Braswell Platform – Intel® Trusted Execution Engine (Intel® TXE) 2.0 SKU Firmware for Windows*

Bring-Up Guide

April 2015

Revision 1.3

Intel Confidential

Contents

Figures

Tables

Revision History

Revision Number	Description	Revision Date
0.5	Initial release.	July 2014
0.8	• Updates: — Added instructions on enable QE bit in strap 1 chapter 14 — Removed TOS & STFC from strap 1 in FITc Soft Straps (Chapter 14) — Updated Figures 1-16	October 2014
1.0	• Updates: — Flamingo Chapter: updated Flamingo usage & new FPF config file view — Updated SOC Strap 2 — Updated SOC Strap 2 — Removed SOC Strap 5 — Updated SOC Strap 6	December 2014
1.1	• Updates: — Updated supported operating systems (removed Android support). — Removed Widevine support. — Added LPCCLK_FREQUENCY control to Soft Strap 4 (bit 19) — Added LPCCLK1_EN control to Soft Strap 5 (bit 19) — Updated SoC brand to “N-Series” only (removed J-Series)	February 2015
1.2	• Updates — Added Step 5 in section 3.4 (add SPI part VSCC table entry in image)	March 2015
1.3	• Updates — Updated Soft Strap configurations	April 2015

§

1 Introduction

This document covers the Intel® Pentium® processor and Intel® Celeron® processor N-series based platform (Braswell platform) firmware bring-up procedure for the Intel® quad-core technology SoC. The processor numbers will start with the prefix 'N' in continuation with the Bay Trail-M SKUs. The prefix 'J' will not be used.

The bring-up procedure primarily involves building a FW image. Once the FW image is built, it can be programmed to the Braswell platform. All the paths mentioned in this guide are relative path to the root of the given kit.

1.1 Terminology

Term	Description
FITc	Flash Image Tool creation
FPF	Filed Programmable Fuse
FPT	Flash Programming Tool
Intel® TXE	Intel® Trusted Execution Engine (Intel® TXE)
Intel® TXEI	Intel® Trusted Execution Engine Interface (Intel® TXEI)

§

2 Quick Start Check List

2.1 First Boot of Braswell

Before running the first basic boot of the Braswell platform, follow the below instructions:

- Build the SPI FW image
    - Build the image using FITc tool as described in Section 3.4 and flash the image components from the FW kit.
- Flash the SPI FW image
    - Flash the image using Dediprog* or FPT method as described in Section 3.6, Flashing Target.
- If using Windows OS: Install Intel® Trusted Execution Engine Interface (Intel® TXEI) Driver as described in Chapter 7
    - Once the platform boots, install the Intel TXEI driver found in the FW Kit
- Verify Intel® Trusted Execution Engine (Intel® TXE) information
    - Run TXEInfo tool found in the FW kit "\System Tools\ TXEInfo\" directory
- Verify Intel TXE status using the TXEManuf tool
    - Run TXEManuf tool found in the FW kit "\System Tools\ TXEManuf\" directory
    - Use TXEManuf.cfg to enable/disable tests of interest

For more details on each of these steps, refer to the appropriate chapter within the Intel TXE FW Bring-Up Guide (this guide).

§

3 Procedure

3.1 Prerequisites

- fitc.exe: can be found under \\System tools\Flash_Image_Tool folder
- DediProg* SF100
- FPT can be found under \\System tools\Flash_Programming_Tool

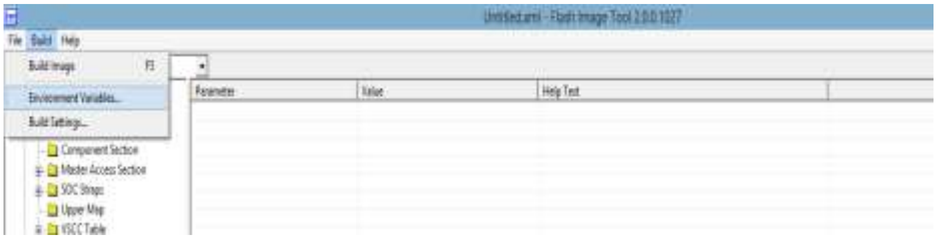
3.2 Start FITc

- Invoke Flash Image Tool by navigating to \\System tools\Flash_Image_Tool folder
- Double click fitc.exe.

3.3 Set Up Build Environment

In the main menu select Build→ Environment Variables.

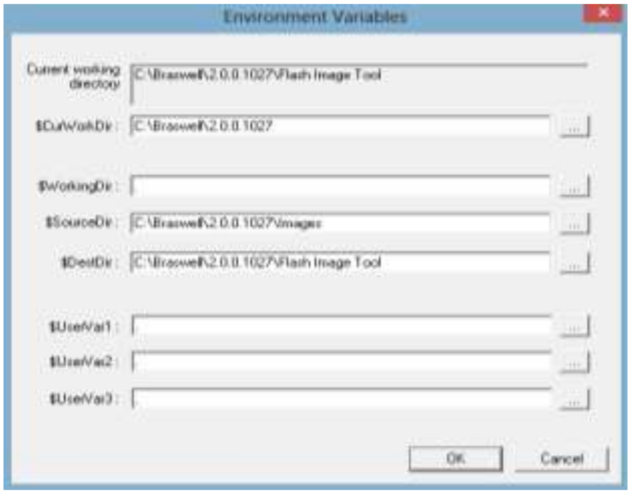
Figure 1. Enviroment Variables



Edit your configuration as shown below.

- **$Source Dir:** The location where FITc will look for binary images during the image creation process
- **$DestDir:** The location where FITc will save the binary image
- **$WorkingDir:** The location where fitc.exe is running. Keep it as "."

Figure 2. Environment Variables



Current working directory: C:\Braswell\2.0.0.1027\Flash Image Tool

$CurWorkDir : C:\Braswell\2.0.0.1027

$WorkingDir :

$SourceDir : C:\Braswell\2.0.0.1027\Images

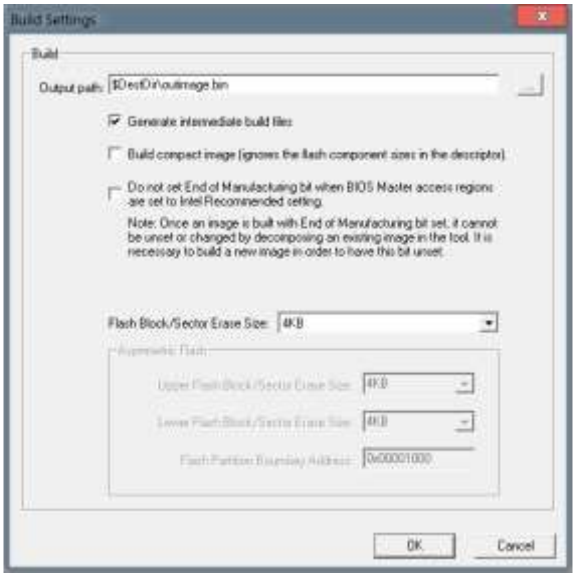
$DestDir : C:\Braswell\2.0.0.1027\Flash Image Tool

$UserVar1 :

$UserVar2 :

$UserVar3 :

Figure 3. FIT Set Up



Output path: $DestDir\outimage.bin

☑ Generate intermediate build files

☐ Build compact image (ignores the flash component sizes in the descriptor)

☐ Do not set End of Manufacturing bit when BIOS Master access regions are set to Intel Recommended setting.

Note: Once an image is built with End of Manufacturing bit set, it cannot be unset or changed by decomposing an existing image in the tool. It is necessary to build a new image in order to have this bit unset.

Flash Block/Sector Erase Size: 4KB

Upper Flash Block/Sector Erase Size: 4KB

Lower Flash Block/Sector Erase Size: 4KB

Flash Partition Boundary Address: 0x00001000

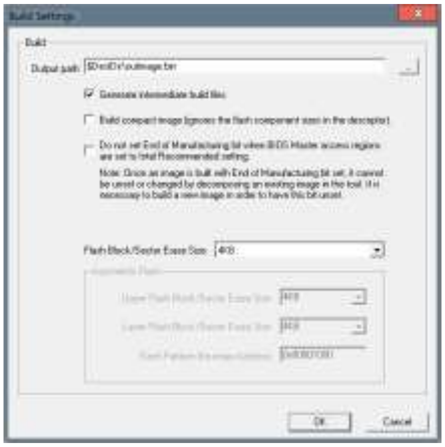
Note: Use the environment variables when defining output path in Build → Build Settings, as shown above.

3.4 Create Flash Image

3.4.1 Using GUI

1. Run FITc.exe.
2. Define output image name and path.
3. Build → Build Settings → Output path.

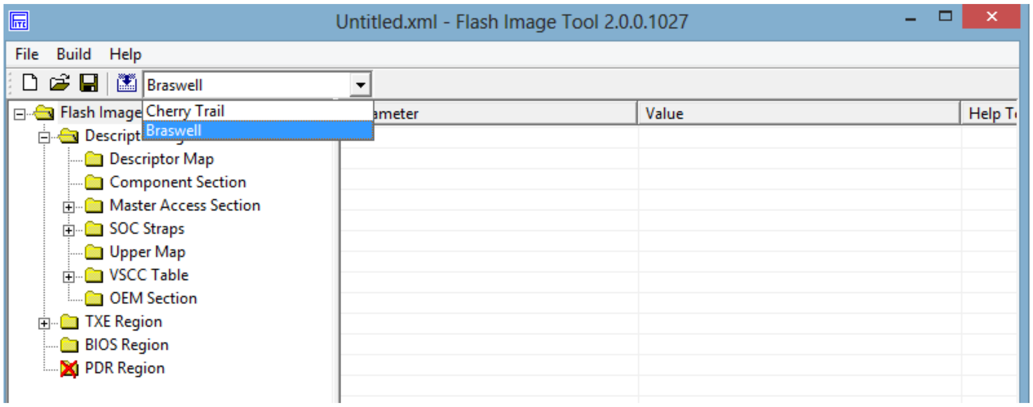
Figure 4. Define Output Path


Build Settings
Build
Output path
Generate intermediate build files
Flash Block/Sector Erase Size
4KB
OK
Cancel

4. Select platform type
   a. On the platform selection list, select Braswell before modifying and building the flash image.

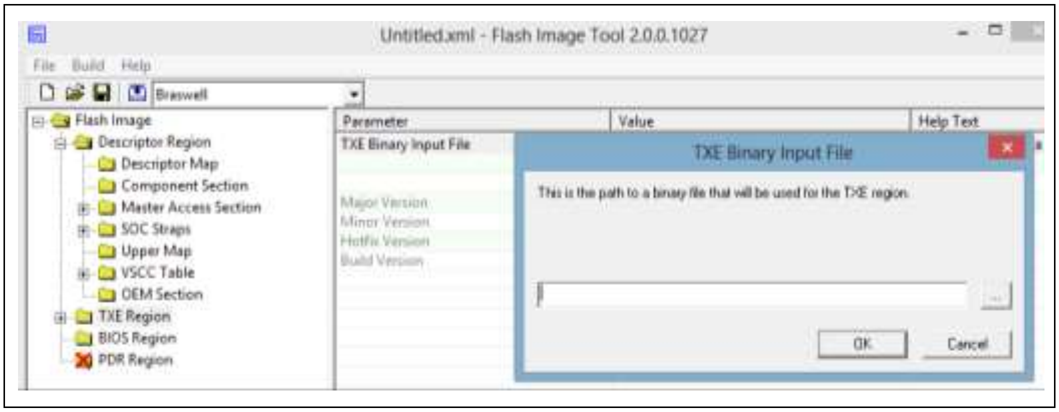
SOC straps definition will be changed upon Platform selection.

Figure 5. Select Platform Type



5. Add your SPI part VSCC table entry (please refer to System Tools user guide in chapter 3.4 for details).
6. Fill in Intel TXE Region:
   a. Select “TXE Region” and double click “TXE Binary Input File”
   b. Select \\Image Components\TXE\. .*.bin

Figure 6. Intel® TXE Region

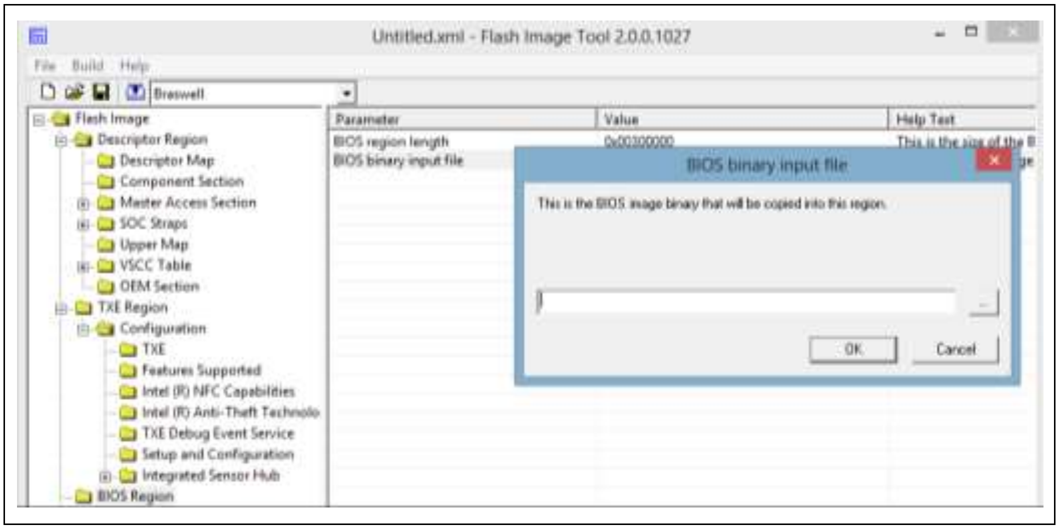


TXE Binary Input File

This is the path to a binary file that will be used for the TXE region.

