



MaaXBoards

**(iMX8M, iMX8mini and
iMX8nano)**

Yocto Zeus 5.4 User

Manual

Revision History

Rev.	Description	Author	Date
V0.0	Initial version	Mitsuki	12142022

Contents

Revision History	2
Chapter 1 Introduction.....	5
1.1 Brief Introduction.....	5
1.2 General workflow	5
1.3 Metadata/input	6
1.4 Build system (source files).....	6
Chapter 2 Setup project	9
2.1 Host setup.....	9
2.1.1 Operating system.....	9
2.1.2 Host packages.....	9
2.1.3 Install the repo utility	9
2.2 Yocto setup	10
2.2.1 Configure GIT	10
2.2.2 Download meta layers from NXP	11
2.2.3 Clone meta-maaxboard git repository	12
2.2.4 Setup the yocto build enviroment.....	13
2.2.5 Source files and bbappend files	14
2.2.6 Build Yocto.....	15
2.2.7 Build output	15
2.2.8 Returning to this project at a later date.....	15
2.2.9 Wi-Fi and Bluetooth	15

Images

Image 1.1 Yocto build workflow5

Image 2.1 Final meta-maaxboard layer.....12

Image 2.2 U-boot bbappend14

Image 2.3 kernel bbappend14

Image 2.5 Wi-Fi and BLE bbappend15

Chapter 1 Introduction

1.1 Brief Introduction

This document demonstrate how to get started with a successful Yocto Zeus 5.4 build for the Avnet MaaXBoards(MaaXboard, MaaXboard mini and MaaXboard nano).

The Yocto Project (YP) is an open-source collaboration project that helps developers create custom Linux-based systems regardless of the hardware architecture. The project provides a flexible set of tools and a space where embedded developers worldwide can share technologies, software stacks, configurations, and best practices that can be used to create tailored Linux images for embedded and IOT devices. For more information on Yocto project, see the Yocto project page: www.yoctoproject.org/.

This document will be focus on metadata/inputs and build system (source files).

