USER'S GUIDE

Apollo5 MRAM Recovery

Ultra-Low Power Apollo5 SoC Family

A-SOCAP5-UGGA01EN 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

Revision	Date	Description
1.0	April 2025	Initial Release.

Reference Documents

Document ID	Description
A-SOCAP5-UGGA02EN	Apollo5 Secure Update User's Guide
A-SOCAP5-UGGA03EN	Apollo5 OEM Provisioning Update and Tools User's Guide
A-SOCAP5-UGGA04EN	Apollo5 Security 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

List of Figures

SECTION

1

Introduction

This document describes the MRAM Recovery feature in the Apollo5 family of SoCs.

1.1 Background

Magnetic RAM (MRAM) is Ambiq's preferred solution for non-volatile memory (NVM) in the Apollo5 family of products. However, in certain applications, exposure to very strong external magnetic interference can lead to the corruption of MRAM on the device. The magnetic interference can be instantaneous and strong or there can be an accumulation over time. Both can lead to corruption. The Apollo5 family of devices were designed to be used in compact, battery-operated devices such as smart watches and bands. These compact designs often do not allow for sufficient magnetic shielding that could prevent the interference from disrupting the device.

1.2 MRAM Recovery Support in Apollo5 Family

To address these issues, the Apollo5 family of devices includes the addition of several architectural improvements to harden the device when compared to the Apollo4 family, including:

- Apollo5 has been hardened to hold its trims in integrated OTP inside the memory.
- Ambiq and Customer trims, security configurations and key assets are now held in dedicated anti-fuse-based OTP memory.
- Secure BootROM (SBR) has been moved to ROM.
- MRAM Recovery has been added to the Secure Boot flow.

The remainder of this document describes the functionality of the MRAM Recovery flow, its configuration, and hardware design considerations.

1.3 MRAM Recovery Initiation

MRAM corruption can be detected in different ways during the normal boot flow of the Apollo5 device:

- Secure BootROM (SBR) can detect corruption in the Secure Bootloader (SBL) area of MRAM.
- Secure Bootloader can detect corruption in the customer (OEM) image[1].
- Watchdog Timer (WDT) protection during the transitions from SBL to OEM image.

Additionally, corruption can be detected by various forms of active monitoring during normal execution.

- Active monitoring by the OEM application.
- Active monitoring by an external processor.

In the cases of the SBR or SBL detecting corruption or WDT expiration inside the SBL, MRAM Recovery will be automatically initiated. In the case of the OEM application detecting corruption, the OEM can initiate an "Application Initiated" recovery. In the case of an external processor detecting corruption, the external processor can initiate an MRAM recovery via a configurable GPIO pin.

1.4 MRAM Recovery Options

The MRAM Recovery supports a variety of options to facilitate retrieval of "golden" assets/images including:

- eMMC device
- Multi-bit SPI device
- UART interface (external processor)
- SPI interface (external processor)

[1]This assumes the customer is using Secure Boot mode and has provisioned OEM content certificates.

SECTION

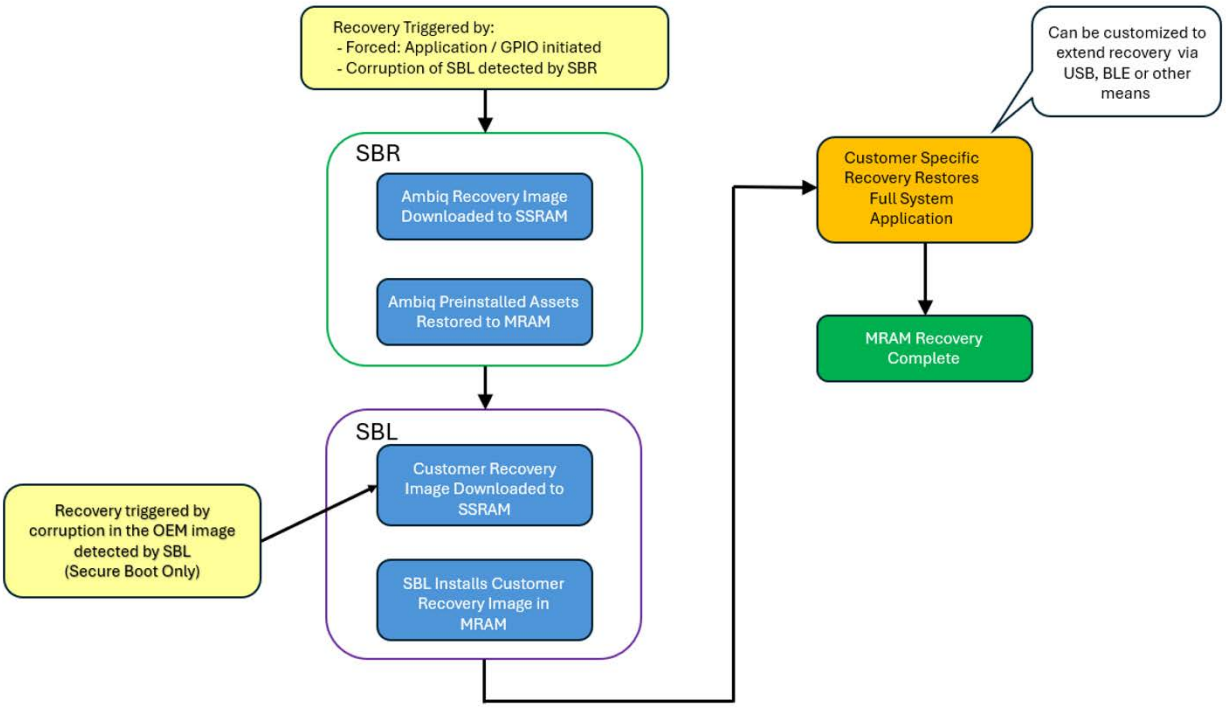
2

MRAM Recovery Flow

MRAM recovery involves three stages of processing as shown in Figure 2-1.

