CALYPSO MCU BOOT ROM application

Specification

CAL208

Ver 1.3

File: C:\TEMP\CAL208.doc

Department: Application Specific Product / Wireless Communications System

	Originator	Approval	Quality
Name	Francois AMAND	Alain BOYADJIAN	
Date	19-Oct-00	19-Oct-00	
Signature			

PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY

HISTORY

Version	Date	Author	Notes
Ver: 1.0	19-Oct-00	Francois AMAND	1
Ver: 1.1	12-Jul-01	Francois AMAND	2
Ver: 1.2	13-Jul-01	Francois AMAND	3
Ver: 1.3	11-Feb-02	Francois AMAND	4

NOTES :

1. Creation of the document
2. Memory mapping update (section 4.3).
   Note concerning IT vector in BOOT ROM (section 4.4).
   Clarification on the checksum processing (section 3.2.2.1).
3. Specification update for MCU BOOT ROM application version 2.0.0:
   - Addition of code identifier (section 4.5).
   - Optimization of the memory mapping (section 4.3).
   - Addition of ROM code checksum processing function (section 4.5).
4. Specification update for MCU BOOT ROM application version 3.0.0:
   - Correction of BUG01716 - BOOT ROM application switches from Supervisor to User mode.
   - Correction of BUG01712 - Interrupt vector in BOOT ROM does not run (see section 4.4).

PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY

SUMMARY

PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY

1 Introduction

This document explains the implementation of the CALYPSO boot application. This application will be contained in a 8kbytes of Internal ROM and executed during the boot phase according to the user selection. The boot application will be in charge to check if the application is already programmed and to download a FLASH programmer.

2 Reference documents

Reference number	Specification name	Version	Description
1	CAL000		CALYPSO top level specification
2	CAL207		CALYPSO register mapping
3	SAM207	3.2	SAMSON register mapping
4	Technical memo	26-May-2000	SAMSON-How to use internal BOOT RAM?

3 BOOT Application

3.1 Requirements

- ROM code < 8Kbytes
- All 8Kbytes must be used
- Use Internal RAM for code and data (independent of the external devices (WS, buffer size,…))
- Let the possibility to use Boot ROM to contain interrupt vector
- Use UARTs to transfer data (UART_MODEM and UART_IRDA)
- Configure core for full performance (dynamic configuration)
- Automatic detection of UART module to use
- Configure the VTCXO_26MHZ bit according to the input frequency (Automatic detection thanks to UART module)
- Have a robust protocol for the download
- The PC RAM loader application must be integrated in another FLASH programmer application
  - ➔ Provide library of Windows RAM loader
- The BOOT ROM code must be associated to a specific Windows Application
- The FLASH programmer is not dependant of the boot ROM code (only reboot procedure)
- The BOOT ROM can contain the interrupt vector table
- The boot application must determinate if an application is already programmed in FLASH
- The code will not be able to be patched.

PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

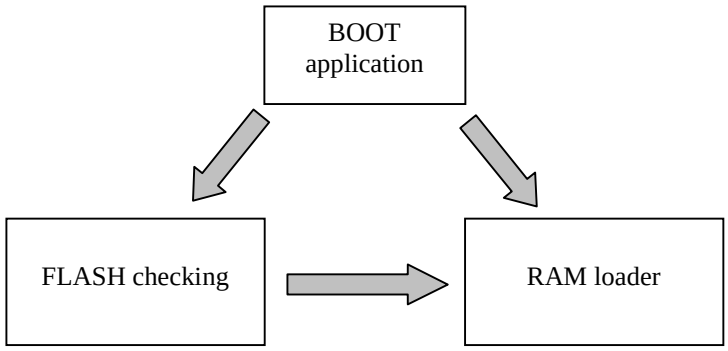
UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY

3.2 Protocol

The boot application contains the following features:

- Boot on the application if no download required,
- Download a FLASH programmer in Internal RAM and execute it (RAM loader),
- Reboot the ARM if BOOT ROM code not used to contain interrupt vector or execute the new application.



