KeyStone Architecture Bootloader

User Guide

Literature Number: SPRUGY5B
June 2012

Release History

Release	Date	Description/Comments
SPRUGY5B	June 2012	• Added and corrected details describing the Ethernet boot process. (Page 3-5) • Added and corrected details describing the I2C boot process. (Page 3-12) • Added and corrected details describing the PCIeboot process. (Page 3-10) • Added more details describing the SRIO boot process. (Page 3-3) • Added or modified some statements describing the boot process (Page 2-2) • Added details about the boot process (Page 1-2) • Added some new terms and abbreviations. (Page 1-3) • Added the abbreviation RBL to differentiate from Intermediate Boot Loader (IBL) (Page 1-1) • Changed all the reference of bootloader to RBL. (Page 1-2) • Modified the wordings to improve clarity. (Page 1-2)
SPRUGY5A	September 2011	• Added the I2C Slave Mode Device configuration table (Page 3-18) • Modified the I2C Master Mode Device Configuration tables to match the correct device configuration (Page 3-12) • Modified the byte offsets (Page 2-3) • Moved memory map information to its respective device-specific data manual (Page 2-3) • Added the interrupt control procedure used in the PCIe boot process (Page 3-11) • Changed device configuration tables for various boot modes to accommodate the structure defined in the ROM code (Page 2-2) • Corrected the secondary core boot process (Page 2-3) • Modified the general structure and clarified the description of bootloader initialization after hard reset and soft reset (Page 2-6)
SPRUGY5	November 2010	Initial Release

Contents

List of Tables

List of Figures

Preface

About This Manual

This document describes the features of the on-chip bootloader provided with C66x_Digital Signal Processors (DSP).

This document should be used in conjunction with the device-specific data manuals and user guides for peripherals used during the boot. This document supports only non-secure boot mode.

Notational Conventions

This document uses the following conventions:

- Commands and keywords are in **boldface** font.
- Arguments for which you supply values are in *italic* font.
- Terminal sessions and information the system displays are in `screen font`.
- Information you must enter is in **`boldface screen font`**.
- Elements in square brackets ([ ]) are optional.

Notes use the following conventions:

Note—Means reader take note. Notes contain helpful suggestions or references to material not covered in the publication.

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

CAUTION—Indicates the possibility of service interruption if precautions are not taken.

WARNING—Indicates the possibility of damage to equipment if precautions are not taken.

Related Documentation from Texas Instruments

C66x CorePac User Guide	SPRUGW0
DDR3 Memory Controller for KeyStone Devices User Guide	SPRUGV8
External Memory Interface (EMIF16) for KeyStone Devices User Guide	SPRUGZ3
HyperLink for KeyStone Devices User Guide	SPRUGW8
Inter Integrated Circuit (I^2C) for KeyStone Devices User Guide	SPRUGV3
Multicore Shared Memory Controller (MSMC) for KeyStone Devices User Guide	SPRUGW7
Peripheral Component Interconnect Express (PCIe) for KeyStone Devices User Guide	SPRUGS6
Phase Locked Loop (PLL) Controller for KeyStone Devices User Guide	SPRUGV2
Power Sleep Controller (PSC) for KeyStone Devices User Guide	SPRUGV4
Serial Peripheral Interface (SPI) for KeyStone Devices User Guide	SPRUGP2
Serial RapidIO (SRIO) for KeyStone Devices User Guide	SPRUGW1

Trademarks

Chapter 1

Introduction

IMPORTANT NOTE—The information in this document should be used in conjunction with information in the device-specific Keystone Architecture data manual that applies to the part number of your device.

This document describes the features of the on-chip ROM Boot Loader (RBL) provided with C66x Digital Signal Processors (DSP).

This document should be used in conjunction with the device-specific data manuals and user guides for peripherals used during the boot. This document applies to non-secure boot mode only.

1.1 Bootloader Features

The ROM Boot Loader (RBL) is DSP code that transfers application code from slow non-volatile external memory or from an external host into high-speed internal memory for execution after the DSP is taken out of reset. The RBL is permanently stored in the internal ROM of the DSP starting at the base address of ROM 0x20B00000.

The boot process can be broadly divided into host boot and memory boot. In memory boot the boot process loads the initial application code from an external memory to the internal memory to execute. In case of the host mode, the boot process configures the DSP in slave mode and wait for the code to be transferred by an external host and kick the start the DSP to execute the application code.

To accommodate different system requirements, the RBL offers several boot modes to execute the boot process. The different boot modes and their operations are listed below. See the device-specific data manual to identify the list of peripherals supported by that device.

- **EMIF16 boot:** The user's application is executed from the external asynchronous memory through a 16-bit interface.
- **Serial Rapid IO (SRIO) boot:** An external host loads the user's application through the SRIO peripheral either in messaging mode or in directIO mode.
- **Ethernet boot:** An external host loads the user's application through an Ethernet peripheral with the packet accelerator driven by either the core reference clock or the SerDes reference clock.
- **PCIe boot:** The external host loads the user's application into the on-chip memory through PCI Express.
- **Master I^2C boot:** The user's application is loaded in blocks of data in boot table format from an I^2C EEPROM.
- **Slave I^2C boot:** An external I^2C master sends the user's application to the DSP in the form of a boot table. This mode is typically used to support single I^2C EEPROM with multiple DSPs to minimize the boot time.
- **SPI boot:** The user's application is loaded in blocks of data in boot table format from a flash memory device connected through Serial Port Interface.
- **HyperLink boot:** This is a passive boot mode where the host takes the responsibility of configuring the memory and loading the user's application directly to boot the DSPs.

The boot process is determined by two factors. one is the type of reset triggered the boot process and the other is the configuration that is set through the boot strap pins. The effects of these factors are discussed in the next chapter.

1.2 Terms and Abbreviations

Term	Definition
I²C	Inter-Integrated Circuit
MSMC	Multicore Shared Memory Controller
PCIe	Peripheral Component Interconnect Express
POR	Power on Reset
RBL	ROM Boot Loader
SPI	Serial Peripheral Interface
SRIO	Serial Rapid Input/Output

Chapter 2

Reset Types and Boot Configurations

2.1 Introduction

As mentioned in the previous chapter, the boot process is determined by two factors. While the boot process is triggered by a device reset, the boot process flow is determined by the type of reset asserted and the boot configuration set by the bootstrap pins. The boot process is executed by CorePac0. There are four types of reset supported in the KeyStone architecture:

- Power-on reset (POR)
- Hard Reset (Chip 0 Reset + Chip 1 Reset)
- Soft Reset (Chip 1 Reset)
- Local Reset

The first three types of reset are grouped as global resets because they affect the entire device. For local reset, the boot process is not triggered.

2.2 Bootloader Initialization After Power on Reset

Power-on reset can be initiated by either POR pin or RESETFULL pin. While POR pin is asserted during the power up sequence, the RESETFULL is asserted by a host to reset the entire device. In the case of RESETFULL pin assertion, it is assumed that the device is already powered up. All the peripherals on the device are reset to their default states and initiate the boot process. During the Power on Reset sequence the 13 bootloader pins are sampled to latch the boot configuration. These configurations are stored in the device status register. The RBL uses this device status register to architect the direction of the boot process.

Using the device configuration setup from the device status registers, the boot process executes an initialization code. The initialization settings that are executed by the RBL are listed below:

- The ROM code enables the reset isolation in all peripherals that support it. The power state of these peripherals is not changed. The list of peripherals that support reset isolation in KeyStone devices are SmartReflex, DDR3, embedded trace, Ethernet SGMII, Ethernet switch, SRIO, and AIF2 (if available).
- The ROM code also ensures that the power and clock domains are enabled for any peripherals that are required for boot.
- The ROM code configures the system PLL to set the device speed based on the three PLL selection bits in the DEVSTAT register. A post divider of two is always applied (postdiv value = 2).

  Table 2-1 lists PLL configurations that are achievable by different combinations of the PLL selection bits for different device speeds.

