Lab Guide: i.MX8MM Hands on Lab Guide

Introduction

Introduction, overview, and advanced topics on NXP's i.MX8M Mini Multimedia Processor. This is a two-part class consisting of lecture and hands-on lab sections. Join this session to learn about NXP's latest Industry-Leading Video and Audio family of iMX8M CPUs featuring up to four 1.8 GHz Cortex-A53 processors, advanced peripherals and more. Learn how this innovative CPU enables Advance HMI Solutions supporting Industrial and consumer HMI, Enriched user experience, Immersive Audio and Video processing, Voice Solutions, and Interconnected Devices (smarter edge devices) among other applications.

This class will also provide the attendees the knowledge they need to setup and demonstrate the key aspects of the i.MX8MMini on the EVK.

Prerequisites

Terminal program running on a host computer
USB Mouse
1080 Monitor
A USB memory stick to copy files from your PC to the EVK.
2 or 4 port USB hub (usb2 or better)

To complete this entire Lab series you will need to download and install the following software:

1) Terminal Emulator such as PUTTY: https://www.putty.org/
2) MfgtoolV3 (uuu); this tool will be used for installing Linux and Android onto the boards and can be found https://github.com/NXPmicro/mfgtools/releases Download the latest Released version.
3) Released and pre-built Linux and Android images can be found https://www.nxp.com/support/developer-resources/evaluation-and-development-boards/i.mx-evaluation-and-development-boards/i.mx-software-and-development-tool:IMX-SW. Look for

i.MX 8MMini EVK. The latest Linux image is “Linux 4.14.98_2.0.0” and for android is “P9.0.0 Pie (P9.0.0_2.0.0, 4.14 kernel)”.

4) Android and Linux release Documentation: Download and unzip from https://www.nxp.com/support/developer-resources/run-time-software/i.mx-developer-resources/i.mx-software-and-development-tool:IMX_SW, Scroll down to the Documentation heading.
5) Google Chrome for Android: Download and save for the class: https://www.appsapk.com/chrome-browser
6) DDR Stress Test Tool; Download from the community page (download both the MX8M_Mini_LPDDR4_RPA_EVK_v9.xlsx, and the mscale_ddr_tool_v210_setup.exe.zip: https://community.nxp.com/docs/DOC-340179
7) i.MX8MMini EVK Quick Start Guide is the reference for board setup and use. This can be found in the box (printed copy) or download from this link https://www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/i.mx-applications-processors/i.mx-8-processors/i.mx-8m-mini-arm-cortex-a53-cortex-m4-audio-voice-video:i.MX8MMINI?tab=Documentation_Tab. Scroll to Quick reference guide and download “**i.MX 8M Mini EVK Quick Start Guide.pdf”** file.
8) i.MX8MMini EVK User’s Guide will be referenced for boot mode and boot device switch settings. This can be downloaded from https://www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/i.mx-applications-processors/i.mx-8-processors/i.mx-8m-mini-arm-cortex-a53-cortex-m4-audio-voice-video:i.MX8MMINI?tab=Documentation_Tab. Scroll to Quick reference guide and download “**i.MX 8M Mini EVK Quick Start Guide.pdf”** file.

Lab Agenda

Labs:

8M Mini hands on work shop
Hardware required per workstation

1) 1080P monitor
2) Mipi camera
3) Internet connection (wireless)
4) USB Hub (2 port or better)
5) Usb power meter

Lab exercises:

1) Out of box
    a. Setup
    b. Connecting video display.
    c. Using the MIPI2HDMI adapter connect to the HDMI display
2) DDR RPA & Stress Test
    a. Open and Run RPA

    b. Run Stress Test (with script from RPA above)
3) Imaging the Board
    a. Program SD memory with Linux (using MFGtool V3)
4) Running Linux
    a. Connecting video display.
    b. Using the MIPI2HDMI adapter connect to the HDMI display
    c. Run graphics demos in: /opt/imx-gpu-sdk/GLES2/
    d. Power modes
        i. With USB power meter connected run the various power modes in Linux
    e. Using the Camera - Linux
        i. Simple camera out to the screen
        ii. Video conferencing demo
5) Imaging the Board
    a. Program eMMC memory with Android
6) Running Android (default environment shipped pre-installed)
    a. Connect video display using MIPI2HDMI adapter
    b. Connecting to a camera
    c. Connecting Wifi
    d. Installing Chrome web browser from the USB drive
7) Virtualization
    a. Run Jailhouse

Lab: Initial Setup and operation

This lab will cover the unboxing, initial setup and operation of the EVK.

