Yocto Project*-based Board Support Package for 11th Gen Intel® Core™ Processors (Code Name: Tiger Lake UP3) on IoT Platforms

Get Started Guide

August 2021

Maintenance Release 2 (MR2) - kernel 5.4
Maintenance Release 3 (MR3) - kernel 5.10

Document Number: 635120-3.0

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.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described 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.

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

Intel, processors, chipsets, and desktop boards 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 technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Performance varies depending on system configuration. No product or component can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Other names and brands may be claimed as the property of others.

Contents

August 2021
Document Number: 635120-3.0

Figures

Tables

Revision History

Date	Revision	Description
August 2021	3.0	Maintenance Release 3 (MR3) - kernel 5.10 release
June 2021	2.4	Updated the Intel® TCC Tools link in Section 4.5. Updated the *11th Gen Intel® Core™ Processors Real-Time Tuning Guide* document ID in Section 1.4.
May 2021	2.3	Added the *meta-intel-wireless* file info. Refer to Section 3.1.2.2 for the integration steps.
May 2021	2.2	Maintenance Release 2 (MR2)
February 2021	2.1	Maintenance Release 1 (MR1) Added Step 8 in Section 3.2

§

1.0 Introduction

This document provides instructions on how to build the Yocto Project*-based board support package (BSP) for the 11th Gen Intel® Core™ Processors (Code Name: Tiger Lake - UP3) RVP. This requires preparing and setting up a host system, steps in selecting the components and building a Yocto Project*-based image, as well as preparing the bootable image with the USB flash drive to install to the RVP.

You are recommended to review the release information before proceeding with this *Get Started Guide*. For release information, notes, and reference, refer to the following documents:

Yocto Project-based Board Support Package for 11th Gen Intel® Core™ Processors (Code Name: Tiger Lake - UP3) Release Notes – Kernel 5.4* (Document number: 615079)

Yocto Project-based Board Support Package for 11th Gen Intel® Core™ Processors (Code Name: Tiger Lake - UP3) Release Notes – Kernel 5.10* (Document number: 646531)

Note: The Yocto Project*-based build system version and the corresponding open-source software components suggested for use with the BSP are only for reference purposes. If you decide to use Yocto Project*, it is your responsibility to integrate the latest functional and/or security updates when they are available from the open-source community.

1.1 Terminology

Table 1. Terminology

Term	Description
BSP	Board Support Package
GCC	GNU Compiler Collection
GPIO	General-Purpose Input/Output
ID	Identity
Intel® AMT	Intel® Active Management Technology
Intel® TCC	Intel® Time Coordinated Computing
NVMe*	NVM Express*
RVP	Reference Validation Platform

Term	Description
SDK	Software Development Kit
TSN	Time-Sensitive Networking
USB*	Universal Serial Bus

1.2 Intended Audience

This guide is for users of the Yocto Project*-based BSP for the 11th Gen Intel® Core™ processors.

1.3 Customer Support

Contact your Intel representative for support or submit an issue to http://premiersupport.intel.com.

1.4 Reference Documents

Log in to the Resource and Design Center (rdc.intel.com) to search for and download the document numbers listed in the following table. Contact your Intel field representative for access.

Note: Third-party links are provided as a reference only. Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Table 2. Reference Documents

Document	Document No./Location
11th Gen Intel® Core™ Processors Real-Time Tuning Guide	640980
Tiger Lake – UP3 for IoT Platforms – Message of the Week	614057
How-to Video: Setting Up Build Environment in the Host System for Yocto Project*	608732
Intel Atom® and Intel® Core™ Processor Build Environment Setup for Yocto Project*	334828
Ethernet Time-Sensitive Networking for Elkhart Lake/Tiger Lake UP3 - Getting Started Guide	616446
Wireless Connectivity Product for Yocto Project* User Guide	617199

Document	Document No./Location
Tiger Lake UP3 Platform Reference Validation Platform (RVP) User Guide	612924
Generate Key for Secure Boot with the Yocto Project*-based Image and Bootloader/UEFI BIOS	633630
Yocto Project*-based Board Support Package for 11th Gen Intel® Core™ Processors (Formerly Known as Tiger Lake UP3) on IoT Platforms Release Notes	615079
Intel® In-Band Manageability Framework x86 Release Notes	630741
Yocto Project*-based Board Support Package for 11th Gen Intel® Core™ Processors (Code Name: Tiger Lake - UP3) Release Notes - Kernel 5.10	646531

§

2.0 Prepare and Set up the Host System

This chapter describes the steps to prepare and set up the host system to build the Yocto Project*-based image that will be used to boot up the 11th Gen Intel® Core™ Processor board. The preparation and setup are only required for the first time.

2.1 Setting up the Host System for the First Time

2.1.1 Prerequisite for the Host System

The following are the minimum host system configurations to build the BSP for Yocto Project*. For the 11th Gen Intel® Core™ Processors, this configuration supersedes the other prerequisites that may be listed in the Host System Build Environment Setup Guide (Document number: 334828).

- Intel® Core™ i7 processor (4 cores)
- Linux* OS of choice for the Yocto Project* build is Ubuntu* 16.04 LTS OS or higher. Refer to Yocto Project Quick Start for more information.
- Minimum of 32 GB Random Access Memory (RAM) and 500GB disk space are recommended.
- High-speed network connectivity.
- USB flash drive 64GB minimum (to prepare the bootable Yocto Project*-based image).