Table 2-1 PLL Clock Configuration (Part 1 of 2)

Boot Pll Select	Input Clock Freq (MHz)	Core = 800 MHz		Core = 1000MHz		Core = 1200MHz	
		Clkr	Clkf	Clkr	Clkf	Clkr	Clkf
0	50.00	0	31	0	39	0	47
1	66.67	0	23	0	29	0	35
2	80.00	0	19	9	24	0	29
3	100.00	0	15	0	19	0	23
4	156.25	24	255	4	63	24	383
5	250.00	4	31	0	7	4	47

Table 2-1 PLL Clock Configuration (Part 2 of 2)

Boot Pll Select	Input Clock Freq (MHz)	Core = 800 MHz		Core = 1000MHz		Core = 1200MHz	
		Clkr	Clkf	Clkr	Clkf	Clkr	Clkf
6	312.50	24	127	4	31	24	191
7	122.88	47	624	28	471	31	624
End of Table 2-1							

- The main PLL stays in bypass mode for no-boot, SPI, and I²C boot. For other boot modes, a PLL initialization sequence executes inside the boot ROM to configure the main PLL in PLL mode.
- The ROM code reserves the last 0xD23F bytes for this purpose in all the CorePacs. This area of the RAM is used to store the initial configuration of the boot process like the boot parameter table. Table 2-2 lists the configuration inputs stored in the local L2 for CorePac0 in C66xx devices. The base of the CorePac memory usage table can be obtained from the data manual for the specific device

Table 2-2 CorePac0 Memory Usage by the ROM Bootloader

Bytes offset	Size	Description
0x0	0x0040	ROM boot version string, (Unreserved)
0x40	0x0400	Boot Code Stack
0x520	0x0020	Boot Progress Register Stack (copies of boot program on mode change)
0x540	0x0100	Boot Internal Stats
0x640	0x0100	Boot variables (FAR data)
0x740	0x0100	DDR Configuration table
0x840	0x0080	RAM table functions
0x8C0	0x0080	Boot Parameter table
0x940	0x3600	Clear text packet scratch
0x5240	0x7f80	Ethernet/SRIO packet/message/descriptor memory
0xD1C0	0x80	Small Stack
0xD23C	0x04	Boot Magic Address

Note that for EMIF16 boot, no memory is reserved by the RBL; memory usage depends entirely on the image stored in, and executed from, the NOR flash.

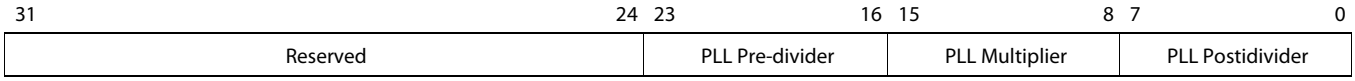
- During the boot process, the bootloader executes an IDLE command on the secondary CorePacs and keeps the secondary CorePacs waiting for an interrupt. After the application code to be loaded in these secondary CorePacs are loaded and the BOOT_MAGIC_ADDRESS in individual corePacs are populated, the application code in the corePac0 can trigger the IPC interrupt to wake up the secondary cores and branch up to the address specified in the BOOT_MAGIC_ADDRESS.
- All L1D and L1P memory is configured by the boot code as cache memory. But L2 memory is configured as addressable memory.
- All the interrupts are disabled except host interrupts that are needed for the PCIe, SRIO (DirectIO), and HyperLink boot modes.
- The bootloader also has a DDR configuration table that, by default, is initialized to all zeros. During the bootloader process, the parameter table is polled after every boot table section is complete. The DDR3 is configured if the bootloader finds that the enable bitmap field is non-zero. This allows a single boot table to

configure the DDR table, then load data to DDR. The DDR configuration boot parameter table is shown in Table 2-3.

Table 2-3 DDR Configuration

Byte offset	Name	Description
0	Enable bitmap	One bit per configuration value. Bit 0 corresponds to the PLL config entry, bit 1 to the SDRAM config entry, etc. The corresponding value will only be set if the bit is set in this bitmap.
4	PLL config	See Figure 2-1
8	config	SDRAM Config Register
12	config 2	SDRAM Config 2 Register
16	Refresh ctl	SDRAM Refresh Control Register
20	Timing 1	SDRAM Timing 1 Register
24	Timing 2	SDRAM Timing 2 Register
28	Timing 3	SDRAM Timing 3 Register
32	Nvm timing	LPDDR2-NVM Timing Register
36	Pwr management	Power Management Control Register
40	IODFT_TLGC	IODFT Test Logic Global Control Register
44	Perf ctl cfg	Performance Counter Config Register
48	Perf ctl sel	Performance Counter Master Region Select Register
52	Read idle ctl	Read Idle Control Register
56	Irq enable	System VBUSM Interrupt Enable Set Register
60	Zq config	SDRAM Output Impedance Calibration Config Register
64	Temp alert cfg	Temperature Alert Config Register
68	Phy ctrl 1	DDR PHY Control 1 Register
72	Phy ctrl 2	DDR PHY Control 2 Register
76	Pri cos map	Priority to Class of Service Mapping Register
80	Mst id cos map 1	Master ID to Class of Service 1 Mapping Register
84	Mst id cos map 2	Master ID to Class of Service 2 Mapping Register
88	Ecc ctrl	ECC Control Register
92	Ecc addr rng 1	ECC Address Range 1 Register
96	Ecc addr rng 2	ECC Address Range 2 Register
100	Rw/exc thresh	Read Write Execution Threshold Register
End of Table 2-3		

Figure 2-1 DDR PLL Configuration



- The part of the device status register that holds the boot configurations is shown in Table 2-4.

Table 2-4 Boot Mode Pins

12	11	10	9	8	7	6	5	4	3	2	1	0
PLL Mult	I²C/SPI Ext Device Cfg		Device Configuration							Boot Device		

The device configuration bits are interpreted by the RBL based on the boot mode selected. Table 2-5 lists the boot mode values. Please see the device data manual to check for the support of the specific boot mode listed.

Table 2-5 Device Configuration Pin Values

Boot Mode Pins: Boot Device Values	
Value	**Boot Device**
0	Sleep / EMIF16[1]
1	Serial RapidIO
2	Ethernet (SGMII) (PA driven from core clk)
3	Ethernet (SGMII) (PA driver from PA clk)
4	PCI
5	I^2C
6	SPI
7	HyperLink

1. See the device-specific data manual for information.

The RBL uses the device configuration to setup the initial configuration structure in a form of a boot parameter table which is stored in the reserved section of the L2 in corePac0. Even though the boot parameter table format varies based on the boot mode selected, the first 12 byte offsets are common across the boot modes. Table 2-6 lists these parameters.

Table 2-6 Boot Parameter Common Values

Byte Offset	Name	Description
0	Length	The length of the table, including the length field, in bytes.
2	Checksum	The 16 bits ones complement of the ones complement of the entire table. A value of 0 will disable checksum verification of the table by the boot ROM.
4	Boot Mode	0-7: Specifies the boot device.
6	Port Num	Identifies the device port number to boot from, if applicable
8	SW PLL, MSW	PLL configuration, MSW
10	SW PLL, LSW	PLL configuration, LSW
End of Table 2-6		

The PLL configuration values are interpreted based on the Table 2-7.

Table 2-7 PLL Configuration

31 30	29 16	15 8	7 0
PLL Config Ctl • 00 - PLL not configured • 01 - PLL configured only if it is already in bypass or disable mode • 10 - PLL is configured • 11- PLL is disabled and put into bypass mode	Pll Multiplier (can be between 0-8191)	Pll Pre-Divider (can be between 0-255)	Pll Post-Divider (can be between 0-255)

The rest of boot parameter table formats are explained under individual boot mode description later in this user guide. The RBL uses the BOOTCOMPLETE register, which controls the BOOTCOMPLETE pin status, to indicate the completion of the ROM boot process. Due to legacy implementation, the BOOTCOMPLETE bit is set before the image is loaded in the CorePac0.

