PSA NuBL2 User Guide

Application Notefor 32-bit NuMicro®Family

Document Information

Abstract	NuBL2 is a secure boot loader to provide trusted boot for Arm PSA certification level2.
Apply to	NuMicro®Cortex®- M2351 series

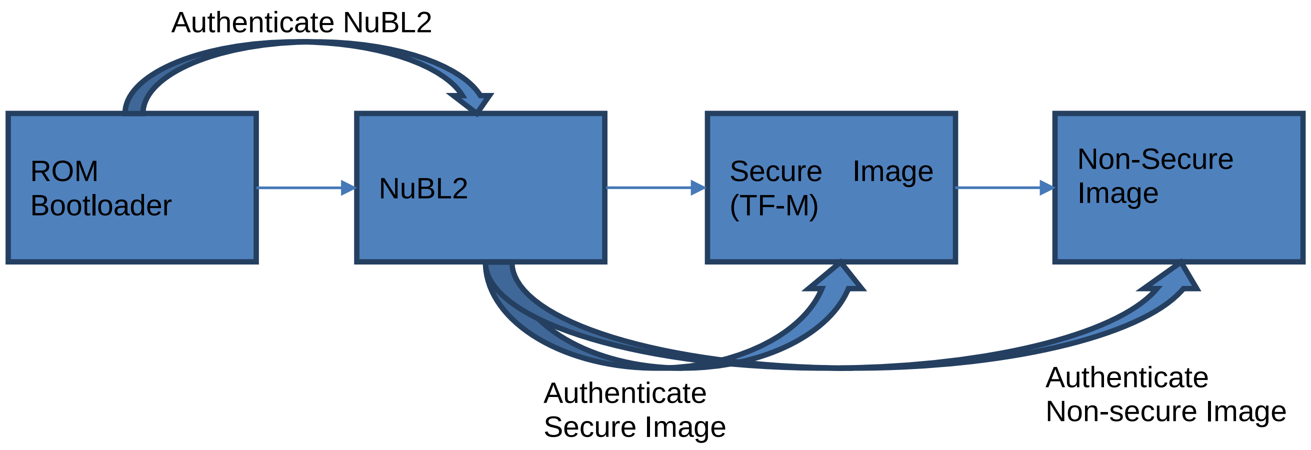
Table of Contents

1 Overview

M2351 support immutable ROM bootloader to provide the Root-of-Trust. To compliant the requirement of PSA level2 certification, NuBL2 is also implemented for authenticating and booting TF-M and Non-secure code.

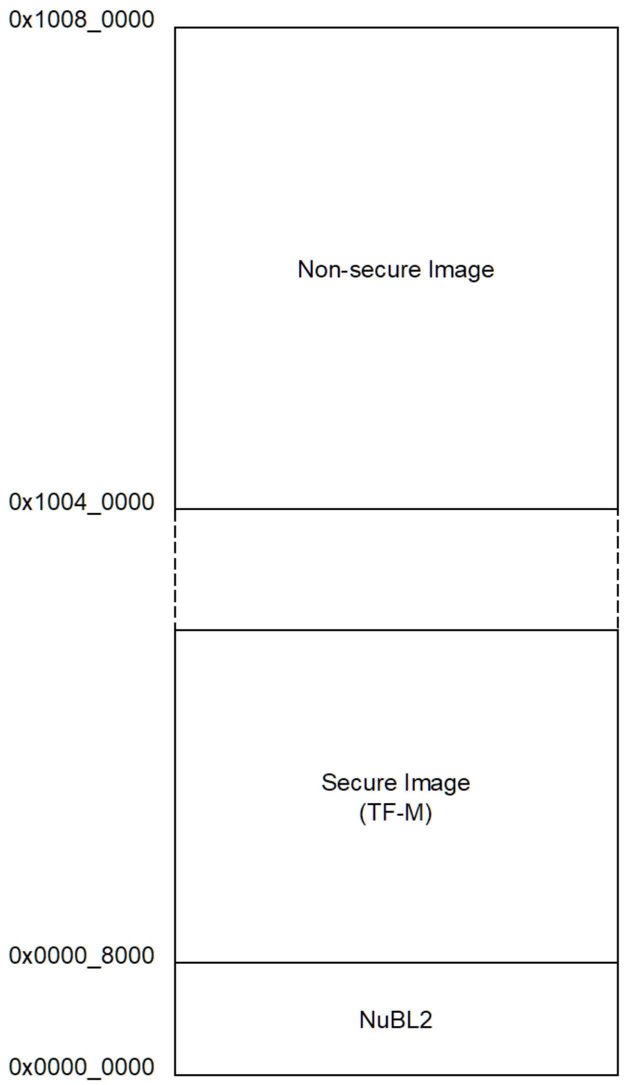
The source code of NuBL2 could be found at https://github.com/wschang0/NuBL2.

When booting, the ROM bootloader is used to authenticate NuBL2 and boot to NuBL2, then NuBL2 will authenticate TF-M and Non-secure code. Finally, NuBL2 will boot to TF-M if both secure image and non-secure image are validated ok.



2 Memory Mapping of the System

The reserved space size for NuBL2 is 32 KB. Secure image space is from 0x8000 to 0x3FFFF and Non-secure image space is from 0x10040000 to 0x1007FFFF.



