Cherry Trail/Braswell Platform System Tools for Intel® Trusted Execution Engine and Intel® Integrated Sensor Solution Firmware

User Guide

March 2015

Revision 1.0

Intel Confidential

Contents

Figures

Tables

§

Revision History

Document Number	Revision	Description	Date
550215	0.5	Initial release	July 2014
550215	0.7	• Milestone release • Updated tool screenshot from latest took kit. • Removed Intel AT. • Updated command line option for FITc/FPT/TXEManuf/TXEInfo/FWUpdLcl section. • Added details on SVN usage and update command line example of manifest creation.	November 2014
550215	0.8	• Updated section 3.4.24 for FITc changes, ISS sensor calibration and configuration should be modified by PDT editor from now on. • Updated table18 to add –ISH option of TXEInfo tool • Updated calibration tool command line usage and examples • Updated Manifest Generation Tool section with ISS support for FLAMInGo.exe	December 2014
550215	1.0	• Added detailed description of –ISH option in Table 16. • Updated Section 2.5 Operating System Support Matrix. • Used consistent naming rule for Flash Descriptor Security Override strap to avoid customer confusion.	March 2015

§

1 Introduction

This document is intended to describe the tools that are used in the platform design, manufacturing, testing, and validation process.

1.1 Terminology

Acronym/Term	Definition
3PDS	3rd Party Data Storage
AC	Alternating Current
Agent	Software that runs on a client PC with OS running
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BBBS	BIOS Boot Block Size
BIN	Binary file
BIOS	Basic Input Output System
BIOS-FW	Basic Input Output System Firmware
BIST	Built In Self Test
CCM	Client Control Mode (Host Based Setup and Configuration)
CLI	Command Line Interface
CHY	Cherry Trail
CPU	Central Processing Unit
CRB	Customer Reference Board
DHCP	Dynamic Host Configuration Protocol
DIMM	Dual In-line Memory Module
DLL	Dynamic Link Library
DNS	Domain Naming System
EC	Embedded Controller
EEPROM	Electrically Erasable Programmable Read Only Memory
EFI	Extensible Firmware Interface
EHCI	Enhanced Host Controller Interface
EID	Endpoint ID
End User	The person who uses the computer (either Desktop or Mobile or Tablet). In corporate, the user usually does not have administrator privileges.
EOP	End Of Post

Acronym/Term	Definition
FCIM	Full Clock Integrated Mode
FCSS	Flex Clock Source Select
FDI	Flexible Display Interface
FITC	Flash Image Tool
FLOCKDN	Flash Configuration Lock-Down
FMBA	Flash Master Base Address
FOV	Fixed Offset Variable
FPT	Flash Programming Tool
FPTW	Flash Programming Tool Window
FQDN	Fully Qualified Domain Name
FRBA	Flash Region Base Address
FW	Firmware
FWUpdate	Firmware Update
G3	A system state of Mechanical Off where all power is disconnected from the system. A G3 power state does not necessarily indicate that RTC power is removed.
GPIO	General Purpose Input/Output
GUI	Graphical User Interface
GUID	Globally Unique Identifier
HECI (deprecated)	Host Embedded Controller Interface
Host or Host CPU	The processor running the operating system. This is different than the security engine controller running the Intel® TXE FW.
Host Service/ Application	An application running on the host CPU
HostIF	Host Interface
HTTP	HyperText Transfer Protocol
HW	Hardware
IBEN	Input Buffer Enable
IBV	Independent BIOS Vendor
ID	Identification
IDER	Integrated Drive Electronics Redirection
INF	An information file (.inf) used by Microsoft operating systems that support the Plug & Play feature. When installing a driver, this file provides the OS with the necessary information about driver filenames, driver components, and supported hardware.
Intel® TXE	Intel® Trusted Execution Engine. The embedded processor residing in the Silicon.

Acronym/Term	Definition
Intel® ISS	Intel® Integrated Sensor Solution. In the Silicon as well.
PDT	Platfomr Descriptor Table
Intel® AT	Intel® Anti-Theft Technology
Intel® DAL	Intel® Dynamic Application Loader (Intel® DAL)
Intel® TXEI	Intel® Trusted Execution Environment Interface
Intel® TXEI driver	Intel® TXE host driver that runs on the host and interfaces between ISV Agent and the Intel® TXE HW.
Intel TXEINFO	Intel® TXE Setting Checker Tool
Intel TXEInfoWin	Windows* version of Intel TXEINFO
Intel TXEManuf	Intel TXEManuf validates Intel® TXE functionality on the manufacturing line
TXEManufWin	Windows* version of Intel TXEManuf
FWUPDLCL	Firmware Update Local Tool
ISV	Independent Software Vendor
IT User	Information Technology User. Typically very technical and uses a management console to ensure multiple PCs on a network function.
JEDECID	Joint Electronic Device Engineering Councils ID. Standard Manufacturer's Identification Code that is assigned, maintained and updated by the JEDEC office
JTAG	Joint Test Action Group
KVM	Keyboard, Video, Mouse
LAN	Local Area Network
LED	Light Emitting Diode
LMS	Local Management Service. An SW application which runs on the host machine and provides a secured communication between the ISV agent and the Intel® Management Engine Firmware.
LPC	Low Pin Count Bus
M0	Intel® TXE power state where all HW power planes are activated. Host power state is S0.
M-Off	No power is applied to the security engine processor subsystem. Intel® TXE is shut down.
MAC address	Media Access Control address
NM	Number of Masters
NVAR	Named Variable
NVM	Non-Volatile Memory
NVRAM	Non-Volatile Random Access Memory
OCKEN	Output Clock Enable
ODM	Original Device Manufacturer

Acronym/Term	Definition
OEM	Original Equipment Manufacturer
OEM ID	Original Equipment Manufacturer Identification
OOB	Out Of Band
OOB interface.	Out Of Band interface. An SOAP/XML interface over secure or non-secure TCP protocol.
OS	Operating System
OS Hibernate	OS state where the OS state is saved on the hard drive.
OS not Functional	The Host OS is considered non-functional in Sx power state in any one of the following cases when the system is in S0 power state: OS is hung After PCI reset OS watch dog expires OS is not present
OVR	Override
PAVP	Protected Video and Audio Path
PC	Personal Computer
PCI	Peripheral Component Interconnect
PCIe*	Peripheral Component Interconnect Express
PDR	Platform Descriptor Region
PHY	Physical Layer
PID	Provisioning ID
PKI	Public Key Infrastructure
PM	Power Management
PRTC	Protected Real Time Clock
PSK	Pre-Shared Key
RCS	Remote Connectivity Service
RCFG	Remote Configuration
RNG	Random Number Generator
ROM	Read Only Memory
RPAS	Remote Connectivity Service
RSA	A public key encryption method
RTC	Real Time Clock
S0	A system state where power is applied to all HW devices and the system is running normally.
S1, S2, S3	A system state where the host CPU is not running but power is connected to the memory system (memory is in self refresh).
S4	A system state, where the host CPU and memory are not active.

Acronym/Term	Definition
S5	A system state where all power to the host system is off but the power cord is still connected.
SDK	Software Development Kit
SEBP	Single Ended Buffer Parameters
SHA	Secure Hash Algorithm
SMB	Small Medium Business mode
SMBus	System Management Bus
Snooze mode	Intel® TXE activities are mostly suspended to save power. Intel® TXE monitors HW activities and can restore its activities depending on the HW event.
SOAP	Simple Object Access Protocol
SOL	Serial over LAN
SPI	Serial Peripheral Interface
SPI Flash	Serial Peripheral Interface Flash
Standby	OS state where the OS state is saved in memory and resumed from the memory when the mouse/keyboard is clicked.
Sx	All S states which are different than S0
SW	Software
System States	Operating System power states such as S0, S1, S2, S3, S4, and S5.
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
UI	User Interface
UIM	User Identifiable Mark
UMA	Unified Memory Access
Un-configured state	The state of the Intel® TXE FW when it leaves the OEM factory. At this stage the Intel® TXE FW is not functional and must be configured.
UNS	User Notification Services
UPDPARAM	Update Parameter Tool
USB	Universal Serial Bus
USBr	Universal Serial Bus Redirection
UUID	Universally Unique IDentifier
VE	Virtualization Engine
VLAN	Virtual Local Area Network
VSCC	Vendor Specific Component Capabilities
Windows* PE	Windows* Preinstallation Environment
WIP	Work in Progress
WLAN	Wireless Local Area Network

Acronym/Term	Definition
XML	Extensible Markup Language. Intel® AMT's XML-based protocol has 3 parts: An envelope that defines a framework for describing what is in a message and how to process it A set of encoding rules for expressing instances of application-defined data types A convention for representing remote procedure calls and responses
ZTC	Zero Touch Configuration
IPC	Inter-Process Communication, which is hardware block used for communication between SeC and the host.
FPF	Field Programmable Fuses
IBB	Initial Boot Block
SB	Secure Boot
ADB	Android* Debug Bridge

1.2 Reference Documents

Document	Document No./Location
Cherry Trail FW Bring Up Guide	FW kit
Cherry Trail SoC External Design Specification (EDS) (Volume 1 of 1) Cherry Trail SoC External Design Specification (EDS) (Volume 2 of 2)	539071 543698
Cherry Trail SoC SPI Programming Guide – Design Guide	FW kit 540557
Cherry Trail SoC - BIOS Writers Guide (Volume 1 of 2) Cherry Trail SoC - BIOS Writers Guide (Volume 2 of 2)	540142 540143
Cherry Trail and Braswell SoC Intel® Trusted Execution Engine (Intel® TXE) - BIOS Writer's Guide	545425
Cherry Trail (MSP-T4) Platform - Design Guide Cherry Trail (VMS-T3) Platform - Design Guide	537901 540558

§

2 Preface

This document covers the system tools used for creating, modifying, and writing binary image files, manufacturing testing, Intel® Trusted Execution Engine (Intel® TXE) setting information gathering, and Intel® TXE FW updating.

The tools are located in **Kit directory\Tools**. For information on other tools, refer to the corresponding tool user guide in the other directories in the FW release.

The system tools described in this document are platform specific in the following ways:

- Cherry Trail/Braswell Platform – All tools in the Cherry Trail/Braswell FW release kit are designed for Cherry Trail/Braswell platforms only. These tools do not work properly on any other legacy platforms (Cedar Trail, Oak Trail) and previous SoC based platform (Medfield, Clover Trail and Bay Trail). Tools designed for other platforms also do not work properly on the Cherry Trail platform.
- Intel® TXE Firmware SKU – A common set of tools are provided for the following
  - Intel® TXE FW SKUs: Tablet, Entry Level Netbook and Desktop SKU.

Note: The system tools including FITc, FPT, TXEinfo and TXEManuf should be used for SPI image development and manufacturing purpose only. OEMs and ODMs who using those tools need ensure follow all legal agreements and never release or expose those tools to end user. Contact local Intel field representative, if you have any further question regarding legal or license agreement.

2.1 Intel® TXE System Tools Changes

Intel developed the following system tools enhancements for Intel® TXE platforms:

- Firmware status of each tool changes from Intel® TXE.
- FITC support ISS firmware and sensor SKU/Interrupt/Calibration configuration.
- One image for both FITC and FW update.
- FPT Support ISS manufacturing line configuration setting including Active SSKU, ISS enable/disable and calibration file flashing.
- TXEInfo support ISS FW version, status and setting display.
- Intel TXEMANUF option changes, supporting –ISS option for ISS FW aliveness tests, sensor calibration, sensor connectivity and Sensor BIST.
- Add Multiple Operation System support for broader enablement.

Note: More details are available in the corresponding tool documentation.

2.2 Image Editing Tools

The following tools create and write flash images:

- FITC:

  Combines the Descriptor, BIOS, PDR, Intel® TXE and ISS FW binaries into one image

  Configures softstraps and NVARs for Intel® TXE and ISS FW settings that can be programmed by a flash programming device or the FPT Tool.

- FPT:

  Programs the flash memory of individual regions or the entire flash device

  Modifies some Intel® TXE and ISS settings (FOV) after Intel® TXE and ISS is flashed on the flash part.

  Programming FPF file, Widevine keybox and sensor calibration data.

- FWUpdate:

  Updates the Intel® TXE, ISS FW code region and calibration file on a flash device that has already been programmed with a complete SPI image

Note: The firmware update tool provided by Intel only works on the platforms that support this feature.

2.3 Manufacturing Line Validation Tools

The manufacturing line validation tools (Intel TXEMANUF) allow the Intel® TXE functionality to be tested immediately after the SoC is generated. These tools are designed to be able to run quickly. They can run on simple operating systems, such as UEFI shell. The Windows* versions are written to run on Windows* 8.1 and Windows* PE. Android* version required to be pushed onto device via ADB interface. These tools are mostly run on the manufacturing line to do manufacturing testing.

2.4 Intel® TXE Setting Checker Tool

The Intel® TXE setting checker tool (Intel TXEINFO) retrieves and displays information about some of the Intel® TXE/ISS FW settings, the Intel® TXE/ISS FW version, and the FW capability on the platform. Widevince keybox provision status and sensor status also be retrieved from this tool.

2.5 Operating System Support

Table 1. OS Support for Tools

Intel® TXE and Manufacturing Tools	UEFI Shell 32/64 bit	Windows* PE 3.x/4.0/5.x 32/64 bit (Based on Win* 8.1)	Windows* 8.1 32/64 bit	Windows* 7 32/64 bit	Android* 32/64bit (Kitkat/ Lollipop)
FITC			x	x	
FPT	x	x	x		x
FWUPDATE	x	x	x		x
FWUPDATE UEFI library	x				
TXEMANUF	x	x	x		x
TXEINFO	x	x	x		x
Manifest Generation Tool			x	x	
Calibration Tool			x		x

NOTES:

1. Android* based tools have to be pushed onto device via ADB and set to executable before running.
2. Maximum EFI environment variable size supported by BIOS and TXE manufacturing tools is 136KB (0x22000). Anything beyond this specification is neither supported nor validated. It is recommended to stay below this limit to avoid unexpected EFI and manufacturing tools behavior.
3. Customer should use OS version for each tools. In other word, 32 bits tool should not be used on 64 bits OS as it's not POR and vice versa.

2.6 Generic System Requirements

The installation of the following services is required by integration validation tools that run locally on the system under test with the Intel® Trusted Execution Engine:

- Intel® TXEI driver

See the description of each tool for exact requirements.

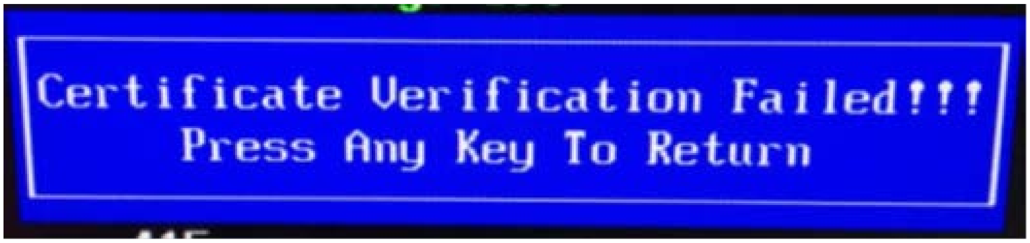
Table 2. Tools Summary

Tool Name	Feature Tested	Runs on Intel® TXE Device
TXEManuf	Connectivity between Intel® TXE Devices	X
TXEInfo	Firmware Aliveness – outputs certain Intel® TXE parameters	X
FPT	Programs the image onto the flash memory	X
FWUpdate	Updates the FW code while maintaining the previously set values	X

2.6.1 UEFI Tools

EFI system tools will not run without disabling "UEFI Security Boot" in the BIOS setup menu. If you attempting to do so result in the error "Certificate Verification Failed" error message. Otherwise OEM has to sign any EFI utility with OEM key which is not recommended for manufacturing tool segment for security concern.

Figure 1. Certificate Verification Failed Error



Certificate Verification Failed!!!
Press Any Key To Return

2.6.2 Manufacturing Tools Requirement

There are some manufacturing requirements Intel strongly recommended our customer have to follow to prevent unexpected failure returned from Intel Manufacturing Tool kit and test case.

- On the manufacturing flow, SPI write protection should only be enabled after all TXE manufacturing flow has done including FOV update, TXEManuf, FPF programming and FPT –closemnf.
- In repair flow, SPI write protection should be able to be temporarily disabled during the operation of TXE repair, testing, refurbish.

2.6.3 Android* Tools

In order to use Intel TXE System tools locally on SUT (System Under Test), you must push the tools using ADB (Android* Debug Bridge) to a directory and set to executable that can be accessed using Terminal Emulator or using ADB itself.

Obtain ADB and fastboot tool from Intel VIP or MCP (comes as part of the Phone Flash Tool installation, which is not part of the Intel TXE FW Kit currently). There are Linux* and Windows* versions of the tool. Usage of this tool will be the same with respect to Intel TXE System tools.

Command line example to ADB push tool to device and change permission in AOS:

ADB Push:

```
C:\adb push C:\CHT_FW_KIT\Flash_Programming_Tool\Android\TXEInfo
/data/local
```

Change permission and command line example:

```
C:\adb shell chmod 777 /data/local/TXEInfo
C:\adb shell TXEInfo -verbose
```

2.7 Error Return

Tools always return 0/1 for the error level (0 = success, 1= error). A detail error code is displayed on the screen and stored on an error.log file in the same directory as the tools. (See Tool Detail Error Codes for a list of these error codes.)

2.8 Usage of the Double-Quote Character (")

The EFI version of the tools handle multi-word argument is different than the Windows* version. If there is a single argument that consists of multiple words delimitated by spaces, the argument needs to be entered as following:

FPT.exe –r “^” this is an example”^”.

The command shell used to invoke the tools in EFI and Windows* has a built-in CLI.

The command shell was intended to be used for invoking applications as well as running in batch mode and performing basic system and file operations. For this reason, the CLI has special characters that perform additional processing upon command.

The double-quote is the only character which needs special consideration as input. The various quoting mechanisms are the backslash escape character (/), single-quotes ('), and double-quotes ("). A common issue encountered with this is the need to have a double-quote as part of the input string rather than using a double-quote to define the beginning and end of a string with spaces.

For example, the user may want these words – one two – to be entered as a single string for a vector instead of dividing it into two strings ("one", "two"). In that case, the entry – including the space between the words – must begin and end with double-quotes ("one two") to define this as a single string.

When double-quotes are used in this way in the CLI, they define the string to be passed to a vector, but are NOT included as part of the vector. The issue encountered with this is how to have the double-quote character included as part of the vector as well as bypassed during the initial processing of the string by the CLI. This can be resolved by preceding the double-quote character with a backslash (\").

For example, if the user wants these words to be input – input"string – the command line is: input\"string.

2.9 PMX Driver Limitation

Several Windows* tools (Intel TXEINFO, Intel TXEMANUF, and FPT) use the PMX library to get access to the PCI device. Only one tool can get access to the PMX library at a time because of library limitation. Therefore, running multiple tools to get access to PMX library will result in an error (failure to load driver).

The PMX driver is not designed to work with the latest Windows* driver model (it does not conform to the new driver's API architecture).

In Windows* 7 and higher, the verifier sits in kernel mode, performing continual checks or making calls to selected driver APIs with simulations of well-known driver related issues.

Warning: Running the PMX driver with the Windows* 7 and higher driver verifier turned on causes the OS to crash. Do not include PMX as part of the verifier driver list if the user is running Windows* 7 and higher with the driver verifier turned on.

§

3 Intel® Flash Image Tool

The Flash Image tool (**FITC.exe**) creates and configures a complete SPI image file for Cherry Trail platforms in the following way:

1. FITC creates and allows configuration of the Flash Descriptor Region, which contains configuration information for platform hardware and FW.
2. FITC assembles the following into a single SPI flash image:

   Binary files of the following regions:
   - BIOS
   - Intel® TXE
   - Intel® ISS
   - Platform Descriptor Region (PDR)
   - The Flash Descriptor Region created by FITC
3. The user can manipulate the completed SPI image via a GUI and change the various SoC parameters to match the target hardware. Various configurations can be saved to independent files, so the user does not have to recreate a new image each time.

FITC supports a set of command line parameters that can be used to build an image from the CLI or from a makefile. When a previously stored configuration is used to define the image layout, the user does not have to interact with the GUI.

Note: FITC just generates a complete SPI image file; it does not program the flash device. This complete SPI image must be programmed into the flash with FPT, any third-party flash burning tool, or some other flash burner device.

3.1 System Requirements

FITC runs on Windows* 7 and Windows* 8/8.1. The tool does not have to run on an Intel® TXE-enabled system with Cherry Trail SoC mounted.

3.2 Flash Image Details

A flash image is composed of four regions. The locations of these regions are referred to in terms of where they can be found within the total memory of the flash.

Figure 2. SPI Flash Image Regions

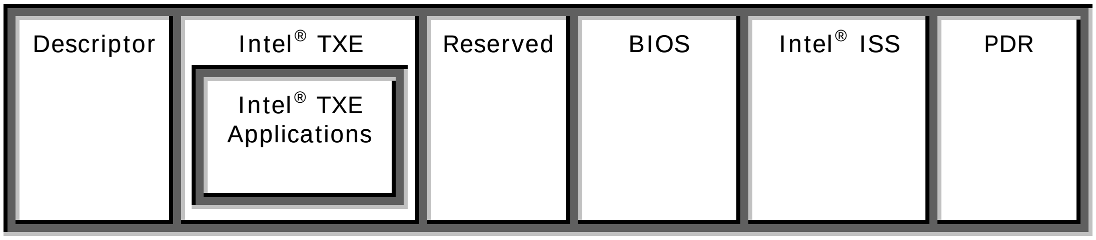


Table 3. Flash Image Regions – Description

Region	Description
Descriptor	This region contains information such as the space allocated for each region of the flash image, read-write permissions for each region, and a space which can be used for vendor-specific data. It takes up a fixed amount of space at the beginning of the flash memory. **NOTE:** This region MUST be locked before the serial flash device is shipped to end users. See section 3.4.11 below for more information. Failure to lock the Descriptor Region leaves the Intel® TXE device vulnerable to security attacks.
Intel® TXE	This region contains code and configuration data for Intel® TXE applications, such as Intel® PTT. It takes up a variable amount of space at the end of the Descriptor. Intel ISS configuration data is located in the Intel TXE data region (NVAR)
Intel® ISS	Contains code for Intel® ISS firmware and OEM ISS code.
Reserved	This region is reserved for future use.
BIOS	This region contains code and configuration data for the entire computer.
PDR	This region lets system manufacturers describe custom features for the platform. (this region is optional for cost reduction program which using 2MB or 4MB SPI flash as main storage)

3.2.1 Flash Space Allocation

Space allocation for each region is determined as follows:

1. Each region can be assigned a fixed amount of space. If a region is not assigned a fixed amount of space, it occupies only as much space as it requires.
2. If there is still space left in the flash after allocating space to all of the regions, the Intel® TXE/ISS region expands to fill the remaining space.
3. If there is leftover space and Intel® TXE/ISS region is not implemented, the BIOS region expands to occupy the remaining space.
4. If only the Descriptor region is implemented, it expands to occupy the entire flash.

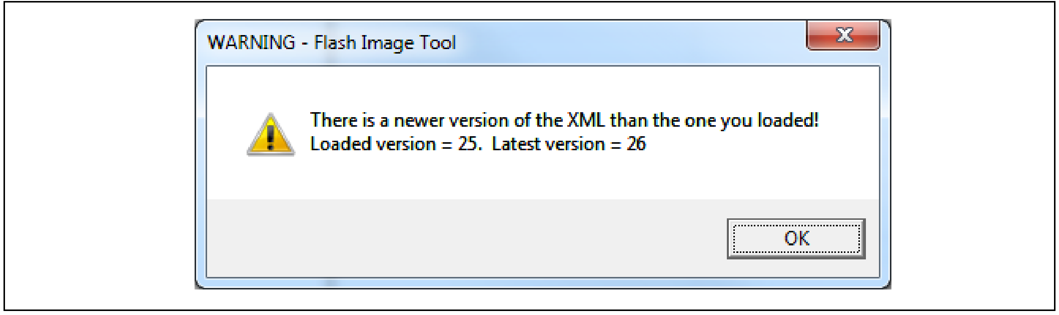
3.3 Required Files

The FITC main executable is **fitc.exe**. The following files must be in the same directory as **fitc.exe**:

- fitctmpl.xml
- newfiletmpl.xml
- vsccommn.bin
- fitc.ini

FITC does not run correctly if any of the .xml and .bin files listed above are missing. FITC creates a blank **fitc.ini** file if there is no **fitc.ini** file in the folder.

Note: When using a ‘Newfiletmp.xml’ from previous kit releases FITc will display a message to the user that the file being used is older than the version FITc expecting (See the following example).



There is a newer version of the XML than the one you loaded!
Loaded version = 25. Latest version = 26

After the user selects the **OK** radio button, FITc automatically updates the ‘Newfiletmp.xml’ with any missing / new or changed variables and pre-populates those variables with the firmware defaults. Once this is completed the user can then re-save this new ‘Newfiletmp.xml’ back to retain the updates made by FITc.

3.4 FITC

See the following for further information:

- General configuration information – See the FW Bring Up Guide from the appropriate Intel® TXE FW kit.
- Detailed information on how to configure SoC Soft Straps and VSCC information – See the Cherry Trail SoC SPI programming guide from the appropriate Intel® TXE FW kit.

3.4.1 Configuration Files

The flash image can be configured in many different ways, depending on the target hardware and the required FW options. FITC lets the user change this configuration in a graphical manner (via the GUI). Each configuration can be saved to an XML file. These XML files can be loaded at a later time and used to build subsequent flash images. Note that the newfiletmpl.xml under FITC folder is just a template which should not be loaded without any modifications.

3.4.2 Creating New Configuration

FITC provides a default configuration file that the user can use to build a new image. This default configuration file can be loaded by clicking **File** > **New**.