2.1.2 Host System Preparation and Installation

Follow the *Host System Build Environment Setup Guide* (Document number: 334828) *and How-to Video: Host System Build Environment Setup* (Video number: 608732) to prepare the host system to build the Yocto Project-based image.

After that, proceed to the next section to install additional dependencies on the host system.

2.2 Install Additional Dependencies

Install the following dependencies for this project.

2.2.1 Install the Required Toolchain

1. Install the necessary toolchain if it is not available in your host system:
```
$ sudo apt-get -y install socat gawk wget git-core diffstat
unzip texinfo build-essential chrpath libncurses5-dev
patchutils curl
```

2. Set up the git-lfs in the host machine:

   Install git-lfs on the host/dev/build machine.

```
$ curl -s
https://packagecloud.io/install/repositories/github/git-
lfs/script.deb.sh | sudo -E bash
$ sudo apt-get install git-lfs
```

2.2.2 Install the Required GCC Version

1. Check for the current GCC version using the following codes:
```
$ gcc --version

$ g++ --version

# If your GCC version is below 8.2, proceed to the next step:
```

2. Install the required GCC using the following codes:
```
# Add PPA repository
$ sudo add-apt-repository ppa:ubuntu-toolchain-r/test

# If unable to add the PPA
$ sudo -E apt-get install --reinstall ca-certificates
$ sudo -E add-apt-repository ppa:ppaname/ppa

# Update and install GCC to take effect
$ sudo -E apt-get update
$ sudo -E apt-get install gcc-8 g++-8

# Alternatively, update and configure GCC
$ sudo update-alternatives --install /usr/bin/gcc gcc
/usr/bin/gcc-8 60 --slave /usr/bin/g++ g++ /usr/bin/g++-8

# Select which GCC to use. Select with GCC8 or the latest GCC
you have installed as prompted
$ sudo update-alternatives --config gcc
```

§

3.0 Getting Started with the Board Support Package

This section consists of two major sub-sections: Getting Started with Kernel 5.4 BSP Release (MR2) and Getting Started with Kernel 5.10 BSP Release (MR3) respectively. MR3 is the latest release. It is recommended to migrate to the latest Board Support Package (BSP) for more fixes and features.

Note: Regularly check on available updates for the Ubuntu* build system that you set up from Section 2.0. This is to make sure that all toolchains are up to date.

3.1 Getting Started with Kernel 5.4 BSP Release (MR2)

Refer to the *Yocto Project*-based Board Support Package for 11th Gen Intel® Core™ Processors (Code Name: Tiger Lake - UP3) Release Notes - Kernel 5.4* (Document number: 615079) for further information on MR2 BSP Release.

3.1.1 Obtain the Repository Source and Prepare to Build Image

This section describes the essential steps to obtain the BSP source and getting the system ready to build Yocto Project*-based image. This section is compulsory, and any missing step might cause the build to fail.

1. Create a bin/ directory in the home directory and include the path:
```
$ mkdir ~/bin
$ PATH=~/bin:$PATH
```

2. Get the repo source and make it executable:
```
$ curl https://storage.googleapis.com/git-repo-downloads/repo
>~/bin/repo
$ chmod a+x ~/bin/repo
```

3. Make a new directory:
```
$ mkdir <work_dir>
$ cd <work_dir>
```

4. Git clone the repo manifest. This manifest will help you to clone all required repositories to create the base BSP:
```
repo init -u https://github.com/intel/iotg-yocto-ese-manifest
-b refs/tags/release-68_tgl_u_mr2 -g all
```

5. Pull the repository meta-layers:
```
$ repo sync -c -j8 --force-sync
```

6. Make a branch:
```
$ repo forall managed/* -c git branch -f BUILD HEAD
```

7. The Yocto Project*-based image supports both secure and non-secure booting options. In either case, security keys must be generated and integrated into the image; otherwise, the build may fail. Choose one of the following options to generate the keys:

   Option 1 (Default):
   Security keys will be generated randomly in this option and it is the default configuration. If randomly generated security keys are not affecting your application, no action is required in this step. You can proceed to the next step.

   Option 2:
   You need to generate your own security keys and place them in the 'cert' folder in the *<work_dir>/build* folder of the build. For steps on how to use a common command-line tool (OpenSSL*) to generate the keys required for the secure boot, including how to create the Keys and Certificates, place the created keys in the correct folder. Then install the keys into the UEFI BIOS. See *Generate Key for Secure Boot with the Yocto Project*-based Image and Slim Bootloader/UEFI BIOS*, (Document number: 633630).

Note: To use Slim Bootloader or secure boot with UEFI BIOS, you must choose Option 2. Make sure the same keys are used for the Yocto Project*-based image and Slim Bootloader/UEFI BIOS.

3.1.2 Adding Optional Components

This section describes the steps required to integrate Software Tools/Packages, Proprietary Components, and workarounds to the build. For more information about software tools or packages, refer to Section 5.1. For more information about proprietary components and workarounds, see *Yocto Project*-based Board Support Package for 11th Gen Intel® Core™ Processors (Formerly Known as Tiger Lake-UP3) on IoT Platforms Release Notes* (Document number: 615079).

