USER'S GUIDE

Apollo5 Family Secure Update

Ultra-Low Power Apollo SoC Family

A-SOCAP5-UGGA02EN v1.0

Legal Information and Disclaimers

AMBIQ MICRO INTENDS FOR THE CONTENT CONTAINED IN THE DOCUMENT TO BE ACCURATE AND RELIABLE. THIS CONTENT MAY, HOWEVER, CONTAIN TECHNICAL INACCURACIES, TYPOGRAPHICAL ERRORS OR OTHER MISTAKES. AMBIQ MICRO MAY MAKE CORRECTIONS OR OTHER CHANGES TO THIS CONTENT AT ANY TIME. AMBIQ MICRO AND ITS SUPPLIERS RESERVE THE RIGHT TO MAKE CORRECTIONS, MODIFICATIONS, ENHANCEMENTS, IMPROVEMENTS AND OTHER CHANGES TO ITS PRODUCTS, PROGRAMS AND SERVICES AT ANY TIME OR TO DISCONTINUE ANY PRODUCTS, PROGRAMS, OR SERVICES WITHOUT NOTICE.

THE CONTENT IN THIS DOCUMENT IS PROVIDED "AS IS". AMBIQ MICRO AND ITS RESPECTIVE SUPPLIERS MAKE NO REPRESENTATIONS ABOUT THE SUITABILITY OF THIS CONTENT FOR ANY PURPOSE AND DISCLAIM ALL WARRANTIES AND CONDITIONS WITH REGARD TO THIS CONTENT, INCLUDING BUT NOT LIMITED TO, ALL IMPLIED WARRANTIES AND CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHT.

AMBIQ MICRO DOES NOT WARRANT OR REPRESENT THAT ANY LICENSE, EITHER EXPRESS OR IMPLIED, IS GRANTED UNDER ANY PATENT RIGHT, COPYRIGHT, MASK WORK RIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT OF AMBIQ MICRO COVERING OR RELATING TO THIS CONTENT OR ANY COMBINATION, MACHINE, OR PROCESS TO WHICH THIS CONTENT RELATE OR WITH WHICH THIS CONTENT MAY BE USED.

USE OF THE INFORMATION IN THIS DOCUMENT MAY REQUIRE A LICENSE FROM A THIRD PARTY UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF THAT THIRD PARTY, OR A LICENSE FROM AMBIQ MICRO UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF AMBIQ MICRO.

INFORMATION IN THIS DOCUMENT IS PROVIDED SOLELY TO ENABLE SYSTEM AND SOFTWARE IMPLEMENTERS TO USE AMBIQ MICRO PRODUCTS. THERE ARE NO EXPRESS OR IMPLIED COPYRIGHT LICENSES GRANTED HEREUNDER TO DESIGN OR FABRICATE ANY INTEGRATED CIRCUITS OR INTEGRATED CIRCUITS BASED ON THE INFORMATION IN THIS DOCUMENT. AMBIQ MICRO RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN. AMBIQ MICRO MAKES NO WARRANTY, REPRESENTATION OR GUARANTEE REGARDING THE SUITABILITY OF ITS PRODUCTS FOR ANY PARTICULAR PURPOSE, NOR DOES AMBIQ MICRO ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT, AND SPECIFICALLY DISCLAIMS ANY AND ALL LIABILITY, INCLUDING WITHOUT LIMITATION CONSEQUENTIAL OR INCIDENTAL DAMAGES. "TYPICAL" PARAMETERS WHICH MAY BE PROVIDED IN AMBIQ MICRO DATA SHEETS AND/OR SPECIFICATIONS CAN AND DO VARY IN DIFFERENT APPLICATIONS AND ACTUAL PERFORMANCE MAY VARY OVER TIME. ALL OPERATING PARAMETERS, INCLUDING "TYPICALS" MUST BE VALIDATED FOR EACH CUSTOMER APPLICATION BY CUSTOMER'S TECHNICAL EXPERTS. AMBIQ MICRO DOES NOT CONVEY ANY LICENSE UNDER NEITHER ITS PATENT RIGHTS NOR THE RIGHTS OF OTHERS. AMBIQ MICRO PRODUCTS ARE NOT DESIGNED, INTENDED, OR AUTHORIZED FOR USE AS COMPONENTS IN SYSTEMS INTENDED FOR SURGICAL IMPLANT INTO THE BODY, OR OTHER APPLICATIONS INTENDED TO SUPPORT OR SUSTAIN LIFE, OR FOR ANY OTHER APPLICATION IN WHICH THE FAILURE OF THE AMBIQ MICRO PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR. SHOULD BUYER PURCHASE OR USE AMBIQ MICRO PRODUCTS FOR ANY SUCH UNINTENDED OR UNAUTHORIZED APPLICATION, BUYER SHALL INDEMNIFY AND HOLD AMBIQ MICRO AND ITS OFFICERS, EMPLOYEES, SUBSIDIARIES, AFFILIATES, AND DISTRIBUTORS HARMLESS AGAINST ALL CLAIMS, COSTS, DAMAGES, AND EXPENSES, AND REASONABLE ATTORNEY FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PERSONAL INJURY OR DEATH ASSOCIATED WITH SUCH UNINTENDED OR UNAUTHORIZED USE, EVEN IF SUCH CLAIM ALLEGES THAT AMBIQ MICRO WAS NEGLIGENT REGARDING THE DESIGN OR MANUFACTURE OF THE PART.

Revision History

Rev	Date	Description
1.0	May 2025	Initial Release

Reference Documents

Document ID	Description
A-SOCAP5-UGGA04EN	Apollo5 Security User's Guide
A-SOCAP5-UGGA03EN	Apollo5 Family OEM Provisioning Tools User's Guide
A-SOCAP5-UGGA01EN	Apollo5 Family MRAM Recovery User's Guide
A-SOCAP5-QSGA01EN	Apollo5 MRAM Recovery Quick Start Guide
A-SOCAP5-QSGA02EN	Apollo5 MRAM Recovery UART Quick Start Guide

Table of Contents

1. Introduction 8

1.1 Terminology 8

2. Security Features of Secure Image Update 9

2.1 Integrity Check 9
2.2 Authentication 9
2.3 Decryption 9
2.4 Anti-Rollback 10
2.5 Image Protection 10

3. Image Formats 11

3.1 Common Field Definitions 13
3.2 Optional Field Definitions 14
3.2.1 Firmware Update 14
3.2.2 INFO0 Update 15
3.2.3 Wired Update (Download) 15
3.2.4 Certificate Chain Update 16
3.2.5 OEM Key Revocation Update 17

4. Secure Image Upgrade Flow 18

4.1 Image Download 18
4.2 OTA Descriptor and OTA Pointer 18
4.3 Reset 20
4.4 Upgrade Verification 20
4.5 Installation 20
4.6 Feedback 20

5. Wired Update Flow 22

5.1 Wired Update Messages 26
5.1.1 Acknowledgment (ACK) Message 26
5.1.2 Connection Establishment (HELLO and STATUS) Messages 27
5.1.3 Secure Upgrade (OTADESC) Message 27
5.1.4 Wired Download (UPDATE and DATA) Messages 28
5.1.5 Termination (ABORT) Message 29
5.1.6 Reboot (RESET) Message 29

6. OEM Secondary Bootloader 30

6.1 Device Programming Considerations for Secondary Bootloader 30

6.2 Secondary Bootloader Implementation Considerations 31

6.2.1 Asset Protections 31

6.2.2 Debugger Support 31

7. OEM MRAM Recovery Image 32

List of Tables

Table 1-1 Terminology 8
Table 3-1 Common Field Definitions 13
Table 3-2 Field Definitions for Firmware Updates 14
Table 3-3 Field Definitions for INFO0 Update 15
Table 3-4 Field Definitions for Wired Update 16
Table 3-5 Field Definitions for Certificate Chain Update 16

List of Figures

Figure 3-1 OTA Image Generic Format with Signature and Encryption Info 12
Figure 3-2 OTA Image Blob with Encryption Info 12
Figure 3-3 Optional Field Definitions for Firmware Updates (Secure or Nonsecure) 14
Figure 3-4 OTA Image Blob with a Content Certificate 15
Figure 3-5 Certificate Chain Update 15
Figure 3-6 Wired Update 15
Figure 3-7 Certificate Chain Update 16
Figure 3-8 OTA Image Blob for Certificate Chain Refresh 16
Figure 3-9 Key Revocation Update 17
Figure 4-1 OTA Descriptor and OTA Pointer 19
Figure 5-1 Message Exchange Using the Configured Security Polices of Device 23
Figure 5-2 Process for Update/Recovery Using IOS SPI Interface 25
Figure 5-3 Wired Update Messages 26
Figure 5-4 Acknowledgment (ACK) Message 26
Figure 5-5 Connection Establishment (HELLO and STATUS) Messages 27
Figure 5-6 Secure Upgrade (OTADESC) Message 28
Figure 5-7 Wired Download (UPDATE and DATA) Messages 28
Figure 5-8 Termination (ABORT) Message 29
Figure 5-9 Reboot (RESET) Message 29
Figure 7-1 OEM MRAM Recovery Image 33

SECTION

1

Introduction

This document describes the methods, image formats, and protocols supported by the Apollo5 Secure Bootloader (SBL) for both secure image and wired updates.