3.4.3 Opening an Existing Configuration

To open an existing configuration file:

1. Go to **File > Open**. The **Open File** dialog appears.
2. Select the XML file to load.
3. Click **Open**.

Note: The user can also open a file by dragging and dropping a configuration file into the main window of the application.

3.4.4 Saving a Configuration

To save the current configuration in an XML file:

1. Go to **File > Save** or **> Save As**. The Save File dialog appears if the configuration has not been given a name or if **File > Save As** was chosen.
2. Select the path and enter the file name for the configuration.
3. Click **Save**.

3.4.5 Environment Variables

A set of environment variables is provided to make the image configuration files more portable. The configuration is not tied to a particular root directory structure because all of the paths in the configuration are relative to environment variables. The user can set the environment variables appropriate for the platform being used, or override the variables with command line options.

It is recommended that the environment variables be the first thing that the user sets when working with a new configuration. This ensures that FITC can properly substitute environment variables into paths to keep them relative. Doing this also speeds up configuration because many of the **Open File** dialogs default to particular environment variable paths.

To modify the environment variables:

1. Go to **Build > Environment Variables**. A dialog appears displaying the current working directory on top, followed by the current values of all the environment variables:

   $CurWorkDir – the current FITc working directory.

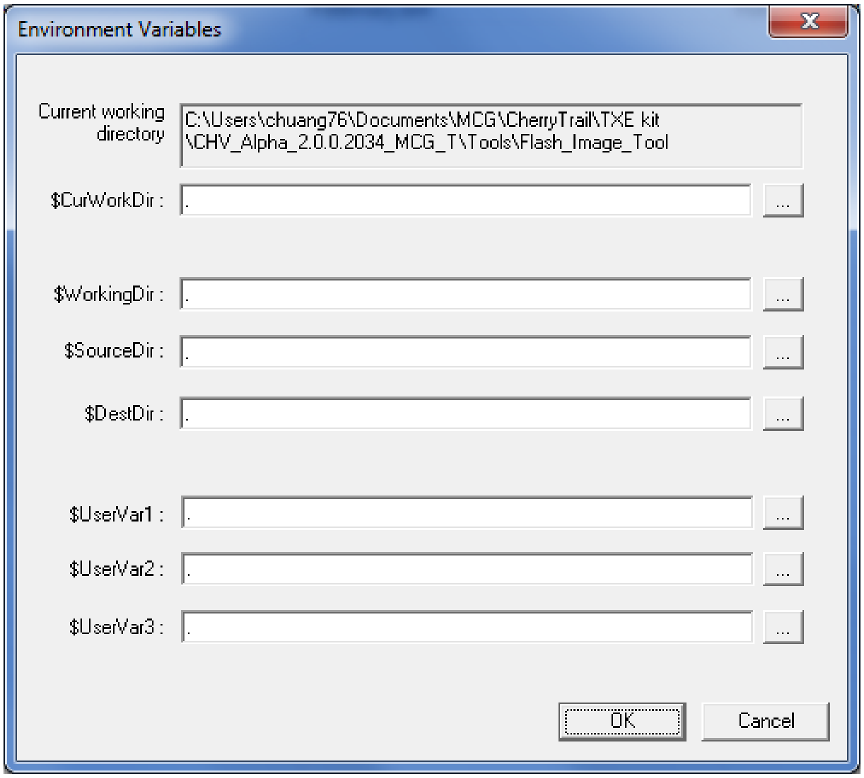
   $WorkingDir – the directory where the log file is kept and where the components of an image are stored when an image is decomposed.

   $SourceDir – the directory that contains the base image binary files from which a complete flash image is prepared. Usually these base image binary files are obtained from Intel® VIP on the Web, a BIOS programming resource, or another source.

   $DestDir – the directory in which the final combined image is saved, as well as all intermediate files generated during the build.

   $UserVar1-3 – used when the above variables are not populated.

Figure 3. Environment Variables Dialog



Current working directory C:\Users\chuang76\Documents\MCG\CherryTrail\TXE kit \CHV_Alpha_2.0.0.2034_MCG_T\Tools\Flash_Image_Tool

$CurWorkDir : .

$WorkingDir : .

$SourceDir : .

$DestDir : .

$UserVar1 : .

$UserVar2 : .

$UserVar3 : .

2. Click ... button next to an environment variable and select the directory where that variable's files will be stored; the name and relative path of that directory appears in the field next to the variable's name.
3. Repeat Step 2 until the directories of all relevant environment variables are defined.
4. Click **OK**.

Note: The environment variables are saved in the application's INI file, not the XML configuration file. This allows the configuration files to be portable across different computers and directory structures.

3.4.6 Build Settings

FITC lets the user set several options that control how the image is built. The options that can be modified are described in Table 4.

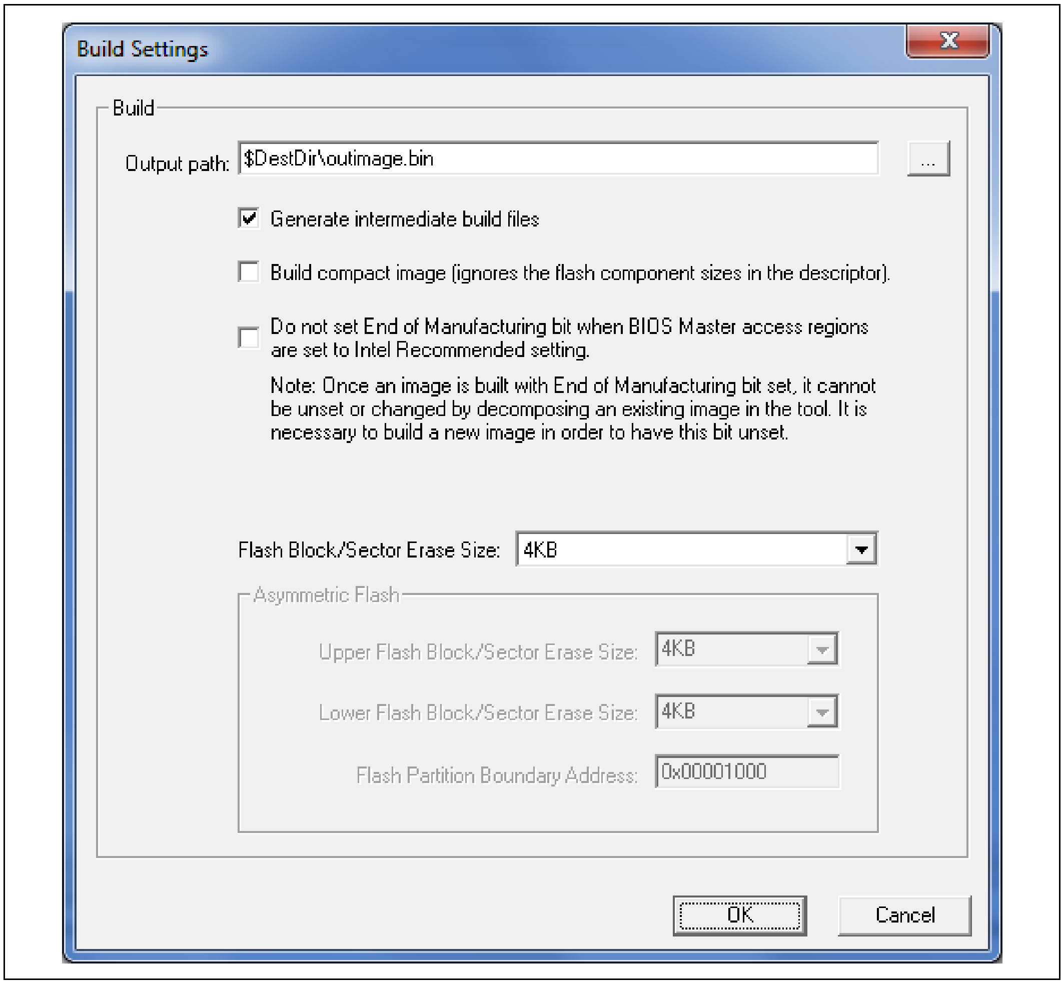
To modify the build setting:

1. Go to **Build** > **Build Settings**. A dialog box appears showing the current build settings.
2. Modify the relevant settings in the **Build Settings** dialog.
3. Click **OK**. The modified build settings are saved in the XML configuration file.

Table 4. Build Settings Dialog Options

Option	Description
Output path	The path and filename where the final image should be saved after it is built. **NOTE:** Using the $DestDir environment variable makes the configuration more portable.)
Generate intermediate build files	Causes the application to generate separate (intermediate) binary files for each region, in addition to the final image file (see Figure 3). These files are located in the specified output folder's INT subfolder. These image files can be programmed individually with the FPT.
Build Compact Image	Creates the smallest flash image possible. (By default, the application uses the flash component sizes in the Descriptor to determine the image length.)
Do not set End of Manufacturing bit …	When descriptor permissions are set to production values, do not select the **Do not set End of Manufacturing bit** box unless not closing End of Manufacturing is explicitly desired. Intel strongly recommends that the Global Lock Bit/End of Manufacturing bit be set on all production platforms.
Flash Block/Sector Erase Size	All regions in the flash conform to the **4KB sector erase size**. It is critical that this option is set correctly to ensure that the flash regions can be properly updated at runtime.
Asymmetric Flash	Allows the user specify a different sector erase size for the upper and lower flash block. **Only 4KB erase is supported for Intel® TXE FW**. This option also lets user modify the flash partition boundary address.

Figure 4. Build Settings Dialog



Build Settings

Build

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

Asymmetric Flash

Upper Flash Block/Sector Erase Size: 4KB

Lower Flash Block/Sector Erase Size: 4KB

Flash Partition Boundary Address: 0x00001000

End of manufacturing bit is simply a byte in the image. This is not an NVAR, or FOV. In previous generation, when creating an image, the user can set the Intel® TXE manufacturing done bit (Global Lock bit) automatically based on BIOS being set to production Master Access section. However, in order to allow some customers not to set it, we show this checkbox. This checkbox only does something if:

- Intel® TXE manufacturing done bit is not set, BIOS is not set to production → FITc will not set Intel® TXE manufacturing done bit – independent of this checkbox
- Intel® TXE manufacturing done bit is not set, BIOS is set to production, checkbox is unchecked → FITc will set Intel® TXE manufacturing done bit
- Intel® TXE manufacturing done bit is not set, BIOS is set to production, checkbox is checked → FITc will not set Intel® TXE manufacturing done bit
- Intel® TXE manufacturing done bit set → will stay set

A dumped image is never reflected in this checkbox – it does not show the actual value of Intel® TXE manufacturing done bit. It shows what should be done in the next build. But if Intel® TXE manufacturing done bit is set, this checkbox will never uncheck it.

3.4.7 Selecting Platform SKU

The ability to select Platform SKU allows the user configure "Full Featured Engineering samples" to test how the firmware behaves like the production Intel® Atom™ Processor, with the following reservations:

- Certain features only work with particular SoC SKUs and FW kits.
- SKU Manager Selection has no effect on the Production SoC chip.

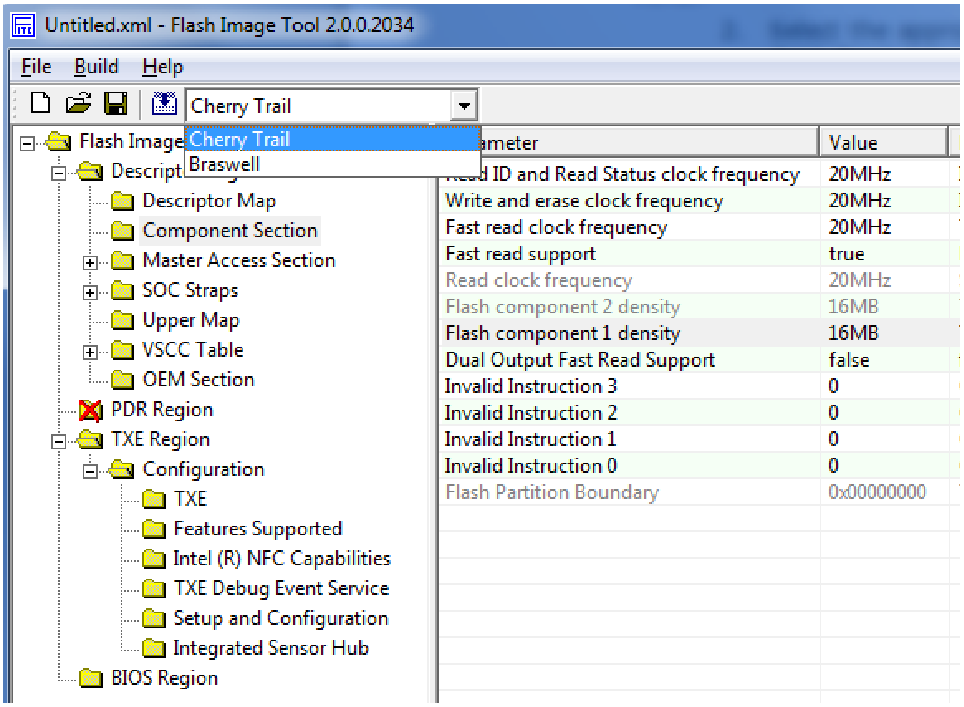
To select a Platform SKU:

1. Load the Intel® TXE region.

Note: Loading the Intel® TXE region first ensures that the proper FW settings are loaded into FITC.

2. Select the appropriate platform type for the specific SoC from the SKU Manager drop-down list.

Figure 5. Selected SKU Platform in FITC



ameter	Value
d ID and Read Status clock frequency	20MHz
Write and erase clock frequency	20MHz
Fast read clock frequency	20MHz
Fast read support	true
Read clock frequency	20MHz
Flash component 2 density	16MB
Flash component 1 density	16MB
Dual Output Fast Read Support	false
Invalid Instruction 3	0
Invalid Instruction 2	0
Invalid Instruction 1	0
Invalid Instruction 0	0
Flash Partition Boundary	0x00000000

3.4.8 Modifying Flash Descriptor Region

The FDR contains information about the flash image and the target hardware. This region contains the read/write values. It is important for this region to be configured correctly or the target computer may not function as expected. This region also needs to be configured correctly in order to ensure that the system is secure.

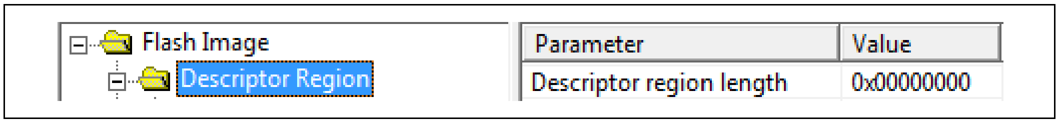
3.4.9 Descriptor Region Length

The Descriptor Region Length parameter sets the size of the Descriptor region.

To set the value of the Descriptor Region Length parameter:

1. Select **Descriptor Region** in the left pane. The **Descriptor Region Length** parameter appears in the right pane.
2. Double-click the **Descriptor Region Length** parameter. The **Descriptor Region Length** dialog appears.
3. Enter any non-zero value into the dialog to set the length of the region and click **OK**.

Figure 6. Descriptor Region Length Parameter



- Flash Image
  - Descriptor Region

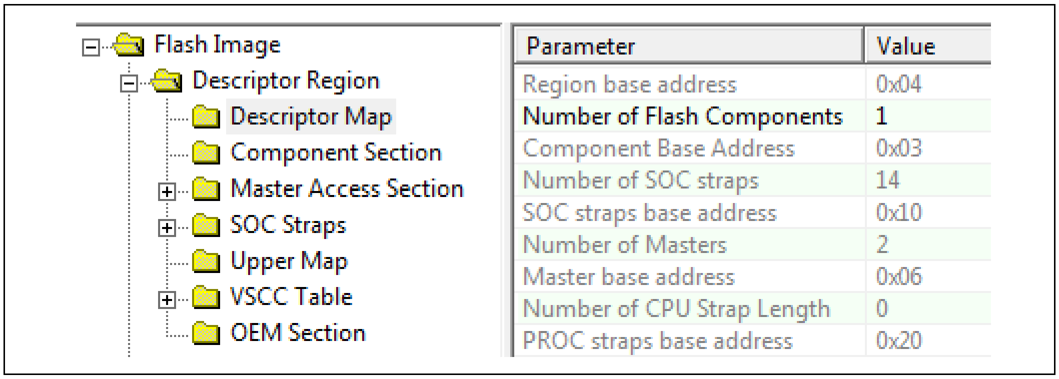
Parameter	Value
Descriptor region length	0x00000000

3.4.10 Setting Number and Size of Flash Components

To set the number of flash components:

1. Expand the **Descriptor Region** node of the tree in the left pane.
2. Select **Descriptor Map** (see Figure 7). All the parameters in the Descriptor Map section are listed in the right pane.

Figure 7. Descriptor Region > Descriptor Map Parameters

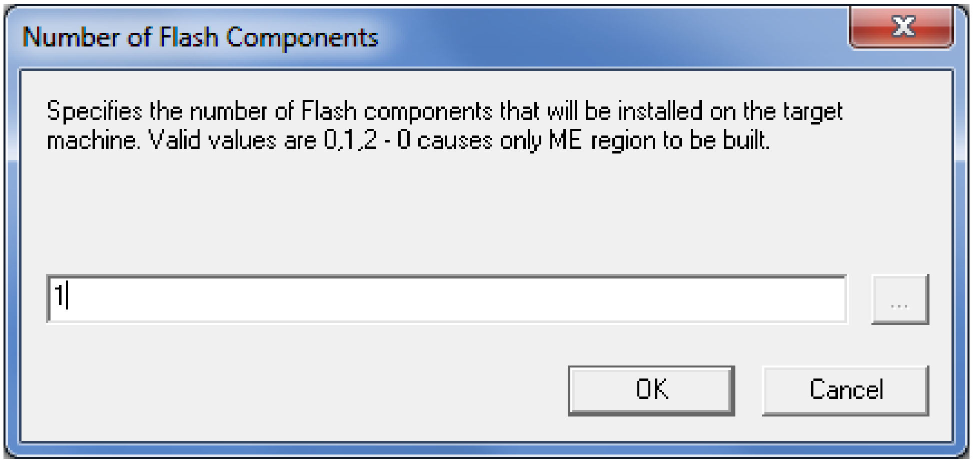


- Flash Image
  - Descriptor Region
    - Descriptor Map
    - Component Section
    - Master Access Section
    - SOC Straps
    - Upper Map
    - VSCC Table
    - OEM Section

Parameter	Value
Region base address	0x04
Number of Flash Components	1
Component Base Address	0x03
Number of SOC straps	14
SOC straps base address	0x10
Number of Masters	2
Master base address	0x06
Number of CPU Strap Length	0
PROC straps base address	0x20

3. Double-click **Number of Flash Components** in the right pane (see Figure 8). The Flash Components dialog appears.
4. Enter the number of flash components (valid values are 0, 1 or 2).
5. Click **OK**. The parameter is updated.

Figure 8. Flash Components Dialog



Number of Flash Components

Specifies the number of Flash components that will be installed on the target machine. Valid values are 0,1,2 - 0 causes only ME region to be built.

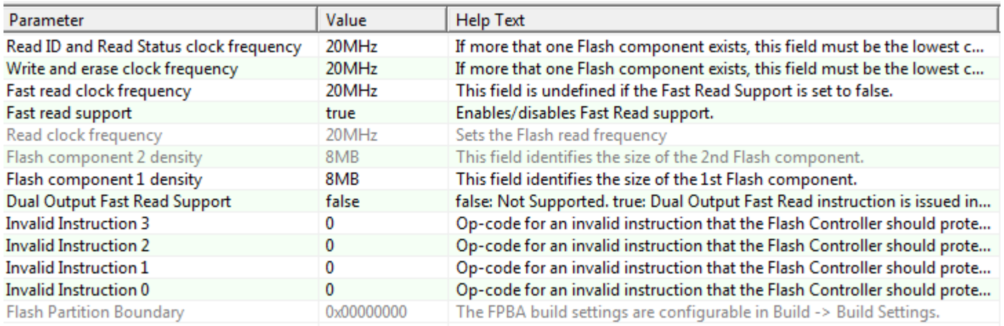
1

To set the size of each flash component:

1. Expand **Descriptor Region** node in the left pane and select **Component Section**. The Component Section parameters appear in the right pane.
2. In the **Flash component 1 density** and **Flash component 2 density** parameters specify the size of each flash component.
3. Double-click one of these parameters. A dialog appears.
4. Select the correct component size from the dialog's drop-down list and click **OK**. The parameter is updated.
5. Repeat steps 2-3 for the other parameter.

Note: The size of the second flash component is only editable if the number of flash components is set to 2.

Figure 9. Descriptor Region > Component Section Parameters



Parameter	Value	Help Text
Read ID and Read Status clock frequency	20MHz	If more that one Flash component exists, this field must be the lowest c...
Write and erase clock frequency	20MHz	If more that one Flash component exists, this field must be the lowest c...
Fast read clock frequency	20MHz	This field is undefined if the Fast Read Support is set to false.
Fast read support	true	Enables/disables Fast Read support.
Read clock frequency	20MHz	Sets the Flash read frequency
Flash component 2 density	8MB	This field identifies the size of the 2nd Flash component.
Flash component 1 density	8MB	This field identifies the size of the 1st Flash component.
Dual Output Fast Read Support	false	false: Not Supported. true: Dual Output Fast Read instruction is issued in...
Invalid Instruction 3	0	Op-code for an invalid instruction that the Flash Controller should prote...
Invalid Instruction 2	0	Op-code for an invalid instruction that the Flash Controller should prote...
Invalid Instruction 1	0	Op-code for an invalid instruction that the Flash Controller should prote...
Invalid Instruction 0	0	Op-code for an invalid instruction that the Flash Controller should prote...
Flash Partition Boundary	0x00000000	The FPBA build settings are configurable in Build -> Build Settings.

3.4.11 Region Access Control

Regions of the flash can be protected from read or write access by setting a protection parameter in the Descriptor Region. The Descriptor Region must be locked before Intel® TXE devices are shipped. If the Descriptor Region is not locked, the Intel® TXE device is vulnerable to security attacks. The level of read/write access provided is at the discretion of the OEM/ODM. A cross-reference of access settings is shown below.

Table 5. Region Access Control Table

		Regions that can be accessed			
		PDR	**Intel® TXE**	**BIOS**	**Descriptor**
Region to Grant Access	**Intel® TXE**	None/Read/Write	Intel® TXE can always read from and write to Intel® TXE Region	None/Read/Write	None/Read/Write
	BIOS	None/Read/Write	None/Read/Write	BIOS can always read from and write to BIOS Region	None/Read/Write

There are three parameters in the Descriptor that specify access for each chipset. The bit structure of these parameters is shown in the following table:

Key:

- 0 – Denied access
- 1 – Allowed access
- NC – bit may be either 0 or 1 since it is unused.

Table 6. CPU/BIOS Access

Read Access								
	Unused			**PDR**	**Severed**	**Intel® TXE**	**BIOS**	**Desc**
Bit Number	7	6	5	4	3	2	1	0
Bit Value	X	X	X	0/1	0/1	0/1	NC	0/1

Write Access								
	Unused			**PDR**	**Severed**	**Intel® TXE**	**BIOS**	**Desc**
Bit Number	7	6	5	4	3	2	1	0
Bit Value	X	X	X	0/1	0/1	0/1	NC	0/1

Example:

If the CPU/BIOS needs read access to the Intel® TXE and write access to Intel® TXE, then the bits are set to:

- Read Access – 0b 0000 1110 (0x 0E in hexadecimal)
- Write Access – 0b 0000 0110 (0x 06 in hexadecimal)

To set these access values in FITC:

1. Select **Descriptor Region > Master Access > CPU/BIOS** in the left pane. The access parameters are listed in the right pane (see Figure 10).
2. Double-click on each parameter and set its access value in one of the following ways:

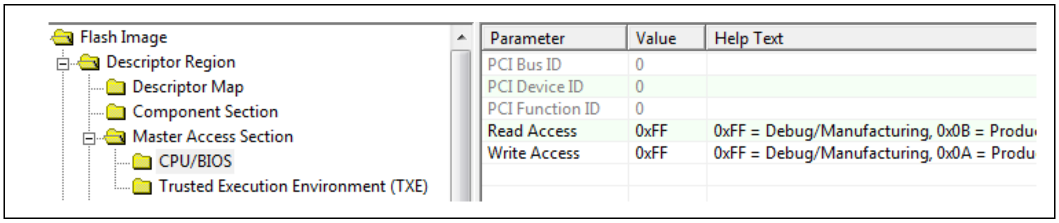
- To generate an image for debug purposes or to leave the SPI region open: Select 0xFF for both read and write access in all three sections.
- To generate a production image with BIOS access to the PDR region: Select read access 0x0B and write access 0x0A.

Note: These settings should only be used if the PDR region is implemented.

- To lock the SPI in the image creation phase: Select the recommended setting for production (e.g., select 0x0D for Intel® TXE read access and 0x0C for Intel® TXE write access).

Note: If all Read/Write Master access settings for Intel® TXE are set to production platform values, then the Intel® TXE manufacturing mode done (Global Lock) bit is automatically set. If the Intel® TXE manufacturing mode done (Global Lock) bit is set, the FOV mechanism is not available.

