Read / Write) - FFFE:3002

REG_SPI_SET2 is dedicated to the configuration of the serial port.

Bit	Name	Function	Reset
4:0	C	Active edge of the clock for each device in TX 0 = Falling 1 = Rising	0
9:5	P	Format of enable signals nTSPEN 0 = negative level 1 = positive level	0
14:10	L	Format of enable signals nTSPEN 0 = level trigger 1 = Edge trigger	0
15	*unused*		

22.4 Control SPI (REG_CTRL) (Read / Write) - FFFE:3004

REG_SPI_CTRL2 is dedicated to the activation of the serial port and starts the operation of the interface as soon as one of its 2 bits is set. It defines:

- a WRITE activation of the serial port (transmit only)
- a 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 5)

Bit	Name	Function	Reset
0	RD	Read and write process activation (toggle at 1)	0
1	WR	Write process activation (toggle at 1)	0
6:2	NB	transmission length of NB+1 bits. 00000 = one bit transmit .... 11111 = 32 bits transmit	0
9:7	AD	addressed device index (5 devices maximum)	0
15:10	*unused*		

22.5 Status register (REG_STATUS) (Read) - FFFE:3006

Bit	Name	Function	Reset
0	RE	Read End 1 = Received registers loaded	0
1	WE	Write End 1 = the serialization is finish	0
15:2	*unused*		

To read the status register or to write in the setup register 2, the internal clock must be running (reg_set1 (0) = 1).

CAUTION : at reset value 0 does NOT mean that transfer is on-going.

22.6 Transmit registers (REG_TX_LSB/MSB) (Read / Write) - FFFE:3008

The data to transmit are loaded in 2 word registers (REG_TX_MSB, REG_TX_LSB).
These registers are accessible by the Rhea bus in read or write.

Bit	Name	Function	Reset
15:0	REG_TX_LSB	Data to transmit (word low)	0
15:0	REG_TX_MSB	Data to transmit (word high)	0

This choice of implementation implies that, whatever its size, the word cough to be aligned on the msb side.

22.7 Receive registers (REG_RX_ LSB/MSB) (Read) - FFFE:300C

The received data are accessible from the Rhea bus through two registers of 16 bits (REG_RX_MSB and REG_RX_LSB).

Bit	Name	Function	Reset
15:0	REG_RX_LSB	Receive data (word low)	0
15:0	REG_RX_MSB	Receive data (word high)	0

23. UWIRE REGISTERS - FFFE:4000

23.1 *UWIRE register mapping*

6 registers are mapped in the I/O port space of the MCU :

register	address	access	reset value
TDR	FFFE:4000	16bits W	???? ???? ???? ????
RDR	FFFE:4000	16bits R	???? ???? ???? ????
CSR	FFFE:4002	16bits R/W	0000 ???? ???? ????
SR1	FFFE:4004	12bits R/W	???0 ???? ?0??
SR2	FFFE:4006	12bits R/W	???0 ???? ?0??
SR3	FFFE:4008	3bits R/W	000

Table 30: UWIRE registers

TDR	Transmit data Register.
RDR	Receive data Register.
CSR	Control & Status Register.
SR1	Setup Register 1.
SR2	Setup Register 2.
SR3	Setup register 3.

23.2 *Transmit Data Register (TDR)* (Write) - FFFE:4000

Bit	Name	Function	Reset
15:0	TD	Data to transmit.	Undef

Msb (bit 15) is the first transmitted bit.

Whatever its size, the word must be aligned on the msb side.

23.3 *Receive Data Register (RDR)* (Read) - FFFE:4000

Bit	Name	Function	Reset
15:0	RD	Received data	Undef

Lsb (bit 0) is the last received bit.

Whatever its size, the word is aligned on the lsb side.

23.4 *Control & Status Register (CSR)* (Write/Read) - FFFE:4002

Bit	Name	Access	Function	Reset
4:0	NB_BITS_RD	R/W	Number of bits to receive	*Undef*
9:5	NB_BITS_WR	R/W	Number of bits to transmit	*Undef*
11:10	INDEX	R/W	Index of the external device 00 : CS0 10 : CS2 01 : CS1 11 : CS3	*Undef*
12	CS_CMD	R/W	Chip select: 1 = Set the Chip Select of the selected device to its active level	0
13	START	R/W	Start process: 1 = Start a WRITE or / and 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 register) 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 register	0
14	CSRB	R	0 = the Control&Status register (CSR) is ready to receive new data. After writing in the Control & Status register, either the CS_CMD or the START bits, this bit is set to 1. When the corresponding action has been done, CSRB is reset.	0
15	RDRB	R	1 = receive register (RDR) is full. When the controller read the content of the RDR register, this bit is cleared.	0

23.5 Setup Register 1 (SR1) (Read / Write) - FFFE:4004

Content of this register must not be changed when a READ or WRITE process is running.
Setup the serial interface for the first and the second external components.

Bit	Name	Function	Reset
0	CS0_EDGE_RD	Active edge of the serial clock SCLK used to read when CS0 is selected: 1 = falling 0 = rising *(Input data are strobed on this edge)*	*undef*
1	CS0_EDGE_WR	Active edge of the serial clock SCLK used to write when CS0 is selected 1 = falling 0 = rising *(Output data are generated on this edge)*	*undef*
2	CS0CS_LVL	Active level of the chip select CS0	0
4:3	CS0_FRQ	SCLK clock Frequency when the CS0 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).*	*undef*
5	CS0_CHK	Before activating a WRITE process, checks if external device is ready. 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 done and the WRITE process is immediately executed. Used when CS0 is selected	*undef*
6	CS1_EDGE_RD	Idem CS0_EDGE_RD when CS1 is selected	*undef*
7	CS1_EDGE_WR	Idem CS0_EDGE_WR when CS1 is selected	*undef*
8	CS1CS_LVL	Defines the active level of the CS1 chip select	0
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).*	*undef*
11	CS1_CHK	Idem CS0_CHK. Used when the CS1 is selected	*undef*

23.6 Setup Register 2 (SR2) (Read / Write) - FFFE:4006

Content of this register must not be changed when a READ or WRITE process is running.
Setup the serial interface of the third and the fourth external component.

Bit	**Name**	**Function**	**Reset**
0	CS2_EDGE_RD	Idem CS0_EDGE_RD when CS2 is selected	*undef*
1	CS2_EDGE_WR	Idem CS0_EDGE_WR when CS2 is selected	*undef*
2	CS2CS_LVL	Defines the active level of the CS2 chip select	*0*
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).*	*undef*
5	CS2_CHK	Idem CS0_CHK. Used when CS2 is selected	*undef*
6	CS3_EDGE_RD	Idem CS0_EDGE_RD when CS3 is selected	*undef*
7	CS3_EDGE_WR	Idem CS0_EDGE_WR when CS3 is selected	*undef*
8	CS3CS_LVL	Defines the active level of theCS3 chip select	0
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).*	*undef*
11	CS3_CHK	Idem CS0_CHK. Used when CS3 is selected	*undef*

23.7 Setup Register 3 (SR3) (Read / Write) - FFFE:4008

Content of this register must not be changed when a READ or WRITE process is running.
Setup the serial interface for the internal clock.

Bit	**Name**	**Function**	**Reset**
0	CLK_EN	Switch off the clock if 0. Switch on the clock if 1.	0
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	00

24. ARMIO REGISTERS - FFFE:4800

24.1 ARMIO register mapping

Nota : reflects GPIO or KBD external input values.

register	address	access	reset value
ARMIO_LATCH_IN	FFFE:4800	16bits R	iiii iiii iiii iiii
ARMIO_LATCH_OUT	FFFE:4802	16bits R/W	0000 0000 0000 0000
IO_CNTL_REG	FFFE:4804	16bits R/W	1111 1111 1111 1111
ARMIO_CNTL_REG	FFFE:4806	4bits R/W	1111 1111 1101 0011
ARMIO_LOAD_TIM	FFFE:4808	16bits R/W	???? ???? ???? ????
KBR_LATCH_REG	FFFE:480A	5bits R	1111 1111 111i iiii
KBC_REG	FFFE:480C	5bits R/W	???? ???? ???1 1111
BUZZER_LIGHT_REG	FFFE:480E	2bits R/W	???? ???? ???? ??00
LIGHT_LEVEL_REG	FFFE:4810	6bits R/W	???? ???? ??11 1111
BUZZER_LEVEL_REG	FFFE:4812	6bits R/W	???? ???? ??11 1111
GPIO event mode	FFFE:4814		???? ???? ??00 0000
Keyboard/GPIO irq register	FFFE:4816		???? ???? ???? ??11
Keyboard/GPIO mask irq	FFFE:4818		???? ???? ???? ??00
GPIO debouncing reg.	FFFE:481A		???? ???? ???? 0000
GPIO_LATCH register	FFFE:481C		???? ???? ???? ????

Table 31: ARMIO registers

ARMIO_LATCH_IN	general purpose input register.
ARMIO_LATCH_OUT	general purpose output register.
ARMIO_CNTL_REG	IO control register.
ARMIO_LOAD_TIM	Timer control register
IO_CNTL_REG	control for genral purpose io
KBR_LATCH_REG	keyboard row inputs control register.
KBC_REG	keyboard column outputs control register.
BUZZER_LIGHT_REG	light &buzzer outputs control register.
LIGHT_LEVEL_REG	light level control register.
BUZZER_LEVEL_REG	buzzer control register.

24.2 Input register ARMIO_LATCH_IN (Read) - FFFE:4800

Bit	Name	Function	Reset
15:0	INPUT_LATCH	General purpose inputs	Input pins

24.3 Output register (ARMIO_LATCH_OUT) (Read / Write) - FFFE:4802

Bit	Name	Function	Reset
15:0	OUTPUT_REG	General purpose ouputs	0000

24.4 Input/output control (IO_CNTL_REG) (Read / Write) - FFFE:4804

Bit	Name	Function	Reset
15:0	IO_CNTL	inout control for general purpose io 1 = Input 0 = Output	FFFF

24.5 *Control ARMIO (ARMIO_CNTL_REG)* (Read / Write) - FFFE:4806

Bit	Name	Function	Reset
1:0	*Unused*		11
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 regerdless of the SOFT bit	0
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 will halt immediately, either retaining or discarding the current state 1 = the peripheral will stop after completion of the current task	0
4	*Unused*		1
5	CLOCK_ENABLE	ARMIO MODULE clock enable	0
15:6	*Unused*		1

24.6 *Load timer (ARMIO_LOAD_TIM)* (Read / Write) - FFFE:4808

Bit	Name	Function	Reset
15:0	LOAD_TIM	This value is loaded when timer passes through 0 or when it starts	Undef

24.7 *Keyboard row inputs (KBR_LATCH_REG)* (Read) - FFFE:480A

Bit	Name	Function	Reset
4:0	KBR_LATCH	Keyboard row inputs	input pins
15:5	*Unused*		

24.8 *Keyboard column outputs (KBC_REG)* (Read / Write) - FFFE:480C

Bit	Name	Function	Reset
4:0	KBC_REG	Keyboard columns outputs	1F
15:5	*Unused*		

24.9 *Buzzer & light ctrl (BUZZ_LIGHT_REG)* (Read / Write) - FFFE:480E

Bit	Name	Function	Reset
0	BUZZER	Buzzer on/off: 0 = Stop 1 = Start	0
1	LIGHT	Light on/off: 0 = Stop 1 = Start	0
15:2	*Unused*		

24.10 Light power level (LIGHT_LEVEL_REG) (Read / Write) - FFFE:4810

Bit	Name	Function	Reset
5:0	LIGHT_LEVEL_REG	Value for light power level 000000 = level 0 (no light) 000001 = level 1 111111 = level 63	111111
15:6	*Unused*		

24.11 Buzzer power level (BUZZ_LEVEL_REG) (Read / Write)- FFFE:4812

Bit	Name	Function	Reset
5:0	BUZZER_LEVEL_REG	Value for buzzer power level 000000 = level 0 (no sound) 000001 = level 1 111111 = level 63	111111
15:6	*Unused*		

24.12 GPIO mode (GPIO_EVENT_MODE_REG) (Read/Write) - FFFE:4814

Bit	Name	Function	Reset
0	SET_GPIO_EVENT_ MODE	GPIO event mode 0 = disable 1 = enable	0
1:4	PIN_SELECT	Select ARMIO_IN[15:0] pin to generate interrupt 0000 = pin 0 1111 = pin 15	0
5	EDGE_SELECT	Set interrupt on falling/rising edge 0 = Falling edge 1 = Rising Edge	0
6:15	unused	-	-

24.13 Keyboard/GPIO Irq Register (KBD_GPIO_INT) (Read) - FFFE:4816

Bit	Name	Function	Reset
0	KBD_INT	Keyboard interrupt (active low)	1
1	GPIO_INT	GPIO interrupt (active low)	1
2:15	*unused*	-	-

GPIO_INT is reset on read access to KBD_GPIO_INT register
KBD_INT is a status bit only (duplication of the level of the corresponding interrupts signal)

24.14 Keyboard/GPIO mask irq (KBD_GPIO_MASKIT) (R/W) - FFFE:4818

Bit	Name	Function	Reset
0	MASKIT_KBD	Mask keyborad interrupt: 0 = un-mask 1 = mask	0
1	MASKIT_GPIO	Mask GPIO interrupt: 0 = un-mask 1 = mask	0
2:15	*unused*	-	-

24.15 GPIO debouncing (GPIO_DEBOUNCING_REG) (R/W) - FFFE:481A

Bit	Name	Function	Reset
0:3	GPIO_DEBOUNCING _REG	Debouncing time(step: 500 μs): 0000 = 500 μs 1111 = 8 ms	0
4:15	*unused*		

24.16 GPIO Latch (GPIO_LATCH_REG) (Read) - FFFE:481C

Bit	Name	Function	Reset
0:15	GPIO_LATCH_REG	After debouncing time, ARMIO_IN bus is latched in this register.	U

24.17 Keyboard interface

Keyboard input rows are read on the armio interrupt generated when one of the line is at low level. The interrupt to the processor is then a AND of the five input rows filtering during an armio clock period.