Address Mapping

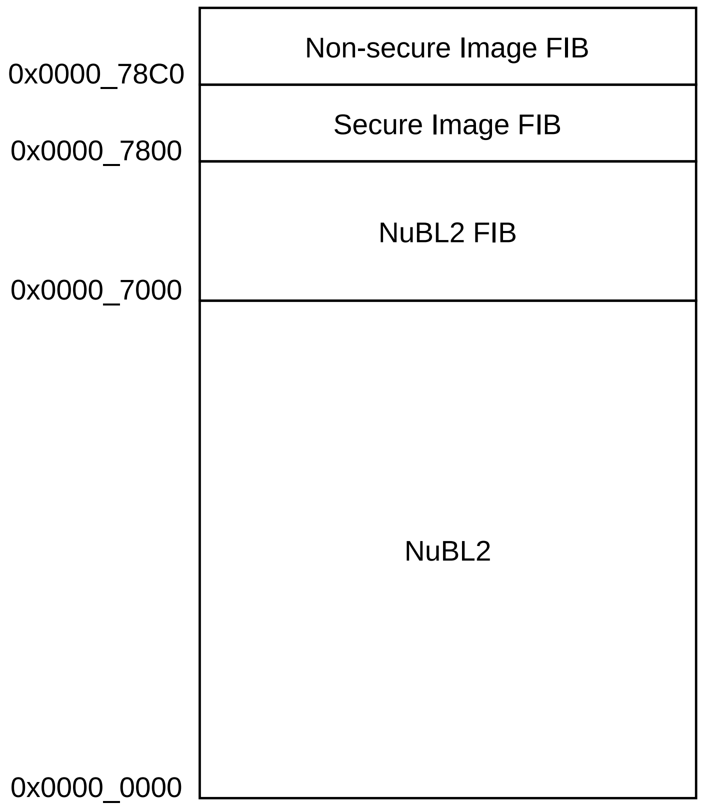
3 PSA NuBL2

The major function of NuBL2 is to authenticate secure and non-secure image then booting to secure image. It also supports to update secure and non-secure image

The NuBL2 will be authenticated by MKROM, thus it also needs to prepare the signature of NuBL2.

3.1 Memory Mapping of NuBL2

NuBL2 is the second stage of M2351 trusted boot. It will be authenticated by M2351 ROM code and responsible to authenticate next boot image. The firmware information block (FIB) is the signature file for authentication purpose. Therefore, all boot images have its own firmware information block. The figure in bellow shows the memory mapping of NuBL2 including the FIBs.



NuBL2 Memory Mapping

3.2 Private Key for Signing NuBL2

The signature file of NuBL2 is signed by ECDSA algorithm with 256 bits. This could be done when building NuBL2. The private key is required for this procedure and it must be input to file FwSign.ini which located in the same folder of NuBL2 project file.

The contents of the FwInfo.ini is

```
[KEY]
Private Key=380a67fcfc01ca7073da7c2c54296a61327f77262a7d4674c3d8e29a63e3fa20
Public Key 1=755B3819F05A3E9F32D4D599062834AAC5220F75955378414A8F63716A152CE2
Public Key 2=91C413F1915ED7B47473FD797647BA3D83E8224377909AF5B30C530EAAD79FD7

[ECDSA]
R=10FB61AE3C753D218A4678EAA916E81D4C24FED9907E246F5019F9C338E18316
S=8E52D7702615A98FD3DFAE8C5DBDAE131550B5C6685E4055B1A971C9B37105F2

[HASH]
Firmware=FF297309AF05F78F3D741458B46D1AD045957C458AB3D3330DF37C6E9C50D5DB
Info=869C868BDCDD6CEB1E0B0281E66FD8D82A4B69E26AC3E62813DE0668C965E1EE
```

To change private key, user needs to modify the value of “Private Key” and leave “Public Key 1” and “Public Key 2” blank. If the public key is not blank, it won’t be re-calculated according to new private key. It would look like this:

New Private Key (256 bits)

```
[KEY]
Private Key=14B1AC49A1D1ED79D8AB069A18D23254916C6C599082F1FD640D10468C7BDDCB
Public Key 1=
Public Key 2=

[ECDSA]
R=10FB61AE3C753D218A4678EAA916E81D4C24FED9907E246F5019F9C338E18316
S=8E52D7702615A98FD3DFAE8C5DBDAE131550B5C6685E4055B1A971C9B37105F2

[HASH]
Firmware=FF297309AF05F78F3D741458B46D1AD045957C458AB3D3330DF37C6E9C50D5DB
Info=869C868BDCDD6CEB1E0B0281E66FD8D82A4B69E26AC3E62813DE0668C965E1EE
```

After re-building NuBL2, the FwInfo.ini will be updated automatically as:

```
[KEY]
Private Key=14B1AC49A1D1ED79D8AB069A18D23254916C6C599082F1FD640D10468C7BDDCB
Public Key 1=06DF446C65C6951E5ADD673ED19211DB2F0FB689B61D79E1E1A709E8E36AAB1B
Public Key 2=697C248D540F56FD2F2739615E0007B48DB75BBCB0C1442647265AF6EAAA95B7

[ECDSA]
R=D84DFFD09060D76E3CAEA64A8C6CCB8918C7FCC85048E15FA6E00F993A73FC94
S=CB7304CB1206759CDF8210D36230B08C61607B30FBB0D914FB462ECCF85429E6

[HASH]
Firmware=FF297309AF05F78F3D741458B46D1AD045957C458AB3D3330DF37C6E9C50D5DB
Info=C70BA2248D87F19F47625ED96E873BEEB894C781165646783BF5CF11BA9A57F9
```

The private key could be generated by Nuvoton Crypto Tool or any other way. However, it should always be kept secret. Nuvoton Crypto Tool could be download form

http://www.nuvoton.com/opencms/system/modules/com.thesys.opencms.nuvoton/pages/download/download.jsp?file=http://www.nuvoton.com/hq/resource-download.jsp?tp_GUID=SW1020180907173742&version=1.1.3

3.3 Build NuBL2 and Create the Signature of NuBL2

NuBL2 only support Keil MDK build environment. To get the Keil MDK, please refer to the here http://www2.keil.com/nuvoton.