Figure 10. Descriptor Region > Master Access Section



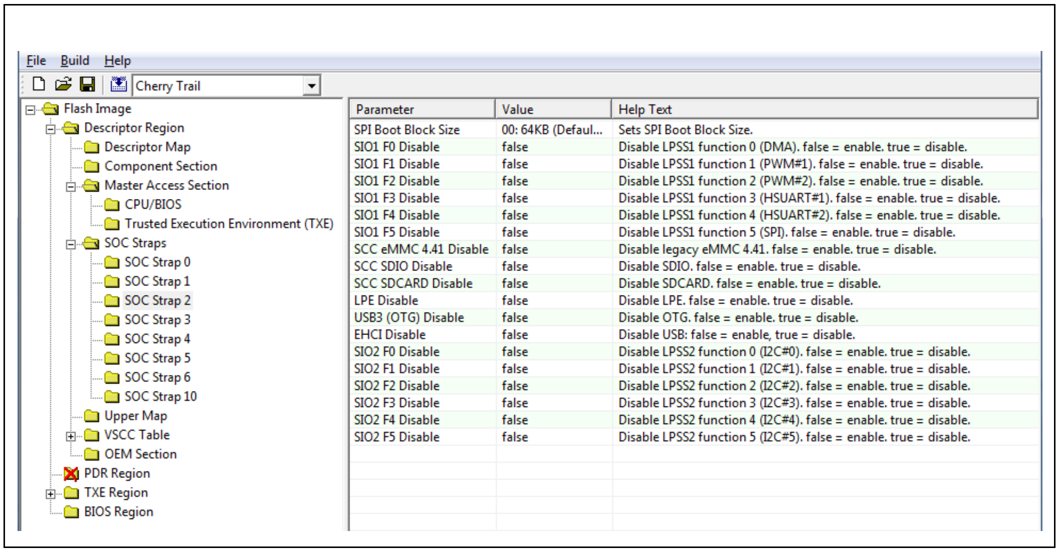
Parameter	Value	Help Text
PCI Bus ID	0	
PCI Device ID	0	
PCI Function ID	0	
Read Access	0xFF	0xFF = Debug/Manufacturing, 0x0B = Produ
Write Access	0xFF	0xFF = Debug/Manufacturing, 0x0A = Produ

3.4.12 SoC Straps

These sections contain configuration options for the SoC. The number of Soft Strap sections and their functionality differ based on the target SoC. Improper settings could lead to undesirable behavior from the target platform.

Note: The SoC is different for each SKU which means only relevant strap shows up for modification for certain SKU. (Refer to Figure10 and for more information on how to set them correctly, see the FW BringUp Guide or the Cherry Trail SoC SPI programming guide)

Figure 11. SoC Straps



Parameter	Value	Help Text
SPI Boot Block Size	00: 64KB (Defaul...	Sets SPI Boot Block Size.
SIO1 F0 Disable	false	Disable LPSS1 function 0 (DMA). false = enable. true = disable.
SIO1 F1 Disable	false	Disable LPSS1 function 1 (PWM#1). false = enable. true = disable.
SIO1 F2 Disable	false	Disable LPSS1 function 2 (PWM#2). false = enable. true = disable.
SIO1 F3 Disable	false	Disable LPSS1 function 3 (HSUART#1). false = enable. true = disable.
SIO1 F4 Disable	false	Disable LPSS1 function 4 (HSUART#2). false = enable. true = disable.
SIO1 F5 Disable	false	Disable LPSS1 function 5 (SPI). false = enable. true = disable.
SCC eMMC 4.41 Disable	false	Disable legacy eMMC 4.41. false = enable. true = disable.
SCC SDIO Disable	false	Disable SDIO. false = enable. true = disable.
SCC SDCARD Disable	false	Disable SDCARD. false = enable. true = disable.
LPE Disable	false	Disable LPE. false = enable. true = disable.
USB3 (OTG) Disable	false	Disable OTG. false = enable. true = disable.
EHCI Disable	false	Disable USB: false = enable, true = disable.
SIO2 F0 Disable	false	Disable LPSS2 function 0 (I2C#0). false = enable. true = disable.
SIO2 F1 Disable	false	Disable LPSS2 function 1 (I2C#1). false = enable. true = disable.
SIO2 F2 Disable	false	Disable LPSS2 function 2 (I2C#2). false = enable. true = disable.
SIO2 F3 Disable	false	Disable LPSS2 function 3 (I2C#3). false = enable. true = disable.
SIO2 F4 Disable	false	Disable LPSS2 function 4 (I2C#4). false = enable. true = disable.
SIO2 F5 Disable	false	Disable LPSS2 function 5 (I2C#5). false = enable. true = disable.

3.4.13 VSCC Table

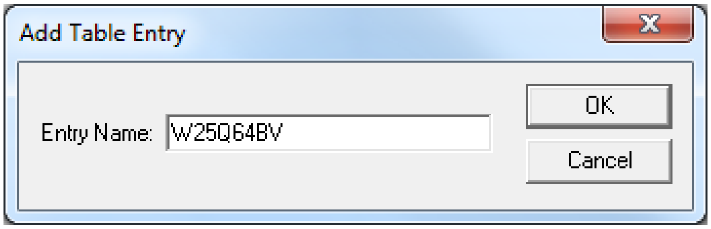
This section is used to store information to setup flash access for Intel® TXE. This does not have any effect on the usage of the FPT. **If the information in this section is incorrect, Intel® TXE FW may not communicate with the flash device.** The information provided is dependent on the flash device used on the system. (For more information, see the Cherry Trail SoC SPI Programming Guide)

3.4.14 Adding New Table

To add a new table:

1. Right-click **Descriptor Region > VSCC table**.
2. Select **Add Table Entry** from the pop-up menu. The **Add Table Entry** dialog appears.

Figure 12. Add VSCC Table Entry Dialog



Entry Name: W25Q64BV

3. Enter a name into the **Entry Name** field.

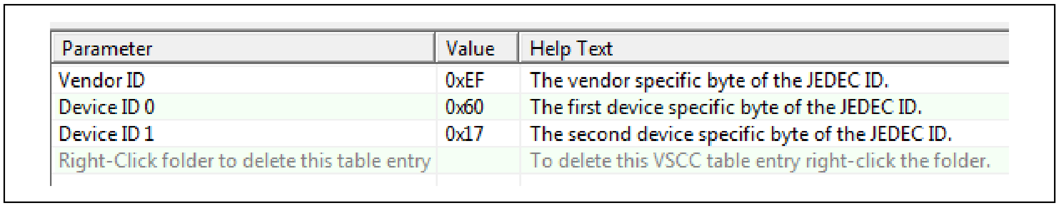
Note: To avoid confusion it is recommended that each table entry name be unique. There is no checking mechanism in FITC to prevent table entries that have the same name and no error message is displayed in such cases.

4. Click **OK**, the new table is listed in the left pane under **VSCC Table** and user can enter into it the values for the flash device. (See Figure 13), which shows the parameters of a new VSCC table.)

Note: The VSCC register value will be automatically populated by FITc using the vscccommn.bin file the appropriate information for the Vendor and Device ID.

Note: If the descriptor region is being built manually the user will need to reference the VSCC table information for the parts being supported from the manufacturers' serial flash data sheet. The Cherry Trail SoC SPI Programming Guide should be used to calculate the VSCC values.

Figure 13. Sample VSCC Table Entry



Parameter	Value	Help Text
Vendor ID	0xEF	The vendor specific byte of the JEDEC ID.
Device ID 0	0x60	The first device specific byte of the JEDEC ID.
Device ID 1	0x17	The second device specific byte of the JEDEC ID.
Right-Click folder to delete this table entry		To delete this VSCC table entry right-click the folder.

3.4.15 Removing an Existing VSCC Table

To remove an existing table:

1. Right-click on the name of the table in the left pane that the user wants to remove.
2. Choose **Remove Table Entry**. The table and all of the information will be removed.

3.4.16 Modifying Intel® TXE Region

The Intel® TXE region contains all of the FW data for the Intel® TXE (including the Intel® TXE FW Kernel and Intel® NFC capabilities).

3.4.17 Setting Intel® TXE Region Binary File

To select Intel® TXE region binary file:

1. Select Intel® TXE Region tree node.
2. Double-click on the **TXE Binary Input file** in the list. A dialog appears that lets the user select the Intel® TXE file to be used.
3. Click **OK** to update the parameter. When the flash image is built, the content of this file is copied into the Intel® TXE Region.

3.4.18 Configuration

The Configuration parameters are visible and editable only after a valid Intel® TXE FW image has been loaded.

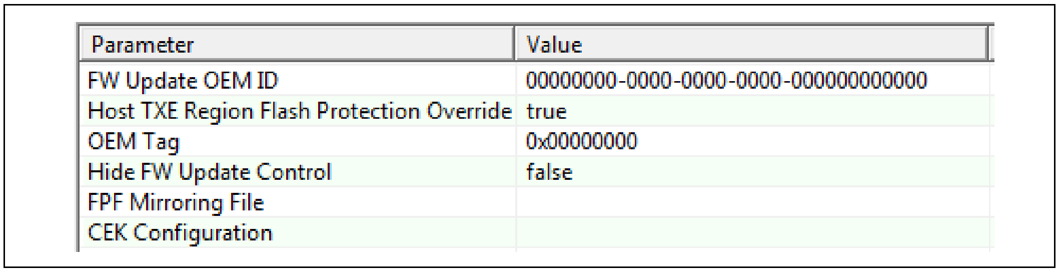
If any of the parameters do not have the Intel-recommended value, the offending row is highlighted yellow but no errors are reported. The highlighted yellow is designed to draw attention to these values to ensure these parameters are set correctly.

3.4.19 Intel® TXE Section

This section describes Intel® TXE FW Kernel parameters. (See the FW Bringup guide for general information and see Appendix for more details.)

The Intel® TXE section lets the user define the system features. The parameter values can be found in the **Help Text** next to the parameter value as shown in Figure 14.

Figure 14. Intel® TXE Section



Parameter	Value
FW Update OEM ID	00000000-0000-0000-0000-000000000000
Host TXE Region Flash Protection Override	true
OEM Tag	0x00000000
Hide FW Update Control	false
FPF Mirroring File	
CEK Configuration	

3.4.20 Features Supported

The Features Supported section determines which features are supported by the system. If a system does not meet the minimum hardware requirements, no error message is given when programming the image. (See the FW Bringup guide for general information for re details.)

These options control the availability and visibility of FW features.

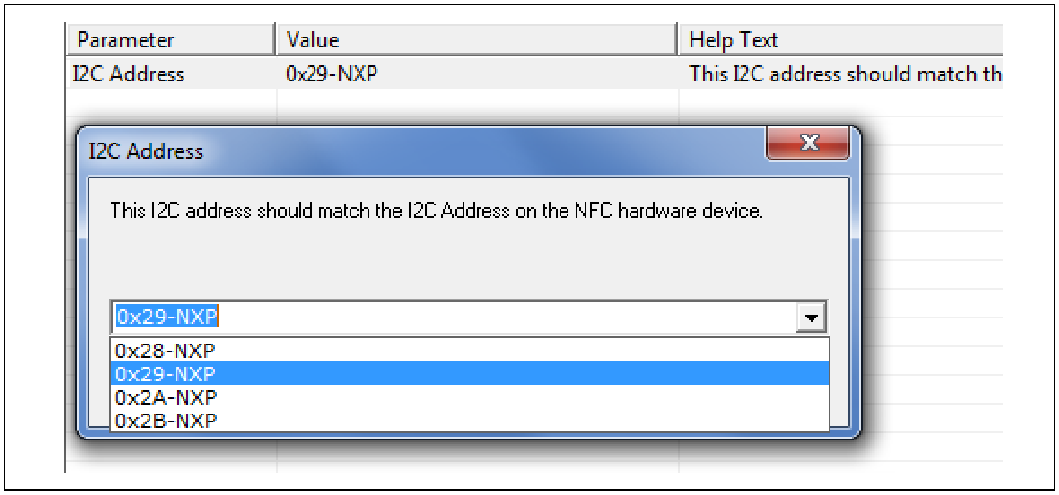
In cases where a specific feature is configurable in the UEFI BIOS, permanently disabling it through the **Features Supported** section hides/disables that feature in UEFI BIOS.

The ability to change certain options is SKU-dependent and – depending on the SKU selected – some of default values will be disabled and cannot be changed.

3.4.21 Intel® NFC Capabilities

This section allows the end user to either enable or disable Near Field Communication which connects to TXE via I2C bus. I2C/SMbus salve address of NFC devices also can be configured with different value. This address may vary from one NFC module vendor to another. Make sure you know the SMBUS address used by your NFC HW module.

Figure 15. Intel NFC Capabilities



Parameter	Value	Help Text
I2C Address	0x29-NXP	This I2C address should match th

I2C Address

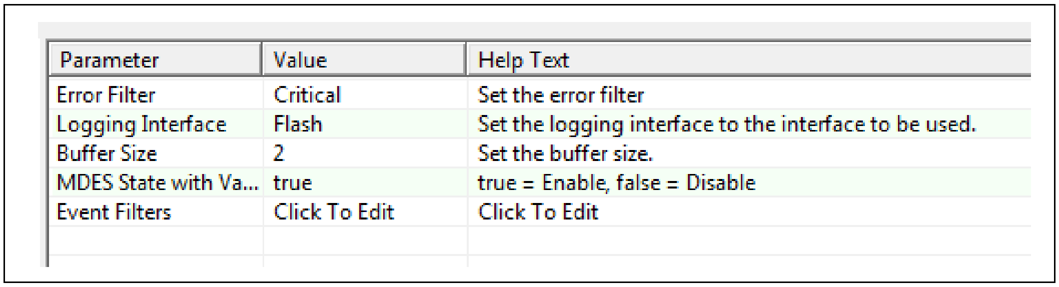
This I2C address should match the I2C Address on the NFC hardware device.

0x29-NXP

3.4.22 TXE Debug Event Service

The TXE Debug Event Service section allows the end user to specify the configuration settings of TXE Debug error and event filters. (See the FW Bringup guide for general information)

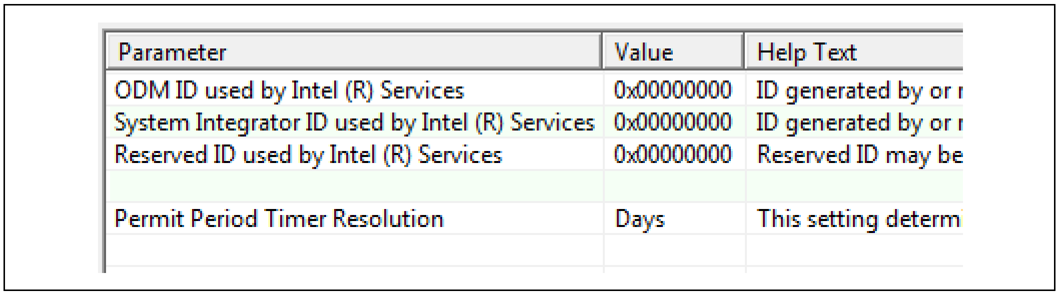
Figure 16. TXE Debug Event Services



Parameter	Value	Help Text
Error Filter	Critical	Set the error filter
Logging Interface	Flash	Set the logging interface to the interface to be used.
Buffer Size	2	Set the buffer size.
MDES State with Va...	true	true = Enable, false = Disable
Event Filters	Click To Edit	Click To Edit

3.4.23 Setup and Configuration Section

The Setup and Configuration section allows the end user to specify the configuration settings. (See the FW Bringup guide for general information and see Appendix E for more details)



Parameter	Value	Help Text
ODM ID used by Intel (R) Services	0x00000000	ID generated by or r
System Integrator ID used by Intel (R) Services	0x00000000	ID generated by or r
Reserved ID used by Intel (R) Services	0x00000000	Reserved ID may be
Permit Period Timer Resolution	Days	This setting determ

Figure 17. Setup and Configuration Section

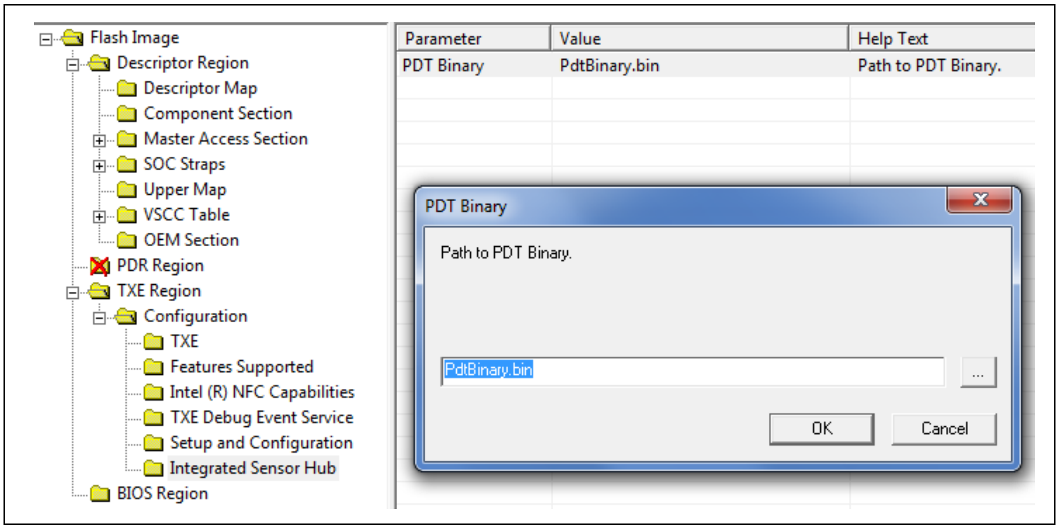
3.4.24 Setting Intel® Integrated Sensor Solution

PDT binary contains sensor lists and configuration and calibration data file which can be generated and modified from PDT editor of Intel® ISS FDK kit. To select binary file for Intel® ISS firmware:

4. Select Intel® Integrated Sensor Solution node.
5. Double-click on the **PDT Binary** in the list. A dialog appears that lets the user select the PDT binary file to be used.

Click **OK** to update the path to PDT Binary. When the flash image is built, the content of this file is copied into the Intel® TXE Region for Intel® ISS to use as need.

Figure 18. Integrated Sensor Solution



Parameter	Value	Help Text
PDT Binary	PdtBinary.bin	Path to PDT Binary.

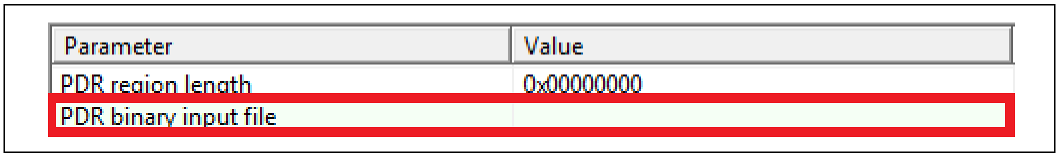
Path to PDT Binary.

PdtBinary.bin

3.4.25 Modifying PDR Region

The PDR Region contains various configuration parameters that let the user customize the computer's behavior.

Figure 19. PDR Region Options



Parameter	Value
PDR region length	0x00000000
PDR binary input file	

3.4.26 Setting PDR Region Length Option

The PDR Region length option should not be altered. A value of 0x00000000 indicates that the PDR Region will be auto-sized by FITc tool based on PDR binary input file.

3.4.27 Setting PDR Region Binary File

To select the PDR region binary file:

1. Select **PDR Region** in the left pane; the PDR Region parameters are listed in the right pane.
2. Double-click the **PDR binary input file** parameter. A dialog appears that lets the user specify which PDR file to use.
3. Click **OK** to update the parameter. When the flash image is built, the contents of this file is copied into the BIOS region.

3.4.28 Enabling / Disabling PDR Region

The PDR Region can be excluded from the flash image by disabling it in FITC.

To disable the PDR Region:

1. Right-click **PDR Region** in the left pane.
2. Select **Disable Region** from the pop-up menu. When the flash image is built, there is no PDR Region in it.

Note: This region is disabled by default.

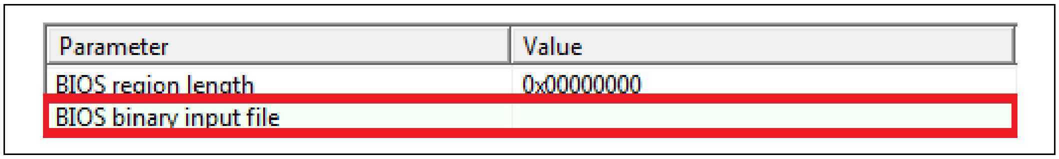
To enable the PDR Region:

1. Right-click **PDR Region** in the left pane.
2. Select **Enable Region** from the pop-up menu.

3.4.29 Modifying BIOS Region

The BIOS Region contains the BIOS code run by the host processor. This is done so that if the flash descriptor becomes corrupt for any reason, the SoC defaults to legacy mode and looks for the reset at the end of the flash memory. By placing the BIOS Region at the end there is a chance the system will still boot. It is also important to note that the BIOS binary file is aligned with the end of the BIOS Region so that the reset vector is in the correct place. This means that if the binary file is smaller than the BIOS Region, the region is padded at the beginning instead of at the end.

Figure 20. BIOS Region Parameters



Parameter	Value
BIOS region length	0x00000000
BIOS binary input file	

3.4.30 Setting BIOS Region Length Parameter

The value of the BIOS Region length parameter should not be altered. A value of 0x00000000 indicates that the BIOS Region will be auto-sized by FITc tool based on BIOS binary input file.

3.4.31 Setting BIOS Region Binary File

To select the BIOS region binary file:

1. Select **BIOS Region** in the left pane; the BIOS Region parameters are listed in the right pane.
2. Double-click **BIOS binary input file** parameter. A dialog appears that lets the user specify which BIOS file to use.
3. Click **OK** to update the parameter. When the flash image is built, the contents of this file are copied into the BIOS region.

3.4.32 Enabling/Disabling BIOS Region

The BIOS Region can be excluded from the flash image by disabling it in FITC.

To disable BIOS Region:

1. Right-click **BIOS Region** in the left pane.
2. Select **Disable Region** from the pop-up menu. When the flash image is built, there is no BIOS Region in it.

To enable BIOS Region:

1. Right-click **BIOS Region** in the left pane.
2. Select **Enable Region** from the pop-up menu.

3.4.33 Building Flash Image

The flash image can be built with the FITC GUI interface.

To build a flash image with the currently loaded configuration:

- Go to **Build** > **Build Image**.

  – OR –

- Specify an XML file with the /b option in the command line.

FITC uses an XML configuration file and the corresponding binary files to build the SPI flash image. The following is produced when an image is built:

- Binary file representing the image
- Text file detailing the various regions in the image
- Optional set of intermediate files (see Section 3.4.6).
- Multiple binary files containing the image broken up according to the flash component sizes

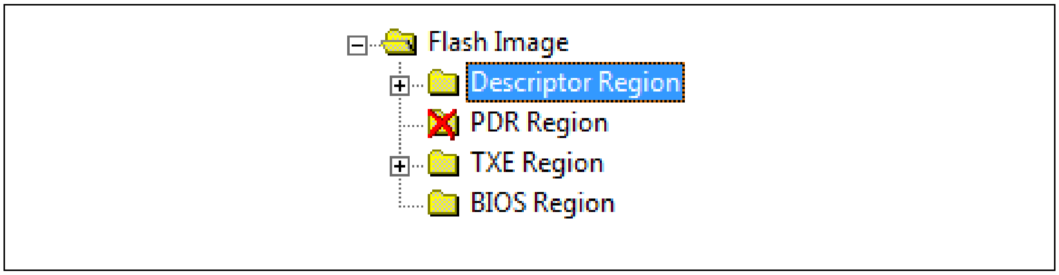
Note: These files are only created if two flash components are specified.

The individual binary files can be used to manually program independent flash devices using a flash programmer. However, the user should select the single larger binary file when using FPT.

3.4.34 Change Region Order on SPI Device

The order and placement of the regions in the full SPI image created by FITC can be altered. The location of each region is determined by the order of the PDR, TXE and BIOS regions as they are displayed in left pane of the FITC window.

Figure 21. Region Order



Each region is added to the full SPI image in the order in which they appear in the list. The order of the regions in the full SPI image created from the regions listed in Figure 23 in order immediately after the Descriptor Region:

1. TXE Region
2. BIOS Region

This can be useful when programming a system with two SPI devices. It is possible to change the order of the PDR, TXE and BIOS regions by clicking and dragging the region to the required location. Figure 23 shows that the Intel® TXE is placed on the first SPI device and the BIOS Region is placed on the second SPI device. The length of each region and the order determines if that region is on the first or second SPI device.

3.4.35 Decomposing an Existing Flash Image

FITC is capable of taking an existing flash image and decomposing it in order to create the corresponding configuration. This configuration can be edited in the GUI like any other configuration (see below). A new image can be built from this configuration that is almost identical to the original, except for the changes made to it.

To decompose an image:

1. Go to **File** > **Open.**
2. Change the file type filter to the appropriate file type.
3. Select the required file and click **Open**. The image is automatically decomposed, the GUI is updated to reflect the new configuration, and a folder is created with each of the regions in a separate binary file.

Note: It is also possible to decompose an image by simply dragging and dropping the file into the main window. When decomposing an image, there are some NVARs will not be able to be decomposed by FITC. FITC will use Intel default value instead. User might want to check the log file to find out which NVARs were not parsed.

Note: FITC will decompose only Cherry Trail SoC TXE firmware images. FITC will read the TXE region firmware version from the binary to determine if the image is a SoC image. If the image is not a SoC image an error should be displayed to the user and the image should not be decomposed.

3.4.36 Command Line Interface

FITC supports command line options.

To view all of the supported options: Run the application with the -? option.

The command line syntax for FITC is:

```
FITC [/h] [/?][/b] [/o <file>] [/platform <value>] [/txe <file>]
        [/bios <file>] [/pdr <file>] [/w <path>] [/s <path>] [/d <path>]
        [/u1 <value>] [/u2 <value>] [/u3 <value>]  [/i <enable|disable>]
        [/flashcount <1|2>] [/flashsize1 <size>] [/flashsize2 <size>]
        [/save <file>] [/fpf <file>] [XML or BIN file]
```

Table 7. FITC Command Line Options

Option	Description
-h	Display Help
-?	Displays the command line options.
-b	Automatically builds the flash image. The GUI does not appear if this flag is specified. This option causes the program to run in auto-build mode. If there is an error, a valid message is displayed and the image is not built. If a BIN file is included in the command line, this option decomposes it.
-o <file>	Path and filename where the image is saved. This command overrides the output file path in the XML file.
-txe <file>	Overrides the binary source file for the Intel® TXE Region with the specified binary file.
-bios <file>	Overrides the binary source file for the BIOS Region with the specified binary file.
-pdr <file>	Overrides the binary source file for the PDR Region with the specified binary file.
-w <path>	Overrides the working directory environment variable $WorkingDir. It is recommended that the user set these environmental variables first. (Suggested values can be found in the OEM Bringup Guide.)