2.3 Bootloader Initialization After (Hard Reset or Soft Reset)

The boot process carried out by the RBL first checks if the hibernation is enabled in the power status control register. If the hibernation is enabled, the boot process carries out the hibernation sequence mentioned in section 2.4. If the hibernation is not enabled, the RBL follows the boot flow similar to the one mentioned in section 2.2, except that the boot configuration pins are not sampled to update the device status registers. In addition the power state controller will not reset any peripherals that are reset isolated in the device. Refer to the Data Manual for detail description of the different reset types.

2.4 The Effect of Hibernation on Bootloader Initialization During Chip Reset0 and Chip Reset1

To reduce power consumption, hibernation can be used to shutdown CorePacs and peripherals that are not used. The RBL's involvement in the hibernation process is minimal. Before shutting down the cores, the user needs to set the hibernation enable bit and the mode bit and the branch address offset to which corePac0 will jump to execute the wake up code sequence in the power state control register. After a hard or soft reset, the bootloader samples the power state control register to verify that hibernation is enabled. If hibernation is enabled, the bootloader resets a set of peripherals and avoids resets to other peripherals based on the hibernation mode set. In KeyStone devices, there are two hibernation modes: Hibernation1 and Hibernation2.

In Hibernation1 mode, the critical status values and information are stored in the MSMC SRAM. A chip-level register is added to control the reset MSMC parity RAM. Before entering Hibernation1 mode, the user must correctly configure the chip-level register Chip Miscellaneous Control register so the parity SRAM will not reset when a hard or soft reset is triggered to exit Hibernation1. During this hibernation mode, only MSMC SRAM content is preserved; the MSMC MMR is not preserved. Therefore, the user should save the MSMC MMR to a known memory location inside the MSMC SRAM before entering this hibernation mode. In general, when Hibernation1 is enabled, CorePac0 will disable DDR self-refresh and branch-to-address specified in PWRSTATECTL and the other CorePacs will be in the IDLE state and after wake-up the user should make sure that the PWRSTATECTL's standby and hibernation fields have become 0, then branch to the boot magic address. The response time in the case of Hibernation1 is less than 100 ms.

In Hibernation2 mode, the critical status values and data are stored in DDR memory. When the hard or soft reset is triggered to exit from Hibernation2 mode, the MSMC is also reset. The MSMC configuration is set to the default value after reset, so only the lower two gigabytes of four gigabytes of space in the DDR memory will be visible. Therefore, the branching address for exiting Hibernation2 mode should be set at the lower two-gigabyte boundary, between 0x8000_0000 to 0xFFFF_FFFF. In general, when Hibernation2 is enabled, CorePac0 will also disable DDR self-refresh, reset MSMC parity, and branch-to-address specified in PWRSTATECTL; the other CorePacs will be in the IDLE state and after wake up will verify that PWRSTATECTL's standby and hibernation fields have become 0, then branch to the boot magic address. The response time is less than two seconds.

Before entering hibernation mode, the user can enable or disable reset isolation for SRIO. When SRIO has reset isolation enabled before entering hibernation, the user should also make the LPSC for SRIO active because packet forwarding requires the VBUS clock to function. When SRIO has reset isolation disabled before entering hibernation, the SRIO block must be disabled by the user; this includes stopping the VBUS clock and disabling the PHY layer. Stopping the VBUS clock without disabling the PHY layer can cause system congestion and system hang.

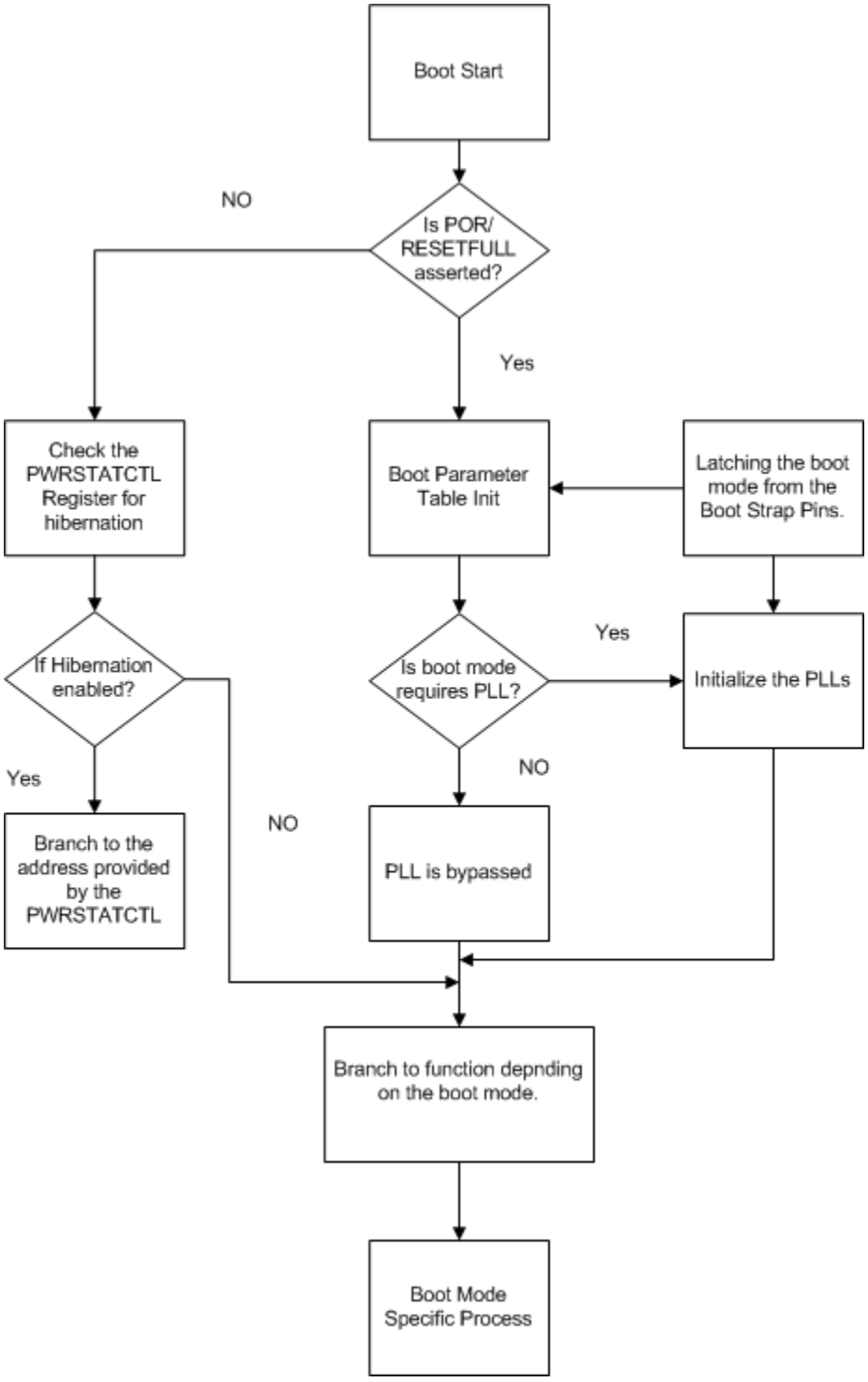
When PCIe is used in EP mode, the user must put PCIe in the L1 powerdown mode before it enters the hibernation mode; the PCIe power domain should be kept on during hibernation mode. When the device exits hibernation mode, the RC device can issue a reset request to all the EP points to bring the PCIe endpoint alive.

To avoid resetting the DDR during hard reset, reset isolation is provided for the DDR. The boot code enables the DDR reset isolation by default and the user has the option to turn off the reset isolation feature if it is not needed (See the *Power Sleep Controller (PSC) for KeyStone Devices User Guide* (SPRUGV4) in "Related Documentation from Texas Instruments" on page ø-viii for disabling the reset isolation for DDR3). Because the DDR contents are preserved, the PLL for DDR3 EMIF must stay locked and DDR PHY must be active to preserve the DDR3 content. This avoids full calibration when resuming the normal operation; full calibration can corrupt the DDR3 content. In summary, the DDR is alive during both hibernation modes and the DDR3 can be put into self-refresh mode to save power.

