Firmware User Manual

IMX8X AI_ML Reference Design

Date: February, 2021 | Version 3.1

CONTENTS

FIGURES

ACRONYMS AND ABBREVIATIONS

Definition/Acronym/Abbreviation	Description
cd	Change directory
scp	Secure copy over the network
dfl	Default
Wi-Fi	Wireless fidelity
LTE	Long-Term Evolution
ML	Machine Learning
SVM	Support Vector Machine
CNN	Convolutional Neural Network
ARM NN	ARM Neural Network
AI_ML board	Artificial intelligence and Machine Learning board featuring the NXP i.MX 8X MPU
AES	Advanced Encryption Standard
AHAB	Advanced High Assurance Boot
AWS	Amazon Web Services
BSP	Board Support Package
CA	Certificate Authority
CAAM	Cryptographic Acceleration and Assurance Module
CMS	Cryptographic Message Syntax
CSF	Command Sequence File
CSR	Certificate Signing Request
CST	Code Signing Tool
DCD	Device Configuration Data
GG	AWS Greengrass
OS	Operating System
OTP	One-Time Programmable
PKI	Public Key Infrastructure
SA	Signature Authority
SCFW	SCU Firmware
SDP	Serial Download Protocol
SECO	Security Controller
SPL	Secondary Program Loader
SRK	Super Root Key
SSK	Security Starter Kit
TPM	Trusted Platform Module
USB	Universal Serial Bus
TLS	Transport Layer Security
RSA	Rivest–Shamir–Adleman
IoT	Internet of Things
HSM	Hardware Security Module

PKCS#11	PKCS#11 (Public Key Cryptography Standards) defines an API to communicate with cryptographic security tokens such as smart cards, USB keys and HSMs
HW	Hardware
MQTT	Message Queuing Telemetry Transport
SSL	Secure Sockets Layer
SHA	Secure Hash Algorithm
SDK	Software Development Kit
ECC	Elliptic Curve Cryptography
ARN	Amazon Resource Name
SECO	Security Controller
FW	Firmware
NV RAM	Non Volatile Random Access Memory
API	Application Programming Interface
UUID	Universally Unique Identifier
SCP	Secure Copy Protocol
SCU	System Control Unit
IAM	AWS Identity and Access Management
TSS	TPM2 Software Stack

1 INTRODUCTION

1.1 Purpose of the Document

Purpose of this document is to use / understand / flash / demonstrate interfaces on iMX8ML_RD firmware.

1.2 About The System

This system contains iMX8X reference design with multiple interface. This is used for Machine learning experience.

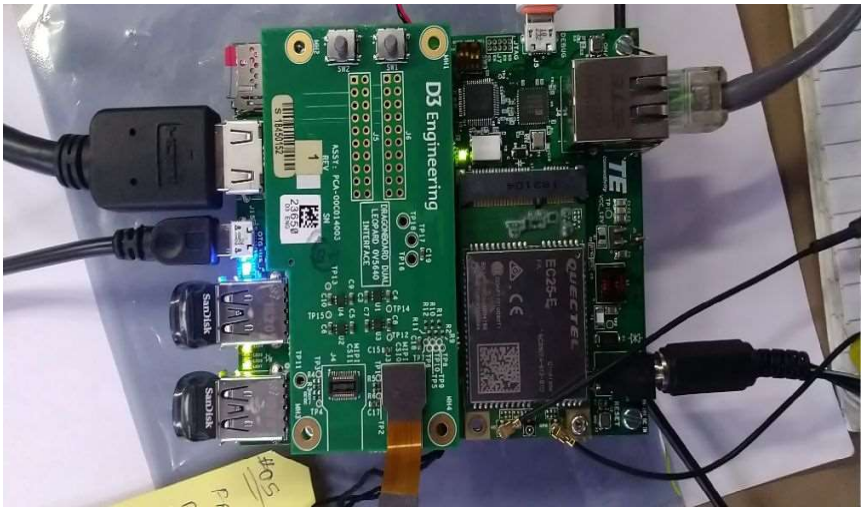


Figure 1: iMX8XML RD AIML board

1.3 Intended Audience

This document is for developers and end-users who want to demonstrate all the interface of AIML boards.

1.4 Prerequisites

Below are the list of Hardware and software needed to enable demonstration.

- x86 host system having Linux Ubuntu 16.04 LTS or Ubuntu 18.04 installed (to build Yocto image)
- For building machine-learning components, at least 250 GB disk space is recommended.
- Linux PC (Configuration:: Optimal 16GB-RAM,Processor:Octa-core).
- Basic understanding of Linux commands
- Setup will require following:
    - AI_ML Board
    - SD-card -32GB
    - MicroUSB debug cable
    - Power Supply –
        - MEANWELL GST60A12-P1J
        - 5.5/2.1mm to 4.75/1.7mm cable DC plug converter
- Internet connectivity (Wi-Fi/Ethernet) of Board and Linux PC should be on same Network

2 ENVIRONMENT SETUP

2.1 Steps to build Yocto Image

We already prepared release package **AIML_L5_4_Rel_3_1.zip,** which contains all the packages and BSP changes required patches for AIML firmware image. User need to download meta-layer first to build image for AIML.