7. Fill in BIOS Region:
   a. Select “BIOS Region” and double click “BIOS binary input file”
   b. Load BIOS image (*.ROM)

Figure 7. BIOS Region



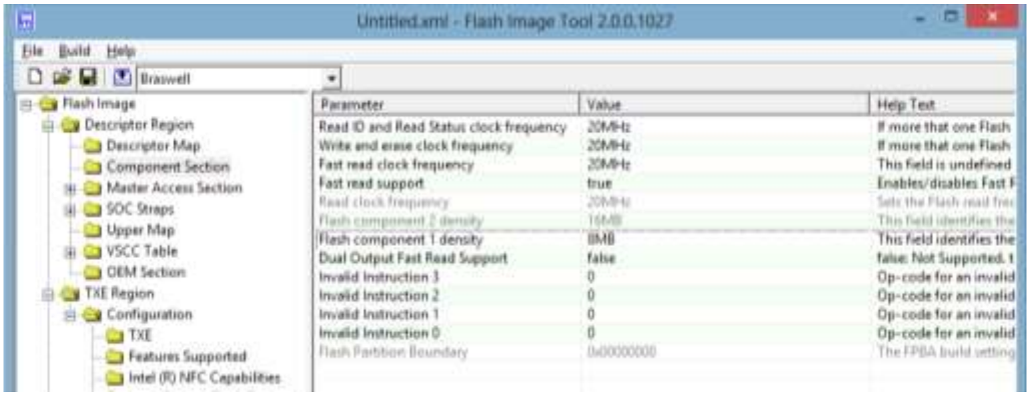
Parameter	Value	Help Text
BIOS region length	0x00300000	This is the size of the B
BIOS binary input file		

This is the BIOS image binary that will be copied into this region.

8. Configure SPI Flash image size
   a. Select Flash Image/ Component Section/ Flash component 1 density

b. Configure the size of the SPI Flash image (such as 8MB, 4MB)
c. Save XML

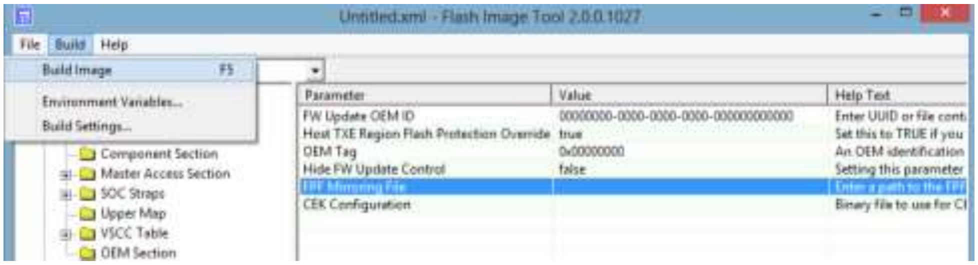
Figure 8. Configure Flash Image Size



Parameter	Value	Help Text
Read ID and Read Status clock frequency	20MHz	If more that one Flash
Write and erase clock frequency	20MHz	If more that one Flash
Fast read clock frequency	20MHz	This field is undefined
Fast read support	true	Enables/disables Fast F
Read clock frequency	20MHz	Sets the Flash read frec
Flash component 2 density	16MB	This field identifies the
Flash component 1 density	8MB	This field identifies the
Dual Output Fast Read Support	false	false: Not Supported, 1
Invalid Instruction 3	0	Op-code for an invalid
Invalid Instruction 2	0	Op-code for an invalid
Invalid Instruction 1	0	Op-code for an invalid
Invalid Instruction 0	0	Op-code for an invalid
Flash Partition Boundary	0x00000000	The FPBA build setting

9. Build image

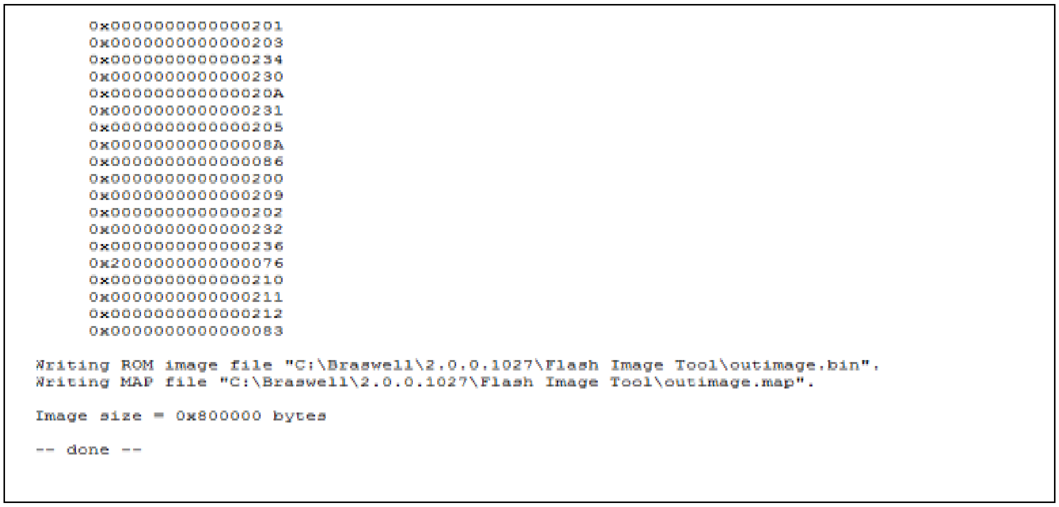
Figure 9. Build Image



Parameter	Value	Help Text
FW Update OEM ID	00000000-0000-0000-0000-000000000000	Enter UUID or file cont
Host TXE Region Flash Protection Override	true	Set this to TRUE if you
OEM Tag	0x00000000	An OEM identification
Hide FW Update Control	false	Setting this parameter
FPF Mirroring File		Enter a path to the FPF
CEK Configuration		Binary file to use for CI

The output image will be located at the path given at step 3.4.1.

Figure 10. Image Output



```
        0x0000000000000201
        0x0000000000000203
        0x0000000000000234
        0x0000000000000230
        0x000000000000020A
        0x0000000000000231
        0x0000000000000205
        0x000000000000008A
        0x0000000000000086
        0x0000000000000200
        0x0000000000000209
        0x0000000000000202
        0x0000000000000232
        0x0000000000000236
        0x2000000000000076
        0x0000000000000210
        0x0000000000000211
        0x0000000000000212
        0x0000000000000083

Writing ROM image file "C:\Braswell\2.0.0.1027\Flash Image Tool\outimage.bin".
Writing MAP file "C:\Braswell\2.0.0.1027\Flash Image Tool\outimage.map".

Image size = 0x800000 bytes

-- done --
```

3.4.2 Using Command Line

To use the command line for creating the image run the below command:

fitc.exe newfiletmpl.xml -b -txe PRODUCTION_TXE_Region.bin -bios BIOS_Region.ROM

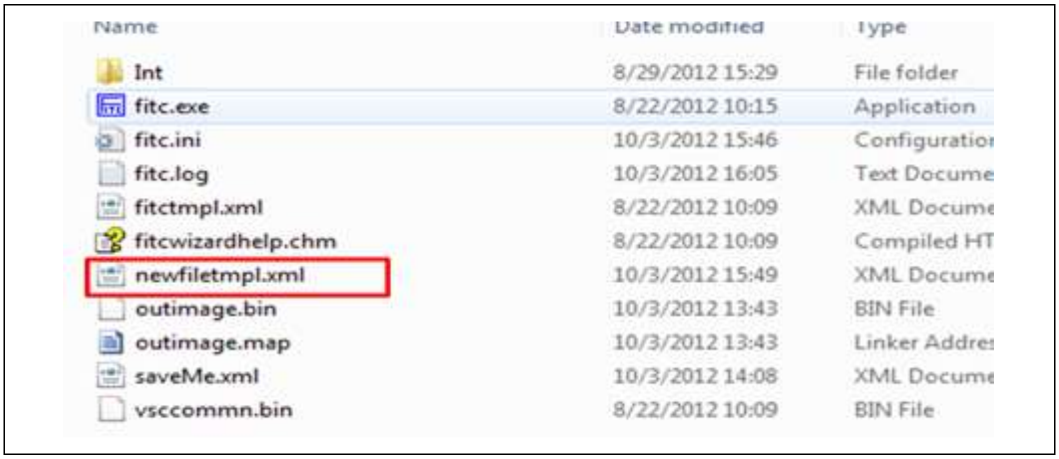
Note: If the Intel TXE Region or BIOS region is not at the same directory as FIT tool, you will need to specify the relevant path.

3.5 XML Configuration

3.5.1 Save Your Settings

10. When opening fitc.exe, it loads default settings defined in newfiletmpl.xml (located in the same folder as fitc.exe).

Figure 11. FITc Configuration



Name	Date modified	Type
Int	8/29/2012 15:29	File folder
fitc.exe	8/22/2012 10:15	Application
fitc.ini	10/3/2012 15:46	Configuratio
fitc.log	10/3/2012 16:05	Text Docume
fitctmpl.xml	8/22/2012 10:09	XML Docume
fitcwizardhelp.chm	8/22/2012 10:09	Compiled HT
newfiletmpl.xml	10/3/2012 15:49	XML Docume
outimage.bin	10/3/2012 13:43	BIN File
outimage.map	10/3/2012 13:43	Linker Addre
saveMe.xml	10/3/2012 14:08	XML Docume
vsccommn.bin	8/22/2012 10:09	BIN File

11. To save your custom settings, in the main menu select File→Save As. Select a name and location for the XML file that contains all the settings configured so far. It is recommended that you save this file in the same directory as fitc.exe is located for easy access.

Figure 12. Save Configuration

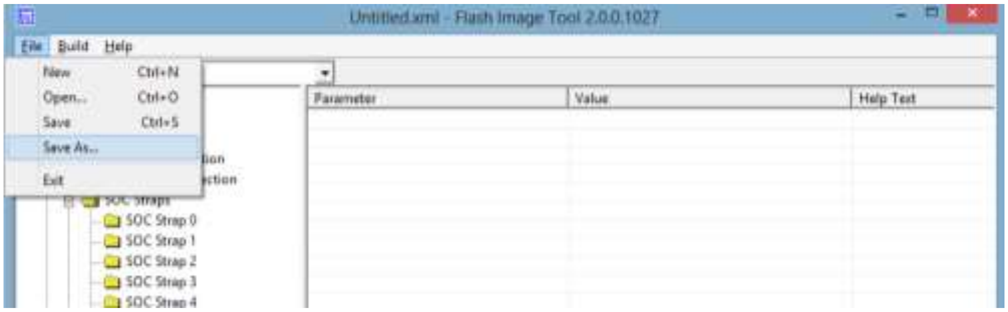
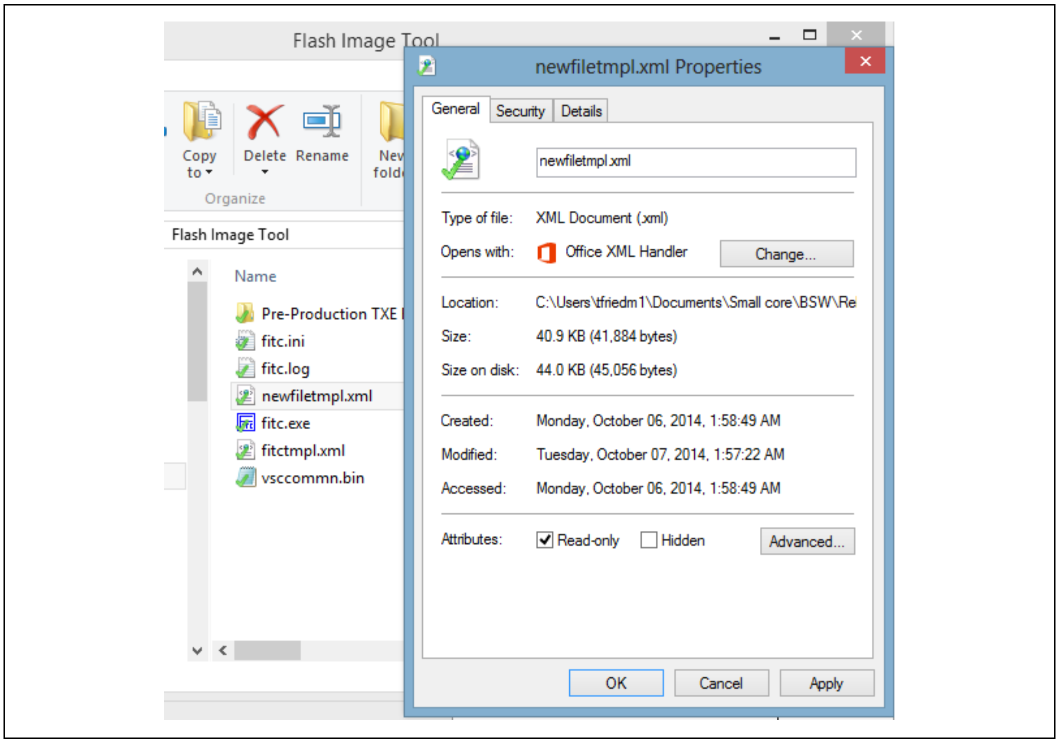


