APU Initialization

The BootROM configures the APU and MIO multiplexer to support the boot process. The state of the MIO pins for each boot mode is described in tables in the Boot Device sections (for example, Table 6-9 for Quad-SPI). The BootROM uses CPU 0 to execute the ROM code. CPU 1 executes the WFE instruction. The caches and TLBs are invalidated. The BootROM configures the MMU and other system resources to meet the needs of the BootROM execution. The state of the APU is described in section 6.3.13 Post BootROM State.

Note: FSBL/User code and operating system software must configure the APU for their own needs and should consider the CPU initialization steps described in section Chapter 3, Application Processing Unit.

6.3.2 BootROM Header

The BootROM requires a header for all master boot modes (flash devices). In JTAG slave boot mode, the BootROM Header is not used and the BootROM code does not load the FSBL/User code.

The BootROM Header parameters are shown in Table 6-5 with their word number, byte address offset, and applicability for the three types of device boot modes.

Note: The BootROM Header is referred to as the Boot Image Header in UG821, *Zynq-7000 All Programmable SoC Software Developers Guide*. They both refer to the same table of parameters provided in Table 6-5.

Table 6-5: **BootROM Header Parameters**

Header Address	32-bit Word	Parameter	Boot Device		
			Secure Usage	Non-Secure Usages	
			OCM	OCM	Execute In Place [6]
0x000 - 0x01F	0 - 7	Interrupt Table for Execution-in-Place	no	no	yes
0x020	8	Width Detection	Quad-SPI	Quad-SPI	Quad-SPI
0x024	9	Image Identification	yes	yes	yes
0x028	10	Encryption Status	yes	yes	yes
0x02C	11	FSBL/User Defined [3]	~	~	~
0x030	12	Source Offset	yes	yes	~
0x034	13	Length of Image	yes	yes	set = 0
0x038	14	FSBL Load Address	~	~	set = 0
0x03C	15	Start of Execution	yes	yes	yes
0x040	16	Total Image Length	note [1]	note [2]	set = 0
0x044	17	QSPI Config Word ,set to 0x01	~	~	~
0x048	18	Header Checksum	yes	yes	yes
0x04C - 0x097	19 - 39	FSBL/User Defined (76-Byte)[3]	~	~	~

Table 6-5: **BootROM Header Parameters** *(Cont'd)*

Header Address	32-bit Word	Parameter	Boot Device		
			Secure Usage	Non-Secure Usages	
			OCM	OCM	Execute In Place [(6)]
`0x0A0 - 0x89F`	40 - 551	Register Initialization (2048-Byte) [(4)]	yes	yes	yes
`0x8A0 - 0x8BF`	552 - 559	Image Header[(3)]	yes	yes	yes
`0x8C0 - 0x8FF`	560 - 576	Partition Header	yes	yes	yes
`0x900`	577 and up	FSBL Image or User Code	192 KB	192 KB	see [(5)]

Notes:

1. In secure mode, the Total Image Length parameter is greater than Length of Image parameter because of encryption.
2. In non-secure OCM mode, the Total Image Length parameter must be set equal to the Length of Image parameter.
3. The FSBL/User Defined parameter and FSBL Image and User Code fields are not used by the BootROM code and do not affect the hardware. See UG821, *Zynq-7000 All Programmable SoC Software Developers Guide* for more information.
4. The addresses that can be accessed by Register Initialization is restricted, see Table 6-7. The secure boot mode has more address restrictions than a non-secure boot.
5. The size of the FSBL image (or User code) for Execute-in-place depends on the allowed capacity of the boot device less the 0x8C0 (the size of the BootROM Header). The maximum Quad-SPI size is described in section 6.3.4 Quad-SPI Boot. For NOR, refer to section 6.3.6 NOR Boot.
6. To select the execute-in-place feature, set the Length of Image and Total Image Length parameters to 0 and load the PS interconnect address into the Source Offset.

Interrupt Table for Execution-in-Place — `0x000 to 0x01C`

Eight 32-bit words are reserved for interrupt mapping. This is useful for execute-in-place for NOR and Quad-SPI devices. It allows the CPU vector table to be managed in two ways. The first is to use the MMU to remap the flash linear address space to `0x0`. The second method to manage the vector table location is to use the coprocessor VBAR register. For more information on setting this register, refer to the *ARM v7-AR Architecture Reference Manual* (listed in Appendix A, Additional Resources).