MSMC on this device does not support the PSC Disable interface. Therefore, MSMC could not detect the status of DDR3 EMIF when DDR3 EMIF is disabled or enabled by PSC. Because accesses to disabled DDR3 EMIF would hang the device, the user can tie off the PSC control to DDR_EMIF to enable it on the chip level (always on). This linking of the PSC control plus the reset isolation of DDR EMIF make it impossible to reset EMIF4F independently of the rest of the chip.

The hibernation sequence is explained in detail in the power management app note. The flow of the boot process is shown in Figure 2-2.

Figure 2-2 Boot Process



2.5 Boot Configuration Format

2.5.1 Introduction

The RBL uses a set of tables to determine the boot flow. Before going through individual boot mode operation, it is necessary to understand these different types of tables. There three types of tables used by the RBL. they are,

- Boot Parameter Table
- Boot Table
- Boot Configuration Table

2.5.2 Boot Parameter Table

The boot parameter table is the most common format the RBL employs to determine the boot flow. As mentioned in the section 2.2, they have first few parameters in the table common across all the boot modes while the rest of the table format is dependant on the boot mode selected.The RBL copies a default boot parameter table destined for each boot mode into the reserved L2 section of corePac0 and modifies the default values based on the boot configuration selected through the bootstrap pins. This table forms the maps for the RBL to execute the boot process.

2.5.3 Boot Table

The image to be loaded into the device is converted to a format recognizable by the RBL. This format is called the boot table. Code and data sections are inserted into the boot table automatically by the hex conversion utility. The hex conversion utility uses information embedded by the linker in the application file to determine each section's destination address and length. Adding these sections to the boot table requires no special intervention by the user. The hex conversion utility adds all initialized sections in the application to the boot table. The remaining information included in this section describes the format of the sections in the boot table.

Each section is added to the boot table in the same format. The first entry is a 32-bit count representing the length of the section in bytes. The next entry is a 32-bit destination address, where the first byte of the section is copied.

The RBL continues to read and copy these sections until it encounters a section whose byte count is zero. This indicates the end of the boot table; then, the bootloader branches to the entry point address (specified at the beginning of the boot table) and begins executing the application.

The boot table format is as follows:

- 32-bit header record indicating where the bootloader should branch after it has completed copying the data
- For each COFF section
  - 32-bit Section byte count
  - 32-bit Section address (destination address for the copy)
  - The data to be copied
- A 32-bit Termination record (0x00000000)

2.5.4 Boot Configuration Table

A boot configuration table is used if certain peripherals must be programmed with values that differ from their reset values before loading an application. For example, if the application needs to be loaded into DDR memory, the boot configuration table can be used to program the DDR registers and enable the DDR peripheral before loading the application code in DDR.

Each table entry in the boot configuration table has three elements:

- The address to be modified
- The set mask
- The clear mask

The RBL reads the specified address, then sets any bits that are set in the set mask element and clears any bits that are set in the clear mask element. If both the set and clear mask elements are 0, the value in the address field is branched via a standard call with the return address stored in register B3. The boot configuration table is terminated when all three elements are 0.

2.5.5 Utilities to Generate Different Tables

Below is the list of the utilities to generate different table formats.

- Hex6x utility is used to convert the application code into a boot table format.
- Romparse is used to append the boot parameter to boot table or a boot configuration table.
- Bootconvert6x utility is used to convert the boot table derived from a little endian application code to a big endian format. This is required as RBL assumes all the images to be in big endian mode.
- B2i2c is an utility used to convert the boot table into a i2c/spi format table. This table can be loaded into a EEPROM that is connected through I2C to the device.
- Bootpacket is an utility to break the boot table into packets that can be sent from the host to the device booted in ethernet boot mode.
- Pcsendpkt is another tool used that can help the host to send the packets generated by bootpacket to the boot device.

Chapter 3

Boot Modes

3.1 EMIF16 Boot

3.1.1 Basic Boot Operation

In this mode, the ROM code configures the EMIF16 interface and sets the boot complete, then branches to the EMIF CS2 data memory at 0x70000000. No return is expected. No memory is reserved by the bootloader. The device configuration for EMIF16 boot mode is shown in Table 3-1.

Table 3-1 EMIF16 Boot Mode Device Configurations

6	5	4	3	2	1	0
submode 00 - Sleep boot 01 - EMIF16 boot 10 - 11 - Reserved		Wait Enable 0 - Wait disabled 1- Wait enabled	Reserved		SR Index	

3.2 SRIO Bootloader Operation

3.2.1 Basic Boot Operation

The SRIO boot operates in two modes: Messaging mode and DirectIO mode. Both the Messaging mode and DirectIO mode are enabled by default, but mailbox can be disabled using the I²C boot parameter table. In Messaging mode, the RBL receives the application code through the SRIO messaging interface in the form of the boot table (see "Master Boot—Boot Table Mode" on page 3-14). The RBL also populates the boot magic address during this process. Transmission is terminated after the end-of-boot packet is received.This signals the completion of the boot process and also wakes up the DSP to execute the boot image from the entry point specified in the boot table. In DirectIO mode, the RBL keeps polling the boot magic address and when the value is non-zero, the RBL jumps the core0 to the address in the boot magic address and let it execute the application code. Because there is no provision in messaging mode to handle the messaging output, the host application should provide a delay between message transmissions to avoid an out-of-order message scenario.

The SRIO boot mode parameters and the bit field descriptions are listed in Table 3-2.

Table 3-2 SRIO Boot Mode Parameter Table

Byte Offset	Name	Description
12	Options	Bit 0 Tx enable 0 SRIO Transmit disable 1 SRIO Transmit Enable Bit 1 Mailbox Enable 0 Mailbox mode disabled. (SRIO boot is in DirectIO mode). 1 Mailbox mode enabled. (SRIO boot is in Messaging mode). Bit 2 Bypass Configuration 0 Configure the SRIO 1 Bypass SRIO configuration Bit 3-15 = Reserved
14	Lane Setup	0b0000 = SRIO configured as four 1x ports 0b0001 = SRIO configured as 3 ports (2x, 1x, 1x) 0b0010 = SRIO configured as 3 ports (1x, 1x, 2x) 0b0011 =SRIO configured as 2 ports (2x, 2x) 0b0100 = SRIO configured as 1 4x port 0b 0101 - 0bffff = Reserved
16	Config Index	Specifies the template used for RapidIO configuration. Must be 0 for KeyStone Architecture
18	Node ID	The node ID value to set for this device
20	SerDes ref clk	The SerDes reference clock frequency, in 1/100 MHZ
22	Link Rate	Link rate, MHz
24	PF Low	Packet forward address range, low value
26	PF High	Packet Forward address range, high value
End of Table 3-2		

Apart from some of the parameters in Table 3-2 that are latched from the bootstrap settings, all the other parameters are populated with default values by the RBL. These parameters can be modified only through the I²C boot parameter table through the I²C master boot and reenter the bootloader in SRIO mode. For example, although there are five possible lane setups, only two lane setups are possible from the bootstrap pin configuration for primary SRIO boot and other lane setups can be configured only through the I²C boot parameter table. The lane setups and the reference clock setups

can be derived from the SRIO device configuration field of the DEVSTAT register. The SRIO device configuration field is described in Table 3-3 and Table 3-4.

Table 3-3 SRIO Boot Mode Device C

6	5	4	3	2	1	0
Lane Setup	Data Rate		Ref Clock		SR ID	

Table 3-4 SRIO Boot Mode Device Configuration Field Descriptions

Bit Field	Description
SR ID	0-3 = Smart Reflex ID
Ref Clock	0 = Reference Clock = 156.25 MHz 1 = Reference Clock = 250 MHz 2 = Reference Clock = 312.5 MHz
Data Rate	0 = Data Rate = 1.25 Gbps 1 = Data Rate = 2.5 Gbps 2 = Data Rate = 3.125 Gbps 3 =Data Rate = 5.0 Gbps
Lane Setup	0 = Port Configured as 4 ports each 1 lane wide (4x1ports) 1 = Port Configured as 2 ports 2 lanes wide (2x2 ports)
End of Table 3-4	

Before configuring SRIO, SerDes should be configured. The RBL initializes the PLL and the receive and the transmit channel registers. After configuring SerDes and SRIO for message mode, the boot code initializes the QMSS to configure transmit and receive queues. The boot code allocates a 32k block of local L2 memory on CorePac0 for packet reception. This is divided into seven buffers of 4096 bytes each with the remainder for the packet descriptors. After the boot table processing is complete, the boot code resets the queue manager, sets the boot complete bit, and branches to the address specified by the boot table; the interrupt maps are restored to their default values.