Note: **Since the components are optional, the build will not fail if you choose to skip this section. You can still add the components in the future and re-build the image.**

3.1.2.1 Integrate Software Tools or Packages

1. This step integrates the bmaptool into the Yocto Project*-based image. This tool is required only if you need to install the image into another storage on the 11th Gen Intel® Core™ processor RVP (Refer to Section 4.4 for the steps to install from one storage to another). Otherwise, you may skip this step.

```
$ cd <work_dir>/build/conf
$ vi local.conf
```

```
#append bmap-tools

IMAGE_INSTALL_append = " bmap-tools"
```

2. **[Required for TSN]** This step integrates the TSN Reference Software into the image. TSN Reference Software is needed if you want to explore the TSN technology in your application. Otherwise, you may skip this step. For more information about TSN, refer to Section 3.6 of the *Yocto Project*-based BSP Release Notes* (Document number: 615079).

```
$ vi <work_dir>/build/conf/multiconfig/x86-common.inc

# Add the following lines
IMAGE_INSTALL_append = " iotg-tsn-ref-sw"
```

3.1.2.2 Integrate Proprietary Components

This section provides the steps to integrate proprietary components into the Yocto Project*-based image. This section is optional, and you may skip this section if none of the proprietary components is applicable to you. You may choose to integrate one or more components. For more information on the proprietary components, see *Yocto Project*-based Board Support Package for 11th Gen Intel® Core™ Processors (Formerly Known as Tiger Lake UP3) Release Notes – Kernel 5.4* (Document number: 615079).

Two files from Section 2.2 of the *11th Gen Intel® Core™ Processors (Code Name: Tiger Lake - UP3) Release Notes - Kernel 5.4* are relevant in this section:

a. yocto_project_mr2_release_v2.zip (Document number: 617124)

   The zip file consists of recipes and binaries for wireless (meta-intel-wireless.tar.gz) and cellular (meta-intel-cellular.tar.gz).

b. meta-ptcm.tar.gz (Document number: 646618)

   The tarball contains the Real-Time Configuration Manager that required for **Intel® TCC Tools**.

Follow the instructions below if you choose to integrate the proprietary components.

1. Create a new folder (for example: *Proprietary*) in the working directory and unzip the **yocto_project_mr2_release_v2.zip and meta-ptcm.tar.gz** into the newly created *Proprietary* folder.

```
$mkdir <work_dir>/proprietary
$unzip yocto_project_mr2_release_v2.zip
$ tar -xvf meta-ptcm.tar.gz
```

2. Multiple tarballs will be available in the *Proprietary* folder after unzipping the file. Untar the tarballs that you would like to integrate in the next steps. Replace<file_name> with the tarball file name.

```
$ tar -xvf <file_name>.tar.gz
```

3. Change the directory to the conf folder where *local.conf* and *bblayers.conf* are located.
```
$ cd <work_dir>/build/conf
```

4. To integrate the Wi-Fi/Bluetooth® component into the image:
```
$ vi bblayers.conf
# Add below lines
{TOPDIR}/../proprietary/meta-intel-wireless \
```

5. To integrate the cellular component into the image:
```
$ vi bblayers.conf
# Add below lines
{TOPDIR}/../proprietary/meta-intel-cellular \

$ vi local.conf
# Add the following lines
# Enable Cellular
# Meta intel cellular packagegroup

IMAGE_INSTALL_append
="${@bb.utils.contains('LAYERSERIES_CORENAMES', 'dunfell',
"packagegroup-meta-intel-cellular", "", d)}"
KERNEL_PROVIDERS_EXTRA_MODULES[5.4] += " iosm-driver"
KERNEL_PROVIDERS_EXTRA_MODULES[5.4] += " mdm-ctrl"
```

6. **[Required for Intel® TCC Tools]** Integrate the real-time configuration manager (RTCM) by updating the following files:
```
$ vi bblayers.conf
# Add the following lines
${TOPDIR}/../proprietary/meta-ptcm \

$ vi local.conf

# Add the following lines
IMAGE_INSTALL_append = " ptcm"
PTCM_INSTALL = "1"
```

7. **[Required for Intel® TCC Tools]** Add the itt-static library:
```
$ vi local.conf

# Add the following line
IMAGE_INSTALL_append =
"${@bb.utils.contains("IMAGE_FEATURES","dev-pkgs", " itt-
staticdev", "", d)}"
```

8. **[Required for Intel® TCC Tools]** TGPIO Feature: Integrate the patch by following steps:

a. Download the TGPIO patch from https://github.com/intel/linux-intel-quilt/blob/5.4/yocto/patches/0001-ptp-S-W-workaround-for-PMC-TGPIO-h-w-bug.210615T223128Z into the build machine.

b. Create a directory **tgpio-5.4** under **meta-intel-ese-bsp/recipes-kernel/linux/files**.

```
$ mkdir <work_dir>/intel-embedded-system-enabling/meta-
intel-embedded-system-enabling/meta-intel-ese-bsp/recipes-
kernel/linux/files/tgpio-5.4
```

c. Copy the patch file *0001-ptp-S-W-workaround-for-PMC-TGPIO-h-w-bug.patch* to **tgpio-5.4** directory.

```
$ cp 0001-ptp-S-W-workaround-for-PMC-TGPIO-h-w-bug.patch
<work_dir>/intel-embedded-system-enabling/meta-intel-
embedded-system-enabling/meta-intel-ese-bsp/recipes-
kernel/linux/files/tgpio-5.4
```

d. Create **tgpio-5.4.scc** file at **meta-intel-ese-bsp/recipes-kernel/linux/files/** and add the line as follows:

```
$ vi <work_dir>/intel-embedded-system-enabling/meta-intel-
embedded-system-enabling/meta-intel-ese-bsp/recipes-
kernel/linux/files/tgpio-5.4.scc

#add this line
patch tgpio-5.4/0001-ptp-S-W-workaround-for-PMC-TGPIO-h-w-
bug.patch

#Save and exit
```