1.2 General workflow

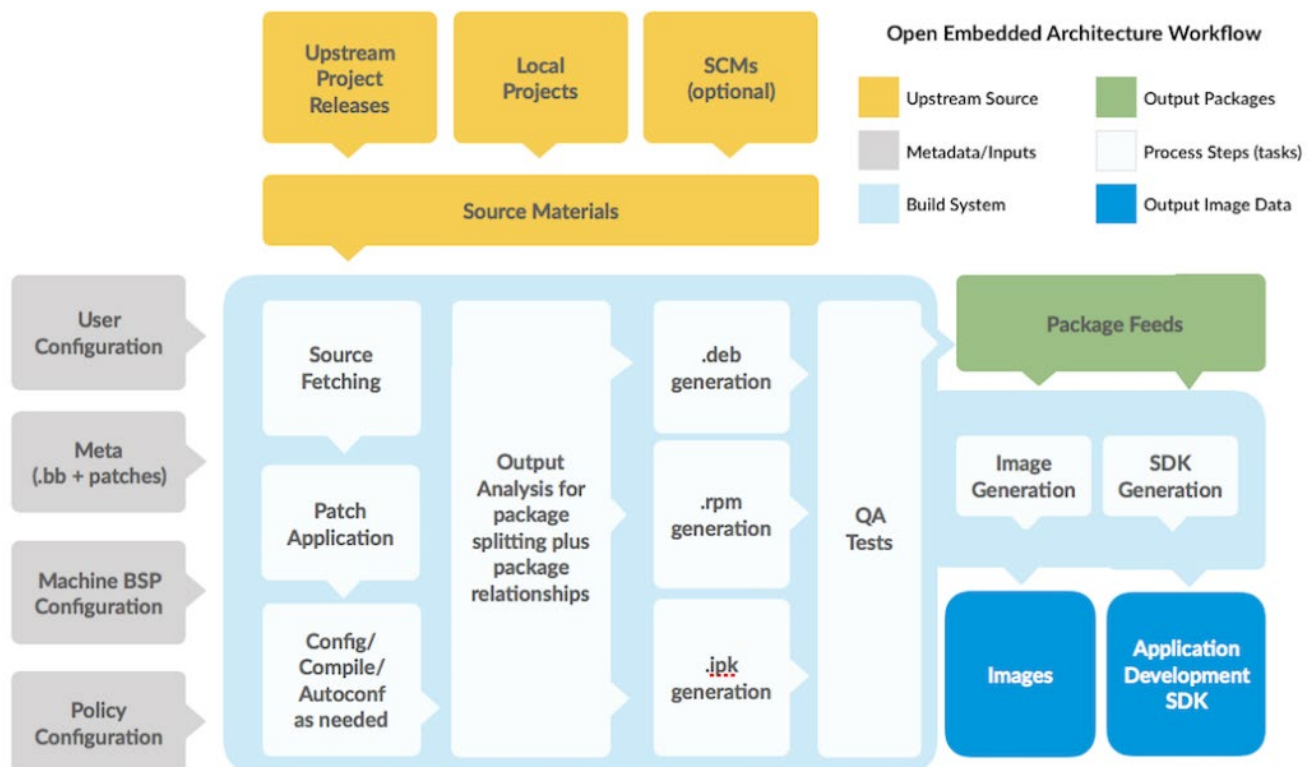


Image 1.1 Yocto build workflow

1.3 Metadata/input

The layers to be use in this document will be all the layers described in the NXP documentation (IMX_YOCTO_PROJECT_USERS_GUIDE) and the meta-maaxboard layer which contains MaaXBoards specific recipes, such as firmware, connectivity for Bluetooth and Wi-Fi. You can find all the meta-maaxboard layers on Avnet's [github](#) here.

- ◆ meta-maaxboard\conf\local.conf – contains all the local user configurations for your build environment
- ◆ meta-maaxboard\conf\bblayers.conf – defines layers, which are directory trees, traversed (or walked) by BitBake
- ◆ meta-maaxboard\conf\layer.conf – holds configuration files for the layer
- ◆ meta-maaxboard\conf\machine\maaxboard-ddr4-2g-sdcard.conf – maaxboard machine configuration
- ◆ meta-maaxboard\conf\machine\maaxboard-mini-ddr4-2g-sdcard.conf – maaxboard mini machine configuration
- ◆ meta-maaxboard\conf\machine\maaxboard-nano-ddr4-1g-sdcard.conf – maaxboard nano machine configuration

1.4 Build system (source files)

In order to customizes the MaaXboards configurations we need to locate the most important files for our kernel and bootloader, the needed files are listed below:

Kerne/linux-imx:

- ◆ linux-imx/arch/arm64/configs/maaxboard_defconfig – to retain a known set of kernel configurations
- ◆ linux-imx/arch/arm64/configs/maaxboard_mini_defconfig – to retain a known set of kernel configurations
- ◆ linux-imx/arch/arm64/configs/maaxboard_nano_defconfig – to retain a known set of kernel configurations
- ◆ linux-imx/arch/arm64/boot/dts/freescale/maaxboard.dtsi – device tree source include SoC level definitions maaxboard

