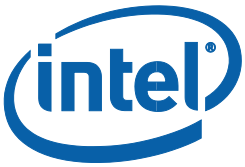


Intel® Integrated Sensor Solution Bring-Up Guide

Software Guide

Revision 1.4

Nov. 2022

Intel Confidential

By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

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.

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." 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.

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

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

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

Copyright © 2017 Intel Corporation. All rights reserved.

Contents

Figures

Revision History

Document Number	Revision Number	Description	Revision Date
571022	0.5	Initial Release	January 2017
	0.6	Remove MEManufu tool from doc	August 2017
	0.7	Adjusted signing steps to support additional platforms	August 2017
	0.8	Adjusted signing and driver installer part	November 2019
	0.9	Added flow of ISH FW/PDT update	January 2020
	1.0	Add more information about SW drivers	August 2020
	1.1	Add CH3.11 for signing change for ADL	May 2021
	1.2	Correct ME version for CH3.11	August 2021
	1.3	Adjusted CH1.3 used tools Adjusted CH8 for issues trouble shooting Adjusted Appendix B Added Appendix E	June 2022
	1.4	Adjusted CH12	November 2022

1 Introduction

This document covers the Intel® Integrated Sensor Solution bring-up for platforms after Intel® Gemini Lake.

This bring-up guide is intended to help the user achieve the following tasks:

1. Install the necessary software drivers to the SUT
2. Modify the platform descriptor table (PDT) to match the system sensor configuration
3. Add the necessary ISS firmware components (FW, PDT) to the binary image

Note: If customer will add third party algorithms or micro driver to the FW, the added models that should be developed using the Firmware Development Kit (FDK) should be created before the bring-up of the Intel® Integrated Sensor Solution.

1.1 Expected Outcome

The expected outcome of this guide for the user is the following:

1. Create a binary image containing the ISS FW components that can be written to the SPI flash of the SUT
2. Prepare the host OS of the SUT for using the Intel® Integrated Sensor Solution by installing the necessary SW drivers.

The location of the ISS FW components within the flash binary image are:

1. ISS FW – stored in an IUP within the ME region
2. ISS PDT – stored within the DFS of the ME region

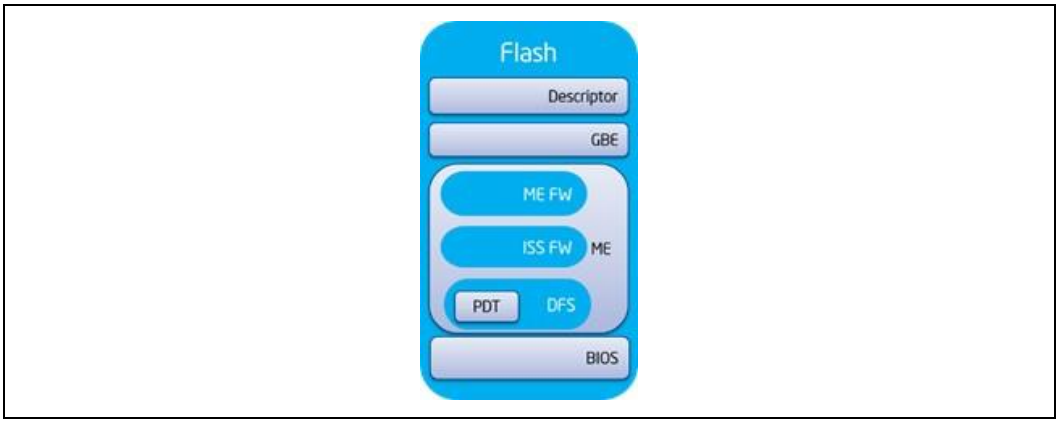


Figure 1-1: Location of FW Components in SPI Binary Image

1.2 Pre-Requisites

The user should download and install the following kits.

- Latest Intel® ME FW kit
- Intel® Integrated Sensor Solution FW kit

The kits can be downloaded from https://platformsw.intel.com/

1.3 Tools Used

The following tools are used within this document:

- Flash Image Tool (FIT) [Intel® ME Kit]
- PDT Editor [Intel® Integrated Sensor Solution FW Kit]
- SW Calibration Tool [Intel® Integrated Sensor Solution FW Kit]
- TraceTool [Intel® Integrated Sensor Solution FW Kit]
- ISSU [Intel® Integrated Sensor Solution FW Kit]

1.4 Terminology

Term	Description
DFS	Data File System (section within the ME Region)
FDK	Firmware Development Kit
IUP	Independent Upgradable Partition
ISH	Integrated Sensor Hub
I²C	Inter-Integrated Circuit, referred to as I-squared-C, I-two-C, or IIC.
MEU	Intel® Manifest Extension Utility
PDT	Platform Descriptor Table consisting of two sections: (1) sensor configuration including sensor driver and I2C address. (2) Calibration data including the sensor driver and I2C address.
Per-Model Calibration	Calibration process that is done for a specific platform design (SKU) and not for an individual system
TPV	Third Party Vendor
SUT	System Under Test
Virtual sensor	A sensor that takes as input measurements from other sensor(s), either physical or virtual, and calculates a new measurement that may be exposed to the OS as a new sensor or consumed internally by ISS firmware.

2 Adding ISS Components

A high-level overview of adding ISS components to the binary image is described below. The key steps include:

1. [Optional] Creating the custom ISS FW components (PDT, FW) with the FDK 2. Modifying the PDT to match the sensor configuration on the system.
3. Adding the ISS components (PDT, FW) to the binary image using the FIT tool.

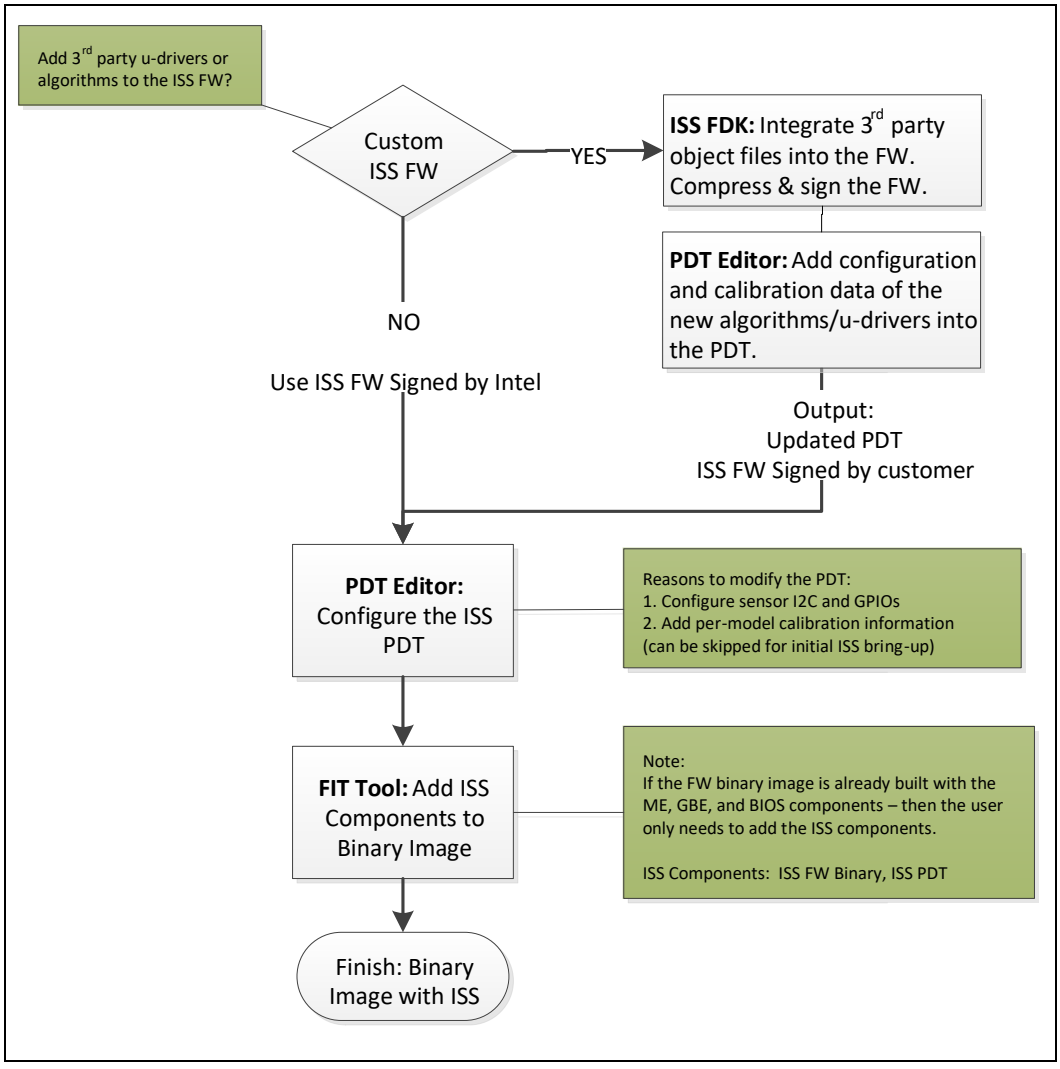


Figure 2-1: High Level Overview of Adding ISS Components to Binary Image

§

3 Signing ISH Binary

Note: This chapter is only for GLK, WHL, CML, ICL.
After TGL, please refer "Signing and Manifesting User Guide.pdf" from ME kit.

3.1 Intel® ISS BOM vs. Intel FDK

Intel ISS supports two use cases of ownership of the ISH code:

Full Intel ownership – Intel ISS defined BOMs

Intel ISS code is signed and released by Intel. It can be integrated into the Intel® CSME binary using the FIT Tool without additional action required from the customer side. OEM/ODM can recreate, modify, update and configure the Intel ISS PDT data region using the Intel PDT Editor tool, while Intel also releases 5 default BOM configuration that can be utilized by customers.

Full OEM ownership – Develop using Intel ISS FDK

Customers who are interested to modify Intel ISS firmware code by, for example, adding new sensors uDrivers/Algorithms that are not included in the default Intel ISS FW, can use Intel FDK to support their development work.

In this case customer will be the owner of the new Intel ISS binary that is created by the Intel FDK and will have to sign the firmware binary by applying his own private and public certificates.

3.2 Basic Concept of ISH Signing

The next section reviews the public key infrastructure.

Intel ISS signing (similar to the Boot Guard technology) employs RSA 2048 public key infrastructure (PKI) mechanism to sign and verify the Firmware image. The private key is used to sign the image as shown in Figure 3.

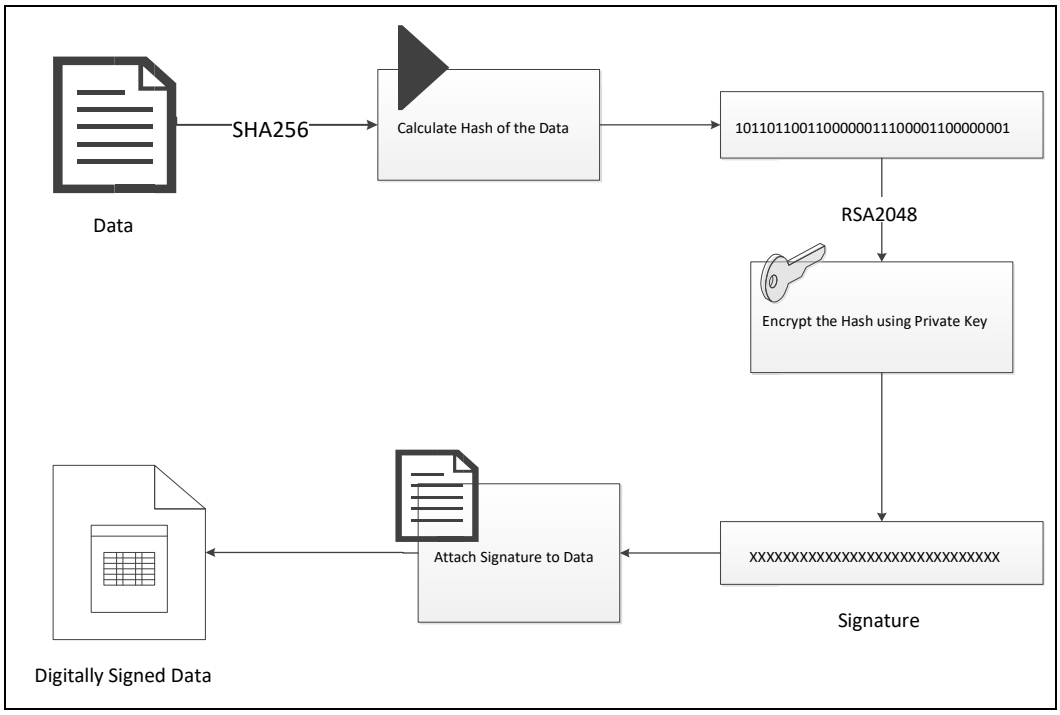


Figure 3-1: Schematic View of ISH Firmware Signing Process

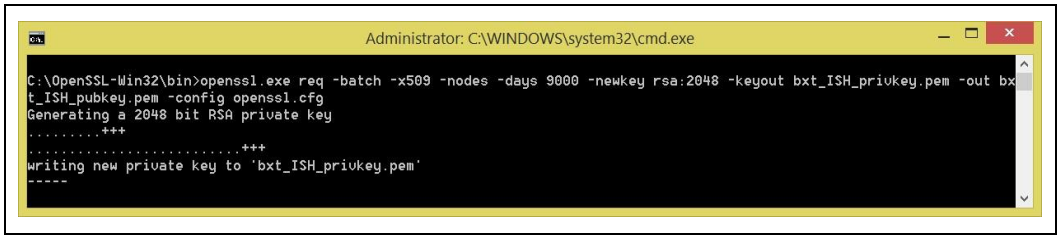
Note: Private keys should be always stored securely and kept secret to provide a robust secure boot flow or ISH firmware load. Public keys are used to verify the integrity of the images. Public keys are stored inside the manifest and hash of the public key is stored in the fuses on the PSH.

3.3 Generating Key Pair for Signing

Users are requested to generate key pairs to be used for signing in the RSA-2048 PKC-1.5 format. **This is the only key format which is supported for ISH signing flow.**

It is recommended to use *openssl.exe* from Shining Light Productions to generate the key pairs:

```
# openssl.exe req -batch -x509 -nodes -days 9000 -newkey
rsa:2048 -keyout privkey.pem -out pubkey.pem -config
openssl.cfg
```



NOTES:

1. Use OpenSSL 1.0.2b or higher versions
2. If the user has the private key from third party – he/she must verify that the key complies with the above requirements for keys that are used in ISH signing. Any other key format or length will trigger signing failure.

