Yocto Project*-based Board Support Package for Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors (Code Name: Elkhart Lake)

Get Started Guide

September 2023

Maintenance Release 7 (MR7) – Kernel 5.15

Document Number: 619566-8.0

Contents

Figures

Tables

Revision History

Date	Revision	Description
September 2023	8.0	• Maintenance Release 7 (MR7) – Kernel 5.15 • Removed Intel® TCC Tools as no longer supported
March 2023	7.0	• Maintenance Release 6 (MR6) – Kernel 5.15
October 2022	6.1	• Updated mandatory workaround for the VC-Intrinsics/IGC URL issue in Section 3.2.1 and 3.31 for MR5
September 2022	6.0	• Maintenance Release 5 (MR5) – Kernel 5.15
August 2022	5.1	• Updated mandatory workaround for GNUlib error in Section 3.2.1
July 2022	5.0	• Maintenance Release 4 (MR4) – Kernel 5.15
April 2022	4.1	• Updated mandatory workaround for Grub re-base URL issue in Section 3.2.1 • Updated mandatory patch to enable the Kernel Boot options in 5.15 Bullpen RT SBL in Section 3.2.1
March 2022	4.0	• Maintenance Release 3 (MR3) – Kernel 5.15 • Updated Getting Started steps in Section 3.0 to contain both Kernel 5.4 (MR2) and Kernel 5.15 (MR3)
March 2022	3.2	Updated mandatory workaround in Section 3.1 for invalid source URL issue.
December 2021	3.1	Hotfix - released a patch that resolved S4 issue. Updated Section 3.2.3 for the steps to integrate the patch.
November 2021	3.0	MR2 Release.
August 2021	2.0	MR1 Release.
June 2021	1.2	• Updated Intel® TCC Tools link in Section 3.9. • Updated *Intel Atom® x6000E Series Processors Real-Time Tuning Guide* Document# in Section 1.4.
March 2021	1.1	Updated document title. Refer to Document #616424, Section 2.1 for supported silicon.
March 2021	1.0	PV Release and added a Note on workaround for Intel® TCC Mode issue in Section 4.2.
February 2021	0.9	Pre-production QS Release.

§

1.0 Introduction

This document provides instructions on how to build the Yocto Project*-based board support package (BSP) for the Elkhart Lake customer reference board (CRB). 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 CRB.

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 Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors (Kernel 5.15) - Release Notes (Document number: 732903)*

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
API	Application Programming Interface
ATT	Attribute Protocol
BSP	Board Support Package
CAN	Controller Area Network
CRB	Customer Reference Board
DMA	Direct Memory Access
eMMC	Embedded Multi-Media Card
GbE	Gigabit Ethernet
GCC	GNU Compiler Collection
HDCP	High-bandwidth Digital Content Protection
HECI	Host Embedded Controller Interface
IBECC	In-Band Error Checking and Correction

Term	Description
IFWI	Integrated Firmware Image
Intel® CSE	Intel® Converged Security Engine
Intel® PSE	Intel® Programmable Services Engine
Intel® TCC	Intel® Time Coordinated Computing
LTS	Long-Term Support
PXE	Pre-Boot eXecution Environment
RPMB	Replay Protected Memory Blob
SCSI	Small Computer System Interface
SOF	Sound Open Firmware
TSN	Time-Sensitive Networking
USB	Universal Serial Bus

1.2 Intended Audience

This document is for users of the Yocto Project*-based BSP for the Intel Atom® x6000E series, and Intel® Pentium® and Celeron® N and J Series processor CRBs.

1.3 Customer Support

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

1.4 Reference Documents

Table 2. Reference Documents

Document	Document Number/Location
Ethernet Time-Sensitive Networking for Elkhart Lake/Tiger Lake UP3 - Getting Started Guide	616446
Intel® PSE SDK Developer Guide	611877
Intel® PSE SDK API Guide	611876
Intel® Programmable Services Engine (Intel® PSE) SDK User Guide	611827
How to Video: Host System Build Environment Setup	608732
Intel® PSE SDK Get Started Guide	608527
Host System Build Environment Setup Guide	334828

Document	Document Number/Location
Wireless Connectivity Product for Yocto Project User Guide	617199
ECMA-393 Network Proxy Technology Support Using Linux* User Space Library (Elkhart Lake)	613398
Intel Atom® x6000E Series Processors Real-Time Tuning Guide	640979
Programming Elkhart Lake MAC Addresses Using Capsule Update	620481
Intel® In-Band Manageability Framework x86 for Intel Atom® x6000E Series processors, Intel® Pentium® and Celeron® N and J Series processors (Code name: Elkhart Lake) Release Notes	635491
Generate Key for Secure Boot with the Yocto Project*-based Image and Bootloader/UEFI BIOS	633630
Pre-boot Execution Environment (PXE) Boot with Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors (Code Name: Elkhart Lake) - Application Notes	635874
Yocto Project*-based Board Support Package for Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors (Code Name: Elkhart Lake) (Kernel 5.15) Release Notes	732903

Note: To download or search for a specific document, type the document number on the search bar in the Intel website.