Reading the input rows consists of latching the input pins of the keyboard rows in “kbr_latch” when the Chip Select (CS) is equal to ARMIO_CS .

24.17.1 Keyboard connection

The keyboard is connected to the chip using:

- KBR(4:0) input pins for row lines.
- KBC(4:0) output pins for column lines

If a key button of the keyboard matrix is pressed, the corresponding row and column lines are shorted together.

To allow a key press detection, all input pins (KBR) are pulled up to VCC and all output pins (KBC) are driving a low level. Any action on a button will generate an interrupt to the Ucontroller which will, as answer, scan the column lines with the sequence describe below.

This sequence is written to allow detection of simultaneous press actions on several key buttons.

	IDLE	**KEYBOARD SCAN**						**IDLE**
KBC(0)	0	1	0	1	1	1	1	0
KBC(1)	0	1	1	0	1	1	1	0
KBC(2)	0	1	1	1	0	1	1	0
KBC(3)	0	1	1	1	1	0	1	0
KBC(4)	0	1	1	1	1	1	0	0

24.18 Pulse Width Modulation (PWM)

In order to control power level of light and buzzer, a basic pulse width modulation is implemented within the ARMIO module.

6 levels are available for both light and buzzer. 2 memory-mapped registers are used to define those power levels:

Light_level and buzzer_level registers are 6 bit wide, and allow to program up to 64 levels.

An internal signal from the internal timer can also control the frequency of the buzzer, it is functionally ANDed with the audio level sown below.

As an example, here are the levels needed for 6 audio levels:
log[(10 : 6) * 1] = 0.22 => 0.22 * 63 = level 14
log[(10 : 6) * 2] = 0.52 => 0.52 * 63 = level 33
log[(10 : 6) * 3] = 0.70 => 0.70 * 63 = level 44
log[(10 : 6) * 4] = 0.82 => 0.82 * 63 = level 52
log[(10 : 6) * 5] = 0.92 => 0.92 * 63 = level 58
log[(10 : 6) * 6] = 1 => 1 * 63 = level 63

The bandwidth is 3.4 kHz. The PWM frequency should be around (10 * 3.4 Khz), and the reference clock at: (64 * 10 * 3.4 Khz) = 2 Mhz

24.18.1 Tones creation

A internal timer is used to produce tones whose frequency is constant. This is done in combination with the PWM (Pulse Width Modulation) capability. The combination with the PWM function performs the power level control while the timer makes the frequency control. When this function is enabled, the generated interrupt is used to toggle TONE_FRQ signal, which commands the PWM activation.

The tone frequency is equal to the interrupt period divided by 2.

The 8-bit timer interrupt period is defined by the value of the load register (LOAD_TIM)
The tone period is as follow:

$$Ttone = Tclk*(LOAD\_TIM+1)* 2^{(9)}$$

LOAD_TIM	Fclk=10 MHz	Fclk=20 MHz
1	102.4 us - 9.75kHz	51.2 us - 19.5 kHz
2	153.6 us - 6.5 Khz	76.8 us - 13 KHz
21	1.13ms - 887.8 Hz	0.56 ms - 1769.9 Hz
43	2.25 ms - 443.9 Hz	1.12 ms - 887.9 Hz
88	4.56 ms - 219.3 Hz	2.28 ms - 438.9 Hz
127	6.55 ms - 152 Hz	3.27 ms - 305 Hz
255	13.1 ms - 76 Hz	6.55 ms - 152 Hz

25. SIM REGISTERS - FFFE:0000

25.1 SIM register mapping

8 registers are mapped in the I/O port space of the MCU :

register	address	access	reset value
REG_SIM_CMD	FFFE:0000	5bits R/W	00 0000
REG_SIM_STAT	FFFE:0002	4bits R	1010
REG_SIM_CONF1	FFFE:0004	16bits R/W	0000 0000 0000 1100
REG_SIM_CONF2	FFFE:0006	16bits R/W	0000 1001 0100 0000
REG_SIM_IT	FFFE:0008	5bits R	0 0000
REG_SIM_DRX	FFFE:000A	9bits R	0 ???? ????
REG_SIM_DTX	FFFE:000C	8bits R/W	0000 0000
REG_SIM_MASKIT	FFFE:000E	6bits R/W	11 1111
REG_SIM_IT_CD	FFFE:0010	1bits R/W	0

Table 32: SIM registers

REG_SIM_CMD	control register.
REG_SIM_STAT	status registers.
REG_SIM_CONF1	configuration registers.
REG_SIM_CONF2	time-delay parameters.
REG_SIM_IT	interrupt status register.
REG_SIM_IT_CD	Card-detect interrupt status register.
REG_SIM_DRX	receive byte register.
REG_SIM_DTX	transmit byte register.
REG_SIM_MASKIT	interrupt mask register.

25.2 Reg_sim_cmd register (R/W) - FFFE:0000

b15:b5	b4	b3	b2	b1	b0
unused	module_clk_en	cmdstart	cmdstop	cmdifrst	cmdcardrst
-	0	0	0	0	0

Never do an OR/AND with reading REG_SIM_CMD.

Bit	Name	Function	Reset
0	CMDCARDRST	SIM card reset sequence 0 = disable 1 = enable	0
1	CMDIFRST	SIM interface software reset 1 = activation (toggle bit)	0
2	CMDSTOP	SIM card stop procedure 1 = activation (toggle bit)	0
3	CMDSTART	SIM card start procedure 0 = no effect 1 = active	0
4	MODULE_CLK_EN	Clock of the module 0 = disable 1 = enable	0
b15:b5	*unused*		-

Nota: *these command bits are sensitive on the writing of a 1 (event generation). The writing of a 0 is inactive.*

25.3 Reg_sim_stat register (R) - FFFE:0002

b15:b4	b3	b2	b1	b0
Unused	statfifoempty	statfifofull	stattxpar	statnocard
0	1	0	1	0

Bit	Name	Function	Reset
0	STATNOCARD	card presence 0 = no card 1 = card detected	0
1	STATTXPAR	parity check for transmit byte 0 = parity error 1 = parity OK	1
2	STATFIFOFULL	FIFO content 1 = FIFO full	0
3	STATFIFOEMPTY	FIFO content 1 = FIFO empty	1
b15:b4	*unused*		-

25.4 Reg_sim_conf1 register (R/W) - FFFE:0004

b15	b14:b11	b10	b9	b8
SIOlow	trig	srstlev	Svcclev	bypass
0	0	0	0	0

b7	b6	b5	b4	b3	b2	b1	b0
etu	sclklev	sclkdiv	*reserved*	sclken	txrx	conv	chkpar
0	0	0	0	1	1	0	0

Bit	Name	Function	Reset
0	CONFCHKPAR	enable parity check on reception 0 = disable 1= enable	0
1	CONFCODCONV	coding convention: (TS character) 0 = direct 1 = inverse	0
2	CONFTXRX	SIO line direction 0 = receive mode 1 = transmit mode	1
3	CONFSCLKEN	SIM clock: 0 = standby mode 1 = normal	1
4	reserved	ETU period: 0 = CONFETUPERIOD 1 = 4*1/F_{SCLK}	0
5	CONFSCLKDIV	SIM clock frequency: 0 = 13/4 MHz 1 = 13/8 MHz	0
6	CONFSCLKLEV	SIM clock idle level: 0 = low 1 = high	0
7	CONFETUPERIOD	ETU period 0 = 372*1/F_{SCLK} 1 = 512/8*1/F_{SCLK}	0
8	CONFBYPASS	bypass hardware timers and start and stop sequences	0
9	CONFSVCCLEV	logic level on SVCC *(used if CONFBYPASS = 1)*	0
10	CONFSRSTLEV	logic level on SRST *(used if CONFBYPASS = 1)*	0
14:11	CONFTRIG	FIFO trigger level: 0000 = 1 .... 1111 = 16	0000
15	CONFSIOLOW	SIO: 0 = no effect 1 = force low	0

Nota: *The bit CONFTXRX 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 doesn't define the direction of the line at the electrical level which is monitored by the state-machine of the interface.*

25.5 Reg_sim_conf2 register (R/W) - FFFE:0006

b15:b8	b7:b4	b3:b0
confwaiti	tdsim	tfsim
0000 1001	0100	0000

Bit	Name	Function	Reset
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)	00
7:4	CONFTDSIM	time delay for contact activation/deactivation time unit = 8 * T_{CKETU}	04
15:8	CONFWAITI	overflow wait time between two received character time unit = 960 *D * T_{CKETU} with D parameter = 1 or 8 (TA1 character)	09

25.6 Reg_sim_it register (R) - FFFE:0008

b15:b5	b4	b3	b2	b1	b0
unused	sim_rx	sim_tx	sim_ov	sim-wt	sim_natr
0	0	0	0	0	0

Bit	Name	Function (IRQ flags)	Reset
0	SIM_NATR	0 = on read access to REG_SIM_IT 1 = no answer to reset	0
1	SIM_WT	0 = on read access to REG_SIM_IT 1 = character underflow	0
2	SIM_OV	0 = on read access to REG_SIM_IT 1 = receive overflow	0
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	0
4	SIM_RX	0 = on read access to REG_SIM_DRX 1 = waiting characters to be read	0
15:5	*unused*	-	0

25.7 Reg_sim_drx register (R) - FFFE:000A

b15:b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
unused	statrxpar	msb							lsb
0	?	?	?	?	?	?	?	?	?

Bit	Name	Function	Reset
7:0	SIM_DRX	next data byte in FIFO available for reading	?
8	STATRXPAR	parity-check for received byte 0 = parity error 1 = no parity error	0
15:9	*unused*		-

25.8 Reg_sim_dtx register (R/W) - FFFE:000C

b15:b8	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
unused	msb							lsb
0	0	0	0	0	0	0	0	0

Bit	**Name**	**Function**	**Reset**
7:0	SIM_DTX	next data byte to be transmitted	0
15:8	*unused*	-	-

25.9 Reg_sim_maskit register (R/W) - FFFE:000E

b15:b8	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
unused	sim_cd	sim_rx	sim_tx	sim_ov	sim-wt	sim_natr
0	1	1	1	1	1	1

Bit	**Name**	**Function**	**Reset**
0	MASK_SIM_NATR	No-answer-to-reset interrupt: 0 = unmask 1 = mask	1
1	MASK_SIM_WT	Character wait-time overflow interrupt: 0 = unmask 1 = mask	1
2	MASK_SIM_OV	Receive overflow interrupt: 0 = unmask 1 = mask	1
3	MASK_SIM_TX	Waiting character to transmit interrupt: 0 = unmask 1 = mask	1
4	MASK_SIM_RX	Waiting characters to be read interrupt: 0 = unmask 1 = mask	1
5	MASK_SIM_CD	SIM card insertion/extraction interrupt: 0 = unmask 1 = mask	1
15:6	*unused*	-	-

***Nota**: If mask is set to 1 the interrupt is disabled. However, the corresponding status bit remains sensitive.*

25.10 Reg_sim_it_cd register (R) - FFFE:0010

b15:b2	**b0**
unused	it_cd
0	0

Bit	**Name**	**Function (IRQ flags)**	**Reset**
0	IT_CD	0 = on read access to REG_SIM_IT_CD 1 = SIM card insertion/extraction	0
15:1	*unused*	-	-

26. TSP REGISTERS - FFFE:0800

26.1 Parallel bit interface – 0x06 / 0x07

The parallel bit interface directly monitors 14 output signals TSPACTi with i E [0-13]. It can control independently the activation and deactivation of each output signal with a quarter of GSM bit time accuracy (923 Ns).

The interface is based on 2 eight-bit registers **REG_TSP_ACT_L** and **REG_TSP_ACT_U** loaded by the TPU with a MOVE instruction.
The 2 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 is composed of 5 bits defining 32 potential addressable registers.
The mapping of the Parallel Port registers is given below.

Register name	TPU address (MOVE instruction)	Access	Reset
REG_TSP_ACT_L	0x06	8 bits	0
REG_TSP_ACT_U	0x07	8 bits	0

26.2 Reg_tsp_act_l register – 0x06

Bit	Name	Description	Value at reset
7:0	REG_TSP_ACT_L	Activation TSPACT 7:0 signals	h00

26.3 Reg_tsp_act_u register – 0x07

Bit	Name	Description	Value at reset
4:0	REG_TSP_ACT_U	Activation TSPACT 13:8 signals	h00
7:5	Unused		

This interface 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 GSM CODEC of TEXAS INSTRUMENTS.

26.4 *TPU sequencer internal address mapping – 0x00 / 0x11*

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.

The mapping of the TSP registers is given below.

Register name	TPU address	access	Value at reset
REG_TSP_CTRL1	00	8bits W	???? ???? 0000 0000
REG_TSP_CTRL2	01	2bits W	???? ???? ???? ??00
REG_TX_1	04	8bits W	???? ???? 0000 0000
REG_TX_2	03	8bits W	???? ???? 0000 0000
REG_TX_3	02	8bits W	???? ???? 0000 0000
REG_TX_4	05	8bits W	???? ???? 0000 0000
REG_TSP_SET1	09	8bits W	???? ???? ?000 ?000
REG_TSP_SET2	0A	8bits W	???? ???? ?000 ?000
REG_TSP_SET3	0B	3bits W	???? ???? ???? ?000
REG_GAUGING_ENABLE	11	1bits W	???? ???? ???? ???0

Table 33: TSP registers

Nota : *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.*

Important: from Calypso C035 – F751619: TSP reg follow nota below (not above):

Nota : *TSP registers with the exception of REG_TSP_SET1, REG_TSP_SET2, REG_TSP_SET3 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.*

26.5 *TSP register mapping*

Both receive and transmit registers are mapped in the MCU address space. Transmit registers access is limited to debug purposes as these registers are controlled by the TPU only. All these registers are accessible thru Rhea interface.