3.4 Reasons to Use MEU Tool

Using the Intel FDK, a customer can integrate other third party code (uDrivers, algorithms) into the unsigned FW included in the Intel FDK. The output FW binary must be transformed into a signed IUP before it can be used on the system. MEU tool provided by Intel is designed to execute Intel ISS FW signing.

The MEU tool therefore:

1. Calls an external compression tool to compress the binary.
2. Calls an external signing tool to sign the FW binary.

The resulting signed FW binary can be added to an Independent Updateable Partition (IUP) within the ME FW region using the Firmware Integration Tool (FIT).

NOTES:

1. The MEU tool supports only the LZMA tool to compress the Intel ISS FW.
2. The MEU tool supports only the OpenSSL for Intel ISS FW signing tool.
3. Other compression and signing tools cannot be used with MEU

Tool Name	Released By	Location
Intel ISS FDK	Intel	FDK Kit on Intel VIP portal
MEU Tool	Intel	Intel® ME FW Kit on VIP portal
LZMA Tool	Intel	After TGL, please download from official website Internet: Lzma SDK The rest platforms, please download it from Intel® ISS FW Kit on VIP portal or Intel FDK package
OpenSLL	Open source	Internet: Shining Light Productions

NOTES:

1. Use OpenSSL 1.0.2b or higher versions (32b or 64b)
2. If the FDK is not being used – the customer can use Intel ISS signed FW binary that is included in the Intel® ISS FW kits released to the Intel VIP portal.

3.5 Before Running MEU Tool

Gather the following components:

- The unsigned FW binary created using the Intel FDK
- Intel® Manifest Extension Utility (MEU)
- LZMA tool (LZMA.exe)
- OpenSSL.exe tool – ver 1.0.2b and above

- Customer specific Private key[1]

3.6 Expected Outcome

The expected outcome is a signed Intel ISS FW binary that has been readied for adding to the Intel® ME region to build the SPI full image. This includes:

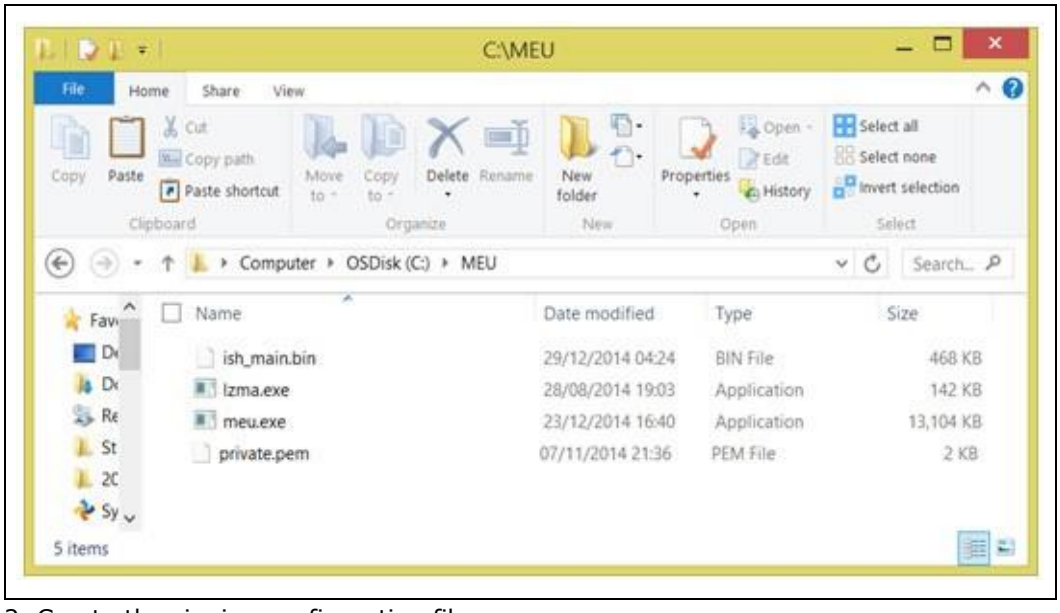
- Adding a manifest to the Intel ISS binary.
- Compressing and signing the binary.

Note: The hash of the public key is added to the silicon/PCH and is used during the authentication of the signed FW binary, which happens during the system boot.

3.7 Signing Flow Steps

1. Place the required components into the same folder:
   a. MEU Tool
   b. Compression tool (LZMA.exe)
   c. Signing Tool (OpenSSL.exe) – should be placed in the openSSL folder (<path>:\OpenSSL-Win32\bin\)
   d. FW binary created using FDK (e.g. ISH_SI.bin)
   e. Private Key (e.g. Private.pem)

Figure 3-2: View of Components Needed to Run MEU Sequence



2. Create the signing configuration file:
   a. Open a command prompt as an administrator and
   b. Run the following commands.

```
# MEU.exe -gen meu_config
```

[1] Private keys are owned by the customer and not provided by Intel.

```
# MEU.exe –gen CodePartition
```

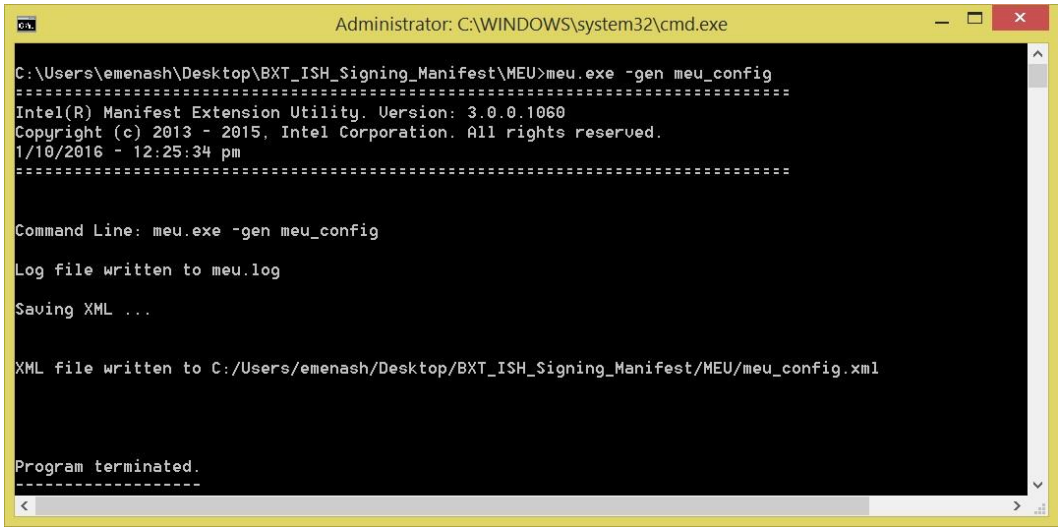


Figure 3-3: Creating me_config file using MEU Tool

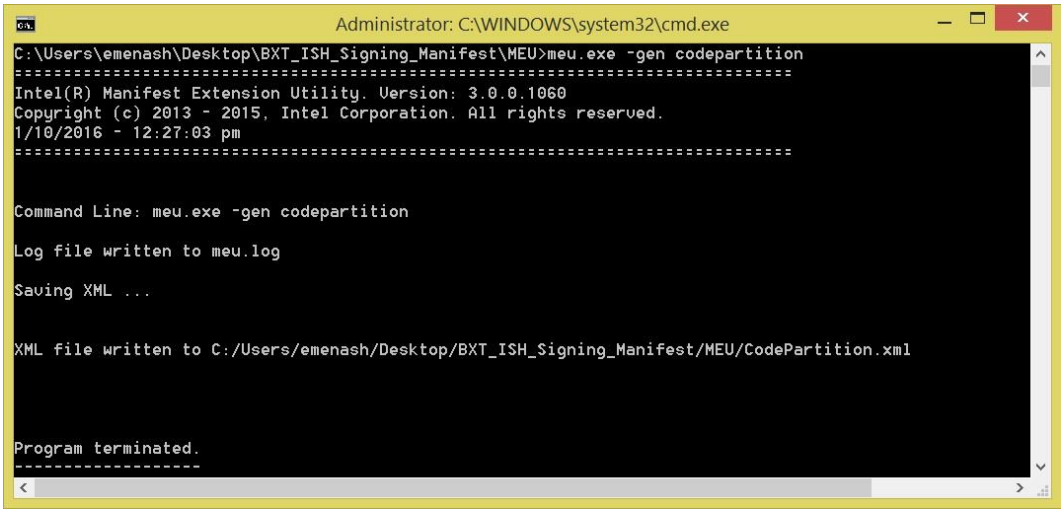


Figure 3-4: Creating CodePartition file using MEU Tool

c. Verify the creation of two files (meu_config.xml, CodePartition.xml) – that are the MEU configuration files for signing
d. Change the CPDataModule parameter in the CodePartition.xml file to the following:
    a) Set "**`VersionMajor.VersionMinor.VersionHotfix.VersionBuild`**" to be FW version. This version is shown in FIT tool.
    For example, FW version is 5.1.0.4333 then VersionMajor should be 5 and VersionBuild should be 4333.

b) Set your VendorId < OEM/ODM should use non-intel vendor ID@0x8086 >
c) Set *Input File value* = <ISH image full path>ISH_Si.bin Example 1

`CodePartition.xml` file that is being ready for **Intel** key signing

```
<?xml version="1.0" encoding="UTF-8"?>
- <CodePartition version="1.1">
<Name value="ISHC"/>
<Length value="0x0"/>
<InstanceId value="0x1"/>
<VersionMajor value="0x5" label="Version Major"/>
<VersionMinor value="0x1" label="Version Minor"/>
<VersionHotfix value="0x0" label="Version Hotfix"/>
<VersionBuild value="0x4333" label="Version Build"/>
<VendorId value="0x8086"/>
<PartitionFlags value="0x00000000"/>
<PartitionVersion value="0x10000000"/>
<DataFormatVersion value="0x00010000"/>
<VersionControlNumber value="0x00000000"/>
<SecurityVersionNumber value="0x0"/>
-<CPModules>
-<CPDataModule enabled="true" name="ISH_si.bin">
<InputFile value="c:\ISH\ISH_si.bin"/>
<CompressionType value="LZMA" value_list="NOT_COMPRESSED,,LZMA"/>
<ProcessId value="0xf6"/>
</CPDataModule>
</CPModules>
</CodePartition>
```

3. Change the following parameters in the meu_config.xml file:
   a) SigningTool value="OpenSSL"
   b) <SigningToolPath value = C:\OpenSSL-Win32\bin\OpenSSL.exe> (according to the actual tools path)
   c) <PrivateKeyPath = c:\OpenSSL-Win32\bin\Private.pem> The private key file should be the one you have generated in the previous section with openssl tool
   d) <CompressionConfig LzmaToolPath value –in MEU directoty> = c:\MEU\LZMA.exe>

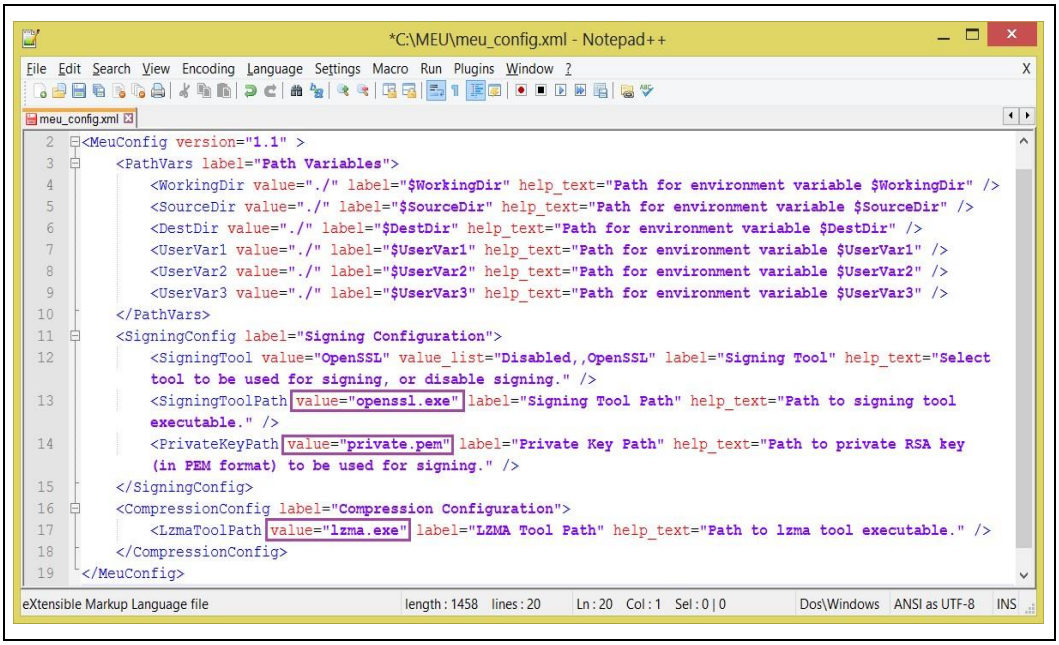


Figure 3-5: Edits to meu_config.xml File

4. Run the MEU tool to create the compressed & signed ISH firmware image.
   a) Open command prompt as administrator
   b) Run the commands with the following parameters: (case sensitive)
      # MEU.exe –f CodePartition.xml –o ISHC_MEU.bin
   c) Verify that this command has generated and the signed binary file exists.