To build AIML firmware on LINUX HOST PC, user will follow below steps:

- Now we need to download new repo for AIML. Currently we are using kernel version 5.4.47 zeus release repo.

```
$: sudo apt-get install repo
$: git config --global user.name "Your Name"
$: git config --global user.email "Your Email"
$: git config --global user.email "Your Email"
$: git config --list
```

- Now Download and extract the **AIML_L5_4_Rel_3_1.zip** it contains below:

AIML_L5_4_Rel_3_1
├── Prebuilt_Image
├── AIML_L5_4_47_Rel_3_1_patches
├── yocto_build_setup_aiml.sh
└── Yocto_build_setup_steps.txt

- Now Download Yocto Project environment into local directory

```
$: cd AIML_L5_4_Rel_3_1/
$ sudo chmod 755  yocto_build_setup_aiml.sh
$ ./yocto_build_setup_aiml.sh
```

Note:[Building firmware image it will take ~10 hours to download all packages and build,the time may vary based on your HOST PC configurations]

- After successful build final sd card image reside at below location:
imx-yocto-bsp/bld-xwayland-aiml/tmp/deploy/images/imx8qxpaiml/

- Filename should be imx-image-full-imx8qxpaiml.wic.bz2 which is soft link of original build image file imx-image-full-imx8qxpaiml-<TIMESTAMP>.rootfs.wic.bz2

- If user want to clean previously build image and want to run it again then we must first clean it with command **“cleanall” or “cleansstate”**

```
$: bitbake imx-image-full -c cleanall
$: bitbake -v imx-image-full     (If user want to turn on verbose)
```

- If user want to clean any particular package then also we can do that with command **“cleanall”** or **“cleansstate”**

```
$: bitbake <PACKAGE_NAME> -c cleanall
$: bitbake <PACKAGE_NAME>

e.g.
$: bitbake linux-imx -c cleanall
$: bitbake linux-imx        (Build linux kernel only)

Same way
$: bitbake u-boot-imx -c cleanall
$: bitbake u-boot-imx       (Build uboot code only)

$: bitbake imx-gpu-sdk -c cleanall
$: bitbake imx-gpu-sdk      (Build gpu sdk only)

$: bitbake opencv -c cleanall
$: bitbake opencv           (Build opencv package)

$: bitbake python3-scipy -c cleanall
$: bitbake python3-scipy    (Build scipy python package for python3)
```

- Please note that, if you re-build any module then it is better to re-build all modules, which are dependent on that module. For example, if you change anything in Linux kernel code and rebuild it using above commands then you must need to re-build kernel-module-laird, imx-gpu-sdk etc. packages to avoid conflicts.

2.2 Download firmware package

- Download the provided SD card (**wic.bz2**) image on Linux PC
- Open terminal in Host PC from left desktop panel or using keyboard shortcut (**ctrl + t)**
- From command terminal traverse to the location where firmware has been downloaded using **cd** command
  # cd /home/user/download/imximages/
- use **ls** command to verify the existence of image downloaded
  # ls -l
- *Verify md5 check sum of downloaded image with given md5sum.*
  # md5sum <image name>.wic.bz2
- Extract the provided **.bz2** image using **bunzip2** command, which will take couple of minutes.
  bunzip2 -dkf <image_name>.wic.bz2
- Once done, will end with **.wic** image in the same directory and can again be verified using **ls -l** command.

2.3 Flash the firmware image to SD Card in LINUX HOST PC

- Plugin micro SD card into x86 Linux Host PC
- *Verify the node created for SD card into /dev directory*
  # ls -l /dev/sd*
- Open terminal and traverse to the location where downloaded firmware image is residing using **cd** command
- Ensure the extracted firmware image's file format is **.wic** using **ls -l** command
- Use below command for flashing if the SD card's entry in Linux is **/dev/sdb**
  # sudo dd if=<image_name>.wic of=/dev/sdb bs=1M conv=fsync ;sync
- Above command will take couple of minutes or more (depending upon PC config) to flash the SD card
- Once done remove and insert the SD card, two drives will get mounted if the above command is successful, named <boot> and <rootfs>
- **Eject (safely remove) SD card from host PC and plug it into board's SD card slot**

2.4 Flash the firmware image to SD Card in Windows HOST PC

- Plugin micro SD card into x86 Windows Host PC
- Install win32 Disk Imager (*https://sourceforge.net/projects/win32diskimager/*)
- Format SD card with **FAT** file system.
- Plug SD card with card reader. It must shows any drive like **"E:"**
- Download appropriate production image ***.wic.bz2**
- Extract ***.wic.bz2** image using winzip or 7-zip. It will create ***.wic** image.
- Run Win32 Disk Imager
- Select .sdcard image file and target drive i.e. E: for input Image File. (see below figure)

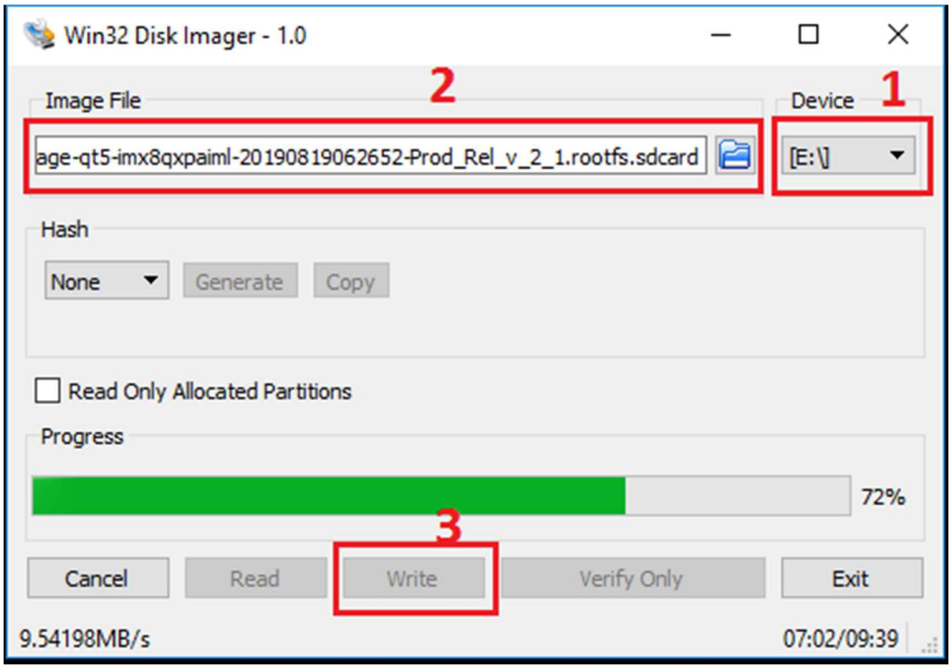


Figure 2: Win32 Disk Imager for flashing SD Card

- Click on **Write**.
- After successful transfer, success message will pop up and we got around

