Libero® SoC v2021.3

PolarFire® System Services SgCore User Guide

Introduction

The `PF_SYSTEM_SERVICES` SgCore provides access to the System Services supported by the PolarFire® and PolarFire SoC devices. System services are system controller actions initiated by asynchronous events from the user design via the system controller's system service and mailbox interface. They are not accessible when the system controller suspend mode is enabled. Microchip provides PolarFire System Services SgCore to execute the system services on the PolarFire and PolarFire SoC devices. The PolarFire System Services SgCore provides an Arm® AMBA® APB interface for controlling the registers implemented in it. Microchip provides PolarFire System Services SgCore firmware driver with a set of functions for controlling the PolarFire System Services SgCore from a general purpose processor.

For more information on PolarFire system services, see PolarFire FPGA and PolarFire SoC FPGA System Services User Guide.

Note: The APB3 protocol standard uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Initiator” and “Target”, respectively.

Features

- Supports Device and Data System Services
- Supports Fabric Services
- Supports AMBA interface to perform System services
- Supports APB target interface for access to System Service related registers

The following feature is not supported:

- Queuing of System Services is not supported

Supported Families

- PolarFire
- PolarFire SoC

Device Usage and Performance

Usage and performance data is listed in the following table for the PolarFire device family. The data listed in this table is indicative only. The overall device usage and performance of the core is system dependent.

Table 1. Device Usage and Performance

FAMILY	DCSERVICE	DIGESTCHECK	DIGSIGSERVICE	DVSERVICE	FFSERVICE	IAPAUTOUPD	IAPSERVICE	NONCESERVICE	PUFEMSERVICE	QUERYSECSERVICE	RDDEBUGINFO	RDDIGEST	SECNVMRD	SECNVMWR	SNSERVICE	UCSERVICE	Logic Elements				Frequency (MHz)
																	Combinational	Sequential	Total	Utilization (%)	
PolarFire	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	451	384	835	0.18	182
PolarFire	1	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	436	390	826	0.18	180
PolarFire	0	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	469	395	864	0.18	193
PolarFire	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	459	382	841	0.18	183

Note: The data in this table is achieved using typical synthesis and layout settings. Frequency (in MHz) was set to 100 and speed grade was -1. FPGA resources and performance data for the PolarFire SoC family is similar to PolarFire family.

Table of Contents

1. Overview

System Services are System Controller actions initiated by the user design through the System Controller's System Service Interface (SSI). System Services are invoked by writing a 16-bit system service descriptor consisting of the command and mailbox offset address.

The lower 7-bits of the descriptor specify the service to be performed and the upper nine bits are used to provide additional information such as the address of a location in the 2 KB mailbox RAM. The mailbox address specifies the location of a service-specific data structure, which is used for any additional input parameters and any outputs from the service. Mailbox addresses are specified using a word offset (0-511).

The following table lists the system service request descriptor bits.

Table 1-1. 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.

Each service has some service-specific data structure, which is used for any additional input parameters and any outputs from the service. The mailbox interface is used for passing data related to the system service between the fabric and System Controller (in both directions).

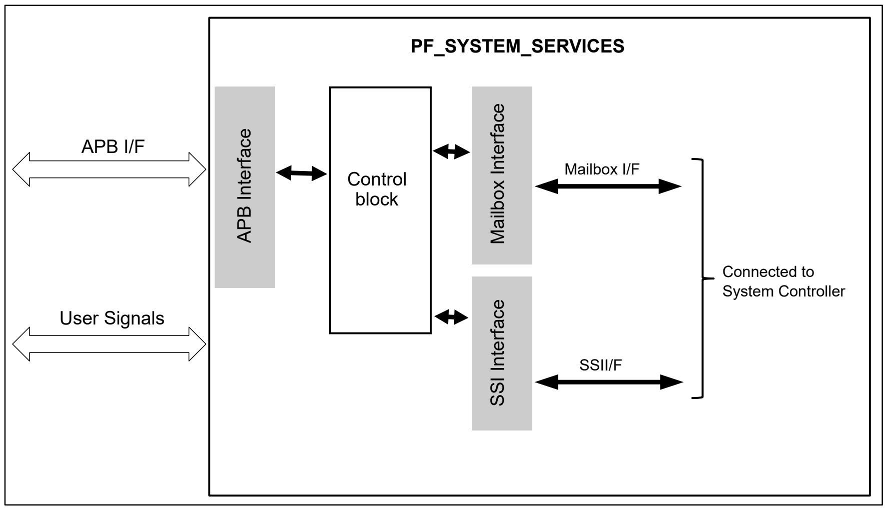
Upon completion of a service, a status code is written to the status register and the interface returns to a state ready for the next request. The `USR_BUSY` signal is used to indicate that the service is in progress.

Note: If a new service request is initiated when the `USR_BUSY` is in progress, the new request is ignored until the current service completes.

1.1 Core Description

The CoreSysServices Soft IP consists of sub-modules as shown in the following block diagram.

Figure 1-1. PF_SYSTEM_SERVICES Block Diagram



1.1.1 SSI Interface (System Services Interface)

The SSI interface is used to support the command and status protocol for the system services between the `PF_SYSTEM_SERVICES` IP and the System Controller.

Note: The SSI interface is not exposed, instead it is connected via the corresponding Libero signals using `SYS_SERVICES` macro. The SSI interface is asynchronous to the `PF_SYSTEM_SERVICES` IP. All the asynchronous signals are synchronized with the internal IP clock in the `PF_SYSTEM_SERVICES` IP block.

1.1.2 MailBox Interface

The Mailbox interface is used to provide additional information, typically the address of a location for the specified service or data for the requested service returns the response data related to the requested service. It provides a mailbox between the `PF_SYSTEM_SERVICES` IP and the System Controller.

Note: The Mailbox interface is not exposed, instead it is connected via the corresponding Libero signals using `SYS_SERVICES` macro. The mailbox interface is asynchronous to the `PF_SYSTEM_SERVICES` IP. All the asynchronous signals are synchronized with the internal IP clock in the `PF_SYSTEM_SERVICES` IP block.

1.1.3 APB Target Interface Logic

The core provides APB3 target interface to connect to the APB3 initiator or APB3 interconnect bus interface. The APB3 target interface of the core complies with AMBA APB3 protocol specifications.

The APB interface is also used to provide access to the 32-bit registers in the System Services IP. For more information about APB registers, see 5. Register Map and Descriptions.

Note: The System Service registers are required to be configured for each of the service.

1.1.4 Control Block

The control block contains control state machines to perform system service requests and response to or from the System Controller interface and the registers. This block consists of three FSMs:

- Main state machine
- SSI state machine
- Mailbox state machine

The main state machine controls the other two state machines depending on the configured service. For more information, see Figure 1-2 and Figure 1-3.

The System Service request is initiated by setting the bit[0] of the `SYS_SERV_REQ` register. It invokes the main control FSM.

The control block determines whether the configured service requires additional service specific data structure to be sent on the System Controller's mailbox interface. Accordingly, it is determined whether to initiate write transactions on the mailbox interface. The mailbox address is specified using the `MBX_WADRDESC` register. The `MBX_WCNT` register specifies the number of data words to be written, which is specific to the System service.

The 16-bit System Service descriptor containing the command and mailbox address is initiated by the SSI state machine on the System Controller command response interface. The SSI System descriptor is configured using `SYS_SERV_CMD` (for 7-bit command ID) and the upper 9-bit mailbox address is specified using the `MBX_WADRDESC` register.

When the given service returns response data output on the System Controller's mailbox interface, the control block System Controller's stores the response received to the `MBX_RDATA` register. The core then asserts the `USR_RDVLD` to the user design indicating that read data is available for read. When the APB read to the `MBX_RDATA` register is done, the `USR_RDVLD` is de-asserted. It is important to note that the APB initiator must poll the `USR_RDVLD` signal before reading the `MBX_RDATA` register. The core performs the read to the System Controller's mailbox interface until the required number of reads are complete. The number of word reads are determined by the value configured in the `MBX_RCNT` register. The `MBX_ECCSTATUS` is used to read the ECC status of each mailbox read.

The response status of the service can be read from the `SYS_SERV_STATUS` register.

Figure 1-2 and Figure 1-3 show a basic flow diagram of the design for `PF_SYSTEM_SERVICES`.

Figure 1-2. PF_SYSTEM_SERVICES Flow Diagram (a)

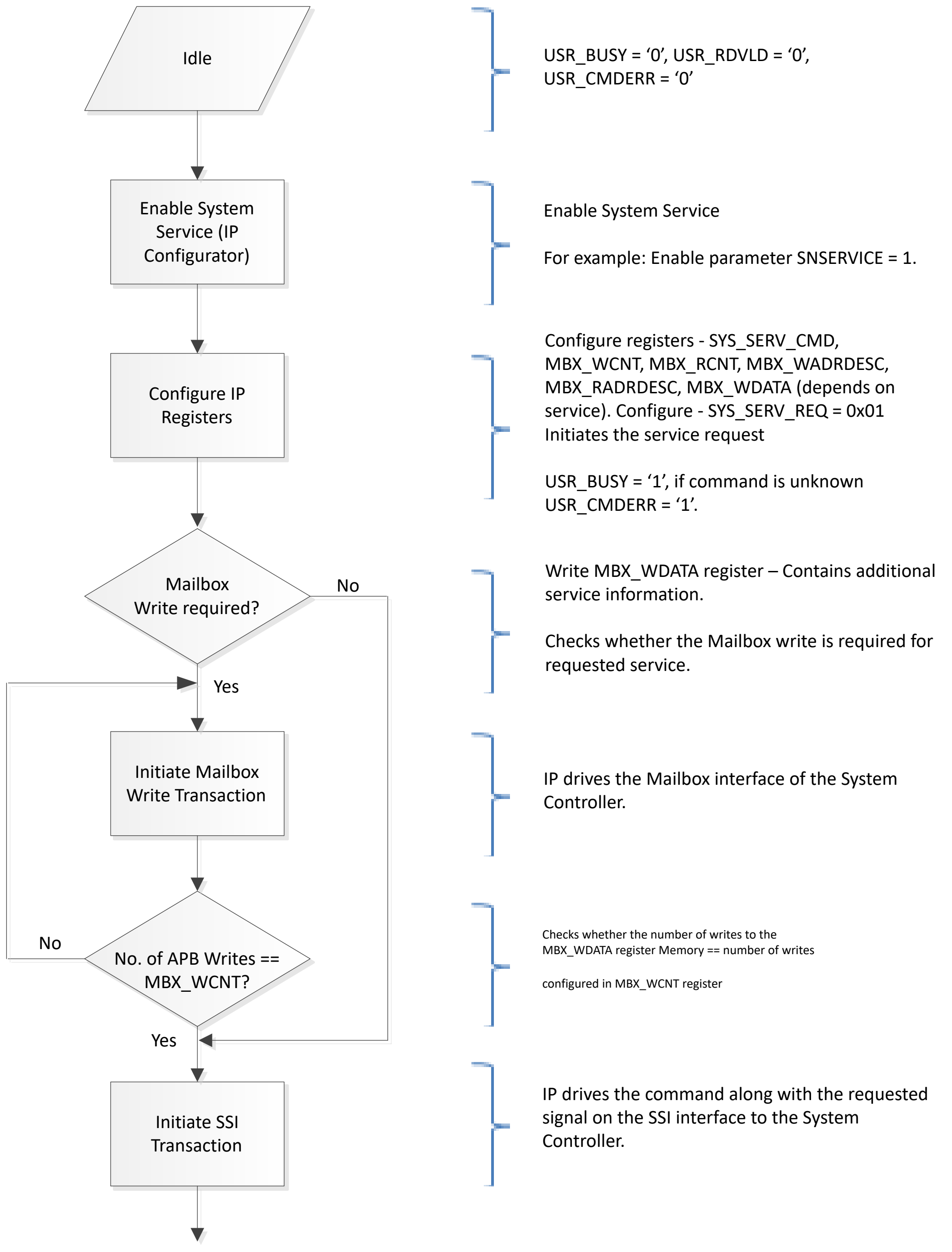
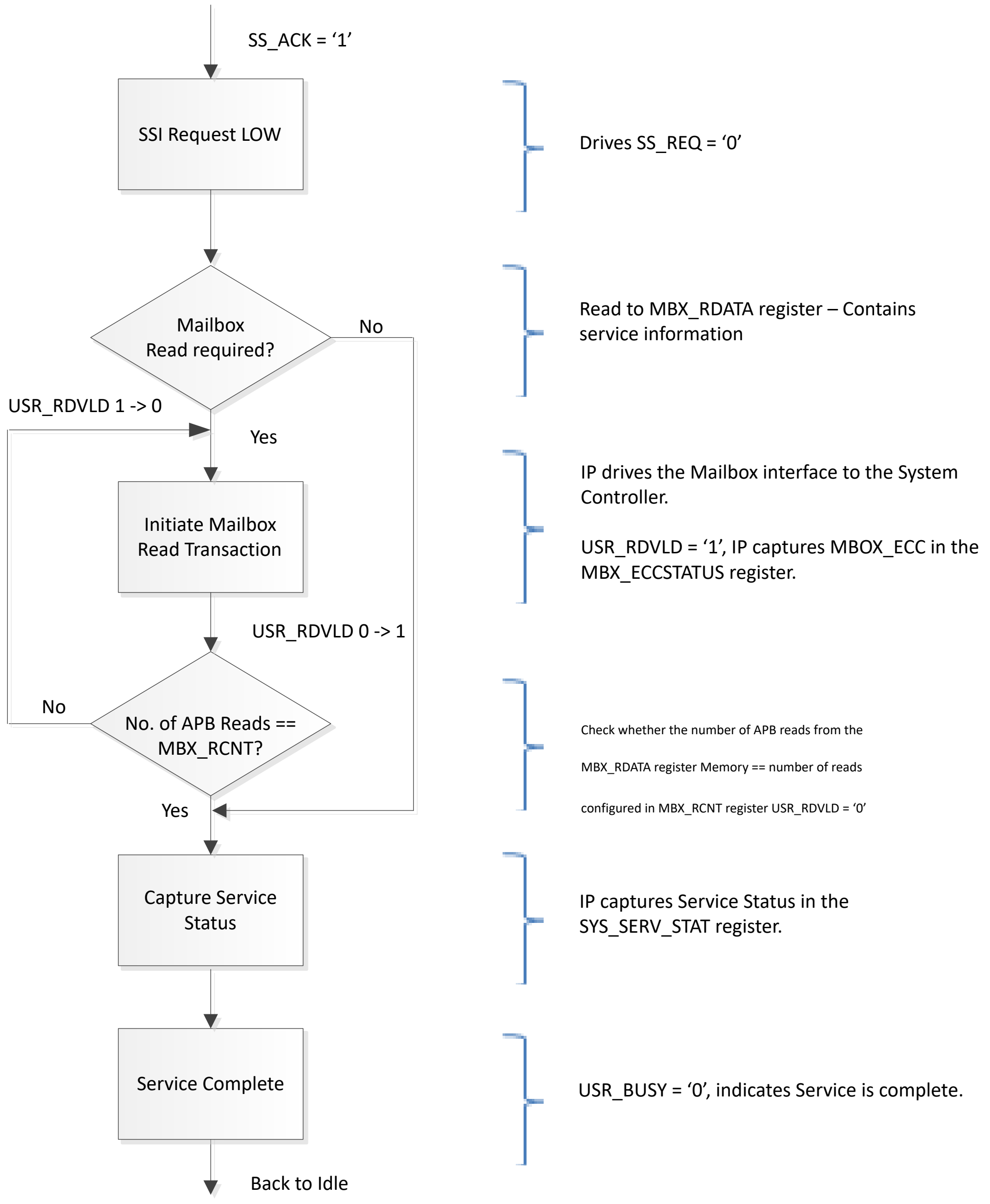