1.1 Terminology

This section defines some of the terminologies used in this document.

Table 1-1: Terminology

Abbreviation	Definition
KEK	Key Encryption Key
PK	Public Key
OTA	The name of the image format used for delivering updated images processed by the SBL, which may be received Over-the-Air.
OTP	One Time Programmable
RSA	A public-key cryptosystem that is widely used for secure data transmission.
SBL	Secure Bootloader – Installed by Ambiq at the factory, that facilitates updating of OEM assets, and the SBL itself, within the configured security policies

SECTION

2

Security Features of Secure Image Update

The following sections describe the specific security features of the Image update flow using Secure Bootloader (SBL).

2.1 Integrity Check

Integrity Check using CRC32 (Ethernet FCS) function.

2.2 Authentication

Authentication uses RSA 3072 signature verification on the Signature field using the PK from the existing certificate chain.

The **Auth Key** Index field (0-2) is specified to determine which PK is used based on the 3-stage pre-installed Certificate chain.

Since the authentication relies on pre-existing certificate chain, this feature is only supported if Secure Boot is enabled.

2.3 Decryption

Decryption is performed using AES-128 CTR and the Encryption Info field. The Encryption Info comprises of the Nonce/IV and the Encrypted Key. The Encrypted key is decrypted using the provisioned Key Encryption Key (KEK) on the device, and is thereafter used to decrypt the ciphertext in conjunction with the supplied Nonce/IV.

KEK Index determines which of the preinstalled keys is used to decrypt the Encrypted key. Values 0 and 1 correspond to the hardware keys, KCP and KCE

respectively. Indices 0x80 onwards signify keys in the Key bank area, with each 128b key corresponding to one index.

2.4 Anti-Rollback

When in Secure Mode, the SBL uses a combination of the **INFOC_HBK1MINVER*** words and the swVer in the OEM Content Certificate to provide rollback prevention. This will prevent an OTA from upgrading to an OEM image using a certificate with a version that is lower than that specified in the current certificate chain. All the Certificates in a chain should all have the same version.

Hence, to implement the anti-rollback, the full certificate chain needs to be refreshed with an updated version number, using a **Certificate Chain Update** operation.

2.5 Image Protection

The SBL can be instructed to apply image protection features to assets in the MRAM. The MRAM blocks for protection are specified in 16K granularity.

The blocks can be marked as *Write-Protected* to prevent overwriting the images either intentionally using malicious programs, or unintentionally. Such Write-protected images can be upgraded only through SBL by maintaining the secure upgrade trust chain.

The blocks can be marked as *Copy-Protected* to prevent Read access to the pages. This can be used to avoid exposing sensitive algorithms or programs from prying eyes. Care must be taken when using this feature to protect executable code—to generate the code using appropriate tool options so that there are no data reads in the protected code memory.

These protection attributes are specified in **INFOC (INFOC_SBL_WPROT0-7**, and **INFOC_SBL_RPROT0-7)**, as part of provisioning, and hence are statically predefined.

The SBL also supports dynamic MRAM protection through the content certificate. On successful installation/verification, the OEM Content Certificate can also be used to direct the SBL to apply specified protections for each image it refers to.

SECTION

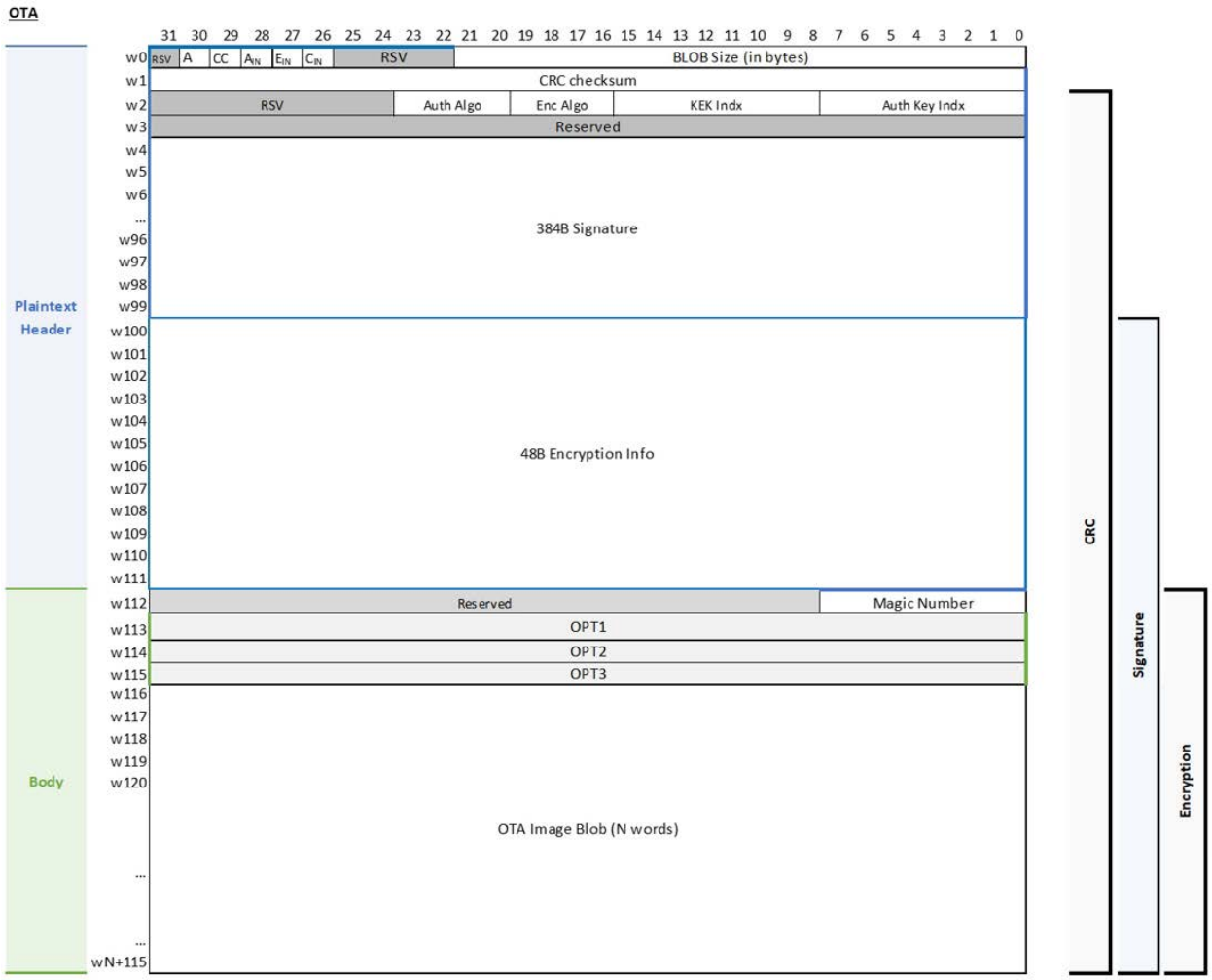
3

Image Formats

This section describes the generic format for all the OTA images:

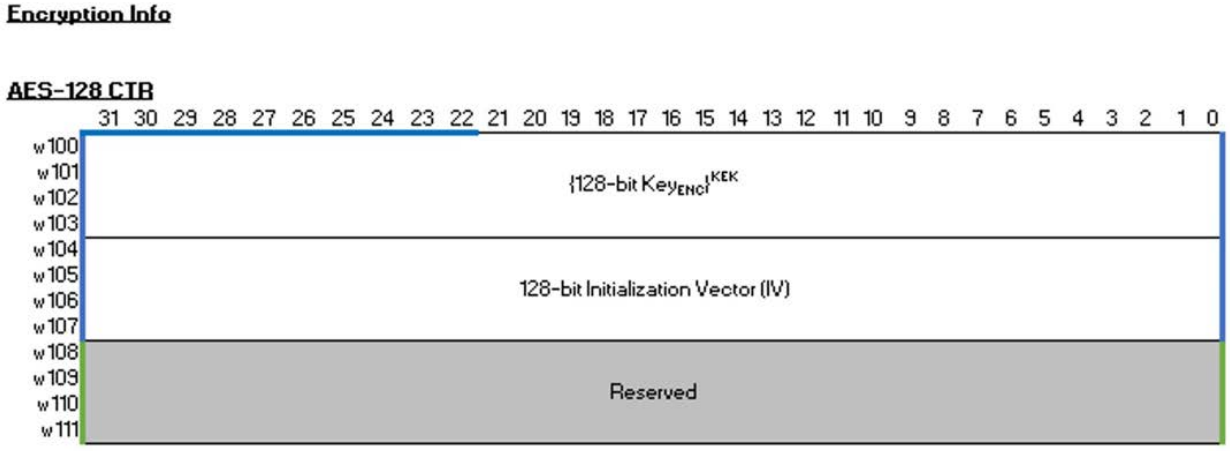
- This is the format for both Secure and Non-Secure OTA Updates, as well as non-firmware updates.
  - The image format supports error detection, authentication, and decryption.
- The **Signature** field is optional, based on value of $\mathbf{A_{IN}}$ (Value of 0 implies no Authentication, and hence no signature).
- The **Encryption Info** field is optionally present, if the $\mathbf{E_{IN}}$ is non-zero.
- The image format and the offset of the fields will change if the optional Signature and/or **Encryption Info** is not present. The generic format in Figure 3-1 and Figure 3-2 on page 12 depicts the case when both are present.

Figure 3-1: OTA Image Generic Format with Signature and Encryption Info



If present, the **Encryption Info** field is formatted as shown in Figure 3-2.

Figure 3-2: OTA Image Blob with Encryption Info



3.1 Common Field Definitions

Table 3-1: Common Field Definitions

Field	Definitions
Magic Number	Unique 8-bit value used to indicate the image type. 0xC0 = SECURE FIRMWARE UPDATE 0xCC = OEM CERT CHAIN UPDATE 0xCB = NONSECURE FIRMWARE 0xCE = OEM KEY REVOKE 0xCD = FIRMWARE DOWNLOAD 0xCF = INFO0 UPDATE
BLOB Size	Size of the image blob body (in bytes)
A	Ambiq Owned (Determines the restrictions on the magic#, and the keys for Authentication/Encryption)
CRC Checksum	CRC checksum signature for the image. CRC is computed over the CRC region marked on the decrypted blob.
CC	Contains embedded Content Cert
A_{IN}	Install-Authenticate enable bit - If '1', the image must be authenticated on installation.
C_{IN}	Install-CRC enable bit - If '1', the image must be CRC verified on installation.
E_{IN}	Install-Decryption enable bit - If '1', the image must be decrypted on installation.
KEK Indx	Key-Encryption-Key Index. This index specifies which KEK should be used within the KEK bank for all key unwrap functions, the MSB indicates if it is a predefined key, or from key bank. Key bank keys are indexed in 128b increment.
Enc Algo	Encryption Algorithm - specifies which algorithm is to be used for decrypting the image. 0: None 1: AES-128 CTR others: not supported
Auth Key Indx	Authentication Key Index. This index specifies which authentication key should be used within the Auth Key bank for authentication of this image. Only three values are supported right now, which will be the PK of the existing Cert chain.
Auth Algo	Authentication Algorithm. Specifies which algorithm is to be used for authenticating the image. 0: None 1: RSA3072 others: not supported

Table 3-1: Common Field Definitions

Field	Definitions
Signature	384B RSA3072 signature of the Install Blob following.
128-bit IV	128-bit Initialization Vector used for seeding the encryption/ decryption of the image.
{128-bit KeyENC}^{KKK}	128-bit KEK wrapped encryption key
OPT*	Optional parameters (depends on Magic number)

3.2 Optional Field Definitions

Optional fields are interpreted based on the type of update (magic#).

3.2.1 Firmware Update

Figure 3-3: Optional Field Definitions for Firmware Updates (Secure or Nonsecure)

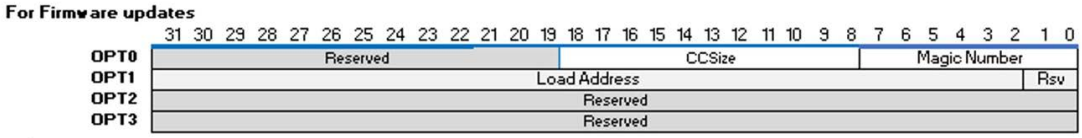


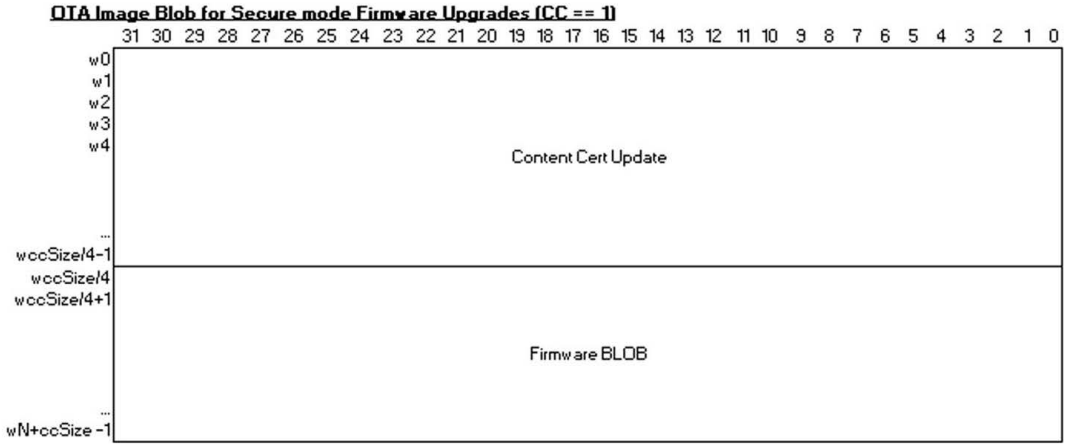
Table 3-2: Field Definitions for Firmware Updates

Field	Definitions
CCSize	Size of the bundled Content Certificate (only relevant if CC is set in W0 for SECURE FIRMWARE update)
Load Address	msb of Address to which the image should be installed.

In the non-secure mode, the embedded OTA Blob inside contains the raw image.

For Secure Mode, the OTA Blob is shown in Figure 3-4 on page 15 as the updated content certificate is also bundled:

Figure 3-4: OTA Image Blob with a Content Certificate



3.2.2 INFO0 Update

Figure 3-5: Certificate Chain Update

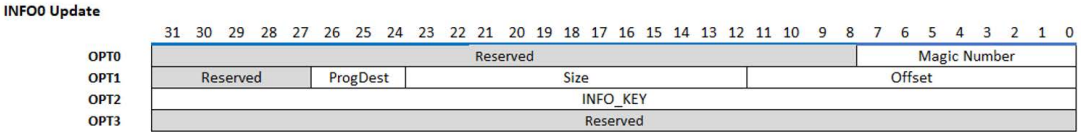


Table 3-3: Field Definitions for INFO0 Update

Field	Definitions
INFO_KEY	32-b key required for programming the INFO0
ProgDest	0 = Current Active, 1 = OTP, 2 = MRAM, 3= Both
Size	The size of the update
Offfset	The offset in the Infospace where the update needs to happen.

3.2.3 Wired Update (Download)

Figure 3-6: Wired Update

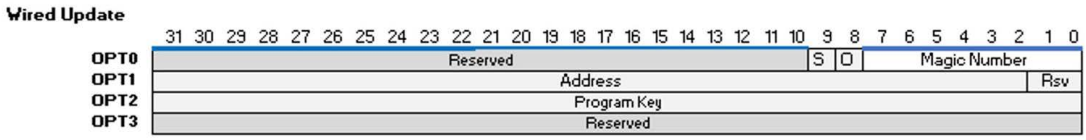


Table 3-4: Field Definitions for Wired Update

Field	Definitions
O	Implying an OTA process request
S	Implying an SBL OTA
Program Key	32-bit key required for programming the MRAM (PROG_KEY)
Address	msb of Address in flash to load the image to

The embedded OTA blob contains the raw download image.