- MRAM Recovery in the Secure BootROM (SBR) that restores the SBL.
- MRAM Recovery in the Secure Boot Loader (SBL), that restores the OEM's initial recovery image.
- Additional customer MRAM Recovery processing steps as designed and implemented by the OEM.

Figure 2-1: MRAM Recovery Stages



MRAM recovery typically starts in the SBR which is responsible for restoring the SBL and its certificate chain. This can start autonomously by SBR detecting the corruption in the SBL or through an explicit trigger initiated by the application or an external host. In addition, it may also begin in the SBL if corruption is detected in the secured customer assets and code. Once the Ambiq assets have been restored, the MRAM recovery flow turns control over to the customer's specific recovery image (code) to complete the recovery of the customer's application, and other NVM assets.

2.1 Recovery Stages

2.1.1 MRAM Recovery in the Secure BootROM (SBR)

The first recovery stage can occur autonomously during the secure boot flow or can be initiated by the customer's background passive or active monitoring. As part of the normal secure boot flow, the SBR must validate the certificate chain used to authenticate the SBL. If the authentication operation fails, then MRAM corruption is suspected in either the SBL image or its content certificates. In this case, the SBR autonomously triggers a recovery of the SBL assets when it is configured and enabled. If the customer is using a non-volatile backup such as MSPI Flash or eMMC, then the recovery flow will continue through the next two stages. If the customer is using one of the wired interfaces, the corruption event may be flagged using an optional status GPIO and then wait for the Host processor to initiate the recovery stages.

2.1.2 MRAM Recovery in the Secure Boot Loader (SBL)

The second stage of MRAM recovery occurs autonomously during the secure boot flow[2]. Like the SBR, the SBL must validate the certificate chain supplied by the customer for their secondary bootloader or application images. Failure of authentication will trigger a recovery of the customer assets like with SBR. It should be noted that during any secure boot flow, the SBR stage may pass while the SBL stage could fail. These stages operate independently and recover only the assets required. However, once a corruption is detected, all subsequent images are recovered.

> **NOTE:** When provisioning the device, the customer chooses a secure or non-secure configuration for boot mode. In the non-secure variant, there is no customer security chain to verify the customer image against, so detection of corruption in the customer's MRAM code and data must be done via their active (or passive) monitoring as discussed later.

[2] Note that the customer security configuration in INFO0 must be configured for SecureBoot if SBL detection is desired.

2.1.3 Customer MRAM Recovery

The final stage of MRAM Recovery is implemented by the customer, based on their product's design requirements. Once the SBL restores the Customer's Recovery Image (which is limited to 256KB), the SBL will pass control to this image. The customer is responsible for the functionality of this image to recover remaining NVM assets. It may include communications to other outside devices via USB (Universal Serial Bus) or Bluetooth Low Energy FOTA (Firmware Over-the-Air) update services to complete the recovery.

2.2 Recovery Interface Options

For both the SBL, and the Customer's Recovery images, several configurable options are available for retrieving or loading the recovery images.

2.2.1 Generic MSPI Driver

This interface uses one of the MSPI modules on Apollo5 to attach to an external flash device. The flash device will contain the Recovery Assets. Since there is no uniform interface to all MSPI flash devices, the configuration information allows for specific MSPI device settings, as well as the ability to send up to 4 "pre-commands" to the device to get it into the proper state.

2.2.2 eMMC Device

This interface uses one of the SDIO (Secure Digital Input Output) modules on Apollo5 to attach to an external eMMC device. The eMMC device contains the Recovery Assets. The protocol to the eMMC device is mostly standardized, so there are only a couple of configuration options (e.g. CLK Speed).

2.2.3 UART or SPI Wired Interfaces From Host Processor

Apollo5 devices support wired interfaces from a Host processor or PC via both SPI and UART. The protocol is a subset of the existing SBL wired update protocol. Since the MRAM Recovery algorithm is active in both the SBR and SBL, the response STATUS message to the initial HELLO connection establishment indicates whether SBR or SBL is executing, and which recovery operation is in progress. If both UART and SPI interfaces are enabled[3], the SBR/SBL will look for activity on the UART prior to switching to SPI after the configured timeout period.

[3] Note that this configuration is allowed but is not typical. Ambiq recommends that each customer choose either UART or SPI and configure the MRAM Recovery for a single wired interface.

2.3 Recovery Assets

The recovery assets are uniquely formatted for interpretation by SBR/SBL. The AmbiqSuite SDK includes tools to generate OEM specific MRAM recovery asset images.

2.3.1 Ambiq Recovery Image

This image contains the SBL, and the certificate chain required to authenticate it. This image will be provided by Ambiq in the SDK.

2.3.2 Customer Recovery Image

This image contains a customer specific recovery image and optionally the certificate chain required to authenticate it. Additional assets as needed by the customer's specific recovery application may also be required, but they are beyond the scope of this MRAM recovery process.

2.3.3 Meta Data (when using non-volatile memory option)

For either the MSPI/Flash or SDIO/eMMC NVM options, the configuration data contains a single location that specifies the Meta Data file on the recovery media, that contains the specific locations (sectors or blocks) on the device of where to find the recovery images.

2.3.4 Recovery Image Maintenance

Note that these assets must be maintained on the external non-volatile device or in the Host processor. If the customer installs an Ambiq SBL update, new application images, or monitored data, then it is up to the customer to keep the recovery assets up to date including the proper versioning if anti-rollback features are enabled.

2.4 Passive and Active Surveillance

Customers may elect to passively monitor for MRAM corruption that trigger a Hardfault or cause a Watchdog timeout. Additionally, more active surveillance measures may be warranted, including periodic CRC checks on static images or dynamic data in addition to the passive methods.