Option	Description
-s <path>	Overrides the source file directory environment variable $SourceDir. It is recommended that the user set these environmental variables before starting a project.
-d <path>	Overrides the destination directory environment variable $DestDir. It is recommended that the user set these environmental variables before starting a project.
-u1 <value>	Overrides the $UserVar1 environment variable with the value specified. Can be any value required.
-u2 <value>	Overrides the $UserVar2 environment variable with the value specified. Can be any value required.
-u3 <value>	Overrides the $UserVar3 environment variable with the value specified. Can be any value required.
-i <enable|disable>	Enables or disables intermediate file generation.
-flashcount <0, 1 or 2>	Overrides the number of flash components in the Descriptor Region. If this value is zero, only the Intel® TXE Region is built.
-flashsize1 <0, 1, 2, 3, 4 or 5>	Overrides the size of the first flash component with the size of the option selected as follows: 0 = 512KB 1 = 1MB 2 = 2MB 3 = 4MB 4 = 8MB 5 = 16MB.
-flashsize2 <0, 1, 2, 3, 4 or 5>	Overrides the size of the first flash component with the size of the option selected as follows: 0 = 512KB 1 = 1MB 2 = 2MB 3 = 4MB 4 = 8MB 5 = 16MB.
-save	Save's the XML file.
-fpf <file>	Overrides the FPF Mirroring NVAR using input FPF Mirroring file with file path point to the FPF mirroring file. FITC will take in the FPF mirroring file convert the contents of the FPF mirroring file into the format required for the FPF Mirroring NVAR before setting these values. Then decompose the FPF Mirror NVAR and write the contents to an FPF Mirroring File. This file shall be saved at the same path as the TXE region. If decompose fails an error shall be displayed to the user. **NOTE:** FITC will warn the user if they attempt add a file to this NVAR on production-fw images. Pre-production images should not have a warning.
<XML File>	Used when generating a flash image file. A sample xml file is provided along with the FITC. When an xml file is used with the /b option, the flash image file is built automatically.

Option	Description
<BIN File>	Decomposes the BIN file. The individual regions are separated and placed in a folder with the same name as the BIN file.

3.4.37 Example – Decomposing an Image and Extracting Parameters

The NVARS variables and the current value parameters of an image can be viewed by dragging and dropping the image into the main window, which then displays the current values of the image's parameters.

An image's parameters can also be extracted by entering the following commands into the command line:

```
Fitc.exe output.bin /b
```

This command would create a folder named "output". The folder contains the individual region binaries (Descriptor, Intel® TXE, and BIOS) and the Map file.

The xml file contains the current Intel® TXE parameters.

The Map file contains the start, end, and length of each region.

3.4.38 More Examples of FITC CLI

Note: If using paths defined in the KIT, be sure to put "" around the path as the spaces cause issues.

Build image with assigned BIOS and TXE binary:

```
fitc.exe /b /bios "..\..\..\Image Components\BIOS\BIOS.ROM" /txe
"..\..\..\Image Components\Firmware\CHT_TXE_PreProduction.BIN" <file.bin
or file.xml>
```

Take an existing image and put in a new BIOS binary:

```
fitc.exe /b /bios "..\..\..\Image Components\BIOS\BIOS.ROM" <file.bin or
file.xml>
```

Take an existing image and put in a different Intel® TXE region:

```
fitc.exe /b /txe "..\..\..\Image
Components\Firmware\CHT_TXE_PreProduction.BIN" <file.bin or file.xml>
```

§

4 Intel® Flash Programming Tool

The Flash Programming Tool (FPT) is used to program a complete SPI image into the SPI flash device(s).

FPT can program each region individually or it can program all of the regions with a single command. The user can also use FPT to perform various functions such as:

- View the contents of the flash on the screen.
- Write the contents of the flash to a log file.
- Perform a binary file to flash comparison.
- Write to a specific address block.
- Program fixed offset variables.
- FPF programming and lock.

Note: For proper function in a multi-SPI configuration the Block Erase, Block Erase Command and Chip Erase must all match.

4.1 System Requirements

The EFI version of FPT (**fpt.efi**) runs on an EFI environment.

The Windows* version (**fptw.exe**) requires administrator privileges to run under Windows* OS. The user needs to use the **Run as Administrator** option to open the CLI in Windows* 32 bit OS.

The Windows* 64 bit version (fpt64w.exe) is designed for running in native 64 bit OS environment which does not have 32 bit compatible mode available for example Windows* PE 64.

The Android* version (FPT) requires root privileges to run under Android* OS using ADB shell. FPT and fparts.txt should be push onto SUT and same path before performing.

FPT requires that the platform is bootable (i.e. working BIOS) and an operating system to run on. It is designed to deliver a custom image to a computer that is already able to boot and is not a means to get a blank system up and running. FPT must be run on the system with the flash memory to be programmed.

One possible workflow for using FPT is:

1. A pre-programmed flash with a bootable BIOS image is plugged into a new computer.
2. The computer boots.
3. FPT is run and a new BIOS/Intel® TXE image is written to flash.
4. The computer powers down.

5. The computer powers up, boots, and is able to access its Intel® TXE capabilities as well as any new custom BIOS features.

4.2 Flash Image Details

A flash image is composed of up to five regions. The locations of these regions are referred to in terms of where they can be found within the overall layout of the flash memory.

Figure 22. Flash Image Regions

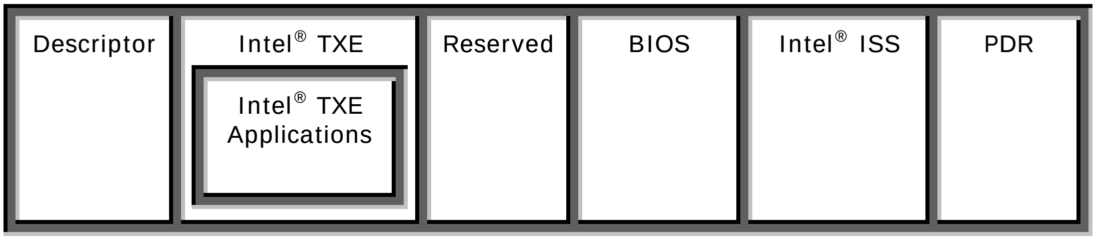


Table 8. Flash Image Regions – Description

Component	Description
Descriptor	Region that takes up a fixed amount of space at the beginning of the flash memory. Contains information such as: Space allocated for each region of the flash image. Read/write permissions for each region. A space that can be used for vendor-specific data.
Intel® TXE	Contains code and configuration data for Intel® TXE applications, such as Intel® PTT technology. Intel ISS configuration data is located in the Intel TXE data region (NVAR)
Intel® ISS	Contains code for Intel® ISS firmware and OEM ISS code.
Reserved	This region is reserved for future use.
BIOS	Contains code and configuration data for the entire platform.
PDR	Region that allows system manufacturers to define custom features for the platform.

4.3 Microsoft Windows* Required Files

The Microsoft Windows* version of the FPT executable is **fptw.exe**. The following files must be in the same directory as **fptw.exe**:

- fparts.txt – contains a comma-separated list of attributes for supported flash devices. The text in the file explains each field. An additional entry may be required in this file to describe the flash part which is on the target system. Examine the target board before adding the appropriate attribute values. The supplied file is already populated with default values for SPI devices used with Intel CRBs.
- fptw.exe – the executable used to program the final image file into the flash.
- pmxdll.dll

- idrvdll.dll
- For tools to work under the Windows* PE environment, you must manually load the driver with the .inf file in the Intel® TXEI driver installation files. Once you locate the .inf file you must use the Windows* PE cmd `drvload ipc.inf` to load it into the running system each time Windows* PE reboots. Failure to do so causes errors for some features.

Table 9. FPT Windows* OS Requirements

FPT version	Target OS	Support Drivers
FPTW.EXE	Windows* 32 / 64 bit w/WOW64	idrvdll.dll, pmxdll.dll
FPTW64.EXE	Windows* Native 64 bit	idrvdll32e.dll, pmxdll32e.dll

Note: In the Windows* environment for operations involving global reset you should add a pause or delay when running FPTW using a batch or script file.

4.4 EFI Required Files

The EFI version of the FPT executable is **fpt.efi**. The following files must be in the same directory as **fpt.efi**:

- **fparts.txt** – contains a comma-separated list of attributes for supported flash devices. The text in the file explains each field. An additional entry may be required in this file to describe the flash part which is on the target system. Examine the target board before adding the appropriate attribute values. The supplied file is already populated with default values for SPI devices used with Intel CRBs.
- **fpt.efi** – the executable used to program the final image file into the flash.

4.5 Programming Flash Device

Once the Intel® TXE is programmed, it runs at all times. Intel® TXE is capable of writing to the SPI flash device at any time as need.

4.5.1 Stopping Intel® TXE SPI Operations

FPT will automatically halt Intel® TXE SPI access prior to erasing or writing data in the TXE region. Customers do not have use either of the following steps listed below when updating platforms unless the descriptor has been locked.

Intel® TXE SPI Operations can be stopped in the following ways:

- Assert GPIO_SUS[5] pin low (Flash Descriptor Security Override Strap) on the rising edge of PMC_PWROK during power transition. (Refer to Cherry Trail Platform Design Guide for more detail and implementation recommendation)
- Send the HMRFPO TXEI message from BIOS to Intel® TXE (Refer to Intel® TXE BIOS Writer's Guide for more detail and implementation recommendation)

Note: When updating the entire Intel TXE region using the FPT tool, FPT will automatically stop Intel TXE before programming. No action is required in this case.

4.6 Usage

The EFI, Windows* and Android* versions of the FPT can run with command line options.

To view all of the supported commands: Run the application with the -? option.

The commands in EFI and Windows* versions have the same syntax. The command line syntax for fpt.efi, fptw.exe and fptw64.exe is:

```
FPTw.exe [-H|?] [-VER] [-EXP] [-VERBOSE] [-Y] [-P] [-LIST] [-I]

      [-F] [-ERASE] [-VERIFY] [-D] [-DESC] [-BIOS] [-TXE] [-PDR]

      [-C] [-B] [-E] [-REWRITE] [-ADDRESS|A] [-LENGTH|L] [-FOVS]

      [-CFGGEN] [-U] [-O] [-IN] [-N] [-ID] [-V] [-LOCK] [-DUMPLOCK]

      [-PSKFILE] [-CLOSEMNF] [-GRESET] [-PAGE] [-SPIBAR] [-R] [-VARS]

      [-COMMIT] [-COMPARE] [-HASHED] [-PROVKB] [-READKB] [-WRITEFPF]

      [-READFPF] [-READFPFATTRIB] [-COMPAREFPF] [-FPFS] [-WRITEGLOBAL]

      [-READGLOBAL] [-GETFPFLOCKSTAT] [-WRITEFPFBATCH] [-
COMPAREFPFBATCH]
```

Table 10. Command Line Options for fpt.efi, fptw.exe and fptw64.exe

Option	Description
-H|-?	Displays the list of command line options supported by FPT tool.
-VER	Shows the version of the tools.
-EXP	Shows examples of how to use the tools.
-VERBOSE [<file>]	Displays the tool's debug information or stores it in a log file.
-Y	Bypasses Prompt. FPT does not prompt user for input. This confirmation will automatically be answered with "y".
-P <file>	Flash parts file. Specifies the alternate flash definition file which contains the flash parts description that FPT has to read. By default, FPT reads the flash parts definitions from **fparts.txt**.
-LIST	Supported Flash Parts. Displays all supported flash parts. This option reads the contents of the flash parts definition file and displays the contents on the screen.
-I	Info. Displays information about the image currently used in the flash.

Option	Description
-F <file> <NOVERIFY>	Flash. Programs a binary file into an SPI flash. The user needs to specify the binary file to be flashed. FPT reads the binary, erases the flash, and then programs the binary into the flash. After a successful flash, FPT verifies that the SPI flash matches the provided image. Without specify the length with –L option, FPT will use the total SPI size instead of an image size. The NOVERFY sub-option *must* follow the file name. This will allow flashing the SPI without verifying the programming was done correctly. The user will be prompted before proceeding unless '-y' is used.
-ERASE:	Block Erase. Erases all the blocks in a flash. This option does not use the chip erase command but instead erases the SPI flash block by block. This option can be used with a specific region argument to erase that region. This option cannot be used with the `–f, -b, -c, -d` or `–verify` options.
-VERIFY <file>:	Verify. Compares a binary to the SPI flash. The image file name has to be passed as a command line argument if this flag is specified.
-D <file> :	Dump. Reads the SPI flash and dumps the flash contents to a file or to the screen using the STDOUT option. The flash device must be written in 4KB sections. The total size of the flash device must also be in increments of 4KB.
-DESC:	Read/Write Descriptor region. Specifies that the Descriptor region is to be read, written, or verified. The start address is the beginning of the region.
-BIOS:	Read/Write BIOS region. Specifies that the BIOS region is to be read, written, or verified. Start address is the beginning of the region.
-TXE:	Read/Write Intel® TXE region. Specifies that the Intel® TXE region is to be read, written, or verified. The start address is the beginning of the region.
-PDR:	Read/Write PDR region. Specifies that the PDR region is to be read, written, or verified. The start address is the beginning of the region.
-C:	Chip erase. Erases the contents of SPI flash device(s). This function does NOT erase block by block.
-B:	Blank Check. Checks whether the SPI flash is erased. If the SPI flash is not empty, the application halts as soon as contents are detected. The tool reports the address at which data was found.
-E:	Skip Erase. Does not erase blocks before writing. This option skips the erase operation before writing and should be used if the part being flashed is a blank SPI flash device.
-REWRITE	Rewrite the SPI flash with file data even if flash is identical
-A<value>, -ADDRESS <value>	Write/Read Address. Specifies the start address at which a read, verify, or write operation must be performed. The user needs to provide an address. This option is not used when providing a region since the region dictates the start address.
-L <value>, LENGTH <value>	Write/Read Length. Specifies the length of data to be read, written, or verified. The user needs to provide the length. This option is not used when providing a region since the region/file length determines this.
-FOVS:	Supported Fixed Offset Variables. Displays all supported FOVs supported by FPT. This option displays names and IDs of supported FOVs.

Option	Description
-CFGGEN	FOV Input file generation option. This creates a file which can be used to update the FOVs. If no file name is specified the default name "FPT.CFG" will be used.
-U:	Update. Updates the FOVs in the flash. The user can update the multiple FOVs by specifying their names and values in the parameter file. The parameter file must be in an INI file format (the same format generated by the `–cfggen` command). The `-in <file>` option is used to specify the input file.
-O <file>	Output File. The file used by FPT to output FOV information.
-IN <file>	Input File. The file used by FPT for FOV input. This option flag must be followed by a text file (i.e., `fpt –u –in FPT.cfg`). The tool updates the FOVs contained in the text file with the values provided in the input file. User can also use FPT –cfggen to generate this file.
-N <value>	Name. Specifies the name of the FOV that the user wants to update in the image file or flash. The name flag must be used with Value (`-v`).
-ID <value>	ID. The names of certain FOVs are quite lengthy. This option lets the user update the FOV by providing its unique identification number instead of its name. The ID for each FOV is specified in the configuration file.
-V <value>	Value. Specifies the value for the FOV variable. The name of variable is specified in the Name flag. The Value flag must follow the Name flag.
-LOCK:	Region Lock. Sets the SPI flash region access to the Intel recommended values)
-DUMPLOCK <PDR>:	Dump Lock Settings. Displays the current lock settings on the screen. The lock settings are read from the descriptor region.
-PSKFILE <file>	PID/PPS/Password pair file. Specifies the input file that contains the one or more PID/PPS/Password key value pairs. This option is used to update the PID, PPS, and Password FOVs whose values are read from the input file. This option only support version 1 FiletypeHeader UUID

Option	Description
-CLOSEMNF <NO> <PDR>:	End of Manufacturing. This option is executed at the end of manufacturing phase. This option does the following: Sets the Intel® TXE manufacturing mode done bit (Global Lock bit). Verifies that the Intel® TXE manufacturing mode done bit (Global Lock bit) is set. Sets the master region access permission in the Descriptor region to its Intel-recommended value Verifies that flash regions are locked. If the image was properly set before running this option, FPT skips all of the above and reports PASS. If anything was changed, FPT automatically forces a global reset through the CF9GR mechanism. The user can use the no reset option to bypass the reset. If nothing was changed, based on the current setting, the tool reports PASS without any reset. The "NO" addition will prevent the system from doing a global reset following a successful update of the Intel® TXE Manufacturing Mode Done, the Region Access permissions, or both. The "PDR" addition will allow CPU\BIOS Read & Write access to the PDR region of flash. **Note**: Running `FPT -closemnf` also sets the default value for any unprovisioning process. Run `FPT –closemnf` first if the user wants to test any unprovisioning related process. In order to allow FPT to perform a global reset, BIOS should not lock CF9GR when Intel® TXE is in manufacturing mode. This step is highly recommended to the manufacturing process. Without doing proper end of manufacturing process would lead to ship platform with potential security/privacy risk.
-GRESET <NO> :	Global Reset. FPT performs a global reset. On mobile platforms this includes driving GPIO30 low. Mobile platforms require a SUS Well power-down acknowledge-driven low before the global reset occurs or the platform may not boot up from the reset. The "NO" afterwards disables the driving of GPIO30 for mobile SKUs.
-PAGE	Pauses the screen when a page of text has been reached. Hit any key to continue.
-SPIBAR:	Display SPI BAR. FPT uses this option to display the SPI BAR.
-R <name>	NVAR Read. FPT uses this option to read a variable stored as a NVAR in the FW. The value of the variable is displayed. By default, all non- secure variables are displayed in clear-text and secure NVAR will be displayed in HASH. The `-hashed` option can be used to display the hash of a value instead of the clear-text value.
-VARS:	Display Supported Variables. FPT uses this option to display all variables supported for the `-R` and `-COMPARE` commands.
-COMMIT:	Commit. FPT uses this option to commit FOVs changes to NVAR and cause relevant reset accordingly If no pending variable changes are present, Intel® TXE does not reset and the tool displays the status of the commit operation.
-COMPARE <file>	NVAR Compare. FPT uses this option to compare a NVAR with the expected value filled in a text file. The compare entry should have the following format: `"<name>" = <value>` **NOTE:** <value> should have the form "xx ", where xx is a hexadecimal value. Each byte must be separated by a space and start with the least significant followed by the next significant byte.

Option	Description
-HASHED:	Hash Variable Output. FPT uses this option to distinguish whether the displayed output is hashed by the FW. For variables that can only be returned in hashed form this option has no effect – the data displayed is hashed regardless.
-PROVKB <file>	Provision Widevine* Google* DRM using KeyBox file.
-READKB	Display Widevine keybox device ID.
-WRITEFPF	Writes as a value to an FPF if not locked.
-READFPF	Reads the FPF value – register or Fuses depending on if the fuses have been committed or not.
-READFPFATTRIB	Display the attributes for the selected FPF.
-COMPAREFPF	Compares the stored FPF register against the expected value, provided on the command line, prior to committing.
-FPFS	Display the list of FPFs.
-WRITEGLOBAL	Writes the Global Valid Fuse.
-READGLOBAL	Reads the Global Valid Fuse.
-GETFPFLOCKSTAT <name>	Display the lock status of the specified FPF.
-WRITEFPFBATCH <f>[NoVerify]	Writes the FPF fuses from a file.
-COMPAREFPFBATCH <f>[NoVerify]	Compare the FPF fuses from a file to the actual fuses or FPF mirroring.

Table 11. FPT –closemnf Behavior

Condition before FPT -closemnf			Condition after FPT -closemnf			Other FPT Action	
TXE Mfg Done Bit Set	Flash Access Set to Intel Rec Values	TXE Mfg Mode	TXE Mfg Done Bit Set	Flash Access Set to Intel Rec Values?	TXE Mfg Mode	FPT Return Value **	Global Reset
No	No	Enabled	**Yes**	**Yes**	**Disabled**	0	Yes
No	Yes	Enabled	No	Yes	Enabled	1	No
Yes	No	Enabled	Yes	**Yes**	**Disabled**	0	Yes
Yes	Yes	Disabled	Yes	Yes	Disabled	0	No
** Return value 0 indicates successful completion. In the second case, FPT –closemnf returns 1 (= error) because it is unable to set the TXE Mfg Done bit, because flash permissions are already set to Intel recommended values (host cannot access TXE Region).							

4.7 Programming Fixed Offset Variables

FPT can program the fixed offset variables and change the default values of the parameters. The modified parameters are used by the Intel® TXE FW after a global reset (Intel® TXE + HOST reset) or upon returning from a G3 state. The fixed offset variables can be continuously changed until the Intel® TXE manufacturing mode done (formerly Global Lock bit) bit is set to 0x01. The parameters can **NOT** be modified after this bit is set. To modify the default settings for the parameters, the entire flash device must be re-programmed.

The variables can be modified individually or all at once via a text file.

Table 12. Fixed Offset Variables Options

Option	Description
fptw.exe –FOVs	Displays a list of the supported variables.
fptw.exe –cfggen	Creates an empty text file that lets the user update multiple fixed offset variables. The variables have the following format in the text file: <Parameter name> = <Value>
fptw.exe –U –IN <Text file>	Updates the fixed offset variables with the values as they are entered in the text file.
fptw.exe –U –n <name> -v <value>	Update certain variable with assigned value.
fptw.exe -commit	Commit updated FOVs to SPI flash.

See Fixed Offset Variables for a description of all the Fixed Offset Variable parameters.

Table 13. Intel Recommend Access Settings

	Intel® TXE	BIOS
Read	0b 0000 1101 = 0x0d	0b 0000 0011 = 0x0B
		0b 0001 1011 = 0x1B – BIOS access to PDR
Write	0b 0000 1100 = 0x0c	0b 0000 0010 = 0x0A
		0b 0001 1010 = 0x1A – BIOS access to PDR

4.8 Fparts.txt File

The **fparts.txt** file contains a list of all flash devices that are supported by FPT. The flash devices listed in this file must contain a 4KB erase block size. If the flash device is not listed, the user will receive the following error:

Intel (R) Flash Programming Tool. Version: x.x.x.xxxx

Copyright (c) 2007-2014, Intel Corporation. All rights reserved.

Platform: “Intel(R) Atom Zxxxx”

Error 75: “fparts.txt” file not found.

If the device is not located in **fparts.txt**, the user is expected to provide information about the device, inserting the values into **fparts.txt** in same format as is used for the rest of the devices. Detailed information on how to derive the values in **fparts.txt** is found in the Cherry Trail Platform SoC SPI Programming Guide. The device must have a **4KB erase sector** and the total size of the SPI Flash device must be a multiple of 4KB. The values are listed in columns in the following order:

- Display name
- Device ID (2 or 3 bytes)
- Device Size (in bits)
- Block Erase Size (in bytes - 256, 4K, 64K)
- Block Erase Command
- Write Granularity (1 or 64)
- Unused
- Chip Erase Command.

4.9 FPF

Field Programmable Fuses (FPF) is implemented as 4 banks of one time programmable area inside Cherry Trail SoC. Objective of FPF is to let OEMs choose platform configuration before shipping their platform to end users. Default FPF value in one time programmable area inside virgin SoC will be all zero. It can be programmed once by OEM/ODM at their factory once then configuration is final and platform is ready to be ship and sold. Main usages of FPF are listed in following:

- Enabled/Disable TXE Features (ex: Secure Boot, Intel® PTT)
- Store Hash of OEM public key used for signing BIOS, etc.

FPF is recommended to be programmed end of manufacturing step, all fuse banks would be locked once Global Valid fuse set to 1 at end of manufacturing as well for security and manufacturing flow perspective. All FPF fuse will be read only after Global Valid fuse set and after post manufacturing stage. Global Valid fuse can be referred as OEM-end of manufacturing for access control purpose and OEM manufacturing flow support. This is the responsibility of the OEM/ODM to program the Global Valid fuse (1 bit) after all the OEM manufacturing FPF fuse files were programmed correctly.

If the objective is to test enable/disable features, use FPF mirroring in SPI flash and then no need to program the real fuse itself. This will prevent wasting platform due to manual mistakes or wrong Fuse configuration. If the objective is to test the FPF itself, test using SPI image stitching with FPF mirroring by FITC tool first and program the real FPF using FPT once the test pass.

