# UG10195

## i.MX FRDM Software User Guide

**Rev. 1.0 — 20 December 2024** **User guide**

**Document information**

| Information | Content |
|---|---|
| Keywords | UG10195, i.MX, LF6.6.36_2.1.0, Yocto, FRDM-IMX93 |
| Abstract | This document describes how to build an image for an i.MX FRDM board by using a Yocto Project build environment and provides steps to install and run the image on an i.MX FRDM board. |

# 1 Overview

i.MX Freedom Development (FRDM) software release contains prebuilt images, documentation, and i.MX FRDM Yocto layer for i.MX FRDM boards.

This document describes how to build an image for an i.MX FRDM board by using a Yocto Project build environment and provides steps to install and run the image on an i.MX FRDM board. It also covers special i.MX FRDM features and how to use them.

i.MX FRDM layer contains Yocto recipes to support i.MX FRDM boards. It is based on i.MX Yocto Project and i.MX Software Release LF6.6.36_2.1.0. For more information on the i.MX Software Release, see IMXLINUX webpage.

## 1.1 Audience

This document is intended for software, hardware, and system engineers who are planning to use the product, and for anyone who wants to know more about the product.

## 1.2 Supported boards

Currently only the FRDM-IMX93 platform is supported.

## 1.3 References

This release includes the following references and additional information:

- *FRDM-IMX93 Quick Start Guide*: The document provides the steps to start the board quickly.
- *FRDM-IMX93 Board User Manual*: The document provides system setup, configurations, and detailed information on the design.
- *i.MX Linux Release Notes* (document RN00210): The document provides the release information.
- *i.MX Linux User's Guide* (document UG10163): The document provides information on installing U-Boot and Linux OS and using i.MX-specific features.
- *i.MX Yocto Project User's Guide* (document UG10164): The document describes the board support package (BSP) for NXP development systems using Yocto Project to set up host, install the tool chain, and build source code to create images.
- *i.MX Porting Guide* (document UG10165): The document provides instructions on porting the BSP to a new board.
- *i.MX Machine Learning User's Guide* (document UG10166): The document provides the machine learning information.
- *GoPoint for i.MX Applications Processors* (document GPNTUG): The document explains how to run GoPoint for i.MX Applications Processors.

## 1.4 Release contents

This release consists of the following:

- Documentation
- Prebuilt binaries:
  - SD card prebuilt image for the release target SoC
  - Kernel and device tree file (`.dtb`)
  - Boot images
  - Rootfs
- i.MX FRDM Yocto project release layer on GitHub

The i.MX FRDM software releases are named `imx-frdm-x.y`.

Where:

- `<x.y>`: Semantic versioning specification; x is the major version and y is the minor version.

Table 1 lists the contents included in each package.

**Table 1. Release contents**

| Component | Description |
|---|---|
| Linux OS Kernel and device trees | 6.6.36 |
| U-Boot | v2024.04 |
| SD card images | Prebuilt images for target i.MX FRDM boards |
| i.MX FRDM layer | To build images for target i.MX FRDM boards |

i.MX FRDM layer available at GitHub, aims to release the updated or new Yocto Project recipes and machine configurations for i.MX FRDM platform, which are stored in layers as follows:

- `meta-imx-bsp`: Updates `meta-imx/meta-imx-bsp` layer for the i.MX FRDM platform.
  - Linux Kernel recipe: Resides in the `recipes-kernel` folder and integrates Kernel patches for the i.MX FRDM platform.
  - U-Boot recipe: Resides in the `recipes-bsp` folder and integrates U-Boot patches for i.MX FRDM platform.
- `meta-nxp-demo-experience`: Updates `meta-nxp-demo-experience` layer to add i.MX FRDM platform support.
- `meta-nxp-connectivity`: Updates `meta-nxp-connectivity` layer to add Matter support for the i.MX FRDM platform.

### 1.4.1 Kernel and device trees

Table 2 describes the Kernel and device trees included in this release. A list of device tree files is provided to work with different accessories.