- ◆ `linux-imx/arch/arm64/boot/dts/freescale/maaxboard-nano.dtsi` – device tree source include SoC level definitions maaxboard nano
- ◆ `linux-imx/arch/arm64/boot/dts/freescale/maaxboard-mini.dtsi` – device tree source include SoC level definitions maaxboard mini
- ◆ `linux-imx/arch/arm64/boot/dts/freescale/maaxboard-dcss-hdmi.dts` – enable dcss hdmi maaxboard
- ◆ `linux-imx/arch/arm64/boot/dts/freescale/maaxboard-dcss-mipi.dts` – enable dcss mipi maaxboard
- ◆ `linux-imx/arch/arm64/boot/dts/freescale/maaxboard-lcdif-mipi.dts` – enable lcd mipi maaxboard
- ◆ `linux-imx/arch/arm64/boot/dts/freescale/maaxboard-dual-display.dts` – enable dual display maaxboard
- ◆ `linux-imx/arch/arm64/boot/dts/freescale/maaxboard-usb0-device.dts` – enable usb maaxboard
- ◆ `linux-imx/arch/arm64/boot/dts/freescale/maaxboard-mini-mipi.dts` – enable usb maaxboard mini
- ◆ `linux-imx/arch/arm64/boot/dts/freescale/maaxboard-nano-mipi.dts` – enable usb maaxboard nano
- ◆ `linux-imx/arch/arm64/boot/dts/freescale/maaxboard-mini-device.dts` – enable usb maaxboard mini
- ◆ `linux-imx/arch/arm64/boot/dts/freescale/maaxboard-extended-gpio.dtsi` – gpios functionality maaxboard
- ◆ `linux-imx/arch/arm64/boot/dts/freescale/maaxboard-mini-extended-gpio.dtsi` – gpios functionality maaxboard mini
- ◆ `linux-imx/arch/arm64/boot/dts/freescale/maaxboard-nano-extended-gpio.dtsi` – gpios functionality maaxboard nano
- ◆ `linux-imx/arch/arm64/boot/dts/freescale/maaxboard-dsi-common.dtsi` – display serial interface maaxboard

BSP/u-boot-imx:

- ◆ u-boot-imx/configs/maaxboard_defconfig – to retain a known set of bootloader configurations
- ◆ u-boot-imx/configs/maaxboard_mini_defconfig – to retain a known set of bootloader configurations
- ◆ u-boot-imx/configs/maaxboard_nano_defconfig – to retain a known set of bootloader configurations
- ◆ u-boot-imx/include/configs/maaxboard.h – maaxboard configuration board
- ◆ u-boot-imx/include/configs/maaxboard_mini.h – maaxboard mini configuration board
- ◆ u-boot-imx/include/configs/maaxboard_nano.h – maaxboard nano configuration board
- ◆ u-boot-imx/include/configs/embest_env.h – to make the kernel configurable (run hdmi, dsi display, dual, etc)
- ◆ uEnv-mq.txt – uboot environment to boot from different dtb(created by Avnet not in u-boot-imx)
- ◆ u-boot-imx/arch/arm/mach-imx/imx8m/Kconfig – add config board definitions
- ◆ u-boot-imx/board/embest/maaxboard – DDR, SPL and maaxboard init functions
- ◆ u-boot-imx/board/embest/maaxboard_mini – DDR, SPL and maaxboard mini init functions
- ◆ u-boot-imx/board/freescale/maaxboard_nano – DDR, SPL and maaxboard nano init functions
- ◆ u-boot-imx/arch/arm/dts/maaxboard.dts – device tree source for maaxboard
- ◆ u-boot-imx/arch/arm/dts/maaxboard-mini.dts – device tree source for maaxboard mini
- ◆ u-boot-imx/arch/arm/dts/maaxboard-nano-mipi.dts – device tree source for maaxboard nano
- ◆ u-boot-imx/arch/arm/dts/maaxboard-nano.dtsi – device tree source for maaxboard nano
- ◆ u-boot-imx/arch/arm/dts/maaxboard-nano-extended-gpio – to configure the gpios(all versions)
- ◆ u-boot-imx/arch/arm/dts/Makefile – we need to add the maaxboards dtb in make file in order to compile it

Chapter 2 Setup project

This chapter will introduce the setup host and Yocto project build as described in NXP documentation.

2.1 Host setup

2.1.1 Operating system

These instructions assume that you are using one of these two versions of standard desktop Ubuntu:

- ◆ Ubuntu 18.04 LTS "Bionic Beaver" is the officially supported version for Yocto Zeus.
- ◆ Ubuntu 20.04 LTS "Focal Fossa" is what we are running on our machines, Yocto Zeus seems to work just fine.

These instructions also assume that you are using the default Bash shell that comes with Ubuntu.

