Intel® Programmable Services Engine SDK Get Started Guide

Rev. 1.0

Doc. No.: 728880, Rev.: 1.0

Contents

1. Intel® Programmable Services Engine Get Started Guide
2. Development System Requirements
3. Set Up the Build Environment and Get the Intel® PSE Source Code
4. Build the Hello World Sample Application
5. Update the PSE Firmware in the Platform
6. Load the Hello World Firmware Image
7. Appendix A Terminology
8. Appendix B Revision History
9. Appendix C
10. Notices and Disclaimers

Intel® Programmable Services Engine Get Started Guide

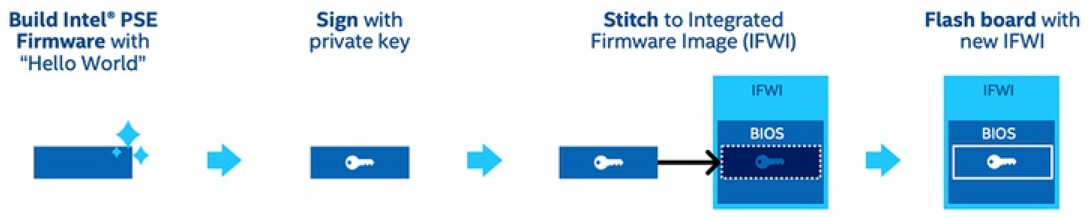
The Intel® Programmable Services Engine (Intel® PSE) is an independent System-on- a-Chip (SoC) subsystem that is integrated in the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors (previously known as Elkhart Lake, will be known as "the Processor" throughout the document) Platform Controller Hub (PCH). This Get Started Guide helps you to set up the Zephyr* Real-Time Operating System (RTOS) Software Development Kit (SDK) and other necessary tools to develop applications for the Intel® PSE.

In this document, you will go through the following steps:

1. Obtain an overview of the development system requirements and prerequisites.
2. Obtain source code and set up build environment for the Intel® PSE.
3. Build a Hello World sample application.
4. Sign and stitch the Hello World firmware image to the Integrated Firmware Image (IFWI).
5. Flash the new IFWI image with the Hello World project, to the platform. Refer to the following figure for a visual overview of step 3 to step 5.

The following figure shows the stages of the process.

Figure 1. Software Components and Steps to Build an IFWI with Hello World Project


Build Intel® PSE Firmware with "Hello World"
Sign with private key
Stitch to Integrated Firmware Image (IFWI)
Flash board with new IFWI
IFWI
BIOS
IFWI
BIOS

The Appendix A Terminology section lists the full names of the acronyms used in this document.

Development System Requirements

Prerequisites

Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors Platform Setup
Verify that the platform has been booted with the Windows* OS or Yocto Project* based Linux* OS to ensure that the board is functional, before proceeding to the next steps in this Get Started Guide.

Experience Required

- Basic knowledge of Linux* OS
- Knowledge of configuring the BIOS

Software
Development Machine

- Ubuntu* OS version 18.04 LTS or 20.04 LTS

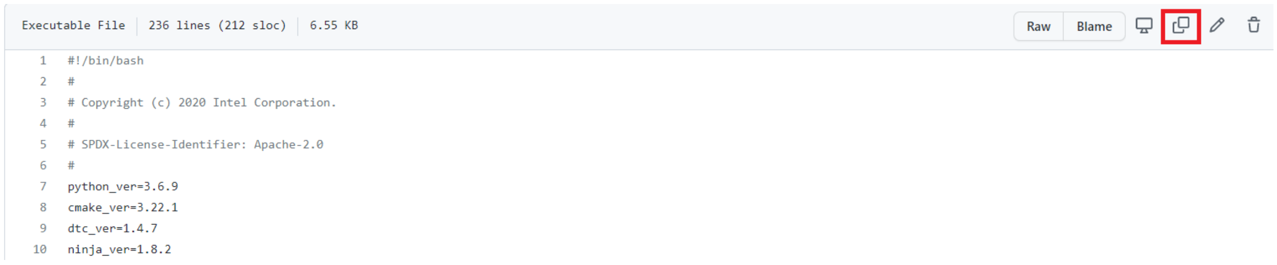
Hardware

Table 1. Hardware Requirements

Device/Accessories	Requirements
The **platform** consists of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors platform.	Accessories including a power adapter, power cord, board standoff, and screws.
The **Development Machine** consists of the Linux* OS PC, laptop, or notebook.	• Ubuntu* OS version 18.04 LTS or 20.04 LTS. • Display, keyboard, and mouse. • Three USB ports.
Serial log cables.	For Intel® PSE Logs and host side logs.
A DediProg* SF600/SF600 Plus SPI flash IC programmer (optional).	