Idle
Enable System Service (IP Configurator)
Configure IP Registers
Mailbox Write required?
No
Yes
Initiate Mailbox Write Transaction
No. of APB Writes == MBX_WCNT?
No
Yes
Initiate SSI Transaction
USR_BUSY = '0', USR_RDVLD = '0', USR_CMDERR = '0'
Enable System Service
For example: Enable parameter SNSERVICE = 1.
Configure registers - SYS_SERV_CMD, MBX_WCNT, MBX_RCNT, MBX_WADRDESC, MBX_RADRDESC, MBX_WDATA (depends on service). Configure - SYS_SERV_REQ = 0x01 Initiates the service request
USR_BUSY = '1', if command is unknown USR_CMDERR = '1'.
Write MBX_WDATA register – Contains additional service information.
Checks whether the Mailbox write is required for requested service.
IP drives the Mailbox interface of the System Controller.
Checks whether the number of writes to the MBX_WDATA register Memory == number of writes
configured in MBX_WCNT register
IP drives the command along with the requested signal on the SSI interface to the System Controller.

Figure 1-3. PF_SYSTEM_SERVICES Flow Diagram (b)


SS_ACK = '1'
SSI Request LOW
Drives SS_REQ = '0'
Mailbox Read required?
No
Read to MBX_RDATA register – Contains service information
USR_RDVLD 1 -> 0
Yes
Initiate Mailbox Read Transaction
IP drives the Mailbox interface to the System Controller.
USR_RDVLD = '1', IP captures MBOX_ECC in the MBX_ECCSTATUS register.
USR_RDVLD 0 -> 1
No
No. of APB Reads == MBX_RCNT?
Check whether the number of APB reads from the MBX_RDATA register Memory == number of reads configured in MBX_RCNT register USR_RDVLD = '0'
Yes
Capture Service Status
IP captures Service Status in the SYS_SERV_STAT register.
Service Complete
USR_BUSY = '0', indicates Service is complete.
Back to Idle

1.1.5 User Interface

The user interface provides the busy (`USR_BUSY`) and other service related status to the user design. It indicates whether the current service is a valid system service on the `USR_CMD_ERROR` output. `USR_RDVLD` is used to read the contents of the read data from the mailbox read data register.

1.2 Device and Design Information Services

These services return information about the device and current user design. The requested information is copied to a location in the mailbox RAM whose address is included in the service descriptor. The size of the returned data is service dependent. These services are available on all devices.

For more information about the return status of device and design information services, see 8. System Service Return Status Codes.

1.2.1 Serial Number Service

Serial Number Service fetches the 128-bit Device Serial Number (DSN). The DSN is a 128-bit quantity unique to every device, set during manufacturing. It comprises of two parts—the Factory Serial Number (FSN) and the Serial Number Modifier (SNM). The first part of the device serial number is the 64-bit FSN that uniquely identifies a device. The DSN is zeroized if the unrecoverable zeroization action is performed on the device.

Table 1-2. Serial Number Service Request

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. See Table 1-3
6:0	00H	Serial number service command

The following table lists the Serial Number Service mailbox format.

Table 1-3. Serial Number Service Mailbox Format

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

1.2.2 USERCODE Service

USERCODE service fetches the 32-bit USERCODE/Silicon signature.

Table 1-4. USERCODE Service Request

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. See Table 1-5.
6:0	01H	USERCODE service command

The following table lists the USERCODE Service mailbox format.

Table 1-5. USERCODE Service Mailbox Format

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

1.2.3 Design Information Service

Design Information service fetches design information including 256-bit user defined design ID, 16-bit design version, and 16-bit design back-level protection value.

Table 1-6. Design Information Service Request

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. See Table 1-7.
6:0	02H	Design Information service command

The following table lists the Design Information Service mailbox format.

Table 1-7. Design Information Service Mailbox Format

Offset	Length (bytes)	Parameter	Direction	Description
0	32	DESIGNID	Output	256-bit user-defined design ID

...........continued

Offset	Length (bytes)	Parameter	Direction	Description
32	2	DESIGNVER	Output	16-bit design version (DCV)
34	2	BACKLEVEL	Output	16-bit design back-level (DCB)

1.2.4 Device Certificate Service

Device Certificate service fetches the device's Supply Chain Assurance Certificate from pNVM.
Note: The certificate data is stored as a 1024-bit entity but the actual certificate size might be smaller. Any excess bytes must be discarded.

The device validates the certificate by checking its signature using the Microchip Public Key (MCPK). In addition,

- The certificate DSN is checked against the DSN.
- The certificate public key is checked by recalculating the value using factory key and comparing against the certificate.

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

Table 1-8. Design Certificate Service Request

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. See Table 1-9.
6:0	03H	Design certificate service command.

The following table lists the Design Certificate Service mailbox format.

Table 1-9. Design Certificate Service Mailbox Format

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

1.2.5 Read Digests Service

Read Digests service returns the stored digests for the device.

Table 1-10. Read Digests Service Request

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. See Table 1-11.
6:0	04H	Read digests service command.

The following table lists the Read Digests Service mailbox format.

Table 1-11. Read Digests Service Mailbox Format

Offset	Length (bytes)	Parameter	Direction	Description
0	416	DIGESTS	Output	Digest array

Table 1-12. Returned Digests Format

Offset (byte)	Size (bytes)	Value	Description
0	32	FD	Fabric digest

...........continued

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

1.2.6 Query Security Service

Query Security service fetches nonvolatile states of user security locks.

Table 1-13. Query Security Service Request

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. See Query Security Service Mailbox Format.
6:0	05H	Query security service command

The following table lists the Query Security Service mailbox format.

Table 1-14. Query Security Service Mailbox Format

Offset	Length (bytes)	Parameter	Direction	Description
0	33	LOCKS	Output	Lock array

The following table lists the description of the returned LOCKS array.

Table 1-15. Returned LOCKS Array

Byte	Bit	Lock	Description
0	0	UL_DEBUG	Debug instructions disable
0	1	UL_SNVM_DEBUG	sNVM debug disable
0	2	UL_LIVEPROBE	Live probes disable
0	3	UL_UJTAG	User JTAG interface disable
0	4	UL_JTAG_BS	JTAG boundary scan disable
0	5	UL_TVS_MONITOR	External access to System TVS monitor disable
0	6	UL_JTAG_MONITOR	JTAG fabric monitor enable

...........continued

Byte	Bit	Lock	Description
0	7	UL_JTAG	JTAG TAP disable
1	0	UL_PLAINTEXT	Plaintext passcode unlock disable
1	1	UL_FAB_PROTECT	Fabric erase/write disable
1	2	UL_EXT_DIGEST	External digest check disable
1	3	UL_VERSION	Replay protection enable
1	4	UL_FACT_UNLOCK	Factory test disable
1	5	UL_IAP	IAP disable
1	6	UL_EXT_ZEROIZE	External zeroization disable
1	7	UL_SPI_SLAVE	SPI port disable
2	0	UL_USL	UFS UL segment protect
2	1	UL_BS_AUTHENTICATE	External bit stream authentication disable
2	2	UL_BS_PROGRAM	External bit stream program mode disable
2	3	UL_BS_VERIFY	External bit stream verify mode disable
2	4	UL_BITS_KEYMD[0]	Bitstream key mode disable
2	5	UL_BITS_KEYMD[1]	Bitstream key mode disable
2	6	UL_BITS_KEYMD[2]	Bitstream key mode disable
2	7	UL_BITS_KEYMD[3]	Bitstream key mode disable
3	0	UL_BITS_KEYMD[4]	Bitstream key mode disable
3	1	UL_BITS_KEYMD[5]	Bitstream key mode disable
3	2	UL_BITS_KEYMD[6]	Bitstream key mode disable
3	3	UL_BITS_KEYMD[7]	Bitstream key mode disable
3	4	UL_BITS_KEYMD[8]	Bitstream key mode disable
3	5	UL_BITS_KEYMD[9]	Bitstream key mode disable
3	6	UL_BITS_KEYMD[10]	Bitstream key mode disable
3	7	UL_BITS_KEYMD[11]	Bitstream key mode disable
4	0	UL_BITS_KEYMD[12]	Bitstream key mode disable
4	1	UL_BITS_KEYMD[13]	Bitstream key mode disable
4	2	UL_BITS_KEYMD[14]	Bitstream key mode disable
4	3	UL_BITS_KEYMD[15]	Bitstream key mode disable
4	4	UL_KEYMD[0]	Global key mode disable
4	5	UL_KEYMD[1]	Global key mode disable
4	6	UL_KEYMD[2]	Global key mode disable
4	7	UL_KEYMD[3]	Global key mode disable
5	0	UL_KEYMD[4]	Global key mode disable
5	1	UL_KEYMD[5]	Global key mode disable

...........continued

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

1.2.7 Read Debug Information Service

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.

Table 1-16. Read Debug Information Service Request

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. See Debug Information.
6:0	06H	Read debug service command.

The following table lists the debug information reported by the Read Debug Information service.

Table 1-17. 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 types used to program device: 1: JTAG 2: IAP 3: SPI_SLAVE
4	37	Reserved	Reserved
1	41	FRAME_ERRORCODE	An error has occurred during bit stream frame processing and the error is identified by the FRAME_ERRORCODE field. See Table 1-18.
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 1-18.
7	65	Reserved	Reserved
4	72	IAP Location	SPI address that was last run in IAP
4	76	SYSCTRL_STATUS	System Controller reset status
4	80	RESET_REASON	Reason for last device reset

The following table lists the error codes and their description.

Table 1-18. ERRORCODE

ERRORCODE	Description	Additional Notes
0	No error	—
1	Bitstream authentication failed	Invalid bit stream or wrong key used.
2	Unexpected data received	Additional data is received after end of bit stream 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 bit stream
5	Back level not satisfied	Bitstream version is older than that of the current back level value set in the device.
6	Illegal bit stream key mode	Bitstream key mode is not initialized or bit stream key mode is disabled by user security.

...........continued

ERRORCODE	Description	Additional Notes
7	DSN binding mismatch	Bitstream is rejected because DSN in the bit stream does not match with the DSN present in the device. A bit stream can be bound to device's unique DSN such that only a specific device can be programmed with that bit stream.
8	Illegal component sequence	Incorrect bit stream
9	Insufficient device capabilities	Bitstream is rejected because the capabilities specified in the bit stream do not match the target device's capabilities.
10	Incorrect DEVICEID	Bitstream is rejected because an attempt by the DEVICEID specified in the bit stream does not match the part identification field (for example, MPF300, MPF500 and so on) of the target device.
11	Unsupported bit stream protocol version (bit stream regeneration required)	Bitstream is rejected because of an attempt made by the old version of a device to decode a bit stream created in new format or by the new version of a device to decode a bit stream created in old format.
12	Verify not permitted on this bit stream	Verify programming action is disabled in the bit stream.
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 Initiater Mode	Error might occur only when the bit stream is executed through IAP mode. The System Controller SPI controller is not configured in the Initiater mode.
22	No valid images found	Error might occur when bit stream is executed through Auto Update mode. Occurs when no valid image pointers are found.
23	No valid images found	Error might occur when bit stream is executed through IAP mode via Index mode. Occurs when No valid image pointers are found.
24	Programmed design version is the same as the Auto Update image found	Error might occur when bit stream is executed through Auto Update mode.
25	Reserved	Reserved
26	Selected image was invalid and no recovery was performed due to valid design in device.	Error might occur only when bit stream is executed through Auto Update or IAP mode. Error can also occur due to BACKLEVEL protection.
27	Selected and Recovery image failed to program	Error might occur only when bit stream is executed through Auto Update or IAP mode.
127	Abort	Non-bit stream instruction is executed during bit stream loading.
128	NVMVERIFY	Fabric or security segment verification failed.
129	PROTECTED	Device security is prevented modification of nonvolatile memory.
130	NOTENA	Programming mode not enabled.
131	PNVMVERIFY	pNVM verify operation failed.