2.5 Triggered Recovery

If any of the passive or active monitoring detects MRAM corruption, the customer may also trigger the MRAM recovery flow through an application API call or through a configurable GPIO. In this case, the customer will use their own methods for detecting corruption. For example, a test could be added to the Hard Fault or Watchdog ISR to trigger recovery through the API call when an abnormal condition is detected.

2.6 Recovery Status

Depending upon the customer application, recovery status is reflected in several ways.

2.6.1 GPIO Status Pin

An optional GPIO can be designed to be used by a Host processor that can be monitored to determine if a recovery is in process.

2.6.2 RSTGEN->STAT Register

The RSTGEN->STAT register includes results for MRAM Recovery performed by the SBR and SBL during the current boot cycle:

- Ambiq MRAM Recovery Occurred, Success/Fail
- OEM MRAM Recovery Occurred, Success/Fail
- Where the Ambiq Recovery Image was loaded from, NV or Wired.
- Where the OEM Recovery Image was from NV or Wired.
- The Reason for MRAM recovery: SBL Certs, OEM Certs, GPIO, App Request
- Source of Wired Recovery images (UART or SPI)
- MRAM Recovery in Process, (internal status and typically not visible the App)

2.6.3 OTP_INFO1_MRAM_RCVY_CNT0/1

A non-volatile count is maintained (in readable INFO1-OTP) of how many times MRAM recovery has been successful. These words are 'bit-counts', each bit set represents a successful recovery. The count saturates at 63 (2 words). In addition, both the SBL and the OEM recovery example application output the count in the SWO logs. Note that the example OEM recovery application also keeps a count of successful OEM recovery operations in a similar fashion, in the last 2 words of INFO0-OTP (customer defined area).

2.7 Retries

In some cases, an initial recovery attempt may not succeed due to ongoing exposure/reoccurrence of magnetic interference. The MRAM recovery algorithm provides a configurable option to have a specific number of retries during the recovery of the Ambiq Recovery Image, with increasing intervals between each retry attempt. The first retry attempt is tried after a minimum time interval. The interval is then increased by the minimum interval. This continues until the interval reaches the maximum configured interval. All remaining retries will then use the maximum interval until the maximum number of retries is reached. If unsuccessful after all retry attempts, a power cycle of the device is required to restart the MRAM recovery process over again.

Any recovery of the Ambiq SBL will always initiate recovery of the OEM image. The OEM recovery will always be tried twice, with an intervening POI that will reverify the ICV certs and Ambiq SBL image.

The retry settings only apply to the recovery of Ambiq's MRAM recovery image, during which the recovery process looks for and verifies that there is no ongoing magnetic interference or continued corruption taking place before completing the SBL recovery. Then after a successful SBL recovery, during the subsequent OEM recovery, it is expected that the OEM recovery will proceed without the need for additional retries. If the OEM recovery fails for any reason, the device will perform a POI reset (which initiates a recheck of the ICV certs and SBL), and if further corruption is detected will initiate the SBL recovery over again, with retries if no corruption is found then the OEM recovery will be attempted a second time, if it again fails the device will require a power cycle to restart the recovery process, the same as when the SBL retries are exhausted.

2.8 External Device Power up and Reset for External NV Device

For external devices (MSPI Flash or eMMC) there is a specific power-up and reset sequence as follow (with each of these steps being optionally enabled or disabled):

- Powered on via specified pin and polarity (if configured).
- Followed by configurable delay (12-bit field, specified as µS or mS)
- Reset via reset pin and polarity specified (10us pulse)
- Followed by device reset delay time (12-bit field, specified as µS or mS)
- JEDEC reset for MSPI (if enabled)
- Followed by device reset delay (same reset delay field)
- Pre commands sent in 1-1-1 mode (for MSPI flash devices)[4]

The device is now ready to read the recovery images that will be authenticated and loaded into the MRAM. After reading the assets from the device, the device is reset

[4] Pre commands can be used to put external MSPI flash devices into data bus mode corresponding to the configuration in INFO0, but their use is very device dependent.

and powered down in the opposite order. All GPIOs are then returned to their reset state.

2.9 Recovery Configuration Strategy for Production

In the initial Apollo5 devices delivered to the customer, INFO0 is unprogrammed. The devices are provided in DM LCS. The configuration for MRAM recovery must be programmed by the customer into INFO0 (and INFOC for wired recovery) during the production manufacturing process.

MRAM Recovery is not designed to be used during the production process to program assets. This should be done in a normal fashion through existing means. INFO0 should be programmed coincident with the MRAM Recovery assets.

As with all development, the production programming process should be developed and tested using INFO0-MRAM based trims, then transitioned to OTP based trims once the final configuration is verified, prior to field deployment.

2.10 MRAM Recovery Design Considerations

MRAM Recovery must be considered during the product design phases, depending upon the customer's application and external device considerations. Specifically, the customer needs to consider:

- Location of the Recovery Assets (NVM or Host Processor)
- Selection and configuration of the MSPI[5] or SDIO instance to use for MRAM Recovery
- Power and reset pin selection for an external device.
- GPIO Control and Status pins, polarity, and pullup/pulldown requirements.
- Powerup/reset delay requirements for the selected external device.
- MSPI sector/page and eMMC partition selection/management for Recovery Assets.

[5] Many MSPI Flash devices include non-volatile configuration registers which may be programmed by customers to suite their application. MRAM recovery is designed to avoid changing these non-volatile registers.

SECTION

3

Enabling MRAM Recovery

To familiarize and enable the MRAM recovery process, the customer should perform the steps as outlined in the following sections. This section focuses on the use of eMMC or MSPI NV devices as the source of the recovery assets, however, the principles extend to the UART or SPI interfaces as well.

3.1 Quick Start

The AmbiqSuite SDK provides a set of "canned" assets to aid in the understanding of the MRAM Recovery flow and tools. As a first step, it is highly recommend that the customer follows the *MRAM Recovery Quick Start Guide* and the provided "canned" assets and tools to gain an initial understanding.

