NXP Semiconductors
User Guide

Document identifier: RTEYOCTOUG
Rev. 2.0, 29-Jul-2021

Real-time Edge Yocto Project User Guide

Chapter 1
Overview

This document describes how to build an Real-time Edge image for both i.MX and QorIQ (Layerscape) boards by using a Yocto Project build environment. It describes Real-time Edge Software Yocto layer and its usage.

The Yocto Project is an open source collaboration focused on embedded Linux® OS development. For more information on Yocto Project, see the Yocto Project page: www.yoctoproject.org. There are several documents on the Yocto Project homepage that describe in detail how to use the system. To use the basic Yocto Project without the Real-time Edge release layer, follow the instructions in the Yocto Project Quick Start found at www.yoctoproject.org/docs/current/yocto-project-qs/yocto-project-qs.html.

Real-time Edge layer is based on i.MX Yocto project and LSDK Yocto release.

- i.MX Yocto project provides i.MX boards support. For more information, refer to i.MX Yocto Project User's Guide.pdf.
- LSDK Yocto project provides Layerscape boards support. For more information, refer to https://www.nxp.com.cn/docs/en/user-guide/LSDKYOCTOUG.pdf.

Files used to build an image are stored in layers. Layers contain different types of customizations and come from different sources. Some of the files in a layer are called recipes. Yocto Project recipes contain the mechanism to retrieve source code, build, and package a component. The following list show the layers used in this release.

Real-time Edge layer

- **dynamic-layers**: includes updates for board-related recipes of i.MX and Layerscape.

```
├── imx-layer
└── qoriq-layer
```

- **recipes-extended**: includes recipes for Real-time Networking, Real-time System, and Industrial.
- **recipes-nxp**: Real-time Edge Image recipes

NOTE

1.1 End user license agreement

During the setup environment process of the Real-time Edge Yocto Project Community BSP, the NXP End-User License Agreement (EULA) is displayed. To continue to use the Real-time Edge Proprietary software, users must agree to the conditions of this license. The agreement to the terms allows the Yocto Project build to untar packages.

1.2 Related documentation

- For more information about i.MX Yocto project, refer to i.MX Yocto Project User's Guide.pdf.
- For more information about LSDK Yocto project, refer to https://www.nxp.com.cn/docs/en/user-guide/LSDKYOCTOUG.pdf.
- For more information about the real time features, refer to the *Real-time Edge Software User Guide* on the URL http://nww.preview.nxp.com/design/software/development-software/real-time-edge-software:REALTIME-EDGE-SOFTWARE.
- For flashing an SD card image for I.MX boards, refer i.MX Linux® User's Guide (IMXLUG).

Chapter 2
Features

Real-time Edge Yocto Project has the following features:

- Linux kernel recipe:
  - The kernel recipe contains two folders:
    - `dynamic-layers/imx-layer/recipes-kernel`: Linux for i.MX boards
    - `dynamic-layers/qoriq-layer/recipes-kernel`: Linux for Layerscape boards
  - `Linux 5.10.9-rt24` is the base of Linux kernel released for the Yocto Project.
- U-Boot recipe
  - The U-Boot recipe has two folders:
    - `dynamic-layers/imx-layer/recipes-bsp/u-boot/`: U-Boot for i.MX boards.
    - `dynamic-layers/qoriq-layer/recipes-bsp/u-boot/`: U-Boot for Layerscape boards.
  - `U-Boot 2020.04` is the U-Boot base released for Yocto Project.
  - `U-boot-script-distro boot` recipe provides distro boot script for normal and BareMetal images.
- Real Time Networking recipes:
  - `avahi`
  - `iproute2`
  - `genavb-tsn`
  - `libredblack`
  - `libyang`
  - `lldpd`
  - `linuxptp`
  - `netopeer2-cli`
  - `netopeer2-keystored`
  - `netopeer2-server`
  - `real-time-edge-nodejs-lbt`
  - `real-time-edge-prl`
  - `real-time-edge-sysrepo`
  - `sysrepo`
  - `tsn-scripts`
  - `tsntool`
- Real-Time System recipes:
  - `real-time-edge-baremetal`: Real-time Edge BareMetal recipe resides in `recipes-extended` directory. It provides BareMetal binary run on slave cores.
  - `real-time-edge-icc`: icc recipe resides in `recipes-extended` directory. It provides a tool to community between master/slave and slave/slave cores.
  - `jailhouse`