To build NuBL2, it needs to open the project file NuBL2.uvprojx at NuBL2\SampleCode\BootLoader\SecureBoot\NuBL2\Keil.

After open the project file with Keil MDK, just press key “F7” to build the project. Then press key “F8” to download NuBL2 to M2351.

When building NuBL2, the signature file (FIB) will be generated automatically. The NuBL2 FIB file is located at Keil\NuBL2 and the file name is FwInfo.bin or FwInfo.hex in hex file formate.

When download NuBL2 to M2351 by press “F8” in Keil MDK, the FwInfo will also be download to 0x7000.

3.4 Enable Root-of-Trust

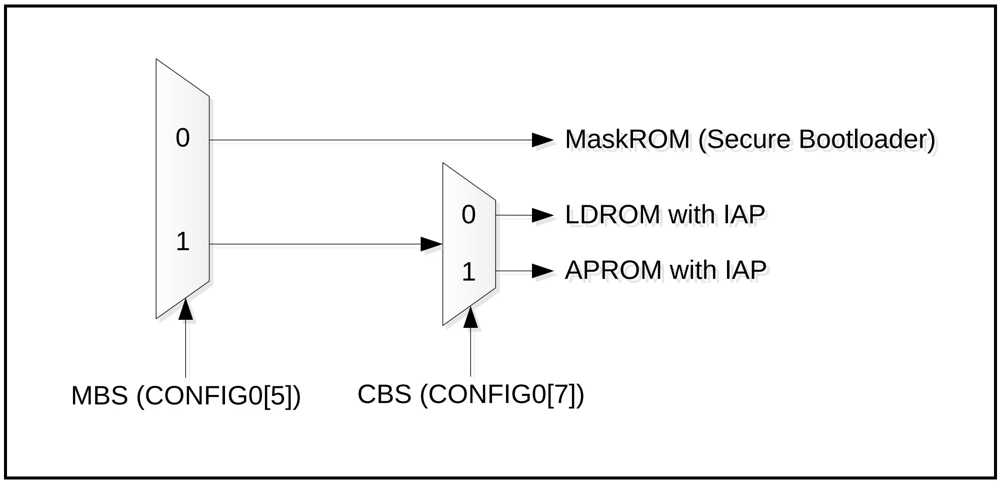
After download NuBL2 and execute it, the message on the debug console would look like this:

```
=== Nuvoton PSA Bootloader v1.0 ===
Root of Trusted Function ........ Disabled!
```

The message shows RoT of M2351 is not enabled yet. To enable RoT of M2351, we need to set user configuration to enable boot loader and write ROTPK Hash to OTP.