3.2.4 Certificate Chain Update

Figure 3-7: Certificate Chain Update

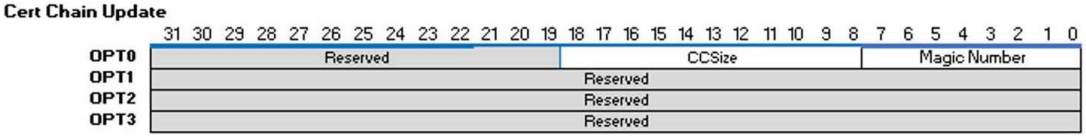
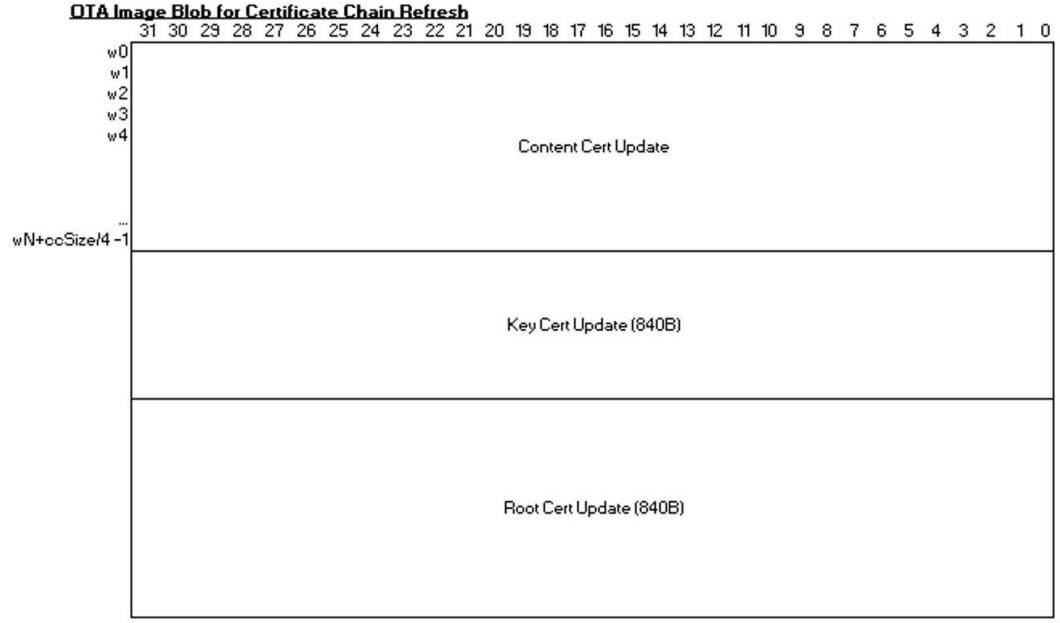


Table 3-5: Field Definitions for Certificate Chain Update

Field	Definitions
CCSize	Size of the bundled Content Certificate

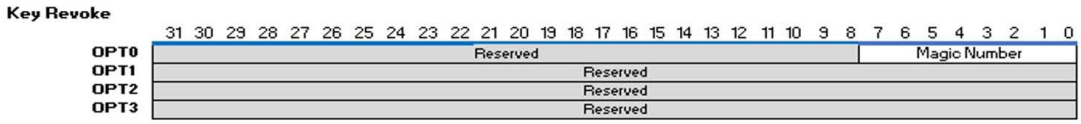
The embedded OTA blob contains the full certificate chain, as shown in Figure 3-8.

Figure 3-8: OTA Image Blob for Certificate Chain Refresh



3.2.5 OEM Key Revocation Update

Figure 3-9: Key Revocation Update



The Image blob section of the OTA image contains 2 words which are a 64b bit-mask of the key(s) to revoke, with each bit corresponding to a 128b key in the Customer's Keybank, (LSB = KEY0). A value of 1 indicates a request to revoke the corresponding key 0.

SECTION

4

Secure Image Upgrade Flow

The Apollo5 SoC supports a secure in-field image upgrade flow using the Secure Bootloader (SBL). The Secure Bootloader has access to the key storage in the INFOC-OTP memory and can be used to optionally decrypt, authenticate and validate the upgrade images before installing them. When configured (based on the security policy configurations in INFOC-OTP), only properly signed and protected images will be allowed for updates.

The Secure Bootloader can be used to securely update itself, the customer application image, a customer's secondary bootloader, or other assets. The following sections lists out steps in the secure in-field image update flow. The same update flow is also used for accomplishing updates for INFO0, the Certificate Chain, as well as to revoke one or more keybank keys.

4.1 Image Download

The in-field upgrade process is initiated by a user application downloading an image blob to the MRAM. This part of the upgrade process is specific to individual deployment scenarios and the implementation is left to the customers. The Secure Upgrade framework does not mandate any specifics for this process. Depending on the deployment model, the image download could happen over traditional wired interfaces (e.g., SPI, or UART), or wirelessly OTA (Over the Air) using BLE, or other wireless protocols. The Upgrade application running on the Apollo5 SoC and its counterpart on the host/cloud side could implement their own protocol to ensure integrity, secrecy, and authenticity of the image blob itself.

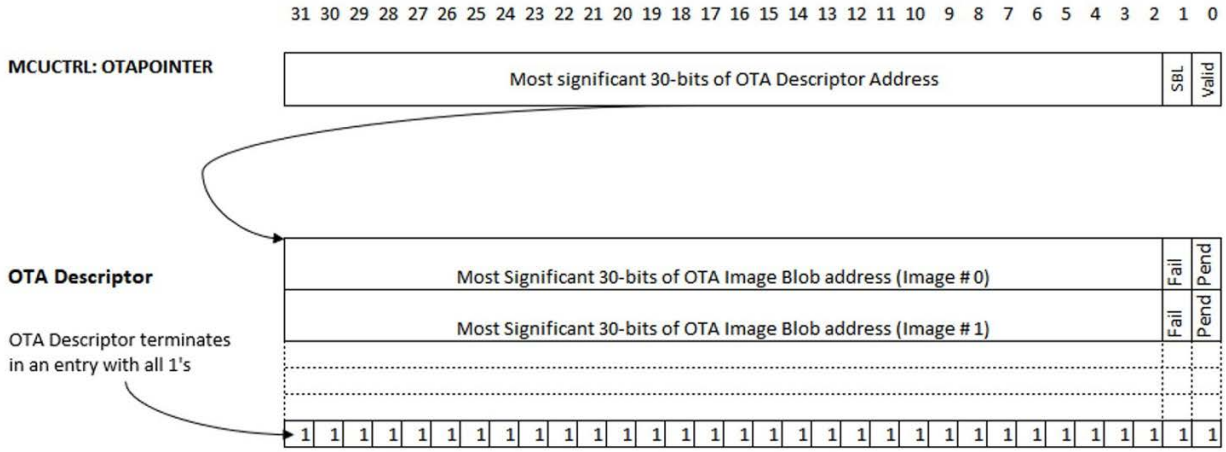
4.2 OTA Descriptor and OTA Pointer

The customer application builds an OTA Descriptor containing the information about the upgrade image blob(s) corresponding to the update requests.

OTA Descriptor needs to be built in MRAM. The framework supports multiple updates in same operation. The OTA Descriptor consists of a list of upgrade image blobs, with a special End of List Marker. This allows the upgrade application to update more than one images in one boot cycle. For example, the upgrade application could construct a single OTA Descriptor to upgrade main firmware as well as certain third-party library images located separately in the MRAM. The OTA Descriptor can support up to eight images[1].

The download application communicates the location of the OTA Descriptor information with the Secure Bootloader by initializing a special register, **MCUCTRL:OTAPOINTER**.

Figure 4-1: OTA Descriptor and OTA Pointer



OTAPOINTER will be initialized as follows:

- Most significant 30-bits will correspond to most significant 30-bits of OTA Descriptor.
- Least Significant bit (bit 0) should be initialized to 1 to indicate a valid OTA Descriptor.
- Bit 1 should be initialized to 1 if the list contains an SBL OTA.

The **OTA Descriptor** contains a list of entries (up to 8), each corresponding to an OTA blob, with the list terminating in 0xFFFFFFFF. Each list entry word are comprised of following:

- Most significant 30-bits will correspond to most significant 30-bits of OTA blob pointer.
- Least Significant 2-bits should be initialized to '11' to indicate a valid OTA Pending.

Note that the OTA Image Blobs needs to be aligned at a 16-byte boundary to ensure that the encryption portion falls at 128b alignment (block size).

[1] Note that when using multiple SECURE FIRMWARE updates together, the embedded content certificate of each image should reflect the updated content at each respective stage

4.3 Reset

After accepting the required updates and programming them in MRAM, the download application constructs the OTA Descriptor List and initializes the **MCUCTRL:OTAPOINTER** register accordingly. The actual Update is then initiated on the next Reset, which kicks in the bootloader.

