UG0905
User Guide
PolarFire SoC FPGA System Services

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

Contents

Figures

Tables

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the current publication.

1.1 Revision 1.0

The first publication of this document.

2 System Services

PolarFire SoC devices include a microprocessor-based system controller, which handles the system service requests. System services are system controller actions initiated by PolarFire SoC Microprocessor Subsystem (MSS). MSS communicates with the system controller over system controller bus (SCB) by write/read to the MSS SCB register space.

Writing a 16-bit system service descriptor to the SCB by the MSS triggers a service request interrupt to the system controller. The lower seven bits of the descriptor specify the service to be performed. The upper nine bits specify address offset (0–511) in the 2 KB mailbox RAM to access word length of four bytes memory. The mailbox address (MBOXADDR[10:0]) specifies the service-specific data structure that is used for any additional inputs to or outputs from the service. The MSS writes additional parameters to the mailbox before requesting system service. The following table lists the system service descriptor bits.

Table 1 • **System Service Request Descriptor**

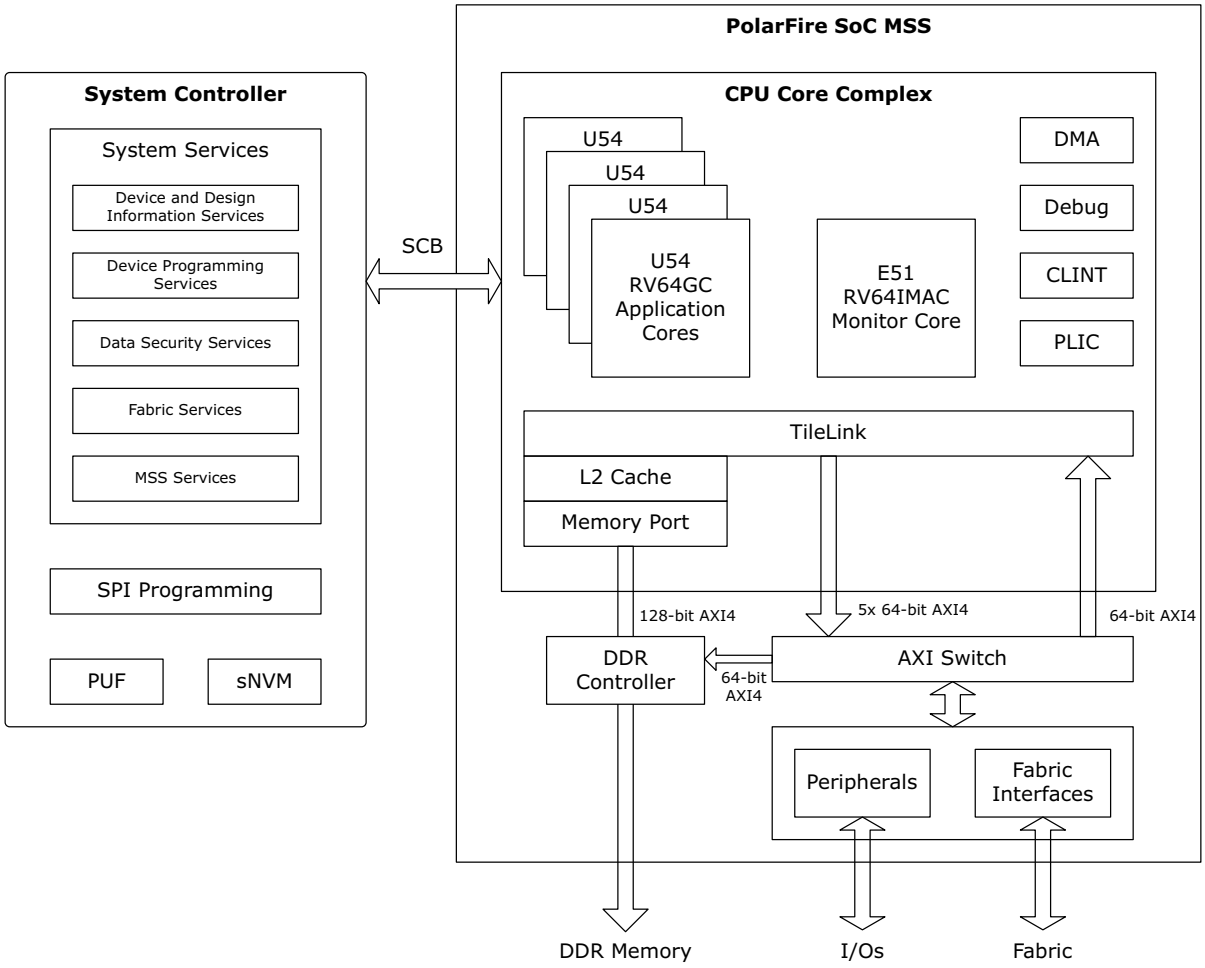
System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Specifies the address in mailbox RAM to access minimum four bytes of memory.
6:0	SERVICEID	Service command for system controller to execute the request.

The system services are grouped into the following categories:

- Device and Design Information Services
- Design Programming Services
- Data Security Services
- Fabric Services
- MSS Services

The following figure shows the PolarFire SoC with System Controller.

Figure 1 • **PolarFire SoC with System Controller**



2.1 Device and Design Information Services

These services return information about the device and current user design. The requested information is copied to a location whose address is included in the service descriptor. The size of the data returned is service dependent. For information about the return status of device and design information services, see Table 75, page 30.

2.1.1 Serial Number Service

Fetches the 128-bit device serial number.

Table 2 • **Serial Number Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 3, page 4.
6:0	00H	Serial number service command

The following table lists the Serial Number Service mailbox format.

Table 3 • **Serial Number Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	16	DSN	Output	Device Serial Number

2.1.2 USERCODE Service

Fetches the 32-bit USERCODE.

Table 4 • **USERCODE Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 5, page 4.
6:0	01H	USERCODE service command

The following table lists the USERCODE Service mailbox format.

Table 5 • **USERCODE Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	4	USERCODE	Output	Device USERCODE

2.1.3 Design Information Service

Returns design information including 256-bit user defined design ID, 16-bit design version, and 16-bit design back-level protection value.

Table 6 • **Design Information Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 7, page 4.
6:0	02H	Design Information service command

The following table lists the Design Information Service mailbox format.

Table 7 • **Design Information Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	32	DESIGNID	Output	256-bit user-defined design ID
32	2	DESIGNVER	Output	16-bit design version
34	2	BACKLEVEL	Output	16-bit design back-level

2.1.4 Device Certificate Service

Fetches the device's Supply Chain Assurance Certificate from pNVM. The certificate data is stored as a 1024-bit entity but the actual certificate size may be smaller. Any excess bytes should be discarded by the user.

The device validates the certificate by checking its signature using the Microsemi public key, MCPK. In addition:

- The certificate DSN is checked against the device's serial number (DSN)
- The certificate public key is checked by recalculating the value using factory key and comparing against the certificate

In the event of an error, the certificate content is still returned for inspection.

Table 8 • **Design Certificate Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 9, page 5.
6:0	03H	Design Certificate service command

The following table lists the Design Certificate Service mailbox format.