Set Up the Build Environment and Get the Intel® PSE Source Code

1. For the purposes of this guide, create **intelpse** as your working directory at the user home (~) directory. Launch a shell terminal from your Development Machine and use the following commands:
   @DEV_MACHINE:
   ```
   cd ~
   mkdir intelpse
   cd intelpse
   ```
2. At browser, visit URL https://github.com/intel/pse-fw to get to PSE Firmware source. The script to install the enviroment variable is located at https://github.com/intel/pse-fw/tree/main/tools/sdk_installer/install_sdk_env_2.sh . Copy the raw contents to a file **install_env.sh** at your Development Machine directory, such as ~/intelpse.



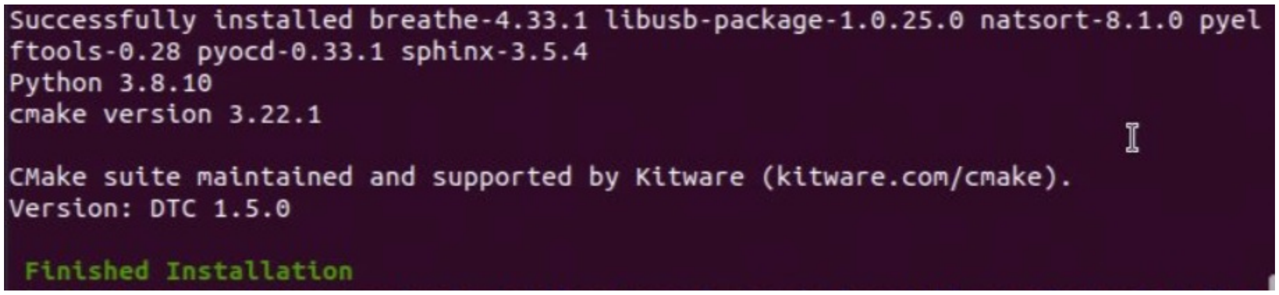
```
#!/bin/bash
#
# Copyright (c) 2020 Intel Corporation.
#
# SPDX-License-Identifier: Apache-2.0
#
python_ver=3.6.9
cmake_ver=3.22.1
dtc_ver=1.4.7
ninja_ver=1.8.2
```

3. **Run the script** to install the build environment:
   @DEV_MACHINE:
   ```
   chmod +x install_env.sh
   ./install_env.sh
   ```
4. You will be asked to enter the Zephyr* SDK installation directory. Press **Enter** when prompted to choose the default installation path **/opt/zephyr-sdk2**.

The installation from this script will take approximately 30 minutes.

5. When the script completes, the versions of the software installed are printed on the screen in this order: Python*, CMake, DTC, and Ninja, as shown in the following image:

Figure 3. Software Versions



```
Successfully installed breathe-4.33.1 libusb-package-1.0.25.0 natsort-8.1.0 pyel
ftools-0.28 pyocd-0.33.1 sphinx-3.5.4
Python 3.8.10
cmake version 3.22.1

CMake suite maintained and supported by Kitware (kitware.com/cmake).
Version: DTC 1.5.0

 Finished Installation
```

Note: If there are installation errors, the script will inform you the software to install manually.

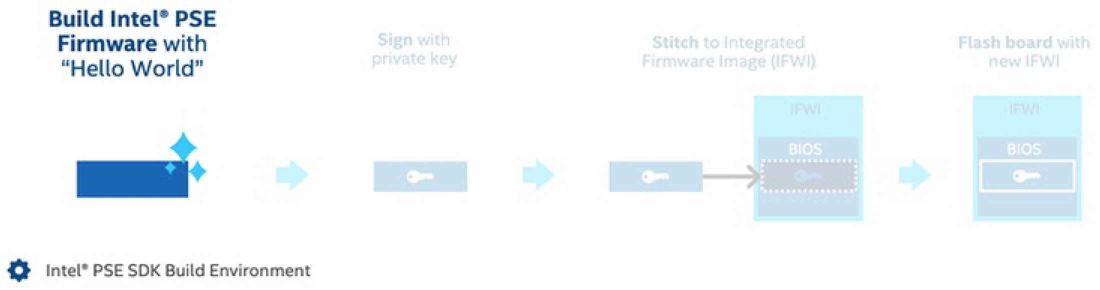
6. Download the latest Intel® Programmable Services Engine source code from the GitHub repository.
@DEV_MACHINE:

```
   cd ~/intelpse
west init -m http://github.com/intel/pse-fw --mr main
west update
```

