Understanding BSP Bootup Sequence for LOCOSTO

ABSTRACT

This Application report gives an overview of the requirements for initializing the LOCOSTO device. It takes you through Pre-Initialization phase, reset sequence, BootROM code, Bootstrap loader and OS startup.

Table of Contents:

Assumptions

This document is addressed for 13202_tcs3[1].2.afinal. The scope of this document largely pertain to the BootRom and the following files: **int.s, inc.c, rlc.c, lic.c, eri.c, hii.c, tci.c, mib.c, qui.c, pii.c, smi.c, eri.c, pmi.c, dmi.c, tmi.c, ioi.c, main.c, tct.c, init.c, frame.c, create_RVtasks.c**.

CHIPSET = 15 is assumed for Locosto.

Every effort has made to ensure that the information contained in this document is accurate at the time of printing. However, the software described in this document is subject to continuous development and improvement.

List of References

#	Title	Version	Description
1	L1S_LOC016_1.doc		Locosto MCU BootROM Code PRD
2	L1S_LOC010	1.1	Locosto MCU BOOT ROM application
3	L1S_CALP010	1.32	Calypso Plus MCU BootROM Application Specification
4	L1S_LOC_TM016_1		Peripheral Booting without security for Locosto
5	locosto_bum_Initialization.pdf	3.0	Introduction to the LOCOSTO device initialization.
5	L1S_LOC022_1.doc	1.0	L1 Calypso Plus to Locosto Porting Guide.
6	locosto_bum_memory_mapping.pdf	3.1	Introduction to the MPU and the DSP memory address spaces of the LOCOSTO device.

Acronyms Abbreviations and Definitions

CS: Chip Select.

DPLL: Digital PLL.

ID: Identifier.

IRQ: Interrupt.

MPU: Micro Processor Unit.

nCS3: Active low Chip Select number three

NOR: A type of Flash.

OS: Operating System.

PLL: Phase Locked Loop.

PRRM: Power, Reset and Clock Management module.

RAM: Random Access Memory. Refers to the OMAP internal static RAM.

ROM: Read Only Memory.

SRAM: Static Random Access Memory.

TOC: Table Of Contents – a structure within an executable image.

USB: Universal Serial Bus.

UART: Universal Asynchronous Receiver/Transmitter.

Terminology

This chapter provides an overview of the requirements for initializing a LOCOSTO device from the point of power-on to the point of OS load and applications running.

- **Bootstrap**: Initial Software that is launched by the Boot ROM Code during the Memory Booting phase.
- **Certificate**: Data block plus trusted signature of the data block. Used during memory and Peripheral Booting for HS devices.
- **E-Fuse**: A one-time programmable memory location usually set at the factory.
- **Flash Loader**: Downloaded Software launched by the ROM Code in Pre-Flashing and which programs an image into external NOR Flash memories.
- **GP device**: General Purpose device. In this type of device Hash (MPK) is zero. Hence the security features are disabled.
- **HS device**: High Security device. This is the type of device where Hash (MPK) is non-zero. In this device all the security features are enabled.
- **Memory Booting**: ROM Code mechanism that allows software residing in external memory (NOR Flash memory) to start execution.
- **MPK, Hash (MPK), Hash (ManPubKey)**: Manufacturer Public Key. Hash of a particular Manufacturer's Public Key is e-fused at manufacturing phase. This is used for authentication for both memory as well as Peripheral Booting.

- **Peripheral Booting**: ROM Code mechanism that consists of polling selected interfaces, downloading and executing initial Software (in this case Flash Loader) in internal RAM.
- **Pre-Flashing**: Pre-Flashing is a specific case of Peripheral Booting, when the ROM Code mechanism is used as part of a Flashing process to program a NOR Flash memory.
- **Protected Applications**: Application that is signed by authenticated author and meant to be executed in Secure Execution Environment.
- **ROM Code**: ROM Code is the on-chip software located in Secure ROM implementing Booting and Security features.
- **Secure Mode**: Protected execution mode of the device. Some peripherals (like internal Secure RAM and Secure ROM) are visible and allowed to be accessed only in this mode.
- **Secure RAM**: Writable memory inside the chip. Can be accessed only when the device is in Secure Mode.
- **Secure ROM**: Read-only memory inside the chip programmed at chip manufacturing phase. Can be accessed only when the chip is in Secure Mode.
- **TOC, Table of Contents**: A set of data blocks containing basic information like code size, entry point of the firmware, etc. Required during Peripheral Booting for GP device.

1 Introduction

This document gives a brief overview of the startup of the software on the LOCOSTO (I-Sample) covering the Pre-initialization, Ramp sequence, ROM Code ,Bootstrap ,OS start.

Initialization of a LOCOSTO device consists of multiple steps as shown below.

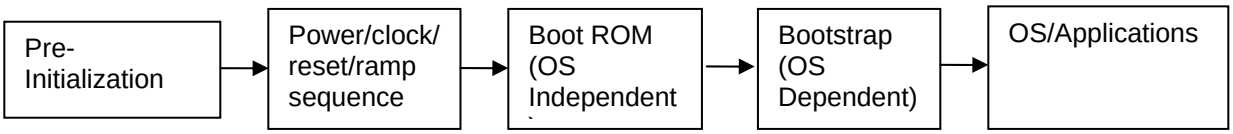

Figure 1. Locosto Initialization

Each block is explained in the subsequent chapters.

2 Pre-Initialization

The LOCOSTO device boot-up operation requires certain hardware configuration settings. Clock, reset, and power connections, as well as the pins involved in setting the boot memory space for the microprocessor unit (MPU), must be connected and driven properly for successful device initialization.

This section details the specific requirements for the pre-initialization stage.

2.1 Power Connections

Power to the LOCOSTO device can be supplied by an external companion chip. Texas Instruments provides a global solution for the LOCOSTO device using an external power device: the TWL3031 power integrated circuit (IC).

For more information on the TWL3031 IC, see the *TWL3031 Data Sheet*.

The below Figure presents the power connections between the LOCOSTO device and the TWL3031 device.

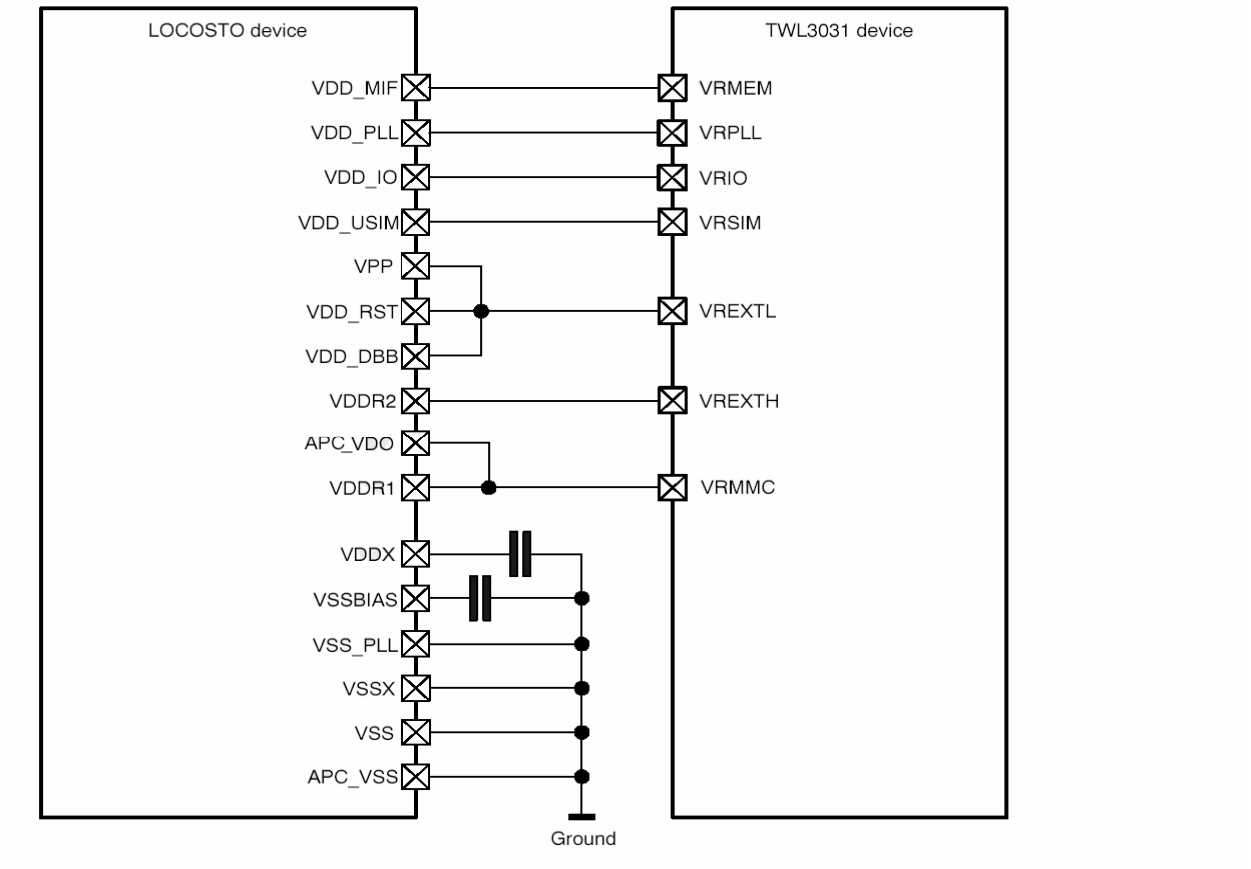

Figure 2. Power connections between the LOCOSTO and TWL3031.

The table below details the Pin Voltage Name, the ABB Names and their description

LOCOSTO Pin Voltage Name	TWL3031 Source	Description
VDD_MIF	VRMEM	External memory interface (EMIF) I/O power supply
VDD_PLL	VRPLL	Supply voltage for the digital baseband phase-locked loop (DBB PLL)
VDD_IO	VRIO	Generic I/O power supply
VDD_USIM	VRSIM	Universal subscriber identity module (USIM) I/O power supply
VPP	VREXTL	Digital radio processor (DRP) core digital supply
VDD_RST	VREXTL	Reset I/O power supply
VDD_DBB	VREXTL	DBB application-specific IC (ASIC) core power supply, including the internal static RAM (SRAM) memory and the internal ROM
VDDR2	VREXTH	Pre-regulated input to the low lropout (LDOX) DRP embedded LDO
APC_VDO	VRMMC	Automatic power control (APC) output amplifier power supply
VDDR1	VRMMC	Pre-regulated input to the LDOOS, LDOA and LDORF DRP embedded LDOs

Table 1. Pin Voltage Name and their description

2.2 Clock and Reset

2.2.1 Clock and Reset Overview:

The figure below shows the LOCOSTO clock and reset environment, which gathers all of the signals related at the system level to clocks and resets.