Table 9 • **Design Certificate Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	1024	CERTIFICATE	Output	Device Certificate

2.1.5 Read Digests Service

Returns the stored digests for the device. The following table lists the returned digests format.

Table 10 • **Read Digest Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 11, page 5.
6:0	04H	Read Digest service command

The following table lists the Read Digest Service mailbox format.

Table 11 • **Read Digests Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	576	DIGESTS	Output	Digest Array

Table 12 • **Returned Digests Format**

Offset (byte)	Size (bytes)	Value	Description
0	32	FD	Fabric digest
32	32	CCDIGEST	Fabric configuration data digest
64	32	SNVMDIGEST	sNVM ROM pages digest
96	32	ULDIGEST	User security segment digest

Table 12 • **Returned Digests Format *(continued)***

Offset (byte)	Size (bytes)	Value	Description
128	32	UKDIGEST0	Digest of user key segment containing SRAM-PUF data
160	32	UKDIGEST1	Digest of user key segment containing UEK (User EC key)
192	32	UKDIGEST2	Digest of user key segment containing UPK1
224	32	UKDIGEST3	Digest of user key segment containing UEK1
256	32	UKDIGEST4	Digest of user key segment containing DPK
288	32	UKDIGEST5	Digest of user key segment containing UPK2
320	32	UKDIGEST6	Digest of user key segment containing UEK2
352	32	UPDIGEST	Digest of permanent lock security segments
384	32	FDIGEST	Digest of factory lock segment, factory key segment in pNVM, and system controller ROM.
416	32	UKDIGEST7	User Key digest 7
448	32	ENVMDIGEST	ENVM Digest
480	32	UKDIGEST8	UKDIGEST8 for MSS Boot Information
512	32	UKDIGEST9	SNVM_RW_ACCESS_MAP Digest
544	32	UKDIGEST10	SBIC revocation digest

2.1.6 Query Security Service

Fetches non-volatile states of user security locks. The following table lists the description of returned LOCKS array.

Table 13 • **Query Security Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 14, page 6.
6:0	05H	Query Security service command

The following table lists the Query Security Service mailbox format.

Table 14 • **Query Security Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	33	Locks	Output	Lock Array

Table 15 • **Returned LOCKS Array**

Byte	Bit	Lock	Description
0	0	UL_DEBUG	Debug instructions disable
	1	UL_SNVM_DEBUG	sNVM debug disable
	2	UL_LIVEPROBE	Live probes disable
	3	UL_UJTAG	User JTAG interface disable
	4	UL_JTAG_BS	JTAG boundary scan disable
	5	UL_TVS_MONITOR	External access to System TVS monitor disable
	6	UL_JTAG_MONITOR	JTAG fabric monitor enable
	7	UL_JTAG	JTAG TAP disable
1	0	UL_PLAINTEXT	Plaintext passcode unlock disable
	1	UL_FAB_PROTECT	Fabric erase/write disable
	2	UL_EXT_DIGEST	External digest check disable
	3	UL_VERSION	Replay protection enable
	4	UL_FACT_UNLOCK	Factory test disable
	5	UL_IAP	IAP disable
	6	UL_EXT_ZEROIZE	External zeroization disable
	7	UL_SPI_SLAVE	SPI port disable
2	0	UL_USL	UFS UL segment protect
	1	UL_BS_AUTHENTICATE	External bitstream authentication disable
	2	UL_BS_PROGRAM	External bitstream program mode disable
	3	UL_BS_VERIFY	External bitstream verify mode disable
	4	UL_BITS_KEYMD[0]	Bitstream key mode disable
	5	UL_BITS_KEYMD[1]	Bitstream key mode disable
	6	UL_BITS_KEYMD[2]	Bitstream key mode disable
	7	UL_BITS_KEYMD[3]	Bitstream key mode disable
3	0	UL_BITS_KEYMD[4]	Bitstream key mode disable
	1	UL_BITS_KEYMD[5]	Bitstream key mode disable
	2	UL_BITS_KEYMD[6]	Bitstream key mode disable
	3	UL_BITS_KEYMD[7]	Bitstream key mode disable
	4	UL_BITS_KEYMD[8]	Bitstream key mode disable
	5	UL_BITS_KEYMD[9]	Bitstream key mode disable
	6	UL_BITS_KEYMD[10]	Bitstream key mode disable
	7	UL_BITS_KEYMD[11]	Bitstream key mode disable

Table 15 • **Returned LOCKS Array *(continued)***

Byte	Bit	Lock	Description
4	0	UL_BITS_KEYMD[12]	Bitstream key mode disable
	1	UL_BITS_KEYMD[13]	Bitstream key mode disable
	2	UL_BITS_KEYMD[14]	Bitstream key mode disable
	3	UL_BITS_KEYMD[15]	Bitstream key mode disable
	4	UL_KEYMD[0]	Global key mode disable
	5	UL_KEYMD[1]	Global key mode disable
	6	UL_KEYMD[2]	Global key mode disable
	7	UL_KEYMD[3]	Global key mode disable
5	0	UL_KEYMD[4]	Global key mode disable
	1	UL_KEYMD[5]	Global key mode disable
	2	UL_KEYMD[6]	Global key mode disable
	3	UL_KEYMD[7]	Global key mode disable
	4	UL_KEYMD[8]	Global key mode disable
	5	UL_KEYMD[9]	Global key mode disable
	6	UL_KEYMD[10]	Global key mode disable
	7	UL_KEYMD[11]	Global key mode disable
6	0	UL_KEYMD[12]	Global key mode disable
	1	UL_KEYMD[13]	Global key mode disable
	2	UL_KEYMD[14]	Global key mode disable
	3	UL_KEYMD[15]	Global key mode disable
	4	UL_SNVM_PROTECT	sNVM bitstream write protection enable
	5	UL_EXT_CHALLENGE	CHALLENGE instruction disable
	6	UL_UEK_PROTECT	UEK overwrite protection
	7	UL_HWM	High Water Mark Reset disable
7	0	UL_ENVM_PROTECT	Disable bitstream programming of eNVM
	1	UL_USER_KEY	User Key1 write protect
	2	UL_USER_KEY2	User Key2 write protect
	3	UP_FACTORY	Permanent factory test disable
	4	UP_DEBUG	Permanent debug disable
	5	UP_FABRIC	Permanent fabric write protect
	6	UP_UPK1	Permanent disable of UPK1
	7	UP_UPK2	Permanent disable of UPK2
8	0	UP_DPK	Permanent disable of DPK
	1	UP_PROTECT	Write disable for UPERM segment
9:11		RESERVED	
12	0	UATHENA_ENA	User F5200 Enable
12	1:2	UATHENA_MODE	User F5200 mode

Table 15 • **Returned LOCKS Array *(continued)***

Byte	Bit	Lock	Description
12	3:5	U_CLKMON_FREQ	System Controller Clock Frequency monitor configuration
13		RESERVED	Reserved
14-15		PORDIGEST[15:0]	This field specifies a mask of device digests that should be checked upon each power-up of the user design
16-31		HWM	High Water Mark value All 1s are returned if the HWM is invalid.
32		PORDIGEST[23:16]	This field specifies a mask of device digests that should be checked upon each power-up of the user design

2.1.7 Read Debug Information Service

Fetches debug information on programming, user initialization, device programming cycle count, and In-application programming (IAP) actions. The device programming cycle count increases for device PROGRAM and ERASE actions. The following table lists the debug information reported by this service.