Build the Hello World Sample Application

In this section, you will build a Hello World project to verify that the Development Machine environment is set up correctly. The sample Hello World source files are in the **zephyr/samples/hello_world** project folder. The following figure shows the current stage of the process.

Figure 4. Build the Intel® PSE Firmware with the Hello World Project


Build Intel® PSE Firmware with "Hello World"
Sign with private key
Stitch to Integrated Firmware Image (IFWI)
Flash board with new IFWI
IFWI
BIOS
IFWI
BIOS
Intel® PSE SDK Build Environment

1. Source the build environment. **Run the following command each time you start a terminal window to build the firmware image**:
@DEV_MACHINE:

```
cd ~/intelpse/ehl_pse-fw/
source ~/.zephyrrc
```

2. Remove any previous builds:
@DEV_MACHINE:

```
./build.sh clean
```

3. Build the Intel® PSE firmware sample application by including the application folder path in the command line, for example in this case we are building the Hello World sample application:
   @DEV_MACHINE:

```
./build.sh ../zephyr/samples/hello_world/
```

4. After the application is built successfully, the output firmware (**PSE_FW.bin**) required for stitching can be found at **ehl_pse-fw/tools/pse_image_tool/output**
   @DEV_MACHINE:

```
ls -la ~/intelpse/ehl_pse-fw/tools/pse_image_tool/output/PSE_FW.bin
```

Update the PSE Firmware in the Platform

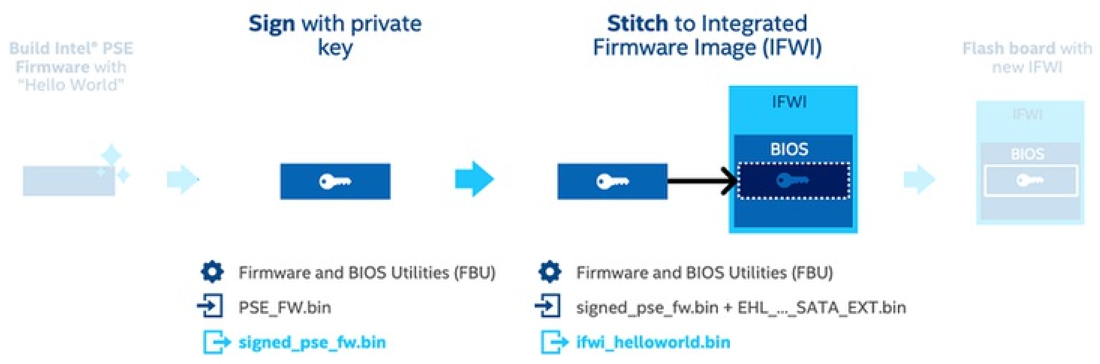
There are two methods to update the Intel® PSE Firmware Image.

1. **Sign and Stitch PSE Firmware with Integrated Firmware Image**.
2. **Update PSE Firmware via Signed Capsule**. For this step, please contact your platform provider.

Sign and Stitch Integrated Firmware Image

To load the newly built **Hello World** firmware image, you must sign and stitch it into the BIOS region of the Integrated Firmware Image (IFWI). **Firmware and BIOS Utilities (FBU)** is a tool that will be used to sign and stitch the image.

Figure 5. Sign and Stitch the Image


Build Intel® PSE Firmware with "Hello World"
Sign with private key
Stitch to Integrated Firmware Image (IFWI)
Flash board with new IFWI
IFWI
BIOS
IFWI
BIOS
Firmware and BIOS Utilities (FBU)
PSE_FW.bin
signed_pse_fw.bin
Firmware and BIOS Utilities (FBU)
signed_pse_fw.bin + EHL_..._SATA_EXT.bin
ifwi_helloworld.bin

This section covers the steps to download the necessary tools to sign and stitch.