Required Equipment

MCIMX8MMini-EVK

Monitor with resolution of 1080P or better and HDMI cable

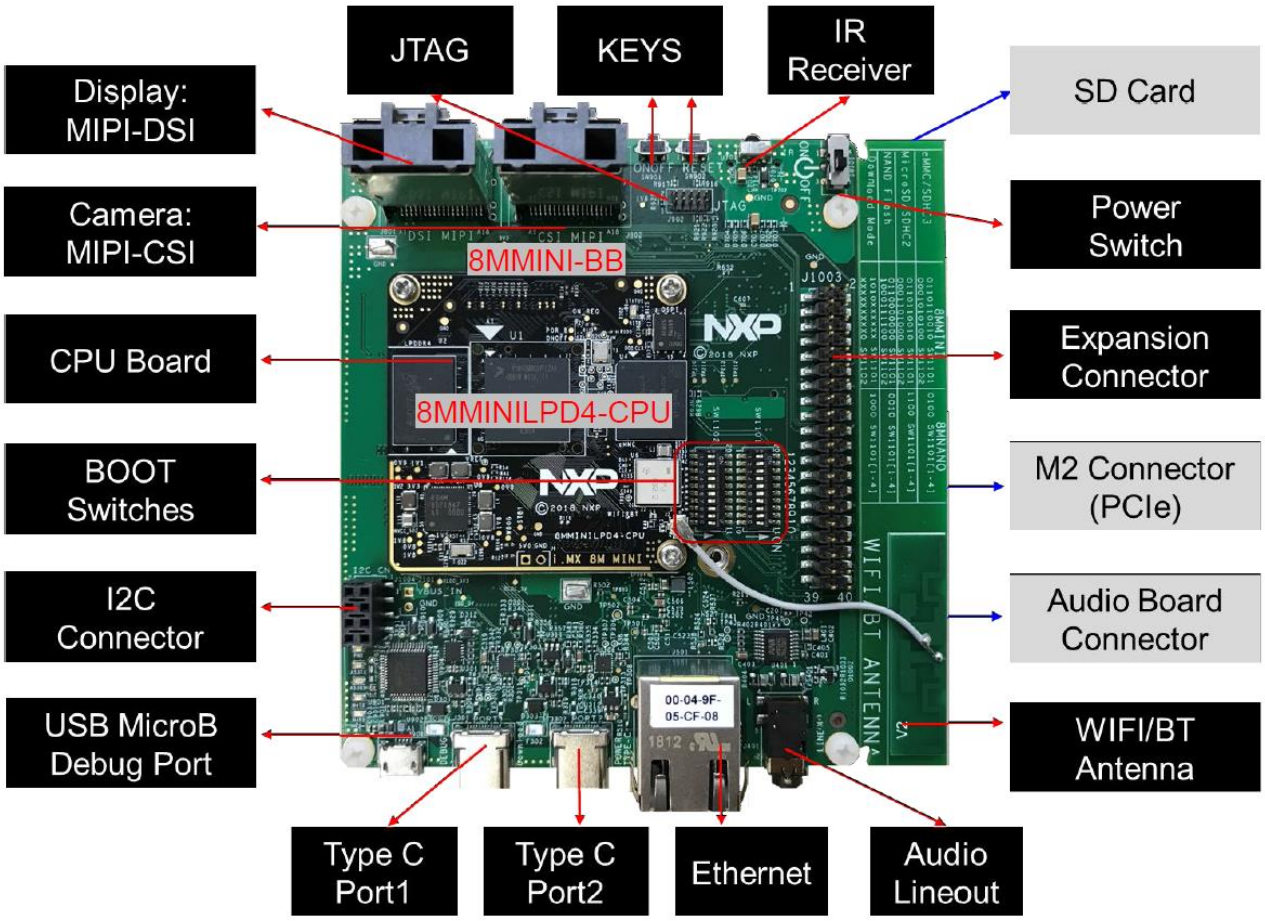
Mouse

2 or 4 port USB hub

PC with a suitable terminal program

Note: Be sure the boot switch settings are to boot from eMMC memory.

Using the included Quick Start Guide (IMX8MMINI EVKQSG) set up the board as instructed.



Connect the following cables:

Power Brick to Type C Port 2

USB cable Type C to PC (Port1 for flashing download)

USB Debug Type A to microB (for Debug Port)

HDMI to Monitor/TV

At this time, verify the boot switch settings to boot the eMMC memory, now the board can be powered on via the power switch, and the system will boot and provide an Android image on the monitor.

Lab: Using the DDR RPA and Stress Test Tools

Download and install the MX8M DDR Tool.

Required Equipment

MCIMX8MM-EVK with Power Supply
USB Cable, type C
USB Cable, type B
Computer with Terminal Program
RPA Spreadsheet
DDR Stress Test

Procedure

This an abbreviated procedure, for a more detailed look at the procedure, see the document, MX8M_DDR_Tool_User_Guide (included with the download of the tool).

Run the RPA Tool

1. Open the RPA: MX8M_Mini_LPDDR4_RPA_v12.xlsx
    a. For the lab computers, this will open OpenOffice.
2. Review the *How To Use* Tab
3. Go to the *Register Configuration* Tab

The steps below are needed to complete the RPA which will generate the datafile for the stress test tool.

Step 1. Obtain the desired DRAM data sheet from the DRAM vendor

Step 2. Update the Device Information table to include the DRAM information and system usage

Device Information	
Memory type:	LPDDR4
Manufacturer:	Micron
Memory part number:	MT53D512M32D2DS-053 WT:D
Density per channel per chip select (Gb)[1]:	8
Number of Channels	2
Number of Chip Selects used[2]	1
Total DRAM density (Gb)	16
Number of ROW Addresses[2]	16
Number of COLUMN Addresses[2]	10
Number of BANK addresses[2]	3
Number of BANKS[2]	8
Bus Width	32
Clock Cycle Freq (MHz)[3]	1500
Clock Cycle Time (ns)	0.666666667
FREQ1 setpoint Clock Cycle Freq (MHz)	200
FREQ1 Clock Cycle Time (ns)	5
FREQ2 setpoint Clock Cycle Freq (MHz)	50
FREQ2 Clock Cycle Time (ns)	20

Step 3. Go through the various shaded cells in the spread sheet to update with data from the DRAM sheet (take special note of the “Legend” table to ascertain the meaning of different shaded cells; in many cases, the cells may not need to be updated).

Step 4. Go to the BoardDataBusConfig tab and correctly fill out the MX8 data bus mapping to the memory device. The user should take special care to ensure this worksheet is configured correctly or else the LPDDR4 system may not work properly.

Note: changes to the Register Configuration and BoardDataBusConfig worksheets are automatically updated in the DDR stress test file worksheet tab described next.

Step 5. The final worksheet tab "DDR stress test file" is the output of the RPA and represents the DRAM initialization for use with the DDR Stress Test. To create a DDR Stress Test script, the user must copy the contents in this worksheet tab and paste it to a text document**, naming the document with the “.ds” file extension**. The user will later select this file when executing the DDR stress test.
It is important that the user must make sure to copy all of the contents from the DDR stress test file worksheet tab. One recommended method to ensure that all of the contents are selected before copying is to click on the arrow in the upper left-hand corner of this sheet between row 1 and column A as shown below.

Now you are ready to run the DDR Stress Test

1) Connect USB C to PC cable to Port #1.
2) Set the boot switches to Serial Download mode

Serial Download

Boot Switches for Serial Download Mode

SW1101 (1-10)	1	0	1	0	X	X	X	X	X	X
SW1102 (1-10)	X	X	X	X	X	X	X	X	X	X

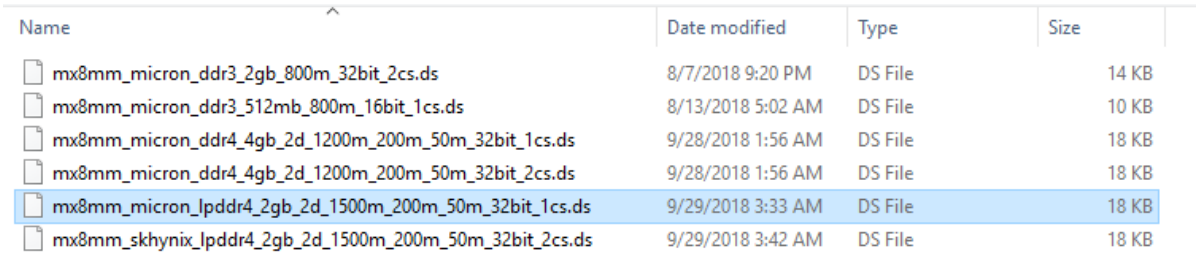
.

