



# **Bay Trail-M/D/T SoC - System Tools for Intel® Trusted Execution Engine Firmware**

**User Guide**

---

***October 2013***

***Revision: 1.3***

**Intel Confidential**



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. Go to: [http://www.intel.com/products/processor\\_number](http://www.intel.com/products/processor_number)

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

\*Other names and brands may be claimed as the property of others.

Copyright © 2013, Intel Corporation. All rights reserved.

# Contents

|        |   |    |
|--------|---|----|
| 1      | Introduction .....                                | 9  |
| 1.1    | Terminology .....                                 | 9  |
| 1.2    | Reference Documents .....                         | 14 |
| 2      | Preface .....                                     | 15 |
| 2.1    | Overview .....                                    | 15 |
| 2.2    | Intel® TXE System Tools Changes .....             | 15 |
| 2.3    | Image Editing Tools .....                         | 16 |
| 2.4    | Manufacturing Line Validation Tools .....         | 16 |
| 2.5    | Intel® TXE Setting Checker Tool .....             | 16 |
| 2.6    | Operating System Support .....                    | 17 |
| 2.7    | Generic System Requirements .....                 | 17 |
| 2.8    | Error Return .....                                | 18 |
| 2.9    | Usage of the Double-Quote Character (") .....     | 18 |
| 2.10   | PMX Driver Limitation .....                       | 19 |
| 3      | Intel® Flash Image Tool .....                     | 20 |
| 3.1    | System Requirements .....                         | 20 |
| 3.2    | Flash Image Details .....                         | 20 |
| 3.2.1  | Flash Space Allocation .....                      | 21 |
| 3.3    | Required Files .....                              | 21 |
| 3.4    | FITC .....  | 22 |
| 3.4.1  | Configuration Files .....                         | 22 |
| 3.4.2  | Creating New Configuration .....                  | 22 |
| 3.4.3  | Opening an Existing Configuration .....           | 23 |
| 3.4.4  | Saving a Configuration .....                      | 23 |
| 3.4.5  | Environment Variables .....                       | 23 |
| 3.4.6  | Build Settings .....                              | 24 |
| 3.4.7  | Selecting the Platform SKU .....                  | 27 |
| 3.4.8  | Modifying Flash Descriptor Region .....           | 28 |
| 3.4.9  | Descriptor Region Length .....                    | 28 |
| 3.4.10 | Setting Number and Size of Flash Components ..... | 28 |
| 3.4.11 | Region Access Control .....                       | 29 |
| 3.4.12 | SoC Soft Straps .....                             | 31 |
| 3.4.13 | VSCC Table .....                                  | 32 |
| 3.4.14 | Adding New Table .....                            | 32 |
| 3.4.15 | Removing an Existing VSCC Table .....             | 33 |
| 3.4.16 | Modifying Intel® TXE Region .....                 | 33 |
| 3.4.17 | Setting Intel® TXE Region Binary File .....       | 33 |
| 3.4.18 | Configuration .....                               | 33 |
| 3.4.19 | Intel® TXE Section .....                          | 34 |
| 3.4.20 | Features Supported .....                          | 34 |
| 3.4.21 | Setup and Configuration Section .....             | 35 |
| 3.4.22 | Modifying PDR Region .....                        | 35 |
| 3.4.23 | Setting PDR Region Length Option .....            | 35 |
| 3.4.24 | Setting PDR Region Binary File .....              | 35 |



|   |         |   |    |
|---|---------|---|----|
|   | 3.4.25  | Enabling/Disabling PDR Region .....   | 36 |
|   | 3.4.26  | Modifying BIOS Region .....   | 36 |
|   | 3.4.27  | Setting BIOS Region Length Parameter .....  | 36 |
|   | 3.4.28  | Setting BIOS Region Binary File .....   | 36 |
|   | 3.4.29  | Enabling/Disabling BIOS Region .....  | 37 |
|   | 3.4.30  | Building Flash Image .....  | 37 |
|   | 3.4.31  | Change Region Order on SPI Device .....   | 37 |
|   | 3.4.32  | Decomposing an Existing Flash Image .....   | 38 |
|   | 3.4.33  | Command Line Interface .....  | 39 |
|   | 3.4.34  | Example – Decomposing an Image and Extracting Parameters .....                                    | 40 |
|   | 3.4.35  | More Examples of FITC CLI .....   | 41 |
| 4 |         | Intel® Flash Programming Tool .....   | 42 |
|   | 4.1     | System Requirements .....   | 42 |
|   | 4.2     | Flash Image Details .....   | 43 |
|   | 4.3     | Microsoft Windows* Required Files .....   | 43 |
|   | 4.4     | EFI Required Files .....  | 44 |
|   | 4.5     | Programming Flash Device .....  | 44 |
|   | 4.5.1   | Stopping Intel® TXE SPI Operations .....  | 44 |
|   | 4.6     | Usage .....   | 45 |
|   | 4.7     | Programming Fixed Offset Variables .....  | 49 |
|   | 4.8     | Updating Hash Certificate through FOV .....   | 50 |
|   | 4.9     | Fparts.txt File .....   | 52 |
|   | 4.10    | FPF .....   | 52 |
|   | 4.10.1  | FPF Programming .....   | 53 |
|   | 4.10.2  | FPF Mirroring .....   | 55 |
|   | 4.11    | Examples .....  | 56 |
|   | 4.11.1  | Complete SPI Flash Device with Binary File .....  | 56 |
|   | 4.11.2  | Program Specific Region .....   | 57 |
|   | 4.11.3  | Program SPI Flash from a Specific Address .....   | 57 |
|   | 4.11.4  | Dump Full Image .....   | 57 |
|   | 4.11.5  | Dump Specific Region .....  | 58 |
|   | 4.11.6  | Display SPI Information .....   | 58 |
|   | 4.11.7  | Verify Image with Errors .....  | 59 |
|   | 4.11.8  | Verify Image Successfully .....   | 60 |
|   | 4.11.9  | Get Intel® TXE Settings .....   | 60 |
|   | 4.11.10 | Compare Intel® TXE Settings .....   | 61 |
|   | 4.11.11 | FOV Configuration File Generation (-cfggen) .....   | 61 |
| 5 |         | Intel® TXEManuf .....   | 64 |
|   | 5.1     | Windows* PE Requirements .....  | 64 |
|   | 5.2     | How to use Intel TXEMANUF .....   | 64 |
|   | 5.3     | Usage .....   | 64 |
|   | 5.3.1   | Host Based Tests .....  | 66 |
|   | 5.4     | Intel TXEMANUF –EOL Check .....   | 66 |
|   | 5.4.1   | TXEMANUF.cfg File .....   | 66 |
|   | 5.4.2   | TXEMANUF –EOL Variable Check .....  | 68 |
|   | 5.4.3   | TXEMANUF –EOL Config Check .....  | 69 |
|   | 5.4.4   | Output/Result .....   | 69 |
|   | 5.5     | Examples .....  | 69 |
|   | 5.5.1   | Example for TXEMANUF running on a full image with some BIST failures Intel® TXE FW platform ..... | 69 |
| 6 |         | Intel® TXEInfo .....  | 72 |

|            |   |     |
|------------|---|-----|
| 6.1        | Windows* PE Requirements .....  | 72  |
| 6.2        | Usage .....   | 72  |
| 6.3        | Examples .....  | 76  |
| 6.3.1      | Dump Full Detail Info about Intel® TXE and its Application Feature Values .....     | 77  |
| 6.3.2      | Retrieve the Current Value of the Flash Version .....                               | 78  |
| 6.3.3      | Checks whether the Computer has completed the setup and configuration process ..... | 79  |
| 7          | Intel® TXE Firmware Update .....  | 80  |
| 7.1        | Requirements .....  | 80  |
| 7.2        | Windows* PE Requirements .....  | 80  |
| 7.3        | Usage .....   | 81  |
| 7.4        | Examples .....  | 82  |
| 7.4.1      | Updates Intel® TXE with Firmware Binary File .....                                  | 82  |
| 7.4.2      | Display Supported Commands .....  | 83  |
| 8          | Intel® Manifest Generation Tool .....   | 84  |
| 8.1        | Manifest Generation Tool .....  | 85  |
| 8.2        | Signing Tool .....  | 89  |
| 8.3        | Example Command .....   | 90  |
| 8.3.1      | Secure Boot Manifest Creation .....   | 90  |
| 8.3.2      | Key Manifest Creation .....   | 90  |
| Appendix A | Fixed Offset Variables .....  | 92  |
| Appendix B | Tool Detail Error Codes .....   | 98  |
| Appendix C | Tool Option Dependency on BIOS/Intel® TXE Status .....                              | 114 |

## Figures

|            |  |    |
|------------|--|----|
| Figure 1.  | SPI Flash Image Regions .....                          | 20 |
| Figure 2.  | Environment Variables Dialog .....                     | 24 |
| Figure 3.  | Build Settings Dialog .....                            | 26 |
| Figure 4.  | Selected SKU Platform in FITC .....                    | 27 |
| Figure 5.  | Descriptor Region Length Parameter .....               | 28 |
| Figure 6.  | Descriptor Region > Descriptor Map Parameters .....    | 28 |
| Figure 7.  | Flash Components Dialog .....                          | 29 |
| Figure 8.  | Descriptor Region > Component Section Parameters ..... | 29 |
| Figure 9.  | Descriptor Region > Master Access Section .....        | 31 |
| Figure 10. | SoC Straps .....                                       | 32 |
| Figure 11. | Add VSCC Table Entry Dialog .....                      | 32 |
| Figure 12. | Sample VSCC Table Entry .....                          | 33 |
| Figure 13. | Intel® TXE Section .....                               | 34 |
| Figure 14. | Features Supported Section .....                       | 34 |
| Figure 15. | Setup and Configuration Section .....                  | 35 |
| Figure 16. | PDR Region Options .....                               | 35 |
| Figure 17. | BIOS Region Parameters .....                           | 36 |
| Figure 18. | Region Order .....                                     | 38 |
| Figure 19. | Flash Image Regions .....                              | 43 |
| Figure 20. | Raw Hash Values from Certificate File .....            | 51 |
| Figure 21. | Sample Hash.txt File .....                             | 51 |



|   |    |
|---|----|
| Figure 22. Manifest Generation Flow .....   | 85 |
| Figure 23. Manifest File Signing Tool ..... | 89 |

## Tables

|   |    |
|---|----|
| Table 1. OS Support for Tools.....  | 17 |
| Table 2. Tools Summary .....  | 18 |
| Table 3. Flash Image Regions – Description.....                           | 21 |
| Table 4. Build Settings Dialog Options .....                              | 25 |
| Table 5. Region Access Control Table.....                                 | 30 |
| Table 6. CPU/BIOS Access .....  | 30 |
| Table 7. FITC Command Line Options .....                                  | 39 |
| Table 8. Flash Image Regions – Description.....                           | 43 |
| Table 9. FPT Windows* OS Requirements.....                                | 44 |
| Table 10. Command Line Options for fpt.efi, fptw.exe and fptw64.exe ..... | 45 |
| Table 11. FPT –closemfn Behavior.....                                     | 49 |
| Table 12. Fixed Offset Variables Options .....                            | 49 |
| Table 13. Intel-Recommend Access Settings.....                            | 50 |
| Table 14. FPT command for FPF access .....                                | 53 |
| Table 15. Options for the Tool .....                                      | 65 |
| Table 16. TXEMANUF - EOL Config Tests .....                               | 69 |
| Table 17. Intel® TXEInfo Command Line Options .....                       | 73 |
| Table 18. Components Lists Displayed in Intel® TXEInfo .....              | 74 |
| Table 19. Image File Update Options .....                                 | 81 |
| Table 20. Tool Options for Public Key Hash Generation .....               | 85 |
| Table 21. Tool Options for Partial Secure Boot Manifest Generation .....  | 86 |
| Table 22. Tool Options for Secure Boot Manifest Generation.....           | 86 |
| Table 23. Tool Options for Partial Key Manifest Generation.....           | 87 |
| Table 24. Tool Options for Key Manifest Generation .....                  | 87 |
| Table 25. Tool Options for BIOS Image Verification.....                   | 88 |
| Table 26. Fixed Offset Item Descriptions .....                            | 93 |