Download Firmware and BIOS Utilities (FBU) Tool

1. Download the FBU tool from https://github.com/intel/iotg-fbu
2. Copy the downloaded FBU tool to your working directory. E.g., to the **~/intelpse** folder.
3. Go to the **iotg-fbu-master** folder and install the required modules for the FBU tool:
   @DEV_MACHINE:

```
cd iotg-fbu-master
sudo pip3 install -r -H requirements.txt
```

Download the Integrated Firmware Image

Download the latest Integrated Firmware Image (IFWI) from your platform provider and move to the iotg-fbu-master/siiptool/scripts folder.

Sign and Stitch the Intel® PSE Image into the IFWI

With all dependencies ready now, you can sign and stitch the image. The following figure shows the current stages of the process.

1. Go to the tool's **scripts** directory:
   @DEV_MACHINE:

   ```
   cd ~/intelpse/iotg-fbu-master/siiptool/scripts
   ```

2. Obtain the PSE Firmware Signing Key from your platform provider and copy to the **~/intelpse/iotg-fbu-master/siiptool/scripts** folder.
3. Copy the **PSE_FW.bin** built previously to the same folder:
   @DEV_MACHINE:

   ```
   cp ~/intelpse/ehl_pse-fw/tools/pse_image_tool/output/PSE_FW.bin .
   ```

4. **Sign** the Intel® PSE firmware image using the key:
   @DEV_MACHINE:

   ```
   python3 siip_sign.py sign -i PSE_FW.bin -k <key from your platform provider> -o signed_pse_fw.bin
   ```

   The following is a quick explanation about the flags used in the command:
   -i is the input file to be signed
   -k is the private key to apply for signing
   -o is the output filename after signing
5. **Stitch** the image as follows. You may refer to the following table.
   @DEV_MACHINE:

   ```
   python3 siip_stitch.py -ip pse -o ifwi_helloworld.bin <IFWI> signed_pse_fw.bin
   ```

   Table 1. Stitch Command Argument and Description

Argument	**Description**
ifwi_helloworld.bin	This is the stitched image that will be created after running this command. This filename will be used in subsequent sections of this document.
IFWI	This is the IFWI downloaded from your platform provider.
signed_pse_fw.bin	This is the signed version of the Intel® PSE firmware image created in step 4.

6. Confirm that **ifwi_helloworld.bin** was created:
   @DEV_MACHINE:

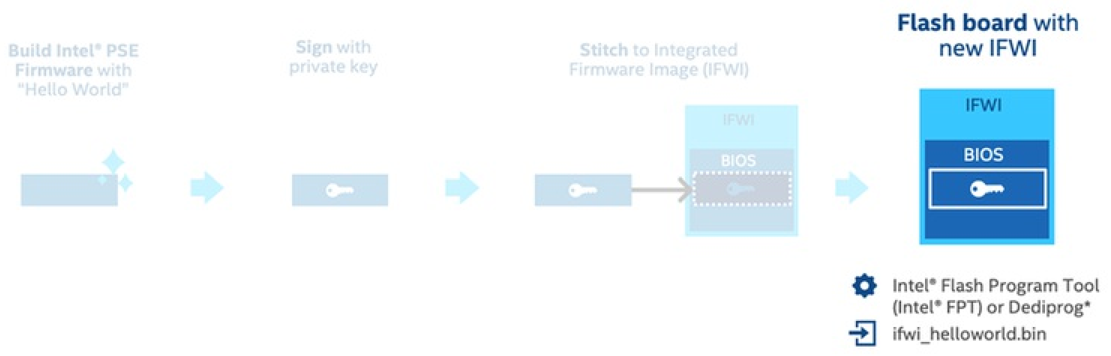
   ```
   ls -la ifwi_helloworld.bin
   ```

Load the Hello World Firmware Image

To load the firmware image, you must flash the Integrated Firmware Image (IFWI) to the platform.