Figure 13. Configuration Protection



newfiletmpl.xml

Type of file: XML Document (.xml)

Opens with: Office XML Handler

Location: C:\Users\tfriedm1\Documents\Small core\BSW\Re

Size: 40.9 KB (41,884 bytes)

Size on disk: 44.0 KB (45,056 bytes)

Created: Monday, October 06, 2014, 1:58:49 AM

Modified: Tuesday, October 07, 2014, 1:57:22 AM

Accessed: Monday, October 06, 2014, 1:58:49 AM

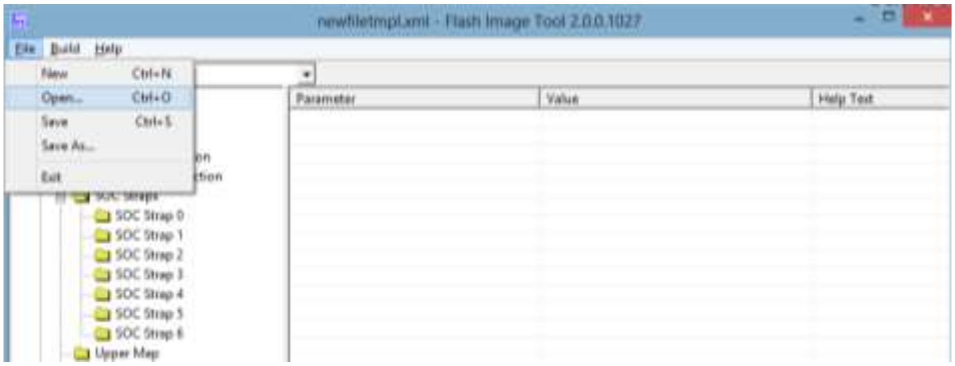
Attributes: ☑ Read-only ☐ Hidden

12. Protect configuration XML file from accidental changes by checking the “Read-only” attribute, as shown above.

3.5.2 Load FITc Configuration

Default or custom settings can be loaded by selecting File→Open in the main menu, and navigating to the desired xml configuration file.

Figure 14. Load Configuration

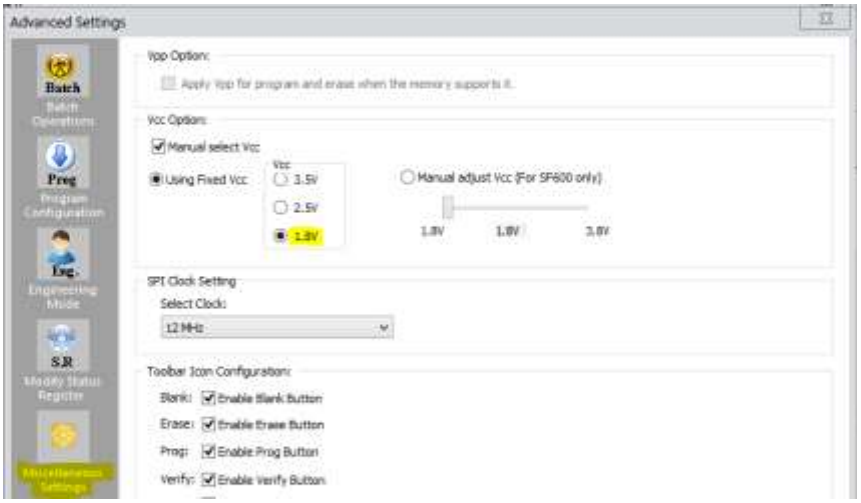


3.6 Flashing Target

3.6.1 Using DediProg*

13. Run DediProg Software.
14. Click “Detect” to verify SPI flash detection.
15. Make sure the voltage is set to 1.8v by clicking Config-> Miscellaneous settings.
16. Click “File” button and select the FW image built in Section 3.4.

Figure 15. Set Voltage



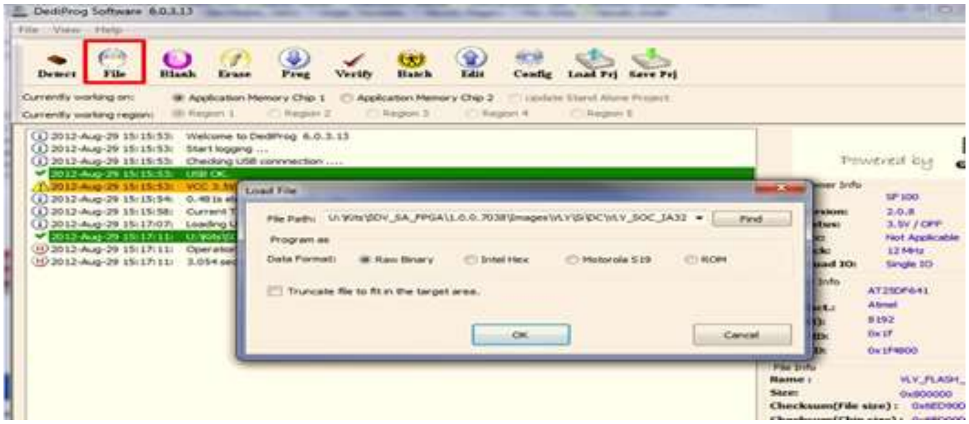
Vpp Option:
☐ Apply Vpp for program and erase when the memory supports it.

Vcc Option:
☑ Manual select Vcc
● Using Fixed Vcc
Vcc
○ 3.5V
○ 2.5V
● 1.8V
○ Manual adjust Vcc (For SF600 only)
1.8V 1.8V 3.8V

SPI Clock Setting
Select Clock: 12 MHz

Toolbar Icon Configuration:
Blank: ☑ Enable Blank Button
Erase: ☑ Enable Erase Button
Prog: ☑ Enable Prog Button
Verify: ☑ Enable Verify Button

Figure 16. Select Image



Currently working on: ● Application Memory Chip 1 ○ Application Memory Chip 2 ☐ Update Stand Alone Project
Currently working region: ● Region 1 ○ Region 2 ○ Region 3 ○ Region 4 ○ Region 5

```
2012-Aug-29 15:15:53: Welcome to DediProg 6.0.3.13
2012-Aug-29 15:15:53: Start logging ...
2012-Aug-29 15:15:53: Checking USB connection ....
2012-Aug-29 15:15:53: USB OK
2012-Aug-29 15:15:53: VCC 3.5V
2012-Aug-29 15:15:54: 0.48 is el
2012-Aug-29 15:15:58: Current T
2012-Aug-29 15:17:07: Loading U
2012-Aug-29 15:17:11: U:\Vols\D
2012-Aug-29 15:17:11: Operation
2012-Aug-29 15:17:11: 3.054 sec
```

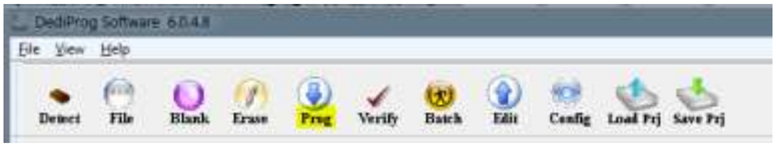
File Path: U:\Vols\SDV_SA_FPGA\1.0.0.7038\Images\VLV\SI\DC\VLV_SOC_JA32
Program as
Data Format: ● Raw Binary ○ Intel Hex ○ Motorola S19 ○ ROM
☐ Truncate file to fit in the target area.

SF100
2.0.8
3.5V / OFF
Not Applicable
12 MHz
Single IO
AT25DF641
Atmel
8192
0x1F
0x1F4800

File Info
Name : VLV_FLASH_
Size: 0x800000
Checksum(File size) : 0x8ED900

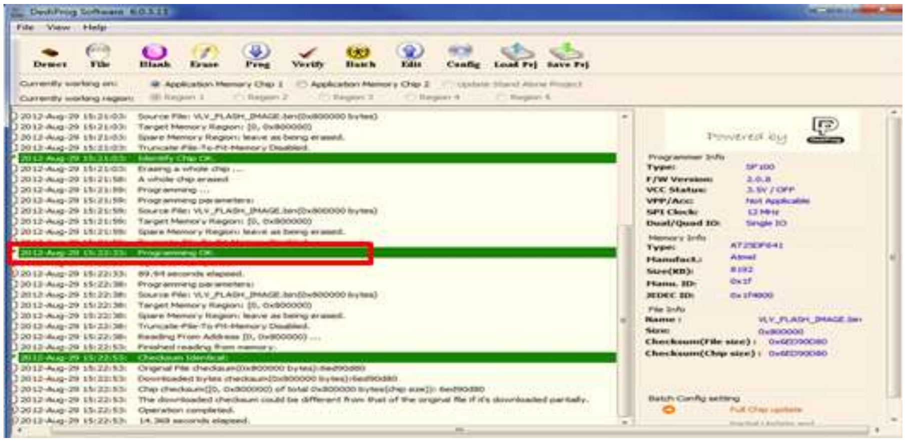
17. Click “Prog” to flash the image to the target.

Figure 17. Prog Option



18. Verify flashing was performed correctly.

Figure 18. Result Expected



3.6.2 Using FPT

1. FPT is a Windows* based tool aimed to program FW on the platform (FPT is running from the platform). Under \\System tools\Flash_Programming_Tool
2. Copy the FW image to the root folder of the FPT tool and rename it to outimage.bin. (For simplicity, we will use \\Flash_Programming_Tool\Windows when referring to FPT tool directory)
3. Open command line with administrative privileges, navigate to \\Flash_Programming_Tool\Windows or Windows64 and type:

```
Ftpt.exe -LIST
```

The system should respond with the number of SPI Flash devices available. For example:

```
--- Flash Devices Found ---

W25Q64BV ID:0xEF4017 Size: 8192KB (65536Kb)

W25Q64BV ID:0xEF4017 Size: 8192KB (65536Kb)
```

4. Program the SPI Flash image to the Flash device(s) by issuing the following command at the prompt:

```
fptw.exe /f outimage.bin
```

5. If the programming was successful, then the following message will be shown:

```
FPT Operation Passed
```

§

4 FPF Configuration File

- FPF configuration file is an input to FPT, FITc, and to Manifest Generation tool.
- FPF configuration file can be found under \\System tools\FpfConfigFile.txt.
- The file contains a list of fuses, where each line describes a fuse file in the following pattern: ***[ID]:[Value]:[Locked]***
  ID: Fuse file ID
  Value: Desired value of fuse file in **hex** digits, must be byte-aligned (For single bit file, should be 00 or 01)
  Locked: Boolean indicates if the file should be locked (TRUE/FALSE).
  Example: FUSE_FILE_ALT_BIOS_LIMIT:1FFF:FALSE

Table 1. Fuse Files

Name	Description
OEM_KEY_HASH	Hash of the key material used by the OEM to sign the Secure Boot Manifest
ALT_BIOS_LIMIT	Alternative BIOS limit. Used to locate the Alternative copy of the IBB and the manifests. This is actually the13 MSbits of the physical address. LSbits assumed to be 0xFFF
SB_EN	This bit indicates that SBit is enable and the other values where already configured
KEY_MANIFEST_ID	This is the ID of the of the Key Manifest (if 0, no Key Manifest is required)
FUSE_FILE_GLOBAL_VALID	FW Flag that marks that all OEM Fuses have been programed.
FUSE_FILE_TPM_DISABLE	FW SKU flag: marks if Firmware TPM is enabled in the Platform.