3) Power on the board.
4) Run the MX8M-DDR_Tool.exe in **ADMINISTRATOR** mode. (right click on the program and select "Run as administrator")
5) Open Debug UART Note: Be sure to disconnect serial program as used above)
    a. Click Search
    b. Using the drop-down menu select the proper COM port. (use the device manager to determine which ports are assigned to the EVK, then select the highest number of the 2).
    c. Click "Connect"
6) Open the RPA: MX8M_Mini_LPDDR4_RPA_v12.xlsx
    a. For the lab computers, this will open OpenOffice.
7) Load the DDR Script.
    a. Click on "Load DDR Script"
    b. Select the proper script. For the EVK select

    Use the file that you generated from above.

    OR use the preconfigured file below

    "mx8mm_micron_lpddr4_2gb_2d_1500mhz_32bits.ds"



    c. Select the following
        d) Target "MX8M-mini"

e) Clock "Default"

f) DDR "LPDDR4"

g) Density "Default"

d. Click on "Download"

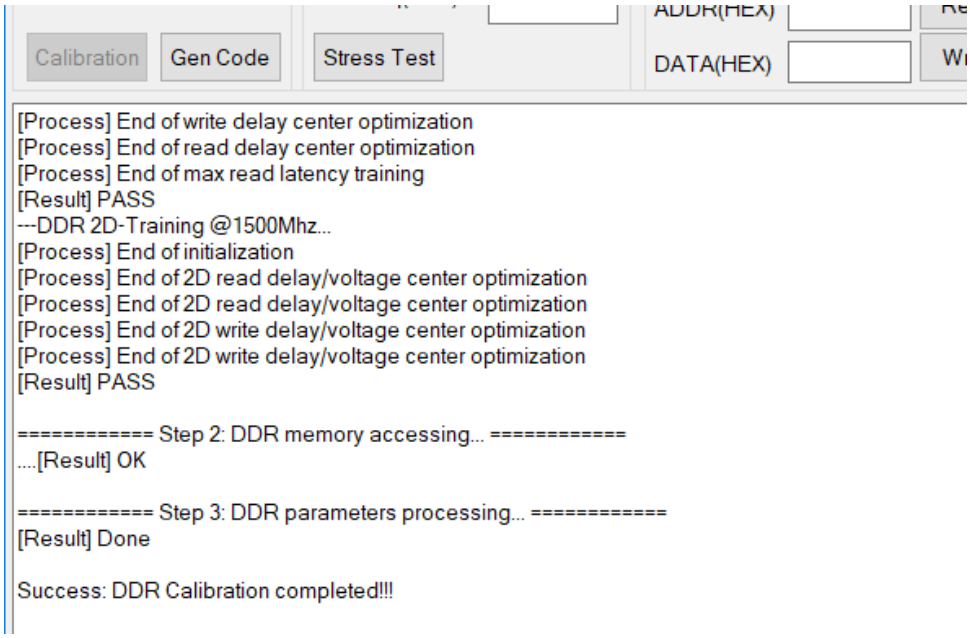
This loads the scripts and prepares to run the Calibration. These scripts are generated from the Register Programming Aid (RPA).

```
************************************************************************
        ARM clock(CA53) rate: 300MHz
        DDR Clock: 750MHz

========================================
     DDR configuration
DDR type is LPDDR4
Data width: 32, bank num: 8
Row size: 16, col size: 10
One chip select is used
Number of DDR controllers used on the SoC: 1
Density per chip select:  2048MB
Density per controller is: 2048MB
Total density detected on the board is: 2048MB
========================================

MX8M-mini: Cortex-A53 is found

************************************************************************
```

8) Run the calibration by Clicking on the "Calibration" button.

a. With Calibration completed (successfully) move onto the next phase

b.



With DDR Calibration completed, you can now run the Stress Test.

9) Now to run the "Stress Test" by clicking on the "Stress Test Button.

For more detailed information on the DDR Stress test, see the DDR documentation *MX8M_DDR_ToolUser_Guide.docx*.

Lab: Imaging the Board

In this lab, the participant will image the board's eMMC with the latest Android and Linux images using the new MFGTool V3.

Required Equipment

MCIMX8MMini-EVK
USB Cable; type C
USB Cable, type B
SD Card 4GB minimum (class 10)
Computer with Terminal Program
Imaging software MFGToolV3 (uuu)

Loading Software

For the most up to date software images, see Table 3 above.

This series of labs, you will program the eMMC with Android and SD card with Linux.

Programming the eMMC memory with Android

The EVK has Android pre-installed on the eMMC memory by default. These instructions are included in the event you need to replace or update the image. The installed Android system uses multiple partitions, the files provided in the download have separate image (img) files for each partition. Also in the files are partition table img files. There are three partition files:

Partition File Name	Supports memory size
partition-table-28GB.img	32GB memory or larger
partition-table-7GB.img	8GB memory or larger
partition-table.img	16GB memory or larger (default file)

In order to image a memory size that is different than the 16GB default, save the original partition-table.img file (rename to partition-table-16GB.img) then rename the desired partition table size file to partition-table.img. This is the file name used in the programming scripts.

1) Download the Android file (see Table 3 above) to a subdirectory. For this instance we will assume the download file is android_p9.0.0_2.0.0-ga_image_8mmevk.
2) Unpack the android_p9.0.0_2.0.0-ga_image_8mmevk .tar.gz
3) Unpack the android p9.0.0_2.0.0-ga_image_8mmevk.tar

4) Copy the latest version of uuu.exe for windows (uuu for linux) into the same subdirectory that you have the Android images.
5) Set the boot switches to Serial Download mode. SW 1101 (sw 1, 2, 3, & 4)

Serial Download

Boot Switches for Serial Download Mode

SW1101 (1-10)	1	0	1	0	X	X	X	X	X	X
SW1102 (1-10)	X	X	X	X	X	X	X	X	X	X

6) Open an **administrator** command prompt window to the subdirectory that the zip files were unpacked in.
   a. Type `uuu_imx_android_flash.bat -f imx8mm -a -e` <RETURN>
   b. The eMMC is now being programmed.

```
This script is validated with uuu 1.2.91 version, please align with this version.
dtbo is supported
dual slot is supported
generate lines to flash partition-table.img to the partition of gpt
generate lines to flash u-boot-imx8mm.imx to the partition of bootloader0
generate lines to flash dtbo-imx8mm.img to the partition of dtbo_a
generate lines to flash boot.img to the partition of boot_a
generate lines to flash system.img to the partition of system_a
generate lines to flash vendor.img to the partition of vendor_a
generate lines to flash vbmeta-imx8mm.img to the partition of vbmeta_a
uuu script generated, start to invoke uuu with the generated uuu script
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.2.91-0-g3799f4d

Success 1    Failure 0

1:2   21/21   [Done                          ] FB: done
```

7) Power off the system.
8) Reset the boot switches to boot the eMMC device
9) Power on the system.
10) Android is now up and running.

