User Guide

Alif Security Toolkit

V1.104.0

Table of Contents

Introduction

The Alif Security Toolkit (SETOOLS) contains several tools (written in Python3) required to program and provision an Alif Semiconductor SoC device.

Provisioning means being able to add images to the internal NVM storage (MRAM) as well as adding Security assets such as Keys to the One Time Programmable memory (OTP).

ALIF images, known as System TOC (STOC), for the device are already provisioned into the MRAM, a user cannot change these, but can update to a new STOC version supplied by Alif. User images such as Application programs (binaries written for A32 or M55 cores) need to be packaged and written to the MRAM. The packages are called an Application Table of Contents (ATOC).

There are also pre-built binary images for the ALIF Table of Contents (STOC) Package. The STOC contains the latest binary version of SERAM. The STOC also contains debug stubs which will execute on the Application cores to allow connection from the Debugger. These stubs are only loaded to Application CPUs that are not configured to run.

NOTE: These tools are designed to work with the following environment:

- Windows PC Executables (Windows 10/11)
- LINUX Executables (Ubuntu 18.4/20.4/22.4)
- MacOS Executables (tested with OS Sonoma 14.1).
- Ensemble DevKit Gen 2

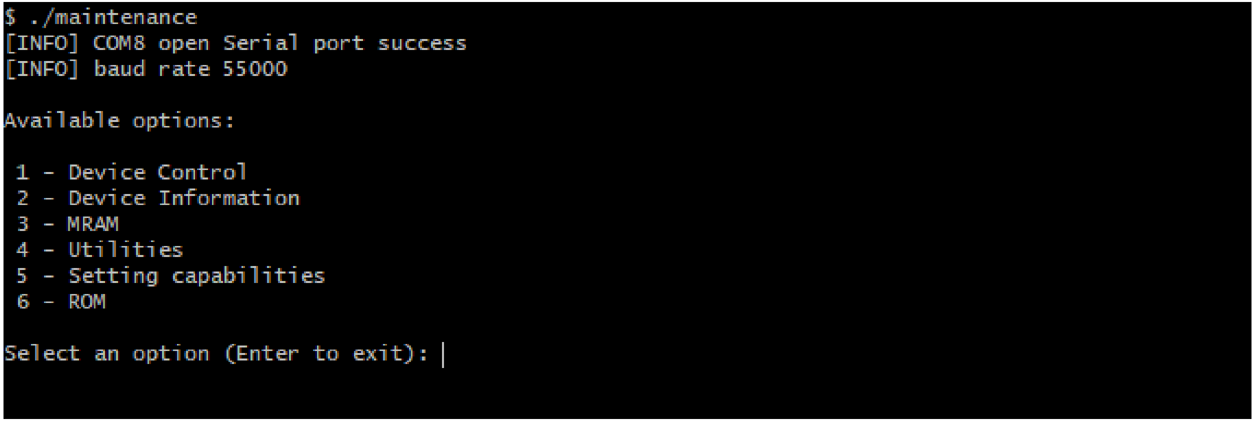
The Alif Ensemble DevKit User Guide (Gen 2) includes instructions on how to connect the SE-UART output of the CPU to the host computer. This guide will then refer to the Alif Security Toolkit Quick Start Guide which contains instructions for installation of these tools.

This release version is **app-*release-exec-windows(linux)(macos)-SE_FW_1.103.0_DEV.zip***

1. Python Executables

The Security tools are executable images, previously these python scripts were delivered as source code. These executables do not require python or any pre-requisite libraries to be installed.

Example:



```
$ ./maintenance
[INFO] COM8 open Serial port success
[INFO] baud rate 55000

Available options:

 1 - Device Control
 2 - Device Information
 3 - MRAM
 4 - Utilities
 5 - Setting capabilities
 6 - ROM

Select an option (Enter to exit):
```

Previously you were required to run

```
$ python3 <tool-name>.py
```

Now you just run

```
$ <tool-name>
$ ./<tool-name>
```

NOTE: Screenshots in this document and other application notes or user guides might show the previous format but apply to this version or later releases if you enter just <tool-name> instead of “python3 <tool-name>.py”.

2. Quick User Guide

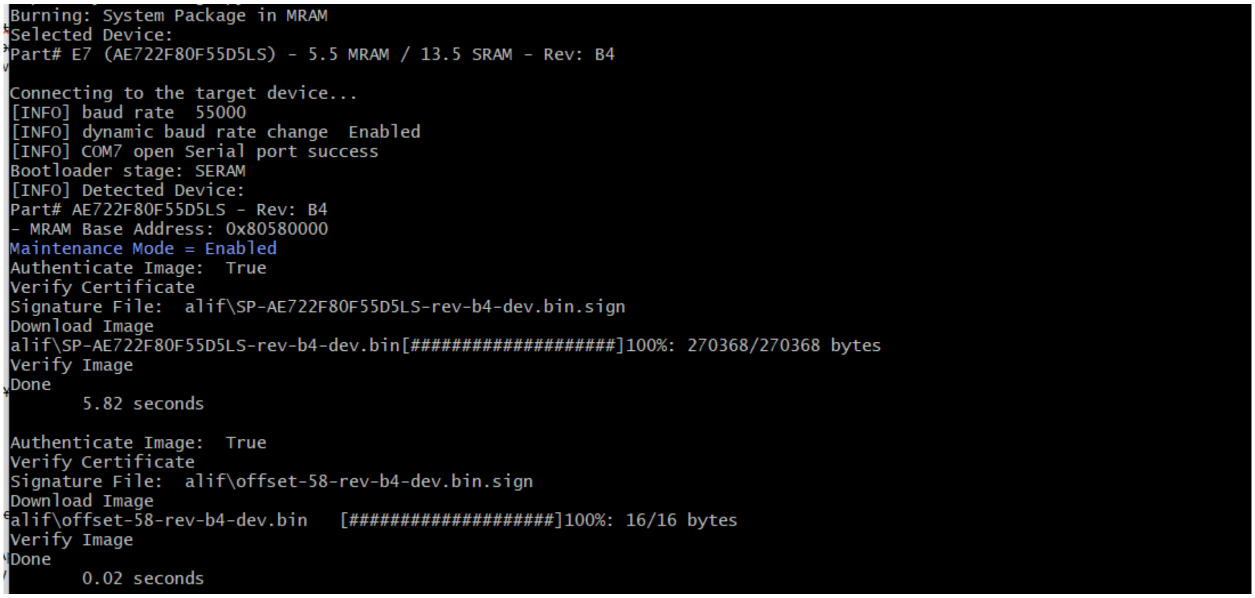
Please ensure you have followed the Installation Guide in the Alif Security Toolkit Quick Start Guide before proceeding.

The recommended <release-location> installation directories are:

Windows:	C:\app-release-exec
Linux:	/home/$USER/app-release-exec-linux
MAC OS:	/Users/$USER/app-release-exec-macos

Updating to the latest ALIF release

```
$ cd <release-location>
$ updateSystemPackage
```



```
Burning: System Package in MRAM
Selected Device:
Part# E7 (AE722F80F55D5LS) - 5.5 MRAM / 13.5 SRAM - Rev: B4

Connecting to the target device...
[INFO] baud rate  55000
[INFO] dynamic baud rate change  Enabled
[INFO] COM7 open Serial port success
Bootloader stage: SERAM
[INFO] Detected Device:
Part# AE722F80F55D5LS - Rev: B4
- MRAM Base Address: 0x80580000
Maintenance Mode = Enabled
Authenticate Image:  True
Verify Certificate
Signature File:  alif\SP-AE722F80F55D5LS-rev-b4-dev.bin.sign
Download Image
alif\SP-AE722F80F55D5LS-rev-b4-dev.bin[####################]100%: 270368/270368 bytes
Verify Image
Done
       5.82 seconds

Authenticate Image:  True
Verify Certificate
Signature File:  alif\offset-58-rev-b4-dev.bin.sign
Download Image
alif\offset-58-rev-b4-dev.bin   [####################]100%: 16/16 bytes
Verify Image
Done
       0.02 seconds
```

Adding / Changing an application image

To change or update an image to add to MRAM do the following:

```
$ cd <release-location>
$ edit build/config/app-cfg.json and add new binaries images

$ app-gen-toc

(Or use below command to use a different configuration)

$ app-gen-toc -f build/config/app-myfile.json

$ app-write-mram
```

The app-gen-toc tool performs the packaging of your binary ready for writing to the MRAM.

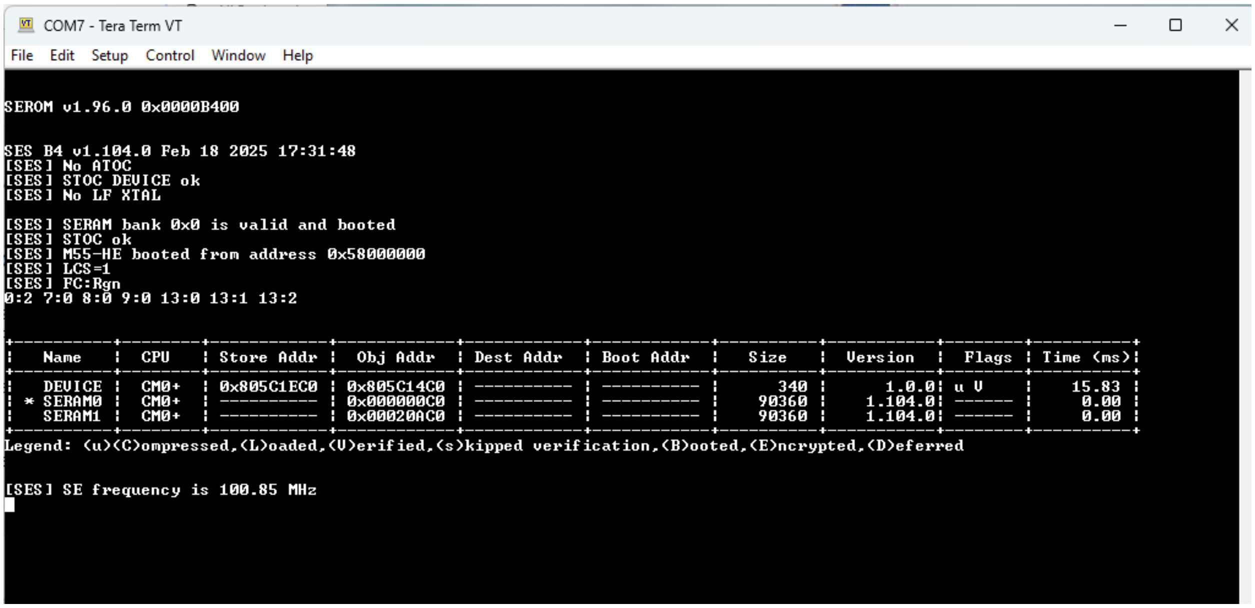
The app-write-mram.py tool performs the writing of the application package to the MRAM. After this is completed, it will reset your target platform and the contents of the MRAM will be executed.

3. TOC Packages

The device will arrive already provisioned. This means that the SES is already programmed into MRAM.

System Table of Contents (STOC)

On the initial power up of the device you will see the following SE-UART output:



```
SEROM v1.96.0 0x0000B400


SES B4 v1.104.0 Feb 18 2025 17:31:48
[SES] No ATOC
[SES] STOC DEVICE ok
[SES] No LF XTAL

[SES] SERAM bank 0x0 is valid and booted
[SES] STOC ok
[SES] M55-HE booted from address 0x58000000
[SES] LCS=1
[SES] FC:Rgn
0:2 7:0 8:0 9:0 13:0 13:1 13:2


+----------+--------+------------+------------+------------+------------+----------+-----------+-------+----------+
|   Name   |  CPU   | Store Addr |  Obj Addr  | Dest Addr  | Boot Addr  |   Size   |  Version  | Flags | Time (ms)|
+----------+--------+------------+------------+------------+------------+----------+-----------+-------+----------+
|   DEVICE |  CM0+  | 0x805C1EC0 | 0x805C14C0 | ---------- | ---------- |      340 |     1.0.0| u V   |    15.83 |
| * SERAM0 |  CM0+  | ---------- | 0x000000C0 | ---------- | ---------- |    90360 |   1.104.0| ------|     0.00 |
|   SERAM1 |  CM0+  | ---------- | 0x00020AC0 | ---------- | ---------- |    90360 |   1.104.0| ------|     0.00 |
+----------+--------+------------+------------+------------+------------+----------+-----------+-------+----------+
Legend: (u)(C)ompressed,(L)oaded,(V)erified,(s)kipped verification,(B)ooted,(E)ncrypted,(D)eferred


[SES] SE frequency is 100.85 MHz
```

SE-UART Output Fields

The output from the SE-UART shows what SES has processed during the boot sequence of the device.

SEROM

Shows the version of the SEROM software and the SoC revision register contents.

Banner

Shows the version of the SES software.

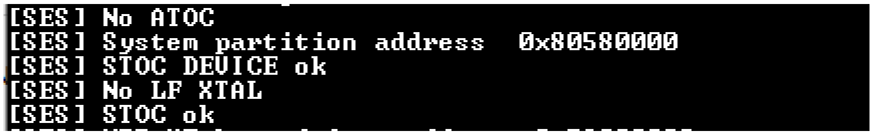
LCS (Life Cycle State)

Shows the LCS state of the device.

[SES] Output

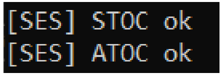
Shows the state of processing the STOC and ATOC images found in MRAM.

Device configuration



```
[SES] No ATOC
[SES] System partition address   0x80580000
[SES] STOC DEVICE ok
[SES] No LF XTAL
[SES] STOC ok
```

Other images processing –



```
[SES] STOC ok
[SES] ATOC ok
```

[SES] Table

Shows the details of all TOC objects processed from MRAM.

Name	ASCII string name from the JSON file.
- DEVICE	Shows a DEVICE Config object.
- SERAM0/1	Shows the two Banks of SES software. An asterisk ('*') denotes which Bank was booted from.
- HE_DBG	Shows the default Debug stub executed if no ATOC is present.
CPU	Which CPU is being targeted.
Store Address	Where in MRAM is this object stored.
Obj Address	TOC address
Dest Addr	Where in RAM is this object being copied to.
Boot Addr	What address is being used to Boot from.
Size	Size of the binary object
Version	Version ID specified in the JSON file.
Flags	
C	Image is Compressed
u	Image is Uncompressed

L	Image is LOADED to a RAM location
V	Image is VERIFIED
s	Image is SKIPPED
B	Image has been BOOTED
D	Image is DEFERRED, Booting will happen later
Time	Time in microseconds from the start of TOC processing until completion, this could include booting.

Deferred images will not display their details in the initial SES Banner print. The reason is that SES SKIPS these images on boot and so there are no details.

Once the object is Un deferred then the details can be displayed.

SES Boot Error messages

SES can report the following Boot loader messages:

`BL_STATUS_OK`	`0x00`
`BL_ERROR_APP_INVALID_TOC_ADDRESS`	`0x01`
`BL_ERROR_APP_INVALID_TOC_OFFSET`	`0x02`
`BL_ERROR_UNALIGNED_ADDRESS`	`0x03`
`BL_ERROR_INVALID_TOC_ADDRESS_RANGE`	`0x04`
`BL_ERROR_INVALID_TOC_FLAGS`	`0x05`
`BL_ERROR_INVALID_IMAGE_DATA`	`0x06`
`BL_ERROR_CERTIFICATE_NO_VERIFY_IN_MEMORY`	`0x07`
`BL_ERROR_CERTIFICATE_NO_VERIFY_IN_FLASH`	`0x08`
`BL_ERROR_CERTIFICATE_INVALID_LOAD_ADDRESS`	`0x09`
`BL_ERROR_CERTIFICATE_INVALID_CHAIN`	`0x0A`
`BL_ERROR_CERTIFICATE_STORAGE_ADDRESS_INVALID`	`0x0B`
`BL_ERROR_DEVICE_ADDRESS_INVALID`	`0x0C`
`BL_ERROR_UNCOMPRESS_FAILED`	`0x0D`
`BL_ERROR_SIGNATURE_VERIFY_FAILED`	`0x0E`
`BL_ERROR_APP_ACCESSING_PROTECTED_AREA`	`0x0F`
`BL_ERROR_ICV_ACCESSING_PROTECTED_AREA`	`0x10`
`BL_ERROR_FAILED_TOC_CRC32`	`0x11`
`BL_ERROR_INVALID_TOC`	`0x12`
`BL_ERROR_EXCEED_MAXIMUM_TOC_ENTRIES`	`0x13`
`BL_ERROR_NOT_IMAGE_NOT_DEVICE_CFG`	`0x14`
`BL_ERROR_INVALID_TOC_ENTRY_ID`	`0x15`
`BL_ERROR_INVALID_CPU_ID`	`0x16`
`BL_ERROR_ENTRY_NOT_SIGNED`	`0x17`
`BL_ERROR_LOAD_TO_MRAM_NOT_ALLOWED`	`0x18`
`BL_ERROR_NO_FREE_SLOTS`	`0x19`
`BL_ERROR_INVALID_M55_BOOT_ADDRESS`	`0x1A`
`BL_TOC_OBJECT_NOT_FOUND`	`0x1B`
`BL_TOC_OBJECT_FOR_CPU_NOT_FOUND`	`0x1C`
`BL_TOC_IMAGE_NOT_BOOTABLE`	`0x1D`
`BL_TOC_IMAGE_NOT_FOUND`	`0x1E`
`BL_ERROR_COMPRESSION_NOT_SUPPORTED`	`0x1F`
`BL_TOC_IMAGE_DEVICE_MISMATCH`	`0x20`
`BL_ERROR_UPD_SIGNATURE_INCORRECT`	`0x21`
`BL_ERROR_UPD_IMG_IN_MRAM_NOT_SUPPORTED`	`0x22`

Firewall Exception messages

Firewall Components can raise exceptions at run time when they reject bus transactions because they didn’t pass the various checks. Such exceptions are reported on the SES output screen.

Firewall exception from FC13, caused by Master ID 0x11 at address 0x80580000, transaction properties 0x00020000

Meaning:

FC <number>	Firewall component which generated the exception.
Master ID	The Corstone Master ID that issued the transaction causing the exception. It can be an APP CPU, but can also be another Corstone master like a DMA controller, USB host, etc.
Address	The address targeted by the transaction.

Transaction properties are the value of the Corstone register FE_TP. The bits in it indicate the transaction properties read/write, data/instruction, unprivileged/privileged, and secure/non-secure.

Firewall Configuration Errors

On Boot up you may see messages such as:

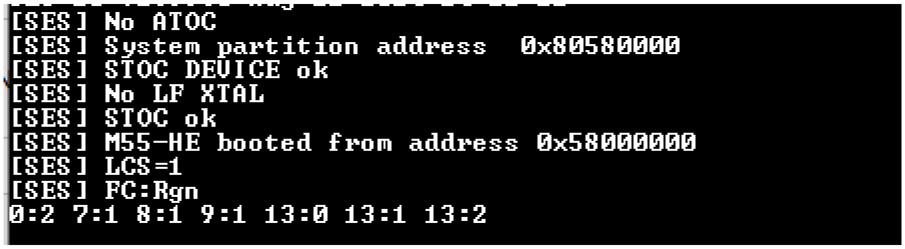
```
[WARNING] ATOC firewall configuration skipped: FC:11 Rgn:1 error=3
```

This indicates that some firewall region configuration fragments in the ATOC device configuration files was rejected for some reason. The error code indicates the reason.

Error	Definition	Description
1	FW_CFG_ERR_MASTER_ID	The region uses the Secure Enclave’s Master ID (0), which is not allowed
2	FW_CFG_ERR_ICV_CONFIG	The region is configured by the SE firmware and cannot be reconfigured
3	FW_CFG_ERR_ICV_PROT	The region overlaps with an Alif protected area (partially or fully)
4	FW_CFG_ERR_REGION_SIZE	Invalid region size code. The maximum region size code is 32 (4GB)

Error messages – no ATOC in MRAM

If there is no ATOC in MRAM, the following error messages are displayed -



```
[SES] No ATOC
[SES] System partition address  0x80580000
[SES] STOC DEVICE ok
[SES] No LF XTAL
[SES] STOC ok
[SES] M55-HE booted from address 0x58000000
[SES] LCS=1
[SES] FC:Rgn
0:2 7:1 8:1 9:1 13:0 13:1 13:2
```

Any other errors during the boot process will be reported.

Application Table of Contents (ATOC)

The default Application TOC package contains a Factory image, which Blinks the LEDs on the Development board.

Our releases consist of:

- M55_HE_TCM LED Blinky program.

This code is stored in MRAM and copied to a RAM location (The TCM for the M55_HE_CPU) and booted.

To try the Blinky program

```
$ cd <release-location>
$ app-write-mram
```

To rebuild this ATOC package

```
$ cd <release-location>
$ app-gen-toc -f build/config/app-cfg.json
$ app-write-mram
```

NOTE: When using an E1C or B1C device you must use the app-cfg-1c.json file.

ATOC Placed Images

ATOC packages allow for the absolute placement of user images in MRAM. See “mramAddress” option in the `app-gen-toc` configuration file.

Users must ensure that these images conform to the MRAM alignment rules and always be 16-byte aligned.

Images that are managed by the ATOC tool i.e., not placed at an absolute address are guaranteed to be aligned correctly.

NTOC Images

NTOC (No TOC) images are Execute in place (XIP) images stored at the base address of MRAM.

M55_HE is the *only* CPU that executes these images.

These images are not generated by the Security Toolkit i.e., they are not ATOC images as they have restrictions such as they cannot be signed.

SES uses the following algorithm to process these objects:

```
IF ATOC is present
   Process ATOC and Boot
ELSE
  IF MTOC is present
    Process MTOC and Boot
ELSE
  IF (0x80000000 and 0x80000004 has valid $SP and $PC)
      ReleaseM55_HE
  ELSE
      Load STOC Debug stub (if present)
```

SES reads the first two 32-bits of MRAM:

- Word1 = $SP (Stack Pointer)
- Word2 = $PC (Program Counter)

Then uses the following logic:

- Is the Program counter in the valid Application address range of MRAM?
- Is the Stack Pointer in a valid RAM memory range?

If these above criteria are not met, then the image will not be released (booted).

NOTE: MTOC support was dropped in CMSIS release V0.9.1.

Change Sets

Change Sets are used to allow for Static initialization of registers at Boot up time.

This is known as Static device configuration. Change Set data is processed and executed as part of the TOC processing. A User could use this for peripheral / register / memory initialization without the necessity to write any code. Pin muxing, Firewall, Clock, Event, and Interrupt masking can be programmed using this method.

Change Sets can be added to the STOC packages (by ALIF only) and to the Application TOC packages by Users.

What is a Change Set?

A Change Set is a group of 32-bit values representing an Address, a Mask, and a Value. With these three elements, any memory or register address can be modified using Change Sets.

Adding a Change Set

If we take the application configuration file app-cfg.json

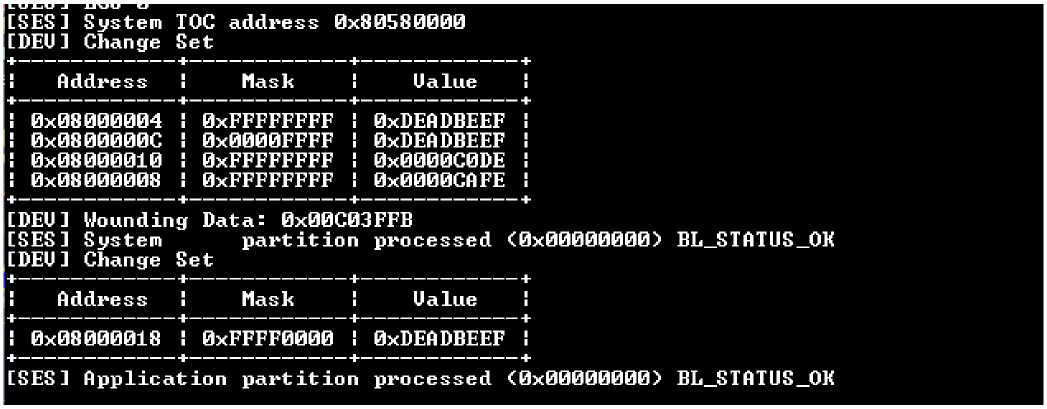
```
{
	"BLINK-HE": {
		"binary": "m55_blink_he.bin",
		"version" : "1.0.0",
		"cpu_id": "M55_HE",
		"loadAddress": "0x60000000",
		"flags": ["load", "boot"]
	}
},
	"DEVICE": {
		"binary": "device-config.json",
		"version" : "0.5.00"
},
```

Here we have added a DEVICE configuration section.
The device-config.json file is as follows:

```
{
    "clocks": {
        "ChangeSets" : [
        {
            "address" : "0x08000018",
            "mask" : "0xFFFFFFFF",
            "value" : "0xdeadbeef"
        }
        ]
}
```

For this example, we are adding a Change Set to a location in Global memory SRAM_1 (0x080000018).

When SES boots it will process the DEVICE Configuration section in MRAM



```
[SES] System TOC address 0x80580000
[DEV] Change Set
+------------+------------+------------+
¦   Address  ¦    Mask    ¦    Value   ¦
+------------+------------+------------+
¦ 0x08000004 ¦ 0xFFFFFFFF ¦ 0xDEADBEEF ¦
¦ 0x0800000C ¦ 0x0000FFFF ¦ 0xDEADBEEF ¦
¦ 0x08000010 ¦ 0xFFFFFFFF ¦ 0x0000C0DE ¦
¦ 0x08000008 ¦ 0xFFFFFFFF ¦ 0x0000CAFE ¦
+------------+------------+------------+
[DEV] Wounding Data: 0x00C03FFB
[SES] System      partition processed (0x00000000) BL_STATUS_OK
[DEV] Change Set
+------------+------------+------------+
¦   Address  ¦    Mask    ¦    Value   ¦
+------------+------------+------------+
¦ 0x08000018 ¦ 0xFFFF0000 ¦ 0xDEADBEEF ¦
+------------+------------+------------+
[SES] Application partition processed (0x00000000) BL_STATUS_OK
```

In this example, there are some DEVICE configurations for the ALIF System TOC then the Application TOC is processed we see the Change Set being applied.

Example – Setting the Ethernet external Clock

The Ethernet external clock can be enabled using the following Change Set configuration.

```
"ChangeSets" : [
     {
        "address" : "0x71007408",
        "mask" : "0x00000001",
        "value" : "0x00000001"
     }
]
```

After SERAM processes the above Change Set, the Ethernet external clock will be enabled.

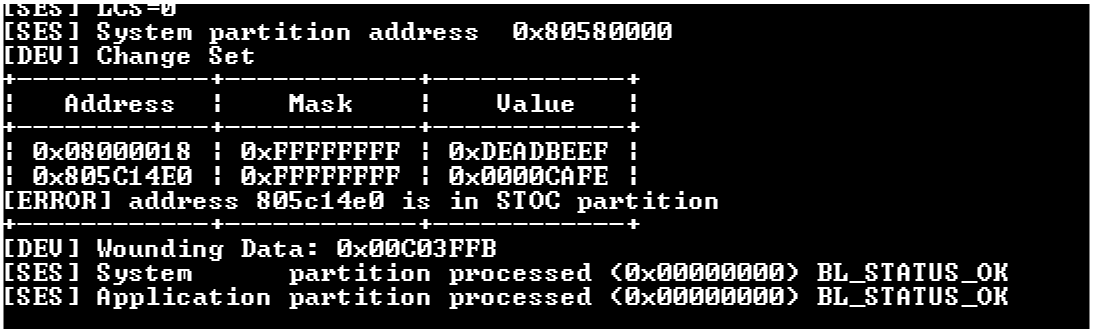
Error handling with a Change Set

There are some checks done within SES for addresses added through Changesets. These include:

- Checking the address is not within the ALIF MRAM area.

Invalid address in STOC partition

In the example below an attempt is made to write to the STOC MRAM area



```
[SES] LCS=0
[SES] System partition address  0x80580000
[DEV] Change Set
+------------+------------+------------+
¦   Address  ¦    Mask    ¦    Value   ¦
+------------+------------+------------+
¦ 0x08000018 ¦ 0xFFFFFFFF ¦ 0xDEADBEEF ¦
¦ 0x805C14E0 ¦ 0xFFFFFFFF ¦ 0x0000CAFE ¦
[ERROR] address 805c14e0 is in STOC partition
+------------+------------+------------+
[DEV] Wounding Data: 0x00C03FFB
[SES] System      partition processed (0x00000000) BL_STATUS_OK
[SES] Application partition processed (0x00000000) BL_STATUS_OK
```

4. ISP (In System Programming)

ISP is a mechanism for connecting to the Secure Enclave using the Secure Enclave UART (SE-UART) and performing various operations with the device.

Please ensure the prerequisite python packages are installed before starting. See the installation instructions in the Alif Security Toolkit Quick Start Guide.

SE-UART Connection

UART Settings

Baud Rate	55000 (default)
Data	8 bits
Parity	None
Stop Bits	1 bit
Flow control	None

Dynamic Baud Rate Change

Tools updating MRAM (updateSystemPackage, app-write-mram) will increase the baud rate from the default value to 921600 baud. Once the operation is completed the Baud rate is set back to the default value.

Use the ‘-s’ option to override this dynamic change facility.

SES supports MRAM updates at the standard HF RC/XO frequencies, with baud rate bumping. I.e., no need to specify '-s'. If the device is running at a non-standard frequency, via the 1/X or 1/Z dividers - the MRAM updates should be done via the '-s' switch.

