MYD-JX8MX

LinuxDevelopmentGuide

MYiR™ Make Your Idea Real

Table of Contents

MYD-JX8MX Linux Development Guide

Introduction

This document aim to introduce how to setup environment,compile source code ,install application,use the hardware API , develop QT application on MYD-JX8MX series development boards.

History

Version	Note	Date
V1.0	Initial Version	2019.03.29
V1.1	1.Distinguish DDR size of uboot 2.Modify the update method with SD card	2019.06.02
V1.2	1.Correct the linux-imx-src.inc path 2.Use Fn-link6222bmodule	2019.08.12

Hardware Version

This document applies to MYD-JX8MX series Development Board of MYIR currently.

1. Software resources

The MYD-JX8MX Development Board offers linux system which kernel version is 4.9.88. It provides rich system resources and software resources.Some of the functions have to be working with the extension modules.

A list of resources as bellow：

Type	Name	Description	Source Code
Bootstrap	U-boot	bootstrap	YES
Linux kernel	Image	base on imx_4.9.88_2.0.0_ga of NXP	YES
Device Driver	PMIC	BD71873 driver	YES
Device Driver	USB Host	USB Host 3.driver	YES
Device Driver	USB OTG	USB OTG 3.driver	YES
Device Driver	I2C	I2C bus driver	YES
Device Driver	SPI	SPI bus driver	YES
Device Driver	Ethernet	10M/100M/1000M driver	YES
Device Driver	MMC	MMC/EMMC/TFcard stoage driver	YES
Device Driver	HDMI	HDMI display driver	YES
Device Driver	LCD	MIPI-LVDS driver	YES
Device Driver	PWM	PWM control driver	YES
Device Driver	RTC	Real Time Clock driver	YES
Device Driver	GPIO	GPIO driver	YES
Device Driver	Touch	Touch of Capactive driver	YES
Device Driver	Audio	WM8904 driver	YES
Device Driver	Camera	Ov5640 driver	YES
Device Driver	WiFi & BT	QCA6174 driver	YES
Device Driver	Watchdog	Watchdog driver	YES
Device Driver	LTE module	Support EC20 module，use USB driver	YES
Device Driver	M.2	NVME driver	YES
Rootfs	Yocto rootfs	Base on Yocto `s xwayland rootfs	YES
Application	GPIO KEY/LED	GPIO button and led appication	YES
Application	NET	TCP/IP Sokect C/S appication	YES
Application	RTC	Real Time Clock application	YES
Application	Audio	Audio application	YES
Application	LCD	Dispaly screen application	YES
Application	Camera	Multiple cameras application	YES
toolchain	Cross compiler	Yocto GCC 7.3.0 Hardfloat	YES

2. Deploy development environment

To get the Yocto Project expected behavior in a Linux Host Machine,it is recommonded to use the Ubuntu 16.04 64bit distribution. The DDR size should be larger than 8GB and the hard disk space requird at least 150G(500GB,recommended).And then connect and configure the network，some packages should be installed.

Repalce the Ubuntu sources-list

Notice that this only applies to the developer in china. If you are in china ,the tuna resource should instead of the default.

The website is as follows：

https://mirrors.tuna.tsinghua.edu.cn/help/ubuntu/
Choose 16.04 ，and then copy the contents to update /etc/apt/source.list .

Operating procedures ：

1.Backup the default file

```
cp /etc/apt/sources.list /etc/apt/sources.list-bak
```

2.Copy contents from the tuna 16.04 column to /etc/apt/sources.list file.

3.Update source

```
apt-get update
```

PS: If error happens to appstream3 ,you can remove the package first and then update.

```

apt-get remove libappstream3

apt-get update
```

Host packages

Essential Yocto Project host packages are:

```
sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat libsd
l1.2-dev u-boot-tools

sudo apt-get install libsdl1.2-dev xterm sed cvs subversion coreutils texi2html docbook-utils python-pysqlite2
help2man make gcc g++ desktop-file-utils  libgl1-mesa-dev libglu1-mesa-dev mercurial autoconf automake groff cu
rl lzop asciidoc
```

Set repo file

The repo file is used to download resources, and it is located in 03-Tools/Repo for chinese, it should be installed on your computer by performing these step:

```
mkdir ~/bin
cp ~/03-Tools/Repo/repo ~/bin
chmod a+x ~/bin/repo
export PATH=~/bin:$PATH
```

Please follow below steps instead of above ,if you are able to access google.

```
mkdir ~/bin
curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
chmod a+x ~/bin/repo
export PATH=~/bin:$PATH
```

Configure git

Make sure that git is set up properly with the commands below.

```
git config --global user.name "abc123"
git config --global user.email "def456@gmail.com"
git config --list
```

3. Yocto build system

There are many open-source system framework based on Linux platform. Developers can build the system and customize development easily.At present, Buildroot,Yocto ,openEmbedded are widely used. Here,Yocto is suggested to be used to build a Linux system for embedded products. Yocto is not only a tool to build rootfs system but also to provide a complete set of Linux-based development and maintenance workflows,enabling the underlying embedded developers and upper-level application developers to develop under a unified framework.

Yocto is an open source "umbrella" project, which means there are many sub-projects under it. Yocto just integrates all the projects together and provides a reference build project Poky to guide developers on how to apply these projects. It contains Bitbake, OpenEmbedded-Core, board support packages, and configuration files for various packages. With Poky, you can build systems with different types of requirements, such as the smallest system coreimage-minimal, the full-featured command line system core-image-base, and the fsl-image-qt5 with Qt5 graphics library.

This chapter mainly introduces the MYD-JX8MX development board, builds the system using yocto, and compiles the image.

The MYD-JX8MX Linux system consists of the following sections:

- imx-boot: consists of SPL，U-boot ，ARM trust firmware and HDMI firmware .Bootstrap support EMMC and SD card.
- Linux Kernel: 4.9.88 linux kernel with many peripheral resources.
- device tree file：associated with hardware configuration.
- rootfs： rootfs system.

3.1 Yocto compile complete image

This chapter mainly introduces how to use yocto to compile the complete image.

Decompress the yocto packages which store in the 04-Source directory:fsl-release-yocto.tar.gz and compile image use fsl-image-qt5-validation-imx as target,perform as follows:

```
tar -zxvf  fsl-release-yocto.tar.gz
cd fsl-release-yocto
DISTRO=fsl-imx-xwayland MACHINE=imx8mqevk source fsl-setup-release.sh -b build-xwayland
bitbake fsl-image-qt5-validation-imx
```

The generated images would be located in below direction when it compiled.

${WORK_DIR}/build-xwayland/tmp/deploy/images/imx8mqevk

Introduce some important resources：

Name	Function
imx-boot-imx8mqevk-sd.bin	bootstrap image
Image	kernel
myd-fsl-imx8mq-evk.dtb	dtb configure file
fsl-image-qt5-validation-imx-imx8mqevk-20190401035300.rootfs.ext4	the rootfs system with EXT4 format
fsl-image-qt5-validation-imx-imx8mqevk-20190401035300.rootfs.tar.bz2	the rootfs use tar -jxf to compress
fsl-image-qt5-validation-imx-imx8mqevk-20190401035300.rootfs.sdcard.bz2	the compete images consist of imx-uboot+dtb+kernel+rootfs

Besides the target of fsl-image-qt5-validation-imx，there are some other ones to be chosen from below：

Target Nanem	feature
core-image-minimal	the minimal rootfs system
fsl-image-validation-imx	the image with gui without any QT content
fsl-image-qt5-validation-imx	the image with qt5

3.2 Yocto compile U-Boot

The I.mx8m bootstrap which named imx-boot is consist of u-boot,SPL,ARM trust firmware and HDMI firmware.

Configure the DDR size

There are two different type of memory for configure，1G DDR or 2G DDR.

Determined by this file ：

sources/meta-fsl-bsp-release/imx/meta-bsp/recipes-bsp/u-boot/u-boot-imx_2017.03.bb

Main content as bellow:

```
UBOOT_SRC ?= "git:///${PWD}/../uboot-imx;protocol=file"
SRCBRANCH = "imx_v2017.03_4.9.88_2.0.0_ga"
SRC_URI = "${UBOOT_SRC};branch=${SRCBRANCH}"

SRCREV = "947e79fa7b49ae66ba7ae8edde094d7e4d433df4"
```

You can find the name of uboot source code directory called uboot-imx,branch name is imx_v2017.03_4.9.88_2.0.0_ga and git commit value is 947e79fa7b49ae66ba7ae8edde094d7e4d433df4 .

Choose the requested branch and commit value can set the correct ddr configure.

View branch command as bellow：

```
duxy@myir:~/linux_8m_software/myd-j8mx/04-Source/fsl-release-yocto/uboot-imx$ git branch
* imx_v2017.03_4.9.88_2.0.0_ga
  imx_v2017.03_4.9.88_2.0.0_ga-2g-ddr
```

Checkout branch command as below:

```
git checkout imx_v2017.03_4.9.88_2.0.0_ga-2g-ddr
```

view the git commit value command as below:

```
git log
```

After configured the file, we can compile the imx-boot by performing below step:

```
   cd fsl-release-yocto
   . ./setup-environment build-xwayland
   bitbake -c cleansstate u-boot
   bitbake -c compile u-boot -f
   bitbake -c deploy u-boot
   bitbake imx-boot
```

The generated images would be located in below direction when it compiled:

build-xwayland/tmp/deploy/images/imx8mqevk/imx-boot-imx8mqevk-sd.bin

3.3 Yocto compile Kernel

The kernel is the second stage after imx-uboot launch,it can be compiled by performing the following steps:

```
. ./setup-environment build-xwayland
  bitbake -c cleansstate virtual/kernel
  bitbake -c patch virtual/kernel
  bitbake -c compile virtual/kernel -f
```

The generated images would be located in below direction when it compiled:

build-xwayland/tmp/work/imx8mqevk-poky-linux/linux-imx/4.9.88-r0/build/arch/arm64/boot/Image

Kernel is same as uboot to set the configuration through the file: sources/meta-fsl-bsp-release/imx/meta-bsp/recipes-kernel/linux/linux-imx-src.inc

3.4 Yocto generate SDK tools

This chapter mainly introduces how to use Yocto to generate SDK toolchain and use it to compile applications which can run on development board.