- Protocols recipes:
  - `canfestival`
  - `igh-ethercat`
  - `libnfc-nci`
  - `libopen62541`
  - `real-time-edge-libbee`
  - `real-time-edge-libblep`

Chapter 3
Host setup

To set up the Yocto Project with expected behavior on a Linux Host Machine, the packages and utilities described in the following sections must be installed.

An important consideration is the hard disk space required in the host machine. For example, when building on a machine running Ubuntu, the minimum hard disk space required is about 50 GB. It is recommended that at least 120 GB disk space is provided, which is enough to compile all backends together.

The recommended minimum Ubuntu version is 18.04 or later. The Chromium version 74 requires Ubuntu 18.04. The latest release supports Chromium V74, which requires an increase to the ulimit (number of open files) to 4098. If Chromium is not used, 18.04 should work.

NOTE

Earlier versions before 16.04 may cause the Yocto Project build setup to fail, because it requires python versions only available starting with Ubuntu 12.04. See the Yocto Project Reference Manual for more information.

Ubuntu 16.04 users have commented on errors during build for SDL. To fix this issue, comment out the following lines in `local.conf`, such as adding the `#` character:

```
#PACKAGECONFIG_append_pn-qemu-native = " sdl"
#PACKAGECONFIG_append_pn-nativesdk-qemu = " sdl"
```

3.1 Host packages

A Yocto Project build requires that some packages be installed for the build. These packages are documented under the Yocto Project. Go to Yocto Project Quick Start and check for the packages that must be installed for your build machine.

Essential Yocto Project host packages are:

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
build-essential chrpath socat cpio python python3 python3-pip python3-pexpect \
xz-utils debianutils iputils-ping python3-git python3-jinja2 libegl1-mesa \
libsdl1.2-dev pylint3 xterm rsnc curl
```

The configuration tool uses the default version of grep that is on your build machine. If there is a different version of grep in your path, it might cause builds to fail. One workaround is to rename the special version to something not containing "grep".

3.2 Setting up the repo utility

'Repo' is a tool based on Git that makes it easier to manage projects containing multiple repositories, provided they do not need to be on the same server. Repo complements very well the layered nature of the Yocto Project, making it easier for users to add their own layers to the Board Support Package (BSP).

To install the “repo” utility, perform these steps:

1. Create a `bin` folder in the home directory.

   ```
   $ mkdir ~/bin (this step may not be needed if the bin folder already exists)
   $ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
   $ chmod a+x ~/bin/repo
   ```

2. Add the following line to the `.bashrc` file to ensure that the `~/bin` folder is in your `PATH` variable.

   ```
   $ export PATH=~/bin:$PATH
   ```

Chapter 4
Yocto Project setup

First, make sure that git is set up properly using the commands below:

```
$ git config --global user.name "Your Name"
$ git config --global user.email "Your Email"
$ git config --list
```

The Real-time Edge Yocto Project Release directory contains a sources directory. This directory contains the recipes used to build one or more build directories, and a set of scripts used to set up the environment.

The recipes used to build the project come from both the community and Real-time Edge. The Yocto Project layers are downloaded to the `sources` directory. In this directory, the recipes that are used to build the project are set up.

The following example shows how to download the Real-time Edge recipe layers. For this example, a directory called `yocto-real-time-edge` is created for the project. Any other name can also be used, instead of this name.