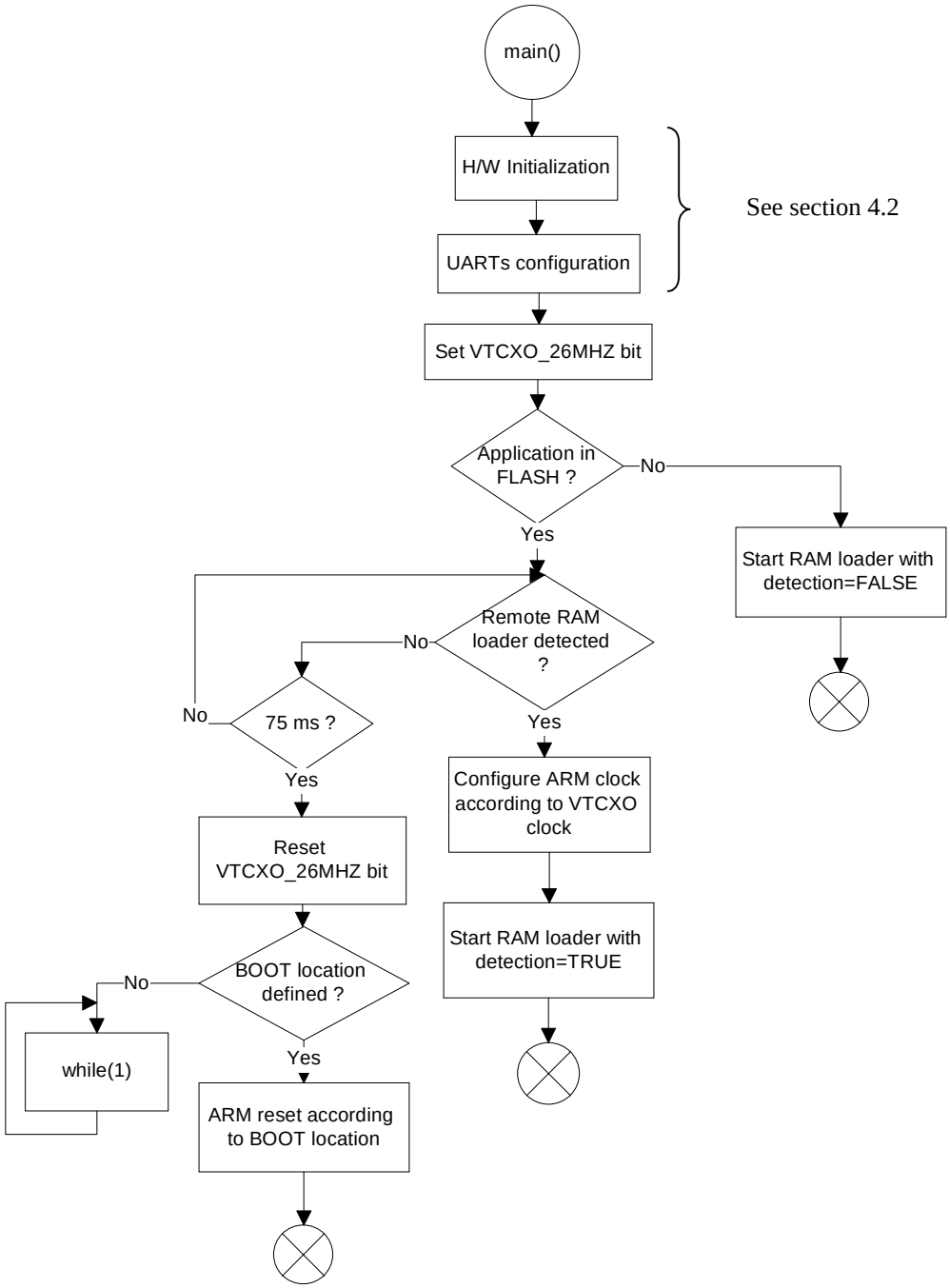
PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY

3.2.1 Boot procedure

The following algorithm is used during the boot step:
The timeout used to determinate if a remote RAM loader has been found depends on the VTCXO input clock. This timeout is configured to 75ms at 13MHz. This value has been chosen in order to have a



coherent timeout both at 26MHz and 13MHz. Note that at 26MHz, the timeout becomes around 33ms.

PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY

3.2.1.1 Detection of application in FLASH

The boot application must check several addresses in the FLASH in order to determinate if the FLASH is already programmed.

The address 0000:2000 contains the BOOT ROM configuration, which corresponds to the location of the ARM interrupt vector. The possible configurations are:

- 0x00000000: Interrupt vector in BOOT ROM.
- 0x00000001: Interrupt vector in FLASH.