3.4.1 Select Boot Source

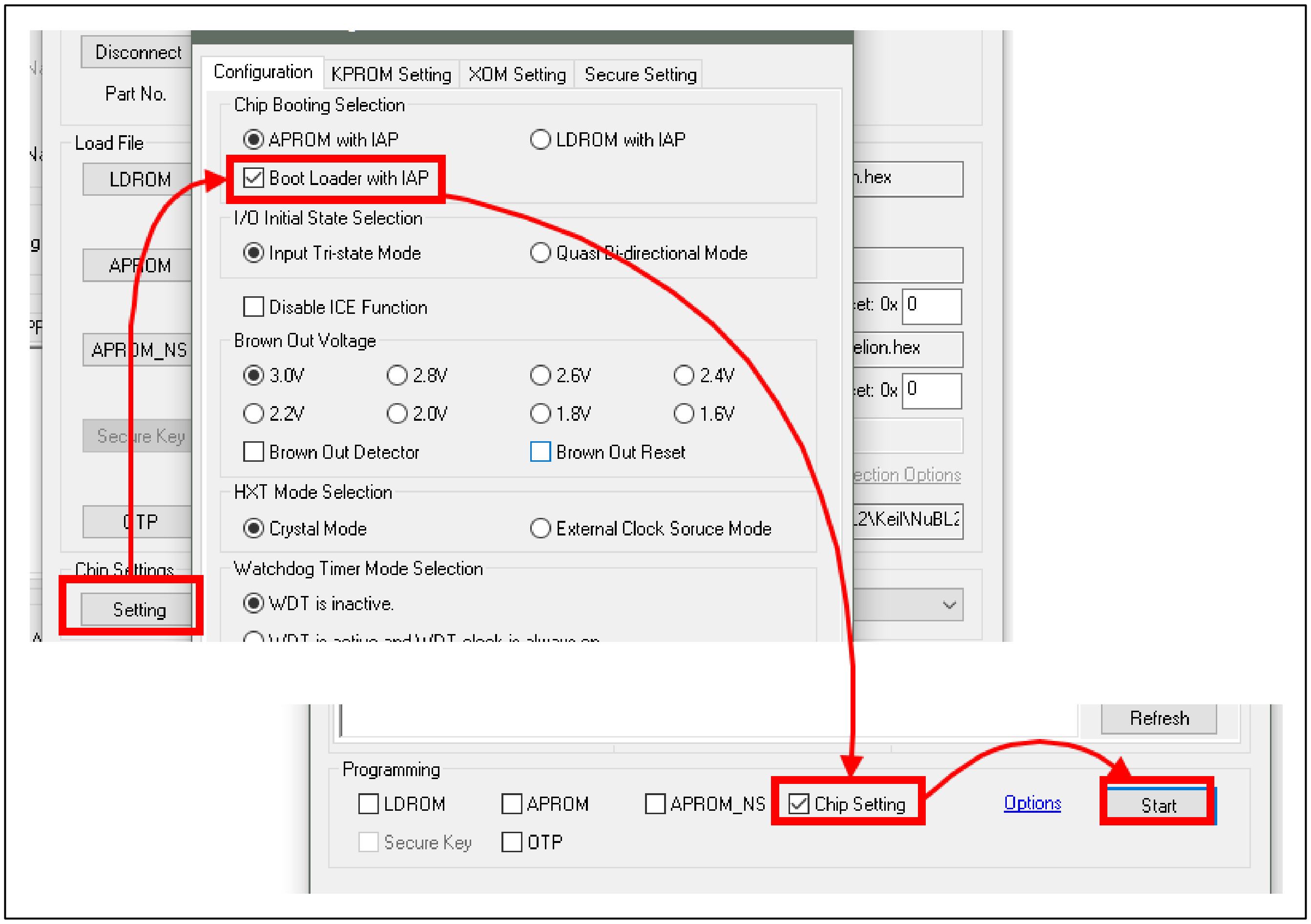
In M2351, the target booting source after CHIP powered on or reset can be selected by setting CBS (CONFIG0[7]) and MBS (CONFIG0[5]) as the below figure shown.



If MBS has been configured as ‘0’, the system will boot from MKROM (immutable ROM code). Otherwise, the system will execute the executable code by CBS ‘0’ from LDROM and CBS ‘1’ from APROM.

According the requirement of PSA certification, MBS (CONFIG0[5]) must be ‘0’ to boot from MKROM.

The easiest way to set to boot from MKROM is using ICP programming tool and enable “boot loader with IAP”. The setting is shown as below figure:



Chip Booting Selection
[✓ APROM with IAP] LDROM with IAP
[✓ Boot Loader with IAP]

I/O Initial State Selection
[✓ Input Tri-state Mode] Quasi Bi-directional Mode

Disable ICE Function

Brown Out Voltage
[✓ 3.0V] 2.8V 2.6V 2.4V
2.2V 2.0V 1.8V 1.6V
Brown Out Detector
Brown Out Reset

HXT Mode Selection
[✓ Crystal Mode] External Clock Soruce Mode

Watchdog Timer Mode Selection
[✓ WDT is inactive.]