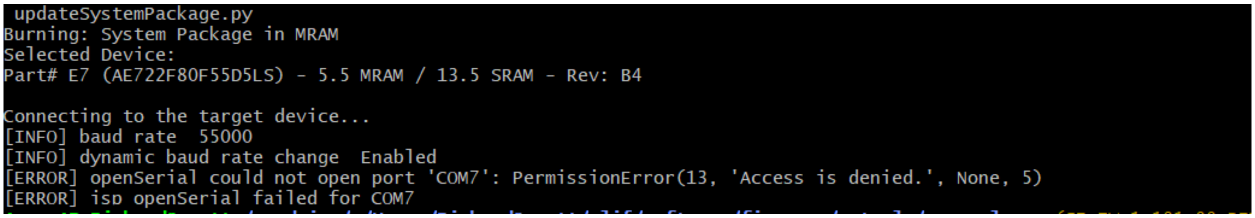
Baud Rate Override

The ISP method uses a default baud rate. For REV_A1 devices this is 100000.
You can override the default baud rate by doing either:

- Edit the local configuration file isp_config_data.cfg and change baudrate.
- Use the -b option e.g. -b 55000

All ISP tools have this option.

UART Errors – PermissionError



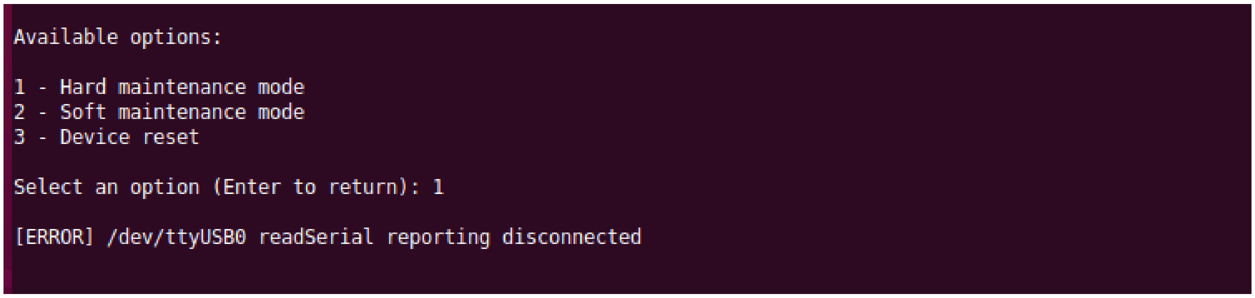
```
 updateSystemPackage.py
Burning: System Package in MRAM
Selected Device:
Part# E7 (AE722F80F55D5LS) - 5.5 MRAM / 13.5 SRAM - Rev: B4

Connecting to the target device...
[INFO] baud rate  55000
[INFO] dynamic baud rate change  Enabled
[ERROR] openSerial could not open port 'COM7': PermissionError(13, 'Access is denied.', None, 5)
[ERROR] isp openSerial failed for COM7
```

There is only one SE-UART on the device. If you are using ISP, please ensure you have no other Tera term or putty sessions using the same SE-UART. The following shows the output if the SE-UART is already being used by another program:

This indicates that the UART is already being used (e.g., a Tera-Term session is still running).

UART Errors – Disconnected

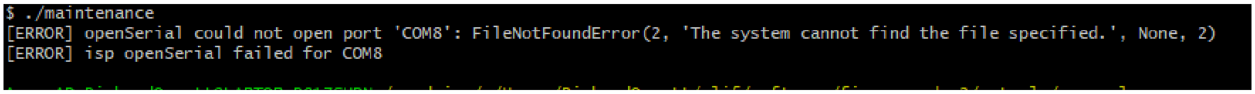
This error occurs when a Disconnect event occurs such as Powering off the FDTI UART whilst in an ISP operation.



```
Available options:

1 - Hard maintenance mode
2 - Soft maintenance mode
3 - Device reset

Select an option (Enter to return): 1

[ERROR] /dev/ttyUSB0 readSerial reporting disconnected
```

In this example, we are in Hard Maintenance mode waiting for input from the Target board. The UART power is turned off and we will see the following error message printed that there has been a disconnection event.

UART Errors – FileNotFoundError



```
$ ./maintenance
[ERROR] openSerial could not open port 'COM8': FileNotFoundError(2, 'The system cannot find the file specified.', None, 2)
[ERROR] isp openSerial failed for COM8
```

This error usually indicates that your target system is not powered.

ISP Discovery

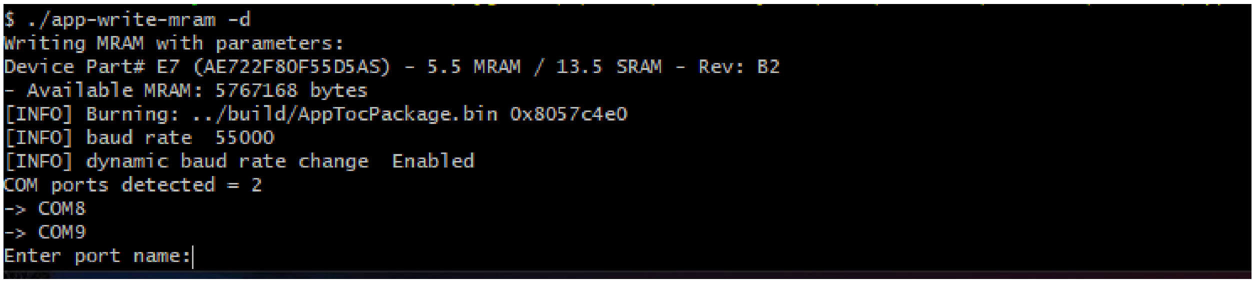
This command discovers the available Serial communication ports. The first time you execute an SETOOLS script you will be prompted for the required serial port.

When the ports are presented, just enter the port name and press [ENTER]. Just pressing [ENTER] will pick the last item in the list.

This port data is saved in a local configuration file (isp_config_data.cfg).

The next time an SETOOLS command is invoked if this configuration file is present, it will use the parameters from this file.

To override this option simply use the -d option:



```
$ ./app-write-mram -d
Writing MRAM with parameters:
Device Part# E7 (AE722F80F55D5AS) - 5.5 MRAM / 13.5 SRAM - Rev: B2
- Available MRAM: 5767168 bytes
[INFO] Burning: ../build/AppTocPackage.bin 0x8057c4e0
[INFO] baud rate  55000
[INFO] dynamic baud rate change  Enabled
COM ports detected = 2
-> COM8
-> COM9
Enter port name:
```

This will force a re-discovery of the Serial ports.
All the MRAM ISP tools have the discovery option which will prompt you for the serial port.

ISP when in Low power mode

When an application has put the device into STOP power mode, ISP will not be operational (the device is Powered off).

5. Tool Flow

The tools allow a user to provision the device.

This includes programming MRAM with application binaries for A32 and M55 embedded processors available for user applications.

a. Use `tools-config` to review and modify current tool options.

b. Generate RoT with app-gen-rot.py only once (or anytime new keys are required) – **Not required** as this release packet already contains a sample RoT and corresponding keys and certificates, but it can be used if user wants to re-generate the RoT.

c. Edit `build/config/app-cfg.json` file (or a json file with other name) with images to include in the desired **AppTocPackage.bin** image and copy the binaries into the build/ directory. Run app-gen-toc.py to generate the package.

d. Power on the board and connect the Serial interface for the following steps.

  d.1 (Optional) Use updateSystemPackage.py to update latest Alif Secure Enclave firmware.

  d.2 Use app-write-mram.py to upload the AppTocPackage.bin image into the MRAM.

    d.2.1 (Optional) Reset the board to reboot the chip to verify that the SERAM image loads and boots properly.

Debug Stubs

To enable connection with the debugger such as ARM-DS, the target core needs to be running.

This is achieved by running debug stubs on the cores. If the M55_HE Application core does not have an application loaded for it a debug stub will be installed for it. This is done automatically by SES if no Application for the M55_HE is detected. This is not a TOC based debug stub and so does not appear in any Table of contents enquiries. (NOTE: It is important by wiring instructions directly to the TCM memory of the M55_HE and releasing the core)

For the other APP cores, the user must instantiate debug stubs by entering them in an ATOC configuration file.

You can create an ATOC image with 3 SRAM debug stubs for the 3 CPU cores by running

```
app-gen-toc -f build/config/app-cpu-stubs.json
```

Then erase the application part of MRAM by running

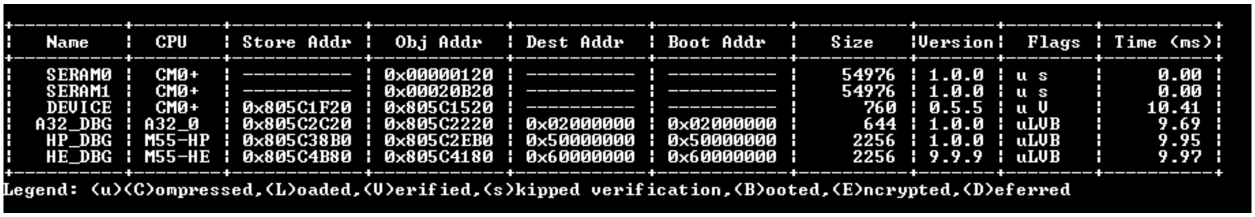
```
app-write-mram -e app
```

And then write the image with 3 debug stubs to MRAM by running

```
app-write-mram
```

Details of these programs are described below.

The following is the SE-UART output on boot when the Application MRAM is not programmed.



```
+-----------+---------+------------+------------+------------+------------+------------+-------+----------+-----------+
¦   Name    ¦   CPU   ¦ Store Addr ¦  Obj Addr  ¦ Dest Addr  ¦ Boot Addr  ¦    Size    ¦Version¦  Flags   ¦ Time (ms)¦
+-----------+---------+------------+------------+------------+------------+------------+-------+----------+-----------+
¦   SERAM0  ¦  CM0+   ¦ ---------- ¦ 0x00000120 ¦ ---------- ¦ ---------- ¦      54976 ¦ 1.0.0 ¦ u s      ¦      0.00 ¦
¦   SERAM1  ¦  CM0+   ¦ ---------- ¦ 0x00020B20 ¦ ---------- ¦ ---------- ¦      54976 ¦ 1.0.0 ¦ u s      ¦      0.00 ¦
¦   DEVICE  ¦  CM0+   ¦ 0x805C1F20 ¦ 0x805C1520 ¦ ---------- ¦ ---------- ¦        760 ¦ 0.5.5 ¦ u V      ¦     10.41 ¦
¦  A32_DBG  ¦ A32_0   ¦ 0x805C2C20 ¦ 0x805C2220 ¦ 0x02000000 ¦ 0x02000000 ¦        644 ¦ 1.0.0 ¦ uLVB     ¦      9.69 ¦
¦   HP_DBG  ¦ M55-HP  ¦ 0x805C38B0 ¦ 0x805C2EB0 ¦ 0x50000000 ¦ 0x50000000 ¦       2256 ¦ 1.0.0 ¦ uLVB     ¦      9.95 ¦
¦   HE_DBG  ¦ M55-HE  ¦ 0x805C4B80 ¦ 0x805C4180 ¦ 0x60000000 ¦ 0x60000000 ¦       2256 ¦ 9.9.9 ¦ uLVB     ¦      9.97 ¦
+-----------+---------+------------+------------+------------+------------+------------+-------+----------+-----------+
Legend: (u)(C)ompressed,(L)oaded,(V)erified,(s)kipped verification,(B)ooted,(E)ncrypted,(D)eferred
```

The debug stubs, denoted by the _DBG names, are loaded by SES for each relevant Application core.

MRAM Burners

Two tools used to write MRAM

- `updateSystemPackage`
- `app-write-mram`

One will update the System TOC (STOC) and the other will update the Users Application MRAM area.

When using a generated TOC package this will be guaranteed to correctly aligned and padded for writing to MRAM.

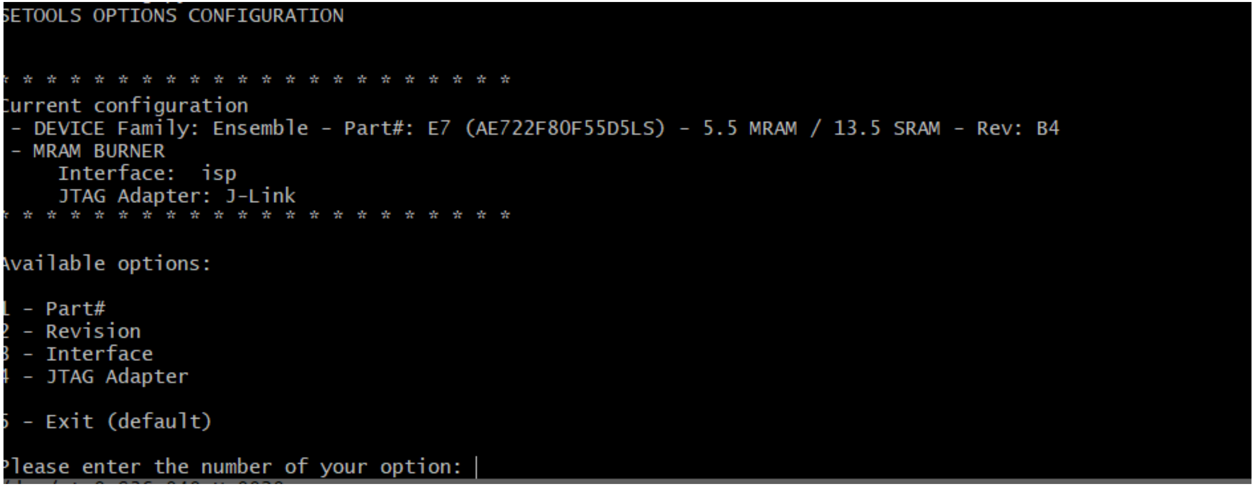
In the case of User supplied images the burning tool will warn of any alignment issues such as not being aligned on 16-byte boundaries. The user will need to correct the issue(s) and re-run the burning tool.

Both tools require that SES is running on the Target device. If you are in Recovery mode, these tools will not work as the ISP protocol command sets are different between SEROM and SERAM.

6. Running the Tools

tools-config tool

From the command prompt.

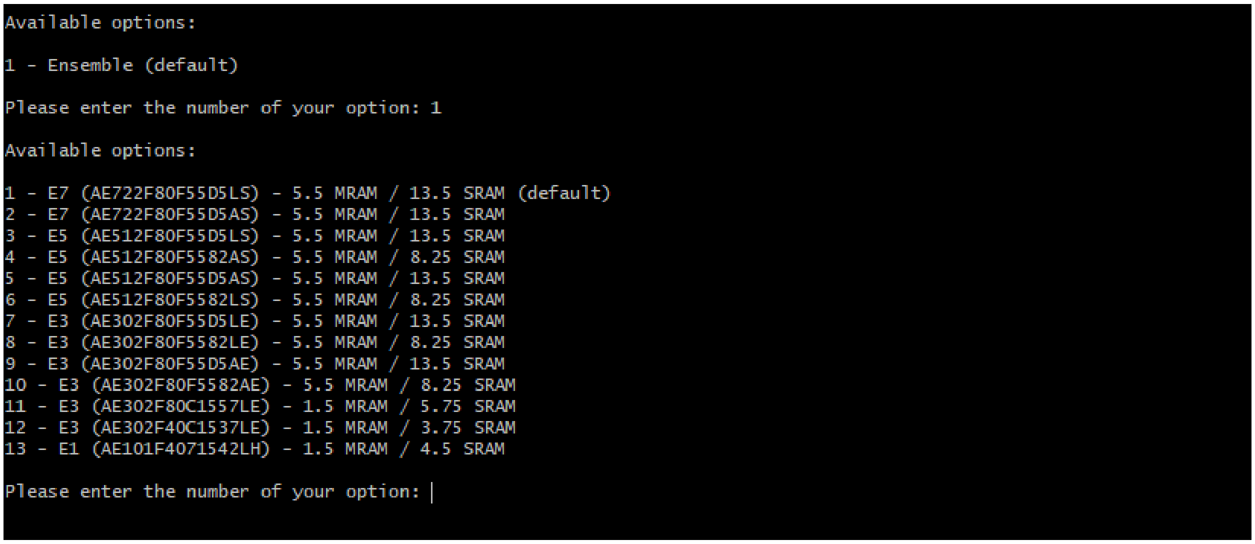


```
SETOOLS OPTIONS CONFIGURATION

* * * * * * * * * * * * * * * * * * * * * *
Current configuration
 - DEVICE Family: Ensemble - Part#: E7 (AE722F80F55D5LS) - 5.5 MRAM / 13.5 SRAM - Rev: B4
 - MRAM BURNER
     Interface:  isp
     JTAG Adapter: J-Link
* * * * * * * * * * * * * * * * * * * * * *

Available options:

1 - Part#
2 - Revision
3 - Interface
4 - JTAG Adapter

5 - Exit (default)

Please enter the number of your option: |
```

```
$ tools-config
```

This script will show the current tool options and allow for their modification.

Press the menu number to show available options.



```
Available options:

1 - Ensemble (default)

Please enter the number of your option: 1

Available options:

1 - E7 (AE722F80F55D5LS) - 5.5 MRAM / 13.5 SRAM (default)
2 - E7 (AE722F80F55D5AS) - 5.5 MRAM / 13.5 SRAM
3 - E5 (AE512F80F55D5LS) - 5.5 MRAM / 13.5 SRAM
4 - E5 (AE512F80F5582AS) - 5.5 MRAM / 8.25 SRAM
5 - E5 (AE512F80F55D5AS) - 5.5 MRAM / 13.5 SRAM
6 - E5 (AE512F80F5582LS) - 5.5 MRAM / 8.25 SRAM
7 - E3 (AE302F80F55D5LE) - 5.5 MRAM / 13.5 SRAM
8 - E3 (AE302F80F5582LE) - 5.5 MRAM / 8.25 SRAM
9 - E3 (AE302F80F55D5AE) - 5.5 MRAM / 13.5 SRAM
10 - E3 (AE302F80F5582AE) - 5.5 MRAM / 8.25 SRAM
11 - E3 (AE302F80C1557LE) - 1.5 MRAM / 5.75 SRAM
12 - E3 (AE302F40C1537LE) - 1.5 MRAM / 3.75 SRAM
13 - E1 (AE101F4071542LH) - 1.5 MRAM / 4.5 SRAM

Please enter the number of your option: |
```

Choose the desired option pressing the option number (or press Enter key to choose the default one, shown in parentheses).

Once the tool options are configured, the rest of the tools will take these options automatically.

(Note: every time a new Part Number or device Revision is changed, it’s recommended to re-generate the APP TOC Package using the app-toc.py tool).

Tools-config command line mode:

Besides the interactive mode explained above (with menus and selections), the tool also allows the user to select the Part# (option 1 in the menu) and Revision (option 2) using the following command line options:

- -p PART# (use double quotes). Examples:
  - -p "E7 (AE722F80F55D5AE) - 5.5 MRAM / 13.5 SRAM"
  - -p "E1 (AE101F4071542UE) - 1.5 MRAM / 4.25 SRAM"
  - -p "e7 (AE722F80F55D5AE) - 5.5 mRAM / 13.5 srAM"
- -r REVISION. Examples:
  - -r B0
  - -r B2
  - -r B4

app-gen-rot tool

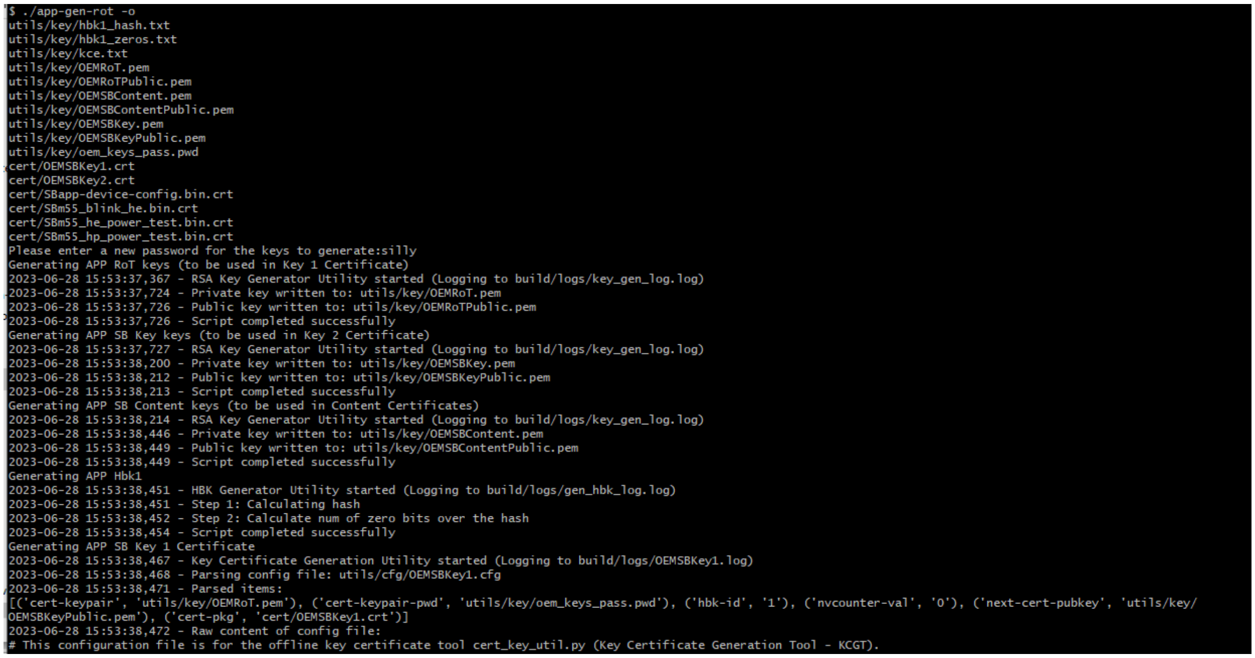
From the command prompt.

```
$ app-gen-rot (-h for help)
```

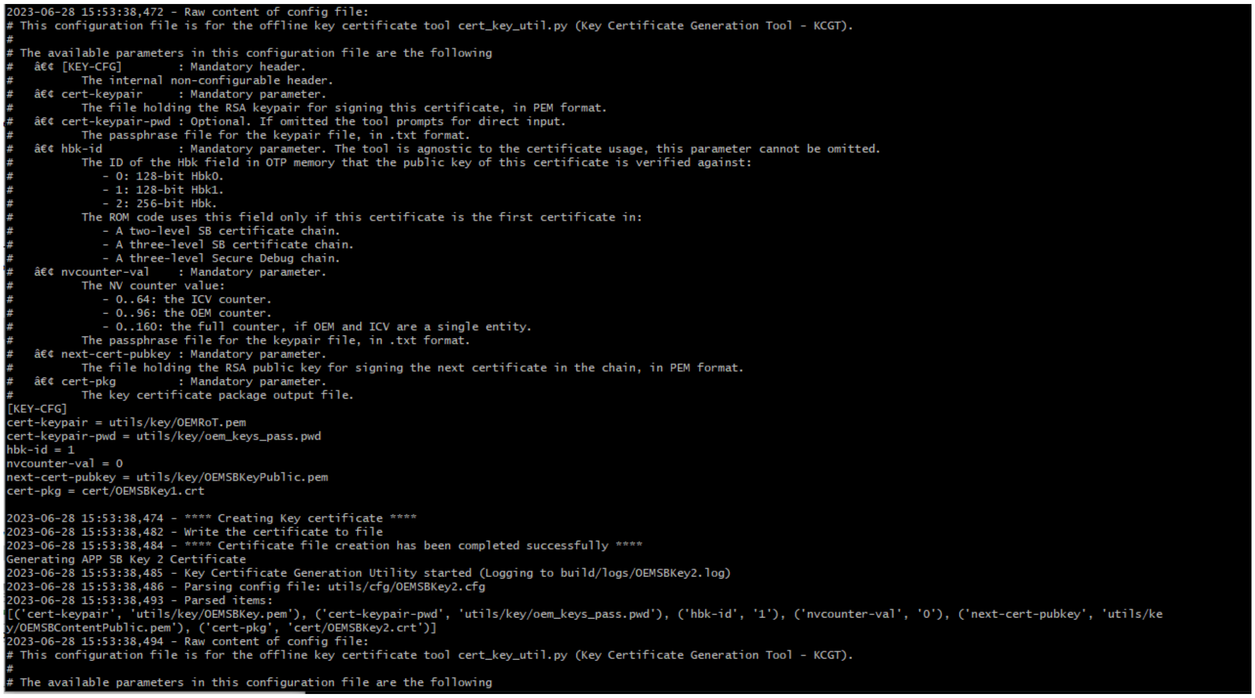
The Security Tools already have a development key embedded.

You will need to use the -o, --overwrite flag to create a new one.

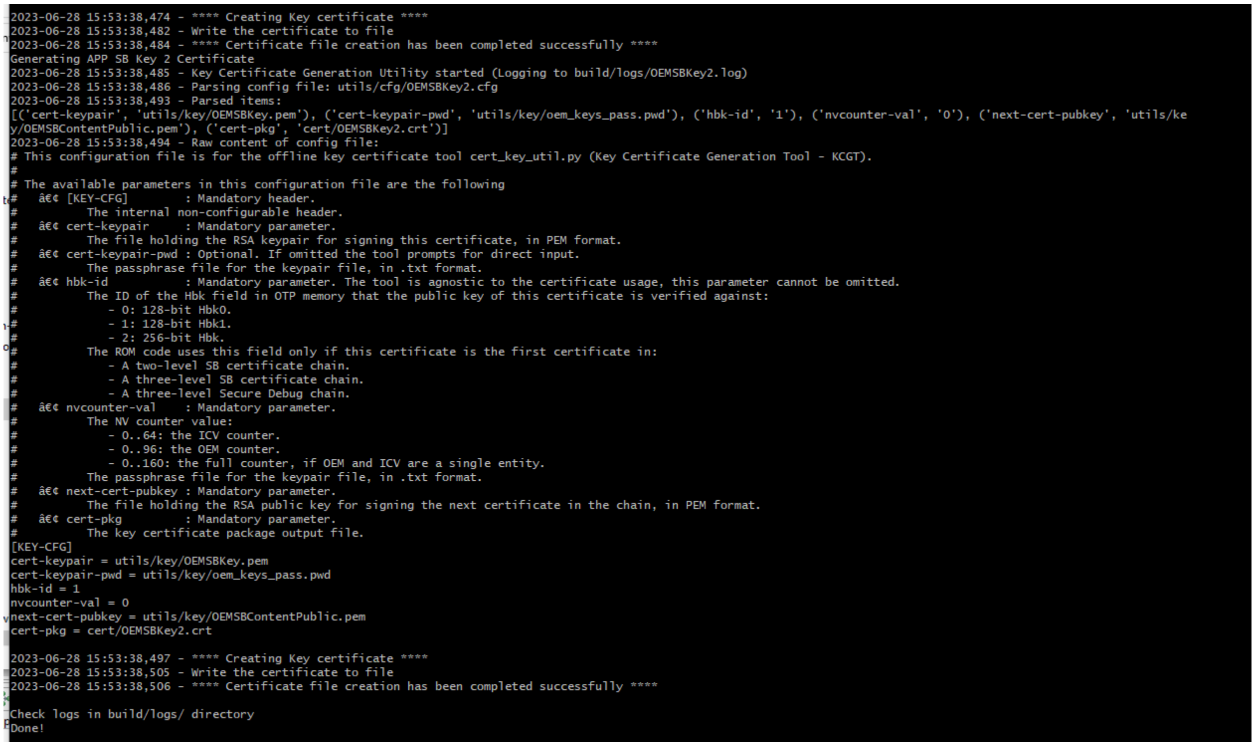
The tool will prompt the user to enter a password to encrypt the generated keys before saving them in the disk. The password will be saved as ‘oem_keys_pass.pwd ‘and it will be needed for each operation that requires reading the encrypted keys.



```
$ ./app-gen-rot -o
utils/key/hbk1_hash.txt
utils/key/hbk1_zeros.txt
utils/key/kce.txt
utils/key/OEMRoT.pem
utils/key/OEMRoTPublic.pem
utils/key/OEMSBContent.pem
utils/key/OEMSBContentPublic.pem
utils/key/OEMSBKey.pem
utils/key/OEMSBKeyPublic.pem
utils/key/oem_keys_pass.pwd
cert/OEMSBKey1.crt
cert/OEMSBKey2.crt
cert/SBapp-device-config.bin.crt
cert/SBm55_blink_he.bin.crt
cert/SBm55_he_power_test.bin.crt
cert/SBm55_hp_power_test.bin.crt
Please enter a new password for the keys to generate:silly
Generating APP RoT keys (to be used in Key 1 Certificate)
2023-06-28 15:53:37,367 - RSA Key Generator Utility started (Logging to build/logs/key_gen_log.log)
2023-06-28 15:53:37,724 - Private key written to: utils/key/OEMRoT.pem
2023-06-28 15:53:37,726 - Public key written to: utils/key/OEMRoTPublic.pem
2023-06-28 15:53:37,726 - Script completed successfully
Generating APP SB Key keys (to be used in Key 2 Certificate)
2023-06-28 15:53:37,727 - RSA Key Generator Utility started (Logging to build/logs/key_gen_log.log)
2023-06-28 15:53:38,200 - Private key written to: utils/key/OEMSBKey.pem
2023-06-28 15:53:38,212 - Public key written to: utils/key/OEMSBKeyPublic.pem
2023-06-28 15:53:38,213 - Script completed successfully
Generating APP SB Content keys (to be used in Content Certificates)
2023-06-28 15:53:38,214 - RSA Key Generator Utility started (Logging to build/logs/key_gen_log.log)
2023-06-28 15:53:38,446 - Private key written to: utils/key/OEMSBContent.pem
2023-06-28 15:53:38,449 - Public key written to: utils/key/OEMSBContentPublic.pem
2023-06-28 15:53:38,449 - Script completed successfully
Generating APP Hbk1
2023-06-28 15:53:38,451 - HBK Generator Utility started (Logging to build/logs/gen_hbk_log.log)
2023-06-28 15:53:38,451 - Step 1: Calculating hash
2023-06-28 15:53:38,452 - Step 2: Calculate num of zero bits over the hash
2023-06-28 15:53:38,454 - Script completed successfully
Generating APP SB Key 1 Certificate
2023-06-28 15:53:38,467 - Key Certificate Generation Utility started (Logging to build/logs/OEMSBKey1.log)
2023-06-28 15:53:38,468 - Parsing config file: utils/cfg/OEMSBKey1.cfg
2023-06-28 15:53:38,471 - Parsed items:
[('cert-keypair', 'utils/key/OEMRoT.pem'), ('cert-keypair-pwd', 'utils/key/oem_keys_pass.pwd'), ('hbk-id', '1'), ('nvcounter-val', '0'), ('next-cert-pubkey', 'utils/key/
OEMSBKeyPublic.pem'), ('cert-pkg', 'cert/OEMSBKey1.crt')]
2023-06-28 15:53:38,472 - Raw content of config file:
# This configuration file is for the offline key certificate tool cert_key_util.py (Key Certificate Generation Tool - KCGT).
```



```
2023-06-28 15:53:38,472 - Raw content of config file:
# This configuration file is for the offline key certificate tool cert_key_util.py (Key Certificate Generation Tool - KCGT).
#
# The available parameters in this configuration file are the following
#   â€¢ [KEY-CFG]          : Mandatory header.
#          The internal non-configurable header.
#   â€¢ cert-keypair       : Mandatory parameter.
#          The file holding the RSA keypair for signing this certificate, in PEM format.
#   â€¢ cert-keypair-pwd : Optional. If omitted the tool prompts for direct input.
#          The passphrase file for the keypair file, in .txt format.
#   â€¢ hbk-id             : Mandatory parameter. The tool is agnostic to the certificate usage, this parameter cannot be omitted.
#          The ID of the Hbk field in OTP memory that the public key of this certificate is verified against:
#             - 0: 128-bit Hbk0.
#             - 1: 128-bit Hbk1.
#             - 2: 256-bit Hbk.
#          The ROM code uses this field only if this certificate is the first certificate in:
#             - A two-level SB certificate chain.
#             - A three-level SB certificate chain.
#             - A three-level Secure Debug chain.
#   â€¢ nvcounter-val      : Mandatory parameter.
#          The NV counter value:
#             - 0..64: the ICV counter.
#             - 0..96: the OEM counter.
#             - 0..160: the full counter, if OEM and ICV are a single entity.
#          The passphrase file for the keypair file, in .txt format.
#   â€¢ next-cert-pubkey : Mandatory parameter.
#          The file holding the RSA public key for signing the next certificate in the chain, in PEM format.
#   â€¢ cert-pkg           : Mandatory parameter.
#          The key certificate package output file.
[KEY-CFG]
cert-keypair = utils/key/OEMRoT.pem
cert-keypair-pwd = utils/key/oem_keys_pass.pwd
hbk-id = 1
nvcounter-val = 0
next-cert-pubkey = utils/key/OEMSBKeyPublic.pem
cert-pkg = cert/OEMSBKey1.crt

2023-06-28 15:53:38,474 - **** Creating Key certificate ****
2023-06-28 15:53:38,482 - Write the certificate to file
2023-06-28 15:53:38,484 - **** Certificate file creation has been completed successfully ****
Generating APP SB Key 2 Certificate
2023-06-28 15:53:38,485 - Key Certificate Generation Utility started (Logging to build/logs/OEMSBKey2.log)
2023-06-28 15:53:38,486 - Parsing config file: utils/cfg/OEMSBKey2.cfg
2023-06-28 15:53:38,493 - Parsed items:
[('cert-keypair', 'utils/key/OEMSBKey.pem'), ('cert-keypair-pwd', 'utils/key/oem_keys_pass.pwd'), ('hbk-id', '1'), ('nvcounter-val', '0'), ('next-cert-pubkey', 'utils/ke
y/OEMSBContentPublic.pem'), ('cert-pkg', 'cert/OEMSBKey2.crt')]
2023-06-28 15:53:38,494 - Raw content of config file:
# This configuration file is for the offline key certificate tool cert_key_util.py (Key Certificate Generation Tool - KCGT).
#
# The available parameters in this configuration file are the following
```