- Tool for programming FPF: FPT (Flash Programming Tool)
- Tool for SPI image creation with FPF mirroring: FITC
- FPF mirroring/configuration file is input to both FPT and FITC

4.9.1 FPF Programming

To support FPF programming on Cherry Trail SoC, FPT implemented new set of commands listed in following table:

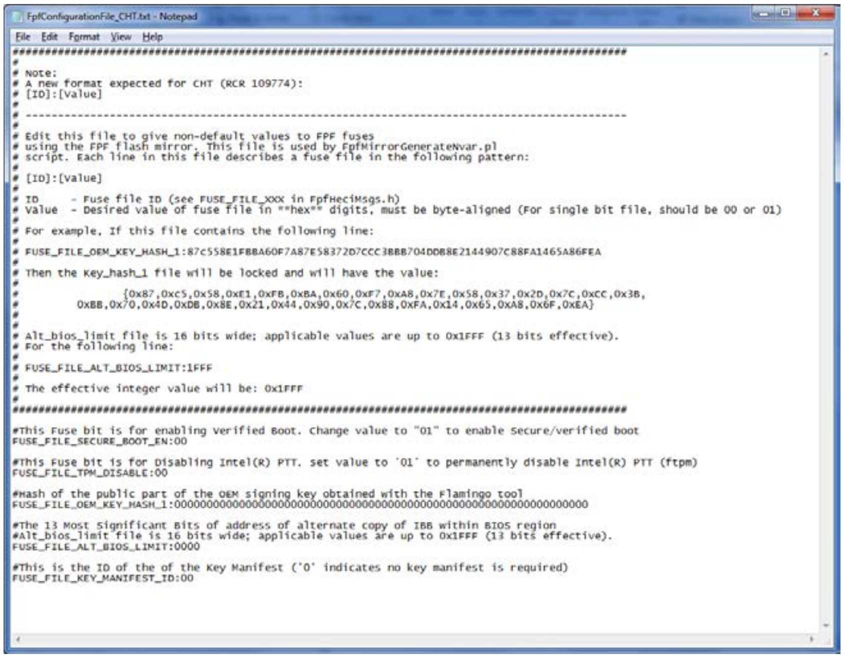
Table 14. FPT Command for FPF Access

Command Example	Usage and Purpose
-WRITEFPF <name> -V <value>	Writes as a value to an FPF if not locked.
-READFPF <name>	Reads the FPF value – register or Fuses depending on if the fuses have been committed or not.
-READFPFATTRIB <name>	Display the attributes for the selected FPF
-COMPAREFPF <name> -V <value>	Compares the stored FPF register against the expected value, provided on the command line, prior to committing.
-FPFS	Display the list of FPFs
-WRITEGLOBAL	Writes the Global Valid Fuse.
-READGLOBAL	Reads the Global Valid Fuse.
-GETFPFLOCKSTAT <name>	Display the lock status of the specified FPF
-WRITEFPFBATCH<f>[NoVerify]	Writes the FPF fuses from a file.
-COMPAREFPFBATCH<f>[NoVerify]	Compare the FPF fuses from a file to the actual fuses or FPF mirroring.

There is a batch process programming command within FPT for FPF programming to allow the OEM to use a FPF configuration file with all FPFs that are desired to be programmed, the value that is needed, and a lock status.

The format of FPF configuration file is shown in the following figure:

Figure 23. FPF Configuration File


FpfConfigurationFile_CHT.txt - Notepad
File Edit Format View Help

```
##########################################################################################
#
# Note:
# A new format expected for CHT (RCR 109774):
# [ID]:[Value]
#
# ----------------------------------------------------------------------------------------
#
# Edit this file to give non-default values to FPF fuses
# using the FPF flash mirror. This file is used by FpfMirrorGenerateNvar.pl
# script. Each line in this file describes a fuse file in the following pattern:
#
# [ID]:[Value]
#
# ID     - Fuse file ID (see FUSE_FILE_XXX in FpfHeciMsgs.h)
# Value  - Desired value of fuse file in **hex** digits, must be byte-aligned (For single bit file, should be 00 or 01)
#
# For example, If this file contains the following line:
#
# FUSE_FILE_OEM_KEY_HASH_1:87c558E1FBBA60F7A87E58372D7CCC3BBB704DDB8E2144907C88FA1465A86FEA
#
# Then the Key_hash_1 file will be locked and will have the value:
#
#                 {0x87,0xc5,0x58,0xE1,0xFB,0xBA,0x60,0xF7,0xA8,0x7E,0x58,0x37,0x2D,0x7C,0xCC,0x3B,
#         0xBB,0x70,0x4D,0xDB,0x8E,0x21,0x44,0x90,0x7C,0x88,0xFA,0x14,0x65,0xA8,0x6F,0xEA}
#
#
# Alt_bios_limit file is 16 bits wide; applicable values are up to 0x1FFF (13 bits effective).
# For the following line:
#
# FUSE_FILE_ALT_BIOS_LIMIT:1FFF
#
# The effective integer value will be: 0x1FFF
#
##########################################################################################

#This Fuse bit is for enabling Verified Boot. Change value to "01" to enable Secure/verified boot
FUSE_FILE_SECURE_BOOT_EN:00

#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_1:0000000000000000000000000000000000000000000000000000000000000000

#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

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

FPF Configuration is created as in a text file which allows to be Input to FITC to create FPF Mirroring, Input to FPT to program and verify the fuses and Input to Manifest tool to get the Public key. It is the same configuration file as input to FITC, FPT and Manifest Tool. Global Valid fuse cannot be locked which it should be set to 1 at end of manufacturing and it make no sense to lock it.

Edit this file to give non-default values to FPF fuses using the FPF flash mirror. Each line in this file describes a fuse file in the following pattern:

[ID]:[Value]

- **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)

For example, if this file contains the following line:

FUSE_FILE_OEM_KEY_HASH_1:87c558E1FBBA60F7A87E58372D7CCC3BBB704DDB8E2144907C88FA1465A86FEA

Then the Key_hash_1 file will be locked and will have the value: {0x87,0xc5,0x58,0xE1,0xFB,0xBA,0x60,0xF7,0xA8,0x7E,0x58,0x37,0x2D,0x7C,0xCC,0x3B,0xBB,0x70,0x4D,0xDB,0x8E,0x21,0x44,0x90,0x7C,0x88,0xFA,0x14,0x65,0xA8,0x6F,0xEA}

As for FUSE_FILE_ALT_BIOS_LIMIT fuse file is 16 bits wide; applicable values are up to 0x1FFF (13 bits effective). For the following line, the effective integer value will be: 0x1FFF

FUSE_FILE_ALT_BIOS_LIMIT:1FFF

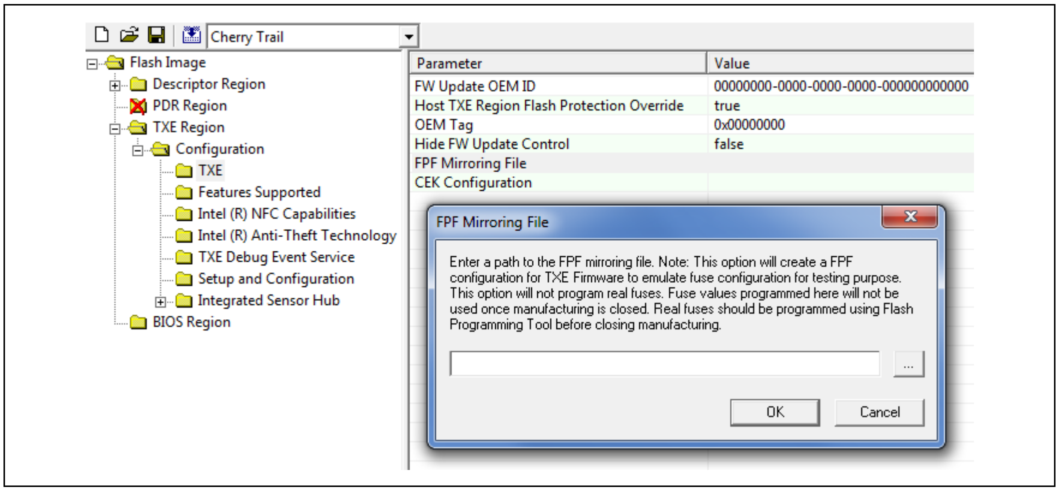
4.9.2 FPF Mirroring

It is used to mirror the Field Programmable Fuse (FPF) setting in the TXE firmware. The following is a guideline for FPF mirroring: (Refer to Intel TXE Firmware Manufacturing Recommendation for more detail)

- FITC uses the FPF configuration file to create an NVAR with the fuse configuration into production SPI image.
- TXE FW uses this NVAR and simulates the FPF settings after SPI image programming and platform boot.
- FPF mirroring enables testing FPF configuration without programming the fuse.
- Allows OEMs to test and finalize the FPF configuration they want to use prior to production.
- Actual fuses should be programmed at manufacturing line with FPT and same FPF configuration file which have been verified before.
- The production SPI image for product will ship to end user should not have FPF mirroring present, as main purpose of FPF mirroring is for early validation in production phase but not for mass production.
- Global Valid fuse should be programmed at end of manufacturing process.

Note: FITC does not update the FPF mirroring NVAR if the user decomposes an existing SPI image, modifies the text file and rebuilds the image. In order to use new or updated FPF mirroring NVAR, the user actually needs to browse to the FPF mirror file setting field from TXE region and reload new or updated text file to read in new values. FPF mirroring will not be updated or removed if you just open an image that already contains the FPF mirror NVAR and delete it from the file dialog box.

Figure 24. Dialog Box for FPF Mirroring File Insertion



Parameter	Value
FW Update OEM ID	00000000-0000-0000-0000-000000000000
Host TXE Region Flash Protection Override	true
OEM Tag	0x00000000
Hide FW Update Control	false
FPF Mirroring File	
CEK Configuration	

Enter a path to the FPF mirroring file. Note: This option will create a FPF configuration for TXE Firmware to emulate fuse configuration for testing purpose. This option will not program real fuses. Fuse values programmed here will not be used once manufacturing is closed. Real fuses should be programmed using Flash Programming Tool before closing manufacturing.

4.10 Examples

The following examples illustrate the usage of the EFI versions of the tool (fpt.efi and fptw.exe respectively). The Windows* version of the tool (Fptw.exe) behaves in the same manner apart from running in a Windows* environment.

4.10.1 Complete SPI Flash Device with Binary File

```
C:\ fptw.exe –f spi.bin

EFI:
>fpt.efi –f spi.bin or fs0:\>fpt.efi –f spi.bin

--------------------------------------------

Intel (R) Flash Programming Tool. Version: x.x.x.xxxx

Copyright (c) 2007-2014, Intel Corporation. All rights reserved.



Platform: “Intel(R) Atom Zxxxx”

Reading HSFSTS register... Flash Descriptor: Valid



    --- Flash Devices Found ---

    AT26DF321 ID:0x1F4700   Size: 4096KB (32768Kb)

    AT26DF321 ID:0x1F4700   Size: 4096KB (32768Kb)
```

```
Warning: There are some addresses that are not defined in any regions.

Read/Write/Erase operations are not possible on those addresses.


PDR Region does not exist.

- Erasing Flash Block [0x800000] – 100% complete.

- Programming Flash [0x800000] 8192KB of 8192KB – 100% complete.

- Verifying Flash [0x800000] 8192KB of 8192KB – 100% complete.

RESULT: The data is identical.


FPT Operation Passed
```

This command writes the data in the spi.bin file into a whole SPI flash from address `0x0`

4.10.2 Program Specific Region

```
fptw.exe –f bios.rom –BIOS
```

EFI:

```
fpt.efi –f bios.rom –BIOS

--------------------------------------------

Intel (R) Flash Programming Tool. Version:  x.x.x.xxxx

Copyright (c) 2007-2014, Intel Corporation. All rights reserved.

Platform: Intel(R) Atom Zxxxx

Reading HSFSTS register... Flash Descriptor: Valid

--- Flash Devices Found ---
    W25Q64BV    ID:0xEF4017    Size: 8192KB (65536Kb)
- Erasing Flash Block [0x800000]... - 100% complete.
- Programming Flash [0x800000]2560KB or 2560KB - 100% complete.
- Verifying Flash [0x800000]2560KB or 2560KB - 100% complete.
RESULT: The Data is identical.
FPT Operation Passed
```

This command writes the data in **bios.rom** into the BIOS region of the SPI flash and verifies that the operation ran successfully.

4.10.3 Program SPI Flash from Specific Address

Windows*:
```
fptw.exe -F image.bin -A 0x100 -L 0x800
```

EFI:
```
fpt.efi -F image.bin -A 0x100 -L 0x800
```

This command loads 0x800 of the binary file **image.bin** starting at address 0x0100. The starting address and the length need to be a multiple of 4KB.

4.10.4 Dump Full Image

```
fptw.exe –d imagedump.bin
```

EFI:

```
fpt.efi –d imagedump.bin


--------------------------------------------

Intel (R) Flash Programming Tool. Version:  x.x.x.xxxx

Copyright (c) 2007-2014, Intel Corporation. All rights reserved.
Platform: Intel(R) Atom Zxxxx

Reading HSFSTS register... Flash Descriptor: Valid

--- Flash Devices Found ---
       W25Q64BV    ID:0xEF4017    Size: 8192KB (65536Kb)
- Reading Flash [0x00800000]... 8192KB of 8192KB - 100% complete.
Writing flash contents to file "imagedump.bin"...
Memory Dump Complete

FPT Operation Passed
```

This command writes the contents of all regions to the file **imagedump.bin**.

4.10.5 Dump Specific Region

```
fptw.exe –d descdump.bin –desc
```

EFI:
```
fpt.efi –d descdump.bin –desc

--------------------------------------------

Intel (R) Flash Programming Tool. Version:  x.x.x.xxxx

Copyright (c) 2007-2014, Intel Corporation. All rights reserved.
Platform: Intel(R) Atom Zxxxx

Reading HSFSTS register... Flash Descriptor: Valid
```

```
--- Flash Devices Found ---
       W25Q64BV    ID:0xEF4017    Size: 8192KB (65536Kb)
- Reading Flash [0x000040]... 4KB of 4KB - 100% complete.
Writing flash contents to file "descdump.bin"...
Memory Dump Complete
FPT Operation Passed
```

This command writes the content of the Descriptor region to the file **descdump.bin**.

4.10.6 Display SPI Information

```
fptw.exe –I

--------------------------------------------
Intel (R) Flash Programming Tool. Version: x.x.x.xxxx
Copyright (c) 2007-2014, Intel Corporation. All rights reserved.

Platform: Intel(R) Atom Zxxxx
Reading HSFSTS register... Flash Descriptor: Valid

        --- Flash Devices Found ---
        AT26DF321 ID:0x1F4700      Size: 4096KB (32768Kb)
        AT26DF321 ID:0x1F4700      Size: 4096KB (32768Kb)

        --- Flash Image Information --
        Signature: VALID
        Number of Flash Components: 2
                 Component 1 - 4096KB (32768Kb)
                 Component 2 - 4096KB (32768Kb)
        Regions:
                 Descriptor - Base: 0x000000, Limit: 0x000FFF
                 BIOS       - Base: 0x600000, Limit: 0x7FFFFF
                 TXE         - Base: 0x003000, Limit: 0x5FFFFF
                 PDR        - Not present
        Master Region Access:
                 CPU/BIOS - ID: 0x0000, Read: 0xFF, Write: 0xFF
                 TXE      - ID: 0x0000, Read: 0xFF, Write: 0xFF

Used Space: 8192KB, Actual Space: 8192KB

FPT Operation Passed
```

This command displays information about the flash devices present in the computer. The base address refers to the start location of that region and the limit address refers to the end of the region. If the flash device is not specified in **fparts.txt**, FPT returns the error message "There is no supported SPI flash device installed".

4.10.7 Verify Image with Errors

```
fptw.exe -verify outimage.bin

EFI:
fpt.efi -verify outimage.bin

--------------------------------------------
Intel(R) Flash Programming Tool. Version:  x.x.x.xxxx
Copyright (c) 2007-2014, Intel Corporation. All rights reserved.
Platform: Intel(R) Atom Zxxxx
Reading HSFSTS register... Flash Descriptor: Valid
```

```
--- Flash Devices Found ---
    W25Q64BV    ID:0xEF4017    Size: 8192KB (65536Kb)
RESULT: Data does not match!
[0x00000000] Expected 0x5A, Found: 0x5A
[0x00000001] Expected 0xA5, Found: 0xA5
Total mismatches found in 64 byte block: 2
Error 204: Data verify mismatch found at address 0x000
```

This command compares the Intel® TXE region programmed on the flash with the specified FW image file **outimage.bin**. If the -y option is not used; the user is notified that the file is smaller than the binary image. This is due to extra padding that is added during the program process. The padding can be ignored when performing a comparison. The -y option proceeds with the comparison without warning.

4.10.8 Verify Image Successfully

```
fptw.exe -verify outimage.bin

EFI:
fpt.efi -verify outimage.bin

--------------------------------------------
Intel (R) Flash Programming Tool. Version: x.x.x.xxxx
Copyright (c) 2007-2014, Intel Corporation. All rights reserved.

Platform: Intel(R) Atom Zxxxx
Reading HSFSTS register... Flash Descriptor: Valid

        --- Flash Devices Found ---
        AT26DF321 ID:0x1F4700      Size: 4096KB (32768Kb)
        AT26DF321 ID:0x1F4700      Size: 4096KB (32768Kb)

- Verifying Flash [0x800000] 8192KB of 8192KB – 100% complete.
RESULT: The data is identical.

FPT Operation Passed
```

This command compares **image.bin** with the contents of the flash. Comparing an image should be done immediately after programming the flash device. Verifying the contents of the flash device after a system reset results in a mismatch because Intel® TXE changes some data in the flash after a reset.

4.10.9 Get Intel® TXE Settings

```
fptw.exe –r “Power Package 1”

------------------------------------------
Intel (R) Flash Programming Tool. Version:  x.x.x.xxxx
Copyright (c) 2007-2014, Intel Corporation. All rights reserved.
Platform: Intel(R) Cherry Trail Platform
Reading HSFSTS register... Flash Descriptor: Valid
--- Flash Devices Found ---
    W25Q64BV    ID:0xEF4017    Size: 8192KB (65536Kb)
Variable: "Power Package 1"
Value: True / 01
Retrieve Operation: Successful
```

Note: Only –r (get command) supports the –hashed optional command argument. When –hashed is used, variable value will be returned in hashed format, otherwise it will be returned in clear txt. There are a few exceptions in the case of PID and PPS, their value will be always returned in hashed format regardless –hashed is used or not. This is primarily because of security concern.

4.10.10 Compare Intel® TXE Settings

FPT –verbose –compare vars.txt compares variables with suggested values in vars.txt, and report result on the screen. Vars.txt can have the following data with verbose information: FPT –VARS can be used to get the VAR list for the platform and get the value/format from FITC advanced mode. There are settings in the ME which are stored encrypted. Users will not be able to compare them using clear text values. Use FPT –R option to read the hash value of those settings and use them as baseline for the expected value.

```
"OEMSkuRule" = EF DC EE 0F
"OEM_TAG" = 78 56 34 12
"Debug Si Features" = 00 00 00 00
"Prod Si Features" = 00 00 00 00
"TXEI TXE Region Unlockable" = True
"Sub System Vendor ID" = 00 00
"FW Update OEM ID" = 12345678-AABB-CCDD-EEFF-55AA11223344
"PROC_MISSING" = No onboard glue logic
"PAVP Permanently Disabled?" = No
"Intel(R) Anti-Theft Technology Permanently Disabled?" = No
"BIOS Reflash Capable" = False
"Boot into BIOS Setup Capable" = False
"Pause during BIOS Boot Capable" = False
"Host Based Setup and Configuration" = True
"Allow Unsigned Assert Stolen" = False
"Intel(R) Anti-Theft BIOS Recovery Timer" = Disabled
"ODM ID used by Intel(R) Service" = <hashed value>
```

4.10.11 FOV Configuration File Generation (-cfggen)

It creates an input file which can be used to update multiple (any or all) FOV’s. The file includes all the current FOV’s. When creating the file, it extracts the fixed offset variables from flash.

Note: The file generated will change every time the list of FOV’s changes.

```
fptw.exe –cfggen [ -o <Output Text File> ][ options ]

        < none >                  Creates an input file which can be
                                  modified to update multiple FOVs.  If no
                                  output file name is provided, the
                                  default “FPT.cfg” file will be created.
        -o <Output File Name>     The desired name of the file generated.
                                  If none is provided the default,
                                  fpt.cfg, will be used.
        -p < file name >          Alternate SPI Flash Parts list file.
        -page                     Pauses at screen / page / window
                                  boundaries.  Hit any key to continue.
        -Verbose [<file name>]    Displays more information.
        -y                        Will not pause to user input to continue
```

Example FPT.CFG output:

```
;
;    Flash Programming Tool FOV Programming File
;
;    Any entry that is not included, or does not have a value
;    following the label will not be updated.
;
;    Comments can be added by using a ';' as the first entry
;    on the line.
;
;    For further explanation  of the required inputs see the
;    System Tools User Guide.doc
;
;    Any entries, FOVs that are displayed with values
;    indicates that the FOV has already been given a value,
;    but has not yet been committed.  Entries without values
;    indicates that the FOV has not been written, at least
;    since the system reset or use of the '-commit' command.
;
OEMSkuRule =
    PAVP                 =
    OEM_TAG =
    ODM_ID =
    SystemIntegratorId =
    ReservedId =
```

§

5 Intel® TXEManuf

Intel TXEManuf validates Intel® TXE functionality on the manufacturing line. It does not check for LAN functionality as it assumes that all Intel® TXE components on the test board have been validated by their respective vendors. It does verify that these components have been assembled together correctly.

The Windows* version of Intel TXEManuf (Intel TXEMANUFWIN) requires administrator privileges to run under Windows* OS. The user needs to use the **Run as Administrator** option to open the CLI in Windows* 8 64/32 bit, Windows* 8 SoC.

Intel TXEManuf validates all components and flows that need to be tested according to the FW installed on the platform to ensure the functionality of Intel® TXE applications: BIOS-FW, Flash, SMBus. This tool is meant to be run on the manufacturing line.

5.1 Windows* PE Requirements

For tools to work under the Windows* PE environment, you must manually load the driver with the .inf file in the Intel® TXEI driver installation files. Once you locate the .inf file you must use the Windows* PE cmd `drvload ipc.inf` to load it into the running system each time Windows* PE reboots. Failure to do so causes errors for some features.

5.2 How to use Intel TXEMANUF

Intel TXEMANUF checks the FW SKU and runs the proper tests accordingly unless an option to select tests is specified.

Intel TXEMANUF is intelligent enough to know if it should run the test or report a result. If there is no test result available for an Intel® TXE enabled platform, TXEMANUF calls the test. Otherwise, it reports the result or the failure message from the previous test.

Intel TXEMANUF tools report the result or cause a reboot. If there is a reboot, Intel TXEMANUF should be run again. **VSCCCOMN.bin** is required to verify the VSCC entry on the platform. This file must be in same folder as the TXEMANUF executable or TXEMANUF reports an error. **ISHLib.dll** is required to verify ISS self test and functionality on the platform. This file must be in same folder as the TXEMANUF executable or TXEMANUF reports an error.

5.3 Usage

The UEFI version of the tool can be operated using the same syntax as the Windows* version. The Windows* version of the tool can be executed by:

```
TXEManufWin.exe [-EXP] [-H|?] [-VER] [-S0] [-F] [-TEST]
                [-EOL] [-CFGGEN] [-VERBOSE] [-PAGE] [-NONFC]
                [NFC] [-NOISH] [-ISH]
```

Table 15. Tool Options

Option	Description
No option	There are differences depending on the firmware SKU type the system is running on: If BIST test result is not displayed after BIST test is done, the tool needs to be run again (with or without any BIST related argument combinations) to retrieve the result, once test result is displayed, it will be cleared. Tool is capable of remembering whether/what tests (including host based tests) have been run from previous invocation. Host based tests will be run for all cases (whether it's retrieving test result or run the actual BIST). Currently there is one host based tests which is VSCC Table validation check. When using –verbose, TXEManuf displays the list of all the tests that have been run and retrieved.
-EXP	Shows examples of how to use the tools.
-H or -?	Displays the help screen.
-VER	Shows the version of the tools.
-S0	The same as No option, except that there is no power reset/hibernation performed at the end of the BIST test. The test result is reported back right after the test is done and cleared.
-F <filename>	Load customer defined .cfg file
-TEST <level>	Run full test
-EOL <Var|Config> -F <filename>	This option runs several checks for the use of OEMs to ensure that all settings and configurations have been made according to Intel requirements before the system leaves the manufacturing process. The check can be configured by the customer to select which test items to run and their expected value (only applicable for Variable Values, FW Version, and BIOS Version). The sub option `config` or `var` is optional. Using `-EOL` without a sub option is equivalent to the `–EOL config`. When –f flag is used along with a file name, the tool will load the file as the configuration file, instead of using TXEManuf.cfg.
-CFGGEN <filename>	Use this option along with a filename to generate a default configuration file. This file (with or without modification) can be used for the `-EOL` option. Rename it **TXEManuf.cfg** before using it. It is highly recommended to use this option to generate a new **TXEManuf.cfg** with an up-to-date variable names list before using the Intel TXEManuf End-Of-Line check feature.
-VERBOSE <file>	Displays the debug information of the tool or stores it in a log file.
–PAGE	When it takes more than one screen to display all the information, this option lets the user pause the display and then press any key to continue on to the next screen.
-NONFC	Skip NFC Test in full/runtime tests.
-NFC	Force NFC Test in full/runtime tests.
-NOISH	Skip ISS tests

Option	Description
-ISH <command> <options>	Triggers the ISS test. Where command include: -h|?: help information -Test <Level>: execute selftest -EXP: valid command line example Where options include: -page : enable paging output -verbose: enable verbose output Test level varies (0,1,2,3) Level 0 - ISS FW aliveness test. Level 1 - Level 0 test + connectivity test. Level 2 - Level 1 test + calibration test. Level 3 - Level 2 + Sensor BIT for each sensor that supports build-in selftest. If a sensor does not have a BIST support and the requested level is 3 the FW shall run a connectivity test instead and return a warning that this sensor does not support self test.

5.3.1 Host Based Tests

TXE/BIOS VSCC validation, Intel TXEManuf verifies that flash SPI ID on the system is described in VSCC table. If found, VSCC entry for relevant SPI part should match the known good values that pre-populated in the file.

Intel® TXE state check, Intel TXEManuf verifies Intel® TXE is in normal state. This is done by checking the value of 4 fields (initialization state, mode of operation, current operation state, and error state) in FW status register1. If any of these fields indicates Intel® TXE is in abnormal state, Intel TXEManuf will report error without running BIST test.

5.3.2 ISS Manufacturing Tests

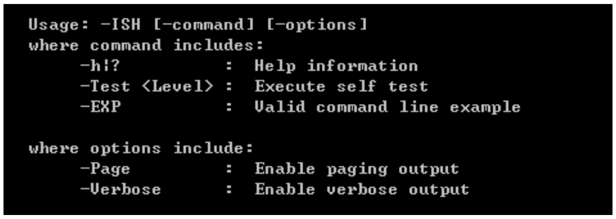
On the top of TXE related test, TXEManuf is recommended tool for production testing of ISS firmware provided by Intel.

It support ISS firmware aliveness tests, Algorithm sanity check, sensor connectivity, and 2nd level diagnostic which is Sensor part BIST.

Unless option specify otherwise, TXEManuf detect which is installed, detected which firmware features are enabled and runs only the appropriate tests.

With TXEManuf –h, you will see all the options for ISS.

Figure 25. TXEManuf for ISS



```
Usage: -ISH [-command] [-options]
where command includes:
     -h|?           :  Help information
     -Test <Level> :  Execute self test
     -EXP           :  Valid command line example

where options include:
     -Page          :  Enable paging output
     -Verbose       :  Enable verbose output
```

Table 16. ISS Tests List

Option	Description
-ISH	Runs only the ISS tests. (without the other TXEManuf tests) From ISS perspective, this is equivalent to –ISH Test 1
-ISH EXP	Prints an example for usage of ISS TXEManuf command line
-ISH Test <Level>	Run a specific ISS Test, Level is an integer parameter: Level 0 = Test basic aliveness. Level 1 = Level 0 + Basic connectivity to each sensor Level 2 = Level 1 + Calibration check for sensor that must have calibration. Level 3 = Level 2 + Sensor BIST for each sensor that support BIST.(Note: this is vendor specific and may not be supported) If devices do not support BIST, the ISS firmware shall run Level2 test only. **Note:** Level 0 - Alivenss only · Not check anything in the HW. · Check only sensors dependency by SW · Every virtual sensor knows that he depends on physical or other virtual sensors so this test check that all mandatory sensors that it depends on are configured in HDT. Level 1 – Connectivity · HW Checking against the physical sensors · the udriver working physically with the sensor by running self-test function to basically verify connection to expected behavior Level2 – Calibration · Every physical sensor required calibration data · This test check that every sensor that require calibration data has its own calibration data (configured in SDT tables) · The test performed by checking that calibration data length in PDT > 0 Level3 – Self test by Sensor · Advanced test for physical sensor that can be implemented by udriver. This test depends on the udriver self-test implementation.
-Page	Enable paging output when option include it
-Verbose	Enable verbose output when option include it

5.4 Intel TXEMANUF –EOL Check

TXEMANUF –EOL check is introduced in the Cherry Trail SoC Family platform to give customers the ability to check Intel® TXE-related configuration before shipping. There are two sets of tests that can be run: variable check and configuration check. Variable check is very similar as FPT –compare option.

5.4.1 TXEMANUF.cfg File

The **TXEMANUF.cfg** file includes all the test configurations for TXEMANUF –EOL check. It needs to be at the same folder that TXEMANUF is run. If there is no **TXEMANUF.cfg** file on that folder, TXEMANUF –EOL config runs the Intel recommended default check only.

Here is an example of the **TXEMANUF.cfg** file:

```
// The end-of-line checks are broken into two categories. One is
// Variable Check, and the other is Configuration Check. If either
// of these check fails, by default TXEManuf will report error and
// continue on to the next check. If a user doesn't wish to continue
// when an error is found, ErrAction field can be used. Please see
// the examples here for detailed explanation:
//
//    SubTestName="TXE VSCC check", ErrAction="ErrorStop"
//
// If the above test fails, TXEManuf will report error and stop. There
// are total of three different error actions user can choose from:
//
// ErrorContinue - report error and continue on to the next check
// ErrorStop - report error and stop any check after the current one
// WarnContinue - report warning and continue on to the next check
//
// To add comment or take out a specific test, leave // at the start
// of a line. This file is processed by TXEManuf line by line as text
// file. Duplication of the same sub-tests are allowed, but TXEManuf
// will always perform the last test to the first test from the file.