e. Edit *linux-intel-ese-lts-rt-5.4_git.bb* for RT kernel and add below lines at the end of the file:

```
$ vi <work_dir>/intel-embedded-system-enabling/meta-intel-
embedded-system-enabling/meta-intel-ese-bsp/recipes-
kernel/linux/linux-intel-ese-lts-rt-5.4_git.bb

#add the lines

# TGPIO WA
SRC_URI_append = " file://tgpio-5.4.scc"

#Save and exit
```

f. Edit linux-intel-ese-lts-5.4_git.bb for LTS kernel and add below lines at the end of the file:

```
$ vi <work_dir>/intel-embedded-system-enabling/meta-intel-
embedded-system-enabling/meta-intel-ese-bsp/recipes-
kernel/linux/linux-intel-ese-lts-5.4_git.bb

#add the lines

# TGPIO WA
SRC_URI_append = " file://tgpio-5.4.scc"
Save and exit

#save and exit
```

3.1.2.3 Integrate Optional Workarounds or Patches

If you have your own kernel patches or patches from Section 2.2 of the *Yocto Project*-based BSP Release Notes* (Document number: 615079) that are relevant to your application, apply it in this step.

3.1.3 Build the Yocto Project*-based Image

This section explains the steps required to start the build process. There are two options to build the image:

Option A: Image boots with LTS kernel by default. This is a validated image with LTS Kernel as the default boot kernel, GUI, secure boot, partition layout (as required for manageability support) and other third-party software/ tools.

Option B: Image boots with real-time kernel by default. This is a validated image with RT kernel as the default boot kernel.

Note: **It is mandatory to choose Option B for Intel® TCC Tools.**

Follow the steps below to build the image:

Option A: Full image with LTS Kernel as Default Boot Image

1. Start the build process:
   ```
   $ cd <work_dir>/build
   $ . ../intel-embedded-system-enabling/oe-init-build-env .
   ```

   Note: The dot at the end of the above command is part of the command.

2. Run the BitBake compilation command:

   To build the .wic image with GUI, secure boot, partition layout (as required for manageability support) and other third party software/ tools:

   ```
   bitbake mc:x86:core-image-sato-sdk
   ```

Location of Images: <work_dir>/build/tmp-x86-glibc/deploy/images/intel-corei7-64/

Required Files: core-image-sato-sdk-intel-corei7-64--<date>.wic and core-image-sato-sdk-intel-corei7-64--<date>.wic.bmap

Note: The bitbake mc:x86: core-image-sato-sdk command will generate an image that boots up with the LTS Kernel by default and the real-time (RT) Kernel as the second option to be selected on the boot menu (refer to Section 4.3 for more details).

Option B: Full Image with Real-time (RT) Kernel as Default Boot Image - Mandatory for Intel® TCC Tools

1. Start the build process:
```
$ cd <work_dir>/build
$ . ../intel-embedded-system-enabling/oe-init-build-env .
```

Note: The dot at the end of the above command is part of the command.

2. Run the BitBake compilation command:
```
bitbake mc:x86-rt:core-image-sato-sdk
```

Location of Images: <work_dir>/build/tmp-x86-rt-glibc/deploy/images/intel-corei7-64/

Required Files: core-image-sato-sdk-intel-corei7-64--<date>.wic and core-image-sato-sdk-intel-corei7-64--<date>.wic.bmap

Note: The bitbake mc:x86-rt:core-image-sato-sdk command will generate an image that boots up with the real-time (RT) Kernel by default and the LTS Kernel as the second option.

3.1.4 Subsequent Build of Image

For subsequent rebuilding of the image, cleaning the kernel is recommended before executing BitBake compilation command.

For Option A:
bitbake -c cleansstate mc:x86:linux-intel-ese-lts-5.4 (To clean sstate of LTS kernel)
bitbake -c cleansstate mc:x86:linux-intel-ese-lts-rt-5.4 (To clean sstate of RT kernel)

For Option B:
bitbake -c cleansstate mc:x86-rt:linux-intel-ese-lts-5.4 (To clean sstate of LTS Kernel)
bitbake -c cleansstate mc:x86-rt:linux-intel-ese-lts-rt-5.4 (To clean sstate of RT kernel)

3.2 Getting Started with Kernel 5.10 BSP release (MR3)

Refer to the *Yocto Project*-based Board Support Package for 11th Gen Intel® Core™ Processors (Code Name: Tiger Lake - UP3) Release Notes - Kernel 5.10* (Document number: 646531) for further information on MR3 BSP Release.

3.2.1 Obtain the Repository Source and Prepare to Build Image

This section describes the essential steps to obtain the BSP source and getting the system ready to build Yocto Project*-based image. This section is compulsory, and any missing step might cause the build to fail.

1. Create a bin/ directory in the home directory and include the path:
```
$ mkdir ~/bin
$ PATH=~/bin:$PATH
```

2. Get the repo source and make it executable:
```
$ curl https://storage.googleapis.com/git-repo-downloads/repo
>~/bin/repo
$ chmod a+x ~/bin/repo
```

3. Make a new directory:
```
$ mkdir <work_dir>
$ cd <work_dir>
```

4. Git clone the repo manifest. This manifest will help you to clone all required repositories to create the base BSP:
```
repo init -u https://github.com/intel/iotg-yocto-ese-manifest
-b refs/tags/release-74_tgl_u_mr3 -g all
```

5. Pull the repository meta-layers:
```
$ repo sync -c -j8 --force-sync
```

6. Make a branch:
```
$ repo forall managed/* -c git branch -f BUILD HEAD
```

   The Yocto Project*-based image supports both secure and non-secure booting options. In either case, security keys must be generated and integrated into the image; otherwise, the build may fail. Choose one of the following options to generate the keys:

   Option 1 (Default):
   Security keys will be generated randomly in this option and it is the default configuration. If randomly generated security keys are not affecting your application, no action is required in this step. You can proceed to the next step.