**Table 2. Kernel and device tree configurations**

| Filename | Description |
|---|---|
| `imx93-11x11-frdm.dtb` | The default dtb supports HDMI Display, MIPI CSI, onboard Wi-Fi/Bluetooth, and so on. |
| `imx93-11x11-frdm-tianma-wvga-panel.dtb` | 24-bit parallel supports 5-inch Tianma LCD |
| `imx93-11x11-frdm-dsi.dtb` | MIPI DSI supports a 7-inch Waveshare LCD |
| `imx93-11x11-frdm-ov5640.dtb` | MIPI CSI supports OV5640 sensors |
| `imx93-11x11-frdm-mt9m114.dtb` | Parallel CSI supports MT9M114 |
| `imx93-11x11-frdm-aud-hat.dtb` | Audio dtb for MX93AUD-HAT |
| `imx93-11x11-frdm-8mic.dtb` | Audio dtb for 8MIC-RPI-MX8 |

## 1.5 Instructions to get the AP1302 firmware

To get the AP1302 firmware from ONSemiconductor GitHub, perform the following steps:

1. Download the firmware image from ap1302_60fps_ar0144_27M_2Lane_awb_tuning.bin.
2. Rename it as `ap1302.fw`.
3. Create a folder `imx/camera/` under `/lib/firmware`.

4. Copy `ap1302.fw` to the target board under `/lib/firmware/imx/camera`.

### 1.6 UART output

Using the Linux PC's default CDC-ACM driver, the UART of FRDM-IMX93 can experience abnormal input/output when the UART cable is replugged.

To resolve this issue, follow the instructions as below:

1. Git Clone `https://github.com/WCHSoftGroup/ch343ser_linux.git`
2. Compile the CH343 Linux driver, remove the CDC-ACM driver, and then install the CH343 Linux driver
3. To use Minicom, ensure that you have upgraded it to version 2.9: http://ftp.hk.debian.org/debian/pool/main/m/minicom/minicom_2.9-5_amd64.deb
4. Connect to Minicom/PuTTy by using `/dev/CH343USB0`

The Windows driver can be downloaded from: https://www.wch.cn/downloads/CH343SER_EXE.html.

## 2 Introduction

i.MX FRDM boards are low-cost platform designed to show the most commonly used features of the i.MX applications processor in a small and low-cost package, which helps developers to get familiar with the processor before investing a large amount of resources in more specific designs.

For information on FRDM-IMX93 board, see FRDM-IMX93 webpage.

This document describes a method to integrate i.MX FRDM support into the NXP i.MX Linux BSP. The Yocto Project is an open source collaboration focused on embedded Linux OS development. For more information on the Yocto Project, see www.yoctoproject.org.

## 3 Getting started

The i.MX FRDM platform comes with a pre-built NXP Linux binary demo image flashed on the eMMC. Without modifying the binary inside, booting from the eMMC provides a default system with certain features for building other applications on top of Linux OS.

One can also copy images (U-Boot, Linux Kernel, device tree, and rootfs) to a boot device and set the boot switches to boot that device. This section describes how to set switches for booting and deploy images.

### 3.1 Basic terminal setup

Access to the i.MX FRDM's serial console UART is available via a CH343 USB serial interface. Connect the USB "DEBUG" port to a USB port on a host computer. CH343 drivers are available on Linux, Mac OS X, and Windows platforms. The first enumerated USB serial port is attached to the i.MX FRDM Cortex-A (U-Boot, Linux) serial console.

Connect a USB cable from the DEBUG port to the computer for console output, respectively with the following setup:

- Bits per second = 115200
- Data = 8 bits
- Parity = None
- Stop = 1 bit
- Flow control = None

## 3.2 Boot switch

The i.MX 93 processor provides multiple boot configurations, which can be selected using SW1 on the FRDM-IMX93 board, or by the boot configuration stored on the internal eFuse of the processor. On the FRDM-IMX93 board, the default Boot mode is from the eMMC device (SW1[1:4] is set to 0100). The other boot device is a microSD connector, which must set SW1[1:4] to 1100. To enter USB serial download, set SW1[1:4] to 1000.

Table 3 shows the switch settings for BOOT_MODE on the FRDM-IMX93 board.