Table 16 • **Read Debug Information Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 17, page 9.
6:0	06H	Read Debug service command

Table 17 • **Debug Information**

Size (Bytes)	Byte Offset	Parameter	Description
32	0	Reserved	Reserved
4	32	TOOL_INFO	Reflects the tool specific data passed in during programming. IAP sets this field to 0.
1	36	TOOL_TYPE	Tool type used to program device: 1: JTAG, 2: IAP, 3: SPI_SLAVE
4	37	Reserved	Reserved
1	41	FRAME_ERRORCODE	An error has occurred during bitstream frame processing and the error is identified by the FRAME_ERRORCODE field. See Table 18, page 10.
6	42	Reserved	Reserved
1	48	UIC_STATUS	Device and design initialization Status. 0 – Successful completion. Others – Device and design initialization failed.
11	49	Reserved	Reserved
4	60	CYCLECOUNT	Programming cycle count.
1	64	IAP ERRORCODE	IAP Error Information. Returns ERRORCODE 21-27, see Table 18, page 10.
7	65	Reserved	Reserved
4	72	IAP Location	SPI address that was last run in IAP

Table 17 • **Debug Information *(continued)***

Size (Bytes)	Byte Offset	Parameter	Description
4	76	SYSCTRL_STATUS	System Controller reset status
4	80	RESET_REASON	Reason for last device reset

Table 18 • **ERRORCODE**

ERRORCODE	Description	Additional Notes
0	No error	
1	Bitstream authentication failed	Invalid bitstream or wrong key used.
2	Unexpected data received	Additional data received after end of bitstream component.
3	Invalid/corrupt encryption key	The requested key mode is disabled or the key could not be read/reconstructed.
4	Invalid component header	Invalid bitstream
5	Back level not satisfied	Bitstream version is older than that of the current back level value set in the device.
6	Illegal bitstream key mode	Bitstream key mode not initialized or bitstream key mode is disabled by user security.
7	DSN binding mismatch	Bitstream rejected because DSN in the bitstream does not match with the DSN present in the device. A bitstream can be bound to device's unique DSN such that only a specific device can be programmed with that bitstream.
8	Illegal component sequence	Incorrect bitstream
9	Insufficient device capabilities	Bitstream is rejected because the capabilities specified in the bitstream do not match the target device's capabilities.
10	Incorrect DEVICEID	Bitstream is rejected because an attempt by the DEVICEID specified in the bitstream does not match the part identification field (for example, MPF300, MPF500 and so on) of the target device.
11	Unsupported bitstream protocol version (bitstream regeneration required)	Bitstream is rejected because of an attempt made by the old version of a device to decode a bitstream created in new format or by the new version of a device to decode a bitstream created in old format.
12	Verify not permitted on this bitstream	Verify programming action disabled in the bitstream.
13	Invalid Device Certificate	Device certificate is invalid or not present.
14	Invalid DIB	Device integrity bits (DIB) are invalid.
21	Device not in SPI Master Mode	Error may occur only when bitstream is executed through IAP mode. The System Controller SPI controller is not configured in master mode.
22	No valid images found	Error may occur when bitstream is executed through Auto Update mode. Occurs when No valid image pointers are found.
23	No valid images found	Error may occur when bitstream is executed through IAP mode via Index Mode. Occurs when No valid image pointers are found.

Table 18 • **ERRORCODE**

ERRORCODE	Description	Additional Notes
24	Programmed design version is the same as the Auto Update image found	Error may occur when bitstream is executed through Auto Update mode.
25	Reserved	
26	Selected image was invalid and no recovery was performed due to valid design in device.	Error may occur only when bitstream is executed through Auto Update or IAP mode. Error could also occur due to BACKLEVEL protection.
27	Selected and Recovery image failed to program	Error may occur only when bitstream is executed through Auto Update or IAP mode.
127	Abort	Non-bitstream instruction executed during bitstream loading.
128	NVMVERIFY	Fabric or security segment verification failed.
129	PROTECTED	Device security prevented modification of non-volatile memory.
130	NOTENA	Programming mode not enabled
131	PNVMVERIFY	pNVM verify operation failed
132	SYSTEM	System hardware error (PUF or DRBG)
133	BADCOMPONENT	An internal error was detected in a bitstream component payload.
134	HVPROGERR	Failure in programming subsystem.
135	HVSTATE	Error in the programming subsystem.

2.1.8 Read eNVM Parameters Service

Retrieves all parameters needed for eNVM operation and programming.

Table 19 • **Read eNVM Parameters Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 20, page 11.
6:0	07H	Read eNVM service command

The following table lists the Serial Number Service mailbox format.

Table 20 • **Serial Number Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	256	eNVM	Output	eNVM Parameters. For information about eNVM parameters, see PolarFire SoC Datasheet (to be released).

2.2 Design Programming Services

An IAP image contains the image descriptor, bitstream, and optional design initialization data. The design programming services are used to authenticate entire IAP image, bitstream portion, or program the device. For information about the return status of Design Programming Services, see Table 75, page 30.

2.2.1 Bitstream Authentication Service

Prior to using the IAP service, it may be required to first validate the new bitstream before committing the device to reprogramming, thus avoiding the need to invoke recovery procedures if the bitstream is invalid.

The bitstream authentication service analyzes a bitstream image stored in SPI flash and checks for all conditions which may result in an authentication error. While the authentication is in progress, the user design continues to operate normally, but without access to SPI flash and system services until the authentication process is complete.

The spi_flash_address parameter passed to this service specifies the address within SPI Flash where the bitstream is stored.

If the authentication service is called while a new bitstream is being loaded through the JTAG interface, the system service takes precedence and the JTAG interface is stalled during the authentication and will ultimately fail. .

Table 21 • **Bitstream Authentication Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 22, page 12.
6:0	23H	Bitstream authentication service command.

The following table describes the bitstream authentication service mailbox format.

Table 22 • **Bitstream Authentication Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	4	SPIADDR	Input	Address of the bitstream in SPI flash. If the external SPI flash device does not support 32-bit addresses, SPIADDR[31:24] is ignored.

2.2.2 IAP Image Authentication Service

Allows the user to validate an IAP image stored in SPI flash. The service authenticates the entire IAP image containing the image descriptor, the referenced bitstream, and optional initialization data. If the image is authenticated successfully, the image is guaranteed to be valid when used by an IAP programming service.

The SPI_IDX parameter passed to this service identifies the index in the SPI directory to be used. To support recovery, SPI_IDX = 1 must be an empty slot and the recovery image must be located in SPI_IDX = 0. Since SPI_IDX = 1 must be an empty slot, it shouldn't be passed into the system service. The following table lists the fields contained in an IAP image authentication service request.