Option 2:
You need to generate your own security keys and place them in the 'cert' folder in the *<work_dir>/build* folder of the build. For steps on how to use a common command-line tool (OpenSSL*) to generate the keys required for the secure boot, including how to create the Keys and Certificates, place the created keys in the correct folder. Then install the keys into the UEFI BIOS. See *Generate Key for Secure Boot with the Yocto Project*-based Image and Slim Bootloader/UEFI BIOS* (Document number: 633630).

Note: To use Slim Bootloader or secure boot with UEFI BIOS, you must choose **Option 2**. Make sure the same keys are used for the Yocto Project*-based image and Slim Bootloader/UEFI BIOS.

3.2.2 Adding Optional Components

This section describes the steps required to integrate Software Tools/Packages, Proprietary Components, and workarounds to the build. For more information about software tools or packages, refer to Section 5.1. For more information about proprietary components and workarounds, see *11th Gen Intel® Core™ Processors (Code Name: Tiger Lake - UP3) Release Notes - Kernel 5.10* (Document number: 646531).

Note: **Since the components are optional, the build will not fail if you choose to skip this section. You can still add the components in the future and re-build the image.**

3.2.2.1 Integrate Software Tools or Packages

1. This step integrates the bmaptool into the Yocto Project*-based image. This tool is required only if you need to install the image into another storage on the 11th Gen Intel® Core™ processor RVP (Refer to Section 4.4 for the steps to install from one storage to another). Otherwise, you may skip this step.

```
$ cd <work_dir>/build/conf
$ vi local.conf

#append bmap-tools

IMAGE_INSTALL_append = " bmap-tools"
```

2. **[Required for TSN]** This step integrates the TSN Reference Software into the image. TSN Reference Software is needed if you want to explore the TSN technology in your application. Otherwise, you may skip this step. For more information about TSN, refer to Section 3.5 of the *11th Gen Intel® Core™ Processors (Code Name: Tiger Lake - UP3) Release Notes - Kernel 5.10* (Document number: 646531).

```
$ vi <work_dir>/intel-embedded-system-enabling/meta-intel-
embedded-system-enabling/meta-intel-ese/conf/include/x86-
common.inc

# Add the following lines
```

```
IMAGE_INSTALL_append = " iotg-tsn-ref-sw"
```

3.2.2.2 Integrate Proprietary Components

This section provides the steps to integrate proprietary components into the Yocto Project*-based image. This section is optional, and you may skip this section if none of the proprietary components is applicable to you. You may choose to integrate one or more components, refer to the *Yocto Project*-based Board Support Package for Yocto Project*-based Board Support Package for 11th Gen Intel® Core™ Processors (Formerly Known as Tiger Lake UP3) Release Notes – Kernel 5.10* (Document number: 646531).

Two files from Section 2.2 of the *11th Gen Intel® Core™ Processors (Code Name: Tiger Lake - UP3) Release Notes - Kernel 5.10* are relevant in this section:

a. yocto_project_mr3_release.zip (Document number: 646693)

   The zip file consists of recipes and binaries for wireless (meta-intel-wireless.tar.gz) and cellular (meta-intel-cellular.tar.gz).

b. meta-ptcm.tar.gz (Document number: 646638)

   The tarball contains the Real-Time Configuration Manager that required for **Intel® TCC Tools**.

Follow the instructions below if you choose to integrate the proprietary components.

1. Create a new folder (for example: *Proprietary*) in the working directory and unzip the **yocto_project_mr3_release.zip** and **meta-ptcm.tar.gz** into the newly created *Proprietary* folder.

   ```
   $mkdir <work_dir>/proprietary
   $unzip yocto_project_mr3_release.zip
   $ tar -xvf meta-ptcm.tar.gz
   ```

   As a result of unzipping, multiple tarballs will be available in the proprietary folder. Untar the tarballs that you would like to integrate in the next steps. Replace <file_name> with the tarball file name.

   ```
   $ tar -xvf <file_name>.tar.gz
   OR
   $ tar -xvf <file_name>.tar.bz2
   ```

2. Change the directory to the conf folder where local.conf and bblayers.conf are located.

   ```
   $ cd <work_dir>/build/conf
   ```

3. To integrate the Wi-Fi/Bluetooth® component into the image:

   ```
   $ vi bblayers.conf
   # Add below lines
   {TOPDIR}/../proprietary/meta-intel-wireless \
   ```

4. To integrate the cellular component into the image:

```
$ vi bblayers.conf
# Add below lines
{TOPDIR}/../proprietary/meta-intel-cellular \

$ vi <work_dir>/intel-embedded-system-enabling/meta-intel-
embedded-system-enabling/meta-intel-ese/conf/include/x86-
common.inc

# Add the following lines
# Enable Cellular
# Meta intel cellular packagegroup

IMAGE_INSTALL_append
="${@bb.utils.contains('LAYERSERIES_CORENAMES', 'dunfell',
"packagegroup-meta-intel-cellular", "", d)}
"KERNEL_PROVIDERS_EXTRA_MODULES[5.10] += " iosm-driver
"KERNEL_PROVIDERS_EXTRA_MODULES[5.10] += " mdm-ctrl"
```