**Table 3. Boot mode switch settings**

| SW1[1:4] | BOOT_MODE[3:0] | Boot core | Boot device |
|---|---|---|---|
| 1000 | 0001 | Cortex-A | Serial downloader (USB) |
| 0100 | 0010 | Cortex-A | uSDHC1 8-bit eMMC 5.1 |
| 1100 | 0011 | Cortex-A | uSDHC2 4-bit SD3.0 |

## 3.3 Downloading a pre-built image

The latest pre-built images for i.MX 93 FRDM are available on the FRDM-IMX93 webpage.

An SD card image file `.wic.zst` contains a partitioned image (with U-Boot, Kernel, rootfs, and so on) suitable for booting the corresponding hardware.

## 3.4 Universal update utility

Universal update utility (UUU) runs on Windows, Linux OS, FWIW, or Mac OS X (not tested yet) host and is used to download images to different devices on an i.MX board.

To use the UUU for i.MX 9, follow the instructions below:

1. Download UUU version 1.5.125 or higher from https://github.com/NXPmicro/mfgtools/releases.
   - On Windows, `uuu.exe` can be downloaded and directly used.
   - On Linux and Mac OS X, uuu can be downloaded and stored to a `/usr/local/bin` folder.
2. Connect a USB cable from a computer to the board's USB OTG/Type C port for downloading.
3. Connect a USB cable from the USB Type-C connector labeled "DEBUG" to a host computer for accessing the serial console described in Section 3.2.
4. Set the boot pin to Serial download mode, see Section 3.2.
   Table 4 shows the boot switch settings for FRDM-IMX93 to enter Serial download mode.

   **Table 4. Set up download mode for FRDM-IMX93**

   | Switch | D1 | D2 | D3 | D4 |
   |---|---|---|---|---|
   | SW1 | ON | OFF | OFF | OFF |

5. Burn the image:
   - To burn a single-boot image and rootfs to SD card, run the following command:

     ```
     uuu -b sd_all imx-boot-imx93frdm-sd.bin-flash_singleboot <rootfs.wic.zst>
     ```

   - To burn a single-boot image and rootfs to eMMC, run the following command:

     ```
     uuu -b emmc_all imx-boot-imx93frdm-sd.bin-flash_singleboot <rootfs.wic.zst>
     ```

6. To boot the board, change the boot switch and reset the board.

**Table 5. eMMC Boot mode on i.MX 93 FRDM**

| Switch | D1 | D2 | D3 | D4 |
|---|---|---|---|---|
| SW1 | OFF | ON | OFF | OFF |

**Table 6. SD Boot mode on i.MX 93 FRDM**

| Switch | D1 | D2 | D3 | D4 |
|---|---|---|---|---|
| SW1 | ON | ON | OFF | OFF |

For detailed instructions on how to use UUU, see https://github.com/nxp-imx/mfgtools/wiki.

## 3.5 Preparing an SD/MMC card to boot

This section describes the steps to prepare an SD/MMC card to boot up an i.MX board using a Linux host machine.

### 3.5.1 Copying the full SD card image

An SD card image file `.wic.zst` contains a partitioned image (with U-Boot, Kernel, rootfs, and so on) suitable for booting the corresponding hardware.

To flash an SD card image, run the following command:

```
zstdcat <image_name>.wic.zst | sudo dd of=/dev/sd<partition> bs=1M conv=fsync
```

For more information on flashing, see section "Preparing an SD/MMC card to boot" in *i.MX Linux User's Guide* (document UG10163).

### 3.5.2 Copying a bootloader image

This section describes how to load only the bootloader image when the full SD card image is not used.

To copy the U-Boot image to the SD/MMC card, run the following command:

```
$ sudo dd if=<boot image> of=/dev/sdx bs=1k seek=<offset> conv=fsync
```

Where:

- `<offset>` is 32 for i.MX 9.

## 3.6 Running Linux OS on the target

This section describes how to run a Linux image on the target using U-Boot.

The basic procedure for running Linux OS on an i.MX board is as follows. This document uses a specific set of environment variable names to make it easier to describe the settings.