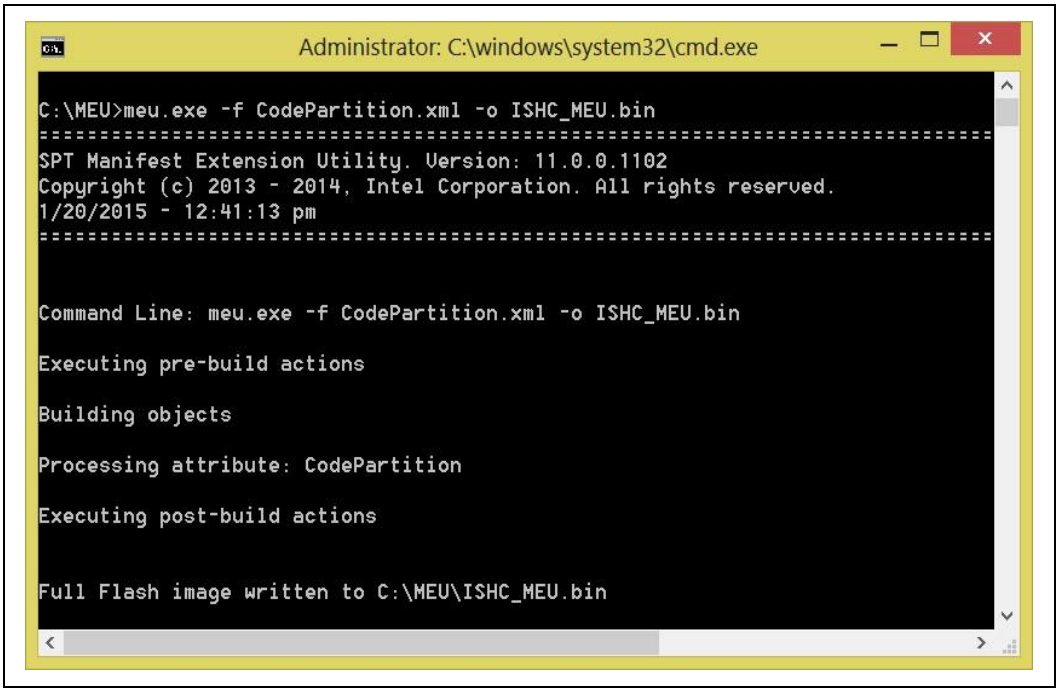


Figure 3-6: Example of Running MEU Tool to Create Signed Binary File

3.8 Creating the Hash

Now after the ISH firmware image has been signed. We will need to create the hash string that will be put into FIT and into the manifest file to create the trust relations between FW and HW.

1. Run the following command:
   a. meu.exe -keyhash **Hash_bxt_ISH** –key privkey.pem"
2. This command will generate a hash file based on the private key generated in the previous section using openSSL tool. The hash key file will be used for creating the trust relations between the HW and ISH firmware image

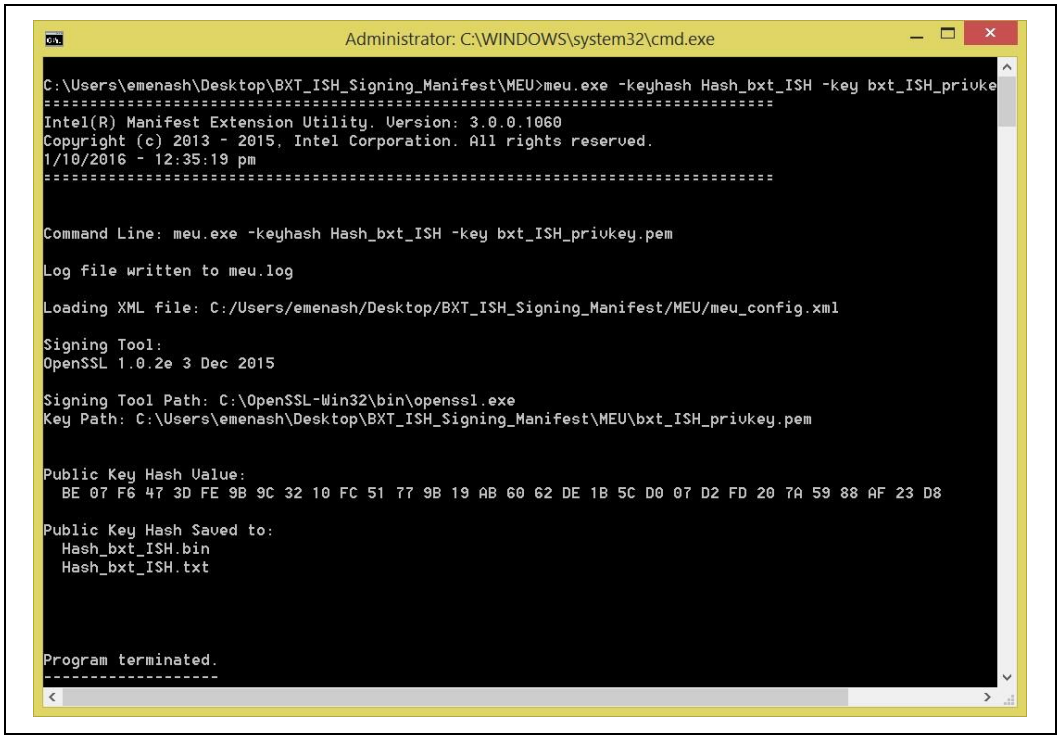


Figure 3-7: Running MEU Tool with meu.exe -keyhash Hash_bxt_ISH –key bxt_ISH_privkey.pem command

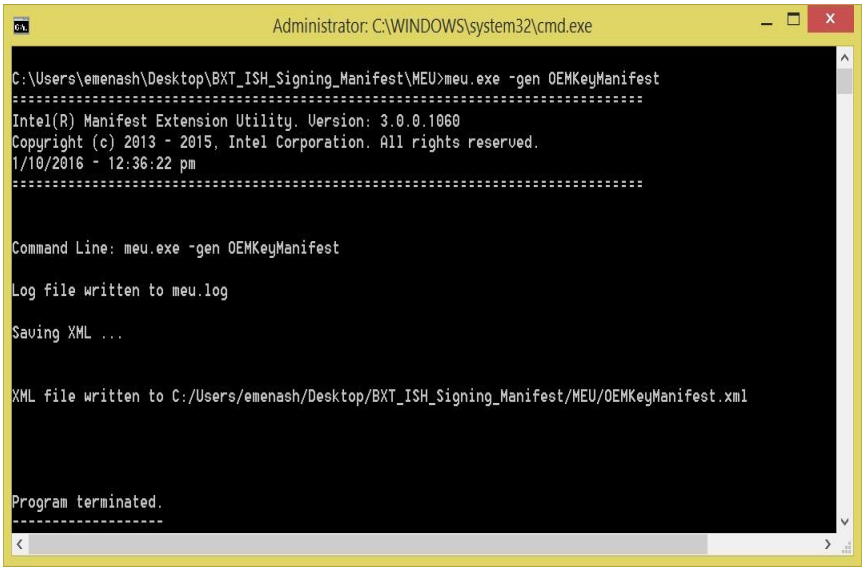
See the hash value shown on the screen:

BE 07 F6 47 3D FE 9B 9C 32 10 FC 51 77 9B 19 AB 60 62 DE 1B 5C D0 07 D2 FD 20 7A 59 88 AF 23 D8

3.9 Creating Manifest File

Now that the ISH image was signed and you have prepared hash files for ISH firmware and other components (only if needed…), it is time to prepare the manifest file itself:

To prepare the key manifest configuration file we will use the MEU tool with the command: meu.exe -gen OEMKeyManifest



The OEMkeymanifest contains two major sections that refer to ISH and all other components. Since we have created ISH and other components hash files – we need to enter in the file the hash files locations.

1. Modify the OEMKeyManifest.xml:

   Remark: the OEMKeyManifest file is configured uniquely by each OEM and it depends on the OEM firmware configuration priorities.

   At the following section I am showing a common configuration where the key manifest file is divided into 2 nodes – one for ISH with a unique key and the other for all the firmware section that are being signed with a different key.

   For more comprehensive data about manifesting –refer to signing and manifesting guide.

   a) *First one – ISH manifest and OemManifest*
      i. *KeyManifestEntry Usage = IshManifest*
      ii. *HashBinary value= Path to binary file containing Public Key Hash (in this case -* **Hash_bxt_ISH is the hash file produced in previous section)**

   b) *Second one – all other sections that are **not** ISH*
      i. *KeyManifestEntry Usage = BootPolicyManifest | iUnitBootLoaderManifest | IfwiManifest | iUnitMainFwManifest | cAvsImage0Manifest* ii. *HashBinary value= Path to binary file containing Public Key Hash (in this case* **general_hash_file is the file we have produced to be used with all other components. But it can come from other groups as the organization decides***)*

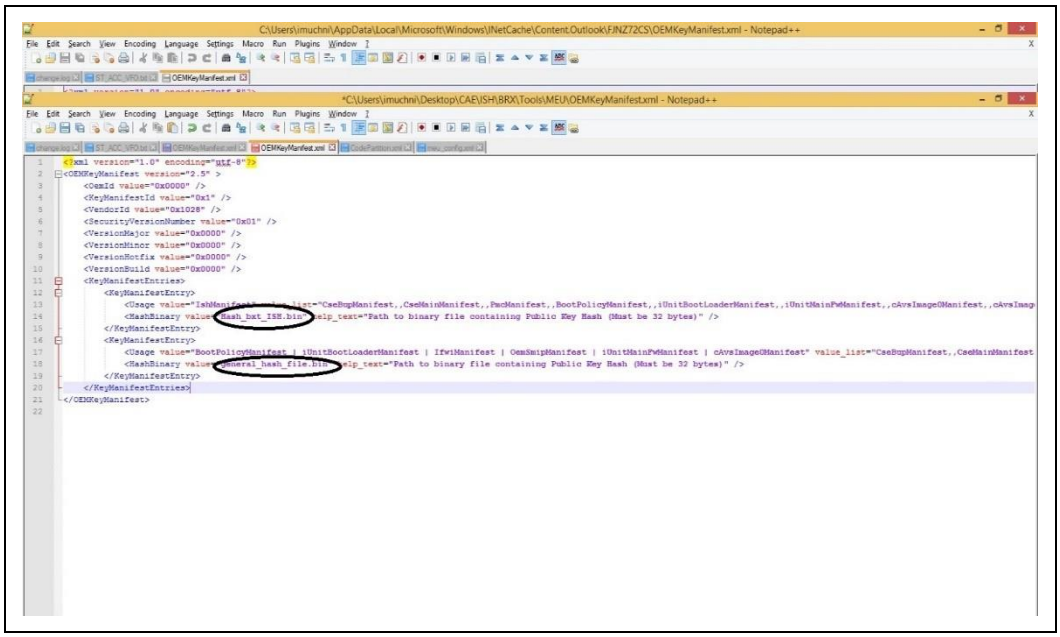


Figure 3-8: OEMKeyManifest.xml file shows Places were Hash Files are Inserted

Now that we have the key manifest configuration file we need to make the actual manifest file and sign it with the key provided as parameter to "-key" command argument.

The command is:

meu.exe -f OEMKeyManifest.xml -o OEMKeyManifest.bin –key <the key file decided to sign the manifest>

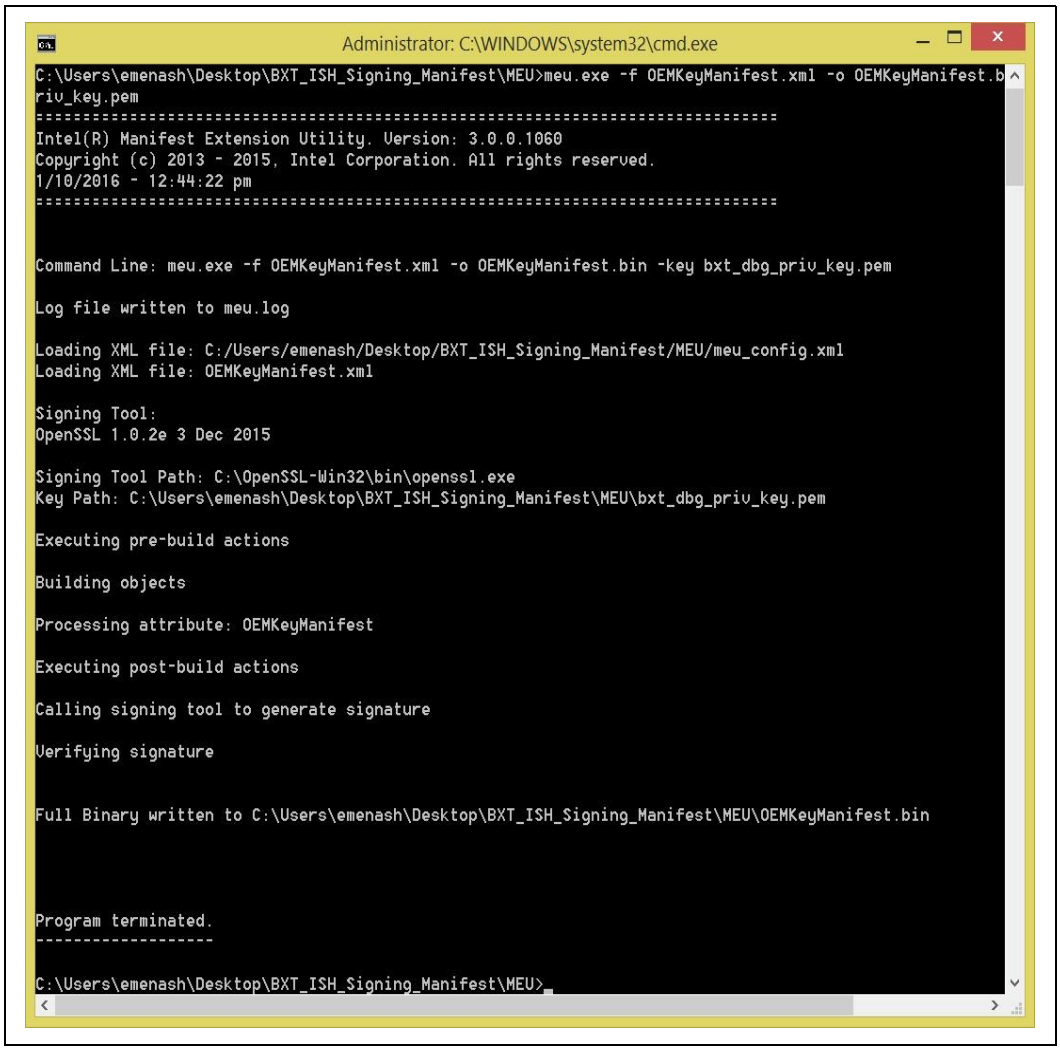


Figure 3-9: Creating and Signing Manifest

The signing and manifest creation is finished.