2.5 Hardware Installation

- Place hardware board on statically clean place
- Insert flashed SD card to J5 SD card slot.
- Attach serial cable's micro end to board's J10 Connector (near Ethernet connector) and USB end to host x86 pc's USB connector.
- Attach Ethernet cable to board's Ethernet connector J12.
- Apply 12V-5A power supply (provided with board) to board on J13 connector once all the other hardware setup done as per requirement.

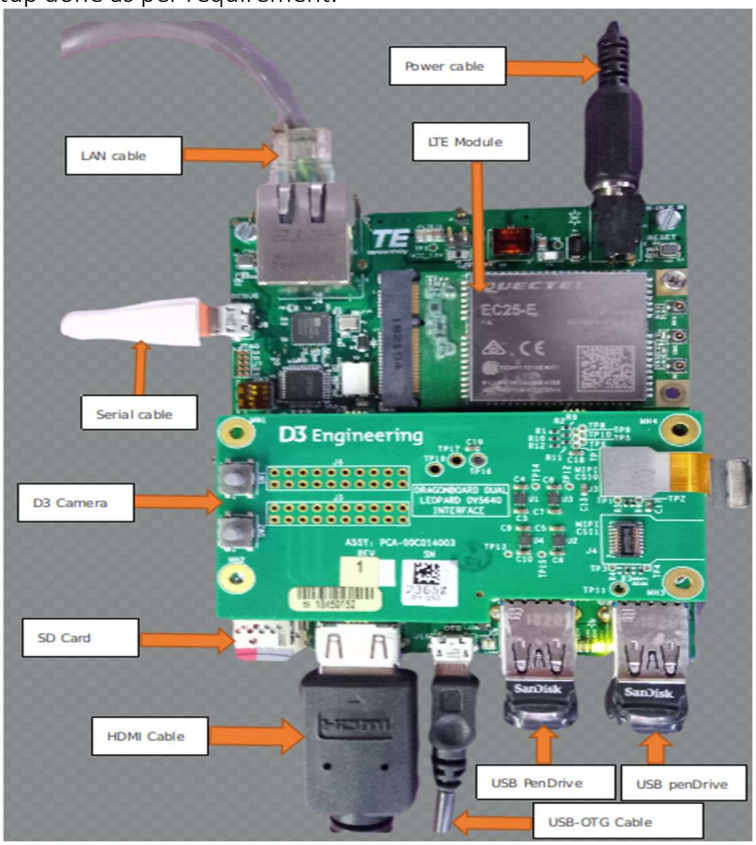


Figure 3: Hardware Setup

3 RUNNING DEMOS

Note: [Demos which require a change in .dtb file has been mentioned in their demo steps,else please keep the .dtb file as imx8qxp-aiml-ei.dtb]

3.1 Ethernet Demo

- Plug in Ethernet cable to target board as per above figure.
- Power up the board.
- Once board gets booted, apply below command using console (minicom require)
  # ifconfig eth0

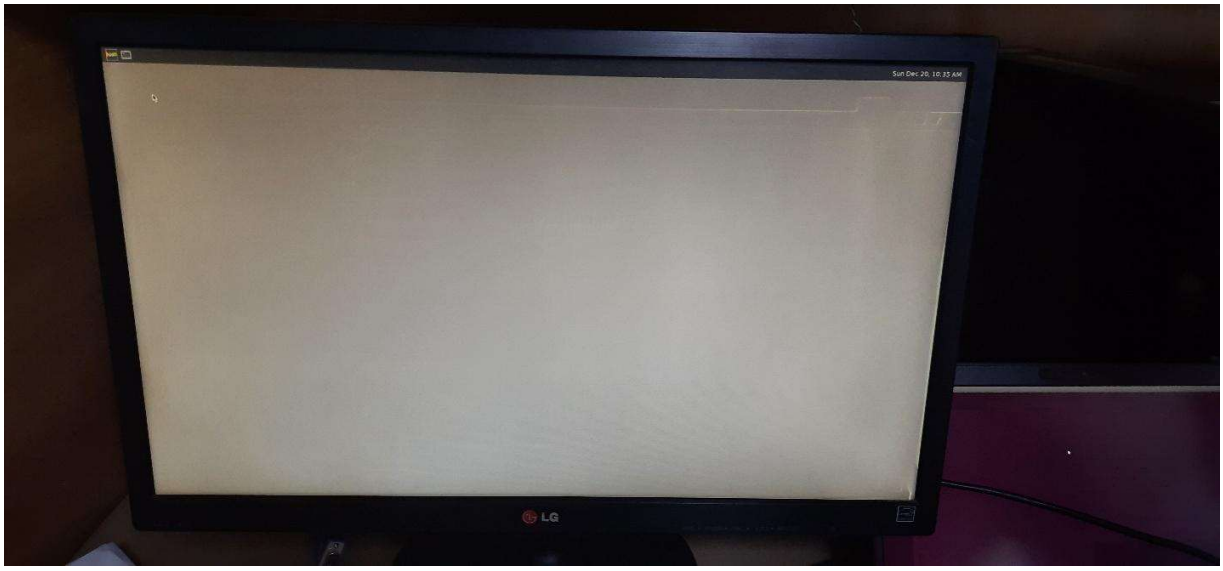
```
root@imx8qxpaiml:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 8e:99:e8:13:f8:41
          inet addr:10.100.134.221  Bcast:10.100.135.255  Mask:255.255.252.0
          inet6 addr: fe80::8c99:e8ff:fe13:f841/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:154 errors:0 dropped:25 overruns:0 frame:0
          TX packets:56 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:13012 (12.7 KiB)  TX bytes:9291 (9.0 KiB)

root@imx8qxpaiml:~#
```