1. Power on the board.
2. When U-Boot begins printing to the console, type a character to halt the automatic boot countdown.
3. Save the environment setup:

```
u-boot=> saveenv
```

4. Run the boot command:

```
u-boot=> run bootcmd
```

The commands `env default -f -a` and `saveenv` can be used to return to the default environment.

## 3.7 Running the Arm Cortex-M image

Some Arm Cortex-M core applications exist from the public Yocto builds. Here is an example to boot the Arm Cortex-M core on FRDM-IMX93:

1. Boot up the board to U-Boot, and load the Arm Cortex-M core image from the SD card to run:

```
=> fatload mmc 1:1 ${loadaddr} imx93-11x11-
evk_m33_TCM_rpmsg_lite_str_echo_rtos.bin
=> cp.b ${loadaddr} 0x201e0000 ${filesize}
=> bootaux 0x1ffe0000 0
```

2. Append `clk_ignore_unused` in U-Boot `mmcargs` environment before booting Linux.
3. Boot to Linux.
4. After login, ensure that the `imx_rpmsg_tty` Kernel module is inserted (`lsmod`) or insert it (`modprobe imx_rpmsg_tty`).
5. After the boot process succeeds, the Arm Cortex-M33 terminal displays the following information:

```
RPMSG String Echo FreeRTOS RTOS API Demo...

Nameservice sent, ready for incoming messages...
```

6. After the Linux RPMsg `tty` module has been installed, the Arm Cortex-M33 terminal displays the following information:

```
Get Message From Master Side : "hello world!" [len : 12]
```

7. The user can then input an arbitrary string to the virtual RPMsg `tty` using the following `echo` command on Cortex-A terminal:

```
echo test > /dev/ttyRPMSG<num>
```

Where:

- `<num>` is the allocated `ttyRPMsg` channel number. Find this number in the file system using a `ls` command.

8. The Cortex-M33 terminal displays the received string content and its length, as shown in the log:

```
Get Message From Master Side : "test" [len : 4]
Get New Line From Master Side
```

The i.MX 93 Cortex-M33 demo is available on the NXP Yocto Project mirror and it can also be retrieved using the following command on the Linux OS:

```
wget https://www.nxp.com/lgfiles/NMG/MAD/YOCTO/imx93-m33-demo-2.16.000.bin
```

The Cortex-M MCUXpresso SDK is distributed by the MCUXpresso Web Builder tool. To obtain the MCUXpresso SDK for the Cortex-M core of your i.MX SoC, visit http://mcuxpresso.nxp.com.

# 4 Image build using Yocto

## 4.1 Host setup

To achieve the Yocto Project's expected behavior in a Linux host machine, install the packages and utilities described below. 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. Provide at least 120 GB, which is enough to compile all backends together. For building machine learning components, at least 360 GB is recommended.

The recommended minimum Ubuntu version is 20.04 or later. The latest release supports Chromium v91, which requires an increase to the ulimit (number of open files) to 4098.

### 4.1.1 Docker

i.MX is now releasing Docker setup scripts in imx-docker. To set up a host build machine using Docker, follow the instructions in the README.

### 4.1.2 Host packages

A Yocto Project build requires specific packages to be installed for the build that are documented under the Yocto Project. Install the essential host packages on your build host. For information on image build using the Yocto Project, see Yocto Project Quick Build.

The following command installs the host packages:

```
$ sudo apt install gawk wget git diffstat unzip texinfo gcc build-essential
 chrpath socat cpio python3 python3-pip python3-pexpect xz-utils debianutils
 iputils-ping python3-git python3-jinja2 python3-subunit zstd liblz4-tool file
 locales libacl1
```

The configuration tool uses the default version of `grep` on your build machine. If a different version of `grep` is present in your path, it can cause build failure. To avoid this failure, rename the special version to a name that does not contain `grep`.

### 4.1.3 Setting up the repo utility

Repo is a tool built on top of Git that simplifies managing projects with multiple repositories, even if they are hosted on different servers.