## Revision History

| Document Number | Revision | Description   | Date           |
|-----------------|----------|---|----------------|
| 519698          | 0.71     | <ul style="list-style-type: none"> <li>Initial Release</li> </ul>   | February 2013  |
| 519698          | 0.72     | <ul style="list-style-type: none"> <li>Fixed the typo in section <a href="#">7.4.2</a>.</li> <li>Updated Reference Documents list.</li> <li>Added FPF mirroring file support into FITC section.</li> <li>Updated figure and screenshots from the latest FW kit release.</li> <li>Updated Fixed Offset Variables options.</li> <li>Added FPF Programming, FPF Mirroring and Guidelines into FPT section.</li> <li>Added –NONFC/NFC Test option and update TXEManuf.cfg file for TXEManuf tool section</li> </ul>   | April 2013     |
| 519698          | 0.8      | <ul style="list-style-type: none"> <li>Added two FPF batch write/compare option into FPT section and update Table 11_for new option and typo.</li> <li>Removed unsupported command from fwupdate tool section.</li> <li>Removed GBE option from FITC command line interface and Table 7_</li> <li>Updated Table 18 and example output for TXEInfo tool</li> <li>Updated TXEManuf.cfg file content.</li> <li>Added Windows 8.1* into OS Support Matrix</li> <li>Added usage and user notice on section <a href="#">4.10.1</a> FPF Programming regarding FPF mirroring file update and reload.</li> </ul> | June 2013      |
| 519698          | 1.0      | <ul style="list-style-type: none"> <li>For PV release.</li> <li>Added three more reference documents. (Bay Trail T/M/D PDG and TXE BIOS Writer guide)</li> <li>Rewrite section <a href="#">Stopping Intel® TXE SPI Operations</a> to avoid confusion.</li> <li>Added reference document for Tablet Platform BIOS signing process.</li> </ul>  | August 2013    |
| 519698          | 1.2      | <ul style="list-style-type: none"> <li>Add Maximum EFI environment variable size definition and notice for UEFI tools in section <a href="#">2.6</a> Operating System Support.</li> <li>Updated Appendix A – Fixed Offset Variables.</li> <li>Removed all references to GbE</li> </ul>  | September 2013 |



| Document Number | Revision | Description   | Date         |
|-----------------|----------|---|--------------|
| 519698          | 1.3      | <ul style="list-style-type: none"><li>• Added details for Widevine* Keybox provisioning flags related to FPT, TXEManuf &amp; TXEInfo</li><li>• Update section Manifest Generation Tool for supporting unsigned manifest and verifybios option</li></ul> | October 2013 |

§



# 1 Introduction

The 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  |
| VLV          | Valleyview  |
| 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. |



| Acronym/Term              | Definition   |
|---------------------------|--|
| EOP                       | End Of Post  |
| 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.  |
| GMCH                      | Graphics and Memory Controller Hub   |
| 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. |

| Acronym/Term       | Definition   |
|--------------------|--|
| Intel® TXE         | Intel® Trusted Execution Engine. The embedded processor residing in the Silicon.   |
| 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  |
| Intel® 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:<br>OS is hung<br>After PCI reset<br>OS watch dog expires<br>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          | <p>Extensible Markup Language. Intel® AMT's XML-based protocol has 3 parts:</p> <p>An envelope that defines a framework for describing what is in a message and how to process it</p> <p>A set of encoding rules for expressing instances of application-defined data types</p> <p>A convention for representing remote procedure calls and responses</p> |
| 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   |

## 1.2 Reference Documents

| Document  | Document No./Location    |
|---|--------------------------|
| FW Bring Up Guide   | FW kit                   |
| Bay Trail-T SoC External Design Specification (EDS)                             | CDI/IBP#515049           |
| Bay Trail-M/D SoC External Design Specification (EDS)                           | CDI/IBP#512177           |
| Bay Trail Platform SoC SPI Programming Guide                                    | FW kit<br>CDI/IBP#514482 |
| Bay Trail-T/I Platform – Intel TXE Firmware Manufacturing Recommendation        | CDI/IBP#515108           |
| Bay Trail-M/D/T SoC Intel TXE BIOS Writer's Guide                               | CDI/IBP#514966           |
| Bay Trail-T Platform Tablet and Convertible Form Factor – Platform Design Guide | CDI/IBP#513059           |
| Bay Trail-M and Bay Trail-D Platform - Design Guide                             | CDI/IBP#512238           |
| Bay Trail-T Platform BIOS Signing – User Guide                                  | CDI/IBP#528651           |



## 2 Preface

---

### 2.1 Overview

This document covers the system tools used for creating, modifying, and writing binary image files, manufacturing testing, Intel® TXE setting information gathering, and Intel® TXE FW updating. The tools are located in **Kit directory\Tools\System tools**. For information on other tools, refer to the tool's user guides in the other directories in the FW release.

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

- Bay Trail Platform – All tools in the Bay Trail FW release kit are designed for Bay Trail platforms only. These tools do not work properly on any other legacy platforms (Cedar Trail, Oak Trail) and previous SoC based platform (Medfield and Clover Trail). Tools designed for other platforms also do not work properly on the Bay 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. Please contact local Intel field representative, if you have any further question regarding legal or license agreement.

### 2.2 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 Wizard will not be supported.
- FITC support SKU manager and FPF mirroring.
- FPT supports the flashing without verifying.
- FTP support FPF read, write, commit, compare and lock.
- One image for both FITC and FW update.
- No legacy DOS support.
- Intel TXEMANUF will save test result in SPI.
- Intel TXEMANUF option changes, supporting –NONFC/NFC option.
- Intel TXEMANUF supports host based BIST and NFC test.

**Note:** More details are available in each tool's documentation.

## 2.3 Image Editing Tools

The following tools create and write flash images:

- FITC:
  - Combines the Descriptor, BIOS, PDR, and Intel® TXE FW binaries into one image
  - Configures softstraps and NVARs for Intel® TXE 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 settings (FOV) after Intel® TXE is flashed on the SPI part.
- FWUpdate:
  - Updates the Intel® TXE FW code region 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.)

## 2.4 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 and Windows\* PE. These tools are mostly run on the manufacturing line to do manufacturing testing.

## 2.5 Intel® TXE Setting Checker Tool

The Intel® TXE setting checker tool (Intel TXEINFO) retrieves and displays information about some of the Intel® TXE settings, the Intel® TXE FW version, and the FW capability on the platform.



## 2.6 Operating System Support

Table 1. OS Support for Tools

| Intel® TXE and Manufacturing Tools | UEFI Shell 32/64 bit | Win* PE 32/64 bit (Based on Win* 8) | Win* 8/8.1 32/64 bit | Win* 7 32/64 bit | Fedora16 Linx 32bit (Kernel 3.5) |
|------------------------------------|----------------------|-------------------------------------|----------------------|------------------|----------------------------------|
| 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                |                                  |

**NOTES:**

1. Currently only tools support UEFI shell 64 bits, 32 bit version will be released and expecting separated binary will be provided once UEFI 32/64 bit support.
2. The Windows\* 64 bit tools will not function when the OS is configured to use EFI / GPT boot capabilities.
3. Tools will be able to run as a 32 bit application on 64Bit OS. (tools not necessary compiled as native 64Bit)
4. 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.

## 2.7 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 its 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                         |

## 2.8 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.)

## 2.9 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 delimited 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`.



## 2.10 PMX Driver Limitation

Several 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 Bay 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
  - 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 chipset 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\* XP, Windows\* 7 and Windows\* 8. The tool does not have to run on an Intel® TXE-enabled system with Bay Trail-M/D/T 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 1. SPI Flash Image Regions**

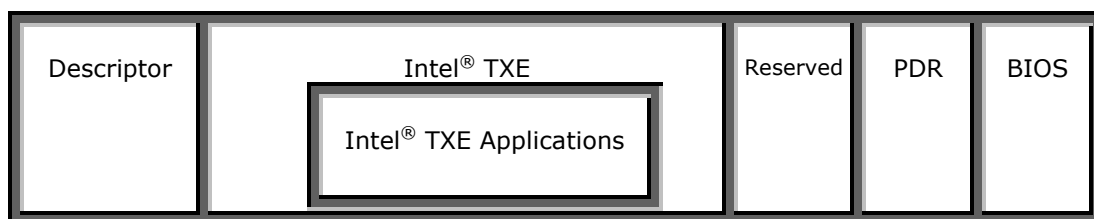


Table 3. Flash Image Regions – Description

| Region     | Description   |
|------------|---|
| Descriptor | <p>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.</p> <p><b>NOTE:</b> This region MUST be locked before the serial flash device is shipped to end users. Please see 3.4.11 below for more information. Failure to lock the Descriptor Region leaves the Intel® TXE device vulnerable to security attacks.</p> |
| Intel® TXE | This region contains code and configuration data for Intel® TXE applications, such as Intel® PTT and Intel® AT. It takes up a variable amount of space at the end of the Descriptor.  |
| 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.  |

### 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 region expands to fill the remaining space.
3. If there is leftover space and Intel® TXE 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.

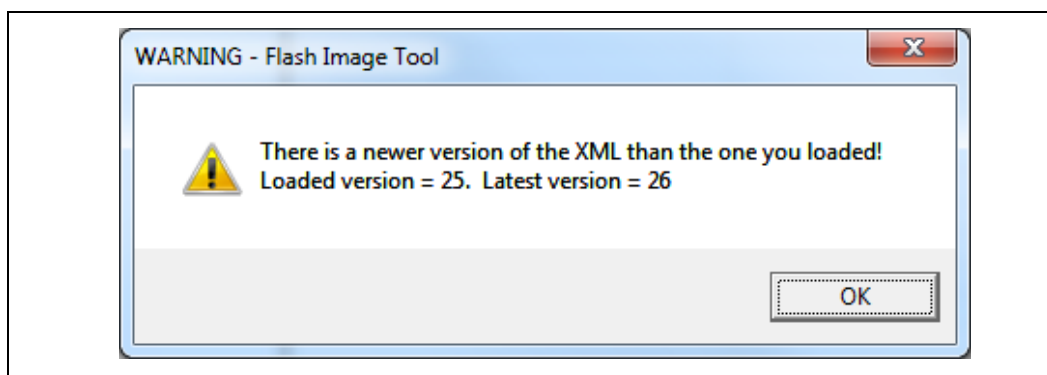
## 3.3 Required Files

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).



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 Bay Trail Platform 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 newfiletmp.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. Choose 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:

Choose File > **Save** or File > **Save As**; the Save File dialog appears if the configuration has not been given a name or if File > **Save As** was chosen.

