Firmware User Manual
iMX8XML Reference Design

Version	2.0
Status	Baseline
Date	15-July-2019

Contents

Figures

Tables

DOCUMENT DETAILS

➢ Document History

Version	Author		Reviewer		Approver	
	Name	Date (DD-MM-YYYY)	Name	Date (DD-MM-YYYY)	Name	Date (DD-MM-YYYY)
0.1	*Rutvij Trivedi*	*14-Nov-2018*	Prajose John	*14-Nov-2018*	Bhavin Patel	*14-Nov-2018*
0.2	*Darshak Patel*	*27-Nov-2018*	Prajose John	*28-Nov-2018*	Bhavin Patel	*28-Nov-2018*
0.3	*Anil Patel*	*7-Dec-2018*	Prajose John	*7-Dec-2018*	Bhavin Patel	*7-Dec-2018*
0.4	*Sandeep Yenugula*	*22-Feb-19*	Prajose John	*22-Feb-19*	Bhavin Patel	*22-Feb-19*
1.0	*Anil Patel*	*18-Mar-2019*	Prajose John	*18-Mar-2019*	Bhavin Patel	*18-Mar-2019*
1.1	*Anil Patel*	*9-Apr-2019*	Prajose John	*9-Apr-2019*	Bhavin Patel	*9-Apr-2019*
1.2	*Anil Patel*	*24-Apr-2019*	Prajose John	*24-Apr-2019*	Bhavin Patel	*24-Apr-2019*
2.0	*Anil Patel*	*15-July-2019*	Prajose John	*15-July-2019*	Bhavin Patel	*15-July-2019*

Version	Description Of Changes
0.1	*initial draft*
0.2	Added new feature as mentioned in Release note V0.2
0.3	Added new feature and ML demos as mentioned in Release note V0.3
0.4	Initial Beta release for ML and added new demo as mentioned in Release note V0.4
1.0	Baselined version
1.1	Added ARM NN SDK
1.2	Added Steps to build Yocto image with meta-einfochips layer
2.0	Production Baseline Version

Table 1: Documents History

➢ Definition, 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 Nueral Network

Table 2: Definition, Acronyms and Abbreviations

➢ References

No.	Document	Version	Remarks
1	Refer the release note V2.0	2.0	Production Meta Release
2	eI_Arrow_iMX8xML_ML_Demos_User_Guide_v0.1	0.1	ML demos user guide