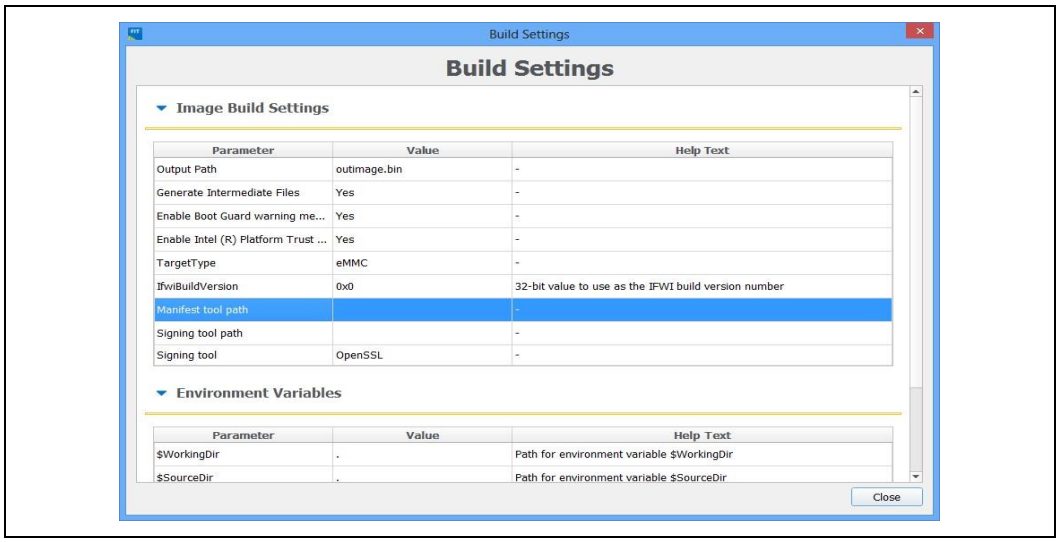
3.10 Creating the Image

This part is optional as it is described in the TXE/IWFI bringup guide. The section is brought as reference only. If any inconsistency exists – the TXE bring up is the official source for data.

We need to configure FIT tool to be able to call MEU use OpenSSL tool for SMIP signing.

1. Goto the following screen in FIT:

Figure 3-10: - Build Setting Tab



And update:
Manifest tool path – MEU path

Signing tool path – OpenSSL path
Signing tool – OpenSSL

2. Goto Platform Protection tab => Platform Integrity => SMIP Signing Key => <the key file that supposed to sign SMIP. Must be compatible to the key file entered into OEMKeyManifest.xml file>
3. Goto *Platform protection Tab => Platform Integrity => OEM Public Key Hash => enter the hash string (actual numbers) generated from making hash of the above key file>*
4. Goto *latform protection Tab => Platform Integrity => OEM Public Key Hash => OEM Key Manifest Binary => <enter the manifest bin file -* OEMKeyManifest.bin>

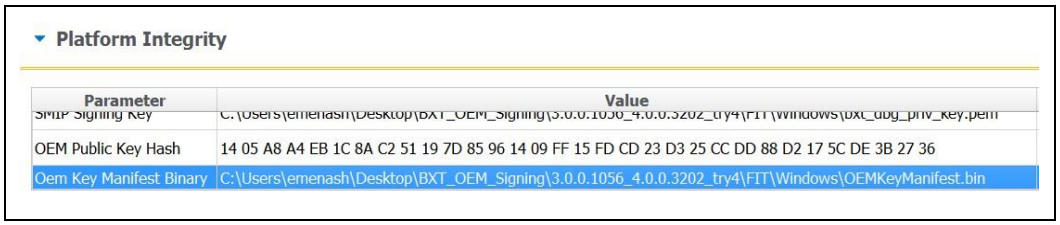
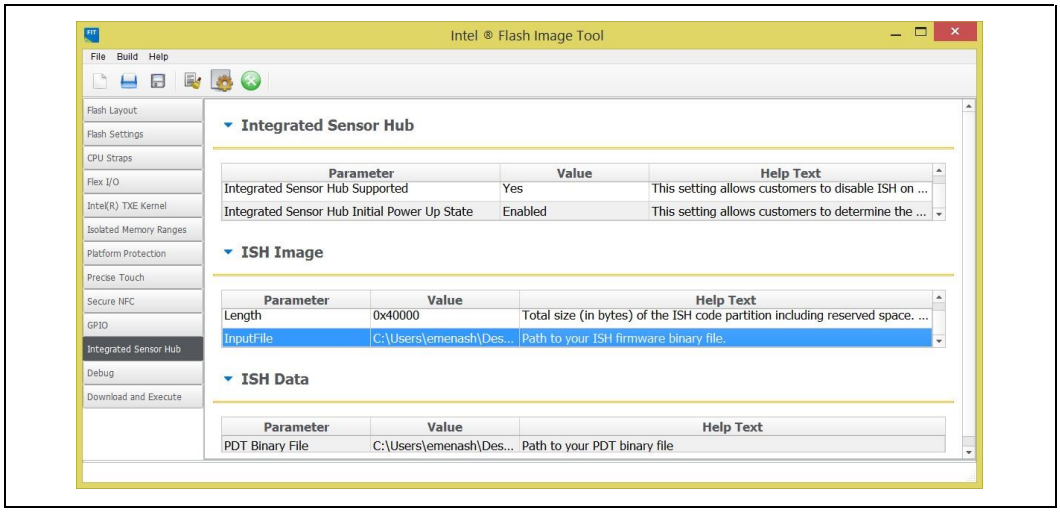


Figure 3-11: Platform Protection Tab

5. Goto *Integrated Sensor Hub Tab => ISH image Table => <add the signed ISH fw image path>*

6. Goto *Integrated Sensor Hub Tab => ISH Data => <Add the PDT file path>*



This concludes the ISH firmware signing flow.

Figure 3-12: ISH Tab

3.11 ISH Bring Up Image

This part is mandatory for platforms after ADL.
When OxM sign ISH image by manual after ME@16.0.0.1302(for ADL), it must have ish_bup.bin which is from FDK folder.Or, system cannot boot successfully.

1. Copy the ish_bup.bin from C:\Intel_ISS_FDK\Tools\EclipsePlugin\FDKTools\versions\ADL\5.4.2.4547(or other version of your ISH FW)
2. Add one more "CPDataModule" with the location of your ish_bup.bin in your Codepartition.xml

```
<CPModules>
    <CPDataModule name="ish_bup" enabled="true">
        <InputFile value="C:\I AM LOCATION\ish_bup.bin" />
        <CompressionType value="NOT_COMPRESSED" value_list="NOT_COMPRESSED,,LZMA" />
        <ProcessId value="0xf5" />
    </CPDataModule>
    <CPDataModule name="ish_main">
        <InputFile value="ish_main.bin" help_text="Path to binary file to load for this module's
        <CompressionType value="LZMA" value_list="NOT_COMPRESSED,,LZMA" help_text="Select compres
        <ProcessId value="0xf6" />
    </CPDataModule>
</CPModules>
```

Figure 3-13: ish_bup entry for CodePartition.xml

§

4 Modifying PDT

4.1 Before Modifying PDT

- Store the ISS default PDTs onto your system [Intel® ISS FW Kit]
- Install the PDT Editor tool to your system [Intel® ISS FW Kit]
- Modify the PDT to include 3rd party micro drivers & algorithms [Optional if creating custom FW using the FDK]

Refer to the PDT Editor user-guide for more detailed usage of this tool. The userguide is located within the ISS FW kit and the ISS FDK.

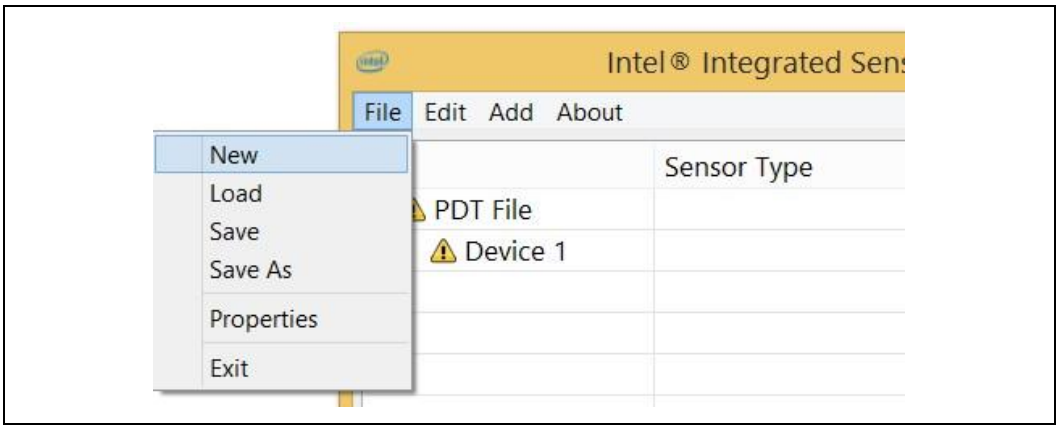
4.2 Expected Outcome

PDT Binary compatible with the sensor configuration of the SUT

4.3 Steps to Modify PDT

1. Run PDTEditor.jar
2. Load PDT.bin by clicking File->Load PDT File and select a pre made PDT.bin.

Figure 4-1: Loading PDT into the PDT Editor



3. After the PDT is loaded it will display the physical and virtual sensors (algorithms) included in the PDT.

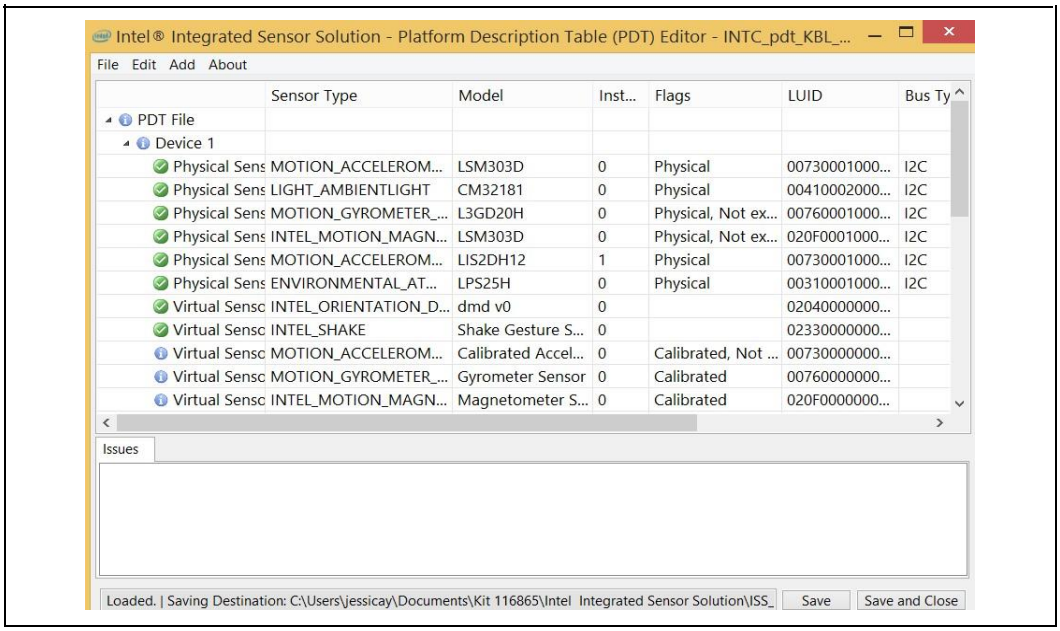


Figure 4-2: PDT Editor View after Loading Default PDT

4.To add a sensor – click "Add New Physical Sensor". Then select the sensor type, vendor, and model...etc in the drop-down box.

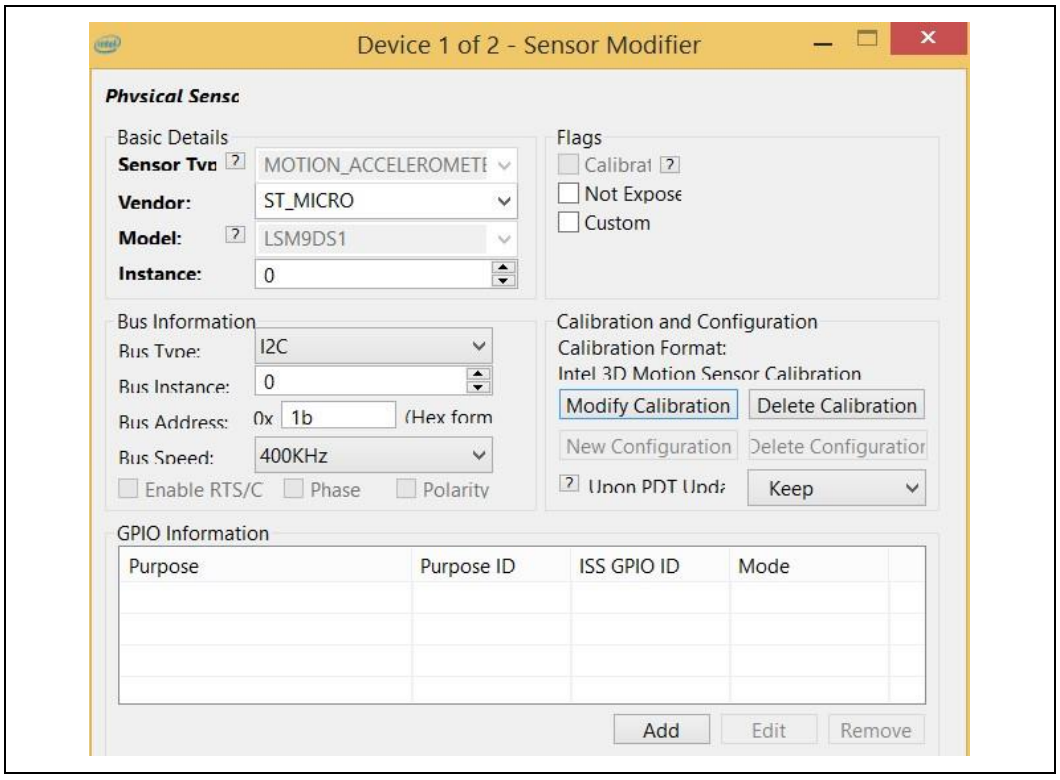
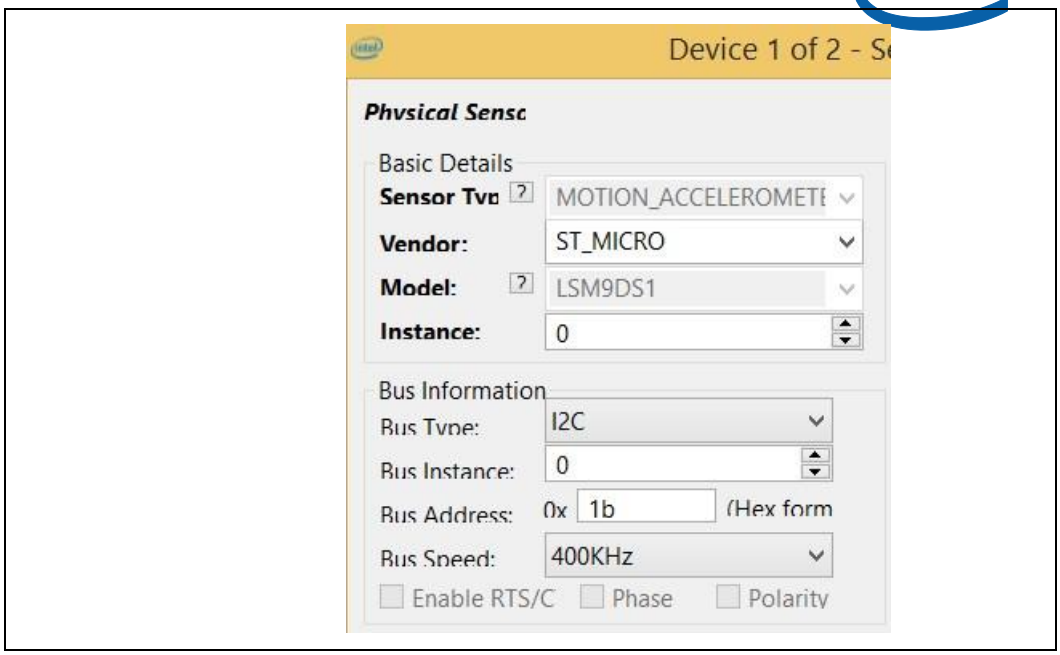