1. Select the path and enter the file name for the configuration.
2. Click Save.

### 3.4.5 Environment Variables

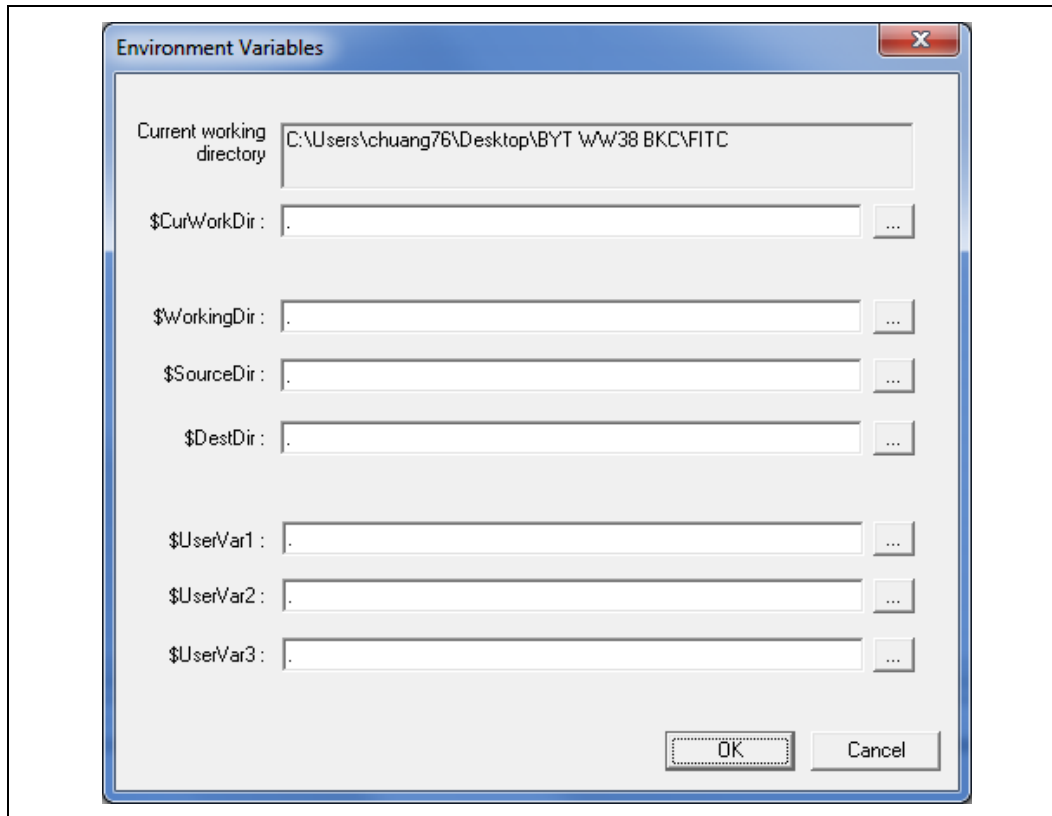
A set of environment variables is provided to make the image configuration files more portable. The configuration is not tied to a particular root directory structure because all of the paths in the configuration are relative to environment variables. The user can set the environment variables appropriate 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. Choose 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 2. Environment Variables Dialog**



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 have been 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:

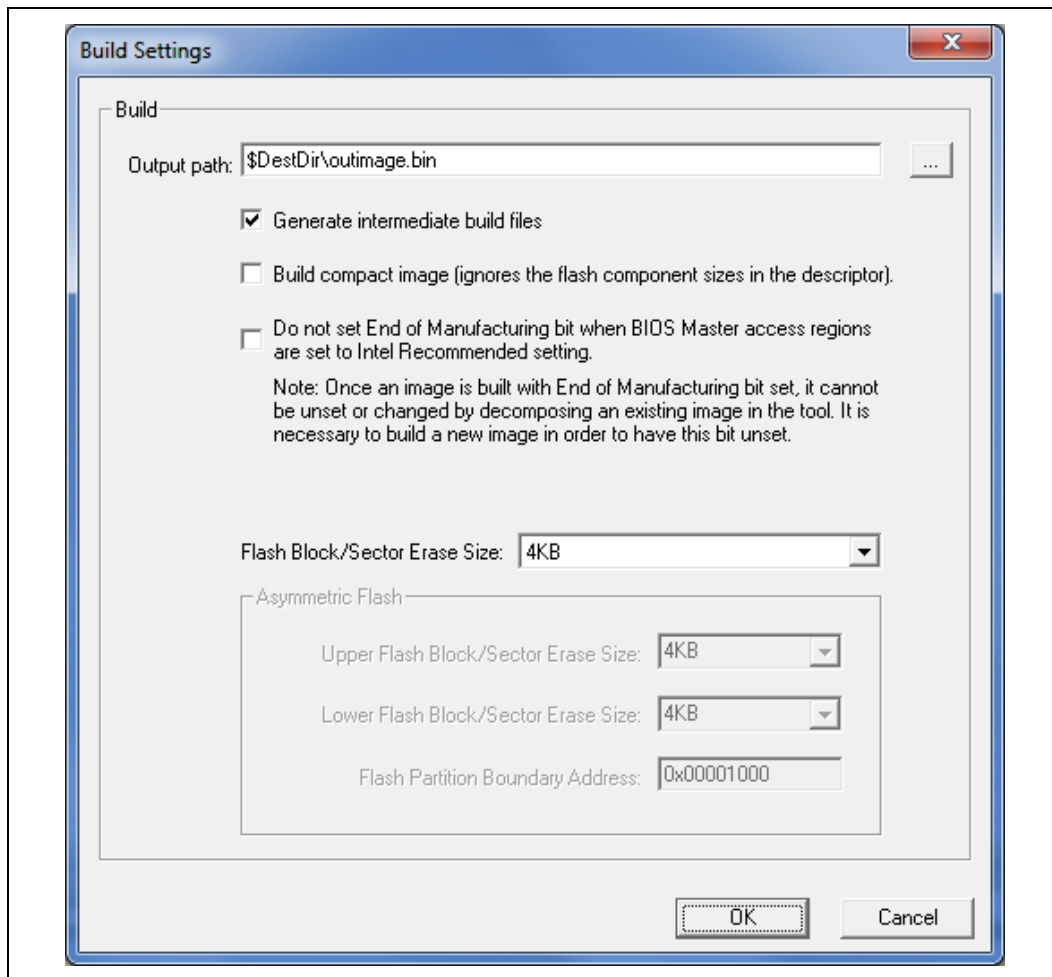
1. Choose **Build > Build Settings**; a dialog appears showing the current build settings.
2. Modify the relevant settings in the **Build Settings** dialog.
3. Click **OK**; the modified build settings are saved in the XML configuration file.



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.<br><b>NOTE:</b> 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 <b>Do not set End of Manufacturing bit</b> 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 <b>4KB sector erase size</b> . 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. <b>Only 4KB erase is supported for Intel® TXE FW</b> . This option also lets user modify the flash partition boundary address.   |

**Figure 3. Build Settings Dialog**



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, but 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 the Platform SKU

The ability to select the Platform SKU lets 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 Chipset SKUs and FW kits.
- SKU Manager Selection has no effect on the Production SoC chip.

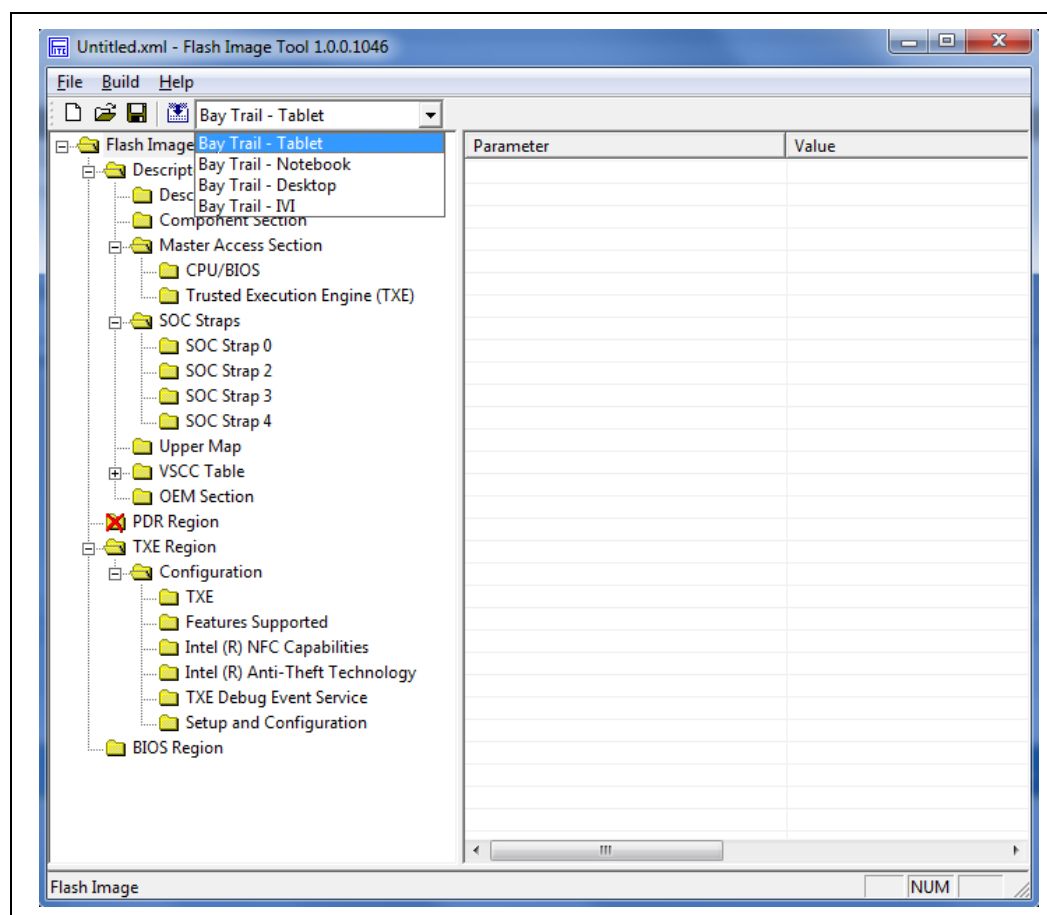
To select a Platform SKU:

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 chipset from the SKU Manager drop-down list.

#### Figure 4. Selected SKU Platform in FITC



### 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 5. Descriptor Region Length Parameter

|  | Parameter                | Value      |
|--|--------------------------|------------|
|  | Descriptor region length | 0x00000000 |

### 3.4.10 Setting Number and Size of Flash Components

To set the number of flash components:

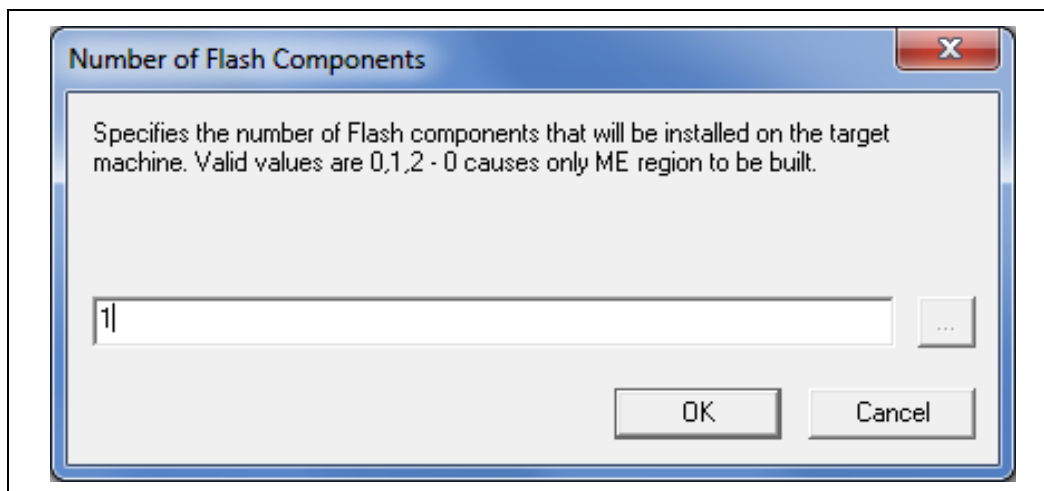
1. Expand the **Descriptor Region** node of the tree in the left pane.
2. Select **Descriptor Map** (see [Figure 6](#)); all the parameters in the Descriptor Map section are listed in the right pane.