4.4 Upgrade Verification

As part of boot process, the SBL inspects **MCUCTRL:OTAPOINTER** for any updates to be processed. If present, each update blob is processed as per the configured security policy.

- The security policy can be configured (via INFOC-OTP memory) to mandate Authentication to ensure only properly signed images will be accepted.
- The Secure Bootloader also supports encrypted image blobs, which also can also be mandated by the security policy.

The SBL enforces the configured security policy and validates the image blobs against the security assets in INFOC-OTP memory, and/or pre-installed certificate chain in the MRAM. After the optional decryption and authentication, if the image is found to be good, the SBL then proceeds with installation of the image. In the case of secure firmware upgrade, SBL also verifies the bundled content certificate update, and installs the new content certificate along with the new image, only if the content certificate checks out (which implies all the images in the content certificate need to match up, if there are more than one).

4.5 Installation

A validated OTA image is installed to its designated place by the Secure Bootloader.

4.6 Feedback

The OTA flow also allows for feedback to the user application using the same OTA Descriptor to communicate the OTA status of individual images, back to the initiating application. This is accomplished using the same OTA Descriptor.

- General OTA descriptor errors are communicated back to the application using the high order bits in the **MCUCTRL:OTAPOINTER** register. Details can be found in **am_hal_secure_ota.h**.
- After the Secure Bootloader processes an OTA, it clears the least significant bit (bit 0).
  - Bit 1 indicates the status of the OTA: 0 for Success, 1 for Failure

- Some of the higher order bits of the Upgrade Descriptor are used to return failure cause. Details can be found in: **am_hal_secure_ota.h.**

– If Bit 0 is still set to 1 (Pending) after the OTA, this implies some other error (e.g., invalid or improperly formed OTA descriptor list) caused SBL to not process this OTA. The OTA can be retried after clearing the issue.

SECTION

5

Wired Update Flow

The Apollo5's SBL provides support for an external host to connect during boot process to upgrade images, or to recover a failing device (see note below). Essentially, the SBL can proxy as the "download application" for the Update process. The SBL enters Application Recovery if any of the boot time validations fail, or if there is no valid image to boot to. In addition, there is an Boot Override feature (configurable through INFOC-OTP), which allows a forced image upgrade that is initiated using a specific (configurable) GPIO during the boot up.

> **NOTE:** The SBL supports and will initiate updates via "Boot Override" and "Application Recovery" which are provided by the SBL and utilized for updating or recovering an OEM application/assets if they were erased. "MRAM Recovery" is a separate process that can be configured where the Apollo5 device will automatically recover the factory installed SBL and OEM's recovery image if the images were corrupted (e.g., Magnetic Corruption). More information about MRAM Recovery is contained in *Apollo5 Family MRAM Recovery Guide User's Guide A-SOCAP5-UGGA01EN*.

The SBL uses the INFOC-OTP configuration to determine which interface(s) to be used (see **INFOC_WIRED_CONFIG**). When enabled, it first looks for UART connection, then the IOS SPI connection. The UART has a default configuration (UART0, 115200 Baud, TX pin 30, RX pin 55) which can be reconfigured using with INFO0 fields **INFO0_SECURITY_WIRED_IFC_CFG0** to **…5**.

The UART interface has a default timeout of 500ms, which can be reconfigured using the INFO0 field **INFO0_WIRED_TIMEOUT** that sets the time the SBL waits for the initial HELLO packet from the Host. The IOS-SPI interface uses the Slave Interrupt pin (see **INFOC_WIRED_CONFIG: SLVINTPIN**), to indicate to the host that it is ready to receive packets. In this case, it waits for a fixed timeout of 500 msec before exiting the update process.

In all cases, an external host needs to use a predefined messaging protocol to instruct the SBL to upgrade assets on the device. Figure 5-1 on page 23 illustrates a high-level message exchange to initiate an "upgrade" using configured security policies of the device when using

the UART. The images are sent in a series of **DATA** packets that can be up to 8KB in size. The process starts with a **HELLO/ STATUS** exchange within the timeout period. It ends with the **RESET/ ACK** exchange which instructs the SBL what type of reset to perform a SWPOR or SWPOI reset (or none).

Figure 5-1: Message Exchange Using the Configured Security Polices of Device

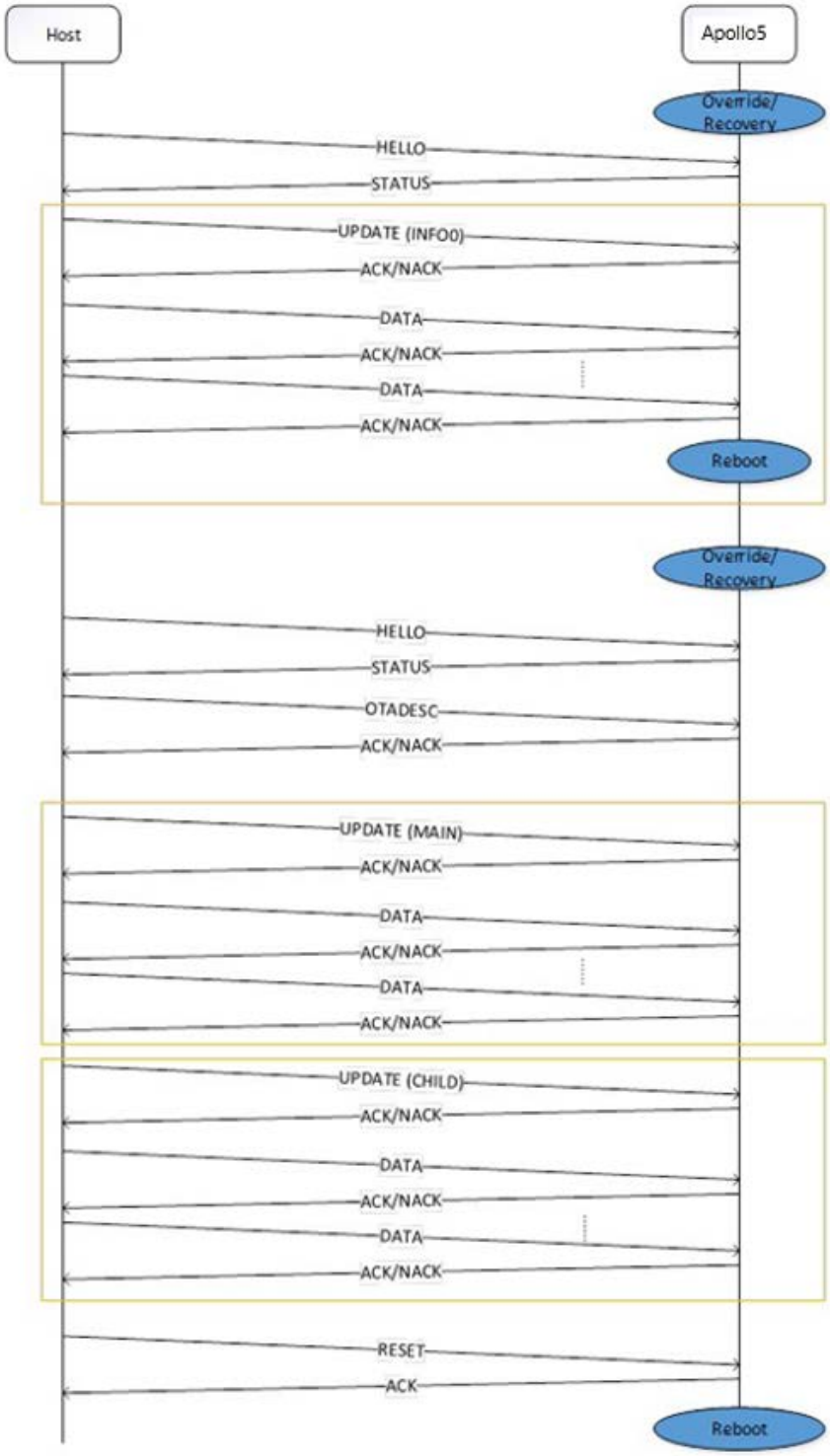


Figure 5-2 on page 25 shows the process for updates when using the IOS SPI interface. The process starts with the Host resetting the Apollo5 device using nRST signal and the device entering Boot Override (because of the configured GPIO level) or Application Recovery because no Application was found or it validation failed (in Secureboot), then the SBL raises the Slave Interrupt pin to indicate it is ready to accept packets over the IOS-SPI interface. All packets from the Host to the SBL are limited to 120 bytes due to the maximum LRAM memory size. This means the packets message packets described previously will be further disassembled for transmission over the IOS-SPI, and then reassembled by the SBL back into the message packets before processing them. Disassembly is done by adding a 32-bit header to each packet as follows:

```
typedef struct
{
      uint32_t length : 16;
      uint32_t resv : 14;
      uint32_t bEnd : 1;
      uint32_t bStart : 1;
} am_secboot_ios_pkthdr_t;
```

