



# NXP Hands-On LKCamp

© 2019 NXP Semiconductor, Inc. All rights reserved.

Version 1.1, December 10, 2019

# Table of Contents

1. Hardware Requirements .....	1
1.1. Getting Familiar to i.MX 8M Mini EVK Board .....	1
1.2. Preparing the i.MX 8M Mini EVK Board .....	2
2. Software Requirements .....	3
2.1. Terminal Emulator .....	3
2.1.1. Minicom Configuration (configured) .....	3
2.1.2. Minicom Usage .....	3
3. Getting Started with NXP BSP Release .....	4
3.1. NXP Linux BSP Release (4.19.35) .....	4
4. Developing Tools .....	5
4.1. GNU Cross-Platform Development Toolchain .....	5
4.1.1. GCC Arm GNU/Linux Toolchain i.MX 32 Bits .....	5
4.1.2. GCC Arm GNU/Linux Toolchain i.MX 64 Bits .....	5
5. Getting Started with i.MX Softwares .....	6
5.1. Solid State Storage Media (SD Card) .....	6
5.2. Bootloader .....	6
5.2.1. Internal U-Boot .....	6
5.2.2. External U-Boot .....	6
5.3. Kernel .....	7
5.3.1. Internal Kernel (4.19.35) .....	7
5.3.2. External Kernel (5.4.2) TODO SECTION .....	8
5.4. Root File System .....	8
6. Multimedia Support .....	9
6.1. Daughter Cards .....	9
6.1.1. MIPI-DSI HDMI display .....	9
6.1.2. OLED Display Panel .....	9
6.1.3. MINISASTOCSI Camera Daughter Card .....	9
6.1.4. Supported Resolutions .....	9
6.2. Changing the Device Tree File .....	10
6.3. Testing the Display and Cameras .....	11
6.3.1. Video Playback .....	11
6.3.2. Camera Playback .....	11
6.4. Changing the Display Resolution/Orientation .....	12
6.5. Multimedia Device Tree nodes .....	13
6.5.1. NXP BSP OLED Display Device Tree .....	13
6.5.2. Kernel Mainline OLED Display Driver Support .....	14
7. Android (P9.0.0_2.0.0_GA) .....	15
8. Introduction to M4 .....	17
8.1. Testing the M4 .....	17
9. Appendix .....	18
9.1. Removing Root Privileges Minicom .....	18
9.2. Additional Packages .....	18

# Chapter 1. Hardware Requirements

The following section describes the steps to setup the *i.MX 8M Mini EVK Board*.