Figure 6. Descriptor Region > Descriptor Map Parameters

|  | Parameter                         | Value    |
|--|-----------------------------------|----------|
|  | Region base address               | 0x04     |
|  | <b>Number of Flash Components</b> | <b>1</b> |
|  | Component Base Address            | 0x03     |
|  | Number of SOC straps              | 7        |
|  | SOC straps base address           | 0x10     |
|  | Number of Masters                 | 2        |
|  | Master base address               | 0x06     |
|  | Number of CPU Strap Length        | 0        |
|  | PROC straps base address          | 0x20     |
|  | Component Section                 |          |
|  | Master Access Section             |          |

3. Double-click **Number of Flash Components** in the right pane (see [Figure 7](#)); 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 7. Flash Components Dialog



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. The **Flash component 1 density** and **Flash component 2 density** parameters specify the size of each flash component.
2. Double-click one of these parameters; a dialog appears.
3. Select the correct component size from the dialog's drop-down list and click **OK**; that parameter is updated.
4. 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 8. Descriptor Region &gt; Component Section Parameters

| Parameter                            | Value      | Help Text   |
|--------------------------------------|------------|---|
| Read ID and Read Status clock fre... | 33MHz      | If more that one Flash component exists, this field must be the lowest c...   |
| Write and erase clock frequency      | 33MHz      | If more that one Flash component exists, this field must be the lowest c...   |
| Fast read clock frequency            | 33MHz      | 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 9).
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 9. Descriptor Region > Master Access Section**

A tree view of a Flash Image structure. The root is 'Flash Image', which contains 'Descriptor Region'. 'Descriptor Region' contains 'Descriptor Map' and 'Component Section'. 'Component Section' contains 'Master Access Section'. 'Master Access Section' contains 'CPU/BIOS' (highlighted with a blue box) and 'Trusted Execution Engine (TXE)'.

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

### 3.4.12 SoC Soft 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 Bay Trail Platform SoC SPI programming guide, Appendix A.)

Figure 10. SoC Straps

| Parameter            | Value    | Help Text  |
|----------------------|----------|--|
| SPI Boot Block Size  | 00: 6... | Sets SPI Boot Block Size.  |
| LPSS1 F0 Disable     | false    | Disable LPSS1 function 0 (DMA). false = enable. true = disable.      |
| LPSS1 F1 Disable     | false    | Disable LPSS1 function 1 (PWM#1). false = enable. true = disable.    |
| LPSS1 F2 Disable     | false    | Disable LPSS1 function 2 (PWM#2). false = enable. true = disable.    |
| LPSS1 F3 Disable     | false    | Disable LPSS1 function 3 (HSUART#1). false = enable. true = disable. |
| LPSS1 F4 Disable     | false    | Disable LPSS1 function 4 (HSUART#2). false = enable. true = disable. |
| LPSS1 F5 Disable     | false    | Disable LPSS1 function 5 (SPI). false = enable. true = disable.      |
| SCC eMMC Disable     | false    | Disable eMMC. false = enable. true = disable.                        |
| SCC SDIO Disable     | false    | Disable SDIO. false = enable. true = disable.                        |
| SCC SDCARD Disable   | false    | Disable SDCARD. false = enable. true = disable.                      |
| SCC eMMCplus Disable | false    | Disable eMMC plus. false = enable. true = disable.                   |
| LPE Disable          | false    | Disable LPE. false = enable. true = disable.                         |
| OTG Disable          | false    | Disable OTG. false = enable. true = disable.                         |
| USH Disable          | false    | Disable USH. false = enable. true = disable.                         |
| USB Disable          | false    | Disable USB. false = enable. true = disable.                         |
| LPSS2 F0 Disable     | false    | Disable LPSS2 function 0 (I2C#0). false = enable. true = disable.    |
| LPSS2 F1 Disable     | false    | Disable LPSS2 function 1 (I2C#1). false = enable. true = disable.    |
| LPSS2 F2 Disable     | false    | Disable LPSS2 function 2 (I2C#2). false = enable. true = disable.    |
| LPSS2 F3 Disable     | false    | Disable LPSS2 function 3 (I2C#3). false = enable. true = disable.    |
| LPSS2 F4 Disable     | false    | Disable LPSS2 function 4 (I2C#4). false = enable. true = disable.    |
| LPSS2 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 Bay Trail Platform SoC SPI Programming Guide, Section 6.4.)

### 3.4.14 Adding New Table

To add a new table:

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

Figure 11. Add VSCC Table Entry Dialog

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.



- 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 12](#), 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 Bay Trail Platform SoC SPI Programming Guide should be used to calculate the VSCC values.

**Figure 12. Sample VSCC Table Entry**

| Parameter                                     | Value | Help Text   |
|---|-------|---|
| Vendor ID                                     | 0xEF  | The vendor specific byte of the JEDEC ID.               |
| Device ID 0                                   | 0x40  | 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:

- Right-click on the name of the table in the left pane that the user wants to remove.
- 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® AT).

### 3.4.17 Setting Intel® TXE Region Binary File

To select the Intel® TXE region binary file:

- Select the Intel® TXE Region tree node.
- 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.
- Click **OK** to update the parameter; when the flash image is built, the contents 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 13.

Figure 13. 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.)

Figure 14. Features Supported Section

| Parameter   | Value |
|---|-------|
| Intel (R) Anti-Theft Technology Permanently Disabled? | No    |

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.

Setting **Intel® Anti-Theft Technology Permanently Disabled?** To "Yes" will permanently disable this features listed above. The only way to re-enable these features is to completely re-burn the Intel® TXE region with this setting set to "No". A FW update using **FWUpdLcl.exe** cannot re-enable features.

### 3.4.21 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)

Figure 15. Setup and Configuration Section

| Parameter                                       | Value      |
|---|------------|
| ODM ID used by Intel (R) Services               | 0x00000000 |
| System Integrator ID used by Intel (R) Services | 0x00000000 |
| Reserved ID used by Intel (R) Services          | 0x00000000 |
| Permit Period Timer Resolution                  | Days       |

### 3.4.22 Modifying PDR Region

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

Figure 16. PDR Region Options

| Parameter             | Value      |
|-----------------------|------------|
| PDR region length     | 0x00000000 |
| PDR binary input file |            |

### 3.4.23 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.

### 3.4.24 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.

### 3.4.25 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. Choose **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. Choose **Enable Region** from the pop-up menu.

### 3.4.26 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 17. BIOS Region Parameters**

| Parameter              | Value      |
|------------------------|------------|
| BIOS region length     | 0x00000000 |
| BIOS binary input file |            |

### 3.4.27 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.

### 3.4.28 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.

### 3.4.29 Enabling/Disabling BIOS Region

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

To disable the BIOS Region:

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

To enable the BIOS Region:

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

### 3.4.30 Building Flash Image

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

To build a flash image with the currently loaded configuration:

- Choose 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

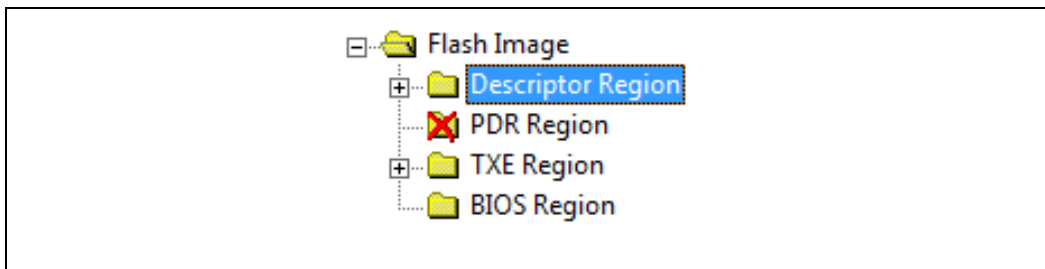
**NOTE:** These files are only created if two flash components are specified.)

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.

### 3.4.31 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 18. 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 19 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 19 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.

### 3.4.32 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. Chose **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 Bay Trail-M/D/T SoC TXE firmware images. FITC will read the SEC 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.

### 3.4.33 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   |
|--------------|---|
| <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.   |
| <BIN File>   | Decomposes the BIN file. The individual regions are separated and placed in a folder with the same name as the BIN file.  |
| -h or -?     | 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.)   |
| -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.  |

| Option                           | Description   |
|----------------------------------|---|
| -u3 <value>                      | Overrides the \$UserVar3 environment variable with the value specified. Can be any value required.  |
| -i<br><enable disable>           | Enables or disables intermediate file generation.   |
| -flashcount<br><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:<br>0 = 512KB<br>1 = 1MB<br>2 = 2MB<br>3 = 4MB<br>4 = 8MB<br>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:<br>0 = 512KB<br>1 = 1MB<br>2 = 2MB<br>3 = 4MB<br>4 = 8MB<br>5 = 16MB.  |
| -platform<br><value>             | This option is used to change the platform configuration being built. Use the words Intel (R) Bay Trail - Tablet, etc. as a reference to a SKU from the drop-down menu (e.g., /sku ??).   |
| -save                            | Saves 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.<br><br><b>NOTE:</b> 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. |

### 3.4.34 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.

### 3.4.35 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\BYT_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\BYT_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 a 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\* 8 64/32 bit, Windows\* 8 SoC.

The Windows 64 bit version (fpt64.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.