Table 3: References

Introduction

➢ Purpose of the document

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

➢ About the System

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

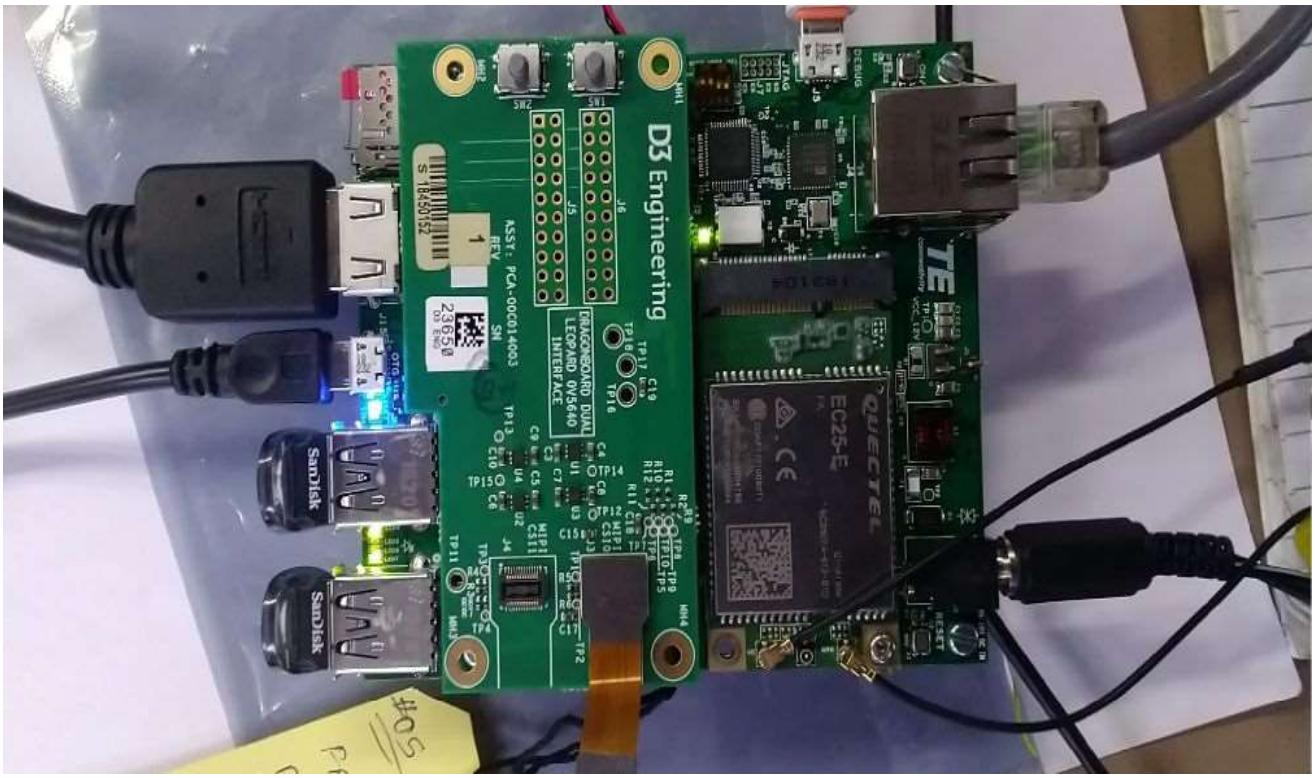

Figure 1: iMX8XML RD AIML board

➢ Pre-requisite

- x86 host system having Linux Ubuntu 16.04 LTS installed
- Basic understanding of Linux commands

➢ Steps to build Yocto Image

We already prepared Meta-layer, which contains all the packages and BSP changes required 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 need to follow below steps:

- Open command prompt (CTRL+ALT+T) and install required packages to build.

$: sudo apt-get install gcc g++ gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat libsdl1.2-dev libsdl1.2-dev xterm sed cvs subversion coreutils texi2html docbook-utils python3-pip python-pip python-pysqlite2 help2man make desktop-file-utils libgl1-mesa-dev libglu1-mesa-dev mercurial autoconf automake groff curl lzop asciidoc u-boot-tools

- Now we need to download new repo for AIML. Currently we are using kernel version 4.14.78-ga release repo.
- To download repo, first we need repo utility. For that need to follow below steps:
  $: mkdir ~/bin
  $: curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
  $: chmod a+x ~/bin/repo
  $: PATH=${PATH}:~/bin
- Now Download Yocto Project environment into local directory
  $: mkdir aiml-yocto-bsp
  $: cd aiml-yocto-bsp
  $: repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-linux-sumo -m imx-4.14.78-1.0.0_machinelearning.xml
  $: repo sync
- Above **repo sync** command will download default NXP source code into our local directory "**aiml-yocto-bsp**".
- Now we need to copy our meta-layer "**meta-einfochips**" into "**sources**" folder.
- "**Meta-einfochips**" contains source code for our both the custom boards AIML and THOR96.
- Therefore, we need to setup our environment based on target machine (board).
- For that we need to follow below steps: (Setup Yocto Build)

  Apply patch **"0001-iMX8XML-Added-BBLAYER-in-local-bblayer.conf-for-AIML.patch"**
  $: cd sources/meta-fsl-bsp-release
  $: git apply ../meta-einfochips/conf/ 0001-iMX8XML-Added-BBLAYER-in-local-bblayer.conf-for-AIML.patch

  In above patch we add three new BBLAYER to our local **bblayers.conf (i.e. meta-webserver, meta-einfochips, meta-imx-machinelearning)**

  Now setup build environments for AIML board.
  $: cd ../../
  $: EULA=1 MACHINE=imx8qxpaiml DISTRO=fsl-imx-xwayland source ./fsl-setup-release.sh -b bld-xwayland-aiml

- After successful setup, we got new build directory **bld-xwayland-aiml,** if it is not there previously. Make sure in build directory "**conf/bblayers.conf"** contains all our required BBLAYERs (which we added through patch).
- Now we are good to go and can build image.