## 1.1. Getting Familiar to i.MX 8M Mini EVK Board

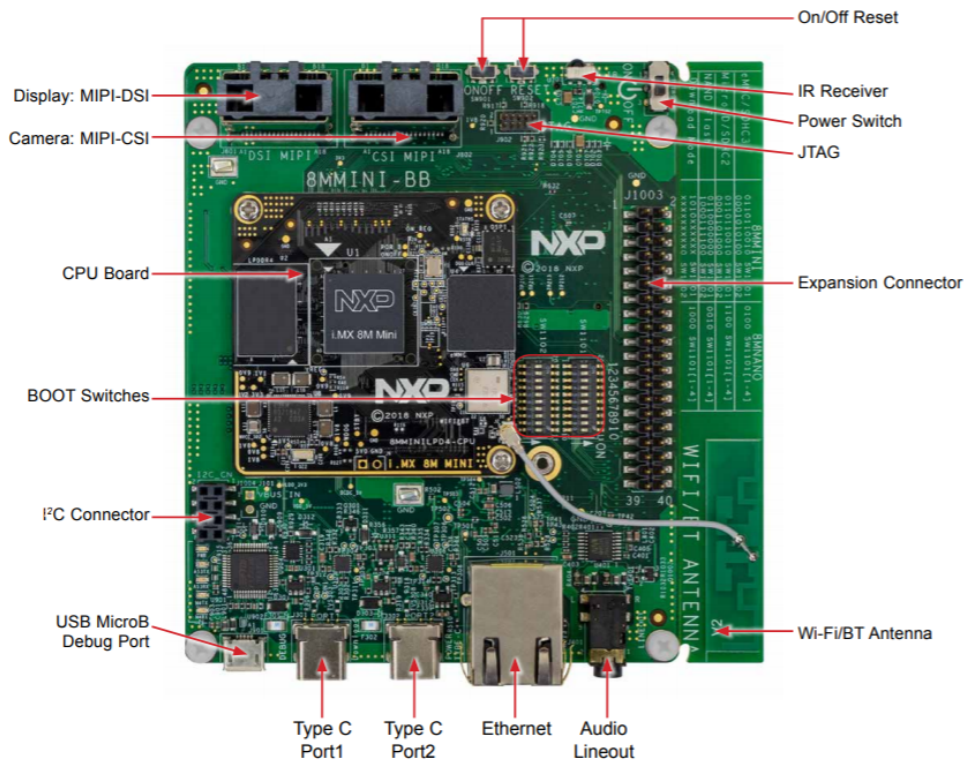


Figure 1. *i.MX 8M Mini EVK Board Top View*

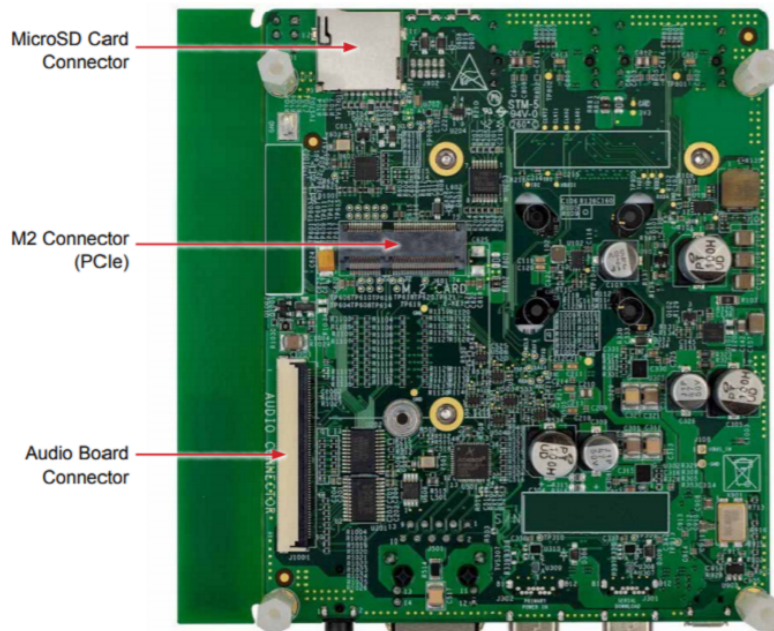


Figure 2. *i.MX 8M Mini EVK Board Back View*

## 1.2. Preparing the i.MX 8M Mini EVK Board

1. Connect the *micro-B* end of the supplied *USB Cable* into *Debug UART* port **J901**.
2. Connect the other end of the cable to the *host GNU/Linux PC*.
3. Insert the *MicroSD Card* into the *MicroSD Card slot J701* in the back board side. To boot the board from the *MicroSD Card*, change the *Boot Switches SW1101* and *SW1102* according to the table below:

Table 1. Boot Device Setting

BOOT Device	SW1101	SW1102
MicroSD/uSDHC2	0110110010	0001101000

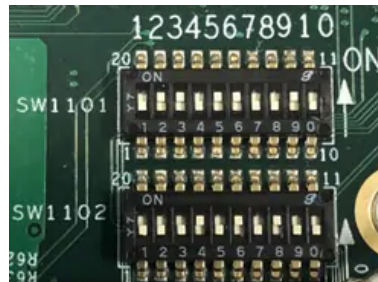


Figure 3. Boot Device Settings

4. Connect the *Power Supply Cable* to the *Power Connector J302* and power **on** the board by flipping the switch **SW101**.



For more details see [i.MX 8M Mini EVK Getting Started](#).

# Chapter 2. Software Requirements

## 2.1. Terminal Emulator



You can use any serial communicator you prefer or has more familiarity.

### 2.1.1. Minicom Configuration (configured)



You need to use root privileges (*sudo*) for steps related to *Minicom* tool.

1. Install the *Minicom* tool:

Host GNU/Linux PC Terminal

```
$ apt-get install minicom ①
```

① using *apt-get* package manager to install *Minicom* tool (step done to avoid spending time).



You need to use root privileges (*sudo*) to use *apt-get* package manager.

2. Go to **serial port setup** and change the settings to:

Host GNU/Linux PC Terminal

```
$ minicom -s ①
```

① configuring the *Minicom* (step done to avoid spending time).

- Serial Device as: */dev/ttyUSB1*
- Bps/Par/Bits as: *115200 8N1*
- Hardware Flow Control: *No*
- Software Flow Control: *No*

3. Then, choose **save setup as dfl**, and **exit**.



The next steps require a bootable image. Go to [Getting Started with NXP BSP Release](#).

### 2.1.2. Minicom Usage

1. Connect the serial cable to the *i.MX Board* and to the *host GNU/Linux PC*, then:

Host GNU/Linux PC Terminal

```
$ minicom ①
```

① open *Minicom* with default settings.

2. Turn the target board power switch to **on**, when it reaches the login prompt, enter: **root**

```
imx8mmevk login: root
Last login: Tue Aug 6 16:53:06 UTC 2019 on tty7
root@imx8mmevk:~#
root@imx8mmevk:~#
```

Figure 4. Prompt log in



Goto [Removing Root Privileges Minicom](#) section if you want to remove the root privileges.

## Chapter 3. Getting Started with NXP BSP Release

The NXP provides the *board support packages (BSP)* with a full and optimized OS, and set of tools to test and maximize the performance of the applications.

### 3.1. NXP Linux BSP Release (4.19.35)



The NXP BSP Releases can be downloaded from [NXP i.MX Software](#).

The BSP for *i.MX 8M Mini EVK Board* is inside the *files* folder located at *~/Desktop/files* directory.