- Above command will show the eth0 “IP” as per above screenshot. If IP is not putting up ,check with ethernet Cable plugged into the board.
- Ping <any server IP>

```
root@imx8qxpaiml:~# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=118 time=17.5 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=118 time=17.7 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=118 time=17.5 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=118 time=17.5 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=118 time=17.6 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=118 time=17.5 ms
^C
--- 8.8.8.8 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5009ms
rtt min/avg/max/mdev = 17.504/17.567/17.671/0.060 ms
root@imx8qxpaiml:~#
```

3.2 HDMI Demo

- Ensure board is not in power up state and SD card is flashed with the latest provided image.
- Insert HDMI cable into board's J2 HDMI connector.
- Apply power to board and go to terminal of x86 host system and open board's console as mentioned above
- Console will show booting logs.
- At the board boots, console will hold on login prompt where user can enter username as **root**. (no password)
- At this time connected HDMI display will show grey image (with small flowers) on desktop and should stop showing "No Signal"



Play local videos on HDMI display

- Ensure Ethernet is connected with board
- Go to board's console and type below command from x86 minicom

  # ifconfig eth0

  #aplay -l (List of PLAYBACK Hardware Devices)

- Get the IP address of Ethernet eth0 interface and note the same.
- Go to x86 host system and extract the provided zip.
- Locate to test video location from command line in x86 (no minicom require)
- Apply below command
  # scp ./<file_name>.mp4 root@<noted ip address of board>:/home/root/
- This will copy the video file from host x86 to board's /home/root location
- Go to board's console (require minicom) and ensure video got copied using ls -l command, will show you <file_name>.mp4 in current directory.

- Apply below command to find the SPDIF audio hw device and card number.

```
root@imx8qxpaiml:~# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: imxspdif [imx-spdif], device 0: S/PDIF PCM snd-soc-dummy-dai-0 [S/PDIF PCM snd-soc-dummy-dai-0]
Subdevices: 1/1
Subdevice #0: subdevice #0
```

- Be in the board's console ,connect handsfree in audio jack of the HDMI display (If internal speaker not available in HDMI display)and apply below command to play video over HDMI Display and audio over SPDIF by configure above noted card and device number in plughw:<card number>,<device number>

```
# gplay-1.0  <file_name>.mp4  --audio-sink=alsasink plughw:0,0
```

- Above command will print logs on console of board and will get played over HDMI display

For HDMI Hot plug detection, we must need to connect HDMI before we power-on board. Otherwise, appropriate framebuffer not created at boot up and HDMI hotplug will not work. Same condition applied when we move from one resolution HDMI to another.

3.3 Camera Demo

Play live video stream from camera on HDMI display

- Attach camera module to high-speed connector.
- To watch live stream over the HDMI, connect HDMI Display too.
- Power up the board
- Go to board's console (require minicom) and immediately stop at **u-boot autoboot** console by pressing any key.
- Apply below commands for changing dtb file

```
# setenv fdt_file imx8qxp-aiml-ei-ov5640.dtb
# saveenv
# boot
```

- Run below command to see preview of camera on HDMI display

```
# export DISPLAY=:0
# gst-launch-1.0 v4l2src device=/dev/video1 ! video/x-raw,width=1280,height=720 ! ximagesink
```

- This will show preview (live) streaming over the attached HDMI.

Capture image from Mipi-camera

- Go to board's console (require minicom) and power up the board with above mentioned dtb change configuration.
- Ensure Ethernet is plugged-in to get image from board to local x86 host pc.
- Apply below command to capture image from camera.

```
# gst-launch-1.0 v4l2src device=/dev/video1  num-buffers=1 ! jpegenc ! filesink location=
/home/root/test.jpg
```

- Above command will capture image named test.jpg in /home/root/ location
- Copy image from board to local pc using below command
  # scp test.jpg <user name of host pc>@<ip of host pc>:/home/user/Desktop
- Go to local pc's /home/user/Desktop and watch image into image viewer to verify captured image from board's camera

Capture image from USB-camera

- Go to board's console (require Minicom) and power up the board with above-mentioned USB-Camera plugged-in.
- Ensure Ethernet is plugged-in to get image from board to local x86-host pc.
  Run below command to see preview of camera on HDMI display

```
#gst-launch-1.0 v4l2src device=/dev/video5 !
'image/jpeg,width=640,height=480,framerate=30/1' ! jpegdec ! autovideosink
```

- Below command will capture image named test.jpg in /home/root/ location

```
# gst-launch-1.0 v4l2src device=/dev/video5 num-buffers=1 ! jpegenc ! filesink location=/home/root/test2.jpg
```

- Copy image from board to local pc using below command

```
# scp test2.jpg <user name of host pc>@<ip of host pc>:/home/user/Desktop
```

Go to local pc's /home/user/Desktop and watch image into image viewer to verify captured image from board's camera

3.4 Wi-Fi Demo

- Open terminal and follow the below commands to run Wi-Fi demo
  # ifconfig wlan0 up
  # ip link show wlan0
- Above command will show the below details of wlan0.

```
3: wlan0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group
default qlen 1000
    link/ether 00:25:ca:14:11:6c brd ff:ff:ff:ff:ff:ff
```