If the address 0000:2000 is different to 0x00000000 or 0x00000001, the boot location is not set and therefore the reboot can not be executed successfully: the boot application calls the RAM loader. For more information about the mapping of the application, see section 4.3.

If the boot location is correct and corresponds to reboot in FLASH, the IRQ interrupt vector is checked thanks to the branch (B) instruction. If values are different to 0xFFFFFFFF, the FLASH is programmed correctly. The format of branch instruction is the following:

31 28	27 25	24	23 0
Cond	1 0 1	L	Offset

In order to perform the check, the interrupt vector table is checked at the address 0000:001C. In order to avoid any problem with the location of the BOOT ROM, these addresses will be accessed thanks to the memory range named nCS0img: the address 0000:001C can be always read from address 0300:001C.

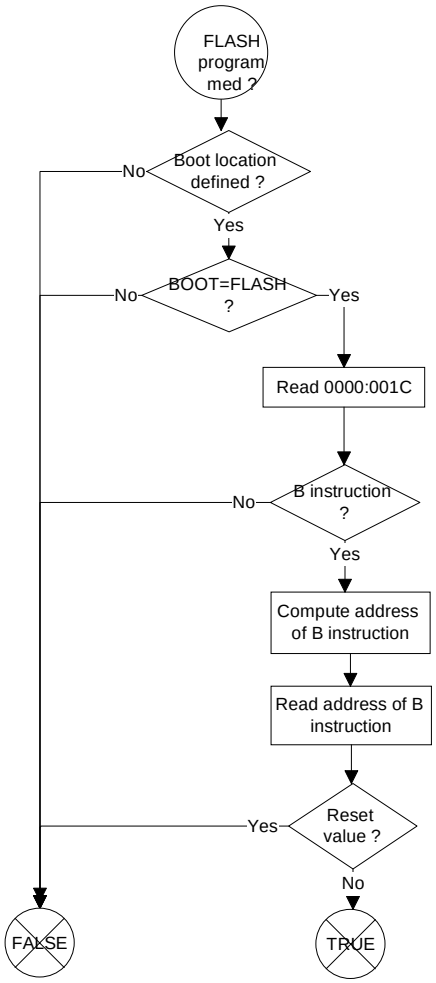
In the interrupt vector table, only the instruction Branch (B) can be present, and without any condition (Link bit (L) is only used with BL instruction.). In consequence, the check of the Branch instruction is compared to: EAXX:XXXX where XX:XXXX bits contain the offset value.
The offset value is used to determinate if an entire application is programmed, and not only the interrupt vectors (case of the first block of the FLASH is not erased).

PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY

In consequence, the following algorithm describes how to check the FLASH:



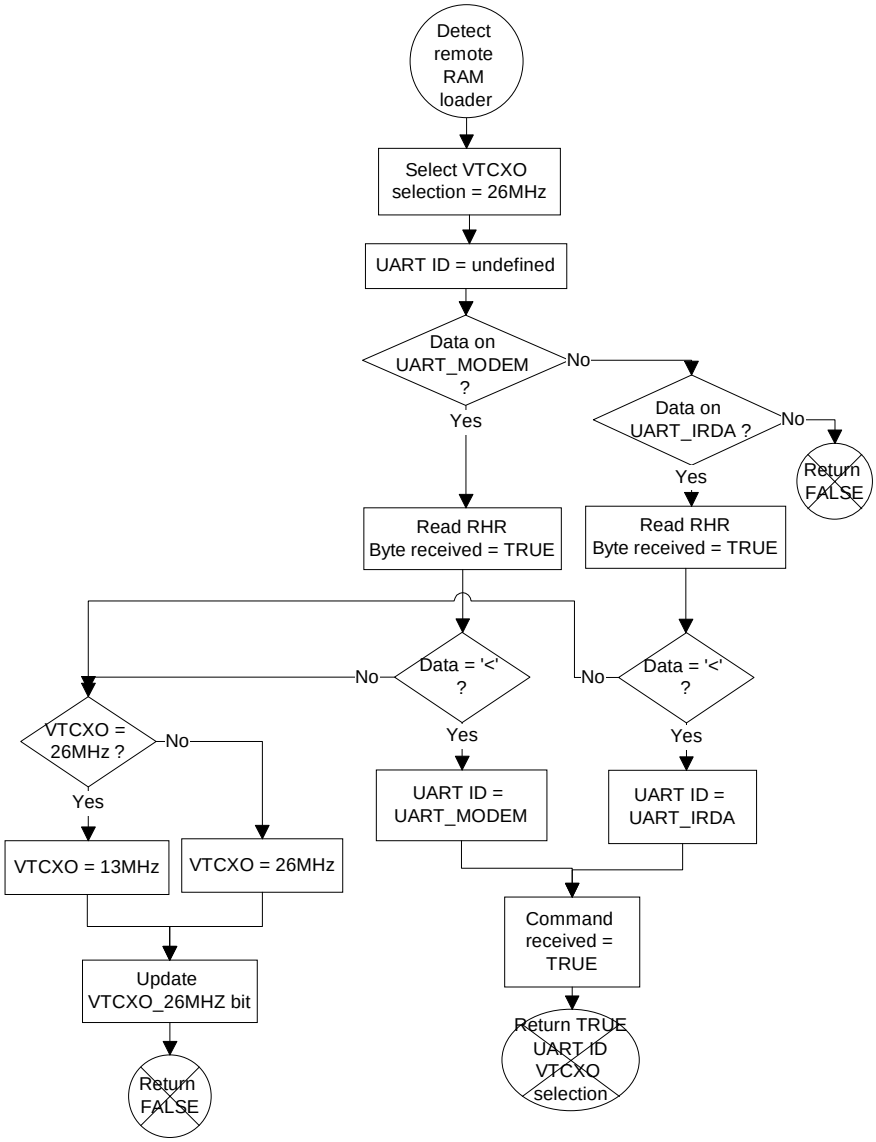
PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY

3.2.1.2 Remote RAM loader detection

The following algorithm shows the procedure to use in order to detect the remote RAM loader (loader executed on PC side):



PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY

3.2.2 RAM loader procedure

3.2.2.1 PC – Mobile commands

The RAM loader contained in BOOT ROM code is considered as a slave application, i.e. the RAM loader can only receive command and send associated response in order to inform the remote application.

PC side		Mobile side	
Description	**Command**	**Description**	**Response**
Signalling request	'<i'	Signalling response	'>i'
Parameter request	'<p[X][Y][Z][W][V]'	Parameter ACK response	'>p[X]'
		Parameter NACK response	'>P'
Write request	'<w[W][X][Y][Z]'	Write ACK response	'>w'
		Write NACK response	'>W[X]'
Checksum request	'<c[X]'	Checksum ACK response	'>c[X]'
		Checksum NACK response	'>C[X]'
Branch request	'<b[X]'	Branch ACK response	'>b'
		Branch NACK response	'>B'
Abort request	'<a'		

When an ABORT request is received, the mobile changes the UART baud rate with the default configuration and goes in SIGNALLING state. Acknowledgment is never sent to the remote application.

Format	Parameters	Definition
<i	None	None
>i	None	None
<p[X][Y][Z][W][V]	X: UART baud rate to use during download	Defined on 8 bits. 0x00 : 115200 bps 0x01 : 57600 bps 0x02 : 38400 bps 0x03 : 19200 bps 0x04 : 9600 bps
	Y: DPLL configuration	Defined on 8 bits. 7 | 6 2 | 1 0 0 | PLL_MULT | PLL_DIV PLL_MULT and PLL_DIV fields correspond to the configuration to put in DPLL register. Nevertheless, the configuration must not take into account the 13/26MHz intput clock: the mobile RAM loader is in charge to divide by 2 when 26MHz is detected.

PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY

	Z: wait state on external and internal memories	Defined on 16 bits. 15 | 14 10 | 9 5 | 4 0 0 | CS7 | CS6 | CS0 CS0, CS6 and CS7 fields correspond to the number of wait state to configure. (MSB byte is sent in first position)
	W: Access factor on strobe 0 and 1	Defined on 8 bits. 7 4 | 3 0 AF strobe 1 | AF strobe 0
	V: UART timeout configuration updated according to the ARM clock configuration. The timeout avoids to put the RAM loader in a dead lock state if a command is not received in its totality.	Defined on 32 bits. Value 0x00000000 disables the timeout. The command's decoder waits that the entire command is received. (MSB byte is sent in first position)
>p[X]	X: Maximum block size (bytes)	Defined on 16 bits, decimal value
>P	None	None
<w[V][W][X][Y][Z]	V: Block index	Defined on 8 bits
	W: Block number	Defined on 8 bits, hexadecimal value
	X: Number of bytes in the block	Defined on 16 bits, hexadecimal value
	Y: Address where the block must be written	Defined on 32 bits, hexadecimal value
	Z: Block	Each data defined on 8 bits
>w	None	None
>W[X]	X: status of the error	Defined on 8 bits 0x01 : Address error 0x02 : Bad block size
<c[X]	X: Checksum computed on PC side	Defined on 8 bits
>c[X]	X: Checksum computed on mobile side	Defined on 8 bits
>C[X]	X: Checksum computed on mobile side	Defined on 8 bits
<b[X]	X: Address of the FLASH programmer	Defined on 32 bits
<a	None	None