// All string comparisons given in this file are case insensitive
// compare. There might be multiple field name/value pairs in one
// entry, but each field needs to be specified in the following
// format where <field name> can be replaced by SubTestName, ReqVal
// or ErrAction, <field value> can be replaced by any string including
// dash and/or spaces surrounded by double quotation marks, or hex-
// decimal number(s) that not surrounded by double quotation marks.
// In case of numeric value, each value (without 0x prefix) needs to
// be specified in byte and eliminated by spaces if there are multiple
// bytes. No line Wrapping is supported:
//
//    <field name>="<field value>", such as ReqVal=" ", or
//    <field name>=<numeric value>, such as ReqVal=78, or
//    <field name>=<numeric value>, such as ReqVal=01 0A 0F FE 7B CD

//////////////////////////////////////////////////////////////////////////
////
// Intel recommends default end-of-line checks includes the following
// list. If a user chooses to use his/her own version of TXEManuf.cfg
// to skip or modify the error action of these checks as WarnContinue,
// TXEManuf will report failure with warnings when these checks are
skipped,
// or have errors. It's suggested that a user should perform these
Intel(R)
// recommended check on all type of SKUs.

SubTestName="EOP status check"
SubTestName="TXE VSCC check"
SubTestName="BIOS VSCC check"
SubTestName="TXE Manufacturing Mode status"
SubTestName="Flash Region Access Permissions"
SubTestName="CF9GR lock check"
SubTestName="FPF Global Valid bit check"
// SubTestName="Security Descriptor Override (SDO) check"
// SubTestName="Validate Keybox Provisioning"

////////////////////////////////////////////////////////////////////////
///////
// The following Configuration Check requires a user to enter an expected
// value after ReqVal=, otherwise the lines without ReqVal field values
will
// be ignored.
//
// Please note that GBE version check will be skipped if Intel Gbe region
// is not present in SPI image.
//
// TXE FW version is a string as <major ver>.<minor ver>.<hotfix
ver>.<build num>
// GBE version is a string as <major ver>.<minor ver>.<revision ver>
// BIOS version is string that vendor specific
////////////////////////////////////////////////////////////////////////
///////

// SubTestName="TXE FW version", ReqVal=
// SubTestName="BIOS version", ReqVal=
// SubTestName="OEM Public Key Hash FPF", ReqVal=
// SubTestName="Perform Secure Boot FPF", ReqVal=
// SubTestName="Key Manifest ID FPF", ReqVal=
// SubTestName="PTT FPF", ReqVal=
// SubTestName="Alternative BIOS Limit FPF", ReqVal=
// SubTestName="OEM Unique Device ID FPF", ReqVal=

//////////////////////////////////////////////////////////////////////
// Variable Check - user needs to put an expected value after ReqVal,
// otherwise the lines without ReqVal field values will be ignored
//
// There are variables that stored in encrypted format. When comparing
// with these variables, ReqVal can only specified as numeric values
// (in encrypted form) in byte order as mentioned above. ReqVal needs
// to be surrounded by double quotation marks if they are string input.
//
// To get an up-to-dated TXEManuf.cfg with a complete variable names
list,
// please run TXEManuf -cfggen <filename>. Please note that variables
// that have # need to be replace by a number. Here defines the number:
//
// Note: The '#' for hash variables should be replaced with an entry
index.
//      The valid range is 0 to 22.
//
```

```
// !!! Please be sure to disable sending EOP or leave platform in ME
// !!! manufacturing mode to run this test, otherwise TXEManuf will
// !!! report failure because this feature is only available in factory
// !!! mode environment.
////////////////////////////////////////////////////////////////////////

// SubTestName="Allow Unsigned Assert Stolen", ReqVal=
// SubTestName="FeatureShipState", ReqVal=
// SubTestName="Flash Protection Override Policy Hard", ReqVal=
// SubTestName="Flash Protection Override Policy Soft", ReqVal=
// SubTestName="FW Update OEM ID", ReqVal=
// SubTestName="Intel (R) Anti-Theft BIOS Recovery Timer", ReqVal=
// SubTestName="Intel (R) Anti-Theft Technology Permanently Disabled?",
ReqVal=
// SubTestName="Intel (R) Dynamic Application Loader Permanently
Disabled?", ReqVal=
// SubTestName="Near Field Communication Enabled", ReqVal=
// SubTestName="Near Field Communication I2C Address", ReqVal=
// SubTestName="ODM ID used by Intel (R) Services", ReqVal=
// SubTestName="OEM_TAG", ReqVal=
// SubTestName="OEMSkuRule", ReqVal=
// SubTestName="PAVP Permanently Disabled?", ReqVal=
// SubTestName="Permit Period Timer Resolution", ReqVal=
// SubTestName="Reserved ID used by Intel (R) Services", ReqVal=
// SubTestName="System Integrator ID used by Intel (R) Services", ReqVal=
// SubTestName="TXEI TXE Region Unlockable", ReqVal=
```

Lines which start with // are comments. They are also used to inform users of the available test group names and the names of specific checks that are included in each test that Intel TXEManuf recognizes.

To select, which test items to run: Create a line that begins with SubTestName="<specific sub test name>".

Here are some other examples that explain how to use this feature:

- To run an Intel TXE version check defined under "Platform Configuration Checkings", a valid Intel TXE version should be equal to string 2.0.0.1008:

  ```
  SubTestName="TXE version", Reqval="2.0.0.1008"
  ```

Note: When running Widevine manufacturing flow, uncomment SubTestName “Validate Keybox Provisioning” from proper EOL testing.

5.4.2 TXEMANUF –EOL Variable Check

TXEMANUF –EOL var check is designed to check the Intel® TXE settings on the platform before shipping. To minimize the security risk in exposing this in an end-user environment, this test is only available in Intel® TXE manufacturing mode or No EOP Message Sent.

Note: -EOL var check. The system must be in Intel® TXE manufacturing mode when **-**EOL var check is run or No EOP Message Sent.

5.4.3 TXEMANUF –EOL Config Check

TXEMANUF –EOL Config check is designed to check the Intel® TXE-related configuration before shipping. Running Intel-recommended tests before shipping is highly recommended.

Table 17. TXEMANUF - EOL Config Tests

Test	Expected Configuration
EOP status check	Enabled
Intel® TXE VSCC check	Set according to the Intel-recommended value
BIOS VSCC check	Set according to the Intel-recommended value
Intel® TXE Manufacturing Mode status	Disabled
Flash Region Access Permissions	Set according to the Intel-recommended value
CF9GR lock check	Locked
FPF Global Valid bit check	Set
Flash Descriptor Security Override (FDSO) check (GPIO_SUS[5])	Disabled
Validate Keybox Provisioning	Not Provisioned (only Provisioned for Android* platform)
NOTE: **–EOL Config** check. If the system is in Intel® TXE manufacturing mode when **–EOL Config** check is run there will be an error report or No EOP Message Sent.	

5.4.4 Output/Result

The following test results can be displayed at the end-of-line checking:

- Pass – all tests passed
- Pass with warning – all tests passed except the tests that were modified by the customer to give a warning on failure. (This modification does not apply to Intel-recommended tests
- Fail with warning - all tests passed except some Intel-recommended tests that were modified by the customer to give a warning on failure.
- Fail - any customer-defined error occurred in the test.

5.5 Examples

5.5.1 TXEMANUF Running on a Full Image with Some BIST Failures Intel® TXE FW Platform

```
TXEMANUF –verbose
```

```
Intel(R) TXEManuf Version: 2.0.0.1003
Copyright(C) 2007 - 2014, Intel Corporation. All rights reserved.
```

```
FW Status Register1: 0x1F0000D5
FW Status Register2: 0x60000000

  CurrentState:                         Normal
  ManufacturingMode:                    Enabled
  TXEMemoryInvalid:                     Valid
  OperationalState:                     Power Gated
  InitComplete:                         Initializing
  BUPLoadState:                         Success
  ErrorCode:                            No Error
  ModeOfOperation:                      Normal
  Phase:                                HOSTCOMM Module

Get FWU info command...done

Get FWU version command...done

Get FWU feature state command...done

Get TXE FWU platform type command...done

Get TXE FWU feature capability command...done
Feature enablement is 0xA0101060
gFeatureAvailability value is 0x1
Intel(R) TXEI device is found to be disabled

Request Intel(R) TXE test result command...done

TXE initialization state valid
TXE operation mode valid
Current operation state valid
TXE error state valid
Verifying FW Status Register1...done

Request Intel(R) TXE test result command...done
vsccommn.bin was created on 18:45:07 03/11/2013 GMT
SPI Flash ID #1 TXE VSCC value is 0x2025
SPI Flash ID #1 (ID: 0xEF6017) TXE VSCC value checked

Error 9271: Flash ID 0xEF6017 Intel(R) BIOS VSCC value mismatch
Programmed value of 0x2005 doesn't match the recommended value of
0x2025
See PCH SPI programming Guide for more details
FPBA value is 0x0

Request Intel(R) TXE Runtime BIST test command...done

Get Intel(R) TXE test data command...done
Total of 4 Intel(R) TXE test result retrieved

MicroKernel - Internal Hardware Tests: Internal Hardware Tests -
Passed

NFC - General: NFC basic configuration - Passed
NFC - General: I2C connection - Passed
NFC - General: Reset pin - Failed
```

```
Error 9372: NFC reset pin failure. Check physical reset pin
connection.

Clear Intel(R) TXE test data command...done

Error 9296: TXEManuf Test Failed
```

§

6 Intel® TXEInfo

Intel® TXEInfoWin and Intel® TXEInfo provide a simple test to check whether the Intel® TXE FW is alive or not. Both tools perform the same test; query the Intel® TXE FW and retrieve data.

It contains a list of the data that each tool returns.

The Windows* version of TXEInfo (TXEInfoWin) requires administrator privileges to run under Windows* OS. The user needs to use the **Run as Administrator** option to open the CLI in Windows* 8 64/32 bit and Windows* 8 SoC.

TXEInfoWin and Intel TXEInfo serve two purposes:

- It provides a means by which the Intel® TXE (Trusted Execution Engine) functionality can be determined (i.e. if it is “alive”).
- It displays a variety of information about the Intel® TXE and Intel® TXE components including versions, capabilities, and functionality

6.1 Windows* PE Requirements

In order for tools to work under the Windows* PE environment, you must manually load the driver with the .inf file in the Intel® TXEI driver installation files. Once you locate the .inf file you must use the Windows* PE cmd `drvload ipc.inf` to load it into the running system each time Windows* PE reboots. Failure to do so causes errors for some features.

ISHLib.dll is required to verify ISS functionality on the platform. This file must be in same folder as the TXEInfo executable or TXEInfo reports an error.

6.2 Usage

The executable can be invoked by:

```
TXEInfoWin.exe [-EXP] [-H|?] [-VER] [-FEAT] [-VALUE]
               [-FWSTS][-VERBOSE] [-PAGE] [-PID] [-DUMPIDLM]
               [-NOISH] [-ISH]
```

```
TXEInfo.efi [-EXP] [-H|?] [-VER] [-FEAT] [-VALUE]
            [-FWSTS][-VERBOSE] [-PAGE] [-PID] [-DUMPIDLM ]
            [-NOISH] [-ISH]
```

Table 18. Intel® TXEInfo Command Line Options

Option	Description
No option:	If the tool is invoked without parameters, it reports information for all components listed in Tablet 19 below.

Option	Description
-EXP	Shows examples about how to use the tools.
-H or -?:	Displays the list of command line options supported by the Intel® TXEInfo tool.
-VER	Shows the version of the tools.
-FEAT <name>	Retrieves the current value for the specified feature. If the feature name is more than one word, the entire feature name must be enclosed in quotation marks. The feature name entered must be the same as the feature name displayed by Intel TXEInfo. Intel TXEInfo can retrieve all of the information detailed below. However, depending on the SKU selected, some information may not appear.
-VALUE <value>	Compares the value of the given feature name with the value in the command line. If the feature name or value is more than one word, the entire name or value must be enclosed in quotation marks. If the values are identical, a message indicating success appears. If the values are not identical, the actual value of the feature is returned. Only one feature may be requested in a command line.
–FWSTS	Decodes the Intel® TXE FW status register value field and breaks it down into the following bit definitions for easy readability:

```
FW Status Register1: 0x1F000255
FW Status Register2: 0x69000004
CurrentState:        Normal
ManufacturingMode:   Enabled
TXEMemoryInvaild:    Valid
OperationalState:    M0 with UMA
InitComplete:        Complete
BUPLoadState:        Success
ErrorCode:           No Error
ModeOfOperation:     Normal
Phase:               HOSTCOMM module
```

Option	Description
-VERBOSE <filename>	Turns on additional information about the operation for debugging purposes. This option has to be used together with the above mentioned option(s). Failure to do so generates the error: "Error 9254: Invalid command line option". This option works with no option and `-feat`.
-PAGE	When it takes more than one screen to display all the information, this option lets the user pause the display and then press any key to continue on to the next screen.
-PID <filename>	Append/Export Platform ID to the binary file
-DUMPIDLM<filename>	Displays Platform ID list in an IDLM binary
-NOISH	Do not display any information related to ISH

Option	Description
-ISH <command> <options>	Display ISH information Where command include: -SensorInfo: sensor information h|?: help information -FWStat: ISS FW status -EXP: valid command line example Where options include: -page : endable paging output -verbose: enable verbose output

Table 19. Components Lists Displayed in Intel® TXEInfo

Feature Name	Feature Data Source	Supported SKUs	Supported OS	Specific Feature Dependency	Field Value
Tools Version	SW (TXEInfo)	Both	All	N/A	Version string Example: 2.x.y.ZZZZ; where x=minor, y = HF/MR, ZZZZ = Build Number.
FWSTS	Intel® TXE Kernel	Both	All	N/A	Two 32bit Hexadecimal numbers and their bit definition breakdown (only available when –verbose is used)
IAFW Version	BIOS/Intel® TXE Kernel	Both	All	N/A	A Version string
VendorID	Intel® TXE Kernel	Both	All	N/A	A number (in Hex)
SoC Version	Intel® TXE Kernel	Both	All	N/A	A Version string
FW Version	Intel® TXE Kernel	Both	All	N/A	A Version string 2.x.y.ZZZZ; where x=minor, y = HF/MR, ZZZZ = Build Number.
TXEI Driver Version	Intel® TXE Kernel	Both	All	NA	A version string
NFC Firmware Version	NFC GUID	Both	All	NA	A version string. If NFC HW device is not found/accessible, display “Not Available”

Feature Name	Feature Data Source	Supported SKUs	Supported OS	Specific Feature Dependency	Field Value
NFC Radio Type	NFC	Both	All	NA	A version string. If NFC HW device is not found/accessible, display “Not Available”
FW Capabilities	Intel® TXE Kernel	Both	All	N/A	Combination of feature name list breakdown (with a Hexadecimal value) *This is a display of the Feature State for the Intel® TXE. Is enabled / disabled on the system. Each bit in the value represents a feature state. Intel® TXE features including PTT and Anti-theft technology etc.
Last Intel® TXE Reset Reason	Intel® TXE Kernel	Both	All	N/A	Power up/ Firmware reset/ Global system reset/ Unknown
Local FWUpdate	Intel® TXE Kernel	Both	All	N/A	Enabled/Disabled / Password Protected
BIOS Config Lock	Other (Directly reading from SPI)	Both	All	N/A	Enabled/Disabled /Unknown If shown as enabled, FLOCKDN for BIOS is set. If shown as disabled, FLOCKDN for BIOS is not set.
Host Read Access to Intel® TXE	Other (Directly reading from SPI)	Both	All	N/A	Enabled/Disabled / Unknown

Feature Name	Feature Data Source	Supported SKUs	Supported OS	Specific Feature Dependency	Field Value
Host Write Access to Intel® TXE	Other (Directly reading from SPI)	Both	All	N/A	Enabled/Disabled / Unknown
SPI Flash ID	Other (Directly reading from SPI)	Both	All	Only when there are flash parts HW installed	A JEDEC ID number (in Hex)
TXE/BIOS VSCC register values	Other (Directly reading from SPI)	Both	All	Only when there are flash parts HW installed	A 32bit VSCC number (in Hex)
BIOS Boot State	Intel® TXE Kernel	Both	All	N/A	Pre Boot/ In Boot/ Post Boot
OEM Id	Intel® TXE Kernel	Both	All	Only if fw image supports OEM Id	UUID for OEM to check during FW Update
OEM Tag	Intel® TXE Kernel	Both	All	N/A	A 32bit Hexadecimal number
Global Valid FPF	Intel® TXE Kernel/FPF	Both	All	N/A	Valid/Invalid
PTT FPF	Intel® TXE Kernel/FPF	Both	All	N/A	Enabled/Disabled
Perform Secure Boot FPF	Intel® TXE Kernel/FPF	Both	All	N/A	Enabled/Disabled
OEM Public Key Hash FPF	Intel® TXE Kernel/FPF	Both	All	N/A	A 32 byte number (in Hex)
Key Manifest ID FPF	Intel® TXE Kernel/FPF	Both	All	N/A	A 8bit number (in Hex)
Alternative BIOS Limit FPF	Intel® TXE Kernel/FPF	Both	All	N/A	A 16bit number (in Hex)
OEM unique Device ID FPF	Intel® TXE Kernel/FPF	Both	All	N/A	A 64bit number (in Hex)
Secure Boot Status	Intel® TXE Kernel/FPF	Both	All	N/A	Not Executed/ Executed
Secure Boot Recovery Status	Intel® TXE Kernel	Both	All	N/A	Not Executed/ Executed
PTT Lockout Override Counter	Intel® TXE Kernel	Both	All	N/A	A Number
Keybox	Other (Directly	Both	All	N/A	Not Provisioned/

Feature Name	Feature Data Source	Supported SKUs	Supported OS	Specific Feature Dependency	Field Value
	reading from SPI)				Provisioned

6.3 Examples

This is a simple test that indicates whether the FW is alive. If the FW is alive, the test returns device-specific parameters. The output is from the Windows* version.

6.3.1 Dump Full Detail Info about Intel® TXE and its Application Feature Values

```
TXEINFOWIN.exe

Intel(R) TXEInfo Version: 2.0.0.1003
Copyright(C) 2007 - 2014, Intel Corporation. All rights reserved.

Intel(R) TXE code versions:

BIOS Version:                           Alpha 1.04
VendorID:                               8086
SOC Version:                            5
FW Version:                             2.0.0.1057
TXEI Driver Version:                    2.0.0.1054
Get NFC Versions command...done
NFC FW Version:                         2.10
NFC Radio Type:                         NXP

FW Capabilities:                        0xA0101060

    Intel(R) Anti-Theft Technology - PRESENT/ENABLED
    Intel(R) Capability Licensing Service - PRESENT/ENABLED
    Protect Audio Video Path - PRESENT/ENABLED
    Intel(R) Dynamic Application Loader - PRESENT/ENABLED

    Intel(R) NFC Capabilities - PRESENT/ENABLED


Last TXE reset reason:                  Power up

Local FWUpdate:                         Enabled


Get BIOS flash lockdown status...done

BIOS Config Lock:                       Enabled


Get flash master region access status...done

Host Read Access to TXE:                Enabled

Host Write Access to TXE:               Enabled

SPI Flash ID #1:                        EF6017

SPI Flash ID VSCC #1:                   20252025

SPI Flash BIOS VSCC:                    20052005

BIOS boot State:                        Post Boot

OEM Id:                                 00000000-0000-0000-0000-
000000000000

Capability Licensing Service:           Enabled

Get TXE FWU OEM Tag command...done

OEM Tag:                                0x00000000

Global Valid FPF:                       Valid

PTT FPF:                                Enabled

Perform Secure Boot FPF:                Enabled

OEM Public Key Hash FPF:
0000000000000000000000000000000000000000000000000000000000000000

Key Manifest ID FPF:                    00

Alternative BIOS Limit FPF:             07DF

Secure Boot Status:                     Not Executed

Secure Boot Recovery Status:            Not Executed

PTT Lockout Override Counter:           10
```

6.3.2 Retrieve Current Value of Flash Version

```
C:\ TXEInfoWin.exe -feat "BIOS boot state"

Intel(R) TXEInfo Version: 2.0.0.1003

Copyright(C) 2007 - 2014, Intel Corporation. All rights reserved.

BIOS boot State: Post Boot

> TXEInfo.efi -feat "BIOS boot state"

Intel(R) TXEInfo Version: 2.0.0.1003

Copyright(C) 2005 - 2012, Intel Corporation. All rights reserved.

BIOS boot State: Post Boot
```

6.3.3 Check if Computer has Completed Setup and Configuration Process

```
C:\ TXEInfoWin.exe -feat "Setup and Configuration" –value "Not Completed"


Intel(R) TXEInfo Version: 2.0.0.1003

Copyright(C) 2007 - 2014, Intel Corporation. All rights reserved.

Local FWUpdate: Success - Value matches FW value.


> TXEInfo.efi -feat "Setup and Configuration" –value "Not Completed"


Intel(R) TXEInfo Version: 2.0.0.1003

Copyright(C) 2007 - 2014, Intel Corporation. All rights reserved.

Local FWUpdate: Success - Value matches FW value.
```

§

7 Intel® TXE Firmware Update

FWUpdate 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.

FWUpdate does not update the BIOS, or Descriptor Regions. It updates the FW code portion that Intel provides on the OEM website. Note that Intel® FWUpdate updates the entire Intel® TXE code area only and keep same data area.

The image file that the tool uses for the update is the same image file that is used by the FITC tool to create a firmware image for use in the SPI. A sample FW image file for updating would be ‘
CHV_SEC_REGION.bin. These file is located in the ‘Image Components\TXE’ sub-folder of the firmware kit package.

FWUpdate takes approximately 1-4 minutes to complete depending on the flash device on the system.

After FWUpdate a host reset is needed to complete FW update. The user can also use the –FORCERESET option to do this automatically.

7.1 Requirements

FWUpdLclWin.exe and FWUpdLclWin64.exe are command line executable that can be run on an Intel® TXE-enabled system that needs updated FW.

FW can only be updated when the system is in an S0 state. FW updates are NOT supported in the S3/S4/S5 state.

Intel® TXE FWUpdate must be enabled in through BIOS.

The Intel® TXE Interface driver must be installed for running this tool in a Windows* environment.

7.2 Windows* PE Requirements

For tools to work under Windows* PE environment, the user will need to manually load a driver by using the .inf file in the Intel® TXEI driver installation files. Once the .inf file located, the user will need to use Windows* PE command `drvload *.inf` to load it into the running system each time Windows* PE reboots. Failure to do so causes a tools reporting error.

7.3 Usage

Note: In this section, <Image File> refers to an Intel-provided image file of the section of the FW to be updated, not the image file used in FITC to program the entire flash memory.