The boot messages are simply a segmented boot table prepended with the header shown in Table 3-5. The application that is loading the message from the host device should be responsible to package the message packets with the header before sending to the device. Also since the reset isolation is enabled by default by RBL, the application code should be responsible to disable the reset isolation.

Table 3-5 SRIO Message Mode Boot Header

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
block Size = 6															
Block Checksum one's Complement															

3.3 Ethernet Bootloader Operation

3.3.1 Basic Boot Operation

The RBL configures SerDes, SGMII, and the switch plus the PASS and the Multicore Navigator (Packet DMA and the QMSS) (if it is available in the DSP) to prepare to receive the boot table from the host connected to the device through the Ethernet interface. These initial configurations are derived by querying the boot mode pins of the DEVSTAT register and the boot parameter table for the Ethernet boot. Based on the boot mode selected, the PA subsystem can be driven either by the main PLL reference clock or by the SerDes reference clock (see Table 3-6 for the boot mode selection). If the main PLL reference clock is used, the PA multiplier configuration can be derived from the three PLL selection bits in the DEVSTAT and the input clock as shown in the table.

Table 3-6 PA PLL Clock Configuration

Boot PLL Select [2:0]	Input Clock Freq (MHz)	PA = 350 MHz	
		Clkr	Clkf
0	50.00	0	41
1	66.67	1	62
2	80.00	3	104
3	100.00	0	20
4	156.25	24	335
5	250.00	4	41
6	312.50	24	167
7	122.88	11	204
End of Table 3-6			

If the PA reference clock is used, the PA PLL clock configuration is selected based on the SerDes Clock Mult section in the Ethernet Boot Mode Device Configuration shown in Table 3-7. If the SerDes clock is used, the RBL first configures the SerDes PLL and the transmit and receive channels. The default values for these registers are configured in the boot parameter table for the Ethernet boot as shown in Table 3-9. The external connection from the DSP to the host is selected through the device configuration bits in the boot mode pins of the DEVSTAT register. The device configuration bit fields and the values for these fields are described in Table 3-7 and Table 3-8. Not all the KeyStone devices have PA sub system. In this case only the SERDES reference clock is used to drive the Ethernet sub system.

Table 3-7 Ethernet(SGMII) Boot Mode Device Configuration

6	5	4	3	2	1	0
SerDes Clock Mult		Ext Connection		DEV ID	DEV ID (SR ID)	

Table 3-8 Ethernet Boot Mode Device Configuration Descriptions

Bit field	Description
Ext connection	0 = Mac to Mac connection, master with auto negotiation 1 = Mac to Mac connection, slave, and Mac to Phy 2 = Mac to Mac, forced link 3 = Mac to fiber connection
Device ID	0-7 = This value is used in the device ID field of the Ethernet ready frame. Bits 1:0 are used for the SR ID.
SerDes Clock Mult The output frequency of the PLL must be 1.25 GBs.	0 = x8 for input clock of 156.25 MHz 1 = x5 for input clock of 250 MHz 2 = x4 for input clock of 312.5 MHz 3 = Reserved
End of Table 3-8	

Table 3-9 Ethernet Boot Parameter Table (Part 1 of 2)

Byte Offset	Name	Description
12	Options	Bits 00 - 02 Interface (Check the Data Manual to see the interfaces available in the device) 000 - MII 001 - RMII 010 - GMII 011 - RGMII 100 - SMII 101 - S3MII 110 - RMII 10Mbps 111 - RMII 100Mbps Bits 03 HD 0 - Half Duplex 1 - Full Duplex Bit 4 Skip TX 0 = Send Ethernet Ready Frame every 3 seconds 1 = Don't send Ethernet Ready Frame Bits 06 - 05 Initialize Config 00 = Switch, SerDes, SGMII and PASS are configured 01 = Only SGMII and PASS are configured 10 = Reserved 11 = None of the Ethernet system is configured.
14	MAC High	The 16 MSBs of the MAC address to receive during boot
16	MAC Med	The 16 middle bits of the MAC address to receive during boot
18	MAC Low	The 16 LSBs of the MAC address to receive during boot
20	Multi MAC High	The 16 MSBs of the multi-cast MAC address to receive during boot
22	Multi MAC Med	The 16 middle bits of the multi-cast MAC address to receive during boot
24	Multi MAC Low	The 16 LSBs of the multi-cast MAC address to receive during boot
26	Source Port	The source UDP port to accept boot packets from. A value of 0 will accept packets from any UDP port
28	Dest Port	The destination port to accept boot packets on.
30	Device ID 12	The first two bytes of the device ID. This is typically a string value, and is sent in the Ethernet ready frame
32	Device ID 34	The 2nd two bytes of the device ID.

Table 3-9 Ethernet Boot Parameter Table (Part 2 of 2)

Byte Offset	Name	Description
34	Dest MAC High	The 16 MSBs of the MAC destination address used for the Ethernet ready frame. Default is broadcast.
36	Dest MAC Med	The 16 middle bits of the MAC destination address
38	Dest MAC Low	The 16 LSBs of the MAC destination address
40	SGMII Config	Bits 0-3 are the config index, bit 4 set if direct config used, bit 5 set if no configuration done
42	SGMII Control	The SGMII control register value
44	SGMII Adv Ability	The SGMII ADV Ability register value
46	SGMII TX Cfg High	The 16 MSBs of the SGMII Tx config register
48	SGMII TX Cfg Low	The 16 LSBs of the SGMII Tx config register
50	SGMII RX Cfg High	The 16 MSBs of the SGMII Rx config register
52	SGMII RX Cfg Low	The 16 LSBs of the SGMII Rx config register
54	SGMII Aux Cfg High	The 16 MSBs of the SGMII Aux config register
56	SGMII Aux Cfg Low	The 16 LSBs of the SGMII Aux config register
58	PKT PLL Cfg MSW	The packet subsystem PLL configuration, MSW
60	PKT PLL CFG LSW	The packet subsystem PLL configuration, LSW

End of Table 3-9

After initializing the clock according to mode, the RBL enables the interface by default in full-duplex gigabit rate. The interface is also configured to receive broadcast packets as specified in the boot parameter table (Table 3-9). The external PHY is also initialized to come up in standard learning mode. The RBL code then initializes the QMSS if available in the device to configure transmit and receive queues. The RBL also allocates a 32k block of local L2 memory on CorePac0 for packet reception. This is divided into 20 buffers each of 1536 bytes, with the remainder for the packet descriptors.

If the PA sub system is available in the device, the RBL loads the custom firmware is to operate the PA subsystem, which enables the PASS to direct all the received traffic to the CPDMA using the above-configured flow. The RBL also configures a transmit channel to optionally send an Ethernet-ready packet, which is sent at the end of the configuration described above. When it is enabled to send, the Ethernet-ready packet is sent every three seconds—the time interval between the packets varies with the input clock—until the first valid boot packet is received by polling the receive queue. After it is detected, the received packet is verified, then sent to the boot table processing functions. When boot-table processing is completed, the RBL resets the PA and the queue manager before branching to the address specified in the boot table. The interrupt maps are also restored to their default values and start executing the application loaded.

3.3.2 Ethernet-Ready Announcement Format

The Ethernet-ready announcement frame is made in the form of a BOOTP request so it can use a standard format. No response is processed for this message and it is constructed so that most—if not all—BOOTP and DHCP servers will discard it. The announcement frame is sent every three seconds (the time interval between the packets varies with the clock input); no retransmission is done.