- Edit **/etc/wpa_supplicant.conf** file as per below in target board

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
ssid="closenet"
scan_ssid=1
key_mgmt=WPA-PSK
psk="123456789"
}
```

- Turn on hotspot from mobile device or from Router.
- Change its ssid name /Network name to "**closenet"** and password to **"123456789"** (Create Close network). Make sure Security Type must be **WPA-PSK or WPA2-PSK.**

  Note: User can select any name (ssid) and password (psk) except any space or special character here and can change above wpa_supplicant.conf file accordingly. For example, ssid "**Anil's iPhone**" is not valid one.
- Use below command to connect with your SSID.
  #wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant.conf
- Now verify connection using below command.
  #iw wlan0 link

```
root@imx8qxpaiml:~# iw wlan0 link
Connected to 8a:a3:03:7d:26:5e (on wlan0)
SSID: hello
freq: 2412
RX: 698 bytes (5 packets)
TX: 3835 bytes (25 packets)
signal: -48 dBm
rx bitrate: 72.2 MBit/s
tx bitrate: 1.0 MBit/s
bss flags: short-slot-time
dtim period: 2
```

```
beacon int: 100
root@imx8qxpaiml:~#
```

#ip address list wlan0

- Discover IP address using below command.
  #udhcpc -i wlan0 -n -q
- List inet address using below command.
  #ip address list wlan0

```
root@imx8qxpaiml:~# ip address list wlan0
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
link/ether 00:25:ca:17:0c:1a brd ff:ff:ff:ff:ff:ff
inet 192.168.43.107/24 brd 192.168.43.255 scope global wlan0
valid_lft forever preferred_lft forever
root@imx8qxpaiml:~#
```

#ip route show
#ping <Any network IP >

- Response should be as per below logs

```
PING 192.168.43.1 (192.168.43.1) 56(84) bytes of data.
64 bytes from 192.168.43.1: icmp_seq=1 ttl=64 time=12.6 ms
64 bytes from 192.168.43.1: icmp_seq=2 ttl=64 time=10.6 ms
64 bytes from 192.168.43.1: icmp_seq=3 ttl=64 time=27.7 ms
64 bytes from 192.168.43.1: icmp_seq=4 ttl=64 time=6.34 ms
64 bytes from 192.168.43.1: icmp_seq=5 ttl=64 time=22.8 ms
64 bytes from 192.168.43.1: icmp_seq=6 ttl=64 time=8.26 ms
```

- The above ping response validates Wi-Fi's working state.

3.5 Bluetooth Demo

- Go to board 's console and apply below command

```
# pactl unload-module  module-bluetooth-discover
# pactl load-module module-bluetooth-discover
# brcm_patchram_plus --baudrate 1500000 --patchram /lib/firmware/brcm/BCM4335C0.hcd --enable_hci --no2bytes --tosleep 1000 /dev/ttyLP0 &
```

- Wait until the above command's complete the command response

```
root@imx8qxpaiml:~# Done setting line discpline
```

- Apply below command's it's enable the blue LED on the board.

```
# hciconfig hci0 up
# hciconfig hci0
```

- User will get the hcio interface

```
root@imx8qxpaiml:~# hciconfig hci0
hci0: Type: Primary Bus: UART
BD Address: 00:25:CA:17:0C:1B ACL MTU: 1021:8 SCO MTU: 64:1
UP RUNNING
RX bytes:1336 acl:0 sco:0 events:70 errors:0
TX bytes:1156 acl:0 sco:0 commands:70 errors:0
root@imx8qxpaiml:~#
```

- Run the "bluetoothctl" utility

```
# bluetoothctl
[bluetooth]# power on
[bluetooth]# agent on
[bluetooth]# default-agent
[bluetooth]# pairable on
[bluetooth]# scan on

> Copy mac address
[bluetooth]# scan off
[bluetooth]# pair <mac address>

> Approve pairing on Device if required
[bluetooth]# trust <mac address>
[bluetooth]# connect <mac address>
[bluetooth]# quit

> Sending file command.
# export $(dbus-launch)
# /usr/libexec/bluetooth/obexd &

# obexctl
[obex]# connect <mac addr>
```

```
[<mac addr>]# send <file>
[<mac addr>]# disconnect
[<mac addr>]# quit
```

- Play the audio over BT commands
- Collect the audio file from the support package folder.
- Get the Bluetooth headset or Bluetooth speaker and connect using **bluetoothctl** utility.
- Check bluetooth device card profile if its not set to a2dpsink then set to a2dpsink to play audio on that device. Use below command to check current audio profile.

```
# pactl list cards
```

- If connected device current profile not set to a2dpsink then set using below command.

```
# pactl set-card-profile <card number> a2dp_sink
I.e pactl set-card-profile 2 a2dp_sink
```

- After setting profile it time to play audio over bluetooth device.

```
# paplay -p --device=bluez_sink.<Device MAC>.a2dp_sink <path of the wav file>
I.e: paplay -p --device=bluez_sink.90_7A_58_33_ED_1B.a2dp_sink Mast_Magan.wav
```

Record audio over Bluetooth

- Connect mobile with our modem using above **bluetoothctl** command.
- Check bluetooth device card profile if its not set to a2dpsource then set to a2dpsource to play audio on that device. Use below command to check current audio profile.
  # pactl list cards
- If connected device current profile not set to a2dpsource then set using below command.