Table 23 • **IAP Image Authentication Service Request**

System Service Descriptor Bit Field	Value	Description
15		Reserved.
14:7	IMAGEID[7:0]	Identifies the image index in the SPI directory for image authentication.
6:0	22H	IAP Image Authenticate service command.

2.3 Data Security Services

The data security services are used to authenticate the device, generate unique random number, and store the encrypted data. For information about the return status of Data Security Services, see Table 75, page 30.

2.3.1 Digital Signature Service

Takes a user-supplied SHA-384 hash and signs it with the device's 384-bit private "factory" EC key, FEK, which is the private half of the key pair whose public key (DCPK) is certified by Microsemi in the device's X.509-compliant supply chain assurance certificate. The resulting P-384 ECDSA signature can either be formatted using ASN.1 DER or simply returned in a raw format compatible with the user cryptoprocessor. Since ECDSA requires the use of a nonce, the service returns a different result each time, even if the hash input is the same.

The system controller cryptoprocessor does not directly support generating a nonce with the required numerical range required for ECDSA. It is therefore possible that the generated nonce is rejected, in which case a new nonce is automatically generated until a good value is found. This makes the execution time of this service non-deterministic, however, the probability of an out-of-range nonce being initially generated is extremely low and the probability of a second bad nonce is infinitesimal.

SIGNATURE = ECDSA (FEK, HASH).

If the Raw format is selected, the `SIGNATURE` field contains two unsigned little-endian 12-word (48 byte) values compatible with the user cryptoprocessor.

If the DER format is selected, the `SIGNATURE` field is returned in a minimal length DER encoding using a maximum of 104 bytes. If the encoded signature is less than 104 bytes, the output is padded with zeroes. The extra bytes, if any, must be deleted by the user.

Table 24 • **Digital Signature Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 25, page 13.
6:0	19H	Digital signature Raw format service command
	1AH	Digital Signature DER format service command

The following table lists the Digital Signature Service mailbox format.

Table 25 • **Digital Signature Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	48	HASH	Input	SHA384 hash to be signed
48	96 (Raw)	SIGNATURE	Output	ECDSA signature (r, s)
	104 (DER)			

2.3.2 Secure NVM (sNVM) Services

sNVM data can be stored in any of the following formats:

- Non-authenticated plaintext,
- Authenticated plaintext
- Authenticated ciphertext

When the data is authenticated, 236 bytes of storage per page is available. When the data is not authenticated, 252 bytes can be stored. Non-authenticated plaintext provides the fastest access time and authenticated ciphertext provides the highest level of security. When authentication is used, a user-provided 96-bit key USK is used to enhance security.

sNVM can be marked as ROM during bitstream programming. In this case, sNVM cannot be modified by the Secure NVM services but can be read.

2.3.2.1 Secure NVM Write Service

Provides write access to pages in the sNVM. Data can be stored as encrypted and authenticated ciphertext, authenticated plaintext, and non-authenticated plaintext.

For authenticated plaintext and authenticated ciphertext, a 512-bit sNVM master key (SMK) is the primary key used, with 256-bits allocated for authentication and 256 bits for encryption. SMK is common for all sNVM pages. In addition, a 96-bit user-supplied key, USK, is used to protect each sNVM page independently. The USK is not stored on the device but must be presented to sNVM read system service to correctly retrieve the data.

For crypto-enabled options, the system controller uses AES-256 in synthetic initialization vector (SIV) mode, which supports authenticated encryption. In SIV mode, the IV used for the encryption function is computed from the data, preventing IV misuse, and doubles as the authentication tag. The computed 128-bit IV is stored in the same page as the user data, reducing the available space for user data by 16 bytes compared to the non-authenticated plaintext-only option.

Besides the user-supplied plaintext data, PolarFire SoC FPGAs also submit additional metadata for authentication that effectively provides a "tweak" to the encryption and authentication functions. Some of the data included are the page address and the page write-counter. This means that the ciphertext and the authentication tag are different even if the same data is written to two different sNVM pages, or even if the same data is written to the same page again (since the page-write counter advances).

The USK is used as another element in the "tweak". Without the same 96-bit USK as was used during the write command, the read command fails authentication (and could not possibly decrypt correctly, either). The user can choose to set this key differently for each page, or for groups of pages, or the same for all pages—either as a secret key for added security, or to a frivolous value such as all zeroes if this feature is not needed.

sNVM modules marked as ROM cannot be overwritten by this service. The service cannot be used to create ROM modules (write-protected pages). ROM is declared when a bitstream is generated, and a page's ROM status can only be changed with a new bitstream, and not at run-time.

Table 26 • **Secure NVM Write Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 27, page 14 and Table 28, page 15.
6:0	10H	Non-authenticated plaintext service command
	11H	Authenticated plaintext service command
	12H	Authenticated ciphertext service command

The following table lists the Secure NVM Write Service Mailbox Format for Non-authenticated plaintext (10H).

Table 27 • **Secure NVM Write Service Mailbox Format (10H)**

Offset	Length (bytes)	Parameter	Direction	Description
0	1	SNVMADDR	Input	sNVM address
1	3	RESERVED		Reserved
4	252	DATA	Input	Data to write to sNVM

The following table lists the Secure NVM Write Service Mailbox Format for authenticated plaintext (11H) and Authenticated ciphertext (12H).

Table 28 • **Secure NVM Write Service Mailbox Format (11H, 12H)**

Offset	Length (bytes)	Parameter	Direction	Description
0	1	SNVMADDR	Input	sNVM address
1	3	RESERVED		Reserved
4	236	DATA	Input	Data to write to sNVM
240	12	USK	Input	User Secret Key

2.3.2.2 Secure NVM Read Service

Provides access to the data stored by the Secure NVM Write service or data programmed via a bitstream. If the data is programmed using authentication, the USK key used at the time of programming must also be provided.

Table 29 • **Secure NVM Write Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 30, page 15.
6:0	18H	Secure NVM Read service command

The following table lists the Secure NVM Read Service Mailbox Format (18H).

Table 30 • **Secure NVM Read Service Mailbox Format (18H)**

Offset	Length (bytes)	Parameter	Direction	Description
0	1	SNVMADDR	Input	sNVM address
1	3	RESERVED		Reserved
4	12	USK	Input	User Secret Key (ignored if page is plaintext)
16	4	ADMIN	Output	Page admin data
20	236 or 252	DATA	Output	Data read from sNVM. 236 bytes of data per page is available when the data is authenticated. 252 bytes of data per page is available when the data is not authenticated.

2.3.3 PUF Emulation Service

Provides a mechanism for authenticating a device, or for generating pseudo-random bit strings that can be used for many different purposes. The service accepts a 128-bit challenge and an 8-bit optype, and returns a 256-bit response unique to the given challenge, optype, and device.

```
RESPONSE = KeyTree(PEK, OPTYPE, CHALLENGE)
```

Where `PEK` is the factory-defined PUF emulation key and `KeyTree` is a function that uses the 8-bit `OPTYPE` concatenated with the 128-bit `CHALLENGE` to navigate a binary key tree with the 256-bit secret `PEK` at its root. The leaf of the tree that is computed as a result of the 136 internal hashing operations (one for each level in the binary tree), is a 256 bit secret. The root key, `PEK`, the result, `RESPONSE`, and the intermediate results are protected against side-channel attacks due to the nature of the protocol, plus the fact that the SHA algorithm implemented in the system controller's cryptoprocessor also has strong DPA countermeasures. The `OPTYPE` and `CHALLENGE` are not protected against side-channel leakage.