Programming
LDROM APROM APROM_NS [✓ Chip Setting]
Secure Key OTP

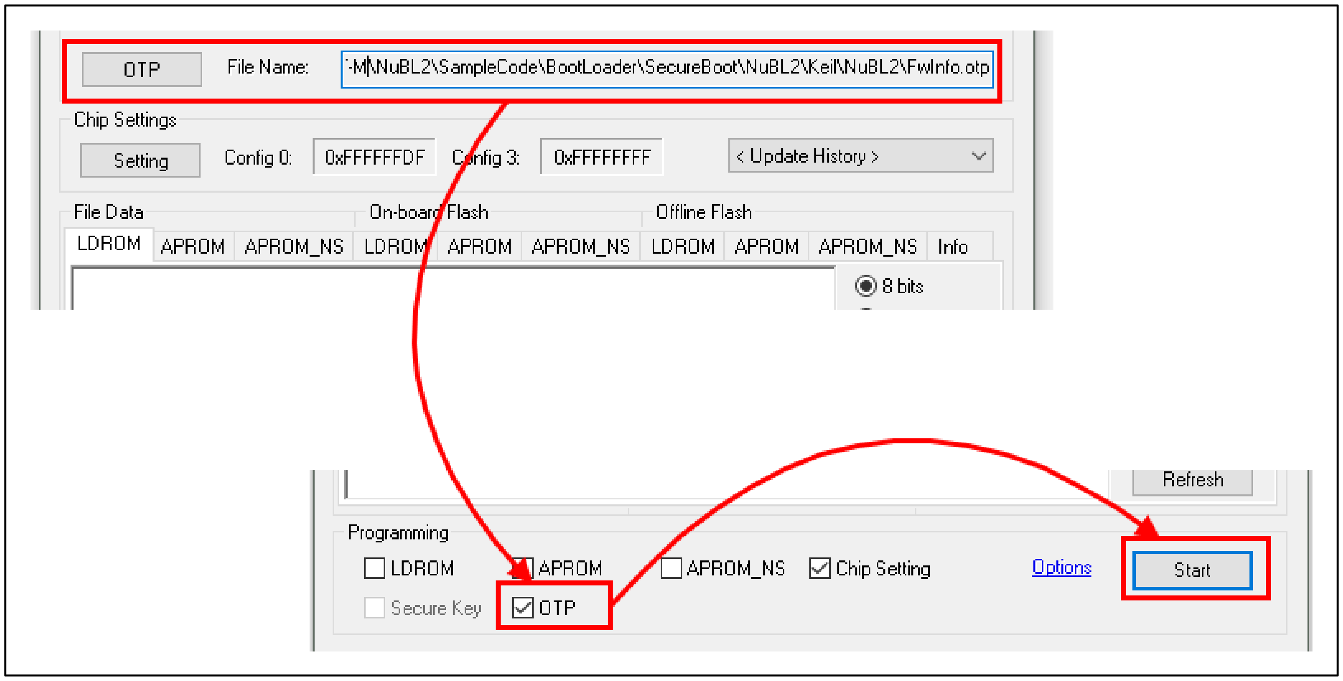
3.4.2 Write ROTPK Hash

M2351 supports one-time programming memory (OTP). To prevent the ROTPK to be changed, in other words, to make sure ROTPK is immutable, the ROTPK hash must be written to OTP. By checking the ROTPK hash, it can prove the ROTPK is immutable.

The ROTPK of NuBL2 could be got from FwInfo.ini file. They are “Public Key 1” and “Public Key 2”. Both of them are 256 bits. For ECDSA, they are just one ROTPK.

```
[KEY]
Private Key=380a67fcfc01ca7073da7c2c54296a61327f77262a7d4674c3d8e29a63e3fa20
Public Key 1=755B3819F05A3E9F32D4D599062834AAC5220F75955378414A8F63716A152CE2
Public Key 2=91C413F1915ED7B47473FD797647BA3D83E8224377909AF5B30C530EAAD79FD7
```

The ROTPK hash will be generated while NuBL2 building. The hash file is stored as FwInfo.otp which located at NuBL2 folder under NuBL2 project file folder. FwInfo.otp file could be load by ICP programming tool to program OTP into M2351. Please note the



The OTP Programming tool could be download from:

http://www.nuvoton.com/hq/resource-download.jsp?tp_GUID=SW0520101208200310&__locale=en

After setting booting from MKROM and write OTP, the message will become as bellow:

```
=== Nuvoton PSA Bootloader v1.0 ===
Root of Trusted Function ........ Enabled!
```

3.5 Verify Next Image

If the NuBL2 can pass the certificationof MKROM, MKROM will start to execute NuBL2. Then, NuBL2 start to certificate next Image including secure image and non-secure image. Therefore, we need to prepare firmware information blocks (FIBs) of secure image and non-secure image to provide certification information of the images.

Furthermore, both secure FIB and non-secure FIB will be included in NuBL2 image. Therefore, it will be more easy to access the FIBs in NuBL2 to certificate the images. The FIBs are included by NuBL3xFwInfo.s which is built with NuBL2 and can be download when downloading NuBL2.

3.5.1 Prepare FIB for Secure Image

Because the secure image here is TF-M, and the TF-M image includes code region and non-secure callable region, we need to split them before we can sign the whole image.

To split TF-M image into code and non-secure callable code, user can use tfm_split.exe utility.

For example:

```
tfm_split .\build\app\secure_fw\tfm_s.bin .\build\app\secure_fw\tfm_s.map
```

Where tfm_s.bin is the TF-M image built from trusted-firmware-m project and tfm_s.map is the mapping file of tfm_s.bin which is generated by Arm Compiler.

tfm_split utility will get memory mapping information to split tfm_s.bin into tfm_s_lr.bin (code region) and tfm_s_nsc.bin (non-secure callable region).

By tfm_s_lr.bin and tfm_s_nsc.bin, user can use FwSign to sign them by below command:

```
FwSign tfm_s_lr.bin+tfm_s_nsc.bin tfm_s_info.bin 0x7800 0x2c800
```

Where tfm_s_info.bin is the prepared FIB file. After execute the command, the tfm_s_info.bin will be updated according to tfm_s_lr.bin and tfm_s_nsc.bin. 0x7800 indicates the location of tfm_s_info.bin. 0x2c800 is used to limit the tfm_s_lr.bin hash calculation range. It means the hash calculation range of tfm_s_lr.bin is from 0x0 to 0x2c800. The data out of this range may be change during TF-M executing due to it is used for secure storage and other storage service in TF-M.

The final tfm_s_info.bin needs to be copy to the folder of NuBL2 project, so the NuBL2 could include it when building.

3.5.2 Prepare FIB for Non-secure Image

Because non-secure image doesn't include non-secure callable, we can just use FwSign to sign it.