Download toolchain

In charter 3.1 you had obtained the decompression of yocto files ，so perform the following steps：

```
cd fsl-release-yocto
. ./setup-environment build-xwayland
bitbake meta-toolchain
```

After download completed, the resources are located in /temp/deploy/sdk.

```
.
├── fsl-imx-xwayland-glibc-x86_64-meta-toolchain-aarch64-toolchain-4.9.88-2.0.0.host.manifest
├── fsl-imx-xwayland-glibc-x86_64-meta-toolchain-aarch64-toolchain-4.9.88-2.0.0.sh
├── fsl-imx-xwayland-glibc-x86_64-meta-toolchain-aarch64-toolchain-4.9.88-2.0.0.target.manifest
├── fsl-imx-xwayland-glibc-x86_64-meta-toolchain-aarch64-toolchain-4.9.88-2.0.0.testdata.json
```

Install to toolchain

Perform the following step:

```
./fsl-imx-xwayland-glibc-x86_64-meta-toolchain-aarch64-toolchain-4.9.88-2.0.0.sh
```

Result:

```
NXP i.MX Release Distro SDK installer version 4.9.88-2.0.0
==========================================================
Enter target directory for SDK (default: /opt/fsl-imx-xwayland/4.9.88-2.0.0): ~/opt/fsl-imx-xwayland/4.9.88-2.0
.0
You are about to install the SDK to "/home/duxy/opt/fsl-imx-xwayland/4.9.88-2.0.0". Proceed[Y/n]? y
Extracting SDK.......................done
Setting it up...done
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to source the environment setup script e.g.
 $ . /home/duxy/opt/fsl-imx-xwayland/4.9.88-2.0.0/environment-setup-aarch64-poky-linux
```

Compile the executable application on board

Perform the following step:

```
source /home/duxy/opt_test/environment-setup-aarch64-poky-linux
$CC main.c -o main
```

Use the file command to check main and view the result:

```
#file main
main: ELF 64-bit LSB executable, ARM aarch64, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-a
arch64.so.1, for GNU/Linux 3.14.0, BuildID[sha1]=75b922974f533df4107a5cd413cb7e7667335e94, not stripped
```

The above toolchain has been provided in 03-Tools/Toolchain. QT toolchain is also provided.

More detail information of yocto, please refer to the following link:

Yocto Project Quick start

Bitback User Manual

Yocto Project Reference Manual

Yocto Project Development Manual

Yocto Project Complete Documentation Set

4. Linux application development

This chapter mainly introduces how to use sample to test MYD-JX8MX peripheral hardware. First of all ,host pc should install the SDK toolchain provided by yocto, if not installed of the toolchain yet, please refer to chapter 3.4. Second ,compile the source code of sample and copy them to the development board.

Compile application

If the toolchain has been installed, we should make sure the gcc version is right.

```
source ~/opt/fsl-imx-xwayland/4.9.88-2.0.0/environment-setup-aarch64-poky-linux
aarch64-poky-linux-gcc --version
aarch64-poky-linux-gcc (GCC) 7.3.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Compile the source code of sample by performing the following step:

```
cd 04-Sources
tar -zxvf example.tar.gz
cd example
make
```

Finally copy the entire example contents to the USB disk and insert into the board use for later testing.

4.1 GPIO-KEY Test

This example demonstrates how to read key status and key values in the Linux user space. After executed the sample ,press or release K3 button . It will output information in the serial port. Press "ctrl+c" to exit the sample.

Hardware connection

Insert the USB disk into the development board,which stored the example in the 4 chapters.

Software test

Execute the sample on the terminal of the development board, and then press the K3 button to print the key value use pressed or released, as follows:

```
cd /run/media/sda/example/gpio_key
./gpio_key /dev/input/event1
Hit any key on board ......
key 2 Pressed
key 2 Released
key 2 Pressed
key 2 Released
```

4.2 GPIO-LED Test

This example demonstrates how to control the D49 and D50 led . After executed the sample , the D49 and D50 led will blink . Press "ctrl+c" to exit the sample.

Hardware connection

Insert the USB disk into the development board,which stored the example in the 4 chapters.

Software test

Execute the sample on the control terminal of the development board , the led blink.

```
./gpio_led   /sys/class/leds/user/brightness   /sys/class/leds/cpu/brightness
```

4.3 USB-HOST Test

Insert the USB disk into the USB HOST J6 or J7 interface, the debug serial port will output information. And then mount the device for writing or reading test.

Hardware connection

Insert the USB disk into the development board,which stored the example in the 4 chapters.

Software test

When the USB disk insert into the board, it appears some information as follows:

```
#  usb 1-1.3: new high-speed USB device number 5 using xhci-hcd
 usb-storage 1-1.3:1.0: USB Mass Storage device detected
 scsi host0: usb-storage 1-1.3:1.0
 scsi 0:0:0:0: Direct-Access     Generic  Flash Disk       8.07 PQ: 0 ANSI: 4
 sd 0:0:0:0: [sda] 31129600 512-byte logical blocks: (15.9 GB/14.8 GiB)
 sd 0:0:0:0: [sda] Write Protect is off
 sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
  sda:
 sd 0:0:0:0: [sda] Attached SCSI removable disk
 FAT-fs (sda): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
```

The system mount the devices automaticly, view the /proc/mounts contents can find the mount point.

```
 # cat /proc/mounts  | grep sda
/dev/sda /run/media/sda vfat rw,relatime,gid=6,fmask=0007,dmask=0007,allow_utime=0020,codepage=437,iocharset=is
o8859-1,shortname=mixed,errors=remount-ro 0 0
```

We can write and read a file in the mount point and view the result.

```
cd /run/media/sda/
echo "Hello" > Hello.txt
cat Hello.txt
Hello
```

4.4 USB-DEVICE Test

This example demonstrates how to use TypeC J8 interface as device module on the development baord, it uses file or memory as device and connect to othear USB HOST interface. The sample use file as storage device for USB HOST interface.

Hardware connection

Use a usb-typec line，one end with USB connected to PC， the othor one with type-c connected to J8 interface on board.

Software test

1.First step to create a file that size of 6M, perform step below:

```
# dd if=/dev/zero of=/home/root/typec_device bs=1M count=6;
```

2.Secondly ,format it using vfat format,perform step below:

```
# mkfs.vfat /home/root/typec_device;
mkfs.fat 4.1 (2017-01-24)
```

3.Finally simulate the file as a storage device, perform step below:

```
# modprobe g_mass_storage file=/home/root/typec_device removable=1 iSerialNumber="1234";
```

The development board output these information.

```
 Mass Storage Function, version: 2009/09/11
 LUN: removable file: (no medium)
 LUN: removable file: /home/root/typec_device
 Number of LUNs=1
 g_mass_storage gadget: Mass Storage Gadget, version: 2009/09/11
 g_mass_storage gadget: g_mass_storage ready
 g_mass_storage gadget: super-speed config #1: Linux File-Backed Storage
```

4.The linux PC detect USB disk inserted ,the SerialNumber is "1234" and the capacity is 6M:

```
[ 6892.908437] usb 3-2: SerialNumber: 1234
[ 6893.178564] usb-storage 3-2:1.0: USB Mass Storage device detected
[ 6893.178771] usb-storage 3-2:1.0: Quirks match for vid 0525 pid a4a5: 10000
[ 6893.178832] scsi host4: usb-storage 3-2:1.0
[ 6893.179045] usbcore: registered new interface driver usb-storage
[ 6893.202270] usbcore: registered new interface driver uas
[ 6894.178633] scsi 4:0:0:0: Direct-Access     Linux    File-Stor Gadget 0409 PQ: 0 ANSI: 2
[ 6894.179747] sd 4:0:0:0: Attached scsi generic sg2 type 0
[ 6894.184176] sd 4:0:0:0: [sdb] 12288 512-byte logical blocks: (6.29 MB/6.00 MiB)
[ 6894.292769] sd 4:0:0:0: [sdb] Write Protect is off
[ 6894.292786] sd 4:0:0:0: [sdb] Mode Sense: 0f 00 00 00
[ 6894.400842] sd 4:0:0:0: [sdb] Write cache: enabled, read cache: enabled, doesn't support DPO or FUA
[ 6894.621878]  sdb:
[ 6894.848615] sd 4:0:0:0: [sdb] Attached SCSI removable disk
```

4.5 Ethernet Test

This sample uses Linux socket API to implement simple C/S structure. Two programs communicate via the TCP/IP stack. The pc endpoint uses pc_server as service and the development board uses arm_client as clinet to communicate.

Hardware connection

Insert the USB disk into the development board,which stored the example in the 4 chapters.Connect the linux pc and development board with net cable.

Software test

Place the program in the linux computer`s home directory,which completed in example/network/pc_server directory. Set a static IP as 192.168.30.3 and start the service，perform below step：

```
ifconfig eth0 192.168.30.3
./pc_server
```

As a client， the development board also need to set a static IP as 192.168.30.133， then start the arm_client program，perform below step：

```
ifconfig eth0 192.168.30.133
./arm_client 192.168.30.3
form server: Make Your idea Real!
```

The linux computer will output below:

```
REC FROM： 192.168.30.133
```

4.6 Audio Test

This example demonstrates how to use arecord command to record audio by line in interface and use aplay to play audio through headphones.

Hardware connection

Need to connect the computer`s headphones and development board LINE-IN(J10) interface with 3.5mm audio AUX cable. The HEADERPHONE of J9 interface connect to a earphone .

Software test

Before excuting arecord command to record audio, the computer should play a audio file first. Press ctrl +c to exit after record about one minute.