Register name	MCU address	Access	Value at reset
REG_RX_LSB	FFFE:0800 – (000)	16 R/W	0
REG_RX_MSB	FFFE:0802 – (001)	16 R/W	0
REG_TX_LSB	FFFE:080C – (110)	16 R	0
REG_TX_MSB	FFFE:080A – (101)	16 R	0

Nota: 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 LSB are significant

REG_RX_LSB receive register (lower bits).
REG_RX_MSB receive register (upper bits).
REG_TX_LSB transmit register (lower bits).
REG_TX_MSB transmit register (upper bits).

The serial port offers input and output registers for, respectively, the loading of the data to serialize (TRANSMIT) or the reading of the data paralleled (RECEIVE).

The characteristics of the serial port are related to configuration bits stored in 2 control registers.

Updating the Read-Write control register sets off the operation of the serial port.

26.6 Transmit registers (reg_tx_1/2/3/4) - 0x02..0x05

As the sizes of the registers in the external devices are variable and in the range 6 to 23, the serial port implements 4 byte registers (REG_TX_1, REG_TX_2; REG_TX_3; REG_TX_4),

Each of them 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 could require one, two or three MOVE instruction(s) if the register size of the external device is lower or equal to one, two, three or four byte(s).

Bit	Name	Address	Description	Reset
7:0	REG_TX_1	04	1ST byte to transmit	0
7:0	REG_TX_2	03	2nd byte to transmit	0
7:0	REG_TX_3	02	3rd byte to transmit	0
7:0	REG_TX_4	05	4th byte to transmit	0

This choice of implementation implies that, whatever its size, the word ought to be aligned on the msb side.

26.7 Receive registers (reg_rx_lsb/msb)- FFFE:0800 .. 0802

The received data are accessible from the MCU through two registers of 16 bits (REG_RX_MSB and REG_RX_LSB) but not from the TPU which implements only a WRITE access to the port.(TPU is unidirectional => WRITE only).

Moreover, for debug purposes, the transmit registers are mapped in the address space of the MCU. However, no synchronization is insured in hardware between the TPU and the MCU.

26.7.1 REG_RX_LSB

Bit	Name	Address	Description	Reset
15:0	REG_RX_LSB	FFFE:0800	16 lower bits or received data	0

26.7.2 REG_RX_MSB

Bit	Name	Address	Description	Reset
15:0	REG_RX_MSB	FFFE:0802	16 upper bits or received data	0

If less than 16 bits are transferred, only the REG_RX_LSB bits are significant.

26.8 TSP setup registers (reg_tsp_set1/2/3): 0x09 – 0x0B

REG_TSP_SET1/2/3 are dedicated to the configuration of the signal waveforms of the serial port. It defines for each device:

- The active edge clock (1 bit)
- The format of the enable signals (2 bits)

26.8.1 REG_TSP_SET1 – 0x09

Bit	Name	Description	Reset
0	C	Active edge clock of device 0 0 = Falling 1 = Rising	0
1	P	Format of enable signal of device 0 0 = Negative level 1 = Positive level	0
2	L	Format of enable signal of device 0 0 = Level trigger 1 = Edge trigger	0
3	*Unused*		
4	C	Active edge clock of device 1 0 = Falling 1 = Rising	0
5	P	Format of enable signal of device 1 0 = Negative level 1 = Positive level	0
6	L	Format of enable signal of device 1 0 = Level trigger 1 = Edge trigger	0
7	*Unused*	-	-

26.8.2 REG_TSP_SET2 – 0x0A

Bit	Name	Description	Reset
0	C	Active edge clock of device 2 0 = Falling 1 = Rising	0
1	P	Format of enable signal of device 2 0 = Negative level 1 = Positive level	0
2	L	Format of enable signal of device 2 0 = Level trigger 1 = Edge trigger	0
3	Unused		
4	C	Active edge clock of device 3 0 = Falling 1 = Rising	0
5	P	Format of enable signal of device 3 0 = Negative level 1 = Positive level	0
6	L	Format of enable signal of device 3 0 = Level trigger 1 = Edge trigger	0
7	Unused		

26.8.3 REG_TSP_SET3 – 0x0B

Bit	Name	Description	Reset
0	C	Active edge clock of device 4 0 = Falling 1 = Rising	0
1	P	Format of enable signal of device 4 0 = Negative level 1 = Positive level	0
2	L	Format of enable signal of device 4 0 = Level trigger 1 = Edge trigger	0
7:3	Unused		

26.9 Control registers (reg_tsp_ctrl1/2): 0x00 - 0x01

Two registers REG_TSP_CTRL1 and REG_TSP_CTRL2 control the operation of the interface.

26.9.1 REG_TSP_CTRL1 - 0x00

REG_TSP_CTRL1 is dedicated to the configuration of the serial port. It defines:

- Number of bits to transfer (in the range 1 to 32)
- External device address (between 5)

This register must be stable during a transfer and has to be loaded before the update of REG_TSP_CTRL2.

Bit	Name	Description	Reset
4:0	NB	Word size (1 to 32 bits) 00000 -> one bit to transmit or receive 11111 -> 32 bits to transmit or receive	0
7:5	AD	Index of the addressed device (0 to 5)	0

26.9.2 REG_TSP_CTRL2 - 0x01

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 2 bits is set. It defines:

- a WRITE activation of the serial port (transmit only)
- a READ activation of the serial port (simultaneous receive and transmit)

Bit	Name	Description	Reset
0	RD	Read activation process	0
1	WR	Write activation process	0

***Note**: RD and WR bits are automatically reset when the serial interface start the corresponding action. No new requests can be executing until the end of the running one.*

26.10 TSP gauging_enable signal (reg_gauging_enable) - 0x11

The GAUGING_ENABLE signal is necessary to gauge the 32Khz oscillator who 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 the deactivation of the GAUGING_ENABLE signal.

Bit	Name	Description	Reset
1	GAUGING_ENABLE	Activation gauging_enable signal	0

27. TPU REGISTERS - FFFF:1000

27.1 TPU register mapping

Both receive and transmit registers are mapped in the MCU address space.

Register name	Address (4:0)	Access	Reset value
REG_TPU_OFFSET	FFFF:100C (00110)	13bits R	0 0000 0000 0000
REG_TPU_SYNCHRO	FFFF:100E (00111)	13bits R	0 0000 0000 0000
REG_TPU_CTRL	FFFF:1000 (00000)	12bits R/W	0000 10?0 ?001
REG_INT_CTRL	FFFF:1002 (00001)	4bits W	0000
REG_INT_STAT	FFFF:1004 (00010)	2bits R	11
REG_IT_DSP_PG	FFFF:1020 (10000)	1bit W	1

Table 34: TPU registers

REG_TPU_CTRL	control & status register.
REG_INT_CTRL	interrupt control register.
REG_INT_STAT	interrupt status register.
REG_TPU_OFFSET	offset operand value register.
REG_TPU_SYNCHRO	synchro operand value register.

27.2 TPU RAM Memory mapping – FFFF:9000

The two pages of the TPU Communication Buffer are addressable by the MCU through the RHEA interface.

***Note**: The MCU see only one page. The page selection management is handled by the TPU.*

Memory	Address (10:0)
PAGE 0 or 1 512*16 bits	1000000000 1111111111

27.3 Control & status register (reg_tpu_ctrl) (Read / Write) - FFFF:1000

The control and status register, TPU_CTRL, gives the possibility to the MCU to:

- reset the TPU module
- check the TPU activity
- reset the TSP module
- control the TPU communication buffer

Bit	Name	Description	Reset	TPU reset
0	TPU_RESET	Reset TPU module 0 = no effect 1 = reset *(excepted GSM timebase)*	1	
1	TPU_PAGE	Page of TPU buffer: 0 = page0 1 = page1 *(visible by TPU)*	0	0
2	TPU_EN	Execution new scenario loaded in the TPU buffer at page TPU_page: 0 = none (*by TPU*). 1 = enable (*by MCU)*	0	0
3	*Unused*	-	-	-
4	DSP_EN*	IT_DSP generation on next it_frame. 0 = Clear when IT_DSP occurs 1 = enable (*by MCU*)	0	0
5	*Unused*	-	-	-
6	MCU_RAM_ACCESS*	RAM read access: 0 = not allowed to MCU 1 = allowed to MCU	0	
7	TSP_RESET	Reset of TSP module: 0 = no effect 1 = reset	1	
8	TPU_IDLE	TPU-scenario execution status 0 = none 1 = on-going	0	0
9	TPU_WAIT	WAIT *or* AT state of TPU 1 = active	0	0
10	TPU_CK_ENABLE	Clock of the TPU module: 0 = disable 1 = enable	0	
11	FULL WRITE	Enable MCU to write anywhere in the RAM even if the TPU is executing a scenario WARNING: TO HANDLE WITH CARE because there is no protection preventing the MCU from writing on the address read by the TPU	0	

FULL_WRITE vs MCU_RAM_ACCESS logic table:

MCU_RAM_ACCESS	FULL_WRITE	MODE
0	0	Write page access mode
0	1	Full write mode
1	X	Full read/write mode

27.4 Interrupt status register (reg_int_stat) (Read) - FFFF:1004

The interrupt status register informs the MCU of the origin of the interrupt:

- new frame occurrence (IT_FRAME)
- execution of a new scenario (IT_PAGE).

Bit	Name	Description	Reset
0	ITF	Frame interrupt occurrence 0 = new frame occurrence	1
1	ITP	Page interrupt occurrence 0 = execution new scenarios	1

27.5 Interrupt control register (reg_int_ctrl) (Write) - FFFF:1002

The interrupt control register gives the possibility to the MCU to:

- mask the frame interrupt (IT_FRAME) generation
- mask the page interrupt (IT_PAGE) generation
- mask the DSP interrupt (IT_DSP) generation

Bit	Name	Description	Reset	TPU reset
0	ITF_M	Frame interrupt for MCU 0 = unmask 1 = mask	0	0
1	ITP_M	Page interrupt 0 = unmask 1 = mask	0	0
2	ITD_M	Frame interrupt for DSP 0 = unmask 1 = mask	0	0
3	ITD_F	Force frame interrupt for DSP 0 = no effect 1 = force	0	0

27.6 DSP interrupt occurrence reg. (reg_it_dsp_pg) (Write) - FFFF:1020

The interrupt control register gives the possibility to the TPU to:

- generate a DSP interrupt (IT_DSP_PG) generation by executing a move instruction to this register

Bit	Name	Description	Active value	Reset
0	IT_DSP_PG	DSP programmable interrupt occurrence	0	1

27.7 Offset register (reg_tpu_offset) (Read) - FFFF:100C

Bit	Name	Description	Reset
12:0	TPU_OFFSET	Value of OFFSET operand	0000
15:13	*Unused*	-	-

***Note**: the content of the register is not latched so its value can be corrupted if MCU read access is simultaneous with a TPU write operation.*

27.8 Synchro register (reg_tpu_synchro) (Read) - FFFF:100E

Bit	Name	Description	Reset
12:0	TPU_SYNCHRO	Value of SYNCHRO operand	0000
15:13	Unused		-

***Note**: the content of the register is not latched so its value can be corrupted if MCU read access is simultaneous with a TPU write operation.*

27.9 TPU-SEQUENCER

27.9.1 Functional description

The sequencer is a micro-programmable machine, which executes an auto-scheduled program code.

The instruction frequency adopted is the one of the quarter of GSM bit: F_{inst} =13/12MHz The corresponding time-period (923.1ns) will be the time accuracy for the execution of a command.

27.9.2 Instruction execution flow

The sequencer implements an instruction cycle based on 4 phases.
For each instruction, the execution flow is:

1| first step:	instruction read (**FETCH**)
2| second step:	instruction store (**STORE**)
3| third step:	instruction decoding (**DECODE**)
4| fourth step:	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 the result of the execution phase of the previous instruction.

The cycle frequency will be the one of the sixteenth of bit: F_{cyc}=13/3MHz.

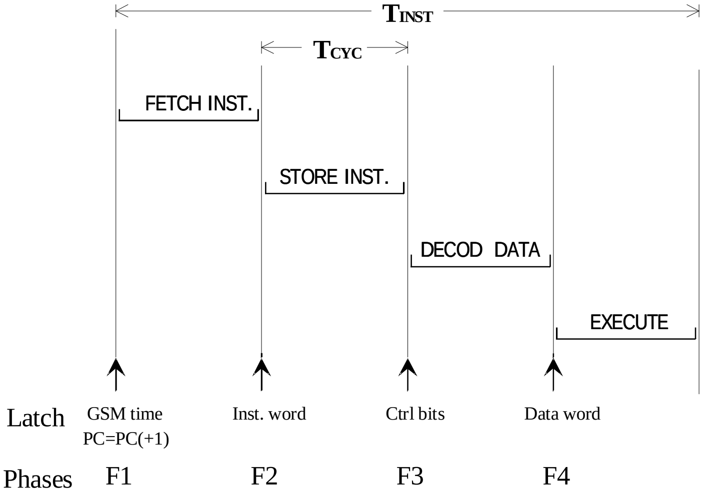


Figure 8: TPU sequencer instruction flow

27.9.3 Micro instructions set definition

The set of micro-instructions has been limited to the essential instructions in order to minimize the complexity of the decoder.