4.1 FPF Mirroring

4.1.1 Motivation

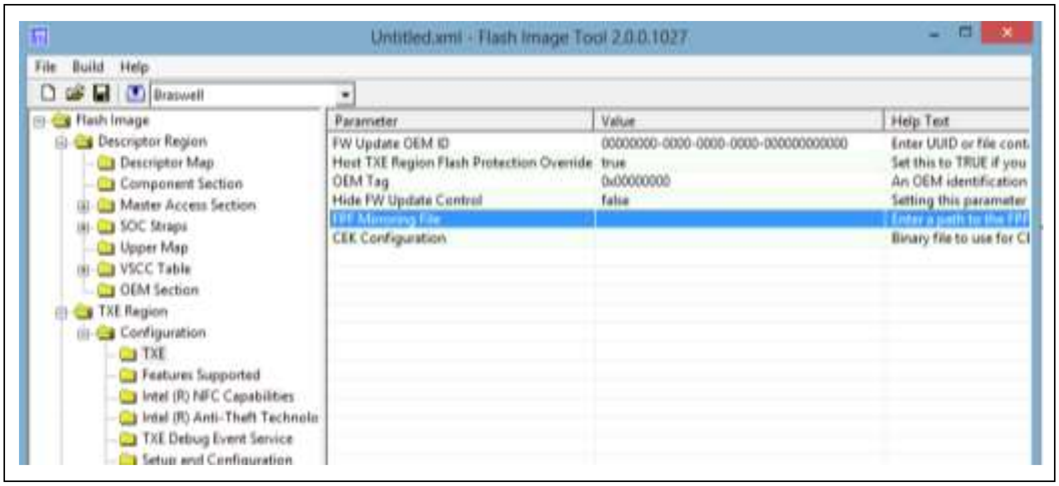
FPF mirroring allows validation\testing of FPF at the FW level, as opposed to the manufacturing phase where the FPFs are HW fused.

FPF mirroring is done using the FPF configuration file.

4.1.2 FPF Mirroring in FITc

1. Edit the FPF configuration file according to the required features. (default matches platform POR)
2. Fill in the Intel TXE region in FITc, as described in Section 3.4.1.
3. Double click on Intel FPF Mirroring File in TXE Region->Configuration->TXE and navigate to the edited FPF configuration file, as depicted.

Figure 19. FPF Mirroring in FITc



Parameter	Value	Help Text
FW Update OEM ID	00000000-0000-0000-0000-000000000000	Enter UUID or file cont
Host TXE Region Flash Protection Override	true	Set this to TRUE if you
OEM Tag	0x00000000	An OEM identification
Hide FW Update Control	false	Setting this parameter
FPF Mirroring File		Enter a path to the FPF
CEK Configuration		Binary file to use for CI

§

5 Intel® TXE Secure Boot Manifest Generation using FLAMInGO Tool

Intel® TXE Secure Boot Manifest is used to authenticate the BIOS Initial Boot Block (IBB)

Key Manifest authenticates the key used to sign the Intel TXE Secure Boot Manifest. Key Manifest is optional.

The Intel recommended method is using Key Manifest.

5.1 FLAMInGo Tool Parameters

Table 2. Parameters

Parameter Name	Description
PublicKeyFile	Public key file to calculate Sha256 from
HashFileout	Name of the file to place the SHA256 digest of the public key
FuseConfigFile	Name of the file that contains the fuses configuration
ManifestName	String that identifies the manifest, same name must be used when completing the manifest generation
IBBFile	Name of the file that contains IBB data (maximum 127kb)
SVN	Security Version Number
SigningKey	Name of the file that contains the public key of the key that is used to sign the manifest
OEMDataFile	Name of the file that contains OEM data (maximum 400 bytes)
KeyManifestFile	Name of the file that contains a valid key manifest generated by this
SignatureFile	Name of the file that contains an RSA signature of the hash file generated when creating a manifest
Unsigned	Creates the blob of the manifest without generating a hash (optional)

- For further information, type “FLAMInGO.exe -?”

5.2 Creating Secure Boot Manifest and Signing IBB

This section covers creating and signing of the SB manifest that is required to enable verified boot.

5.2.1 Prerequisites

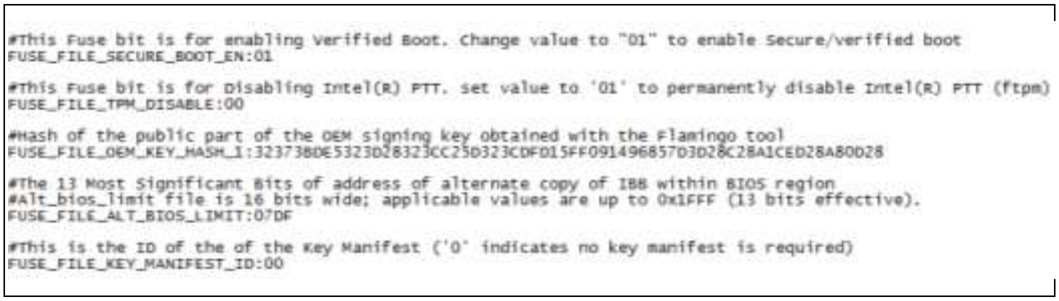
- Public and private key to be used for signing the BIOS
- FLAMinGo tool (can be found in the FW kit)
- SampleSigner (can be found in the FW kit)
- FPFconfigFile.txt (can be found the FW Kit)
- IBB (127KB) – Initial Boot Block

5.2.2 Creation and Signing Procedure

1. Open CMD and use Flamingo to hash the certificate:
   FLAMInGO.exe HashKey --out MyKeyHash.txt --key MyKey.cer
   the output will be the PubKeyHash.txt that will include the hash of the public key.

2. Insert the hash of the public key from the previous step into the FPFconfigFile.txt under FUSE_FILE_OEM_KEY_HASH_1,
   and enable secure boot by setting the FUSE_FILE_SECURE_BOOT_EN value to 01.
   The result should look like the following (with your own hash key inside):

Figure 20. FPFconfigFile.txt Example



```
#This Fuse bit is for enabling Verified Boot. Change value to "01" to enable Secure/verified boot
FUSE_FILE_SECURE_BOOT_EN:01

#This Fuse bit is for Disabling Intel(R) PTT. set value to '01' to permanently disable Intel(R) PTT (ftpm)
FUSE_FILE_TPM_DISABLE:00

#Hash of the public part of the OEM signing key obtained with the Flamingo tool
FUSE_FILE_OEM_KEY_HASH_1:32373BDE5323D28323CC25D323CDFD15FF091496857D3D28C28A1CED28A80D28

#The 13 Most Significant Bits of address of alternate copy of IBB within BIOS region
#Alt_bios_limit file is 16 bits wide; applicable values are up to 0x1FFF (13 bits effective).
FUSE_FILE_ALT_BIOS_LIMIT:07DF

#This is the ID of the of the Key Manifest ('0' indicates no key manifest is required)
FUSE_FILE_KEY_MANIFEST_ID:00
```

3. Use Flamingo to create the Secure Boot manifest
   FLAMInGo.exe ImageManCreate --Name [ManifestName] --Fuse [FuseConfigFile] [--Unsigned UnsignedFile] --KeySign [SigningKey] --SVN [SVN] --Type ImageType [--OEMDataFile <OEMDataFile>] --Image ImageFile [--KeyMan <KeyManifestFile>]
   The output will be the hash of the Secure Boot manifest and an .xml structure file with the name [Manifest Name]

4. Sign the Hash of the Secure Boot Manifest using sampleSigner:
   SampleSigner.exe [HashFileToSign] [PrivateKeyFile] [OutSignatureFile]
   The output will be the signature file.

5. Complete the process by integrating the Secure Boot manifest with the IBB
   FLAMInGo.exe ImageManComplete --Name [ManifestName] --Fuse [FuseConfigFile] [--signature SignatureFile]
   The output here will be a 128KB .bin file that will include the Secure Boot manifest (1KB) and the IBB (rest of the 127KB)

6. Copy the 128KB file to the end of the BIOS or the full SPI image.

Sample run of the tools:

```
FLAMInGO.exe HashKey --out PubKeyHash.txt --key Key.cer
FLAMInGO.exe ImageManCreate --Name manifest --Fuse FpfConfigFile_BSW.txt --KeySign
Key.cer --SVN 2 --Type IBB --OEMData oemdata.bin --Image IBB.bin
SampleSigner.exe manifest_hash.bin Key.cer manifest_sig.bin
FLAMInGO.exe ImageManComplete --Name manifest --Fuse FpfConfigFile_BSW.txt --signature
manifest_sig.bin
```

5.3 Creating Key Manifest and Signing IBB

This section covers creating and signing of the Key manifest that is required to perform verified boot using Key Manifest.

5.3.1 Prerequisites

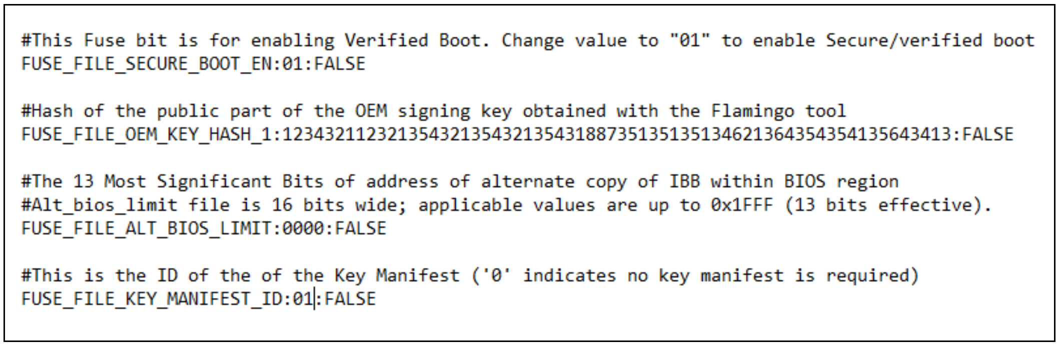
- Create two Public and private key sets and a certificate from each pair – one for the Key manifest (such as KMpubkey) and one for the secure boot manifest (such as SBpubkey).
- FLAMinGo tool (can be found in the FW kit)
- SampleSigner (can be found in the FW kit)
- FPFconfigFile.txt (can be found the FW Kit)
- IBB (127KB)

5.3.2 Creation and Signing Procedure

1. Use Flamingo to hash the Key Manifest certificate:
   FLAMInGO.exe HashKey --out KMPubKeyHash.txt --key KMKey.cer
   the output will be the KMPubKeyHash.txt that will include the hash of the public key of the Key Manifest.

2. In the FPFconfigFile.txt file:
   - Insert the hash of the Key Manifest public key from the previous section under FUSE_FILE_OEM_KEY_HASH_1
   - Enable secure boot by setting the FUSE_FILE_SECURE_BOOT_EN value to 01
   - Set the Key Manifest ID under FUSE_FILE_KEY_MANIFEST_ID, note that the Key Manifest ID should be different form 00.
   - The result should look like this (with your own hash key and Key Manifest ID inside):

Figure 21. FPFconfigFile.txt Example



```
#This Fuse bit is for enabling Verified Boot. Change value to "01" to enable Secure/verified boot
FUSE_FILE_SECURE_BOOT_EN:01:FALSE

#Hash of the public part of the OEM signing key obtained with the Flamingo tool
FUSE_FILE_OEM_KEY_HASH_1:1234321123213543213543213543188735135135134621364354354135643413:FALSE

#The 13 Most Significant Bits of address of alternate copy of IBB within BIOS region
#Alt_bios_limit file is 16 bits wide; applicable values are up to 0x1FFF (13 bits effective).
FUSE_FILE_ALT_BIOS_LIMIT:0000:FALSE

#This is the ID of the of the Key Manifest ('0' indicates no key manifest is required)
FUSE_FILE_KEY_MANIFEST_ID:01:FALSE
```

3. Use Flamingo to generate the hash of the Key Manifest:
   FLAMInGo.exe KeyManCreate --Name [ManifestName] --Fuse [FuseConfigFile] [--Unsigned <UnsignedFile>] --Keysign [SigningKey] --SVN [SVN] --Type [ImageType] –KeyCert [PublicKeytKeyFileToCerify]
   - KeyToCertify is the the certificate of the Secure Boot Manifest to be used (such as, SBpubkey)
   - SVN will be set to 0 for testing (this is relevant for Key revocation)
   - SigningKey is the certificate of the Key Manifest

4. Sign the hash using SampleSigner
   SampleSigner.exe [HashFileToSign] [KMPrivateKeyFile] [OutSignatureFile]
   the output will be the signature file.

5. Complete the creation of the Key Manifest
   FLAMInGo.exe KeyManComplete --Name [ManifestName] --Fuse [FuseConfigFile] [--signature SignatureFile]

- With this step giving the Key Manifest, the next step is to create the Secure Boot Manifest (using the Key Manifest in the process).