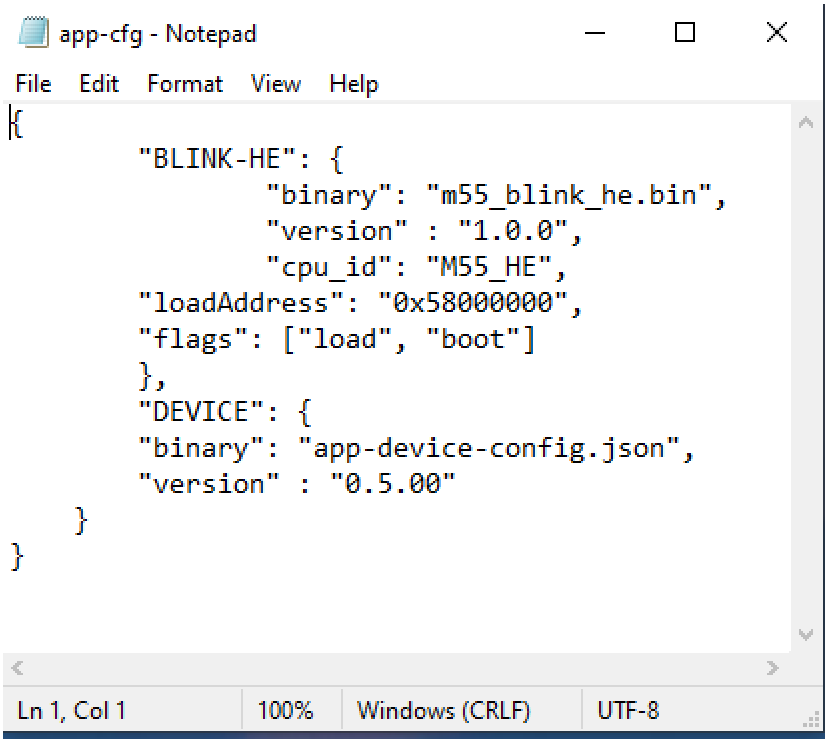
```
2023-06-28 15:53:38,474 - **** Creating Key certificate ****
2023-06-28 15:53:38,482 - Write the certificate to file
2023-06-28 15:53:38,484 - **** Certificate file creation has been completed successfully ****
Generating APP SB Key 2 Certificate
2023-06-28 15:53:38,485 - Key Certificate Generation Utility started (Logging to build/logs/OEMSBKey2.log)
2023-06-28 15:53:38,486 - Parsing config file: utils/cfg/OEMSBKey2.cfg
2023-06-28 15:53:38,493 - Parsed items:
[('cert-keypair', 'utils/key/OEMSBKey.pem'), ('cert-keypair-pwd', 'utils/key/oem_keys_pass.pwd'), ('hbk-id', '1'), ('nvcounter-val', '0'), ('next-cert-pubkey', 'utils/ke
y/OEMSBContentPublic.pem'), ('cert-pkg', 'cert/OEMSBKey2.crt')]
2023-06-28 15:53:38,494 - Raw content of config file:
# This configuration file is for the offline key certificate tool cert_key_util.py (Key Certificate Generation Tool - KCGT).
#
# The available parameters in this configuration file are the following
#   â€¢ [KEY-CFG]          : Mandatory header.
#          The internal non-configurable header.
#   â€¢ cert-keypair       : Mandatory parameter.
#          The file holding the RSA keypair for signing this certificate, in PEM format.
#   â€¢ cert-keypair-pwd : Optional. If omitted the tool prompts for direct input.
#          The passphrase file for the keypair file, in .txt format.
#   â€¢ hbk-id             : Mandatory parameter. The tool is agnostic to the certificate usage, this parameter cannot be omitted.
#          The ID of the Hbk field in OTP memory that the public key of this certificate is verified against:
#             - 0: 128-bit Hbk0.
#             - 1: 128-bit Hbk1.
#             - 2: 256-bit Hbk.
#          The ROM code uses this field only if this certificate is the first certificate in:
#             - A two-level SB certificate chain.
#             - A three-level SB certificate chain.
#             - A three-level Secure Debug chain.
#   â€¢ nvcounter-val      : Mandatory parameter.
#          The NV counter value:
#             - 0..64: the ICV counter.
#             - 0..96: the OEM counter.
#             - 0..160: the full counter, if OEM and ICV are a single entity.
#          The passphrase file for the keypair file, in .txt format.
#   â€¢ next-cert-pubkey : Mandatory parameter.
#          The file holding the RSA public key for signing the next certificate in the chain, in PEM format.
#   â€¢ cert-pkg           : Mandatory parameter.
#          The key certificate package output file.
[KEY-CFG]
cert-keypair = utils/key/OEMSBKey.pem
cert-keypair-pwd = utils/key/oem_keys_pass.pwd
hbk-id = 1
nvcounter-val = 0
next-cert-pubkey = utils/key/OEMSBContentPublic.pem
cert-pkg = cert/OEMSBKey2.crt

2023-06-28 15:53:38,497 - **** Creating Key certificate ****
2023-06-28 15:53:38,505 - Write the certificate to file
2023-06-28 15:53:38,506 - **** Certificate file creation has been completed successfully ****

Check logs in build/logs/ directory
Done!
```

Once finished, keys will be generated in *utils/key/*, certificates are generated in *cert/*, and logs in *build/logs/* folder.

app-gen-toc tool

To generate the APP TOC Package, the `build/config/app-cfg.json` file must be configured. Alternatively, a configuration file can be created with a different name and the tool’s “-f” option can cause that alternate file to be used as input to the program.

Here is an example configuration file:



```
{
        "BLINK-HE": {
                "binary": "m55_blink_he.bin",
                "version" : "1.0.0",
                "cpu_id": "M55_HE",
        "loadAddress": "0x58000000",
        "flags": ["load", "boot"]
        },
        "DEVICE": {
        "binary": "app-device-config.json",
        "version" : "0.5.00"
    }
}
```

This configuration file is in JSON format and must specify each of the binary images the user wants to include in the APP TOC Package. SERAM will process the images in the specified order (from top to bottom).

A JSON file may contain multiple objects, where each object is represented by a set of name-value pairs.

In the configuration file, each image must be specified as an object in the following way;

```
"IMAGE_IDENTIFIER": {
      "attribute_1": "value_1",
      "attribute_2": "value_2",
      ……………
      "attribute_n": "value_n"
}
```

Multiple images will be declared by multiple objects, separated by comma. For example;

```
{
   "IMAGE_1": {

   },
   "IMAGE_2": {

   },
   ………………
   "IMAGE_N": {

   }
}
```

The “IMAGE_IDENTIFIER” is an 8-character length (MAX) description intended to give a short description to the image. The tool will truncate or extend this field’s length as appropriate. This name ***should be unique*** per configuration file.

Inside each image object, different attributes can be used to define the parameters of the binary image. The following attributes are valid:

binary: [MANDATORY] - this attribute declares the name of the binary to be included in the APP TOC Package. It is assumed the binary exists in the build/images/ folder.

version: [OPTIONAL] – this attribute declares the version of the image, and it should follow the ‘X.Y.Z’ format. Example; ‘1.0.0’

loadAddress: [MANDATORY if LOAD flag is set] – this attribute specifies the loading address in (RAM) memory where the code will be executed from. Global Memory Addresses (in hexadecimal format) should be used for this attribute.

mramAddress: [OPTIONAL] – this attribute specifies the MRAM address location where the user wishes the image to reside, and addresses (in hexadecimal format) starting from 0x8000-0100 (for REV_Ax) can be used. This option is to support the “Fixed Address Image” or “absolute placement” use case, where the user needs to position the binary in a specific address in MRAM, so other running code can locate this image in the specified location (Linux use case). If this attribute is used with the BOOT flag, XIP (execution in place) is supported. If this flag is not set, then the image will reside at this address, with no further implication. If this attribute is NOT specified, the tool will define the MRAM position to allocate the binary in memory.

flags: [OPTIONAL] – the following flags can be used – example [“load”, “boot”]:

- LOAD – The image will be loaded into the specified memory address
- BOOT – The specified CPU_ID will be started to execute the image
- ENCRYPT – The image will be encrypted
- COMPRESS – The image will be compressed
- DEFERRED – The image will be skipped at boot time (i.e., no boot or load) and wait for a service request at runtime.

cpu_id: [OPTIONAL] – this attribute indicates the CPU to start once the binary is ready to be executed. Valid values are:

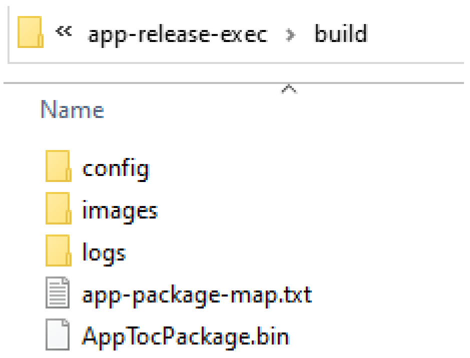
[A32_0, A32_1, A32_2, A32_3, M55_HP, M55_HE]

Note: Only one CPU_ID should be specified.

signed: [OPTIONAL] – this attribute is a binary value (true/false) specifying if the image is signed (“signed”: true) or unsigned (“signed”: false). In a typical case for Secure Boot, all images should be signed, so there is no need to specify this attribute. Only if the user wishes to specify an unsigned image, this attribute should be set as false.

disabled: [OPTIONAL] – this attribute allows a temporary disable of an image in the configuration file for testing purposes, so there is no need to delete the object for that image. Specifying “disabled”: true, the tool will ignore the entry. Either deleting the attribute, or setting it as false, the tool will include back the image in the final APP Package.

The JSON configuration file, along with all the binaries declared in it, should be allocated in the build/ folder (build/config/ and build/images respectively);



Once the *app-cfg.json* file is configured as desired, we need to be sure that all declared images do exist In the *build/images/* folder.

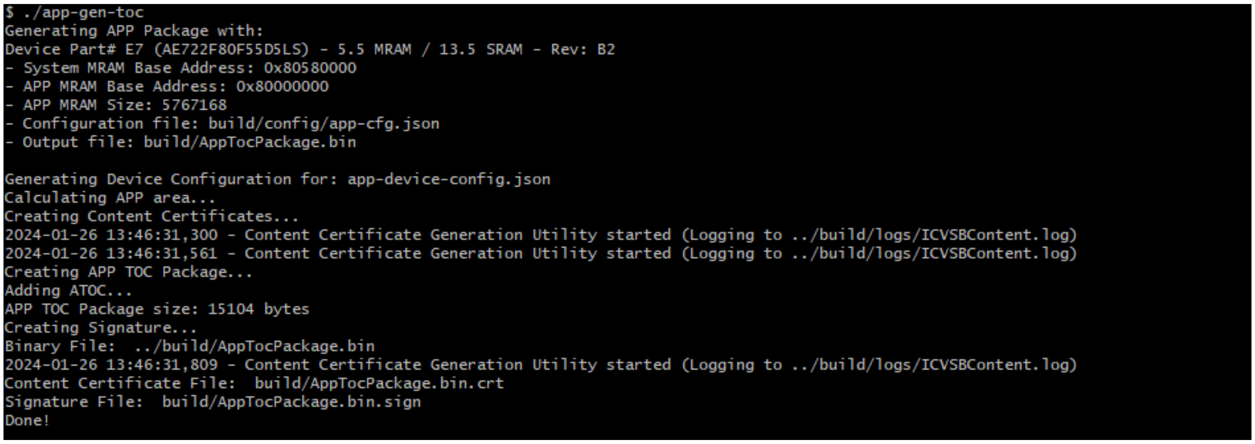
After running the *app-gen-toc.py* tool, the **APP TOC Package** (AppTocPackage.bin) will be generated in the *build/* folder.

The *build/logs/* folder contains a log for the Generation (OEMSBContent.log file)

Once the configuration is done, and all declared images exist in *build/images/* folder, continue with the APP TOC Package generation step.

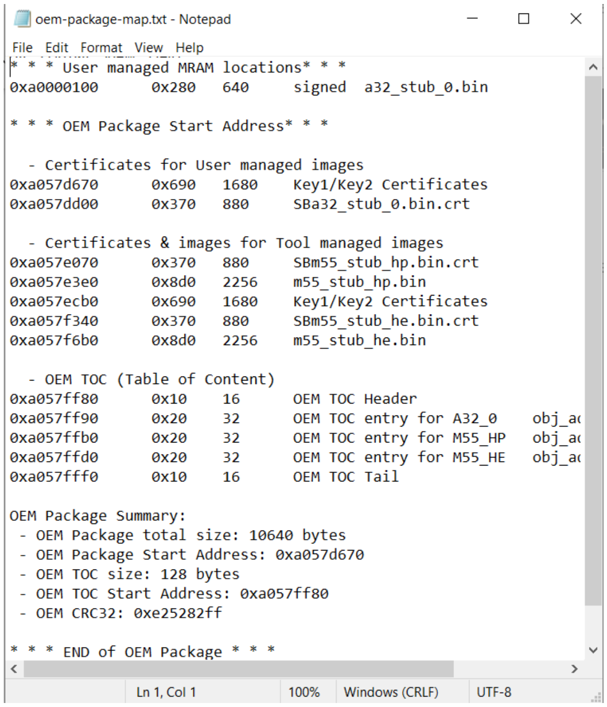
From the command prompt.

```
$ app-gen-toc (-h for help)
```



```
$ ./app-gen-toc
Generating APP Package with:
Device Part# E7 (AE722F80F55D5LS) - 5.5 MRAM / 13.5 SRAM - Rev: B2
- System MRAM Base Address: 0x80580000
- APP MRAM Base Address: 0x80000000
- APP MRAM Size: 5767168
- Configuration file: build/config/app-cfg.json
- Output file: build/AppTocPackage.bin

Generating Device Configuration for: app-device-config.json
Calculating APP area...
Creating Content Certificates...
2024-01-26 13:46:31,300 - Content Certificate Generation Utility started (Logging to ../build/logs/ICVSBContent.log)
2024-01-26 13:46:31,561 - Content Certificate Generation Utility started (Logging to ../build/logs/ICVSBContent.log)
Creating APP TOC Package...
Adding ATOC...
APP TOC Package size: 15104 bytes
Creating Signature...
Binary File:  ../build/AppTocPackage.bin
2024-01-26 13:46:31,809 - Content Certificate Generation Utility started (Logging to ../build/logs/ICVSBContent.log)
Content Certificate File:  build/AppTocPackage.bin.crt
Signature File:  build/AppTocPackage.bin.sign
Done!
```

Once the tool finishes, a map file named *app-package-map.txt* will be created in build/ folder to provide a reference of the memory map and images/certificates packaged in the APP TOC Package created by the tool. This memory map file will have the following structure.



```
* * * User managed MRAM locations* * *
0xa0000100      0x280   640     signed  a32_stub_0.bin

* * * OEM Package Start Address* * *

  - Certificates for User managed images
0xa057d670      0x690   1680    Key1/Key2 Certificates
0xa057dd00      0x370   880     SBa32_stub_0.bin.crt

  - Certificates & images for Tool managed images
0xa057e070      0x370   880     SBm55_stub_hp.bin.crt
0xa057e3e0      0x8d0   2256    m55_stub_hp.bin
0xa057ecb0      0x690   1680    Key1/Key2 Certificates
0xa057f340      0x370   880     SBm55_stub_he.bin.crt
0xa057f6b0      0x8d0   2256    m55_stub_he.bin

  - OEM TOC (Table of Content)
0xa057ff80      0x10    16      OEM TOC Header
0xa057ff90      0x20    32      OEM TOC entry for A32_0    obj_a
0xa057ffb0      0x20    32      OEM TOC entry for M55_HP   obj_a
0xa057ffd0      0x20    32      OEM TOC entry for M55_HE   obj_a
0xa057fff0      0x10    16      OEM TOC Tail

OEM Package Summary:
 - OEM Package total size: 10640 bytes
 - OEM Package Start Address: 0xa057d670
 - OEM TOC size: 128 bytes
 - OEM TOC Start Address: 0xa057ff80
 - OEM CRC32: 0xe25282ff

* * * END of OEM Package * * *
```

Additionally, a script file will be created in bin/ folder to automate the burn of this package into MRAM, using the app-write-mram.py tool. See next section for details.

Valid LoadAddress Regions

The valid address ranges supported are as follows:

Start Address	Size	Name	Name
0x5000-0000	256KB	SRAM2	M55_HP_ITCM_BASE
0x5080-0000	1MB	SRAM3	M55_HP_DTCM_BASE
0x5800-0000	256KB	SRAM4	M55_HE_ITCM_BASE
0x5880-0000	256KB	SRAM5	M55_HE_DTCM_BASE
0x6000 0000	256KB	SRAM9_A	
0x6004 0000	1024KB	SRAM6_A	
0x6200 0000	1024KB	SRAM6_B	
0x6300 0000	512KB	SRAM7	
0x6320 0000	2048KB	SRAM8	
0x8000-0000	6MB	MRAM	MRAM_BASE

Note on Factory (Blinky) for ENSEMBLE E1C and BALLETTO B1C

Security toolset (SETOOLS) contains a Factory Image (“Blinky) for ALIF development boards. This is supplied as a binary and is intended as a quick visual test of SETOOLS.

The ENSEMBLE E1C and BALLETTO B1C series devices differ from the ENSEMBLE E1, E3, E5 and E7 series devices. This means that the default Blinky for an ENSEMBLE device (E1,3,5,7) does not run on an ENSEMBLE E1C device, main difference being the Memory layouts.

For E1C/B1C devices there are added configuration files as configuration files for Balletto devices (both “Ensemble” E1C and B1C).

- app-cfg-1c.json
- app-device-config-1c.json

The app-cfg-1c.json file uses m55_blink_he_1c.bin binary, tailored for Balletto Devkit board. Also, the specific device configuration file is being used in this app configuration.

In SETOOLS, ENSEMBLE is the default family/part. So, in order to burn the Blinky example to a BALLETTO Dev board, the following steps should be followed:

- Select a Balletto B1C (or Ensemble E1C) part using the tools.config.py tool.
- Build the ATOC with:

```
$ ./app-gen-toc -f build/config/app-cfg-1c.json
```

- Burn the ATOC using:

```
$ ./app-write-mram
```

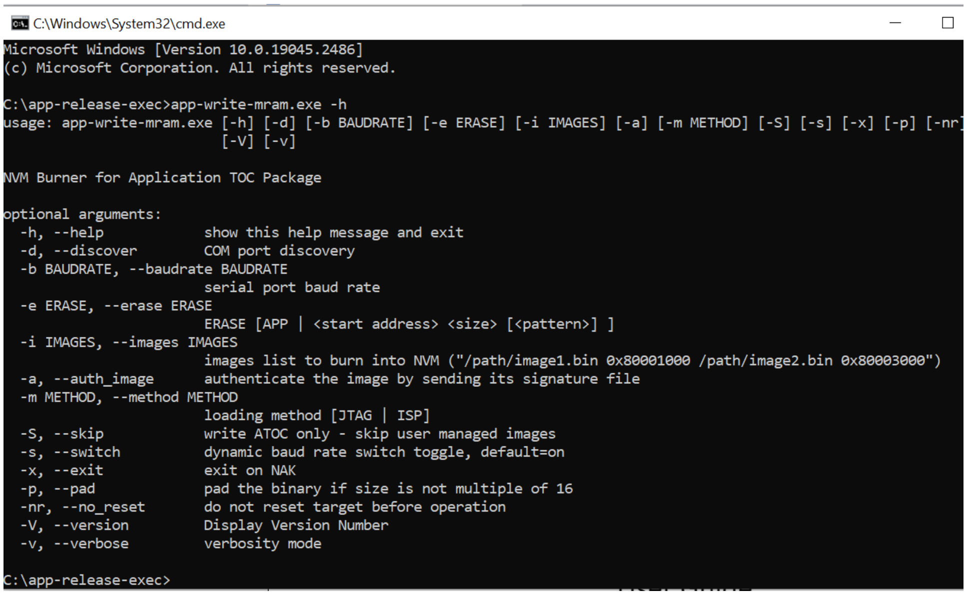
The LED should blink in the BALLETTO Dev Kit

app-write-mram tool

- This utility uses ISP (In System Programming) via the SE-UART connection or JTAG supporting SEGGER J-Link adapter, to update the MRAM.

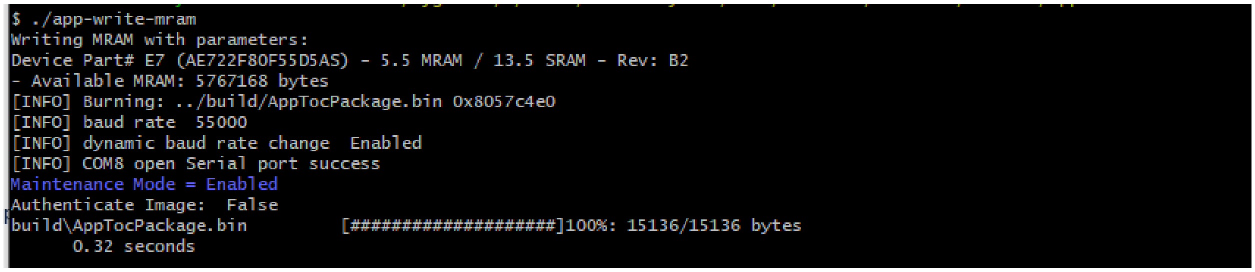
From the Windows Command prompt.

```
$ app-write-mram (-h for help)
```



```
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.

C:\app-release-exec>app-write-mram.exe -h
usage: app-write-mram.exe [-h] [-d] [-b BAUDRATE] [-e ERASE] [-i IMAGES] [-a] [-m METHOD] [-S] [-s] [-x] [-p] [-nr]
                          [-V] [-v]

NVM Burner for Application TOC Package

optional arguments:
  -h, --help            show this help message and exit
  -d, --discover        COM port discovery
  -b BAUDRATE, --baudrate BAUDRATE
                        serial port baud rate
  -e ERASE, --erase ERASE
                        ERASE [APP | <start address> <size> [<pattern>] ]
  -i IMAGES, --images IMAGES
                        images list to burn into NVM ("/path/image1.bin 0x80001000 /path/image2.bin 0x80003000")
  -a, --auth_image      authenticate the image by sending its signature file
  -m METHOD, --method METHOD
                        loading method [JTAG | ISP]
  -S, --skip            write ATOC only - skip user managed images
  -s, --switch          dynamic baud rate switch toggle, default=on
  -x, --exit            exit on NAK
  -p, --pad             pad the binary if size is not multiple of 16
  -nr, --no_reset       do not reset target before operation
  -V, --version         Display Version Number
  -v, --verbose         verbosity mode

C:\app-release-exec>
```

Before running this utility, be sure the CPU board is powered on and the SEUART is connected to the PC. Open Windows Command prompt and change directory to the release directory. Call the script as the example shows.



```
$ ./app-write-mram
Writing MRAM with parameters:
Device Part# E7 (AE722F80F55D5AS) - 5.5 MRAM / 13.5 SRAM - Rev: B2
- Available MRAM: 5767168 bytes
[INFO] Burning: ../build/AppTocPackage.bin 0x8057c4e0
[INFO] baud rate  55000
[INFO] dynamic baud rate change  Enabled
[INFO] COM8 open Serial port success
Maintenance Mode = Enabled
Authenticate Image:  False
build\AppTocPackage.bin         [####################]100%: 15136/15136 bytes
      0.32 seconds
```

The first time this command is executed it will prompt the user for the Serial Communications port name. This information is stored in a local file and will be read again when this command is executed.

You can override this using the -d (discovery) option and you will be prompted again for the port name.

There is a verbose mode (-v) which will print all the communications between the host and the target. During this mode, the progress bar is suppressed.

In SE LCS, the SE firmware requires that the ATOC signature (AppTocPackage.bin.sign) is sent as well. This is done by passing the option ‘-a’ (Authenticate Image) to app-write-mram.py.

```
$ app-write-mram -a
```

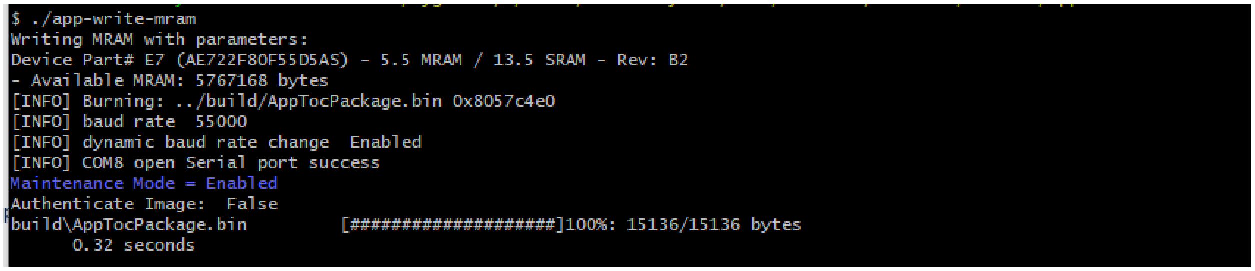
Maintenance Mode Internals

When updating MRAM (using app-write-mram or updateSystemPackage) SES is placed into a forced Maintenance mode. This means all cores are stopped achieved by resetting the device and then entering maintenance mode. This ensures no cores are operating when the MRAM update is performed.

Once the update is completed, the whole device is reset (this is done via an ISP command).

Maintenance Mode reporting

This tool will also report the Maintenance state of the Target system. See the section on the Maintenance tool for more details.



```
$ ./app-write-mram
Writing MRAM with parameters:
Device Part# E7 (AE722F80F55D5AS) - 5.5 MRAM / 13.5 SRAM - Rev: B2
- Available MRAM: 5767168 bytes
[INFO] Burning: ../build/AppTocPackage.bin 0x8057c4e0
[INFO] baud rate  55000
[INFO] dynamic baud rate change  Enabled
[INFO] COM8 open Serial port success
Maintenance Mode = Enabled
Authenticate Image:  False
build\AppTocPackage.bin         [####################]100%: 15136/15136 bytes
      0.32 seconds
```

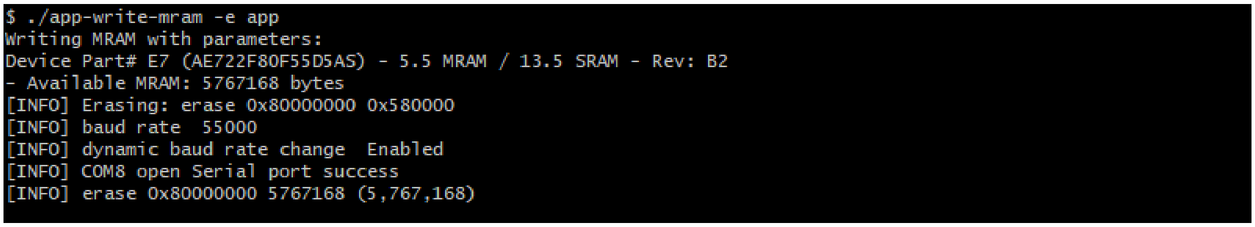
In the above, the maintenance mode is *Enabled*.