Figure 4-3: Add Sensor Micro Drivers

5. Select each physical sensor that requires editing by double-clicking it.

Figure 18 - Detailed view of Physical Sensor

Figure 4-4: Detailed view of Physical Sensor

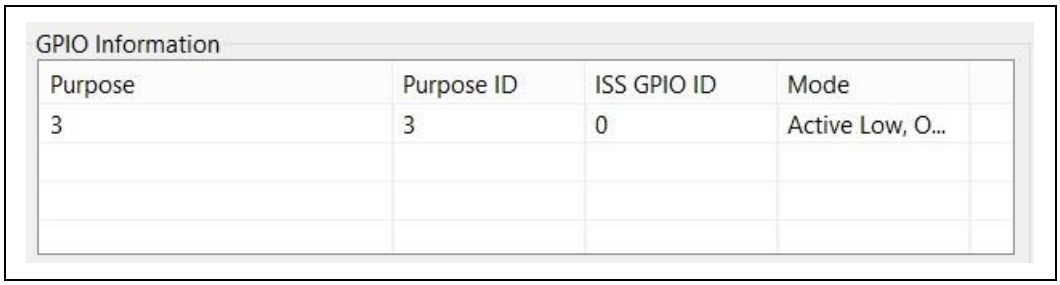


6.Modify the sensor bus address (I2C address) if needed to avoid I2C conflicts. Assign or change the GPIO address (Optional) assigned to the sensor.

a. Double-click on the GPIO section of the physical sensor

b. Edit the GPIO information in the "Interrupts Modifier" window

Figure 4-5: View of GPIO Editor Window



7.Click to save the PDT as a PDT binary file.

§

5 Adding ISS Components to Binary Image

5.1 Before Modifying PDT

- Create a binary image and XML configuration file that includes the BIOS, ME, and GBE region.
- Modify the ISS PDT to match the SUT sensor configuration

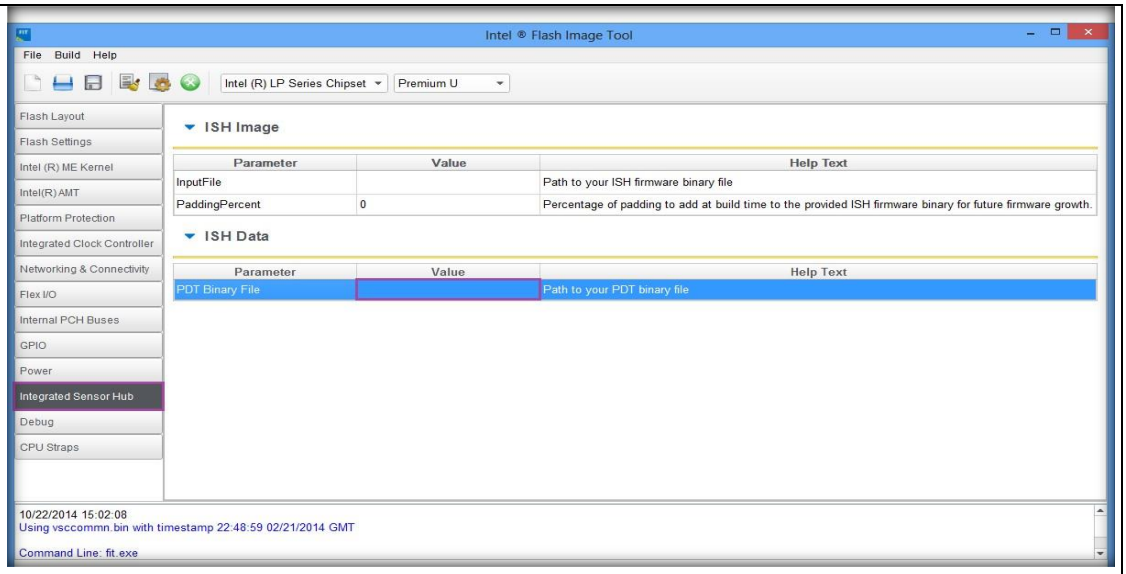
5.2 Expected Outcome

Binary image with ISS that can be flashed to the SUT

5.3 Steps to Add ISS Components to Binary Image

1. Open Flash Image Tool (FIT) and copy to the FIT folder these files:
   a. Existing SPI flash image.
   b. New ISS Code signed and compressed.
2. Run FIT.exe and drag the SPI flash image into the GUI.
3. Navigate to and press on the 'Integrated Sensor Hub'.
4. Press on the Browse button of "ISH Image" and load the new Intel® ISS FW Code bin file.
5. Press on the Browse button of "ISH Data" and load the PDT binary file.
6. Build image (Ctrl+B).
7. Expected result will be an SPI flash binary image created in the FIT folder including ISS FW components.

Figure 5-1: View of FIT Tool when Adding ISS FW Components



Input File	Location for the ISS FW binary. Provide the FIT tool with the location of the binary that should be added to the IUP within the ME region.
Padding Percent	Percentage of extra space to be added to the IUP. • If no value is set – the default IUP size is 256KB • If padding percentage entered – this will be the size of the padding added above the size of the ISS FW binary
PDT Binary File	Location for the ISS PDT binary. Provide the FIT tool with the location of the binary that should be added to the IUP within the ME region.

§

6 Installing ISS Drivers

6.1 Obtaining ISS Drivers

The drivers may be obtained from the software zip file inside the ISH kit. They can be installed to a known location. The drivers included are:

For RS1:	Windows* 10
ISH HECI Driver	✓
ISH Bus Driver	✓
HID PCI Miniport Driver	✓

For previous RS1:	Windows* 10
ISH HECI Driver	✓
ISH Bus Driver	✓
HID PCI Miniport Driver	✓
AdvSensorHIDClassDriverV2	✓

Note: The (✓) means that the driver is required for the operating system version. Android drivers are part of the Android image.

Warning: Use the SW drivers and FW binary from the same FW kit.

6.1.1 Purpose for each driver

a. **ISH.sys** is lowest driver which provide ipc/heci communication to talk with PCI devices – ISH hardware.
b. **ISH_BusDriver.sys** is for sensor HID client to handle HID messages.
c. **HID_PCI.sys** is the miniport driver which is a filter driver to map Microsoft mshidkmdf.sys.

6.1.2 Location

The drivers are installed to **C:\Windows\System32\DriverStore\FileRepository**, you can refer to INF file - DefaultDestDir = 13 control this. (No platform dependency.)

6.2 Installing ISS Drivers – Windows*

There are 2 methods of installing the drivers on the machine:

- Manual installation

- Using a installer file

6.2.1 Exe File Installation

Run the SetupISS.exe file (from <unzipped folder>\ISH_SW\OS Components\Installer) with administrator privileges.

Once finished, it will show ISS stacks in Windows* Device Manager.

6.2.2 Manual Installation

For each one of the drivers in section 5.1 do the following:

1. Enter the driver folder (<unzipped folder>\drivers\<os version>\<driver name>\)
2. Right-click the .inf file and choose install

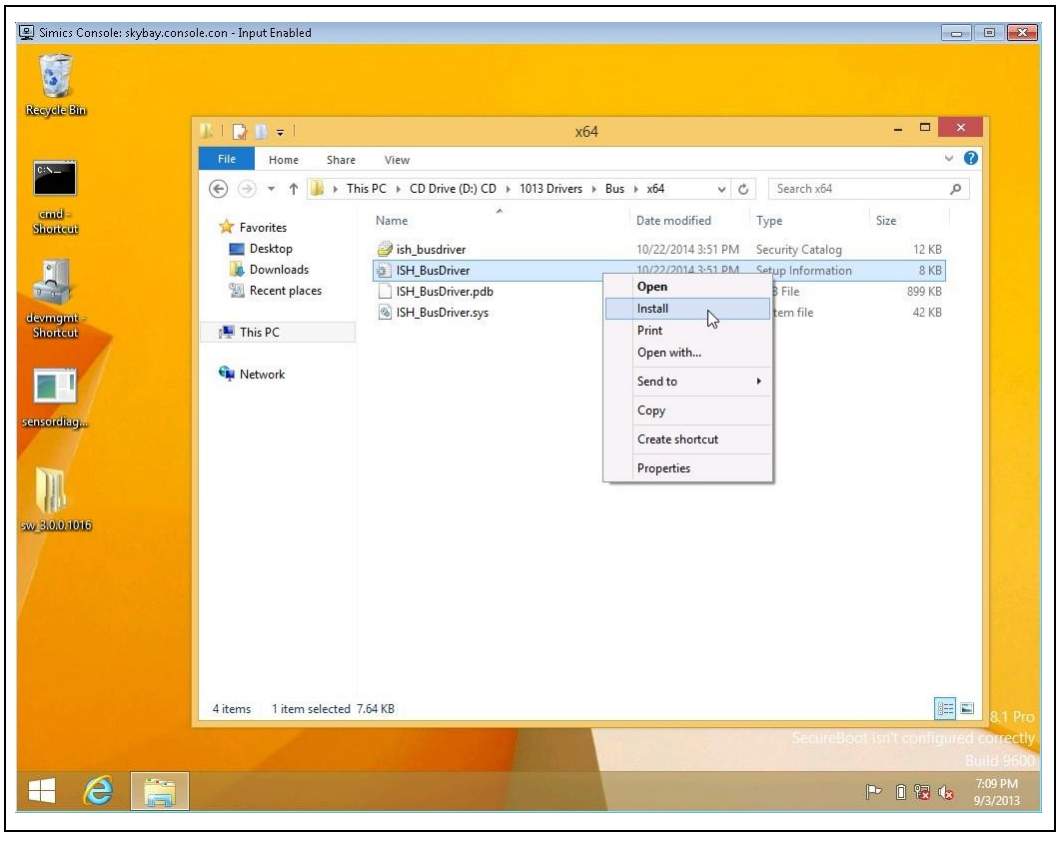


Figure 6-1: View of SW Driver Installation

After installation of all drivers, it is recommended to restart the machine.

6.3 Uninstalling ISS Drivers

Uninstall old installer from Control Panel, then uninstall drivers from Windows* Device Manager with "delete the driver software for this device" selected manually with administrator privileges.

§

7 Confirming ISS Status

7.1 Before Checking ISS Status

Complete the previous sections including (a) adding ISS FW components to the binary image, (b) writing the binary image to flash, and (c) installing the ISS SW drivers.

Locate the **ISSUtil** tool located within the Intel® ISS tool kit.

7.2 Check ISS Status

1. Use the ISSU to check the FW status from the Windows* Desktop:
    - Open the command prompt as an "Administrator"
    - Run the following command: *"ISSU.exe -INFO"*
    - Verify that the tool returns the following status:
        - ISS state
        - ISS Intel FW status
        - ISS Intel FW Version
        - PDT version
        - PDT Description
        - CSE FW version
        - Driver version
        - ISS Sensor Information

7.3 Check Sensor Information

1. Use the ISSU tool to check the FW status from the Windows* Desktop:
    - Open the command prompt as an "Administrator"
    - Run the following command: *ISSU.exe –INFO*
2. Verify that the tool returns the following information for each sensor:

- Sensor LUID
- Sensor Name
- Vendor
- Sensor Sub Type
- Bus Type
- Bus Address
- Calibration Status

7.4 Check ISS FW Functional Test

Use the ISSU to check the FW functional status from the Windows Desktop:

1. Open the command prompt as an "Administrator"
2. Run the following command: *ISSU.exe -bist -test<x> -verbose:*
    - <x> can be one of the following:
    - 1 – referring for sensor connectivity test
    - 2 – referring for sensor calibration test
    - 3 – referring for sensor BIST (Basic Internal Self Test)
3. Verify that the tool returns "test pass for all sensors".