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 19. Flash Image Regions

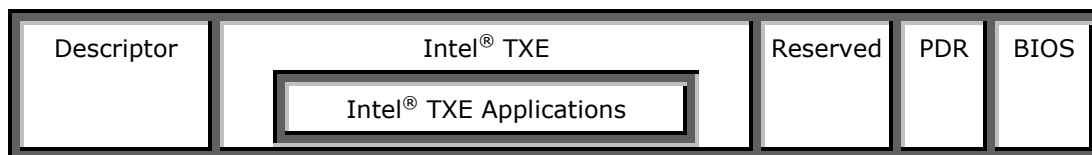


Table 8. Flash Image Regions – Description

| Component  | Description  |
|------------|--|
| Descriptor | Region that takes up a fixed amount of space at the beginning of the flash memory. Contains information such as:<br>Space allocated for each region of the flash image.<br>Read/write permissions for each region.<br>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 and Intel® AT.   |
| 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\_S0\_SC[65] pin low (Flash Descriptor Override Strap) on the rising edge of PMC\_PWROK during power transition. (Refer to Bay Trail-T/M/D Platform Design Guide for more detail and implementation recommendation)
- Send the HMRFP0 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 and Windows 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] [-WRITEFPF] [-READFPF] [-READFPFATTRIB]
          [-COMPAREFPF] [-FPFS] [-WRITEGLOBAL] [-READGLOBAL] [-LOCKFPF]
          [-GETFPFLOCKSTAT] [-WRITEFPFBATCH] [-COMPAREFPFBATCH]
```

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

| Option                  | Description   |
|-------------------------|---|
| Help (-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 <b>fparts.txt</b> .   |
| -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.   |
| -F <file><br><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 specifying the length with -L option, FPT will use the total SPI size instead of an image size.<br><br>The NOVERIFY sub-option <i>must</i> 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. |



| Option                      | Description  |
|-----------------------------|--|
| -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 <code>-f</code> , <code>-b</code> , <code>-c</code> , <code>-d</code> or <code>-verify</code> 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.   |
| -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.  |
| -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 <code>-cfggen</code> command). The <code>-in &lt;file&gt;</code> option is used to specify the input file.                                   |
| -O <file>                   | Output File. The file used by FPT to output FOV information.   |

| Option                   | Description  |
|--------------------------|--|
| -IN <file>               | Input File. The file used by FPT for FOV input. This option flag must be followed by a text file (i.e., <code>fpt -u -in FPT.cfg</code> ). The tool updates the FOVs contained in the text file with the values provided in the input file. User can also use <code>FPT -cfggen</code> 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)   |
| -DUMLOCK:                | 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.<br>This option only support version 1 FiletypeHeader UUID  |
| -CLOSEMNF <NO><br><PDR>: | End of Manufacturing. This option is executed at the end of manufacturing phase. This option does the following:<br>Sets the Intel® TXE manufacturing mode done bit (Global Lock bit).<br>Verifies that the Intel® TXE manufacturing mode done bit (Global Lock bit) is set.<br>Sets the master region access permission in the Descriptor region to its Intel-recommended value<br>Verifies that flash regions are locked.<br>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.<br>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.<br>The "PDR" addition will allow CPU\BIOS Read & Write access to the PDR region of flash.<br><b>Note:</b> Running <code>FPT -closemnf</code> also sets the default value for any unprovisioning process. Run <code>FPT -closemnf</code> 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. |



| Option                          | Description   |
|---------------------------------|---|
| -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.<br>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 <code>-hashed</code> 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 <code>-R</code> and <code>-COMPARE</code> 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: " <code>&lt;name&gt;</code> " = <code>&lt;value&gt;</code><br><b>NOTE:</b> <code>&lt;value&gt;</code> 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. |
| -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   |
| -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                     | Writes the Global Valid Fuse.   |
| -LOCKFPF <name>                 | Locks the specified FPF.  |
| -GETFPFLOCKSTAT<br><name>       | Display the lock status of the specified FPF  |
| -WRITEFPFBATCH<br><f>[NoVerify] | Writes the FPF fuses from a file.   |



| Option                            | Description   |
|-----------------------------------|---|
| -COMPAREFPFBATCH<br><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      | <b>Yes</b>                    | <b>Yes</b>                            | <b>Disabled</b> | 0                   | Yes          |
| No                             | Yes                                  | Enabled      | No                            | Yes                                   | Enabled         | 1                   | No           |
| Yes                            | No                                   | Enabled      | Yes                           | <b>Yes</b>                            | <b>Disabled</b> | 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:<br><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 Updating Hash Certificate through FOV

**Note:** This section is not applicable for special Intel® TXE FW SKU.

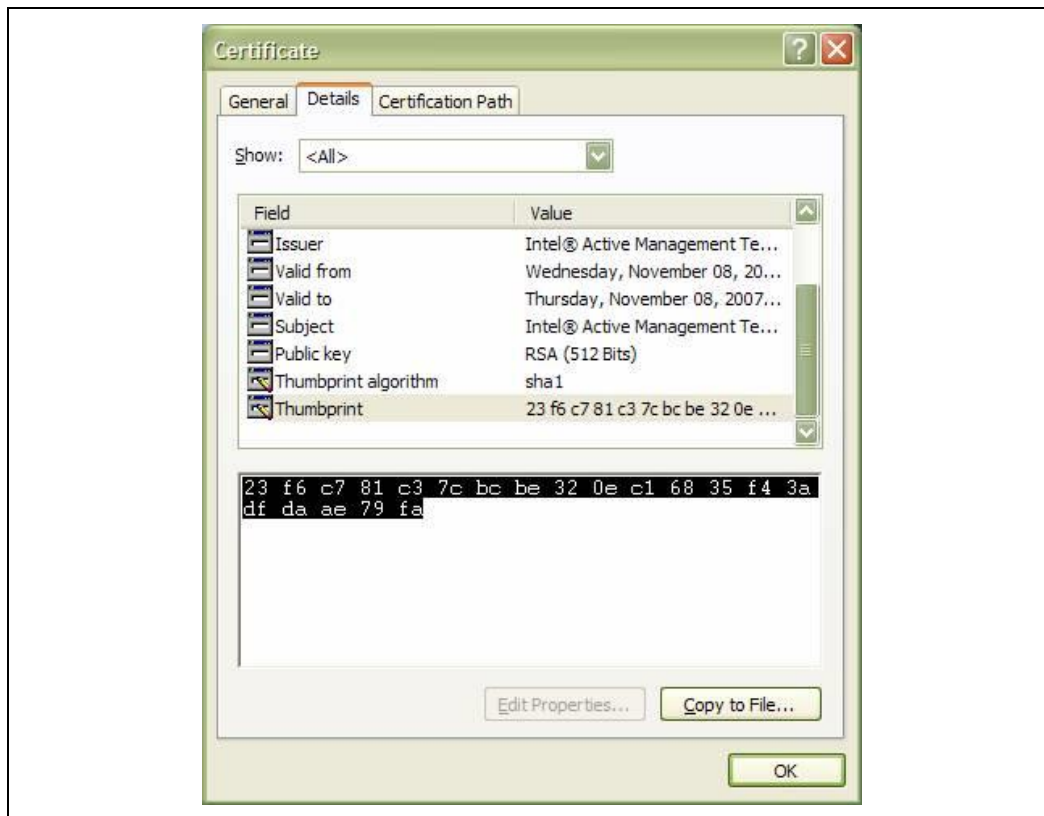
There are some OEMs Customizable certificate hash values that can be stored in the Intel® TXE region:

- The OEM Customizable Certificates 1-3 are not default certificates and are deleted after a full un-provisioning.
- The OEM Customizable Certificates 1-3 are configurable by FOV (with FPT or other flash programming methods) or FITC.

To store certificate hash values in the Intel® TXE region:

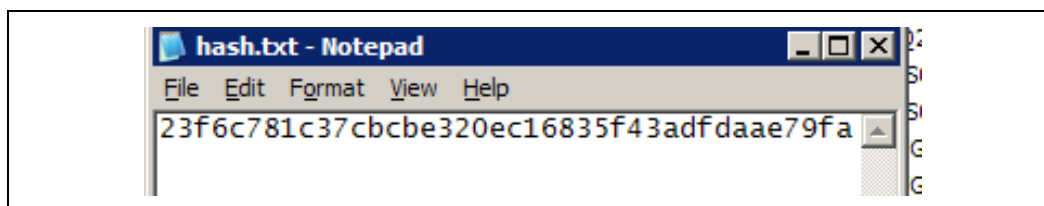
1. Copy the raw hash values from a valid certificate file.

Figure 20. Raw Hash Values from Certificate File



2. Paste the raw hash values into a text file
3. Remove all the spaces from the text file.

Figure 21. Sample Hash.txt File



4. \Save the text file as **hash.txt**.
5. Copy and paste the text saved from hash.txt and add it to **FPT.CFG file** in order to update the FOV:

**EXAMPLE:**

```
; OEMCustomCert1 Certificate
; All data is required to update the certificate.
; See the Tools Users Guide for detailed explanation
; of required data and format.
OEMCustomCert1 IsActive      = 0x01
OEMCustomCert1 FriendlyName  = MyCert
OEMCustomCert1 RawHashFile   = 23f6c781c37cbce320ec16835f43adfdaae79fa
```

6. Flash Hash FOV with FPT's -u -in option (e.g., fpt -u -in sampleparam.txt).



**Note:** **FPT.CFG** is the file that is used to update multiple FOVs.  
(fptw.exe /ex /o FPT.CFG) .

## 4.9 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) 2007-2012, 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 Bay 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.

## 4.10 FPF

Field Programmable Fuses (FPF) is implemented as 4 banks of one time programmable area inside Bay Trail-M/D/T 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, please 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

### 4.10.1 FPF Programming

To support FPF programming on Bay Trail M/D/T 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                   | Writes the Global Valid Fuse.   |
| -LOCKFPF <name>               | Locks the specified FPF.  |
| -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:

```

FpfMirrorNvarValues_WW20.txt - Notepad
File Edit Format View Help
#####
#
# 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]:[Locked]
#
# ID - Fuse file ID (see FUSE_FILE_XXX in FpfHec1Msgs.h)
# Value - Desired value of fuse file in "hex" digits, must be byte-aligned (For single bit file, should be 00 or 01)
# Locked - Boolean indicates if the file should be locked (TRUE/FALSE)
#
# For example, if this file contains the following line:
#
# FUSE_FILE_OEM_KEY_HASH_1:87c558E1FBBA60F7A87E58372D7CCC3BBB704DDB8E2144907C88FA1465A86FEA:TRUE
#
# 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:FALSE
#
# The effective integer value will be: 0x1FFF
#
#####
#This bit indicates that Secure Boot/Verified Boot is enabled. Change value to "01" to enable Secure/verified boot
FUSE_FILE_SECURE_BOOT_EN:00:FALSE
#FW Flag that marks that all Field Programmable Fuses have been programmed and all the values are valid.
#Set to '01' to lock all fuse values.
FUSE_FILE_GLOBAL_VALID:01:FALSE
#set value to '01' to permanently disable Intel(R) PTT (ftpm)
FUSE_FILE_TPM_DISABLE:00:FALSE
#Hash of the public part of the OEM signing key obtained with the Flamingo tool
FUSE_FILE_OEM_KEY_HASH_1:0000000000000000000000000000000000000000000000000000000000000000:FALSE
#The 13 Most Significant Bits of address of alternate copy of IBB within BIOS region
#Alt_bios_limit file is 16 bits wide; applicable values are up to 0x1FFF (13 bits effective).
FUSE_FILE_ALT_BIOS_LIMIT:0000:FALSE
#This is the ID of the of the Key Manifest ('0' indicates no key manifest is required)
FUSE_FILE_KEY_MANIFEST_ID:00:FALSE

```

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]:[Locked]

- **ID:** Fuse file ID
- **Value:** Desired value of fuse file in hex digits, must be byte-aligned (For single bit file, should be 00 or 01)
- **Locked:** Boolean indicates if the file should be locked (TRUE/FALSE)

For example, if this file contains the following line:

```
FUSE_FILE_OEM_KEY_HASH_1:87c558E1FBBA60F7A87E58372D7CCC3BBB704DDB8E2144907C88FA1465A86FEA:TRUE
```

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

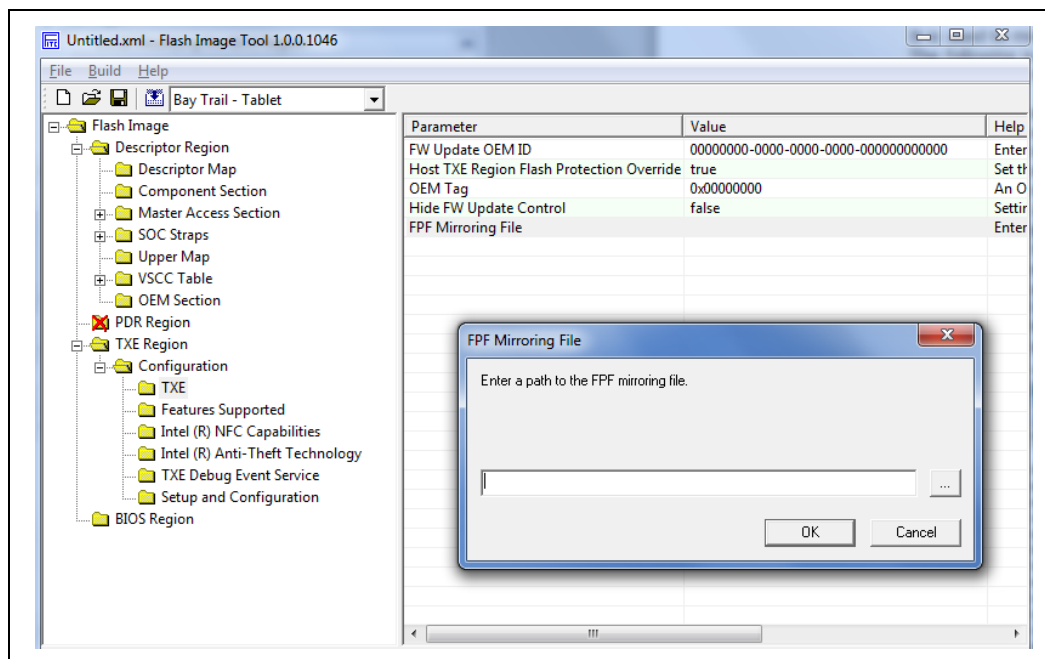
FUSE\_FILE\_ALT\_BIOS\_LIMIT:1FFF:FALSE

## 4.10.2 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 that 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 PF 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.





## 4.11 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.

### 4.11.1 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) 2007-2012, Intel Corporation. All rights reserved.

Platform: "Intel(R) Atom Zxxxx"

Reading HSFSTS register... Flash Descriptor: Valid

--- Flash Devices Found ---

AT26DF321ID:0x1F4700 Size: 4096KB (32768Kb)

AT26DF321ID: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





### 4.11.2 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) 2007-2012, 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.

### 4.11.3 Program SPI Flash from a Specific Address

```
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.