The frame uses the DIX MAC Header format. The MAC header contains:

- Destination MAC address = H-MAC addr (from boot parameters, normally FF:FF:FF:FF:FF:FF)
- Source MAC address == this devices MAC addr (from boot parameters)
- Type = IPV4 (0x800)

The IPV4 header contains:

- Version = 4
- Header length = 0
- TOS = 0
- Len = 328 (300 BOOTP + 8 UDP + 20 IP)
- ID = 0x0001
- Flags + Fragment offset = 0
- TTL = 0x10
- Protocol = UDP (17)
- Header checksum = 0xA9A5
- SRC IP = 0.0.0.0
- DEST IP = 0.0.0.0

The UDP header contains:

- Source port = BOOTP client (68 decimal)
- Destination port = BOOTP server (67 decimal).
- Length = 308 (300 BOOTP + 8 UDP)
- Checksum = 0 (not calculated)

The BOOTP Payload contains:

- Opcode = Request (1)
- HW Type = Ethernet (1)
- HW Addr Len = 6
- Hop Count = 0
- Transaction ID = 0x12345678
- Number of seconds = 1
- Client IP = 0.0.0.0
- Your IP = 0.0.0.0
- Server IP = 0.0.0.0
- Gateway IP = 0.0.0.0
- Client HW Addr = this device's MAC address
- Server hostname = ti-boot-table-svr
- Filename = ti-boot-table-XXXX (where XXXX is the 4 character device ID from boot parameters)
- Vendor info = all zeros

3.3.3 Ethernet Boot Packet Format

The boot table format is encapsulated in Ethernet frames with IPV4 and UDP headers. The following paragraphs describe the Ethernet frames that are accepted. Frames not matching the specified criteria are silently discarded and subsequent frames processed.

Frames using both DIX and 802.3 MAC header formats are accepted as are frames with and without VLAN tags. Any source MAC address is acceptable. A destination MAC address of this device (as specified in boot parameters) or the M-MAC specified in the boot parameters are accepted. VLAN fields (other than type/len) are ignored. If 802.3 format MAC format is used, the SNAP/LLC header will be verified and skipped. The type field will select IPV4 type (0x0800).

The IPV4 header validates the Version (4), flag and fragment fields, and protocol (UDP) field. The header length field is parsed to skip header option words. Any source and destination IP addresses are accepted.

The UDP header verifies that the source and destination port numbers match those specified in the boot parameters. If the boot parameter source port field is 0, any source port will be accepted. The UDP header length is sanity-tested against the adjusted frame length. If the UDP length is too long for the frame or is not a multiple of two, the frame is discarded. The UDP checksum is verified and the frame with incorrect UDP checksum is discarded if the UDP checksum field is non-zero.

The following checks are performed on the boot table frame header. The magic number field and opcode fields are compared to the expected values. The sequence number field is compared to the expected value. The expected value for sequence number is 0 for the first frame processed and increments by one for each processed frame.

The Boot table frame payload (which is a multiple of four bytes in length) is processed by the boot-table processing function.

Table 3-10 Ether Boot Packet Format

Ethernet Header, One of The Following Types:
DIX Ethernet (DMAC, SMAC, type: 14 bytes) 802.3 w/ SNAP/LLC (DMAC, SMAC, len, LLC, SNAP: : 22 bytes) DIX Ethernet w/ VLAN (18 bytes) 802.3 w/ VLAN and SNAP/LLC (26 bytes)
IPV4 (20 to 84 bytes) UDP (8 bytes)
Boot Table Frame Header (4 bytes) Boot Table Frame Payload (min 4 bytes, max limited by max Ethernet frame - previous headers)

Table 3-11 Boot Table Frame Header

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Boot Magic Number = 0x544b															
Opcode = 0x01								Sequence number							

3.4 PCI Express (PCIe) Bootloader Operation

3.4.1 Basic Boot Operation

In the PCIe boot mode, the host configures memory and loads all the sections directly to the memory. The bootloader configures the base address registers, the number of windows, and their size. The PCIe power-up is configured through the external pin PCIESSEN. PCIe boot code can configure the PCIe registers either by getting the values from the I²C or the default values from the boot parameter table (Table 3-12).

Table 3-12 PCIe Boot Parameter Table

Byte Offset	Name	Description
12	Config	PCI configuration options
14	Address Width	PCI address width, can be 32 or 64
16	Link Rate	SerDes frequency, in Mbps. Can be 2500 or 5000
18	Reference clock	Reference clock frequency, in units of 10 kHz. Value values are 10000 (100 MHz), 12500 (125 MHz), 15625 (156.25 MHz), 25000 (250 MHz) and 31250 (312.5 MHz). A value of 0 means that value is already in the SerDes cfg parameters and will not be computed by the boot ROM.
20	Window 1 Size	Window 1 size.
22	Window 2 Size	Window 2 size.
24	Window 3 Size	Window 3 size. Valid only if address width is 32.
26	Window 4 Size	Window 4 Size. Valid only if the address width is 32.
28	Vendor ID	Vendor ID
30	Device ID	Device ID
32	Class code Rev ID MSW	Class code revision ID MSW
34	Class code Rev ID LSW	Class code revision ID LSW
36	SerDes cfg msw	PCIe SerDes config word, MSW
38	SerDes cfg lsw	PCIe SerDes config word, LSW
40	SerDes lane 0 cfg msw	SerDes lane config word, msw lane 0
42	SerDes lane 0 cfg lsw	SerDes lane config word, lsw, lane 0
44	SerDes lane 1 cfg msw	SerDes lane config word, msw, lane 1
46	SerDes lane 1 cfg lsw	SerDes lane config word, lsw, lane 1
End of Table 3-12		

If the PCIe boot is the primary boot, the BAR size configuration is driven by the BAR config fields listed in Table 3-13 and Table 3-14.

Table 3-13 PCIe Boot Mode Device Configuration

6	5	4	3	2	1	0
Reserved	BAR Config				SR ID	

Table 3-14 PCIe Boot Mode Device Configuration Bits Description

Bit Field	Value	Description
SR ID	0-3	Smart Reflex ID
Bar Config	0-0xf	See Table 3-15

Table 3-15 PCIe Window Sizes

BAR Config	BAR0	32-bit Address Translation					64-bit Address Translation	
		BAR1	BAR2	BAR3	BAR4	BAR5	BAR2/3	BAR4/5
0b0000	PCIe MMRs	32	32	32	32	Not Used		
0b0001		16	16	32	64			
0b0010		16	32	32	64			
0b0011		32	32	32	64			
0b0100		16	16	64	64			
0b0101		16	32	64	64			
0b0110		32	32	64	64			
0b0111		32	32	64	128			
0b1000		64	64	128	256			
0b1001		4	128	128	128			
0b1010		4	128	128	256			
0b1011		4	128	256	256			
0b1100							256	256
0b1101							512	512
0b1110							1024	1024
0b1111							2048	2048

End of Table 3-15

If the I²C EEPROM is used to configure the PCI registers, the BAR configurations are driven by the parameter table downloaded through I²C. On the boot reentry, the PCIe boot code is executed with these BAR configurations. In this mode of operation, the I²C peripheral is also configured by the bootloader.

The bootloader code also configures the interrupt subsystem by configuring the chip-level interrupt controller, then executes the IDLE instruction (Mostly MSI or legacy interrupts can be used). After the host transfers the boot table directly to the memory location and if the BOOT MAGIC ADDRESS register is non-zero, the ROM code copies the start address of the application from the BOOT MAGIC ADDRESS to the program counter wakes up the DSP through an interrupt generated by poking the MSI application register and executes the application loaded. If the BOOT MAGIC ADDRESS register is still zero the ROM code keeps the DSP in idle till the value is non-zero.