The following figure shows the current stage of the process:

Figure 6. Flash the Board with the New IFWI


Build Intel® PSE Firmware with "Hello World"
Sign with private key
Stitch to Integrated Firmware Image (IFWI)
IFWI
BIOS
Flash board with new IFWI
IFWI
BIOS
Intel® Flash Program Tool (Intel® FPT) or Dediprog*
ifwi_helloworld.bin

Install DediProg* Software on the Development Machine

1. Install the following prerequisite tool:
   @DEV_MACHINE:
   ```
   sudo apt-get install libusb-dev build-essential
   ```
2. Download the DediProg* software for Linux* OS and compile the code:
   @DEV_MACHINE:
   ```
   cd ~
   git clone https://github.com/DediProgSW/SF100Linux
   cd SF100Linux
   make
   ```
   The dpcmd execution file will be created in the **SF100Linux** folder.
3. Root rights is required before each execution of the “dpcmd” command. Copy **60-dediprog.rules** from the **SF100Linux** folder to **/etc/udev/rules.s** (the extension depends on the OS, the file could sometimes be **rules.d**):
   @DEV_MACHINE:
   ```
   sudo mv 60-dediprog.rules /etc/udev/rules.d/.
   ```
4. Restart the service:
   @DEV_MACHINE:
   ```
   sudo udevadm control --reload
   ```
5. Open the **~/.bashrc file** and run the following command to add the “**SF100Linux**” directory with the dpcmd program:
   @DEV_MACHINE:
   ```
   nano ~/.bashrc
   export PATH=$PATH:~/SF100Linux/dpcmd
   ```
6. Source ~/.bashrc:
   @DEV_MACHINE:
   ```
   source ~/.bashrc
   ```

Flash the New IFWI onto the Platform using the DediProg* Software

The DediProg* SF600 Plus programmer can be used to program the SPI flash device:

1. Power off the platform.

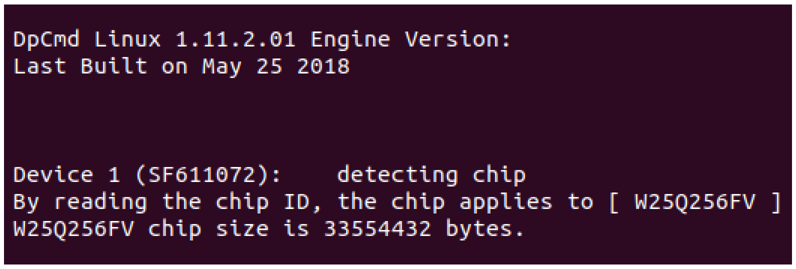
2. Connect the DediProg* programmer USB cable to the Development Machine that has the DediProg* software installed.
3. Connect the Flat Ribbon Cable (FRC) to the SPI DediProg* header on the platform.
4. Change the work directory to the DediProg* folder:
   @DEV_MACHINE:

```
cd ~/SF100Linux
```

5. Detect the SPI:
   @DEV_MACHINE:

```
dpcmd -d
```

6. The output message will appear as follows:

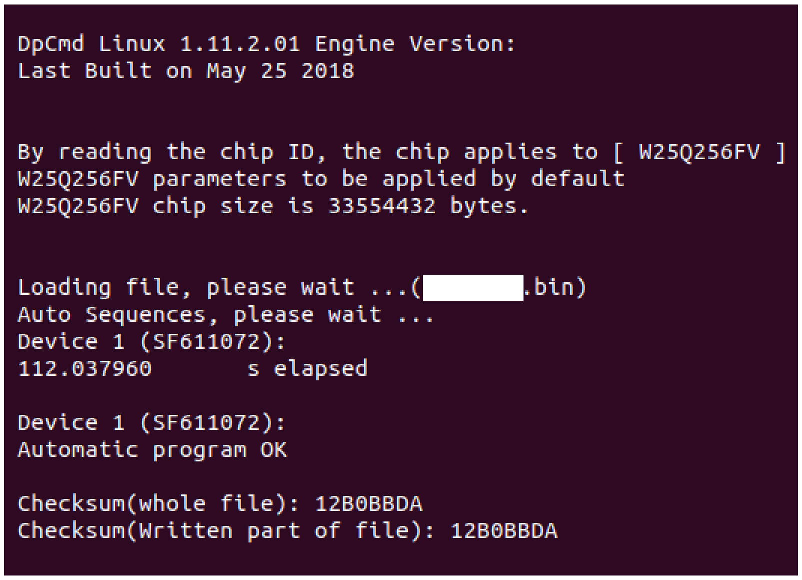


```
DpCmd Linux 1.11.2.01 Engine Version:
Last Built on May 25 2018



Device 1 (SF611072):    detecting chip
By reading the chip ID, the chip applies to [ W25Q256FV ]
W25Q256FV chip size is 33554432 bytes.
```