Erase option

Erase option will allow, via ISP, the contents of the Application MRAM to be erased. Erasing means that the MRAM location is written with zeros. The writing is verified. See later description on how to add your own pattern.

Erasing all the application MRAM

```
$ app-write-mram -e app
```



```
$ ./app-write-mram -e app
Writing MRAM with parameters:
Device Part# E7 (AE722F80F55D5AS) - 5.5 MRAM / 13.5 SRAM - Rev: B2
- Available MRAM: 5767168 bytes
[INFO] Erasing: erase 0x80000000 0x580000
[INFO] baud rate  55000
[INFO] dynamic baud rate change  Enabled
[INFO] COM8 open Serial port success
[INFO] erase 0x80000000 5767168 (5,767,168)
```

Depending on the size of the Application MRAM this can take a few seconds to complete, do not panic, the tool will exit when the operation completes, or an error is detected.

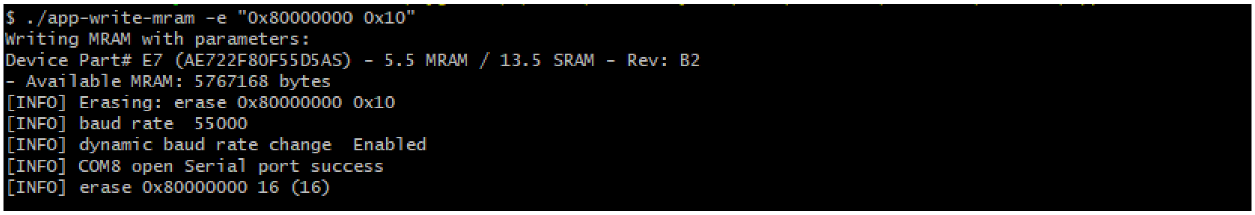
NOTE: The pattern written is always zeros and cannot be changed.

Erasing a specific address of application MRAM

The user can specify a specific address and length to erase.

Erasing a specific address of application MRAM

The user can specify a specific address and length to erase.

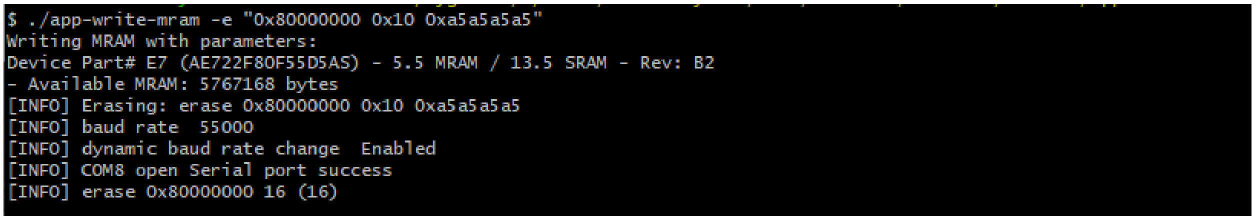


```
$ ./app-write-mram -e "0x80000000 0x10"
Writing MRAM with parameters:
Device Part# E7 (AE722F80F55D5AS) - 5.5 MRAM / 13.5 SRAM - Rev: B2
- Available MRAM: 5767168 bytes
[INFO] Erasing: erase 0x80000000 0x10
[INFO] baud rate  55000
[INFO] dynamic baud rate change  Enabled
[INFO] COM8 open Serial port success
[INFO] erase 0x80000000 16 (16)
```

This will erase from the start of MRAM for 16 Bytes, it will be erased with zeros.

Erasing a specific address of application MRAM with a pattern

The user can specify a specific address and length to erase along with a pattern to be written.

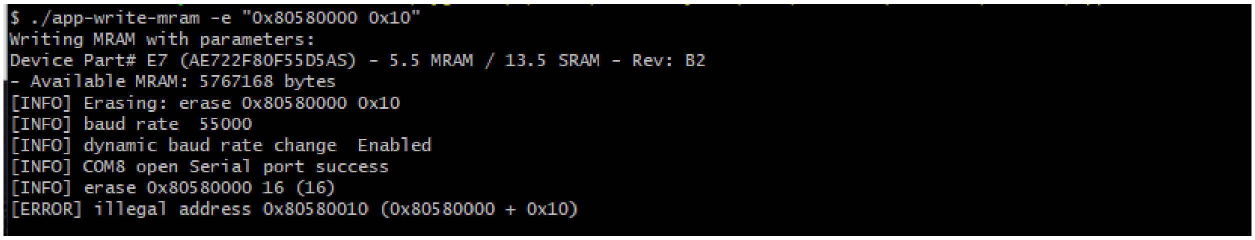


```
$ ./app-write-mram -e "0x80000000 0x10 0xa5a5a5a5"
Writing MRAM with parameters:
Device Part# E7 (AE722F80F55D5AS) - 5.5 MRAM / 13.5 SRAM - Rev: B2
- Available MRAM: 5767168 bytes
[INFO] Erasing: erase 0x80000000 0x10 0xa5a5a5a5
[INFO] baud rate  55000
[INFO] dynamic baud rate change  Enabled
[INFO] COM8 open Serial port success
[INFO] erase 0x80000000 16 (16)
```

In this example, the pattern 0xa5a5a5a5 will be written to the <address> for <length> bytes.

Erasing a specific address of MRAM that is illegal.

If the user specifies a region of MRAM that is not allowed to be accessed



```
$ ./app-write-mram -e "0x80580000 0x10"
Writing MRAM with parameters:
Device Part# E7 (AE722F80F55D5AS) - 5.5 MRAM / 13.5 SRAM - Rev: B2
- Available MRAM: 5767168 bytes
[INFO] Erasing: erase 0x80580000 0x10
[INFO] baud rate  55000
[INFO] dynamic baud rate change  Enabled
[INFO] COM8 open Serial port success
[INFO] erase 0x80580000 16 (16)
[ERROR] illegal address 0x80580010 (0x80580000 + 0x10)
```

In this example, an error is flagged that the address is illegal.

Padding binaries (-p option)

The MRAM controller requires write operations with data size in multiple of 16 bytes.

The tools take care of this requirement by always producing images that satisfy this constraint.

In case the user specifies binaries to be placed in specific MRAM addresses (using the “mramAddress” attribute in the ***app-gen-toc*** configuration file), these binaries should also satisfy the MRAM controller requirement (i.e., the size should be multiple of 16 bytes). When running the ***app-write-mram*** tool, if a user binary does not satisfy this constraint, the tool will throw an error message and abort the process.

The user should invoke the tool using the –p (or --pad) option to force the tool to pad the binaries. Be aware that using this option, the user provided binary will be modified (padded) and no backup will be created.

JTAG Method (-m)

JTAG is supported as a method of burning Application MRAM. The JTAG supported is SEGGER J-LINK

```
$ app-write-mram -m jtag
```

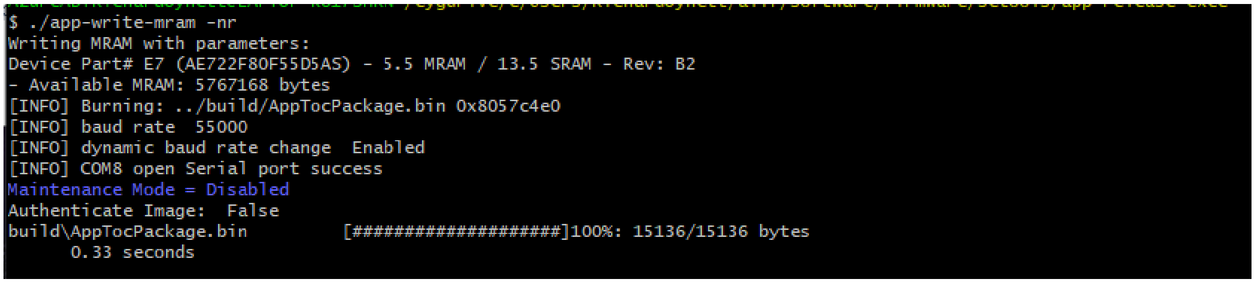
This method can be set as a default by using tools-config.

It should be noted that after the write has occurred a [RESET] is signaled to the Target. With the J-LINK this reset does not appear to have any effect and a manual press of [RESET] is required.

No Reset option (-nr)

The Reset option causes the command to reset the target and put it in 'Maintenance mode', in which no CPU cores are running (the TOCs are not processed). This is useful in cases where it is not desirable to have them running while writing to MRAM. To overwrite this mode (default), use –nr option (no reset)

```
$ app-write-mram -nr
```



```
$ ./app-write-mram -nr
Writing MRAM with parameters:
Device Part# E7 (AE722F80F55D5AS) - 5.5 MRAM / 13.5 SRAM - Rev: B2
- Available MRAM: 5767168 bytes
[INFO] Burning: ../build/AppTocPackage.bin 0x8057c4e0
[INFO] baud rate  55000
[INFO] dynamic baud rate change  Enabled
[INFO] COM8 open Serial port success
Maintenance Mode = Disabled
Authenticate Image:  False
build\AppTocPackage.bin         [####################]100%: 15136/15136 bytes
      0.33 seconds
```

Note: By default, the reset is enabled, the override is '-nr'

Images option (-i)

The -i option was implemented to provide a facility to download individual files at specific addresses. Note, this is not needed to download the *AppTocPackage.bin* file as the tool takes care of this.

Initially JTAG was used (via ARM-DS with ULink Pro) to burn images to MRAM. This uses semi-hosting to download the files. Since the MRAM burner runs from the `app-release/bin` directory, the **file should be specified relative to this path.** Subsequently, both ISP and J-Link support for MRAM burning were developed but still assume this filename approach for compatibility.

As an example, a binary file stored in `app-release/build/images/m55_blink.bin`, should be specified with its *relative path* from the `app-release/bin` directory.

```
$ app-write-mram -i “../build/config/images/m55_blink.bin 0x80001000”
```

Note also that double quotes must be provided.

SKIP Images option (-S)

The -S option will write only the ATOC to MRAM. Any managed images to be written will be ignored.

SWITCH option (-s)

The -s option will toggle the dynamic baud rate change when performing bulk transfer operations. By default, dynamic baud rate change is ENABLED.

No reset option (-nr)

The -nr option will not reset the Target being doing the Application MRAM write.

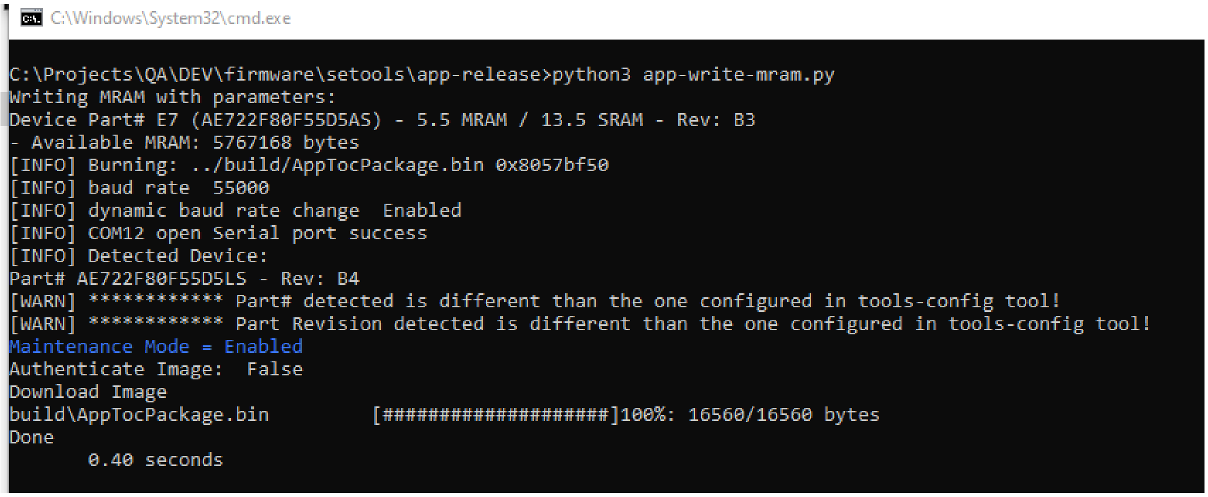
ATOC Policy

The HFRC is not very precise, and it is possible that on some boards it runs at a quite different frequency from its nominal 76.8MHz, this may result in seeing ISP errors (such as Checksum).

The policy was changed and starting with V83 – the policy when there is no ATOC is to assume that the external crystals HFXO and LFXO are present, and to switch the device to using them, including starting the PLL and switching to it.

Part# and/or Part Revision mismatch warnings

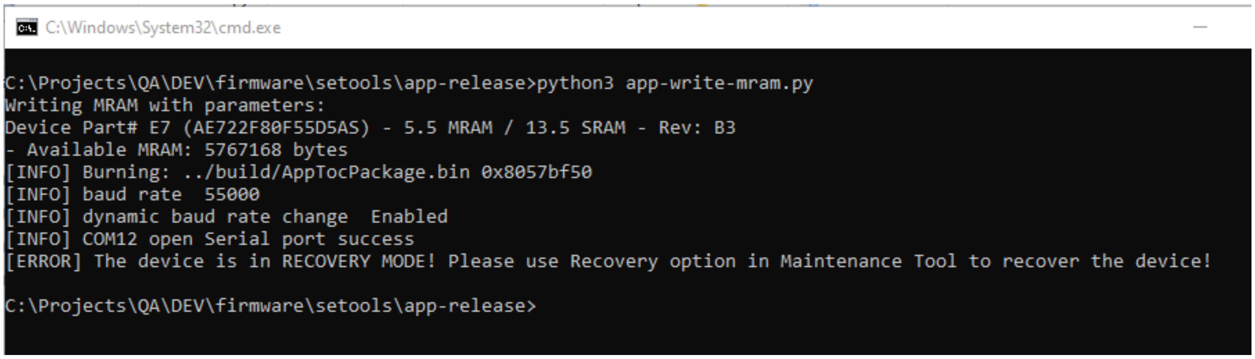
The app-write-mram tool, when using ISP protocol, will probe the device to get the Part# and Revision. If these parameters are different than the one configured in the tools (via tools-config tool), a Warning message will be displayed:



```
C:\Projects\QA\DEV\firmware\setools\app-release>python3 app-write-mram.py
Writing MRAM with parameters:
Device Part# E7 (AE722F80F55D5AS) - 5.5 MRAM / 13.5 SRAM - Rev: B3
- Available MRAM: 5767168 bytes
[INFO] Burning: ../build/AppTocPackage.bin 0x8057bf50
[INFO] baud rate  55000
[INFO] dynamic baud rate change  Enabled
[INFO] COM12 open Serial port success
[INFO] Detected Device:
Part# AE722F80F55D5LS - Rev: B4
[WARN] ************ Part# detected is different than the one configured in tools-config tool!
[WARN] ************ Part Revision detected is different than the one configured in tools-config tool!
Maintenance Mode = Enabled
Authenticate Image:  False
Download Image
build\AppTocPackage.bin         [####################]100%: 16560/16560 bytes
Done
       0.40 seconds
```

Using app-write-mram in SEROM Recovery mode

If for any reason, the device is in SEROM Recovery mode, the app-write-mram tool will not work as it requires an SES image to be running the tool will probe the device, via ISP, to check if SEROM or SES is running. If device is in Recovery mode, then it will warn the user about this condition and exit;



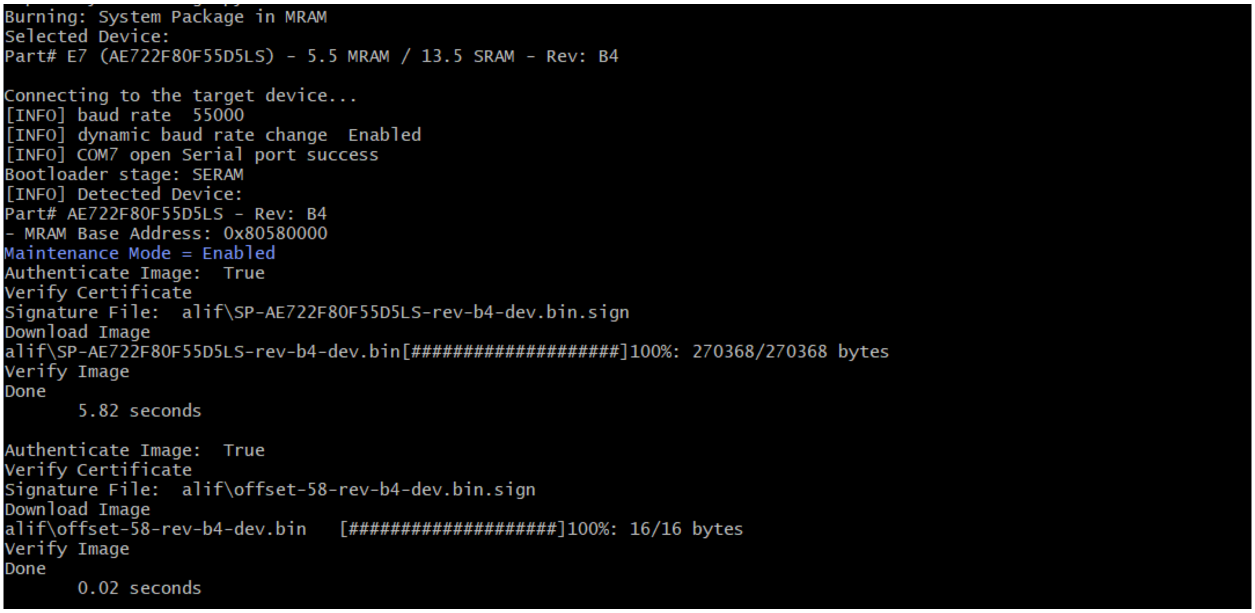
```
C:\Projects\QA\DEV\firmware\setools\app-release>python3 app-write-mram.py
Writing MRAM with parameters:
Device Part# E7 (AE722F80F55D5AS) - 5.5 MRAM / 13.5 SRAM - Rev: B3
- Available MRAM: 5767168 bytes
[INFO] Burning: ../build/AppTocPackage.bin 0x8057bf50
[INFO] baud rate  55000
[INFO] dynamic baud rate change  Enabled
[INFO] COM12 open Serial port success
[ERROR] The device is in RECOVERY MODE! Please use Recovery option in Maintenance Tool to recover the device!

C:\Projects\QA\DEV\firmware\setools\app-release>
```

The user must use the maintenance tool to recover the device.

UpdateSystemPackage tool

Before running this utility, be sure the CPU board is powered on and the SEUART is connected to the PC. Open Windows Command prompt and change directory to the release directory. Call the script as the example shows.

```
$ updateSystemPackage
```



```
Burning: System Package in MRAM
Selected Device:
Part# E7 (AE722F80F55D5LS) - 5.5 MRAM / 13.5 SRAM - Rev: B4

Connecting to the target device...
[INFO] baud rate  55000
[INFO] dynamic baud rate change  Enabled
[INFO] COM7 open Serial port success
Bootloader stage: SERAM
[INFO] Detected Device:
Part# AE722F80F55D5LS - Rev: B4
- MRAM Base Address: 0x80580000
Maintenance Mode = Enabled
Authenticate Image:  True
Verify Certificate
Signature File:  alif\SP-AE722F80F55D5LS-rev-b4-dev.bin.sign
Download Image
alif\SP-AE722F80F55D5LS-rev-b4-dev.bin[####################]100%: 270368/270368 bytes
Verify Image
Done
       5.82 seconds

Authenticate Image:  True
Verify Certificate
Signature File:  alif\offset-58-rev-b4-dev.bin.sign
Download Image
alif\offset-58-rev-b4-dev.bin   [####################]100%: 16/16 bytes
Verify Image
Done
       0.02 seconds
```

ALIF images are signed. SES will check that image is signed (based on the LCS state of the device) and only allow signed images.

The tool will Probe the target device to ensure it matches the device chosen through tools-config. In the above screen shot, we see:

```
Bootloader stage: SERAM
```

This shows that SERAM is running, the alternative is that SEROM is running and in this case, this is the wrong tool to be using.

```
Device Revision: B0
```

This shows the target device that is attached. The tool will exit if there is a mismatch.

No Reset option (-nr)

The Reset option causes the command to reset the target and put it in 'Maintenance mode', in which no CPU cores are running (the TOCs are not processed). This is useful in cases where it is not desirable to have them running while writing to MRAM. To overwrite this mode (default), use –nr option (no reset)

```
$ updateSystemPackage -nr
```

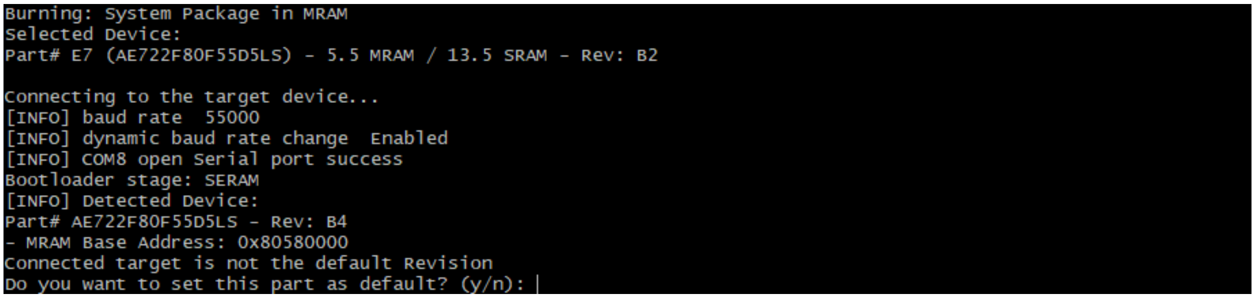
No Authentication (-na) option

If the device is in DM (Device Manufacturer) or SE (Secure Enable) lifecycle state, images written to the System partition of MRAM must be signed by Alif. The APP release package already contains signatures for the STOC images. The script updateSystemPackage.py runs by default in ‘Authenticated Mode’, i.e., it sends the image signatures to the device for verification. The option ‘no-authentication’ can be used to override that behavior and run in ‘non authenticated mode’ instead. This is allowed in CM LCS, but not allowed in DM and SE LCSs.

```
$ updateSystemPackage -na
```

Setting the part as default

You may see the following:



```
Burning: System Package in MRAM
Selected Device:
Part# E7 (AE722F80F55D5LS) - 5.5 MRAM / 13.5 SRAM - Rev: B2

Connecting to the target device...
[INFO] baud rate  55000
[INFO] dynamic baud rate change  Enabled
[INFO] COM8 open Serial port success
Bootloader stage: SERAM
[INFO] Detected Device:
Part# AE722F80F55D5LS - Rev: B4
- MRAM Base Address: 0x80580000
Connected target is not the default Revision
Do you want to set this part as default? (y/n): |
```

This is informing you that the target device does not match the default device selected via the tools-config too.

Answering ‘y’es will tell updateSystemPackage to select the correct package for the *detected* device.

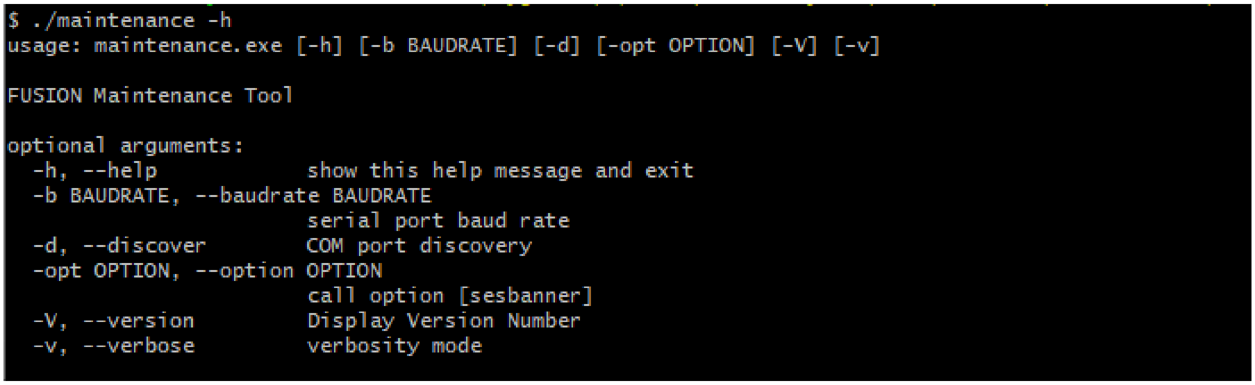
Answering ‘n’o will tell updateSystemPackage to select the correct package based on the default selection from tools-config.

Maintenance tool

Maintenance mode allows for the recovery of MRAM executable images and interaction with SES.

Running the Tool

```
$ maintenance (-h for help)
```



```
$ ./maintenance -h
usage: maintenance.exe [-h] [-b BAUDRATE] [-d] [-opt OPTION] [-V] [-v]

FUSION Maintenance Tool

optional arguments:
  -h, --help            show this help message and exit
  -b BAUDRATE, --baudrate BAUDRATE
                        serial port baud rate
  -d, --discover        COM port discovery
  -opt OPTION, --option OPTION
                        call option [sesbanner]
  -V, --version         Display Version Number
  -v, --verbose         verbosity mode
```

The command presents several options including maintenance mode.

Select an option number from the menu.

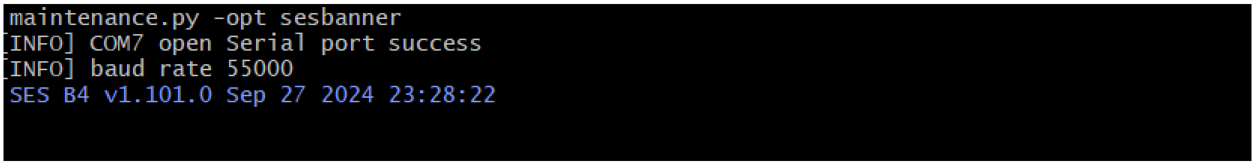
To exit the tool just press [ENTER] at the option selection.

Command line options

```
$ maintenance -opt <option>
```

Option	
sesbanner	Returns the SES Banner string.

This allows a command to be executed by the Maintenance tool without navigating the menu options.



```
maintenance.py -opt sesbanner
[INFO] COM7 open Serial port success
[INFO] baud rate 55000
 SES B4 v1.101.0 Sep 27 2024 23:28:22
```

Maintenance Menu Grouping

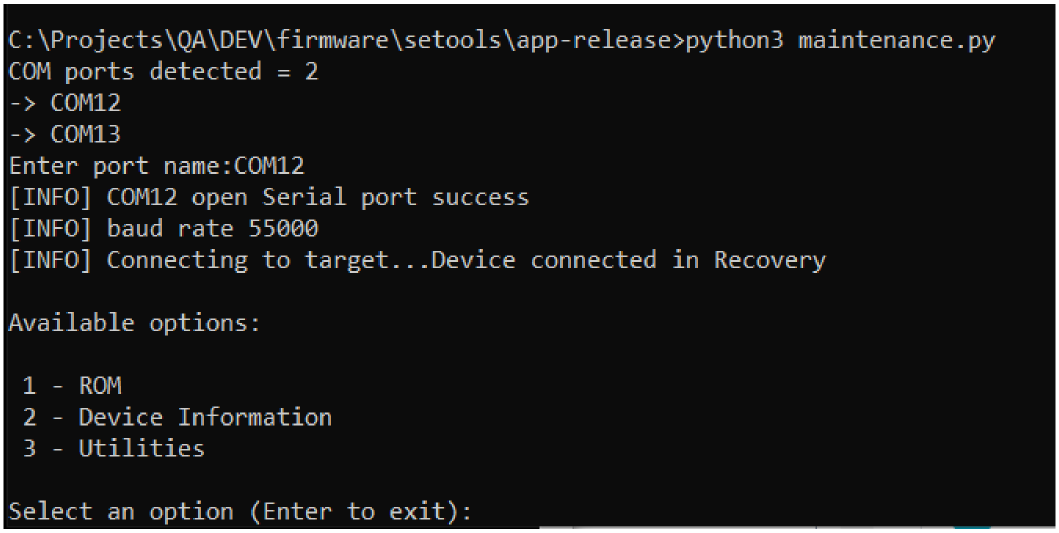
The maintenance menus are grouped into categories.