```
$ mkdir yocto-real-time-edge
$ cd yocto-real-time-edge
$ repo init -u https://github.com/real-time-edge-sw/yocto-real-time-edge.git -b real-time-edge-
gatesgarth -m real-time-edge-2.0.0.xml
$ repo sync
```

When this process is completed, the source code is checked out into the directory `yocto-real-time-edge/sources`. You can perform repo synchronization, with the command `repo sync`, periodically to update to the latest code.

If errors occur during repo initialization, try deleting the `.repo` directory and running the repo initialization command again.

The `repo init` is configured for the latest patches in the line.

Chapter 5
Image building

This section provides the detailed information along with the process for building an image.

5.1 Build configurations

Real-time Edge provides the script `real-time-edge-setup-env.sh`, which simplifies the setup for both i.MX and Layerscape machines. To use the script, the name of the specific machine to be built for and the desired distro must be specified. The script sets up a directory and the configuration files for the specified machine and distro.

Real-time Edge supports the below NXP hardware platforms.

- imx6ull14x14evk
- imx8mmevk
- imx8mpevk
- ls1028ardb
- ls1012ardb
- ls1021atwr
- ls1021atsn
- ls1021aiot
- ls1043ardb
- ls1046ardb
- ls1046afrwy
- lx2160ardb
- lx2160ardb-rev2

Each build folder must be configured in such way that it uses only one distro. Each time the variable `DISTRO_FEATURES` is changed, a clean build folder is needed. Distro configurations are saved in the `local.conf` file in the `DISTRO` setting and are displayed when the bitbake is running. Here is the list of `DISTRO` configurations:

- `nxp-real-time-edge` – The normal image including Real-time and industrial package without BareMetal support.
- `nxp-real-time-baremetal` –The BareMetal image (some boards do not support this distro).

The syntax for the `real-time-edge-setup-env.sh` script is shown below:

```
$ DISTRO=<distro name> MACHINE=<machine name> source real-time-edge-setup-env.sh -b <build dir>
```

`DISTRO`**=<**distro configuration name**>** is the distro, which configures the build environment and it is stored in `meta-real-time-edge/conf/distro.`

`MACHINE`**=<**machine configuration name**>** is the machine name which points to the configuration file in `conf/machine` in `meta-freescale` and `meta-imx.`

`-b <build dir>` specifies the name of the build directory created by the `real-time-edge-setup-env.sh` script.

When the script is run, it prompts the user to accept the End User License Agreement (EULA). Once the EULA is accepted, the acceptance is stored in `local.conf` inside each build folder and the EULA acceptance query is no longer displayed for that build folder.

After the script runs, the working directory is the one just created by the script, specified with the `-b` option. A `conf` folder is created containing the files `bblayers.conf` and `local.conf`.

The <build dir>/conf/bblayers.conf file contains all the metalayers used in the Real-time Edge Yocto Project release. The local.conf file contains the machine and distro specifications:

```
MACHINE ??= 'imx8mpevk'
DISTRO ?= 'nxp-real-time-edge'

ACCEPT_FSL_EULA = "1"
```

The MACHINE configuration can be changed by editing this file, if necessary.

ACCEPT_FSL_EULA in the local.conf file indicates that you have accepted the conditions of the EULA.

5.2 Choosing a Real-time Edge Yocto project image

The Yocto Project provides images that are available on different layers.

Real-time Edge provides nxp-image-real-time-edge image, which contains Real-time Networking, Real-time System, and Protocols packages.

5.3 Building an image

The Yocto Project build uses the bitbake command. For example, bitbake <component> builds the named component. Each component build has multiple tasks, such as fetching, configuration, compilation, packaging, and deploying to the target rootfs. The bitbake image build gathers all the components required by the image and builds in the order of the dependency per task. The first build is the toolchain along with the tools required for the components to build.

The following command is an example of how to build an image:

```
$ bitbake nxp-image-real-time-edge
```

5.4 Bitbake options

The bitbake command used to build an image is bitbake <image name>. Additional parameters can be used for specific activities described below. Bitbake provides various useful options for developing a single component. To run with a bitbake parameter, the command to be used is below:

```
$ bitbake <parameter> <component>
```

<component> **is a desired build package.**

The following table provides some bitbake options.

Table 1. Bitbake options

Bitbake paramater	Description
-c fetch	Fetches if the downloads state is not marked as done.
-c cleanall	Cleans the entire component build directory. All the changes in the build directory are lost. The rootfs and state of the component are also cleared. The component is also removed from the download directory.
-c deploy	Deploys an image or component to the rootfs.
-k	Continues building components even if a build break occurs.
-c compile -f	It is not recommended to change the source code under the temporary directory. However, if it is changed, the Yocto Project might not rebuild it unless this option is used. Use this option to force a recompile after the image is deployed.

Table continues on the next page...

Table 1. Bitbake options (continued)

`-g`	Lists a dependency tree for an image or component.
`-DDD`	Turns on debug 3 levels deep. Each D adds another level of debug.

5.5 Build scenarios

The following are build setup scenarios for various configurations.

Set up the manifest and populate the Yocto Project layer sources using the commands below:

```
$ mkdir yocto-real-time-edge
$ cd yocto-real-time-edge
$ repo init -u https://github.com/real-time-edge-sw/yocto-real-time-edge.git -m default.xml
$ repo sync
```

The following sections give some specific examples. Replace the machine names and the backends specified to customize the commands.

5.5.1 Real-time Edge image on i.MX 8M Plus EVK

This builds a multimedia image with Real-time Edge packages. The command below can be used:

```
$ DISTRO=nxp-real-time-edge MACHINE=imx8mpevk source real-time-edge-setup-env.sh -b build-imx-real-
time-edge
$ bitbake nxp-image-real-time-edge
```

5.5.2 Real-time Edge BareMetal image on i.MX 8M Plus EVK

```
$ DISTRO=nxp-real-time-edge-baremetal MACHINE=imx8mpevk source real-time-edge-setup-env.sh -b build-
imx-baremetal
$ bitbake nxp-image-real-time-edge
```

1. Restarting a build environment

Sometimes, a new terminal window is opened or the machine is rebooted after a build directory is set up. In such cases, the setup environment script should be used to set up the environment variables and run a build again. The full `real-time-edge-setup-release.sh` is not needed.

```
$ source setup-environment <build-dir>
```

Chapter 6
Image deployment

Complete filesystem images are deployed to `<build directory>/tmp/deploy/images`. An image is, for the most part, specific to the machine set in the environment setup. Each image build creates a U-Boot, a kernel, and an image type based on the `IMAGE_FSTYPES` defined in the machine configuration file. Most machine configurations provide an SD card image (.wic) and a rootfs image (.tar). The SD card image contains a partitioned image (with U-Boot, kernel, rootfs, and other such files) suitable for booting the corresponding hardware.

6.1 Flashing an SD card image

An SD card image file '.wic' contains a partitioned image (with U-Boot, kernel, rootfs, and other files) suitable for booting the corresponding hardware. To flash an SD card image, run the following command:

```
$ bunzip2 -dk -f ~<image_name>.wic.bz2
$ sudo dd if=<image name>.wic of=/dev/sd<partition> bs=1M conv=fsync
```

For i.MX, see Section "Preparing an SD/MMC card to boot" in the i.MX Linux® User's Guide (IMXLUG).

For Layerscape, see LSDKYOCTOUG.pdf.

Chapter 7
Building packages Based on i.MX Yocto release

This chapter describes how to add packages of meta-real-time-edge into i.MX Yocto Project.

Table 2. Selected packages on i.MX Yocto Project

Package	Recipe	Real-time Edge Linux	i.MX Linux
IGH EtherCAT master stack	igh-ethercat	Y	Y
LinuxPTP	linuxptp	Y	Y
OPC-UA	libopen62541	Y	Y
real-time-edge-sysrepo	real-time-edge-sysrepo	Y	N
Jailhouse	jailhouse	Y	Y
Real-time Edge BareMetal		Y	N
Preempt-RT Linux		Y	N

7.1 Downloading i.MX Yocto Release and Real-time Edge Yocto Layer

Install i.MX Yocto project, referring to the user guide:

1. Download i.MX Yocto release:

```
$ mkdir imx-yocto-bsp
$ cd imx-yocto-bsp
$ repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-linux-gatesgarth
-m imx-5.10.9-1.0.0.xml
$ repo sync
```

2. Download Real-time Edge Yocto layer:

```
$ cd sources
$ git clone https://github.com/real-time-edge-sw/meta-real-time-edge.git
```

Enabling meta-real-time-edge layer in i.MX Image Build

1. Setup build environment:

```
$ cd imx-yocto-bsp (The top directory of repo)
$ DISTRO=fsl-imx-wayland MACHINE=<machine name> source imx-setup-release.sh -b build-real-time-
edge
```

2. Add meta-real-time-edge to bblayers.conf under specific build folder

```
$ cd build-real-time-edge (The build-directory)
$ vim conf/bblayers.conf
# Add the below setting
BBLAYERS += "${BSPDIR}/sources/meta-real-time-edge"
```

7.2 Selecting Packages of Real-time Edge Yocto Layer

7.2.1 Packages from Real-time Edge Yocto layer

The packages from Real-time Edge Yocto layer can be added into the i.MX image separately. These are the following:

- igh_ethercat
- real-time-edge-sysrepo
- libopen62541 (OPC UA)

To select the package, add it to the `IMAGE_INSTALL` on the `local.conf` as below:

For example, use the below commands to add the `igh-ethercat` package:

Adding igh-ethercat:

```
$ vim conf/local.conf

# Add package
IGH_ETHERCAT_imx8mmevk = " fec "
IGH_ETHERCAT ??= " "
PACKAGECONFIG_append_pn-igh-ethercat = " ${IGH_ETHERCAT} "
IMAGE_INSTALL += " igh_ethercat "
```

NOTE

For i.MX Yocto release, the FEC Ethernet driver is built in kernel and only the EtherCAT generic module could be used. In order to use native EtherCAT-capable module on i.MX 8M Mini platform, users need to compile FEC Ethernet driver as a module by setting "`CONFIG_FEC=m`" in kernel configuration and set `DEVICE_MODULES` to "`fec`" as described in `Chapter 5.1.5.2 IGH EtherCAT Setup` of Real-time Edge Software User Guide.

Adding OPC UA

```
$ vim conf/local.conf
# Add package
# Select OPC UA example application
include ${BSPDIR}/sources/meta-real-time-edge/conf/distro/include/libopen62541.inc
LIBOPEN62541_LOGLEVE = "300"
IMAGE_INSTALL += " libopen62541"
```

7.2.2 Packages in i.MX Yocto layer

The packages that are in i.MX Yocto layer are overridden when adding `meta-real-time-edge` layer. If it is required to keep the original package instead of using Real-time Edge packages, you must add these packages to “`BBMASK`” in the `bblayer.conf` as listed below.

- avahi
- ethtool
- iproute2
- jailhouse
- linuxptp
- lldpd
- tsntool

Below is the configuration that can be used to mask the packages in `bblayer.conf`:

```
$ vim conf/bblayers.conf
# Add the below setting
```

```
BBMASK += "meta-real-time-edge/recipes-extended/jailhouse/*.bbappend"
BBMASK += "meta-real-time-edge/recipes-extended/tsntool/tsntool_%.bbappend"
BBMASK += "meta-real-time-edge/recipes-extended/ethtool/ethtool_%.bbappend"
BBMASK += "meta-real-time-edge/recipes-extended/linuxptp/linuxptp_3.1.bbappend"
BBMASK += "meta-real-time-edge/recipes-extended/avahi/avahi_%.bbappend"
BBMASK += "meta-real-time-edge/recipes-extended/iproute2/iproute2_%.bbappend"
BBMASK += "meta-real-time-edge/recipes-extended/lldpd/lldpd_%.bbappend"
```

The above packages that can be selected running on i.MX Yocto layer.

For example:

```
$ vim conf/local.conf

# Add package
IMAGE_INSTALL +=  "  \
    linuxptp \
    jailhouse \
    iproute2 \
    lldpd \
    avahi-daemon \
    avahi-utils \
"
```

7.3 Building the image

After adding the package, users can start to build an image with the selected Real-time edge packages.

Build the i.MX image using the command below:

```
$ bitbake imx-image-multimedia
```

7.4 Running packages on i.MX release

1. **Running Jailhouse**

   Refer to `Chapter 3.3.2 Running PREEMPT_RT Linux in Inmate` and `Chapter 3.3.3 Running Jailhouse Examples In Inmate` of Real-time Edge Software User Guide.

2. **Running LinuxPTP**

   Refer to `Chapter 4.3.5 Quick Start for IEEE 1588 and` and `Chapter 4.3.6 Quick Start for IEEE 802.1AS` of Real-time Edge Software User Guide.

3. **Running IGH-EtherCAT**

   Refer to `Chapter 5.1.5.2 IGH EtherCAT Setup` of Real-time Edge Software User Guide.

4. **Running OPC-UA**

   Refer to `Chapter 5.3 OPC UA` of Real-time Edge Software User Guide.

Chapter 8
Building packages based on Layerscape Yocto release

This chapter describes how to add packages of meta-real-time-edge into the Layerscape Yocto Project.

Table 3. Selected packages on Layerscape Yocto project

	Recipe	Real-time Edge Linux	Layerscape Linux
IGH EtherCAT master stac	igh-ethercat	Y	Y
LinuxPTP	linuxptp	Y	Y
OPC-UA	libopen62541	Y	Y
real-time-edge-sysrepo	real-time-edge-sysrepo	Y	Y
Jailhouse	jailhouse	Y	Y
Real-time Edge BareMetal		Y	Y
Preempt-RT Linux		Y	Y

8.1 Downloading LSDK Yocto release and Real-time Edge Yocto layer

1. Download LSDK Yocto release using the commands below:

```
$ mkdir yocto-sdk
$ cd yocto-sdk
$ repo init -u https://source.codeaurora.org/external/qoriq/qoriq-components/yocto-sdk -b
gatesgarth
$ repo sync
```

2. Download Real-time Edge Yocto layer using the commands below:

```
$ cd sources
$ git clone https://github.com/real-time-edge-sw/meta-real-time-edge.git
```

8.2 Enabling meta-real-time-edge layer in Layerscape Image Build

1. Setup the build environment:

```
$ . ./setup-env -m ls1028ardb
```

2. Add `meta-real-time-edge` to `bblayers.conf` under the specific build folder.

```
$ vim conf/bblayers.conf
# Add the below setting
BBLAYERS += " ${TOPDIR}/../sources/meta-real-time-edge"
```

8.3 Selecting packages of Real-time Edge Yocto Layer

8.3.1 Packages from Real-time Edge Yocto layer

The packages included in the Real-time Edge Yocto layer can be added into the Layerscape image separately. These packages are:

- igh_ethercat

- real-time-edge-sysrepo
- libopen62541 (OPC UA)

To select the package, add them to the `IMAGE_INSTALL` on the `local.conf` as shown below:

Adding igh-ethercat:

```
$ vim conf/local.conf

# Add package
IGH_ETHERCAT   ??=   " "
IMAGE_INSTALLL_append = " igh_ethercat "
```

Adding real-time-edge-sysrepo

```
$ vim conf/local.conf
# Add package
REAL_TIME_EDGE_SYSREPO_ls1028ardb = ""
REAL_TIME_EDGE_SYSREPO_ls1021atsn = "real-time-edge-sysrepo-tc"
PACAGECONFIG_append_pn-real-time-edge-sysrepo = "${REAL_TIME_EDGE_SYSREPO}" IMAGE_INSTALL_append = "
real-time-edge-sysrepo "
```

Adding OPC UA