In order to check that the download has been done successfully, a checksum is verified by the RAM loader application before to branch in the downloaded application.

The checksum processing is based on the method used in MOTOROLA-S format generated by the HEX470 utility in order to convert COFF to ASCII file.

The checksum is defined on 8 bits and corresponds to the 1's complement of the sum of all block's checksum.
The block's checksum is the 1's complement of the sum of all data plus the block address plus the block size plus 5.

PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT **DO NOT COPY**

$$AppliChecksum = \left[Not\left[\sum_{AllBlock} BlockChecksum \right] \right]_{8LSB}$$

$$BlockChecksum = \left[Not\left[\sum (Data) + 5 + BlockSize + Address[7:0] + Address[15:8] + Address[23:16] + Address[31:24] \right] \right]_{8LSB}$$

PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY

3.2.2.2 RAM loader state machine (Mobile side)

When an error occurs, the RAM loader:

- Sends a NACK response,
- Set default UART baud rate,
- Goes in SIGNALLING state.

STATE	EVENT	ACTION	NEXT STATE
SIGNALLING	SIGNALLING_REQUEST	Send(SIGNALLING_RESPONSE)	SIGNALLING
	PARAMETER_REQUEST && Bad Baud rate	Send(PARAMETER_NACK_RESPONSE)	SIGNALLING
	PARAMETER_REQUEST && Correct Baud rate	Prepare response Send(PARAMETER_ACK_RESPONSE) Set new baud rate	AWAIT_COMMAND_ PHASE
	ABORT_REQUEST	None	SIGNALLING
AWAIT_COMMAND_ PHASE	PARAMETER_REQUEST && Bad Baud rate	Send(PARAMETER_NACK_RESPONSE) Set default baud rate	SIGNALLING
	PARAMETER_REQUEST && Correct Baud rate	Prepare response Send(PARAMETER_ACK_RESPONSE) Set new baud rate	AWAIT_COMMAND_ PHASE
	WRITE_REQUEST && Bad arguments	Prepare response Send(WRITE_NACK_RESPONSE) Set default baud rate	SIGNALLING
	WRITE_REQUEST && Correct arguments	Decrypt data block if necessary Copy block in Internal RAM Send(WRITE_ACK_RESPONSE)	COMMAND_PHASE
	ABORT_REQUEST	Set default baud rate	SIGNALLING
	SIGNALLING_REQUEST	Send(SIGNALLING_RESPONSE)	AWAIT_COMMAND_ PHASE
COMMAND_PHASE	WRITE_REQUEST && Bad arguments	Prepare response Send(WRITE_NACK_RESPONSE) Set default baud rate	SIGNALLING
	WRITE_REQUEST && Correct arguments	Decrypt data block if necessary Copy block in Internal RAM Send(WRITE_ACK_RESPONSE)	COMMAND_PHASE
	CHECKSUM_REQUEST && Correct checksum	Prepare response Send(CHECKSUM_ACK_RESPONSE)	BRANCH_PHASE
	CHECKSUM_REQUEST && Bad checksum	Send(CHECKSUM_NACK_RESPONSE) Set default baud rate	SIGNALLING
	Others event	Set default baud rate	SIGNALLING
	SIGNALLING_REQUEST	Send(SIGNALLING_RESPONSE)	COMMAND_PHASE
BRANCH_PHASE	BRANCH_REQUEST && Bad arguments	Send(BRANCH_NACK_RESPONSE) Set default baud rate	SIGNALLING
	BRANCH_REQUEST && Correct arguments	Send(BRANCH_ACK_RESPONSE) Set default baud rate Execute the FLASH programmer	SIGNALLING
	ABORT_REQUEST	Set default baud rate	SIGNALLING
	SIGNALLING_REQUEST	Send(SIGNALLING_RESPONSE)	BRANCH_PHASE
	CHECKSUM_REQUEST	Send(CHECKSUM_NACK_RESPONSE) Set default baud rate	SIGNALLING
	WRITE_REQUEST	Send(WRITE_NACK_RESPONSE) Set default baud rate	SIGNALLING

PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY

3.2.2.3 RAM loader state machine (PC side)

When NACK response is received, the RAM loader:

- Set default UART baud rate,
- Goes in RESET state,
- Inform the user.