Programming the SD Card memory with Linux

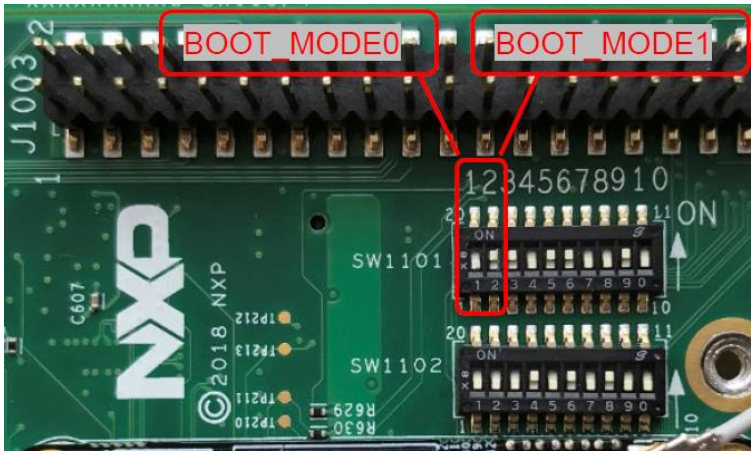
Linux SD Card image

1) Download the Linux file to a subdirectory. For this instance we will assume the download file is L4.14.98_2.0.0_ga_images_MX8MMEVK.zip
2) Copy the latest version of uuu.exe for windows (uuu for linux) into the same subdirectory that you have the Linux images.
3) Set the boot switches to Serial Download mode. SW 1101 (sw 1, 2, 3, and 4)

Serial Download

Boot Switches for Serial Download Mode

SW1101 (1-10)	1	0	1	0	X	X	X	X	X	X
SW1102 (1-10)	X	X	X	X	X	X	X	X	X	X



4) Plug a USB type C cable from your host computer to the EVK on Port 1 (Download).
5) Open a command prompt window to the subdirectory that the zip files were unpacked in.
    a. Modify the uuu.auto text file (not required here, the modified file has been provided for you, its uuu.auto.sd and path for the same is E:\Lab-Imaging_the_EVK\Linux)
        i. Change line 12
            1. from: FB: ucmd setenv mmcdev ${emmc_dev}
            2. to: FB: ucmd setenv mmcdev ${sd_dev}
        ii. Change line 13
            1. From: FB: ucmd mmc dev ${emmc_dev}
            2. To: FB: ucmd mmc dev ${sd_dev}
        iii. Remove line 16.
            1. FB: ucmd mmc partconf ${emmc_dev} 0 1 0
        iv. Save the file as uuu.auto.sd
    b. Type uuu uuu.auto.sd <RETURN>

    The program indicates "Wait for Known USB Device Appear"

```
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.1.81-0-ge39adc4

uuu [-d -m -v -V] <bootloader|cmdlists|cmd>

    bootloader  download bootloader to board by usb
    cmdlist     run all commands in cmdlist file
                If it is path, search uuu.auto in dir
                If it is zip, search uuu.auto in zip
    cmd         Run one command, use -H see detail
                example: SDPS: boot -f flash.bin
    -d          Deamon mode, wait for forever.
    -v -V       verbose mode, -V enable libusb error\warning info
    -m          USBPATH Only monitor these pathes.
                    -m 1:2 -m 1:3

uuu -s          Enter shell mode. uuu.inputlog record all input commands
                you can use "uuu uuu.inputlog" next time to run all commands

uuu -h -H       show help, -H means detail helps

uuu [-d -m -v] -b[run] <emmc|emmc_all|qspi|sd|sd_all|spl> arg...
        Run Built-in scripts
        emmc    burn boot loader to eMMC boot partition
                arg0: _flash.bin
        emmc_all        burn whole image to eMMC
                arg0: _flash.bin
                arg1: _rootfs.sdcard
        qspi    burn boot loader to qspi nor flash
                arg0: _flexspi.bin  bootloader
                arg1: _image[Optional]  image burn to flexspi, default is the same as bootloader
        sd      burn boot loader to sd card
                arg0: _flash.bin
        sd_all  burn whole image to sd card
                arg0: _flash.bin
                arg1: _rootfs.sdcard
        spl     boot spl and uboot
                arg0: _flash.bin

uuu -bshow <emmc|emmc_all|qspi|sd|sd_all|spl>
        Show built-in script

Wait for Known USB Device Appear
```

Note: uuu.exe is a command line program. Just double clicking on it will not work

6) Power on the board.

   a. This process will take a few minutes. The status is indicated on the host PC.

```
Success 0    Failure 0

1:2    4/ 7   [=>          10%            ] FB: flash -raw2sparse all fsl-image-validation-imx-imx8mmevk.sdcard
```

   b. When the programming of the memory is completed, the program will indicate "Done".

```
Success 1    Failure 0

1:2    7/ 7   [Done                        ] FB: done

C:\Users\nxa17243\OneDrive - NXP\Training\Training_2018\DFAE_Oct2018\L4.9.123-ga-8mmevk>
```

7) Power Off the board

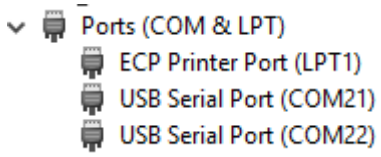
8) Reset the switches to boot from the SD card

Boot Device	SW1101	SW1102
MicroSD/SDHC2	0110110010	0001101000

9) Start your favorite terminal program and connect to the appropriate com port.

   a. In Linux /dev/ttyUSB1

b. For Windows, check the device manager for the USB Serial Port. The A53 debug port will be the highest of the two numbers. In this case, it will be COM22. The M4 debug port will be enumerated as the lower number.



10) Connect the monitor (if not already connected).

11) Power on the board using SW101.

a. You will see many messages cross the console and finally land on a prompt. The log in is "root" with no password.

b. The Wayland desktop will be showing on the screen.

Congratulations you have completed this lab!!

Lab: Virtualization and Jailhouse

This lab you will reprogram the eMMC with the Jailhouse version of Linux, then run the lab.

Reference: i.MX Virtualization User's Guide, which can be found here. The image file can be found here. https://www.nxp.com/pages/alpha-beta-bsps-for-microprocessors:IMXPRERELEASES

Required Equipment

MCIMX8MMini-EVK
USB Cable; type C
USB Cable, type B
Computer with Terminal Program
Pre-Imaged SD Card

Note: This lab requires a bootable Linux OS on both the eMMC and the SD card. This lab is written up to have the Jailhouse image on the SD card.

Using Jailhouse with Prebuilt image

Prebuilt images are only provided for i.MX 8MMini and 8MQuad.

There are two required drivers that need to be loaded, these are jailhouse.ko and uio_ivshmem.ko files which can be found under: /lib/modules/<linux_kernel_version>/extra/driver

The cells and inmates are under: /usr/share/jailhouse.