The OPTYPE allows the user to conceptualize that there are 256 different 128-bit key trees, each with 2^{128} possible output responses, which can be put to different uses without much danger of collision.

The function emulates a "strong PUF", which means that it takes a cryptographically large challenge space and computes a pseudo-random repeatable output response from it, but in this implementation, it does not use unclonable physical properties developed during the manufacturing of the device for the challenge-response calculation, instead using classical cryptographic algorithms; thus the "emulation" disclaimer. The root key PEK is, however, protected as an encrypted/authenticated PUF key code, so the unclonable physical properties of the PolarFire SoC device do enter into the reconstruction of the PUF secret and decryption of the key code to unwrap PEK for use in this function.

There are many uses in cryptography for such a per-device unique, pseudo-random function. One use is to identify a particular chip by first recording (possibly several) challenge-response pairs, then later seeing if the target chip provides the same response as expected for one of the recorded challenges-response pairs. Another application is deriving many keys from one.

Table 31 • **PUF Emulation Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 32, page 16.
6:0	20H	PUF Emulation service command

The following table lists the PUF Emulation Service Mailbox Format.

Table 32 • **PUF Emulation Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	1	OPTYPE	Input	Operational type
1	3	RESERVED		Reserved
4	16	CHALLENGE	Input	Challenge input
20	32	RESPONSE	Output	Response output

2.3.4 Nonce Service

Generates a 256-bit random number derived from the startup states of a dedicated SRAM. The nonce service provides the user with the ability to strengthen the NRBG of the User Cryptoprocessor random bit generator by providing an alternate entropy source to use as additional seed data in its DRBG functions.

NONCE = KeyTree256(PUK, 0, PUFSEED)

Where, PUFSEED is a 256-bit conditioned true random output of the SRAM-PUF. PUK is a 256-bit device-generated nonce set in the factory.

To generate maximum entropy and forward and backward resistance, the SRAM-PUF is automatically power-cycled before generating the seed.

Table 33 • **Nonce Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 34, page 17.
6:0	21H	Nonce service command

The following table lists the Nonce Service Mailbox Format.

Table 34 • **Nonce Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	32	NONCE	Output	Generated nonce

2.4 Fabric Services

Fabric services are used to calculate digests of non-volatile memories and program the device. For information about the return status of Fabric Services, see Table 75, page 30.

2.4.1 Digest Check Service

Recalculates digests of selected non-volatile memories and compares against stored values. The OPTIONS parameter passed in the digest check service indicates the area for which the digest check must be performed.

Table 35 • **Digest Check Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 36, page 17.
6:0	47H	Digest Check service command

The following table lists the Digest Check Service mailbox format.

Table 36 • **Digest Check Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	4	OPTIONS	Input	Digest options. See Table 37, page 17.
4	4	DIGESTERR	Output	See Table 38, page 18.

Table 37 • **OPTIONS[31:0]**

OPTIONS	Name	Description
0	CHECK FABRIC	Enables fabric digest
1	CC	Enables digest of fabric configuration data
2	sNVM	Enables digest of sNVM pages marked as ROM
3	UL	Enables digest of user security segment
4	UKDIGEST0	Enables digest of user key segment containing SRAM-PUF data
5	UKDIGEST1	Enables digest of user key segment containing UEK (User EC key)
6	UKDIGEST2	Enables digest of user key segment containing UPK1
7	UKDIGEST3	Enables digest of user key segment containing UEK1
8	UKDIGEST4	Enables digest of user key segment containing DPK
9	UKDIGEST5	Enables digest of user key segment containing UPK2
10	UKDIGEST6	Enables digest of user key segment containing UEK2

Table 37 • **OPTIONS[31:0]** ***(continued)***

OPTIONS	Name	Description
11	UPERM	Enables digest of permanent lock security segments
12	SYS	Enables digest of factory lock segment, factory key segment in pNVM, and system controller ROM.
13	UKDIGEST7	UKDIGEST7 in User Key segment (HWM)
14	ENVM	ENVMDIGEST
15	UKDIGEST8	UKDIGEST8 for MSS Boot Information
16	UKDIGEST9	SNVM_RW_ACCESS_MAP Digest
17	UKDIGEST10	Secure Boot Image Certificate (SBIC) revocation digest
[31:18]	Reserved	Reserved

If CHECK FABRIC is '1', the FPGA fabric is placed in suspend state and I/Os behave in same way as programming mode. Upon completion of the fabric digest, the suspend state is automatically exited. LSRAMs do not retain the user data after performing digest check on FPGA fabric. The status of the fabric digest check must be monitored by MSS. After checking the status of the fabric digest check, the MSS needs to issue a design reset or device reset depending on the design requirements. Use RESET_DEVICE tamper response signal for device reset.

If CHECK FABRIC is '0', the fabric continues to operate as normal during the requested digest calculations.

If a digest mismatch occurs, DIGESTERR indicates the selected digests are in error as listed in Table 38, page 18. A failure of any digest results in the DIGEST tamper flag being triggered. The DIGESTERR indicates zero when it is successful.

Table 38 • **DIGESTERR[31:0]**

DIGESTERR Bit Field	Name	Description
0	FABRICERR	Fabric digest error (0 if CHECK FABRIC is '0')
1	CCERR	fabric configuration digest error
2	SNVMERR	sNVM (ROM pages) digest error (0 if CHECKSNVM is '0')
3	ULERR	User security segment digest error
4	UK0ERR	Digest error in user security segment containing SRAM-PUF data
5	UK0ERR	Digest error in user security segment containing UEK (User EC key)
6	UK2ERR	Digest error in user security segment containing UPK1
7	UK3ERR	Digest error in user security segment containing UEK1
8	UK4ERR	Digest error in user security segment containing DPK
9	UK5ERR	Digest error in user security segment containing UPK2
10	UK6ERR	Digest error in user security segment containing UEK2
11	UPERR	Digest error in permanent security lock segments
12	SYSERR	Digest error in factory key segment, factory lock segment, or system controller ROM.
13	UK7ERR	UKDIGEST7 in User Key segment (HWM)
14	ENVMERR	ENVMDIGEST
15	UK8ERR	UKDIGEST8 for MSS Boot Info

Table 38 • **DIGESTERR[31:0]**

DIGESTERR Bit Field	Name	Description
16	UK9ERR	SNVM_RW_ACCESS_MAP Digest
17	UK10ERR	SBIC revocation digest
[31:18]	Reserved	Reserved

2.4.2 In-Application Programming (IAP) Service

IAP reprograms the device with a specific programming image. In IAP, regardless of the image version, the device chooses the programming image based on either the image index or the SPI image address. The MSS specifies the programming image and initiates reprogramming of the device using the IAP system service.