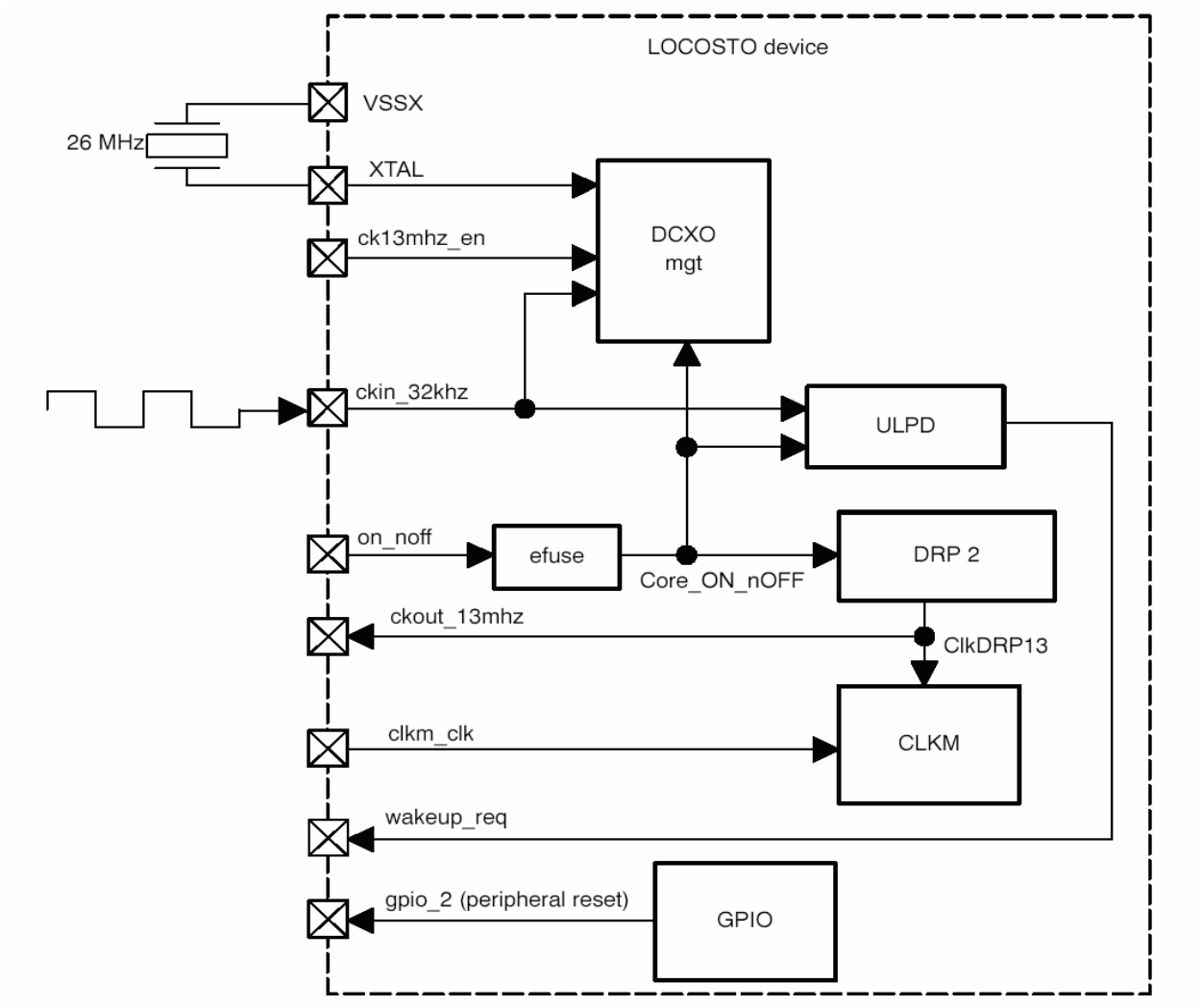

Figure 3. LOCOSTO clock and reset environment.

The LOCOSTO device requires two external clock inputs: the functional and interface clocks are all generated from the ckin_32khz and the XTAL input clocks.

The Table below describes the clock and reset I/Os of the LOCOSTO device.

I/O Name	Type	Description
XTAL and VSSX	Clock input	The 26-MHz system clock is the main source clock of the chip and is connected to a 26-MHz quartz crystal.
ckin_32khz	Clock input	The 32-kHz clock is generated by the TWL3031 power IC, used for low frequency operations, and supplies the wake-up domain for operation in the lowest power mode.
clkm_clk	Clock input	This external input clock can be an alternate solution for the quartz crystal and supports digital clock frequencies up to 40 MHz.
ck13mhz_en	Logic input	Enables the system clock generation
ckout_13mhz	Clock output	This 13-MHz output clock is available by setting the CONF_LOCOSTO_ DEBUG[3] BUT_CLK13M bit to 0.
on_noff	Reset input	Cold reset for the chip.
wakeup_req	Wakeup output	When the LOCOSTO device wakes up from the deep sleep state, the ultra-low power down (ULPD) module drives this signal to a high level.
gpio_2	Reset output	The LOCOSTO gpio_2 is the peripheral reset signal and is connected to the reset pins of the peripheral devices. During booting, the gpio_2 pin is a pull-down; thus, the resets of all peripheral devices are enabled.

Table 2. Clock and reset I/Os of the LOCOSTO device

2.2.2 Clock Configuration:

The digital PLL (DPLL) module and the clock manager (CLKM) module control the LOCOSTO clocks. The DPLL module generates the ClkDPLL104 internal clock configured through its control registers. The CLKM module disables/enables the clocks while dispatching them.

The following conditions apply after the LOCOSTO device is reset and all of the clocks are stable:

- The application software can write to the control registers (CLKM.CNTL_ARM_CLK[2:1] CLKIN_SEL and CLKIN_SEL0 bits) to switch to a desired mode of operation. The CLKM.CNTL_ARM_CLK [2:1] bits should be changed before modifying the clock divider ratios in the DPLL_CNTL_REG register.
- For the 104-MHz DPLL clock, the DPLL_CNTL_REG register configuration is 0x2830 ClkDPLL104 = ClkDRP13 * 16 / 2).
- To determine if the DPLL is locked, poll the DPLL_CNTL_REG[0] LOCK bit.

2.3 Boot Configuration

Two external LOCOSTO pins, USB_BOOT and nBSCAN, are used to select the boot mode. The ROM code is configured to download software from the universal serial bus (USB) interface or the universal asynchronous receiver/transmitter (UART) interface selected by the USB_BOOT GPIO pin.

nBSCAN Pin State	USB_BOOT Pin State	UART	USB	Boundary Scan
1 (high)	1 (high)		X	
1 (high)	0 (low)	X		
0 (low)	1 (high)			X
0 (low)	0 (low)			X

Table 3. USB_BOOT Configuration

This mechanism applies to both initial flashing in production (flash memory is empty) and re-flashing in service (flash memory has content). After booting, the USB_BOOT pin can optionally be used as GPIO or LPG output for other functions.

To set the chip test engineering and the I/O boundary scan, pull down the nBSCAN pin to a low level. To boot the device, the nBSCAN pin should be high (an on-chip pullup is set by default).

The figure below shows the boot configuration environment used to select the boot mode.

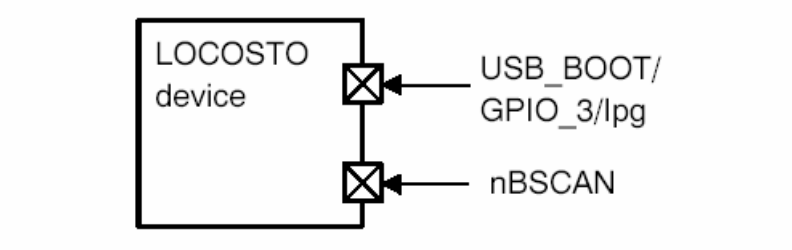

Figure 4. Boot configuration environment

2.4 Pin Multiplexing and Pullup/Pulldown

Each pin with a multiplexing function is assigned a MODE bit field in a configuration register, thus creating up to eight possible multiplexing options per pin (with up to 3 bits in the MODE bit field). At reset (on_noff is low), each pin is forced asynchronously to a default multiplexing configuration mode (mode0). Most pins also can be configured either as a pullup or a pulldown with a 2-bit field.

2.5 Memory Mapping

2.5.1 Memory Address Space Overview

The 32-bit microprocessor unit (MPU) address bus defines a 4G-byte space. This space is divided into regions based on the type of internal bus used.

The MPU memory space is composed of:

- 512K-byte internal memory space through the internal memory interface (IMIF)
- 124M-byte external memory space through the external memory interface (EMIF)
- Internal memory-like peripheral space through the MEM_PERIPHERAL interface (I/F)
- 32K-byte internal DSP shared memory space (the API) through an application programming interface/Texas Instruments peripherals bus (API/TIPB) bridge
- Peripheral space through an API/TIPB bridge

The LOCOSTO DSP memory space is divided into seven 128K-byte pages:

- 1 data memory page
- 5 program memory pages
- 1 XIO memory page

The figure below shows the memory space divided into six spaces: The boot ROM, external memory, internal memory, memory-like peripherals, API memory, and peripherals.

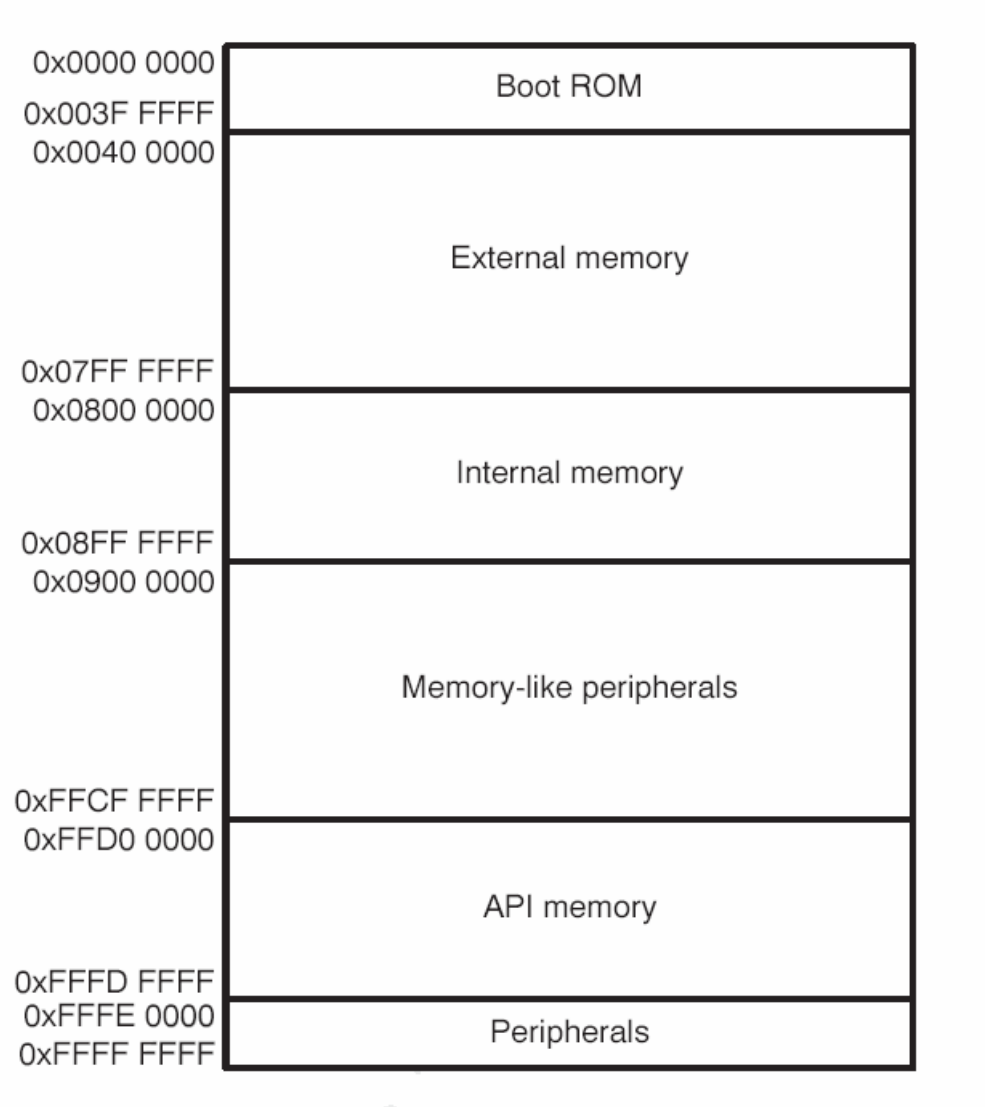

Figure 5. MPU Memory Mapping Overview

The table below describes the address range of each space.