1. Uncompress the image:

*Host GNU/Linux PC Terminal*

```
$ cd ~/Desktop/files
$ unzip L4.19.35_1.1.0_images_MX8MMEVK.zip
```

2. Flash the image into the SD Card device:

*Host GNU/Linux PC Terminal*

```
$ dd if=imx-image-full-imx8mmevk.sdcard of=/dev/sd<x> bs=1M status=progress && sync
```



You need root privileges (sudo) to use `dd` for writing to `/dev/sd<x>`.



`<x>` refers to the SD Card device. You can check it using `lsblk` command.



If you are using the SD Card Reader the device is called `mmcblk<x>`.

3. To boot the board, refers to the [Minicom Usage](#) section.

# Chapter 4. Developing Tools

## 4.1. GNU Cross-Platform Development Toolchain

A *toolchain* is a set of tools that compiles source codes generating executable binary which can run on a specific target device. It includes a *compiler*, *linker*, *run-time libraries* and might also include a *debugger*. The *toolchain* must be able to compile codes written in *Assembly*, *C* and *C++* since these are the most common languages used in the based open source packages.



The described procedures in this document target a *GNU/Linux (Ubuntu 18.04.3 LTS)*.

### 4.1.1. GCC Arm GNU/Linux Toolchain i.MX 32 Bits

1. Install the *GCC*, *G++* cross-compilers and support programs for *i.MX 32 bits*:

*Host GNU/Linux PC Terminal*

```
$ apt-get install gcc-arm-linux-gnueabi g++-arm-linux-gnueabi ①
```

① using *apt-get* to install all the required packages for compiling (**step done to avoid spending time**).

2. To export the cross-compiler, run the following environment variables:

*Host GNU/Linux PC Terminal*

```
$ export ARCH=arm ①
$ export CROSS_COMPILE=/usr/bin/arm-linux-gnueabi- ②
```

① exporting environment variable for defining the platform.

② exporting environment variable for defining the cross-compiler binary.

### 4.1.2. GCC Arm GNU/Linux Toolchain i.MX 64 Bits

1. Install the *GCC*, *G++* cross-compilers and support programs for *i.MX 64 bits*:

*Host GNU/Linux PC Terminal*

```
$ apt-get install gcc-aarch64-linux-gnu g++-aarch64-linux-gnu ①
```

① using *apt-get* to install all the required packages for compiling (**step done to avoid spending time**).

2. To export the cross-compiler, run the following environment variables:

*Host GNU/Linux PC Terminal*

```
$ export ARCH=arm64 ①
$ export CROSS_COMPILE=/usr/bin/aarch64-linux-gnu- ②
```

① exporting environment variable for defining the platform.

② exporting environment variable for defining the cross-compiler binary.



These environment variables are only set for this shell and its children processes. If the shell/terminal is **closed** and starts a new one, these settings **will not be retained**.

## Chapter 5. Getting Started with i.MX Softwares

The NXP provides Open Source softwares for i.MX SoCs, which includes Linux and Android solutions. i.MX applications processors are multicores ARM-based solutions for multimedia and display applications with scalable, high-performance, and low power capabilities.

The NXP provides source codes for *bootloader*, *Kernel*, *Root File System*, and several other software products available externally.



The available NXP Software Releases are published at [codeaurora.org](https://codeaurora.org).