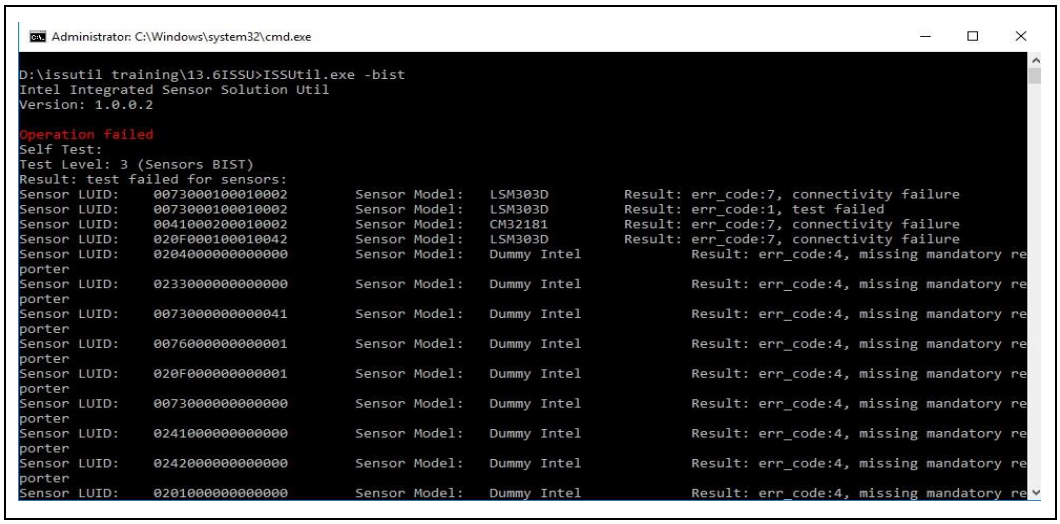


Figure 7-1: View of Functional Test Results Displayed by ISSUtil Tool

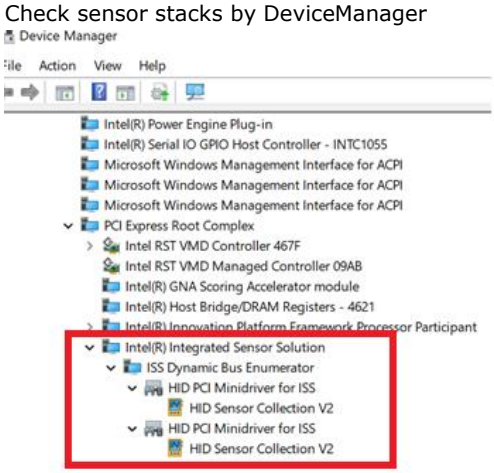
Note: In this example the tool was run without –verbose command.

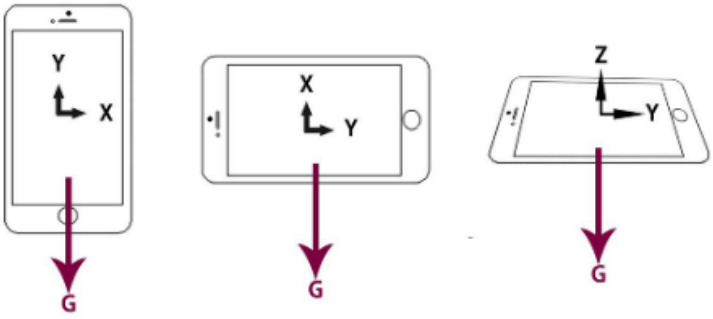
§

8 Common Bring Up Issues and Troubleshooting Table

8.1 Common Bring Up Issues and Troubleshooting Table

Problem / Issue	Solution / Workaround
MEU tool fails to run	Confirm that the MEU_Config and CodePartition files are present in the same folder of the MEU tool. Confirm that both files have been modified properly. *If it still failed, add option "-verbose" into end of command line and provide PDT image.
ISSU BIST test fails	Review the PDT table added to the binary image: • Re-confirm all I2C addresses for each sensor • Check that the sensors listed in the PDT table match those physically present on the system *If it still failed, add option "-verbose" into end of command line and provide PDT image. • Provide FW log
The ISSU BIST "level 3" test for ISH returns a warning for one or more sensors. (i.e. test unsupported)	Included is a list of possible causes and resolutions for sensor failures listed by ISSU: • *Missing calibration* – this failure is raised if the calibration values are set to their defaults (all 0's). This can be resolved by re-running the calibration for this sensor type. • *Connectivity failure* – this indicates that the FW was unable to reach the specific sensor listed. The root-cause can include (a) configuring the wrong I2C address or (b) a HW issue affecting the sensor or the I2C link. • *Test unsupported* – this may related to a to the sensor's built-in self-test. This warning will be raised if the sensor u-driver does not support a sensor self-test that can be called by ISSUtil. • *If it still failed, add option "-verbose" into end of command line and provide PDT image.
The ISSU BIST test for ISH returns sensor is not active or missing sensors	• Make sure that udriver already includes your ISH image o Run command "ISSU.exe -info -sensorlist iss.bin" with unsigned ISH image. • Make sensor already adds into PDT image and all configuration is correct(like i2c bus, i2c speed…etc) * help to get FW log by ISSTraceCollector/FTDI and get SelectedSensors.xml • (workspace\ISS_Sample\settings\SelectedSensors.xml)

Sensor stacks are still showed when the ISS is not enabled	• Check the BIOS to for a setting that has disabled for ISS. • Check BIOS code for ISH enabling.
The ISS is enabled but the sensors are not completed.	• Check the system design to confirm that the I2C addresses of each sensor matches those that are programmed in the PDT (sensor configuration) If this is the problem – use the PDT Editor tool to edit the PDT and correct the problem. Add the updated PDT to the binary image and write this to the system. *If it still failed, add option "-verbose" into end of command line and provide PDT image. • Check sensor stacks by DeviceManager  • Check FW status by ISSU (ISSU.exe -info) • Check sensor status by SensorViewer (check Desktop API in settings) • Check FW log • Check I2C signal by scope. • Check ISH_GP pins in BIOS code. *Please refer Platform EDS document for ISH_GP settings in BIOS
YB issue at sensor stacks	Provide following information: • ME/TXE/ISH/OS version • Failure rate/reproduce way/recovery way • PDT image • Version of drivers (HECI, BUS, MINIPORT) • Error code which is showed by DeviceManager • Check sensor status by SensorViewer • Check FW status by ISSU • Event log, full ramdump(if BSOD), ISH.etl, FW log **NOTE:** Instruction to get full ramdump
Function of PLM sensor does not work	• SUT must be calibrated. • Check value of XYZ-axis from physical accelerometer sensor as below picture

	 x: 0 y: 1 z: 0 — x: 1 y: 0 z: 0 — x: 0 y: 0 z: 1 • Check data of Accelerometer sensor/Gyrometer sensors/Hinge sensor/Hall sensors. It means that data should be correct when ISH calculates hinge angle. • GPIO pins should be configured correctly in BIOS and PDT If GPIO works, check EC part.
Cannot build ISH project by FDK	• Use "Troubleshooter" to see what's software missing (Just click "fix it" if your PC does not have such of SW) • Provide error log, hole workspace folder and FDK version.
Hall sensor does not work(Lid close detection sensor, tablet mode sensor…etc)	• Make sure GPIO pin is already configured correctly in BIOS and in PDT • Dump GPIO information from HW directly by Intel BIOS GPIO Tool in S0 • GPIO purpose for Hall sensors should be 0 • Check signal of GPIO pin directly
Senor Lost	• Check sensor stacks by DeviceManager • Check sensor status by SensorViewer • Provide reproduce way/recovery way/failure rate/ISH Driver version/ISH FW version/PDT image • Provide system log, ISH.etl • Provide FW log through FTDI if reproduce way include power state changes(S3/S4/S5/MS mode). If not, provide FW log through HECI.

NOTES:

There are two ways to get full ramdump.

A. Get <<trace_enable_ISH.reg_>>
   1. Enter regedit
   2. Delete (if exists) the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\WMI\AutoLogger\ISH folder
   3. Run attached trace_enable_ISH.reg (remember to rename from trace_enable_ISH.reg_ to trace_enable_ISH.reg before execution)
   4. Restart the platform
   5. Reproduce the issue again
   6. Create a dump by pressing ctrl+scrlck twice.

7. Send the BSOD dump file to Intel (Make sure it's a completed dump).

If you have problems to get BSOD crash by pressing ctrl+scrlck, try to find and download a freeware tool “Notmyfault” from internet and run this tool to replace step6.

B. Get trace_enable_ISH.reg from CE team Below is snippet of this reg script:

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\WMI\Autologger\ISH]

1. "Start"=dword:00000001
2. "Guid"="{A5371B2E-0B01-45F5-81B5-EFBC009B117C}"
3. "FlushTimer"=dword:00000001
4. "Status"=dword:00000000
5. "LogFileMode"=dword:00088101
6. "ClockType"=dword:00000002
7. "FileName"="%SystemRoot%\\System32\\LogFiles\\WMI\\ISH.etl"
    a. Double click to apply the reg file
    b. Control Panel -> System -> system properties -> startup and recovery -> change to "complete memory dump"
    c. Reboot
    d. Do your test.
    e. When issue occur run the command prompt as admin navigate to the folder contained the tracelog64.exe tool and type: tracelog.exe -stop ISH
    f. Upload the following file to your AE.

§

9 Debug Pre-requisites

9.1 Overview

This document describes the standard set of requirements for submitting issues to the Intel® Integrated Sensor Solution (ISS) Debug team. This facilitates debugging of submitted issues.

The target audience for this document includes OxMs, IHVs, Intel AEs and all those submitting issues to Intel.

9.2 Requirements and Log Requests

9.2.1 Mandatory Information for Sightings

For all newly reported issues, Intel requires the following information:

- **Detailed reproduction steps** - with actual and expected results. It is highly recommended to include supporting information such as screen shots, video etc.
- **Full SPI flash dump after the failure** - The firmware contains different system configurations, such as ME information and critical logs, as well as ISS information and PDT.
- **PDT File** - In case full image cannot be provided, for review and faster reproduction.
- **OS version and build number, ISS FW and SW version –** Can be found in syscope log
- **Answers to the following questions:**
  - Can the issue be reproduced on an RVP?
  - Can the issue be reproduced on a simulator? (such as Simics)
  - Any other information that is relevant to the issue

9.2.2 Additional Information Requested

In addition to the above requirements and in order to speed up the debug process, Provide some of the following information as relevant to the case:

- **ISS FW Log** – Collect trace messages from firmware, it will help us to understand what the FW status is and history on what happened during the reproduction.

Note: Refer to chapter 2.1 Retrieving ISS FW Logs for more information

- **Customer system** - When submitting an issue that is replicated only on customer system, make sure to ship a system to Intel ASAP so the shipping does not delay the debugging process. For shipping information see D Appendix IV Shipping system to Intel
- **System Tools Logs** – Provides simple test to check whether the Intel® ME or Intel® TXE FW are alive. In case no Intel® SST log available, provide ISSUtil logs to put more light on system's status:

For ME:

```
MEInfoWin                       > meinfo.txt
MEInfoWin -fitver               > fitver.txt
ISSU -BIST -verbose          > issu.txt
```

For TXE:

```
TXEInfo                          > txeinfo.txt
TXEInfo -fitver                 > fitver.txt
ISSU -BIST -verbose          > issu.txt
```

- **Customer BIOS** – Clean System BIOS image which never ran on a platform with relevant ISH FW version. BIOS version information and reference code version. It will help reproducing on customer's platform.
- **uDrivers & Algorithms Compiled files** – either *.o or *.a files of the uDrivers and algorithms included in the PDT (if they are not part of Intel's built-in code)
- **System scope log** – Run the System scope tool on the platform and save the log as txt.

  For more information, refer to the Syscope section in the Intel® Management Engine Compliance and Debug Kit

- **Intel® Trace Hub and Intel® Trace Tool (NPK)** – Closed chassis interface for tracing different platform components including Intel® CSE. For sensor hub bringup issues, include STT log with CSE messages. For more information, refer to "Intel® Trace Hub & Direct Connect Interface Implementation Guide for Skylake Platform", CDI #548786.
- **Latest schematics, board files, system layout and relevant documentation** – When I2C logs should be extracted from a closed chassis platform, the schematics and layout are necessary to perform the rework. In addition, when a HW issue is suspected those documents are needed for isolating the issue.

9.2.3 Issue Specific Logs (as Applicable)

- **Simics/Emulauncher issues** – The Wind River Simics hardware simulator is included in the Intel® ISS FDK. It can simulate the full target system or only the firmware to create a shared platform for software development. For Simics/Emulauncher issues, include all the Simics logs. These are located at <FDK installation folder>\Tools\Emulauncher and are named as follows:
  - debug-log.txt
  - debug-log-console.txt
  - debug-log-ish-console.txt
- **Calibration and data error Issues** – The following logs need to be provided:
  - **Calibration Tool Logs** – Include all the files which were generated in the calibration tool folder after running calibration flow (before reproducing the issue).
  - **Sensor Records** – For calibration issues or problems with the sensors data include all relevant sensor records, including physical sensor, un-calibrated sensor, and calibrated sensor data of AGM and algorithm data. Do this as follows:
    1. Open the ISS Sensor Viewer.
    2. Click **Record**.

3. Select sensors to record and click **Start Recording**.
4. To stop recording, click the red button, enter a name for the record file, and click **Save**.

A message informs where the record is located. You can click **Open Location** to access that folder.

- **SW Issues**
  - **For any SW issue** – Include the Windows* Event Log and a screen capture of the device manager, for better understanding of OS status.
  - **For Intel® ISS Drivers issues** - Include Intel® ISS trace prints.

    Refer to chapter 2.2 Retrieving ISS Driver Logs for more information.
  - **For Blue-Screen errors** - Include the Kernel-Dump file.

    Refer to chapter 2.3 "Locating the Windows Dump File" for more information.
- **Documentation issues -** For documentation issues, include document name, revision, and document ID. If possible, also include details (page, section, specific issue).
- **Intel® AWS Issues** - The Intel® Automated Workflow Suite is a tool used to perform ISS compliance testing. For any sighting reported with AWS, include AWS logs and any additional information (such as, do compliance test(s) pass when performed manually or in a different test environment or process).
  Enable Intel® AWS to include Intel® SST logs with Intel® AWS logs.

Note: Refer to AWS User guide in the PETS release for more information.

§

10 Retrieving Logs and Dump Files

10.1 Retrieving ISS FW Logs

FW logs are better being captured using FTDI (I2C) rather than HECI. Follow the steps below to retrieve ISS FW logs:

1. Modify trace configuration using "Trace Config Tool":

   a. Create an xml configuration file. See Appendix II – XML config file for XML content.

   b. Run the trace config tool (from the ISS FW kit) with the below command: "TraceConfigTool.exe setconfig <configFile> [volatile]"

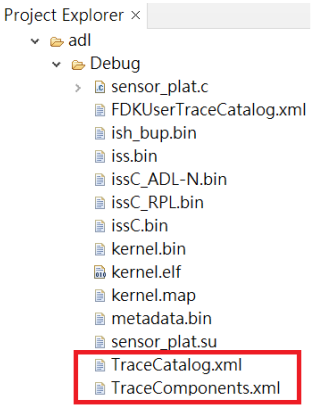
      - Volatile: if volatile, the configuration won't be saved in FW after reset ISS.
      - TraceConfigTool can run and change configuration during the ISSTraceCollector is running, and change configuration meanwhile.
      - ConfigFile = xml file that was created by the user.
      - Configuration fields in the ConfigFile:
        - Trace interfaces: 1 or more from: HECI (bit 0), I2C (bit 1), LPK (bit 2).
        - I2C Bus.
        - I2C Address.
        - Host debug: (0=disable, 1=enable).
        - Trace mode: buffer disable (0), non-cyclic buffered mode (bit 1), cyclic buffered mode (bit 1+bit 2), block thread (bit 4- if msg queue is full, thread calling syslog() is blocked until queue can accept trace msg).
        - Buffer size: for buffered mode, by KB.
        - Lowest priority: 0=emergency, 1=alert, 2=critical, 3=error, 4=warning, 5=notice, 6=info, 7=debug.
        - Components Filter Mask: components to get traces from. Components xml file generated with the FW. 16 major components, that each one contain 16 sub components.