Action	Menu Groups							
	Device Control	**Device Info**	**MRAM**	**Utilities**	Set	**ROM**	**SEROM**	**SES**
Hard maintenance mode	●							●
Soft maintenance mode	●							●
Device Reset	●							●
Device enquiry		●					●	●
Get revision info		●					●	●
Get TOC info		●						●
Get SES banner		●						●
Get CPU boot info		●						●
Get OTP Data		●					●	●
Get MRAM data		●					●	●
Get log data		●						●
Get SEROM trace data		●					●	●
Get SERAM trace data		●						●
Get Power data		●						●
Get Clock data		●						●
Get SERAM metrics				●				●
Terminal mode				●			●	●
Get ECC Key				●				●
Get Firewall configuration				●				●
Erase application MRAM			●					●
Fast erase application MRAM			●					●
Fast erase application NTOC			●					
Get MRAM info			●					●
Recovery							●	
Enable LOGGING					●			●
Disable LOGGING					●			●
Enable PRINTING					●			●
Disable PRINTING					●			●

NOTE: Some ISP commands are common between SEROM and SES. When in Recovery mode the number of commands is limited. This is because the SEROM only supports a limited subset of features compared to ISP running from SES.

maintenance menu in SEROM Recovery mode

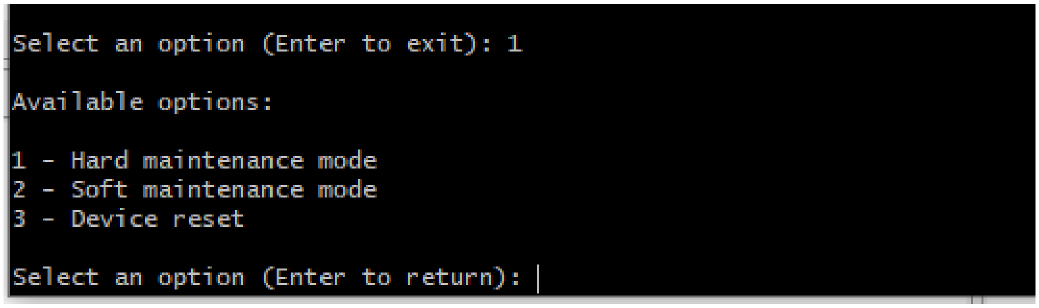
If for any reason, the device enters into SEROM Recovery mode (i.e. no SERAM running), the maintenance tool will detect this stage, and it will adapt the menu options to only show those enabled for this case;



```
C:\Projects\QA\DEV\firmware\setools\app-release>python3 maintenance.py
COM ports detected = 2
-> COM12
-> COM13
Enter port name:COM12
[INFO] COM12 open Serial port success
[INFO] baud rate 55000
[INFO] Connecting to target...Device connected in Recovery

Available options:

 1 - ROM
 2 - Device Information
 3 - Utilities

Select an option (Enter to exit):
```

Device Control

Top level group menu



```
Select an option (Enter to exit): 1

Available options:

1 - Hard maintenance mode
2 - Soft maintenance mode
3 - Device reset

Select an option (Enter to return): |
```

Hard maintenance Mode

Hard maintenance mode will wait until the user has pressed RESET on the target board.

Once]RESET] has been pressed the tool will drop back to the Available options menu. NOTE: This mode skips the TOC processing section of SES, if you use the get TOC info option it will return no data as no TOCs have been loaded.

To exit the tool, just press [ENTER]

Soft maintenance Mode

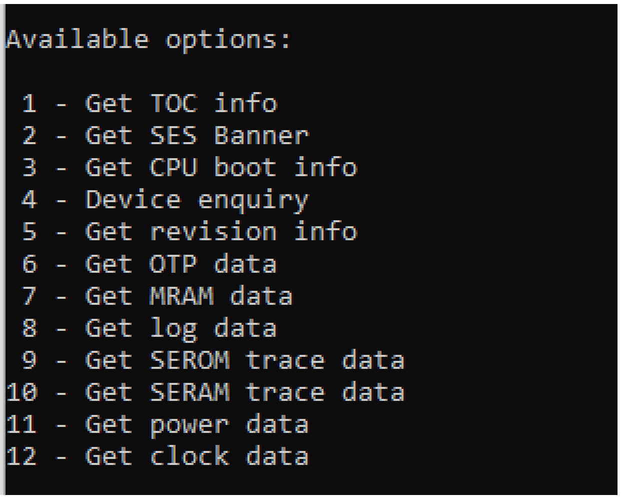
Soft maintenance mode is like Hard maintenance mode, but no [RESET] button needs to be pressed on the target board.

Device Reset

Issues a Reset ISP command to the board.

Device Information

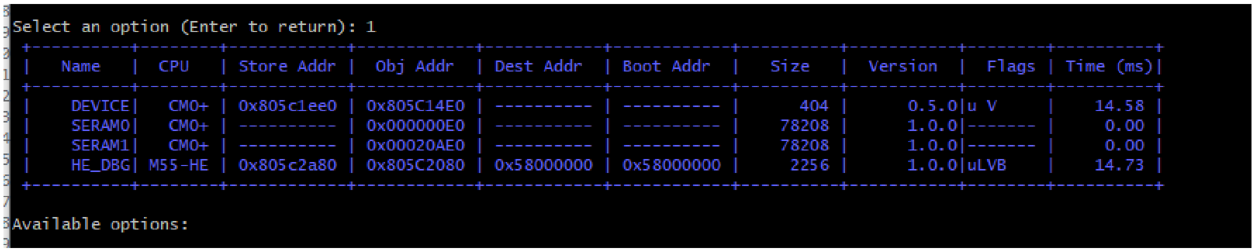
Top level group menu



```
Available options:

 1 - Get TOC info
 2 - Get SES Banner
 3 - Get CPU boot info
 4 - Device enquiry
 5 - Get revision info
 6 - Get OTP data
 7 - Get MRAM data
 8 - Get log data
 9 - Get SEROM trace data
10 - Get SERAM trace data
11 - Get power data
12 - Get clock data
```

Get Table of contents information

This will return the TOC contents in MRAM along with the boot status of the Application CPUs.



```
Select an option (Enter to return): 1
 +----------+--------+------------+------------+------------+------------+----------+-----------+--------+----------+
 |   Name   |  CPU   | Store Addr |  Obj Addr  | Dest Addr  | Boot Addr  |   Size   |  Version  |  Flags | Time (ms)|
 +----------+--------+------------+------------+------------+------------+----------+-----------+--------+----------+
 |    DEVICE|   CM0+ | 0x805c1ee0 | 0x805C14E0 | ---------- | ---------- |      404 |      0.5.0|u V     |    14.58 |
 |    SERAM0|   CM0+ | ---------- | 0x000000E0 | ---------- | ---------- |    78208 |      1.0.0|------- |     0.00 |
 |    SERAM1|   CM0+ | ---------- | 0x00020AE0 | ---------- | ---------- |    78208 |      1.0.0|------- |     0.00 |
 |    HE_DBG| M55-HE | 0x805c2a80 | 0x805C2080 | 0x58000000 | 0x58000000 |     2256 |      1.0.0|uLVB    |    14.73 |
 +----------+--------+------------+------------+------------+------------+----------+-----------+--------+----------+

Available options:
```

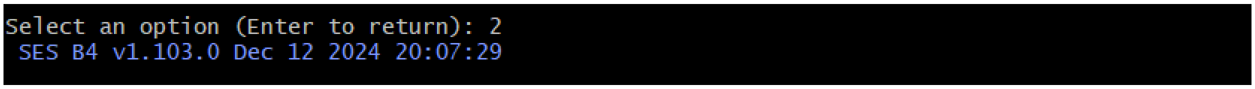
Legend Reference

Name	Name of entry from JSON file.
CPU	Which CPU is this TOC referencing.
Store Address	MRAM location
Object address	MRAM TOC address
Destination address	Copy address for object (if relevant).
Boot Address	Boot address
Size	Size of TOC entry
Version	Version id from JSON file
Flags	Directives for the TOC object

u	Uncompressed image
C	Compressed Image
L	Image is loaded from MRAM
V	Image is verified
s	Image is Skipped
B	Image has been Booted
E	Image is Encrypted
D	Image is Deferred, will be Loaded / Booted later.

When you are in hard maintenance mode, the TOC table contents will look very different. The main reason is that Hard Maintenance skips the TOC processing stage entirely except for some DEVICE CONFIG objects. No Application CPU cores are booted or loaded with images whilst in this mode.

Get SES Banner

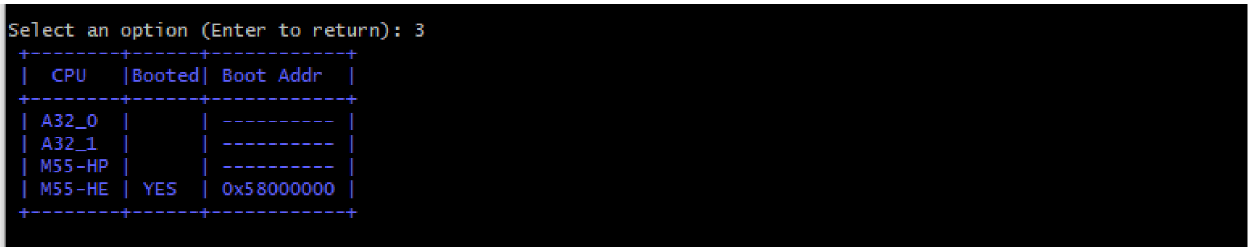


```
Select an option (Enter to return): 2
 SES B4 v1.103.0 Dec 12 2024 20:07:29
```

This returns the version string for the SES image

NOTE: The Banner output is only 80 characters.

Get CPU boot information



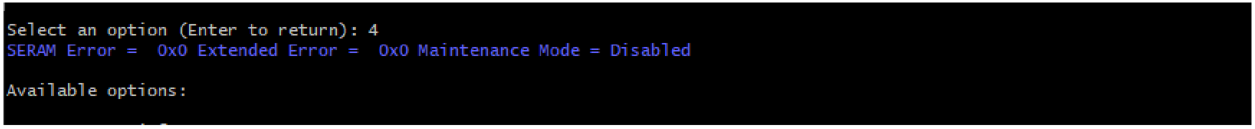
```
Select an option (Enter to return): 3
 +--------+------+------------+
 |  CPU   |Booted| Boot Addr  |
 +--------+------+------------+
 | A32_0  |      | ---------- |
 | A32_1  |      | ---------- |
 | M55-HP |      | ---------- |
 | M55-HE | YES  | 0x58000000 |
 +--------+------+------------+
```

This returns the status of the CPU cores.

Device Enquiry

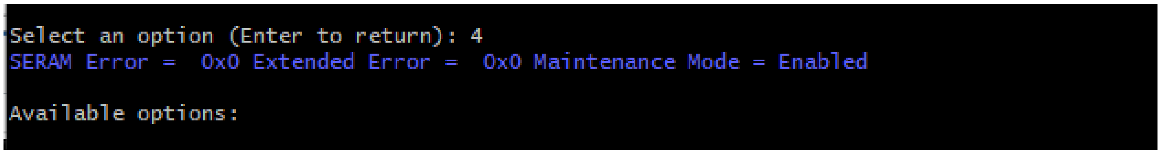
Returns if connected to SEROM or SES. Any errors are reported otherwise 0x0 (Success) is returned.

Device Enquiry - SES



```
Select an option (Enter to return): 4
SERAM Error =  0x0 Extended Error =  0x0 Maintenance Mode = Disabled

Available options:
```

It will also tell you if you are in Maintenance mode, in above case it is DISABLED

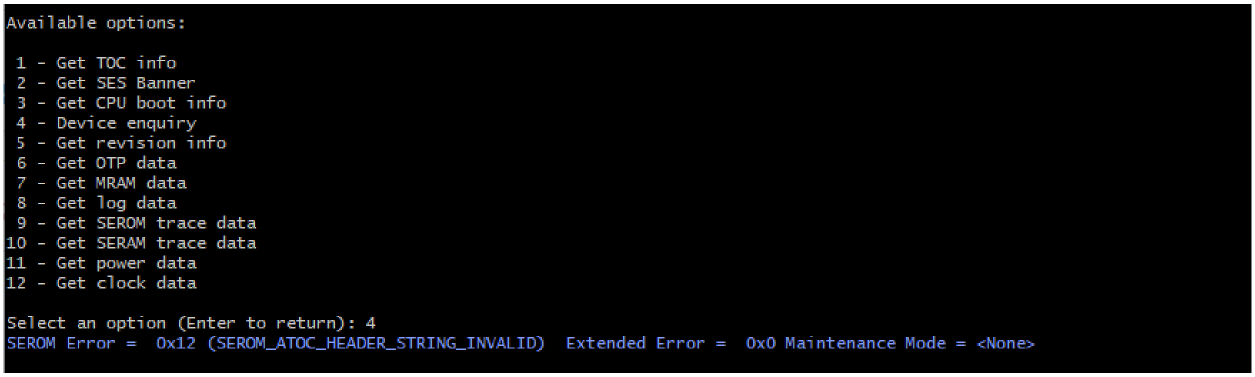


```
Select an option (Enter to return): 4
SERAM Error =  0x0 Extended Error =  0x0 Maintenance Mode = Enabled

Available options:
```

In the above case it is ENABLED

Device Enquiry – SEROM

When connected to SEROM:



```
Available options:

 1 - Get TOC info
 2 - Get SES Banner
 3 - Get CPU boot info
 4 - Device enquiry
 5 - Get revision info
 6 - Get OTP data
 7 - Get MRAM data
 8 - Get log data
 9 - Get SEROM trace data
10 - Get SERAM trace data
11 - Get power data
12 - Get clock data

Select an option (Enter to return): 4
SEROM Error =  0x12 (SEROM_ATOC_HEADER_STRING_INVALID)  Extended Error =  0x0 Maintenance Mode = <None>
```

NOTE: If you are connected to SEROM then this means your MRAM is not provisioned with an SES image. Maintenance mode is not applicable in this scenario.

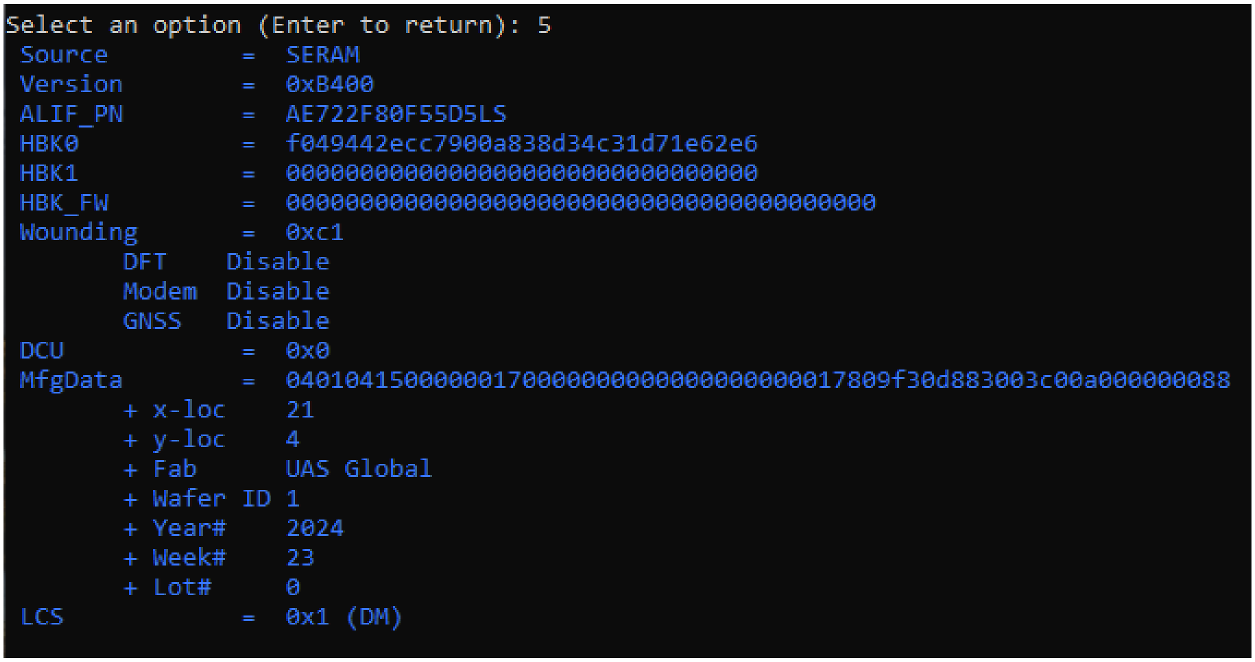
Get Revision Information

Device Information

Item	Meaning
Version	SOC id register
ALIF_PN	ALIF Part Number
HBK0	SHA-128 of ALIF RoT Public key
HBK1	SHA-128 of Customer RoT Public key
HBK_FW	
Wounding	Shows DISABLED Wounding bits
DCU	Debug Lock bits
MfgData	ALIF Manufacturing data
SerialN	ALIF Device Serial Number
LCS	Life Cycle State CM, DM, SE, RMA

Revision B4

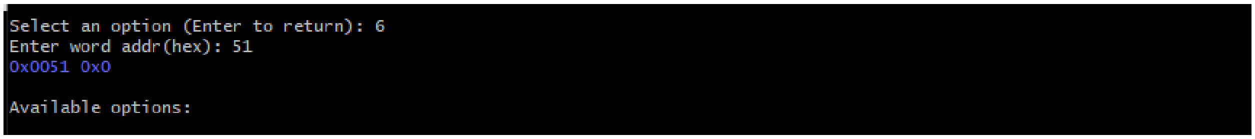
Returns device specific information such as the SOC ID, Part#.



```
Select an option (Enter to return): 5
 Source          =  SERAM
 Version         =  0xB400
 ALIF_PN         =  AE722F80F55D5LS
 HBK0            =  f049442ecc7900a838d34c31d71e62e6
 HBK1            =  00000000000000000000000000000000
 HBK_FW          =  0000000000000000000000000000000000000000
 Wounding        =  0xc1
        DFT    Disable
        Modem  Disable
        GNSS   Disable
 DCU             =  0x0
 MfgData         =  0401041500000017000000000000000000000017809f30d883003c00a000000088
        + x-loc     21
        + y-loc     4
        + Fab       UAS Global
        + Wafer ID 1
        + Year#     2024
        + Week#     23
        + Lot#      0
 LCS             =  0x1 (DM)
```

Get OTP data

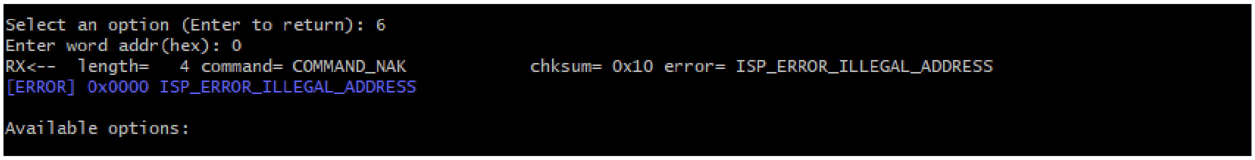
Returns OTP data based on a provided OTP word offset



```
Select an option (Enter to return): 6
Enter word addr(hex): 51
0x0051 0x0

Available options:
```

In this example, we are extracting the ALIF Manufacturing data location (Word offset 0x51)

Not all OTP locations are visible, in this case you will see:



```
Select an option (Enter to return): 6
Enter word addr(hex): 0
RX<--   length=    4 command= COMMAND_NAK             chksum= 0x10 error= ISP_ERROR_ILLEGAL_ADDRESS
[ERROR] 0x0000 ISP_ERROR_ILLEGAL_ADDRESS

Available options:
```

An error message is reported that you have supplied an ILLEGAL address offset.

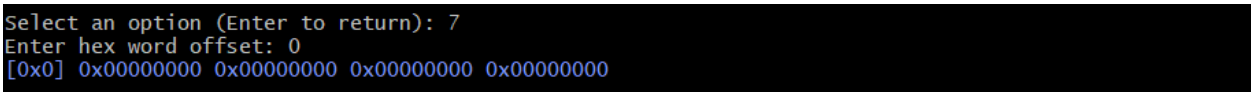
Allowed OTP offsets:

Offset	Contents
0x10 – 0x18	ICV Flags
0x21 – 0x2B	OEM Flags
0x5F – 0x7F	OEM Custom Area

Additionally, in DM LCS the OTP words from 0x19 to 0x20 (8 words) can be read as well. Those store the OEM keys ‘Kcp’ and ‘Kce’.

Get MRAM data

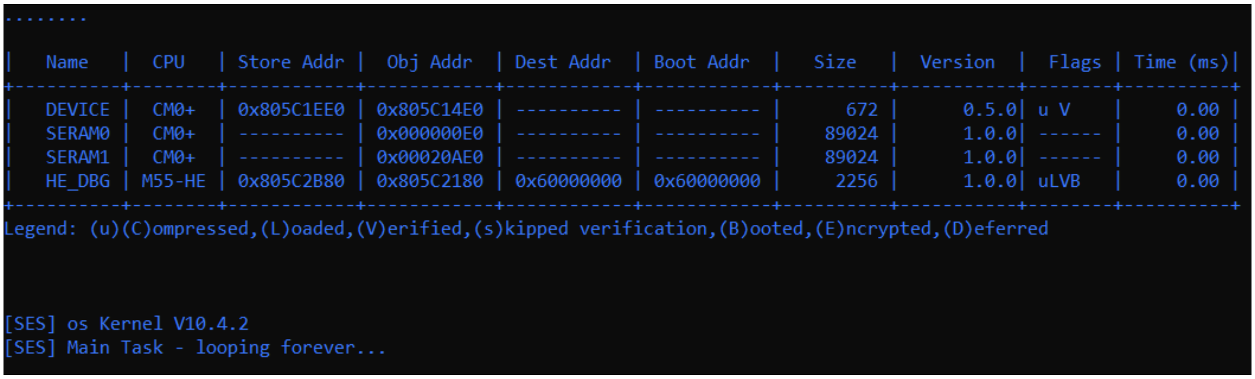
Returns four words of MRAM data based on a provided MRAM word offset. The display shows the word offset entered followed by four words of data.



```
Select an option (Enter to return): 7
Enter hex word offset: 0
[0x0] 0x00000000 0x00000000 0x00000000 0x00000000
```

Get log data

Returns the last SE-UART data sent:



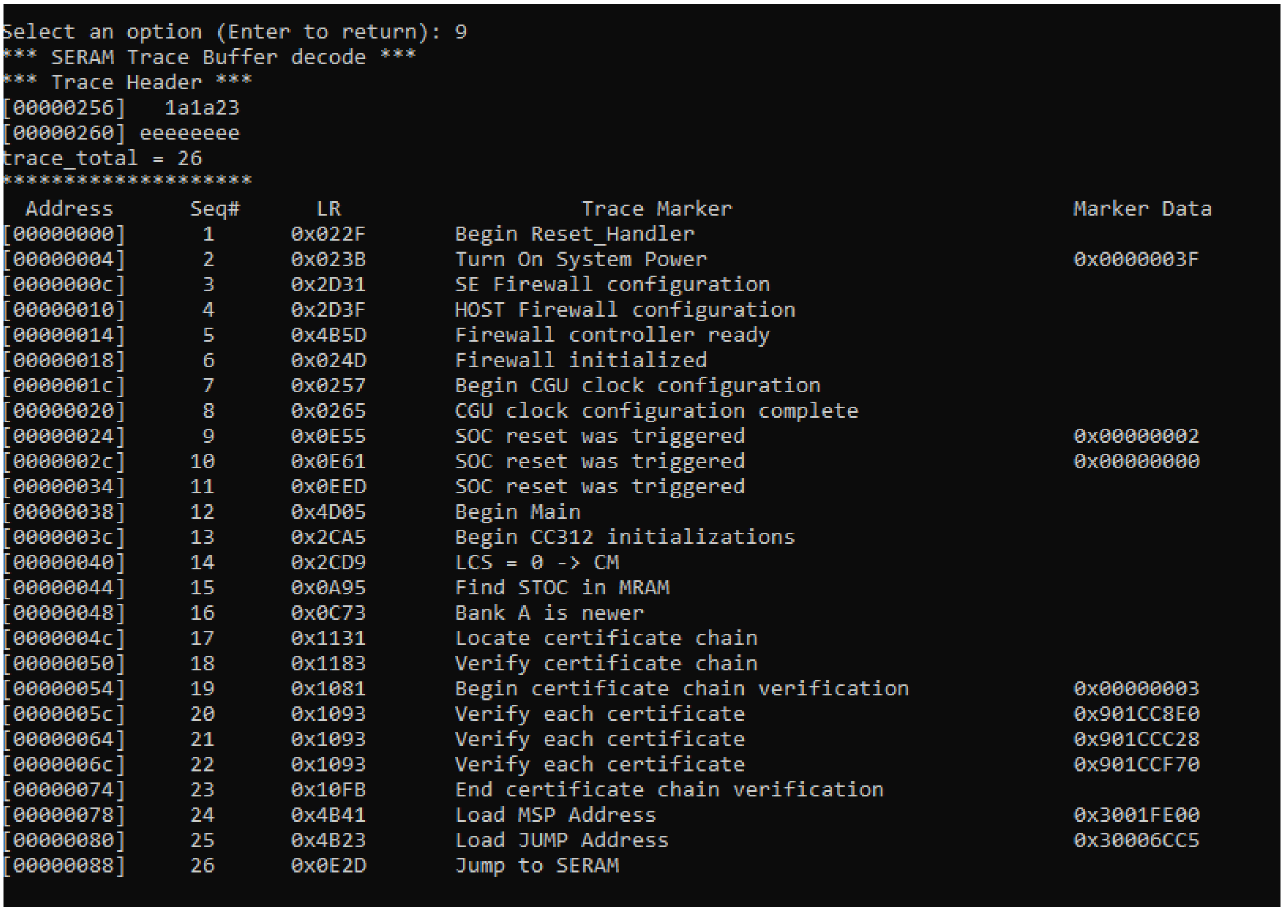
```
........

|   Name   |  CPU   | Store Addr |  Obj Addr  | Dest Addr  | Boot Addr  |   Size   |  Version  |  Flags | Time (ms)|
+----------+--------+------------+------------+------------+------------+----------+-----------+--------+----------+
|   DEVICE |  CM0+  | 0x805C1EE0 | 0x805C14E0 | ---------- | ---------- |      672 |     0.5.0| u V    |     0.00 |
|   SERAM0 |  CM0+  | ---------- | 0x000000E0 | ---------- | ---------- |    89024 |     1.0.0| ------ |     0.00 |
|   SERAM1 |  CM0+  | ---------- | 0x00020AE0 | ---------- | ---------- |    89024 |     1.0.0| ------ |     0.00 |
|   HE_DBG | M55-HE | 0x805C2B80 | 0x805C2180 | 0x60000000 | 0x60000000 |     2256 |     1.0.0| uLVB   |     0.00 |
+----------+--------+------------+------------+------------+------------+----------+-----------+--------+----------+
Legend: (u)(C)ompressed,(L)oaded,(V)erified,(s)kipped verification,(B)ooted,(E)ncrypted,(D)eferred



[SES] os Kernel V10.4.2
[SES] Main Task - looping forever...
```

Get trace data

Trace data are Markers left in SES memory space as SEROM and SES are executed. There are separate buffers used for SEROM and SES.

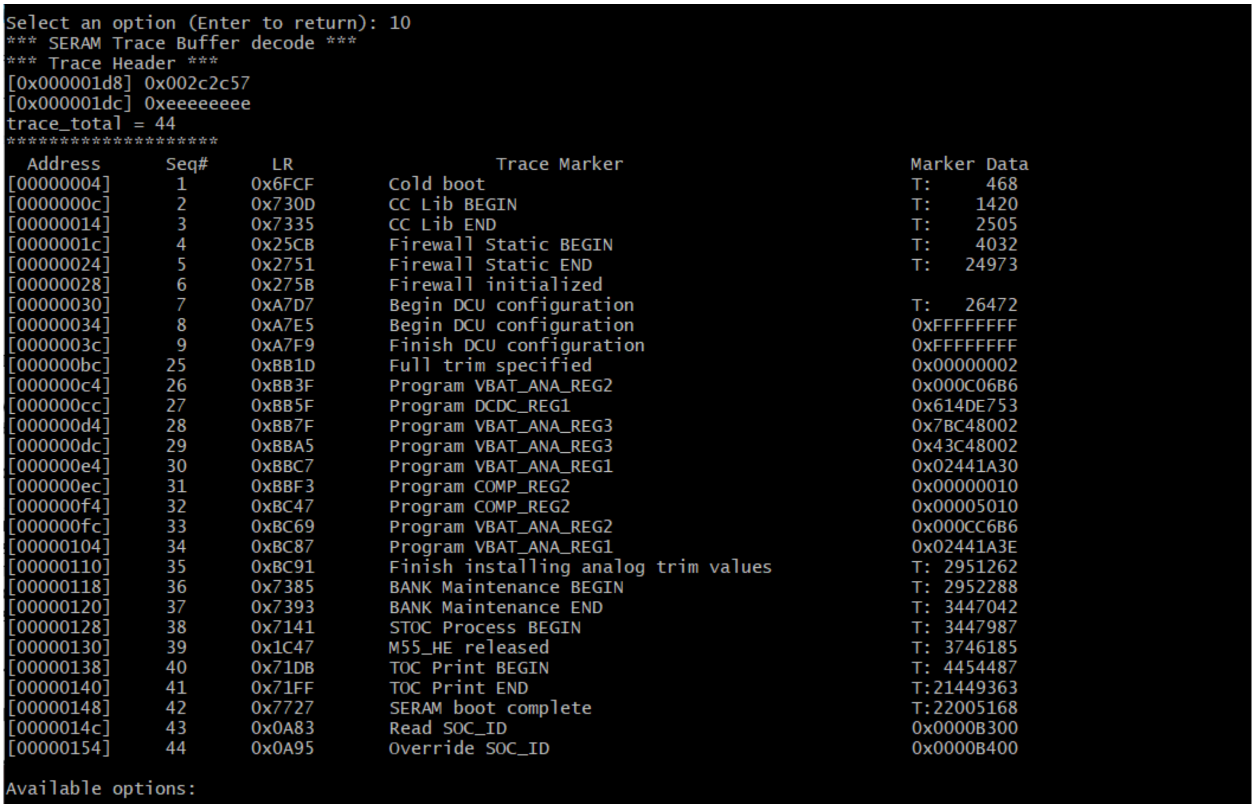
NOTE: If you are in RECOVERY Mode (SEROM only mode) you can only see the SEROM Trace data.