3.2 Creating Recovery Image Binary

The customer can start development using Non-Secureboot in DM-LCS[6]. At this stage no OEM certificates are required, thus the SBL cannot detect corruption of the OEM image. However, other detection methods are operational. This allows the customer to test the NV device provisioning and INFO0 settings with a quick path to recovery if something goes wrong.

The customer has complete freedom to create an OEM recovery application specific to how the product maintains access to program images/data and how they plan to use customer MRAM. Ambiq provides a simplified example OEM recovery application in the SDK (see /examples/mram_recovery/mram_rcv_app) that can be used as the basis for their custom application. Ambiq recommends that the customer provide output status for debug. Ambiq provides the **create_cust_image_blob.py** tool to convert the customer's Recovery Application and certificates into the OEM Recovery Image.

[6] See OTP_INFOC_SECURITY field in OTP-INFOC. CUST_SECBOOT should be set to SBDIS (the default).

3.3 Provisioning the NV Device

The customer then needs to determine where on the NV device the Ambiq and OEM Recovery Image(s), and Metadata will be located. The SDK includes an example (see /examples/mram_recovery/image_loader) which is used to load the NV devices on the Ambiq EVB board, but could be modified to match the customer's product pin selections and NV device[7]. Ambiq's Recovery Image (SBL & Certificates) are also supplied in the SDK. Once the image_loader program has been updated and compiled, the customer should update the **recovery_loader.ini** file (See \tools\apollo510_scripts\mram_recovery) for the placement of the recovery assets, and can then use the **recovery_nvloader.py** tool to load the images into their specific NV device.

3.4 Program INFO0

The next step is to determine the configuration settings and desired options in INFO0 fields. Once this has been done, run the **create_info0.py** tool to create the INFO0 binary that is used for programming either the MRAM or OPT INFO0 spaces. At this stage, INFO0-MRAM should be to verify and test the configuration and NV device, however, actual magnetic interference testing requires INFO0 to be configured to have OTP as the active configuration.

3.5 Final Testing

Once the NV device and INFO0-MRAM configurations are established, the customer should test the MRAM Recovery flow using an application-level call (see am_hal_mram_recovery_init_app_recovery) or using GPIO to initiate recovery.

The final steps for testing MRAM Recovery are:

- Finalize the OEM Recovery application, create the OEM Certificates, are used together to create the **OEM_Recovery_OTA**, and then use the **recovery_nvloader** to program the NV device.
- Program INFO0-OTP with the same settings confirmed in MRAM-INFO0.
- Switch the **CUST_SECBOOT** to SBEN in the **INFO_SECURITY** word.
- Switch the device to boot from OTP (see **INFOC_SHDW_TRIM_INFO0_SEL**)

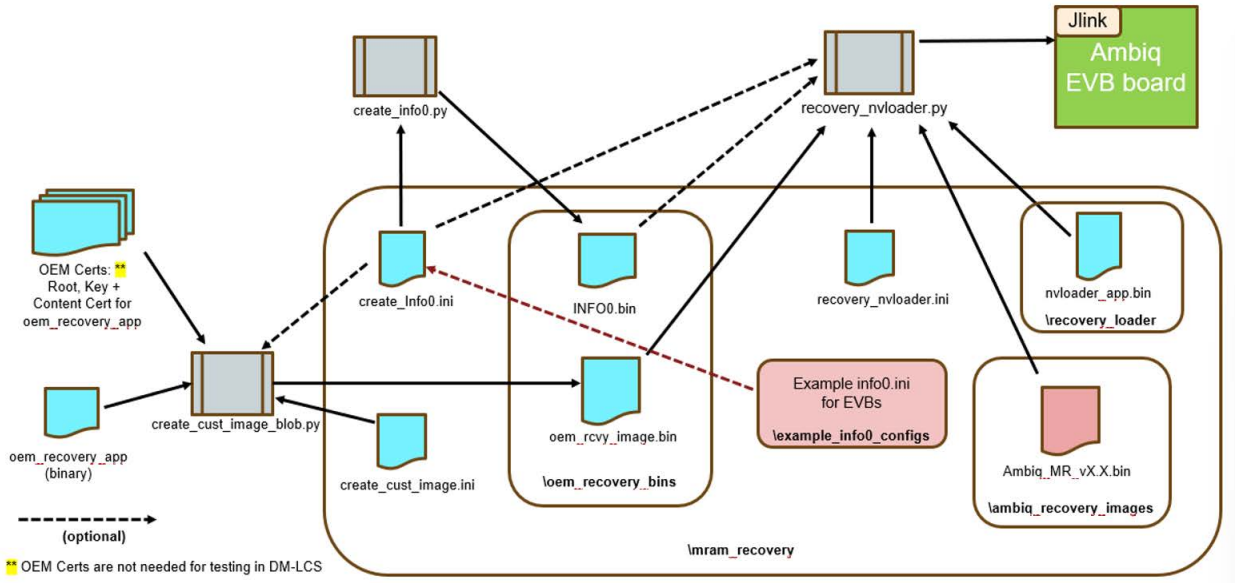
At this point, the system is enabled for actual magnetic interference testing.

[7] The nvLoader example requires only minimal or no modification for eMMC.

3.6 MRAM Recovery Tool Flow

See Figure 3-1 which shows the overall MRAM Recovery tool flow corresponding to the previous sections.

Figure 3-1: MRAM Recovery Tool Flow



3.7 SDK Directory Structure for MRAM Recovery Tools and Example Configs

The MRAM Recovery tools, examples, and configuration files can be found in the following directories:

- **\tools\apollo510_scripts**: (additions to preexisting scripts)
  - **create_cust_image_blob.py** (updated to add OEM_Recovery type)
  - **create_info0.py** (updated to have MRAM Recovery fields allow input via an **.ini** file)
  - **uart_recovery_host.py** (script that acts as a host for UART (and SPI) based MRAM recovery)
- **\mram_recovery**: (directory in **\tools\apollo510_scripts**)
  - Three .ini files for use with **create_cust_image_blob.py** (output builds to .\oem_recovery_builds)
    - **oem_recovery.ini** that creates the OEM's MRAM recovery image
    - **oem_recovery_wired.ini** that creates a "_wired" image for UART/SPI loading from OEM recovery image
    - **ambiq_recovery_wired.ini**, creates a "_wired" Ambiq recovery image for using UART/SPI to load Ambiq recovery image.