...........continued

ERRORCODE	Description	Additional Notes
132	SYSTEM	System hardware error (PUF or DRBG).
133	BADCOMPONENT	An internal error was is detected in a bit stream component payload.
134	HVPROGERR	Failure in programming subsystem.
135	HVSTATE	Error in the programming subsystem.

1.3 Design Programming Services

An IAP image contains the image descriptor, bit stream, and optional design initialization data. The design programming services are used to authenticate entire IAP image, bit stream portion, or program the device. For more information about the return status of Design Programming Services, see 8. System Service Return Status Codes.

1.3.1 Bitstream Authentication Service

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

The bit stream authentication service analyzes a bit stream image stored in SPI Flash and checks for all conditions, which might 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 bit stream is stored.

If the authentication service is called while a new bit stream 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 1-19. Bitstream Authentication Service Request

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. See Bitstream Authentication Service Mailbox Format.
6:0	23H	Bitstream authentication service command.

The following table lists the Bitstream Authentication service mailbox format.

Table 1-20. Bitstream Authentication Service Mailbox Format

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

1.3.2 IAP Image Authentication Service

IAP Image Authentication service allows you to validate an IAP image stored in SPI Flash. The service authenticates the entire IAP image containing the image descriptor, the referenced bit stream, and optional initialization data. If the image is authenticated successfully, the image is ensured 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. As, SPI_IDX

= 1 must be an empty slot, it must not be passed into the system service. The following table lists the fields contained in an IAP image authentication service request.

Table 1-21. IAP Image Authentication Service Request

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

1.4 Data Security Services

The data security services are used to authenticate the device, generate unique random number, and store the encrypted data. For more information about the return status of Data Security Services, see 8. System Service Return Status Codes.

1.4.1 Digital Signature Service

The 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 Microchip 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. As, 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 you.

Table 1-22. Digital Signature Service Request

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. See Table 1-23.
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 1-23. Digital Signature Service Mailbox Format

Offset	Length (bytes)	Parameter	Direction	Description
0	48	HASH	Input	SHA384 hash to be signed

...........continued

Offset	Length (bytes)	Parameter	Direction	Description
48	96 (Raw)	SIGNATURE	Output	ECDSA signature (r, s)
	104 (DER)			

1.4.2 Secure NVM (sNVM) Services

Secure NVM (sNVM) occupies the Sector 1 of the pNVM. Each page of the pNVM in this region constitutes one sNVM module. Three pages of the pNVM sector are reserved for administrative purposes, leaving 221 pages available for sNVM modules. sNVM modules can be marked as ROM during bit stream programming. These modules cannot be modified by the secure NVM functions, but they can be read.

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 bit stream programming. In this case, sNVM cannot be modified by the Secure NVM services, but it can be read.

Note: USK related functionality is not supported in simulation model and the data is always presented to fabric as plaintext.

1.4.2.1 Secure NVM Write Service

The Secure (sNVM) write service provides the write access to pages in the sNVM region of the pNVM. 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 Initiater Key (SMK) is the primary key used, with 256-bit allocated for authentication and 256-bit 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. 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, System Controller uses AES-256 in Synthetic Initialization Vector (SIV) mode, which supports authenticated encryption. In SIV mode, the initialization vector used for the encryption function is computed from the data, preventing IV misuse and doubles as the authentication tag. The computed 128-bit initialization vector 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. It 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 (as the page write counter advances).

USK is used as another element in the tweak. Without the same 96-bit, USK is used during the write command, the read command fails authentication (and could not possibly decrypt correctly, either). You can choose to set this key differently for each page, for groups of pages, or the same for all pages—either as a secret key for added security, or to a invalid 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 bit stream is generated, and a page's ROM status can only be changed with a new bit stream, and not at run-time.

Table 1-24. Secure NVM Write Request

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. See Table 1-25 and Table 1-26
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 1-25. 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	PT	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 1-26. 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	PT	Input	Data to write to sNVM
240	12	USK	Input	User Secret Key

Programming Example for sNVM

An example sequence to configure `PF_SYSTEM_SERVICE` registers to access sNVM is shown in the following steps. It is a basic sequence to read/write one page of sNVM. sNVM page size for non-authenticated plaintext is 252 bytes and for authenticated plaintext or ciphertext is 236 bytes.

Non-Authenticated Plaintext sNVM Write at Address 0x10

1. Write 0x10, sNVM write service command for non-authenticated plaintext in `SYS_SERV_CMD` (register offset – 0x04) register.
2. Write 0x40 in `MBX_WCNT` (register offset – 0x14) register to write total 64 words data (first word – sNVM page address and next 63 Words – non-authenticated plaintext data) into mailbox memory.
3. Write 0x00, mailbox memory write word address offset in `MBX_WADRDESC` (register offset – 0x1C) register.
4. Write 0x01 in `SYS_SERV_REQ` (register offset – 0x0C) register to initiate sNVM write service.
5. Write sNVM page address 0x10 in mailbox memory by writing in `MBX_WDATA` (register offset – 0x28) register.
6. Write 63-words non-authenticated Plaintext data in mailbox memory by writing in `MBX_WDATA` (register offset 0x28) register. `MBX_WDATA` register accepts 1 word at a time to write 63 words, this operation must be repeated 63 times.
7. Read `SYS_SERV_USER` (register offset – 0x30) register until its bit 0- `USER_BUSY` goes low.
8. Read `SYS_SERV_STAT` (register offset – 0x08) register to read the status.

Authenticated Plaintext sNVM Write at Address 0x50

1. Write 0x11, sNVM write service command for authenticated plaintext in `SYS_SERV_CMD` (register offset – 0x04) register.
2. Write 0x3F in `MBX_WCNT` (register offset – 0x14) register to write total 63 words data (first word – sNVM page address, next 59 words - authenticated plaintext data and next 3 words - user secret key) into mailbox memory.
3. Write 0x00, mailbox memory write word address offset in `MBX_WADRDESC` (register offset – 0x1C) register.
4. Write 0x01 in `SYS_SERV_REQ` (register offset – 0x0C) register to initiate sNVM write service.
5. Write sNVM page address 0x50 in mailbox memory by writing in `MBX_WDATA` (register offset – 0x28) register.
6. Write 59-words authenticated plaintext data in mailbox memory by writing in `MBX_WDATA` (register offset 0x28) register. `MBX_WDATA` register accepts 1 word at a time to write 59 words. This operation must be repeated 59 times.
7. Write 3 words user secret key in mailbox memory by writing `MBX_WDATA` (register offset 0x28) register. `MBX_WDATA` register accepts 1 word at a time to write 3 words. This operation must be repeated 3 times.
8. Read `SYS_SERV_USER` (register offset – 0x30) until its bit 0- `USER_BUSY` goes low.
9. Read `SYS_SERV_STAT` (register offset – 0x08) to read the status.

Authenticated Ciphertext sNVM Write at Address 0x08

1. Write 0x12, sNVM write service command for authenticated ciphertext in `SYS_SERV_CMD` (register offset – 0x04) register.
2. Write 0x3F in `MBX_WCNT` (register offset – 0x14) register to write total 63 words data (first word – sNVM page address, next 59 words - authenticated ciphertext data and next 3 words - user secret key) into mailbox memory.
3. Write 0x00, mailbox memory write word address offset in `MBX_WADRDESC` (register offset – 0x1C) register.
4. Write 0x01 in `SYS_SERV_REQ` (register offset – 0x0C) register to initiate sNVM write service.
5. Write sNVM page address 0x08 in mailbox memory by writing in `MBX_WDATA` (register offset – 0x28) register.
6. Write 59-words authenticated ciphertext data in mailbox memory by writing in `MBX_WDATA` (register offset 0x28) register. `MBX_WDATA` register accepts 1 word at a time to write 59 words. This operation must be repeated 59 times.
7. Write 3 words user secret key in mailbox memory by writing in `MBX_WDATA` (register offset 0x28) register. `MBX_WDATA` register accepts 1 word at a time to write 3 words. This operation must be repeated 3 times.
8. Read `SYS_SERV_USER` (register offset – 0x30) register until its bit 0- `USER_BUSY` goes low.
9. Read `SYS_SERV_STAT` (register offset – 0x08) register to read the status.

Figure 1-4. System Services with Mailbox Write Only

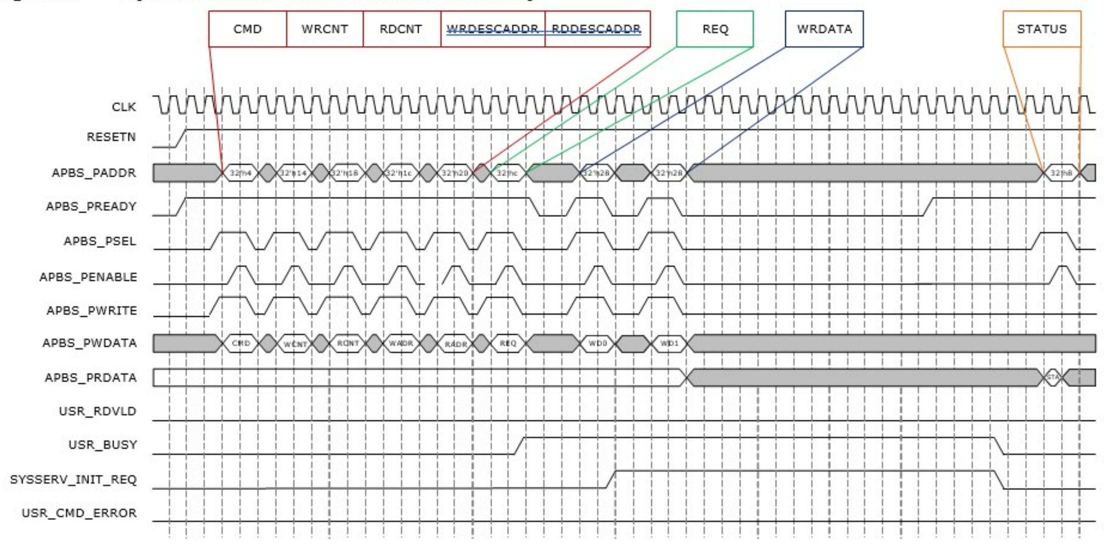
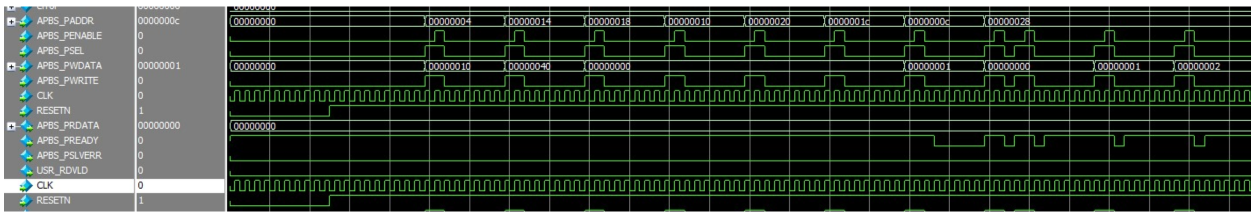


Figure 1-5. sNVM Write Transactions in Simulations



1.4.2.2 Secure NVM Read Service

The Secure NVM read service provides access to the data stored by the Secure NVM Write service or data programmed via a bit stream. If the data is programmed using authentication, USK used at the time of programming must also be provided.

Table 1-27. Secure NVM Write Request

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. See Table 1-28.
6:0	18H	Secure NVM Read service command

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

Table 1-28. 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	PT	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.

Non-Authenticated Plaintext sNVM Read at Address 0x01

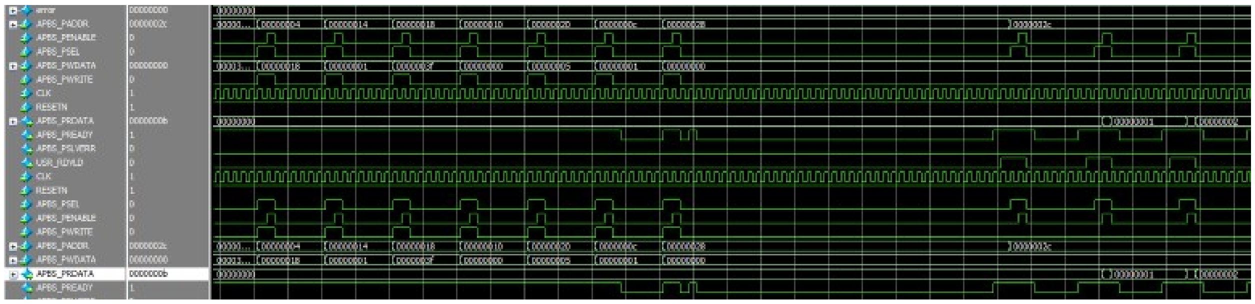
1. Write 0x18, sNVM read service command in `SYS_SERV_CMD` (register offset – 0x04) register.
2. Write 0x1 in `MBX_WCNT` (register offset – 0x14) register to write 1-word sNVM page address into mailbox memory.
3. Write 0x00, mailbox memory write word address offset in `MBX_WADRDESC` (register offset – 0x1C) register.
4. Write 0x3F in `MBX_RCNT` (register offset – 0x18) register to read 63 non-authenticated plaintext data from mailbox memory.
5. Write 0x05, mailbox memory read word address offset in `MBX_RADRDESC` (register offset – 0x20) register.
   Note: In this example, ADMIN data is not read and due to that mailbox memory read word address offset is configured to 5 (20th byte).
6. Write 0x01 in `SYS_SERV_REQ` (register offset – 0x0C) register to initiate sNVM read service.
7. Write sNVM page address 0x01 in Mailbox memory by writing in `MBX_WDATA` (register offset – 0x28) register.
8. Read `SYS_SERV_USER` (register offset – 0x30) until its bit 1- `USER_RDVLD` goes high.
9. Read 1-word non-authenticated plaintext data from mailbox memory by reading `MBX_RDATA` (register offset 0x2C) register.