§

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 Elkhart Lake board. The preparation and setup only need to be done 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 list provides the minimum host system configuration to build the BSP for the Yocto Project*. For Elkhart Lake, this configuration supersedes the other prerequisite 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* 18.04 LTS OS or higher (Refer to the Yocto Project* Quick Start for more information.)
- A minimum of 32 GB RAM and 500 GB disk space are recommended.
- High-speed network connectivity
- USB flash drive 64 GB 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* (Document 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 in 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 required toolchain if it is not available in your host machine:

```
$ sudo apt-get -y install socat gawk wget git-core diffstat
unzip texinfo build-essential chrpath libsdl1.2-dev xterm
libncurses5-dev patchutils curl libelf-dev elfutils
```

2. Download and install the git-lfs in the host 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 the current GCC version in your host machine:

```
$ gcc --version
$ g++ --version
# If your gcc version is not 7.3 or 8.2 and above, proceed to
the next steps
```

2. Install the required GCC version.

```
# Add ppa repository
$ sudo add-apt-repository ppa:ubuntu-toolchain-r/test

# If you are not able to add the ppa
$ sudo apt-get install --reinstall ca-certificates
$ sudo -E add-apt-repository ppa:ppaname/ppa

# Update and install gcc to take effect
$ sudo apt-get update
$ sudo 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

# To 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 BSP

This section consists of Getting Started with Kernel 5.15 BSP Release (MR7), which is the latest release.

Intel recommends migrating to the latest Board Support Package (BSP) release 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. Build steps in this section were performed based on **Python** = **2.7.17** and **Python3** = **3.6.9** in the Ubuntu build machine. Intel recommends using this version or newer for this release. For more information on how to switch or update Python version, refer to the appendix section.

3.1 Getting Started with Kernel 5.15 BSP Release

Refer to the *Yocto Project*-based Board Support Package for Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors (Kernel 5.15) Release Notes* (Document number: 732903) for further information on BSP Release.

3.1.1 Obtain Repository Sources 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 your home directory and include your 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.git -b refs/tags/release-142_ehl-mr7 -g all
```

5. Pull the repository meta-layers (-j8 for simultaneous downloads, increase/decrease per your needs):

```
$ 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 Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors (Kernel 5.15) Release Notes* (Document number: 732903).

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

3.1.2.1 Integrate Software Tools or Packages

1. This step integrates the bmaptool and/ or devmem2 into the Yocto Project*-based image. The bmap-tools is needed if you need to install the image from one storage to another storage on the Elkhart Lake CRB (refer to Section 4.4 for steps to install from one storage to another). Devmem2 is needed if you need to test the IBECC feature. Otherwise, you may skip this step.

```
$ cd <work_dir>/intel-embedded-system-enabling/meta-intel-
embedded-system-enabling/meta-intel-distro/conf/include
$ vi ese-x86-common.inc
#add the following lines
IMAGE_INSTALL:append = " bmap-tools"
IMAGE_INSTALL:append = " devmem2"
```

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: 732903).

```
$ vi <work_dir>/intel-embedded-system-enabling/meta-intel-
embedded-system-enabling/meta-intel-distro/conf/include/ese-
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. It is considered as an **optional** section as you may skip this section if none of the proprietary components is applicable to your interest. You may choose to integrate one or more components. For more information on the proprietary components, refer to the *Yocto Project*-based Board Support Package for Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors (Kernel 5.15) Release Notes* (Document number: 732903).

These files from Section 2.2 of the *Yocto Project-based BSP Release Notes* (Document number: 732903) are relevant in this section:

a. yocto_project_mr7_release.zip (ID# 743612)
   The zip file consists of recipes and binaries for wireless (meta-intel-wireless.zip) and Network Proxy Application (meta-libnetprox.tar.bz2)

b. intel-socwatch-yoctorecipe-ver1.zip (ID# 724016)
   The zip file consists of recipes to integrate the kernel module for **Intel® Soc Watch** and the user guide to download binary via Intel® System Bring-up Toolkit.