To install the repo utility, perform the following 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. To ensure that the `~/bin` folder is in your PATH variable, add the following line to the `.bashrc` file:

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

## 4.2 Yocto project setup

To set up the Yocto project, perform the following steps:

1. Ensure that Git is set up properly with the commands below:

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

```
$ git config --list
```

The i.MX Yocto Project BSP release directory contains a sources directory, which contains the recipes for building one or more build directories, along with a set of scripts used to set up the environment.
The recipes required to build the project are sourced from the community i.MX Yocto Project and i.MX FRDM layer, which are downloaded into the sources directory. This ensures that all necessary recipes are set up to build the project.

2. The following example shows how to download the i.MX Yocto Project Community BSP recipe layers and i.MX FRDM layer. For this example, a directory called `imx-yocto-bsp` is created for the project. Any other name can be used instead of this name.
Download i.MX Yocto Release:

```
$ mkdir imx-yocto-bsp
$ cd imx-yocto-bsp
$ repo init -u https://github.com/nxp-imx/imx-manifest -b imx-linux-scarthgap
 -m imx-6.6.36-2.1.0.xml
$ repo sync
```

3. When this process is completed, the source code is checked out into the directory `imx-yocto-bsp/sources`.
4. 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.
5. Integrate i.MX FRDM layer into i.MX Yocto Project:

```
$: cd ./sources
$: git clone https://github.com/nxp-imx-support/meta-imx-frdm.git
$: cd meta-imx-frdm
$: git checkout imx-frdm-1.0
```

# 5 Image build

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

## 5.1 Build configurations

i.MX FRDM provides a script `imx-frdm-setup.sh` that simplifies the setup for i.MX FRDM machines. To use the script, the name of the specific machine to be built and the desired graphical backend must be specified.

The script sets up a directory and the configuration files for the specified machine and backend.

The following are i.MX machine configuration files (i.MX 9) that can be selected:

`imx93-11x11-lpddr4x-frdm` or `imx93frdm`

Each build folder must be configured in such a way that they only use 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` command is running.

The list of distro configurations are as follows:

- `fsl-imx-wayland`: Pure Wayland graphics.
- `fsl-imx-xwayland`: Wayland graphics and X11; X11 applications using EGL are not supported.

If no distro file is specified, the XWayland distro is set up as default.

To set preferred versions and providers, users can create their own distro file based on one of these configurations to customize their environment without updating the `local.conf`.

The syntax for the `imx-frdm-setup.sh` script is shown below:

```
$ DISTRO=<distro name> MACHINE=<machine name> source sources/meta-imx-frdm/
tools/imx-frdm-setup.sh -b <build dir>
```

Where:

- `DISTRO=<distro configuration name>` is the distro, which configures the build environment, and it is stored in `meta-imx/meta-imx-sdk/conf/distro`
- `MACHINE=<machine configuration name>` is the machine name, which points to the configuration file in `conf/machine` in `meta-imx-frdm`
- `-b <build dir>` specifies the name of the build directory created by the `imx-frdm-setup.sh` script

When the script is run, it prompts the user to accept the 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 running the script, it creates the working directory specified with the `-b` option. The `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 i.MX Yocto Project release and i.MX FRDM release.

The `local.conf` file contains the machine and distro specifications:

```
MACHINE ??= 'imx93frdm'
DISTRO ?= 'fsl-imx-xwayland'
ACCEPT_FSL_EULA = "1"
```

Where:

- 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 an image

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

Table 7 lists various key images, their contents, and the layers that provide the image recipes.

**Table 7. Image recipes**

| Image name | Target |
|---|---|
| `core-image-minimal` | A small image that only allows a device to boot |
| `imx-image-core` | Core image with basic graphics and no multimedia |
| `imx-image-multimedia` | Image with multimedia and graphics without any Qt content |
| `imx-image-full` | Image with multimedia, machine learning, and Qt |