The user application initiates an IAP system service request using the core system services IP. The system service specifies whether the image is used for verification or programming. The system controller automatically reads the bitstream from the SPI flash to verify or program the device contents.

Verify Operation

The verify operation compares the specified programming image contents with the device contents. The following table lists the fields in an IAP system service request using the image index.

Table 39 • **IAP Verify Request by Image Index**

System Service Descriptor Bit Field	Value	Description
15		Reserved.
14:7	SPI_IDX[7:0]	Identifies the image index in the SPI directory for IAP operation.
6:0	44H	IAP verify operation.

An SPI flash memory address can be specified instead of the image index within the SPI directory, as shown in the following table.

Table 40 • **IAP Verify Request by Image Address**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 43, page 20.
6:0	45H	IAP verify operation.

If the IAP verification is successful, the status code 0 is generated. If the IAP verification fails, an 8-bit error code is generated.

Note: Digest Check Service is recommended to verify the integrity of the device contents instead of IAP verify operation. For more information, see *DS0141: PolarFire SoC FPGA Datasheet (to be released).*

Program Operation

The program operation updates the device contents using a specified programming image. The IAP program operation does not authenticate the image before executing the program. The image can be authenticated using the IAP image authentication system service.

The user application cannot obtain the status code in the following scenarios:

- If IAP is successful, the device is automatically restarted to initialize the new design.
- If IAP fails, the IAP recovery procedure attempts to program the device with image 0.

Note: IAP recovery considers image 0 when the pointer to image 1 in the SPI directory is null.

The following table lists the fields in an IAP system service request using the image index.

Table 41 • **IAP Program Request by Image Index**

System Service Descriptor Bit Field	Value	Description
15		Reserved.
14:7	SPI_IDX[7:0]	Identifies the image index in the SPI directory for IAP operation.
6:0	42H	IAP program operation.

An SPI flash memory address can be specified instead of the image index within the SPI directory, as specified in the following table.

Table 42 • **IAP Request by Image Address**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	For the mailbox format, see Table 43, page 20.
6:0	43H	IAP program operation.

The following table describes the mailbox format.

Table 43 • **IAP Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	4	SPIADDR	Input	Programming image address in SPI flash memory. If the attached SPI flash device does not support 32-bit addresses, SPIADDR[31:24] is ignored.

2.4.3 Auto Update Service

In this service, the newest image of the first two images in the SPI directory is chosen to be programmed.

Table 44 • **Digest Check Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	Reserved	Reserved
6:0	46H	Auto Update service command

The user application cannot obtain the status code in the following scenarios:

- If the auto update program is successful, the device is automatically restarted to initialize the new version of the design.
- If the auto update program fails, the auto update recovery procedure attempts to program the device with the valid image again.
- If the device remains blank at the end of auto update, there is no indication through I/O and user intervention is required.

2.5 MSS Services

For information about the return status of the MSS Services, see Table 75, page 30.

2.5.1 SPI Copy Service

Allows data to be copied from the SPI flash to MSS memory. The SPI SCK frequency is specified by a user-defined option allowing for a maximum SCK frequency of 80 MHz.

A SPI flash memory address can be specified instead of the image index within the SPI directory, as specified in the following table.

Table 45 • **SPI Copy Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	For the mailbox format, see Table 46, page 21.
6:0	50H	SPI Copy Service command

The following table lists the SPI Copy Service mailbox format.

Table 46 • **SPI Copy Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	8	DSTADDR	Input	MSS Destination address
8	4	SRCADDR	Input	SPI flash address
12	4	NBYTES	Input	Number of bytes to transfer
16	1	OPTIONS	Input	OPTIONS (See Table 47, page 21)

Table 47 • **OPTIONS[7:0]**

OPTIONS	Name	Description
1:0	CLKDIV	SPI clock divider configuration. Following are the Clock Dividers and their Frequency 0: 80MHz, 1: 40MHz, 2: 20MHz, 3: 13.33MHz
7:2	RESERVED	Reserved

2.5.2 Probe Read Debug Service

Reads the content of a probe module (59 x 18b words). The probe read debug service cannot be used when the fabric is powered down.

Table 48 • **Probe Read Debug Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 49, page 22.
6:0	70H	Probe read debug service command

The following table lists the Probe Read Debug service mailbox format.

Table 49 • **Probe Read Debug Service Mailbox Format**

Offset (Byte)	Offset (Bit)	Length (Bits)	Parameter	Direction	Description
0	0	6	IPSEGADDR[5:0]	Input	Specifies a probe segment address.
0	6	5	IPROWADDR[4:0]	Input	Specifies a probe row address.
4+4*i	0	18	PRDATAi[17:0]	Output	The read data for probe word i (0 to 58) within the probe module.

2.5.3 Probe Write Debug Service

Writes up to 18 bits of data to the selected probe address.

Table 50 • **Probe Write Debug Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 51, page 22.
6:0	71H	Probe Write Debug service command

The following table lists the Probe Write Service mailbox format.

Table 51 • **Probe Write Service Mailbox Format**

Offset (Byte)	Offset (Bit)	Length (Bits)	Parameter	Direction	Description
0	0	6	PRBADDR[5:0]	Input	Specifies one of 59 words within a probe module.
2	0	6	IPSEGADDR[5:0]	Input	Specifies the probe segment address.
2	6	5	IPROWADDR[4:0]	Input	Specifies the probe row address.
4		18	PWMASK[17:0]	Input	Specifies 18 bits of PWDATA is written to selected probe module.
8		18	PWDATA[17:0]	Input	Specifies the value to be written on selected probe registers. Example: If PWMASK[i] is ‘1’, probe register i is written to the value specified by PWDATA[i].

2.5.4 Live Probe Debug Service

Configures channel A or B of the live probe system. A live probe is enabled by writing a local address register within one of the probe segment modules. Each probe segment module generates its own local channel A live probe outputs. The live probe outputs are combined to generate a chip-level live probe channel A signal.

The local live probe may be used internally within the user design or the global channel output may be sent to an IO. To support these options, an option is provided which clears all local channel configurations before configuring a new one.

When configuring channel A, channel B is not affected and vice-versa.

Table 52 • **Live Probe Debug Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 53, page 23.
6:0	72H	Live Probe Channel A service command
	73H	Live Probe Channel B service command

The following table lists the Live Probe Service mailbox format.

Table 53 • **Live Probe Service Mailbox Format**

Offset (Byte)	Offset (Bit)	Length (Bits)	Parameter	Direction	Description
0	0	5	XADDR[4:0]	Input	Specifies the X co-ordinate within target probe module.
0	5	6	YADDR[5:0]	Input	Specifies the Y co-ordinate within target probe module.
2	0	6	IPSEGADDR[5:0]	Input	Specifies the probe segment address and the target address of probe module.
2	6	5	IPROWADDR[4:0]	Input	Specifies the probe row address and the target address of probe module.
4	0	1	CLEAR	Input	Clears the configurations of local channels A or B. If CLEAR is ‘1’, all local channel x (the applicable channel A or B) configurations are cleared before applying the new configuration.
5	0	1	IOEN	Input	Activates the probe output pad. If IOEN is ‘1’, the corresponding live probe output pad is activated. Note that setting IOEN to ‘0’ does not disable the internal live probe configuration.