For example:

```
FwSign tfm_ns.bin tfm_ns_info.bin 0x78C0
```

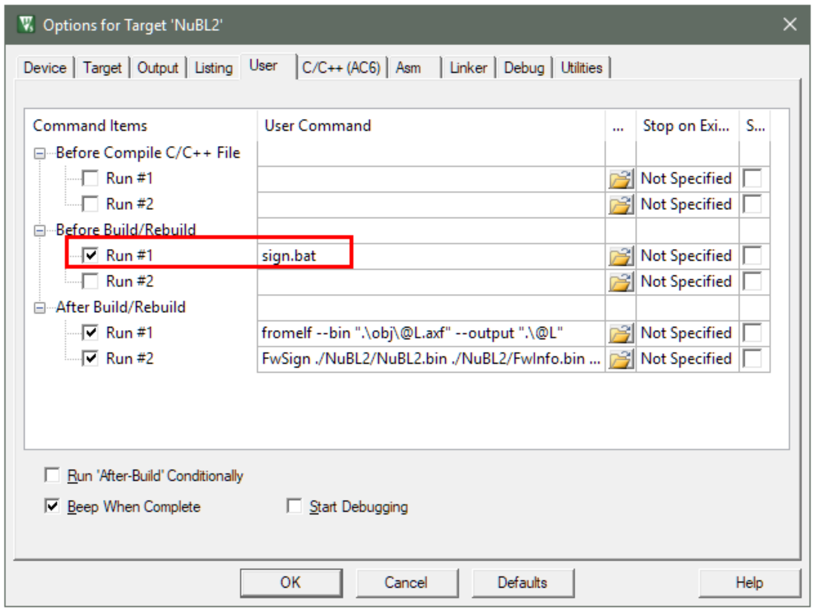
Where tfm_ns.bin is the non-secure image and tfm_ns_info.bin is the prepared FIB of non-secure image.

After executing this command, the tfm_ns_info.bin will be updated to be the signature of tfm_ns.bin.

Finally, user needs to copy the tfm_ns_info.bin to the folder of NuBL2 project, so the NuBL2 could include it when building.

3.5.3 Download secure and non-secure image

Before downloading secure and non-secure image, the images need to be signed. This could be done by DOS batch file and setting this file to Keil to run it before building.


Options for Target 'NuBL2'

Device | Target | Output | Listing | User | C/C++ (AC6) | Asm | Linker | Debug | Utilities

Command Items	User Command	...	Stop on Exi...	S...
Before Compile C/C++ File				
☐ Run #1			Not Specified	☐
☐ Run #2			Not Specified	☐
Before Build/Rebuild				
☑ Run #1	sign.bat		Not Specified	☐
☐ Run #2			Not Specified	☐
After Build/Rebuild				
☑ Run #1	fromelf --bin ".\obj\@L.axf" --output ".\@L"		Not Specified	☐
☑ Run #2	FwSign ./NuBL2/NuBL2.bin ./NuBL2/FwInfo.bin ...		Not Specified	☐

☐ Run 'After-Build' Conditionally

☑ Beep When Complete ☐ Start Debugging

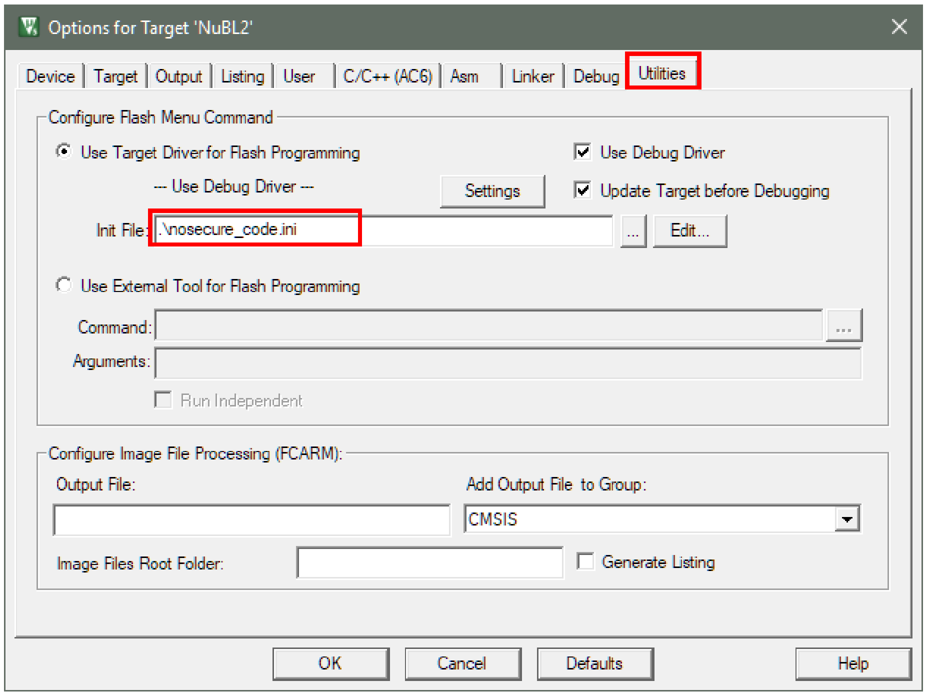
The content of sign.bat looks like below:

```
tfm_split  d:\MCU\TF-M\trusted-firmware-m\build\app\secure_fw\tfm_s.bin  d:\MCU\TF-M\trusted-
firmware-m\build\app\secure_fw\tfm_s.map

FwSigntfm_s_lr.bin+tfm_s_nsc.bintfm_s_info.bin 0x7800 0x28c00

FwSign d:\MCU\TF-M\trusted-firmware-m\build\app\tfm_ns.bin tfm_ns_info.bin 0x78C0
```

The secure and non-secure code could be download by setting nonsecure_code.ini in project option.