Device Name	Start Address	End Address	Size	Data Access
Boot Area 0x0000 0000 - 0x003F FFFF				
Boot ROM	0x0000 0000	0x0000 FFFF	64K bytes	8/16/32 bits R
Reserved	0x0001 0000	0x003F FFFF		
External Memory 0x0400 0000 - 0x07FF FFFF				
External memory	0x0040 0000	0x07FF FFFF	124M bytes	8/16/32 bits R/W
Internal Memory 0x0800 0000 - 0x08FF FFFF				
Protected RAM	0x0800 0000	0x0803 FFFF	256K bytes	8/16/32 bits R/W
Nonprotected RAM	0x0804 0000	0x0804 FFFF	64K bytes	8/16/32 bits R/W
ROM	0x0805 0000	0x0807 FFFF	192K bytes	8/16/32 bits R
Reserved	0x0808 0000	0x08FF FFFF		
Internal Memory-Like Peripherals 0x0900 0000 - 0x0FFF FFFF				
Boot ROM	0x0900 0000	0x0900 FFFF	64K bytes	8/16/32 bits R
Reserved	0x0901 0000	0x096F FFFF		
Camera	0x0970 0000	0x097F FFFF	1M byte	32 bits R/W
SHA-1/MD5	0x0980 0000	0x098F FFFF	1M byte	32 bits R/W
DES/3DES	0x0990 0000	0x099F FFFF	1M byte	32 bits R/W
RNG	0x09A0 0000	0x09AF FFFF	1M byte	32 bits R/W
TI reserved	0x09B0 0000	0x09CF FFFF		
NAND flash	0x09D0 0000	0x09DF FFFF	1M byte	8/16/32 bits R/W
MSSPI	0x09E0 0000	0x09EF FFFF	1M byte	32 bits R/W
Debug unit	0x09F0 0000	0x09FF FFFF	1M byte	32 bits R/W
Reserved	0x0A00 0000	0xFFCF FFFF		
API Memory 0xFFD0 0000 - 0xFFFD FFFF				
API RAM	0xFFD0 0000	0xFFD0 7FFF	32K bytes	16/32 bits R/W
Reserved	0xFFD0 8000	0xFFDF FFFF		
API control register	0xFFE0 0000	0xFFE0 0001	2 bytes	16 bits R/W
Reserved	0xFFE0 0002	0xFFFD FFFF		
Peripherals 0xFFFE 0000 - 0xFFFF FFFF				
TIPB strobe #1	0xFFFE 0000	0xFFFE FFFF	64K bytes	8/16 bits R/W
TIPB strobe #0	0xFFFF 0000	0xFFFF FFFF	64K bytes	8/16 bits R/W

Table 4. MPU Memory Mapping Addresses

2.5.2 External Memory Mapping

The EMIF can control four external devices through four independent chip-selects (CS0, CS1, CS2, and CS3) without adding any external logic. The figure below shows the link between the MPU addresses and the external memory chip-selects. Because of the boot area, the first chip-select (CS0) can only support up to 28M bytes of memory; the other three chip-selects can support up to 32M bytes of memory.

Note: *The nCS1 and nCS2 signals are not available at power-on reset due to multiplexed pins (respectively, gpio_36 and gpio_35).*

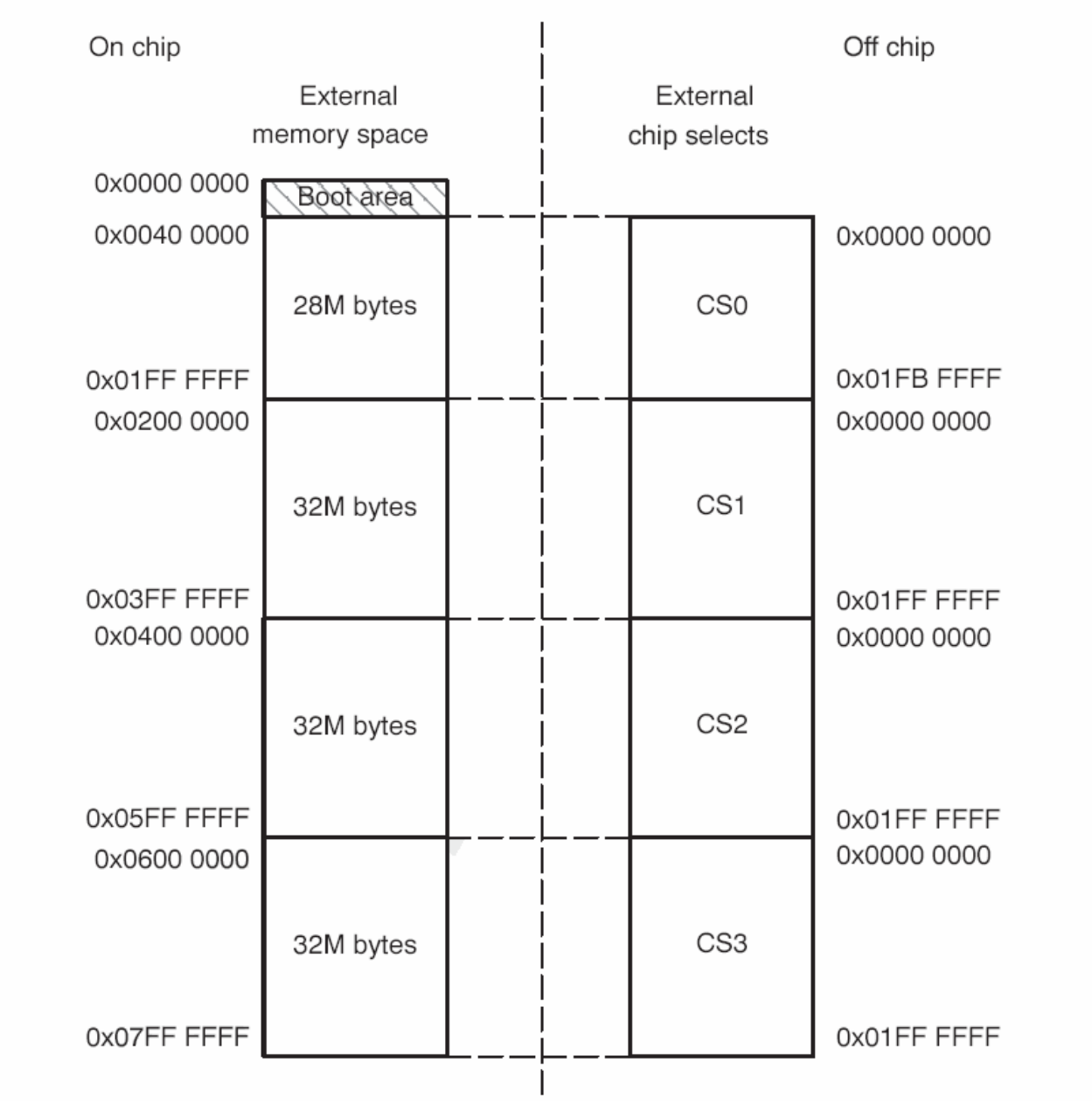

Figure 6. External Memory Mapping

2.5.3 API

These 32K bytes of internal dual access data RAM (DARAM) are shared between different resources: the MPU subsystem, the DSP subsystem, and the DMA subsystem.

2.5.4 Peripheral and Control Registers

This space is accessible through the TIPB STROBE #0 and STROBE #1 lines. The below two tables list the modules that are connected to the STROBE #0 line and the STROBE #1 line, respectively.

Device Name	Start Address	End Address	Size	Data Access
DRP reserved†	0xFFFF 0000	0xFFFF 3FFF		
DRP external memory	0xFFFF 4000	0xFFFF 4FFF	4K bytes	16 bits R/W
APC‡	0xFFFF 5000	0xFFFF 57FF	2K bytes	16 bits R/W
Reserved	0xFFFF 5800	0xFFFF 6FFF		
UART‡	0xFFFF 7000	0xFFFF 727F	640 bytes	8 bits R/W
UART TIPB/OCP switch	0xFFFF 7280	0xFFFF 77FF	1.375K bytes	8 bits R/W
MCSI‡	0xFFFF 7800	0xFFFF 7FFF	2K bytes	16 bits R/W
DRP dynamic switch	0xFFFF 8000	0xFFFF 87FF	2K bytes	16 bits R/W
APC TIPB static switch	0xFFFF 8800	0xFFFF 881F	32 bytes	16 bits R/W
MCSI TIPB static switch	0xFFFF 8820	0xFFFF 883F	32 bytes	16 bits R/W
C-port TIPB static switch	0xFFFF 8840	0xFFFF 885F	32 bytes	16 bits R/W
Reserved	0xFFFF 8860	0xFFFF 8FFF		
TPU RAM	0xFFFF 9000	0xFFFF 97FF	2K bytes	16 bits R/W
DPLL	0xFFFF 9800	0xFFFF 9FFF	2K bytes	16 bits R/W
LCD interface	0xFFFF A000	0xFFFF A7FF	2K bytes	16 bits R/W
USIM interface	0xFFFF A800	0xFFFF AFFF	2K bytes	16 bits R/W
USB	0xFFFF B000	0xFFFF B7FF	2K bytes	16 bits R/W
I^2C	0xFFFF B800	0xFFFF BFFF	2K bytes	16 bits R/W
GEA3	0xFFFF C000	0xFFFF C7FF	2K bytes	16 bits R/W
I^2C Triton Lite	0xFFFF C800	0xFFFF CFFF	2K bytes	16 bits R/W
C-port (control registers)‡	0xFFFF D000	0xFFFF D7FF	2K bytes	16 bits R/W
C-port (FIFO registers)‡	0xFFFF D800	0xFFFF DFFF	2K bytes	16 bits R/W
TI reserved†	0xFFFF E000	0xFFFF E7FF		
DMA controller	0xFFFF E800	0xFFFF EFFF	2K bytes	16 bits R/W
TPU	0xFFFF F000	0xFFFF F7FF	2K bytes	16 bits R/W
Watchdog	0xFFFF F800	0xFFFF F87F	128 bytes	16 bits R/W
Secure watchdog	0xFFFF F880	0xFFFF F8FF	128 bytes	16 bits R/W
API/TIPB bridge	0xFFFF F900	0xFFFF F9FF	256 bytes	16 bits R/W

Interrupt handler	0xFFFF FA00	0xFFFF FA7F	128 bytes	16 bits R/W
Secure interrupt handler	0xFFFF FA80	0xFFFF FAFF	128 bytes	16 bits R/W
EMIF	0xFFFF FB00	0xFFFF FB2D	46 bytes	16 bits R/W
API_RHEA_CNTL register	0xFFFF FB2E	0xFFFF FB2F	2 bytes	16 bits R/W
BOOT_MODE_CONF register	0xFFFF FB30	0xFFFF FB31	2 bytes	16 bits R/W
Reserved	0xFFFF FB32	0xFFFF FBFF		
PRRM	0xFFFF FC00	0xFFFF FCFF	256 bytes	16 bits R/W
CLKM	0xFFFF FD00	0xFFFF FDFF	256 bytes	16 bits R/W
Reserved	0xFFFF FE00	0xFFFF FEFF		
EMPU	0xFFFF FF00	0xFFFF FFFF	256 bytes	16 bits R/W

† To avoid hazardous effects, do not write into this space.
‡ Module behind a configurable switch