3.2.2.3.1 State machine

STATE	EVENT	ACTION	NEXT STATE
RESET	USER_DOWNLOAD_REQUEST	Initialise UART Send(SIGNALLING_REQUEST) StartTimer(SIGNALLING_TIMER)	SIGNALLING
SIGNALLING	SIGNALLING_TIMEOUT	Send(SIGNALLING_REQUEST) StartTimer(SIGNALLING_TIMER)	SIGNALLING
	SIGNALLING_RESPONSE	Prepare request Send(PARAMETER_REQUEST)	AWAIT_PARAMETER
	USER_ABORT_REQUEST	Send(ABORT_REQUEST) Set default baud rate	RESET
AWAIT_PARAMETER	PARAMETER_ACK_RESPONSE	Set new baud rate Ready_to_write = TRUE StartTimer(WATCHDOG_TIMER)	COMMAND_PHASE
	PARAMETER_NACK_RESPONSE	Inform user (ERROR)	RESET
	USER_ABORT_REQUEST	Send(ABORT_REQUEST) Set default baud rate	RESET
COMMAND_PHASE	END_OF_FILE_NOT_REACHED && Ready_to_write == TRUE	StopTimer(WATCHDOG_TIMER) Prepare request Send(WRITE_REQUEST) Ready_to_write = FALSE StartTimer(WATCHDOG_TIMER)	COMMAND_PHASE
	END_OF_FILE_NOT_REACHED && Ready_to_write == FALSE	None	COMMAND_PHASE
	WRITE_ACK_RESPONSE	StopTimer(WATCHDOG_TIMER) Ready_to_write = TRUE StartTimer(WATCHDOG_TIMER)	COMMAND_PHASE
	WRITE_NACK_RESPONSE	Inform user (ERROR) Set default baud rate	RESET
	USER_ABORT_REQUEST	Send(ABORT_REQUEST) Set default baud rate	RESET
	END_OF_FILE_REACHED && Ready_to_write == TRUE	StopTimer(WATCHDOG_TIMER) Prepare request Send(CHECKSUM_REQUEST) StartTimer(WATCHDOG_TIMER)	CHECKSUM_PHASE
	END_OF_FILE_REACHED && Ready_to_write == FALSE	None	COMMAND_PHASE
	WATCHDOG_TIMEOUT	Set default baud rate Inform user (ERROR)	RESET
CHECKSUM_PHASE	CHECKSUM_ACK_RESPONSE	StopTimer(WATCHDOG_TIMER) Prepare request Send(BRANCH_REQUEST) StartTimer(WATCHDOG_TIMER)	BRANCH_PHASE
	CHECKSUM_NACK_RESPONSE	Inform user (ERROR) Set default baud rate	RESET
	USER_ABORT_REQUEST	Send(ABORT_REQUEST) Set default baud rate	RESET
	WATCHDOG_TIMEOUT	Set default baud rate Inform user (ERROR)	RESET
BRANCH_PHASE	BRANCH_ACK_RESPONSE	StopTimer(WATCHDOG_TIMER) Set default baud rate Inform user (SUCCESS) StartTimer(WATCHDOG_TIMER)	RESET

PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT **DO NOT COPY**

	BRANCH_NACK_RESPONSE	Set default baud rate Inform user (ERROR)	RESET
	WATCHDOG_TIMEOUT	Set default baud rate Inform user (ERROR)	RESET
	USER_ABORT_REQUEST	Send(ABORT_REQUEST) Set default baud rate	RESET