27.9.4 Structure of the micro-instruction

The micro-instruction is coded on 16 bits to be consistent with the word format manipulated by the MCU and the DSP.
It is split in several fields with a format specific to each category of instructions.

Instruction for time scheduling

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Instruction Code			Absolute time (Qbit GSM step)												

Instruction for Data transfer

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Instruction Code			Data Operand								Destination Address				

Figure 9: TPU Instruction format

27.10 TPU Instruction Set

27.10.1 Micro instructions 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**

27.10.1.1 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 unit.
The dynamic range is of 1 frame (0 to 5000 qbit) with a quarter of bit time-step.

0	**0**	**1**	τ12	τ11	τ10	τ9	τ8	τ7	τ6	τ5	τ4	τ3	τ2	τ1	τ0
Instruction Code			time versus OFFSET												

27.10.1.2 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 unit.
The dynamic range is of 1 frame (0 to 5000 qbit) with a quarter of bit time-step.

0	**1**	**0**	τ12	τ11	τ10	τ9	τ8	τ7	τ6	τ5	τ4	τ3	τ2	τ1	τ0
Instruction Code			TPU OFFSET versus reference time-base												

27.10.1.3 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 in quarter of GSM bit unit.
The dynamic range is of 1 frame (0 to 5000 qbit) with a quarter of bit time-step.

0	**1**	**1**	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0

Instruction Code IT-FRAME trigger versus reference time-base

27.10.1.4 WAIT instruction

The WAIT instruction allows introducing a waiting-period between the executions of 2 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 unit.
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.

1	**0**	**1**	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0

Instruction Code time to wait

27.10.1.5 SLEEP instruction

The SLEEP instruction allows stopping the sequencer by disabling the bit TPU-ENABLE of the control register.
To restart the TPU, the MCU has to set the enable bit by writing in the control register TPU_CTRL_REG.

0	**0**	**0**													

Instruction Code unused

27.10.2 Micro instruction for data processing

The instruction for data processing allows to:

- Write a word (max 8 bits) to a register **MOVE**

27.10.2.1 MOVE instruction

The MOVE instruction 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.

1	**0**	**0**	D7	D6	D5	D4	D3	D2	D1	D0	A4	A3	A2	A1	A0

Instruction Code Data Operand Register Address

The MOVE instruction can initiate several actions:

- a trigger event (toggle bit principle)
  bit di=1 => event activated
  bit di=0 => no event

Note:

- *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 bi-directional serial port.*

28. RTC REGISTERS - FFFE:1800

28.1 RTC register mapping

register	address	access	reset value
SECOND_REG	FFFE:1800 (00)	8bits R/W	0000 0000
MINUTES_REG	FFFE:1801 (01)	8bits R/W	0000 0000
HOURS_REG	FFFE:1802 (02)	8bits R/W	0000 0000
DAYS_REG	FFFE:1803 (03)	8bits R/W	0000 1111
MONTHS_REG	FFFE:1804 (04)	8bits R/W	0000 1111
YEARS_REG	FFFE:1805 (05)	8bits R/W	0000 0000
WEEK_REG	FFFE:1806 (06)	4bits R/W	0000
reserved	FFFE:1807 (07)	-	-
ALARM_SECOND_REG	FFFE:1808 (08)	8bits R/W	0000 0000
ALARM_MINUTES_REG	FFFE:1809 (09)	8bits R/W	0000 0000
ALARM_HOURS_REG	FFFE:180A (0A)	8bits R/W	0000 0000
ALARM_DAYS_REG	FFFE:180B (0B)	8bits R/W	0000 1111
ALARM_MONTH_REG	FFFE:180C (0C)	8bits R/W	0000 1111
ALARM_YEARS_REG	FFFE:180D (0D)	8bits R/W	0000 0000
reserved	FFFE:180E (0E)	-	-
reserved	FFFE:180F (0F)	-	-
RTC_CTRL_REG	FFFE:1810 (10)	7bits R/W	000 0000
RTC_STATUS_REG	FFFE:1811 (11)	7bits R/W	100 0000
RTC_INT_REG	FFFE:1812 (12)	4bits R/W	0000
RTC_COMP_LSB_REG	FFFE:1813 (13)	8bits R/W	0000 0000
RTC_COMP_MSB_REG	FFFE:1814 (14)	8bits R/W	0000 0000
RTC_RES_PROG_REG	FFFE:1815 (15)	6bits R/W	10 0111

Table 35: RTC registers

28.2 Time and Calendar registers (TC) - FFFE:1800 .. 1806

28.2.1 SECONDS_REG (Read / Write) - FFFE:1800

Bit	Name	Function	Reset
3:0	SEC0	1st digit of seconds: Range is 0 to 9	0
7:4	SEC1	2nd digit of seconds: Range is 0 to 5	0

28.2.2 MINUTES_REG (Read / Write) - FFFE:1801

Bit	Name	Function	Reset
3:0	MIN0	1st digit of minutes: Range is 0 to 9	0
7:4	MIN1	2nd digit of minutes: Range is 0 to 5	0

28.2.3 HOURS_REG (Read / Write) - FFFE:1802

Bit	Name	Function	Reset
3:0	HOUR0	1st digit of hours: Range is 0 to 9	0
6:4	HOUR1	2nd digit of hours: Range is 0 to 2	0
7	PM_nAM	Only used in PM_AM mode (otherwise 0) 0 = AM 1 = PM	0

28.2.4 DAYS_REG (Read / Write) - FFFE:1803

Bit	Name	Function	Reset
3:0	DAY0	1st digit of days: Range from 0 to 9	1
7:4	DAY1	2nd digit of days: Range from 0 to 3	0

28.2.5 MONTHS_REG (Read / Write) - FFFE:1804

Bit	Name	Function	Reset
3:0	MONTH0	1st digit of months: Range from 0 to 9	1
7:4	MONTH1	2nd digit of months: Range from 0 to 1	0

Note: *Usual notation is taken for month value:*

01 = January
02 = February
...
12 = December

28.2.6 YEARS_REG (Read / Write) - FFFE:1805

Bit	Name	Function	Reset
3:0	YEAR0	1st digit of Years: Range from 0 to 9	0
7:4	YEAR1	2nd digit of Years: Range from 0 to 9	0

28.2.7 WEEKS_REG (Read / Write) - FFFE:1806

Bit	Name	Function	Reset
3:0	WEEK	1st digit of Days in a week: Range from 0 to 6	0

28.3 TC alarm registers - FFFE:1808 .. 180D

28.3.1 ALARM_SECONDS_REG (Read / Write) - FFFE:1808

Bit	Name	Function	Reset
3:0	ALARM_SEC0	1st digit of seconds: Range is 0 to 9	0
7:4	ALARM_SEC1	2nd digit of seconds: Range is 0 to 5	0

28.3.2 ALARM_MINUTES_REG (Read / Write) - FFFE:1809

Bit	Name	Function	Reset
3:0	ALARM_MIN0	1st digit of minutes: Range is 0 to 9	0
7:4	ALARM_MIN1	2nd digit of minutes: Range is 0 to 5	0

28.3.3 ALARM_HOURS_REG (Read / Write) - FFFE:180A

Bit	Name	Function	Reset
3:0	ALARM_HOUR0	1st digit of hours: Range is 0 to 9	0
6:4	ALARM_HOUR1	2nd digit of hours: Range is 0 to 2	0
7	ALARM_PM_nAM	Only used in PM_AM mode (otherwise 0) 0 = AM 1 = PM	0

28.3.4 ALARM_DAYS_REG (Read / Write) - FFFE:180B

Bit	Name	Function	Reset
3:0	ALARM_DAY0	1st digit for days: Range from 0 to 9	1
7:4	ALARM_DAY1	2nd digit for days: Range from 0 to 3	0

28.3.5 ALARM_MONTHS_REG (Read / Write) - FFFE:180C

Bit	Name	Function	Reset
3:0	ALARM_MONTH0	1st digit of months: Range from 0 to 9	1
7:4	ALARM_MONTH1	2nd digit of months: Range from 0 to 1	0

28.3.6 ALARM_YEARS_REG (Read / Write) -: FFFE:180D

Bit	Name	Function	Reset
3:0	ALARM_YEAR0	1st digit of Years: Range from 0 to 9	0
7:4	ALARM_YEAR1	2nd digit of Years: Range from 0 to 9	0

28.4 General Registers - FFFE:1810 .. 1812

28.4.1 RTC_CTRL_REG (Read / Write) -: FFFE:1810

Bit	Name	Function	Reset
0	STOP_RTC	RTC control: 0 = freeze 1 = run	0
1	ROUND_30S	Update: 0 = No update 1 = When a one is written, the time is rounded to the closest minute (*See note*)	0
2	AUTO_COMP	Auto compensation: 0 = disable 1 = enable	0
3	MODE_12_24	Hour mode: 0 = 24 hours mode (*See note*) 1 = 12 hours mode (PM-AM mode)	0
4	TEST_MODE	Functional / Test mode: 0 = functional mode 1 = test mode (Auto compensation is enable when the 32 KHz counter reachs at its end)	0
5	SET_32_COUNTER	32K counter: 0 = No action 1 = set the 32 KHz counter with comp_reg value (See note)	0
6	nDELTA_OMEGA	Analog baseband type: 0 = ABB circuit is Iota 1 = ABB circuit is Nausica	0

<u>Note:</u>

- *SET_32_counter must only be used when the RTC is frozen.*
- *ROUND_30S bit is a toggle bit, ARM can only write '1', RTC clears it. If the ARM set the ROUND_30S bit and then read it, the ARM will read '1' until the rounded to the closet minute is perform at the next second.*
- *MODE_12_24 : it is possible to switch between the two mode at any time without disturbed the RTC, read or write are always performed with the current mode.*

28.4.2 RTC_INTERRUPTS_REG (Read / Write) - FFFE:1812

Bit	Name	Function	Reset
1:0	EVERY	Periodic interrupt frequency: 0 = second 1 = minute 2 = hour 3 = day	0
2	IT_TIMER	Periodic interrupt 0 = disabled 1 = enabled	0
3	IT_ALARM	Enable one interrupt when the alarm value is reached (*TC ALARM registers*) by the TC registers	0

<u>Note:</u> *The ARM should respect the BUSY period to prevent spurious interrupt*.

28.4.3 RTC_STATUS_REG - FFFE:1811

Bit	Name	Function	Acc.	Reset
0	BUSY	Event update: 0 = updating event in more than 15 µs 1 = updating event	R	0
1	RUN	RTC status: 0 = frozen 1 = running	R	0
2	1S_EVENT	One second has occurred	R	0
3	1M_EVENT	One minute has occurred	R	0
4	1H_EVENT	One hour has occurred	R	0
5	1D_EVENT	One day has occurred	R	0
6	ALARM	Alarm interrupt 0 = none 1 = has been generated	R/W	0
7	POWER_UP	Reset status: 0 = none 1 = occurred	R/W	1

Note:

- The alarm interrupt keeps its low level, until the ARM writes ‘1’ in the ALARM bit of the RTC_STATUS_REG register.
- The timer interrupt is a low-level pulse (15 µs duration).
- RUN bit show the real state of the RTC, indeed because of STOP_RTC signal was re-synchronized on 32Khz clock, the action of this bit is delayed.
- POWER_UP is set by a reset, is cleared by writing ‘1’ in this bit.

28.5 Compensation Registers - FFFE:1813 .. 1814

28.5.1 RTC_COMP_MSB_REG (Read / Write) - FFFE:1814

Bit	Name	Function	Reset
7:0	RTC_COMP_MSB	Nb of periods to be added in the counter every hour	0

28.5.2 RTC_COMP_LSB_REG (Read / Write) - FFFE:1813

Bit	Name	Function	Reset
7:0	RTC_COMP_LSB	Nb of periods to be added in the counter every hour	0

Note: *This register must be written in 2-complement, that means:*

- *To add one 32 Khz oscillator period every hour, ARM needs to write ‘FFFF’ into RTC_COMP_MSB_REG & RTC_COMP_LSB_REG.*
- *To remove one 32 Khz oscillator period every hour, ARM needs to write ‘0001’ into RTC_COMP_MSB_REG & RTC_COMP_LSB_REG.*
- *The ‘7FFF’ value is forbidden.*

28.6 RTC_RES_PROG_REG (Read / Write) – FFFE:1815

This register is used to set a current supply by switching resistors. This current supply is used to start and to keep quartz oscillations.