  $: bitbake fsl-image-qt5

- 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 fsl-image-qt5 -c cleanall

  $: bitbake -v fsl-image-qt5 (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.
- After successful build final sd card image reside at below location:
  aiml-yocto-bsp/bld-xwayland-aiml/tmp/deploy/images/imx8qxpaiml/
  Filename should be **fsl-image-qt5-imx8qxpaiml.sdcard.bz2** which is soft link of original build image file **fsl-image-qt5-imx8qxpaiml-<TIMESTAMP>.rootfs.sdcard.bz2**

➢ Download firmware package

- Download the provided SD card (sdcad.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 which should be provided over the share point*
  ***# md5sum <image name>*.sdcard.bz2**
- Extract the provided **.bz2** image using **bunzip2** command, which will take couple of minutes.
  bunzip2 -dkf <image_name>.sdcard.bz2
- Once done, will end with **.sdcard** image in the same directory and can again be verified using **ls -l** command.

➢ Flash the firmware image to SD Card

- Plugin micro SD card into x86 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 **.sdcard** 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>.sdcard 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**

➢ 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.

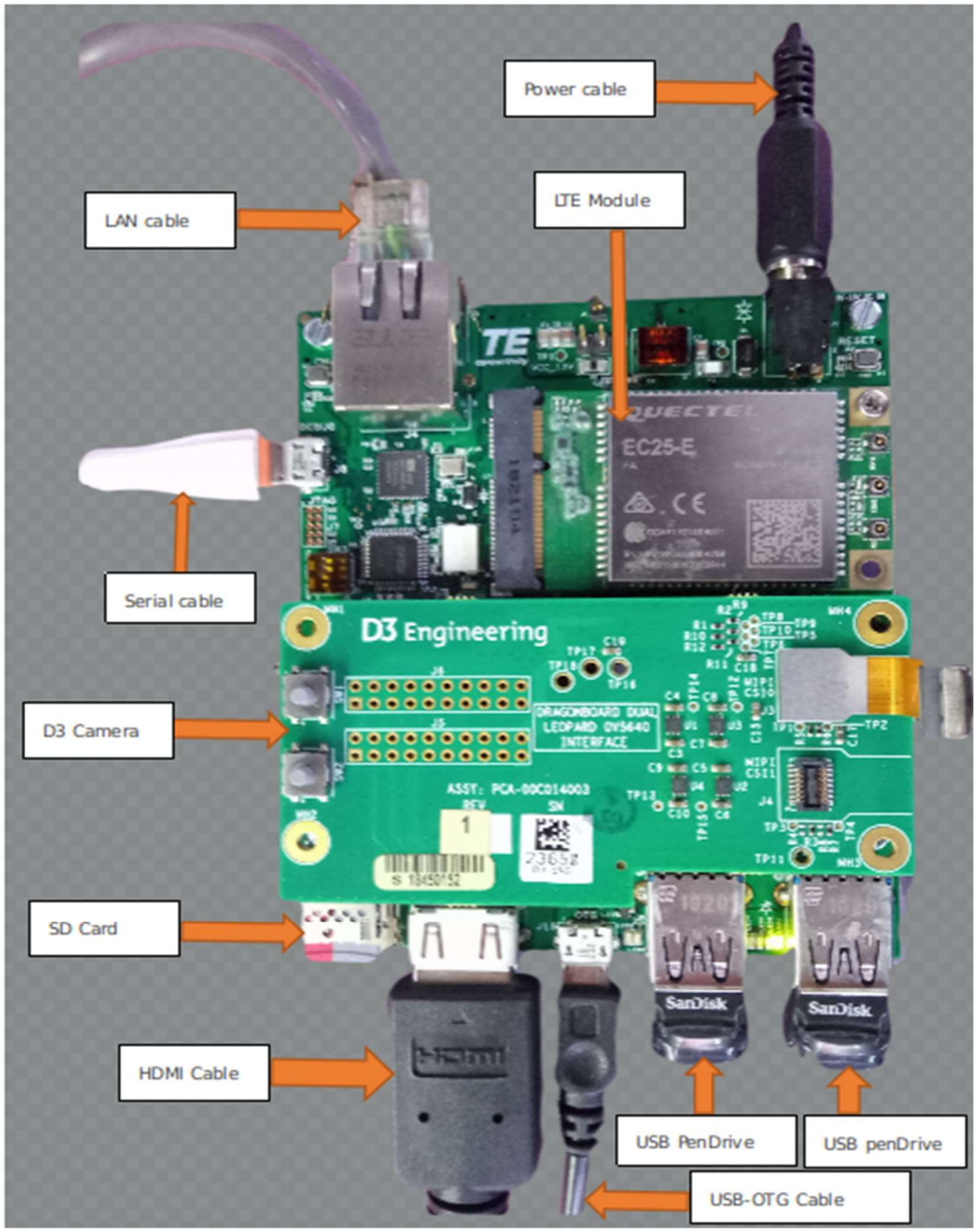

Figure 2: Hardware setup

➢ Open board's terminal- console (minicom) on x86 Host PC

- Ensure SD card is flashed and serial cable is plugged in into board as per mentioned in hardware setup.
- Attached serial cable's USB end to host x86 PC's USB.
- Ensure **minicom** is installed in x86 Ubuntu pc
- Apply below command to open serial command's setting.
  # sudo minicom –s

- **set baud rate and other setting as per below**
- Baud rate 115200, parity none, hardware flow control/software flow control none, serial device /dev/ttyUSB0, **save setup as dfl**.