10. To read the remaining 62 words non-authenticated plaintext data from mailbox memory repeat, steps 8 and 9, 62 times.
11. Read `SYS_SERV_USER` (register offset – 0x30) register until its bit 0- `USER_BUSY` goes low.
12. Read `SYS_SERV_STAT` (register offset – 0x08) register to read the status.

Authenticated Plaintext/Ciphertext sNVM Read at Address 0x20

1. Write 0x18, sNVM read service command in `SYS_SERV_CMD` (register offset – 0x04) register.
2. Write 0x4 in `MBX_WCNT` (register offset – 0x14) register to write total 4 words (first Word sNVM page address and 3 words user secret key) into mailbox memory.
3. Write 0x00, mailbox memory write word address offset in `MBX_WADRDESC` (register offset – 0x1C) register.
4. Write 0x3B in `MBX_RCNT` (register offset – 0x18) register to read 59 authenticated plaintext/ciphertext data from mailbox memory.
5. Write 0x05, mailbox memory read word address offset in `MBX_RADRDESC` (register offset – 0x20) register.
   Note: In this example, ADMIN data is not read and due to that mailbox memory read word address offset is configured to 5 (20th byte).
6. Write 0x01 in `SYS_SERV_REQ` (register offset – 0x0C) register to initiate sNVM read service.
7. Write sNVM page address 0x20 in Mailbox memory by writing in `MBX_WDATA` (register offset – 0x28) register.
8. Read `SYS_SERV_USER` (register offset – 0x30) until its bit 1- `USER_RDVLD` goes high.
9. Read 1-word authenticated plaintext/ciphertext data from mailbox memory by reading `MBX_RDATA` (register offset 0x2C) register.
10. To read the remaining 58 words authenticated plaintext/ciphertext data from Mailbox memory, repeat steps 8 and 9, 58 times.
11. Read `SYS_SERV_USER` (register offset – 0x30) register until its bit 0- `USER_BUSY` goes low.
12. Read `SYS_SERV_STAT` (register offset – 0x08) register to read the status.

Figure 1-6. sNVM Read Transactions in Simulations



1.4.3 PUF Emulation Service

The 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.
- 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. 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 you 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 derives many keys from one.

Table 1-29. PUF Emulation Service Request

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. See Table 1-30.
6:0	20H	PUF Emulation service command

The following table lists the PUF Emulation Service Mailbox Format.

Table 1-30. 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

1.4.4 Nonce Service

The nonce service generates a 256-bit random number derived from the start-up states of a dedicated SRAM. The nonce service provides you 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 1-31. Nonce Service Request

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. See Table 1-32.
6:0	21H	Nonce service command

The following table lists the Nonce Service Mailbox Format.

Table 1-32. Nonce Service Mailbox Format

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

1.5 Fabric Services

Fabric services are used to calculate digests of nonvolatile memories and program the device. Fabric services requires the user design be powered down in a controlled manner for the duration of the service. Upon completion of the service the user design either restores to its previous state or reinitialized, depending on the service requested.

For more information about the return status of fabric services, see 8. System Service Return Status Codes.

1.5.1 Digest Check Service

Digest Check service recalculates digests of selected nonvolatile 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 1-33. Digest Check Service Request

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. See Table 1-34.
6:0	47H	Digest Check service command

The following table lists the Digest Check Service mailbox format.

Table 1-34. Digest Check Service Mailbox Format

Offset	Length (bytes)	Parameter	Direction	Description
0	4	OPTIONS	Input	Digest options. See Table 1-35.
4	4	DIGESTERR	Output	See Table 1-36.

Table 1-35. OPTIONS

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.

...........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.

If CHECK FABRIC is 1, the FPGA fabric is placed in suspend state and I/Os behave in the 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, 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 1-36. A failure of any digest results in the DIGEST tamper flag being triggered. The DIGESTERR indicates zero when it is successful.

Table 1-36. DIGESTERR

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.

1.5.2 In-Application Programming (IAP)/Auto Update 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 bit stream 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 1-37. 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 1-38. IAP Verify Request by Image Address

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	Mailbox address. See Table 1-41.
6:0	45H	IAP verify operation.

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

Note: 1.5.1. Digest Check Service is recommended to verify the integrity of the device contents instead of IAP verify operation.

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 a 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 1-39. 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 1-40. IAP Request by Image Address

System Service Descriptor Bit Field	Value	Description
15:7	MBOXADDR[10:2]	For the mailbox format, see Table 1-41.

...........continued

System Service Descriptor Bit Field	Value	Description
6:0	43H	IAP program operation.

The following table lists the IAP mailbox format.

Table 1-41. 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.

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

Table 1-42. 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 a 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. Interface

The following sections describe the interface for `PF_SYSTEM_SERVICES`.

2.1 I/O Signals

The following table lists the port signals for the `PF_SYSTEM_SERVICES`.

Table 2-1. I/O Signal Descriptions

Port Name	Type	Description
Clocks and Resets		
CLK	Input	Fabric clock.
RESETN	Input	System reset. Active-Low asynchronous reset. RESETN must be synchronized with CLK clock domain externally.
APB Target Interface Signals		
`APBS_PSEL`	Input	Select signal. The APB bridge unit generates this signal to each peripheral bus target. It indicates that the target device is selected.
`APBS_PENABLE`	Input	Enable. This signal indicates the second and subsequent cycles of an APB transfer.
`APBS_PWRITE`	Input	Direction. This signal indicates an APB write access when HIGH and an APB read access when LOW.
`APBS_PADDR[31:0]`	Input	Address. This is the APB address bus. It is 32-bit wide and is driven by the peripheral bus bridge unit.
`APBS_PWDATA[31:0]`	Input	Write Data. This bus is driven by the peripheral bus bridge unit during write cycles when PWRITE is HIGH. This bus is 32-bit wide.
`APBS_PRDATA[31:0]`	Output	Read Data. The selected target drives this bus during read cycles when PWRITE is LOW. This bus is 32-bit wide. **Note:** PRDATA signal is combinatorially generated from the IP. The APB Initiator must sample it on the CLK.
`APBS_PREADY`	Output	Ready. The target uses this signal to extend an APB transfer. **Note:** PREADY signal is combinatorially generated from the IP. The APB Initiator must sample it on the CLK.
`APBS_PSLVERR`	Output	This signal indicates a transfer failure. APB peripherals are not required to support the PSLVERR pin. `PF_SYSTEM_SERVICES` never returns error response. Where a peripheral does not include this pin then the appropriate input to the APB bridge is tied LOW.
User Interface Signals		
`USR_CMD_ERROR`	Output	User Command Error. This indicates that an unsolicited command is received during the response phase. It is asserted in response to an undefined SERVICEID in the command descriptor.

...........continued		
Port Name	**Type**	**Description**
`USR_BUSY`	Output	User Service Busy. When HIGH, indicates that the IP is busy executing the service. It is asserted one clock cycle after the request is initiated from the AMBA. It is de-asserted only when the service is completed. For services requiring the mailbox reads, the signal is de-asserted only when the required number of reads for the service are completed.
`USR_RDVLD`	Output	User Service Read Valid. When HIGH, indicates that the mailbox read data is available in the mailbox read data (MBX_RDATA) register for reading through the APB interface. The `PF_SYSTEM_SERVICES` IP issues read to the System Controller's mailbox interface and fetches data corresponding to the service. Once the data is fetched in to the `MBX_RDATA` register, the IP core asserts `USR_RDVLD` signal indicating to the APB initiator that the data is available for reading. When the APB initiator reads the data, it de-asserts the `USR_RDVLD` signal. It is repeated until all the mailbox reads are completed based on the read count value configured in the `MBX_RCNT` register specific to the service. **Note:** The user design must issue number of APB reads equal to the read count configured in the `MBX_RCNT` register. For this, the `USR_RDVLD` signal must be polled to read the data in the mailbox for a given system service.
`SYSSERV_INIT_REQ`	Output	System Service request from APB interface. This signal indicates that the core has seen the System Service request from the AMBA interface.
`SS_BUSY`	Output	Service Request Busy. Indicates that the System Controller is busy processing the request. This remains set until the System Controller has completed the service request and any requested data is available in the mailbox memory. Indicates non-synchronized busy from the System Controller.

2.2 Core Parameters

The following section describes the core parameters for `PF_SYSTEM_SERVICES`.

2.2.1 PF_SYSTEM_SERVICES Configurable Options

There are a number of configurable options that apply to `PF_SYSTEM_SERVICES` as listed in the following table. If a configuration other than the default is required, use the configuration dialog box in SmartDesign to select appropriate values for the configurable options.

Table 2-2. PF_SYSTEM_SERVICES Configuration Options

Parameter Name	Valid Range	Default	Description
Device and Design Information Services			
SNSERVICE	0-1	1	Fetch the 128-bit Serial Number Service. • 0 – Disable • 1 – Enable

...........continued

Parameter Name	Valid Range	Default	Description
UCSERVICE	0-1	1	Fetch the 32-bit User Code Service. • 0 – Disable • 1 – Enable
DVSERVICE	0-1	1	Design Information Service. Returns 256-bit user-defined design ID, 16-bit design version (DCV) and 16-bit design back-level (DCB). • 0 – Disable • 1 – Enable
DCSERVICE	0-1	1	Device Certificate Service. • 0 – Disable • 1 – Enable Fetch the device Supply Chain Assurance Certificate from pNVM. Returns 1024-byte Device Certificate.
RDDIGEST	0-1	1	Returns the 416-byte stored digests for the device. • 0 – Disable • 1 – Enable
QUERYSECSERVICE	0-1	1	Reads nonvolatile states of user security locks (9-byte). 0 – Disable 1 – Enable
RDDEBUGINFO	0-1	1	Reads out useful debug information for programming, user initialization and IAP. • 0 – Disable • 1 – Enable
RDENVMPARAM	0-1	1	Retrieves all parameters needed for eNVM operation and programming • 0 – Disable • 1 – Enable **Note:** The system service is available for PF SoC devices only.
Design Services			
AUTHBITSTREAM	0-1	1	Used to authenticate the UIC Bit stream before using IAP service. • 0 – Disable • 1 – Enable
AUTHIAPIMG	0-1	1	Allows you to validate an IAP image stored in SPI Flash. • 0 – Disable • 1 – Enable
Data Security Services			

...........continued

Parameter Name	Valid Range	Default	Description
DIGSIGSERVICE	0-1	1	Digital Signature service. • 0 – Disable • 1 – Enable
SECNVMWR	0-1	1	Provides write access to pages in the sNVM region of the pNVM. Performs the following Secure NVM Functions: • Non-authenticated plaintext • Authenticated plaintext • Authenticated ciphertext 0 – Disable 1 – Enable
SECNVMRD	0-1	1	Provides access to the data stored by the Secure NVM Write service or data programmed through a bit stream. 0 – Disable 1 – Enable
PUFEMSERVICE	0-1	1	The PUF emulation service provides a mechanism for authenticating a device. 0 – Disable 1 – Enable
NONCESERVICE	0-1	1	The nonce service provides you with the ability to strengthen the DRBG of the Athena F5200B by providing an alternate entropy source to use as additional seed data in its DRBG functions. • 0 – Disable • 1 – Enable
Fabric Services			
DIGESTCHECK	0-1	1	Recalculates and compares digests of selected nonvolatile memories. • 0 – Disable • 1 – Enable
IAPSERVICE	0-1	1	The IAP service allows you to reprogram the device without the need for an external initiator.Your design writes the bit stream to be programmed into a SPI Flash connected to the SPI port. When the service is invoked, the System Controller automatically reads the bit stream from the SPI Flash and programs the device. 0 – Disable 1 – Enable
IAPAUTOUPD	0-1	1	Performs the auto-update sequence. 0 – Disable 1 – Enable

3. Timing Diagrams

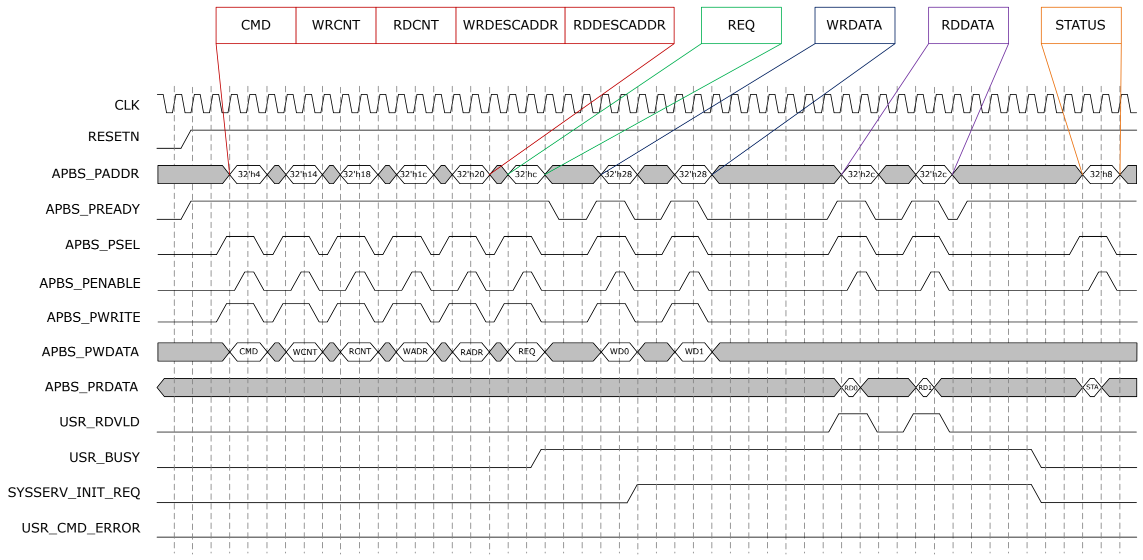
The following sections describe the system services timing diagrams.

3.1 System Services with Both Mailbox Write and Read

The following system services require mailbox write and mailbox read.