```
# arecord -f cd test.wav
```

To excute aplay command to play the recorded audio file.

```
# aplay test.wav
```

4.7 Serial port Test

This example demonstrates how to test the serial port with different baund rates. There are three serial ports can be used:

Serial port	Associate device
J18	ttymxc0 (default serial port)
J19	ttymxc1
J21	ttymxc3

Hardware connection

Insert the USB disk into the development board,which stored the example in the 4 chapters. Connect the J19 interface with a serial cable.

Software test

Set the device of ttymxc1 baund rate to 115200 and send "hello" words.

```
cd /run/media/sda/example/uart_test
./uart_test -d /dev/ttymxc0 -b 115200 -s hello
```

Use the serial tool to configure ,detail parameter as follows :

- baund rate：115200
- data bits: 8bit
- parity：None
- stop bits：1bit
- flow control：Disable

The serial port output:

```
hellohellohello
```

4.8 SPI Test

This example demonstrates how to use J17 interface to test SPI feature.

Hardware connection

Insert the USB disk into the development board,which stored the example in the 4 chapters.Connect the linux pc and development board with net cable.

The SI and SO pins should be connected.

Software test

Excute the spi_test program to test ,which located in example/spi. Perform below step:

```
cd /run/media/sda/example/spi
./spi_test
```

The result as follows:

```
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 KHz)
r_buf[0] = 0
r_buf[1] = 1
r_buf[2] = 2
r_buf[3] = 3
r_buf[4] = 4
r_buf[5] = 5
r_buf[6] = 6
r_buf[7] = 7
r_buf[8] = 8
r_buf[9] = 9
```

4.9 WIFI Test

The MYD-JX8MX development carries two WiFi/BT module , one is 82748-PR, the othear one is 6222B-PRB.

The wifi module of 6222B-PRB is RTL8822B, the module name of 8274B-PR is QCA6714.

This charpter use Client mode to connenct wifi,and then to ping the website.

Hardware connection

It is necessary to connect an antenna to U6 .

Software test

The WiFi module is treated as a clinet device in Client mode.It can connect othear routers.

- View the wifi status

The kernel had been loaded the wifi module drivers automaticly when the system power-on. If the driver loaded successfuly, it will rename wlan0 to wlp1s0, use ifconfig command to confirm.

```
ifconfig wlp1s0
wlp1s0    Link encap:Ethernet  HWaddr 80:5e:4f:b3:a5:20
          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)
```

rfkill command can view the wlan`s status ，if it was blocked，turn on （unblock） it. perform as follows:

```
rfkill list
0: phy0: wlan
        Soft blocked: yes
        Hard blocked: no
```

- Turn on the wlan

The rfkill unblock 0 or rfkill unblock wlan command can unblock the wlan.

```

rfkill unblock wlan
rfkill list
0: phy0: wlan
        Soft blocked: no
        Hard blocked: no
```

- Set wifi SSID name and passwd

The wifi SSID name and passwd should write to the /etc/wpa_supplicant.conf file . The rootfs system provides a wpa_passphrase command to do it . Now we demonstrate a sample to connect a wifi name MYIR_TECH and passwd is myir@2016 as below:

```
head -n 4 /etc/wpa_supplicant.conf > /etc/wpa_supplicant.conf.tmp
wpa_passphrase MYIR_TECH myir@2016 >> /etc/wpa_supplicant.conf.tmp
mv /etc/wpa_supplicant.conf /etc/wpa_supplicant.conf.bak
mv /etc/wpa_supplicant.conf.tmp /etc/wpa_supplicant.conf
```

- Kill wpa_supplicant process

```
killall wpa_supplicant
```

- Connect wifi

After setting SSID name and passwd,it is time to connect wifi and obtain the ip address. Perform below step:

```
wpa_supplicant -B -i wlp1s0 -c /etc/wpa_supplicant.conf -D nl80211
```

Obtain the ip address:

```
udhcpc -i wlp1s0
```

Above two commands excute like this：

```
#wpa_supplicant -B -i wlp1s0 -c /etc/wpa_supplicant.conf -D nl80211

Successfully initialized wpa_supplicant
# udhcpc -i wlp1s0
udhcpc (v1.24.1) started
\wlp1s0: authenticate with 30:fc:68:9a:e8:99
Sending discover...
wlp1s0: send auth to 30:fc:68:9a:e8:99 (try 1/3)
wlp1s0: authenticated
wlp1s0: associate with 30:fc:68:9a:e8:99 (try 1/3)
wlp1s0: RX AssocResp from 30:fc:68:9a:e8:99 (capab=0x431 status=0 aid=2)
wlp1s0: associated
IPv6: ADDRCONF(NETDEV_CHANGE): wlp1s0: link becomes ready
Sending discover...
Sending select for 192.168.40.107...
Lease of 192.168.40.107 obtained, lease time 7200
/etc/udhcpc.d/50default: Adding DNS 223.5.5.5
/etc/udhcpc.d/50default: Adding DNS 201.104.111.114
```

- Test the network

Checking the network whether connected by ping the baidu.

```
# ping www.baidu.com
PING www.baidu.com (61.135.169.121) 56(84) bytes of data.
64 bytes from 61.135.169.121: icmp_seq=1 ttl=55 time=22.6 ms
64 bytes from 61.135.169.121: icmp_seq=2 ttl=55 time=27.9 ms
```

4.10 BT Test

The MYD-JX8MX development carries a WiFi/BT module of 82748-PR(U7)， the part number of wifi module is QCA6714. The BT is a serial module that connect to ttymxc2,this charpter will connect the pc endpoint BT device with development board.

Hardware connection

It is necessary to connect an antenna to U5 avoid the poor signal.

Software test

- Attach the serial port

Use hciattach command to attach ttymxc2 serial port, perform below step:

```
#hciattach /dev/ttymxc2 any 115200 -t120 flow
Setting TTY to N_HCI line discipline
Device setup complete
```

Please note: if the wifi module on the baseboard is Fn-link 6222B, the command should be replaced as follow:

```
rtk_hciattach -n -s 115200 ttymxc2 rtk_h5 &
```

- Turn on wlan

The rfkill unblock 0 or rfkill unblock wlan command can unblock the wlan. Before unblock you should block it first.

```
#rfkill list
0: phy0: wlan
        Soft blocked: no
        Hard blocked: no
1: hci0: bluetooth
        Soft blocked: yes
        Hard blocked: no

#rfkill block wlan
#rfkill unblock wlan
# rfkill list
0: phy0: wlan
        Soft blocked: no
        Hard blocked: no
1: hci0: bluetooth
        Soft blocked: no
        Hard blocked: no
```

- Connect BT

bluetoothctl command can control the BT device to scan or connect.

Function	Command
scan	scan on
view the scan devices	devices
pair	pair target_mac
connenct	connect target_mac

The below step demostrate how to connect BT:

```
# bluetoothctl
Agent registered
[bluetooth]# scan on
Discovery started
[CHG] Controller 80:5E:4F:B3:CC:47 Discovering: yes
[CHG] Device B0:FC:36:3B:CF:0E RSSI: -100
[CHG] Device B0:FC:36:3B:CF:0E TxPower: 0
[bluetooth]# devices
Device B0:FC:36:3B:CF:0E DESKTOP-5SG2HL8
[bluetooth]# pair B0:FC:36:3B:CF:0E
Attempting to pair with B0:FC:36:3B:CF:0E
[CHG] Device B0:FC:36:3B:CF:0E Connected: yes
Request confirmation
[agent] Confirm passkey 903031 (yes/no): yes
[DESKTOP-5SG2HL8]#
```

4.11 4G Test

The MYD-JX8MX development board provides a MINI PCI-E slot for 4G module，this slot uses USB data pin to communicate with 4G module. We support EC20 driver as reference.

Hardware connection

- Insert the EC20 module into the PCI-E U29 slot
- Connect an antenna to EC20 module
- Insert the SIM card into J25 slot
- Insert usb disk stored the 04-source/ppp.tar.gz on the root

Software test

- Check the 4G module

The kernel has added the 4G protocol, when the system power on , under the /dev directory you can see the ttyUSB* nodes.

```
# ls -l /dev/ttyUSB*
crw-rw---- 1 root dialout 188, 0 Apr  3 12:07 /dev/ttyUSB0
crw-rw---- 1 root dialout 188, 1 Apr  3 12:07 /dev/ttyUSB1
crw-rw---- 1 root dialout 188, 2 Apr  3 12:07 /dev/ttyUSB2
crw-rw---- 1 root dialout 188, 3 Apr  3 12:07 /dev/ttyUSB3
```

- Copy the scripts of 4G to the rootfs

Firstly, decompress the ppp.tar.gz and then copy the ppp directory to the rootfs root path.