5. **[Required for Intel® TCC Tools]** Integrate the real-time configuration manager (RTCM) by updating the following files:

```
$ vi bblayers.conf
# Add the following lines
${TOPDIR}/../proprietary/meta-ptcm \

$ vi <work_dir>/intel-embedded-system-enabling/meta-intel-
embedded-system-enabling/meta-intel-ese/conf/include/x86-
common.inc

# Add the following lines
IMAGE_INSTALL_append = " ptcm"
PTCM_INSTALL = "1"
```

3.2.2.3 Integrate Optional Workarounds or Patches

If you have your own kernel patches or patches from Section 2.2 of the *Yocto Project*-based BSP Release Notes* (Document number: 646531) that are relevant to your application, apply it in this step.

3.2.3 Build the Yocto Project*-based Image

This section explains the steps required to start the build process. There are two options to build the image:

Option A:
Image boots with LTS kernel by default. This is a validated image with LTS Kernel as the default boot kernel, GUI, secure boot, partition layout (as required for manageability support) and other third-party software/ tools.

Option B:
Image boots with real-time kernel by default. This is a validated image with RT kernel as the default boot kernel.

Note: **It is mandatory to choose Option B for Intel® TCC Tools.**

Follow the steps below to build the image:

Option A: Full image with LTS Kernel as Default Boot Image

1. Start the build process:
```
$ cd <work_dir>/build
$ . ../intel-embedded-system-enabling/oe-init-build-env .
```

Note: The dot at the end of the above command is part of the command.

2. Run the BitBake compilation command:

To build the .wic image with GUI, secure boot, partition layout (as required for manageability support) and other third-party software/ tools:

```
bitbake mc:x86-2020:core-image-sato-sdk
```

Location of Images: <work_dir>/build/tmp-x86-2020-glibc/deploy/images/intel-corei7-64/

Required Files: core-image-sato-sdk-intel-corei7-64--<date>.wic and core-image-sato-sdk-intel-corei7-64--<date>.wic.bmap

Note: The bitbake mc:x86-2020: core-image-sato-sdk command will generate an image that boots up with the LTS Kernel by default and the real-time (RT) Kernel as the second option to be selected on the boot menu (refer to Section 4.3 for more details).

Option B: Full Image with Real-time (RT) Kernel as Default Boot Image - Mandatory for Intel® TCC Tools

1. Start the build process:
```
$ cd <work_dir>/build
$ . ../intel-embedded-system-enabling/oe-init-build-env .
```

Note: The dot at the end of the above command is part of the command.

2. Run the BitBake compilation command:

```
bitbake mc:x86-rt-2020:core-image-sato-sdk
```

Location of Images: <work_dir>/build/tmp-x86-rt-2020-glibc/deploy/images/intel-corei7- 64/

Required Files: core-image-sato-sdk-intel-corei7-64--<date>.wic and core-image-sato-sdk-intel-corei7-64--<date>.wic.bmap

Note: The bitbake mc:x86-rt-2020:core-image-sato-sdk command will generate an image that boots up with the real-time (RT) Kernel by default and the LTS Kernel as the second option.

3.2.4 Subsequent Build of Image

For subsequent rebuilding of the image, cleaning the kernel is recommended before executing BitBake compilation command.

For Option A:
bitbake -c cleansstate mc:x86-2020:linux-intel-ese-lts-5.10 (To clean sstate of LTS kernel)
bitbake -c cleansstate mc:x86-2020:linux-intel-ese-lts-rt-5.10 (To clean sstate of RT kernel)

For Option B:
bitbake -c cleansstate mc:x86-rt-2020:linux-intel-ese-lts-5.10 (To clean sstate of LTS Kernel)
bitbake -c cleansstate mc:x86-rt-2020:linux-intel-ese-lts-rt-5.10 (To clean sstate of RT kernel)

§

4.0 Next Step to Boot Up 11th Gen Intel® Core™ Processor Reference Validation Platform

This section describes the steps required to prepare a bootable Yocto Project*-based image and to boot up 11th Gen Intel® Core™ Processor Reference Validation Platform (RVP). Last sub-section contains the link to Intel® TCC Tools for next step.

4.1 Prepare a Bootable Image with the USB Flash Drive

Intel recommends using “bmaptool” to prepare a bootable image if the USB flash drive is being used as boot media (minimum of 64GB size is recommended).

1. Download the latest bmaptool release from https://github.com/intel/bmap-tools/releases into the Ubuntu* host system, where you build the Yocto Project-based image.
   ```
   $ curl -Lo bmaptool https://github.com/01org/bmap-tools/releases/download/v3.4/bmaptool && chmod +x bmaptool
   ```

2. Ensure that Python* module six is installed on the system.
   ```
   pip3 install six
   ```

3. Insert the USB flash drive and ensure all partitions of the target device (the USB flash drive in this case) are unmounted. Refer to Section 5.1 on how to find the USB Device.

WARNING: You could wipe off your hard drive if the wrong device is chosen.