- Digital signature RAW format
- Digital signature DER format
- Secure NVM read
- PUF Emulation

Figure 3-1. System Services with Both Mailbox Write and Read

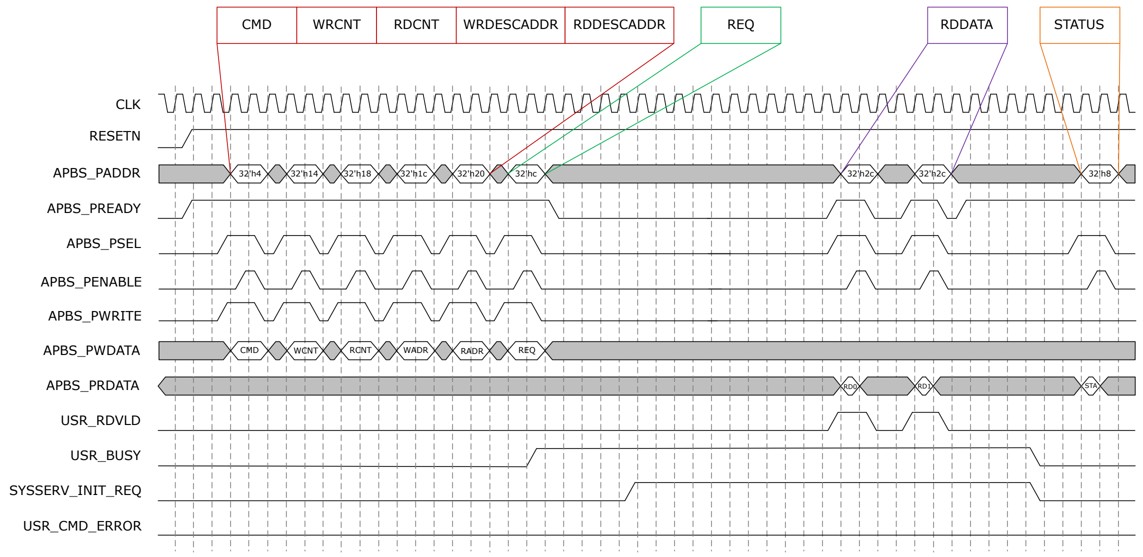


3.2 System Services with Mailbox Read Only

The following system services require mailbox read-only.

- Serial number
- User code
- Design info
- Device certificate
- Read digest
- Query security
- Read debug info
- Nonce

Figure 3-2. System Services with Mailbox Read Only



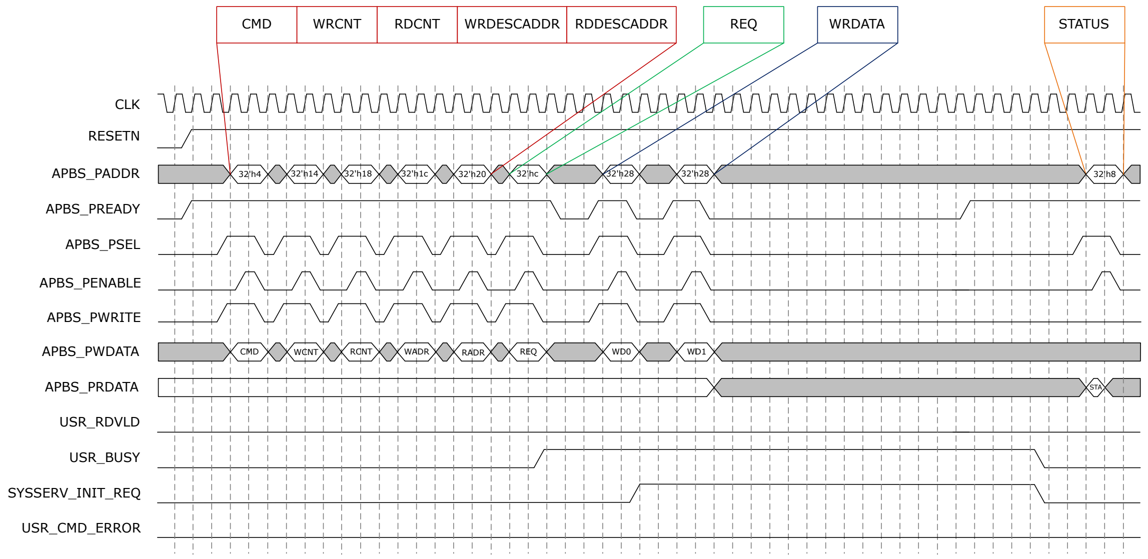
Note: You can skip the registers such as MBX_WCNT (WRCNT), MBX_WADRDESC(WRDESCADDR), and so on, which are not required for Read only System Services.

3.3 System Services with Mailbox Write Only

The following system services require mailbox write-only.

- Bit stream authentication
- Secure NVM write
- Digest check
- IAP verify by SPIADDR
- IAP program by SPIADDR

Figure 3-3. System Services with Mailbox Write Only



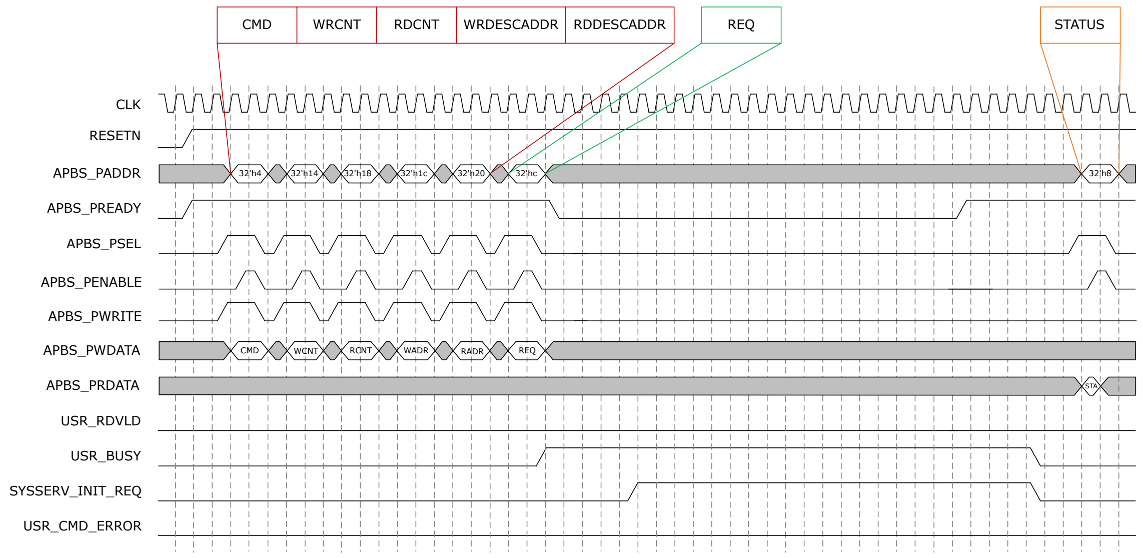
Note: You can skip the registers such as MBX_RDCNT(RDCNT), MBX_RADRDESC (RDDESCADDR), and so on, which are not required for Write only System Services.

3.4 System Services without Mailbox Write or Read

The following system services do not require mailbox write and mailbox read.

- IAP image authentication
- IAP program by index
- IAP verify by index
- IAP auto update

Figure 3-4. System Services without Mailbox Write or Read



4. Tool Flow

The following sections describe the tool flow for `PF_SYSTEM_SERVICES`.

4.1 License

`PF_SYSTEM_SERVICES` does not require a license to be used and instantiated.

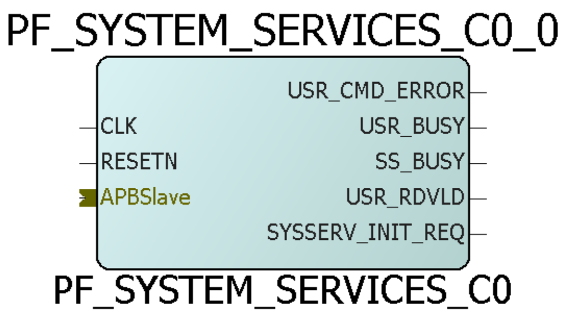
4.1.1 RTL

The complete Verilog RTL source code is provided for `PF_SYSTEM_SERVICES`.

4.2 SmartDesign

`PF_SYSTEM_SERVICES` is pre-installed and can be configured using the SmartDesign IP deployment design environment.

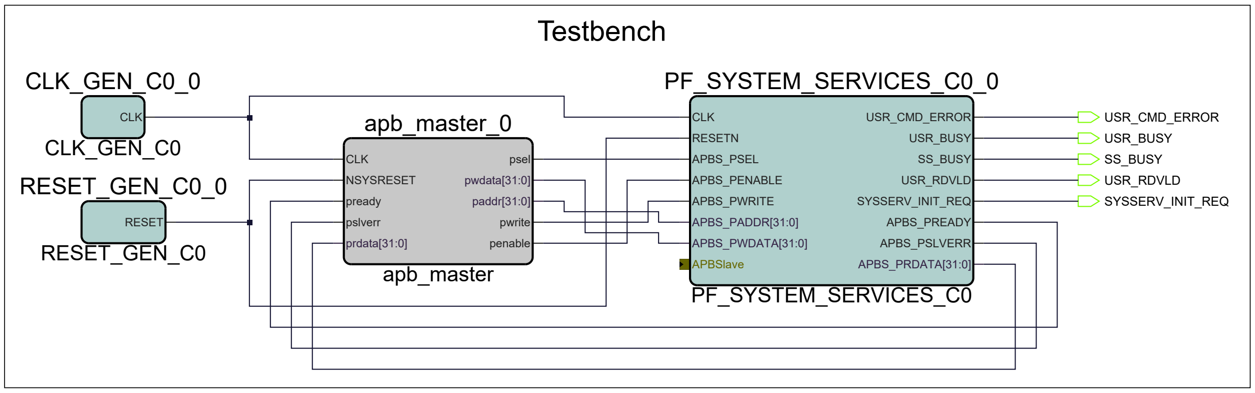
Figure 4-1. SmartDesign PF_SYSTEM_SERVICES Instance View



4.3 Simulation Flows

`PF_SYSTEM_SERVICES` can be simulated using the following simulation environment. The required vsim commands and access flow are explained in individual services.

Figure 4-2. System Services Test Bench



Serial Number Service Simulation

You can pass the desired DSN either by passing the value using the `vsim` command or by port mapping during the instantiation.

```
vsim -L PolarFire -L presynth -L CORESYSSERVICES_PF_LIB -gSRLNUM=128'h12345678 -t 1ps -
gSIM_PA5M300T=0 presynth.ss_tb
```

The following is the Serial Number service simulation log.

```
SysServices: Device Serial Number service request received at time             11450000 ps.
# Serial Number 0x00000000000000000000000123456789 is being written to Mailbox Address 0x000.
```

USERCODE Service Simulation

You can pass the desired USERCODE either by passing the value using the `vsim` command or by port mapping during instantiation.

```
vsim -L PolarFire -L presynth -L CORESYSSERVICES_PF_LIB -gUSRCD =32'h1234 -t 1ps -
gSIM_PA5M300T=0 presynth.ss_tb
```

The following is the USERCODE service simulation log.

```
SysServices: User Code service request received at time             18050000 ps.
#              User Code 0x00000000 is being written to Mailbox Address 0x000
```

Design Information Service Simulation

You can pass the desired DSGNINFO either by passing the value using the `vsim` command or by port mapping during instantiation.

```
vsim -L PolarFire -L presynth -L CORESYSSERVICES_PF_LIB -gDSGNINFO =288'h1234 -t 1ps -
gSIM_PA5M300T=0 presynth.ss_tb
```

The following is the Design Information service simulation log.

```
Services: Design INFO service request received at time             73970000 ps.
#              Design Id 0x0000000000000000000000000000000000000000000000000000000000001234
is being written to Mailbox Address 0x000.
#              Design Version 0x0000 is being written to Mailbox Address 0x00000020.
#              Back Level 0x0000 is being written to Mailbox Address 0x00000022.
#                76210             Device Version Services Completed
```

Design Certificate Service Simulation

You can pass the desired CERT either by passing the value using the `vsim` command or by port mapping during instantiation.

```
vsim -L PolarFire -L presynth -L CORESYSSERVICES_PF_LIB -gCERT=288'h1234 -t 1ps -
gSIM_PA5M300T=0 presynth.ss_tb
```

The following is the Design Certificate service simulation log.

```
SysServices: Device Certificate service request received at time             13370000 ps.
# Design Certificate
0xxxxxxxxxxxxxxxx000000000000000000000000000000000000000000000000000000000000000000001234 is
being written to Mailbox Address 0x000.
# 69950 Device Certificate Service Completed
```

Read Digests Service Simulation

You can pass the desired RDDGST either by passing the value using the `vsim` command or by port mapping during instantiation.

```
vsim -L PolarFire -L presynth -L CORESYSSERVICES_PF_LIB -gRDDGS=3328'h1234 -t 1ps -
gSIM_PA5M300T=0 presynth.ss_tb
```