7. Flash the new IFWI generated in the Sign and Stitch the Intel® PSE Image into the IFWI section using the following command:
   @DEV_MACHINE:

```
./dpcmd --vcc 0 -z ~/intelpse/fbu/siiptool/scripts/ifwi_helloworld.bin
```

8. The output message will appear as follows:



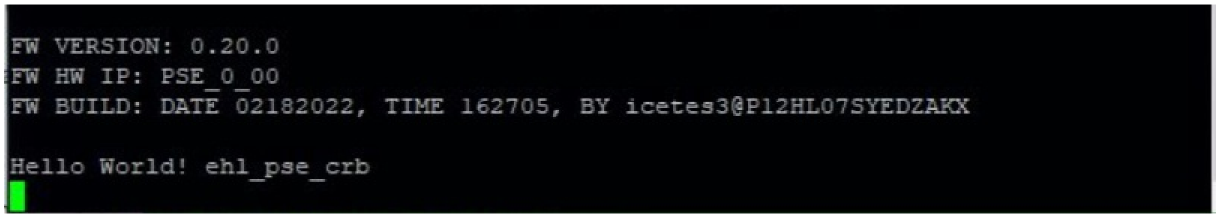
```
DpCmd Linux 1.11.2.01 Engine Version:
Last Built on May 25 2018


By reading the chip ID, the chip applies to [ W25Q256FV ]
W25Q256FV parameters to be applied by default
W25Q256FV chip size is 33554432 bytes.


Loading file, please wait ...([redacted].bin)
Auto Sequences, please wait ...
Device 1 (SF611072):
112.037960       s elapsed

Device 1 (SF611072):
Automatic program OK

Checksum(whole file): 12B0BBDA
Checksum(Written part of file): 12B0BBDA
```

9. After everything is completed, **unplug the FRC** from the platform.
10. Connect the serial cable between PSE UART Serial Connector and Development Machine. Make sure you have a serial utility installed in the Development Machine, for e.g., PuTTY. Please refer to Appendix C for serial connection details.

> Note:
> Users might have to do some BIOS settings to enable PSE Firmware Serial Logs. Refer to the platform specific guides for further details.

11. Power on the platform.

Test the Loaded Image
The Hello World sample application will print the following on the Intel® PSE terminal:



```
FW VERSION: 0.20.0
FW HW IP: PSE_0_00
FW BUILD: DATE 02182022, TIME 162705, BY icetes3@P12HL07SYEDZAKX

Hello World! ehl_pse_crb
```

If your output matches the preceding figure, **congratulations!** You have completed your first Intel® PSE application and flashed it to the platform.

Appendix A Terminology

Term	Description
BIOS	Basic Input/Output System
BKC	Best-Known Configuration
BSP	Board Support Package
dtc	Device-Tree-Compiler
FBU	Firmware and BIOS Utilities
FRC	Flat Ribbon Cable
IFWI	Integrated Firmware Image
Intel® FPT	Intel® Flash Program Tool
Intel® PSE	Intel® Programmable Services Engine
OEM	Original Equipment Manufacturer
OS	Operating System
PCH	Platform Controller Hub
RAM	Random Access Memory

Term	Description
RSA	Rivest–Shamir–Adleman
SDK	Software Development Kit
SIIP	System-on-a-Chip Independent Intellectual Property
SoC	System-on-a-Chip
SPI	Serial Peripheral Interface
UEFI	Unified Extensible Firmware Interface

Appendix B Revision History

Date	Revision	Description
April 2022	1.0	Initial release.