3.5 I2C Bootloader Operations

3.5.1 I2C Boot Basic Operations

I2C is the only peripheral that is configured during I2C boot. Because the data is transferred using I2C from EEPROM, the interrupt subsystem and the EDMA are not enabled. A seven-bit address and eight-bit data modes are the only supported transfer configuration. I2C can operate either in master mode or slave mode. In the master mode the DSP drives I2C slave device where the image is stored. In the slave mode, the DSP is driven by an I2C master device. The master device is mostly another DSP or a FPGA.

3.5.2 Master Mode—Boot Parameter Table

The master boot mode by default looks for a boot parameter table. After the BOOT MAGIC ADDRESS register is cleared, the data is read from the I2C slave device in blocks of data starting at address (0x80 * parameter index). The parameter index can be read from the I2C master mode device configuration bit fields shown in the Table 3-16 and Table 3-17.

Table 3-16 I2C Master Mode Device Configuration

9	8	7	6	5	4	3	2	1	0
Reserved	Speed	Address	Reserved	Mode(0)	Parameter Index				
					SR index				

Table 3-17 I2C Master Mode Field Description

Bit Field	Description
Mode	0 = Master Mode 1 = Passive Mode (bit 6 is always set to 1)
Address	0 = Boot From I2C EEPROM at I2C bus address 0x50 1 = Boot From I2C EEPROM at I2C bus address 0x51
Speed	0 = I2C data rate set to approximately 20 kHz 1 = I2C fast mode. Data rate set to approximately 400 kHz (will not exceed)
Parameter Index	0 - 32= Identifies the index of the configuration table initially read from the I2C EEPROM

Each of these blocks have a maximum size of 128 bytes with first 4 bytes allocated to the block header. The block format is shown in Table 3-18

Table 3-18 EEPROM Block Format

Offset	Field	Value
0	Block Size	The size of the block including the header
2	Checksum	The ones compliment check sum, including the block size and checksum fields. Valid checksum values are 0 and -0
4 - 127	Data	Data
End of Table 3-18		

The data section format of the table in the boot parameter mode is listed in Table 3-19.

Table 3-19 Boot Parameter Table Format

Offset	Field	Value
4	Boot Mode	Extended boot mode that is used by the bootloader (See Table 3-20)
6	Port Num	Not Used
8	PLL configuration (MSW)	See Table 2-7
10	PLL configuration (LSW)	See Table 2-7
12	Option	See Table 3-21
14	Boot Dev Addr	The I²C device address to boot from
16	Boot Dev Addr Ext	Extended boot device address
18	Broadcast Addr	I²C address used to send data in the I²C master broadcast mode.
20	Local Address	The I²C address of this device
22	Device Freq	The operating frequency of the device (MHz)
24	Bus Frequency	The desired I²C data rate (kHz)
26	Next Dev Addr	The next device address to boot (Used only if boot config option is selected)
28	Next Dev Addr Ext	The extended next device address to boot (Used only if boot config option is selected)
30	Address Delay	The number of CPU cycles to delay between writing the address to an I²C EEPROM and reading data.
End of Table 3-19		

Table 3-20 Extended Boot Modes

Boot Type	Extended Boot Mode Value (Decimal)
Ethernet Boot Mode	10
SRIO Boot Mode	20
PCIe Boot Mode	30
I²C Master Boot Mode	40
I²C Passive Boot Mode	41
SPI Boot Mode	50
HyperBridge Boot Mode	60
EMIF 16 Boot Mode	70
Sleep Boot Mode	100

The I²C boot mode can also be used a primary entry to perform some pre-initialization, then re-entered in any other native mode using the boot parameter table. The only difference in this case is that the data section will vary based on the secondary boot mode selected.

Table 3-21 Boot Options

Boot Option value	Boot Option Description
0	Boot Parameter Table
1	Boot Tables
2	Boot Config Table
3	Passive Boot Mode

3.5.3 Master Boot—Boot Table Mode

In this mode, the data is read from the EEPROM through the I²C and the four-byte block header is stripped and the remaining data is passed to the boot table processing function. If the BOOT MAGIC address is found to be non-zero during the boot data transfer, the boot code terminates and sets the boot complete and branches to the address specified in the BOOT MAGIC address. This boot table is a block of data that contains the code and data sections to be loaded by the RBL, as well as other information such as the entry point address. The boot table is created by the hex conversion utility (a standard component of the TMS320C6000 Assembly Language Tools), based on the COFF (common object file format) output of the linker for the application code. The hex conversion utility provides several output options, including industry-standard ASCII formats that can be used to program parallel or serial EEPROMs, and formats that can be used in code for a host to transmit the boot table to the DSP.

To create the boot table, do the following steps:

1. Use the hex conversion utility (hex6x.exe) revision 6.0A or later.
2. Use the -boot option to cause the hex conversion utility to create a boot table.
3. Specify the romwidth and memwidth to both be 32 bits.
4. Specify the entry point using the -e entry_point_address option. The entry point is the address to which the bootloader transfers execution when the boot load is complete.
5. Specify the output filename using the -o output_filename option. If you do not specify an output filename, the hex conversion utility creates a default filename based on the output format.
6. Specify the endianness to match the compilation, -order L for little endian, -order M for big endian.
7. Correct any sections that are not multiples of 32 bits. The C compiler always generates sections whose lengths are multiples of 32 bits. This may not be the case for any sections declared in assembly. For little endian systems, the byte order must be swapped for these remaining bytes.

Example 3-1 Creating a Boot Table for ASCII Output

To create a boot table for the application my_app.out with the following conditions:

- Little endian compilation
- Desired output is ASCII format in a file called my_app.hex

Use the following options on the hex conversion utility command line or command file:

```
-boot; option to create a boot table
-a; ASCII format
-e _c_int00; Standard entry point for C library
-order L; Little endian format
-memwidth32; memory width
-romwidth32; rom width
-o my_app.hex; specify the output filename
my_app.out; specify the input file
```

End of Example 3-1

3.5.4 Master Boot—Config Table Mode

In this mode, the data is read from the I²C contains configuration tables. Each element in the table has three 32-bit fields, as shown in Table 3-22.

Table 3-22 Config Table Layout

Entry 0	Address
	Set Mask
	Clear Mask
Entry 1	Address
	Set Mask
	Clear Mask
…	
Entry N, Table Termination	Address = 0
	Set Mask = 0
	Clear Mask = 0

Table 3-23 shows an example boot configuration. Each of these entries in the table can be a standard entry, branch entry, or termination entry.

Table 3-23 Boot Config Table Format

Offset	Data	Operation
0x0	0x0093001C	Set 16 bits MSBs and clear 16LSBs at address 0x0093001C
0x4	0xFFFF0000	
0x8	0x0000FFFF	
0xC	0x00000000	Termination
0x10	0x00000000	
0x14	0x00000000	

3.5.4.1 Standard Entry

A standard entry has address ≠ 0, and set mask or clear mask ≠ 0. The ROM code reads the 32-bit value at the address, modifies the value as shown in Table 3-24 on a bit-by-bit basis, and writes the value back.

Table 3-24 Standard Boot Config Table Options

Set Mask Bit	Clear Mask Bit	Operation
0	0	Bit value is unchanged
1	0	Bit value is set
0	1	Bit value is cleared
1	1	Bit value is toggled

3.5.4.2 Branch Entry

A branch entry has address ≠ 0, set mask = 0, clear mask = 0. The boot ROM makes a function call to the address. On return (if there is one) the table processing continues.

3.5.4.3 Termination Entry

The table termination field has address, set mask, and clear mask all set to 0. When this entry is found, the boot ROM modifies the current active boot parameter table. The boot mode is changed to I²C master boot parameter table mode, the address is changed to the current next address value, and the boot is rerun.

3.5.5 Master Boot—Transmit Mode

Master transmit mode cannot be initiated from bootstrapped pins. To enter this mode, the device is pin-strapped as I²C Master mode boot. The boot parameter table loaded sets the new boot mode to I²C master transmit mode.

When it is in master transmit mode, the boot ROM reads a block of data from the I²C EEPROM exactly as it does in master mode, supporting both boot tables and boot configuration tables. But in addition to processing the I²C block, the boot ROM retransmits the block from the I²C bus to the address specified in the broadcast address field of the boot parameter table.