The following is the Read Digests service simulation log.

```
SysServices: Read Digest service request received at time             79010000 ps.
# Fabric Digest 0x000000000000000000000000000000000000000000000000000000000000000d is being
written to Mailbox Address 0x000.
# UFS CC Segment Digest 0x000000000000000000000000000000000000000000000000000000000000000c is
being written to Mailbox Address 0x00000020.
# SNVM Digest 0x000000000000000000000000000000000000000000000000000000000000000b is being
written to Mailbox Address 0x00000040.
# UFS UL Segment 0x000000000000000000000000000000000000000000000000000000000000000a is being
written to Mailbox Address 0x00000060.
# User Key Digest 0 0x0000000000000000000000000000000000000000000000000000000000000009 is
being written to Mailbox Address 0x00000080.
# User Key Digest 1 0x0000000000000000000000000000000000000000000000000000000000000008 is
being written to Mailbox Address 0x000000a0.
# User Key Digest 2 0x0000000000000000000000000000000000000000000000000000000000000007 is
being written to Mailbox Address 0x000000c0.
# User Key Digest 3 0x0000000000000000000000000000000000000000000000000000000000000006 is
being written to Mailbox Address 0x000000e0.
# User Key Digest 4 0x0000000000000000000000000000000000000000000000000000000000000005 is
being written to Mailbox Address 0x00000100.
# User Key Digest 5 0x0000000000000000000000000000000000000000000000000000000000000004 is
being written to Mailbox Address 0x00000120.
# User Key Digest 6 0x0000000000000000000000000000000000000000000000000000000000000003 is
being written to Mailbox Address 0x00000140.
# UFS UPERM Segment Digest 0x0000000000000000000000000000000000000000000000000000000000000002
is being written to Mailbox Address 0x00000160.
# Factory Digest 0x0000000000000000000000000000000000000000000000000000000000000001 is being
written to Mailbox Address 0x00000180.
# 103150 Read Digests System Services Completed
```

Query Security Service Simulation
You can pass the desired QRYSEC either by passing the value using the `vsim` command or by port mapping during instantiation.

```
vsim -L PolarFire -L presynth -L CORESYSSERVICES_PF_LIB -gQRYSEC=72'h1234 -t 1ps -
gSIM_PA5M300T=0 presynth.ss_tb
```

The following is the Query Security service simulation log.

```
SysServices: Query Security service request received at time             77650000 ps.
#    Lock Array 0x000000000000001234 is being written to Mailbox Address 0x000.
```

Read Debug Information Service Simulation
You can pass the desired RDDBGINF either by passing the value using the `vsim` command or by port mapping during instantiation.

```
vsim -L PolarFire -L presynth -L CORESYSSERVICES_PF_LIB -gRDDBGINF=672'h1234 -t 1ps -
gSIM_PA5M300T=0 presynth.ss_tb
```

The following is the Read Debug Information service simulation log.

```
SysServices: Read Debug Info service request received at time            104590000 ps.
# Read Debug Info
0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000001234 is being written
to Mailbox Address 0x000.
# 109470 Read Debug Info System Services Completed
```

Bitstream Authentication Service Simulation
In simulation, only the Bitstream Authentication service receive request is displayed.

```
SysServices: Bitstream Authentication service request received at time  11890000 ps.
```

IAP Image Authentication Service Simulation
In simulation, only the IAP image authentication service receive request is displayed.

```
SysServices: IAP Image Authentication service request received at time  17090000 ps.
```

Digital Signature Service Simulation
In simulation model, data moving occurs with the proper display messages, but no processing of the data is done.

```
SysServices: Digital Signature RAW service request received at time  12430000 ps.
# Hash located at Mailbox Address 0x000 is being signed and stored at Mailbox  Address
0x00000030
# 17970  Digital Signature RAW System Services Completed
```

Secure NVM Write Service Simulation
In simulation, the data is written starting from the offset + 4.

```
SysServices: Non-Authenticated PlainText SNVM Write service request received at time
126410000 ps.
# Data located at Mailbox Address 0x00000004 is being written to the SNVM at Address 0x000
# 126550   SEC NVM WRITE Completed
```

Secure NVM Read Service Simulation
In simulation, the data is read starting from the offset + 14.

```
SysServices: SNVM Read service request received at time  128310000 ps.
# Data located at SNVM Address 0x000 is being written to the Mailbox at Address 0x00000014
```

PUF Emulation Service Simulation
In simulation, the first four bytes are not supported. Data is written at offset+4 address.

```
SysServices: PUF Emulation service request received at time 11950000 ps.
# Challenge Input is stored at Mailbox address 0x00000004 and the Response will be written to
the Mailbox at address 0x00000014
```

Nonce Service Simulation
In simulation, a random number is stored in the mailbox offset address and might not match the silicon.

```
SysServices:  Received NONCE service request
# SysServices:  Generated Nonce :
000000000000000000000000000000000000000000000000000000fbb52986 is read from Mailbox Offset
Address :0
#
#13470  Nonce Services Completed
```

Digest Check Service Simulation
In simulation, only the digest check service receive request is displayed.

```
SysServices:  Received Digest options at Mailbox Offset address : 0
# SysServices:  Executed Digest Check service
```

In-Application Programming (IAP)/Auto Update Service Simulation
In simulation, the In-Application Programming (IAP)/Auto update service receive request is displayed.

```
SysServices:  Received IAP Program by Index service request
# SysServices:  Executed IAP Program by Index service
SysServices  :  Received IAP Program by SPIADDR service request
# SysServices:  Executed IAP Program by SPIADDR service
```

```
SysServices:  Received IAP Program by Auto Update service request
# SysServices:  Executed IAP Program by Auto Update service

SysServices:  Received IAP Program by SPIADDR service request
# SysServices:  Executed IAP Program by SPIADDR service
```

Simulation Error Response on System Services
In simulation, you can provide the error/success status within a text file (`.txt`) using the `vsim` command. For example, use the following example command to pass the `ERROR_RESPONSE.txt` file to simulation model.

```
vsim -L PolarFire -L presynth -L CORESYSSERVICES_PF_LIB  -
gSYS_SERVICES_RESPONSE_FILE=ERROR_RESPONSE.txt -t 1ps presynth.ss_tb
```

The following is the format of error status to be written in the text file.

```
[14:8] – SERVICEID
[7:0] – Error Status
```

The following is a snippet from a sample error status text file.

```
0024      // Serial Number Service = 00H, Response = 24H
0105      // USERCODE Service = 01H Response = 05H
0202      // Design Info Service = 02H Response = 02H
0303      // Device Certificate Service = 03H Response = 03H (System error : PUF or storage
failure )
0424      // Read Digests Service = 04H Response = 24H
0524      // Query Security Service  = 05H Response = 24H
0624      // Read Debug Info Service = 06H Response = 24H
0724      // eNVM Parameters Info Service = 07H Response = 24H
2302      // Bitstream Authentication Service = 23H, Response = 02H(Unexpected data received)
2202      // IAP Image Authentication Service  = 22H, Response = 02H(Unexpected data received)
1901      // Digital Signature Service(Raw Format)  = 19H, Response = 01H(FEK Failure)
1A02      // Digital Signature Service(DER Format)  = 1AH, Response = 02H(DRBG Error)
1002      // Secure NVM Write Service(Non-authenticated plaintext) = 10H, Response =
02H(Write failure)
1103      // Secure NVM Write Service(Authenticated plaintext) = 11H, Response = 03H(System
error)
1204      // Secure NVM Write Service(Authenticated ciphertext) = 12H, Response = 04H(Write
Not Permitted)
1803      // Secure NVM Read  Service = 18H, Response = 03H(System error)
2001      // PUF Emulation Service  = 20H, Response = 01H(Internal error)
2101      // Nonce Service   = 21H, Response = 01H(Error fetching PUK)
4701      // Digest Check Service   = 47H, Response = 01H(FABRICERR)
4202      // IAP Program by Index Service   = 42H, Response = 02H(Unexpected data received)
4403      // IAP Verify by Index Service   = 44H, Response = 03H(Invalid/corrupt encryption
key)
4304      // IAP Program by SPIADDR Service   = 43H, Response = 04H(Invalid component header)
4505      // IAP Verify by SPIADDR Service   = 45H, Response = 05H(Back level not satisfied)
4606      // IAP Verify by SPIADDR Service   = 46H, Response = 06H(Illegal bitstream mode)
```

4.4 Synthesis in Libero

To run synthesis on the core, set the design root to the SmartDesign design and click **Synthesis** in Libero SoC. The Synthesis window displays the Synplify® project. To run Synthesis, click **Run**.

4.5 Place-and-Route in Libero

After the design is synthesized, run the compilation and then place-and-route in the tool. `PF_SYSTEM_SERVICES` does not require any special place-and-route settings.

5. Register Map and Descriptions

The PF_SYSTEM_SERVICES registers are accessible through the APB target interface. The following table lists all the registers required to execute system services on PolarFire devices.

Table 5-1. Register Map

Address Offset	Register Name	Type	Width	Reset Value	Description
0x04	SYS_SERV_CMD	R/W	32	0	Request Command
0x08	SYS_SERV_STAT	R/O	32	0	Response Status
0x0C	SYS_SERV_REQ	R/W	32	0	Control Register
0x10	MBX_ECCSTATUS	R/O	32	0	Mailbox ECC Status
0x14	MBX_WCNT	R/W	32	‘d1	Mailbox Write Count
0x18	MBX_RCNT	R/W	32	‘d1	Mailbox Read Count
0x1C	MBX_WADRDESC	R/W	32	0	Mailbox Write Address Descriptor
0x20	MBX_RADRDESC	R/W	32	0	Mailbox Read Address Descriptor
0x28	MBX_WDATA	R/W	32	0	Mailbox Write Data
0x2C	MBX_RDATA	R/O	32	0	Mailbox Read Data
0x30	SYS_SERV_USER	R/O	32	0	System Service User

5.1 Register Map Details

The following section describes the register map details for PF_SYSTEM_SERVICES.

Table 5-2. SYS_SERV_CMD Register

Register Offset	Register Name	Field Name	Width	Bit	Type	Reset Value	Description
0x04	SYS_SERV_CMD	Command	16	0	R/W	0x0	Requested System Service Command
		Reserved	16	16	—		Reserved

Table 5-3. SYS_SERV_STAT Register

Register Offset	Register Name	Field Name	Width	Bit	Type	Reset Value	Description
0x08	SYS_SERV_STATUS	Service Status	16	0	R/O	0x0	Received System Service Status. This register contains the contents reported by SS_STATUS signal
		Reserved	16	16	—		Reserved

Table 5-4. SYS_SERV_REQ Register

Register Offset	Register Name	Field Name	Width	Bit	Type	Reset Value	Description
0x0C	SYS_SERV_REQ	Request	1	0	R/W	0x0	System Service Request is initiated. This bit is set when the service is to be initiated. **Note:** For services that do not require data to be written into mailbox, the request bit can be set to initiate the service.
		SYSSERV_BUSY	1	1	R		Busy due to System Service initiated from AMBA (APB) interface.
		Reserved	1	2	—		Reserved
		SS_BUSY	1	3	R		This contains the busy status of the `SS_BUSY` signal on the SSI interface connected to the System Controller.
		SYSSERV_REQ_INIT	1	4	R		Indicates that the System Service request initiated from AMBA interface is in progress.
		Reserved	27	5	—		Reserved

Table 5-5. MBX_ECCSTATUS Register

Register Offset	Register Name	Field Name	Width	Bit	Type	Reset Value	Description
0x10	MBX_ECCSTATUS	Mailbox ECC Status	2	0	R/W	0x0	Mailbox ECC Status. Indicates the ECC error reported by Mailbox interface.
		Reserved	30	2	—		Reserved

Table 5-6. MBX_WCNT Register

Register Offset	Register Name	Field Name	Width	Bit	Type	Reset Value	Description
0x14	MBX_WCNT	Mailbox Write Burst Count	9	0	R/W	0x1	Mailbox write burst count. This register contains the number of words to be written to Mailbox memory for the specified service.
		Reserved	23	9	—		Reserved

Table 5-7. MBX_RCNT Register

Register Offset	Register Name	Field Name	Width	Bit	Type	Reset Value	Description
0x18	MBX_RCNT	Mailbox Read Burst Count	9	0	R/W	0x1	Mailbox read burst count. This register contains the no of words to be read from Mailbox memory for the specified service.
		Reserved	23	—	—		Reserved

Table 5-8. MBX_WADRDESC Register

Register Offset	Register Name	Field Name	Width	Bit	Type	Reset Value	Description
0x1C	MBX_WADRDESC	Mailbox Write Offset Address	9	0	R/W	0x0	Mailbox Write Offset Address. This register contains the address offset of the Mailbox memory where the data need to be written. **Note:** Mailbox Write Offset Address must be word-aligned.
		Reserved	23	9	—		Reserved

Table 5-9. MBX_RADRDESC Register

Register Offset	Register Name	Field Name	Width	Bit	Type	Reset Value	Description
0x20	MBX_RADRDESC	Mailbox Read Offset Address	9	0	R/W	0x0	Mailbox Read Offset Address. This register contains the address offset of the Mailbox memory where the data need to be read from. **Note:** Mailbox Write Offset Address must be word-aligned.
		Reserved	23	9	—		Reserved

Table 5-10. MBX_WDATA Register

Register Offset	Register Name	Field Name	Width	Bit	Type	Reset Value	Description
0x28	MBX_WDATA	Mailbox Write Data	32	0	R/W	0x0	Mailbox Write Data Register. This register contains the write data to be written to the Mailbox memory.

Table 5-11. MBX_RDATA Register

Register Offset	Register Name	Field Name	Width	Bit	Type	Reset Value	Description
0x2C	MBX_RDATA	Mailbox Read Data	32	0	R/O	0x0	Mailbox Read Data Register. This register contains the read data to be read from the Mailbox memory.

Table 5-12. SYS_SERV_USER

Register Offset	Register Name	Field Name	Width	Bit	Type	Reset Value	Description
0x30	SYS_SERV_USER	User Busy	1	0	R/O	0x0	Service Busy. When HIGH, indicates that the service is in progress. Only one system service command is supported at a time. **Note:** The signal is asserted one clock cycle after the AMBA or non-AMBA request is initiated. It is de-asserted when the service is completed. **Note:** The User Busy bit is updated one cycle late for the assertion and de-assertion cycles as compared to the USR_BUSY signal.
		USR_RDVLD	1	1	R/W		Service Read Valid. When HIGH, indicates that the mailbox read data is available in the mailbox read data register for read. When an APB read is issued on the `MBX_RDATA` register, a read is issued on the mailbox interface that fetches the data. **Note:** The User design must issue number of APB reads equal to the read count configured in the `MBX_RCNT` register. For this, the `USR_RDVLD` signal must be polled to read the data in the mailbox for a given system service.
		USR_CMDERROR	1	2	R		Command Error. This indicates that an unsolicited command is received during the response phase.
		Reserved	29	3	—		Reserved.