### 5.1. Solid State Storage Media (SD Card)

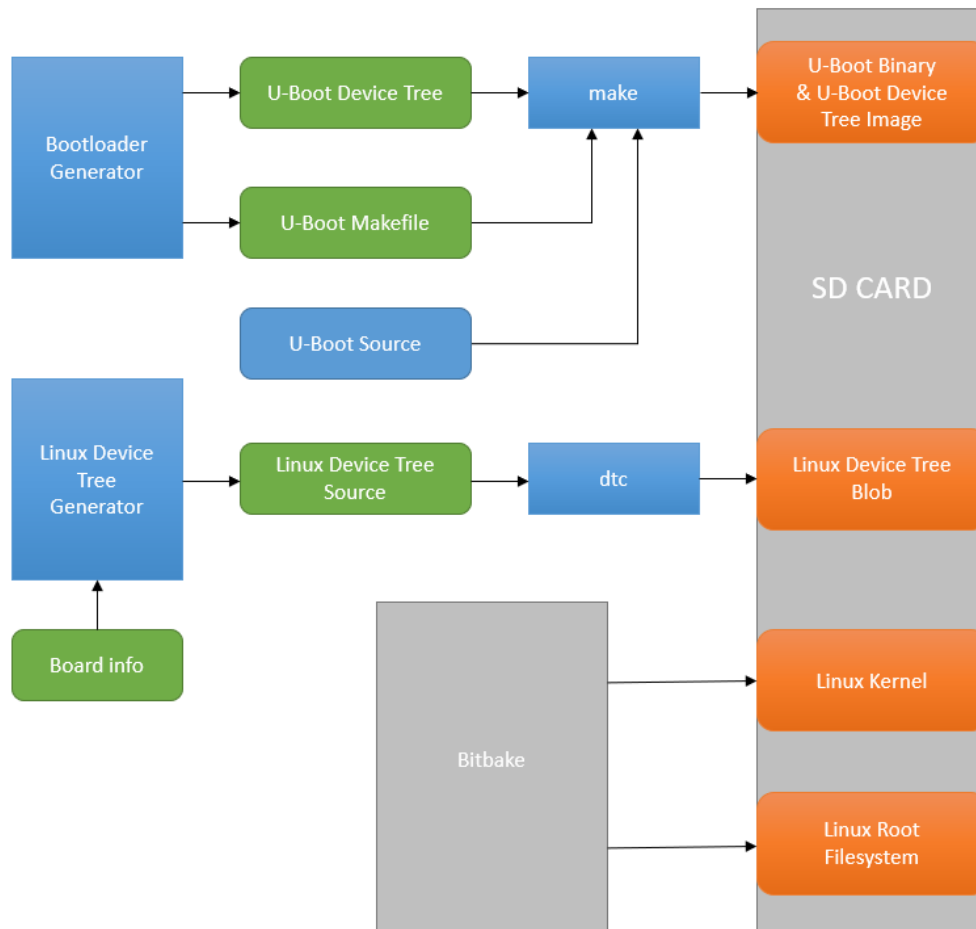


Figure 5. SD Card Layout

## 5.2. Bootloader

### 5.2.1. Internal U-Boot



Follow these steps to compile the internal U-Boot: [README](#).

### 5.2.2. External U-Boot



Follow these steps to compile the mainline U-Boot: [README](#).



## 5.3. Kernel

### 5.3.1. Internal Kernel (4.19.35)

1. Prepare the environment following the 2nd step in [GCC Arm GNU/Linux Toolchain i.MX 64 Bits](#) section.
2. Clone the repository:

*Host GNU/Linux PC Terminal*

```
$ git clone https://source.codeaurora.org/external/imx/linux-imx ①
$ git checkout origin/imx_4.19.35_1.1.0 ②
```

① cloning the repo (**step done to avoid spending time**).

② checking out the 4.19 branch.

3. Compile the Kernel for the i.MX 8M Mini EVK Board 64 bits:

*Host GNU/Linux PC Terminal*

```
$ cd ~/Desktop/files/linux-imx/ ①
$ make defconfig ②
$ make -j<n> ③
```

① entering the folder.

② setting the def config file.

③ compiling the Kernel.



<n> refers to the number of processing units available.



To check the number of processing units available use the `nproc` command.

4. Copy the binaries to the SD Card boot partition:

*Host GNU/Linux PC Terminal*

```
$ cp arch/arm64/boot/Image /media/$USER/<boot_partition>/Image_4.19.35
$ cp arch/arm64/boot/dts/freescale/fsl-imx8mm-evk.dtb /media/$USER/<boot_partition>/4.19.35-imx8mm-evk.dtb
```



Replace the <**boot\_partition**> for the correct name.

5. Stop the *U-Boot* by **pressing any key**. Then, set the following environment variables:

*i.MX Board Minicom Terminal*

```
# setenv image Image_4.19.35 ①
# setenv fdt_file 4.19.35-imx8mm-evk.dtb ②
# saveenv ③
# boot ④
```

① setting the image kernel name.

② setting the device tree binary name.

③ saving the changes.

④ booting the board.

### 5.3.2. External Kernel (5.4.2) TODO SECTION

1. Clone the repository:

*Host GNU/Linux PC Terminal*

```
$ git clone https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git ①
```

① cloning the repo (**step done to avoid spending time**).

#### 2. TODO

- 1. Generate the mainline Kernel binaries.
- 2. Copy these binaries to the SD Card and rename them.
- 3. Change the environment variables.
- 4. Boot the board with the **Kernel 5.4.2**.



The Kernel mainline is located at `~/Desktop/files` folder.

## 5.4. Root File System

To generate the *Root File System* for any *i.MX* board, you have two options:

- Use [Buildroot](#); or
- Use [NXP Linux Yocto BSP](#).

# Chapter 6. Multimedia Support

This section explains how to enable and use different display output and the *OV5640* camera at the *i.MX 8M Mini EVK Board*, and the main difference between the *Device Trees* supported by the *NXP BSP* and the *Kernel mainline*.

## 6.1. Daughter Cards

The *i.MX 8M Mini EVK Board* supports the following daughter cards.

### 6.1.1. MIPI-DSI HDMI display

Using the *IMX-MIPI-HDMI Daughter Card*, the user can connect a *HDMI* monitor by default and reach up to **1080p@60fps** output images on GNU/Linux or Android.



Figure 6. MIPI-DSI Connection Example

### 6.1.2. OLED Display Panel

Other way to reach up to **1080p@60fps** on *i.MX 8M Mini EVK Board* is by using the *OLED display panel*. This accessory can be connected to the *MIPI-DSI* using the **fsl-imx8mm-evk-revb-rm67191.dtb** file.

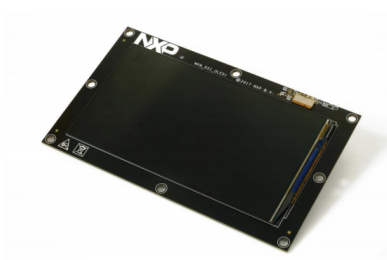


Figure 7. DSI OLED display example

### 6.1.3. MINISASTOCSI Camera Daughter Card

This accessory is a *MIPI-CSI* interface camera kit, based on the **OmniVision OV5640**, which is compatible with the *i.MX 8M Mini EVK Board*. You can enable it by using the default *Device Tree*.