```
# pactl set-card-profile <card number> a2dp_source
I.e pactl set-card-profile 2 a2dp_source
```

- play the music on mobile player
- run below command to capture the audio from Bluetooth

```
#paplay -r –device=bluez_source.<Device MAC>.a2dp_source <path of wav file>
I.e: paplay -r --device=bluez_source.88_A3_03_7D_26_5D.a2dp_source test_rec.wav
```

- copy recorded file in your host PC and verify with any player on host PC

3.6 USB Hub demo

- Connect USB device disk to USB port of target board
- Go to board ‘s console and apply below command
- **lsusb**
- below info should appears on the console where it will show plugged-in USB disk details (i.e. SanDisk in this case)

```
Bus 002 Device 003: ID 0781:5583 SanDisk Corp. Ultra Fit
Bus 001 Device 002: ID 04b4:6502 Cypress Semiconductor Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 002: ID 04b4:6500 Cypress Semiconductor Corp.
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
```

3.7 USB OTG as host

- Connect USB device disk to USB OTG port of target board
- Go to board 's console and apply below command as same as USB hub

```
# lsusb
Bus 002 Device 003: ID 0781:5583 SanDisk Corp. Ultra Fit
Bus 001 Device 002: ID 04b4:6502 Cypress Semiconductor Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 002: ID 04b4:6500 Cypress Semiconductor Corp.
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
```

3.8 USB OTG as Devices

- Connect USB cable (same like debug uart cable) USB OTG port of target board
- Run the below command

```
# dd if=/dev/zero of=/mass_storage bs=1M seek=256 count=0
# mkfs.fat /mass_storage
# cat <<EOT | sfdisk --reorder /mass_storage
>hello
> EOT
# mkfs.vfat /mass_storage
# chmod 777 /mass_storage
# mount -o loop /mass_storage /mnt/
# mount
# modprobe  g_mass_storage  file=/mass_storage
```

- Disconnect and connect the USB cable
- User will see the drive on LINUX host machine.

Figure 4: USB OTG as device in Linux

- Please note that on Window system mass storage been created but not seen the drive (although we have created FAT file system). Must be an issue with Windows system. Therefore, User need to test this with Linux system only.

3.9 LTE Demo

- Connect Quectel module with target board's as per below image
- Set user SIM Card APN using below command.
  Note: please identify your SIM card APN as per your service provider.
  # export LTE_APN=<Your APN>
  i.e export LTE_APN=airtelgprs.com
- Go To Board's console and apply below command
  # pppd call quectel-ppp &
- Edit /etc/resolv.conf and add appropriate name server as per below
  nameserver 59.144.127.117
  nameserver 59.144.144.46
  nameserver 8.8.8.8
- Save above file and apply below command to check connection and IP address.
  # ifconfig ppp0

```
root@imx8qxpaiml:~# ifconfig ppp0
ppp0 Link encap:Point-to-Point Protocol
inet addr:100.78.168.109 P-t-P:10.64.64.64 Mask:255.255.255.255
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
RX packets:4 errors:0 dropped:0 overruns:0 frame:0
TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:3
RX bytes:52 (52.0 B) TX bytes:58 (58.0 B)
root@imx8qxpaiml:~#
```

- Please confirm IP firtst ,then use below command it should be able to ping to google.com
  # ping www.google.com -I ppp0

```
root@imx8qxpaiml:~# ping www.google.com -I ppp0
PING www.google.com (216.58.203.196) from 25.127.216.25 ppp0: 56(84) bytes of data.
64 bytes from bom07s12-in-f4.1e100.net (216.58.203.196): icmp_seq=1 ttl=114 time=43.0
ms
64 bytes from bom07s12-in-f4.1e100.net (216.58.203.196): icmp_seq=2 ttl=114 time=63.8
ms
64 bytes from bom07s12-in-f4.1e100.net (216.58.203.196): icmp_seq=3 ttl=114 time=82.0
ms
```

- Will be able to ping to google.com.

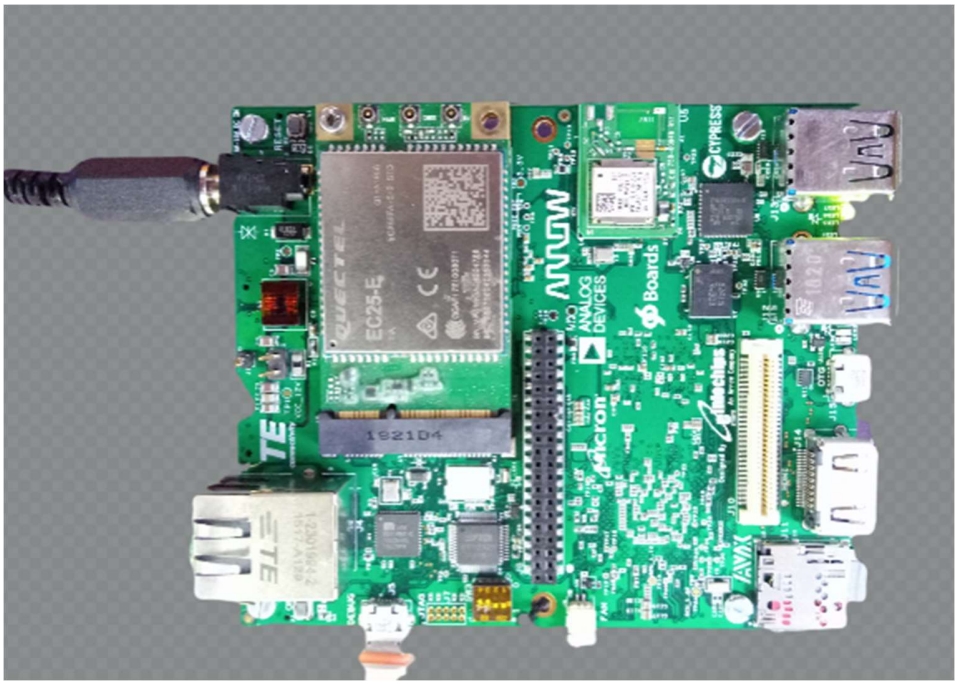


Figure 5: QUECTEL Module on target board

3.10 EEPROM

- Run below command to test EEPROM
- To write in eeprom

```
#echo Hello_einfochips > /sys/bus/i2c/devices/16-0050/eeprom
```

- To read from eeprom