```
FWUpdLclWin.exe  [-H|?] [-VER] [-EXP] [-VERBOSE] [-F] [-Y] [-SAVE]
                 [-FWVER] [-ALLOWSV] [-FORCERESET] [-OEMID] [-GENERIC]


FWUpdLcl.efi   [-H|?] [-VER] [-EXP] [-VERBOSE] [-F] [-Y] [-SAVE]
               [-FWVER] [-ALLOWSV] [-FORCERESET] [-OEMID]
```

Note: Image File is the image file of the FW to be updated. Is the same image file used by FITC.

Table 20. Image File Update Options

Option	Description
-H or -?	Displays the list of command line options supported by the Intel TXEInfo tool.
-VER	Shows the version of the tools.
-EXP	Shows examples about how to use the tools.
-VERBOSE [<FILE>]	Verbose. Enables additional information about the tool's operation to be displayed for debugging purposes.
-F <FILE>	File. Specifies the FWUpdate image file to be used for performing an update.
-Y	Ignore warning. If the warning asks for input "Y/N", this flag makes the tool automatically take "y" as the input.
-SAVE <file>	Restore Point. Retrieves an update image from the FW based on the currently running FW. The update image is saved to the user-specified file.
-FWVER	Display FW version
-ALLOWSV	Allow Same Version. Allows the version of the input FW (based on the file input) to be the same as the version of the FW currently on the platform. Without this option, an attempt to perform an update on the same version will not proceed.
-FORCERESET	Force Reset. The tool automatically reboots the system after the update process with FW is complete. The system reboot is necessary for the new FW to take effect. An attempt to update the FW without this option will end with a message telling the user to reset the platform for the changes to take effect.
-OEMID <UUID>	OEM ID. The tool uses the specified OEM ID during the transaction of the new FW image with the Security Engine. The purpose of the OEM ID is for manufacturers to have an identifier for their system. Using any other OEM ID value other than what is on the FW running on the target platform results in a failure of the FWUpdate process. The full image (including all necessary flash partitions) flashed to the system can be configured with the Flash Image Tool to specify the OEM ID (this tool specifies a default of zeros for the OEM ID.) If this command line option is not used, the default OEM ID used for the update is zeros. The OEM ID is configured in the existing FW image running on the platform. The OEM ID value is specified in the UUID format (8-4-4-4-12).

Option	Description
-GENERIC	Intel® TXEI. Specifies that the tool performs the update over the Intel® TXEI interface. Intel® TXEI is used even if the FW supports a network-based update. **Note**: This option is only supported in the Windows* version of the tool.

7.4 Examples

7.4.1 Updates Intel® TXE with Firmware Binary File

This command updates TXE with FW.BIN file. If the firmware on current platform is newer than then version in FW.BIN file, tools will promote a warning to let user know there will be a firmware downgrade (rollback) event and let user choose Y/N to continue. User can always use –y to skip this warning automatically. If the firmware on the platform is the same as the version in FW.BIN, tools will return an error. User can use –allowsv to allow same version update.

```
FWUpdLclWin.exe –f FW.BIN

EFI:
FWUpdLcl.efi –f FW.BIN


C:\> FWUpdLclWin.exe –f upd.bin –allowsv
Intel (R) Firmware Update Utility version 2.0.0.1003
Copyright (C) 2007-2014, Intel Corporation.  All rights reserved.


Trying to connect to TXEI driver.

Communication Mode: TXEI
Checking firmware parameters...

Warning: Do not exit the process or power off the machine before the
firmware update process ends.
Initiating firmware update process...

Sending the update image to FW for
verification............................................................
..........................
Image successfully sent to FW.
FW verifying the image...

Trying to receive update status...
Trying to connect to TXEI driver.

FW Update is complete and a reboot will run the new FW.
```

Note: The final output message could change as per the reset type required by the update. If the update only requires TXE reset then the success text will be “FW Update is completed successfully” else if the reset type is host or global reset then the success text will be “FW Update is complete and a reboot will run the new FW.” The Reset Type will be automatically determined by the FW on an update.

7.4.2 Display Supported Commands

Display a list of supported command line sequences based on the arguments provided. The arguments relevant for this usage are any of the command line options with the prefix '-' removed. The tool will display all valid command sequences based on the options provided. Below is an example which displays valid command sequences with the -exp option

```
C:\> FWUpdLclWin.exe -exp save

Intel (R) Firmware Update Utility Version: 2.0.0.1003
Copyright (C) 2007 - 2012, Intel Corporation.  All rights reserved.


 The parameters provided are supported in the following command-line
sequences:

  1. SAVE<file> [VERBOSE[<file>]]

 Using -EXP without any additional input will display examples of
 common command-line input.
```

EFI:

```
> FWUpdLcl.efi -exp save

Intel (R) Firmware Update Utility Version: 2.0.0.1003
Copyright (C) 2007 - 2012, Intel Corporation.  All rights reserved.


 The parameters provided are supported in the following command-line
sequences:

  1. SAVE<file> [VERBOSE[<file>]]

 Using -EXP without any additional input will display examples of
 common command-line input.
```

§

8 Calibration Tool

The calibration tool is intend to extract the relevant data (needed for sensor calibration) from the ISS firmware and uses it to calculate the calibration coefficients for the sensor calibration, and burn the new coefficients to the ISS firmware. For more detail, refer to Intel ISH calibration tool user guide release in both FDK and TXE FW kit.

The calibration tool can work in two operation modes:

1. Full calibration mode:

In this mode the calibration tool makes the whole three steps of calibration:

- Extract the needed data for calibration from ISS.
- Calculates the calibration coefficients needed per sensor calibration.
- Burn the calibration coefficients in the ISS firmware image.

2. In conjugation with any other calibration tool:

In this mode the calibration tool will work as extractor of parameters and burner of calibration coefficients into the ISS firmware, the actual calibration task will be done by 3rd party calibration tool– this mode is being supported to make sure ISS can be compatible with various sensors that were not integrated by Intel and are added by customers using the FDK or any other way.

Tools can be used in Windows* and Android* OS environment.

Windows* operation:

The tool has two modes of operations under Windows* OS:

1. Command line version
2. GUI version

Both version are feature identical and are made to be able to support per model and system calibration in development and in the manufacturing line.

Android* operation:

1. Command line version
2. No GUI version available

8.1 Usage

The command line calibration tool executable can be invoked by:

```
WindowsCalibrationTool.exe [-ImportXML <filename>] [-Save <filename>]
```

```
WindowsCalibrationTool.exe –Calibrate <sensor> –HorizontalField <horizontal-magnetic-field> -VerticalField <vertical-magnetic-field> [-LoadDataDump <dump-file-
path>] | DumpData <dump-file-path> [-UpdateFW] [-ExportXml <filename>] [-Save
<filename>]
```

```
WindowsCalibrationTool.exe –CalibrateOffline <sensors> –HorizontalField <horizontal-
magnetic-field> -VerticalField <vertical-magnetic-field> -LoadDataDump <dump-file-
path> -ExportXml <filename>
```

Table 21. Intel® Calibration Command Line Options

Option	Description
-ImportXML <xml-file-path>	Import calibration data from XML file
-Save <xml-file-path>	Save current FW calibration data to XML file
-CalibrateOffline	Run the calibration off line without machine dependency
-Calibrate <sensors>	Run calibration flow on selected sensors
-ExportXML <xml-file-path>	Export the newly calculated calibration data to XML file
-DumpData <dump-file-path>	Dump the data sampled during the calibration to a file
-LoadDataDump <dump-file-path>	Read the sensors data from a file instead of sampling sensors
-HorizontalField <horizontal-magnetic-field>	Horizontal magnetic field measured in this location
-VerticalField <Vertical-magnetic-field>	Vertical magnetic field measured in this location
-UpdateFW	Flush the calculated calibration data to the firmware
-Verbose	Enables verbose output
-NoLog	Disable the automatic log file creation
-H	Show help message

8.2 Examples

Following are some examples of calibration tool CLI usage:

- CalibrationTool.exe -Calibrate motion -HorizontalField 100 -VerticalField 200 -UpdateFW -ExportXml filename.xml –Save filename.xml
- CalibrationTool.exe -Calibrate motion -HorizontalField 100 -VerticalField 200 -UpdateFW -LoadDataDump filename.csv
- CalibrationTool.exe -Save filename.xml
- CalibrationTool.exe -ImportXml filename.xml

§

9 Intel® Manifest Generation Tool

The SPI image creation flow with signed BIOS image included is centered on the Manifest Generation Tool. Intel will deliver the Manifest Generation tool and Manifest Signing tool in FW kit release. Note that SampleSigner is not allowed to be used by customers to generate their production FW image due to legal concerns. Customers need to use their own signing tool and infrastructure for the flow of key pair generation, BIOS signing and key hash generation. Refer to Platform BIOS signing user guide for detail which it's only available for Tablet segment.

The Manifest Generation tool is used to:

- Calculate the Hash of a public key
- Create a Secure boot/Key Manifest

To avoid common errors that may result in a non-bootable IBB, the Manifest Generation tool will return as a unit the IBB and manifests as the result of the last stage in the secure boot manifest creation flow (signature insertion).

The tool will verify as well that the result passes authentication and authorization. In case of any error the tool will return the error and it will not return the IBB and manifests.

The signed BIOS Image with manifest creation stages will be:

1. **Manifest Candidate Creation:**

   Taking as input the IBB, Secure Boot fuses configuration, Public Key, OEM block and other manifest data, the manifest candidate and returns the hash to be signed by the private key.

2. **Singing the Manifest:**

   The hash returned by the Manifest Generation Tool will be signed using the PKCS 1.5 scheme with the private key using the OEM signing infrastructure.

3. **Inserting the signature in the manifest:**

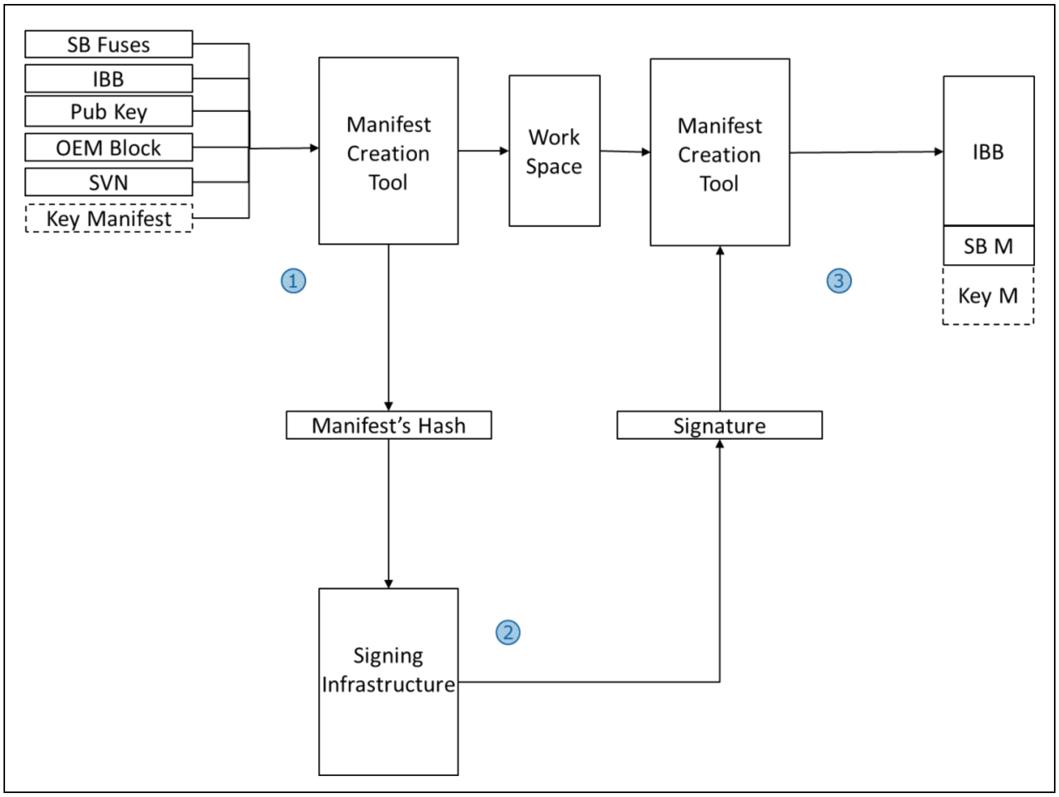
   The Manifest Generation Tool will verify that the signature is OK, then insert it in the manifest candidate making it a valid manifest and return the IBB and the manifest as they should be in the BIOS layout.

4. **Stitching the BIOS:**

   The OEM need to stitch the output of the Manifest Generation Tool with the rest of the BIOS.

The Manifest Generation Tool will verify at every step the coherency of the data and it will fail on error indicating what the problem is. If the Manifest Generation Tool completed successfully the last step of the manifest creation the result MUST successfully pass authentication.

Figure 26. Manifest Generation Flow



9.1 Manifest Generation Tool

FLAMInGo.exe is Windows* based SPI flash Manifest Generation Tool which will be released in FW kit. It can be used to create secure boot Manifest and Key Manifest with command line support only.

To create a SHA256 digest (hash) of a given public key, the executable can be invoked in command line by:

```
FLAMInGo.exe HashKey --out [HashOutputFile] --key [PublicKeyFileToHash]
```

Table 22. Tool Options for Public Key Hash Generation

Option	Description
HashKey	Ask tool to create a SHA256 digest (hash) of a given public key
--out [HashOutputFile]	Name of the file to place the SHA256 digest of the public key
--key [PublicKeyFileToHash]	Public key file to calculate SHA256 form
-?	To displays the list of command line options

Following command example takes the public key form the MyKey.cer file (public and write its hash value to MyKeyHash.txt file:

```
FLAMInGO.exe HashKey --out MyKeyHash.txt --key MyKey.cer
```

To create a partial image manifest and a hash file to sign, the executable can be invoked in command line by:

```
FLAMInGo.exe ImageManCreate --Name [ManifestName] --Fuse [FuseConfigFile]

                    [--Unsigned UnsignedFile] --KeySign [SigningKey] --SVN [SVN]

                    --Type ImageType [--OEMDataFile <OEMDataFile>]

                    --Image ImageFile [--KeyMan <KeyManifestFile>]  [-?]
```

Table 23. Tool Options for Partial Secure Boot Manifest Generation

Option	Description
ImageManCreate	Asks tool to create a partial image manifest and a hash file to sign
--Name ManifestName	String that identifies the manifest, same name must be used when completing the manifest generation process
--Fuse FuseConfigFile	Name of the file that contains the fuses configuration
--Unsigned UnsignedFile	Name of the file that place then unsigned data of the manifest
--KeySign SigningKey	Name of the file that contains the public key of the key that is used to sign the manifest
--SVN SVN	Security Version Number
--Type ImageType	Type of image to sign (IBB or ISH)
--OEMDataFile OEMDataFile	Name of the file that contains OEM data (maximum 400 bytes)
--Image ImageFile	Name of the file that contains image to sign
--KeyMan KeyManifestFile	Name of the file that contains a valid key manifest generated by this tool
-?	To displays the list of command line options

To take an image manifest hash signature and generates an image manifest, the executable can be invoked in command line by:

```
FLAMInGo.exe ImageManComplete --Name [ManifestName] --Fuse [FuseConfigFile]

                                  [--signature SignatureFile] [-?]
```

Table 24. Tool Options for Secure Boot Manifest Generation

Option	Description
ImageManComplete	Asks tool to take a secure boot manifest hash signature and generates a secure boot manifest
--Name ManifestName	String that identifies the manifest, same name must be used when completing the manifest generation process

Option	Description
--Fuse FuseConfigFile	Name of the file that contains the fuses configuration
--signature SignatureFile	Name of the file that contains an RSA signature of the hash file generated when creating a manifest
-?	To displays the list of command line options

To create a partial Key manifest and a hash file to sign, the executable can be invoked in command line by:

```
FLAMInGo.exe KeyManCreate --Name [ManifestName] --Fuse [FuseConfigFile]

            [--Unsigned <UnsignedFile>] --Keysign [SigningKey] --SVN [SVN]

            --Type [ImageType] –KeyCert [PublicKeytKeyFileToCerify] [-?]
```

Table 25. Tool Options for Partial Key Manifest Generation

Option	Description
KeyManCreate	Asks tool to create a partial Key manifest and a hash file
--Name ManifestName	String that identifies the manifest, same name must be used when completing the manifest generation process
--Fuse FuseConfigFile	Name of the file that contains the fuses configuration
--Unsigned <UnsignedtFile>	Name of the file that place then unsigned data of the manifest
--Keysign SigningKey	Name of the file that contains the public key of the key that is used to sign the manifest
--SVN SVN	Security Version Number
--Type ImageType	Type of image to sign (IBB or ISH)
--KeyCert PublicKeyFileToCerify	Name of the file that contains the key to certify by the key manifest
-?	To displays the list of command line options

To take a key manifest hash signature and generates a key boot manifest, the executable can be invoked in command line by:

```
FLAMInGo.exe KeyManComplete --Name [ManifestName] --Fuse [FuseConfigFile]

                      [--signature SignatureFile] [-?]
```

Table 26. Tool Options for Key Manifest Generation

Option	Description
KeyManComplete	Asks tool to take a Key manifest hash signature and generates a Key manifest
--Name ManifestName	String that identifies the manifest, same name must be used when completing the manifest generation process
--Fuse FuseConfigFile	Name of the file that contains the fuses configuration

Option	Description
--signature SignatureFile	Name of the file that contains an RSA signature of the hash file generated when creating a manifest
-?	To displays the list of command line options

To verify if an ingredient image contains a valid manifest and matches the fuse configuration file.

FLAMInGo.exe VerifyImage --Image [ImageFile] --Fuse [FuseConfigFile] [-?]

Table 27. Tool Options for BIOS Image Verification

Option	Description
VerifyImage	Asks tool to verify if given BIOS image contain a valid manifest and match the fuse configuration file or not
--Image ImageFile	Name of the file that contains image to verify
--Fuse FuseConfigFile	Name of the file that contains the fuses configuration
-?	To displays the list of command line options

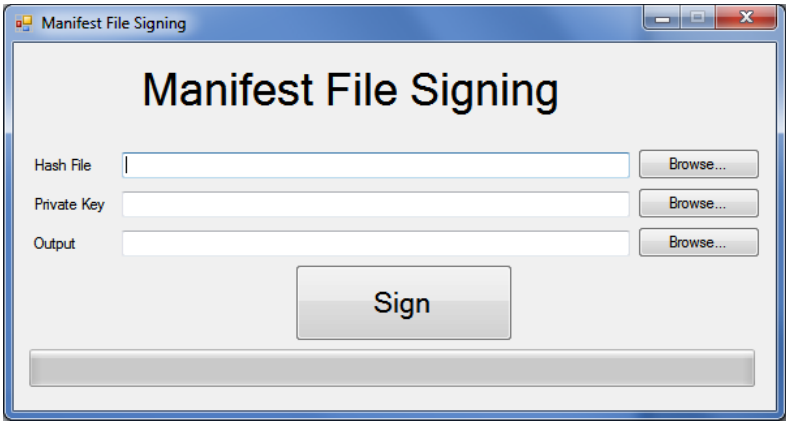
9.2 Signing Tool

SampleSigner.exe is Windows* based tool which will be released in FW kit and used for generating signature for Manifest Hash file by taking Private Key. This tool support both GUI and command line mode. GUI based dialog shows up if you double click `SampleSigner.exe` from Windows* environment.

Parameters:

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

Figure 27. Manifest File Signing Tool



As for command line mode, the executable can be invoked in command line by:

SampleSigner.exe [HashFileToSign] [PrivateKeyFile] [OutSignatureFile]

To displays the list of command line options supported by:

SampleSigner.exe -?

Option	**Description**
HashFileToSign	The Hash file plan to be signed
PrivateKeyFile	Private key from OEM key pair
OutSignatureFile	The output location path of the signed output hash file
-?	To displays the list of command line options

9.3 Example Command

9.3.1 Secure Boot Manifest Creation

FLAMInGO.exe SBManCreate Fuses.txt MySBManifest IBB.bin 1 MySBSigningKey.cer -OEMDataFile myOEMdata.bin -KeyManifestFile MyManifest_manifest.bin

The above example command starts the secure boot manifest creation process. It uses the fuse configuration from the Fuses.txt file, and it certifies the IBB data contained in the IBB.bin file the SVN is 1.

Secure Version Number (SVN) is just a monotonic increasing unsigned integer. The SVN should be incremented only if a critical bug that compromises very sensitive assets was fixed and the usage of previous version should be avoided.

MySBSigningKey.cer file contains the key that is used to sign the manifest. The name for the manifest is MySBManifest and should be used when completing the manifest creation process. This command creates a file call MySBManifest_Hash.bin that contains the hash of the manifest to be signed.

This command will add OEM data from the myOEMdata.bin file to the manifest as well. The manifest will and contains the previously created Key Manifest that is in the MyManifest_manifest.bin file. The last two fields (OEMdata and Key Manifest are not mandatory).

Similarly as in the Key Manifest, we need to sign the manifest hash by following command:

SampleSigner.exe MySBManifest_Hash.bin MySBSigningKey.cer MySBManifest_signature.bin

Once we have the signature, we can complete the process by following command:

FLAMInGO.exe SBManComplete Fuses.txt MSByManifest

MySBManifest_signature.bin

This above command creates a secure boot manifest. It'll use the fuse configuration from the Fuses.txt file - make sure it's the same file used in the previous FLAMInGO command. It uses the same name - MySBManifest, and reads the signature from the MySBManifest_signature.bin created in the last step. Once this command completes, a new file will be created - MSByManifest_manifest.bin which contains the secure boot manifest itself.

9.3.2 Key Manifest Creation

FLAMInGO.exe KeyManCreate Fuses.txt MyManifeste MyKey.cer 1 MySigningKey.cer

This above example command will start the key manifest creation process. It'll use the fuse configuration from the Fuses.txt file, the key to be certified the key in the MyKey.cer file, the SVN is 1.

The MySigningKey.cer file contains the key that is used to sign the manifest. The name for the manifest is MyManifest and should be used when completing the manifest creation process. This command will create a file call MyManifest_Hash.bin that contains the hash of the manifest to be signed.

Now we need to sign the manifest hash, we can use the SampleSigner to sign or any other standard signing infrastructure.

SampleSigner.exe MyManifest_Hash.bin MySigningKey.cer MyManifest_signature.bin

This above command will sign the hash file with the private key from the MySigningKey.cer file. The signature will be written to the MyManifest_signature.bin file. We will use this file as input to complete the manifest creation process.

FLAMInGO.exe KeyManComplete Fuses.txt MyManifest MyManifest_signature.bin

This above command will create a key manifest. It'll use the fuse configuration from the Fuses.txt file - make sure it's the same file used in the previous FLAMInGO command. It uses the same name - MyManifest, and reads the signature from the MyManifest_signature.bin created in the last step. Once this command completes, a new file will be created - MyManifest_manifest.bin which contains the manifest itself.

§

Appendix A Fixed Offset Variables

This appendix only covers fixed offset variables that are directly available to FPT and FPTW. A complete list of fixed offset variables can be found in the *Firmware Variable Structures for Intel® Trusted Execution Engine*. All of the fixed offset variables have an ID and a name. The `-fovs` option displays a list of the IDs and their respective names. The variable name must be entered exactly as displayed below.

Table 28. Fixed Offset Item Descriptions

<table>
<tr><th>Fixed Offset Name</th><th>FPT ID</th><th>Fixed Offset ID</th><th>Description</th><th>Data Length (in Bytes)</th><th>Expected Value</th><th>Secure</th><th>Reset Type</th></tr>
<tr><td colspan="8">Non-Application Specific Fixed Offset Item Descriptions</td></tr>
<tr><td>OEM Sku Rule</td><td>7</td><td>0x000A</td><td>UINT32 (little endian) value. This controls what features are permanently disabled by OEM.

<b>Note:</b>
There are reserved bits that the must not be changed for proper platform operation. The user should only modify the bit(s) for the feature(s) they wish to change. There is NO ability to change features one at a time. This FOV sets OEM Permanents Disable for ALL features. In addition prior updating or changing any of available setting it is highly recommended that the user first retrieves the current OEM Sku Rule and toggling only the desired bits, and then resave them.

This will not enable functionality that is not capable of working in the target hardware SKU. See the respective Firmware Bring-up Guide with what firmware bundle and Hardware SKU of Intel Cherry Trail SoC.</td><td>4</td><td>Feature Capable: 1
Feature Permanently disabled: 0
<table>
<tr><th>Bit</th><th>Description</th><th>Notes</th></tr>
<tr><td>31</td><td>Near Field Communications</td><td></td></tr>
<tr><td>30</td><td>Reserved</td><td></td></tr>
<tr><td>29:22</td><td>Reserved</td><td></td></tr>
<tr><td>21</td><td>Reserved</td><td></td></tr>
<tr><td>20</td><td>DAL</td><td></td></tr>
<tr><td>19</td><td>Reserved</td><td></td></tr>
<tr><td>18</td><td>Reserved</td><td></td></tr>
<tr><td>17</td><td>Reserved</td><td></td></tr>
<tr><td>16</td><td>Reserved</td><td></td></tr>
<tr><td>51:13</td><td>Reserved</td><td></td></tr>
<tr><td>12</td><td>PAVP</td><td></td></tr>
<tr><td>11:6</td><td>Reserved</td><td></td></tr>
<tr><td>5</td><td>Reserved</td><td></td></tr>
<tr><td>4:3</td><td>Reserved</td><td></td></tr>
<tr><td>2</td><td>Reserved</td><td></td></tr>
<tr><td>1</td><td>Reserved</td><td></td></tr>
<tr><td>0</td><td>Reserved</td><td></td></tr>
</table>
</td><td><b>No</b></td><td><b>Global</b></td></tr>
</table>

<table>
<tr><th>Fixed Offset Name</th><th>FPT ID</th><th>Fixed Offset ID</th><th>Description</th><th>Data Length (in Bytes)</th><th>Expected Value</th><th>Secure</th><th>Reset Type</th></tr>
<tr><td>Feature Shipment Time State</td><td>8</td><td>0x000B</td><td>UINT32 (little endian) value. This controls what features are enabled or disabled. These features may be enabled / disabled by mechanisms such as provisioning. This setting is only relevant for features NOT permanently disabled by the OEM Permanent Disable.
This will not enable functionality that is not capable of working in the target hardware SKU. See the respective Firmware Bring-up Guide with what firmware bundle and Hardware SKU of Intel Cherry Trail SoC
<b>NOTE:</b> There are reserved bits that the must not be changed for proper platform operation. The user should only modify the bit(s) for the feature(s) they wish to change. There is NO ability to change features one at a time. This FOV sets OEM Permanents Disable for ALL features. In addition prior updating or changing any of available setting it is highly recommended that the user first retrieves the current Feature Shipment Time State and toggling only the desired bits, and then resave them.</td><td>4</td><td>Feature Capable: 1
Feature Permanently disabled: 0
<table>
<tr><td>Bit</td><td>Description</td><td>Note</td></tr>
<tr><td>31:30</td><td>Reserved</td><td></td></tr>
<tr><td>29</td><td>PTT</td><td>1</td></tr>
<tr><td>28:3</td><td>Reserved</td><td></td></tr>
<tr><td>2</td><td>Reserved</td><td></td></tr>
<tr><td>1:0</td><td>Reserved</td><td></td></tr>
</table>
<b>NOTE:</b> Bit 29 is only applicable to TXE Firmware.</td><td><b>No</b></td><td><b>Global</b></td></tr>
</table>

Fixed Offset Name	FPT ID	Fixed Offset ID	Description	Data Length (in Bytes)	Expected Value	Secure	Reset Type
OEM_TAG	34	0x000F	A human readable 32-bit number to describe the flash image represented by value	4	Readable 32 bit hex value identifying the image. Can be empty (Null).	**No**	**TXE**
Revenue Sharing Related FOV Item Descriptions							
ODM_ID		0x5003	FOV used for setting the ODM ID Used by Intel Services **NOTE:** This FOV / NVAR can be set only once. Once it is set in FITC or committed following being set via the FOV, it cannot be changed. Also Note: Unlike most other NVARs, the value cannot be read until it has been set.	4	32-bit value Value 0x00000000 < n < 0xFFFFFFFF	**No**	**TXE**
SystemIntegratorID		0x5004	Used for setting the System Integrator ID used by Intel® Services **NOTE:** This FOV / NVAR can be set only once. Once it is set in FITC or committed following being set via the FOV, it cannot be changed. Also Note: Unlike most other NVARs, the value cannot be read until it has been set.	4	32-bit value Value: 0x00000000 < n < 0xFFFFFFFF	**No**	**TXE**

Fixed Offset Name	FPT ID	Fixed Offset ID	Description	Data Length (in Bytes)	Expected Value	Secure	Reset Type
ReservedID		0x5005	**NOTE:** This FOV / NVAR can be set only once. Once it is set in FITC or committed following being set via the FOV, it cannot be changed. Also Note: Unlike most other NVARs, the value cannot be read until it has been set.	4	32-bit value Value: 0x00000000 < n < 0xFFFFFFFF	No	**TXE**

NOTE: All Fixed Offset Variables (FOVs) have corresponding Named Variables (NVARs) however not all Named Variables (NVARs) have Firmware Offset Variables (FOVs) associated with them.

Additionally some Fixed Offset Variables (FOVs) have different name designations than Named Variable (NVARs) counterparts.

FPT NVAR Retrieve command:

```
fpt.exe –r <name> | all [-f <file>] [options]
```

Required Parameters

<name> Name of NVAR OR All retrieves all the NVARs

FPT FOV / NVAR naming Comparison

Named Variables (NVARs)	Fixed Offset Variables (FOVs)
OEMSkuRule	OEMSkuRule
FeatureShipState	FeatureShipState
OEM_TAG	OEM_TAG
ODM ID used by Intel (R) Services	ODM_ID
System Integrator ID used by Intel (R) Services	SystemIntegratorId
Reserved ID used by Intel (R) Services	ReservedId
All remaining NVARS	All remaining NVARs do not have corresponding FOVs to allow configuration post image creation

§

Appendix B Tool Detail Error Codes

B.1 Common Error Code for all Tools

Error Code	Error Message	Response
0	Success	
1	Memory allocation error occurred	Ensure there is enough memory in the system
2	Invalid descriptor region	Check descriptor region
3	Region does not exist	Check region to be programmed
4	Failure. Unexpected error occurred	Contact Intel
5	Invalid data for Read ID command	Contact Intel
6	Error occurred while communicating with SPI device	Check SPI device
7	Hardware sequencing failed. Make sure that access permissions are correct for the target flash area	Check descriptor region access settings
8	Software sequencing failed. Make sure that access permissions are correct for the target flash area	Check descriptor region access settings
9	Unrecognized value in the HSFSTS register	Unrecognized value in the HSFSTS register
10	Hardware Timeout occurred in SPI device	Hardware Timeout occurred in SPI device
11	AEL is not equal to zero	AEL is not equal to zero
12	FCERR is not equal to zero	FCERR is not equal to zero
25	The host CPU does not have write access to the target flash area. To enable write access for this operation the user needs to modify the descriptor settings to give host access to this region.	Check descriptor region access settings
26	The host CPU does not have read access to the target flash area. To enable read access for this operation the user needs to modify the descriptor settings to give host access to this region.	Check descriptor region access settings
27	The host CPU does not have erase access to the target flash area. To enable erase access for this operation the user needs to modify the descriptor settings to give host access to this region.	Check descriptor region access settings

Error Code	Error Message	Response
28	Protected Range Registers are currently set by BIOS, preventing flash access. Contact the target system BIOS vendor for an option to disable Protected Range Registers.	Assert Flash Descriptor Security Override Strap (GPIO_SUS[5]) to Low, Power Cycle, and Retry. If Protected Range Registers are still set, contact the target BIOS vendor.
50	General Erase failure	Attempt the command again. If it fails again, contact Intel.
51	An attempt was made to read beyond the end of flash memory	Check address
52	An attempt was made to write beyond the end of flash memory	Check address
53	An attempt was made to erase beyond the end of flash memory	Check address
54	The address <address> of the block to erase is not aligned correctly	Check address
55	Internal Error	Contact Intel
56	The supplied zero-based index of the SPI Device is out of range.	The supplied zero-based index of the SPI Device is out of range.
57	AEL or FCERR is not equal to zero for Software Sequencing	AEL or FCERR is not equal to zero for Software Sequencing
75	File not found	Check file location
76	Access was denied opening the file	Check file location
77	An unknown error occurred while opening the file	Verify the file is not corrupt
78	Failed to allocate memory for the flash part definition file	Check system memory Verify the file is not corrupt
79	Failed to read the entire file into memory	Check system memory Verify the file is not corrupt
80	Parsing of file failed	Check system memory Verify the file is not corrupt
100	This error can occur if both Software and Hardware sequencing are not available and the SPI Flash configuration registers are write protected by the Flash Configuration Lock-Down bit (FLOCKDN). Contact the BIOS vendor to unlock this bit or enable hardware sequencing in descriptor mode.	Check with BIOS vendor or SPI programming Guide
101	No SPI flash device could be identified. Verify if Fparts.txt has support for this part	Verify **Fparts.txt** contains device supported.
102	Failed to read the device ID from the SPI flash part	Verify **Fparts.txt** has correct values

Error Code	Error Message	Response
103	There are no supported SPI flash devices installed. Check connectivity and orientation of SPI flash device	Verify **Fparts.txt** has correct values. Check SPI Device
104	The two SPI flash devices do not have compatible command sets	Verify both SPI devices on the system are compatible
105	An error occurred while writing to the write status register of the SPI flash device. This program will not be able to modify the SPI flash	Check SPI Device
202	Confirmation is not received from the user to perform operation.	
203	Flash is not blank	
204	Data verify mismatch found	
205	Unexpected failure occurred	
207	Invalid parameter value specified by user. The option specified cannot be run on a platform with Intel (R) ME Ignition FW	
208	Intel® TXE is disabled	
209	Intel® TXE failed to reset	
210	Requesting Intel® TXE FW Reset failure.	
211	Communications error between FPT and the ME.	
212	The request to disable the ME failed.	
213	Intel® TXE disable is not required	
214	Intel® TXE is already disabled	
215	The attempt to commit the FOVs has failed.	
216	The Close Manufacturing process failed.	
217	Setting Global Reset Failed	
240	Access was denied opening the file	
241	Access was denied creating the file	
242	An unknown error occurred while opening the file	
243	An unknown error occurred while creating	
244	Not a valid file	
245	file not found error	
246	Failed to read the entire file into memory	
247	Failed to write the entire flash contents to file	
248	file already exists	
249	The file is longer than the flash area to write.	
250	The file is smaller than the flash area to write.	
251	Length of image file extends past the flash area.	

Error Code	Error Message	Response
252	Image file not found.	
253	file does not exist	
254	Not able to open the file	
255	Error occurred while reading the file	
256	Error occurred while writing to the file	
280	Failed to disable write protection for the BIOS space	
281	The Enable bit in the LPC RCBA register is not set. The value of this register cannot be used as the SPI BIOS base address.	
282	Failed to get information about the installed flash devices	
283	Unable to write data to flash.	
284	Fail to load driver (PCI access for Windows). The tool needs to run with an administrator privilege account.	
320	FPT General failure error	
321	The address is outside the boundaries of the flash area.	
360	Invalid Block Erase Size value in	
361	Invalid Write Granularity value in	
362	Invalid Enable Write Status Register Command value	
363	Invalid Chip Erase Timeout value	
360	Invalid Block Erase Size value in	
361	Invalid Write Granularity value in	
362	Invalid Enable Write Status Register Command value	
363	Invalid Chip Erase Timeout value	
360	Invalid Block Erase Size value in	
361	Invalid Write Granularity value in	
362	Invalid Enable Write Status Register Command value	
363	Invalid Chip Erase Timeout value	
440	Invalid Fixed Offset variable name	
441	FOV invalid variable ID	
442	Param file is already opened	
443	FOV exists already	
444	Invalid name or Id of FOV	

Error Code	Error Message	Response
445	Invalid length of FOV value. Check FOV configuration file for correct length	
446	Password does not match the criteria.	
447	Error occurred while reading FOV configuration file	
448	Invalid hash certificate file	
449	Valid PID/PPS/Password records are not found in	
450	Invalid ME Manufacturing Mode Done value entered	
451	Unable to get master base address from the descriptor.	
452	Verification of End Of Manufacturing settings failed	
453	End Of Manufacturing Operation failure - Verification failure on ME Manufacturing Mode Done settings	
454	End Of Manufacturing Operation failure - Verification failure on Intel® TXE Manuf counter.	
455	End Of Manufacturing Operation failure - Verification failure on Descriptor Lock settings.	
456	Invalid hexadecimal value entered for the FOV	
457	Parsing of file failed	
480	The setup file header has an illegal UUID	
481	The setup file version is unsupported	
482	Reserved	
483	the given buffer length is invalid	
484	the record chunk count cannot contain all of the setup file record data	
485	the setup file header indicates that there are no valid records (RecordsConsumed >= RecordCount)	
486	the given buffer is invalid	
487	A record entry with an invalid Module ID was encountered.	
488	A record was encountered with an invalid record number.	
489	The setup file header contains an invalid module ID list.	
490	The setup file header contains an invalid byte count.	
491	The setup file record id is not found	
492	The list of data record entries is invalid.	

Error Code	Error Message	Response
493	Reserved	
494	Reserved	
495	The PID is invalid.	
496	The PPS is invalid.	
497	The PID checksum failed.	
498	The PPS checksum failed.	
499	Reserved	
500	Reserved	
501	The data record is missing a PID entry.	
502	The data record is missing a PPS entry.	
503	The header chunk count cannot contain all of the setup file header data.	
504	The requested index is invalid.	
505	Failed to write to the given file.	
506	Failed to read from the given file.	
507	Failed to create random numbers.	
508	The data record is missing a PKI DNS Suffix entry.	
509	The data record is missing a Config Server FQDN entry.	
510	The data record is missing a ZTC entry.	
511	The data record is missing a Pre-Installed Certificate enabled entry.	
512	The data record is missing a User defined certificate config entry.	
513	The data record is missing a User defined certificate Add entry.	
514	The data record is missing a SOL/IDER enable entry.	
515	OEM Firmware Update Qualifier data missing in USB file.	
1000	Invalid command line option(s)	
1001	Unsupported OS	
8192	General error	
8193	Cannot locate ME device	
8194	Memory access failure	
8195	Write register failure	
8196	OS failed to allocate memory	

Error Code	Error Message	Response
8197	Circular buffer overflow	
8198	Not enough memory in circular buffer	
8199	Communication error between application and Intel® TXE <HECI command name>	Contact Intel
8200	Unsupported HECI bus message protocol version	
8201	Unexpected interrupt reason	
8202	Reserved	
8203	Unexpected result in command response <HECI command name>	Contact Intel
8204	Unsupported message type	
8205	Cannot find host client	
8206	Cannot find Intel® TXE client	
8207	Client already connected	
8208	No free connection available	
8209	Illegal parameter	
8210	Flow control error	
8211	No message	
8212	Requesting HECI receive buffer size is too large	
8213	Application or driver internal error	
8214	Circular buffer not empty	

B.2 Firmware Update Errors

Error Code	Error Message
0	Success
1	Reserved
2	Reserved
3	Reserved
4	Reserved
8193	Intel® TXE Interface : Cannot locate Intel® TXE device driver
8704	Firmware update operation not initiated due to a SKU mismatch
8705	Firmware update not initiated due to version mismatch
8706	Firmware update not initiated due to integrity failure or invalid FW image
8707	Firmware update failed due to an internal error
8708	Firmware Update operation not initiated because a firmware update is already in progress
8710	Firmware update tool failed due to insufficient memory
8713	Firmware update not initiated due to an invalid FW image header
8714	Firmware update not initiated due to file open or read failure
8716	Invalid usage
8718	Update operation timed-out; cannot determine if the operation succeeded
8719	Firmware update cannot be initiated because Local Firmware update is disabled
8722	Intel® TXE Interface : Unsupported message type
8723	No Firmware update is happening
8724	Platform did not respond to update request.
8725	Failed to receive last update status from the firmware
8727	Firmware update tool failed to get the firmware parameters
8728	This version of the Intel I® FW Update Tool is not compatible with the current platform.
8741	FW Update Failed.
8743	Unknown or unsupported Platform.
8744	OEM ID verification failed.
8745	Firmware update cannot be initiated because the OEM ID provided is incorrect
8746	Firmware update not initiated due to invalid image length
8747	Firmware update not initiated due to an unavailable global buffer
8748	Firmware update not initiated due to invalid firmware parameters
8754	Encountered error writing to file.

Error Code	Error Message
8757	Display FW Version failed.
8758	The image provided is not supported by the platform.
8759	Internal Error.
8760	Update downgrade vetoed.
8761	Firmware write file failure.
8762	Firmware read file failure.
8763	Firmware delete file failure.
8764	Partition layout NOT compatible.
8765	Downgrade NOT allowed, data mismatched.
8766	Password did not match.
8768	Password Not provided when required.
8769	Polling for FW Update Failed.
8772	Invalid usage, -allowsv switch required to update the same version firmware
8778	Unable to read FW version from file. Verify the update image used.
8787	Password exceeded maximum number of retries.

B.3 TXEManuf Errors

Error Codes	Error Messages
9248	Intel® TXE internal communication error (BIST)
9249	Intel® TXE internal communication error (FW)
9250	Reserved
9251	Fail to create verbose log file %s Where %s is the log file name user specified
9252	Reserved
9254	Reserved
9255	Internal error
9256	Communication error between host application and Intel® TXE FW
9257	Reserved
9261	Hibernation is not supported by the OS, Intel® TXE test cannot run
9262	Reserved
9263	Reserved
9264	Reserved
9265	Reserved
9266	Reserved

Error Codes	Error Messages
9267	Fail to establish a communication with SPI flash interface
9268	Fail to load vsccommn.bin
9269	Zero flash device found for VSCC check
9270	Fail to load driver (PCI access for Windows) Tool needs to run with an administrator privilege account.
9271	Flash ID 0x%06X Intel® TXE VSCC mismatch Programmed value of 0x%X does not match the recommended value of 0x%X See Cherry Trail SoC SPI programming Guide for more details
9272	No recommended Intel® TXE VSCC value found for flash ID 0x%06X
9273	Reserved
9275	Reserved
9276	Fail to read FW Status Register value 0x%X
9277	Reserved
9278	Cannot locate hardware platform identification This program cannot be run on the current platform. Unknown or unsupported hardware platform or A %s hardware platform is detected This program cannot be run on the current platform. Unknown or unsupported hardware platform Where %s is the official name of the hardware platform
9279	SPI flash Intel® TXE region is not locked
9280	Intel® TXE has read or write access to BIOS region
9281	SPI flash descriptor region is not locked
9282	BIOS has granted Intel® TXE access to its region
9283	Region access permissions do not match Intel recommended values
9284	Read firmware flash master region permission failure
9285	Reserved
9286	Reserved
9287	Reserved
9288	Reserved
9289	Reserved
9290	Reserved
9291	Reserved
9292	Reserved
9295	Reserved

Error Codes	Error Messages
9296	TXEManuf Test Failed Or TXEManuf End-Of-Line Test Failed Or TXEManuf Operation Failed
9297	Reserved
9298	Reserved
9299	Single flash part found, Flash Partition Boundary Address must be zero
9300	Flash Partition Boundary Address should be in between flash parts
9301	The two flash parts on this platform require different BIOS VSCC values
9302	Reserved
9303	Memory allocation failed for checking variable "<Variable Name>"
9304	Variable "<Variable Name>" mismatch, actual value is - <Variable Value>
9305	Intel® TXE firmware version mismatch, actual value is - <Version String> BIOS version mismatch, actual value is - <Version String>
9306	Reserved
9307	Reserved
9308	Security Descriptor Override Strap (SDO) is enabled
9309	End-Of-Post message is not sent
9310	Unable to determine Intel® TXE Manufacturing Mode status Intel® TXE is still in Manufacturing Mode
9311	Intel® TXE test failed to start, error 0x%X returned
9312	Intel® TXE test timeout (exceeded 30 seconds)
9313	Reserved
9314	Reserved
9315	Intel® TXE test is currently running, try again
9316	Reserved
9317	Reserved
9318	TXEManuf End-Of-Line Test config file generation failed
9319	Reserved
9320	Internal error
9321	TXEManuf End-Of-Line Test Failed
9322	TXEManuf Operation Failed
9324	Reserved
9325	Reserved
9326	Reserved

Error Codes	Error Messages
9327	Reserved
9328	Internal error
9329	Internal error
9330	Internal error
9331	SMBus hardware is not ready
9332	Internal error
9333	SMBus encountered time-out
9334	Failed to retrieve password from SPI
9335	Internal error
9336	Internal error
9337	Internal error
9338	Failed to retrieve test result from SPI
9339	Failed to retrieve power rule from SPI
9340	Failed to retrieve power source
9341	Reserved
9342	Reserved
9343	Internal error
9344	Reserved
9345	Reserved
9346	Reserved
9347	Power source is not AC
9348	Internal error
9349	Internal error
9350	Internal error
9351	Reserved
9352	Reserved
9353	Reserved
9354	Reserved
9355	Reserved
9356	Reserved
9357	Reserved
9358	Reserved
9359	Reserved
9360	Reserved
9361	Reserved

Error Codes	Error Messages
9362	Internal error
9363	Internal error
9364	The compressed data is incorrect
9365	Reserved
9366	Reserved
9367	Firmware is in recovery mode
9368	SMBus address is not configured correctly
9369	Could not register for SMBus alert
9370	Communication interference
9371	SMBUS connection failed. Check connection or SMBUS address
9372	GPIO connection failed. Check connection or GPIO configuration
9373	NFC Radio – Unknown error
9374	NFC RF Test – Error returned from radio
9375	NFC RF Test – Communication interference or bad response returned from radio
9376	NFC RF Test – Timeout
9377	NFC RF Test – Unknown error
9400	Reserved
9401	Reserved
9402	Reserved
9403	Reserved

B.4 TXEInfo Errors

Error Code	Error Messages
9450	Reserved
9451	Reserved
9452	Communication error between application and Intel® TXE module (iCLS client)
9455	Failed to read FW Status Register value 0x%X
9457	Failed to create verbose log file %s: Where %s is the log file name user specified
9458	Communication error between application and Intel® TXE module (FW Update client)
9459	Internal error (Could not determine FW features information)
9460	Cannot locate hardware platform identification This program cannot be run on the current platform. Unknown or unsupported hardware platform

Error Code	Error Messages
	Or A %s hardware platform is detected This program cannot be run on the current platform. Unknown or unsupported hardware platform Where %s is the official name of the hardware platform
9461	Communication error between application and Intel® TXE module (HCI client)
9462	Communication error between application and Intel® TXE module (Kernel Client)
9467	Cannot use zero as SPI Flash ID index number
9468	Could not find a matching SPI Flash ID
9469	Access to SPI Flash device(s) failed
9470	Failed to load driver (PCI access for Windows) Tool needs to run with an administrator privilege account.
9471	Invalid feature name XXXXX: Where XXXXX is the feature name
9472	XXXXX feature was not available: Where XXXXX is the feature name
9473	XXXXX actual value is – YYYYY: Where XXXXX is the feature name Where YYYY is the feature value
9474	Error reporting revenue share information – Invalid index used
9475	Error reporting revenue share information – Index already in use
9476	Error reporting revenue share information – Slot is empty
9478	End of file encountered when reading first record
9479	Non-Intel chipset is found in first record
9480	Invalid marker found in first record
9481	Unable to locate CODE manifest marker Or Failed to locate DATA manifest marker
9482	Failed to locate PID module entry
9483	This PID cannot be used since the PID matches the known PID for Pre-Production SoCs

B.5 FPT Errors

Error Code	Error
Invalid Parameters	
200	Invalid parameter value specified by the user. Use -? Option to see help.
Invalid Verbose File	
254	Not able to open the file <FILENAME>.
Unsupported Platform	
201	<EXENAME> cannot be run on the current platform. Contact your vendor.
Unsupported OS	
9254	Unsupported OS
Commit FOVs Operation	
517	Get NVAR - Read Failed
518	Get NVAR - Invalid NVAR specified
519	Get NVAR - Out of Memory
520	Get NVAR - Blob Integrity Failed
8193	Intel® TXE Interface : Cannot locate ME device driver
8199	Intel® TXE Interface : TXE Device not ready for data transmission
8204	Intel® TXE Interface : Unsupported message type
8213	Intel® TXE Interface : Buffer too small
Compare FOV(s) Operation	
517	Get NVAR - Read Failed
518	Get NVAR - Invalid NVAR specified
519	Get NVAR - Out of Memory
520	Get NVAR - Blob Integrity Failed
8193	Intel® TXE Interface : Cannot locate ME device driver
8199	Intel® TXE Interface : TXE Device not ready for data transmission
8204	Intel® TXE Interface : Unsupported message type
8213	Intel® TXE Interface : Buffer too small
Retrieve NVAR Operation	
517	Get NVAR - Read Failed
518	Get NVAR - Invalid NVAR specified
519	Get NVAR - Out of Memory
520	Get NVAR - Blob Integrity Failed

Error Code	Error
8193	Intel® TXE Interface : Cannot locate TXE device driver
8199	Intel® TXE Interface : TXE Device not ready for data transmission
8204	Intel® TXE Interface : Unsupported message type
8213	Intel® TXE Interface : Buffer too small
Updating Parameters Operations	
493	Reserved
506	Failed to read from the given file.
3003	Error occurred while opening image file
3004	Parsing of image file failed
3005	Heci communication failed
3006	File does not exist
3007	Operating system is not supported
3008	Reserved
3009	User defined certificate hash table is full
3010	Unable to start HECI
3011	Invalid input file name
3012	Chipset not supported by the tool
3013	PID value is NULL
3014	PPS value is NULL
3015	Configuration Server FQDN value is NULL
3016	PKI DNS Suffix value is NULL
3017	Host Name value is NULL
3018	Domain Name value is NULL
3054	Unable to create Logfile
3055	System failed to retrieve current firmware feature state.
3056	Unable to Save updated parameter as factory defaults on FW image.
3057	Unable to complete FOV commit option.
Widevine Keybox Provision	
576	Keybox file invalid.
581	Invalid keybox status in data out response

§

Appendix C Tool Option Dependency on BIOS/Intel® TXE Status

Tools Options	Intel® TXE Manufacturing Mode Done Bit		End of Post		CF9GR Locking	
	1	0	Yes	No	Yes	No
FPT -Greset	Not related	Not related	Not related	N/A Not related	Fail	Work
FPT –R	Depends on End of post status	Work	Depends on Intel® TXE manufacturing mode donebit status	Work	Not related	Not related
Intel TXEMANUF –EOL config	Depends on End of post status	Work	Depends on Intel® TXE manufacturing mode donebit status	Work	Not related	Not related

§