The lab process:

1) Set the board to boot from the SD card – imaged with the virtualization image that was downloaded from the i.MX Software and Development Tool site.

2) Boot the board into uboot. Stop the boot process in u-boot.

3) In U-Boot, type `run jh_mmcboot`. It loads dedicated DTB for Jailhouse usage, then boots into linux

4) Login as `root`

   Note: For easier reading, and to extend the automatic word wrap on the command line, enter:
   ```
   # stty rows 45 cols 135 && bash
   ```

5) As a baseline let us check the cpu count by entering:

   ```
   # cat /proc/cpuinfo
   ```

```
root@imx8mmevk:~# cat /proc/cpuinfo
processor       : 0
BogoMIPS        : 16.00
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 1
BogoMIPS        : 16.00
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 2
BogoMIPS        : 16.00
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 3
BogoMIPS        : 16.00
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4
```

   Notice there are 4 cores reported.

6) Now load the drivers:

```
# insmod /lib/modules/`uname -r`/extra/driver/jailhouse.ko
# insmod /lib/modules/`uname -r`/extra/driver/uio_ivshmem.ko
```

7) Now its time to load up the jail, cells, and inmates.

```
# jailhouse enable /usr/share/jailhouse/cells/imx8mm.cell
```

```
root@imx8mmevk:~# jailhouse enable /usr/share/jailhouse/cells/imx8mm.cell

Initializing Jailhouse hypervisor v0.10 (66-g2217029) on CPU 1
Code location: 0x0000ffffc0200800
Page pool usage after early setup: mem 39/994, remap 0/131072
Initializing processors:
 CPU 1... OK
 CPU 3... OK
 CPU 2... OK
 CPU 0... OK
Initializing unit: irqchip
Initializing unit: ARM SMMU
No SMMU
Initializing unit: PCI
Adding virtual PCI device 00:00.0 to cell "imx8mm"
iommu_config_commit imx8mm
Page pool usage after late setup: mem 58/994, remap 144/131072
Activating hypervisor
[ 1090.216683] OF: PCI: host bridge /pci@0 ranges:
[ 1090.221294] OF: PCI:   MEM 0xbb900000..0xbb901fff -> 0xbb900000
[ 1090.227725] pci-host-generic bb800000.pci: ECAM at [mem 0xbb800000-0xbb8fffff] for [bus 00]
[ 1090.236286] pci-host-generic bb800000.pci: PCI host bridge to bus 0000:00
[ 1090.243112] pci_bus 0000:00: root bus resource [bus 00]
[ 1090.248376] pci_bus 0000:00: root bus resource [mem 0xbb900000-0xbb901fff]
[ 1090.255556] pci 0000:00:00.0: BAR 0: assigned [mem 0xbb900000-0xbb9000ff 64bit]
[ 1090.263283] virtio-pci 0000:00:00.0: enabling device (0000 -> 0002)
[ 1090.269949] uio_ivshmem 0000:00:00.0: using jailhouse mode
[ 1090.275912] The Jailhouse is opening.
root@imx8mmevk:~#
```

```
# jailhouse cell create /usr/share/jailhouse/cells/imx8mm-ivshmem-demo.cell
```

```
root@imx8mmevk:~# jailhouse enable /usr/share/jailhouse/cells/imx8mm.cell
JAILHOUSE_ENABLE: Device or resource busy
root@imx8mmevk:~# jailhouse cell create /usr/share/jailhouse/cells/imx8mm-ivshmem-demo.cell
[ 1486.284452] CPU3: shutdown
[ 1486.287170] psci: CPU3 killed.
Adding virtual PCI device 00:00.0 to cell "ivshmem-demo"
Shared memory connection established: "ivshmem-demo" <--> "imx8mm"
iommu_config_commit ivshmem-demo
Created cell "ivshmem-demo"
Page pool usage after cell creation: mem 71/994, remap 144/131072
[ 1486.315331] Created Jailhouse cell "ivshmem-demo"
root@imx8mmevk:~#
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyUSB1
```

```
# jailhouse cell load 1 /usr/share/jailhouse/inmates/ivshmem-demo.bin
```

```
root@imx8mmevk:~# jailhouse cell load 1 /usr/share/jailhouse/inmates/ivshmem-demo.bin
Cell "ivshmem-demo" can be loaded
root@imx8mmevk:~#
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyUSB1
```

```
# jailhouse cell start 1
```

```
root@imx8mmevk:~# jailhouse cell load 1 /usr/share/jailhouse/inmates/ivshmem-demo.bin
Cell "ivshmem-demo" can be loaded
root@imx8mmevk:~# jailhouse cell start 1
Started cell "ivshmem-demo"
IVSHMEM: Found 1af4:1110 at 00:00.0
IVSHMEM: shmem is at 0x00000000bba00000
IVSHMEM: bar0 is at 0x00000000bbc00000
IVSHMEM: mapped shmem and bars, got position 1
IVSHMEM: Hello from bare-metal ivshmem-demo inmate!!!
IVSHMEM: Enabled IRQ:0x6c
IVSHMEM: Enabling IVSHMEM_IRQs
IVSHMEM: Done setting up...
IVSHMEM: waiting for interrupt.
root@imx8mmevk:~#
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyUSB1
```

At this point the virtual machine is running. To check this out, check the number of cores available to the system by typing # cat /proc/cpuinfo

```
root@imx8mmevk:~# cat /proc/cpuinfo
processor       : 0
BogoMIPS        : 16.00
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 1
BogoMIPS        : 16.00
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 2
BogoMIPS        : 16.00
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

root@imx8mmevk:~#
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyUSB1
```

And now there are only 3 cores reported. Jailhouse has 1 inmate using 1 core.

8) Check interrupts by entering

   # cat /proc/interrupts | grep uio

   You should be able to see the interrupts triggered once.

9) This concludes this exercise, not enter:

   # jailhouse cell destroy 1

```
root@imx8mmevk:~# jailhouse cell destroy 1
Closing cell "ivshmem-demo"
iommu_config_commit ivshmem-demo
Page pool usage after cell destruction: mem 59/994, remap 144/131072
[ 4411.976477] Detected VIPT I-cache on CPU3
[ 4411.976504] GICv3: CPU3: found redistributor 3 region 0:0x00000000388e0000
[ 4411.976542] CPU3: Booted secondary processor [410fd034]
[ 4412.008474] Destroyed Jailhouse cell "ivshmem-demo"
root@imx8mmevk:~#
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyUSB1
```

10) Now recheck # cat /proc/cpuinfo

```
root@imx8mmevk:~# cat /proc/cpuinfo
processor       : 0
BogoMIPS        : 16.00
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 1
BogoMIPS        : 16.00
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 2
BogoMIPS        : 16.00
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 3
BogoMIPS        : 16.00
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

root@imx8mmevk:~#
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyUSB1
```

And we are back to 4 cores.