6. System Integration

This section provides guidelines to ease the integration of `PF_SYSTEM_SERVICES`.

Figure 6-1. System Level Integration of PF_SYSTEM_SERVICES in Libero

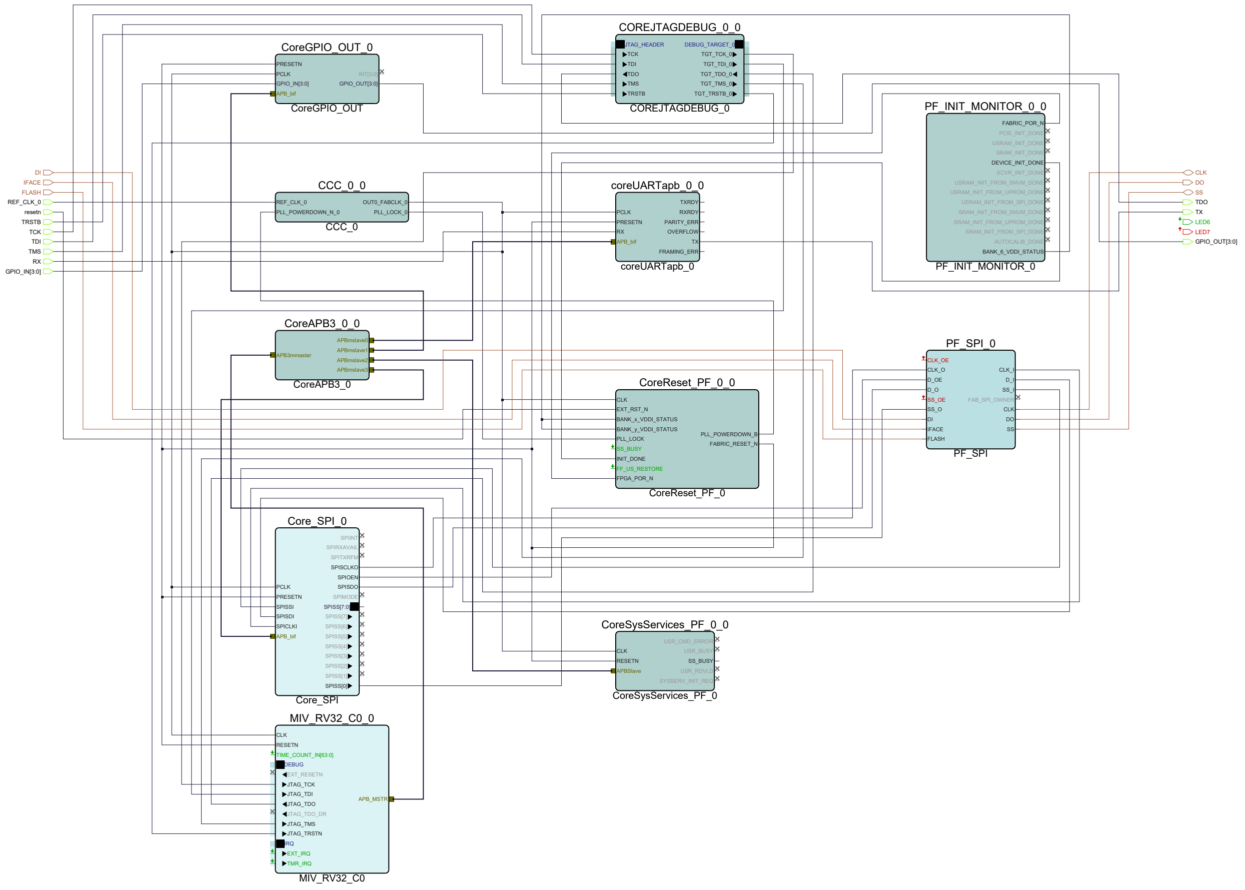


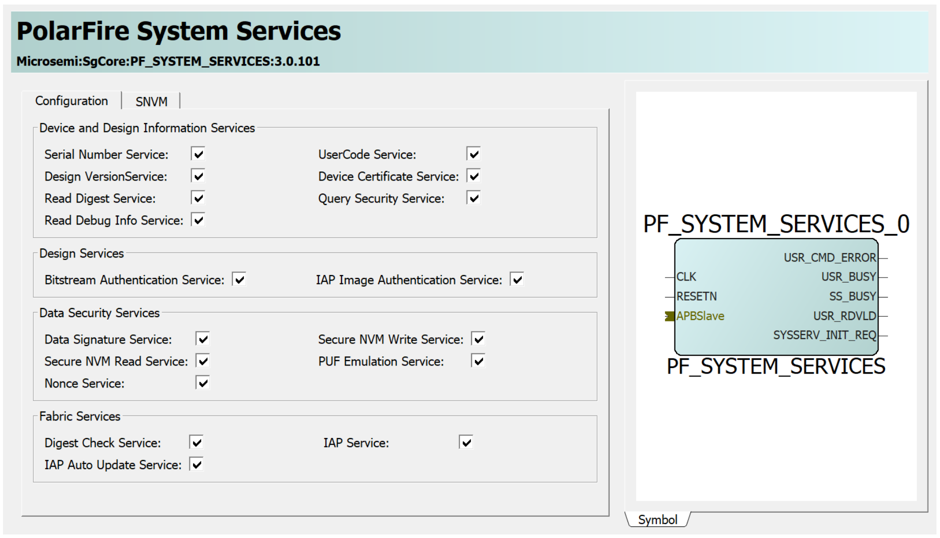
Table 6-1. Signals and Descriptions

Signal	Description
REF_CLK_0	Input 50 MHz clock from the on-board 50 MHz oscillator
resetn	On-board reset push button for the PolarFire device
RX	Input signals received from the serial UART terminal
TX	Output signals transmitted to the serial UART terminal
GPIO_OUT[3:0]	On-board LED outputs

6.1 PF_SYSTEM_SERVICES Configuration

`PF_SYSTEM_SERVICES` provides access to the System Controller. It is connected to Mi-V soft processor as an APB target. By default, all the service check boxes of the `PF_SYSTEM_SERVICES` are selected. The application can initiate these selected services. The configuration settings of the `PF_SYSTEM_SERVICES` are shown in the following figure.

Figure 6-2. PF_SYSTEM_SERVICES Configuration


PolarFire System Services

Microsemi:SgCore:PF_SYSTEM_SERVICES:3.0.101

Configuration | SNVM

Device and Design Information Services

Serial Number Service:	✓	UserCode Service:	✓
Design VersionService:	✓	Device Certificate Service:	✓
Read Digest Service:	✓	Query Security Service:	✓
Read Debug Info Service:	✓		

Design Services

Bitstream Authentication Service:	✓	IAP Image Authentication Service:	✓

Data Security Services

Data Signature Service:	✓	Secure NVM Write Service:	✓
Secure NVM Read Service:	✓	PUF Emulation Service:	✓
Nonce Service:	✓		

Fabric Services

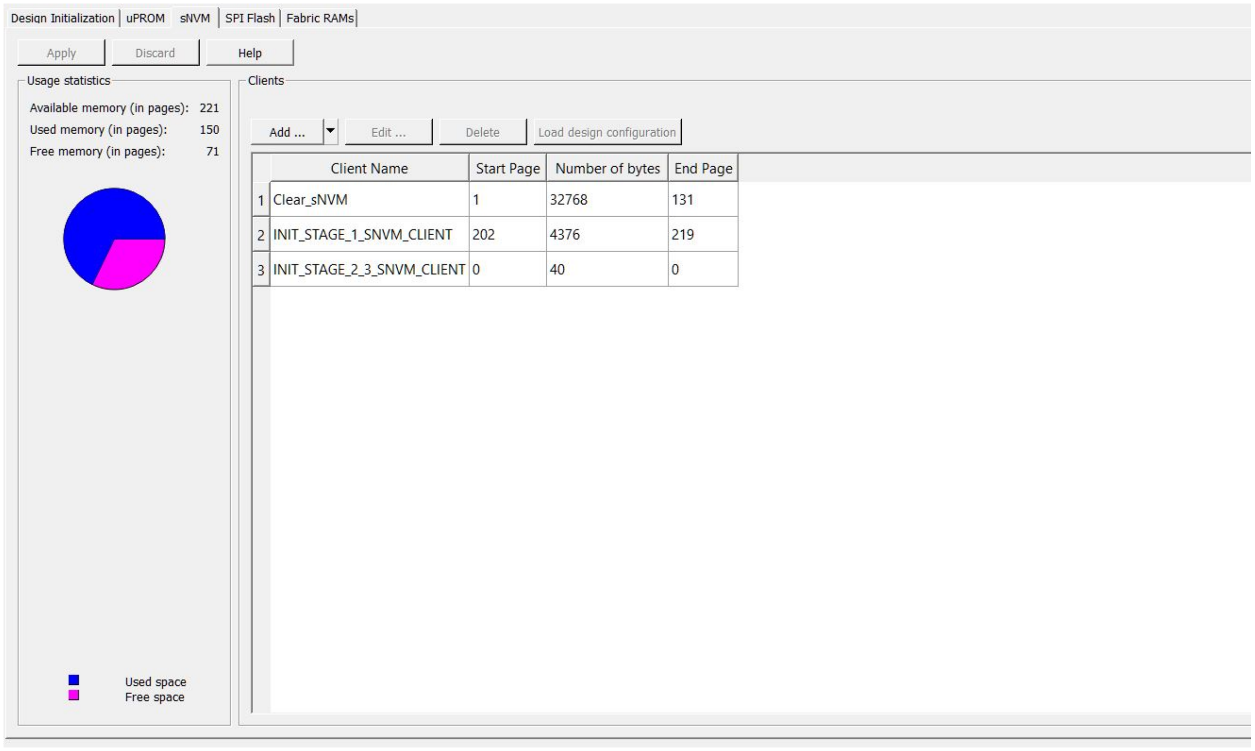
Digest Check Service:	✓	IAP Service:	✓
IAP Auto Update Service:	✓		

7. sNVM Configuration

`PF_SYSTEM_SERVICES` configurator also gives you the option of specifying data storage clients in the user nonvolatile Flash memory to store your application firmware image or other types of data, such as data co-efficients.

To add a new client to the sNVM, click **Add** in the sNVM tab.

Figure 7-1. sNVM Configuration using PF_SYSTEM_SERVICES Configurator



Available memory (in pages): 221
Used memory (in pages): 150
Free memory (in pages): 71

	Client Name	Start Page	Number of bytes	End Page
1	Clear_sNVM	1	32768	131
2	INIT_STAGE_1_SNVM_CLIENT	202	4376	219
3	INIT_STAGE_2_3_SNVM_CLIENT	0	40	0

7.1 Usage Statistics

Usage statistics displays the total memory size of the sNVM, the size of used memory, and free memory. All memory sizes are expressed in terms of pages.

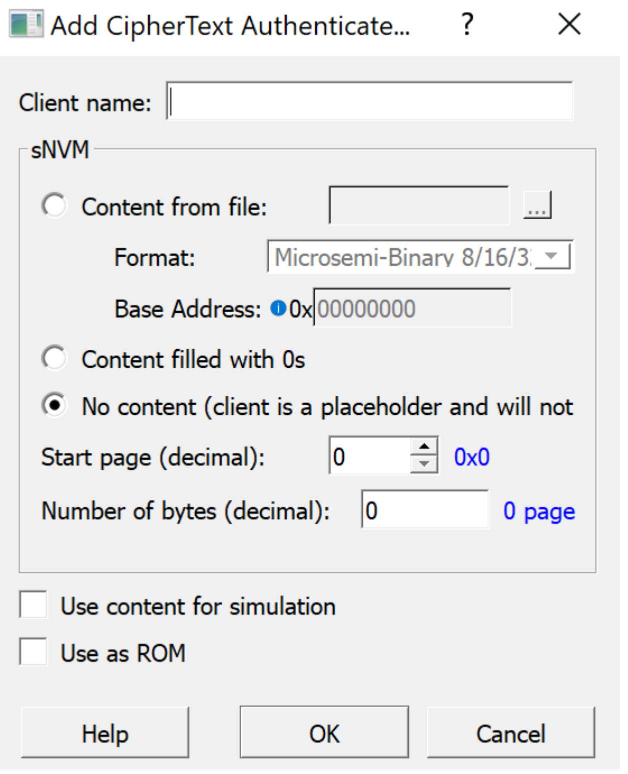
Note: Each PolarFire FPGA has 56 Kbytes of sNVM. The sNVM is organized into 224 pages, each page is of 256 bytes in size. Three pages are reserved for administrative purposes, the remaining 221 pages are available for user data. See the device data sheet for specific information on Flash Memory size.

7.2 Add Clients to System

You can choose one of the following client types:

- PlainText NonAuthenticated Client
- PlainText Authenticated Client
- CipherText Authenticated Client
- USK Client

Figure 7-2. Add Client Dialog Box


Add CipherText Authenticate...

Client name:

sNVM

( ) Content from file:

Format: Microsemi-Binary 8/16/3

Base Address: 0x 00000000

( ) Content filled with 0s

(•) No content (client is a placeholder and will not

Start page (decimal): 0 0x0

Number of bytes (decimal): 0 0 page

[ ] Use content for simulation

[ ] Use as ROM

To Add Clients to the system, click **Add** to open the Client dialog box. This dialog box enables you to specify the following options:

- Client name
- Memory content for Client (Content from file or Content filled with 0s or No Content)
- Start page
- Use content for simulation
- Use as ROM

7.2.1 Client Name

Enter a name for your memory client.

7.2.2 Content from File

You can load your memory client from a memory file with this option.

1. Click **Browse** to navigate to the location of the memory file you want to load.
2. You need to choose the appropriate type of file for the filter (Files of Type Combo Box) as sNVM supports Microchip-Binary, Intel-Hex, Motorola-S, and Simple-Hex formats for the memory content.