Bit	Description	Reset C05	Reset C035
2:0	Control analog switches from R1 to R3	111	011
5:3	Control analog switches from R2 to FB	100	101
7:6	*unused*	-	

28.6.1 Current resistance value for Calypso C05

Crystal registor value	Programmed resistor value	Oscillator current
0 x 2 F	0K	
0 x 2 7	57 K	Default
0 X 2 E	80 K	
0 X 2 6	137 K	
0 X 2 C	160 K	
0 X 2 4	217 K	
0 X 2 A	240 K	
0 X 2 2	297 K	
0 X 2 8	320 K	
0 X 2 0	377 K	

Table 36: **Current resistance value versus register contents for C05**

29. ULPD REGISTERS - FFFE:2000

29.1 ULPD register mapping

registers	address	access	reset value
SETUP_RF_REG	FFFE:2024 (12)	8 bits R/W	???? ???? 0000 0000
SETUP_FRAME_REG	FFFE:2022 (11)	5 bits R/W	???? ???? ???0 0000
SETUP_VTCXO_REG	FFFE:2020 (10)	8 bits R/W	???? 1111 1111 1111
SETUP_SLICER_REG	FFFE:201E (0F)	12 bits R/W	???? 1111 1111 1111
SETUP_CLK13_REG	FFFE:201C (0E)	6 bits R/W	???? ???? ??11 1111
GSM_TIMER_IT_REG	FFFE:201A (0D)	1 bit R	???? ???? ???? ???0
GSM_TIMER_VALUE_REG	FFFE:2018 (0C)	16 bits R	0000 0000 0000 0001
GSM_TIMER_INIT_REG	FFFE:2016 (0B)	16 bits R/W	0000 0000 0000 0000
GSM_TIMER_CTRL_REG	FFFE:2014 (0A)	2 bits R/W	???? ???? ???? ??10
GAUGING_STATUS_REG	FFFE:2012 (09)	3 bits R	???? ???? ???? ?000
GAUGING_CTRL_REG	FFFE:2010 (08)	3 bits R/W	???? ???? ???? ?000
COUNTER_HI_FREQ_MSB_REG	FFFE:200E (07)	6 bits R	???? ???? ??00 0000
COUNTER_HI_FREQ_LSB_REG	FFFE:200C (06)	16 bits R	0000 0000 0000 0001
COUNTER_32_MSB_REG	FFFE:200A (05)	4 bits R	???? ???? ???? 0000
COUNTER_32_LSB_REG	FFFE:2008 (04)	16 bits R	0000 0000 0000 0001
SIXTEENTH_STOP_REG	FFFE:2006 (03)	15 bits R	?000 0000 0000 0000
SIXTEENTH_START_REG	FFFE:2004 (02)	15 bits R	?000 0000 0000 0000
INC_SIXTEENTH_REG	FFFE:2002 (01)	12 bits R/W	???? 0000 0000 0000
INC_FRAC_REG	FFFE:2000 (00)	16 bits R/W	0000 0000 0000 0000

Table 37: ULPD registers

29.2 GSM TIMER registers - FFFE:2014 .. 201A

29.2.1 GSM_TIMER_INIT_REG (Read / Write) - FFFE:2016

Bit	Name	Function	R/W	Reset
15:0	timer_init	load value of the timer	R/W	0

29.2.2 GSM_TIMER_VALUE_REG (Read) - FFFE:2018

Bit	Name	Function	R/W	Reset
15:0	timer_value	current value of the timer	R	1

29.2.3 GSM_TIMER_CTRL_REG (Read / Write) - FFFE:2014

Bit	Name	Function	R/W	Reset
0	load	load the timer with timer_init value (toggle bit)	R/W	0
1	freeze	GSM timer activity: 1 = frozen 0 = running	R/W	1

Note: *The bit load is cleared after loading timer_init value.*
The bit freeze is set when the timer reached to zero.

29.2.4 GSM_TIMER_IT_REG (Read) - FFFE:201A

Bit	Name	Function	R/W	Reset
0	it_gsm_timer_event	GSM timer interrupt: 0 = none 1 = pending	R	0

Note: *The bit it_gsm_timer_event is cleared on reading this register.*

29.3 SETUP registers - FFFE:201C .. 2024

29.3.1 SETUP_RF_REG (Read / Write) - FFFE:2024

Bit	Name	Function	R/W	Reset
7:0	setup_rf	Number of 32 khz clock periods to enable the RF from the beginning of the wake up phase	R/W	0

Note: *The RF is enabled (setup_rf + 1) period at 32 khz, after the gsm timer equals setup_frame.*

29.3.2 SETUP_FRAME_REG (Read / Write) - FFFE:2022

Bit	Name	Function	R/W	Reset
4:0	setup_frame	number of frames to begin the wake up phase (*in TDMA frame unit*)	R/W	0

29.3.3 SETUP_VTCXO_REG (Read / Write) - FFFE:2020

Bit	Name	Function	R/W	Reset
11:0	setup_vtcxo	Number of 32 khz clock periods to enable the vtcxo from the beginning of the wake up phase	R/W	FFF

Note: *The vtcxo is enabled (setup_vtcxo + 1) period at 32 khz, after gsm timer equal setup_frame*.

29.3.4 SETUP_SLICER_REG (Read / Write) - FFFE:201E

Bit	Name	Function	R/W	Reset
11:0	setup_slicer	Number of 32 khz clock periods to enable the slicer from vtcxo enable	R/W	FFF

Note: *The slicer is enabled (setup_slicer + 1) period at 32 kHz, after vtcxo goes active*.

29.3.5 SETUP_CLOCK_13MHZ_REG (Read / Write) - FFFE:201C

Bit	Name	Function	R/W	Reset
5:0	setup_clk13	Number of 32 khz clock periods to enable the 13 Mhz clock from slicer enable	R/W	3F

Note: *The 13 MHz clock is enabled (setup_clk13 + 1) period at 32 kHz, after slicer goes active.*

29.4 GAUGING registers - FFFE:2004 .. 2012

29.4.1 GAUGING_CTRL_REG (Read / Write) - FFFE:2010

Bit	Name	Function	R/W	Reset
0	gauging_en	Gauging activity: 0 = stoped 1 = running	R/W	0
1	gauging_type	Gauging versus: 0 = gsm network time 1 = high frequency clock	R/W	0
2	select_hi_freq_clock	High frquency clock: 0 = 13 Mhz clock 1 = DSP PLL clock	R/W	0
3-15	-	Reserved	R/W	-

Note:

- *The gauging is really finished after it_gauging occurrrence.*
- *After gauging_en is cleared, if gauging_en is set before it_gauging happens, the current gauging may not be stopped.*
- *If gauging_en is set and cleared in a short time (less than 2 periods at 32 Khz), the it_gauging may not be generated*

29.4.2 GAUGING_STATUS_REG (Read) - FFFE:2012

Bit	Name	Function	R/W	Reset
0	It_gauging	Gauging interrupt: 0 = none 1 = pending	R	0
1	overflow_hi_freq	High-frequency-counter overflow error: 0 = none 1 = overflow occurred (*during gauging vs high frequency clock*)	R	0
2	overflow_32	32 kHz-counter overflow error: 0 = none 1 = overflow occurred	R	0

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.*

29.4.3 COUNTER_HI_FREQ_MSB_REG (Read) - FFFE:200E

Bit	Name	Function	R/W	Reset
5:0	counter_hi_freq_msb	Upper value of the number of high frequency clock during gauging time	R	0

29.4.4 COUNTER_HI_FREQ_LSB_REG (Read) - FFFE:200C

Bit	Name	Function	R/W	Reset
15:0	counter_hi_freq_lsb	Lower value of the number of high frequency clock during gauging time	R	1

29.4.5 COUNTER_32_MSB_REG (Read) - FFFE:200A

Bit	Name	Function	R/W	Reset
3:0	counter_32_msb	Upper value of the number of clock 32 kHz during gauging time	R	0

29.4.6 COUNTER_32_LSB_REG (Read) - FFFE:2008

Bit	Name	Function	R/W	Reset
15:0	counter_32_lsb	Lower value of the number of clock 32 kHz during gauging time	R	1

29.4.7 SIXTEENTH_STOP_REG (Read) - FFFE:2006

Bit	Name	Function	R/W	Reset
14:0	sixteenth_stop	value of GSM-time-base at the stop of gauging	R	0

29.4.8 SIXTEENTH_START_REG (Read) - FFFE:2004

Bit	Name	Function	R/W	Reset
14:0	sixteenth_start	value of GSM-time-base at the start of gauging	R	0

29.5 GSM TIME BASE registers: FFFE:2000 .. 2002

29.5.1 INC_SIXTEENTH_REG (Read / Write) - FFFE:2002

Bit	Name	Function	R/W	Reset
11:0	inc_sixteenth	Integer part of the duration of the 32 kHz clock period in 1/16 GSM bit unit	R/W	0

29.5.2 INC_FRAC_REG (Read / Write) - FFFE:2000

Bit	Name	Function	R/W	Reset
15:0	inc_frac	Fractional part of the duration of the 32 kHz clock period in 1/16 GSM bit unit	R/W	0

***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 unit multiply by 65536.*

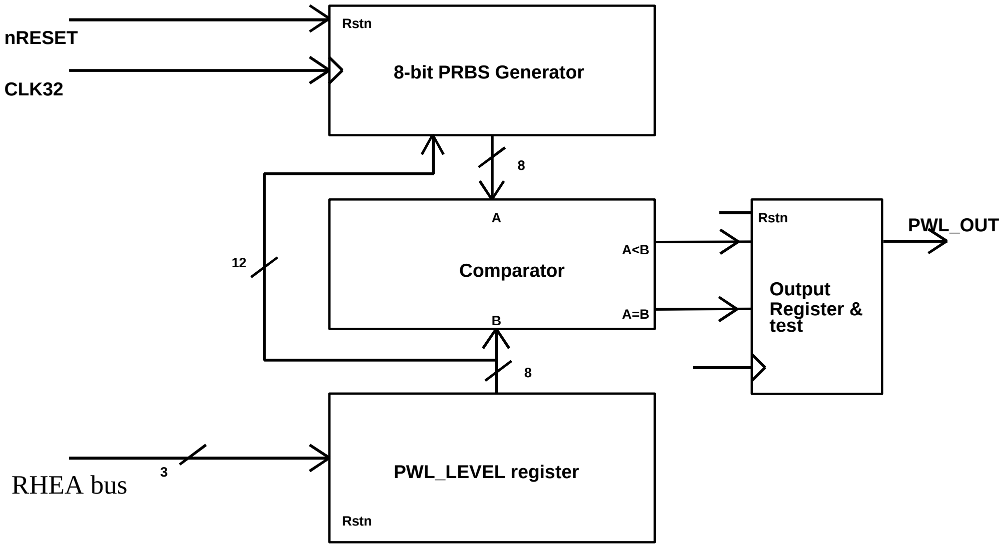
30. PWL REGISTERS - FFFE:8000

30.1 *Functionality*

This module is composed of a pseudo-random 8-bit data generator and a programmable threshold comparator.

- The pseudo-random 8-bit data generator is build using a 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 or equal than the programmable threshold.
  - ⇨ 1 if the random value is less than the programmable threshold.

Considering the random sequence is normal, it generates a sequence whose mean value is proportional to the comparator threshold.



Pic .52 PWL block diagram

30.2 *PWL Register mapping*

Register	Address	Access	reset value
PWL_LEVEL_REG	FFFE:8000 (00)	8 bits R/W	0000 0000
PWL_CTRL_REG	FFFE:8001 (01)	1 bit R/W	0

Table 38: PWL registers

30.3 *Level Register (PWL_LEVEL_REG) (Read / Write) - FFFE:8000*

Bit	Name	Function	Acc.	Reset
7:0	PWL_LEVEL	Mean value of the PWL output signal. 0 = continuous ‘0’ output. ... 255 = continuous ‘1’ output	R/W	0

30.4 *Control Register (PWL_CTRL_REG) (Read / Write) - FFFE:8001*

Bit	Name	Function	Acc.	Reset
0	CLK_ENABLE	Internal clock is enabled when ‘1’, else it is cut	R/W	0
7:1	-	Reserved	R/W	-

31. PWT REGISTERS - FFFE:8800

31.1 PWT Register mapping

Register	Address	Access	reset value
FRC_REG	FFFE:8800	6 bits R/W	00 0000
VCR_REG	FFFE:8801	7 bits R/W	000 0000
GCR_REG	FFFE:8002	1 bit R/W	0

Table 39: PWT registers

31.2 Frequency Control Register (FRC_REG) (Read / Write) - FFFE:8800

Bit	Name	Function	R/W	Reset
1:0	OCT*	octave select	R/W	0
5:2	FRQ*	frequency select (12 frequencies)	R/W	0

(*) resynchronous writing, asynchronous reading

Buzzer frequencies

FRC bits 5-2 1-0	frequency (Hz)		FRC bits 5-2 1-0	frequency (Hz)	
0000 00	5274	e^5	0000 10	1319	e^3
0001 00	4978	dis^5	0001 10	1245	dis^3
0010 00	4699	d^5	0010 10	117	d^3
0011 00	4435	cis^5	0011 10	1109	cis^3
0100 00	4186	c^5	0100 10	1047	c^3
0101 00	3951	h^4	0101 10	988	h^2
0110 00	3729	ais^4	0110 10	932	ais^2
0111 00	3520	a^4	0111 10	880	a^2
1000 00	3322	gis^4	1000 10	831	gis^2
1001 00	3136	g^4	1001 10	784	g^2
1010 00	2960	fis^4	1010 10	740	fis^2
1011 00	2794	f^4	1011 10	698	f^2
0000 01	2637	e^4	0000 11	659	e^2
0001 01	2489	dis^4	0001 11	622	dis^2
0010 01	2349	d^4	0010 11	587	d^2
0011 01	2217	cis^4	0011 11	554	cis^2
0100 01	2093	c^4	0100 11	523	c^2
0101 01	1976	h^3	0101 11	494	h^1
0110 01	1865	ais^3	0110 11	466	ais^1
0111 01	1760	a^3	0111 11	440	a^1
1000 01	1661	gis^3	1000 11	415	gis^1
1001 01	1568	g^3	1001 11	392	g^1
1010 01	1480	fis^3	1010 11	370	fis^1
1011 01	1397	f^3	1011 11	349	f^1
			11?? ??	***not allowed***	-

31.3 Volume Control Register (VRC_REG) (Read / Write) - FFFE:8801

Bit	Name	Function	R/W	Reset
0	ONOFF*	switch tone 0 = off 1 = on	R/W	0
6:1	VOL*	volume select	R/W	0

(*) resynchronous writing, asynchronous reading

Buzzer volume

VRC bits 5 - 1 ; 0	buzzer / loadspeaker
111111 1	loud
000000 1	quiet
????x 0	off

31.4 General Control Register (GCR_REG) : FFFE:8802

Bit	Name	Function	R/W	Reset
0	CLK_EN	PWT clock: 0 = disable 1 = enable	R/W	0
1	TESTIN	divider 1/154: 0 = on 1 = off	R/W	0

(*) asynchronous writing and reading

32. LPG REGISTERS - FFFE:7800

32.1 *LPG Register mapping*

Register	Address	access	reset value
LCR_REG	FFFE:7800	8 bits R/W	0000 0000
PM_REG	FFFE:7801	1 bits R/W	0

Table 40: LPG registers

32.2 *LPG control register (LCR_REG) : FFE:7800*

Bit	Name	Function	R/W	Reset
2:0	PERCTRL*	LED blink frequency	R/W	0
5:3	ONCTRL*	time LED is on parameter	R/W	0
6	LPGRES*	LPG counter reset, active low	R/W	0
7	PERM_ON*	set high to force permanent light on	R/W	0

(*) Asynchronous writing and reading

With the LCR bits 2-0 the blinking period of the LED is determined.