2.1.2 Host packages

Install the following packages

```
sudo apt-get update && sudo apt-get install -y gawk wget git-core diffstat unzip texinfo  
gcc-multilib build-essential chrpath socat libstd1.2-dev xterm sed cvs subversion coreuti  
ls texi2html docbook-utils python-pysqlite2 help2man make gcc g++ desktop-file-utils libg  
l1-mesa-dev libglu1-mesa-dev mercurial autoconf automake groff curl lzop asciidoc u-boot-  
tools cpio sudo locales
```

2.1.3 Install the repo utility

Repo is a tool built on top of Git that makes it easier to manage projects that contain multiple repositories, which do not need to be on the same server

```
mkdir ~/bin (this step may not be needed if the bin folder already exists)  
  
chmod a+x ~/bin/repo  
curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > /usr/bin/repo  
  
export PATH="${HOME}/bin:${PATH}"
```

2.2 Yocto setup

This section will introduce all the needed setup we will need in order to build the MaaXboards recipes we choose.

2.2.1 Configure GIT

- ◆ List present configuration

```
$ git config -list
```

If the above command demonstrates that you already have a username and email configured, then you can skip the remainder of this section and continue with configuring Repo.

- ◆ Configure Git

```
$ git config --global user.name "Firstname Lastname"
$ git config --global user.email "EmailAddress@Domain.com"
```

- ◆ Confirm that Git is configured properly

```
$ git config -list
```

You should see at least these two lines with your name and email address

```
user.name=Firstname Lastname
user.email=EmailAddress@Domain.com
```

2.2.2 Download meta layers from NXP

Make a new directory called imx-yocto-bsp. We'll be downloading the board support package (BSP) and other meta layers here:

```
mkdir imx-yocto-bsp
sudo chown -R [USERNAME] imx-yocto-bsp
cd imx-yocto-bsp
```

Install the i.MX BSP repo and download the Yocto Project Layers. I'll be using Zeus here:

```
$ repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-linux-zeus -
m imx-5.4.70-2.3.11.xml
repo sync
```

You should now see the following folders / files

```
imx-setup-release.sh
README
README-IMXBSP
setup-environment
sources
```

2.2.3 Clone meta-maaxboard git repository

All the needed files to build the kernel and universal bootloader can be downloaded from github:

[HinoAM/meta-maaxboard: Yocto meta-layer for MaaXBoard/Mini/Nano \(github.com\)](https://github.com/HinoAM/meta-maaxboard)

Go to sources folder

```
$ cd sources
```

Clone the meta-maaxboard layer

```
$ git clone https://github.com/HinoAM/meta-maaxboard.git  
$ cd meta-maaxboard  
$ git checkout zeus
```

At the end of this steps, you will have some repository like below image



Image 2.1 Final meta-maaxboard layer

2.2.4 Setup the yocto build enviroment

First some Yocto definitions

MACHINE=<machine>

Use EVK names for <machine> listed in Yocto Project Users Guide, section 5.1 "Build configurations"

DISTRO=fsl-imx-<backend> where <backend> refers to the graphics type:

xwayland = Wayland with X11 support - default distro

wayland = Wayland only

fb = Framebuffer (not supported for imx8)

Note: Each build folder can only support a single DISTRO

Setup the Yocto build environment

```
mkdir maaxboard-wayland
MACHINE=imx8mqevk DISTRO=fsl-imx-xwayland source imx-setup-release.sh -b maaxboard-wayland
```

This operation will generate two conf files under the path maaxboard-wayland

- ◆ local.conf
- ◆ bblayers.conf

You will need to modify both conf files according to your settings. Under the meta-maaxboard folder you will find a script that you can run and replace the files. You need to put the folder destination and the type of maaxboard you want, for example:

```
./replace-conf.sh maaxboard-wayland maaxboard
bblayers.conf maaxboards copied ... [OK]
local.conf maaxboard copied ... [OK]
Done
```

2.2.5 Source files and bbappend files

Inside of the meta-maaxboard folder, you will find the folder structure and files needed to replace the files in kernel and universal bootloader, this is possible with the bbappend files, after bitbake downloaded the source code from the repositories the bbappend files will replace the needed source files in order to build the kernel and universal bootloader for MaaXboards.