### 6.1.4. Supported Resolutions

- QXGA: 2592x1944@15fps;
- Full HD: 1920x1080@30fps;
- HD: 1280x720@60fps.



Figure 8. MINISASTOCSI Camera Daughter Card

## 6.2. Changing the Device Tree File

The *NXP BSP L4.19.35\_1.0.0* and the *i.MX 8M Mini EVK Board* supports the following *Device Trees* regarding multimedia use cases:

Table 2. *i.MX 8M Mini EVK Board DTB Files Available*

DTB Name	Supported outputs	daughter card	Max Resolution
fsl-imx8mm-evk-revb.dtb	MIPI-DSI	IMX-MIPI-HDMI	1920x1080@60fps
fsl-imx8mm-evk-revb-rm67191.dtb	MIPI-DSI	MIPI-DSI-OLED	1920x1080@60fps
fsl-imx8mm-evk-revb.dtb	MIPI-CSI	MINISASTOCSI	2592x1944@15fps

To change the DTB files on Linux, follow the steps below:

1. Reboot the board and press any key to stop the boot process at U-Boot.
2. Type the desired DTB file following the example command line below:

*i.MX Board Minicom Terminal*

```
# setenv fdt_file fsl-imx8mm-evk-revb.dtb ①
# saveenv ②
# boot ③
```

① setting the fdt\_file.

② saving the variable.

③ booting the board.



Check all these Device Trees in details at this [link](#).

## 6.3. Testing the Display and Cameras

Once connected the display and the camera to the *MIPI-DSI* and *MIPI-CSI* peripherals, boot the board with the correct *Device Tree* and try the following use cases.

### 6.3.1. Video Playback

1. Video playback example:

*i.MX Board Minicom Terminal*

```
# gst-launch-1.0 videotestsrc ! video/x-raw,width=1280,height=720 ! glimagesink
```

2. **TODO**

- 1. Try some tests by changing the **video/x-raw** resolution and the **sink** element, such as the **autovideosink** and **waylandsink window-widht=640 window-height=480**.

### 6.3.2. Camera Playback

1. Camera playback example:

*i.MX Board Minicom Terminal*

```
# gst-launch1.0 v4l2src device=/dev/video0 ! autovideosink
```

2. **TODO**

- 1. Use different camera resolution by changing the **caps** value, i.e., by including the **video/x-raw,width=1920,height=1080,framerate=30/1** property to the *GStreamer* pipeline.
- 2. Use the **glimasink** instead of **autovideosink**.

## 6.4. Changing the Display Resolution/Orientation

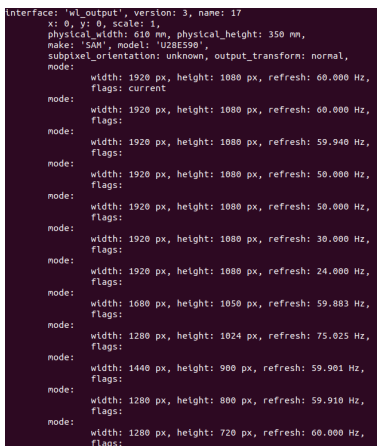
To change the display *resolution* and *orientation*, the first step is to check the available resolutions with the following command:

1. Check the available resolutions:

*i.MX Board Minicom Terminal*

```
# weston-info
```

2. The results will be something like the image below:



```
Interface: 'wl_output', version: 3, name: 17
x: 0, y: 0, scale: 1
physical_width: 610 mm, physical_height: 350 mm,
make: 'SAM', model: 'U28E590',
subpixel_orientation: unknown, output_transform: normal,
mode:
  width: 1920 px, height: 1080 px, refresh: 60.000 Hz,
  flags: current
mode:
  width: 1920 px, height: 1080 px, refresh: 60.000 Hz,
  flags:
mode:
  width: 1920 px, height: 1080 px, refresh: 59.940 Hz,
  flags:
mode:
  width: 1920 px, height: 1080 px, refresh: 50.000 Hz,
  flags:
mode:
  width: 1920 px, height: 1080 px, refresh: 50.000 Hz,
  flags:
mode:
  width: 1920 px, height: 1080 px, refresh: 30.000 Hz,
  flags:
mode:
  width: 1920 px, height: 1080 px, refresh: 24.000 Hz,
  flags:
mode:
  width: 1680 px, height: 1050 px, refresh: 59.883 Hz,
  flags:
mode:
  width: 1280 px, height: 1024 px, refresh: 75.025 Hz,
  flags:
mode:
  width: 1440 px, height: 900 px, refresh: 59.901 Hz,
  flags:
mode:
  width: 1280 px, height: 800 px, refresh: 59.910 Hz,
  flags:
mode:
  width: 1280 px, height: 720 px, refresh: 60.000 Hz,
  flags:
```