### 4.11.4 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) 2007-2012, 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**.

#### 4.11.5 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) 2007-2012, 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**.

#### 4.11.6 Display SPI Information

```
fptw.exe -I
```

-----

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

Copyright (c) 2007-2012, 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: 0x7FFFFFF

TXE - Base: 0x003000, Limit: 0x5FFFFFF

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".

#### 4.11.7 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) 2007-2012, 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.

#### 4.11.8 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) 2007-2012, 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.

#### 4.11.9 Get Intel® TXE Settings

```
fptw.exe -r "Power Package 1"
```

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

```
Copyright (c) 2007-2011, Intel Corporation. All rights reserved.
```

```
Platform: Intel(R) Qxx Express Chipset
```

```
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.

#### 4.11.10 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>
```

#### 4.11.11 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 =

Intel (R) Anti-Theft Technology =

PAVP =

OEM\_TAG =



ODM\_ID =

SystemIntegratorId =

ReservedId =

ATFPOPHard =

ATFPOPSoft =

§

## 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.

**VSCC.COMN.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.

### 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:

```
TXEMANUF [-EXP] [-H|?] [-VER] [-S0] [-F] [-TEST] [-EOL] [-CFGGEN]
          [-VERBOSE][-PAGE][-NONFC][NFC]
```



Table 15. Options for the Tool

| Option   | Description   |
|--|---|
| No option  | <p>There are differences depending on the firmware SKU type the system is running on:</p> <p>If BIST test result isn't 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.</p> <p>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.</p> <p>When using <code>-verbose</code>, TXEManuf displays the list of all the tests that have been run and retrieved.</p>  |
| <code>-EXP</code>  | Shows examples of how to use the tools.   |
| <code>-H</code> or <code>-?</code>   | Displays the help screen.   |
| <code>-VER</code>  | Shows the version of the tools.   |
| <code>-S0</code>   | 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.  |
| <code>-F &lt;filename&gt;</code>   | Load customer defined .cfg file   |
| <code>-TEST</code>   | Run full test   |
| <code>-EOL</code><br><code>&lt;Var Config&gt;</code><br><code>-F &lt;filename&gt;</code> | <p>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 <code>config</code> or <code>var</code> is optional. Using <code>-EOL</code> without a sub option is equivalent to the <code>-EOL config</code>. Host Based Tests</p> <p>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.</p> <p>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.</p> <p>Intel TXEMANUF <code>-EOL CheckWhen -f</code> flag is used along with a file name, the tool will load the file as the configuration file, instead of using TXEManuf.cfg.</p> |
| <code>-CFGGEN</code><br><code>&lt;filename&gt;</code>                                    | <p>Use this option along with a filename to generate a default configuration file. This file (with or without modification) can be used for the <code>-EOL</code> option. Rename it <b>TXEManuf.cfg</b> before using it. It is highly recommended to use this option to generate a new <b>TXEManuf.cfg</b> with an up-to-date variable names list before using the Intel TXEManuf End-Of-Line check feature.</p>  |
| <code>-VERBOSE</code><br><code>&lt;file&gt;</code>                                       | Displays the debug information of the tool or stores it in a log file.  |

| Option | Description  |
|--------|--|
| -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.  |

### 5.3.1 Host Based Tests

1. 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.
2. 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.

## 5.4 Intel TXEMANUF –EOL Check

TXEMANUF -EOL check is introduced in the Bay Trail-M/D/T 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. Please refer that section.

### 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 delimit 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.
//
//
// TXE FW version is a string as <major ver>.<minor ver>.<hotfix
// ver>.<build num>
// 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="IBB Size 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 SMBus 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 1.0.0.7000:

```
SubTestName="TXE version", Reqval="1.0.0.7000"
```

**Note:** When running Widevine manufacturing flow, please uncomment SubTestName "Validate Keybox Provisioning" from proper EOL testing.

## 5.4.2 TXEMANUF –EOL Variable Check

TXEMANUF -EOL variable 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 Variable check. The system must be in Intel® TXE manufacturing mode when -EOL Variable 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 16. 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  |
| Security Descriptor Override (SDO) check (GPIO_S0_SC[65])   | Disabled                                     |
| <b>NOTE:</b> -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 Example for TXEMANUF running on a full image with some BIST failures Intel® TXE FW platform

```
TXEMANUF -verbose
```

```
Intel(R) TXEManuf Version: 1.0.0.1041
Copyright(C) 2005 - 2013, 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  
vsccommon.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

---

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.

TXEInfo reports an LMS error. This behavior is expected as the LMS driver cannot be installed on Windows\* PE.

### 6.2 Usage

The executable can be invoked by:

```
TXEInfoWin.exe [-EXP] [-H|?] [-VER] [-FEAT <filename>] [-VALUE <value>]
               [-FWSTS] [-VERBOSE <output filename>] [-PAGE] [-PID <filename>]
               [-DUMPIDLM <filename>]