2.5.5 MEM Select Debug Service

Specifies a target fabric memory for the MEM read and MEM write services to access. A handshake mechanism is used to request access to the target memory. The memory lock can be acquired immediately allowing multiple read/write operations to perform as one logical transaction or the lock can be acquired and released by individual read/write operations.

Table 54 • **MEM Select Debug Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 55, page 24.
6:0	74H	MEM select debug service command

The following table lists the MEM Select Service mailbox format.

Table 55 • **MEM Select Service Mailbox Format**

Offset (Byte)	Offset (Bit)	Length (Bits)	Parameter	Direction	Description
0	0	3	IPBLKADDR[2:0]	Input	Specifies the block address of fabric memory.
0	3	6	IPSEGADDR[5:0]	Input	Specifies the segment address.
0	9	5	IPROWADDR[4:0]	Input	Specifies the row address of fabric memory to be accessed by MEM read and MEM write services.
2	0	3	MEMTYPE	Input	Specifies the type of fabric memory to be used for MEM read and write services.
3	0	2	MEMLOCKMODE	Input	Specifies the memory lock states for supported MEMLOCKMODE values.
4	0	13	TIMEOUT[12:0]	Input	When a lock is requested, the user design may not complete the requested handshake. To prevent the firmware from waiting indefinitely, the user must specify a timeout after which the handshake is aborted.

2.5.6 MEM Read Debug Service

An interface to read data from the memory peripheral that is specified. A handshake mechanism is used to request access to the target memory. The memory lock can be acquired immediately allowing multiple read/write operations to be performed as one logical transaction.

Table 56 • **MEM Read Debug Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 55, page 24.
6:0	75H	MEM read debug service command

The following table lists the MEM Read Service mailbox format.

Table 57 • **MEM Read Service Mailbox Format**

Offset (Byte)	Offset (Bit)	Length (Bits)	Parameter	Direction	Description
0	0	15	MEMADDR[14:0]	Input	Sets the target address within the selected memory peripheral for subsequent MEM write and MEM read instructions.
2	0	15	NWORDS	Input	Depends on memory type. The maximum limit is the size of memory.
4	0	64	MSSADDR	Input	Specifies the MSS RAM area where to copy the MEM Read data to. Note that all accesses are done with MSS User privileges.

2.5.7 MEM Write Debug Service

An interface to write data to the memory peripheral that is specified. A handshake mechanism is used to request access to the target memory. The memory lock may be acquired immediately allowing multiple read/write operations to be performed as one logical transaction.

Table 58 • **MEM Write Debug Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 59, page 25.
6:0	76H	MEM write debug service command

The following table lists the MEM Write Service mailbox format.

Table 59 • **MEM Write Service Mailbox Format**

Offset (Byte)	Offset (Bit)	Length (Bits)	Parameter	Direction	Description
0	0	15	MEMADDR[14:0]	Input	Sets the target address within the selected memory peripheral for subsequent MEM write and MEM read instructions.
2	0	15	NWORDS	Input	Depends on memory type. The maximum limit is the size of memory.
4	0	64	MSSADDR	Input	Specifies the MSS RAM area where to copy the MEM Write data from. Note that all accesses are done with MSS User privileges.

2.5.8 APB Read Debug Service

Reads a specified number of bytes from the fabric APB bus to the specified MSS RAM area. The operation makes the required number of read transactions using the selected transaction size, APBDWSIZE.

The addressed fabric peripheral generates an error or fail to respond within a user-defined window, in that case any subsequent transfers are aborted and corresponding error flags are returned.

Table 60 • **APB Read Debug Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 61, page 25.
6:0	77H	APB Read Debug service command

The following table lists the APB Read Debug Service mailbox format.

Table 61 • **APB Read Debug Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	4	APBADDR[28:0]	Input	Specifies the target address and transfer size for the APB write and APB read operations.

Table 61 • **APB Read Debug Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
4	1	APBDWSIZE[1:0]	Input	Specifies the data transfer size to be used by the APB write and APB read operations.
8	2	MAXBYTES[11:0]	Input	Calculates specified number of bytes from the fabric APB bus to the Shared Buffer. Number of bytes = MAXBYTES + 1
16	8	MSSADDR	Input	Specifies the MSS RAM area where to copy MAXBYTES+1 of the APB Read data to. Note that all accesses are done with MSS User privileges.

2.5.9 APB Write Debug Service

Writes bytes of data to the current fabric APB address as specified by APBADDR. The addressed fabric peripheral generates an error or fail to respond within a user-defined window, in that case the corresponding error flags are returned.

Table 62 • **APB Write Debug Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 63, page 26.
6:0	78H	APB Write Debug service command

The following table lists the APB Write Debug Service mailbox format.

Table 63 • **APB Write Debug Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	4	APBADDR[28:0]	Input	Specifies the target address and transfer size for the APB write and APB read operations.
4	1	APBDWSIZE[1:0]	Input	Specifies the data transfer size to be used by the APB write and APB read operations.
8	2	MAXBYTES[11:0]	Input	Calculates specified number of bytes from the fabric APB bus to the Shared Buffer. Number of bytes = MAXBYTES + 1
16	8	MSSADDR	Input	Specifies the MSS RAM area where to copy from MAXBYTES+1 of data to the APB. Note that all accesses are done with MSS User privileges.

2.5.10 Debug Snapshot Service

Generates a snapshot of the volatile fabric content. The volatile fabric data is read from each LSRAM, µRAM and probe module and copied to the fabric APB debug port.

Table 64 • **Debug Snapshot Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 65, page 27.
6:0	79H	Debug Snapshot service command.

The following table lists the Debug Snapshot Service mailbox format.

Table 65 • **Debug Snapshot Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	4	PORTADDR[28:0]	Input	Sets the address of the APB port. Bits 1:0 are ignored.
4	1	APBFASTWR	Input	Specifies the usage of fast APB protocol. If apb_fast_write is '1' during write transfers, the fast APB protocol is used and the address range is limited to PORTADDR[15:2] and PORTADDR[28:16] is ignored.

2.5.11 Generate OTP Service

Sets up the device to receive a one-time passcode. A 128-bit nonce, N_{FPGA}, is generated and stored in volatile memory for later use in the rest of the protocol. A 128-bit user nonce, N_{USER}, is supplied by the user.

This service only unlocks the software debug lock SWL_DEBUG.

Table 66 • **Generate OTP Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 67, page 27.
6:0	7AH	Generate OTP service command

The following table lists the Generate OTP Service mailbox format.

Table 67 • **Generate OTP Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	1	KEYMODE	Input	Specifies the key mode used to transport the encrypted passcode. The KEYMODE parameter is not checked for validity until the MATCH OTP service is executed.
4	16	N_{USER}	Input	Specifies the user nonce, and is supplied by the user.
20	16	N_{FPGA}	Output	The 128-bit nonce, N_{FPGA}, is generated and stored in volatile memory for later use in the rest of the protocol.