```
cd /run/media/sda/
tar -zxvf ppp.tar.gz
cd ppp
tree
.
├── etc
│   └── ppp
│       ├── ip-up
│       └── peers
│           ├── quectel-chat-connect
│           ├── quectel-chat-disconnect
│           └── quectel-ppp
└── sbin
    ├── chat
    └── pppd

4 directories, 6 files

cp * / -rf
```

- Dial

Perform this command ：pppd call quectel-ppp &

```
# pppd call quectel-ppp &
```

Debug info as below ：

```
[1] 3325
root@imx8mqevk:~# pppd options in effect:
debug           # (from /etc/ppp/peers/quectel-ppp)
nodetach                # (from /etc/ppp/peers/quectel-ppp)
```

```
dump            # (from /etc/ppp/peers/quectel-ppp)
noauth          # (from /etc/ppp/peers/quectel-ppp)
user test               # (from /etc/ppp/peers/quectel-ppp)
password ??????         # (from /etc/ppp/peers/quectel-ppp)
remotename 3gppp                # (from /etc/ppp/peers/quectel-ppp)
/dev/ttyUSB3            # (from /etc/ppp/peers/quectel-ppp)
115200          # (from /etc/ppp/peers/quectel-ppp)
lock            # (from /etc/ppp/peers/quectel-ppp)
connect chat -s -v -f /etc/ppp/peers/quectel-chat-connect               # (from /etc/ppp/peers/quectel-ppp)
disconnect chat -s -v -f /etc/ppp/peers/quectel-chat-disconnect         # (from /etc/ppp/peers/quectel-ppp)
nocrtscts               # (from /etc/ppp/peers/quectel-ppp)
modem           # (from /etc/ppp/peers/quectel-ppp)
hide-password           # (from /etc/ppp/peers/quectel-ppp)
novj            # (from /etc/ppp/peers/quectel-ppp)
novjccomp               # (from /etc/ppp/peers/quectel-ppp)
ipcp-accept-local               # (from /etc/ppp/peers/quectel-ppp)
ipcp-accept-remote              # (from /etc/ppp/peers/quectel-ppp)
ipparam 3gppp           # (from /etc/ppp/peers/quectel-ppp)
noipdefault             # (from /etc/ppp/peers/quectel-ppp)
ipcp-max-failure 30             # (from /etc/ppp/peers/quectel-ppp)
defaultroute            # (from /etc/ppp/peers/quectel-ppp)
usepeerdns              # (from /etc/ppp/peers/quectel-ppp)
noccp           # (from /etc/ppp/peers/quectel-ppp)
abort on (BUSY)
abort on (NO CARRIER)
abort on (NO DIALTONE)
abort on (ERROR)
abort on (NO ANSWER)
timeout set to 30 seconds
send (AT^M)
expect (OK)
AT^M^M
OK
 -- got it

send (ATE0^M)
expect (OK)
^M
ATE0^M^M
OK
 -- got it

send (ATI;+CSUB;+CSQ;+CPIN?;+COPS?;+CGREG?;&D2^M)
expect (OK)
^M
^M
Quectel^M
EC20F^M
Revision: EC20CEFDKGR06A04M2G^M
^M
SubEdition: V09^M
^M
+CSQ: 29,99^M
^M
+CPIN: READY^M
^M
+COPS: 0,0,"CHINA MOBILE",7^M
^M
+CGREG: 0,1^M
^M
OK
 -- got it

send (AT+CGDCONT=1,"IP","3gnet",,0,0^M)
expect (OK)
^M
^M
OK
 -- got it
```

```
send (ATD*99#^M)
expect (CONNECT)
^M
^M
CONNECT
 -- got it

Script chat -s -v -f /etc/ppp/peers/quectel-chat-connect finished (pid 3334), status = 0x0
Serial connection established.
using channel 1
Using interface ppp0
Connect: ppp0 <--> /dev/ttyUSB3
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x700123c2> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x0 <asyncmap 0x0> <auth chap MD5> <magic 0x360a419a> <pcomp> <accomp>]
sent [LCP ConfAck id=0x0 <asyncmap 0x0> <auth chap MD5> <magic 0x360a419a> <pcomp> <accomp>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x700123c2> <pcomp> <accomp>]
rcvd [LCP DiscReq id=0x1 magic=0x360a419a]
rcvd [CHAP Challenge id=0x1 <4162163c6dc48a1c2b2e7cd5b23aaa32>, name = "UMTS_CHAP_SRVR"]
sent [CHAP Response id=0x1 <8e39e62928ffd0b59b8fdbd4a2d2fb97>, name = "test"]
rcvd [CHAP Success id=0x1 ""]
CHAP authentication succeeded
CHAP authentication succeeded
sent [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
rcvd [IPCP ConfReq id=0x0]
sent [IPCP ConfNak id=0x0 <addr 0.0.0.0>]
rcvd [IPCP ConfNak id=0x1 <addr 100.72.97.144> <ms-dns1 211.137.58.20> <ms-dns2 211.137.64.163>]
sent [IPCP ConfReq id=0x2 <addr 100.72.97.144> <ms-dns1 211.137.58.20> <ms-dns2 211.137.64.163>]
rcvd [IPCP ConfReq id=0x1]
sent [IPCP ConfAck id=0x1]
rcvd [IPCP ConfAck id=0x2 <addr 100.72.97.144> <ms-dns1 211.137.58.20> <ms-dns2 211.137.64.163>]
Could not determine remote IP address: defaulting to 10.64.64.64
local  IP address 100.72.97.144
remote IP address 10.64.64.64
primary   DNS address 211.137.58.20
secondary DNS address 211.137.64.163
Script /etc/ppp/ip-up started (pid 3346)
Script /etc/ppp/ip-up finished (pid 3346), status = 0x0
```

- Test the network

Checking the network whether connected by ping the baidu.

```
#ping www.baiud.com

PING www.a.shifen.com (111.13.100.92) 56(84) bytes of data.
64 bytes from 111.13.100.92: icmp_seq=1 ttl=53 time=78.5 ms
64 bytes from 111.13.100.92: icmp_seq=2 ttl=53 time=63.9 ms
```

The D23 led will be on if dailed successful, and check the ppp0 status.

```
# ifconfig ppp0
ppp0      Link encap:Point-to-Point Protocol
          inet addr:100.72.97.144  P-t-P:10.64.64.64  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:17 errors:0 dropped:0 overruns:0 frame:0
          TX packets:23 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:1106 (1.0 KiB)  TX bytes:1413 (1.3 KiB)
```

4.12 RTC Test

This sample demostrates how to write and read the RTC time ,as well as when the system power off for a while , the system time whether synchronize to the RTC.

Hardware connection

Install RTC battery into J16.

Software test

RTC write and read test

1.View the system time

```
# date
Wed Apr  3 12:09:40 UTC 2019
```

2.Update the system time to RTC

```
hwclock -w
```

3.View the time of RTC

```
# hwclock
Wed Apr  3 12:11:09 2019  0.000000 seconds
```

4.Set a system time

```
#date -s 20170402
Sun Apr  2 00:00:00 UTC 201
```

5.Update the RTC`s time to system

```
#hwclock -s
#date
Wed Apr  3 12:13:10 UTC 2019
```

Perform RTC synchronization system time test

The above process has set the time for the RTC. Now power off first for a minute, then power on , the system time will be update by RTC .

```
# date
Wed Apr  3 12:14:20 UTC 2019
```

4.13 M.2 Test

M.2 interface work on the NVME agreement，this sample use nvme2280 SSD hard disk of TOSHIBA RC100 for validation purpose.

Hardware connection

Insert the SSD into J13 interface.

Software test

The Kernel has added the NVME protocol by default,so we can use SSD without configure any more.

- Check the ssd is mounted By below command to check whether it mounted or not.

```
#df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       6.4G  1.9G  4.2G  31% /
devtmpfs        237M  4.0K  237M   1% /dev
tmpfs           494M     0  494M   0% /dev/shm
tmpfs           494M  8.6M  485M   2% /run
tmpfs           494M     0  494M   0% /sys/fs/cgroup
tmpfs           494M  4.0K  494M   1% /tmp
tmpfs           494M  144K  494M   1% /var/volatile
/dev/nvme0n1p1  110G  560M  104G   1% /run/media/nvme0n1p1
/dev/mmcblk0p1  500M   21M  480M   5% /run/media/mmcblk0p1
tmpfs            99M   88K   99M   1% /run/user/0

# cat /proc/mounts  | grep nvme
/dev/nvme0n1p1 /run/media/nvme0n1p1 ext4 rw,relatime,data=ordered 0 0
```

- Write and read SSD

Write a hello strings to the test.txt on the SSD mount point, and then read it,compare with what bas been written.

```
cd /run/media/nvme0n1p1
echo hello > test.txt
cat test.txt
hello
```

4.14 HDMI Test

This charpter will use gst-play command to play video file on the display of HDMI.

Hardware connection

- Connect the HDMI port and display with HDMI cable
- Insert usb disk with a video file

Software test

The gst-play command can use to play video , below step will play a 4K file on the display.

```
gst-launch-1.0   playbin uri=file:///run/media/sda/LG.SUPER.UHDTV_4K.HDR.DEMO_New.York.ts audio-sink="alsasink
 device=hw:2,0  sync=false async=false"
```

Additionally:

The resolution of HDMI can be set at the uboot stage by the mmcargs parameter, perform below step:

1. Press any key during the power on and enter the uboot cmdline
2. add video parameter to mmcargs and save
3. re-power the baord

Set the HDMI resolution to 1080P as below:

```
printenv mmcargs
mmcargs=setenv bootargs console=${console} root=${mmcroot}

setenv mmcargs 'mmcargs=setenv bootargs console=${console} root=${mmcroot} video=HDMI-A-1:1920x1080-32@60'
save
```

4.15 MIPI-DSI(LVDS) Test

The U37 conponent on the development board is used to transform MIPI-DSI to LVDS signal. the system supports DCSS or LCDIF to output .This sample will use 10.1 inch screen to test DCSS and LCDIF output,the interface is J20.

Hardware connection

Connect the 10.1 inch screen to the J20 FPC interface.

Software test

- About configure file

There are some configuration files ending with .dtb in /run/media/mmcblk0p1/ dirctory.If the files are not existing, you can find them in 02-Images/imx8mqevk-xddr and copy them into the /run/emdia/mmcblk0p1 directory.

File name	Fearture
myb-fsl-imx8mq-evk.dtb	HDMI use DCSS
myb-fsl-imx8mq-evk-dcss-tc358775-10.dtb	10.1 inch screen with DCSS
myb-fsl-imx8mq-evk-lcdif-tc358775-10.dtb	10.1 inch screen with LCDIF
myb-fsl-imx8mq-evk-tc358775-dual-display-10.dtb	HDMI with DCSS，lvds screen with LCDIF

At uboot stage the value of fdt_file will decide to use which dtb file for the kernel. In yhis sample ,we set the fdt_file value to myb-fsl-imx8mq-evk-lcdif-tc358775-10.dtb for testing.

Enter the uboot and set the fdt_file value as below:

1.Press any key

2.Power on the board and enter the cmdline

3.Set fdt_file value and save

4.Re-power the board

Set and save the fdt_file value during uboot stage as below：