TXEInfo.efi [-EXP] [-H|?] [-VER] [-FEAT <filename>] [-VALUE <value>]
            [-FWSTS] [-VERBOSE <output filename>] [-PAGE] [-PID <filename>]
            [-DUMPIDLM <filename>]
```



Table 17. Intel® TXEInfo Command Line Options

| Option                          | Description  |
|---------------------------------|--|
| -FEAT < name><br>-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.   |
| -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.<br><br>Intel TXEInfo can retrieve all of the information detailed below. However, depending on the SKU selected, some information may not appear.   |
| -FWSTS                          | Decodes the Intel® TXE FW status register value field and breaks it down into the following bit definitions for easy readability:<br><br>FW Status Register1: 0x1F000255<br>FW Status Register2: 0x60000000<br>CurrentState: Normal<br>ManufacturingMode: Enabled<br>TXEMemoryInvalid: Valid<br>OperationalState: M0 with UMA<br>InitComplete: Complete<br>BUPLoadState: Success<br>ErrorCode: No Error<br>ModeOfOperation: Normal<br>Phase: HOSTCOMM module |
| -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".<br><br>This option works with no option and -feat.  |
| -H or -?:                       | Displays the list of command line options supported by the Intel® TXEInfo tool.  |
| -VER                            | Shows the version of the tools.  |
| -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.   |
| -EXP                            | Shows examples about how to use the tools.   |
| -PID <filename>                 | Append/Export Platform ID to the binary file   |
| -DUMPIDLM<filename>             | Displays Platform ID list in an IDLM binary  |
| No option:                      | If the tool is invoked without parameters, it reports information for all components listed in <b>Error! Reference source not found.</b> below.  |

**Table 18. 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<br>Example:<br>1.x.y.ZZZZ;<br>where x=minor,<br>y = HF/MR, ZZZZ<br>= Build Number.   |
| SoC Version          | Intel® TXE Kernel      | Both           | All          | N/A                         | Version string  |
| FW Version           | Intel® TXE Kernel      | Both           | All          | N/A                         | Version string<br>1.x.y.ZZZZ;<br>where x=minor,<br>y = HF/MR, ZZZZ<br>= Build Number.   |
| BIOS Version         | BIOS/Intel® TXE Kernel | Both           | All          | N/A                         | A Version string  |
| NFC Firmware Version | NFC GUID               | Both           | All          | NA                          | A version string.<br>If NFC HW<br>device is not<br>found/accessibl<br>e, display "Not<br>Available"   |
| NFC Loader Version   | NFC GUID               | Both           | All          | NA                          | A version string.<br>If NFC HW<br>device is not<br>found/accessibl<br>e, display "Not<br>Available"   |
| VendorID             | Intel® TXE Kernel      | Both           | All          | N/A                         | A number (in<br>Hex)  |
| TXEI Driver Version  | FW update              | Both           | All          | NA                          | A version string  |
| FW Capabilities      | Intel® TXE Kernel      | Both           | All          | N/A                         | Combination of<br>feature name<br>list breakdown<br>(with a<br>Hexadecimal<br>value)<br>*This is a display<br>of the Feature<br>State for the<br>Intel® TXE. Is |

| Feature Name                    | Feature Data Source                      | Supported SKUs | Supported OS | Specific Feature Dependency                | Field Value  |
|---------------------------------|--|----------------|--------------|--|--|
|                                 |  |                |              |  | enabled / disabled on the system. Each bit in the value represents a feature state. Intel® TXE features including PTT and Anti-theft technology etc. |
| BIOS Config Lock                | Other (Directly reading from SPI)        | Both           | All          | N/A  | Enabled/Disabled /Unknown<br>If shown as enabled, FLOCKDN for BIOS is set.<br>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   |
| Host Write Access to Intel® TXE | Other (Directly reading from SPI)        | Both           | All          | N/A  | Enabled/Disabled / Unknown   |
| Last Intel® TXE Reset Reason    | Intel® TXE Kernel                        | Both           | All          | N/A  | Power up/<br>Firmware reset/<br>Global system reset/<br>Unknown  |
| BIOS Boot State                 | Intel® TXE Kernel                        | Both           | All          | N/A  | Pre Boot/<br>In Boot/<br>Post Boot   |
| OEM Id                          | Intel® TXE Kernel                        | Both           | All          | Only if fw image supports OEM Id           | UUID for OEM to check during FW Update   |
| Local FWUpdate                  | Intel® TXE Kernel                        | Both           | All          | N/A  | Enabled/Disabled / Password Protected  |
| Intel TXEI Driver version       | Other (Reading Windows registry entries) | Both           | Windows*     | Only when Windows TXEI driver is installed | A version string   |

| Feature Name                  | Feature Data Source               | Supported SKUs | Supported OS | Specific Feature Dependency                  | Field Value  |
|-------------------------------|-----------------------------------|----------------|--------------|--|--|
| 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)                                     |
| OEM Tag                       | Intel® TXE Kernel                 | Both           | All          | N/A  | A 32bit Hexadecimal number                                       |
| FWSTS                         | Intel® TXE Kernel                 | Both           | All          | N/A  | Two 32bit Hexadecimal numbers and their bit definition breakdown |
| Global Valid FPF              | Intel® TXE Kernel                 | Both           | All          | N/A  | Valid/Invalid  |
| PTT FPF                       | Intel® TXE Kernel                 | Both           | All          | N/A  | Enabled/Disabled   |
| Perform Secure Boot FPF       | Intel® TXE Kernel                 | Both           | All          | N/A  | Enabled/Disabled   |
| OEM Public Key Hash FPF       | Intel® TXE Kernel                 | Both           | All          | N/A  | A 32 byte number (in Hex)  |
| Key Manifest ID FPF           | Intel® TXE Kernel                 | Both           | All          | N/A  | A 8bit number (in Hex)   |
| Alternative BIOS Limit FPF    | Intel® TXE Kernel                 | Both           | All          | N/A  | A 16bit number (in Hex)  |
| Secure Boot Status            | Intel® TXE Kernel                 | Both           | All          | N/A  | Not Executed/ Executed   |
| Secure Boot Recovery Status   | Intel® TXE Kernel                 | Both           | All          | N/A  | Not Executed/ Executed   |
| Keybox                        | Other (Directly reading from SPI) | Both           | All          | N/A  | Not Provisioned/ 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. The EFI version does not display the UNS version, or LMS version numbers.



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

TXEINFOWIN.exe

Intel(R) TXEInfo Version: 1.0.0.1041

Copyright(C) 2005 - 2013, Intel Corporation. All rights reserved.

Intel(R) TXE code versions:

BIOS Version: Alpha 1.04

VendorID: 8086

SOC Version: 5

FW Version: 1.0.0.1057

TXEI Driver Version: 1.0.0.1054

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



BIOS Config Lock: Enabled

Host Read Access to TXE: Enabled

SPI Flash ID #1: EF6017

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

[illegible]

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

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

Intel(R) TXEInfo Version: 1.0.0.7000

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



BIOS boot State: Post Boot

> TXEInfo.efi -feat "BIOS boot state"

Intel(R) TXEInfo Version: 1.0.0.7000

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

BIOS boot State: Post Boot

### 6.3.3 Checks whether the Computer has completed the setup and configuration process

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

Intel(R) TXEInfo Version: 1.0.0.7000

Copyright(C) 2005 - 2012, 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: 1.0.0.7000

Copyright(C) 2005 - 2012, 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. Please 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 **'VLV\_SEC\_REGION.bin'**. This 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.

If Intel® Anti-theft technology is enabled, a system restart must occur to complete the FW update process.

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 19. Image File Update Options**

| Option               | Description  |
|----------------------|--|
| -VERBOSE<br>[<FILE>] | Verbose. Enables additional information about the tool's operation to be displayed for debugging purposes.   |
| -Y                   | Ignore warning. If the warning asks for input "Y/N", this flag makes the tool automatically take "y" as the input.   |
| -F <FILE>            | File. Specifies the FWUpdate image file to be used for performing an update.   |
| -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.  |
| -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 |



| Option   | Description  |
|----------|--|
|          | image running on the platform. The OEM ID value is specified in the UUID format (8-4-4-4-12).  |
| -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.<br><b>Note:</b> This option is only supported in the Windows version of the tool. |
| -FWVER   | Display FW version   |
| -H or -? | Displays the list of command line options supported by the Intel TXEInfo tool.   |
| -EXP     | Shows examples about how to use the tools.   |
| -VER     | Shows the version of the tools.  |

## 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 1.0.0.7000  
Copyright (C) 2007-2012, 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: 1.0.0.7000  
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: 1.0.0.7000  
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 *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. Please 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. Please 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. Signing 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:**

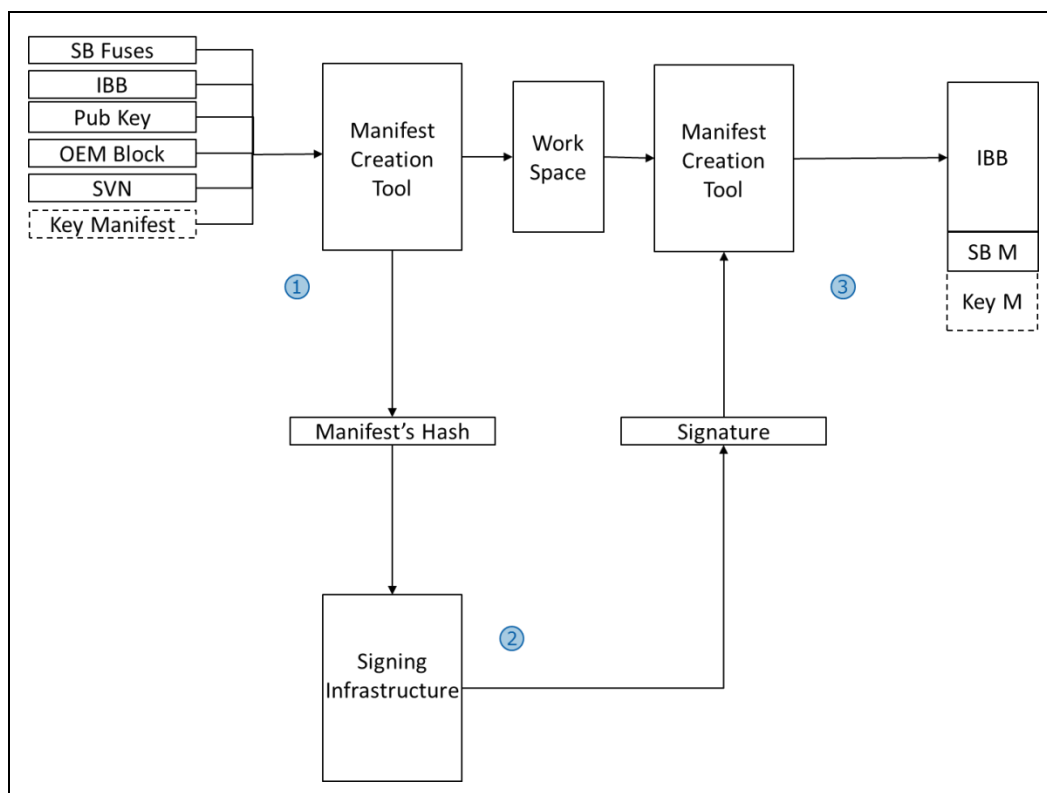
The Manifest Generation Tool will verify that the signature is OK, then insert it in the manifest candidate making it a valid manifest and return the IBB and the manifest as they should be in the BIOS layout.

### **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 22. Manifest Generation Flow



## 8.1 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 [PublicKeyFile] [HashFileOut] [VerifyBios] [-?]

Table 20. Tool Options for Public Key Hash Generation

| Option        | Description  |
|---------------|--|
| HashKey       | Ask tool to create a SHA256 digest (hash) of a given public key    |
| PublicKeyFile | Public key file to calculate SHA256 form                           |
| HashFileOut   | File Name of the file to place the SHA256 digest of the public key |
| -?            | To displays the list of command line options                       |



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

```
FLAMInGo.exe HashKey MyKey.cer MyKeyHash.txt
```

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

```
FLAMInGo.exe SBManCreate [FuseConfigFile] [ManifestName] [IBBFile] [SVN]  
[SigningKey] [-OEMDataFile <OEMDataFile>]  
[-KeyManifestFile <KeyManifestFile>]  
[-Unsigned <UnsignedManifestFile>] [-?]
```

**Table 21. Tool Options for Partial Secure Boot Manifest Generation**

| Option                                | Description   |
|---------------------------------------|---|
| SBManCreate                           | Asks tool to create a partial Secure Boot manifest and a hash file  |
| FuseConfigFile                        | Name of the file that contains the fuses configuration  |
| ManifestName                          | String that identifies the manifest, same name must be used when completing the manifest generation process   |
| IBBFile                               | Name of the file that contains IBB data (maximum 127kb)   |
| SVN                                   | Security Version Number   |
| SigningKey                            | Name of the file that contains the public key of the key that is used to sign the manifest  |
| -OEMDataFile<br><OEMDataFile>         | Name of the file that contains OEM data (maximum 400 bytes)   |
| -KeyManifestFile<br><KeyManifestFile> | Name of the file that contains a valid key manifest generated by this tool  |
| -Unsigned<br><UnsignedManifestFile>   | Name of the file that place then unsigned data of the manifest, by using this will create the blob of the manifest without generating a hash (optional) |
| -?                                    | To displays the list of command line options  |

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

```
FLAMInGo.exe SBManComplete [FuseConfigFile] [ManifestName]  
[SignatureFile] [-?]
```

**Table 22. Tool Options for Secure Boot Manifest Generation**

| Option         | Description  |
|----------------|--|
| SBManComplete  | Asks tool to take a secure boot manifest hash signature and generates a secure boot manifest |
| FuseConfigFile | Name of the file that contains the fuses configuration                                       |

| Option        | Description   |
|---------------|---|
| ManifestName  | String that identifies the manifest, same name must be used when completing the manifest generation process |
| 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 [FuseConfigFile] [ManifestName]
                        [KeyToCerify] [SVN] [SigningKey]
                        [-Unsigned <UnsignedManifestFile>] [-?]
```

**Table 23. Tool Options for Partial Key Manifest Generation**

| Option                              | Description   |
|-------------------------------------|---|
| KeyManCreate                        | Asks tool to create a partial Key manifest and a hash file  |
| FuseConfigFile                      | Name of the file that contains the fuses configuration  |
| ManifestName                        | String that identifies the manifest, same name must be used when completing the manifest generation process   |
| KeyToCerify                         | Name of the file that contains the key to certify by the key manifest   |
| SVN                                 | Security Version Number   |
| SigningKey                          | Name of the file that contains the key to certify by the key manifest   |
| -Unsigned<br><UnsignedManifestFile> | Name of the file that place then unsigned data of the manifest, by using this will create the blob of the manifest without generating a hash (optional) |
| -?                                  | 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 [FuseConfigFile] [ManifestName]
                        [SignatureFile] [-?]
```

**Table 24. Tool Options for Key Manifest Generation**

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



| Option        | Description   |
|---------------|---|
| 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 a BIOS image contains a valid manifest and matches the fuse configuration file.

FLAMInGo.exe VerifyBios [FuseConfigFile] [Image] [-?]

**Table 25. Tool Options for BIOS Image Verification**

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



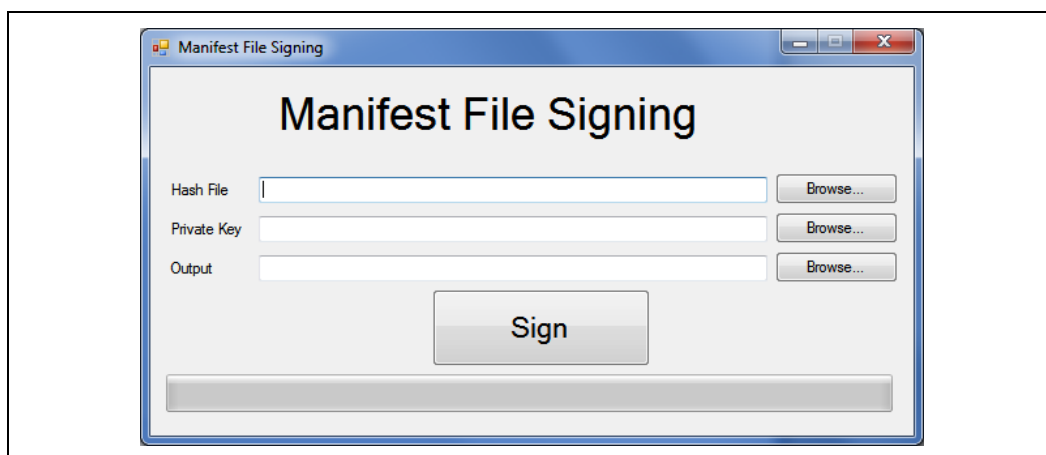
## 8.2 Signing Tool

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 23. 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            |

## 8.3 Example Command

### 8.3.1 Secure Boot Manifest Creation