<u>Follow the instructions below if you choose to integrate the proprietary components</u>

1. Create a new folder (for example: proprietary) in the working directory and unzip the yocto_project_mr7_release.zip into the proprietary folder.

   ```
   $ mkdir <work_dir>/proprietary
   $ unzip yocto_project_mr7_release.zip
   ```

   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. Integrate components of the **Wi-Fi* and Bluetooth®** technology in the BSP

```
$ vi bblayers.conf

# Add the following lines
${TOPDIR}/../proprietary/meta-intel-wireless \
```

4. To integrate the **ECMA-393 network proxy** technology in the BSP:

   a. Edit bblayers.conf and ese-x86-common.inc

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

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

   # Add the following lines and change the path to where you
   have placed the binaries.
   IMAGE_INSTALL:append = " libnetprox"
   ```

   b. Update networkproxy.cfg ONLY if you need to change port. The config file is set for PSE GbE0 by default. If you need to use PSE GbE1, follow the path below:

   ```
   $ vi <work_dir>/intel-embedded-system-enabling/meta-intel-
   embedded-system-enabling/meta-intel-iot-bsp/recipes-
   kernel/linux/linux-config/bsp/networkproxy.cfg

   # change from "CONFIG_STMMAC_NETWORK_PROXY_PORT=0" to
   "CONFIG_STMMAC_NETWORK_PROXY_PORT=1"
   ```

   c. Edit the lts 5.15 kernel configuration file for updating the network proxy configurations:

   ```
   # vi <work_dir>/intel-embedded-system-enabling/meta-intel-
   embedded-system-enabling/meta-intel-iot-bsp/recipes-
   kernel/linux/linux-intel-iot-lts-5.15_git.bb
   ```

   Append this line at the end of the file:

   ```
   SRC_URI:append = " file://bsp/networkproxy.scc"
   ```

5. To integrate the kernel module for **Intel® SoC Watch**, edit the *bblayers.conf* and *ese-x86-common.inc*

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

$ vi <work_dir>/intel-embedded-system-enabling/meta-intel-
embedded-system-enabling/meta-intel-
distro/conf/include/ese-x86-common.inc
# Add the following line
KERNEL_PROVIDERS_EXTRA_MODULES[5.15] += " intel-socwatch-
driver"
```

3.1.2.3 Integrate Workarounds or Patches

1. **[Workaround]** This step includes the workaround for the known issue in ASPM (Active State Power Management). Follow the steps if the feature is required in your application. Otherwise, you can skip this step.

   Refer to workaround ID: 1508932757 from Section 3.1.6 Workarounds of the *Yocto Project*-based Board Support Package for Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors (Kernel 5.15) Release Notes* (Document number: 732903).

Note: This workaround is not applicable if Real-Time Performance is required.

```
$ vi <work_dir>/build/conf/multiconfig/x86-common.inc
# Add the following lines
APPEND:append = " pcie_aspm.policy=powersupersave"
#save and exit
```

If you have your own kernel patches that are relevant to your application, apply them 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 a few options to build different types of images:

Note: It is mandatory to choose Option B for Real Time use case

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.

Option C: Minimal image with LTS kernel as default boot kernel. This is a **non-validated** image that contains the same kernel driver as the image in Option A but with minimal user space software (for example, without GUI) for boot up. Wi-Fi drivers are provided by the default kernel and not from iwlwifi-backports.

Option D: Netboot image with PXE boot. This is a **non-validated** image to demonstrate the PXE boot feature in this release. This image is not recommended to be used for other features testing.

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

```
$ bitbake mc:x86-2021:core-image-sato-sdk
```

Location of Images: <work_dir>/build/tmp-x86-2021-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:x86-2021: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 the 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.

```
$ bitbake mc:x86-rt-2021:core-image-sato-sdk
```

Location of Images: <work_dir>/build/tmp-x86-rt-2021-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-2021: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.