```
$ vim conf/local.conf
# Add package
# Select OPC UA example application
include ../sources/meta-real-time-edge/conf/distro/include/libopen62541.inc
LIBOPEN62541_LOGLEVE = "300" IMAGE_INSTALL_append = " libopen62541"
```

8.3.2 Packages in Layerscape Yocto layer

The below packages that are in Layerscape Yocto layer are overridden when adding the `meta-real-time-edge` layer.

- `avahi`
- `ethtool`
- `iproute2`
- `jailhouse`
- `linuxptp`
- `lldpd`
- `tsntool`

If you require to keep the original package instead of using Real-time Edge packages, add these packages to `"BBMASK"` in the `bblayer.conf` file as shown below.

```
$ vim conf/bblayers.conf
# Add the below setting
BBMASK += "meta-real-time-edge/recipes-extended/jailhouse/*.bbappend"
BBMASK += "meta-real-time-edge/recipes-extended/tsntool/tsntool_%.bbappend"
BBMASK += "meta-real-time-edge/recipes-extended/ethtool/ethtool_%.bbappend"
BBMASK += "meta-real-time-edge/recipes-extended/linuxptp/linuxptp_3.1.bbappend"
BBMASK += "meta-real-time-edge/recipes-extended/avahi/avahi_%.bbappend"
BBMASK += "meta-real-time-edge/recipes-extended/iproute2/iproute2_%.bbappend"
BBMASK += "meta-real-time-edge/recipes-extended/lldpd/lldpd_%.bbappend"
```

The above packages can be selected running on Layerscape Yocto layer. To select the package, the package name needs to added into "IMAGE_INSTALL".

For example:

```
$ vim conf/local.conf

# Add package
IMAGE_INSTALL_append = " \
    linuxptp \
    jailhouse \
    iproute2 \
    lldpd \
    avahi-daemon \
    avahi-utils \
"
```

8.4 Building the image

After adding the package, one can start to build an image with selected Real-time edge package.

Build Layerscape image using the command below:

```
$ bitbake fsl-image-networking
```

8.5 Running packages on Layerscape

1. Running Jailhouse

   The below process takes LS1028ARDB as example.

   a. Get `fsl-ls1028a-rdb-jailhouse.dtb` from Real-time Edge Release. Other images are built according to above process.

   b. Run the below command under U-Boot to boot up the board.

   ```
   # setenv bootargs "root=/dev/ram0 rw earlycon=uart8250,0x21c0500
   console=ttyS0,115200  ramdisk_size=0x10000000"
   # tftp 0x82000000 Image; tftp 0xa0000000 fsl-image-networking-ls1028ardb.ext2.gz.u-
   boot;tftp 0x90000000 fsl-ls1028a-jailhouse-rdb.dtb;
   # booti 0x82000000 0xa0000000 0x90000000;
   ```

   c. Transfer Linux kernel binary "Image" to folder `/usr/share/jailhouse/inmates/kernel/`.

   For other steps, Please refer to `Chapter 3.3.2 Running PREEMPT_RT Linux in Inmate` and `Chapter 3.3.3 Running Jailhouse Examples In Inmate` of Real-time Edge Software User Guide.

2. Running LinuxPTP

   Please refer to `Chapter 4.1.5 Quick Start for IEEE 1588 and` and `Chapter 4.1.6 Quick Start for IEEE 802.1AS` of Real-time Edge Software User Guide.

3. Running IGH-EtherCAT

   Please refer to `Chapter 5.1.5.2 IGH EtherCAT Setup` of Real-time Edge Software User Guide.

4. Running OPC-UA

   Please refer to `Chapter 5.3 OPC UA` of Real-time Edge Software User Guide.

Chapter 9
Revision history

The table below describes the revisions to this document.

Table 4. Revision history

Date	Version	Cross-reference	Changes
30-Jul-2021	2.0	-	First release for Real Time Edge software Rev 2.0

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 29-Jul-2021

Document identifier: RTEYOCTOUG

arm