- **recovery_nvloader.ini** file for use with recovery_nvloader.py (that loads NV assets into the flash devices in the Ambiq Boards)
- **\ambiq_recovery_images** (contains Ambiq recovery images for official SBL releases)
- **\example_info0_configs** (directory with .ini files to generated INFO0.bin files for various configurations)
- **\recovery_loader** (directory with .py script, and nvloader_app.bin to load EVB's NV devices.
  - **recovery_nvloader.py** (script to automate the loading of EVB's NV devices (and INFO0 config)
  - **nvloader_app.bin** prebuilt executable from \examples\mram_recovery\image_loader run by recovery_nvloader.py wrapper script
- **\oem_recovery_bins** (default output for OEM recovery image builds)
  - Populated with pre_built OEM_App binary, and Certs

– **\boards\examples\MRAM_recovery**:

- **\mram_rcv_app** (example OEM recovery app – demos various MRAM corruption detections)
  - **crc_to_bin.py** (script to calculate and add a CRC into a .bin file)
- **\image_loader** (Loader for NV devices on Ambiq EVB boards)

3.8 Consistency Checking of Fields in the *.ini Scripts

There are several fields that need to be consistent between the various scripts and what is programmed in INFO0 space:

– MAINPTR
– CERTCHAINPTR
– NV_METADATA_OFFSET_META_OFFSET
– MRAM_RCVY_CTRL_NV_RCVY_TYPE
– MRAM_RCVY_CTRL_NV_MODULE_NUM
– MRAM_RCVY_CTRL_EMMC_PARTITION

Each of the related **.ini** files has an optional keyword **info0Cfg** with the path to the **info0.ini** for the **INFO0** that will be used with the file being generated. The fields above will be checked for consistency when the **info0** pathname is specified. The fields can be omitted when the **info0Cfg** path is specified and the values from the info0.ini will then be used. Any inconsistencies will be flagged as an error when multiple scripts specify them differently. All scripts have the same defaults for these fields when not specified.

If the **info0Cfg** keyword is not specified, the scripts work separately (standalone) and no consistency checks are done. The examples in the SDK all use the **info0Cfg** keyword so inconsistencies will be flagged when changes are made.

The **recovery_nvloader.py** script can also (optionally) generate and load INFO0 if the **info0Cfg** path is supplied along with the **regenInfo0** and **loadInfo0** keywords in the **.int** file. However, **INFO0** can always be loaded separately using the **jlink-prog-info0.txt** or **jlink-prog-info0_otp.txt** as in previous SDKs.

SECTION

4

MRAM Recovery Configuration

MRAM Recovery is configured and enabled in INFO0[8]. The fields are defined in detail in the register definitions included with the SDK. There are five main areas of configuration for MRAM Recovery:

- Enables (1 word)
  - MRAM Recovery feature enabled.
  - Application Initiated MRAM Recovery enabled.
  - GPIO designated for MRAM recovery status.
  - GPIO to control triggering MRAM recovery.
  - Source of recovery data: MSPI-Flash, eMMC, Wired-UART, Wired-SPI.
  - Instance number for the MSPI or eMMC to use.
- Metadata Location (1 word)
  - Location of the page (MPSI) or block (eMMC) that contains the metadata that further defines the location and sizes of the recovery images.
- Resets, Power, and Device Pin configurations (6 words)
  - External device power and reset GPIO pin configurations and delays.
  - CE, CMD, CLK, Data, and DQS GPIO pin configurations
- Device Specific Configuration (MSPI/eMMC) (6 words)
- Retry Configuration (1 word)

[8] Note that INFO0 may be sourced from MRAM during development so it can be easily updated and changed. INFO0 must be sourced from OTP for production devices, and for any magnetic corruption testing.