6. Create the Secure Boot Manifest (the OEM data file is optional)
   FLAMInGo.exe ImageManCreate --Name [ManifestName] --Fuse [FuseConfigFile] [--Unsigned UnsignedFile] --KeySign [SigningKey] --SVN [SVN] --Type ImageType [--OEMDataFile <OEMDataFile>] --Image ImageFile [--KeyMan <KeyManifestFile>]
   - The FuseConfigFile (FPFconfigFile.txt file) is the original one from the Key Manifest creating
   - SVN will be set to 0 for testing
   - The KeyManifestFile is the output of step 5
   - The output of this section is the hash of the Secure Boot Manifest

7. Sign the hash using sample signer
   SampleSigner.exe [HashFileToSign] [PrivateKeyFile] [OutSignatureFile]
   - The output of this step is the signature file ([OutSignatureFile]).

8. Complete the process by integrating the Secure Boot Manifest, the Key Manifest and the IBB

FLAMInGo.exe ImageManComplete --Name [ManifestName] --Fuse [FuseConfigFile] [--signature SignatureFile]

- The FuseConfigFile (FPFconfigFile.txt file) here is the original one used in the creation of the Key Manifest.
- The output from this step is a 132KB file that includes the Key Manifest [4KB], the Secure Boot Manifest [1KB] and the IBB [127KB].

9. Copy the 132KB file to the end of the BIOS or the full SPI image.

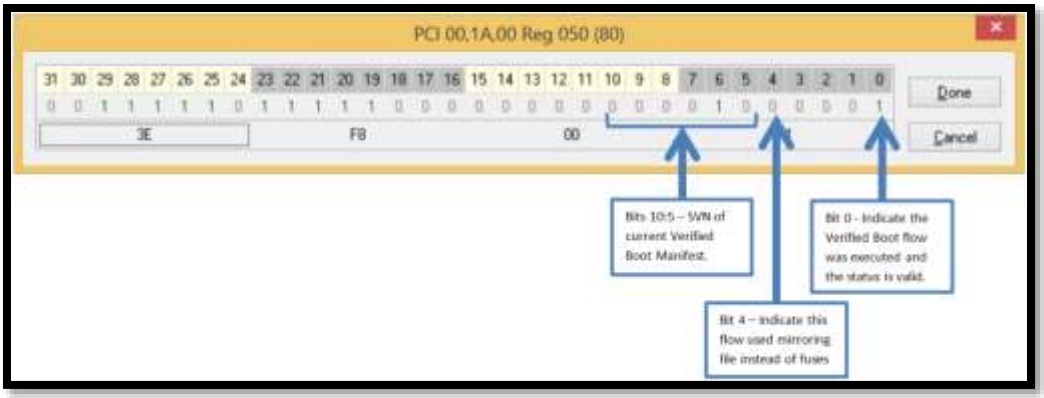
5.4 How to Confirm Verified Boot Executed Successfully

10. Dump PCI Config Space for TXE

    a. From the EFI shell run "*PCI 1A 0*"

    b. If SVN = 2, verify offset 50h shows "**5C**": verified boot executed using mirroring flow, if not executed the value will be "**00**".

    - When enabling verified boot using fused silicon FPF, the value of offset 50h will be "**41**".



PCI 00.1A.00 Reg 050 (80)

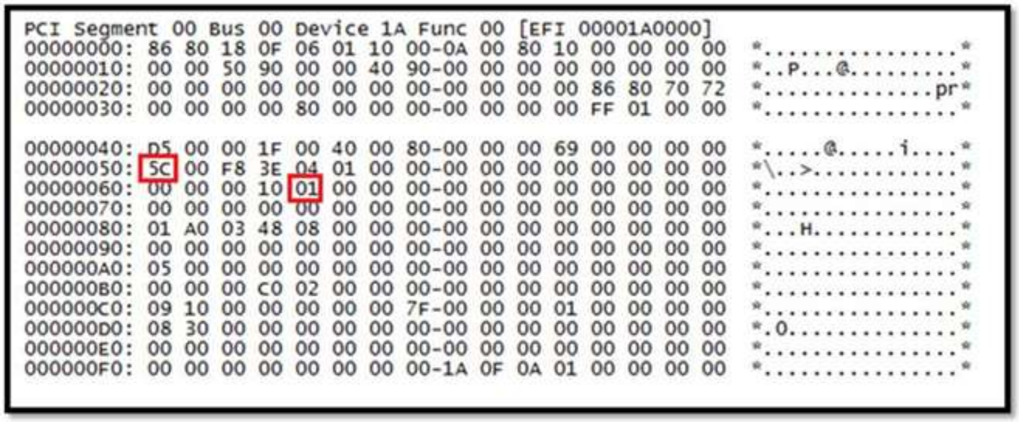
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1

3E F8 00

Bits 10:5 – SVN of current Verified Boot Manifest.

Bit 0 - Indicate the Verified Boot flow was executed and the status is valid.

Bit 4 – Indicate this flow used mirroring file instead of fuses



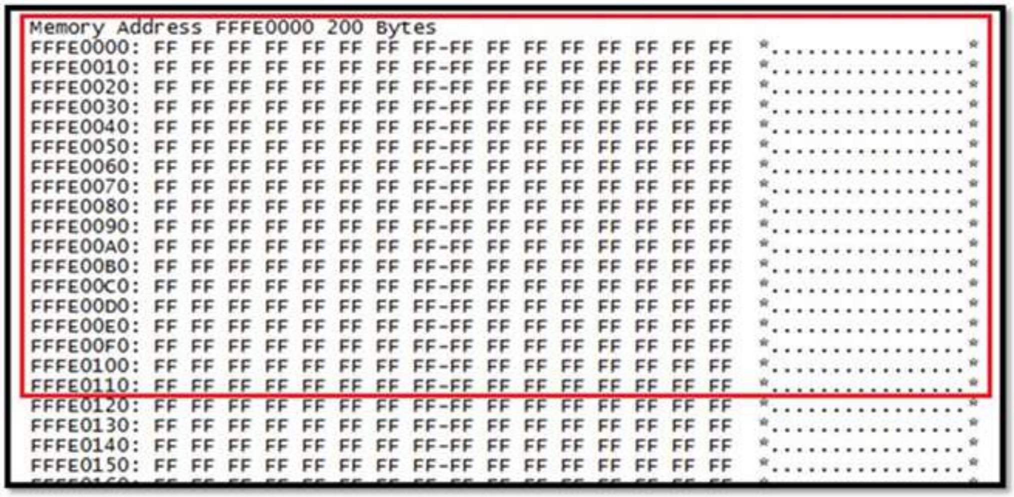
```
PCI Segment 00 Bus 00 Device 1A Func 00 [EFI 00001A0000]
00000000: 86 80 18 0F 06 01 10 00-0A 00 80 10 00 00 00 00  *................*
00000010: 00 00 50 90 00 00 40 90-00 00 00 00 00 00 00 00  *..P...@.........*
00000020: 00 00 00 00 00 00 00 00-00 00 00 86 80 70 72  *..............pr*
00000030: 00 00 00 00 80 00 00 00-00 00 00 FF 01 00 00  *................*

00000040: D5 00 00 1F 00 40 00 80-00 00 00 69 00 00 00 00  *.....@.....i....*
00000050: 5C 00 F8 3E 04 01 00 00-00 00 00 00 00 00 00 00  *\..>............*
00000060: 00 00 00 10 01 00 00 00-00 00 00 00 00 00 00 00  *................*
00000070: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
00000080: 01 A0 03 48 08 00 00 00-00 00 00 00 00 00 00 00  *...H............*
00000090: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
000000A0: 05 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
000000B0: 00 00 00 C0 02 00 00 00-00 00 00 00 00 00 00 00  *................*
000000C0: 09 10 00 00 00 00 00 7F-00 00 00 01 00 00 00 00  *................*
000000D0: 08 30 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *.0..............*
000000E0: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
000000F0: 00 00 00 00 00 00 00 00-1A 0F 0A 01 00 00 00 00  *................*
```

c. Run: “MEM FFFE0000 –b”

- Verify at this offset in memory that IBB is shielded from the user (had been copied to the SRAM) as shown below.

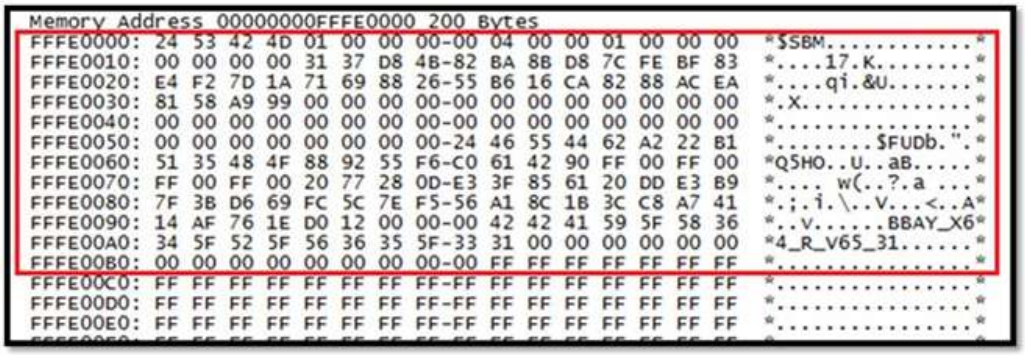
  If verified boot had not been executed, the IBB will be exposed in this offset.

Figure 22. Verified Boot Enabled – IBB is shielded from User



```
Memory Address FFFE0000 200 Bytes
FFFE0000: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE0010: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE0020: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE0030: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE0040: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE0050: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE0060: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE0070: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE0080: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE0090: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE00A0: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE00B0: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE00C0: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE00D0: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE00E0: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE00F0: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE0100: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE0110: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE0120: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE0130: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE0140: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE0150: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
```

Figure 23. Verified Boot had Not Been Executed – IBB is visible



```
Memory Address 00000000FFFE0000 200 Bytes
FFFE0000: 24 53 42 4D 01 00 00 00-00 04 00 00 01 00 00 00  *$SBM............*
FFFE0010: 00 00 00 00 31 37 D8 4B-82 BA 8B D8 7C FE BF 83  *....17.K........*
FFFE0020: E4 F2 7D 1A 71 69 88 26-55 B6 16 CA 82 88 AC EA  *....qi.&U.......*
FFFE0030: 81 58 A9 99 00 00 00 00-00 00 00 00 00 00 00 00  *.X..............*
FFFE0040: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
FFFE0050: 00 00 00 00 00 00 00 00-24 46 55 44 62 A2 22 B1  *........$FUDb.".*
FFFE0060: 51 35 48 4F 88 92 55 F6-C0 61 42 90 FF 00 FF 00  *Q5HO..U..aB.....*
FFFE0070: FF 00 FF 00 20 77 28 0D-E3 3F 85 61 20 DD E3 B9  *.... w(..?.a ...*
FFFE0080: 7F 3B D6 69 FC 5C 7E F5-56 A1 8C 1B 3C C8 A7 41  *.;.i.\..V...<..A*
FFFE0090: 14 AF 76 1E D0 12 00 00-00 42 42 41 59 5F 58 36  *..v......BBAY_X6*
FFFE00A0: 34 5F 52 5F 56 36 35 5F-33 31 00 00 00 00 00 00  *4_R_V65_31......*
FFFE00B0: 00 00 00 00 00 00 00 00-00 FF FF FF FF FF FF FF  *................*
FFFE00C0: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE00D0: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
FFFE00E0: FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF  *................*
```

§

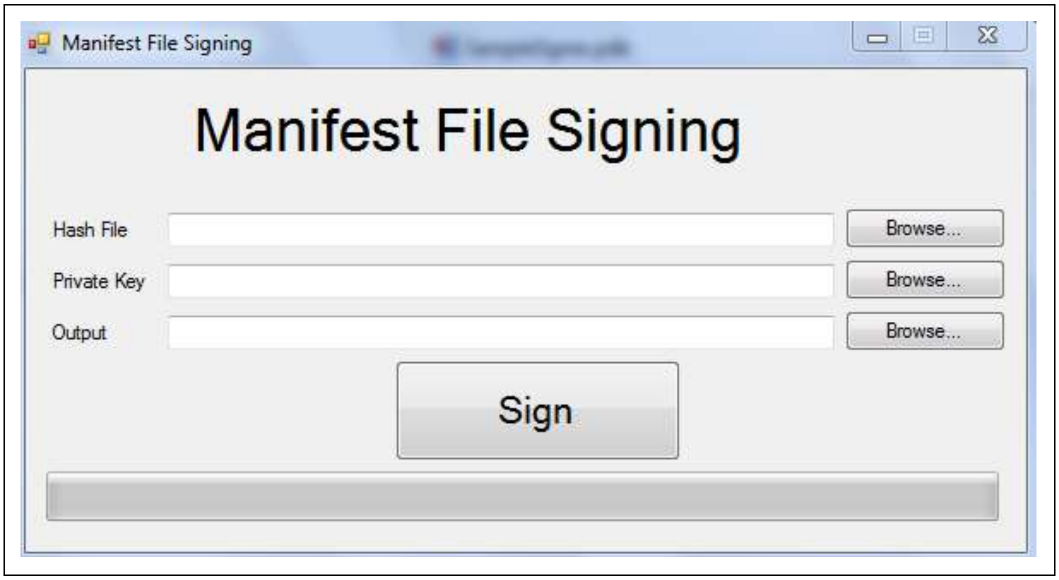
6 Sample Signer – Verified Boot Manifest Signing Reference Tool

Signing reference tool can be found under \\Flash Manifest Generation Tool \SampleSigner.exe.