Figure 9. Weston-info command example

3. Then, open the *weston.ini* file and change it according:

*i.MX Board Minicom Terminal*

```
# vi /etc/xdg/weston/weston.ini
```

- a. MIPI-TO-HDMI resolution example:

```
[output]
name=HDMI-A-1
mode=1280x720@60
#transform=90
```

- b. OLED display orientation example:

```
[output]
name=DSI-1
mode=720x1280@60
transform=90
```

4. Save the file and reset the board:

*i.MX Board Minicom Terminal*

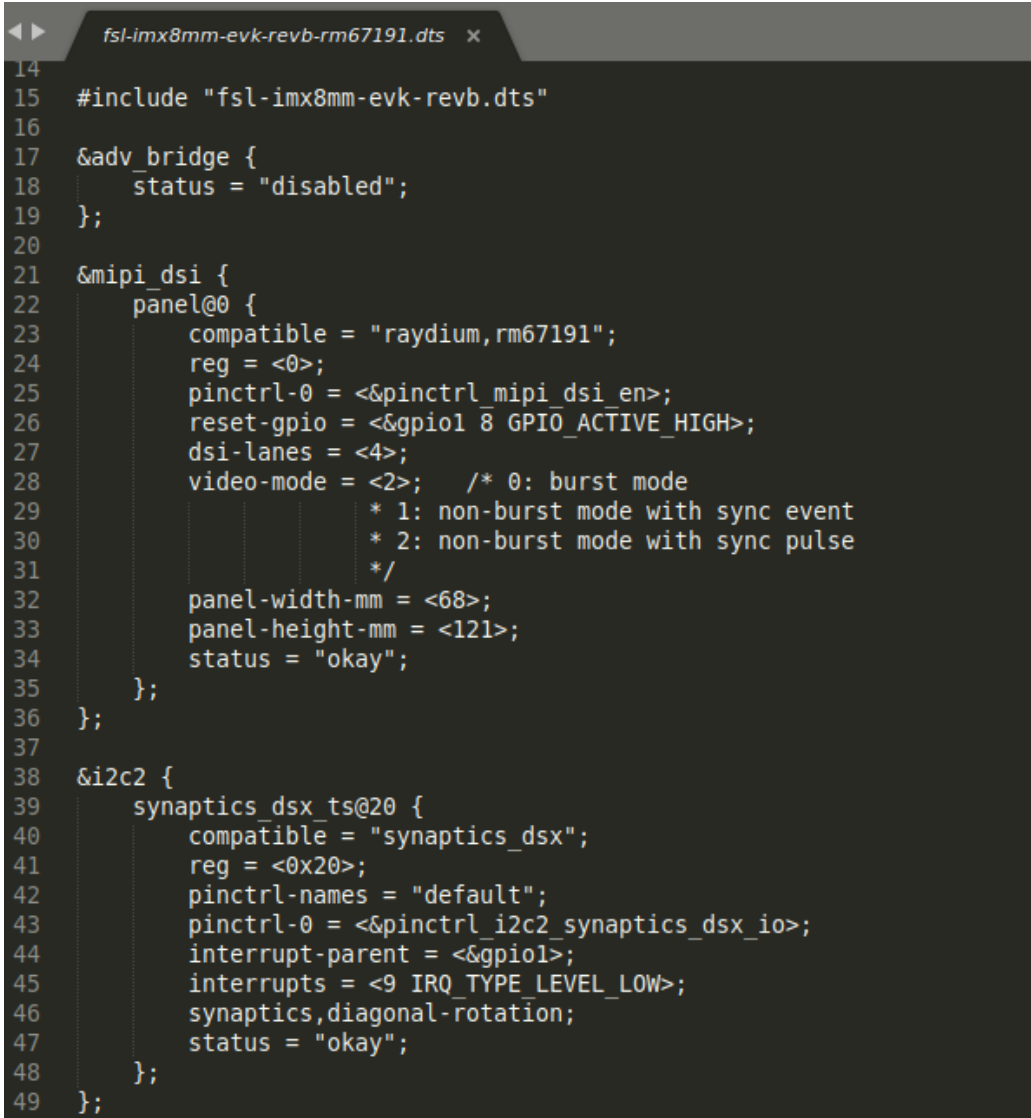
```
# reboot
```

## 6.5. Multimedia Device Tree nodes

This section describes some Device Tree multimedia nodes and the comparison with the NXP BSP and Kernel Mainline.

### 6.5.1. NXP BSP OLED Display Device Tree

You can check the OLED display support at the image below:



```

14
15 #include "fsl-imx8mm-evk-revb.dts"
16
17 &adv_bridge {
18     status = "disabled";
19 };
20
21 &mipi_dsi {
22     panel@0 {
23         compatible = "raydium,rm67191";
24         reg = <0>;
25         pinctrl-0 = <&pinctrl_mipi_dsi_en>;
26         reset-gpio = <&gpio1 8 GPIO_ACTIVE_HIGH>;
27         dsi-lanes = <4>;
28         video-mode = <2>; /* 0: burst mode
29                            * 1: non-burst mode with sync event
30                            * 2: non-burst mode with sync pulse
31                            */
32         panel-width-mm = <68>;
33         panel-height-mm = <121>;
34         status = "okay";
35     };
36 };
37
38 &i2c2 {
39     synaptics_dsx_ts@20 {
40         compatible = "synaptics_dsx";
41         reg = <0x20>;
42         pinctrl-names = "default";
43         pinctrl-0 = <&pinctrl_i2c2_synaptics_dsx_io>;
44         interrupt-parent = <&gpio1>;
45         interrupts = <9 IRQ_TYPE_LEVEL_LOW>;
46         synaptics,diagonal-rotation;
47         status = "okay";
48     };
49 };