2.5.12 Match OTP Service

This service is the second part of the one-time passcode protocol. Before using this service, the GENERATE OTP service must first be used to obtain a nonce, N_{FPGA}, from the device.

Table 68 • **Match OTP Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 69, page 28.

Table 68 • **Match OTP Service Request *(continued)***

System Service Descriptor Bit Field	Value	Description
6:0	7BH	Match OTP service command

The following table lists the Match OTP Service the mailbox format.

Table 69 • **Match OTP Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	16	UID	Input	User ID
16	32	V	Input Output	Input: Validator Output: Set to 0 during system service execution.
48	32	OTP	Input Output	Input: One Time Passcode Output: Set to 0 during system service execution.

2.5.13 Unlock Debug Passcode Service

Attempts to match the user debug pass code using the key loaded into the mailbox. If the match is successful, the software debug lock SWL_DEBUG is temporarily inactive.

Table 70 • **Unlock Debug Passcode Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. For the format, see Table 71, page 28.
6:0	7CH	Unlock Debug Passcode service command

The following table lists the Unlock Debug Passcode service mailbox format.

Table 71 • **Unlock Debug Passcode Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	32	DEBUG PASSCODE	Input Output	Input: Value of debug passcode Output: This is set to 0 during system service execution.

2.5.14 One Way Passcode Service

Provides a mechanism for overriding the software debug lock SWL_DEBUG without requiring any interaction with an external intelligence. This service is similar to One-Time Passcode but has selective unlocking capability and does not require a FPGA nonce.

Table 72 • **One Way Passcode Service Request**

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address.
6:0	7DH	One-way Passcode service command

The following table lists the One Way Passcode service mailbox format.

Table 73 • **One Way Passcode Service Mailbox Format**

Offset	Length (bytes)	Parameter	Direction	Description
0	16	MSG ID	Input	Message identification
16	32	V	Input	Input Validator
48	2	BSVER	Input	Not used. Value Ignored[1]
50	1	CTYPE	Input	
51	1	KEYMODE	Input	
52	16	UID	Input	Not used. Value Ignored[1]
68	16	DSN	Input	
84	256	USERDATA	Input	Not used. Value Ignored[1]
340	1	OPTIONS	Input	
341	1	RESERVED	Input	Reserved
342	2	SWVER	Input	Not used. Value Ignored[1]
344	8	RESERVED	Input	Reserved
352	32	HASH	Input	Hash of next 96 bytes
384	32	Plaintext Passcode	Input	Encrypted via bitstream mechanism
416	16	HWM	Input	Encrypted via bitstream mechanism
432	8	LOCKMASK	Input	Encrypted via bitstream mechanism Not used. Value Ignored[1] SWL_DEBUG always chosen.
440	1	PCYTPE	Input	Encrypted via bitstream mechanism Not used. Value Ignored[1] PCTYPE=1 DPK always chosen.
441	7	RESERVED	Input	Reserved
448	32	TNEXT	Input	Not used. Value Ignored[1]

1. Value of 0 must be used for value ignored.

2.5.15 Terminate Debug Service

This service is called to end the debug session. Its purpose is to re-lock all the software based debug locks (SWL_DEBUG) if needed and to release any memories previously locked using the MEM Select Debug Service.

Table 74 • **Terminate Debug Service Request**

System Service Descriptor Bit Field	Value	Description
15:7		Reserved
6:0	7EH	System service command

2.6 System Service Return Status Codes

The following table lists all the system services with their command values and return status.

Table 75 • **PolarFire SoC FPGA System Services**

Category	System Service Name	Command Value in Hexadecimal	Response Status
Device and Design Information Services	Serial Number Service	00	0: Success
	USERCODE Service	01	
	Design Information Service	02	
	Design Certificate Service	03	0: Success 1: Device mismatch 2: Signature invalid 3: System error
	Read Digests Service	04	0: Success
	Query Security Service	05	
	Read Debug Information Service	06	
	Read eNVM Parameters Service	07	0: Success 1: eNVM page mismatch error
Design Programming Services	Bitstream Authentication Service	23	0: Success 1: ERRORCODE (see Table 18, page 10)
	IAP Image Authentication Service	22	
Data Security Services	Digital Signature Service	19, 1A	0: Success 1: FEK failure 2: DRBG error 3: ECDSA error
	Secure NVM Write Service	10, 11, 12	0: Success 1: Invalid SNVMADDR 2: Write failure 3: System error 4: Write not permitted 5: Access failure
	Secure NVM Read Service	18	0: Success 1: Invalid SNVMADDR 2: Authentication failure 3: System error 5: Access failure
	PUF Emulation Service	20	0: Success 1: Internal error
	Nonce Service	21	0: Success 1: Error fetching PUK 2: Error generating seed

Table 75 • **PolarFire SoC FPGA System Services**

Category	System Service Name	Command Value in Hexadecimal	Response Status
Fabric Services	Digest Check Service	47	0: Success 1: returned if any of DIGESTERR bits are set See Table 38, page 18 for digest error codes.
	In-application Programming Service	42, 43, 44, 45	0: Success 1: ERRORCODE (see Table 18, page 10)
	Auto Update Service	46	See Table 18, page 10 for error codes.
MSS Services	SPI Copy Service	50	0: Success 1: Device not configured for master mode 2: AXI Error
	Probe Read Debug Service	70	0: Success 1: SECERR (Blocked by Device security)
	Probe Write Debug Service	71	
	Live Probe Debug Service	72, 73	
	MEM Select Debug Service	74	0: Success 1: SECERR (Blocked by Device security) 2: TIMEOUTERR 3: LOCKERR (Target memory fail to lock)
	MEM Read Debug Service	75	
	MEM Write Debug Service	76	
	APB Read Debug Service	77	0: Success 1: SECERR (Blocked by Device security) 2: SLVERR (Bus transaction error) 3: TIMEOUT
	APB Write Debug Service	78	
	Debug Snapshot Service	79	0: Success 1: SECERR (Blocked by Device security) 2: BUSERR (the fabric power is off, the APB slave flags an error or the APB slave is slow to assert READY signal)
	Generate OTP Service	7AH	0: Success 1: SECERR (Blocked by Device security) 2: PROTOCOLERR
	Match OTP Service	7BH	0: Success 1: PROTOCOLERR 2: MISMATCHERR
	Unlock Debug Passcode Service	7CH	0: Success 1: SECERR (Blocked by Device security) 2: PASSCODE_ERR (tamper event PASSCODE_FAIL generates and all unlocked passcodes are re-locked)
	One Way Passcode Service	7DH	0: Success 1: OWPERR (tamper event PASSCODE_FAIL generates and all unlocked passcodes are re-locked)
	Terminate Debug Service	7EH	0: Success

2.7 How to Use System Services

To access the system services implemented by the PolarFire SoC System Controller, Microsemi will provide system services firmware drivers along with example project. The system services drivers will be available in future for user applications. The system services APIs must be called in the user application code to access system services. Each system service API returns a service response to know the status of the service.