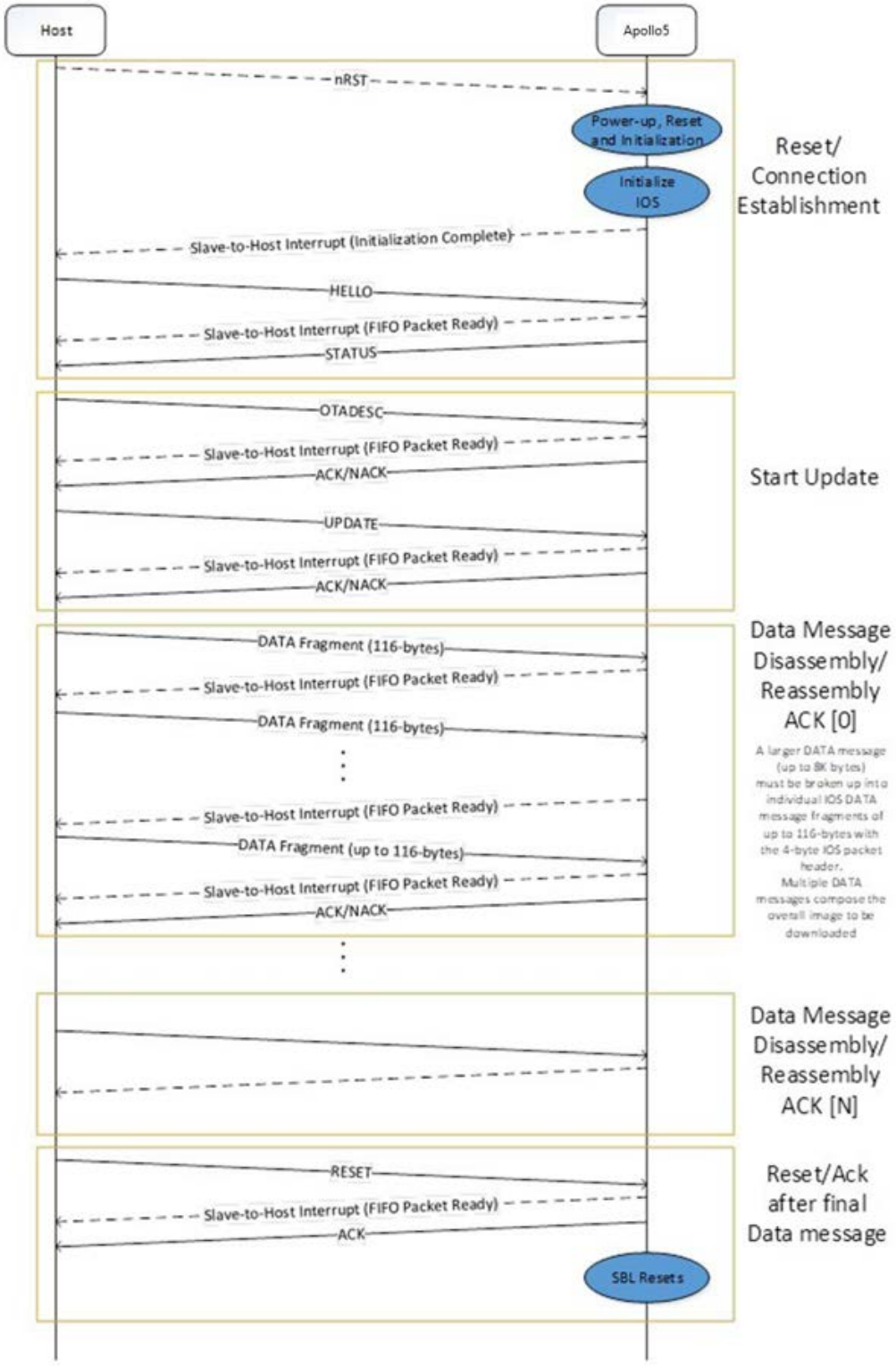
The start and end flags indicate the packets that represent the first and last of an original SBL wired protocol packet. The original packet data follows with up to 116 bytes. The 116-byte Data Fragments compose an entire original DATA message and are reassembled by SBL up to maximum 8KB size of the original packet. Error checking is then done on the reassembled packet using the CRC and message length contained in the original message, or it will be rejected.

Note that the FIFO mode is used for Slave to Host transfers, so disassembly/reassembly is not required in that direction and messages will not exceed 1023 bytes in that direction.

AmbiqSuite provides an example application that uses the ISO interface to do SBL updates. Using a second Apollo5 EVB board running the example **uart_ios_bridge**that works together with the **uart_wired_update.py** script, to provide a bridge from the UART update protocol done by the **uart_wired_update.py** script and will deliver update images via the IOS-SPI interface to the target Apollo5 device. The example can be found at:

ambiqsuite\boards\apollo510_evb\examples\peripherals\uart_ios_bridge

Figure 5-2: Process for Update/Recovery Using IOS SPI Interface

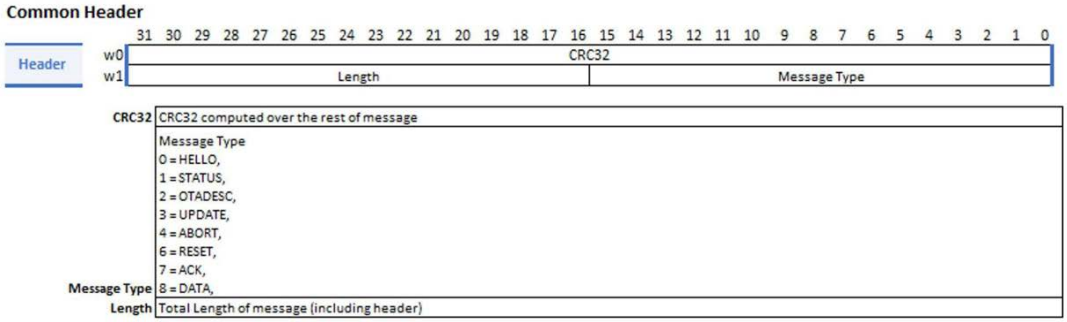


5.1 Wired Update Messages

All the message formats below assume little-endian byte order.

Each message starts with a common header which defines the following message type and length, along with a CRC for error checking.

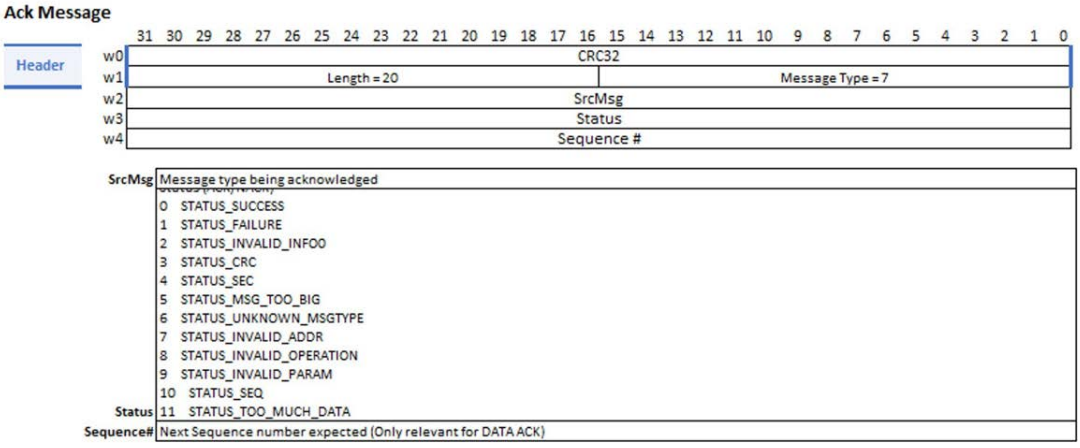
Figure 5-3: Wired Update Messages



5.1.1 Acknowledgment (ACK) Message

The SBL acknowledges most of the messages received using an ACK message (except for HELLO, which is acknowledged using STATUS message). Acknowledgments for DATA messages (described later), also include a sequence number, which can be used to implement a retransmission mechanism at host side if the connection medium is lossy.