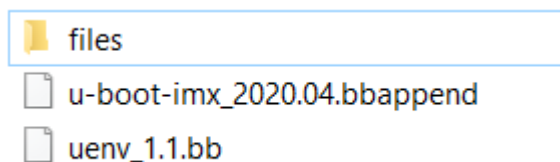


Image 3.2 U-boot bbappend

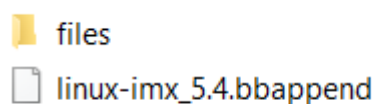


Image 4.3 kernel bbappend

2.2.6 Build Yocto

The Yocto Project provides some images that are available on different layers. Poky provides some images, meta-freescale and meta-freescale-distro provide others, and additional image recipes are provided in the meta-imx layer. The following table lists various key images, their contents, and the layers that provide the image.

Image name	Target	Provided by layer
core-image-minimal	A small image that only allows a device to boot.	poky
core-image-base	A console-only image that fully supports the target device hardware.	poky
core-image-sato	An image with Sato, a mobile environment and visual style for mobile devices. The image supports a Sato theme and uses Pimlico applications. It contains a terminal, an editor and a file manager.	poky
imx-image-core	An i.MX image with i.MX test applications to be used for Wayland backends. This image is used by our daily core testing.	meta-imx/meta-sdk
fsl-image-machine-test	An FSL Community i.MX core image with console environment - no GUI interface.	meta-freescale-distro
imx-image-multimedia	Builds an i.MX image with a GUI without any Qt content.	meta-imx/meta-sdk
imx-image-full	Builds an opensource Qt 6 image with Machine Learning features. These images are only supported for i.MX SoC with hardware graphics. They are not supported on the i.MX 6UltraLite, i.MX 6UltraLiteLite, i.MX 6SLL, i.MX 7Dual, i.MX 8MNanoLite, or i.MX 8DXL	meta-imx/meta-sdk

Image 2.4 Target Yocto builds

From maaxboard-wayland directory, run bitbake:

```
bitbake imx-image-full
```

2.2.7 Build output

Once it's done building, the build output is located under path:

```
imx-yocto-bsp/maaxboard-wayland/build/tmp/deploy/images/maaxboard-ddr4-2g-  
sdcard
```

You will find a wic file which is the needed file to flash into the SD card.

2.2.8 Returning to this project at a later date

Bitbake will not run if the environment is not configured. If you close the present shell (terminal) then you will lose the environment set up by imx-setup-release.sh. To set up our environment again:

```
$ source setup-environment maaxboard-xwayland
```

2.2.9 Wi-Fi and Bluetooth

Maaxboard include an Azure Wave which include the wi-fi and ble , the embedded zip already have the needed files the enable the features, the firmware comes in binary and blob we just need to tell bitbake where to put them in order to use them.

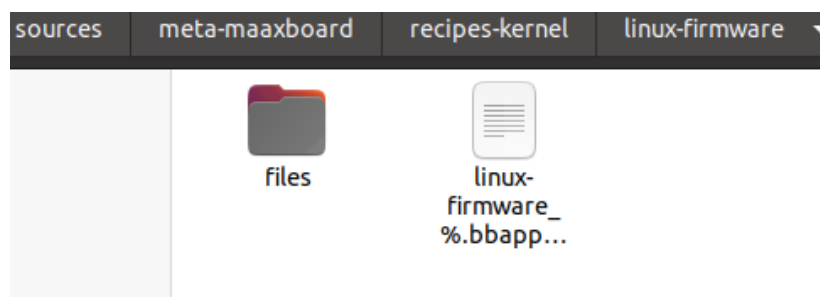


Image 5.5 Wi-Fi and BLE bbappend