SEROM Trace Data



```
Select an option (Enter to return): 9
*** SERAM Trace Buffer decode ***
*** Trace Header ***
[00000256]   1a1a23
[00000260] eeeeeeee
trace_total = 26
********************
  Address      Seq#      LR                   Trace Marker                            Marker Data
[00000000]      1    0x022F         Begin Reset_Handler
[00000004]      2    0x023B         Turn On System Power                              0x0000003F
[0000000c]      3    0x2D31         SE Firewall configuration
[00000010]      4    0x2D3F         HOST Firewall configuration
[00000014]      5    0x4B5D         Firewall controller ready
[00000018]      6    0x024D         Firewall initialized
[0000001c]      7    0x0257         Begin CGU clock configuration
[00000020]      8    0x0265         CGU clock configuration complete
[00000024]      9    0x0E55         SOC reset was triggered                           0x00000002
[0000002c]     10    0x0E61         SOC reset was triggered                           0x00000000
[00000034]     11    0x0EED         SOC reset was triggered
[00000038]     12    0x4D05         Begin Main
[0000003c]     13    0x2CA5         Begin CC312 initializations
[00000040]     14    0x2CD9         LCS = 0 -> CM
[00000044]     15    0x0A95         Find STOC in MRAM
[00000048]     16    0x0C73         Bank A is newer
[0000004c]     17    0x1131         Locate certificate chain
[00000050]     18    0x1183         Verify certificate chain
[00000054]     19    0x1081         Begin certificate chain verification              0x00000003
[0000005c]     20    0x1093         Verify each certificate                           0x901CC8E0
[00000064]     21    0x1093         Verify each certificate                           0x901CCC28
[0000006c]     22    0x1093         Verify each certificate                           0x901CCF70
[00000074]     23    0x10FB         End certificate chain verification
[00000078]     24    0x4B41         Load MSP Address                                  0x3001FE00
[00000080]     25    0x4B23         Load JUMP Address                                 0x30006CC5
[00000088]     26    0x0E2D         Jump to SERAM
```

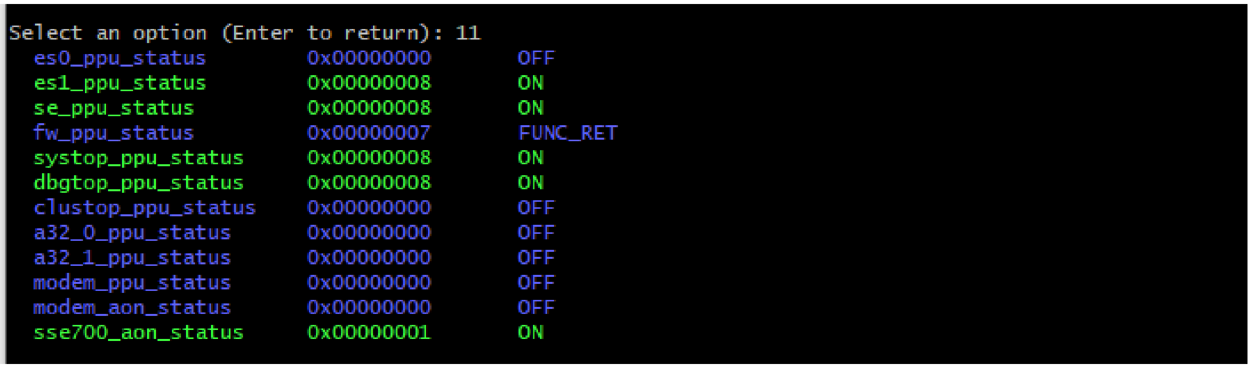
SERAM Trace Data



```
Select an option (Enter to return): 10
*** SERAM Trace Buffer decode ***
*** Trace Header ***
[0x000001d8] 0x002c2c57
[0x000001dc] 0xeeeeeeee
trace_total = 44
********************
  Address      Seq#      LR                    Trace Marker                           Marker Data
[00000004]       1      0x6FCF       Cold boot                                         T:     468
[0000000c]       2      0x730D       CC Lib BEGIN                                      T:    1420
[00000014]       3      0x7335       CC Lib END                                        T:    2505
[0000001c]       4      0x25CB       Firewall Static BEGIN                             T:    4032
[00000024]       5      0x2751       Firewall Static END                               T:   24973
[00000028]       6      0x275B       Firewall initialized
[00000030]       7      0xA7D7       Begin DCU configuration                           T:   26472
[00000034]       8      0xA7E5       Begin DCU configuration                           0xFFFFFFFF
[0000003c]       9      0xA7F9       Finish DCU configuration                          0xFFFFFFFF
[000000bc]      25      0xBB1D       Full trim specified                               0x00000002
[000000c4]      26      0xBB3F       Program VBAT_ANA_REG2                             0x000C06B6
[000000cc]      27      0xBB5F       Program DCDC_REG1                                 0x614DE753
[000000d4]      28      0xBB7F       Program VBAT_ANA_REG3                             0x7BC48002
[000000dc]      29      0xBBA5       Program VBAT_ANA_REG3                             0x43C48002
[000000e4]      30      0xBBC7       Program VBAT_ANA_REG1                             0x02441A30
[000000ec]      31      0xBBF3       Program COMP_REG2                                 0x00000010
[000000f4]      32      0xBC47       Program COMP_REG2                                 0x00005010
[000000fc]      33      0xBC69       Program VBAT_ANA_REG2                             0x000CC6B6
[00000104]      34      0xBC87       Program VBAT_ANA_REG1                             0x02441A3E
[00000110]      35      0xBC91       Finish installing analog trim values              T: 2951262
[00000118]      36      0x7385       BANK Maintenance BEGIN                            T: 2952288
[00000120]      37      0x7393       BANK Maintenance END                              T: 3447042
[00000128]      38      0x7141       STOC Process BEGIN                                T: 3447987
[00000130]      39      0x1C47       M55_HE released                                   T: 3746185
[00000138]      40      0x71DB       TOC Print BEGIN                                   T: 4454487
[00000140]      41      0x71FF       TOC Print END                                     T:21449363
[00000148]      42      0x7727       SERAM boot complete                               T:22005168
[0000014c]      43      0x0A83       Read SOC_ID                                       0x0000B300
[00000154]      44      0x0A95       Override SOC_ID                                   0x0000B400

Available options:
```

Get Power data

This option returns the Power state of the SoC:

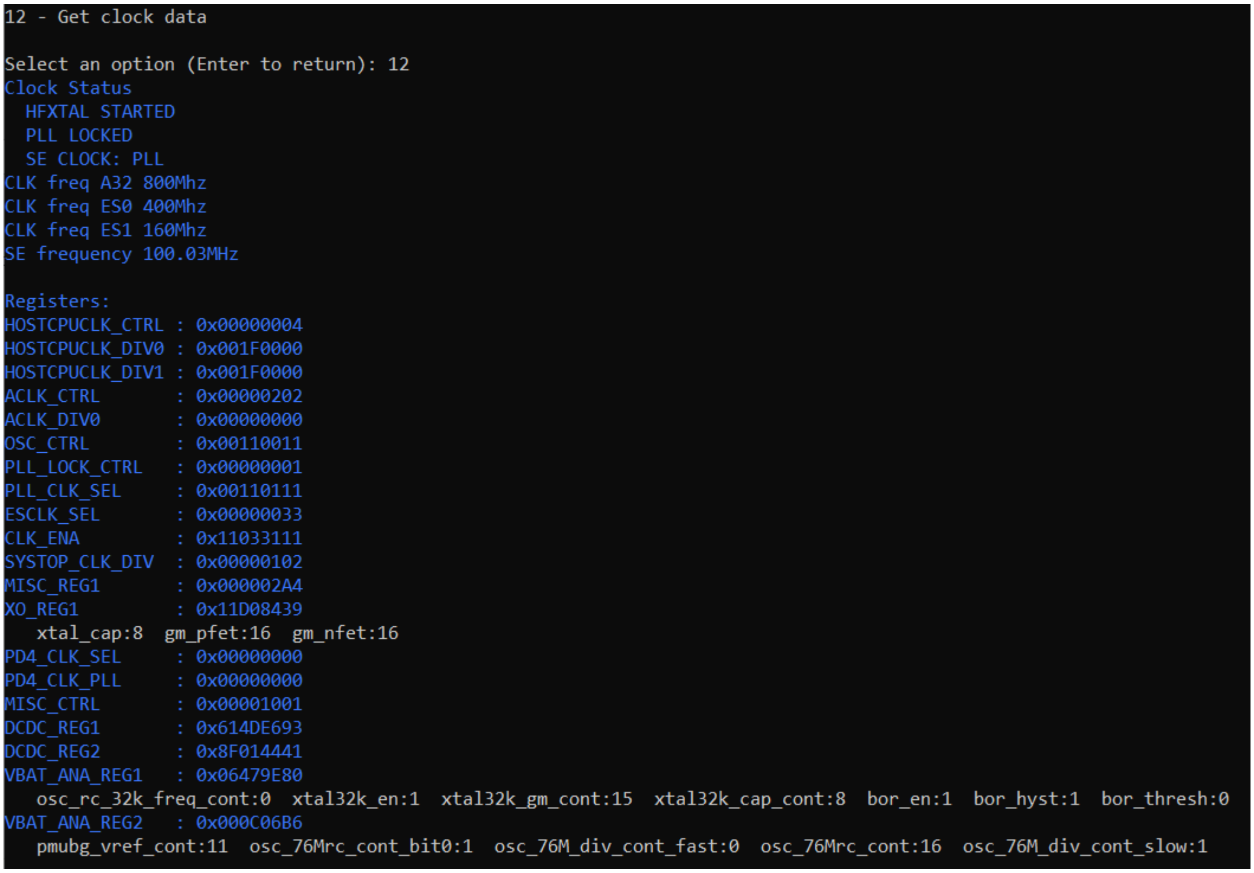


```
Select an option (Enter to return): 11
  es0_ppu_status         0x00000000       OFF
  es1_ppu_status         0x00000008       ON
  se_ppu_status          0x00000008       ON
  fw_ppu_status          0x00000007       FUNC_RET
  systop_ppu_status      0x00000008       ON
  dbgtop_ppu_status      0x00000008       ON
  clustop_ppu_status     0x00000000       OFF
  a32_0_ppu_status       0x00000000       OFF
  a32_1_ppu_status       0x00000000       OFF
  modem_ppu_status       0x00000000       OFF
  modem_aon_status       0x00000000       OFF
  sse700_aon_status      0x00000001       ON
```

NOTE: If you are in Hard Maintenance mode these values will not be accurate as the state changes in this mode.

Get Clock data

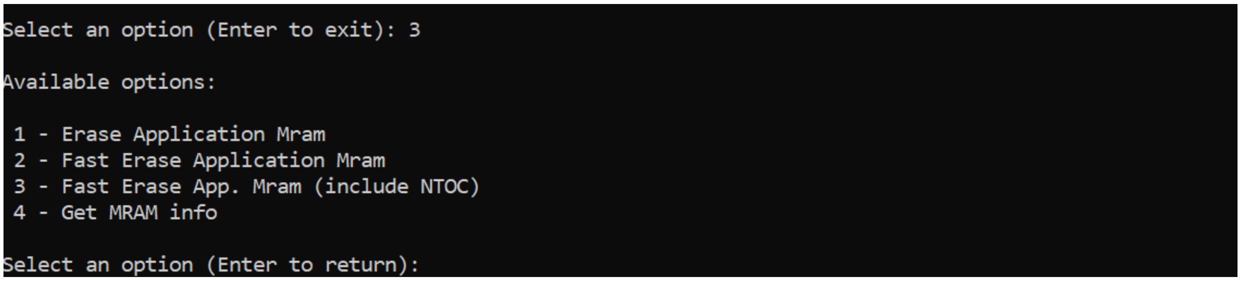
This option returns the Clock and PLL state of the SoC:



```
12 - Get clock data

Select an option (Enter to return): 12
Clock Status
  HFXTAL STARTED
  PLL LOCKED
  SE CLOCK: PLL
CLK freq A32 800Mhz
CLK freq ES0 400Mhz
CLK freq ES1 160Mhz
SE frequency 100.03MHz

Registers:
HOSTCPUCLK_CTRL : 0x00000004
HOSTCPUCLK_DIV0 : 0x001F0000
HOSTCPUCLK_DIV1 : 0x001F0000
ACLK_CTRL       : 0x00000202
ACLK_DIV0       : 0x00000000
OSC_CTRL        : 0x00110011
PLL_LOCK_CTRL   : 0x00000001
PLL_CLK_SEL     : 0x00110111
ESCLK_SEL       : 0x00000033
CLK_ENA         : 0x11033111
SYSTOP_CLK_DIV  : 0x00000102
MISC_REG1       : 0x000002A4
XO_REG1         : 0x11D08439
   xtal_cap:8  gm_pfet:16  gm_nfet:16
PD4_CLK_SEL     : 0x00000000
PD4_CLK_PLL     : 0x00000000
MISC_CTRL       : 0x00001001
DCDC_REG1       : 0x614DE693
DCDC_REG2       : 0x8F014441
VBAT_ANA_REG1   : 0x06479E80
   osc_rc_32k_freq_cont:0  xtal32k_en:1  xtal32k_gm_cont:15  xtal32k_cap_cont:8  bor_en:1  bor_hyst:1  bor_thresh:0
VBAT_ANA_REG2   : 0x000C06B6
   pmubg_vref_cont:11  osc_76Mrc_cont_bit0:1  osc_76M_div_cont_fast:0  osc_76Mrc_cont:16  osc_76M_div_cont_slow:1
```

MRAM

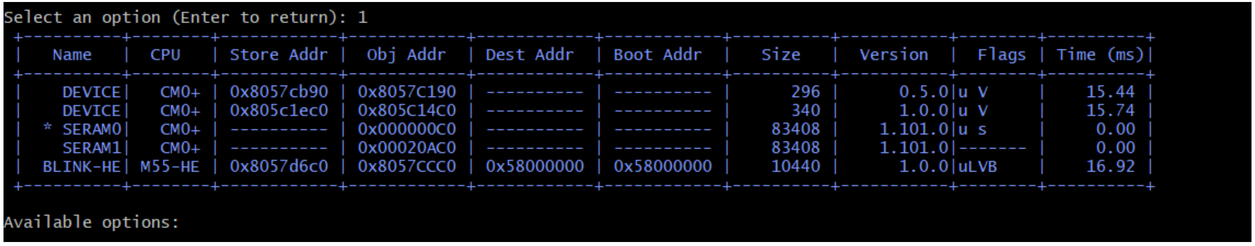
Top level menu group



```
Select an option (Enter to exit): 3

Available options:

 1 - Erase Application Mram
 2 - Fast Erase Application Mram
 3 - Fast Erase App. Mram (include NTOC)
 4 - Get MRAM info

Select an option (Enter to return):
```

NOTE: If you are in Recovery (ROM) mode, these commands will not operate as SEROM does not support these MRAM erase commands.

Erase Application MRAM

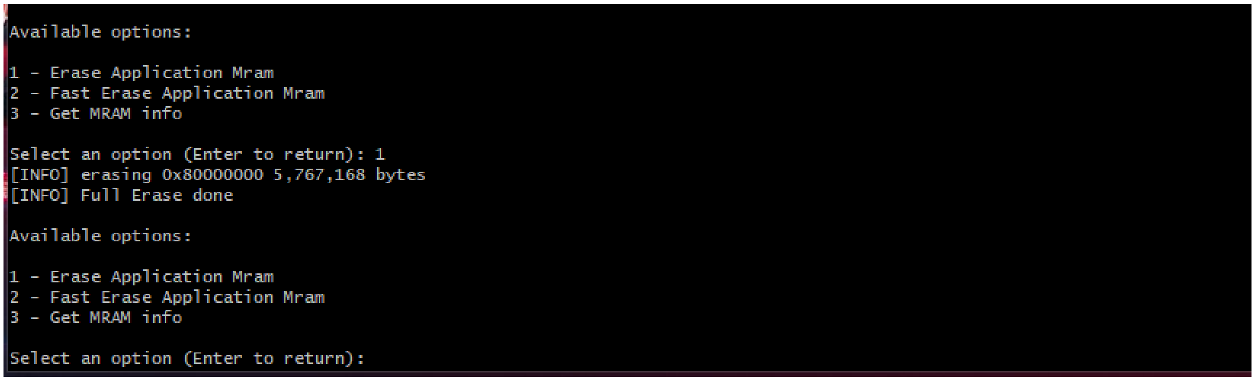


```
Select an option (Enter to return): 1
 +----------+--------+-----------+-----------+-----------+-----------+----------+-----------+--------+----------+
 |   Name   |  CPU   | Store Addr |  Obj Addr  | Dest Addr  | Boot Addr  |   Size   |  Version  |  Flags | Time (ms)|
 +----------+--------+-----------+-----------+-----------+-----------+----------+-----------+--------+----------+
 |    DEVICE|   CM0+ | 0x8057cb90 | 0x8057C190 | ---------- | ---------- |      296 |     0.5.0|u V     |    15.44 |
 |    DEVICE|   CM0+ | 0x805c1ec0 | 0x805C14C0 | ---------- | ---------- |      340 |     1.0.0|u V     |    15.74 |
 |  * SERAM0|   CM0+ | ---------- | 0x000000C0 | ---------- | ---------- |    83408 |   1.101.0|u s     |     0.00 |
 |    SERAM1|   CM0+ | ---------- | 0x00020AC0 | ---------- | ---------- |    83408 |   1.101.0|------- |     0.00 |
 |  BLINK-HE| M55-HE | 0x8057d6c0 | 0x8057CCC0 | 0x58000000 | 0x58000000 |    10440 |     1.0.0|uLVB    |    16.92 |
 +----------+--------+-----------+-----------+-----------+-----------+----------+-----------+--------+----------+

Available options:
```

This option will erase the Application MRAM. It is the equivalent of using the ‘-e app’ option of app-write-mram.

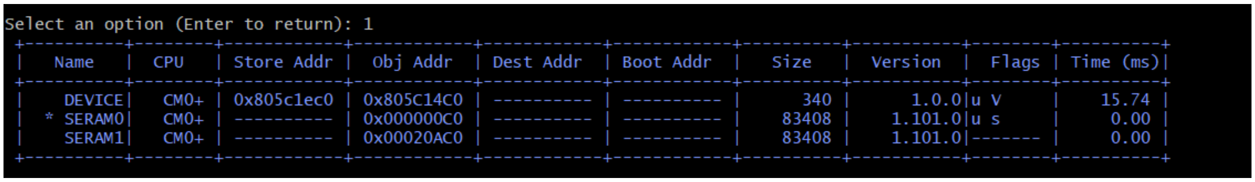
In this example, we see that we have one Application (BLINK-HE) running from Application MRAM.

We will now erase this. Choose the Erase Application MRAM. This will instruct SES to erase the Application MRAM. The size of the MRAM to erase is known as you selected the device type using the tools-config.



```
Available options:

1 - Erase Application Mram
2 - Fast Erase Application Mram
3 - Get MRAM info

Select an option (Enter to return): 1
[INFO] erasing 0x80000000 5,767,168 bytes
[INFO] Full Erase done

Available options:

1 - Erase Application Mram
2 - Fast Erase Application Mram
3 - Get MRAM info

Select an option (Enter to return):
```

If we inspect the TOC information, we can see that the BLINK-HE application has been removed from MRAM.



```
Select an option (Enter to return): 1
 +---------+--------+------------+------------+------------+------------+----------+-----------+--------+----------+
 |   Name  |  CPU   | Store Addr |  Obj Addr  | Dest Addr  | Boot Addr  |   Size   |  Version  |  Flags | Time (ms)|
 +---------+--------+------------+------------+------------+------------+----------+-----------+--------+----------+
 |   DEVICE|   CM0+ | 0x805c1ec0 | 0x805C14C0 | ---------- | ---------- |      340 |      1.0.0|u V     |    15.74 |
 |  * SERAM0|  CM0+ | ---------- | 0x000000C0 | ---------- | ---------- |    83408 |    1.101.0|u s     |     0.00 |
 |   SERAM1|   CM0+ | ---------- | 0x00020AC0 | ---------- | ---------- |    83408 |    1.101.0|------- |     0.00 |
 +---------+--------+------------+------------+------------+------------+----------+-----------+--------+----------+
```

Fast Erase Application MRAM

This option will invalidate the Application TOC without erasing the Application MRAM. While the effect will be the same as the Erase Application MRAM option, this option does not *clear* the Application MRAM area.

Fast Erase Application MRAM (including NTOC)

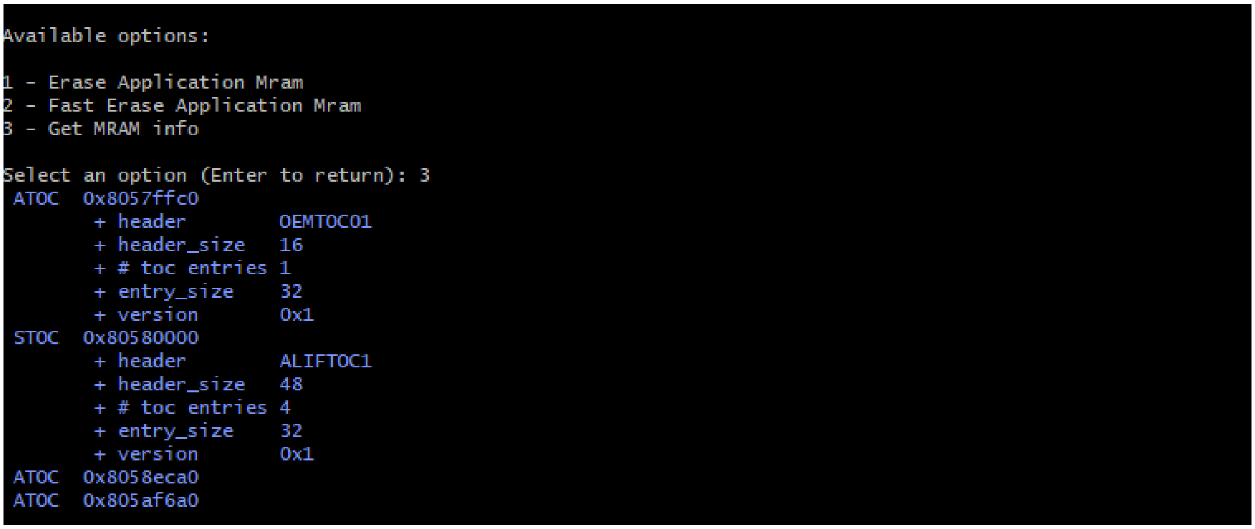
This option will invalidate the Application TOC and erase the first 16 bytes of Application MRAM.

NTOC is the method by which the M55_HE boots from the start of MRAM (address 0x80000000)

This option is needed when the user wants to invalidate a binary previously burnt in MRAM address 0x80000000, otherwise the ATOC has been erased but SES will find still find a valid NTOC and boot it.

Get MRAM info (MRAM Walker)

This walks the entire MRAM memory looking for Table of Contents objects:

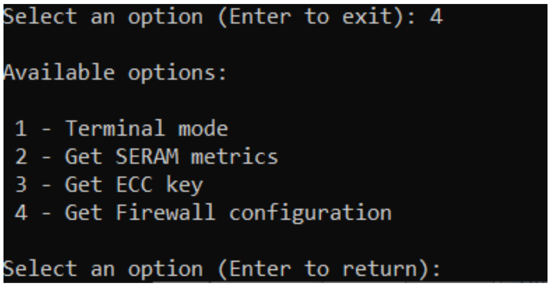


```
Available options:

1 - Erase Application Mram
2 - Fast Erase Application Mram
3 - Get MRAM info

Select an option (Enter to return): 3
 ATOC  0x8057ffc0
        + header        OEMTOC01
        + header_size   16
        + # toc entries 1
        + entry_size    32
        + version       0x1
 STOC  0x80580000
        + header        ALIFTOC1
        + header_size   48
        + # toc entries 4
        + entry_size    32
        + version       0x1
 ATOC  0x8058eca0
 ATOC  0x805af6a0
```

In this example, two System TOC objects were found in MRAM.

Utilities

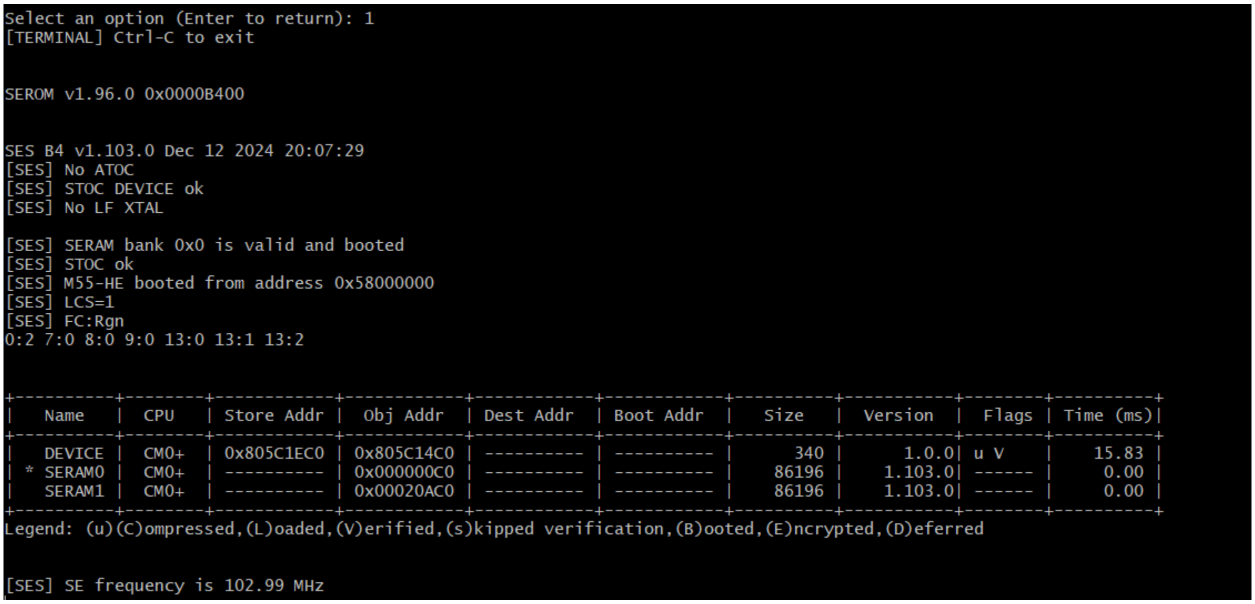
Top level menu group



```
Select an option (Enter to exit): 4

Available options:

 1 - Terminal mode
 2 - Get SERAM metrics
 3 - Get ECC key
 4 - Get Firewall configuration

Select an option (Enter to return):
```

Terminal Mode

This option echoes the SE-UART output.

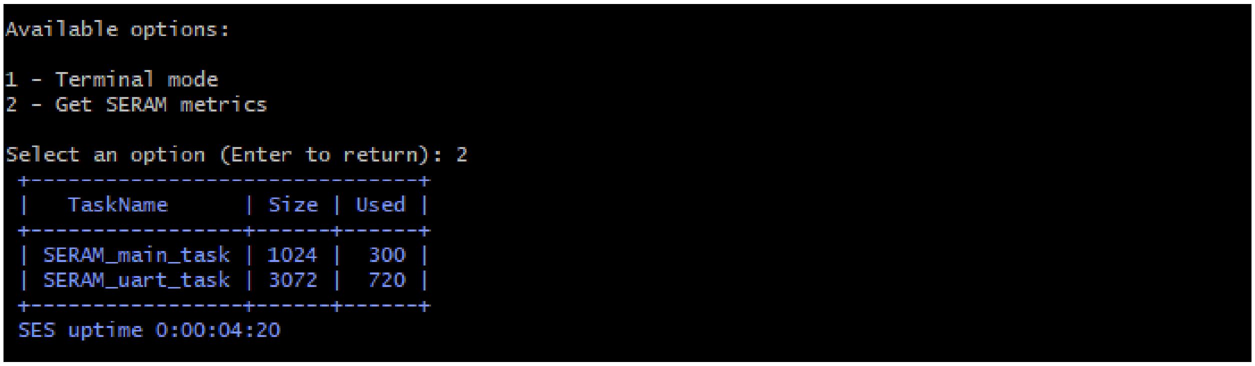


```
Select an option (Enter to return): 1
[TERMINAL] Ctrl-C to exit


SEROM v1.96.0 0x0000B400


SES B4 v1.103.0 Dec 12 2024 20:07:29
[SES] No ATOC
[SES] STOC DEVICE ok
[SES] No LF XTAL

[SES] SERAM bank 0x0 is valid and booted
[SES] STOC ok
[SES] M55-HE booted from address 0x58000000
[SES] LCS=1
[SES] FC:Rgn
0:2 7:0 8:0 9:0 13:0 13:1 13:2


+----------+--------+------------+------------+------------+------------+-----------+-----------+--------+----------+
|   Name   |  CPU   | Store Addr |  Obj Addr  | Dest Addr  | Boot Addr  |   Size    |  Version  |  Flags | Time (ms)|
+----------+--------+------------+------------+------------+------------+-----------+-----------+--------+----------+
|   DEVICE |  CM0+  | 0x805C1EC0 | 0x805C14C0 | ---------- | ---------- |       340 |      1.0.0| u V    |    15.83 |
| * SERAM0 |  CM0+  | ---------- | 0x000000C0 | ---------- | ---------- |     86196 |    1.103.0| ------ |     0.00 |
|   SERAM1 |  CM0+  | ---------- | 0x00020AC0 | ---------- | ---------- |     86196 |    1.103.0| ------ |     0.00 |
+----------+--------+------------+------------+------------+------------+-----------+-----------+--------+----------+
Legend: (u)(C)ompressed,(L)oaded,(V)erified,(s)kipped verification,(B)ooted,(E)ncrypted,(D)eferred


[SES] SE frequency is 102.99 MHz
```

In this example, [RESET] was pressed on the board, and we see the complete boot up information echoed.

To exit this mode press CTRL-C and it will drop back to the maintenance menu.

SES Metrics



```
Available options:

1 - Terminal mode
2 - Get SERAM metrics

Select an option (Enter to return): 2
 +--------------------------------+
 |   TaskName       | Size | Used |
 +------------------+------+------+
 | SERAM_main_task  | 1024 |  300 |
 | SERAM_uart_task  | 3072 |  720 |
 +------------------+------+------+
 SES uptime 0:00:04:20
```

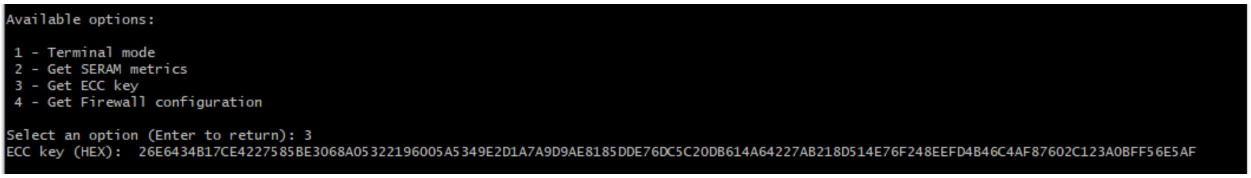
Returns (for now) stack information details used by SES as well as the uptime since SES booted. The uptime format is Days:Hours:Minutes:Seconds

Get address/Set address

These diagnostic commands are not supported.