Table 5. Peripheral Space Addresses – TIPB Strobe #0

- Peripheral Space Addresses – TIPB Strobe #1

Device Name	Start Address	End Address	Size	Data Access
Reserved	0xFFFE 0000	0xFFFE 07FF		
TPU–2–OCP	0xFFFE 0800	0xFFFE 0FFF	2K bytes	16 bits R
Reserved	0xFFFE 1000	0xFFFE 1FFF		
ULPD	0xFFFE 2000	0xFFFE 27FF	2K bytes	16 bits R/W
Reserved	0xFFFE 2800	0xFFFE 37FF		
TIMER1	0xFFFE 3800	0xFFFE 3FFF	2K bytes	16 bits R/W
Reserved	0xFFFE 4000	0xFFFE 47FF		
GPIO	0xFFFE 4800	0xFFFE 4FFF	2K bytes	16 bits R/W
GPIO1	0xFFFE 5000	0xFFFE 57FF	2K bytes	16 bits R/W
GPIO2	0xFFFE 5800	0xFFFE 5FFF	2K bytes	16 bits R/W
Reserved	0xFFFE 6000	0xFFFE 67FF		
TIMER2	0xFFFE 6800	0xFFFE 6FFF	2K bytes	16 bits R/W
Reserved	0xFFFE 7000	0xFFFE 77FF		

Peripheral Space Addresses – TIPB Strobe #1

Device Name	Start Address	End Address	Size	Data Access
LPG	0xFFFE 7800	0xFFFE 7FFF	2K bytes	8 bits R/W
PWL	0xFFFE 8000	0xFFFE 87FF	2K bytes	8 bits R/W
PWT	0xFFFE 8800	0xFFFE 8FFF	2K bytes	8 bits R/W
Software debug registers	0xFFFE 9000	0xFFFE 97FF	2K bytes	16 bits R/W
Reserved	0xFFFE 9800	0xFFFE B7FF		
Keyboard controller	0xFFFE B800	0xFFFE BFFF	2K bytes	16 bits R/W
Reserved	0xFFFE C000	0xFFFE EFFF		
JTAG PART_ NUMBER (ID-CODE[27:12])	0xFFFE F000	0xFFFE F001	2 bytes	16 bits R
JTAG VERSION (ID-CODE[31:28])	0xFFFE F002	0xFFFE F003	2 bytes	16 bits R
DieID [63:0]	0xFFFE F004	0xFFFE F00B	8 bytes	16 bits R
Man_pub Key	0xFFFE F00C	0xFFFE F01B	16 bytes	16 bits R
LOCOSTO core configuration	0xFFFE F01C	0xFFFE F03F	36 bytes	16 bits R/W
DieID [127:64]	0xFFFE F040	0xFFFE F047	8 bytes	16 bits R
LOCOSTO version register	0xFFFE F048	0xFFFE F049	2 bytes	16 bits R
Reserved	0xFFFE F04A	0xFFFE F0FF		
I/O configuration registers	0xFFFE F100	0xFFFE F1FF	256 bytes	16 bits R/W
Reserved	0xFFFE F200	0xFFFE FFFF		

Table 6. Peripheral Space Addresses – TIPB Strobe #1 (Continued)

3 Power, Clock, and Reset Sequence on Power-Up

This section details the ClkDRP13 clock generation from the battery insertion event. The Figure below shows the block and timing diagram of the power-up sequence.

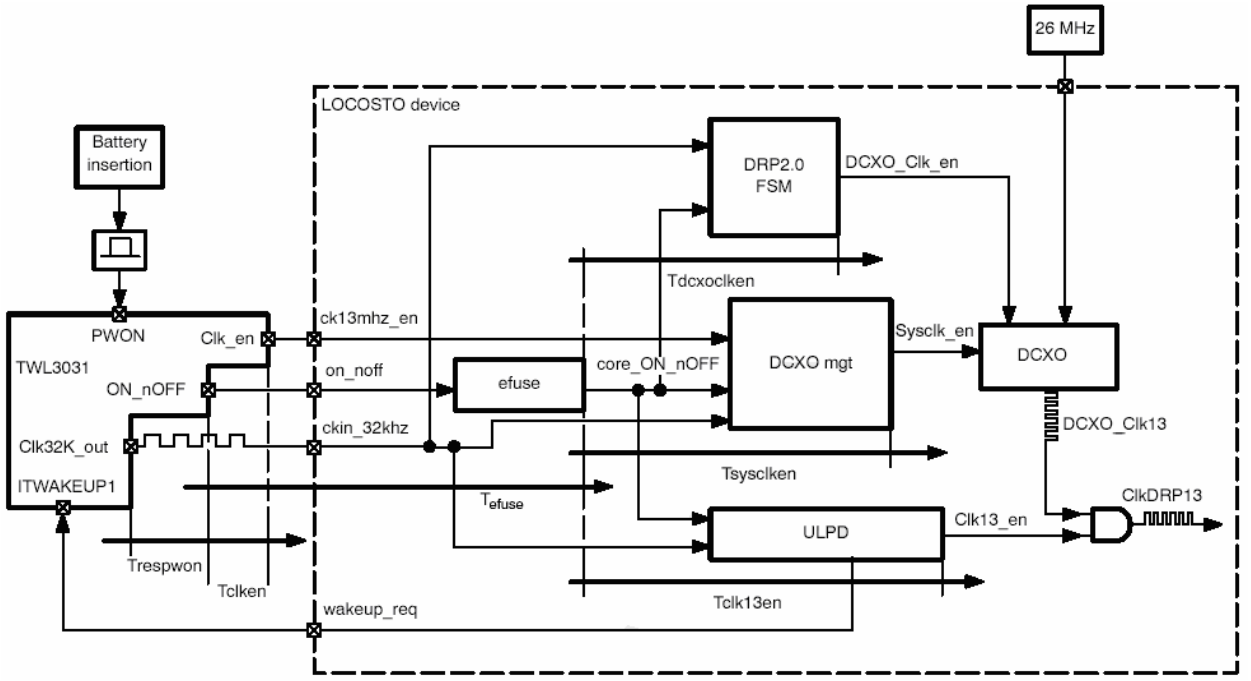

Figure 7. The block and timing diagram of the power-up sequence

The LOCOSTO device power-up is managed by the TWL3031 device in cooperation with the LOCOSTO device power-management finite state-machine (FSM).

The following steps comprise the power up sequence:

- When the main battery is initially plugged in, the TWL3031 device is supplied and clocked after the 32-kHz oscillator start-up phase.
- When the ON/OFF button is pressed, the TWL3031 power-management FSM accepts the start condition as valid only after a stable low-level period of the PWON signal.
- Once a valid switch-on condition is recognized, the TWL3031 FSM provides the correct sequence for power supplies rising up. The VPP, VDD_RST, and VDD_DBB LOCOSTO power domains are supplied first by the VREXTL TWL3031 power domain.
- After the VREXTL supply rail setup time, the VDDR2, VDD_MIF, and VDD_PLL LOCOSTO lower domains are supplied simultaneously by the VREXTH, VRPLL, and VRMMC TWL3031 power outputs.

- After the VREXTH, VRPLL, and VRMMC supply rails setup time, the VDD_IO LOCOSTO power domain is supplied by VRIO TWL3031 power output.
- When the VRIO power domain is stable, the TWL3031 FSM enables the 32-kHz clock to the external world.
- Once the 32-kHz clock is stabilized: after the Trespwon time interval, the FSM releases the system reset, driving the on_noff signal to a high level.
- As a final action, the TWL3031 FSM enables the system clock generation, thus driving the ck13mhz_en signal to a high logical level at Tclken time after reset release.
- On the LOCOSTO device side, after the on_noff signal is driven high by the TWL3031 device, the efuse module releases the core_ON_nOFF signal after a Tefuse delay (see Table 30-1).
- The following three state-machines start in parallel to activate the 13-MHz clock (ClkDRP13):
  - The DCXO management module generates the Sysclk_en signal to the DRP DCXO module following a Tsysclken delay.
  - The DRP2.0 FSM activates the DCXO clock with the DCXO_Clk_en following a Tdcxoclken delay.
  - The ULPD FSM enables the 13-MHz clock by releasing the Clk13_en signal following a Tclk13en delay.

Note: Because the three FSMs run in parallel, the values of the different delays must satisfy a consistent enable of the system clock; that is, the DCXO_Clk_en signal must be released before the Sysclk_en signal, which must be released before the Clk13_en signal.

The figure shows a chronogram of the 10 power-up steps:

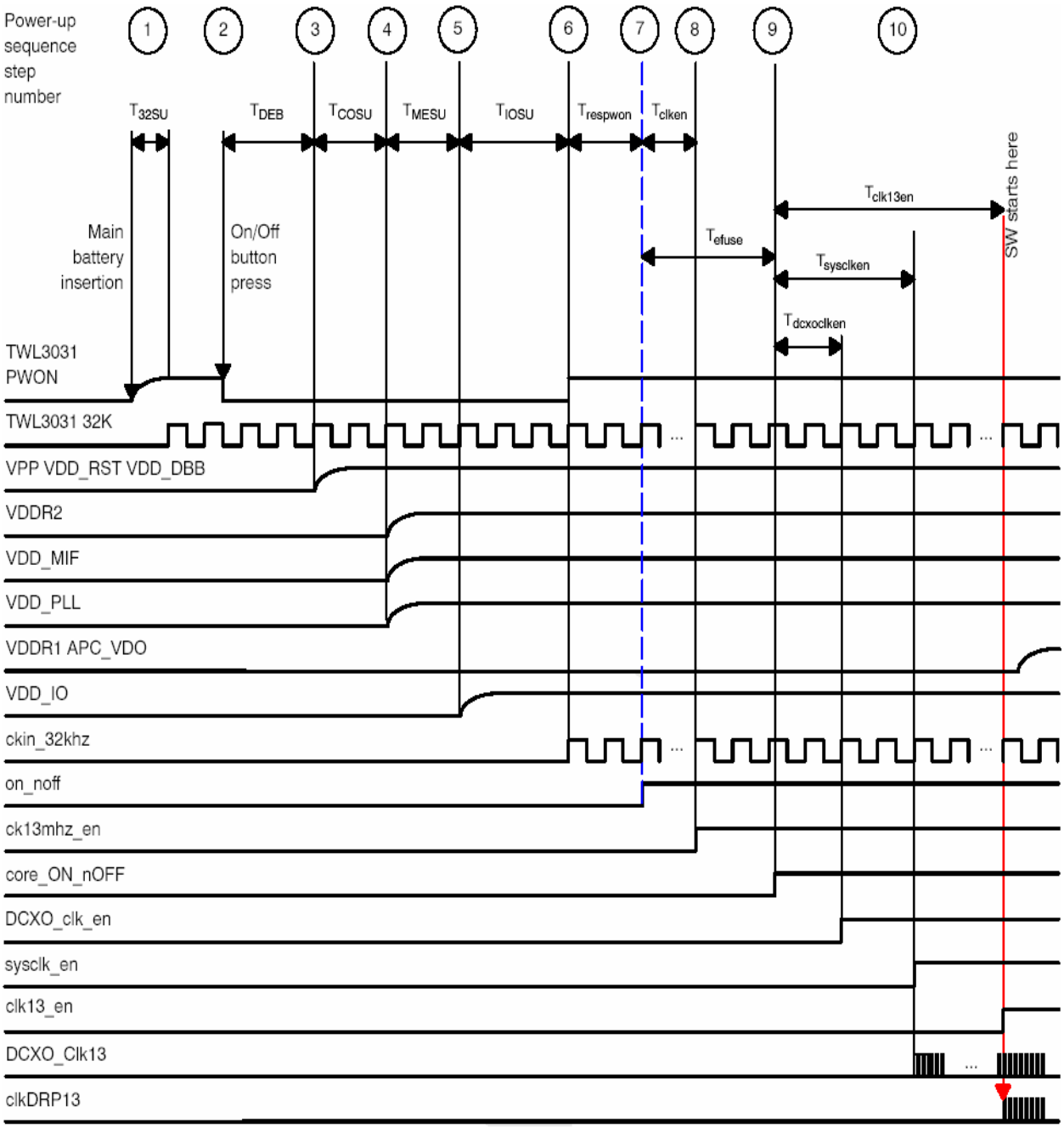

Figure 8. A chronogram of the 10 power-up steps:

The table below describes the delay settings:

Delay	Default Value	Configuration
T32SU	See the *TWL3031 Data Sheet*	
TDEB		
TCOSU		
TIOSU		
TMESU		
Trespwon	7.1411 ms	See the *TWL3031 Data Sheet*
Tclken	19 x 32 khz clock periods (0.5798 ms)	
Tefuse	744 x 32 khz clock periods (22.705 ms)	None
Tdcxoclken	183 x 32 khz clock periods (5.585 ms)	Reset value
Tsysclken	510 x 32 khz clock periods	Reset value
Tclk13en	(15.564 ms) 8253 x 32 khz clock periods (251.86 ms)	Reset value

Table 7. Delay settings

After the voltages and clocks are ramped and stable, the USB_BOOT and nBSCAN pins are sampled and the ROM code is executed.