Width Detection — `0x020`

Width Detection is required for the Quad-SPI boot mode. Ensure that the BootROM Header includes the value of `0xAA995566` so the BootROM can determine the maximum hardware I/O data connection width of the flash device(s). This value helps the BootROM to determine the data width of a single Quad-SPI device (x1, x2, or x4) and to detect a second device in 8-bit parallel I/O configuration. If this value is not present for the Quad-SPI boot mode then the BootROM lockdowns the system and generates an error code. The error code number depends on other conditions. The error codes are listed in Table 6-20, page 199. Details of the detection operation are explained in section 6.3.4 Quad-SPI Boot.

Image Identification — `0x024`

This word has a mandatory value of `0x584C4E58`, `'XLNX'`. This value allows the BootROM (along with the header checksum field) to determine that a valid BootROM Header is present. If the value is not matched, the BootROM code performs a Header search if the boot mode is either Quad-SPI, NAND, or NOR. If the boot mode is SD card, the BootROM lockdowns the system and generates an error code.

Encryption Status — `0x028`

Encryption Status determines if the boot is secure (the boot image is encrypted) or non-secure mode. Valid values for this field are:

- `0xA5C3C5A3` Encrypted FSBL/User code (requires eFUSE key source).
- `0x3A5C3C5A` Encrypted FSBL/User code (requires battery-backed RAM key source).
- Not `0xA5C3C5A3` or `0x3A5C3C5A`. Non-encrypted FSBL/User code (no key).

The eFuse states and the encryption status word determines the source of the encryption key, if any. The valid combination are shown in Table 6-6.

Table 6-6: **BootROM Requirements for Encryption Status Word**

eFuse States (described in Table 32-2, page 779)			
eFuse Secure Boot	**Not Blown**	**Not Blown**	**Blown**
BBRAM Key Disable	**Not Blown**	**Blown**	**Don't Care**
Encryption Status Word			
Non-secure	Okay	Okay	Lockdown
Secure with BBRAM Key	Okay	Lockdown	Lockdown
Secure with eFuse Key	Okay	Okay	Okay

FSBL/User Defined — `0x02C`

This word is used to store the BootROM header version. Refer to UG821, *Zynq-7000 All Programmable SoC Software Developers Guide* for more information. The BootROM does not interpret or use this field.

Source Offset — `0x030`

This parameter contains the number of bytes from the beginning of the valid BootROM Header to where the FSLB/User code image resides. This offset must be aligned to a 64-byte boundary and must be at or above address offset `0x8C0` from the beginning of the BootROM Header.

Length of Image — `0x034`

This word contains the byte count of the load image to transfer to the OCM. For non-secure mode, the Length of Image equals the Total Image Length parameter and has a maximum value of 192 KB. For secure mode, the Length of Image is set equal to the length of the image after it has gone through the authentication and decryption process steps. In this case, the Length of Image is always less than 192 KB because of the encryption overhead.

A value of zero with a Quad-SPI or NOR flash mode causes the BootROM to execute the FSBL/User code from the associated flash device without copying the image to OCM (execute-in-place).

FSB Load Address— `0x038`

Destination address to which to copy the FSBL.

Start of Execution — `0x03C`

- This is a byte address that is relative to the start of system memory and is used for both executing the FSBL/User code from the OCM or using the optional execute-in-place feature of Quad-SPI and NOR boot modes. The byte address must be aligned to a 64-byte boundary. FSBL/User code executes from OCM:
  - Execution attempts outside of the OCM memory address space cause a secure lockdown.
  - Non-secure mode: address must be equal to or greater than `0x0` and less than `0x30000`.
  - Secure mode: the address must equal `0x0`.
- Execute-in-place FSBL/User code:
  - The address must point to a location within the first 16 MB of memory for x4 Quad-SPI and the first 32 MB of memory for NOR and dual x8 parallel Quad-SPI boot modes.

Total Image Length — `0x040`

This is the total number of bytes loaded into the OCM by the BootROM from the flash memory.

For non-secure boot, the Total Image Length parameter must be set equal to the Length of Image parameter.

For secure images, the Total Image Length parameter includes the HMAC header, the encryption overhead and the alignment requirements and is always larger than the Length of Image parameter. The Total Image Length parameter is provided by the design tools.

QSPI Config Word — `0x044`

QSPI configuration word, hard-coded to `0x01`.