2. Collect logs with one of the below methods:

   a. Collecting logs through FTDI (I2C) – see section 10.2

b. Collecting logs through HECI (using ISSTraceCollector tool from ISS kit) :

- To be used on the system itself, able to retrieve information starting kernel phase only when buffer mode is disabled
  - Better to have buffer mode enabled in order to capture logs during boot time.
- ISH driver is required to be up and running for retrieving messages
- Run the following command "ISSTraceCollector.exe HECI continues TraceCatalog.xml TraceComponents.xml log.txt"
  - TraceCatalog.xml and TraceComponents.xml files should be found in the ISS kit if ISH image is from ISH kit.
  - TraceCatalog.xml and TraceComponents.xml files are from FDK project if ISH image is built by FDK.



10.2 Collecting Logs through FTDI

Connecting FTDI:

1. Download and install the FTDI driver on your console.
2. Connect the FTDI to the platform. Use either the UMTF201XA or UMTF201XB.

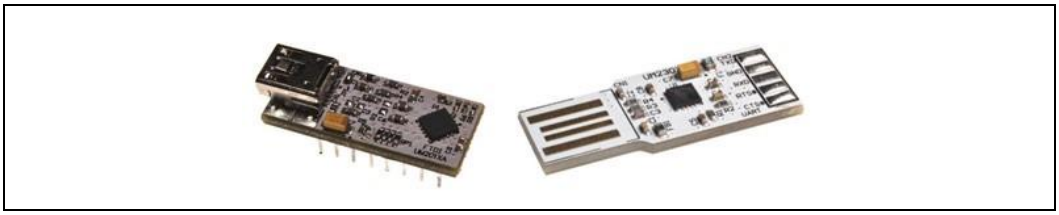


Figure 10-1: FTDI UMFT201XA vs UMFT201XB

The connection uses 3 wires (SDA, SCL and GRD) from the platform to the FTDI. This can be done manually or by dedicated connectors.

The following figure shows a connection via NS.

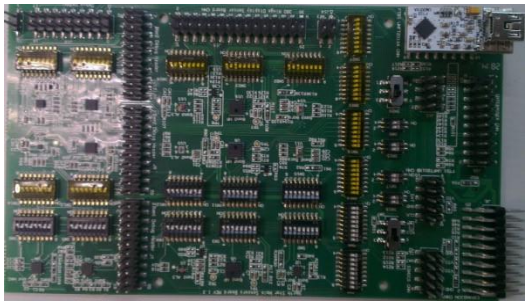
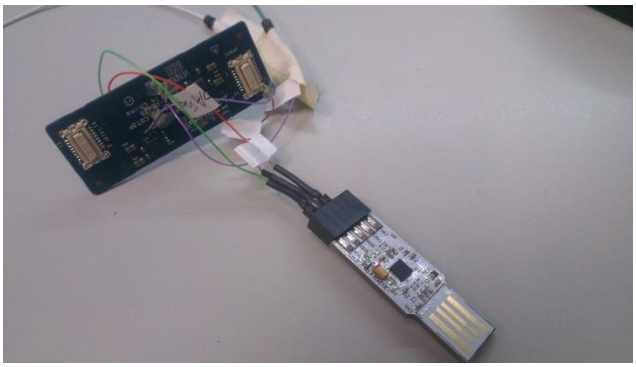


Figure 10-2: FTDI on NS

The following figure shows a manual connection.

Figure 10-3: FTDI Connected via Wires



3. Configure the FTDI (see Appendix III – FTDI configuration for instructions.)
4. Run "ISSTraceCollector.exe I2C_COM <FTDI COM PORT #> TraceCatalog.xml TraceComponent.xml log.txt"

   • < FTDI COM PORT #> is the FTDI port # which was determined in the previous bullet

10.3 Retrieving ISS Driver Logs

This procedure describes how to enable the logging mechanism to collect ISS driver logs.

1. Enable ISS logging in the Windows registry as follows:
   a. Create a new text file, edit it, and copy into this file the contents see Appendix I - Windows Registry Integrated Sensor Solution Logging.
   b. Save the file with a .reg suffix
   c. Run the file with administrator privileges.
   d. Restart the system 2. Run the flow.

3. When you have a reproduction hold the Ctrl button and press the Scroll lock twice to initiate a BSOD
4. Restart the system and find the dump file
5. Create an etl file from the dump that was created (optional but can reduce download time, hence is very recommended) a. Open the dump file using Windbg
   b. Type in the command line:
      !wmitrace.logsave ISH c:\logs\badFlow.etl
6. Upload the etl / dump file , together with the exact versions of the bus, HECI - ISH and miniport driver to the HSD
7. Verify the driver version as follows:
   a. Open the Windows* Device Manager.
   b. Click View→Devices by Connection.
   c. Double click all of the following:

      - Intel® Integrated Sensor Solution
      - Integrated Sensor Solution Dynamic Bus Enumerator
      - HID PCI Minidriver for Integrated Sensor Solution
      - Advanced sensor collection
      - Advanced sensor collection V2

   d. View the driver version in the Driver tab for each of the above items.

Note: In case BSOD is not functioning and flow does not include power transactions, collect SW logs manually:

8. Save the ISS drivers log in a file as follows:
   i. Create a new text file, edit it, and copy into this file the contents of Appendix I - Windows Registry Integrated Sensor Solution Logging.
   ii. Save the file with a .reg suffix
   iii. Run the file with administrator privileges.
   iv. Restart the system
   v. Obtain the Tracelog tool from WDK.
   vi. Open a command prompt with administrator privileges.
   vii. The logging process has been running since driver startup. To stop it, type: **tracelog.exe –stop ISH**. A log file is created and saved in *%SystemRoot%\System32\LogFiles\WMI\ISH.etl* where %SystemRoot% is the folder where system32 folder is located (usually C:\Windows).

10.4 Locating Windows Dump File

1. Right-click Computer, select Properties, and click Advanced System Settings.
2. In the Startup and Recovery area, click Settings.

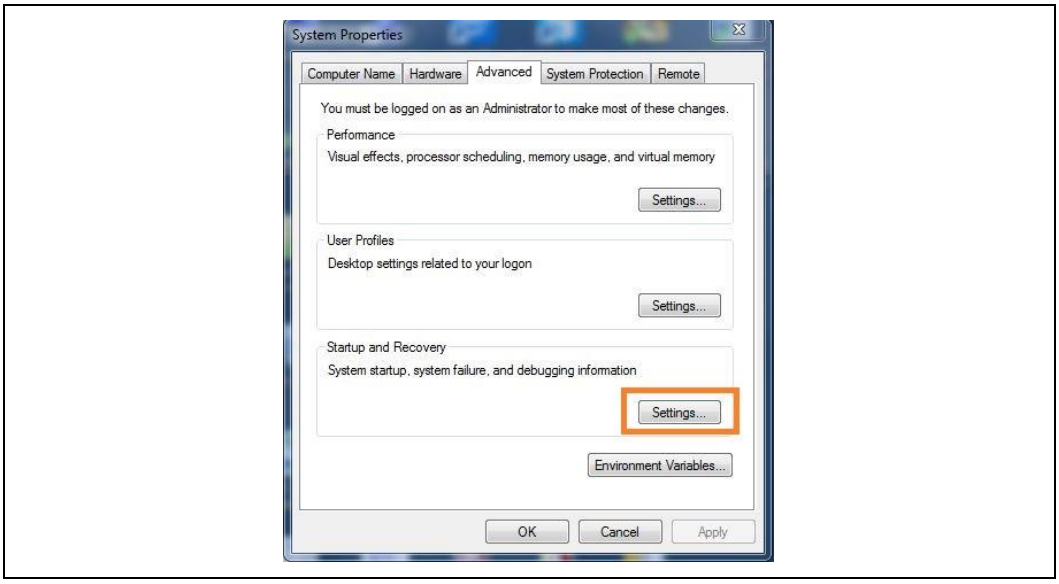


Figure 10-4: Startup and Recovery

3. Note the location of the dump file as displayed in the Dump File field.

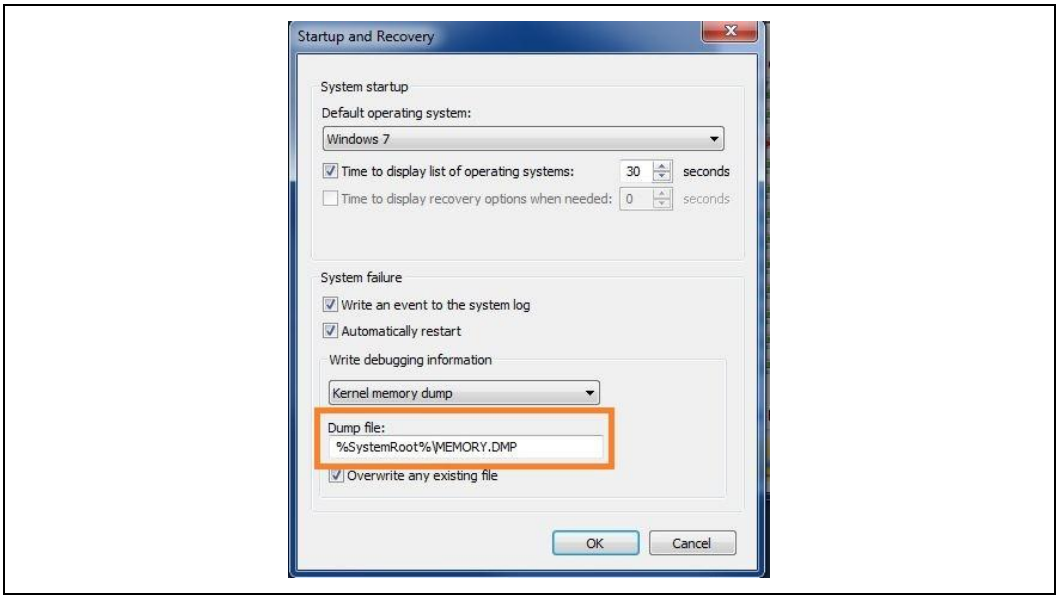


Figure 10-5: Dump File Location

§

11 Reference Documents

Document or Tool	Location
Intel® Management Engine Compliance and Debug Kit (includes Intel® PETS)	VIP
Intel® System Scope Tool (Intel® SST)	VIP Being posted with Intel® ME Compliance and Debug Kit and as a separate kit
Skylake U and Y Platform Design Guide	CDI #543016 Chapter 31.3.2 – "Sensors Debug Hooks" Chapter 44 – "Platform Debug and Test Hooks"
Kaby Lake Platform Design Guide	CDI # 561150 Chapter 30.3.2 – "Sensors Debug Hooks" Chapter 43 – "Platform Debug and Test Hooks"
Broxton Platform Design Guide	CDI # 551546 Chapter 7.6.2 – "Sensors Debug Hooks"
Apollo Lake Platform Design Guide	CDI # 557503 Chapter 3.2 – "Sensors Debug Hooks"
Cherry Trail T3 Platform – Design Guide	CDI #540558 Chapter 6.3 – "SoC Debug Port (XDP) Design Guidelines"
Intel® Trace Hub & Direct Connect Interface Implementation Guide for Skylake Platform	CDI #548786

§

12 FWUpdate for ISH Components

12.1 Purpose and scope of this document

After CNL, ISH partial update is supported.

The document is created to define the flow of ISH FW/PDT update through ME BIOS.

12.2 Flow to update ISH FW in the field

Calling **FwUpdatePartialBuffer()** API via FWUpdateLib to update ISH.
To get more detail, please refer system tool guide from ME kit.

12.3 Flow to update PDT in the field (RS Only)

Since PDT is protected when ME is locked, the PDT UNLOCK is needed. And, it's one time effective.

Here are steps to implement PDT image update in BIOS code:

a. Enabling HECI interface in BIOS code.
To get more detail, please refer CH3.4.7 Sending Receiving Intel® Management Engine Interface (Intel® MEI) Messages from ME BIOS spec.

Platform	Document ID
CML	608436
CNL/WHL/CFL	572579
ICL	574509
TGL	612229
ADL	627331
RPL	679995
MTL	729124

b. Sending PDT_UNLOCK command (MKHI messages) before End-Of-Post.
To get more detail, please refer CH25.6 PDT Unlock Data Message from Platform PCH BIOS spec.

Platform PCH	Document ID
CNL/CML-H	570374
ICL	574625
TGL	613495
ADL	630603
RPL	630603

c. Entering EFI shell without configuration reset if needed.
d. Calling **FwuSetIshConfig()** API via FWUpdateLib to update PDT image.

To get more detail, please refer system tool guide from ME kit.
For this step, **ME mode cannot be in normal mode and be not after End-Of-Post**.

**Note: In development stage, customers could run the following commands to validate quickly. However, please be aware that FWUpdate.efi is not validated. Suggested that customers port reference code into BIOS.