4 Boot ROM Code Overview

4.1 Introduction:

The main features of the secure BootROM are memory booting and peripheral booting. Memory Booting is to apply an authentication and verification mechanisms on the external Firmware before their execution on the device. Peripheral Booting is downloading code from external host and authenticating it before execution. The downloaded code is usually a flash programmer i.e. used for programming the firmware in the external memory. Memory and Peripheral Booting can be secure or not depending upon whether the device is a High Security device or General Purpose device respectively. Locosto supports peripheral booting over UART (in Modem mode) and USB links.

In addition the BootROM also has secure services, which can be invoked by the firmware to work in a secure environment. Also there are services to perform engineering tests.

4.2 BOOT Application

4.2.1 Device Types

Based on the Hash_ManufacturerPublicKey bits, which are stored in e-fuse, Locosto may operate as a High Secure (HS) Device or a General Purpose (GP) Device.

- **HS Device** is a production device in which all the security features are enabled. In this device Hash_MPK bits e-fused are non-zero. On a HS Device ROM code requires the authentication of the SW (Firmware or Flash Programmer) before their execution. Authentication of the code is handled by routines in the secure ROM.
- **GP Device** is a production device in which all the security features are disabled. (ROM code cannot still be read). In this device Hash_MPK bits are zero. On a GP device doesn't perform authentication on any S/W before execution and any reference to authentication shall be ignored. Emulation is always open.

4.2.2 Architecture of the ROM code

4.2.2.1 High-level overview

The Secure Boot ROM code is split with following features:

- Secure Boot ROM code
  - *Synchronization on UART/USB*: ROM Code supports USB or UART Modem interface for interaction with the host to download the flash programmer code. At reset ROM code polls the interface selected via a GPIO pin, for Signaling Command to be transmitted from PC. If Signaling command is received, proceeds to download the flash programmer code.
  - *Firmware Authentication* (Secure Reset Boot Checker): On the other hand if the signaling command is not detected in the selected interface before the timeout period, ROM Code assumes that a firmware is located at CS3. For HS Device proceeds to authenticate and execute the same if authentication passes. For GP device BootROM directly jumps to the CS3 location (0x06000000).

  - *FLASH Programmer download* (Secure Boot ROM): If signaling command is detected in the selected interface, ROM Code interacts with the host, according to the predefined protocol and then downloads the flash programmer code. *The commands exchanged differ for HS device and GP device.*
  - *FLASH Programmer Authentication*: For HS Device, once the flash programmer is downloaded ROM Code authenticates the code, before handing over control to it. This authentication is similar to the firmware authentication step. For GP device control is immediately handed over to the flash programmer code.

- Secure Services
  - *Secure Run-Time Checker*: This service allows a firmware during run time, to check the firmware certificate. This allows checking the firmware's reliability.
  - *Secure Run-Time Loader*: This service allows to create a *certificate* of a piece of code or data, in order to avoid them being cloned in another mobile.
  - *Secure Run-Time Platform Data Checker*: This service provides protection against cloning by providing the application secure means to verify the unique association of the data/code to the hardware platform

- **Services to perform engineering tests:** These are services that are used to validate the Boot ROM code and also other codes that are used for initial system level validation.
- **Re-mapping of interrupt vectors:** This part of the code re-maps the interrupt vectors (which by default map to Boot-ROM) to a predefined location in the Internal SRAM