Option C: Minimal 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

```
$ bitbake mc:x86-2021-minimal:core-image-full-cmdline
```

Location of Images: <work_dir>/build/tmp-x86-2021-minimal-glibc/deploy/images/intel-corei7-64/
Required Files: core-image-full-cmdline-intel-corei7-64--<date>.hddimg

Option D: Netboot image with PXE boot

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:x86-2021-minimal-netboot:core-image-full-cmdline
```

Location of Images: <work_dir>/build/tmp-x86-2021-minimal-netboot-glibc/deploy /images/intel-corei7-64/
Required File: core-image-full-cmdline-intel-corei7-64--<date>.rootfs.uefi-netboot.tar.bz2

Refer to the *PXE Boot Application Notes* (Document number: 635874) for more details.

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:x86-2021:linux-intel-iot-lts-5.15 (To clean sstate of LTS kernel)
bitbake -c cleansstate mc:x86-2021:linux-intel-iot-lts-rt-5.15 (To clean sstate of RT kernel)

For Option B:
bitbake -c cleansstate mc:x86-rt-2021:linux-intel-iot-lts-5.15 (To clean sstate of LTS kernel)

bitbake -c cleansstate mc:x86-rt-2021:linux-intel-iot-lts-rt-5.15 (To clean sstate of RT kernel)

§

4.0 Next Step to Boot Up CRB

This section describes the steps required to prepare a bootable Yocto Project*-based Board Support Package for Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors CRB. Last sub-section contains the link to Intel® TCC Tools for the 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 (minimum size of 64 GB 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, Further References, 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 have UEFI Secure Boot. Copy all of the files below, which you have generated earlier by following "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:

1. If you are copying to a different USB flash drive, make sure the USB flash drive is in an FAT32 format.
2. 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 Elkhart Lake CRB with USB Flash Drive

This section provides the steps to boot up the CRB with a bootable USB Flash Drive prepared in the previous section. For more information on CRB, refer to the *Elkhart Lake CRB User Guide* (Document number: 615859).

1. Insert a USB flash drive into the Elkhart Lake CRB and ensure the IFWI has been installed.

2. Boot up the Elkhart Lake CRB by pressing the power button. Press F2 if you need to enter the BIOS menu for configuration or to select the boot option.

3. **[Optional]** At the GRUB boot menu, press the key “e” on the keyboard. Append “i915.force_probe=*” to the end of the kernel boot parameters and press Ctrl + x to continue the booting process. This step will enable the multi-display function.

4. 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).

5. 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.

NOTES:

1. "Install" option is no longer applicable in this image. Make sure the bmaptool is integrated in the image as guided in Section 3.1.2.1. Refer to Section 4.4 to install the image into the Elkhart Lake CRB local drive.

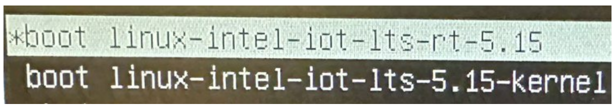
2. eMMC storage on the Elkhart Lake CRB is limited. Only the minimal image (refer to Section 3.1.3) is small enough to install in the eMMC.

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. If you wish to boot with the RT kernel by default, refer Section 3.1.3 for more information.

To choose the RT kernel, select "boot linux-intel-iot-lts-rt-5.15" from the boot menu as shown in the figure below.

Figure 1. Boot Menu



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 https://github.com/intel/linux-intel-lts

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 Elkhart Lake CRB 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. eMMC storage on the Elkhart Lake CRB is limited. Only the minimal image (refer to Section 3.1.3) is small enough to install in the eMMC.

4.4.1 First Time Image Installation into the CRB'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 onto 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 in the CRB.

8. Boot up the CRB 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 target's 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 CRB and remove the USB flash drive. Start the CRB again. It will be booted from the target's local drive.

4.4.2 Subsequent Image Installation into the CRB'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. 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 CRB 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 CRB boots up from the USB flash drive.

1. Boot the target 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 target 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 4.4.1 for the new installation.

4.5 Next Steps for Intel® Time Coordinated Computing