```
FLAMInGO.exe SBManCreate Fuses.txt MySBManifest IBB.bin 12 MySBSigningKey.cer  
-OEMDataFile myOEMdata.bin -KeyManifestFile MyManifest_manifest.bin
```

The above 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 12.

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.

### 8.3.2 Key Manifest Creation

```
FLAMInGO.exe KeyManCreate Fuses.txt MyManifeste MyKey.cer 34 MySigningKey.cer
```

This above 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 34.

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 26. Fixed Offset Item Descriptions

| Fixed Offset Name                                       | FPT ID | Fixed Offset ID | Description   | Data Length (in Bytes) | Expected Value  | Secure                    | Reset Type |
|---|--------|-----------------|---|------------------------|---|---------------------------|------------|
| Non-Application Specific Fixed Offset Item Descriptions |        |                 |   |                        |   |                           |            |
| OEM Sku Rule  | 7      | 0x000A          | UINT32 (little endian) value. This controls what features are permanently disabled by OEM.<br><br><b>Note:</b><br>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.<br><br>This will not enable dunctionality that is not capable of working in the target hardware SKU. Please see the respective Firmware Bring-up Guide with what firmware bundle and Hardware SKU of Intel Bay Trail SoC. | 4                      | Feature Capable: 1<br>Feature Permanently disabled: 0 | No                        | Global     |
|   |        |                 |   |                        | Bit   |                           |            |
|   |        |                 |   |                        | 31  | Near Field Communications |            |
|   |        |                 |   |                        | 30  | Reserved                  |            |
|   |        |                 |   |                        | 29:22   | Reserved                  |            |
|   |        |                 |   |                        | 21  | Reserved                  |            |
|   |        |                 |   |                        | 20  | DAL                       |            |
|   |        |                 |   |                        | 19  | Reserved                  |            |
|   |        |                 |   |                        | 18  | Reserved                  |            |
|   |        |                 |   |                        | 17  | Reserved                  |            |
|   |        |                 |   |                        | 16  | Reserved                  |            |
|   |        |                 |   |                        | 51:13   | Reserved                  |            |
|   |        |                 |   |                        | 12  | RAVP                      |            |
|   |        |                 |   |                        | 11:6  | Reserved                  |            |
|   |        |                 |   |                        | 5   | Intel AT                  |            |
|   |        |                 |   |                        | 4:3   | Reserved                  |            |
|   |        |                 |   |                        | 2   | Reserved                  |            |
|   |        |                 |   |                        | 1   | Reserved                  |            |
|   |        |                 |   |                        | 0   | Reserved                  |            |



| Fixed Offset Name   | FPT ID | Fixed Offset ID | Description  | Data Length (in Bytes) | Expected Value   | Secure | Reset Type |             |             |    |       |          |  |    |     |   |      |          |  |   |          |  |     |          |  |
|---|--------|-----------------|--|------------------------|--|--------|------------|-------------|-------------|----|-------|----------|--|----|-----|---|------|----------|--|---|----------|--|-----|----------|--|
| Feature Shipment Time State                               | 8      | 0x000B          | <p>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.</p> <p>This will not enable dunctionality that is not capable of working in the target hardware SKU. Please see the respective Firmware Bring-up Guide with what firmware bundle and Hardware SKU of Intel Bay Trail SoC</p> <p><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 feautres. 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.</p> | 4                      | Feature Capable: 1<br>Feature Permanently disabled: 0  | No     | Global     |             |             |    |       |          |  |    |     |   |      |          |  |   |          |  |     |          |  |
|   |        |                 |  |                        | <table><tr><td>Bit</td><td>Description</td><td>No</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> |        |            | Bit         | Description | No | 31:30 | Reserved |  | 29 | PTT | 1 | 28:3 | Reserved |  | 2 | Reserved |  | 1:0 | Reserved |  |
|   |        |                 |  |                        | Bit  |        |            | Description | No          |    |       |          |  |    |     |   |      |          |  |   |          |  |     |          |  |
|   |        |                 |  |                        | 31:30  |        |            | Reserved    |             |    |       |          |  |    |     |   |      |          |  |   |          |  |     |          |  |
|   |        |                 |  |                        | 29   |        |            | PTT         | 1           |    |       |          |  |    |     |   |      |          |  |   |          |  |     |          |  |
|   |        |                 |  |                        | 28:3   |        |            | Reserved    |             |    |       |          |  |    |     |   |      |          |  |   |          |  |     |          |  |
|   |        |                 |  |                        | 2  |        |            | Reserved    |             |    |       |          |  |    |     |   |      |          |  |   |          |  |     |          |  |
|   |        |                 |  |                        | 1:0  |        |            | Reserved    |             |    |       |          |  |    |     |   |      |          |  |   |          |  |     |          |  |
| <b>NOTE:</b> : Bit 29 is only applicable to TXE Firmware. |        |                 |  |                        |  |        |            |             |             |    |       |          |  |    |     |   |      |          |  |   |          |  |     |          |  |
|   |        |                 |  |                        |  |        |            |             |             |    |       |          |  |    |     |   |      |          |  |   |          |  |     |          |  |
|   |        |                 |  |                        |  |        |            |             |             |    |       |          |  |    |     |   |      |          |  |   |          |  |     |          |  |
|   |        |                 |  |                        |  |        |            |             |             |    |       |          |  |    |     |   |      |          |  |   |          |  |     |          |  |
|   |        |                 |  |                        |  |        |            |             |             |    |       |          |  |    |     |   |      |          |  |   |          |  |     |          |  |

| 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<br><br><b>NOTE:</b> 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<br>Value 0x00000000 < n < 0xFFFFFFFF                     | No     | TXE        |
| SystemIntegratorID                            |        | 0x5004          | Used for setting the System Integrator ID used by Intel® Services<br><br><b>NOTE:</b> 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<br>Value:<br>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          | <b>NOTE:</b> 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<br>Value:<br>0x00000000 < n < 0xFFFFFFFF                       | No        | <b>TXE</b> |
| <b>Intel® AT Related FOV Item Descriptions</b>     |        |                 |   |                        |   |           |            |
| AT FW Flash Protection Override Policy Hard FDSO   | 27     | 0x6001          | Indicates whether Hardware flash descriptor security override (FDSO) is allowed, and under what conditions.   | 1                      | Always Allowed: <b>0x01</b><br>Allowed when AT NOT provisioned: <b>0x02</b> | <b>No</b> | <b>TXE</b> |
| AT FW Flash Protection Override Policy Soft HMRPFO | 28     | 0x6002          | Indicates whether Software descriptor override (HMRPFO) is allowed, and under what conditions.  | 1                      | Always Allowed: <b>0x01</b><br>Allowed when AT NOT provisioned: <b>0x02</b> | <b>No</b> | <b>TXE</b> |

**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  |
| Flash Protection Override Policy Hard           | ATFPOPHard  |
| Flash Protection Override Policy Soft           | ATFPOPSoft  |
| 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.<br>Contact the target system BIOS vendor for an option to disable Protected Range Registers.  | Assert Flash Descriptor Override Strap (GPIO_S0_SC[65]) to Low, Power Cycle, and Retry.<br>If Protected Range Registers (memory location: SPIBAR + 74h -> 8Fh) 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<br>Verify the file is not corrupt   |
| 79         | Failed to read the entire file into memory  | Check system memory<br>Verify the file is not corrupt   |
| 80         | Parsing of file failed  | Check system memory<br>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).<br>Contact the BIOS vendor to unlock this bit or enable hardware sequencing in descriptor mode. | Check with BIOS vendor or SPI programming Guide   |



| Error Code | Error Message   | Response  |
|------------|---|---|
| 101        | No SPI flash device could be identified. Please verify if Fparts.txt has support for this part  | Verify <b>Fparts.txt</b> contains device supported.           |
| 102        | Failed to read the device ID from the SPI flash part  | Verify <b>Fparts.txt</b> has correct values                   |
| 103        | There are no supported SPI flash devices installed. Check connectivity and orientation of SPI flash device                                  | Verify <b>Fparts.txt</b> 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   |   |

| Error Code | Error Message   | Response |
|------------|---|----------|
| 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.   |          |
| 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  |          |



| Error Code | Error Message  | Response |
|------------|--|----------|
| 441        | FOV invalid variable ID  |          |
| 442        | Param file is already opened   |          |
| 443        | FOV exists already   |          |
| 444        | Invalid name or Id of FOV  |          |
| 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.  |          |

| Error Code | Error Message  | Response |
|------------|--|----------|
| 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.                              |          |
| 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)   |          |



| Error Code | Error Message  | Response      |
|------------|--|---------------|
| 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   |               |
| 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                         |



| Error Code | Error Message   |
|------------|---|
| 8754       | Encountered error writing to file.  |
| 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. Please 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<br>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 isn't supported by the OS, Intel® TXE test cannot run                  |
| 9262        | Reserved   |
| 9263        | Reserved   |
| 9264        | Reserved   |
| 9265        | Reserved   |

| Error Codes | Error Messages   |
|-------------|--|
| 9266        | Reserved   |
| 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)<br>Tool needs to run with an administrator privilege account.   |
| 9271        | Flash ID 0x%06X Intel® TXE VSCC mismatch<br>Programmed value of 0x%X doesn't match the recommended value of 0x%X<br>See Bay Trail Platform 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<br>This program cannot be run on the current platform.<br>Unknown or unsupported hardware platform<br>or<br>A %s hardware platform is detected<br>This program cannot be run on the current platform.<br>Unknown or unsupported hardware platform<br>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 don't 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   |



| Error Codes | Error Messages  |
|-------------|---|
| 9295        | Reserved  |
| 9296        | TXEManuf Test Failed<br>Or<br>TXEManuf End-Of-Line Test Failed<br>Or<br>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><br>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<br>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  |

| Error Codes | Error Messages                          |
|-------------|---|
| 9326        | Reserved                                |
| 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                                |



| Error Codes | Error Messages   |
|-------------|--|
| 9360        | Reserved   |
| 9361        | Reserved   |
| 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:<br>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)                          |

| Error Code | Error Messages   |
|------------|--|
| 9460       | Cannot locate hardware platform identification<br>This program cannot be run on the current platform.<br>Unknown or unsupported hardware platform<br>Or<br>A %s hardware platform is detected<br>This program cannot be run on the current platform.<br>Unknown or unsupported hardware platform<br>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       | Couldn't find a matching SPI Flash ID  |
| 9469       | Access to SPI Flash device(s) failed   |
| 9470       | Failed to load driver (PCI access for Windows)<br>Tool needs to run with an administrator privilege account.   |
| 9471       | Invalid feature name XXXXX:<br>Where XXXXX is the feature name   |
| 9472       | XXXXXX feature was not available:<br>Where XXXXX is the feature name   |
| 9473       | XXXXXX actual value is – YYYYY:<br>Where XXXXX is the feature name<br>Where YYYYY 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<br>Or<br>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   |
|---------------------------------|---|
| <b>Invalid Parameters</b>       |   |
| 200                             | Invalid parameter value specified by the user. Use -? Option to see help.       |
| <b>Invalid Verbose File</b>     |   |
| 254                             | Not able to open the file <FILENAME>.   |
| <b>Unsupported Platform</b>     |   |
| 201                             | <EXENAME> cannot be run on the current platform.<br>Please contact your vendor. |
| <b>Unsupported OS</b>           |   |
| 9254                            | Unsupported OS  |
| <b>Commit FOVs Operation</b>    |   |
| 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   |
| <b>Compare FOV(s) Operation</b> |   |
| 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   |
| <b>Retrieve NVAR Operation</b>  |   |
| 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                           |
| <b>Updating Parameters Operations</b> |   |
| 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.                             |



## 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 |

§