Parameters:

1. Hash file to sign
2. Private key
3. The output location of the signed file.

Figure 24. Signing Tool



§

7 Intel® Trusted Execution Engine Interface (Intel® TXEI) Driver

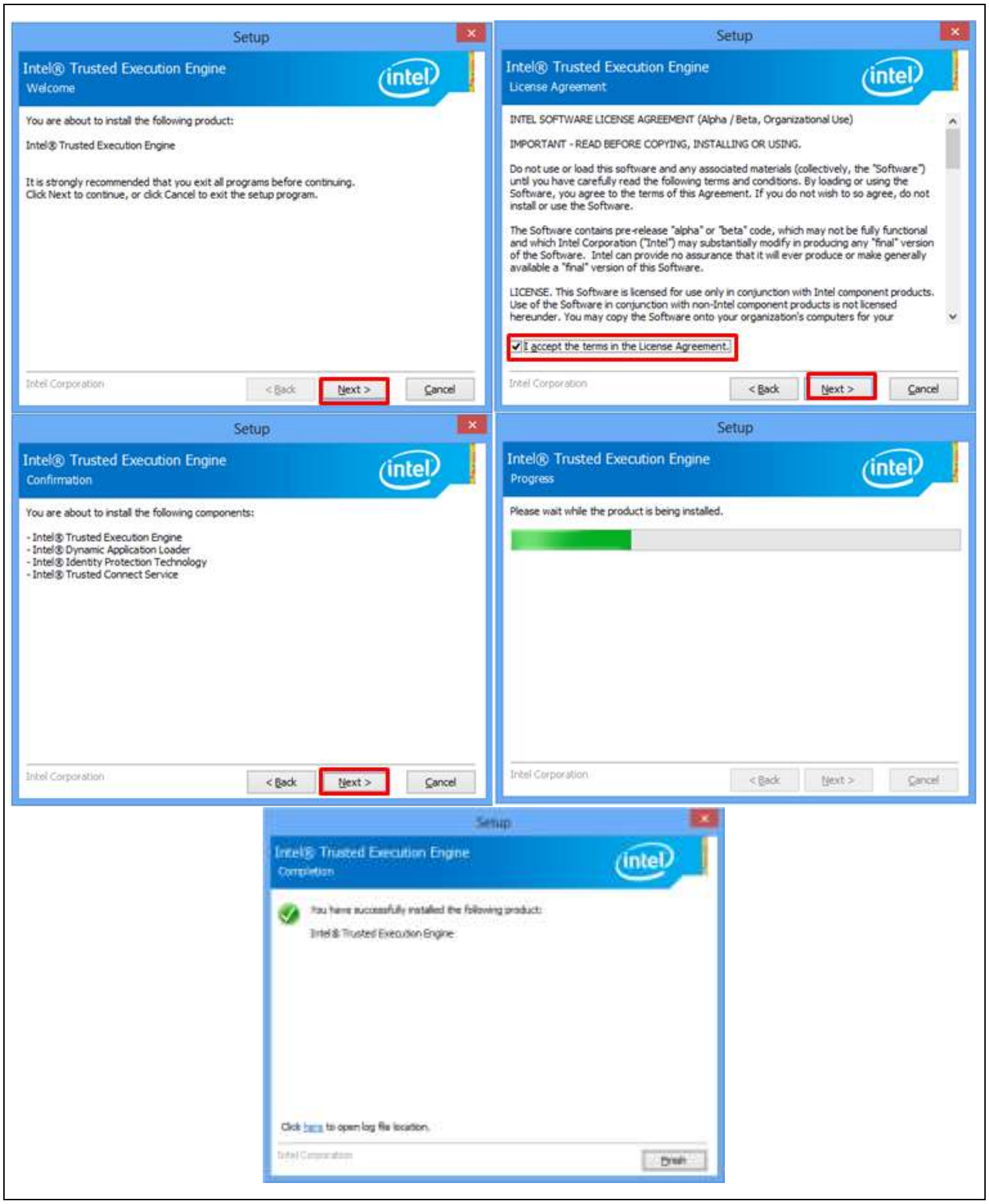
7.1 Install the Intel® TXEI Driver using Installer

1. Navigate to the root folder of the Intel TXE Installer (\\Installers)
2. Double click "SetupTXE.exe"
3. Follow the installation procedure as shown in Figure 25. Intel® TXE Installation Steps.
4. For Windows 7 users only: Intel® Trusted Execution Engine Interface (Intel® TXEI) Driver uses KMDF (WDF) 1.11, which is built-in on Windows 8 and Windows 8.1. However, Windows 7 does not have it.

Install Kernel-Mode Driver Framework (KMDF) version 1.1. Otherwise, yellow bang appears on Intel TXEI device upon installation.

Follow instructions in this link: KB2685811

Figure 25. Intel® TXE Installation Steps



You are about to install the following product:

Intel® Trusted Execution Engine

It is strongly recommended that you exit all programs before continuing. Click Next to continue, or click Cancel to exit the setup program.

INTEL SOFTWARE LICENSE AGREEMENT (Alpha / Beta, Organizational Use)

IMPORTANT - READ BEFORE COPYING, INSTALLING OR USING.

Do not use or load this software and any associated materials (collectively, the "Software") until you have carefully read the following terms and conditions. By loading or using the Software, you agree to the terms of this Agreement. If you do not wish to so agree, do not install or use the Software.

The Software contains pre-release "alpha" or "beta" code, which may not be fully functional and which Intel Corporation ("Intel") may substantially modify in producing any "final" version of the Software. Intel can provide no assurance that it will ever produce or make generally available a "final" version of this Software.

LICENSE. This Software is licensed for use only in conjunction with Intel component products. Use of the Software in conjunction with non-Intel component products is not licensed hereunder. You may copy the Software onto your organization's computers for your

☑ I accept the terms in the License Agreement.

You are about to install the following components:

- Intel® Trusted Execution Engine
- Intel® Dynamic Application Loader
- Intel® Identity Protection Technology
- Intel® Trusted Connect Service

Please wait while the product is being installed.

You have successfully installed the following product:

Intel® Trusted Execution Engine

Click here to open log file location.

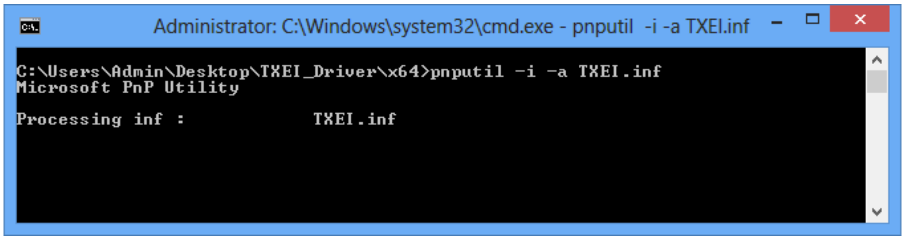
7.2 Install the Intel® TXEI Driver using Command Line

Use this option as an alternative method. Installing the driver using the Installer is the recommended method.

1. Verify Intel TXEI driver is not installed on your system by:
    a. Open Device Manager (right click on My Computer -> Manage)
    b. Go to Device Manager subcategory
    c. Open System devices subcategory
    d. Look for device called "PCI Encryption/ Decryption Controller".

    If “Intel® Trusted Execution Engine Interface” is already installed, uninstall it by right click->uninstall and check the "*Delete the driver software for this device*" checkbox.
2. Open command line with admin privileges and navigate to the root folder of TXEi.inf (\\TXEI_Driver\x64 or x86)
3. Type “pnputil.exe –i –a TXEI.inf

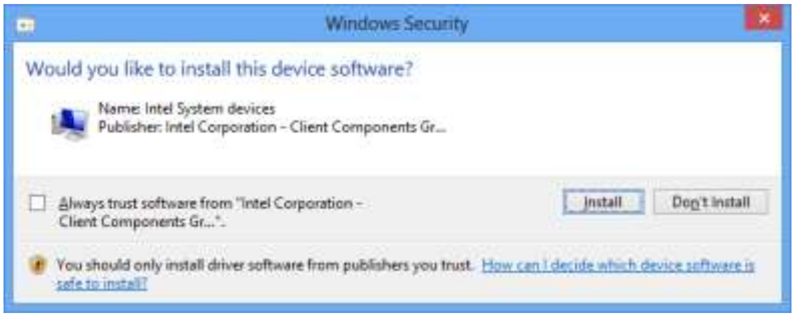
Figure 26. Intel® TXEI Installation


Administrator: C:\Windows\system32\cmd.exe - pnputil -i -a TXEI.inf

```
C:\Users\Admin\Desktop\TXEI_Driver\x64>pnputil -i -a TXEI.inf
Microsoft PnP Utility

Processing inf :            TXEI.inf
```

1. Select “*Install*”

Figure 27. Windows* Security Prompt



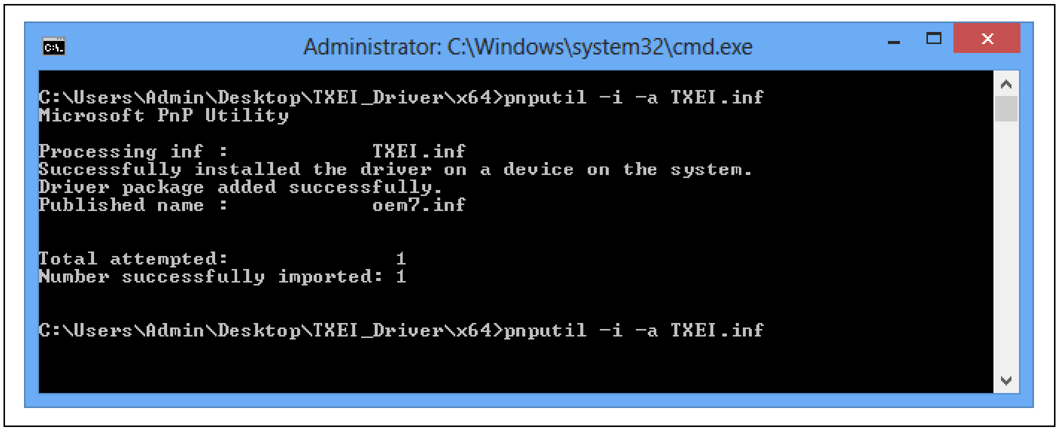
Would you like to install this device software?

Name: Intel System devices
Publisher: Intel Corporation - Client Components Gr...

☐ Always trust software from "Intel Corporation - Client Components Gr...".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

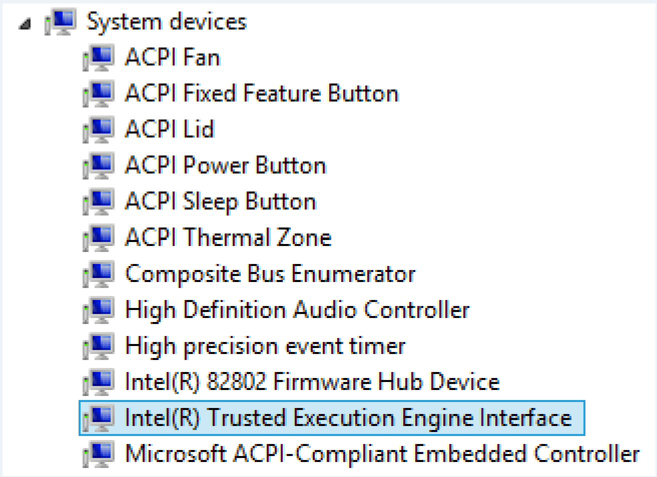
Figure 28. Finishing Intel® TXEI Installation



```
C:\Users\Admin\Desktop\TXEI_Driver\x64>pnputil -i -a TXEI.inf
Microsoft PnP Utility

Processing inf :            TXEI.inf
Successfully installed the driver on a device on the system.
Driver package added successfully.
Published name :            oem7.inf


Total attempted:              1
Number successfully imported: 1


C:\Users\Admin\Desktop\TXEI_Driver\x64>pnputil -i -a TXEI.inf
```

5. Verify Intel® TXEI is installed by referring to device manager→System devices.

Figure 29. Verify Intel® TXEI Installation in Device Manager



- System devices
  - ACPI Fan
  - ACPI Fixed Feature Button
  - ACPI Lid
  - ACPI Power Button
  - ACPI Sleep Button
  - ACPI Thermal Zone
  - Composite Bus Enumerator
  - High Definition Audio Controller
  - High precision event timer
  - Intel(R) 82802 Firmware Hub Device
  - Intel(R) Trusted Execution Engine Interface
  - Microsoft ACPI-Compliant Embedded Controller

§

8 Using WinPE Tools

When using Windows* FW tools in WinPE, remember to load the TXEI driver at every boot.