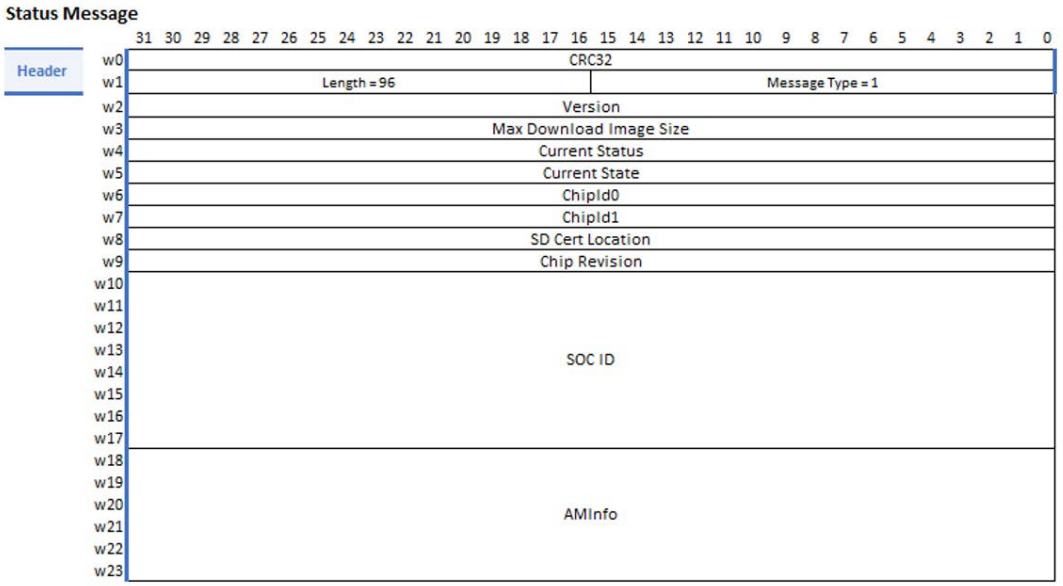
Figure 5-4: Acknowledgment (ACK) Message



5.1.2 Connection Establishment (HELLO and STATUS) Messages

There is an initial handshake between the host and SBL, which can assist in determining the reason why SBL got into the wired update mode. A HELLO message is sent from host, to which the SBL responds with a STATUS message. The STATUS message also provides information about the largest size of the image blob that can be downloaded using SBL, along with other useful information about the device.

Figure 5-5: Connection Establishment (HELLO and STATUS) Messages



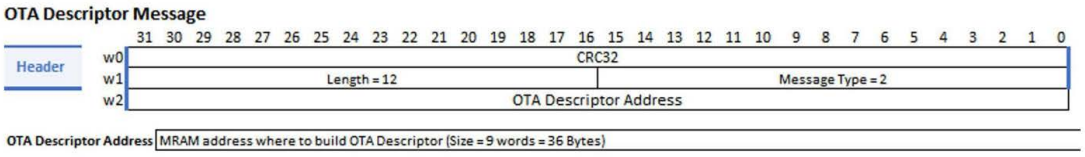
Version	Bootloader Version information
Max Image Size	Maximum size of blob that can be transferred using UPDATE command
Status	Current Boot Status
State	Current Bootloader state
ChipId0	Child Id 0 for the device
ChipId1	Child Id 0 for the device
SD Cert Location	Address at which the Secure Debug Cert needs to be loaded
ChipRev	Chip Revision
SOC ID	256b SOCID for the device (Only valid if in Secure LCS)
AMInfo	Aux information - only for Ambiq usage

5.1.3 Secure Upgrade (OTADESC) Message

Upgrade for the firmware images using one of the wired interfaces provides secure update process for a host to load update images it may have received via its OTA process (e.g. wireless OTA). The SBL just provides a means to do wired download for the image blobs. There are messages which instruct SBL to create an OTA Descriptor, followed by download of one or more image blobs (using UPDATE and DATA messages) that can then be processed on one batch update.

SBL reserves a fixed size in the MRAM starting at the specified address for the purpose of building the OTA Descriptor (to allow for up to 8 update images) before being processed.

Figure 5-6: Secure Upgrade (OTADESC) Message

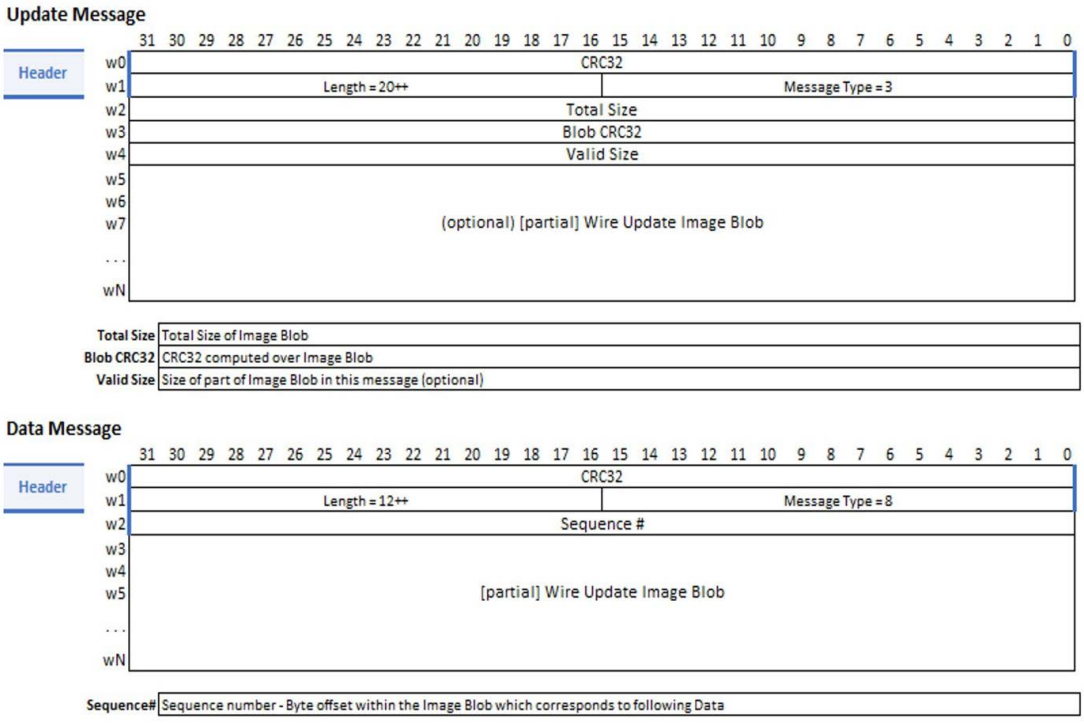


5.1.4 Wired Download (UPDATE and DATA) Messages

Any upgrade on the device is comprised of an UPDATE message, which describes the nature and size of an upgrade. The UPDATE message is then followed by zero or more DATA messages to send the actual image blob. After all the data is received, the SBL verifies the integrity and validates the image(s) using the configured security policy.

To avoid corrupting existing MRAM space with corrupted/unverified downloads, the image blobs are downloaded to SRAM, and written to MRAM only after verification for integrity and authentication. This implies that individual wired downloads are limited in size based on the available SRAM. The SBL only uses DTCM for this purpose, and the amount of available memory for this could be further reduced, if some part of the memory is being reserved by the application (using **INFO0_SECURITY_SRAM_RESV**). Bigger size blobs can still be transferred by splitting them accordingly[2].

Figure 5-7: Wired Download (UPDATE and DATA) Messages

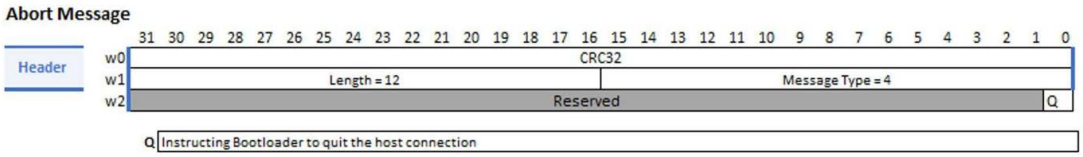


[2] Tools bundled with AmbiqSuite SDK provide means to handle disassembly/reassembly of large size images into multiple OTA images.

5.1.5 Termination (ABORT) Message

A Download in progress can be aborted using ABORT message. Host has a choice to continue the connection, or instruct SBL to quit the connection.