Running Linux in a Jailhouse Cell

This porting of the lab, will have 2 instantiations of Linux running. The host OS Linux will have the its standard console port and the inmate Linux will use the console that is typically used for the M4.

Be sure to have both consoles open. In the example case below, the ports will be on ttyUSB1 (host Linux) and ttyUSB0 (Inmate Linux).

If you are continuing on from the last exercise, then the is no need to load the drivers, but included for completion sake.

Load the Drivers, Cells, Tools, & Inmate

```
# insmod /lib/modules/`uname -r`/extra/driver/jailhouse.ko
# insmod /lib/modules/`uname -r`/extra/driver/uio_ivshmem.ko
# stty rows 45 cols 135 && bash
```

Load the jail cells, and set the $PATH to some needed tools

```
# jailhouse enable /usr/share/jailhouse/cells/imx8mm-veth.cell
```

```
# export PATH=$PATH:/usr/share/jailhouse/tools/
```

```
jailhouse cell linux /usr/share/jailhouse/cells/imx8mm-linux-demo.cell /run/media/mmcblk1p1/Image \
-d /run/media/mmcblk1p1/fsl-imx8mm-evk-inmate.dtb -c "clk_ignore_unused console=ttymxc3,115200 \
earlycon=ec_imx6q,0x30890000,115200 root=/dev/mmcblk2p2 rootwait rw"
```

> Note: The current setup for the Linux inmate requires a Linux root fs to be located on the eMMC partition.

The inmate OS will have its console output on the console usually set for the M4. At the login prompt, sign in as root.

Now that the inmate is running, you can check this be entering on the host OS

> # jailhouse cell list

Lab: U

```
root@imx8mmevk:~# jailhouse cell list
ID      Name                    State             Assigned CPUs          Failed CPUs
0       imx8mm                  running           0-1
1       linux-inmate-demo       running           2-3
root@imx8mmevk:~#
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyUSB1
```

On the host OS, enter the following:

```
# Ifconfig eth1 192.168.200.10
```

On the jailhouse OS, enter the following:

```
# ifconfig eth0 192.168.200.10
```

Ping from the inmate OS to the host OS:

```
# ping 192.168.200.10
```

Quit the ping by entering CNTRL-C

```
root@JH_imx8mmevk:~# ping 192.168.200.10
PING 192.168.200.10 (192.168.200.10) 56(84) bytes of data.
64 bytes from 192.168.200.10: icmp_seq=1 ttl=64 time=0.166 ms
64 bytes from 192.168.200.10: icmp_seq=2 ttl=64 time=0.110 ms
64 bytes from 192.168.200.10: icmp_seq=3 ttl=64 time=0.138 ms
^C
--- 192.168.200.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2041ms
rtt min/avg/max/mdev = 0.110/0.138/0.166/0.022 ms
root@JH_imx8mmevk:~#
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyUSB0
```

On the host OS, ping the inmate OS

```
# ping 192.168.200.20
```

```
root@imx8mmevk:~# ping 192.168.200.20
PING 192.168.200.20 (192.168.200.20) 56(84) bytes of data.
64 bytes from 192.168.200.20: icmp_seq=1 ttl=64 time=0.179 ms
64 bytes from 192.168.200.20: icmp_seq=2 ttl=64 time=0.123 ms
64 bytes from 192.168.200.20: icmp_seq=3 ttl=64 time=0.117 ms
64 bytes from 192.168.200.20: icmp_seq=4 ttl=64 time=0.098 ms
^C
--- 192.168.200.20 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3072ms
rtt min/avg/max/mdev = 0.098/0.129/0.179/0.031 ms
root@imx8mmevk:~#
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyUSB1
```

Go ahead and experiment with ssh from one to the other.

```
# ssh root@192.168.200.xx
```
//fill in the appropriate xx with 10 or 20 depending on which OS you are typing on.

Experiment with a shell open from one OS to the other OS.

This concludes this lab.

Lab Running Graphics

This lab will provide the information so one could demonstrate the graphics demos that are included in the NXP Linux load.

Note: The board comes pre-programmed with Android, so Linux must be loaded on the board prior to running this lab. See Imaging the Board lab.

Required Equipment

MCIMX8MMini-EVK
MIPI to HDMI adapter
Monitor connected via HDMI
USB Cable, type C to type B
Computer with Terminal Program

Procedure

1) Boot the EVK into Linux
2) To run the **3D Demo**:
    a. Login using root as user with no password
    b. Type on the following commands on the console:
    c. cd /opt/imx-gpu-sdk/GLES2/ <RETURN>
    d. ./ModelViewer/ModelViewer_Wayland <RETURN>
    e. To exit the demo, use CTRL + C

3) For another image, type the following
    a. ./S08_EnvironmentMappingRefraction/S08_EnvironmentMappingRefraction_Wayland <RETURN>

    Other demos can be found in the directory: /opt/imx-gpu-sdk/GLES3, use the same syntax as above.

Benchmark Tests:
    /usr/bin/glmark2-es2-wayland

Lab: Power Measurements

With this lab, we will run the board in 2 different modes and look at the current draw. This lab will use the graphics demo as the power consumer.

Required Equipment

MCIMX8MMini-EVK with Power Supply
Monitor connected via HDMI
USB Cable, type C to type A
USB type C Power Meter
Computer with Terminal Program

1) Add the USB-C Power Meter between the Power and the board
2) Boot the EVK into Linux
3) Run the 3D Demo
    a. Enter the following command

i. Login
ii. cd /opt/imx-gpu-sdk/GLES2/ <RETURN>
iii. ./S08_EnvironmentMappingRefraction/S08_EnvironmentMappingRefraction_Wayland & <RETURN>

4) Check and note the power.
5) Put the system into a Power Save mode
    a. Enter the following command
        i. echo mem > /sys/power/state
6) Check and note the power.
7) Press the On/Off Button (the one near MIPI CSI) to bring the system back to operational mode.

At this time you should have the graphics image running again. Note the power again. Also note the messages. The resume time takes approximately 40 milliseconds.

To resume normal operation, either kill the 3D image or reboot.

This concludes this Lab.

Lab: Connecting Wifl (Linux)

The i.MX8MMini EVK has wifi included. This lab will guide you through the steps in setting up the wifi via the command line on linux. (This is adapted from the i.MX_Linux_Reference_Manual under connectivity) The commands listed below work well for the first time connections)

Required Equipment

MCIMX8MMini-EVK with Power Supply
Monitor connected via HDMI
USB Cable, type A to type Micro
Computer with Terminal Program

1) Connect the terminal program and power up the board with Linux on the boot device (if not already running linux).
2) Log in at the prompt as "root", there is no password.
3) Enter the following commands
    a. `wpa_passphrase ***ssid passcode*** >> /etc/wpa_supplicant.conf`
    fill in the ssid and password for the desired wifi access point that you want to connect to. For example: SSID=classroom_5 and PASSCODE=nxp_2018.
    b. `wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant.conf -D nl80211`
    c. `udhcpc -i wlan0`
    This line requests the IP address from the dhcp server.