This can be done by: *X:\Windows\System32>drvload.exe <path>\TXEI.inf.*

TXEI.inf can be found in every FW kit release.

§

9 Using EFI System Tools in UEFI Shell with UEFI Secure Boot Enabled Option

Due to Microsoft’s mandatory UEFI Shells and related applications requirement (System.Fundamentals.Firmware.UEFISecureBoot), when running Intel or customer manufacturing utilities in UEFI shell, customers are required to disable UEFI Secure boot via BIOS setup menu or UEFI variable. If the OEM/ODM wants to run a specific EFI tool that needs to run with UEFI secure boot, the OEM/ODM will sign that EFI tool with their OEM key.

§

10 Intel® TXEManuf

Intel TXEManuf tool will auto-detect the hardware/firmware SKU, and automatically runs tests to check functionality of their related features on the manufacturing line.

10.1 Prerequisites

Intel® TXEI driver must be installed. (Refer to Chapter 8 for instructions on how to install Intel® TXEI driver).

10.2 TXEManuf Usage

For detailed instructions, refer to "Intel TXEManuf" section in "System Tools User Guide" document, located at the System Tools folder.

§

11 Intel® TXE FW Update

Intel FWUpdate tool allows an end user, such as an IT administrator, to update Intel TXE FW without having to reprogram the entire flash device. It then verifies that the update was successful.

11.1 Prerequisites

Intel® TXEI driver must be installed. (Refer to Chapter 8 for instructions on how to install Intel® TXEI driver).

11.2 FWUpdate Usage

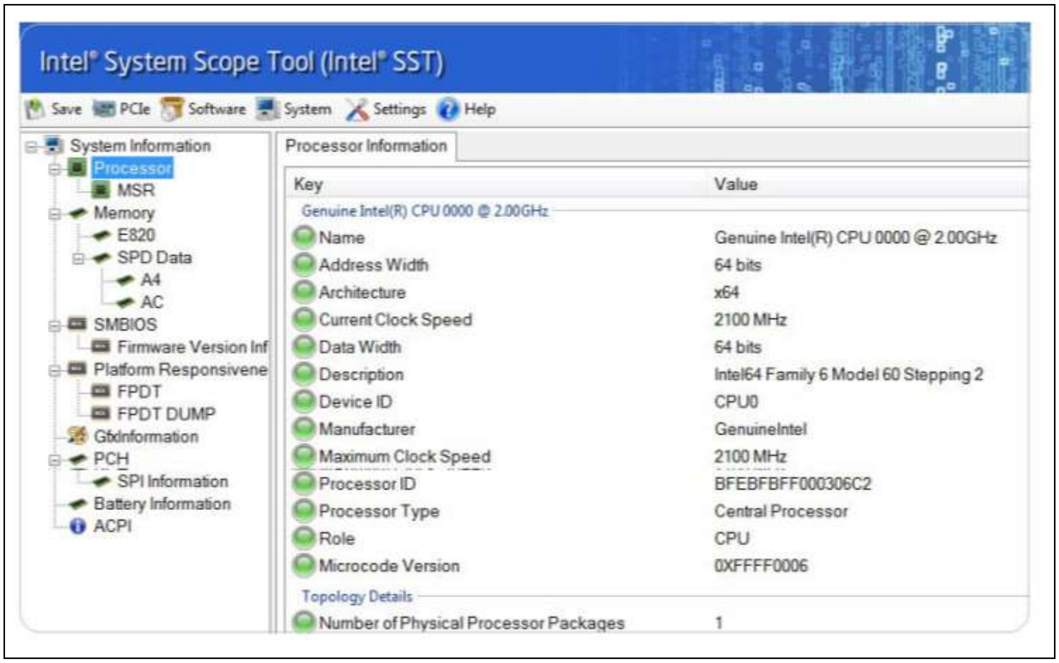
For detailed instructions, refer to “Intel TXE FW Update” section in the “System Tools User Guide” document located at the root folder.

§

12 Intel® System Scope Tool (Intel® SST)

The Intel® System Scope Tool (Intel® SST) is a tool that provides the complete snapshot of the system including both hardware and the software details.

Figure 30. Intel® System Scope Tool Screen Shot



Key	Value
Genuine Intel(R) CPU 0000 @ 2.00GHz	
Name	Genuine Intel(R) CPU 0000 @ 2.00GHz
Address Width	64 bits
Architecture	x64
Current Clock Speed	2100 MHz
Data Width	64 bits
Description	Intel64 Family 6 Model 60 Stepping 2
Device ID	CPU0
Manufacturer	GenuineIntel
Maximum Clock Speed	2100 MHz
Processor ID	BFEBFBFF000306C2
Processor Type	Central Processor
Role	CPU
Microcode Version	0XFFFF0006
Topology Details	
Number of Physical Processor Packages	1

- This tool is useful for providing full platform information for debugging purposes.
- An output file can be saved in .html format and attached to Braswell sightings.
- This tool can be found on Intel® VIP inside the Braswell Compliance Kits.

§

13 FITc Soft Straps

Table 3. SOC Strap 0

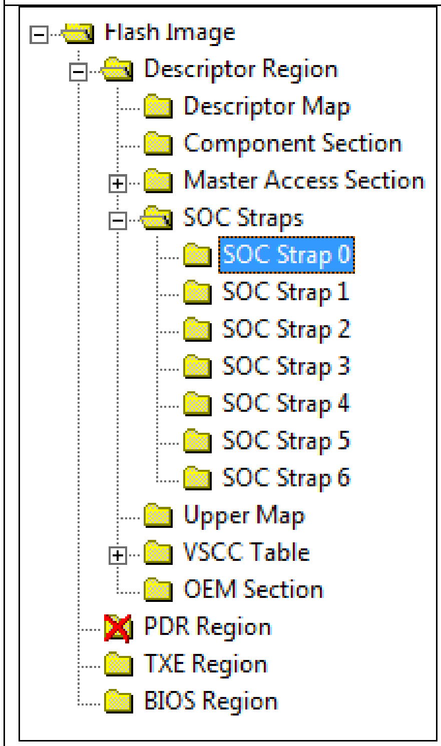
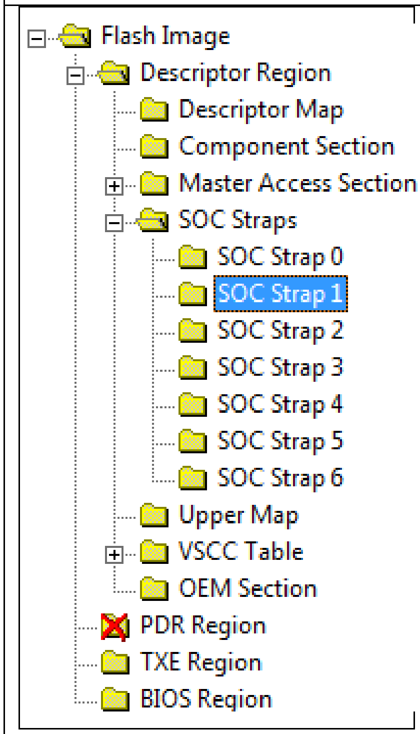
Location	Parameter	Values	Description
 Flash Image Descriptor Region Descriptor Map Component Section Master Access Section SOC Straps SOC Strap 0 SOC Strap 1 SOC Strap 2 SOC Strap 3 SOC Strap 4 SOC Strap 5 SOC Strap 6 Upper Map VSCC Table OEM Section PDR Region TXE Region BIOS Region	BIOS Protected Range 4 Base	0x0000	Specifies the lower base of the BIOS protected range number 4. Address bits [11:0] are assumed to be 12'h000 for the base comparison. (goes to bits [12:0] at register: [Protected_Range_4] PR4 (@0x84)).
	BIOS Protected Range 4 Limit	0x0000	Specifies the upper limit of the BIOS protected range number 4. Address bits [11:0] are assumed to be 12'hFFF for the limit comparison. (Goes to bits [28:16] at register: [Protected_Range_4] PR4 (@0x84)).
	BIOS Protected Range 4 Write Protection Enable	True False (default)	When set, this bit indicates that the Base and Limit fields are valid and that writes directed to addresses between them (inclusive) must be blocked by hardware. The base and limit fields are ignored when this bit is cleared. Disabling this protected range could be done also by the security override pin strap. (this soft strap and the security override pin strap are reflected into bit 31 at register: [Protected_Range_4] PR4 (@0x84)).

Table 4. SOC Strap 1

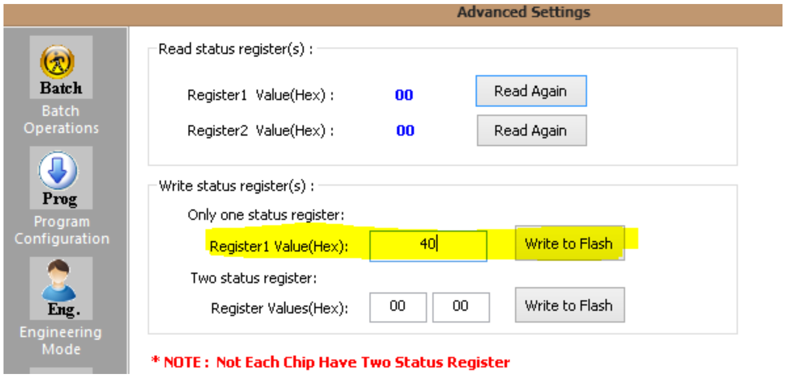
Location	Parameter	Values	Description
 Flash Image Descriptor Region Descriptor Map Component Section Master Access Section SOC Straps SOC Strap 0 SOC Strap 1 SOC Strap 2 SOC Strap 3 SOC Strap 4 SOC Strap 5 SOC Strap 6 Upper Map VSCC Table OEM Section PDR Region TXE Region BIOS Region	Dual Output Read Enable	False (default) True	False: Dual output read is disabled True: Dual output read is enabled This soft-strap only has effect if dual output read is discovered as supported via SFDP. If SFDP table is not detected, this strap has no effect and dual output read is controlled via flash descriptor.
	Dual I/O Read Enable	False (default) True	False: Dual I/O read is disabled True: Dual I/O read is enabled This soft-strap only has effect if dual output read is discovered as supported via SFDP. If SFDP table is not detected, this strap has no effect.
	Quad Output Read Enable	False (default) True	False: Quad output read is disabled True: Quad output read is enabled This soft-strap only has effect if dual output read is discovered as supported via SFDP. If SFDP table is not detected, this strap has no effect.
	Quad I/O Read Enable	False (default) True	False: Quad I/O read is disabled True: Quad I/O read is enabled This soft strap has effect only if quad I/O read is discovered as supported via SFDP

Keeping QORE (Quad Output Read Enable) and QIORE (Quad I/O Read Enable) enabled as shown below, SPI part QE (Quad Enable) bit has to be set in the SPI part SR (Status Register). Follow the following instructions in order to properly enable your platform to boot with this capability:

1. Using FITC tools, create your boot image with QORE and QIORE set to “True” (default).

This image cannot currently be displayed.