```
setenv fdt_file myb-fsl-imx8mq-evk-lcdif-tc358775.dtb
save
```

Now you will see the 10.1 inch screen display the weston.

4.16 MIPI-CSI Test

The J26 and J2 on the development board are equipped with two MIPI-CSI interfaces, and the camera module(MY-CAM003) is used for preview and taking photos to verify the MIPI-CSI function.

Hardware connection

- Connect the camera module to J26 interface.
- Insert the USB disk into the development board,which stored the example in the 4 chapters.

Software test

The gst-play can use to preview and the v4l2grab can use to take phote test.

There are two CSI interfaces on the development board, J26 is connected to the /dev/video0 node , J2 is connected to the /dev/video1 node.

Preview

By using J26 to preview, perform as below:

```
gst-launch-1.0 v4l2src device=/dev/video0 ! 'video/x-raw,width=640,height=480,framerate=30/1' ! glimagesink
```

Taking photos

The v4l2grab is located in the example/csi directory on the usb disk.It can be used to take photo and save the file on the CSI directory,perform as below:

```
cd /run/media/sda/example/csi
./v4l2grab -d /dev/video0 -W 640 -H 480 -I 30 -o picture.jpg
Supported palettes:
0: YUYV (YUYV 4:2:2)
1: RGB3 (RGB3)
2: BGR3 (BGR3)
3: YU12 (YU12)
4: YV12 (YV12)
Supported framesize:
0: witdth = 640 height = 480
1: witdth = 720 height = 480
2: witdth = 1280 height = 720
3: witdth = 1920 height = 1080
4: witdth = 2592 height = 1944
5: witdth = 0 height = 0
[  189.228012] alloc_contig_range: [68c00, 68c96) PFNs busy
[  189.239325] alloc_contig_range: [68c00, 68c96) PFNs busy
[  189.282109] ov5640_mipi 0-003c: s_stream: 1
[  189.486510] ov5640_mipi 0-003c: s_stream: 0
```

Dual preview

When the J26 and J2 all connencted with the camera modules, the gst-play can dual camera preview ,perform as below:

```
gst-launch-1.0 v4l2src device=/dev/video0 ! 'video/x-raw,width=640,height=480,framerate=30/1' ! glimagesink & g
st-launch-1.0 v4l2src device=/dev/video1 ! 'video/x-raw,width=640,height=480,framerate=30/1' ! glimagesink
```

5. QT application development

Qt is a cross-platform graphical application development framework that is applied to devices and platforms of different sizes, and is available in different copyright versions for users to choose from. MYD-JX8MX uses Qt 5.9.4 for application development. In Qt application development, it is recommended to use the QtCreator integrated development environment, you can develop Qt applications under Linux PC, and automatically cross-compile into the ARM architecture of the development board.

This chapter uses the SDK tool built by Yocto as a cross-compilation system to quickly develop graphics applications with QtCreator. Before starting this chapter, please complete the Yocto build process in Chapter 3. Or use the precompilation provided on the CD SDK toolkit. Before starting this chapter, please install the application SDK development tool.

5.1 Install Qt toolchain and QtCreator

Install qt toolchain

The QT toolchain locates in 03-Tools/Toolchain/fsl-imx-xwayland-glibc-x86_64-meta-toolchain-qt5-aarch64-toolchain-4.9.88-2.0.0.sh

Install the toolchain as below：

```
./fsl-imx-xwayland-glibc-x86_64-meta-toolchain-qt5-aarch64-toolchain-4.9.88-2.0.0.sh
```

Result：

```
NXP i.MX Release Distro SDK installer version 4.9.88-2.0.0
===========================================================
Enter target directory for SDK (default: /opt/fsl-imx-xwayland/4.9.88-2.0.0): /home/duxy/opt_qt
You are about to install the SDK to "/home/duxy/opt_qt". Proceed[Y/n]? y
Extracting SDK.....................................................................................
.....................................................................................................
.......done
Setting it up...done
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to source the environment setup script e.g.
 $ . /home/duxy/opt_qt/environment-setup-aarch64-poky-linux
```

Install QtCreator

The QtCreator installation package is a binary program that can be installed directly.

The toolchain can download through the yocto and also can find in the 03-Tools/Qt/qt-opensource-linux-x64-5.9.4.run

```
cd 03-Tools/Qt/
chmod a+x qt-opensource-linux-x64-5.9.4.run
 ./qt-opensource-linux-x64-5.9.4.run
```

Keep clicking the next step, install by default, or choose another directory to install. In this example, select the installation and then the user directory.

Start the program with following command.

```
~/Qt5.9.4/Tools/QtCreator/bin/qtcreator.sh
```

5.2 Configure QtCreator

It is necessary to configure the toolchain in QtCreator if your want the program can run on the development baord.

perform below command to start the qtcreate.

```
~/opt/Qt5.9.4/Tools/QtCreator/bin/qtcreator.sh
```

Make a new configure：

- Configure a gcc and g++ compile
- Configure a QTversion
- Configure a QTdebug
- Add a new device
- Create a kit and add the above 4 items together to compile the QT configuration

1.Configure GCC and G++

After started Qtcreator, first open a sample, make the "project" become configurable, and then click "Project" -> Manage kits, as shown below.

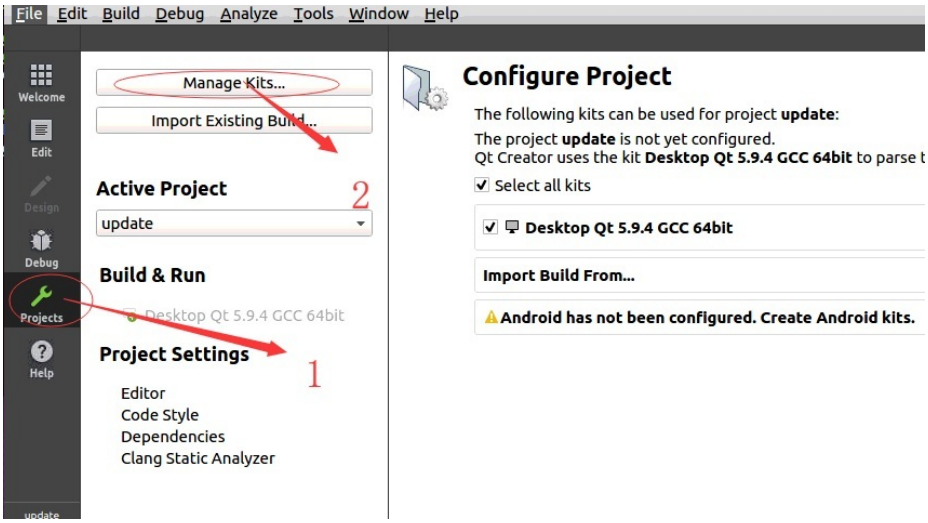

Figure 5-2-1 qtcreate interface

Then go to "Options" -> Compiler -> "Add" -> GCC->C/C++. The process is as follows:

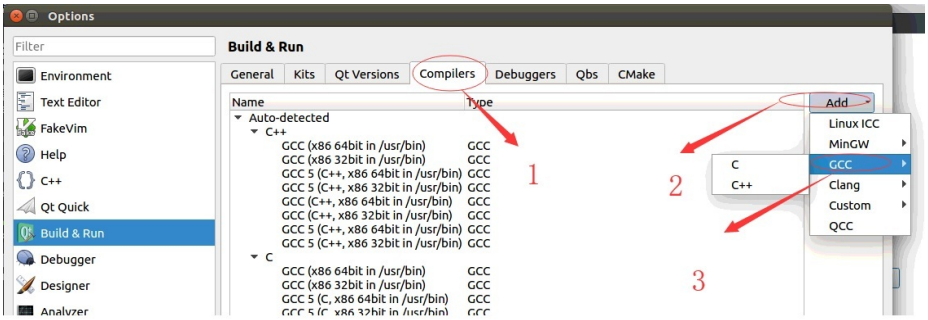

Figure 5-2-2 option interface

Then configure C and C++, you need to fill in the "name", "compiler path", "abi" select arm architecture

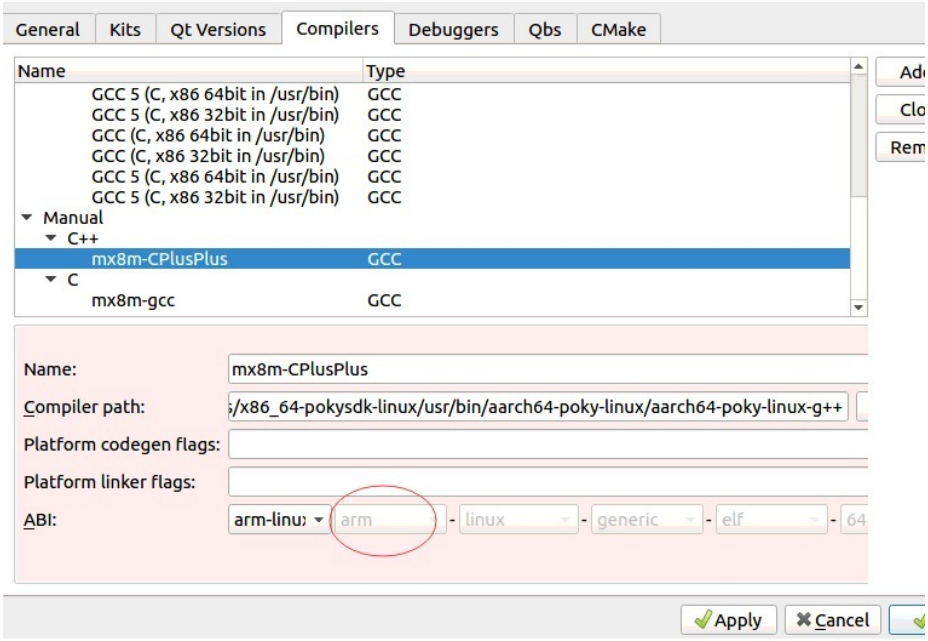

Figure 5-2-3 C++ configure

The C++ compiler path is:

```
/home/duxy/opt_qt/sysroots/x86_64-pokysdk-linux/usr/bin/aarch64-poky-linux-musl/aarch64-poky-linux-musl-g++
```

The C compiler path is:

```
/home/duxy/opt_qt/sysroots/x86_64-pokysdk-linux/usr/bin/aarch64-poky-linux-musl/aarch64-poky-linux-musl-gcc
```

2.Configure QT versions

"Option"-> Qt Versions -> "Add"，select the following file.