You are now connected to the network.
An inspection of the file "/etc/wpa_supplicant.conf shows:

```
network={
      ssid="classroom_5"
      #psk="nxp_2018"
      psk=8e9bbd5aeec2cb42e5ff6c83a355843fcf9257ea9d1dd2c35628389722051f06
}
```

And now take a look at ifconfig:

```
root@imx8mmevk:/etc# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:04:9f:05:9e:ee
          UP BROADCAST MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:722 errors:0 dropped:0 overruns:0 frame:0
          TX packets:722 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:52300 (51.0 KiB)  TX bytes:52300 (51.0 KiB)

wlan0     Link encap:Ethernet  HWaddr a0:c9:a0:5e:1d:d9
          inet addr:192.168.100.109  Bcast:192.168.100.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13854 errors:0 dropped:39 overruns:0 frame:0
          TX packets:126 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3000
          RX bytes:1762547 (1.6 MiB)  TX bytes:11234 (10.9 KiB)
```

This concludes this lab.

Lab: Using the MIPI CSI Camera (Linux)

This lab will walk you through the steps in setting up the MIPI CSI camera on the EVK under the installed Android OS.

Required equipment

MCIMX8MMini-EVK with Power Supply

Monitor (1080P) with HDMI cable

MIPI CSI camera accessory; part number: MINISASTOCSI

- MIPI Camera
- MIPI DSI to HDMI Converter

With the board **powerd off**, connect the camera cable to the MIPI Camera interface connector labeled J802 CSI MIPI.
Boot the board into Linux and log on

For 1080p @60 fps camera capture issue the following command:
gst-launch-1.0 v4l2src device=/dev/video0 ! video/x-raw,width=1920,height=1080 ! waylandsink

For 1080p @30 fps:
gst-launch-1.0 v4l2src device=/dev/video0 ! video/x-raw,width=1920,height=1080, framerate=30/1 ! waylandsink

For 720p @60fps:
gst-launch-1.0 v4l2src device=/dev/video0 ! video/x-raw,width=1280,height=720 ! waylandsink

Below is a copy of what you should see on the command line, and then you should see what the camera sees on the monitor.

```
root@imx8mmevk:~# gst-launch-1.0 v4l2src device=/dev/video0 ! video/x-raw,width=1920,height=1080, framerate=30/1 !
waylandsink
Setting pipeline to PAUSED ...
Pipeline is live and does not need PREROLL ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
[  993.414998] alloc_contig_range: [82c00, 82ff5) PFNs busy
[  993.421164] alloc_contig_range: [82c00, 830f5) PFNs busy
[  993.427159] alloc_contig_range: [82e00, 831f5) PFNs busy
[  993.433221] alloc_contig_range: [82f00, 832f5) PFNs busy
[  993.452248] ov5640_mipi 2-003c: s_stream: 1
```

Lab: Creating Video Chat

Required Equipment

Requires 2 sets of each:

MCIMX8MMini-EVK with Power Supply
MIPI to HDMI adapter
Monitor connected via HDMI
MIPI to CSI Camera adapter

USB Cable, type C to type B
Computer with Terminal Program

Lab Summary

In the lab, we will use included Linux commands to emulate a video chat. While these commands will create two-way video conferencing, it is for demonstration of capabilities of the i.MX8MMini SOC and EVK.

This lab demonstrates the built in video encoding using the VPU and

This lab requires 2 boards, so look to your right or to your left and partner with another team so you have two boards.

1) Boot both the units to the serial prompt
2) Copy the video chat scripts from the usb thumb drive to /home/root using the command
   `cp /run/media/sda1/Lab-Running Linux/Video_chat_scripts/* ~/. <RETURN>`
3) Connect both boards to the wifi.

Note: The boards should have a connection from the earlier lab exercise, if not please get the network connection.

Edit the vid_chat_rx.sh script to point to the transmitting/partner board. Change the ip that is pointed to by the `host=`
Use the command ifconfig to find the ip addresses of each board.

4) cd /home/root/vid_chat <- these scripts are provided for the class work, it is not included on the standard Linux distribution provided by NXP.
5) Execute "./vid_chat_tx.sh" on the **both** the boards first.
6) Start the client by executing "`./vid_chat_rx_preview.sh <ip_address_of_tx>`". Insert the IP address of the board you are connecting to. Use the `ifconfig` command (as above) to find the ip address. Make sure to start the RX after both the TX are started.

Contents of vid_chat_tx.sh

```
gst-launch-1.0 -v v4l2src device=/dev/video0 ! "video/x-raw,width=1920,\
height=1080,framerate=30/1" ! queue  ! videoconvert !  vpuenc_h264 qos =true ! \
h264 parse ! queue  ! matroskamux ! queue ! tcpserversink port=5004 host=0.0.0.0 &
```

Contents of vid_chat_rx.

```
#!/bin/bash
If [ -z "$1"]; then
   echo 1>&2 usage: $0 ip address of transmitter
    exit 1
if
gst-launch-1.0 -v tcpclientsrc host=$1 port=5004 ! typefind ! matroskademux ! /
multiqueue !  vpudec !  waylandsink sync=false &
```

Contents of vid_chat_rx_preview.sh

```
#!/bin/bash
If [ -z "$1"]; then
   echo 1>&2 usage: $0 <ip address of transmitter>
```

```
    exit 1
if
gst-launch-1.0 imxcompositor_g2d name=c \
sink_0::xpos=0 sink_0::ypos=0 sink_0::width=1920 sink_0::height=1080\
sink_1::xpos=0 sink_1::ypos=0 sink_1::width=640 sink_1::height=480 ! queue ! waylandsink sync=false \
tcpclientsrc host=$1 port=5004  timeout=10 ! typefind ! matroskademux ! multiqueue ! vpudec \
 ! queue ! videoconvert ! c.sink_0 tcpclientsrc host=127.0.0.1 port=5004  timeout=10 ! typefind ! \
  matroskademux ! multiqueue ! vpudec ! queue ! videoconvert ! c.sink_1
```

Lab: Running Android

This lab will take you through the steps in installing Chrome Web Browser, setting up WiFi, using the MIPI DSI and CSI interfaces on the EVK under the installed Android OS.

Required equipment

MCIMX8MM-EVK
Monitor (1080P) with HDMI cable
USB Hub
Mouse
Keyboard (optional)

Connect the board, as described above in Initial Setup and operation. Be sure you have a USB hub, as this will allow you to use the mouse and USB thumb drive together.