```
$ cat /sys/bus/i2c/devices/16-0050/eeprom | hexdump -C
```

- Compare the output will contain data which was written to EEPROM

```
root@imx8qxpaiml:~# cat /sys/bus/i2c/devices/16-0050/eeprom | hexdump -C
00000000 48 65 6c 6c 6f 5f 65 69 6e 66 6f 63 68 69 70 73 |Hello_einfochips|
00000010 0a 00 00 00 00 00 00 00 00 00 00 00 37 36 34 30 |............7640|
00000020 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |................|
*
00000800 2e 20 50 72 6f 69 6e 20 65 6c 69 74 20 74 75 72 |. Proin elit tur|
00000810 70 69 73 2c 20 61 6c 69 71 75 61 6d 20 6e 65 63 |pis, aliquam nec|
00000820 20 65 6e 69 6d 20 73 69 74 20 61 6d 65 74 2c 20 | enim sit amet, |
00000830 76 65 68 69 63 75 6c 61 20 65 6c 65 6d 65 6e 74 |vehicula element|
00000840 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |................|
*
00008000
root@imx8qxpaiml:~#
```

3.11 Sensors (Acc/Gyro)

Command to test Accelerometer Sensors

- Note current values of co-ordinates using below command.
  # cat /sys/bus/i2c/devices/16-006a/iio\:device1/in_accel_x_raw
  # cat /sys/bus/i2c/devices/16-006a/iio\:device1/in_accel_y_raw
  # cat /sys/bus/i2c/devices/16-006a/iio\:device1/in_accel_z_raw
- Now move change the direction of device note values of co-ordinates using above command. It should vary.

Command to test Gyro meter Sensors

- Note current values of angle using below command.
  #cat /sys/bus/i2c/devices/16-006a/iio\:device0/in_anglvel_x_raw
  #cat /sys/bus/i2c/devices/16-006a/iio\:device0/in_anglvel_y_raw
  #cat /sys/bus/i2c/devices/16-006a/iio\:device0/in_anglvel_z_raw
- Change anlge of device and again run above commands note the values it should vary and

Command to test Temperature Sensors

- Set values of temperature sensor using below command.
  # i2cset -f -y 16 0x6a 0x11 0x10
  # i2cset -f -y 16 0x6a 0x10 0x10
- Check the value of temperature using below command.
- Check row 20 of i2cdump it should change.
  # i2cdump -f -y 16 0x6a

```
No size specified (using byte-data access)
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 00 00 00 00 00 00 00 80 00 00 00 00 00 08 00 69    .......?.....?.i
10: 10 10 44 00 00 00 00 00 38 38 00 0f 00 10 07 bb    ??D.....88.?.???
20: e6 00 10 02 0c fc 9e fe ac ff 5e fb d1 c1 00 00    ?.???????.^???..
30: 00 00 00 00 00 00 00 00 00 00 00 00 10 00 00 ff ff  ..........?....
40: 86 f8 09 00 00 00 00 00 00 00 00 00 00 00 00 00    ???.............
50: 00 00 00 00 00 00 00 00 80 00 00 00 10 00 00 00    ........?...?...
60: 00 00 00 00 00 ff 00 00 00 00 00 00 00 00 00 00    ................
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
```

3.12 LOW Speed Expansion

- UART is validated with Bluetooth
- SPI validated using SPI based chipset (SPI to CAN interface) at EI LAB
- I2C is validated with D3 camera and NXP Display
- GPIO is validated using multi-meter (Set high and low from user space)

Pin 23

```
# echo 29 > /sys/class/gpio/export
# echo out > /sys/class/gpio/gpio29/direction
# cat /sys/class/gpio/gpio29/value
# echo 1 > /sys/class/gpio/gpio29/value
# echo 0 > /sys/class/gpio/gpio29/value
```

Pin 24

```
# echo 35 > /sys/class/gpio/export
# echo out > /sys/class/gpio/gpio35/direction
# cat /sys/class/gpio/gpio35/value
# echo 1 > /sys/class/gpio/gpio35/value
# echo 0 > /sys/class/gpio/gpio35/value
```

Pin 25

```
# echo 39 > /sys/class/gpio/export
# echo out > /sys/class/gpio/gpio39/direction
# cat /sys/class/gpio/gpio39/value
# echo 1 > /sys/class/gpio/gpio39/value
# echo 0 > /sys/class/gpio/gpio39/value
```

Pin 26

```
# echo 63 > /sys/class/gpio/export
# echo out > /sys/class/gpio/gpio63/direction
# cat /sys/class/gpio/gpio63/value
# echo 1 > /sys/class/gpio/gpio63/value
# echo 0 > /sys/class/gpio/gpio63/value
```

Pin 27

```
# echo 20 > /sys/class/gpio/export
# echo out > /sys/class/gpio/gpio20/direction
# cat /sys/class/gpio/gpio20/value
# echo 1 > /sys/class/gpio/gpio20/value
# echo 0 > /sys/class/gpio/gpio20/value
```

Pin 28