Options for Target 'NuBL2'
Device | Target | Output | Listing | User | C/C++ (AC6) | Asm | Linker | Debug | Utilities
Configure Flash Menu Command
Use Target Driver for Flash Programming
--- Use Debug Driver ---
Settings
Use Debug Driver
Update Target before Debugging
Init File: .\nosecure_code.ini
Edit...
Use External Tool for Flash Programming
Command:
Arguments:
Run Independent
Configure Image File Processing (FCARM):
Output File:
Add Output File to Group:
CMSIS
Image Files Root Folder:
Generate Listing
OK
Cancel
Defaults
Help

The nonsecure_code.ini content is as below:

```
load ".\\NuBL2\\FwInfo.hex"
load "d:\\MCU\\TF-M\\trusted-firmware-m\\build\\app\\secure_fw\\tfm_s.axf" INCREMENTAL
load "d:\\MCU\\TF-M\\trusted-firmware-m\\build\\app\\tfm_ns.axf" INCREMENTAL
```

By this ini file, Keil will download FwInfo.hex, tfm_s.axf and tfm_ns.axf before downloading NuBL2.

4 Execute PSA NuBL2

The following steps are for the trusted boot of M2351

Step1. Link NuMaker-PFM-M2351 through ICE USB port.

Step2. Open NuBL2 project

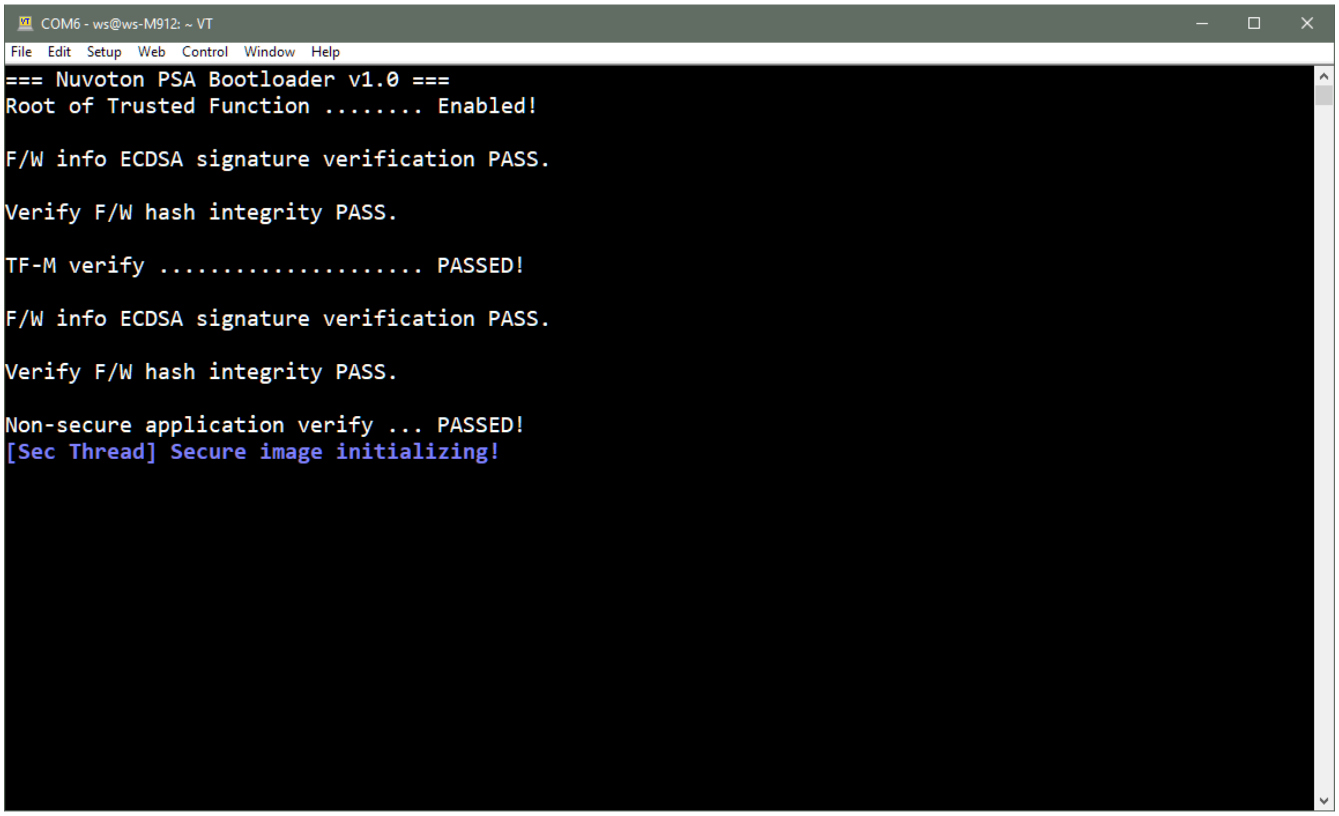
Step3. Built TF-M to generated tfm_s.axf and tfm_ns.axf

Step4. Press ‘F7’ to build the NuBL2

Step5. Press ‘F8’ to download secure, non-secure image and NuBL2

Step6. Press ‘RESET” key on board to run the code.

The screen shot of PSA NuBL2 executing is as below:



```
=== Nuvoton PSA Bootloader v1.0 ===
Root of Trusted Function ........ Enabled!

F/W info ECDSA signature verification PASS.

Verify F/W hash integrity PASS.

TF-M verify ..................... PASSED!

F/W info ECDSA signature verification PASS.

Verify F/W hash integrity PASS.

Non-secure application verify ... PASSED!
[Sec Thread] Secure image initializing!
```

5 Firmware Update

NuBL2 supports firmware update by micro SD card. User needs to prepare new firmware and relative firmware information file. The filename is fixed as

- tfm_s_c.bin (Under NuBL2\Keil)
- tfm_s_i.bin (Under NuBL2\Keil)
- tfm_ns.bin (Under trusted-frimware-m\build\app)
- tfm_ns_i.bin (Unser NuBL2\Keil)

where tfm_s_c.bin is the secure image of secure firmware which is generated when building NuBL2.