4. Run the following command (assume the USB flash drive is using /dev/sdc) to generate a bootable USB flash drive:
   ```
   $ sudo ./bmaptool copy --bmap <path>/core-image-sato-xxx-<date>.wic.bmap <path>/core-image-sato-<date>.wic /dev/sdc
   ```

5. This step is required only if you want to use secure boot. Copy all the files below, which you have generated earlier by following the instructions in *Generate Key for Secure Boot with the Yocto Project*-based Image and Bootloader/UEFI BIOS* (Document number: 633630) into the USB flash drive:
   - DB.cer, KEK.cer and PK.cer.

NOTES:

- If you are copying to a different USB flash drive, make sure the USB flash drive is in an FAT32 format.
- If you are copying into the USB flash drive that is the same as the bootable version, copy it to the same level of /BOOT/EFI/bootx64.efi dir.

4.2 Boot up 11th Gen Intel® Core™ Processor Reference Validation Platform with USB Flash Drive

This section provides the steps to boot up the Reference Validation Platform (RVP) with a bootable USB flash drive prepared in the previous section. For more information on RVP, refer to the *Tiger Lake UP3 Platform Reference Validation Platform (RVP) User Guide* (Document number: 612924).

1. Insert a USB flash drive into the 11th Gen Intel® Core™ platform and ensure the BIOS has been flashed into the platform. For more details about UEFI BIOS and steps to flash, refer to UEFI Reference BIOS in Section 2.2 of *BSP Release Notes*.

2. Boot up the 11th Gen Intel® Core™ processor RVP by pressing the power button. Press F2 if you need to enter the BIOS menu for configuration or to select the boot option.

3. The Secure Boot feature is disabled by default in the UEFI BIOS. If you would like to use the UEFI Secure Boot feature, you will need to insert the security key that was generated in the previous section through the UEFI BIOS.

   For steps on how to install the keys into the UEFI BIOS menu, see *Generate Key for Secure Boot with the Yocto Project*-based Image and Bootloader/UEFI BIOS* (Document number: 633630).

4. Once the UEFI BIOS has passed, the log in screen will be shown. Type "root" to log in and no password is required by default.

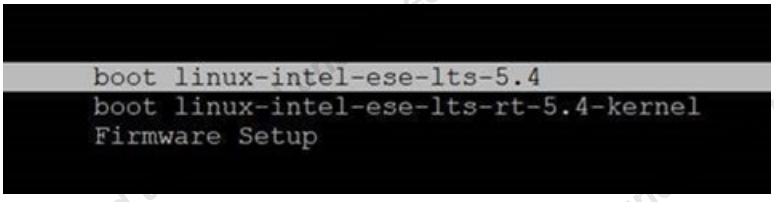
Note: The “Install” option is no longer applicable for this image. Make sure that the bmaptool is integrated in the image as guided in Section 3.2.2.1. Refer to Section 4.4 to install the image into the 11th Gen Intel® Core™ processor local drive.

4.3 Boot with PREEMPT_RT Kernel

If you build the Yocto Project*-based with **Option A**, by default, the system will be booted with the LTS kernel if no selection has been made on the boot menu.

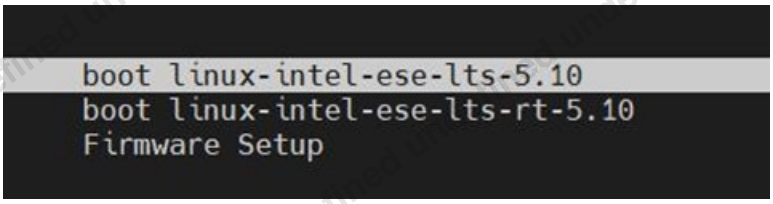
To choose the RT kernel for kernel 5.4, select "boot linux-intel-ese-lts-rt-5.4-kernel" from boot menu as shown in Figure 1.

Figure 1. Boot Menu with LTS Kernel as Default Boot Option (Kernel 5.4)



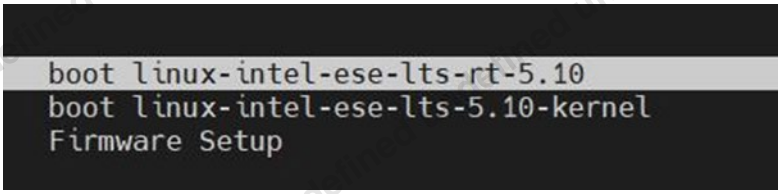
To choose the RT kernel for kernel 5.10, select "boot linux-intel-ese-lts-5.10" from boot menu as shown in Figure 2.

Figure 2. Boot Menu with LTS Kernel as Default Boot Option (Kernel 5.10)



If you wish to boot with the RT kernel by default, follow **Option B** from Section 3.2.3. Figure 3 shows the boot menu on kernel 5.10 with RT kernel as default boot option.

Figure 3. Boot Menu with RT Kernel as Default Boot Option (Kernel 5.10)



Disclaimer: Yocto Project* Board Support Package (BSP) kernel recipe resides in the recipes-kernel folder and integrates PREEMPT_RT Linux* kernel from the source downloaded from the following links:

Kernel 5.4: https://github.com/intel/linux-intel-lts/tree/5.4/preempt-rt

Kernel 5.10: https://github.com/intel/linux-intel-lts/tree/5.10/preempt-rt

Note that the PREEMPT_RT Linux* kernel is based on the same version of Real-Time Linux* Project without RT specific changes made by Intel. However, the kernel is built with different configurations (settings) to further reduce latency.

4.4 Install the Yocto Project*-based image into the 11th Gen Intel® Core™ Processor Local Drive

This section describes the steps to install the Yocto Project*-based image from one storage to another storage (for example, from the USB flash drive to the NVMe Storage). You may skip this section if you do not plan to install the image to another storage device.