- Once board gets powered up , terminal will show logs on x86 and one can interact with board using this terminal
- Username for board is “root” without any password.

RUNNING DEMOS

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

- Ping <any server IP>

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

- 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 ./big_buck_bunny.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 big_buck_bunny.mp4 in current directory.
- Be in the board's console and apply below command to play video over HDMI Display
  # gst-launch-1.0 filesrc location=big_buck_bunny_720p_1mb.mp4 ! decodebin name=dec ! videoconvert ! autovideosink dec. ! audioconvert ! audioresample ! alsasink device=plughw:0,0

  Or

  # gplay-1.0 big_buck_bunny_720p_1mb.mp4
- 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. 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 fsl-imx8qxp-aiml-mipi-ov5640.dtb
  # saveenv
  # boot
- Run below command to see preview of camera on HDMI display
  # gst-launch-1.0 v4l2src device=/dev/video0 ! video/x-raw,width=1280,height=720 ! glimagesink
- This will show preview (live) streaming over the attached HDMI.

Capture image from 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 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

4. Wi-Fi Demo

- Open board's console.
- Ensure wlan0 node is prepared.
  #ip link show 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.

  #wpa_supplicant -i wlan0 -c /etc/wpa_supplicant.conf &
  #iw wlan0 link
  #ip address list wlan0
  #udhcpc -i wlan0 -n –q
  #ip address list wlan0
  #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

5. LTE Demo

- Connect Quectel module with target board's as per below image
- 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
- Save above file and apply below command
  # ifconfig ppp0
  # ping www.google.com -I ppp0
- Will be able to ping to google.com