Header Checksum — `0x048`

The is the checksum value of the header which is checked prior to using the data within the header. The checksum is calculated by summing the words from `0x020` to `0x044` and inverting the result.

FSBL/User Defined— `0x04C to 0x097`

This can be used in Bootgen using the udf_data field of the BIF file. Refer to Appendix-A in UG821, *Zynq-7000 All Programmable SoC Software Developers Guide* for more information.

Boot Header Table Offset — 0x098

Pointer to Image Header table.

QSPI Config Word — 0x09C

Pointer to the Partition Header table.

Register Initialization Parameters — 0x0A0 to 0x89C

This region contains 256 pairs of address and data words that can be used to initialize PS registers for the MIO multiplexer, boot device clocks, and other functions before the FSBL/User code is accessed from the boot device; either to copy the image to the OCM or to execute-in-place. The register writes are commonly used to optimize the boot device interface and set its clock frequency to maximize performance.

A register initialization pair appears as two 32-bit words, first a register address, then a register write value. Register initializations can be in any order, and the same register can be initialized with different values as many times as desired. The register initialization is performed prior to copying the FSBL/User code so the user can modify the default reset register values to reduce the time to access the code and process it.

The BootROM stops processing the Register Initialization list when either the address register = 0xFFFF_FFFF or the end of the list (256 address/write data pairs).

Usage of the register initialization parameters are explained in the “Register Initialization to Optimize Boot Times” section of section 6.3.3 BootROM Performance.

Restricted Addresses

The register address space for the Register Initialization address-data writes is restricted. Register addresses outside of the allowed address range cause the BootROM to lockdown the system and generate error code 0x2111. The allowed register accesses depend on the boot mode and are listed in Table 6-7.

These restrictions are enforced by the BootROM. They do not apply when the FSBL/User code begins to execute. The BootROM screens the register initialization writes and disallows certain addresses from being accessed.

Table 6-7: **BootROM Accessible Address Ranges for Register Initialization**

Control Registers	Non-Secure Boot Mode		Secure Boot Mode
	Ranges	Exceptions to Range[1]	
UART 1	E000_1000 to E000_1FFC	~	No
Quad-SPI	E000_D000 to E000_DFFC	~	No
SMC	E000_E000 to E000_EFFC	~	No
SDIO 0	E010_0004 to E010_0FFC	E010_0058	No
DDR Memory	F800_6000 to F800_6FFC	~	No

Table 6-7: **BootROM Accessible Address Ranges for Register Initialization *(Cont'd)***

Control Registers	Non-Secure Boot Mode		Secure Boot Mode
	Ranges	Exceptions to Range[1]	
SLCR registers			
PLL, Peripheral, AMBA and CPU clock controls	F800_0100 to F800_0234	Reserved: F800_01B0 PS Reset Ctrl: F800_0200	PLL, Peripheral and PL clock controls: F800_0100 to F800_01AC
SWDT Reset	F800_024C	~	
SWDT clock, TZ configuration, PS ID code, DDR configuration, MIO pins, SD card WP/CD routing	F800_0304 to F800_0834	~	
Reserved	F800_0A00 to F800_0A8C	~	
Reserved, GPIO and DDR I/O controls	F800_0AB0 to F800_0B74	~	
UART 0, USB, I2C, SPI, CAN, GPIO, GigE, TTC, DMAC, SWDT, DDR, DevC, AXI HP	Not accessible		Not accessible

Notes:

1. The registers in this column are not accessible by the Register Initialization writes.

FSBL/User Defined — `0x8A0 - 0x8BF`

This memory area may be used by the FSBL or User code. Refer to UG821, *Zynq-7000 All Programmable SoC Software Developers Guide* for more information.

FSBL Image or User Code Start Address — `0x8C0`

The FSBL Image or User Code must start at or above this location. The location is pointed to by the Source Offset parameter and must be aligned to 64 bytes.

6.3.3 BootROM Performance

The BootROM performance is an important factor to the total bring-up time of the system that includes: Power-up, BootROM execution, FSBL/User code execution, U-boot time, and OS loading time. The entire boot and configuration process is explained in section 6.4 Device Boot and PL Configuration.

Below are a few topics related to BootROM execution that include using the Register Initialization mechanism in the BootROM Header to optimize the bandwidth of the flash device interface. The flash device bandwidth is the single most important factor in speeding up boot times.