SIGNALLING_TIMEOUT : 10ms
DOWNLOAD_TIMEOUT : 2mn

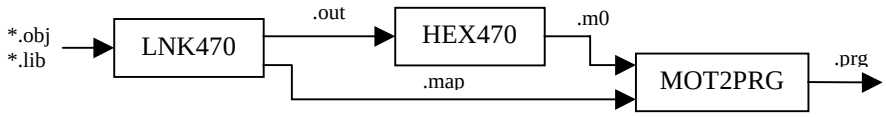
3.2.2.3.2 PC – API interface

RAM loader event	Parameter definition	Format
USER_DOWNLOAD_REQUEST	Input file name	String
USER_ABORT_REQUEST	None	Void
USER_DOWNLOAD_RESPONSE	Status of the download procedure 0x00 : Success 0x01 : Bad parameters 0x02 : Error during write 0x03 : Bad checksum 0x04 : Bad address of branch 0x05 : Watchdog timer reached	8 bits

3.2.2.3.3 File format

Start address of the FLASH downloader application
MOTOROLA-S Format

The start address of the FLASH downloader application is defined thanks to the .map file.



PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY

4 BOOT ROM implementation

4.1 Files structure

4.1.1 Inc: include files (.h)

- map.h: Registers and application addresses
- ram_loader.h: RAM loader include file
- serial.h: UART include file
- standard.h: global types
- arm_rst.h: ARM reset include file
- start.h: include file of the start applciation

4.1.2 Src: source files (.c and .s)

- arm_rst.c: ARM reset function. Must be downloaded in Internal RAM before execution
- bootloader.s: Entry point of the application and interrupt vector table
- download.c: function used to download ARM reset procedure in Internal RAM
- isr_ind.c: indirect call for all interrupts sources
- ram_loader.c: RAM loader state machine
- serial.c: UART functions
- start.c: main function of the boot application
- reboot_cs0.c: reboot function executable at the address 0000:0000
- indcall_ram.asm: duplicated IND_CALL ASM RTS routine used during a call thanks to a function pointer

4.1.3 obj: output directory contained the object files

All object files are compiled according to the debug flag. Extension '_0' corresponds to the version without debug functionality. Extension '_1' corresponds to the version with debug feature.

4.1.4 Tools: makefile and link command file

- Makefile: 'debug' flag must be set during compilation in order to select the debug mode
- Bootrom_0.cmd: link command file used for the final application
- Bootrom_1.cmd: link command file used for debug

4.1.5 Bin: output files

This directory contains the executable files both with and without debug feature and the corresponding final mapping file.

4.1.6 Doc: documentation

PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY

4.2 H/W configuration

The H/W is reconfigured during the first step of the boot application.

The function *f_initialize_hardware()* is in charge to set the H/W in a stable configuration:
- All interrupts are masked,
- Watchdog timer is disabled,
- Rhea interface is configured in order to allow the access to all registers whatever the ARM clock configuration:
  - Strobe 0: access factor = 2,
  - Strobe 1: access factor = 2.
- DPLL is set in bypass mode (division by 1),
- ARM uses VTCXO without division factor,
- VTCXO_26MHZ bit is reset,
- Wait-state on CS0, CS6 and CS7 configured:
  - CS0: 4 WS,
  - CS6: 0 WS,
  - CS7: 0 WS.

The second step consists to configure the 2 UART modules:
- Baud rate : 19200 bps,
- 8 bits, no parity, 1 stop bit,
- Modem mode.

This H/W configuration will be able to be ameliorated thanks to Parameter command, which allows the modification of ARM speed, memories wait state and rhea accesses.

The VTCXO_26MHZ bit is set in order to have a compliant clock on UART modules. This configuration will be used to determinate the VTCXO clock thanks to data received on serial ports.

Timeout, used to determinate if a remote RAM loader is present, is measured thanks to an ARMIO output pin in debug mode. The code is compiled in 32 bits mode with option –o2. For more information about the value of the timeout, see *start.h* file.

PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY

4.3 Mapping of the BOOT ROM code

Address	Definition	Memory
0000:0000	Interrupt vector table	BOOT ROM
0000:0020	ARM reset procedure (load address)	
0000:0244	Boot program	
0000:1FFE	Code identifier on 16 bits	
0000:2000	Boot ROM location status: - 0x00000000: IT vector in BOOT ROM - 0x00000001: IT vector in FLASH	External FLASH
0000:2004	Entry point address of the application (used when the boot ROM contains the interrupt vector)	
0080:0000	Shared memory between application and boot ROM when interrupt vectors are contained in it.	Internal SRAM
0080:001C	Run address of interrupt indirect call	
0080:0038	Boot switch function (run address)	
0080:0104	Data memory	
0080:05C0	Stack memory (400 bytes)	
0080:0750	FLASH programmer application (downloaded thanks to RAM loader) (Around 510.2Kbytes)	

The application programmed in FLASH must:
- Indicates the boot ROM location at a specific address in the FLASH,
- Indicates the entry point of the application at a specific address in the FLASH.