Get ECC key

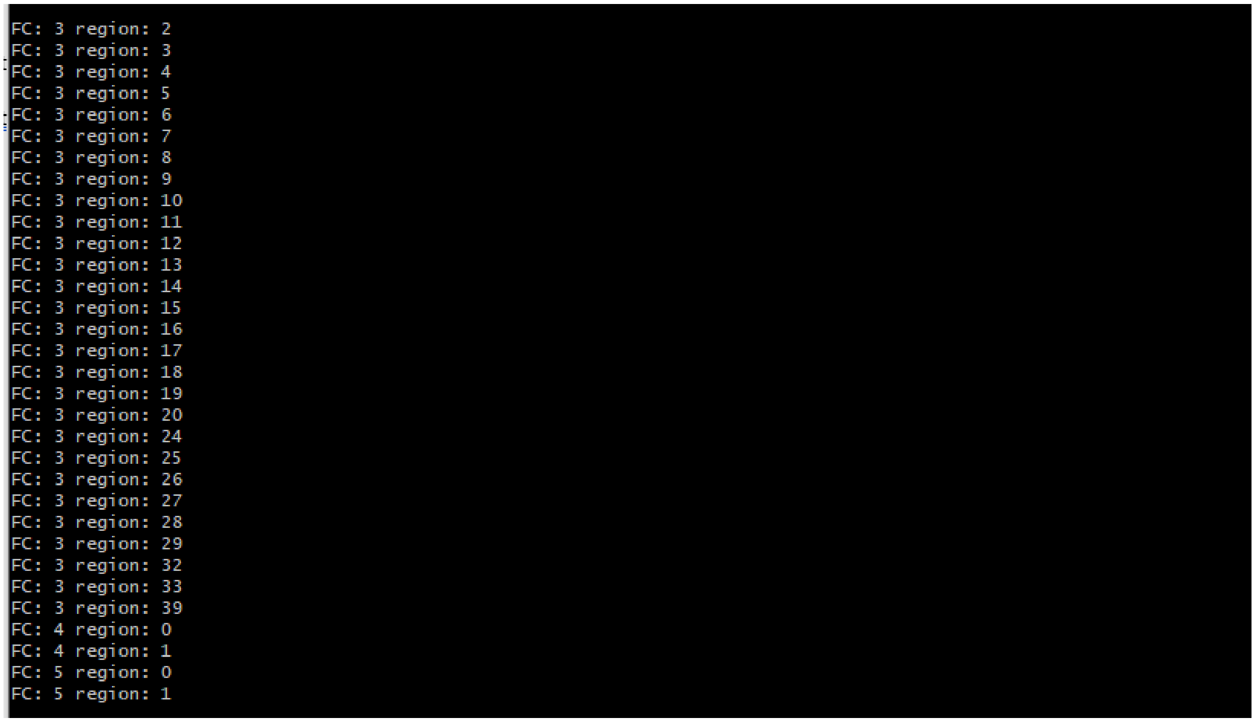
This option returns the public ECC key of the device. If the ECC key is not provisioned yet, all 0's is returned.



```
Available options:

 1 - Terminal mode
 2 - Get SERAM metrics
 3 - Get ECC key
 4 - Get Firewall configuration

Select an option (Enter to return): 3
ECC key (HEX):  26E6434B17CE4227585BE3068A05322196005A5349E2D1A7A9D9AE8185DDE76DC5C20DB614A64227AB218D514E76F248EEFD4B46C4AF87602C123A0BFF56E5AF
```

Get Firewall configuration

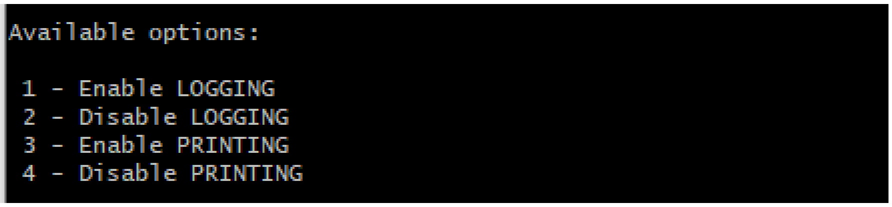
This option returns the current Firewall configuration status, i.e., what regions have been configured in which Firewall Components ('FC: x').



```
FC: 3 region: 2
FC: 3 region: 3
FC: 3 region: 4
FC: 3 region: 5
FC: 3 region: 6
FC: 3 region: 7
FC: 3 region: 8
FC: 3 region: 9
FC: 3 region: 10
FC: 3 region: 11
FC: 3 region: 12
FC: 3 region: 13
FC: 3 region: 14
FC: 3 region: 15
FC: 3 region: 16
FC: 3 region: 17
FC: 3 region: 18
FC: 3 region: 19
FC: 3 region: 20
FC: 3 region: 24
FC: 3 region: 25
FC: 3 region: 26
FC: 3 region: 27
FC: 3 region: 28
FC: 3 region: 29
FC: 3 region: 32
FC: 3 region: 33
FC: 3 region: 39
FC: 4 region: 0
FC: 4 region: 1
FC: 5 region: 0
FC: 5 region: 1
```

Setting capabilities

Top level menu group



```
Available options:

 1 - Enable LOGGING
 2 - Disable LOGGING
 3 - Enable PRINTING
 4 - Disable PRINTING
```

Enable / Disable LOGGING

Controls whether the SES messages are added to the log buffer. The log buffer can be read using the 'Device Information' option.

Enable / Disable PRINTING

Toggle PRINTING – controls whether the SERAM messages are printed directly to the SE UART or put in a buffer from where they are periodically printed out.

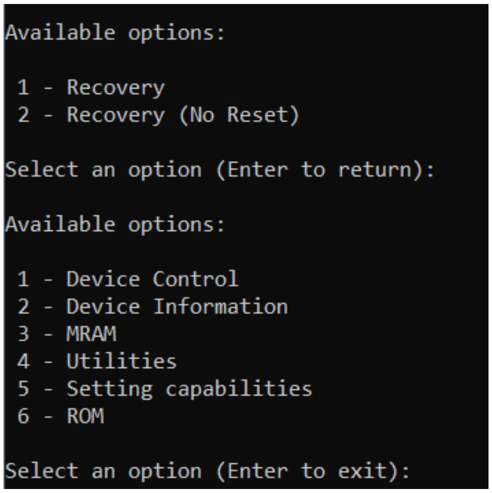
ROM

Devices support a limited subset of ISP commands built into the ROM.

If the NVM contains no viable SES image to load, SEROM will enter an ISP mode to allow recovery.

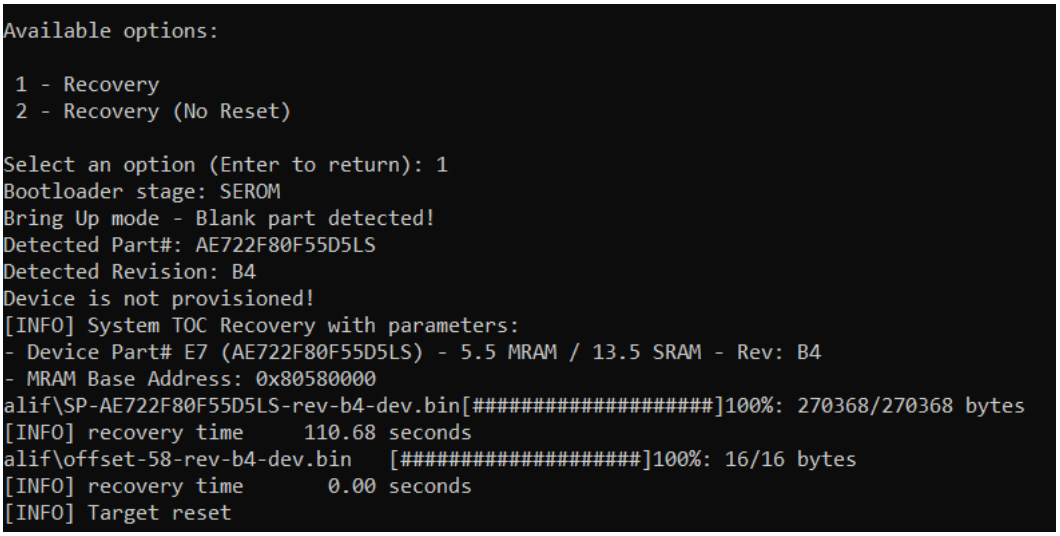
Recovery mode will take an SES image and write it to the NVM using a specific ISP protocol supported by SEROM, it is not the same as used for wiring the Application MRAM.

Top level menu group



```
Available options:

 1 - Recovery
 2 - Recovery (No Reset)

Select an option (Enter to return):

Available options:

 1 - Device Control
 2 - Device Information
 3 - MRAM
 4 - Utilities
 5 - Setting capabilities
 6 - ROM

Select an option (Enter to exit):
```

Recovery Session



```
Available options:

 1 - Recovery
 2 - Recovery (No Reset)

Select an option (Enter to return): 1
Bootloader stage: SEROM
Bring Up mode - Blank part detected!
Detected Part#: AE722F80F55D5LS
Detected Revision: B4
Device is not provisioned!
[INFO] System TOC Recovery with parameters:
- Device Part# E7 (AE722F80F55D5LS) - 5.5 MRAM / 13.5 SRAM - Rev: B4
- MRAM Base Address: 0x80580000
alif\SP-AE722F80F55D5LS-rev-b4-dev.bin[####################]100%: 270368/270368 bytes
[INFO] recovery time     110.68 seconds
alif\offset-58-rev-b4-dev.bin   [####################]100%: 16/16 bytes
[INFO] recovery time       0.00 seconds
[INFO] Target reset
```

The device is in recovery mode and has loaded the SES image via SEROM into MRAM.

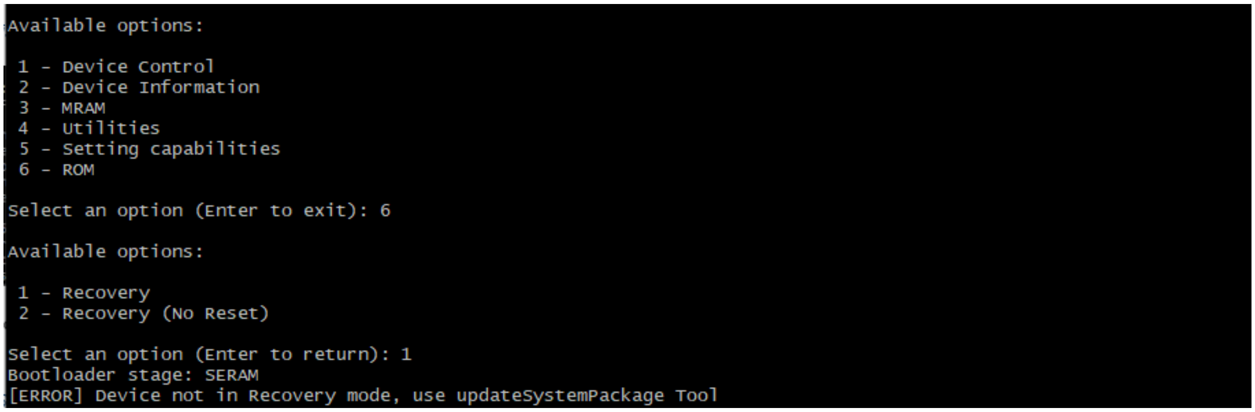
Recovery Session – No Reset

By default, after the recovery operation is performed then the tool will send a request to perform a RESET of the platform.

There is also an option to NOT perform this reset after recovery. This can be useful for observing potential failures that would be masked by the actual reset operation.

An example of this would be a device failing with say a CC312 error code (0x0B000006 - illegal KCEICV) because of the automatic reset, and the user resetting the board one more time after that, both SERAM banks are being marked as invalid, so the reported error code is 0x26 in the recovery screen, which hides the actual issue.

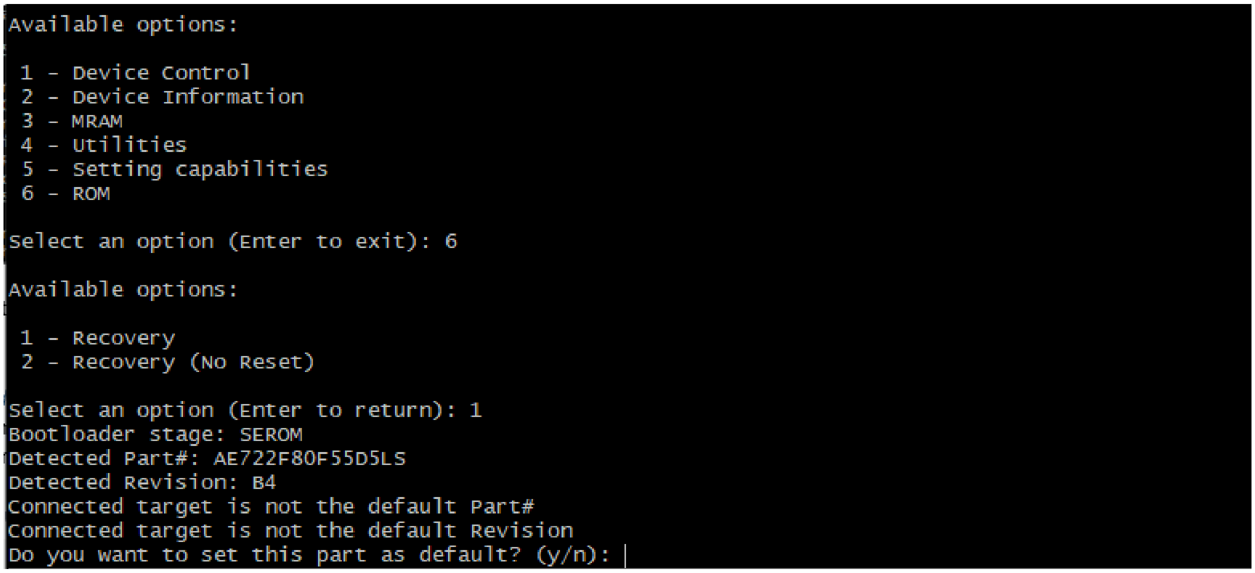
Target not in recovery mode



```
Available options:

 1 - Device Control
 2 - Device Information
 3 - MRAM
 4 - Utilities
 5 - Setting capabilities
 6 - ROM

Select an option (Enter to exit): 6

Available options:

 1 - Recovery
 2 - Recovery (No Reset)

Select an option (Enter to return): 1
Bootloader stage: SERAM
[ERROR] Device not in Recovery mode, use updateSystemPackage Tool
```

This message is shown if the target device is not in recovery mode probably because it is running SES already.

Target device mismatch



```
Available options:

 1 - Device Control
 2 - Device Information
 3 - MRAM
 4 - Utilities
 5 - Setting capabilities
 6 - ROM

Select an option (Enter to exit): 6

Available options:

 1 - Recovery
 2 - Recovery (No Reset)

Select an option (Enter to return): 1
Bootloader stage: SEROM
Detected Part#: AE722F80F55D5LS
Detected Revision: B4
Connected target is not the default Part#
Connected target is not the default Revision
Do you want to set this part as default? (y/n): |
```

This message shows that the target Device is different from the one selected by tools-config for this session.

Recovery UART Speed

When in RECOVERY mode the device is running ISP from the ROM which runs at RC frequency, so you will notice it is slower than when SES is performing MRAM updates.

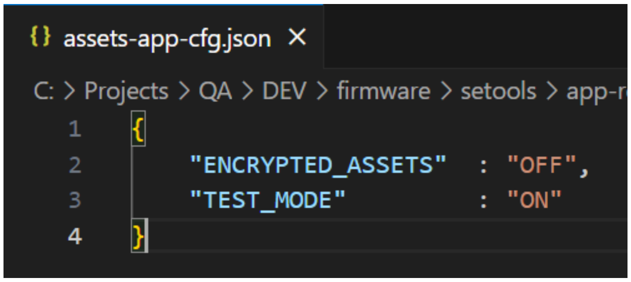
- There is no PLL support inside the ROM code so the baud rate cannot be increased like it is when SES is doing updates (Using app-mram-write, updateSystemPackage).
- ISP protocol used in the ROM is MRAM line based i.e. 128bits at a time with no buffering.

app-assets-gen

The app-assets-gen tool generates the “assets package” required to provision the device with the app-provision tool (see next section).

This file is loaded by the app-provision tool and contains assets and options for the provisioning. Provisioning means keys and other values to be programmed (OTP) into the device.

The options can be configured with a JSON file. Default configuration file is `build\config\assets-app-cfg.json`



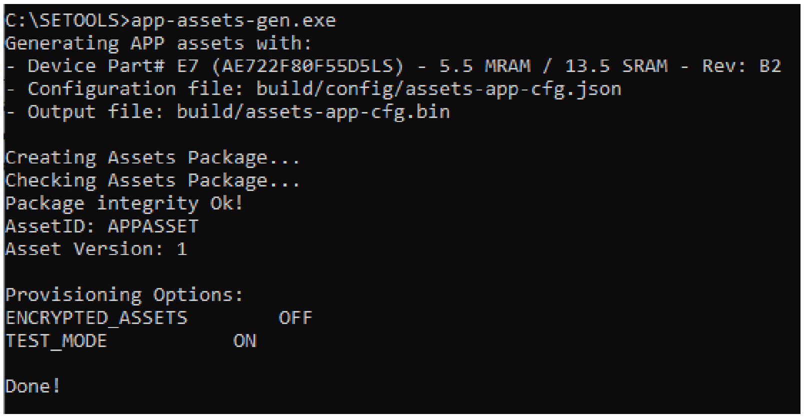
```
{
    "ENCRYPTED_ASSETS"  : "OFF",
    "TEST_MODE"         : "ON"
}
```

Options description:

- ENCRYPTED_ASSETS:
    - Defines if the assets are encrypted or not. Unencrypted assets should only be used in trusted environments.
    - NOTE: Encrypted assets are not supported yet.
- TEST_MODE:
    - This option DOES NOT burn the OTP area, so the program can run multiple times without advancing the LCS to SE.
    - This is provided to test the process without affecting the device.

Once the desired configuration is done, the assets package can be generated using the tool:

```
$ app-assets-gen (-h for help)
```

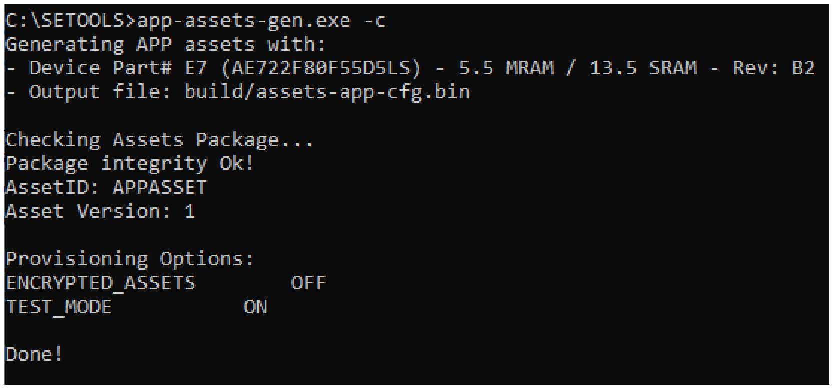


```
C:\SETOOLS>app-assets-gen.exe
Generating APP assets with:
- Device Part# E7 (AE722F80F55D5LS) - 5.5 MRAM / 13.5 SRAM - Rev: B2
- Configuration file: build/config/assets-app-cfg.json
- Output file: build/assets-app-cfg.bin

Creating Assets Package...
Checking Assets Package...
Package integrity Ok!
AssetID: APPASSET
Asset Version: 1

Provisioning Options:
ENCRYPTED_ASSETS        OFF
TEST_MODE           ON

Done!
```

The command generates the assets package and checks the generated binary to extract the Options selected (it reads back the options from the binary, not from the json file).

Also, using the –c flag, the tool will skip the assets package generation, and it just checks the options used to generate it.

$ app-assets-gen -c



```
C:\SETOOLS>app-assets-gen.exe -c
Generating APP assets with:
- Device Part# E7 (AE722F80F55D5LS) - 5.5 MRAM / 13.5 SRAM - Rev: B2
- Output file: build/assets-app-cfg.bin

Checking Assets Package...
Package integrity Ok!
AssetID: APPASSET
Asset Version: 1

Provisioning Options:
ENCRYPTED_ASSETS        OFF
TEST_MODE           ON

Done!
```

Using the –f option, we can specify an input file (.json format if assets package is being generated, or .bin file if only checking the options used to generate the binary).

Example:

$ app-assets-gen -f build\config\assets-cfg.json
$ app-assets-gen -c –f build\assets-cfg.bin

The output file is always created in the build\ folder and the name will be the same as the configuration file.

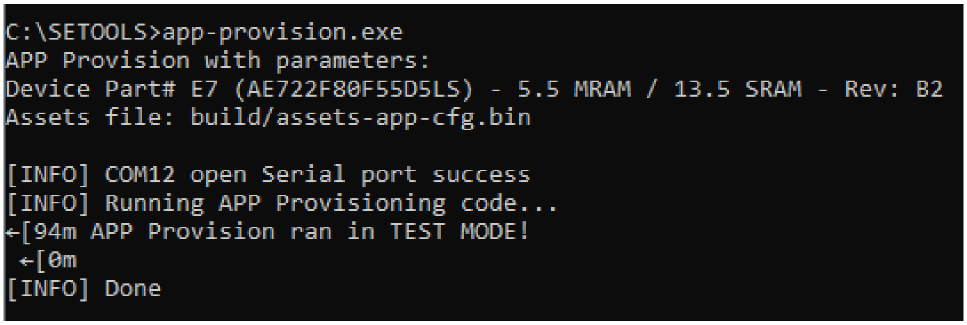
app-provision

The app-provision tool will inject user assets (keys and secrets) into the device and move the device from DM (Device Manufacturing) to SE (Secure Enable) LCS (Life-Cycle State).

The SE LCS is used for devices from the manufacturing line and “in the field”. It permits the execution of security functions but blocks all debugging and testing capabilities.

Using Secure Boot is mandatory in this SE LCS.

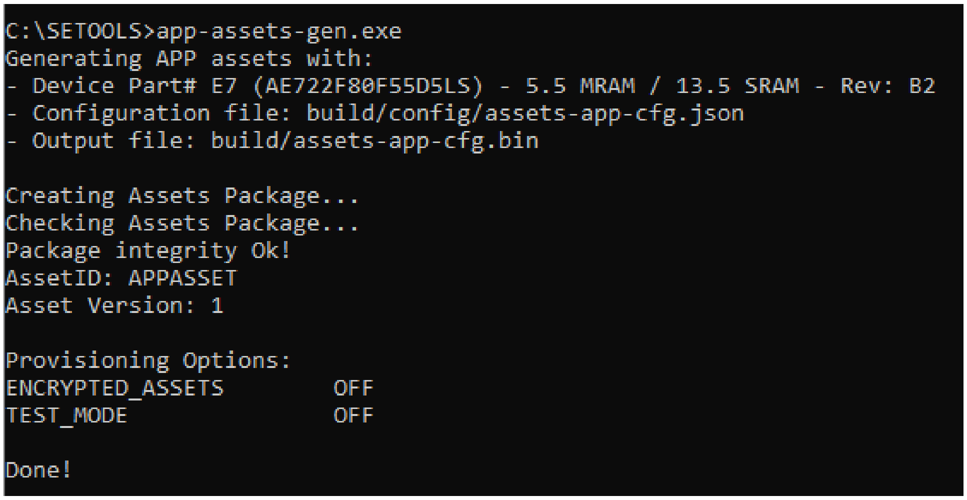
```
$ app-provision  (-h for help)
```



```
C:\SETOOLS>app-provision.exe
APP Provision with parameters:
Device Part# E7 (AE722F80F55D5LS) - 5.5 MRAM / 13.5 SRAM - Rev: B2
Assets file: build/assets-app-cfg.bin

[INFO] COM12 open Serial port success
[INFO] Running APP Provisioning code...
←[94m APP Provision ran in TEST MODE!
 ←[0m
[INFO] Done
```

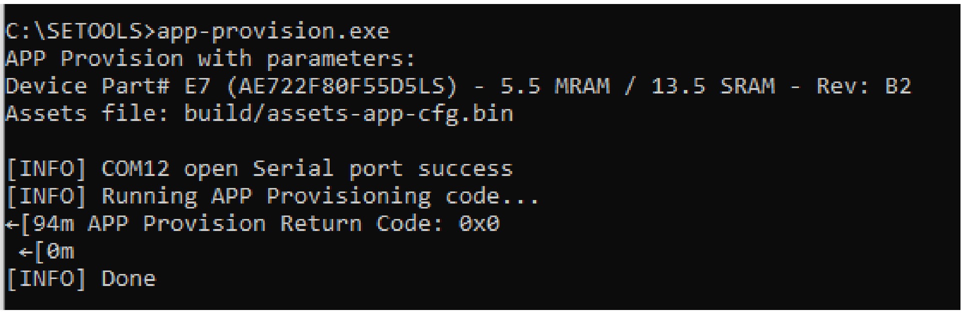
The default assets file is `build\assets-app-cfg.bin`. Use the –a option to specify other assets file.

To perform a real provision, configure TEST_MODE OFF in the assets configuration file, and re-build the assets using the app-assets-gen tool;



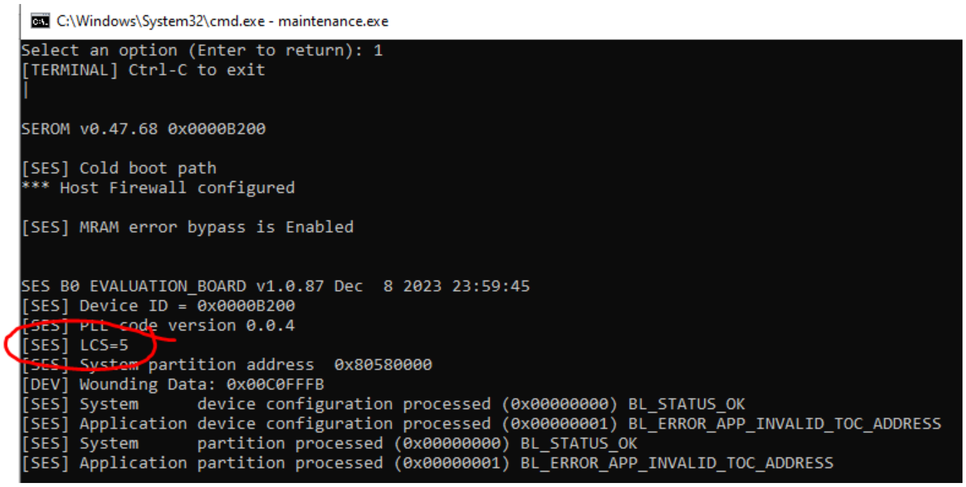
```
C:\SETOOLS>app-assets-gen.exe
Generating APP assets with:
- Device Part# E7 (AE722F80F55D5LS) - 5.5 MRAM / 13.5 SRAM - Rev: B2
- Configuration file: build/config/assets-app-cfg.json
- Output file: build/assets-app-cfg.bin

Creating Assets Package...
Checking Assets Package...
Package integrity Ok!
AssetID: APPASSET
Asset Version: 1

Provisioning Options:
ENCRYPTED_ASSETS        OFF
TEST_MODE               OFF

Done!
```

Running the app-provision tool with this new assets binary, it will provision the device;



```
C:\SETOOLS>app-provision.exe
APP Provision with parameters:
Device Part# E7 (AE722F80F55D5LS) - 5.5 MRAM / 13.5 SRAM - Rev: B2
Assets file: build/assets-app-cfg.bin

[INFO] COM12 open Serial port success
[INFO] Running APP Provisioning code...
←[94m APP Provision Return Code: 0x0
 ←[0m
[INFO] Done
```

Once this script finishes, the device will advance from DM to SE (Secure Enable) LCS, after reset.



C:\Windows\System32\cmd.exe - maintenance.exe

```
Select an option (Enter to return): 1
[TERMINAL] Ctrl-C to exit
|

SEROM v0.47.68 0x0000B200

[SES] Cold boot path
*** Host Firewall configured

[SES] MRAM error bypass is Enabled


SES B0 EVALUATION_BOARD v1.0.87 Dec  8 2023 23:59:45
[SES] Device ID = 0x0000B200
[SES] PLL code version 0.0.4
[SES] LCS=5
[SES] System partition address  0x80580000
[DEV] Wounding Data: 0x00C0FFFB
[SES] System      device configuration processed (0x00000000) BL_STATUS_OK
[SES] Application device configuration processed (0x00000001) BL_ERROR_APP_INVALID_TOC_ADDRESS
[SES] System      partition processed (0x00000000) BL_STATUS_OK
[SES] Application partition processed (0x00000001) BL_ERROR_APP_INVALID_TOC_ADDRESS
```

app-secure-debug (--rma)

A device in SE LCS (Secure State) blocks all debugging and testing capabilities. The Secure Debug tool gives the user a mechanism to:

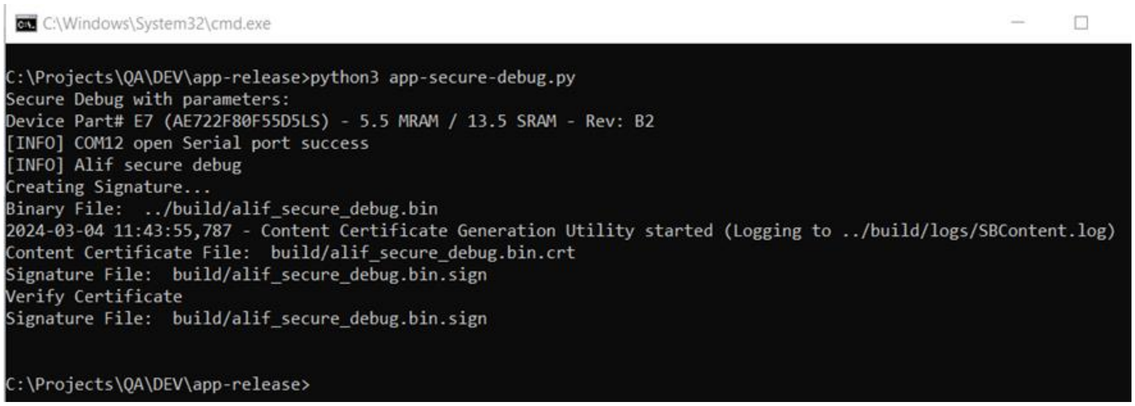
- Open debugger access to the APP cores and the related debugging infrastructure, in a device in SE LCS.