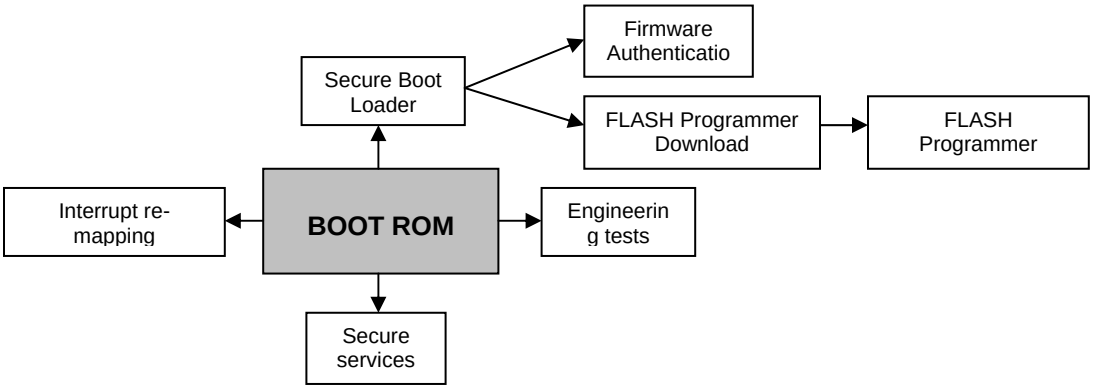

Figure 9. ROM Code Features

4.3 Boot ROM Code Sequence:

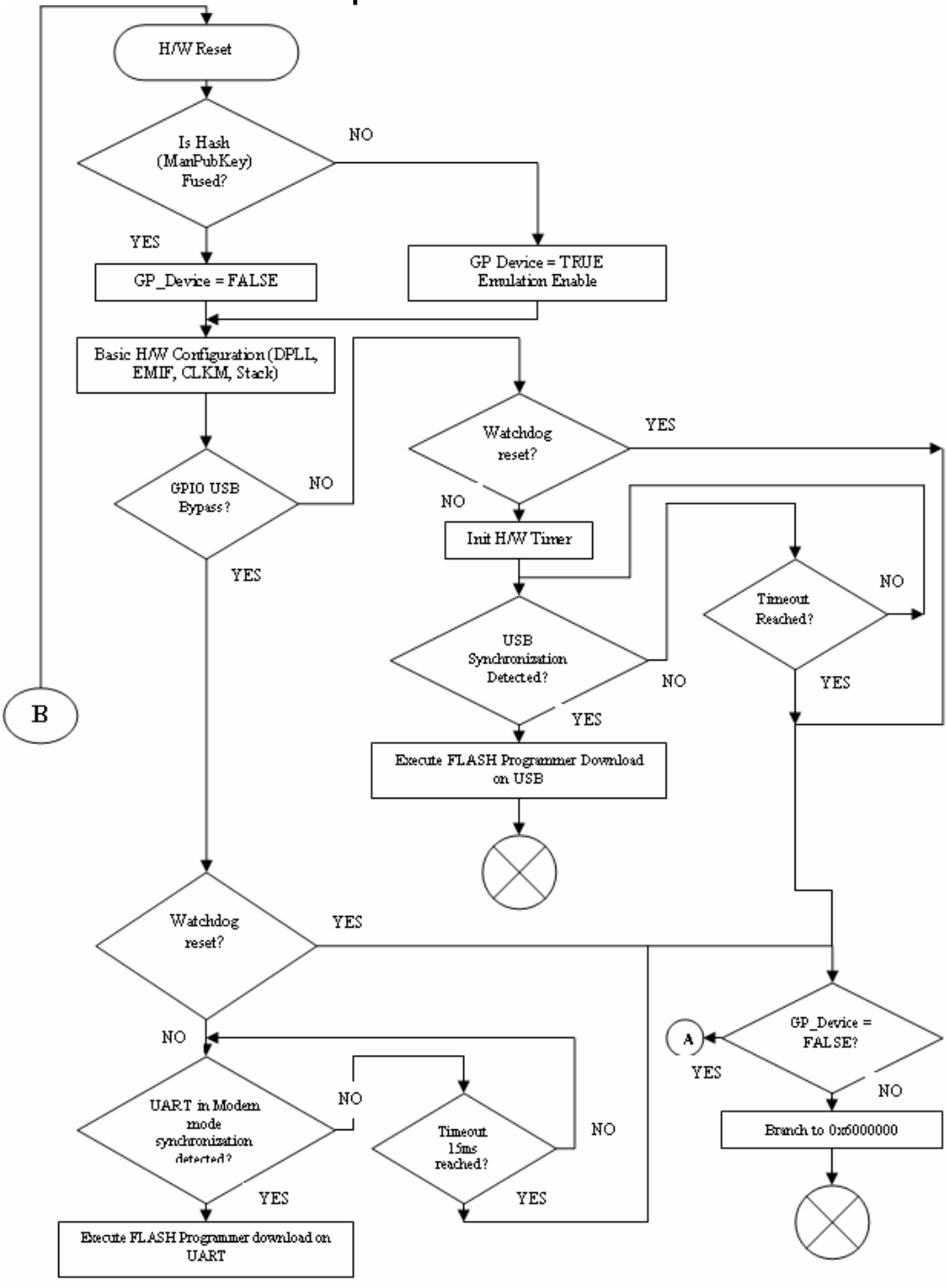

Figure 10. Boot ROM Code Sequence

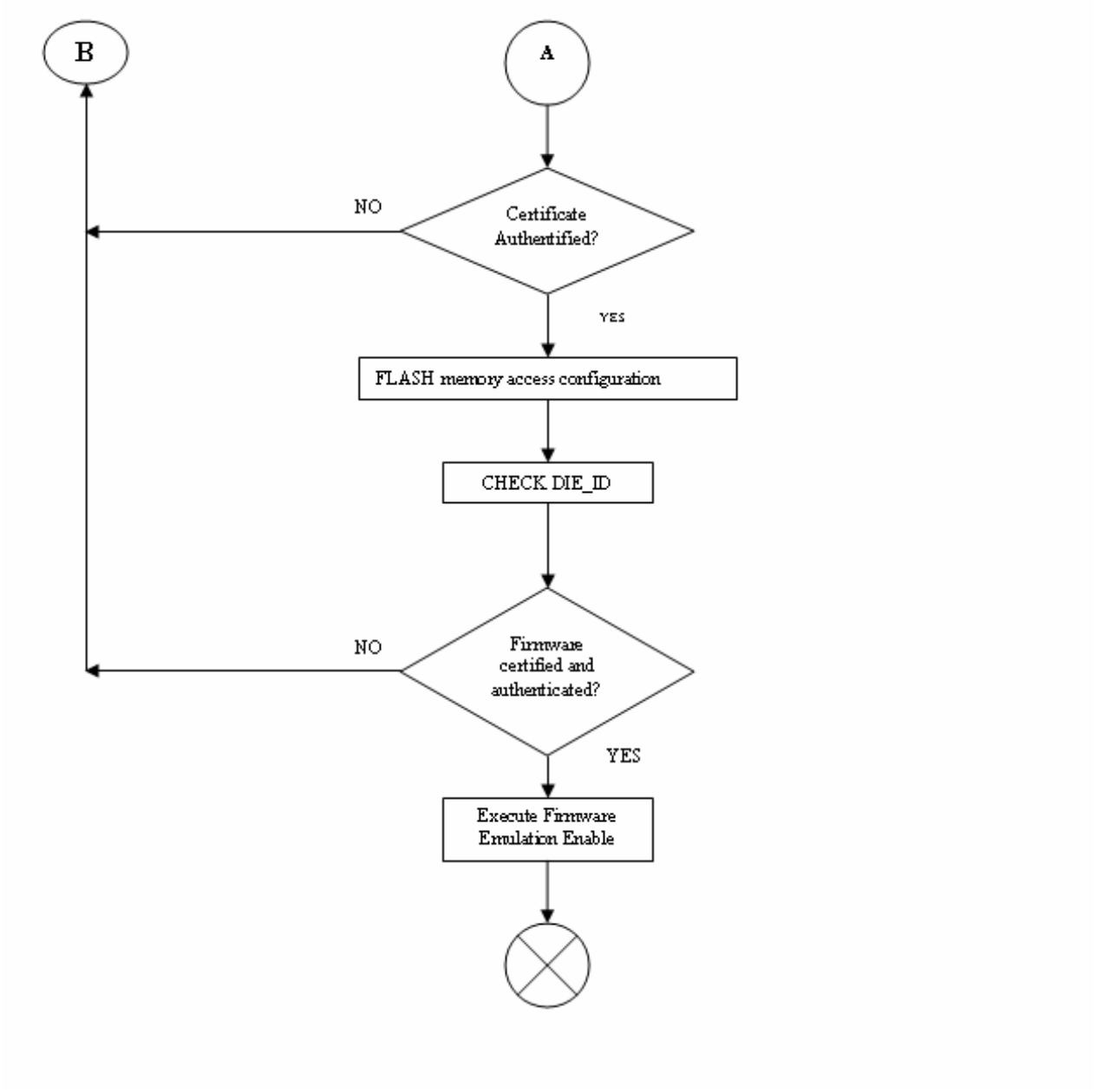

Figure 11. Boot ROM Code Sequence

4.3.1 Boot ROM Code sequence:

- Whatever the source of the reset (Power-on reset, ON/OFF or watchdog reset), the mobile always starts the code execution in the Secure Boot ROM. This step can not be bypassed by any means and is the base of the security. The JTAG port is always disabled whatever the reset source, which implies that a hacker cannot plug an emulator to bypass the Boot sequence or to dump sensitive data located in Internal SRAM. The H/W manages the disable of the JTAG port.

  For HS Device the enable of the emulation JTAG port by the ROM code is always the last operation to be done after authentication and integry check, and before giving the hand to the firmware or the flash programmer. Also the enable is conditionned by the value of the DEBUG parameter in the certificate and Die-ID field.

- For GP Device the enable of the emulation is done at the starting of the Boot-ROM code.

- As soon as the first code execution, the Secure Boot ROM checks the content of the Hash(ManPubKey) fused on the chip, based on which it finds whether the device is a GP Device or a HS Device. Acoordingly the GP_Device is set to FALSE or TRUE.

- The next step for Secure Boot ROM is to perform basic Initialization like configuration of the DPLL and the ARM clock. The ARM clock frequency is provided from DPLL, which is configured at 104MHz. DPLL receives a VTCXO input frequency of 13MHz.

- In parallel, the PRRM module erases the content of the Internal SRAM (first 2Mbits) and the API memory. This module uses directly the ARM clock, which is configured at 52MHz.

- All data managed by the Secure Boot ROM are located in the upper 1.5Mbits (0802:0000 to 0805:0000) of the Internal SRAM. .

- After optimal H/W configuration, the Secure Boot ROM checks the reset source.
  - In case of watchdog reset (CNTL_RST[3] bit at ‘1’), the Secure Boot ROM bypasses the synchronisation on the selected interface in order to reduce the impact on the user. Moreover, on watchdog reset occurrence, the PRRM can be configured by the manufacturer in order to not reset the content of the Internal SRAM. For example, the configuration could be to set FULL_RST_EN bit to ‘0’ and PRRM_INT_START and PRRM_INT_END bits to 0x0000 in the PRRM module.

    The Secure Boot ROM does not reset the bit CNTL_RST[3], which lets the reset source information to the Firmware. Thanks to this mechanism, the manufacturer can put in place a fast recovery mechanism in order to not disturb the user.
  - In other cases, the GPIO pin is read to see which interface is selected for Peripheral Booting. Accoridingly GPIO_USB_Bypass is set. Based on this setting USB or UART is configured. UART is configured in Modem mode and USB is configured in Full Speed mode.

- A S/W timer is started and the Secure Boot ROM waits a synchronisation from an external PC. The selected interfaces is polled sequentially. The S/W timeout is fixed to 15ms in case of UART which allows to receive 15 signalling commands at 19200 bps. The timeout is fixed to 3s in case of USB. In case of USB the timeout is stopped as soon as the host moves the USB to the configured state as this indicates the presence of an active host. And also to avoid a dead-lock condition whereby the host moves the usb peripheral to the configured state and the device

is reset before the signalling command is received. Then the host has to send the signalling command and start further communication.

- In case synchronisation on the selected interface is detected before the timeout, the Secure Boot ROM enters in the sequence to download a FLASH Programmer.Once the code is downloaded JTAG is enabled as per the following condition.
  - Debug Request Field in the certificate needs to be enabled AND
  - Die-ID field in certificate should be zero OR should match the Die-ID of the target Locosto device.
- In case of the timeout is reached (or watchdog reset), the Secure Boot ROM starts Memory Booting. For GP Device Boot ROM jumps and starts execution from **nCS3**. For HS Device a sequence of authentication and integrity verification of the Firmware located in external FLASH. **This Firmware must be located on the CS3,** which is the chip-select that is always enabled on Locosto, **and the certificate must be located at the first address of the CS3**.
- In case of success of the certificate authentication and integrity check, the Secure Boot ROM configures the external memory interface according to the parameters contained in the Firmware's certificate. This configuration is done in order to speed-up the authentication phase, which consists to hash the Firmware. The configurations are the Burst Mode capability, the number of wait-state and dummy cycle to access the FLASH memory at 52MHz.
- After H/W configuration, the DIE_ID field of the certificate and the DIE_ID registers are compared:
  - If the DIE_ID field value is equal to 0, the Manufacturer certificate is not uniquely associated to the given hardware platform.
  - If the DIE_ID field value is equal to the DIE_ID registers value, the subsequent usage of PLATFORM_DATA field value is valid. The Firmware is informed that it can securly use the PLATFORM_DATA field value after successful firmware authentication and integrity check.
  - If the DIE_ID field value is **NOT** equal to the DIE_ID registers value, the subsequent usage of PLATFORM_DATA field value is not recommended, because there are not associated to the given hardware platform. Nevertheless, the Firmware is just informed about it and will be executed after succesfull authentication and integrity check.
- After Die-ID comparison, JTAG emulation is enabled only if both
  - Debug Request is enabled in the certificate AND
  - **Die-ID in certificate matches the Die-ID** in target Locosto Device.(NOTE: A firmware certificate with Blank DieID will not enable emulation).
- In case of successful firwmare authentication and integrity check, the Firmware is executed with a parameter representing the status of the DIE_ID field check. The parameter is located in R0 register of the ARM processor with the following signification:
  - DIE_ID field value is equal to the DIE_ID registers: R0 = 1
  - DIE_ID field value is NOT equal to the DIE_ID registers: R0 = 4
  - DIE_ID field value is equal to 0: R0 = 5

The address of the entry point of the Firmware is located in the certificate.

- In case of error of the certification or authentication verifications, the Secure Boot ROM performs a S/W reset in order to restart the boot sequence.

4.4 Basic Hardware Initializations

4.4.1 Reset Considerations

On Reset, Boot-ROM reads Hash (MPK) to detect whether the device is a GP Device or HS Device. Then it reads the GPIO pin to detect the selected interface for peripheral booting – USB or UART. For improving reliability and because of implementation GPIO Pin needs to be kept the configured level for at least 750 μs from starting point.

4.4.2 Clock configuration

The H/W is configured during the first step of the boot application:

- DPLL at 104MHz;
- ARM uses DPLL
- RHEA Access Factor:
  - 0 on strobe 0
  - 0 on strobe 1
- API wait state:
  - 0WS in HOM
  - 6WS in SAM
- EMIF Configuration – BootROM writes the relevant EMIF registers like emif_conf_reg, emif_conf_reg_cs3, etc to its reset value to be safe. But configured after Firmware Certificate authentication phase with values programmed in the certificate.
- Disable of Watchdog timer and Secure Watchdog timer.
- Initialize RNG module.
- Initialize the timer1 module based on selected interface

Note: At reset, ARM interrupts are masked in the ARM CPSR register, then it is neither mandatory to configure CPSR, nor to mask interrupt in the Interrupt Handler module.

4.4.3 Interface configuration

The interfaces used in the Secure Boot Loader are:

Physical interface	Mode
UART	MODEM
USB	Full Speed (12 Mbps)

Table 8. Physical interface

If UART is selected, UART is configured with the following default value:

- Baud rate: 19200 bps,
- 8 bits, no parity, 1 stop bit

As the UART module is shared between MCU & DSP, The SSW switch needs to be configured to give MCU the access. Also the CLKM module needs to be configured to give the UART module a 48 MHZ clock.

4.5 Memory mapping

The first 1Mbits of the internal SRAM and the full API memory are not used. This implies that the boot application data are located in the upper 1.5Mbits of internal SRAM and in the Secure RAM.

The first 1Mbits are reserved to download the FLASH Programmer, which is allowed after synchronization on the UART/USB interface. In this case, the duration is less critical and it is acceptable to have the ARM in waits to finish the erase of the 2.5Mbits.

Address	Function	Code	Location
0000:0000	IT Vectors	B _SecureBoot ADD PC, PC, #0x08000000 ADD PC, PC, #0x08000000 ADD PC, PC, #0x08000000 ADD PC, PC, #0x08000000 ADD PC, PC, #0x08000000 ADD PC, PC, #0x08000000 ADD PC, PC, #0x08000000	BOOT ROM CODE
0000:0020	TEST vector	MOV PC, #0x84	
0000:0024	Service vector	MOV PC, #0x88	
0000:0028	Reserved for future management	BX LR	
0000:0060	Secure TEST vector	MOV PC, #0x84	
0000:0064	Secure Service vector	MOV PC, #0x88	
0000:0068	Reserved for future management	BX LR	
0000:0080	ROM code identifier	0x00000500	
0000:0084	Dynamic redirection to test subroutine	B _ROM_TestsSubroutine	
0000:0088	Service routine management		
0000:00XX	ROM code	…	

0600:0000	Firmware certificate	…	CS3
0600:0XXX	Code	Firmware	
0600:0XXX	Data		
0600:0XXX	*Free*	…	

0800:0000	*Free*	…	Internal SRAM (1Mbits)
0800:000C	Indirect IT Vectors (not initialised by the ROM code)	LDR PC, [PC, #0x14] LDR PC, [PC, #0x14] LDR PC, [PC, #0x14] LDR PC, [PC, #0x14] LDR PC, [PC, #0x14] LDR PC, [PC, #0x14] LDR PC, [PC, #0x14] Note: the offset 0x14 takes into account the ARM pipeline: 0x14=7*4-8	
0800:0028	Address of interrupt subroutine (not initialised by the ROM code)	Undefined subroutine address SWI subroutine address Abort Prefetch subroutine address Abort Data subroutine address Reserved subroutine address IRQ subroutine address FIQ subroutine address	
0800:0044 to 0802:0000	Reserved to download FLASH Programmer	…	
0802:0000 to 0804:0000	Used by ROM code	…	Internal SRAM (1Mbits)
0804:0000 to 0800:4FFF	Not used. This is where BootROM can be remapped in case of un-protected devcies	Not used	Internal SRAM (0.5Mbits)
09B0:0000 to 09B0:0200	Used by ROM code	…	Secure RAM (512 bytes)

Table 9. Memory mapping

5 Control Flow of the Bootstrap Loader

Whatever the source of the reset (Power-on reset, ON/OFF or watchdog reset), the mobile always starts the code execution in the Secure Boot ROM Code. This step can not be bypassed by any means and is the base of the security. The JTAG port is always disabled whatever the reset source. For HS Device the enable of the emulation JTAG port by the ROM code is always the last operation to be done after authentication and integry check, and before giving the hand to the firmware or the flash programmer..For GP Device the enable of the emulation is done at the starting of the Boot-ROM code.