For Intel® TCC, refer to the link:

https://www.intel.com/content/www/us/en/developer/topic-technology/edge-5g/real-time/overview.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, which you can search for Linux* command and learn the usage. For example, this includes 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://docs.yoctoproject.org/bitbake/2.0/index.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 Elkhart Lake, 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 and unzip the Proprietary software package from the *Integrate Proprietary Components section* of this guide, to retrieve **audio_fw_<milestone_release>.zip** for quick deployment and testing by following the steps.

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 Elkhart Lake, unzip the audio_fw_<release_milestone>.zip file and copy sof-ehl.ri into the /lib/firmware/intel/sof/ folder.
2. Copy sof-ehl-rt5660.tplg into the /lib/firmware/intel/sof-tplg folder.

5.3 Kernel Configuration for Intel® Converged Security Engine (Intel® CSE)

Enable the following kernel configurations to allow the Intel® Converged Security Engine (Intel® CSE) to interact with the Yocto Project*-based OS.

CONFIG_INTEL_MEI=Y

Description: The Intel® Management Engine (Intel® ME) provides manageability, security, and media services for the system containing Intel® chipsets. If selected, */dev/mei0 misc device* will be created.

CONFIG_INTEL_MEI_HDCP=Y

Description: Enables HDCP support in conjunction with Intel® HD Graphics.

CONFIG_PCI=Y

Description: This must be enabled. Intel CSE communication (HECI) is based on PCI*.

CONFIG_INTEL_MEI_ME=Y

Description: Intel® ME-enabled Intel® chipsets.

CONFIG_RPMB=Y

Description: This is required for RPMB support.

CONFIG_RPMB_INTF_DEV=Y

Description: RPMB is one of the partitions in the eMMC*/NVMe* flash storage. Enable this configuration to ensure the Elkhart Lake platform supports Trusty secure storage.

CONFIG_SCSI_DMA=Y

Description: This is required for DMA support.

CONFIG_SCSI_LOWLEVEL=Y

Description: SCSI low-level drivers.

5.4 GPIO Control

5.4.1 BIOS Settings

These are some of the relevant BIOS settings to control GPIO:

Intel Advanced Menu > PCH IO Configuration > Security configuration > "Force unlocked all GPIO pads" > Enable

Intel Advanced Menu > PCH IO Configuration > PSE configuration > Eclite > Disable

Intel Advanced Menu > PCH IO Configuration > PSE configuration > GPIO/TGPIO group 0 & 1> Host owned

Intel Advanced Menu > PCH IO Configuration > PSE configuration > GPIO/TGPIO 0 & 1 Pin selection > Uncheck any pin that will be tested

Intel Advanced Menu > PCH IO Configuration > PSE configuration > Disable all the I2C, SPI Controller

5.4.2 GPIO Control via sysfs Interface

1. You can see the pins number and name from the following command:
   ```
   $ cat /sys/kernel/debug/pinctrl/INTC1020\:00/pins
   ```
   Example of the output:
   ```
   registered pins: 315
   pin 0 (CORE_VID_0) mode 1 0x44000700 0x00000018 0x00000100
   [LOCKED full, ACPI]
   pin 1 (CORE_VID_1) mode 1 0x44000700 0x00000019 0x00000100
   [LOCKED full, ACPI]
   pin 2 (VRALERTB) GPIO 0x82880102 0x0000001a 0x00000000 [LOCKED
   tx, ACPI]
   pin 3 (CPU_GP_2) GPIO 0x44000300 0x0000001b 0x00000100 [LOCKED
   full, ACPI]
   ```

2. Check the GPIO Controller number
   ```
   $ls /sys/class/gpio/
   ```
   Example of the output:
   ```
   Export gpiochip56 unexport
   ```

3. To control the GPIO pin, export it with a **number = A + B**; where
   A = pin number that wanted to test from step 1, and
   B = gpiochip controller number from step 2.

   For example, to test pin 20.
   number = 20 + 56 =76
   ```
   $echo 76 > /sys/class/gpio/export
   ```

4. Go to exported 76
   ```
   $cd /sys/class/gpio/gpio76
   ```
   a. read current direction and value.
      Example of the relevant commands:
      ```
      $cat direction
      $cat value
      ```
   b. change direction and value. Example of the relevant commands:
      ```
      $echo out > direction
      $echo in > direction

      $echo 0 > value
      $echo 1 > value
      ```

5. Intel recommends unexporting the GPIO pin when you no longer need to use the pin.
   ```
   $ $echo 76 > /sys/class/gpio/unexport
   ```

5.4.3 GPIO Offset

To find out the GPIO offset, take pin T43 as an example below:

1. Identify the "EDS Signal Name" from 613952_GPIO_Table based on "GPIO (EDS)" or "Pin Number"

   For example: GP_A11

GPIO (EDS)	Pin Numbe	EDS Signal Name
GP_A11	T43	GP_A11/PSE_GBE1_RGMII_TXD3

2. Find out the desired pin belongs to which "GPIO Community" from 613952_GPIO_Table. It is Community 3 in this example (GP_A11)

GPIO Community #
3

3. Check the pin number (in Linux) using this command:

   cat /sys/kernel/debug/pinctrl/INTC1020\:03/pins