2. Enable the QUAD MODE on the SPI part installed on your Braswell design (only done once on SPI part)
   a. BSW RVP board by setting the QE bit on SPI part MX25U6435F, it can be done by writing 0x40 to the status register by DediProg
      i. In Config → Modify Status Register → Write Status register(s) → Register1 Value(Hex)



Read status register(s) :

Register1 Value(Hex) : 00

Register2 Value(Hex) : 00

Write status register(s) :

Only one status register:

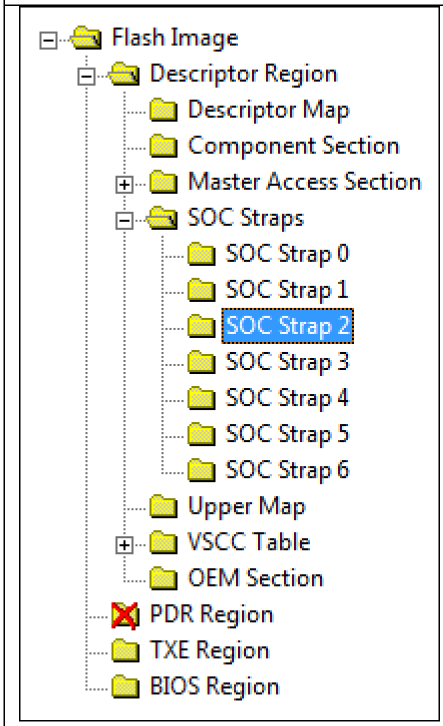
Register1 Value(Hex): 40

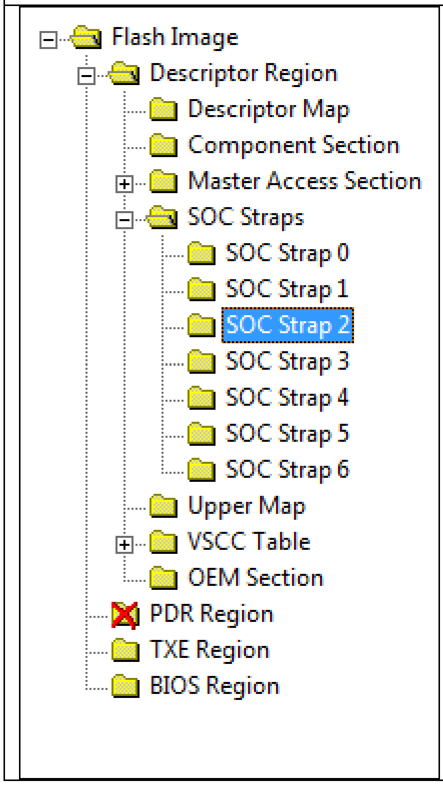
Two status register:

Register Values(Hex): 00 00

*** NOTE : Not Each Chip Have Two Status Register**

Table 5. SOC Strap 2

Location	Parameter	Values	Description
 Flash Image Descriptor Region Descriptor Map Component Section Master Access Section SOC Straps SOC Strap 0 SOC Strap 1 SOC Strap 2 SOC Strap 3 SOC Strap 4 SOC Strap 5 SOC Strap 6 Upper Map VSCC Table OEM Section PDR Region TXE Region BIOS Region	SIO1_F3_Disable	False (default) True	Disable LPSS1 function 3 (HSUART#1). false = enable. true = disable
	SIO1_F4_Disable	False (default) True	Disable LPSS1 function 4 (HSUART#2). false = enable. true = disable
	SCC eMMC 4.41 Disable	False (default) True	Disable legacy eMMC 4.41
	SCC SDIO Disable	False (default) True	Disable SDIO. false = enable. true = disable
	SCC SDCARD Disable	False (default) True	Disable SDCARD. false = enable. true = disable
	SCC Disable	False (default) True	Disable the Storage and Communications Cluster (SCC). The Storage and Communications cluster includes the following controllers: SD card reader, eMMC.
	HDA Disable	False (default) True	Disable HD Audio. false = enable. true = disable
	USB3 (OTG) Disable	True (default) False	Disable OTG
	SATA Disable	False (default) True	Disable SATA. False = enable. true = disable
	EHCI Disable	False (default) True	Disable USB: false = enable, true = disable
	DMA2 Disable	False (default) True	Disable DMA2

Location	Parameter	Values	Description
 Flash Image Descriptor Region Descriptor Map Component Section Master Access Section SOC Straps SOC Strap 0 SOC Strap 1 SOC Strap 2 SOC Strap 3 SOC Strap 4 SOC Strap 5 SOC Strap 6 Upper Map VSCC Table OEM Section PDR Region TXE Region BIOS Region	SIO2 F1 Disable	False (default) True	Disable LPSS2 function 1 (I2C#1). false = enable. true = disable

	SIO2 F2 Disable	False (default)
		True
			Disable LPSS2 function 2 (I2C#2).
			False = enable.
			True = disable

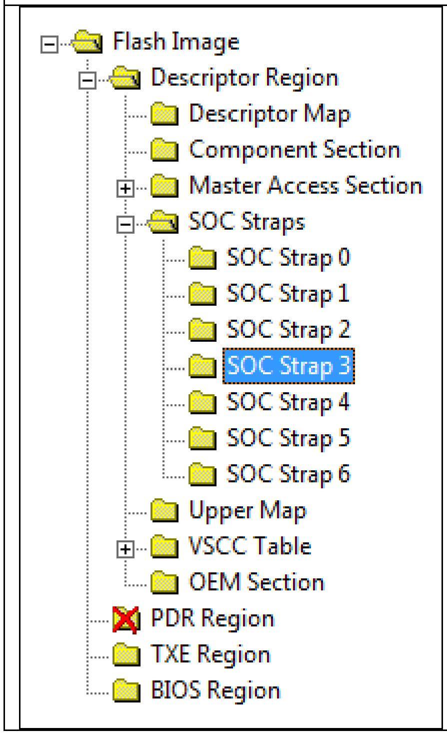
	SIO2 F3 Disable	False (default)
		True
			Disable LPSS2 function 3 (I2C#3).
			False = enable.
			True = disable

	SIO2 F4 Disable	False (default)
		True
			Disable LPSS2 function 4 (I2C#4).
			False = enable.
			True = disable

	SIO2 F5 Disable	False (default)
		True
			Disable LPSS2 function 5 (I2C#5).
			False = enable.
			True = disable

Table 6. SOC Strap 3

Location	Parameter	Values	Description

 Flash Image
Descriptor Region
Descriptor Map
Component Section
Master Access Section
SOC Straps
SOC Strap 0
SOC Strap 1
SOC Strap 2
SOC Strap 3
SOC Strap 4
SOC Strap 5
SOC Strap 6
Upper Map
VSCC Table
OEM Section
PDR Region
TXE Region
BIOS Region
	SIO2 F6 Disable	False (default)
		True
			Disable LPSS2 function 6 (I2C#6).
			False = enable.
			True = disable

	SIO2 F7 Disable	False (default)
		True
			Disable LPSS2 function 7 (I2C#7).
			False = enable.
			True = disable

	No Reboot	False (default) True	Disable PLTRST after TCO WDT second time expiration Permanently disables the SOC’s watchdog timer mechanism from resetting the system.

	SMBus Disable	False (default)
		True
			Disable SMBUS.
			False = enable.
			True = disable

Table 7. SOC Strap 4

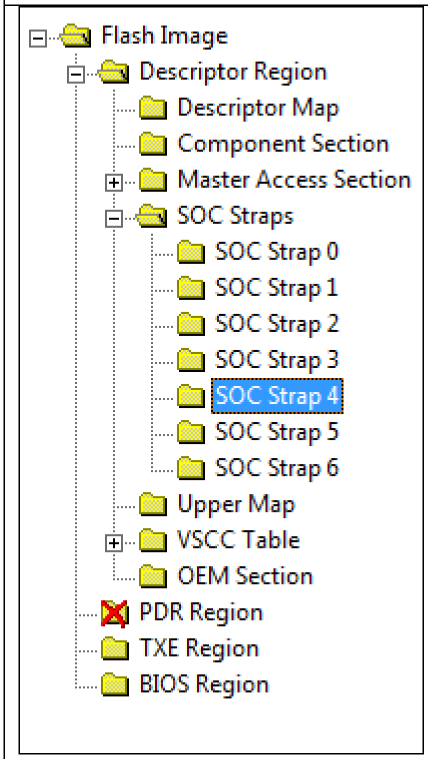
Location	Parameter	Values	Description
 Flash Image Descriptor Region Descriptor Map Component Section Master Access Section SOC Straps SOC Strap 0 SOC Strap 1 SOC Strap 2 SOC Strap 3 SOC Strap 4 SOC Strap 5 SOC Strap 6 Upper Map VSCC Table OEM Section PDR Region TXE Region BIOS Region	LPC GPIO Select	1'b0: LPC (default) 1'b1: GPIO	Select the usage of LPC pins
	LPCCLK_FREQUENCY	1’b1:25 Mhz	LPC Clock frequency selection
	Vddq_voltage	False (default) True	Enable Vddq_voltage
	Suspwrdnack_cfg_mode	False (default) True	Enable Suspwrdnack behavior (S5=>G3): True = enable, False = disable.
	PCIe0_Disable_New	False (default) True	Disable PCIE Port0: False = enable, True = disable
	PCIe1_Disable_New	False (default) True	Disable PCIE Port1: False = enable, True = disable
	PCIe2_Disable_New	False (default) True	Disable PCIE Port2: False = enable, True = disable
	PCIe3_Disable_New	False (default) True	Disable PCIE Port3: False = enable, True = disable
	PCIE_clkreq_enable_wake	True (default) False	False = enable, True = disable

Table 8. SOC Strap 5

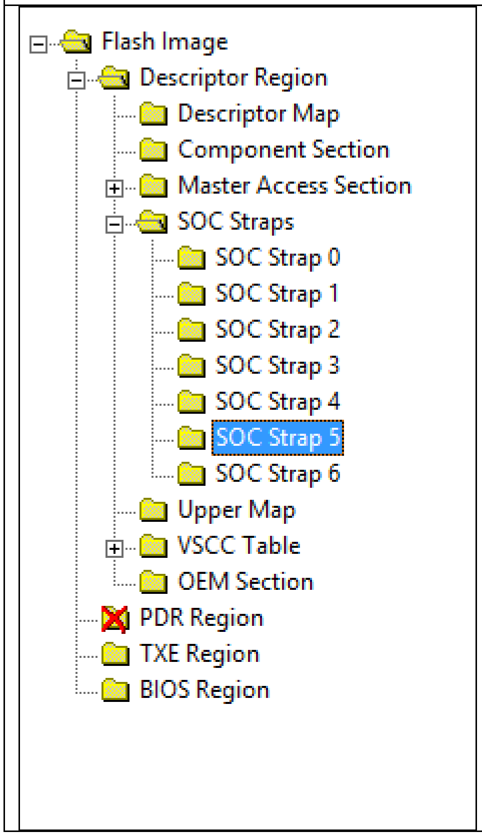
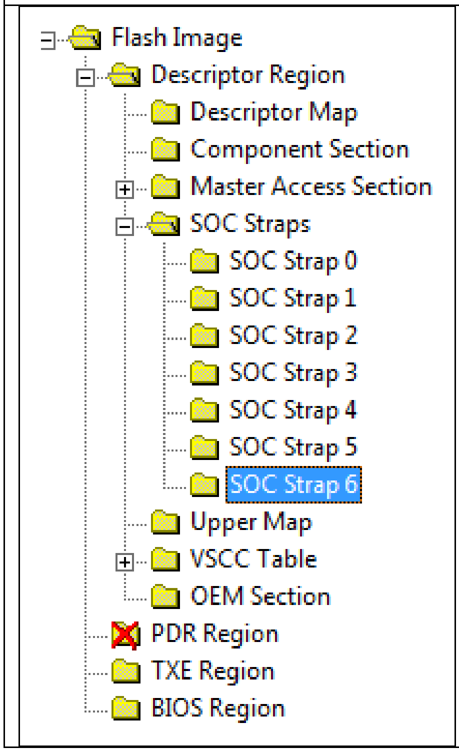
Location	Parameter	Values	Description
 Flash Image Descriptor Region Descriptor Map Component Section Master Access Section SOC Straps SOC Strap 0 SOC Strap 1 SOC Strap 2 SOC Strap 3 SOC Strap 4 SOC Strap 5 SOC Strap 6 Upper Map VSCC Table OEM Section PDR Region TXE Region BIOS Region	LPCCLK1_EN	1'b0: Disable LPC CLK#1 1'b1: Enable LPC CLK#1	Enable clock output on LPCCLK1

Table 9. SOC Strap 6

Location	Parameter	Values	Description
 Flash Image Descriptor Region Descriptor Map Component Section Master Access Section SOC Straps SOC Strap 0 SOC Strap 1 SOC Strap 2 SOC Strap 3 SOC Strap 4 SOC Strap 5 SOC Strap 6 Upper Map VSCC Table OEM Section PDR Region TXE Region BIOS Region	Root Port Configuration	11: 1x4 Port 1 (x4) 10 :2x2 Port 1 (x2), port 3 (x2) 01: 1x2, 2 x1s Port 1 (x2), port 3 (x1), port4 (x1) 00: 4x1s Port 1 (x1), Port 2 (x1), Port 3(x1), Port 4 (x1) (default)	See below note.
	Lane reversal	False (default) True	Lane reversal

NOTES: For "Root Port Configuration":

1. If the Value of "Root Port Configuration" has been changed to "1": 1x2, 2 x1s Port 1 (x2), Port 3 (x1), Port 4 (x1):
   a. Require change of "PCIe 1 Disable_New" value in PCH Strap 4 from "false" to "true".
2. If the Value of "Root Port Configuration" has been changed to "2": 2x2 Port 1 (x2), Port 3 (x2):
   a. Require change of "PCIe 1 Disable_New" and "PCIe 3 Disable_New" in PCH Strap 4 from "false" to "true".
3. If the Value of "Root Port Configuration" has been changed to "3": 1x4 Port 1 (x4)):
   a. Require change of "PCIe 1 Disable_New" and "PCIe 2 Disable_New" as well as "PCIe 3 Disable__New" in PCH Strap 4 from "false" to "true".
4. PCIE Clock Req: For all the above settings, ensure to enable "PCIE_clkreq_enable_wake" Soft Strap.