```

Figure 10. OLED display Device Tree support

As you can see, we have tree nodes at this [Device Tree](#):

- `adv_bridge`, which has been disabled because we are not using the IMX-MIPI-HDMI daughter card anymore;
- `mipi_dsi`, which is describing the Raydium RM67191 OLED display;
- `i2c2`, which supports the Synaptics DSX touchscreen drive.

### 6.5.2. Kernel Mainline OLED Display Driver Support

In the Kernel mainline, we just have the [Raydium RM67191 Driver Support](#). There is no example of how to enable it on the *Device Tree* but the driver was based on the *NXP BSP driver*, so the *Device Tree* should be very similar to the image above.

```

1 Raydium RM67191 OLED LCD panel with MIPI-DSI protocol
2
3 Required properties:
4 - compatible:      "raydium,rm67191"
5 - reg:             virtual channel for MIPI-DSI protocol
6                   must be <0>
7 - dsi-lanes:       number of DSI lanes to be used
8                   must be <3> or <4>
9 - port:            input port node with endpoint definition as
10                   defined in Documentation/devicetree/bindings/graph.txt;
11                   the input port should be connected to a MIPI-DSI device
12                   driver
13
14 Optional properties:
15 - reset-gpios:     a GPIO spec for the RST_B GPIO pin
16 - v3p3-supply:     phandle to 3.3V regulator that powers the VDD_3V3 pin
17 - v1p8-supply:     phandle to 1.8V regulator that powers the VDD_1V8 pin
18 - width-mm:        see panel-common.txt
19 - height-mm:       see panel-common.txt
20 - video-mode:      0 - burst-mode
21                   1 - non-burst with sync event
22                   2 - non-burst with sync pulse
23
24 Example:
25
26     panel@0 {
27         compatible = "raydium,rm67191";
28         reg = <0>;
29         pinctrl-0 = <&pinctrl_mipi_dsi_0_1_en>;
30         pinctrl-names = "default";
31         reset-gpios = <&gpio1 7 GPIO_ACTIVE_LOW>;
32         dsi-lanes = <4>;
33         width-mm = <68>;
34         height-mm = <121>;
35
36         port {
37             panel_in: endpoint {
38                 remote-endpoint = <&mipi_out>;
39             };
40         };
41     };
42

```

Figure 11. OLED Display Kernel Mainline Driver Support



This was an example of how you can add a support by comparing the *NXP BSP* and *Kernel Mainline*.



# Chapter 7. Android (P9.0.0\_2.0.0\_GA)



The NXP Android BSP Releases can be downloaded from NXP [i.MX Software](#).

The Android BSP for *i.MX 8M Mini EVK Board* is located at `~/Desktop/files directory`.

1. Uncompress the image:

*Host GNU/Linux PC Terminal*

```
$ cd ~/Desktop/files/
$ tar -vzxf android_p9.0.0_2.0.0-ga_image_8mmevk.tar.gz
```

Image name	Download target
u-boot-imx8mm.imx	33 KB offset of MMC for a board with LPDDR4 on it.
u-boot-imx8mm-ddr4.imx	33 KB offset of MMC for a board with DDR4 on it.
u-boot-imx8mm-trusty.imx	33 KB offset of MMC for a board with LPDDR4 on it.
u-boot-imx8mm-evk-uuu.imx	Bootloader used by UUU for the i.MX 8M Mini board with LPDDR4 on it. It is not flashed to MMC.
u-boot-imx8mm-ddr4-evk-uuu.imx	Bootloader used by UUU for the i.MX 8M Mini board with DDR4 on it. It is not flashed to MMC.
imx8mm_mcu_demo.img	5120 KB offset of MMC.
partition-table.img	0 offset of MMC. If the actually size of the SD card is larger than 13 GB, use the default partition-table.img.
partition-table-7GB.img	0 offset of MMC. If the actually size of the SD card is larger than 7 GB, use this image as partition-table.img.
partition-table-28GB.img	0 offset of MMC. If the actually size of the SD card is larger than 28 GB, use this image as partition-table.img.
boot.img	boot_a and boot_b partitions.
vbmeta-imx8mm.img	vbmeta_a and vbmeta_b partitions to support LPDDR4 and MIPI-to-HDMI output and Direct Stream Digital (DSD) playback.
vbmeta-imx8mm-ddr4.img	vbmeta_a and vbmeta_b partitions to support DDR4 and MIPI-to-HDMI output.
vbmeta-imx8mm-m4.img	vbmeta_a and vbmeta_b partitions to support LPDDR4, MIPI-to-HDMI output, and audio playback based on Cortex-M4 FreeRTOS.
vbmeta-imx8mm-mipi-panel.img	vbmeta_a and vbmeta_b partitions to support LPDDR4 and MIPI panel output.
system.img	system_a and system_b partitions.
vendor.img	vendor_a and vendor_b partitions.
dtbo-imx8mm.img	dtbo_a and dtbo_b partitions to support LPDDR4, MIPI-to-HDMI output, and DSD playback.
dtbo-imx8mm-ddr4.img	dtbo_a and dtbo_b partitions to support DDR4 and MIPI-to-HDMI output.
dtbo-imx8mm-m4.img	dtbo_a and dtbo_b partitions to support LPDDR4, MIPI-to-HDMI output, and audio playback based Cortex-M4 FreeRTOS.
dtbo-imx8mm-mipi-panel.img	dtbo_a and dtbo_b partitions to support LPDDR4 and MIPI panel output.