```
/home/duxy/opt_qt/sysroots/x86_64-pokysdk-linux/usr/bin/qt5/qmake
```

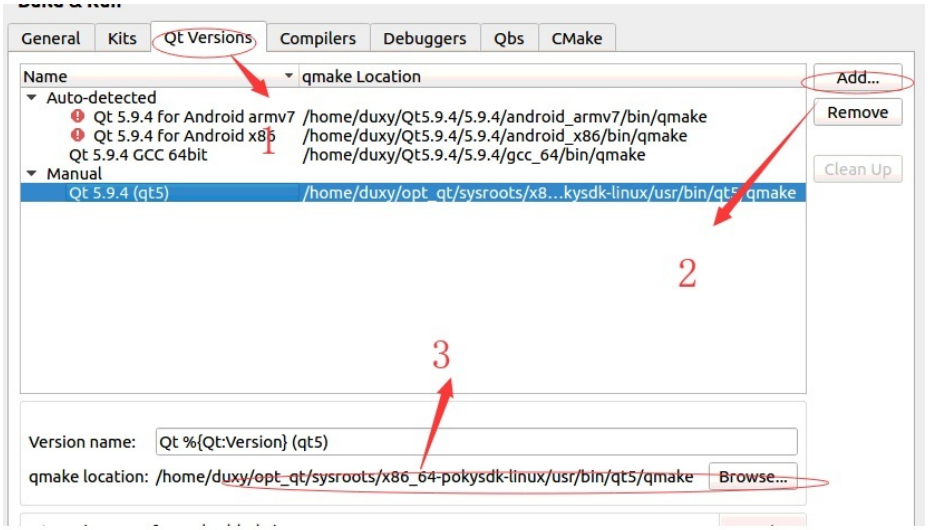

Figure 5-2-4 QTversion configure

3.Configure QT debug "Option"->"Debuggers"->"add" ，fill name and path.

path ：

```
/home/duxy/opt_qt/sysroots/x86_64-pokysdk-linux/usr/bin/aarch64-poky-linux-musl/aarch64-poky-linux-musl-gdb
```

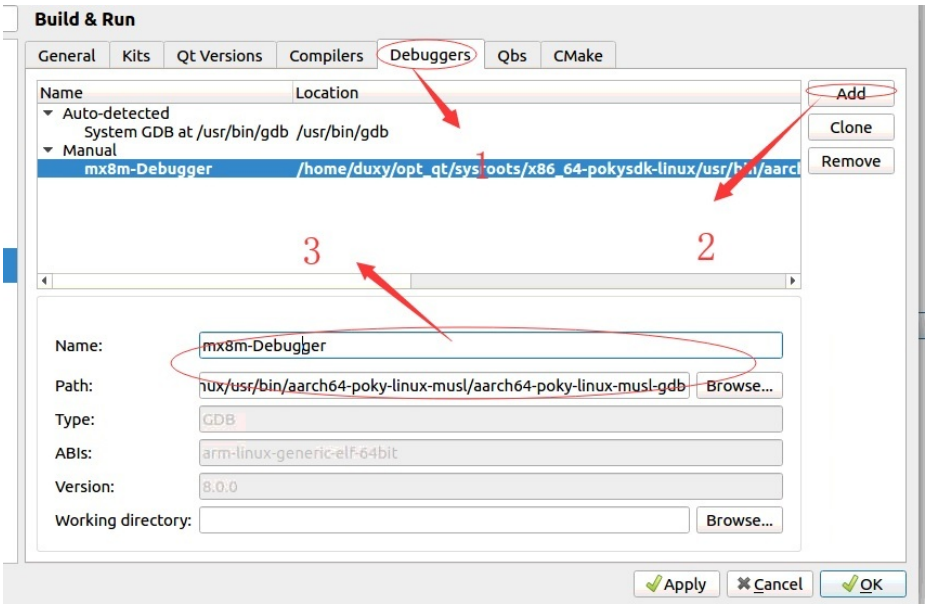

Figure 5-2-5 QTdebug configure

4.Add new device "device"->"add"->"general Linux device" Then configure the name, IP address, login user name, account number and password.

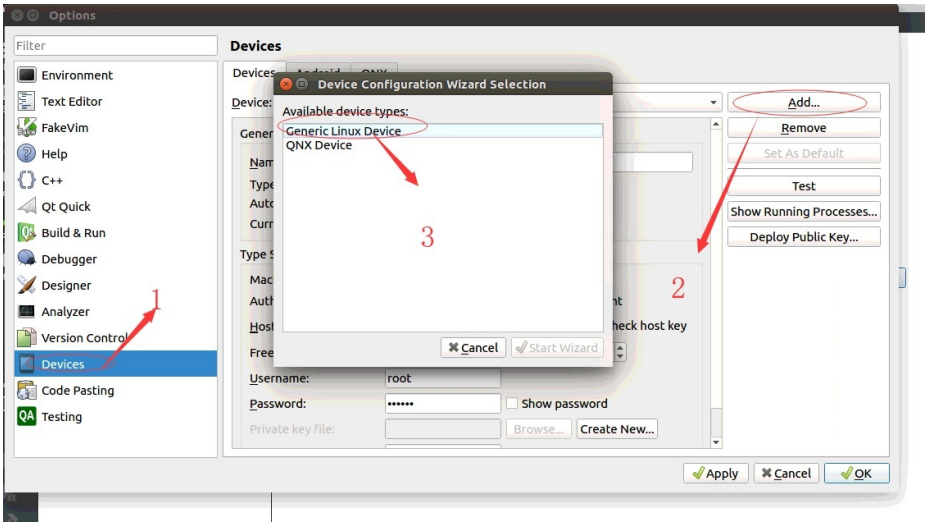

Figure 5-2-6 device configure01

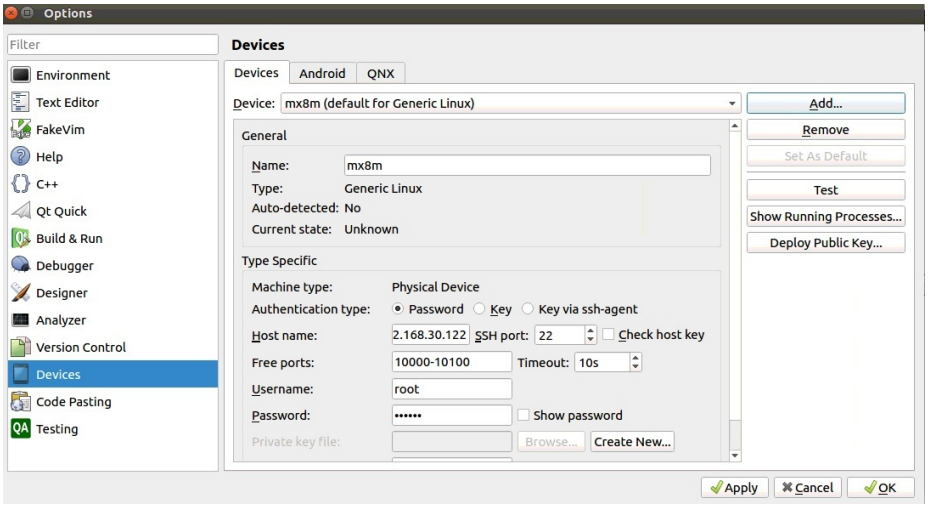

Figure 5-2-7 device configure01

5.Create kit

"Build and run" -> build kit (kit) -> "add".

Here you need to combine the four-step configuration files of the appeal and fill in the sysroot:

```
/home/duxy/opt_qt/sysroots
```

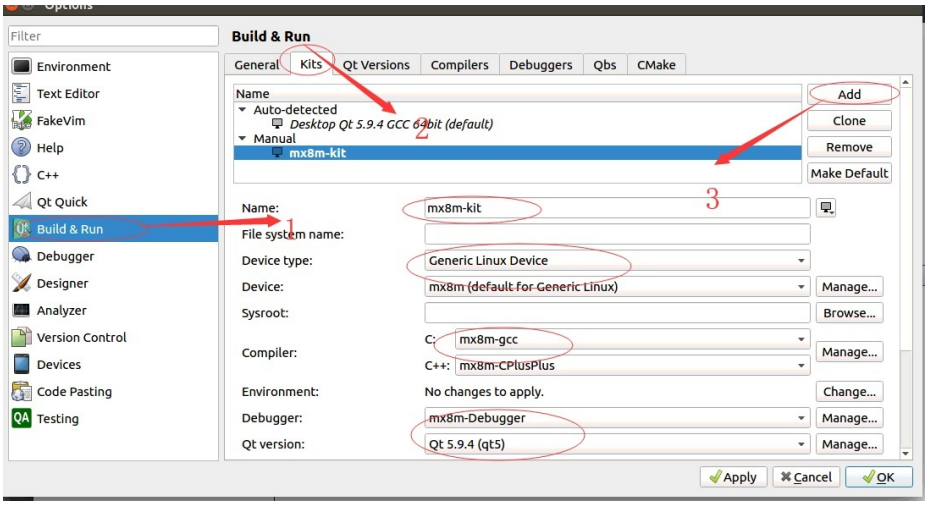

Figure 5-2-8 kit configure

Finally configure as follows:

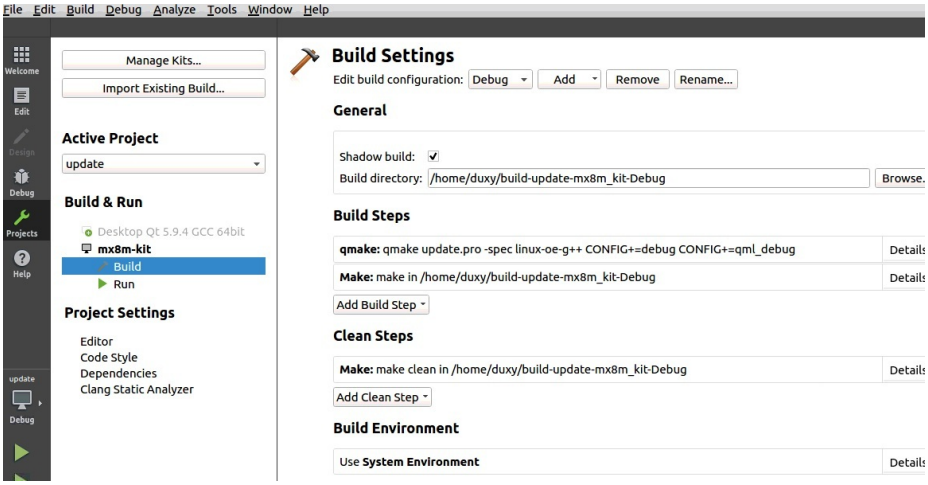

Figure 5-2-9 kit configure complete

5.3 Test qt application

This example compiles a QT demo and runs on the development board.

The QT example source code is located in 04-Source/borderlayout.tar.gz.

1.Decompress file

```
cd 04-Source
tar -zxvf borderlayout.tar.gz
```

2.Qtcreator compile demo

Start Qtcreator

```
~/opt/Qt5.9.4/Tools/QtCreator/bin/qtcreator.sh
```

"Welcome"->open project ->select the borderlayout.pro project->"Project"->select mx8m-kit->compile

Then go to "Options" -> Compiler -> "Add" -> GCC->C/C++. The process is as follows:

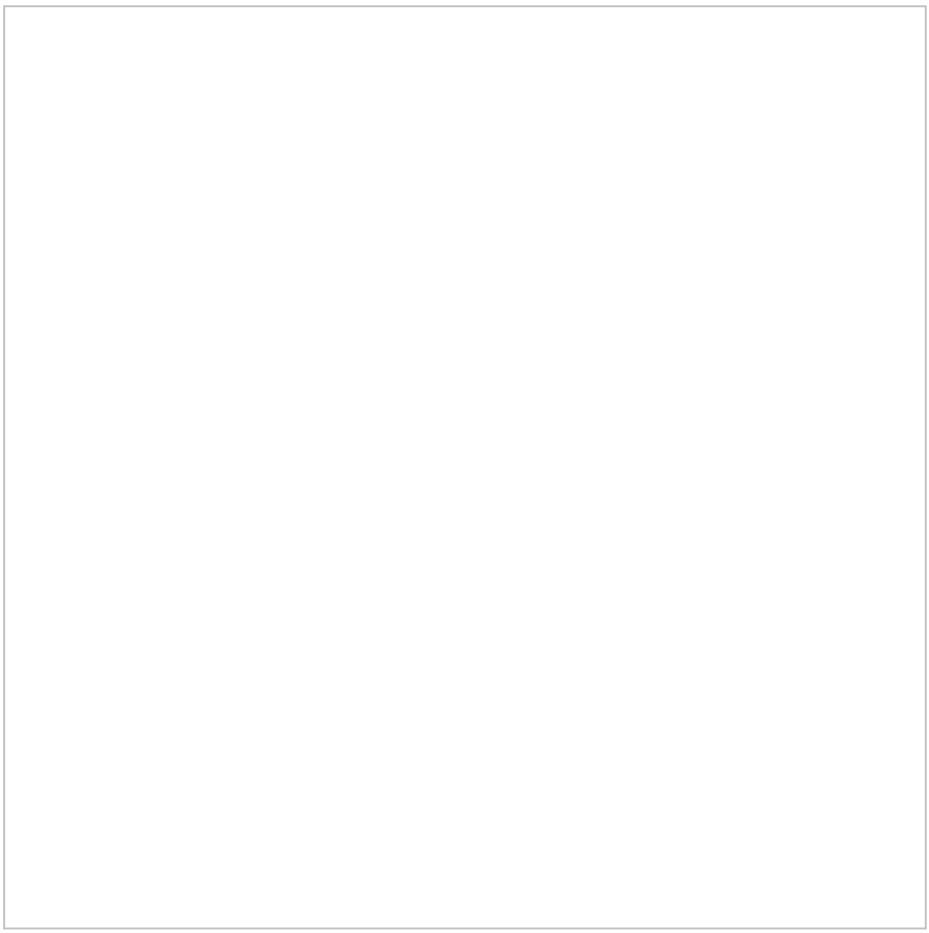

Figure 5-3-1 compile qt

After the sample is completed, the build-borderlayout-mx8m_kit-Debug directory exists in the same directory.

The borderlayout in the directory is an executable file. After copying this file to the root directory of the USB disk, insert it into the development board and power it on.

3.Start the qt demo

Power on and excute the follow file:

```
./run/media/sda/borderlayout
```

We can see this on the display

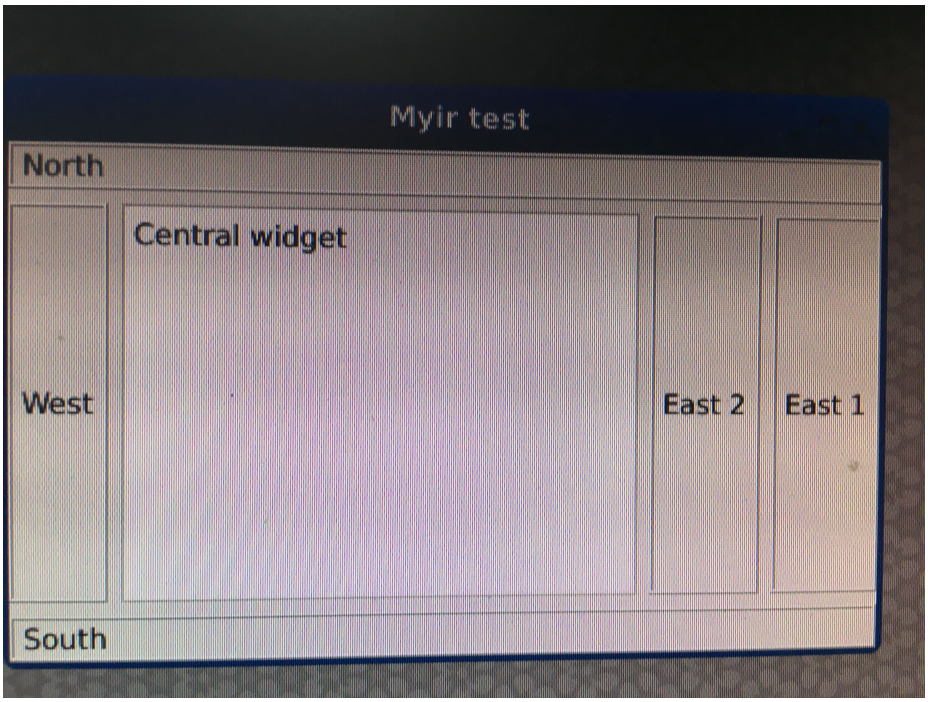

Figure 5-3-2 excute qt

6. Update system

The MYD-JX8MX series development board provides two ways to update Linux systems, UUU updates and SD card updates.

- UUU update:

Switch the Download mode and use the UUU tool to program the file into the EMMC of the development board. Need to connect the PC to the development board.

- SD card update:

Set the board to boot form SD card . After booting up, program the file from the SD card to the EMMC of the development board. It it the way to be used for mass production .

6.1 UUU update system

The Universal Update Utility (UUU) runs on a Windows or Linux OS host and is used to download images to different devices on an i.MX board.

Programing process：

1. Set the development board to download mode.
2. Type the command of program.
3. Set the internal mode after programing successful.

1.PC preparation

Programing the image requires the following file, which can be generated after yocto is compiled.

about "imx-uboot bootstrap， entirely image suffix sdcard.bz2"

File	Compile name
imx-uboot	imx-boot-imx8mqevk-sd.bin-flash_evk
sdcard	fsl-image-qt5-validation-imx-imx8mqevk.sdcard.bz2

The fsl-image-qt5-validation-imx-imx8mqevk.sdcard.bz2 needs to be decompressed first.

Put 03-Tools/UUU/UUU_MYD-JX8MQ6-8E2D-130-E in the root directory of the D disk in the window, and replace imx-boot-imx8mqevk-sd.bin-flash_evk and fsl-image-qt5-validation-imx-imx8mqevk.sdcard image.

2.Development board preparation

Set the SW1 to EMMC startup and SW2 to download mode when the board power off.

SW1 boot dev（1~4）	Boot device
0010	EMMC startup
1100	SD card startup

SW2 boot mode （1~2）	Boot mode
00	fuse mode（internal start priority）
10	download mode
01	internal mode
11	reserve

If you want to program the image to emmc，set SW1 to 0010 and set SW2 to 10 .

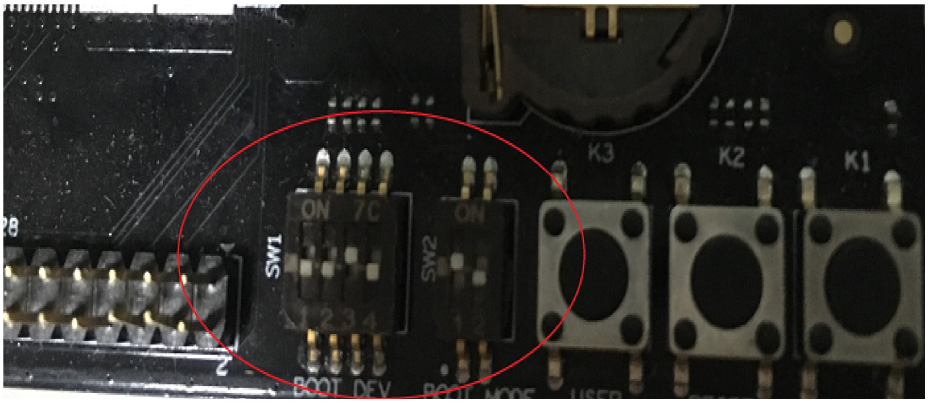

Figure 6-1 set EMMC download mode

Connect one end of the Type-C to the PC, the other end to the typeC interface of the board, one end of the serial cable connect to the PC, and the other end connect to the J18 serial port (the serial port only plays the role of viewing the log)

3.Start programing

Before powering up, execute the following command in the window cmd command line.