Figure 4-1: MRAM Recovery Configuration in INFO0

INFO0 Index

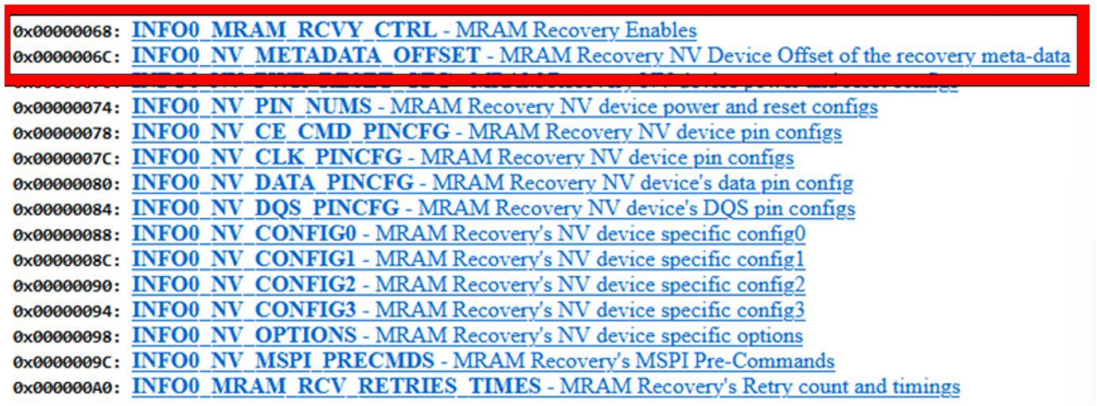
INFO Base Address = 0x42000000

0x00000000: **INFO0_SIGNATURE0** - INFO0 Signature
0x00000004: **INFO0_SIGNATURE1** - INFO0 Signature
0x00000008: **INFO0_SIGNATURE2** - INFO0 Signature
0x0000000C: **INFO0_SIGNATURE3** - INFO0 Signature
0x00000010: **INFO0_BLRESET** - Security protection bits
0x00000014: **INFO0_CUSTOMER_TRIM** - Customer trim values
0x00000028: **INFO0_SECURITY_WIRED_IFC_CFG0** - Security Wired Interface configuration word0
0x0000002C: **INFO0_SECURITY_WIRED_IFC_CFG1** - Security Wired Interface configuration word1
0x00000030: **INFO0_SECURITY_WIRED_IFC_CFG2** - Security Wired Interface configuration word2
0x00000034: **INFO0_SECURITY_WIRED_IFC_CFG3** - Security Wired Interface configuration word3
0x00000038: **INFO0_SECURITY_WIRED_IFC_CFG4** - Security Wired Interface configuration word4
0x0000003C: **INFO0_SECURITY_WIRED_IFC_CFG5** - Security Wired Interface configuration word5
0x00000040: **INFO0_SECURITY_VERSION** - Security version field
0x00000044: **INFO0_SECURITY_SRAM_RESV** - DTCM Reserved for Application Scratch space
0x00000048: **INFO0_SECURITY_RMAOVERRIDE** - RMA override.
0x00000054: **INFO0_WIRED_TIMEOUT** - Wired timeout
0x00000058: **INFO0_SBR_SDCERT_ADDR** - SBR SDCERT ADDR
0x00000060: **INFO0_MAINPTR** - Pointer to the main application.
0x00000064: **INFO0_CERTCHAINPTR** - OEM certificate chain
0x00000068: **INFO0_MRAM_RCVY_CTRL** - MRAM Recovery Enables
0x0000006C: **INFO0_NV_METADATA_OFFSET** - MRAM Recovery NV Device Offset of the recovery meta-data
0x00000070: **INFO0_NV_PWR_RESET_CFG** - MRAM Recovery NV device power and reset configs
0x00000074: **INFO0_NV_PIN_NUMS** - MRAM Recovery NV device power and reset configs
0x00000078: **INFO0_NV_CE_CMD_PINCFG** - MRAM Recovery NV device pin configs
0x0000007C: **INFO0_NV_CLK_PINCFG** - MRAM Recovery NV device pin configs
0x00000080: **INFO0_NV_DATA_PINCFG** - MRAM Recovery NV device's data pin config
0x00000084: **INFO0_NV_DQS_PINCFG** - MRAM Recovery NV device's DQS pin configs
0x00000088: **INFO0_NV_CONFIG0** - MRAM Recovery's NV device specific config0
0x0000008C: **INFO0_NV_CONFIG1** - MRAM Recovery's NV device specific config1
0x00000090: **INFO0_NV_CONFIG2** - MRAM Recovery's NV device specific config2
0x00000094: **INFO0_NV_CONFIG3** - MRAM Recovery's NV device specific config3
0x00000098: **INFO0_NV_OPTIONS** - MRAM Recovery's NV device specific options
0x0000009C: **INFO0_NV_MSPI_PRECMDS** - MRAM Recovery's MSPI Pre-Commands
0x000000A0: **INFO0_MRAM_RCV_RETRIES_TIMES** - MRAM Recovery's Retry count and timings

NOTE: In addition to these configuration values in **INFO0**, there is also the **OTP_INFOC_WIRED_CONFIG** word in **INFOC** that determines the wireless configuration used.

4.1 MRAM Recovery Enables and Metadata Configurations

Figure 4-2: MRAM Recovery Enables and Metadata Configurations



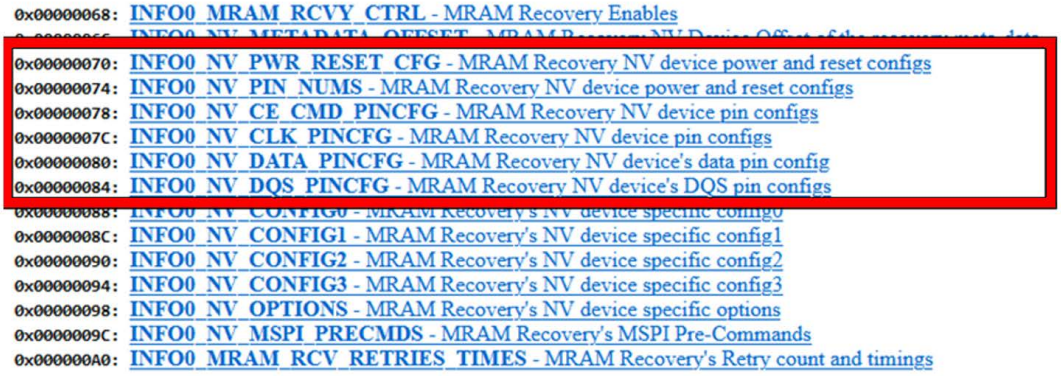
- The MRAM recovery Enables - 1 word
  - Master Enable (for MRAM recovery to be enabled **INFO0** must be valid, and master enable set)
  - GPIO Pin for Recover in Progress indication (0xFF means unused)
  - NV type and module number (EMMC, MSPI or none)
  - Wired Enable and type
    - For UART or SPI, additional configs are specified in the existing **INFOC WIRED_CONFIG** fields
  - App initiated Recovery enable
  - GPIO initiated Recovery enable (Pin # and Polarity)
  - App fail option (Reboot or spin if App or GPIO initiated recovery fails)
- Metadata Offset – 1 word
  - The Metadata is a set of records that define location and size of the recovery images
    - Ambiq's MRAM Recovery defines and uses the first two records
      - The first record specifies the SBL recovery image
      - The second is the OEM Recovery image
    - Additional records are customer defined and used
    - Each record is 2-words and defines the offset and size of a recovery image

4.2 NV Device Configurations (Power/Reset/Pin # & Configs)

During MRAM recovery, the NV device will be powered on via specified pin and polarity (if enabled) which is followed by a configurable delay (12-bit field, specified either as µS or mS). Next, the device will be reset via reset pin (if enabled) with specified polarity with a 10us pulse which is followed by a device reset delay time (12-bit field, specified as either µS or mS). Finally, an optional JEDEC reset for MSPI (if enabled) which is followed by device reset delay (same value as used for the reset delay field above).