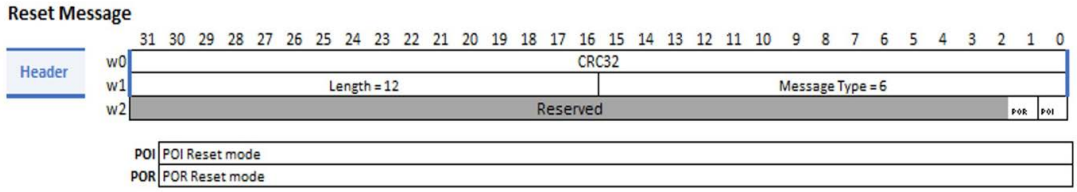
Figure 5-8: Termination (ABORT) Message



5.1.6 Reboot (RESET) Message

The Image blobs downloaded through UPDATE/DATA messages will have 'O' bit set to instruct SBL to schedule an OTA using the downloaded image. Multiple images can be combined together for the download step using this process. Actual Image upgrade is only initiated when a RESET message is received, as part of regular OTA processing by SBL.

Figure 5-9: Reboot (RESET) Message



SECTION

6

OEM Secondary Bootloader

The Apollo5 Secure Boot/Update flow allows the provision to incorporate an OEM secondary bootloader. A secondary bootloader can be used to supplement the core capabilities of the native Apollo5 SBL. Potential reasoning for implementing one can be:

- Need to support different authentication/encryption algorithms
- Need to support external memory device
- Other vendor specific enhancements

For designs incorporating a secondary bootloader, the latter replaces the OEM's main image. SBL treats the secondary bootloader as the main image and verifies/updates the same using native boot/update flow. The secondary bootloader can then implement the additional features before passing control to the OEM's main application image.

6.1 Device Programming Considerations for Secondary Bootloader

There are certain considerations while OEM programming a design with a secondary bootloader.

- Ensure **INFOC_SECUIRTY:PLONEXIT** field is set to 1 during OEM provisioning in DM LCS.
- Ensures the INFOC-OTP key bank will remain accessible to be used for image verification purposes.
- It also allows Secondary bootloader to implement page lockouts for Read/Write protection by clearing additional bits in the **MCUCTRL:FLASHWPROT*** and **MCUCTRL:FLASHRPROT*** words.

6.2 Secondary Bootloader Implementation Considerations

6.2.1 Asset Protections

It is the responsibility of the secondary bootloader code to do the following before transferring to the main image to ensure proper security. It needs to "lock" the INFOC-OTP key bank and restrict further access to flash protection registers, by asserting the protection lock by writing 1 to **PROTLOCK** bit in register **MCUCTRL:BOOTLOADER**.

6.2.2 Debugger Support

If Debugger Support is enabled for the main image, but disabled during the Secondary Bootloader phase (Controlled by **INFOC_DCU_DISABLEOVERRIDE**), the Secondary Bootloader needs to implement additional logic to check if a halt is requested by the debugger after the bootloader, and if so, halt the processor to allow a debugger to connect, after re-enabling the debugger using **MCUCTRL:DEBUGGER**.

- Checking for a Halt request from Debugger – Check if Least Significant Bit of register **MCUCTRL:SCRATCH0** is set
- If set, clear the bit and halt the processor using **DHCSR** register.

NOTE: The HAL function **am_hal_bootloader_exit()** in the file **am_hal_security.c** performs the sequence described above and is intended for use in a OEM's secondary bootloader to enable the debugger and initiate the debugger halt as it hands control to the main image.

SECTION

7

OEM MRAM Recovery Image

The Apollo5 OEM Recovery Image format is used to encode the various pieces of the OEM's MRAM recovery image. The image is similar to the OTA image described in *Section 3 Image Formats on page 11*and all the field descriptions apply to this format as well. This format does not use the standard OTA header. During MRAM recovery the OEM certs used for the RSA validation are likely corrupted, so this image also includes the Public Key for the selected key index specified for authentication.

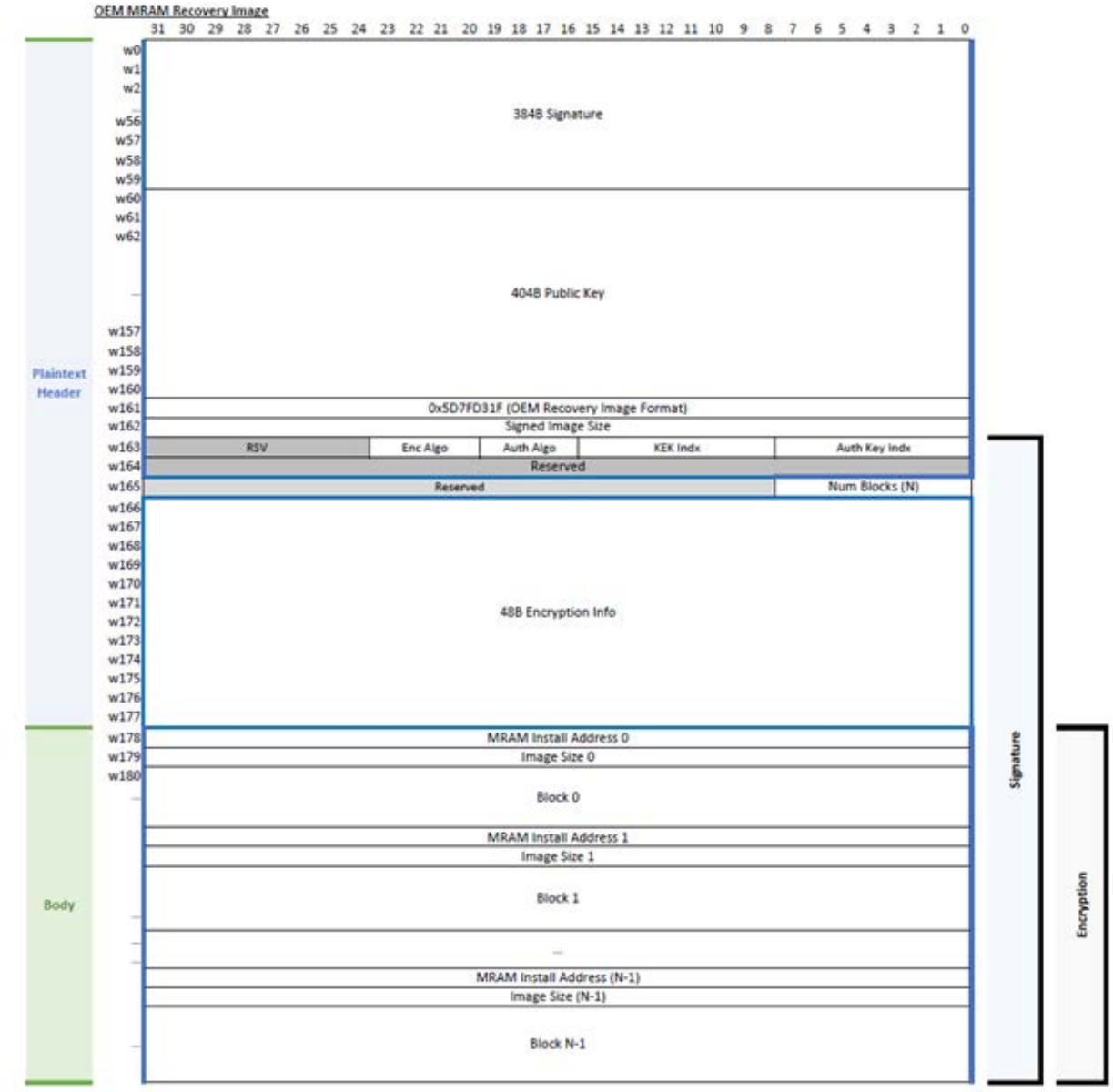
Note that the OEM MRAM Recovery image type does not include a CRC over the image. It is expected that this image will always be authenticated when deployed which makes the CRC redundant. The image can be created without authentication (and encryption) for test purposes when testing MRAM Recovery in DM-LCS before OEM keys may have been provisioned, but should not be deployed unless the image is signed, as bit-error in the image will not be detected, and security could also be compromised.

The OEM MRAM Recovery image is made up of 5 blocks, comprised of:

- OEM Recovery Application binary
- OEM Content Cert*
- OEM Key Cert*
- OEM Root Cert*
- Cert Chain Pointers*

The blocks can be specified in any order. The Ambiq tools build the OEM image in the order specified above. The Certs and Cert Chain Pointers are optional and will be not included if not provided (e.g. when not using SecureBoot), which results in the image having only one data block, the OEM Recovery Application binary).

Figure 7-1: OEM MRAM Recovery Image



6500 River Place Boulevard, Building 7, Suite 200, Austin, TX 78730
www.ambiq.com
sales@ambiq.com
+1 512. 879.2850

A-SOCAP5-UGGA02EN v1.0
May 2025