- Initiate the process of transitioning a device to RMA LCS (end of product cycle), before sending the device to Alif for failure analysis or further inspection.

Both operations are done by generating a secure debug certificate and sending it to the device via ISP. The secure debug certificate is signed using the same keys as the ones used when generating the ATOC, i.e. for secure booting. The certificates are session-specific and become invalid after the device is reset.

To open debugger access to the APP cores, run:

```
$ app-secure-debug  (-h for help)
```

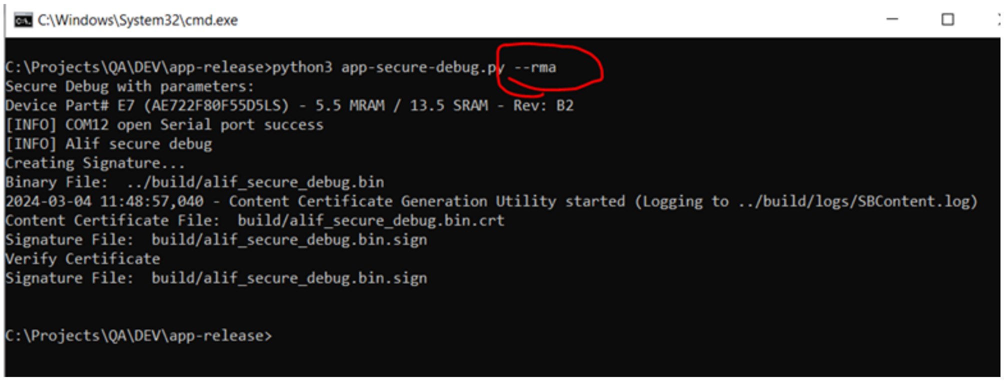


```
C:\Projects\QA\DEV\app-release>python3 app-secure-debug.py
Secure Debug with parameters:
Device Part# E7 (AE722F80F55D5LS) - 5.5 MRAM / 13.5 SRAM - Rev: B2
[INFO] COM12 open Serial port success
[INFO] Alif secure debug
Creating Signature...
Binary File:  ../build/alif_secure_debug.bin
2024-03-04 11:43:55,787 - Content Certificate Generation Utility started (Logging to ../build/logs/SBContent.log)
Content Certificate File:  build/alif_secure_debug.bin.crt
Signature File:  build/alif_secure_debug.bin.sign
Verify Certificate
Signature File:  build/alif_secure_debug.bin.sign

C:\Projects\QA\DEV\app-release>
```

After this command, a debugger can connect to the application cores and do a debug session.

To initiate the transition to RMA LCS, run:

```
$ app-secure-debug --rma
```



```
C:\Projects\QA\DEV\app-release>python3 app-secure-debug.py --rma
Secure Debug with parameters:
Device Part# E7 (AE722F80F55D5LS) - 5.5 MRAM / 13.5 SRAM - Rev: B2
[INFO] COM12 open Serial port success
[INFO] Alif secure debug
Creating Signature...
Binary File:  ../build/alif_secure_debug.bin
2024-03-04 11:48:57,040 - Content Certificate Generation Utility started (Logging to ../build/logs/SBContent.log)
Content Certificate File:  build/alif_secure_debug.bin.crt
Signature File:  build/alif_secure_debug.bin.sign
Verify Certificate
Signature File:  build/alif_secure_debug.bin.sign

C:\Projects\QA\DEV\app-release>
```

After this command, the device is still in its current LCS (SE or DM) but the RMA transition is initiated. The device can then be sent to Alif, where the transition to RMA LCS will be completed.

Tool Exit codes

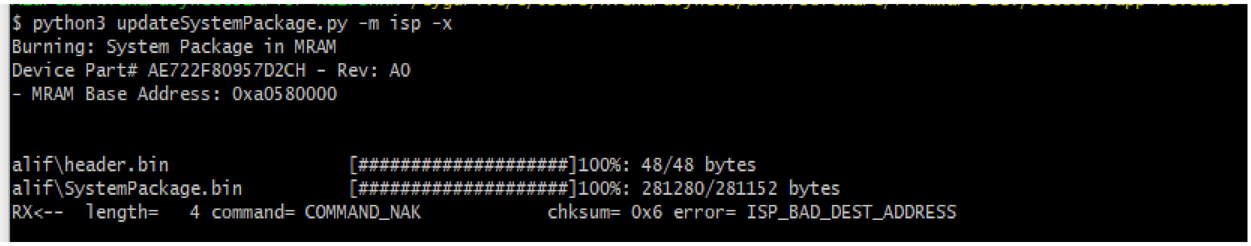
The tools will return the following runtime execution Error codes.

0	Indicates success.
1	Indicates failure.

NACK Error Handling

The ISP protocol uses an Acknowledge (ACK) / Not Acknowledge (NACK) protocol scheme. If a NACK is seen, then it will be followed by an Error code.

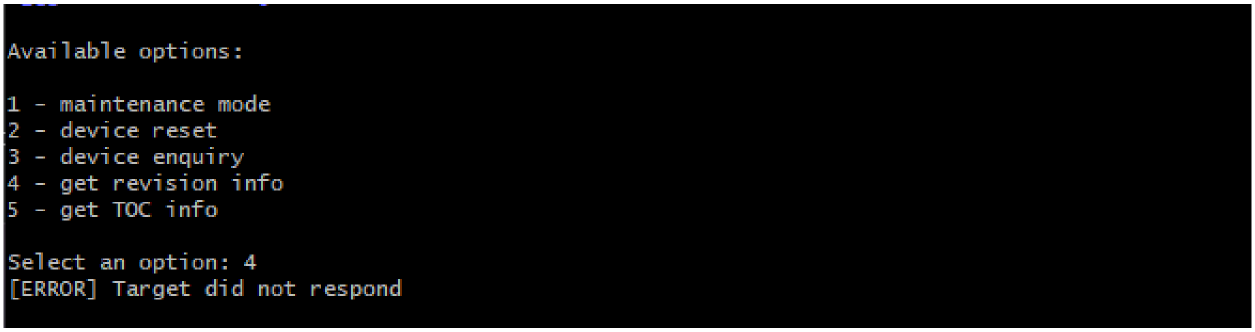
If a NACK packet is seen it will always be printed, and the tool will EXIT.



```
$ python3 updateSystemPackage.py -m isp -x
Burning: System Package in MRAM
Device Part# AE722F80957D2CH - Rev: A0
- MRAM Base Address: 0xa0580000


alif\header.bin              [####################]100%: 48/48 bytes
alif\SystemPackage.bin       [####################]100%: 281280/281152 bytes
RX<--  length=   4 command= COMMAND_NAK           chksum= 0x6 error= ISP_BAD_DEST_ADDRESS
```

TIMEOUT Error Handling

An ISP session requires an initial ISP START command sent to the Target to put it into ISP mode. If this command fails, then the ISP session will not occur, and an Error is reported that the Target did not respond



```
Available options:

1 - maintenance mode
2 - device reset
3 - device enquiry
4 - get revision info
5 - get TOC info

Select an option: 4
[ERROR] Target did not respond
```

This Error will exit the program.

Possible causes for this Error:

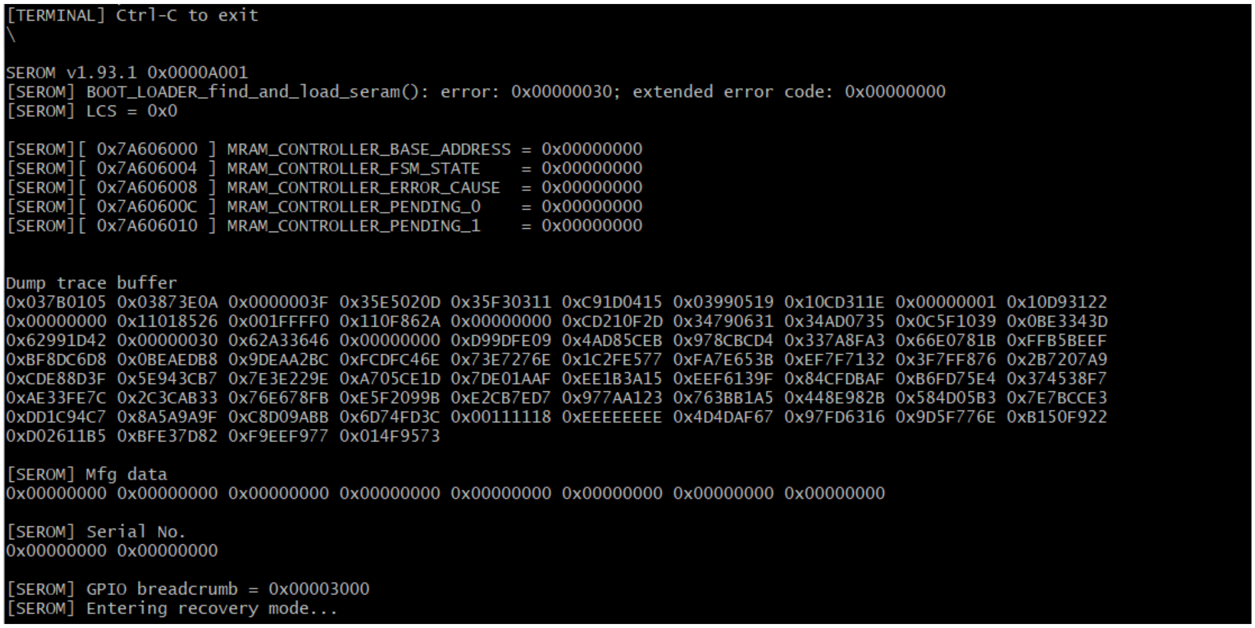
- Incorrect baud rate setting
- Target is not powered or is in a low power state such as STOP mode.

7. RECOVERY MODE

Recovery mode occurs when SEROM

- Fails to find an SES image in MRAM that it can boot.
- Fails to verify an SES image from MRAM.
- Fails to decrypt an SES image from MRAM.
- Finds the dual SES Banks are both marked as “bad”.

If your device is in Recovery mode, you will see the following SE-UART output



```
[TERMINAL] Ctrl-C to exit
\

SEROM v1.93.1 0x0000A001
[SEROM] BOOT_LOADER_find_and_load_seram(): error: 0x00000030; extended error code: 0x00000000
[SEROM] LCS = 0x0

[SEROM][ 0x7A606000 ] MRAM_CONTROLLER_BASE_ADDRESS = 0x00000000
[SEROM][ 0x7A606004 ] MRAM_CONTROLLER_FSM_STATE    = 0x00000000
[SEROM][ 0x7A606008 ] MRAM_CONTROLLER_ERROR_CAUSE  = 0x00000000
[SEROM][ 0x7A60600C ] MRAM_CONTROLLER_PENDING_0    = 0x00000000
[SEROM][ 0x7A606010 ] MRAM_CONTROLLER_PENDING_1    = 0x00000000


Dump trace buffer
0x037B0105 0x03873E0A 0x0000003F 0x35E5020D 0x35F30311 0xC91D0415 0x03990519 0x10CD311E 0x00000001 0x10D93122
0x00000000 0x11018526 0x001FFFF0 0x110F862A 0x00000000 0xCD210F2D 0x34790631 0x34AD0735 0x0C5F1039 0x0BE3343D
0x62991D42 0x00000030 0x62A33646 0x00000000 0xD99DFE09 0x4AD85CEB 0x978CBCD4 0x337A8FA3 0x66E0781B 0xFFB5BEEF
0xBF8DC6D8 0x0BEAEDB8 0x9DEAA2BC 0xFCDFC46E 0x73E7276E 0x1C2FE577 0xFA7E653B 0xEF7F7132 0x3F7FF876 0x2B7207A9
0xCDE88D3F 0x5E943CB7 0x7E3E229E 0xA705CE1D 0x7DE01AAF 0xEE1B3A15 0xEEF6139F 0x84CFDBAF 0xB6FD75E4 0x374538F7
0xAE33FE7C 0x2C3CAB33 0x76E678FB 0xE5F2099B 0xE2CB7ED7 0x977AA123 0x763BB1A5 0x448E982B 0x584D05B3 0x7E7BCCE3
0xDD1C94C7 0x8A5A9A9F 0xC8D09ABB 0x6D74FD3C 0x00111118 0xEEEEEEEE 0x4D4DAF67 0x97FD6316 0x9D5F776E 0xB150F922
0xD02611B5 0xBFE37D82 0xF9EEF977 0x014F9573

[SEROM] Mfg data
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000

[SEROM] Serial No.
0x00000000 0x00000000

[SEROM] GPIO breadcrumb = 0x00003000
[SEROM] Entering recovery mode...
```

This means you are now back in SEROM. There are two error codes printed error and extended error.

SEROM Error codes are described below.
Extended error codes are for reporting issues (specifically) with the CC312 Crypto Engine.

SEROM Error Codes

```
// Error codes
#define SEROM_STATUS_SUCCESS                          0x0

// Crypto errors
#define SEROM_BSV_INIT_FAIL                           0x1
#define SEROM_BSV_LCS_GET_AND_INIT_FAIL               0x2
#define SEROM_BSV_LCS_GET_FAIL                        0x2
#define SEROM_BSV_SEC_MODE_SET_FAIL                   0x3
#define SEROM_BSV_PRIV_MODE_SET_FAIL                  0x4
#define SEROM_BSV_CORE_CLK_GATING_ENABLE_FAIL         0x5
```

```
// MRAM errors
#define SEROM_MRAM_INITIALIZATION_FAILURE             0x6
#define SEROM_MRAM_INITIALIZATION_TIMEOUT             0x7
#define SEROM_MRAM_WRITE_FAILURE                      0x8


// ATOC errors
#define SEROM_ATOC_EXT_HDR_OFFSET_ZERO                0x9
#define SEROM_ATOC_EXT_HDR_OFFSET_TOO_LARGE           0xA
#define SEROM_ATOC_OBJECT_OFFSET_ZERO                 0xB
#define SEROM_ATOC_OBJECT_OFFSET_MISALIGNED           0xC
#define SEROM_ATOC_OBJECT_OFFSET_TOO_LARGE            0xD
#define SEROM_ATOC_OBJECT_OFFSET_TOO_SMALL            0xE
#define SEROM_ATOC_EXT_HDR_OFFSET_MISALIGNED          0xF
#define SEROM_ATOC_HEADER_OFFSET_INVALID              0x10
#define SEROM_ATOC_HEADER_CRC32_ERROR                 0x11
#define SEROM_ATOC_HEADER_STRING_INVALID              0x12
#define SEROM_ATOC_NUM_TOC_ENTRIES_INVALID            0x13

// Certificate errors
#define SEROM_CONTENT_CERTIFICATE_NULL                0x14
#define SEROM_CERTIFICATE_NULL                        0x15
#define SEROM_CERTIFICATE_CHAIN_INVALID               0x16
#define SEROM_INVALID_OEM_ROT                         0x17
#define SEROM_CERTIFICATE_ERROR_BASE                  0x18
#define SEROM_CERTIFICATE_1_ERROR                     0x19
#define SEROM_CERTIFICATE_2_ERROR                     0x1A
#define SEROM_CERTIFICATE_3_ERROR                     0x1B

// BOOT errors
#define SEROM_BOOT_CODE_LOAD_ADDR_INVALID             0x1C
#define SEROM_BOOT_VERIFY_IN_MEMORY_CASE_INVALID      0x1D
#define SEROM_BOOT_ZERO_IMAGE_LENGTH_INVALID          0x1E
#define SEROM_BOOT_ENCRYPTED_IMAGE_INVALID            0x1F
#define SEROM_BOOT_VERIFY_IN_FLASH_CASE_INVALID       0x20
#define SEROM_BOOT_IMAGE_LENGTH_TOO_LARGE             0x21
#define SEROM_BOOT_RAW_IMAGE_LOADING_NOT_ALLOWED      0x22
#define SEROM_BOOT_SERAM_JUMP_RETURN_ERROR            0x23
#define SEROM_BOOT_FAILED                             0x24
#define SEROM_BOOT_JUMP_ADDRESS_NOT_VALID             0x25

// Bank selection errors
#define SEROM_BOTH_BANKS_INVALID                      0x26

#define SEROM_ATOC_EXT_HDR_OFFSET_TOO_SMALL           0x27

#define SEROM_BOOT_END_OF_MAIN_ERROR                  0x28
#define SEROM_INVALID_NULL_PTR                        0x29
#define SEROM_INVALID_TOC_OFFSET                      0x30
```

8. CONDUCTOR Tool Flow

The ALIF CONDUCTOR tool is a web-based application that allows a user to configure a Device. Please refer to the CONDUCTOR documentation for more details. This section will focus on the CONDUCTOR material generated for STATIC configuration and how to get this configuration data built using SETOOLS.

CONDUCTOR can be used to generate configurations for

- Static Pin Mux
- Static Clocks
- Static Application space Firewall

The output of the CONDUCTOR tool is a JSON file that can be included into an application as a DEVICE Config object. This object is executed by SES on boot up of the device, so by the time the Application CPUs are booted settings for PinMux, Clock and Firewall are already set.

This method is known as STATIC configuration. As the configuration is stored in Application MRAM space (as part of the ATOC) it will remain there until deleted or overwritten.

Configuration Options

There are three main methods of configuring the device for PinMux, Clock and Firewall:

- STATIC
- DYNAMIC
- Wild-West

STATIC configuration

Configuration data is part of the ATOC (DEVICE Configuration object) and will be installed on each boot of the device before the Application cores are booted.

DYNAMIC configuration

Configuration is done using SERVICE library calls from an application core once booted. This means the initializations are done at run-time or dynamically. These settings can also be changed by subsequent SERVICE library calls.

Wild-West configuration

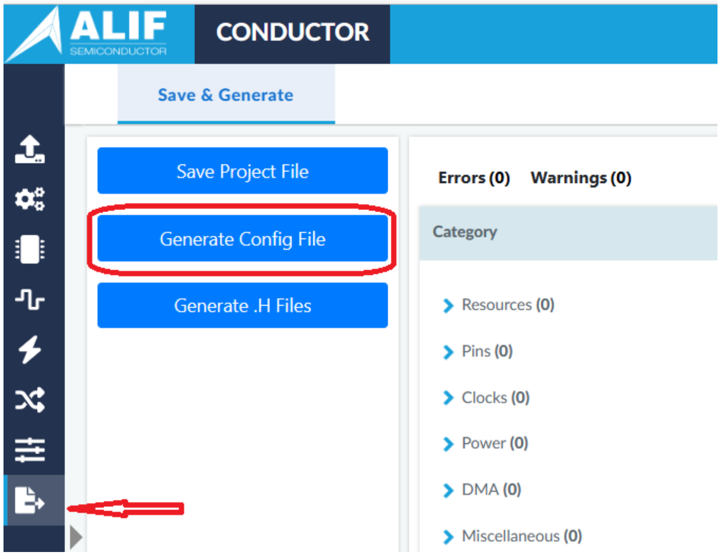
In this mode the application core will be setting the registers directly. There is no protection against multiple CPUs overwriting each other’s settings.

Using the CONDUCTOR Tool with SETOOLS

The Conductor tool generates two types of output – Device Configuration and C header files. SERAM uses the Device Configuration output. It is a JSON file that needs to be copied to the APP release folder (under build\config) and then referenced by the DEVICE entry in the ATOC config file.

Steps –

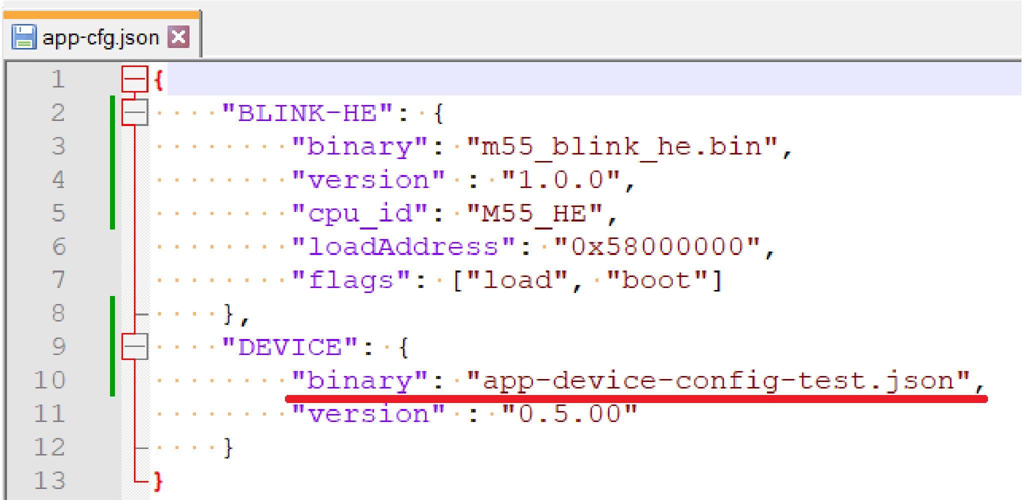
1. Generate a Device Configuration File from the Conductor tool -



The default output file name is something like ‘Config__AE722F80F55D5AS_20230705_0910.json’.

You can rename the file at this point or do that later.

2. Copy the file to the <APP release folder>**\build\config**.
   The Conductor tool saves the generated file in the browser’s default Downloads folder, so copy it from there to your APP release.
3. Optionally, rename the file, e.g., to ‘**app-device-config-test.json**’.
4. Reference the file from the ATOC config file –



```
{
    "BLINK-HE": {
        "binary": "m55_blink_he.bin",
        "version" : "1.0.0",
        "cpu_id": "M55_HE",
        "loadAddress": "0x58000000",
        "flags": ["load", "boot"]
    },
    "DEVICE": {
        "binary": "app-device-config-test.json",
        "version" : "0.5.00"
    }
}
```

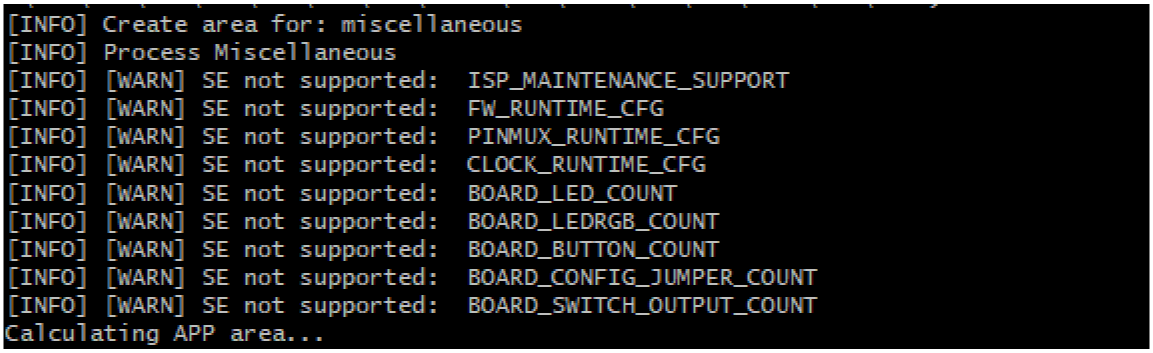
Miscellaneous Feature set supported from CONDUCTOR tool

The MISECELLANEOUS section of the Conductor tool has several configuration options that will be passed into SES for processing at Boot time.

The options that are supported are as follows:

- LFXO_CAP_CTRL
- LFXO_GM_CTRL
- HFXO_CAP_CTRL
- HFXO_PFET_GM_CTRL
- HFXO_NFET_GM_CTRL
- SE_BOOT_INFO

The unsupported options are printed as Warnings when running the app-gen-toc tool



```
[INFO] Create area for: miscellaneous
[INFO] Process Miscellaneous
[INFO] [WARN] SE not supported:  ISP_MAINTENANCE_SUPPORT
[INFO] [WARN] SE not supported:  FW_RUNTIME_CFG
[INFO] [WARN] SE not supported:  PINMUX_RUNTIME_CFG
[INFO] [WARN] SE not supported:  CLOCK_RUNTIME_CFG
[INFO] [WARN] SE not supported:  BOARD_LED_COUNT
[INFO] [WARN] SE not supported:  BOARD_LEDRGB_COUNT
[INFO] [WARN] SE not supported:  BOARD_BUTTON_COUNT
[INFO] [WARN] SE not supported:  BOARD_CONFIG_JUMPER_COUNT
[INFO] [WARN] SE not supported:  BOARD_SWITCH_OUTPUT_COUNT
Calculating APP area...
```

You must use the '-v' (Verbose) option to see these Warnings.

SE_BOOT_INFO

The default for this is '0' which means all SES Prints are enabled on boot up.

Value	Output
0	All prints enabled
1	
2	Suppress all prints

9. Device Debug Strategies

The following section describes some debug strategies that can be applied to a device that appears to be not working.

Recovery vs SES Mode

In normal operational mode SES will be running on the device. IN rare cases this may not be the case, and your Device will be in Recovery mode. This means that SEROM failed to find a valid image to boot.

In recovery mode you will use the Maintenance tool to put back (Recover) an SES image into the device.

In recovery mode you are communicating with SEROM only. The ISP command set for SEROM is limited compared to when talking to SES. You may see ISP_UNKNOWN_GET_OBJECT_ERROR, this means that the object being requested is not there. An example would be to try to get the TOC data from SEROM.

HARD Maintenance mode

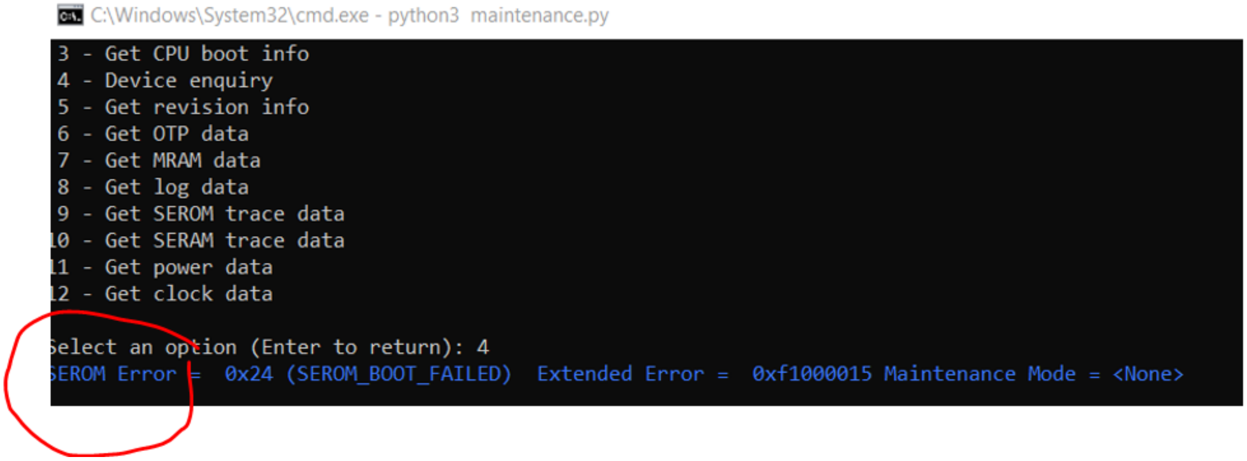
This mode allows the user to override the SE Boot sequence and get access to the Secure Enclave.

Quick Fix guide

Anyone who is troubleshooting a bricked or unresponsive board should do these basic first steps –

- Check if the board is in SEROM or SERAM mode,

If you are not sure what mode you are in, use SETOOLS



```
3 - Get CPU boot info
4 - Device enquiry
5 - Get revision info
6 - Get OTP data
7 - Get MRAM data
8 - Get log data
9 - Get SEROM trace data
10 - Get SERAM trace data
11 - Get power data
12 - Get clock data

Select an option (Enter to return): 4
SEROM Error =  0x24 (SEROM_BOOT_FAILED)  Extended Error =  0xf1000015 Maintenance Mode = <None>
```

- SEROM mode – run the SEROM recovery command.

NOTE: Hard Maintenance mode does not make sense in SEROM mode and is not supported.
Recommendation – run the recovery using the option '**2. Recovery (No Reset)**'. When the recovery is complete, open a terminal, then reset the board and monitor the output on the SE UART.
If, after Booting, you are still in recovery mode look for extended Error code in Recovery screen and dump SEROM Trace log

- SERAM mode – enter Hard Maintenance mode and then Erase the Application MRAM.

Maintenance -> Option#3 -> Option#3 (or Option#1)

Reporting Issues to ALIF

When reporting issues to ALIF always provide the following information

- Version of SETOOLS used.
- Version of Device used
- Maintenance -> Option#2 Device information -> Option#5 Get Revision Information
- SE-UART output on boot up
    - This is the most useful for triaging issues
- SEROM, SERAM TRACEs (Maintenance -> Option#2 Device Information -> Option#9 and Option#10

10. Document History

Version	Date	Change Log
0.84.0	October 2023	Web release for Security Toolkit v0.84.0 tools
0.85.0	November 2023	Initial support for MAC OS in SETOOLS
0.87.0	December 2023	Adding assets-gen and app-provision tool details
1.104.0	February 2025	New Maintenance menus when in Recovery mode