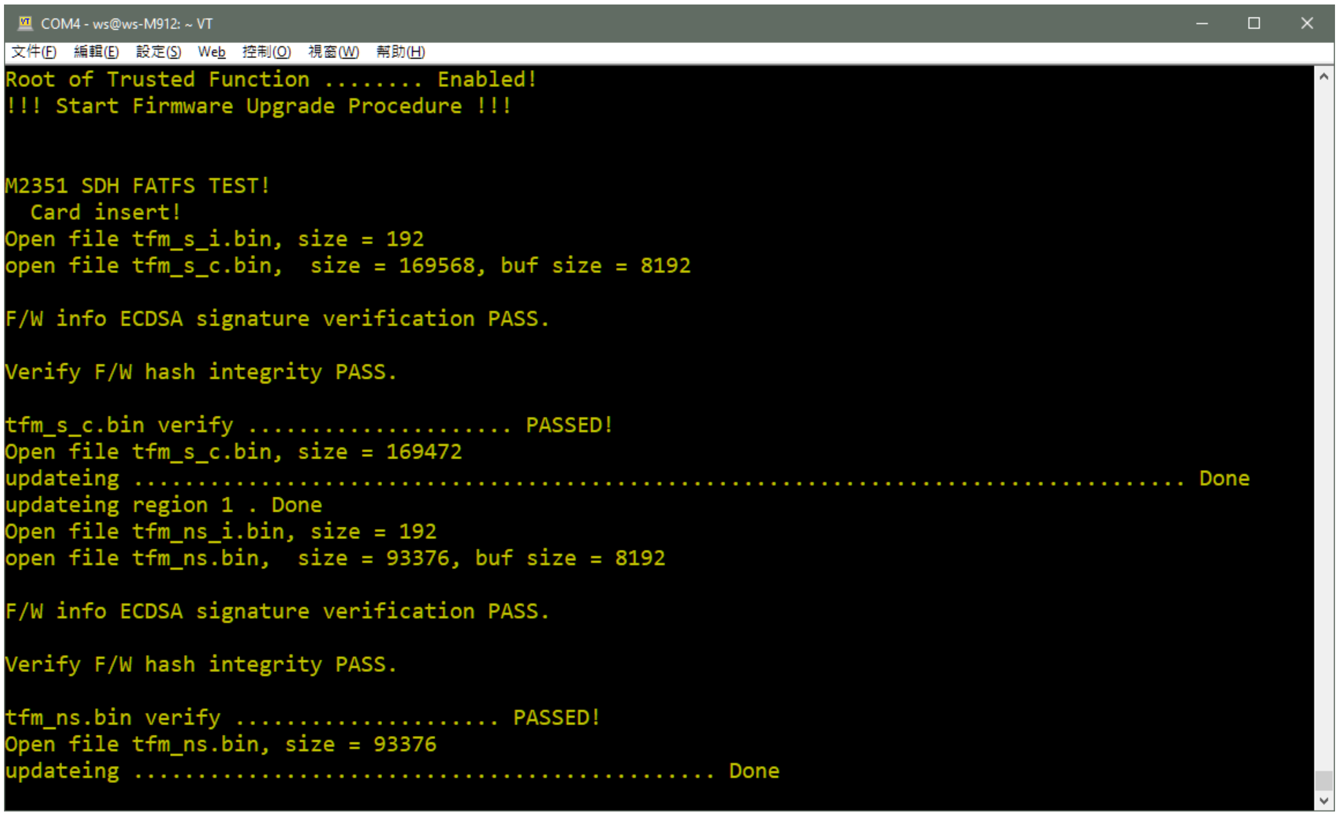
The tfm_ns.bin is non-secure image which is generated by trusted-firmware-m

tfm_s_i.bin and tfm_ns_i.bin are the firmware information file of secure and non-secure image.

User needs to copy the files to the root directory of micro SD card and insert the card to NuMaker-PFM-M2351 board.

To update the firmware, user need to keep pressing key “SW2” while reset the system.

Then NuBL2 will start to verify the images on micro SD card and start update process.



```
Root of Trusted Function ........ Enabled!
!!! Start Firmware Upgrade Procedure !!!


M2351 SDH FATFS TEST!
  Card insert!
Open file tfm_s_i.bin, size = 192
open file tfm_s_c.bin,  size = 169568, buf size = 8192

F/W info ECDSA signature verification PASS.

Verify F/W hash integrity PASS.

tfm_s_c.bin verify ..................... PASSED!
Open file tfm_s_c.bin, size = 169472
updateing ................................................................................ Done
updateing region 1 . Done
Open file tfm_ns_i.bin, size = 192
open file tfm_ns.bin,  size = 93376, buf size = 8192

F/W info ECDSA signature verification PASS.

Verify F/W hash integrity PASS.

tfm_ns.bin verify ..................... PASSED!
Open file tfm_ns.bin, size = 93376
updateing ............................................. Done
```

User needs to reset system to reboot after updating firmware.