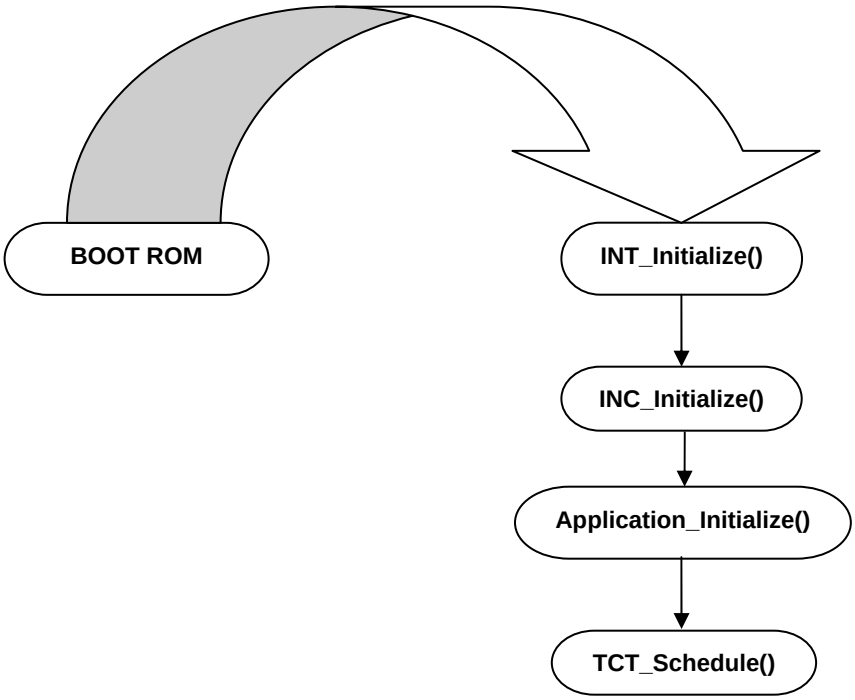

Figure 12. Control Flow

The First function called in the firmware is INT_Initialize().This function sets up the global system stack variable and transfers control to the target independent initialization function INC_Initialize.

INC_Initialize is the main initialization function of the system. All components are initialized by this function. After system initialization is complete, the Application_Initialize routine is called. After all initialization is complete, this function calls TCT_Schedule to start scheduling tasks.

TCT_Schedule waits for a thread to become ready. Once a thread is ready, this function initiates a transfer of control to that thread.