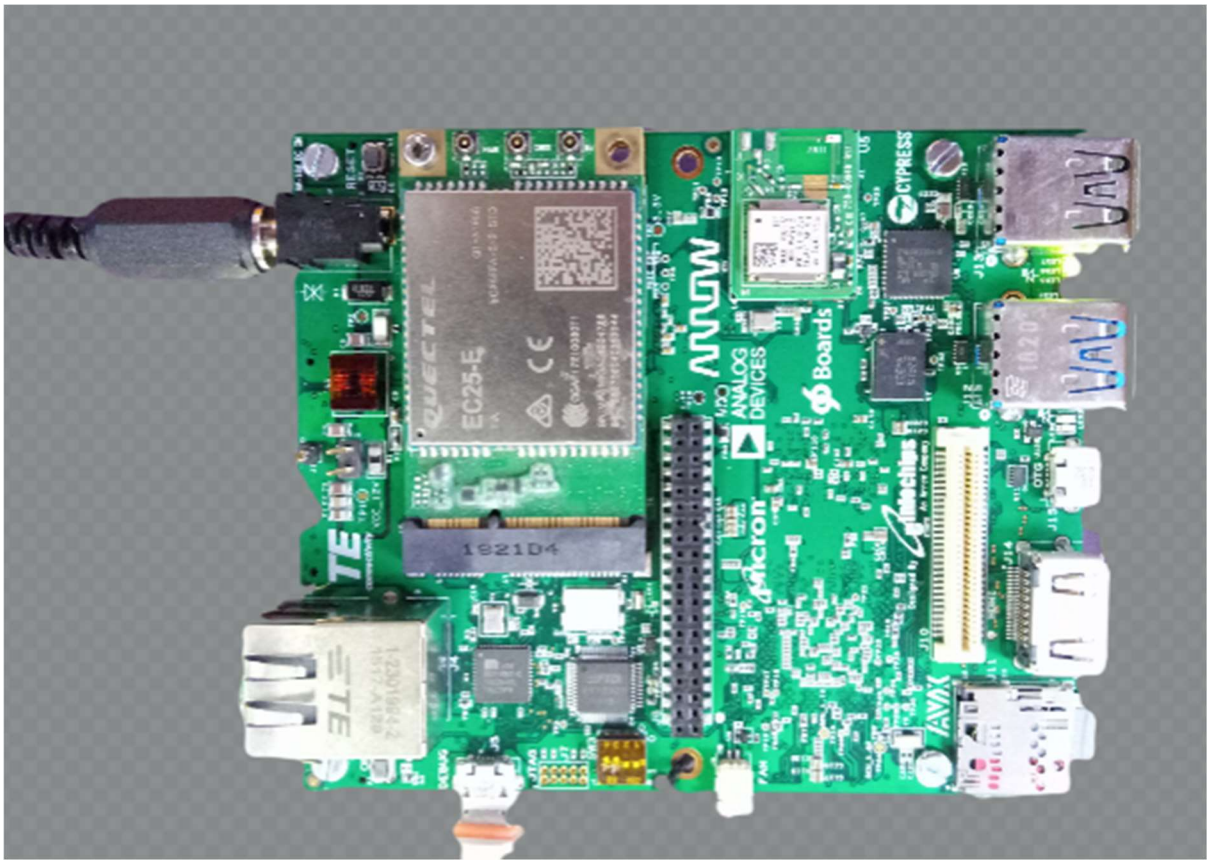

Figure 3: QUECTEL Module on target board

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

7. Bluetooth

- Go to board ‘s console and apply below command
  # brcm_patchram_plus -baudrate 1500000 --patchram /lib/firmware/brcm/4339.hcd --enable_hci --no2bytes --tosleep 1000 /dev/ttyLP0 -d &
- Wait until the complete the command response
  # hciconfig hci0 up
  # hciconfig hci0

- User will get the hcio interface
- 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.
  # aplay -D bluealsa:HCI=hci0,DEV=<mac addr>,PROFILE=a2dp play_audio.wav

- Get the Mobile headset,
- Connect mobile with our modem using above **bluetoothctl** command.
- play the music on mobile player

- run below command to capture the audio from Bluetooth

  # arecord -D bluealsa:HCI=hci0,DEV=<mac addr>,PROFILE=a2dp record_audio.wav

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

8. 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

9. 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.

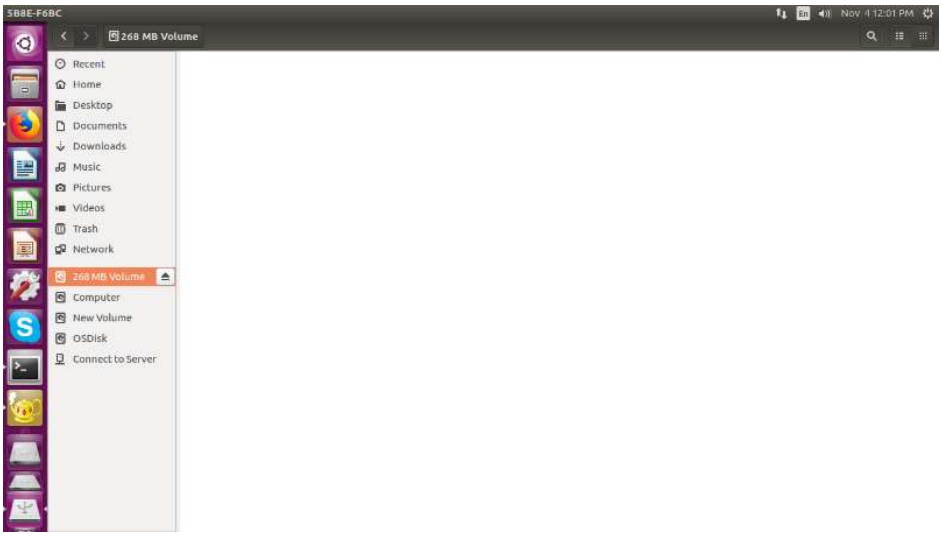

Figure 4: USB OTG as device in Linux

- 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.