```
C:\Users\duxy>d:

D:\>cd UUU_MYD-JX8MQ6-8E2D-130-E

D:\UUU_MYD-JX8MQ6-8E2D-130-E>uuu.exe uuu.auto
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.2.91-0-g3799f4d

Wait for Known USB Device Appear...
```

Figure 6-2 run the UUU command

After power on

```
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.2.39-0-gdcc404f

Success 0    Failure 0

1:4    4/ 7   [==>         11%           ] FB: flash -raw2sparse all fsl-image-qt5-validation-imx-imx8mqevk.sdcard
```

Figure 6-3 begin to program

Finally program successfully.

```
Success 1    Failure 0

1:4    7/ 7   [Done                       ] FB: done
```

Figure 6-4 program successfully

When the programming is completed. Set the boot mode change to internal mode (01) . Now the board is ready for normal start up.

6.2 SD card update system

To program images with the mfg tool, you must have a computer with one board, which is not suitable for mass production. Here is how to use the SD card for programing. The SD card programing steps are:

1.Program the image to the SD card

2.Insert the sd card into the development board and set the boot mode to SD boot first.

3.Power on and program the image , the simple progress of qt display on the screen.

Introduce how to create SD card update image

Preparing files：

The imx-boot, kernel, dtb and rootfs which want to be programed into EMMC.

According to the EMMC size, decompress 03-Tools/MksdcardTool/Mksdcard_MYD-JX8MQ6-8E1D-130-E.tar.gz or Mksdcard_MYD-JX8MQ6-8E2D-130-E.tar.gz.

```
tar -zxf Mksdcard_MYD-JX8MQ6-8E2D-130-E.tar.gz
```

Contents as follows:

```
.
├── factory
├── firmware
├── lib
├── mfgimages   #stored image prepare to pgrgram into emmc
├── mkfs.ext3
├── mkfs.ext4
├── mkfs.ext4.real
├── myir_mkscard.sh      #the scripts to create the image
├── ppp
├── README.txt
└── rootfs
```

After replacing the image you need to program into mfgimages, you can execute below command.

```
sudo ./myir_mkscard.sh
```

After the scripts completed the image will generated JX8MX-Update-20190602144250-2gddr.rootfs.scard.

Program to the SD card with dd if command , or program it to the SD card with 03-Tools/MksdcardTool/Win32DiskImager-1.0.0-binary.zip.

Insert the SD card into the board, set the SD card to boot. It will run the burn_emmc.sh script to program the image under mfgimages to EMMC automaticly.

Appendix 1 Warranty & Technical Support Services

MYIR Tech Limited is a global provider of ARM hardware and software tools, design solutions for embedded applications. We support our customers in a wide range of services to accelerate your time to market.

MYIR is an ARM Connected Community Member and work closely with ARM and many semiconductor vendors. We sell products ranging from board level products such as development boards, single board computers and CPU modules to help with your evaluation, prototype, and system integration or creating your own applications. Our products are used widely in industrial control, medical devices, consumer electronic, telecommunication systems, Human Machine Interface (HMI) and more other embedded applications. MYIR has an experienced team and provides custom design services based on ARM processors to help customers make your idea a reality.

The contents below introduce to customers the warranty and technical support services provided by MYIR as well as the matters needing attention in using MYIR's products.

Service Guarantee

MYIR regards the product quality as the life of an enterprise. We strictly check and control the core board design, the procurement of components, production control, product testing, packaging, shipping and other aspects and strive to provide products with best quality to customers. We believe that only quality products and excellent services can ensure the long-term cooperation and mutual benefit.

Price

MYIR insists on providing customers with the most valuable products. We do not pursue excess profits which we think only for short-time cooperation. Instead, we hope to establish long-term cooperation and win-win business with customers. So we will offer reasonable prices in the hope of making the business greater with the customers together hand in hand.

Delivery Time

MYIR will always keep a certain stock for its regular products. If your order quantity is less than the amount of inventory, the delivery time would be within three days; if your order quantity is greater than the number of inventory, the delivery time would be always four to six weeks. If for any urgent delivery, we can negotiate with customer and try to supply the goods in advance.

Technical Support

MYIR has a professional technical support team. Customer can contact us by email (support@myirtech.com), we will try to reply you within 48 hours. For mass production and customized products, we will specify person to follow the case and ensure the smooth production.

After-sale Service

MYIR offers one year free technical support and after-sales maintenance service from the purchase date. The service covers:

1. Technical support service

```
* MYIR offers technical support for the hardware and software materials which have provided to customers;
* To help customers compile and run the source code we offer;
* To help customers solve problems occurred during operations if users follow the user manual documents;
* To judge whether the failure exists;
* To provide free software upgrading service.
```

However, the following situations are not included in the scope of our free technical support service:

```
* Hardware or software problems occurred during customers' own development;
* Problems occurred when customers compile or run the OS which is tailored by themselves;
* Problems occurred during customers' own applications development;
```

```
     * Problems occurred during the modification of MYIR’s software source code.
```

1. After-sales maintenance service

The products except LCD, which are not used properly, will take the twelve months free maintenance service since the purchase date. But following situations are not included in the scope of our free maintenance service:

```
* The warranty period is expired;
* The customer cannot provide proof-of-purchase or the product has no serial number;
* The customer has not followed the instruction of the manual which has caused the damage the product;
* Due to the natural disasters (unexpected matters), or natural attrition of the components,
or unexpected matters leads the defects of appearance/function;
* Due to the power supply, bump, leaking of the roof, pets, moist, impurities into the boards,
all those reasons which have caused the damage of the products or defects of appearance;
* Due to unauthorized weld or dismantle parts or repair the products which has caused
the damage of the products or defects of appearance;
* Due to unauthorized installation of the software,
system or incorrect configuration or computer virus which has caused the damage of products.
```

Warm tips:

1. MYIR does not supply maintenance service to LCD. We suggest the customer first check the LCD when receiving the goods. In case the LCD cannot run or no display, customer should contact MYIR within 7 business days from the moment get the goods.
2. Please do not use finger nails or hard sharp object to touch the surface of the LCD.
3. MYIR suggests user purchasing a piece of special wiper to wipe the LCD after long time use, please avoid clean the surface with fingers or hands to leave fingerprint.
4. Do not clean the surface of the screen with chemicals.
5. Please read through the product user manual before you using MYIR’s products.
6. For any maintenance service, customers should communicate with MYIR to confirm the issue first. MYIR’s support team will judge the failure to see if the goods need to be returned for repair service, we will issue you RMA number for return maintenance service after confirmation.

1. Maintenance period and charges
    - MYIR will test the products within three days after receipt of the returned goods and inform customer the testing result. Then we will arrange shipment within one week for the repaired goods to the customer. For any special failure, we will negotiate with customers to confirm the maintenance period.
    - For products within warranty period and caused by quality problem, MYIR offers free maintenance service; for products within warranty period but out of free maintenance service scope, MYIR provides maintenance service but shall charge some basic material cost; for products out of warranty period, MYIR provides maintenance service but shall charge some basic material cost and handling fee.
2. Shipping cost

During the warranty period, the shipping cost which delivered to MYIR should be responsible by user; MYIR will pay for the return shipping cost to users when the product is repaired. If the warranty period is expired, all the shipping cost will be responsible by users.

1. Products Life Cycle

MYIR will always select mainstream chips for our design, thus to ensure at least ten years continuous supply; if meeting some main chip stopping production, we will inform customers in time and assist customers with products updating and upgrading.

Value-added Services

1. MYIR provides services of driver development base on MYIR's products, like serial port, USB, Ethernet, LCD, etc.
2. MYIR provides the services of OS porting, BSP drivers' development, API software development, etc.
3. MYIR provides other products supporting services like power adapter, LCD panel, etc.
4. ODM/OEM services.

MYIR Tech Limited

Room 1306, Wensheng Center, Wenjin Plaza,

North Wenjin Road, Luohu District, Shenzhen, China 518020

Support Email: support@myirtech.com

Sales Email: sales@myirtech.com

Phone: +86-755-22984836

Fax: +86-755-25532724

Website: www.myirtech.com