```
pin 30 (RGMII1_TXDATA_3) 30:INTC1020:03 GPIO 0xc4000300 0x0000004b 0x00000100 [ACPI]
```

4. Check the pin range (in Linux) under Community 3 using this command in the screenshot:

```
root@intel-corei7-64:~# cat /sys/kernel/debug/pinctrl/INTC1020\:03/gpio-ranges
GPIO ranges handled:
0: INTC1020:03 GPIOS [797 - 813] PINS [0 - 16]
17: INTC1020:03 GPIOS [814 - 815] PINS [17 - 18]
19: INTC1020:03 GPIOS [816 - 839] PINS [19 - 42]
43: INTC1020:03 GPIOS [840 - 843] PINS [43 - 46]
```

5. Count the offset

```
19: INTC1020:03 GPIOS [816 - 839] PINS [19 - 42]
```

   Pin 30 is fall within this range and the offset is 827.

5.5 Entering S0ix State via Opportunistic Idle

1. Identify LPSS UART and update autosuspend_delay_ms and control

   Run the below command and check how many ttySx present
   ```
   $ udevadm info /dev/ttyS* | grep -i devpath
   ```

   Sample output (your output might be slightly different)
   ```
   root@intel-corei7-64:~# udevadm info /dev/ttyS* | grep -i
   devpath
   E: DEVPATH=/devices/pci0000:00/0000:00:19.2/dw-apb-
   uart.2/tty/ttyS4
   E: DEVPATH=/devices/pci0000:00/0000:00:1e.0/dw-apb-
   uart.3/tty/ttyS5
   E: DEVPATH=/devices/pci0000:00/0000:00:1e.1/dw-apb-
   uart.4/tty/ttyS6
   E: DEVPATH=/devices/pci0000:00/0000:00:11.0/tty/ttyS7
   E: DEVPATH=/devices/pci0000:00/0000:00:11.1/tty/ttyS8
   ```

   From the above output, five serial ports are available. Update the autosuspend_delay_ms and control for each of them by using the commands below:
   ```
   $echo "20000" > /sys/bus/pci/devices/0000:00:19.2/dw-apb-
   uart.7/power/autosuspend_delay_ms
   $echo "auto" > /sys/bus/pci/devices/0000:00:19.2/dw-apb-
   uart.7/power/control

   $echo "2000" > /sys/bus/pci/devices/0000:00:1e.0/dw-apb-
   uart.8/power/autosuspend_delay_ms
   $echo "auto" > /sys/bus/pci/devices/0000:00:1e.0/dw-apb-
   uart.8/power/control

   $echo "2000" > /sys/bus/pci/devices/0000:00:1e.1/dw-apb-
   uart.9/power/autosuspend_delay_ms
   $echo "auto" > /sys/bus/pci/devices/0000:00:1e.1/dw-apb-
   uart.9/power/control

   $echo "2000" > /sys/bus/pci/devices/0000:00:11.0/tty/
   ttyS7/power/autosuspend_delay_ms
   $echo "auto" > /sys/bus/pci/devices/0000:00:11.0/tty/
   ttyS7/power/control

   $echo "2000" > /sys/bus/pci/devices/0000:00:11.1/tty/ttyS8/
   power/autosuspend_delay_ms
   $echo "auto" > /sys/bus/pci/devices/0000:00:11.1/tty/ttyS8/
   power/control
   ```

2. Update autosuspend_delay_ms for Intel® PSE UART by using the commands below:
   ```
   $echo 2000 >
   /sys/bus/pci/devices/0000\:00\:11.0/power/autosuspend_delay_ms
   $echo 2000 >
   /sys/bus/pci/devices/0000\:00\:11.1/power/autosuspend_delay_ms
   ```

3. Stop the Net-SNMP Master agent and update control for Intel® PSE GbE by using the commands below:
```
$systemctl stop snmpd
$echo "auto" > /sys/devices/pci0000\:00/0000\:00\:1d.1/power/
control
$echo "auto" > /sys/devices/pci0000\:00/0000\:00\:1d.2/power/
control
```
4. Use powertop to set all tunable options to their good settings.
```
$powertop --auto-tune
```
5. Turn off display by using the commands below:
```
$export DISPLAY=:0
$xset dpms force off
```
6. Remove all USB devices and disconnect the external putty / terminal, if any.
7. Observe the system is off.
8. After a moment, wake the system up via pressing power button or re-connect USB devices.
9. To confirm if the system entered S0ix state (non-zero value should be shown and the number increase every time the system wake from S0ix state), use the command below:
```
$cat /sys/kernel/debug/pmc_core/slp_s0_resedency_use
```

§