ISH FW:

a. FWupdate – FWUpdLcl.efi -F <image.bin> -PARTID ISHC
b. FWUpdate(RS version) - fwupdate.efi -i <image.bin>

ISH PDT:

a. Disable EPO in BIOS menu
b. FWUpdate(RS version) - fwupdate.efi /d <image.bin>

§

Appendix A – Windows* Registry Integrated Sensor Solution Logging

Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\WMI\AutoLogger\ISH]

"Start"=dword:00000001

"Guid"="{A5371B2E-0B01-45F5-81B5-EFBC009B117C}"

"FlushTimer"=dword:00000001

"Status"=dword:00000000

"LogFileMode"=dword:00088101

"ClockType"=dword:00000002

"FileName"="%SystemRoot%\\System32\\LogFiles\\WMI\\ISH.etl"

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\WMI\AutoLogger\ISH\{739119E6-5336-4582-880D-041989565D5F}]

"Enabled"=dword:00000001

"EnableFlags"=dword:ffffffff

"EnableLevel"=dword:00000005

"LoggerName"="ISH"

"Status"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\WMI\AutoLogger\ISH\{2432b9fb-2987-4602-af88-b237175db491}]

"Enabled"=dword:00000001

"EnableFlags"=dword:ffffffff

"EnableLevel"=dword:00000005

"LoggerName"="ISH"

"Status"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\WMI\AutoLogger\ISH\{D23A0C5A-D307-4f0e-AE8E-E2A355AD5DAC}]

"Enabled"=dword:00000001

"EnableFlags"=dword:ffffffff

"EnableLevel"=dword:00000005

"LoggerName"="ISH"

"Status"=dword:00000000

;----------------------------------trace umdf drivers-------------------------------------

;

;trace provider for adv umdf driver for win8.1

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\WMI\Autologger\ISH\{3804C675-7658-4B7E-AD5A-86E21631B7CD}]

"Enabled"=dword:00000001

"EnableFlags"=dword:ffffffff

"EnableLevel"=dword:00000005

"LoggerName"="ISH"

"Status"=dword:00000000

;trace provider for adv umdf driver for win10

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\WMI\Autologger\ISH\{06C6C767-2179-40E1-8D39-9CD55AC6ABF8}]

"Enabled"=dword:00000001

"EnableFlags"=dword:ffffffff

"EnableLevel"=dword:00000005

"LoggerName"="ISH"

"Status"=dword:00000000

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\i8042prt\Parameters]

"CrashOnCtrlScroll"=dword:00000001

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\kbdhid\Parameters]

"CrashOnCtrlScroll"=dword:00000001

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\WMI\Autologger\Sensors Trace]

"BufferSize"=dword:00000008

"FlushTimer"=dword:00000000

"MaximumBuffers"=dword:00000000

"MinimumBuffers"=dword:00000000

"Start"=dword:00000001

"FileName"="C:\\Windows\\tracing\\SensorsTrace.etl"

"ClockType"=dword:00000001

"MaxFileSize"=dword:00000000

"LogFileMode"=dword:00001200

"Guid"="{384195BC-9E6C-4493-BE41-3993902CAAEC}"

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\WMI\Autologger\Sensors Trace\{32E38BD8-AC95-4605-894E-A5D815CA0F3B}]

"Enabled"=dword:00000001

"EnableLevel"=dword:000000ff

"EnableProperty"=dword:00000000

"MatchAllKeyword"=hex(b):00,00,00,00,00,00,00,00

"MatchAnyKeyword"=hex(b):ff,ff,ff,ff,ff,ff,ff,ff

§

Appendix B – XML Config File

FTDI :

```
<?xml version="1.0" encoding="UTF-8"?>
<myConfiguration>
  <TraceInterfacesMask>2</TraceInterfacesMask>
  <TraceModeMask>1</TraceModeMask>
  <BufferSize>16</BufferSize>
  <UARTPort>0</UARTPort>
  <UARTBaudrate>115200</UARTBaudrate>
  <I2CBUS>0</I2CBUS>
  <I2CAddress>0x5e</I2CAddress>
  <HostDebugEnabled>1</HostDebugEnabled>
  <LowestPriority>7</LowestPriority>
  <CompFilterMask>ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff</CompFilterMask>
</myConfiguration>
```

HECI :

```
<?xml version="1.0" encoding="UTF-8"?>
<myConfiguration>
  <TraceInterfacesMask>1</TraceInterfacesMask>
  <TraceModeMask>1</TraceModeMask>
  <BufferSize>16</BufferSize>
  <UARTPort>0</UARTPort>
  <UARTBaudrate>115200</UARTBaudrate>
  <I2CBUS>0</I2CBUS>
  <I2CAddress>0x5e</I2CAddress>
  <HostDebugEnabled>1</HostDebugEnabled>
```

<LowestPriority>7</LowestPriority>

<CompFilterMask>ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff</CompFilterMask>

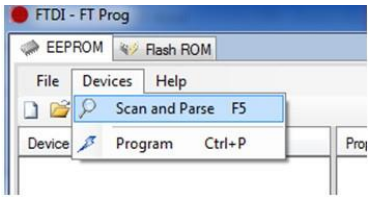
</myConfiguration>

§

Appendix C – FTDI configuration

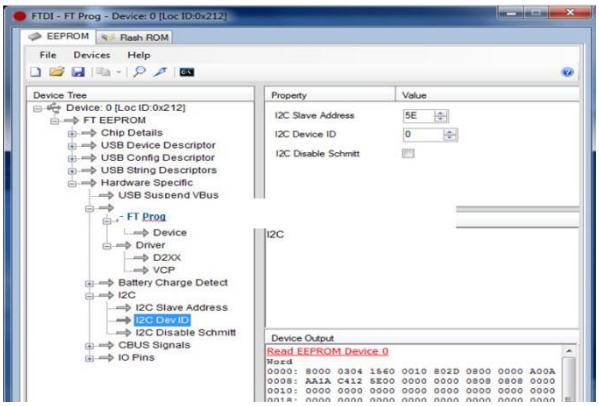
1. Download the FTDI software FT_prog to configure FTDI with FT PROG.
2. Configure the FTDI connector (this only needs to be done once):
3. From EEPROM -> Devices, click **Scan and Parse**. It may take a while until the device is exposed.

Figure 0-1: FT Prog Scan and Parse



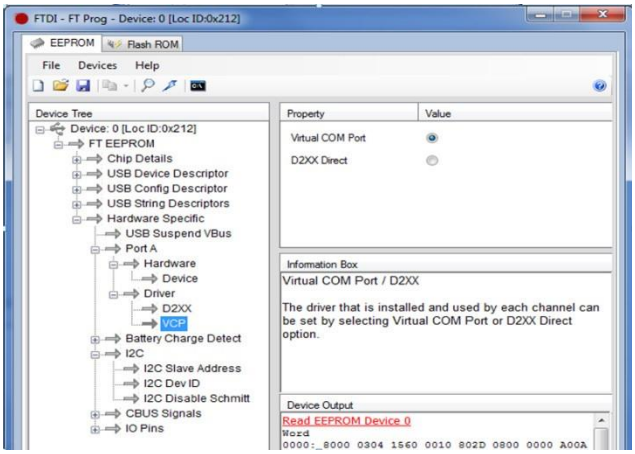
In Hardware Specific -> I2C -> I2C Dev ID, make sure the I2C Slave Address is set to 0x5E.

Figure 0-2: FT Prog I2C Configuration



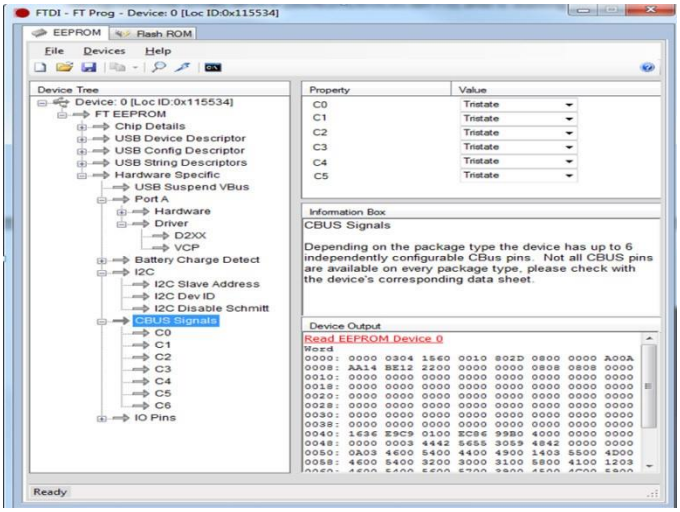
4. In Hardware Specific -> Port A-> Driver -> VCP verify that Virtual COM Port is selected.

Figure 0-4: FT Prog Virtual COM Port



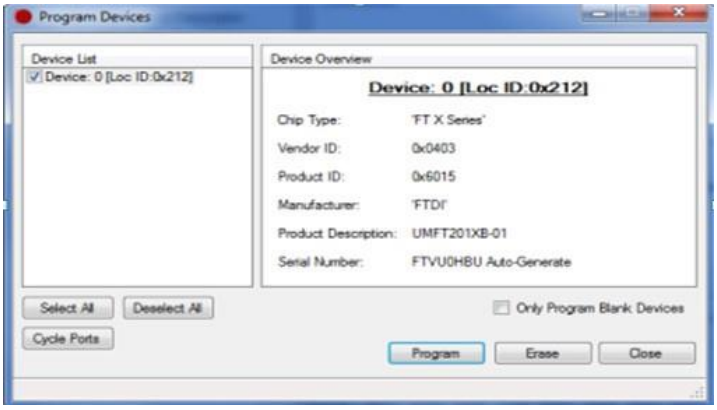
5. In Hardware Specific-> CBUS Signals, change all the signals to Tristate.

Figure 0-4: FT Prog CBus Signals



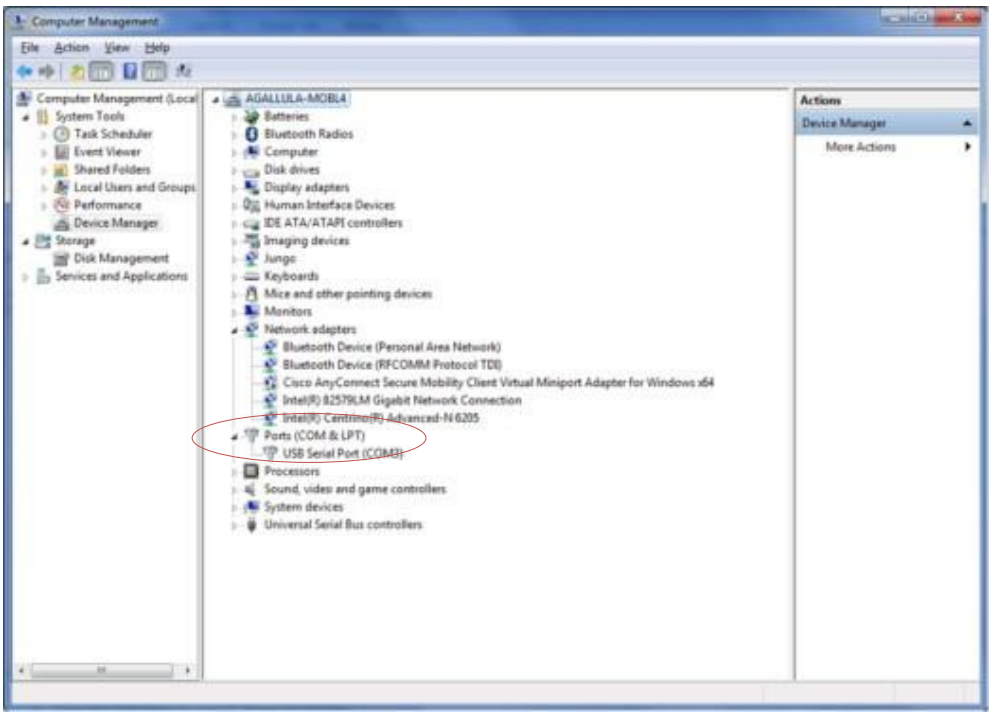
6. Program the device under Devices->Program and click Program.

Figure 0-5: FT Prog Program Devices



7.Find the FTDI COM PORT # under Computer ->Manage->Device Manager->Ports ->USB Serial Port (in the following example, the COM number is 3).

Figure 0-6: FTDI COM Port



§

Appendix D - Shipping System to Intel

When sending a system to Intel, ensure the following information is available:

- **Pre-shipping form –** Ensure to fill the pre shipping form (See below) and send it to the system receiver as soon as you can (recommended before shipping) in order to speed up the delivery.

  Note:

  - If there is no wireless card, there should be a written declaration with the confirmation. Note that this is preferred method to send platforms to Israel.
  - In case there is a wireless card included, the card's details must be shared as well in the pre-shipping form.
  - Note that a system will be released from Israel customs minimum 10 business days after sharing this information.
  - For more information, contact customer enablement.
- **Rework** - Ensure to solder relevant connectors if they have foot print, or to put the relevant reworks if they do not have:

  - **JTAG connector rework** – For source code debugging and HW register reading, refer to the Debug Port Design Guide to design and rework PCH XDP 60-pin or CMC port. For more details, refer to "Skylake U and Y Platform Design Guide" CDI #543016, chapter 44 – "Platform Debug and Test Hooks" or "Cherry Trail T3 Platform – Design Guide"
    CDI #540558, chapter 6.3 – "SoC Debug Port (XDP) Design Guidelines".
  - **Intel® ISS Debug Connector Rework** – for connecting to the ISH I2C buses and ISH GPIOs, refer to the debug connector and rework I2C buses (0 & 1), VIO and relevant GPIOs. For more details, refer to Platform Design Guide
  - **SPI socket** – This is very helpful during debugging, as well as for recovering a system.

Pre Shipping Form			
Category	**Question**	**Answer**	**Notes**
General	Who is the end recipient of the platform (make sure to add him in the CC)		
	Platform type (e.g., Skylake)		
	Code name of the platform (Model)		

Pre Shipping Form			
Category	**Question**	**Answer**	**Notes**
	Battery Working Hours – if more than 40 WH it should be shipped separately)		
Wi-Fi	model name		
	brand name		
	bands		
	SKU		
Bluetooth	model name		
	brand name		
	bands		
	SKU		
CPU/PCH	CPU Step		
	PCH Step		
	QDF		
	Production /Pre-Production		
Shipping	Including which Shipping Method used (Intel shipping, FedEx, DHL)		
	Send the Shipping memo number and tracking number if available		
	Is this shipment above $1000? if yes include commercial invoice (Israel only)		

§

Appendix E – Format For Issue Reporting

1. Issue Title on IPS system : [OEM_PROJECTNAME] ISSUE_SYMPOTOM
2. Project name :
3. Platform : ADL?
4. Sku : P?H?M?
5. OS version : 22H2?
6. ISH FW version : 5.4.2.4564?
7. ISH Driver version :
8. Issue symptom :
9. How to reproduce issue :
10. How to recover issue :
11. Failure rate :
12. Provide FW log, ISH.etl, system log, PDT image, screen shot of sensor stacks in DM
13. Provide Sensor Information if needed

Sensor Model	Location (Panel/Keyboard)	I2C_BUS	I2C Address	ISH_GP pin	GPIO mode (ex:active low, push-pull…etc)

§