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

With the LCR bits 5-3 the on time of the LED is determined.

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
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

32.3 *Power management register (PM_REG) (Read / Write) - FFFE:7801*

Bit	Name	Function	R/W	Reset
0	CLK_EN	Functional clock: 0 = disable 1 = enable	R/W	0

(*) Asynchronous writing and reading

32.3.1 Design constraint

Clearing the LCR bit 6 ('0') resets the whole PWM circuit (but not the control register) and switches off the LED.

It's possible to switch on the LED independently from the PWM circuit by setting the LCR bit 7 ('1' = permanent light).

The reset PORN is low active and resets 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 as asynchronous, some meta-stability problems can happen.
Consequently, the LED output could, in the worst case, be switched on at maximum intensity during one additional blink period

33. UART 16C750 REGISTERS - FFFF:5000/6000

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.

33.1 UART registers mapping

The following registers are accessible by the MCU. The chip-select is defined by ARM_UART_CS.

Address IRDA	Address MCU MODEM	Address DSP MODEM	Register	Access
	FFFF:6000		UIR	2 bits R/W
FFFF:5000	FFFF:5800	DATA:8000	RHR	8bits R
			THR	8 bits W
			DLL	8 bits R/W
FFFF:5001	FFFF:5801	DATA:8001	IER	8 bits R/W
			DLH	8 bits R/W
FFFF:5002	FFFF:5802	DATA:8002	IIR	8 bits R
			FCR	8 bits W
			EFR	8 bits R/W
FFFF:5003	FFFF:5803	DATA:8003	LCR	8 bits R/W
FFFF:5004	FFFF:5804	DATA:8004	MCR	6 bits R/W
			XON1	8 bits R/W
			ADDR1	8 bits R/W
FFFF:5005	FFFF:5805	DATA:8005	LSR	8 bits R
			XON2	8 bits R/W
			ADDR2	8 bits R/W
FFFF:5006	FFFF:5806	DATA:8006	MSR	4 bits R
			TCR	8 bits R/W
			XOFF1	8 bits R/W
FFFF:5007	FFFF:5807	DATA:8007	SPR	8 bits R/W
			TLR	8 bits R/W
			XOFF2	8 bits R/W
FFFF:5008	FFFF:5808	DATA:8008	MDR1	6 bits R/W
FFFF:5009			MDR2	2 bits R/W
FFFF:500A			SFLSR	4 bits R
			TXFLL	8 bits W
FFFF:500B			RESUME	8 bits R
			TXFLH	5 bits W
FFFF:500C			SFREGL	8 bits R
			RXFLL	8 bits R/W
FFFF:500D			SFREGH	4 bits R
			RXFLH	4 bits R/W
FFFF:500E			BLR	8 bits R/W
	FFFF:580E	DATA:800E	UASR	8 bits R
FFFF:500F			ACREG	5 bits R/W
			DIV1.6	8 bits R/W
FFFF:5010	FFFF:5810	DATA:8010	SCR	6 bits R/W
FFFF:5011	FFFF:5811	DATA:8011	SSR	2 bits R
FFFF:5012			EBLR	8 bits W

Table 41: UART/Modem registers

33.2 UART/IrDA registers mapping

The following registers are accessible by the MCU. The chip-select is defined by ARM_UART_CS.

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[c]	IER[c]	DLH	DLH	DLH	DLH
0x02	IIR	FCR[b]	IIR	FCR[b]	EFR	EFR
0x03	LCR	LCR	LCR	LCR	LCR	LCR
0x04	MCR[b]	MCR[b]	MCR[b]	MCR[b]	XON1/ADDR1	XON1/ADDR1
0x05	LSR	-	LSR	-	XON2/ADDR2	XON2/ADDR2
0x06	MSR/TCR[a]	TCR[a]	MSR/TCR[a]	TCR[a]	XOFF1/TCR[a]	XOFF1/TCR[a]
0x07	SPR/TLR[a]	SPR/TLR[a]	SPR/TLR[a]	SPR/TLR[a]	XOFF2/TLR[a]	XOFF2/TLR[a]
0x08	MDR1	MDR1	MDR1	MDR1	MDR1	MDR1
0x09	MDR2	MDR2	MDR2	MDR2	MDR2	MDR2
0x0A	SFLSR	TXFLL	SFLSR	TXFLL	SFLSR	TXFLL
0x0B	RESUME	TXFLH	RESUME	TXFLH	RESUME	TXFLH
0x0C	SFREGL	RXFLL	SFREGL	RXFLL	SFREGL	RXFLL
0x0D	SFREGH	RXFLH	SFREGH	RXFLH	SFREGH	RXFLH
0x0E	BLR	BLR	-	-	-	-
0x0F	ACREG	ACREG	DIV1.6	DIV1.6	DIV1.6	DIV1.6
0x10	SCR	SCR	SCR	SCR	SCR	SCR
0x11	SSR	-	SSR	-	SSR	-
0x12	EBLR	EBLR	-	-	-	-

Table 42: UART/Irda registers

a) Transmission control register (TCR) and Trigger Level Register (TLR) are accessible only when EFR [4]=’1’ and MCR [6]=’1’.
b) MCR [7:5] and FCR [5:4] can only be written when EFR [4]=’1’.
c) In UART mode, IER [7:4] can only be written when EFR [4]=’1’. In SIR mode, EFR [4] has no impact on the access to IER [7:4].

33.3 UART/modem registers mapping

The following registers are accessible by the either the MCU or the DSP depending on the value of UIR [0]. Either ARM_UART_CS or DSP_UART_CS defines the chip-select.

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[b]	IER[b]	DLH	DLH	DLH	DLH
0x02	IIR	FCR[b]	IIR	FCR[b]	EFR	EFR
0x03	LCR	LCR	LCR	LCR	LCR	LCR
0x04	MCR[b]	MCR[b]	MCR[b]	MCR[b]	XON1	XON1
0x05	LSR	-	LSR	-	XON2	XON2
0x06	MSR/TCR[a]	TCR[a]	MSR/TCR[a]	TCR[a]	XOFF1/TCR[a]	XOFF1/TCR[a]
0x07	SPR/TLR[a]	SPR/TLR[a]	SPR/TLR[a]	SPR/TLR[a]	XOFF2/TLR[a]	XOFF2/TLR[a]
0x08	MDR1	MDR1	MDR1	MDR1	MDR1	MDR1
0x09	-	-	-	-	-	-
0x0A	-	-	-	-	-	-
0x0B	-	-	-	-	-	-
0x0C	-	-	-	-	-	-
0x0D	-	-	-	-	-	-
0x0E	-	-	UASR	-	UASR	-
0x0F	-	-	-	-	-	-
0x10	SCR	SCR	SCR	SCR	SCR	SCR
0x11	SSR	-	SSR	-	SSR	-

a) Transmission control register (TCR) and Trigger Level Register (TLR) are accessible only when EFR [4]='1' and MCR [6]='1'.

b) MCR [7:5], FCR [5:4] and IER [7:4] can only be written when EFR [4]='1'.

33.4 UIR mapping

The following register is permanently accessible only by the micro-controller and is mapped on a distinct chip-select (defined by the input ARM_SPECIAL_CS).

A[4:0]	REGISTERS	
	READ	WRITE
0x0	UIR	UIR

33.5 Receive Holding Register (RHR)

The receiver section consists of the receiver holding register (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 zero of the FIFO is used to store the single data character. (NOTE: If overflow occurs data in the RHR is not overwritten).

Bit	Name	Function	R/W	Reset
7:0	RHR	Receive holding register	R	Undef

33.6 Transmit Holding Register (THR)

The transmitter section consists of transmit holding register (THR) and transmit shift register. Transmit holding register is actually a 64-byte FIFO. The MCU 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.

Bit	Name	Function	R/W	Reset
7:0	THR	Transmit holding register	W	Undef

33.7 FIFO Control Register (FCR)

Bit	Name	Function	R/W	Reset
0	FIFO_EN	Transmit & Receive FIFOs: 0 = Disable 1 = Enable	W	0
1	RX_FIFO_CLEAR	Receive FIFO: 0 = No change 1 = Cleared and counter reseted. (*Will return to zero after clearing FIFO*).	W	0
2	TX_FIFO_CLEAR	Transmit FIFO: 0 = No change 1 = Cleared and counter reseted. (*Will return to zero after clearing FIFO*).	W	0
3	DMA_MODE	DMA mode (**if SCR[0]=0**) 0 = mode 0 1 = mode 1	W	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	W	0
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	W	0

Note: *bits 4 and 5 can only be written when EFR [4]='1'*

33.8 Supplementary Control Register (SCR)

Bit	Name	Function	R/W	Reset
0	DMA_MODE_CTL	DMA mode set with: 0 = FCR[3] 1 = SCR[2:1]	R/W	0
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]	R/W	00
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 shift register are empty.	R/W	0
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.	R/W	0
5	DSR_IT	DSR interrupt 0 = Disabled. 1 = Enabled.	R/W	0

33.9 Line Control Register (LCR)

LCR [6:0] define parameters of the transmission and reception in UART mode.

Bit	Name	Function	R/W	Reset
1:0	CHAR_LENGTH	Word length (transmit & receive): 00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits	R/W	0
2	NB_STOP	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
3	PARITY_EN	Parity bit (transmit & check): 0 = None 1 = Enable	R/W	0
5:4	PARITY_TYPE	Parity value (**if LCR[3] = 1**): 00 = Odd 01 = Even 10 = Mark 11 = space	R/W	0
6	BREAK_EN	Break condition: 0 = None. 1 = Break sending	R/W	0
7	DIV_EN	Divisor Latch: 0 = Disable 1 = Enable. Allows to access DLL, DLH and other registers (*refer to the registers mapping*)	R/W	0

33.10 Line Status Register (LSR)

33.10.1 UART mode LSR

Bit	Name	Function	R/W	Reset
0	RX_FIFO_E	RX FIFO contents status: 0 = Empty 1 = NOT empty	R	0
1	RX_OE	Receiver overrun error: 0 = No error 1 = Overrun error has occurred. *When bit is set, character held in receive shift register is not transferred to the RX FIFO. This case can occurs only when receive FIFO is full*	R	0
2	RX_PE	Receiver parity error: 0 = No 1 = Parity error	R	undef
3	RX_FE	Receiver Framing error: 0 = No Framing 1 = Framing error	R	undef
4	RX_BI	Break condition: 0 = No break condition 1 = Break detected *Bit is set, while the data being read from the RX FIFO was being received.*	R	undef
5	TX_FIFO_E	Transmit hold register status: 0 = NOT empty 1 = Empty.	R	1
6	TX_SR_E	Transmit **AND** Hold register status 0 = Not empty. 1 = Empty	R	1
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	0

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 don’t 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.

33.10.2 SIR mode LSR [UART/IrDA module]

Bit	Name	Function	R/W	Reset
0	RX_FIFO_E	RX FIFO contents status: 0 = NOT Empty 1 = empty	R	1
1	STS_FIFO_E	Status FIFO: 0 = NOT empty 1 = Empty	R	1
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].*	R	?
3	ABORT	Receive frame abort: 0 = None 1 = Received	R	?
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 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	?
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.*	R	0
6	STS_FIFO_FULL	Status FIFO: 0 = Not full. 1 = Full.	R	0
7	THR_EMPTY	Transmit hold register: 0 = Not empty. 1 = Empty. *When the bit is set, the processor can load up to 64 bytes of data into the THR if the TX FIFO is enabled.*	R	1

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 don’t 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.

33.11 Supplementary Status Register (SSR)

Bit	Name	Function	R/W	Reset
0	TX_FIFO_FULL	TX FIFO: 0 = Not full 1 = Full.	R	0
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	R	0

33.12 Modem Control Register (MCR)

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.

Bit	Name	Function	R/W	Reset
0	*Reserved*	*Reserved*	R/W	0
1	RTS*	Force RTS output: 0 = Force RTS* output to inactive (high). 1 = Force RTS* output to active (low). *In loop-back controls MSR [4].* *If auto-RTS* is enabled the RTS* output is controlled by hardware flow control*.	R/W	0
2	*Reserved*	*Reserved*	R/W	0
3	*Reserved*	*Reserved*	R/W	0
4	LOOPBACK_EN	Loop-back mode: 0 = Disable (normal mode) 1 = Enable (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	0
5	XON_EN	XON any: 0 = Disable function 1 = Enable function	R/W	0
6	TCR_TLR	TCR, TLR register access: 0 = Disable 1 = Enable.	R/W	0
7	CLKSEL	Clock input divider: 0 = None 1 = divide by 4	R/W	0

Note: *bits 5, 6 and 7 can be written only when EFR [4]='1'.*

33.13 Modem Status Register (MSR)

This 8-bit register provides information about the current state of the control lines from the modem, data set or peripheral device to the MCU. It also indicates when a control input from the modem changes state.