Lab Connecting to WiFi

This lab will walk you through the steps in setting up WiFi on the EVK under the installed Android OS.

- Open the desktop
- Click on Settings
- Click on Network and Internet
- Select the wifi access point and enter proper credentials
- Wait for things to connect
- You are on the internet (or at least connected to the wifi😊)

Note: Connect to classroom_5 and enter password= nxp_2018

Lab: Installing Chrome Web browser

The default Android image is a minimum image and does not include many applications. So we will install the Chrome Web Browser. The Chrome web browser needs to be downloaded from the internet at

https://www.appsapk.com/chrome-browser and save it to a USB Thumb Drive. Insert the USB Thumb Drive into the USB port of the EVK.

Download file for hands on is copied in this path: E:\Lab-Imaging_the_EVK\Android

- Open the desktop
- Click on the Files icon
- Click (Open) the USB Drive
- Double click on the Chrome Browser APK file
    - Staging app..
    - Vulnerability warning – click Continue
    - Do You want to install this application? Click Install
    - Installing
    - App Installed – click Open

With either a good wifi connection or a proper ethernet connection, you can now browse the internet.

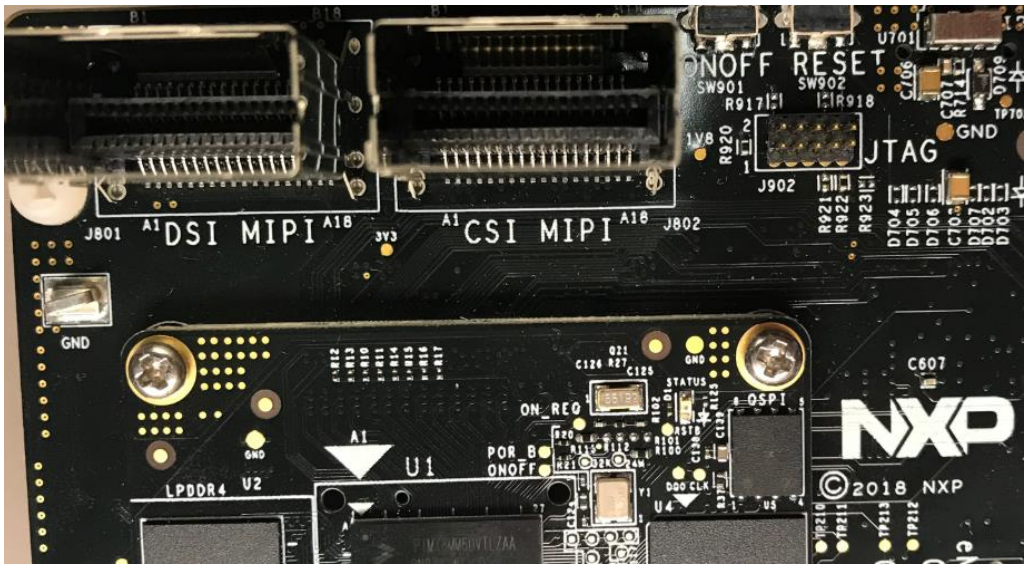
Lab: Using the MIPI CSI Camera (Android)

This lab will walk you through the steps in setting up the MIPI CSI camera on the EVK under the installed Android OS.

Required equipment

MCIMX8M-EVK
Monitor (1080P) with HDMI cable
Mouse
MIPI CSI camera accessory; part number: MINISASTOCSI
MIPI Camera
MIPI DSI to HDMI Converter

With the board **powerd off**, connect the camera cable to the MIPI Camera interface connector labeled CSI MIPI J802.



Connect the monitor and mouse to the appropriate connectors.

Boot the board into Android (default SD card image)

Open the desktop.

Click on Camera.

You will see the camera image on the display.

Appendix A Additional Information

Boot Switches

Boot Modes

There are multiple boot options for the SOC. The sections below outline the main ones used for the EVK. Each mode requires the Boot Switches to be placed in the proper settings.

Serial Download

Boot Switches for Serial Download Mode

SW1101 (1-10)	1	0	1	0	X	X	X	X	X	X

SW1102 (1-10)	X	X	X	X	X	X	X	X	X	X

eMMC

Boot Switches for eMMC boot

SW1101 (1-10)	0	1	1	0	1	1	0	0	0	1
SW1102 (1-10)	0	0	0	1	0	1	0	1	0	0

Note: On the very early boards, the silkscreen on the board for SW1102 is incorrectly labeled for eMMC Boot.

1 = ON, 0 = OFF

Be sure SW1101 and SW1102, Boot Mode Switches, are set for EMMC Boot. After the board images are loaded into the eMMC and the boot switches are correctly configured, the system is ready to run.

Power on the EVK by sliding the power switch SW191 to ON.

During the boot process, the OS logo will appear on the HDMI display.

The OS UI can be seen after the boot process is finished. You can start operating with the mouse.

SD Card

Boot Switches for SD Card

SW1101 (1-10)	0	1	1	0	1	1	0	0	1	0
SW1102 (1-10)	0	0	0	1	1	0	1	0	0	0

1 = ON, 0 = OFF

Be sure SW1101 and SW1102, Boot Mode Switches, are set for SD Card Boot. After the board images are loaded into the SD Card and the boot switches are correctly configured, the system is ready to run.

Power on the EVK by sliding the power switch SW191 to ON.

During the boot process, the OS logo will appear on the HDMI display.

The OS UI can be seen after the boot process is finished. You can start operating with the mouse.

i.MX8MMini Resources

i.MX 8MMini Links: https://www.nxp.com/imx8mmini

EVK board: https://www.nxp.com/support/developer-resources/software-development-tools/i.mx-developer-resources/evaluation-kit-for-the-i.mx-8m-mini-applications-processor:8MMINILPD4-EVK

Debug Serial Console

Windows user's may need to update the serial drivers on your computer. The drivers can be found at https://www.ftdichip.com/Drivers/VCP.htm Be sure to select the proper driver for your OS.

Revision History

Version	Author	Changes	Date
1.0	M Ruthenbeck	Original Release	5 Oct 2018
1.1	M Ruthenbeck	Updated video script and process	29 Oct 2018
1.2	M Ruthenbeck	Updated for Rev C board, added Boot mode switches, Updated SW links, changed programming Linux to SD from eMMC.	27 Feb 2019
1.3	M Bajaj	Moved UUU hands on first and modified the class to flash Android in eMMC	12 April 2019
1.4	M Ruthenbeck	Added RPA section to DDR tools Added Virtualization labs and information on creating the SD card used in the lab	1 May 2019
1.5	M Ruthenbeck	Updated to reflect GA released images	1 Aug 2019

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

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.

Rev 1.
08/2019