When you import a file, you have three choices regarding the physical location of the file in disk. You can select:

- Absolute path— File can be in any folder accessible by Libero software.
- Relative path— File can be in any folder relative to the project folder accessible by Libero software.
- Copy memory file to project directory— File is copied from any folder location accessible by Libero software into the project directory.
  The number of rows in the file (word count) must be less than or equal to the memory space of the sNVM (up to 221 pages).

7.2.3 Content Filled with 0s

Fill the content of the memory client with 0s as a place holder. You can update the memory client after Place-and-Route and before Programming.

7.2.4 No Content

Use this option to reserve pages for memory client as a place holder. You can update the memory client after Place-and-Route and before Programming.

7.2.5 Start Page

Enter the Start page of your client. Valid values are from 0 to 221 (decimal). While specifying the start page number, it must be noted that sNVM pages (up to 17 pages) are reserved for device initialization clients starting from page number 204.

7.2.6 Number of Bytes

Based on contents of file, the number of bytes and the number of pages are computed and displayed. This field is meant for information purposes (read-only).

7.2.7 Use Content for Simulation

Select this check box to include your memory content for simulation. When this box is checked, a `sNVM.mem` file is automatically created in the `<prj_location>/simulation` folder when simulation is invoked in the Design Flow window. The `snvm.mem` file contains byte by byte data of imported Microchip-Binary or Intel-Hex or Motorola-S or Simple-Hex file.

Example:

If you import a Microchip-Binary file having 4 bytes of data: 11000000000001100000000000000010

The `snvm.mem` file for the above content is shown in the following codeblock.

```
@0
00000010
@1
00000000
@2
00000110
@3
11000000
```

Only clients with the Use Client for Simulation check box checked have the contents in simulation folder that is, `snvm.mem` file for simulation.

Note: The clients you have added appear in the User clients in SNVM page.

7.2.8 Use as ROM

Check this box to write-protect the pages used by the client. The ROM status of a page cannot be changed after power-up and it can only be changed by programming a new bit stream (with the check box un-checked).

7.3 DRC Rules and Error Messages

To prevent overlapping of address space, DRC rules are enforced and error messages are given when:

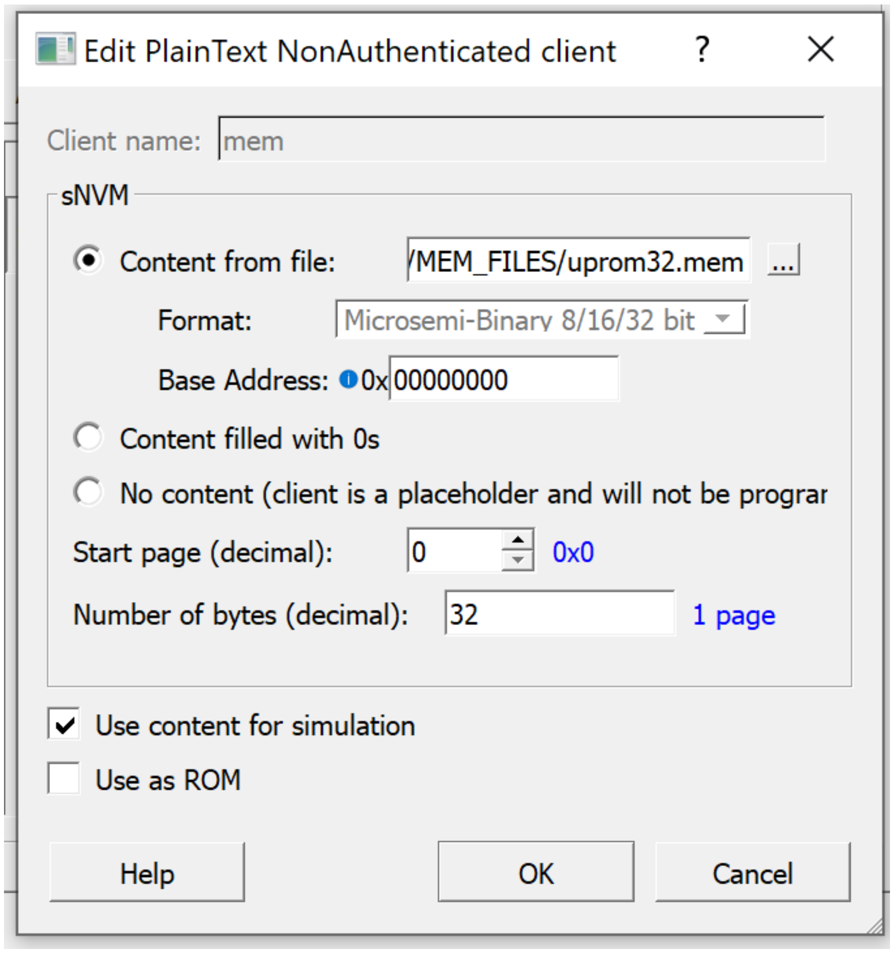
- There is more than one user client and the address range of one client overlaps with that of another.
- The memory file `(*.mem)` size exceeds the total sNVM memory space.

7.4 Editing a Client

To edit a client, follow the steps.

1. Right-click the client and choose **Edit** to open the Edit Data Storage Client dialog box.
2. Make your changes in the Edit Data Storage Client dialog box.
3. Click **OK**.

Figure 7-3. Edit Data Storage Client Dialog Box



Edit PlainText NonAuthenticated client

Client name: mem

sNVM

(●) Content from file: /MEM_FILES/uprom32.mem

Format: Microsemi-Binary 8/16/32 bit

Base Address: 0x 00000000

( ) Content filled with 0s

( ) No content (client is a placeholder and will not be prograr

Start page (decimal): 0 0x0

Number of bytes (decimal): 32 1 page

[✓] Use content for simulation

[ ] Use as ROM

7.5 Deleting a Client

Right-click the client and choose **Delete**.

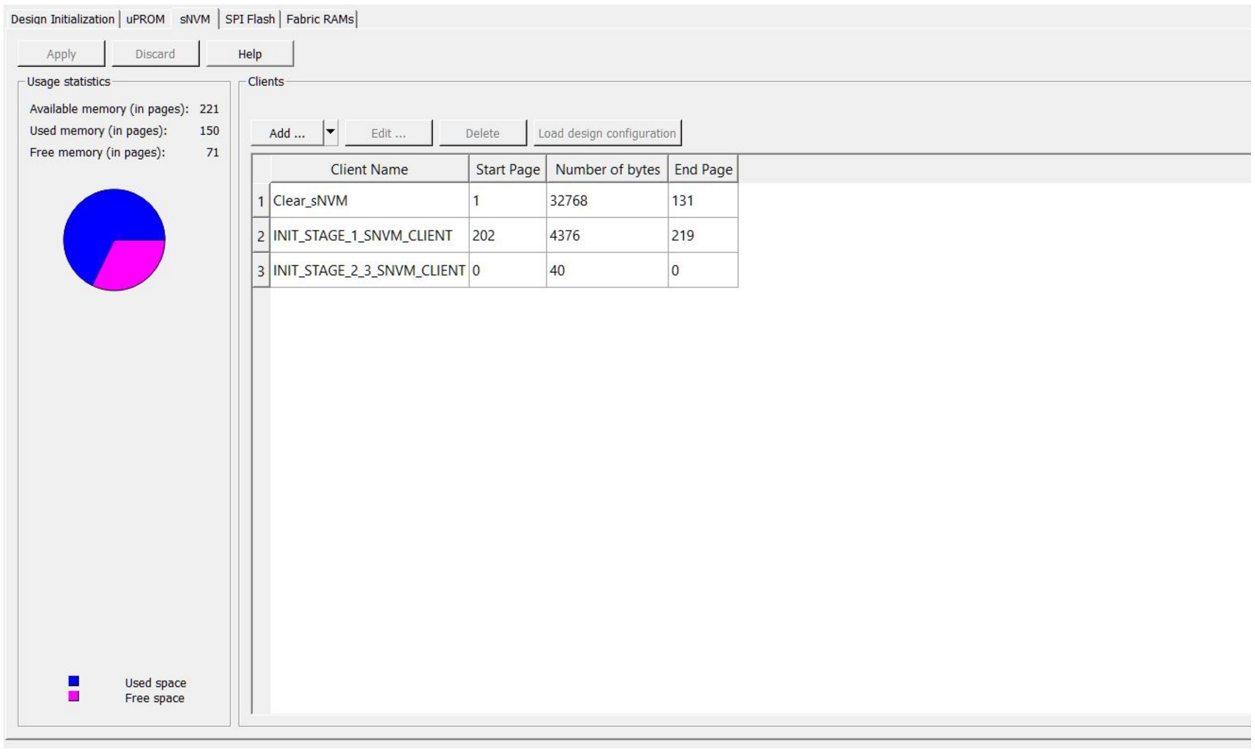
7.6 Update sNVM Memory Content

In Libero Design Flow, after Place-and-Route you can modify the sNVM clients created using `PF_SYSTEM_SERVICES` configurator or add new sNVM clients. This is accomplished using the sNVM tab in Configure Design Initialization Data and Memories tool. After you have made the updates to the sNVM clients, there is no need to rerun Place-and-Route.

7.7 Configure Design and Memory Design Initialization Data and Memories

Use this tab to modify and update sNVM clients.

Figure 7-4. sNVM Tab



Available memory (in pages):	221
Used memory (in pages):	150
Free memory (in pages):	71

	Client Name	Start Page	Number of bytes	End Page
1	Clear_sNVM	1	32768	131
2	INIT_STAGE_1_SNVM_CLIENT	202	4376	219
3	INIT_STAGE_2_3_SNVM_CLIENT	0	40	0

8. System Service Return Status Codes

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

Table 8-1. PolarFire FPGA System Services Status Code

Category	System Service Name	Command Value in Hexadecimal	Response Status
Device and Design Information Services	Serial Number Service	0x0	0: Success
	USERCODE Service	0x1	0: Success
	Design Information Service	0x2	0: Success
	Design Certificate Service	0x3	0: Success—Certificate is valid and consistent with device. 1: Device mismatch—Public key or FSN does not match with a device. 2: Signature invalid—Certificate signature is invalid. 3: System error—PUF or storage failure.
	Read Digests Service	0x4	0: Success
	Query Security Service	0x5	0: Success
	Read Debug Information Service	0x6	0: Success
Design Programming Services	Bitstream Authentication Service	0x23	—
	IAP Image Authentication Service	0x22	—
Data Security Services	Digital Signature Service	0x19 RAW format 0x1A DER format	0: Success. 1: FEK failure—Error retrieving FEK. 2: DRBG error—Failed to generate nonce. 3: ECDSA error—ECDSA failed.

...........continued

Category	System Service Name	Command Value in Hexadecimal	Response Status
Secure NVM (SNVM) Functions	Secure NVM Write Service	0x10 Non-authenticated plain text format 0x11H Authenticated plain text format 0x12 Authenticated ciper text format	0: Success. 1: Invalid SNVMADDR—Illegal page address. 2: Write failure—PNVM program/ verify failed. 3: System error—PUF or storage failure. 4: Write Not Permitted—ROMFLAG is set.
	Secure NVM Read Service	0x18	0: Success. 1: Invalid SNVMADDR—Illegal page address. 2: Authentication failure—Page blank, storage corrupt or incorrect USK. 3: System error—PUF or storage failure.
	PUF Emulation Service	0x20	—
	Nonce Service	0x21	—
Fabric Services	Digest Check Service	0x47	—
	In-application Programming Service	0x42 IAP program by index 0x44 IAP verify by index 0x43 IAP program by SPIADDR 0x45 IAP verify by SPIADDR	—
	Auto Update Service	0x46	—

9. System Services Error Codes

The following table lists the System Services Error Codes.

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

...........continued

Error Code	Description	Additional Comments
26	Selected image was invalid and no recovery was performed due to valid design in device.	Error might occur only when the bit stream 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 might occur only when bit stream is executed through Auto Update or IAP mode.
127	Abort	Non-bit stream instruction is executed during bit stream loading.
128	NVMVERIFY	Fabric or security segment verification failed.
129	PROTECTED	Device security has prevented the modifications of nonvolatile memory.
130	NOTENA	Programming mode not enabled.
131	PNVMVERIFY	pNVM verify operation failed 132 SYSTEM System hardware error (PUF or DRBG).
132	SYSTEM	System hardware error (PUF or DRBG).
133	BADCOMPONENT	An internal error is detected in a bit stream component payload.
134	HVPROGERR	Failure in programming subsystem.
135	HVSTATE	Error in the programming subsystem.

10. Revision History

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

Revision	Date	Description
B	12/2021	• Renamed document to PolarFire System Services SgCore User Guide • Updated Introduction chapter. • Added 1.1. Core Description chapter. • Updated 1.4.2.1. Secure NVM Write Service and 1.4.2.2. Secure NVM Read Service. • Added 2. Interface chapter. • Added 3. Timing Diagrams chapter. • Added 4. Tool Flow chapter. • Added 5. Register Map and Descriptions chapter. • Added 6. System Integration chapter. • Added 7. sNVM Configuration chapter. • Added 9. System Services Error Codes chapter. • Updated all figures for quality enhancements.
A	08/2021	Initial Revision.

Microchip FPGA Support

Microchip FPGA products group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, and worldwide sales offices. Customers are suggested to visit Microchip online resources prior to contacting support as it is very likely that their queries have been already answered.

Contact Technical Support Center through the website at www.microchip.com/support. Mention the FPGA Device Part number, select appropriate case category, and upload design files while creating a technical support case.

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

- From North America, call **800.262.1060**
- From the rest of the world, call **650.318.4460**
- Fax, from anywhere in the world, **650.318.8044**

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.

Legal Notice

Trademarks

ISBN: 978-1-5224-9387-7

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4485-5910
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-72400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-72884388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820