10. EEPROM

- Run below command to test EEPROM
- To write in eeprom
  # echo hello_world > /sys/bus/i2c/devices/i2c-1/1-0050/eeprom
- To read from eeprom
  $ cat /sys/bus/i2c/devices/i2c-1/1-0050/eeprom | hexdump -C
- Compare the output will contain data which was written to EEPROM

11. Sensors (Acc/Gyro)

Command to test Accelerometer Sensors

- Note current values of co-ordinates using below command.
  # cat /sys/bus/i2c/devices/1-006a/iio\:device0/in_accel_y_raw
  # cat /sys/bus/i2c/devices/1-006a/iio\:device0/in_accel_y_scale
  # cat /sys/bus/i2c/devices/1-006a/iio\:device0/in_accel_x_raw
  # cat /sys/bus/i2c/devices/1-006a/iio\:device0/in_accel_x_scale
  # cat /sys/bus/i2c/devices/1-006a/iio\:device0/in_accel_z_raw
  # cat /sys/bus/i2c/devices/1-006a/iio\:device0/in_accel_z_scale
- 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/1-006a/iio\:device1/in_anglvel_x_raw
  # cat /sys/bus/i2c/devices/1-006a/iio\:device1/in_anglvel_y_raw
  # cat /sys/bus/i2c/devices/1-006a/iio\:device1/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 1 0x6a 0x11 0x10
  # i2cset -f -y 1 0x6a 0x10 0x10

- Heat the IC using
- Check the value of temperature using below command.
- Check row 20 of i2cdump it should change.
  # i2cdump -f -y 1 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 00 00 00 00 00 00 08 00 69    ...............i
10: 00 10 44 00 00 00 04 00 38 38 00 00 00 00 06 bb    ..D.....88......
20: 44 00 71 f2 9a 1c c6 0f 00 00 00 00 00 00 00 00    D.q?????........
30: 00 00 00 00 00 00 00 00 00 00 00 00 10 00 00 ff    ................
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
50: 00 00 00 00 00 00 00 00 01 00 00 00 00 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    ................
```

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 31

 # echo 499 > /sys/class/gpio/export

 # echo out > /sys/class/gpio/gpio499/direction

 # cat /sys/class/gpio/gpio499/value

 # echo 1 > /sys/class/gpio/gpio499/value

 # echo 0 > /sys/class/gpio/gpio499/value

Pin 27

 # echo 500 > /sys/class/gpio/export

 # echo out > /sys/class/gpio/gpio500/direction

 # cat /sys/class/gpio/gpio500/value

 # echo 1 > /sys/class/gpio/gpio500/value

 # echo 0 > /sys/class/gpio/gpio500/value

Pin 23

 # echo 509 > /sys/class/gpio/export

 # echo out > /sys/class/gpio/gpio509/direction

 # cat /sys/class/gpio/gpio509/value

 # echo 1 > /sys/class/gpio/gpio509/value

 # echo 0 > /sys/class/gpio/gpio509/value

Pin 29

 # echo 448 > /sys/class/gpio/export

 # echo out > /sys/class/gpio/gpio448/direction

 # cat /sys/class/gpio/gpio448/value

 # echo 1 > /sys/class/gpio/gpio448/value

 # echo 0 > /sys/class/gpio/gpio448/value

Pin 33

 # echo 449 > /sys/class/gpio/export

 # echo out > /sys/class/gpio/gpio449/direction

 # cat /sys/class/gpio/gpio449/value

 # echo 1 > /sys/class/gpio/gpio449/value

 # echo 0 > /sys/class/gpio/gpio449/value

Pin 24

 # echo 451 > /sys/class/gpio/export

 # echo out > /sys/class/gpio/gpio451/direction

 # cat /sys/class/gpio/gpio451/value

 # echo 1 > /sys/class/gpio/gpio451/value

 # echo 0 > /sys/class/gpio/gpio451/value

Pin 25

 # echo 455 > /sys/class/gpio/export

 # echo out > /sys/class/gpio/gpio455/direction

 # cat /sys/class/gpio/gpio455/value

 # echo 1 > /sys/class/gpio/gpio455/value

 # echo 0 > /sys/class/gpio/gpio455/value

Pin 30

 # echo 461 > /sys/class/gpio/export

 # echo out > /sys/class/gpio/gpio461/direction

 # cat /sys/class/gpio/gpio461/value

 # echo 1 > /sys/class/gpio/gpio461/value

 # echo 0 > /sys/class/gpio/gpio461/value

Pin 32