Before sending the first data block, the boot ROM must send the following block to the broadcast address on the I²C bus. The slave boot processors use this field to set their options flag. This is required to allow the slaves to know if the next received data blocks contain boot tables data or boot configuration tables.

Table 3-25 I²C Master Transmit Mode Broadcast Options

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
block Size = 6															
Checksum															
I²C Slave Boot mode Options filed															

3.5.6 Slave Boot

In I²C slave mode, the boot ROM operates the I²C device in receiver mode and waits for an I²C Master to send data using a standard boot table format. The procedure is the same as the I²C Master mode except that the DSP configures its I²C interface for slave reads to the slave address. The device configuration field for I2C Slave mode is shown in Table 3-26. The default slave address is set to value in bits 4-2 in Table 3-26 plus 0x19. Also, the I²C bus on the master side needs to be configured to have the same frequency as the I²C module within the DSP. Secondly, the I²C master need to send six data bytes to the slave DSP before sending the boot table.

Table 3-26 I²C Master Mode Device Configuration

6	5	4	3	2	1	0
Reserved (1)	Mode(1)	Receive I2C Address			Smart Reflex	
					Reserved	

The header format should be in the following format:

19 xx xx yy yy zz zz (for Receive I2C address value in the device configuration of 0x0)

Where:
19 = the slave address (part of the I²C command word not included in the Data block) for the DSP in slave I²C boot mode
xx xx = length
yy yy = checksum
zz zz = boot option

3.5.7 Secondary Stage Bootloader

For each boot mode, the RBL uses a default boot parameter table for that mode and the values for certain parameters from the bootstrap pin configuration by reading the DEVSTAT register. Although this is convenient, it does not give a lot of flexibility in terms of the possible configurations for a given boot mode. Also, there is no provision to configure additional peripherals that might apply to some scenarios. For example, it might be useful to configure DDR to store an application that was downloaded during the Ethernet boot mode.

To facilitate such unique scenarios, the I²C boot mode provides two-stage booting. When the KeyStone devices boot up in I²C mode, they always enter the boot code twice. The first stage might be used to configure the additional peripheral using a boot config table, then point to a second parameter table to be loaded. In the second stage, the second parameter table loaded is used by the RBL to boot in the specified mode. The first stage might also be used to load a boot application using a boot table; at the end of the boot table the RBL transfers control to the entry point of the downloaded boot application to execute.

3.6 SPI Boot Operation

3.6.1 Basic Boot Operation

The boot parameter table (Table 3-27) is used to configure SPI boot. Because SPI is operated through direct register reads and writes, no EDMA configurations are required.

Table 3-27 SPI Boot Parameter Table

Byte Offset	Name	Description
12	Options	Bits 0 & 1 Modes 00 = Load a boot parameter table from the SPI (Default mode) 01 = Load boot records from the SPI (boot tables) 10 = Load boot config records from the SPI (boot config tables) 11 = Reserved Bits 2 - 15 = Reserved
14	Mode	SPI mode, 0-3
16	Address Width	The number of bytes in the SPI device address. Can be 2 or 3 (16 or 24 bit)
18	Data Width	The data width of the device. Can be 8 or 16
20	NPin	The operational mode, 3 or 4 pin
22	Chipsel	The chip select used (valid in 4 pin mode only). Can be 0-3.
24	Read Addr MSW	The first address to read from, MSW (valid for 24 bit address width only)
26	Read Addr LSW	The first address to read from, LSW
28	CPU Freq MHz	The speed of the CPU, in MHz
30	Bus Freq, MHz	The MHz portion of the SPI bus frequency. Default = 5 MHz
32	Bus Freq, kHz	The kHz portion of the SPI buf frequency. Default = 0
End of Table 3-27		

The SPI boot configuration derives the parameters to fill the boot parameter table from the boot device configuration as shown in Table 3-28 and Table 3-29.

Table 3-28 SPI Boot Mode Device Configuration

9	8	7	6	5	4	3	2	1	0
Mode (Clk Pol/Phase)		4, 5 Pin	Addr Width	Chip Select		Parameter Index (1 - 0 bits also used for SR ID)			

Table 3-29 SPI Boot Mode Device Configuration Description

Bit Field	Description
Mode	0 = Data is output on the rising edge of SPICLK. Input data is latched on the falling edge. 1 = Data is output one half-cycle before the first rising edge of SPICLK and on subsequent falling edges. Input data is latched on the rising edge of SPICLK. 2 = Data is output on the falling edge of SPICLK. Input data is latched on the rising edge. 3 = Data is output one half-cycle before the first falling edge of SPICLK and on subsequent rising edges. Input data is latched on the falling edge of SPICLK.
4,5 pin	0 = 4-pin mode used 1 = 5-pin mode used
Addr Width	0 = 16 bit address values are used 1 = 24 bit address values are used
Chip Select	0-3 = The chip select field value
Parameter Table Index	0-3 = Specifies which parameter table is loaded
SR Index	0-3 = Smart Reflex Index
End of Table 3-29	

After initializing the peripherals by the boot parameter table shown in Table 3-21, the boot code reads either a boot parameter table or boot config table that is at the address specified by the first boot parameter table and executes it directly. Similar to the I²C in the boot config table mode, the boot code consist of data in blocks with the first four bytes of data as header which is stripped and the rest of the data blocks are used for booting process.

In Boot Parameter mode, SPI reads the data block from the EEPROM into the internal boot parameter table and reenters the RBL. The table loaded can contain a boot parameter table for any boot mode.

After the boot process completes, the interrupt maps are restored to their default values.

3.7 HyperLink Boot Operation

3.7.1 Basic Boot Operation

HyperLink boot through the ultra-short-range (HyperLink) connection is configured using the four SerDes lanes. The SerDes clock configurations are obtained from the boot configuration field bits as shown in Table 3-30 and Table 3-31.

Table 3-30 HyperLink Boot Mode Device Configuration

6	5	4	3	2	1	0
Reserved	Date Rate		Ref Clock		SR Index	

Table 3-31 HyperLink Boot Device Configuration Description

Bit Field	Description
SR Index	0-3 = Smart Reflex Index
Ref Clock	0 = 156.25 MHz 1 = 250 MHz 2 = 312.5 MHz
Data Rate	0 = 1.25 Gbaud 1 = 3.125 Gbaud 2 = 6.25 Gbaud 3 = 12.5 Gbaud

HyperLink boot code initializes the chip-level interrupt controller to interrupt the DSP after the boot. After setting up the interrupt configuration, the boot ROM executes an idle instruction. The remote device loads the memory directly and generates the HyperLink interrupt. When the boot code is awoken, the interrupt maps are restored to their default values and the DSP branches to the address in DSP BOOT MAGIC address. As with PCI, if the value in DSP BOOT MAGIC address is still 0 after wakeup, the ROM again executes an idle.

HyperLink mapping can be configured by the master, but the bootloader sets up the initial mappings. Table 3-32 shows the initial mappings for KeyStone devices.

Table 3-32 Boot ROM Initialized HyperLink Segment Mapping

Segment	Size	Translated Address	Description
0-N	Size of L2	Global address of core0 L2 to Core N L2	Global L2
N + 1	128k	0x08000000	XMC Config
N + 2	1 MB	0x0bc00000	MSMC Config
N + 3	Size of MSMC memory	0x0c000000	MSMC memory
N + 4	4MB	0x01C00000	Config Regs
N + 5	4MB	0x02000000	Config Regs
N + 6	4MB	0x02400000	Config Regs
N + 7	2MB	0x02800000	Config Regs
N + 8	512B	0x21000000	Config Regs
(N + 9) - 64	4MB	0x80000000 (4MB steps)	DDR memory

Index

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 TI 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. Information of third parties may be subject to additional restrictions.

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.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

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

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Mobile Processors	www.ti.com/omap		
Wireless Connectivity	www.ti.com/wirelessconnectivity		
	TI E2E Community Home Page		e2e.ti.com

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