Bit	Name	Function	R/W	Reset
0	CTS*_STS	CTS changed: 0 = *This bit is cleared on a read.* 1 = CTS* input state has changed (*or MCR [1]] in loop-back mode*).	R	0
1	DSR*_STS	DSR changed: 0 = *This bit is cleared on a read* 1 = DSR input state has changed. (*or MCR [0] in loop-back mode*)	R	0
3:2	*Reserved*	*Reserved*	R	0
4	NCTS*_STS	This bit is the compliment of the CTS* input. In loop-back mode it is equivalent to MCR[1]	R	Input signal
5	NDSR*_STS	This bit is the compliment of the DSR input. In loop-back mode it is equivalent to MCR[0]	R	Input signal
7:6	*Reserved*	*Reserved*	R	0

33.14 Interrupt Enable Register (IER)

33.14.1 UART mode IER

There are seven types of interrupt in this mode, receiver error, RHR interrupt, THR interrupt, XOFF received and CTS*/RTS* change of state from low to high and they can be enabled/disabled individually. There is also a sleep mode enable bit.

Bit	Name	Function	R/W	Reset
[0]	RHR_IT	RHR interrupt 0 = Disable 1 = Enable	R/W	0
[1]	THR_IT	THR interrupt 0 = Disable 1 = Enable	R/W	0
[2]	LINE_STS_IT	Receiver line status interrupt 0 = Disable 1 = Enable	R/W	0
[3]	MODEM_STS_IT	modem status register interrupt 0 = Disable 1 = Enable	R/W	0
[4]	SLEEP_MODE	sleep mode 0 = Disable 1 = Enable (*baud rate clock stopped if module inactive*)	R/W	0
[5]	XOFF_IT	XOFF interrupt 0 = Disable 1 = Enable	R/W	0
[6]	RTS*_IT	RTS* interrupt 0 = Disable 1 = Enable	R/W	0
[7]	CTS*_IT	CTS* interrupt 0 = Disable 1 = Enable	R/W	0

Note: *bits 4, 5, 6 and 7 can only be written when EFR [4]='1' in UART mode.*

33.14.2 SIR mode IER [UART/IrDA module]

There are 8 types of interrupt in these modes, received EOF, LSR interrupt, TX under-run, status FIFO interrupt, RX overrun, last byte in RX FIFO, THR interrupt and RHR interrupt and they can be enabled/disabled individually.

Bit	Name	Function	R/W	Reset
0	RHR_IT	RHR interrupt: 0 = Disable 1 = Enable	R/W	0
1	THR_IT	THR interrupt: 0 = Disable 1 = Enable	R/W	0
2	LAST_RX_BYTE_IT	Frame Last-byte RX FIFO interrupt: 0 = Disable 1 = Enable	R/W	0
3	RX_OVERRUN_IT	RX overrun interrupt: 0 = Disable 1 = Enable	R/W	0
4	STS_FIFO_TRIG_IT	Status FIFO trigger level interrupt: 0 = Disable 1 = Enable	R/W	0
5	TX_UNDERRUN_IT	TX underrun interrupt: 0 = Disable 1 = Enable	R/W	0
6	LINE_STS_IT	Receiver line status interrupt: 0 = Disable 1 = Enable	R/W	0
7	EOF_IT	Received EOF interrupt: 0 = Disable 1 = Enable	R/W	0

33.15 Interrupt Identification Register (IIR)

33.15.1 UART mode IIR

The IIR is a read-only 8-bit register, which provides the source of the interrupt in a prioritized manner.

<table>
<tr><th>Bit</th><th>Name</th><th colspan="8">Function</th><th>R/W</th><th>Reset</th></tr>
<tr><td>0</td><td>IT_PENDING</td><td colspan="8">Interrupt status:
0 = IT pending
1 = None</td><td>R</td><td>1</td></tr>
<tr><td rowspan="9">5:1</td><td rowspan="9">IT_TYPE</td><td rowspan="2">Priority</td><td colspan="6">bits</td><td rowspan="2">Source</td><td rowspan="9">R</td><td rowspan="9">0</td></tr>
<tr><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></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 timeout</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 char</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>
<tr><td>7:6</td><td>FCR_MIRROR</td><td colspan="8">Mirror the contents of FCR (0).</td><td>R</td><td>0</td></tr>
</table>

33.15.2 SIR mode IIR [UART/IrDA module]

The nIRQ output is activated whenever one of the 8 interrupts is active.

Bit	Name	Function	R/W	Reset
0	RHR_IT	RHR interrupt 0 = None 1 = Pending	R	0
1	THR_IT	THR interrupt 0 = None 1 = Pending	R	0
2	RX_FIFO_LAST_BYTE_IT	Frame Last-byte RX FIFO interrupt: 0 = None 1 = Pending	R	0
3	RX_OE_IT	RX overrun interrupt 0 = None 1 = Pending	R	0
4	STS_FIFO_IT	Status FIFO trigger level interrupt 0 = None 1 = Pending	R	0
5	TX_UE_IT	TX under-run interrupt 0 = None 1 = Pending	R	0
6	LINE_STS_IT	Receiver line status interrupt 0 = None 1 = Pending	R	0
7	EOF_IT	Received EOF interrupt 0 = None 1 = Pending	R	0

33.16 Enhanced Feature Register (EFR)

This 8-bit register enables or disables enhanced features. Most of the enhanced functions only apply to UART mode but EFR [4] enables write accesses to FCR [5:4], the TX trigger level, which is also used in SIR mode.

Bit	Name	Function	R/W	Reset
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 $XON_1/XOFF_1$, $XON_2/XOFF_2$	R/W	0
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 $XON_1/XOFF_1$, $XON_2/XOFF_2$	R/W	0
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.	R/W	0
5	SPECIAL_CHAR _DETECT	Special character detect: 0 = Disable 1 = 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 a special character has been detected.*	R/W	0
6	AUTO_RTS*_EN	Auto-RTS flow control: 0 = Disable 1 = Enable *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,*	R/W	0
7	AUTO_CTS*_EN	Auto-CTS flow control: 0 = Disable 1 = Enable *Transmission is halted when the CTS* pin is high (inactive).*	R/W	0

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 should be set to different values if the software flow control is enabled.*

33.17 XON1/ADDR1 register

Bit	Name	Function	R/W	Reset
7:0	XON_WORD1	Used to store the 8-bit XON1 character in UART mode and ADDR1 address 1 for SIR mode	R/W	undef

33.18 XON2/ADDR2 register

Bit	Name	Function	R/W	Reset
7:0	XON_WORD2	Used to store the 8-bit XON2 character in UART mode and ADDR2 address 2 for SIR mode	R/W	undef

33.19 XOFF1 register

Bit	Name	Function	R/W	Reset
7:0	XOFF_WORD1	Used to store the 8-bit XOFF1 character in used in UART mode.	R/W	undef

33.20 XOFF2 register

Bit	Name	Function	R/W	Reset
7:0	XOFF_WORD2	Used to store the 8-bit XOFF2 character in used in UART mode.	R/W	undef

33.21 Scratchpad Register (SPR)

This 8-bit Read/Write Register does not control the module in anyway. It is intended as a scratchpad register to be used by the programmer to hold temporary data.

Bit	Name	Function	R/W	Reset
7:0	SPR_WORD	Scratchpad register	R/W	undef

33.22 Divisor Latches (DLL, DLH)

These are two 8-bit registers, which store the 16-bit divisor for generation of the baud clock in the baud rate generator. DLH stores the most significant part of the divisor. DLL stores the least significant part of the divisor.
Note that DLL and DLH can only be written to before sleep mode is enabled (i.e before IER[4] is set).

33.22.1 Choosing the appropriate divisor value:

UART/SIR mode: Divisor value = Operating Freq. / (16 x Baud-Rate)

In order to achieve the required baud-rate DLL/DLH must be programmed with the integer part of the divisor value.

33.22.2 Divisor latch LSB value (DLL)

Bit	Name	Function	R/W	Reset
7:0	CLOCK_LSB	Used to store the 8-bit LSB divisor value	R/W	undef

33.22.3 Divisor latch MSB value (DLH)

Bit	Name	Function	R/W	Reset
7:0	CLOCK_MSB	Used to store the 8-bit MSB divisor value	R/W	undef

33.23 Transmission Control Register (TCR)

This 8-bit register is used to store receive FIFO threshold levels to start/stop transmission during hardware/software flow control.

Bit	Name	Function	R/W	Reset
3:0	RX_FIFO_TRIG_HALT	RCV FIFO trigger level to HALT transmission (0 - 60)	R/W	F
7:4	RX_FIFO_TRIG_START	RCV FIFO trigger level to RESTORE transmission (0 - 60)	R/W	0

Note:

- *TCR can only be accessed 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-RTS* or software flow control is enabled to avoid spurious operation of the device.*

33.24 Trigger Level Register (TLR)

This 8-bit register is used to store the programmable transmit and receive FIFO trigger levels used for DMA and IRQ generation. Trigger levels from 4 - 60 can be programmed with a granularity of 4, (i.e Trigger level = 4 x [4-bit register value]).

Bit	Name	Function	R/W	Reset
3:0	TX_FIFO_TRIG_DMA	Transmit FIFO trigger level (4 - 60)	R/W	0
7:4	RX_FIFO_TRIG_DMA	RCV FIFO trigger level (4 - 60)	R/W	0

Note:

- *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.*
- *Note that for the Transmit FIFO, the TLR represents the number of empty spaces in the FIFO, above which the THR interrupt will be activated. For example, if TLR [3:0] = 1111, then if there are four or fewer bytes in the transmit FIFO, the THR interrupt will be activated. If TLR [3:0] = 0001, then if there are 60 or less bytes in the transmit FIFO the interrupt will be active.*

33.25 Mode Definition Register 1 (MDR1)

Writing to MDR1 [2:0] can program the mode of operation and therefore the MDR1 must be programmed on start-up after configuration of the configuration registers (DLL, DLH, LCR…). 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 then to the new mode.

Bit	Name	Function	R/W	Reset
2:0	MODE_SELECT	Mode selection: 000 = UART 001 = Slow InfraRed [*reserved in UART/modem*] 010 = UART with auto-baud [*reserved in IrDA*] 111 = reset/ default state *All the other values are reserved*	R/W	7
3	IR_SLEEP	SIR sleep mode: 0 = Disable 1 = Enable *this bit is reserved in UART/modem*	R/W	0
4	*Reserved*	*Reserved*	-	-
5	SCT	Sart 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*	R/W	0
6	*Reserved*	*Reserved*	-	-
7	FRAME_END_MODE	Frame end mode: 0 = Frame-length Method 1 = Set EOT bit method *this bit is reserved in UART/modem*	R/W	0

33.26 UART Autobauding Status Register (UASR) [UART/modem only]

The purpose of this function is to determine the speed, the number of bits by characters, the type of the parity. However the cases 5 and 6 bits are not considered.
In autobauding mode the input frequency of the UART/modem must be fixed to 13 Mhz.

Bit	Name	Function	R/W	Reset
4:0	SPEED	Identified speed: 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*	R	0000
5	BIT_BY_CHAR	Identified character length: 0 = 7 bits 1 = 8 bits	R	0
7:6	PARITY_TYPE	Identified parity: 00 = No Parity 01 = Space 10 = Even 11 = Odd	R	00

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” then to the UART/modem in autobaud mode “010” or UART/modem in standard mode “000”

33.27 UART Interface Register (UIR) [UART/modem only]

This register is accessed only by the micro-controller and it is used to exclusively allow the access on the IrDA/modem to the micro-controller or DSP.
If the access is given to the DSP all the register (except UIR) are accessible by the DSP (Status, Control register and FIFO (RX, TX)). The interrupts are sent to the DSP and not to the micro-controller.

Bit	Name	Function	R/W	Reset
0	UART_ACCESS	UART modem registers accesses by: 0 = MCU 1 = DSP	R/W	0
1	UART_MASK_IT	UART interrupt: 0 = Not-masked 1 = Masked	R/W	0
7:2	*Reserved*	*Reserved*	-	-

Note: *Steps to do a transition from the MCU Rhea bus to the DSP Rhea bus*

(1) MCU disables the DMA channel allocation to the UART/modem in the DMA controller block
(2) MCU disables DMA transfer since DMA bus is mapped on ARM Rhea Bus only. Setting the DMA in mode 0 does this.
(3) MCU disables IT UART_MASK_IT to 1
(4) MCU set UART_ACCESS to 1
(5) MCU enables IT UART_MASK_IT to 0

There should be no activity on DSP_nSTROBE input (input stays high) between steps (3) and (5). To do so, DSP_nSTROBE can be connected to a dedicated strobe line.

Note: *Steps to do a transition from the MCU Rhea bus to the DSP Rhea bus*

(1) MCU disables IT UART_MASK_IT to 1
(2) MCU set UART_ACCESS to 0
(3) MCU enables IT UART_MASK_IT to 0
(4) MCU can enable the DMA channel allocation to the UART/modem in the DMA controller block
(5) MCU can then enable DMA transfer. 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)

33.28 Mode Definition Register 2 (MDR2) [UART/IrDA only]

The functions of MDR2 are only used in SIR mode. This register should be programmed before the mode is programmed in MDR1[2:0].