## 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 build in 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 imx-image-full
```

## 5.4 BitBake options

The `bitbake` command used to build an image is `bitbake <image name>`. More 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 looks as follows:

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

Where:

- `<component>` is a desired build package.

Table 8 provides some BitBake options.

**Table 8. BitBake options**

| BitBake parameter | 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 to build components even if a build break occurs. |
| `-c compile -f` | Do not change the source code directly under the temporary directory. If changes are made, the Yocto Project cannot rebuild it unless this option is used. Use this option to force a recompile after the image is deployed. |
| `-g` | Lists a dependency tree for an image or component. |
| `-DDD` | Turns on debug 3 levels deep. Each D adds another level of debug. |
| `-s, --show-versions` | Shows the current and preferred versions of all recipes. |

## 5.5 Build scenarios

To build a BSP image for FRDM-IMX93, execute the following commands:

1. Download i.MX Linux Yocto Release:

```
$: mkdir ${MY_YOCTO} # this directory will be the top directory of the Yocto source code
$: cd ${MY_YOCTO}
$: repo init -u https://github.com/nxp-imx/imx-manifest -b imx-linux-scarthgap -m
 imx-6.6.36-2.1.0.xml
$: repo sync
```

2. Integrate `meta-imx-frdm` recipes into the Yocto code base:

```
$: cd ./sources
$: git clone https://github.com/nxp-imx-support/meta-imx-frdm.git
$: cd meta-imx-frdm
$: git checkout imx-frdm-1.0
```

3. Change to the top directory of the Yocto source code and execute the command below to set up environment for build:

```
#For i.MX93 FRDM
$: MACHINE=imx93frdm DISTRO=fsl-imx-xwayland source sources/meta-imx-frdm/tools/imx-frdm-setup.sh
 -b frdm-imx93
```

4. To generate Yocto images, run the following command:

```
$: bitbake imx-image-full
```

After executing previous commands, the Yocto images are generated at `<build directory>/tmp/deploy/images`. You can now use the `zstd` and `dd` commands or UUU tool to flash the images to a microSD card.

# 6 Matter support

This repository also contains Yocto recipes to add Matter support based on i.MX Matter 2024 Q3. For more information on i.MX Matter 2024 Q3, see https://github.com/nxp-imx/meta-nxp-connectivity.

To build the Yocto image with an integrated OpenThread Border Router, perform the following steps:

1. Run the following commands:

```
#Install the repo utility:
$: mkdir ~/bin
$: curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$: chmod a+x ~/bin/repo
$: export PATH=${PATH}:~/bin
```

```
#Download i.MX Software Release 2024 Q3:
$: mkdir ${MY_YOCTO} # this directory will be the top directory of the Yocto source code
$: cd ${MY_YOCTO}
$: repo init -u https://github.com/nxp-imx/imx-manifest -b imx-linux-scarthgap -m
 imx-6.6.36-2.1.0.xml
$: repo sync
```

```
#Download i.MX Matter Release 2024 Q3:
$: cd ${MY_YOCTO}/sources/meta-nxp-connectivity
$: git remote update
$: git checkout imx_matter_2024_q3
```

```
#Integrate meta-imx-frdm recipes into the Yocto code base:
$: cd ${MY_YOCTO}/sources
$: git clone https://github.com/nxp-imx-support/meta-imx-frdm.git
$: cd meta-imx-frdm
$: git checkout imx-frdm-1.0
```

2. Run i.MX Linux Yocto Project setup:
Change the current directory to the top directory of the Yocto source code and execute the command below:

```
#For i.MX93 FRDM:
$: MACHINE=imx93frdm-iwxxx-matter DISTRO=fsl-imx-xwayland source sources/meta-imx-frdm/tools/imx-
frdm-matter-setup.sh bld-xwayland-imx93
```

3. To generate Yocto images, run the following command:

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

After executing previous commands, the Yocto images are generated at `<build directory>/tmp/deploy/images`. You can now use the `zstd` and `dd` commands or UUU tool to flash the images to a microSD card.

# 7 Customization

## 7.1 Work with expansion boards

To work with certain expansion boards on FRMD-IMX93, the corresponding `dtb` file must be specified.

Table 9 shows the corresponding relationship.

**Table 9. Corresponding relationship**

| Expansion board | Interface | dtb |
|---|---|---|
| 7-inch Waveshare LCD | MIPI DSI | `imx93-11x11-frdm-dsi.dtb` |
| 5-inch Tianma LCD | 24 bit Parallel | `imx93-11x11-frdm-tianma-wvga-panel.dtb` |
| RPI-CAM-MIPI | MIPI CSI | `imx93-11x11-frdm.dtb` |
| RPI-CAM-INTB | 40pins | `imx93-11x11-frdm-mt9m114.dtb` |
| MX93AUD-HAT | 40pins | `imx93-11x11-frdm-aud-hat.dtb` |
| 8MIC-RPI-MX8 | 40pins | `imx93-11x11-frdm-8mic.dtb` |
| 2EL M.2 Module | M.2 Key E | `imx93-11x11-frdm.dtb` |

The `dtb` file can be specified in U-Boot:

```
u-boot=> setenv fdtfile <dtb name>
u-boot=> saveenv
u-boot=> boot
```

***Note:*** *Bluetooth and Wi-Fi are supported on i.MX FRDM through on-board chip (MAYA-W27x with NXP IW612) or external NXP SDIO IW612 (tested with Murata LBES5PL2EL). By default the on-board chip is used. If external hardware is used, it must rework the board according to the FRDM-IMX93 Board User Manual.*

## 7.2 How to build U-Boot and Kernel in a standalone environment

To build U-Boot and Kernel in a standalone environment, perform the following steps:

1. Generate an SDK that includes the necessary tools, toolchain, and a small rootfs to compile on the host machine.
   To generate an SDK from the Yocto Project build environment, run the following command:

   ```
   $ bitbake core-image-minimal -c populate_sdk
   ```

   The `populate_sdk` generates a script file that sets up a standalone environment without Yocto Project.
2. To build on, copy the `sh` file from the build directory in `tmp/deploy/sdk` to the host machine and execute the script to install the SDK. The default location is in `/opt`, but it can be placed anywhere on the host machine.
3. On the host machine, to build U-Boot and Kernel, perform the following steps:
   a. For i.MX 9 builds on the host machine, set the environment with the following command before building:

      ```
      $ source /opt/fsl-imx-xwayland/6.6-nanbield/environment-setup-aarch64-poky-linux
      $ export ARCH=arm64
      ```

      U-Boot:
      Download source by cloning as follows:

      ```
      $ git clone https://github.com/nxp-imx/uboot-imx -b lf_v2024.04
      $ cd uboot-imx
      $ git checkout lf-6.6.36-2.1.0
      #Apply patches for i.MX FRDM and “DIR” is the directory of this i.MX FRDM Software release.
      ```

```
$ git am DIR/meta-imx-frdm/meta-imx-bsp/recipes-bsp/u-boot/u-boot-imx/*.patch
```

b. For i.MX 93 11x11 FRDM board, run the following command:

```
$ make distclean
$ make imx93_11x11_frdm_defconfig
$ make
```

Kernel:
Download source by cloning as follows:

```
$ git clone https://github.com/nxp-imx/linux-imx -b lf-6.6.y
$ cd linux-imx
$ git checkout lf-6.6.36-2.1.0
#Apply patches for FRDM-IMX and "DIR" is the directory of this release.
$ git am DIR/meta-imx-frdm/meta-imx-bsp/recipes-kernel/linux/linux-imx/*.patch
```

c. To build the Kernel in the standalone environment for i.MX 9, execute the following commands:

```
$ make imx_v8_defconfig
$ make
```

# 8 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2024 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# 9 Revision history

Table 10 summarizes the revisions to this document.

**Table 10. Revision history**

| Document ID | Release date | Description |
|---|---|---|
| UG10195 v.1.0 | 20 December 2024 | Initial public release |

# Legal information

## Definitions

**Draft** — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

## Disclaimers

**Limited warranty and liability** — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

**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.

**Suitability for use** — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

**Applications** — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

**Terms and conditions of commercial sale** — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

**Export control** — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

**Suitability for use in non-automotive qualified products** — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

**HTML publications** — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

**Translations** — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

**Security** — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. 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.

**NXP B.V.** — NXP B.V. is not an operating company and it does not distribute or sell products.

## Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

**NXP** — wordmark and logo are trademarks of NXP B.V.

# Contents