Appendix C

PuTTY GUI Configuration for the Host

Launch the PuTTY software from the desktop by right-clicking on the PuTTY icon and configure it to connect to the Host as follows:

As shown in the following figure,

Step 1: Under **Connection type**, select **Serial from radio button.**

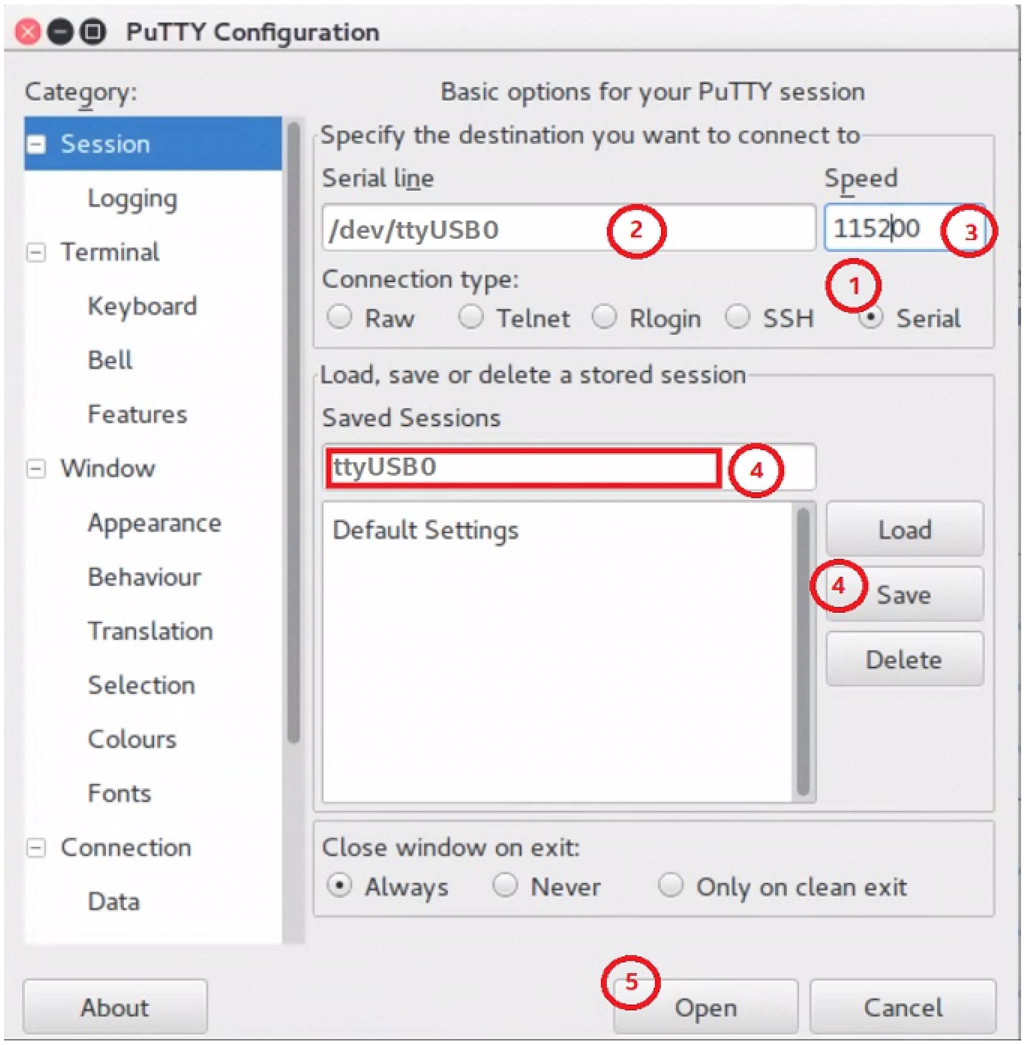
Step 2: Input the **serial line** identified with the "dmesg" command in the preceding figure. For example, **/dev/ttyUSB0.**

Step 3: Set the **Speed** field to **115200**.

Step 4: Name the serial configuration session as **ttyUSB0** and save the session to make it easy to launch in future.

Step 5: Click **Open**. There will be no output until the CRB is powered on in the next section.