This setting is possible by defining a variable as ***const***.

The FLASH programmer must be downloaded in the second block of 1Mbits of the Internal RAM. This choice has been done in order to let the possibility to use 3Mbits of Internal RAM on CALYPSO chip.

Nevertheless, the memory ranges used by the BOOT ROM application and used to download the FLASH programmer must not be protected by the MPU in case of a reboot in the BOOT ROM is performed thanks to a watchdog reset, because the MPU configuration is not reset on this event.

4.4 Interrupt vectors in BOOT ROM

This feature has been implemented in order to let the possibility to have the interrupt vector located in the BOOT ROM and in consequence to decrease the time processing during occurrence of them.

Nevertheless, this feature is not functional in the current version of the BOOT ROM application. Therefore, it is not possible to have the interrupt vector in the internal BOOT ROM.
This constraint implies that the application programmed in FLASH must force the BOOT ROM location to 0x00000001 at the address 0000:2000.

PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY

The branch instructions located in the interrupt vector table at the addresses 0000:0004 to 0000:001C jumps directly in the Internal SRAM at the addresses 0000:001C to 0000:0034 when an interrupt occurs (see BOOT ROM mapping in section 4.3). At this addresses, the ARM fetches the instruction *LDR PC, [PC, #-0x24]* in order to put in PC the address located at the addresses 0080:0000 – 0080:0018. The offset -0x24 is used in order to take into account the ARM pipeline.

4.5 ROM code checksum and identifier

4.5.1 ROM code identifier

A ROM code identifier has been added in the ROM code in order to differentiate the different ROM code's version. The identifier is defined on 16 bits and located at the address 0000:1FFE.

BOOT ROM code version	Identifier
1.0.0	Not applicable
2.0.0	0x0200
3.0.0	0x0300

Table 1: ROM code identifier.

Note: The ROM code identifier is only available from the version 2.0.0.

4.5.2 ROM code checksum

A specific function has been integrated in the BOOT ROM application in order to compute the ROM code checksum.

The function must be called in 32 bits mode at the address specified in Table 2 and has the following prototype:

unsigned short int f_bootrom_checksum(unsigned char b_hw_init);

The valid arguments are:

0x00: the hardware is not reconfigured before to compute the checksum.

0x01: the hardware is reconfigured (ARM sets at 13MHz).

The returned checksum is defined on 16 bits and is computed thanks to the following formula:

$$BootRomChecksum = \sum_{i=0000:0000}^{i<0000:2000} DataRom16Bits[i]$$

BOOT ROM code version	Checksum	Function address
1.0.0	Not applicable	Not applicable
2.0.0	0xB2BA	0000:16B8
3.0.0	0xF1E3	0000:1608

Table 2: ROM code checksum.

PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY

Note: The ROM code checksum processing function is only available from the version 2.0.0.

4.6 RAM loader command

Command and response are decoded on the fly. Nevertheless, in order to avoid a deadlock state, a timeout is implemented during the character reception. When a timeout occurs, the RAM loader state machine stays in the same state and waits the next command.

Each time that a Parameter command is received in SIGNALLING or AWAIT_COMMAND states, a new H/W configuration is performed (DPLL and WS).
Values exchanged thanks to Parameter command in order to configure the DPLL must not take into account the VTCXO clock. Configuration PLL_MULT and PLL_DIV fields must suppose that the VTCXO clock is 13MHz even if it is not the case. The BOOT ROM application is in charge to configure the ARM clock according to the selection of the VTCXO frequency (13 or 26MHz).

Each time that a write command is received in AWAIT_COMMAND or COMMAND_PHASE states, the block address is checked in order to avoid erasing the data and stack sections of BOOT ROM application.

4.7 Code size

The BOOT ROM application is compiled in 32 bits mode with option –o2.

ROM size: 5820 bytes
RAM size: 1212 bytes
Stack size: 400 bytes

Free space: ~2000 bytes

PRELIMINARY documents contain information on a product under development and is issued for evaluation purposes only. Features characteristic data and other information are subject to change.

UNDER NON DISCLOSURE AGREEMENT

DO NOT COPY