Figure 12. NXP Android BSP pre-images content

The *Android pre-image* supports by default the *MIPI-TO-HDMI Daughter Card* and requires a *16GB SD Card*. However, this section uses the *OLED Display* described at the [OLED Display Panel](#) section instead of the *IMX-MIPI-HDMI*.

The *OLED* has touchscreen support and provides a more intuitive experience with the *Android* image.

1. To support the *OLED* display, rename the following files according:

*Host GNU/Linux PC Terminal*

```
$ cd ~/Desktop/files/android_p9.0.0_2.0.0-ga_image_8mmevk/<android_source>
$ mv vbmeta-imx8mm.img vbmeta-imx8mm-default.img
$ mv vbmeta-imx8mm-mipi-panel.img vbmeta-imx8mm.img
$ mv dtbo-imx8mm.img dtbo-imx8mm-default.img
$ mv dtbo-imx8mm-mipi-panel.img dtbo-imx8mm.img
```



Be sure to have a clean SD Card. Use `dd` command or the `GParted` tool.

2. With all the required files and the SD Card formatted, do the following command:

*Host GNU/Linux PC Terminal*

```
$ ./fsl-sdcard-partition.sh -f imx8mm /dev/sd<x>
```



<x> refers to the SD Card device. You can check it using `lsblk` command.



If you are using an 8GB SD Card, include a `-c 7` after the `-f imx8mm` to flash the SD Card correctly.

For more details on it, check the *Android* documentation at the [NXP Webpage](#).

## Chapter 8. Introduction to M4

The *MCUXpresso Software Development Kit (MCUXpresso SDK)* provides comprehensive software source code to be executed in the *i.MX 8M Mini M4 Core*. This guide shows how to run the *imx8mm\_m4\_TCM\_hello\_world.bin* demo provided by the *NXP BSP L4.19.35\_1.0.0* Release. For detailed information on *MCUXpresso SDK* and how to build and deploy custom demos, please see the [MCUXpresso SDK Webpage](#).

### 8.1. Testing the M4

1. Reboot the board and press any key to stop the boot process at U-Boot.
2. Type the following DTB file in order to enable the M4 core:

*i.MX Board Minicom Terminal USB1*

```
# setenv fdt_file fsl-imx8mm-evk-revb-m4.dtb ①
# saveenv ②
```

① changing the dtb file.

② saving variable.

3. Open a **new terminal** and go to **serial port setup** and change the settings to:

*Host GNU/Linux PC Terminal USB0*

```
$ minicom -s ①
```

① configuring the *Minicom*.

- Serial Device as: */dev/ttyUSB0*
- Bps/Par/Bits as: *115200 8N1*
- Hardware Flow Control: *No*
- Software Flow Control: *No*

4. Then, choose **save setup as dfl**, then **exit** and keep this terminal opened.

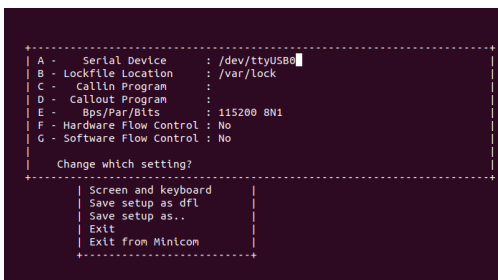


Figure 13. Changing Serial Device at Minicom Console

5. Go back to the terminal with the *U-Boot* parameters (*/dev/ttyUSB1*) and enter with the following command:

*i.MX Board Minicom Terminal*

```
# fatload mmc 0:1 0x7e0000 imx8mm_m4_TCM_hello_world.bin
# bootaux 0x7e0000
```

6. With this command, the **hello world** message should be displayed at the new terminal (*/dev/ttyUSB0*).

## Chapter 9. Appendix

### 9.1. Removing Root Privileges Minicom

1. Removing *root* privileges:

*Host GNU/Linux PC Terminal*

```
$ adduser $USER dialout
$ chgrp dialout /etc/minicom/minirc.dft
$ chmod g+rw /etc/minicom/minirc.dft
```



Be careful using these commands.

### 9.2. Additional Packages

1. Install these additional packages:

*Host GNU/Linux PC Terminal*

```
$ apt-get update
$ apt-get install bison flex make cmake git libssl-dev
```