Bit	Name	Function	R/W	Reset
0	*Reserved*	*Reserved*	-	-
2:1	STS_FIFO_TRIG	Status FIFO Threshold select: 00 = 1 character 01 = 4 characters 10 = 7 characters 11 = 8 characters	R/W	00
4:3	DIV_1.6M	MSB part of DIV_1.6	R/W	00
7:5	*Reserved*	*Reserved*	-	-

33.29 Transmit Frame Length register (TXFLL, TXFLH) [UART/IrDA only]

The registers TXFLL and TXFLH 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.
[*See Section related to the frame length closing method*]

33.29.1 TXFLL

Bit	Name	Function	R/W	Reset
7:0	TXFLL	LSB register used to specify the frame length	W	0

33.29.2 TXFLH

Bit	Name	Function	R/W	Reset
4:0	TXFLH	MSB register used to specify the frame length	W	0
7:5	*Reserved*	*Reserved*		

33.30 Received Frame Length reg. (RXFLL, RXFLH) [UART/IrDA only]

The registers RXFLL and RXFLH 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 to be n + 3 in SIR mode.

33.30.1 RXFLL

Bit	Name	Function	R/W	Reset
7:0	RXFLL	lsb register used to specify the frame length in reception	W	0

33.30.2 RXFLH

Bit	Name	Function	R/W	Reset
3:0	RXFLH	msb register used to specify the frame length in reception	W	0
7:4	-	*Reserved*		

33.31 Status FIFO Line Status Register (SFLSR) [UART/IrDA only]

Reading this register effectively reads frame status information from the status FIFO (i.e. the register doesn’t physically exist). Reading this register increments the status FIFO read pointer.

Bit	Name	Function	R/W	Reset
0	*Not used*	*Not used*	-	-
1	CRC_ERROR	CRC error (in frame at top of FIFO); 0 = No error 1 = CRC Error	R	0
2	ABORT_DETECT	Abort frame (in frame at top of FIFO): 0 = None 1 = Detected	R	0
3	FRAME_LENGTH_ERROR	Frame-length error (in frame at top of FIFO): 0 = No error 1 = Frame Error	R	0
4	OE_ERROR	Overrun error (in frame at top of RX FIFO): 0 = No error 1 = Overrun error.	R	0
7:5	*Not used*	*Not used*	-	-

33.32 RESUME register [UART/IrDA only]

This register is used to clear internal flags, which 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 zeroes being output on the data bus, DI[7:0]. [See underrun, overrun section].

Bit	Name	Function	R/W	Reset
7:0	DI	Dummy read to restart the TX or RX	R	0

33.33 Status FIFO Register (SFREGL, SFREGH) [UART/IrDA only]

The frame lengths of received frames are written into the status FIFO. This information can be read by reading the SFREGL and SFREGH registers (i.e. these registers don’t 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 should be read before the pointer is incremented by reading the SFLSR.

33.33.1 SFREGL

Bit	Name	Function	R/W	Reset
7:0	SFREGL	LSB part of the frame length	R	?

33.33.2 SFREGH

Bit	Name	Function	R/W	Reset
3:0	SFREGH	MSB part of the frame length	R	?
7:4	Not used	Not used		

33.34 BOF Length Register (BLR) [UART/IrDA only]

Bit	Name	Function	R/W	Reset
5:0	Not used	Not used	-	-
6	BOF_TYPE	SIR Start Flag Select. 0 = 0xFF 1 = 0xC0	R/W	1
7	STS_FIFO_RESET	Status FIFO reset. This bit is self-clearing	R/W	0

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, this will always be 0xC0. If (n) more than one start flag are required, then either (n-1) 0xC0 or (n-1) 0xFF flags will be 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 xBOF should 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 xBOF to send depends on the baud rate and the time for the DTE to go from TX to RX State.
The IrDA specification mentions that the allowed number of additional values are: 0, 1, 2, 3, 4, 5, 6, 8, 10, 12, 16, 24, 48.

33.35 DIV1.6 register [UART/IrDA only]

Bit	Name	Function	R/W	Reset
7:0	DIV_1.6	Used to generate the 1.6 us pulse	R/W	0

In SIR the DIV1.6 register is used to generate 1.6 us pulse encoding instead of 3/16 encoding when selected using ACREG [7].

The value of DIV1.6 is coded on ten bits by MDR2[4:3] for its MSB and DIV-1.6[7:0] for LSB.

In SIR mode DIV1.6 should 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.

33.36 Auxiliary Control Register (ACREG) [UART/IrDA only]

Bit	Name	Function	R/W	Reset
0	EOT_EN	EOT (End of Transmission) 0 = when the MCU writes to the THR 1 = MCU just writes last byte to the TX FIFO	R/W	0
1	ABORT_EN	Frame Abort. 0 = No effect 1 = Abort. *The MCU should reset the TX FIFO before the next frame is transmitted.*	R/W	0
2	SCTX_EN	Store and controlled TX start. 0 = *self-clearing bit* 1 = Start frame transmit (**if MDR1[5] = 1**)	R/W	0
3	*Reserved*	*Reserved*	-	-
4	TX_UNDERRUN	TX underrun: 0 = enable 1 = disable (long stop bits can be transmitted)	R/W	0
5	DIS_IR_RX	RXIR input: 0 = enable 1 = disable (for half-duplex purpose)	R/W	0
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	R/W	0
7	PULSE_TYPE	SIR pulse width select: 0 = 3/16 of baud-rate pulse width 1 = 1.6us	R/W	0

33.37 EBLR register [UART/IrDA only]

Bit	Name	Function	R/W	Reset
[7:0]	EBLR	This register allows to define up to 176 xBOFs, the maximum required by IrDA specification	W	0

33.38 I2C REGISTERS – FFFE:2800

The *Master I2C Interface module* has 10 registers for communication between the Rhea Bus 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 Rhea address space.

33.39 I2C register mapping

register	address	access	reset value
DEVICE_REG	FFFE:2800	7 bits R/W	1000 0000
ADDRESS_REG	FFFE:2801	8 bits R/W	0000 0000
DATA_WR_REG	FFFE:2802	8 bits R/W	0000 0000
DATA_RD_REG	FFFE:2803	8 bits R	0000 0000
CMD_REG	FFFE:2804	6 bits R/W	1101 0001
CONF_FIFO_REG	FFFE:2805	4 bits R/W	1111 1111
CONF_CLK_REG	FFFE:2806	6 bits R/W	1100 0000
CONF_CLK_FUNC_REF	FFFE:2807	7 bits R/W	1000 1010
STATUS_FIFO_REG	FFFE:2808	6 bits R	1100 0010
STATUS_ACTIVITY_REG	FFFE:2809	4 bits R	1111 0000

Table 43: I2C registers

33.40 I2C features

The Master I2C Interface Module provides an interface between Rhea Bus and I2C Bus. Rhea Bus through Master I2C Interface Module can control the external peripheral devices on the I2C bus. Fundamentally, Master I2C Interface Module is a parallel to serial and serial to parallel converter. The parallel data received from the Rhea Bus has to be converted to a suitable serial form for external peripheral devices on the I2C Bus. Also, the serial data received from the I2C bus has to be converted to a suitable parallel form for the Rhea Bus. This Master I2C Interface Module is compatible with Rhea Bus specification and Philips Master Only I2C specification.

- The *Master I2C Interface Module* **supports I2C Master Only mode** with
  1. 7 bits address DEVICE
  2. 8 bits sub address
  3. Master write to slave receiver in single or multiple mode (data loop)
     16 bytes deep transmit FIFO
  4. Master simple Read to slave receiver
  5. Read Combined cycle
  6. 3-bit programmable pre-scale internal clock divider and 7-bit programmable SCL clock divider to support wide clock frequency range of module input clock signal. The I2C SCL clock frequency are: I2C Standard Mode: 100 kHz
     I2C Fast Mode: 400 kHz
  7. 3-bit programmable spike filter to provide I2C bus input signal noise filtering ability.
  8. Asynchronous Rhea bus access with Zero Wait State Insertion
     Except for the synchronization of Rhea read access to STATUS_ACTIVITY.
  9. Error Handling Capability during I2C bus access
- The *Master I2C Interface Module* **does not support**
  1. Rhea Abort
  2. Rhea Suspend Mode
  3. Rhea Supervisor Mode
  4. Rhea DMA access
  5. I2C Bus 10-bit addressing
  6. I2C Bus CBUS compatibility
  7. Multi Master I2C

33.41 Device register (DEVICE_REG) – FFFE:2800

At the beginning of the I2C Bus read/write access, it is loaded with the information about 7-bit slave device identification.

Bit	**Name**	**Function**	**R/W**	**Reset**	
				HW	**SW**
6 : 0	DEVICE	Identification code for I2C Bus slave device		0	-
7	*Unused*	-		1	-

33.42 Address register (ADDRESS_REG) – FFFE:2801

Bit	**Name**	**Function**	**R/W**	**Reset**	
				HW	**SW**
7 : 0	ADDRESS	I2C Slave device internal register address	R/W	0	-

33.43 Date write register (DATA_WR_REG) – FFFE:2802

Bit	**Name**	**Function**	**R/W**	**Reset**	
				HW	**SW**
7 : 0	DATA_WRITE	Data to write on I2C bus	R/W	0	0

The data write register is the input register of the 16 bytes transmit FIFO.

33.44 Data read register (DATA_RD_REG) – FFFE:2803

Bit	**Name**	**Function**	**R/W**	**Reset**	
				HW	**SW**
7 : 0	DATA_READ	Data to read on I2C bus	R	0	0

33.45 Command register (CMD_REG) – FFFE:2804

Bit	**Name**	**Function**	**R/W**	**Reset**	
				HW	**SW**
0	SOFT_RESET	Reset the FIFO 0 = No reset soft 1 = Reset Soft	R/W	1	1
1	EN_CLK	Clock enable 0 = Clock is shut off 1 = Clock is enabled	R/W	0	-
2	START	Start the I2C transmission (toggle bit)	R/W	0	-
3	RW	Read Not Write Bit 0 = I2C Bus write access 1 = I2C Bus read access	R/W	0	-
4	COMB_READ	Simple or combined read Access 0 = A master read immediately, if RW = 1 1 = A combined read access, if RW = 1	R/W	1	-
5	IRQ_MSK	Rhea Bus Interrupt Request 0 = disabled 1 = enabled	R/W	0	-
7 : 6	*Unused*		R	3	-

Note: *the START toggle bit is activated when writing a 1. This bit doesn't need to be released to 0. Writing a 0 means no action.*

33.46 Configuration FIFO register (CONF_FIFO_REG) – FFFE:2805

Bit	Name	Function	R/W	Reset	
				HW	SW
3 : 0	FIFO_SIZE	Size of the FIFO (16 max) to generate the FIFO_FULL	R/W	F	-
7 : 4	*Unused*	-	R	F	-

When FIFO_SIZE = 000	then we read 1 value in the FIFO
When FIFO_SIZE = n	then we read n+1 value in the FIFO

33.47 Configuration Clock register (CONF_CLK_REG) – FFFE:2806

Bit	Name	Function	R/W	Reset	
				HW	SW
2 : 0	PTV	Pre-scale clock divider factor: Divisor_1 000 = 1 001 = 2 010 = 4 011 = 8 100 = 16	R/W	0	-
5 : 3	SPK_F	Spike Filter Factor / Check signal stability 000 = No filtering 001 = for 2 master clock 010 = for 3 master clock 011 = for 4 master clock 100 = for 5 master clock 101 = for 6 master clock 110 = for 7 master clock 111 = for 8 master clock	R/W	0	-
7 : 6	*Unused*		R	3	-

33.48 Configuration Clock functional reference register – FFFE:2807

Bit	Name	Function	R/W	Reset	
				HW	SW
6 : 0	CLK_REF	Functional clock reference: Divisor_2 0000001 = 1 .... 0000010 = 2 1111110 = 126 0000011 = 3 1111111 = 127	R/W	A	-
7	*Unused*	-	R	1	-

Note: *The CLK_FUNC_REF is generated by:*

$$CLK\_FUNC\_REF = \frac{Master\_Clock\_Frequency}{(Divisor\_2 + 1)}$$

$$Master\_Clock\_Frequency = \frac{External\_Clock\_Frequency}{Divisor\_1}$$

$$SCL\_OUT = \frac{CLK\_FUNC\_REF}{3}$$

33.49 Status FIFO register (STATUS_FIFO_REG) – FFFE:2808

Bit	Name	Function	R/W	Reset	
				HW	SW
0	FIFO_FULL	Indicate if the FIFO is full 0 = FIFO not full 1 = FIFO full	R	0	0
1	FIFO_EMPTY	Indicate if the FIFO is empty 0 = FIFO not Empty 1 = FIFO Empty	R	1	1
5 : 2	READ_CPT	Indicate the Read FIFO count value	R	0	0
7 : 6	*Unused*	-	R	3	-

33.50 Status activity register (STATUS_ACTIVITY_REG) – FFFE:2809

Bit	Name	Function	R/W	Reset	
				HW	SW
0	ERROR_DATA	Sub address or data transmit flag error: 0 = No error 1 = Error	R	0	0
1	ERROR_DEVICE	Device transmission error: 0 = No error 1 = Error	R	0	0
2	IDLE	Master I2C mode. 0 = Idle 1 = Transfer / Receive When the device is in Idle mode, the valid data is stored in read register and a new access is allowed, else no new access is allowed.	R	0	0
3	INTERRUPT	Interrupt Bit: 0 = Transfer is not completed or Module is in Idle mode 1 = Transfer Completed or aborted on nor acknowledge When the device is in Interrupt mode, the interrupt bit is used to indicate that the I2C 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	R	0	0
7 : 4	*Unused*		R	F	-

To read the status register or to write in the setup register 2, the internal clock must be running (EN_CLK of CMD_REG set to ‘1’).