```
# echo 64 > /sys/class/gpio/export
# echo out > /sys/class/gpio/gpio64/direction
# cat /sys/class/gpio/gpio64/value
# echo 1 > /sys/class/gpio/gpio64/value
# echo 0 > /sys/class/gpio/gpio64/value
```

Pin 29

echo 32 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio32/direction
cat /sys/class/gpio/gpio32/value
echo 1 > /sys/class/gpio/gpio32/value
echo 0 > /sys/class/gpio/gpio32/value

Pin 30

echo 45 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio45/direction
cat /sys/class/gpio/gpio45/value
echo 1 > /sys/class/gpio/gpio45/value
echo 0 > /sys/class/gpio/gpio45/value

Pin 31

echo 19 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio19/direction
cat /sys/class/gpio/gpio19/value
echo 1 > /sys/class/gpio/gpio19/value
echo 0 > /sys/class/gpio/gpio19/value

Pin 32

echo 46 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio46/direction
cat /sys/class/gpio/gpio46/value
echo 1 > /sys/class/gpio/gpio46/value
echo 0 > /sys/class/gpio/gpio46/value

Pin 33

echo 33 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio33/direction
cat /sys/class/gpio/gpio33/value
echo 1 > /sys/class/gpio/gpio33/value
echo 0 > /sys/class/gpio/gpio33/value

Pin 34

echo 103 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio103/direction
cat /sys/class/gpio/gpio103/value
echo 1 > /sys/class/gpio/gpio103/value
echo 0 > /sys/class/gpio/gpio103/value

3.13 High Speed Expansion

- NXP DSI connected and validated the DSI display interface

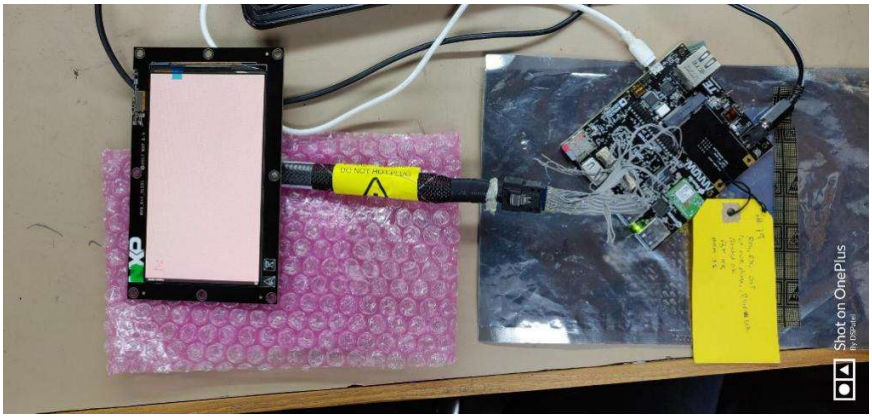


Figure 6: High Speed Expansion

- CSI is validated with D3 Camera
- USB is validated with USB pen drive (pin out connect with external USB connector)

3.14 USER LED

- Run the below command to control the Led

USER_LED1

USER_LED1 ON

echo 255 > /sys/class/leds/green\:user1/brightness

USER_LED1 OFF

echo 0 > /sys/class/leds/green\:user1/brightness

USER_LED2

USER_LED2 ON

echo 255 > /sys/class/leds/green\:user2/brightness

USER_LED2 OFF

echo 0 > /sys/class/leds/green\:user2/brightness

USER_LED3

USER_LED3 ON

echo 255 > /sys/class/leds/green\:user3/brightness

USER_LED3 OFF

echo 0 > /sys/class/leds/green\:user3/brightness

USER_LED4

USER_LED4 ON

echo 255 > /sys/class/leds/green\:user4/brightness

USER_LED4 OFF

echo 0 > /sys/class/leds/green\:user4/brightness

BT_LED

BT_LED ON

echo 255 > /sys/class/leds/blue\:bt/brightness

BT_LED OFF

echo 0 > /sys/class/leds/blue\:bt/brightness

WLAN_LED

WLAN_LED ON

echo 255 > /sys/class/leds/yellow\:wlan/brightness

WLAN_LED OFF

echo 0 > /sys/class/leds/yellow\:wlan/brightness

3.15 NOR Flash demo

- On Board's create text file
  # vi write.txt
- Write some data into it by below command
- Once done writing save and quit the above file by below command.
  # <ESC><:><wq>
- Check for the Nor flash node by below command
  # ls -l /dev/mtd0
- Erase NOR flash using below command
  # flash_eraseall /dev/mtd0
- Write the created file into NOR flash using below command.
  # time dd if=write.txt of=/dev/mtd0
  Read from NOR flash from the same location
  # dd if=/dev/mtd0 of=read.txt
  # cat read.txt
- Compare the read.txt, it should be same as write.txt
- Please note that, when we write data, we write only a few bytes of data. However, when we read, we **read the whole partition** instead of the initial few lines. Due to that, we see junk characters in the place where we did not write anything. So user need to read the file at very first few lines using vim and verify its data.

3.16 DMIC demo

- Record a wav format file using below command.
- Please use command below to check DMIC hardware card and device number.

 # arecord -l

```
**** List of CAPTURE Hardware Devices ****
card 1: imxaudmix [imx-audmix], device 0: HiFi-AUDMIX-FE (*) []
Subdevices: 1/1
Subdevice #0: subdevice #0
card 2: imxaudiosph0645 [imx-audio-sph0645], device 0: imx-sph0645 snd-soc-dummy-dai-0 []
Subdevices: 1/1
Subdevice #0: subdevice #0
root@imx8qxpaiml:~#
```

- Capture sound using below command,configure above noted card and device number in plughw:<card number>,<device number>.

 # arecord -D hw:2,0 -c 4 -r 48000 -f S16_LE tt.wav

Control + C after 20 sec.

- copy it to host system
  # scp tt.wav username@<IP address>:~/
- Play using audacity utility.

3.17 ML and ARM NN demos

Please refer ML demo user guide,
"**ML_Demos_Guide_iMX8_L5_4_Rel_3_1.docx**"for this section.

4 LIMITATIONS

1. DSI Display not validated with this release 3.1

5 REFERENCES

[1] https://www.arrow.com/en/products/i.imx8x-ai-ml/arrow-development-tools
[2] https://www.nxp.com/docs/en/data-sheet/IMX8QXPIEC-DS.pdf
[3] https://www.nxp.com/webapp/Download?colCode=L5.4.47_2.2.0_LINUX_DOCS