Figure 24. PuTTY GUI Configuration for Host



Basic options for your PuTTY session

Specify the destination you want to connect to

Serial line: /dev/ttyUSB0

Speed: 115200

Connection type: Raw Telnet Rlogin SSH [✓ Serial]

Load, save or delete a stored session

Saved Sessions: ttyUSB0

Default Settings

Close window on exit: [✓ Always] Never Only on clean exit

PuTTY GUI Configuration for the Intel® PSE Terminal

Launch the PuTTY software from the desktop by right-clicking on the PuTTY icon and configure it to connect to the Intel® PSE terminal as follows:

As shown in the following figure,

Step 1: Under **Connection type**, select **Serial from radio button.**

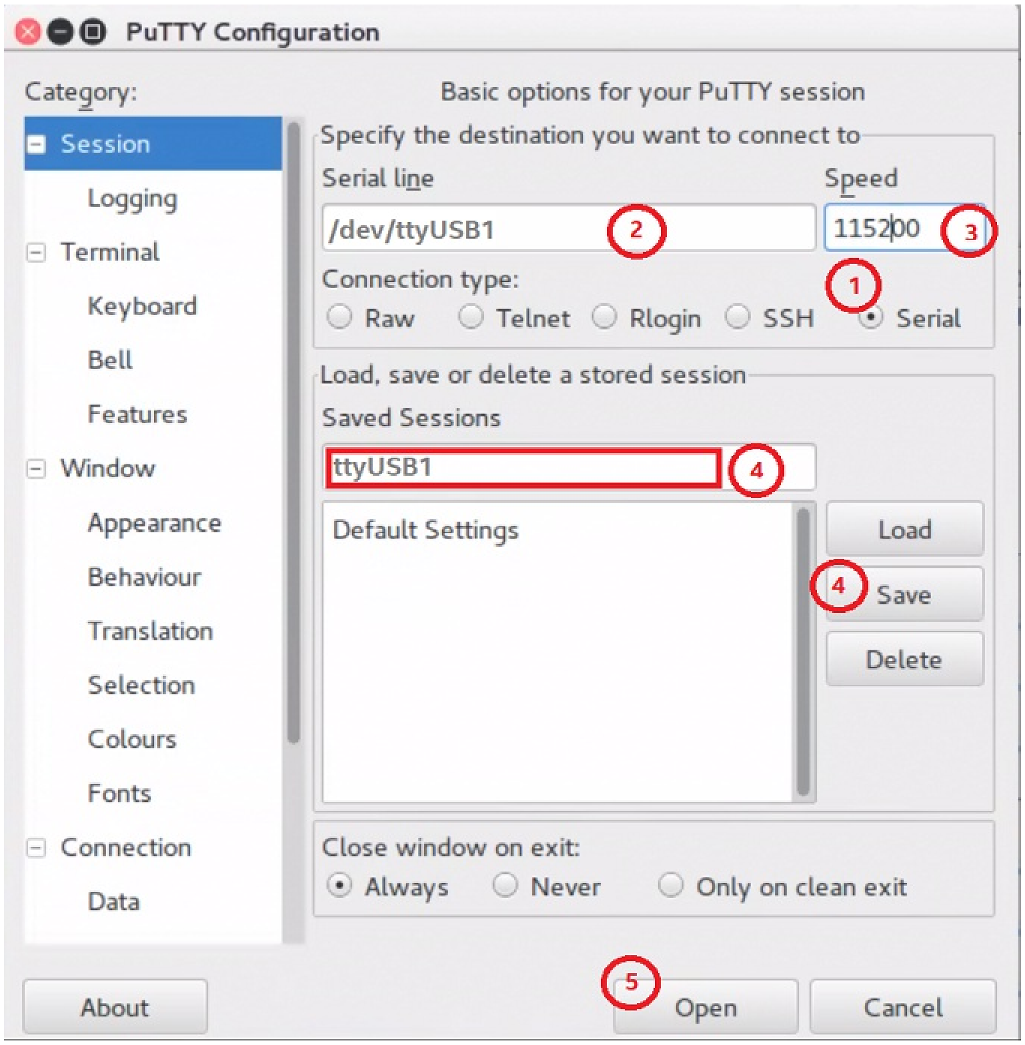
Step 2: Input the **serial line** identified with the "dmesg" command in the preceding figure. For example, **/dev/ttyUSB1.**

Step 3: Set the **Speed** field to **115200**.

Step 4: Name and save the serial configuration session to make it easy to launch in future.

Step 5: Click **Open**.

Figure 7. PuTTY GUI Configuration for Intel® PSE Terminal


PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Serial line: /dev/ttyUSB1

Speed: 115200

Connection type: Raw Telnet Rlogin SSH [✓ Serial]

Load, save or delete a stored session

Saved Sessions: ttyUSB1

Default Settings

Close window on exit: [✓ Always] Never Only on clean exit

Notices and Disclaimers