```
# echo 462 > /sys/class/gpio/export
# echo out > /sys/class/gpio/gpio462/direction
# cat /sys/class/gpio/gpio462/value
# echo 1 > /sys/class/gpio/gpio462/value
# echo 0 > /sys/class/gpio/gpio462/value
```

Pin 26

```
# echo 479 > /sys/class/gpio/export
# echo out > /sys/class/gpio/gpio479/direction
# cat /sys/class/gpio/gpio479/value
# echo 1 > /sys/class/gpio/gpio479/value
# echo 0 > /sys/class/gpio/gpio479/value
```

Pin 28

```
# echo 416 > /sys/class/gpio/export
# echo out > /sys/class/gpio/gpio416/direction
# cat /sys/class/gpio/gpio416/value
# echo 1 > /sys/class/gpio/gpio416/value
# echo 0 > /sys/class/gpio/gpio416/value
```

Pin 34

```
# echo 391 > /sys/class/gpio/export
# echo out > /sys/class/gpio/gpio391/direction
# cat /sys/class/gpio/gpio391/value
# echo 1 > /sys/class/gpio/gpio391/value
# echo 0 > /sys/class/gpio/gpio391/value
```

13. High Speed Expansion

- NXP DSI connected and validated the DSI display interface

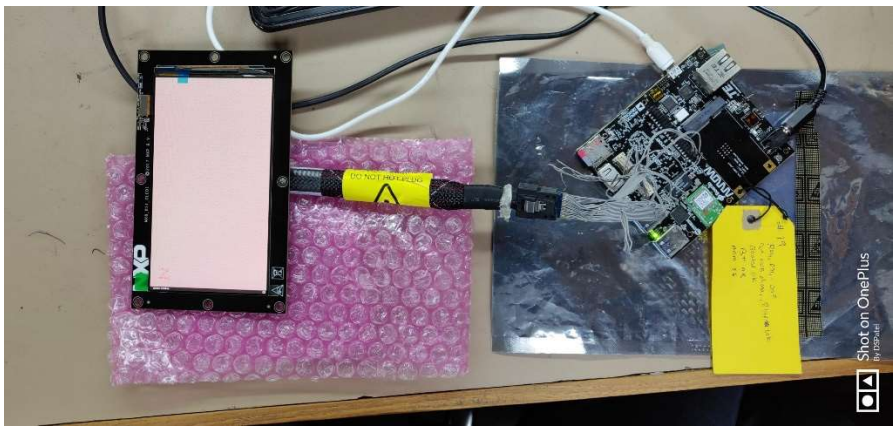

Figure 5: High Speed Expansion

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

14. USER LED

- Run the below command to control the Led

LED1

 # echo 368 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio368/direction
 # echo 1 > /sys/class/gpio/gpio368/value

 # echo 0 > /sys/class/gpio/gpio368/value

LED2

 # echo 486 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio486/direction
 # echo 1 > /sys/class/gpio/gpio486/value

 # echo 0 > /sys/class/gpio/gpio486/value

LED3

#echo 487 > /sys/class/gpio/export
#echo out > /sys/class/gpio/gpio487/direction
#echo 1 > /sys/class/gpio/gpio487/value

#echo 0 > /sys/class/gpio/gpio487/value

LED5

 # echo 369 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio369/direction
 # echo 1 > /sys/class/gpio/gpio369/value

 # echo 0 > /sys/class/gpio/gpio369/value

LED4

 # echo 373 > /sys/class/gpio/export
 # echo out > /sys/class/gpio/gpio373/direction
 # echo 1 > /sys/class/gpio/gpio373/value

 # echo 0 > /sys/class/gpio/gpio373/value

LED6

```
# echo 370 > /sys/class/gpio/export
# echo out > /sys/class/gpio/gpio370/direction
# echo 1 > /sys/class/gpio/gpio370/value

# echo 0 > /sys/class/gpio/gpio370/value
```

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.

16. DMIC demo

- Record a wav format file using below command.
  # arecord -D hw:1,0 -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.

17. ML and ARM NN demos

Please refer ML demo user guide for this section.

KNOWN ISSUES AND LIMITATIONS

- Please refer the Release note V2.0

CONTACT US

For any queries related to product, please contact us at arrow.imx8Xml@einfochips.com