5.1 Flow chart of various functions in the Bootstrap loader:

5.1.1 INT_Initialize()

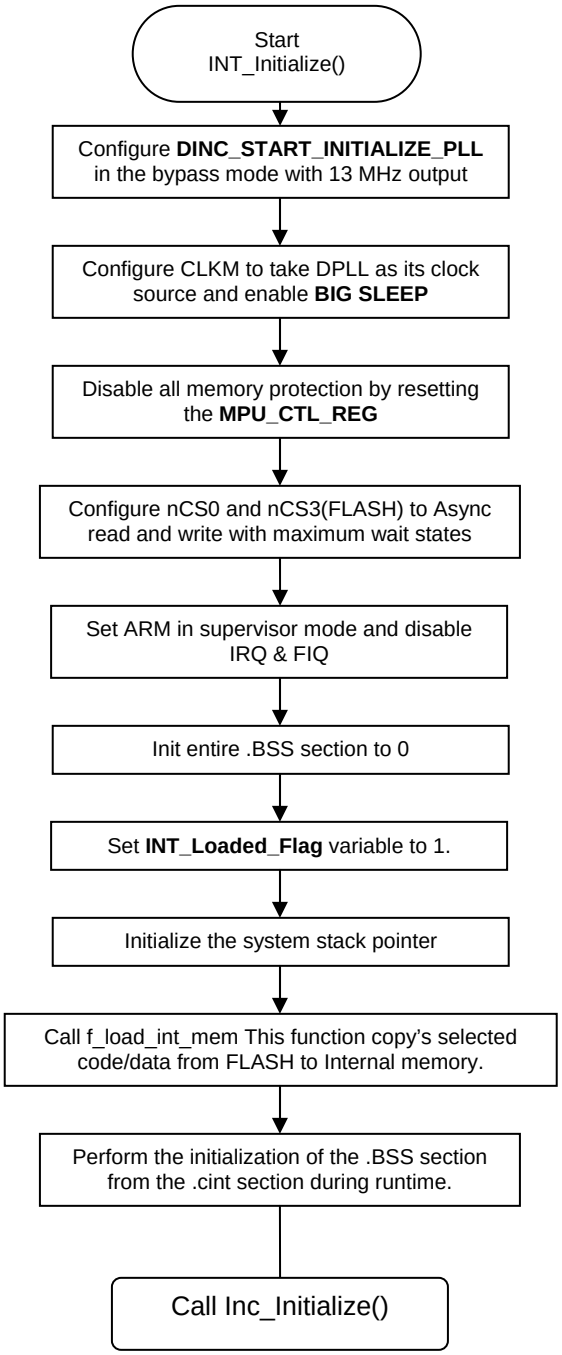

Figure 13. Flow Chart of INT_Initialize

5.1.2 INC_initialize()

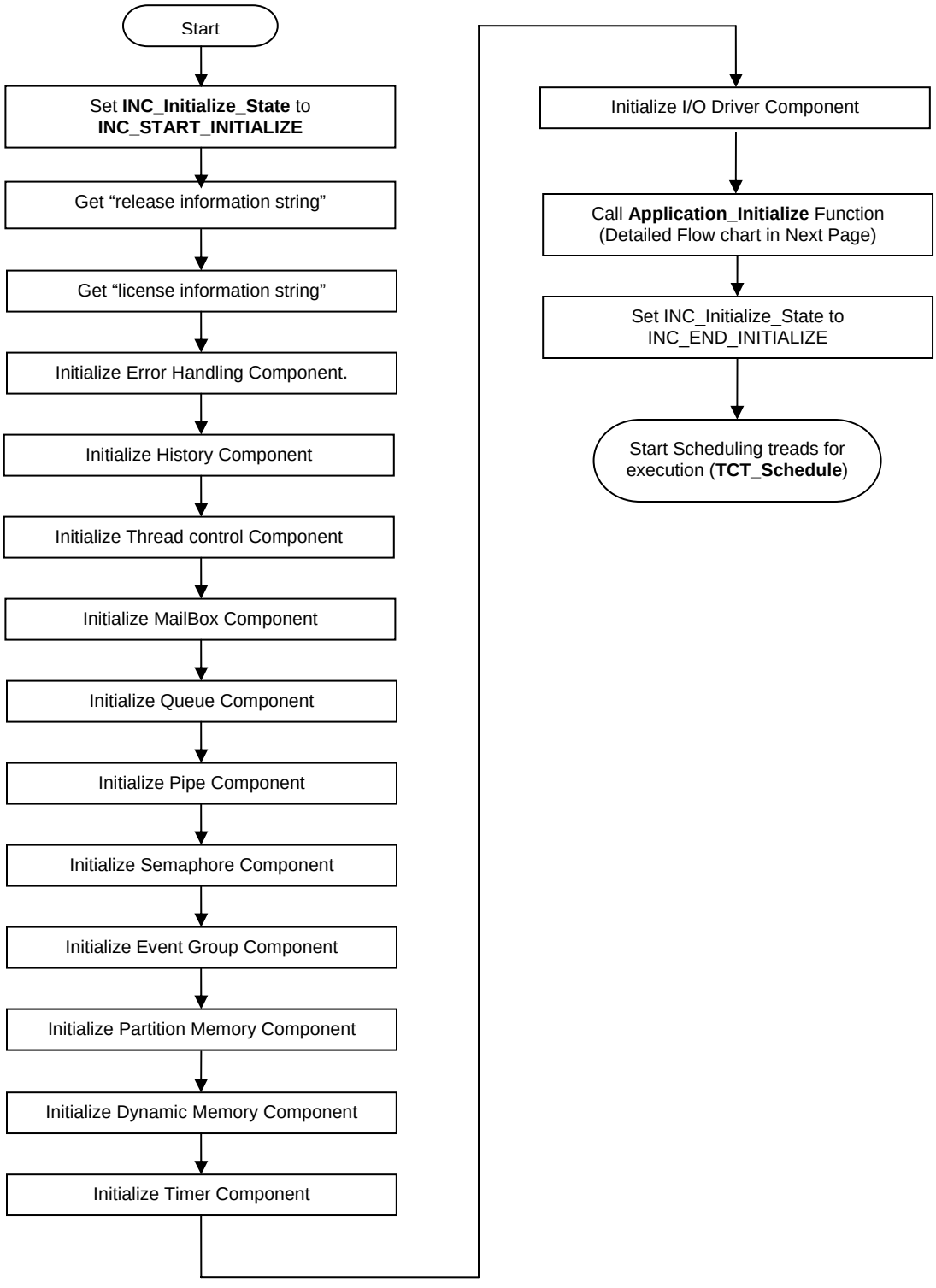

Figure 14. Flow Chart of INC_Initialize

5.1.3 Flow of Application_Initialize()

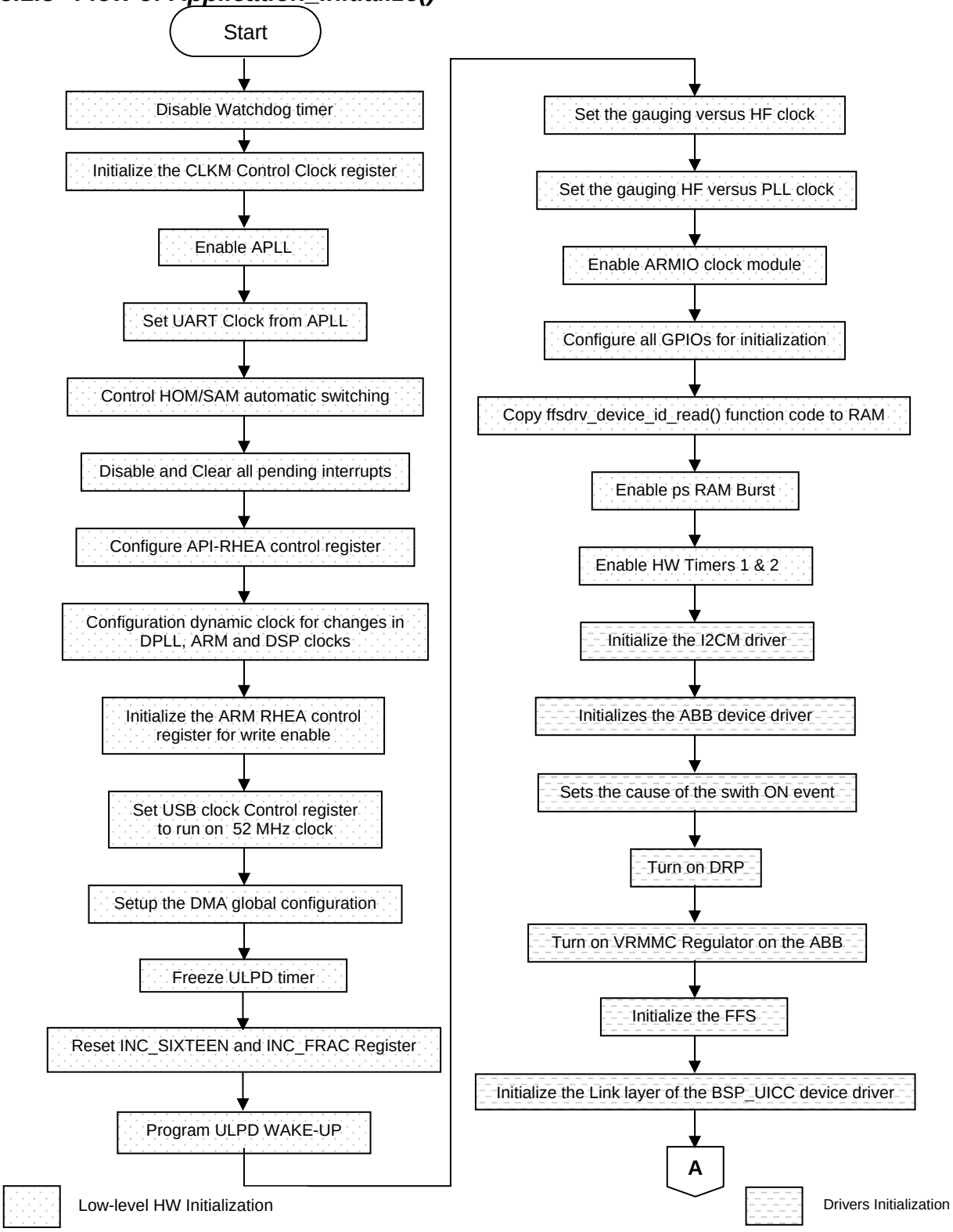

Figure 15. Flow Chart of Application_Initialize

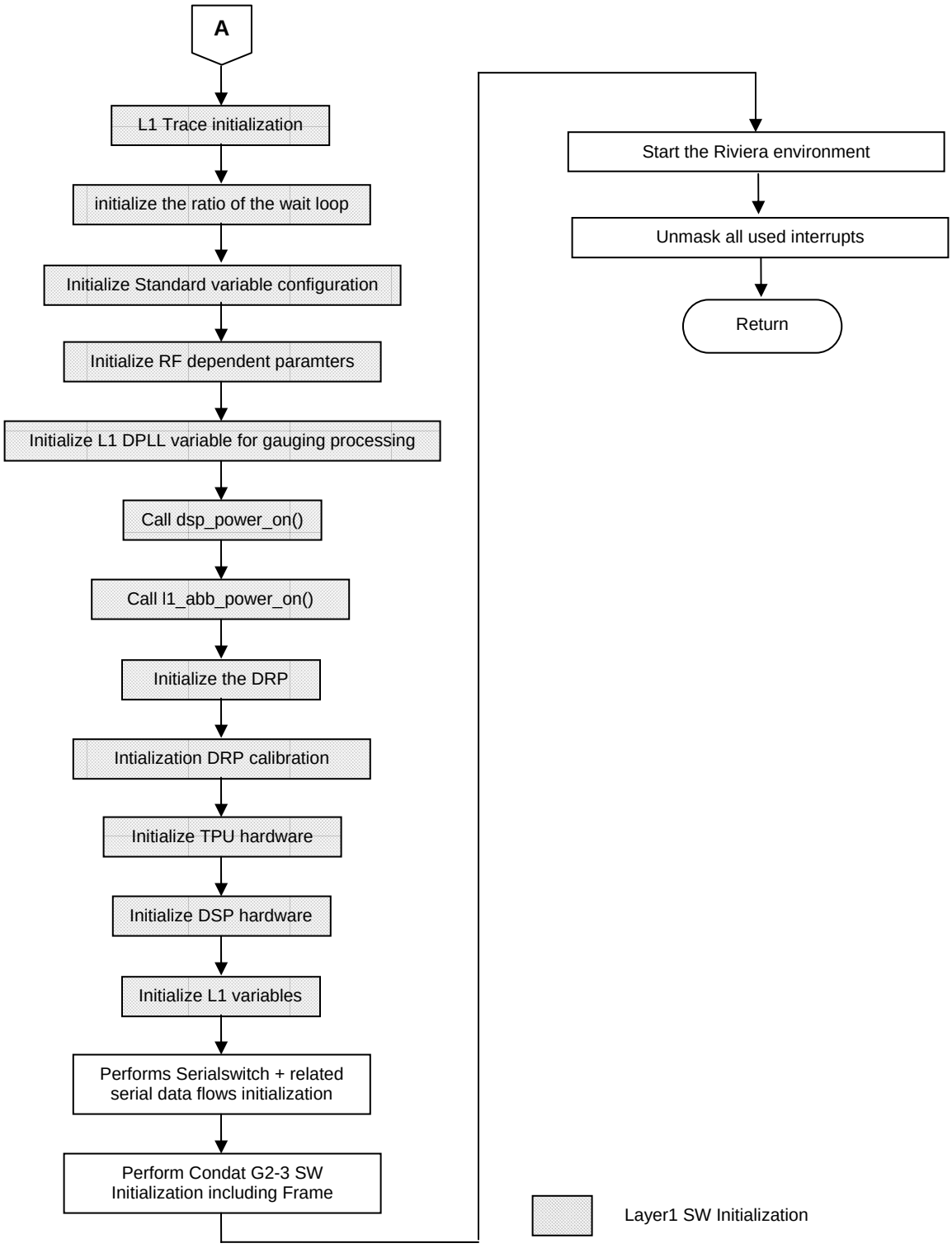

Figure 16. Flow Chart of Application_Initialize (Cond.)

5.2 Function Descriptions

5.2.1 INT_Initialize

Function:	INT_Initialize
Path:	chipsetsw\system\init_common\int.s
Description:	This function sets up the global system stack variable and transfers control to the target independent initialization function INC_Initialize. Responsibilities of this function include the following: - Setup necessary processor/system control registers - Initialize the vector table - Setup the system stack pointers - Setup the timer interrupt - Calculate the timer HISR stack and priority - Calculate the first available memory address - Transfer control to INC_Initialize to initialize all of the system components.

5.2.2 INC_Initialize

Function:	INC_Initialize
Path:	chipsetsw\os\nucleus\inc.c
Description:	This function is the main initialization function of the system. All components are initialized by this function. After system initialization is complete, the Application_Initialize routine is called. After all initialization is complete, this function calls TCT_Schedule to start scheduling tasks. The functions called are: Application_Initialize : Application initialize RLC_Release_Information : Release information LIC_License_Information : License information ERI_Initialize : Error handling initialize HII_Initialize : History initialization TCI_Initialize : Thread control initialize MBI_Initialize : Mailbox initialize QUI_Initialize : Queue initialize PII_Initialize : Pipe initialize SMI_Initialize : Semaphore initialize EVI_Initialize : Event flag initialize PMI_Initialize : Partition memory initialize DMI_Initialize : Dynamic memory initialize TMI_Initialize : Timer initialize IOI_Initialize : I/O Driver initialize TCT_Schedule : Thread scheduling loop

5.2.3 RLC_Release_Information

Function:	RLC_Release_Information
Path:	chipsetsw\os\nucleus\rlc.c
Description:	This function returns a pointer to the release information string. The information string identifies the current version of Nucleus PLUS.

5.2.4 LIC_License_Information

Function:	LIC_License_Information
Path:	chipsetsw\os\nucleus\lic.c
Description:	This function returns a pointer to the license information string. The information string identifies the customer and product line Nucleus PLUS is licensed for.

5.2.5 ERI_Initialize

Function:	ERI_Initialize
Path:	chipsetsw\os\nucleus\eri.c
Description:	This function initializes the data structures of the Error management component (ER).

5.2.6 HII_Initialize

Function:	HII_Initialize
Path:	chipsetsw\os\nucleus\hii.c
Description:	This function initializes the data structures that control the operation of the History component (HI).

5.2.7 TCI_Initialize

Function:	TCI_Initialize
Path:	chipsetsw\os\nucleus\tci.c
Description:	This function initializes the data structures that control the operation of the Thread Ccontrol component. The system is initialized as idle.

5.2.8 MBI_Initialize

Function:	MBI_Initialize
Path:	chipsetsw\os\nucleus\mib.c
Description:	This function initializes the data structures that control the operation of the Mailbox component (MB). There are no mailboxes initially.

5.2.9 QUI_Initialize

Function:	QUI_Initialize
Path:	chipsetsw\os\nucleus\qui.c
Description:	This function initializes the data structures that control the operation of the Queue component (QU). There are no queues initially.

5.2.10 PII_Initialize

Function:	PII_Initialize
Path:	chipsetsw\os\nucleus\pii.c
Description:	This function initializes the data structures that control the operation of the Pipe component (PI). There are no pipes initially

5.2.11 SMI_Initialize

Function:	SMI_Initialize
Path:	chipsetsw\os\nucleus\smi.c
Description:	This function initializes the data structures that control the operation of the Semaphore component (SM). There are no semaphores initially.

5.2.12 EVI_Initialize

Function:	EVI_Initialize
Path:	chipsetsw\os\nucleus\evi.c
Description:	This function initializes the data structures that control the operation of the Event Group component (EV). There are no event groups initially.

5.2.13 PMI_Initialize

Function:	PMI_Initialize
Path:	chipsetsw\os\nucleus\pmi.c
Description:	This function initializes the data structures that control the operation of the Partition Memory component (PM). There are no partition pools initially.

5.2.14 DMI_Initialize

Function:	DMI_Initialize
Path:	chipsetsw\os\nucleus\dmi.c
Description:	This function initializes the data structures that control the operation of the Dynamic Memory component (DM). There are no dynamic memory pools initially.

5.2.15 TMI_Initialize

Function:	TMI_Initialize
Path:	chipsetsw\os\nucleus\tmi.c
Description:	This function initializes the data structures that control the operation of the timer component (TM). There are no application timers created initially.

5.2.16 IOI_Initialize

Function:	IOI_Initialize
Path:	chipsetsw\os\nucleus\ ioi.c
Description:	This function initializes the data structures that control the operation of the I/O driver component (IO). There are no I/O drivers initially.

5.2.17 Application_Initialize

Function:	Application_Initialize
Path:	chipsetsw\system\init_common\main.c
Description:	Initialization Main function, called by Nucleus (INC_Initialize) This function calls Init_Target() Init_Drivers() Cust_Init_Layer1() Init_Serial_Flows() StartFrame() rv_start() Init_Unmask_IT()

5.2.18 TCT_Schedule

Function:	TCT_Schedule
Path:	chipsetsw\os\nucleus\tct.c
Description:	This function waits for a thread to become ready. Once a thread is ready, this function initiates a transfer of control to that thread.

5.2.19 Init_Target

Function:	Init_Target
Path:	chipsetsw\system\init_common\init.c
Description:	Performs low-level HW Initialization.

5.2.20 Init_Drivers

Function:	Init_Drivers
Path:	chipsetsw\system\init_common\init.c
Description:	Performs Drivers Initialization.

5.2.21 Cust_Init_Layer1

Function:	Cust_Init_Layer1
Path:	FILE not included in realease
Description:	Layer1 SW Initialization.

5.2.22 Init_Serial_Flows

Function:	Init_Serial_Flows
Path:	chipsetsw\system\init_common\init.c
Description:	Performs Serialswitch and related serial data flows initialization.

5.2.23 StartFrame

Function:	StartFrame
Path:	*FILE not included in realease*
Description:	Condat G23 SW Initialization including GPF Frame.

5.2.24 rv_start

Function:	rv_start
Path:	chipsetsw\system\init_app\create_RVtasks.c
Description:	This function is called by the RV_START task. It starts the Riviera environment and the TRACE task. This start must be done after Application_initialize().

5.2.25 Init_Unmask_IT

Function:	Init_Unmask_IT
Path:	chipsetsw\system\init_common\init.c
Description:	Unmask all used interrupts.

IMPORTANT NOTICE

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

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

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

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

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

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

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

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

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