4.4.1 First Time Image Installation into the RVP's Local Drive

1. On the host system, compress the image file for faster copying
   ```
   bzip2 -k core-image-sato-xxx-<date>.wic
   ```

2. Find a partition on the USB flash drive that is big enough for the compressed image file. For example, for the partition named data, one method of viewing partitions is to use gparted.
   ```
   sudo gparted
   ```

3. On the host system, create the mount point.
   ```
   sudo mkdir /mnt/usb
   ```

4. Mount the USB flash drive. Replace "sd<letter><number>" with the partition you identified in step 2.
   ```
   sudo mount /dev/sd<letter><number> /mnt/usb
   ```

5. Copy the compressed image file in step 1 and core-image-sato-xxx-<date>.wic.bmap file to USB flash drive.
   ```
   sudo cp <compressed_image_file> /mnt/usb

   sudo cp core-image-sato-sdk-intel-corei7-64-
   <datetime>.wic.bmap /mnt/usb
   ```

6. After copying, unmount the USB flash drive.
   ```
   sudo umount /mnt/usb
   ```

7. Insert the USB flash drive into the RVP.

8. Boot up 11th Gen Intel® Core™ processor by following the steps in Section 4.2.

9. Go to the USB flash drive partition where you copied the compressed image file and bmap (for example, ``/data``).

10. Flash the image to the local drive using bmaptool the same way as you flashed the image. Replace <target_drive> as appropriate.
    ```
    bmaptool copy --bmap <path>/core-image-sato-xxx-
    <date>.wic.bmap <path>/core-image-sato-<date>.wic.bz2
    /dev/<target_drive>
    ```

11. After successfully flashing, shut down the RVP and remove the USB flash drive. Start the RVP again. It will be booted from the target's local drive.

4.4.2 Subsequent Image Installation into the RVP's Local Drive

Note: Once an image is installed on the local drive, the target system will boot from that drive even when you specify boot from USB flash drive. This is an expected behavior as the boot media selection in BIOS menu is meant for different OS selection (for example, Yocto Project*-based image in the local drive and Windows* in USB flash drive).

To make sure the RVP is not booted from the local drive but from USB flash drive, the boot partition of the local drive needs to be removed. Then, a new installation can be done when the RVP boots up from the USB flash drive.

1. Boot the 11th Gen Intel® Core™ Processor (Tiger Lake – UP3) system from the local drive.
2. Run the following command to remove the boot partition from the drive. Replace <target_drive> as appropriate, for example, mmcblk0.
   ```
   dd if=/dev/zero of=/dev/<target_drive> bs=512 count=1
   ```
3. Reboot the 11th Gen Intel® Core™ Processor system.
4. After completing these steps, the target system will no longer find the boot partition on the local drive and will boot from the USB.
5. Follow Section 4.4.2 for the new installation.

4.5 Next Steps for Intel® TCC Tools

For Intel® TCC Tools, continue with the *Intel® TCC Tools Get Started Guide & Installation Instructions* for the specific version that you must download from the link:

https://software.intel.com/content/www/us/en/develop/tools/time-coordinated-computing-tools.html

§

5.0 Appendix

5.1 Further References

Below are some useful links for further understanding on some terms or Linux* commands that are used in this document:

1. https://man7.org/linux/man-pages/index.html

   This link provides online Linux* man-pages, where you can search for Linux* command and learn about the usage. For example, *curl*, *mkdir*, *cp*, and *mount*.

2. https://tldp.org/LDP/intro-linux/html/sect_03_01.html

   This link provides a general overview of the Linux* file system, in which some of them are used in this document (for example, /dev, /mnt).

3. https://github.com/intel/bmap-tools

   This link provides the source code and usage information on the bmap-tools used this document.

4. https://www.atlassian.com/git/tutorials/git-lfs

   This link provides a basic understanding on the usage of Git LFS.

5. https://www.yoctoproject.org/docs/1.6/bitbake-user-manual/bitbake-user-manual.html

   This link provides the BitBake User Manual for Yocto Project*.

6. https://www.tecmint.com/find-usb-device-name-in-linux/

   This link provides some tips on how to find USB device name in Linux*.

7. https://www.kernel.org/doc/Documentation/gpio/sysfs.txt

   This link provides information about GPIO sysfs Interface for userspace.

8. https://thesofproject.github.io/latest/index.html

   This link provides information about Sound Open Firmware.

5.2 I2S* Audio Settings and Configuration

In TGL-UP3, I2S* Audio via DSP is supported by using SOF (sound open firmware) firmware and topology files. For further information, refer https://thesofproject.github.io/latest/index.html.

Download *audio_fw_<release_milestone>.zip* from Section 2.2 of *BSP Release Notes* for quick deployment and testing by following the steps below.

5.2.1 BIOS Settings

These are the required BIOS settings to enable audio for a different codec.

For On-Board I2S* Codec (ALC 5660):

INTEL ADVANCED MENU → PCH-IO CONFIGURATION → HD AUDIO CONFIGURATION → HD AUDIO → AUDIO LINK MODE → SSP (I2S)

5.2.2 Integrate Firmware and Topology

1. After booting up the 11th Gen Intel® Core™ processor, unzip the *audio_fw_<release_milestone>.zip* file and copy *sof-tgl.ri* into the **/lib/firmware/intel/sof/** folder.
2. Copy the *sof-tgl-rt1308.tplg* folder to the **/lib/firmware/intel/sof-tplg** folder.

§