The device can then be read, and the recovery image loaded. After reading the device (successful or not) the device will be reset and powered down in the opposite order. All pins will be de-configured and returned to their reset state following the loading process.

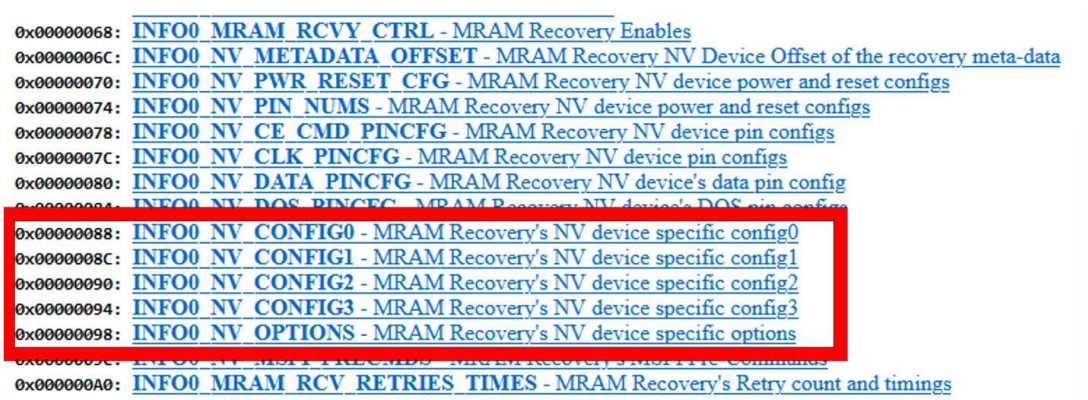
Figure 4-3: NV Device Configuration (Power/ResetPin # & Configs)



- Power and Reset options: - 1 word
  - Power pin polarity
  - Reset pin polarity
  - Delay following power-on (12-bit field, specified as µS or mS)
  - Delay time following reset (12-bit field, specified as µS or mS)
  - JEDEC reset (for MSPI)
- Pin Numbers - 1 word
  - Power and pin numbers (0xFF disables the function)
  - CE Pin number (for MSPI)
  - All other pin numbers are determined by the NV type and module number
- **PINCFGxx** fields for the specified pin(s): - 4 words
  - CLK (The pin # is determined by the NV type and module number)
  - CE/CMD (for EMMC the CMD pin is determined by module number)
  - Data0-7 (again the pin #s are determined by the NV type and module number)
  - DQS (MSPI only, the pin # is determined by the module number)

4.3 Device Specific Configuration (EMMC)

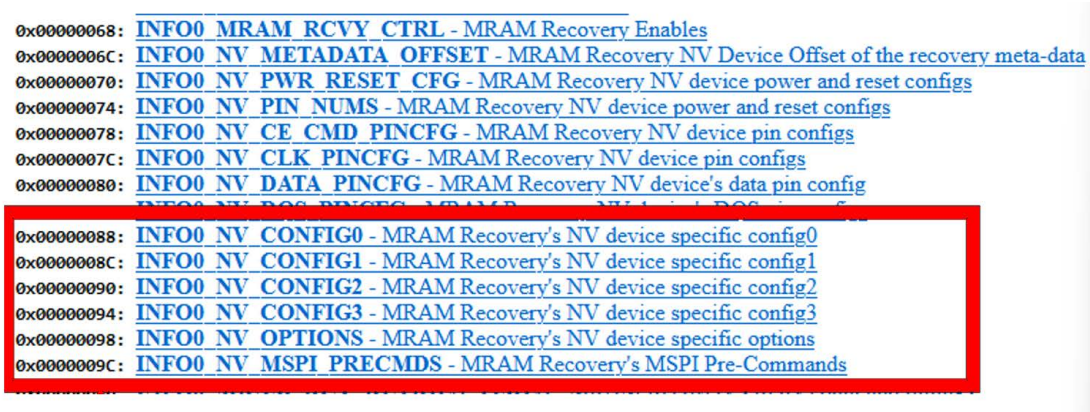
Figure 4-4: Device Specific Configuration (EMMC)



- The MRAM_RCVY_CTRL word has
  - EMMC_PARTITION – User, Boot1, Boot2
- EMMC devices use only **NV_CONFIG0** and **NV_CONFIG1** words
- NV_CONFIG0 specifies the target speed in Hz
  - The device will be configured to closest speed, not to exceed the target
- NV_CONFIG1 word – Specifies the EMMC UHS mode
  - As specified in the **UHSMODESEL** field in **REG_SDIO0_AUTO** register
- NV_OPTIONS Word – EMMC has 2 fields
  - EMMC_BUS_WIDTH – 1, 4, 8-bit SDR/DDR
  - EMMC_VOLTAGE - 1.8V, 3.0V, 3.3V

4.4 Device Specific Configuration (MSPI)

Figure 4-5: Device Specific Configuration (MSPI)



- The MRAM_RCVY_CTRL word has
  - SDIO module number to use (0 or 1)
- MSPI devices use the NV_CONFIG0-4 words and fields in NV_OPTIONS word and MSPI_PRECMDS
- NV_CONFIG0 word written directly to specified module's DEV0CFG register
- NV_CONFIG1 word written directly to specified module's DEV0CFG1 register
- NV_CONFIG2 word written directly to specified module's DEV0DDR register
- NV_CONFIG3 word written directly to specified module's DEV0SCRAMBLING register
- **NV_OPTIONS** Word – MSPI has 5 fields
  - **READCMD** – the 8-bit read command sent when reading the metadata and recovery images
  - **PRECMD_CTRL** – field that indicates how to send pre-commands, if any, before reading images
  - **PRECMD_CLKSEL – CLKSEL** field setting when sending pre-commands
  - **WIDTHS** – Width of the read transfers (loaded into the MSPIx_CTRL1 register PIOMIXED field)
  - **READ_CLKSEL** – The CLSEL register setting used when reading the meta-data and recovery images
- **NV_MSPI_PRECMDS** – 1 word of 4 bytes that can be sent before reading the Metadata/recovery images
  - Can send various combinations of CMDs or CMD with 1 or 2 data bytes

4.5 MRAM Wired Recovery Configuration – UART /SPI

MRAM Wired Recovery Configuration should be the same as Wired Update Config.

Figure 4-6: MRAM Wired Recovery Configuration - UART/SPI

- Wired recovery uses the existing wired configurations in **INFOC** and **INFO0**
- **WIRED_CONFIG** in INFOC (same as Apollo4p – with **SLAVEINT** field size increased)
  - Enables UART and/or SPI
  - Specifies which UART module
  - SPI enable/disable – MRAM Recovery uses IOS (not IOSFD)

  - SLAVEINT pin used for SPI
- INFO0 **SECURITY_WIRED_IFC_CFGx** (0-5) and **WIRED_TIMEOUT** configure the UART parameters (same as Apollo4p)
- UART/SPI Recovery message exchange when interacting with BootROM or SBL
  - The messaging protocol is subset of SBL wired update protocol
    - Same protocol as Apollo4p (The Hello Status response msg has been updated for Apollo5)
  - The HELLO Status response message is used to distinguish between Ambiq and OEM recovery.
- If both UART and SPI are enabled, UART will be tried first and then SPI
- The recovery images are formatted different than previous update images
  - The SDK provides tools to generate MRAM Recovery images

4.6 MRAM Recovery Retry Configuration

Figure 4-7: MRAM Recovery Retry Configuration

- If the MRAM recovery fails for any reason, it will retry a configurable number of times
  - Depending on length of magnetic interference, recovery may not succeed first time
- **MRAM_RCV_RETRIES_TIMES** has 3 fields to allow for retry configurations
  - **MAX_RETRIES** – The maximum number of retries (16-bit, 65K max retries)
  - **MIN_RETRY_TIME** – The initial time between retries (in seconds)
  - **MAX_RETRY_TIME** – The maximum time between retires (in Minutes)
- The time between retries will:
  - Start with the time specified in **MIN_RETRY_TIME** (in seconds) and
  - Will increment by that same value up to, but
  - Not to exceed **MAX_RETRY_TIME** (in Minutes)
- If the maximum retries are exceeded, the device will be "locked" and will require a power cycle
  - Power cycling the device will reinitiate the recovery process, and retry the full recovery number or retries and times.

NOTE: If MRAM recovery was initiated by the Application or GPIO, and there was no real MRAM corruption, and the recovery fails all retries for any reason (e.g. the Recovery images are not loaded into NV parts), the subsequent reset will boot normally if enabled by the **APP_RCVY_REBOOT** field in the **INFO0_MRAM_RCVY** word.

SECTION

5

Appendix A: SDK MRAM Recovery Configuration Examples

The Ambiqsuite SDK has several example configuration files for setting up INFO0 for MRAM Recovery using the devices (eMMC and MSPI) and wired interfaces (UART and SPI) available on the Ambiq EVB. Each of these can be used with the create_info0.py script to automatically generate and load the INFO0 for the selected configuration. They are all contained in the directory:

\tools\apollo510_scripts\mram_recovery\example_info0_configs.

Files: \info0_Recovery_EMMC.ini (the preconfigured default)
\info0_Recovery_MSPI(1-1-1).ini
\info0_Recovery_MSPI(1-8-8).ini
\info0_Recovery_UART_SPI.ini
\info0_Recovery_UART1.ini

Each of these represent a configuration that can be used on the Ambiq Apollo510 EVB.

Note that there are a large number of registers and fields for MRAM recovery, to list all the options use the "-hx" help option in the **create_info0.py script**. The normal -h help option lists only the options not used with MRAM recovery. The user can specify the MRAM recovery options as either a full register word or by the individual fields within the register (if both are specified, the last prevails).

The SDK provides an example .ini file for loading the recovery assets into the EVB's non-volatile memory devices using the **recovery_nvloader.py**:

\tools\apollo510_scripts\mram_recovery\recovery_nvloader.ini

By default it is configured to load the eMMC device, but by changing the "info0_cfg =" to point to one of the MSPI .ini examples the **recovery_nvloader.py** can be used to preload the MRAM recovery images into the onboard MSPI device.

Likewise the script **create_cust_image_blob.py** has an example .ini for taking the example OEM Recovery binary and turning it into a OEM Recovery image. The example .ini file for creating the OEM Recovery image is here:

\tools\apollo510_scripts\mram_recovery\oem_recovery.ini

When doing MRAM recovery via UART or SPI both the Ambiq recovery image and the OEM recovery image (created with the oem_recovery.ini) needs to be processed into a "_wired" image compatible with the wired protocol used by these interfaces. This is done again using **create_cust_image_blob.py** and the "image_type = wired" keyword. The SDK provides two example .ini files, one to create the Ambiq Recovery wired image, and one to create the OEM Recovery wired image. These two ini files are

\tools\apollo510_scripts\mram_recovery\ambiq_recovery_wired.ini
\tools\apollo510_scripts\mram_recovery\oem_recovery_wired.ini

When using the wired interface for MRAM recovery the SDK has an example script **uart_recovery_host.py** that will respond to the "in-progress GPIO" and deliver both the wired Ambiq Recovery image and the wired OEM 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-UGGA01EN v1.0
April 2025