

MYD-JX8MX Linux 开发手册

MYiR™ Make Your Idea Real



Table of Contents

前言	1.1
1. 软件资源介绍	1.2
2. 部署开发环境	1.3
3. Yocto构建系统	1.4
3.1 Yocto编译完整镜像	1.4.1
3.2 Yocto编译U-Boot	1.4.2
3.3 Yocto编译Kernel	1.4.3
3.4 Yocto构建SDK工具	1.4.4
4. Linux 应用开发	1.5
4.1 GPIO-KEY测试	1.5.1
4.2 GPIO-LED测试	1.5.2
4.3 USB-HOST测试	1.5.3
4.4 USB-DEVICE测试	1.5.4
4.5 Ethernet测试	1.5.5
4.6 Audio测试	1.5.6
4.7 串口测试	1.5.7
4.8 SPI测试	1.5.8
4.9 WIFI测试	1.5.9
4.10 BT测试	1.5.10
4.11 4G测试	1.5.11
4.12 RTC测试	1.5.12
4.13 M.2测试	1.5.13
4.14 HDMI测试	1.5.14
4.15 MIPI-DSI测试	1.5.15
4.16 MIPI-CSI测试	1.5.16
4.17 NXP提供例子	1.5.17
5. QT 应用开发	1.6
5.1 安装QT编译链和QtCreate	1.6.1
5.2 配置QtCreate	1.6.2
5.3 测试Qt应用	1.6.3
6. 系统更新	1.7
6.1 UUU 更新系统	1.7.1
6.2 sd卡启动更新系统	1.7.2

附录A	1.8
附录B	1.9

MYD-JX8MX Linux 开发手册

本文档介绍基于MYD-JX8MX 系列开发板，进行Linux系统的编译和安装，硬件接口使用，QT应用开发等。

历史版本

版本号	描述	时间
V1.0	初始版本	2019.03.29
V1.1	1.uboot下区分DDR大小 2.修改使用SD卡升级方式	2019.06.02
V1.2	1.修改linux-imx-src.inc 路径 2.新增Fn-link6222b模组	2019.08.12
V1.3	更新BSPL4.19.35	2020.04.26

硬件版本

本手册适合于以下型号的开发板

- MYD-JX8MX

1. 软件资源介绍

MYD-JX8MX搭载基于Linux 4.19.35内核的操作系统，提供了丰富的系统资源和软件资源。部分资源需要配合相应的扩展模块才能使用。以下是软件资源列表：

类别	名称	描述信息	源码
引导程序	U-boot	第一级引导启动程序	YES
Linux内核	Image	基于官方imx_4.19.35_1.1.1_ga版本	YES
设备驱动	PMIC	BD71873 驱动	YES
设备驱动	USB Host	USB Host 3.0驱动	YES
设备驱动	USB OTG	USB OTG 3.0驱动	YES
设备驱动	I2C	I2C总线驱动	YES
设备驱动	SPI	SPI总线驱动	YES
设备驱动	Ethernet	10M/100M/1000M驱动	YES
设备驱动	MMC	MMC/EMMC/TF卡存储驱动	YES
设备驱动	HDMI	HDMI显示驱动	YES
设备驱动	LCD	MIPI-LVDS驱动	YES
设备驱动	PWM	PWM控制	YES
设备驱动	RTC	实时时钟驱动	YES
设备驱动	GPIO	GPIO驱动	YES
设备驱动	Touch	电容触摸驱动	YES
设备驱动	Audio	WM8904驱动	YES
设备驱动	Camera	Ov5640驱动	YES
设备驱动	WiFi & BT	QCA6174驱动	YES
设备驱动	Watchdog	Watchdog驱动	YES
设备驱动	LTE模块	支持移远EC20，使用USB驱动	YES
设备驱动	M.2	NVME驱动	YES
文件系统	Yocto rootfs	基于Yocto构建带xwayland的文件系统	YES
应用程序	GPIO KEY/LED	GPIO按键指示灯例程	YES
应用程序	NET	TCP/IP Sokect C/S例程	YES
应用程序	RTC	实时时钟例程	YES
应用程序	Audio	Audio例程	YES
应用程序	LCD	显示屏例程	YES
应用程序	Camera	多摄像头显示例程	YES



2.部署开发环境

开发前需要在PC安装好Linux操作系统，推荐使用Ubuntu 16.04 64bit发行版，8G内存，至少150G(推荐500G以上)空闲硬盘。连接网线并配置好网络，后续操作需要连接互联网安装或下载相关软件包。

替换Ubuntu软件源

由于Ubuntu自带的软件源处于国外服务器，国内客户下载资源会很慢，因此需要将软件源替换成清华源，清华源网址如下：

<https://mirrors.tuna.tsinghua.edu.cn/help/ubuntu/>

选择16.04，然后把内容更新到 /etc/apt/source.list 文件中。

操作步骤如下：

1.备份源

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

2.复制16.04清华源内容到此/etc/apt/sources.list 文件。

3.更新源

```
apt-get update
```

PS: 如遇到appstream3报错可以先删除软件包，再更新。

```
apt-get remove libappstream3
```

```
apt-get update
```

安装必备软件包

新装Ubuntu系统还需要安装一些常用软件，执行如下命令进行安装。

```
sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat libsdl1.2-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 curl lzop asciidoc
```

设置repo文件

repo是用来下载资源文件，存放在03-Tools/Repo 目录，如下操作将此文件添加到用户环境中。

```
mkdir ~/bin
```

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

配置git

资源下载时需要用到git，如下操作进行基本配置，用户名和邮箱改成户用对于名称。

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

3.Yocto构建系统

Linux系统平台上有许多开源的系统构建框架，这些框架方便了开发者进行嵌入式系统的构建和定制化开发，目前比较常见的有Buildroot, Yocto, OpenEmbedded等等。其中 Yocto项目使用更强大和定制化的方法，来构建出适合嵌入式产品的Linux系统。Yocto不仅仅是一个制作文件系统工具，同时提供整套的基于Linux的开发和维护工作流程，使底层嵌入式开发者和上层应用开发者在统一的框架下开发，解决了传统开发方式下零散和无管理的开发形态。Yocto是一个开源的“umbrella”项目，意指它下面有很多个子项目，Yocto只是把所有的项目整合在一起，同时提供一个参考构建项目Poky，来指导开发人员如何应用这些项目，构建出嵌入式Linux系统。它包含Bitbake, OpenEmbedded-Core, 板级支持包，各种软件包的配置文件。通过Poky，可以构建出不同类需求的系统，如最小的系统coreimage-minimal、全功能命令行系统core-image-base、带Qt5图形库的imx-image-full。

本章节主要介绍MYD-JX8MX开发板上，使用yocto构建系统，编译镜像。

MYD-JX8MX的Linux系统包含以下部分：

- imx-boot: 包括SPL，U-boot，ARM trust firmware和HDMI firmware。引导程序，支持不同方式启动内核。
- Linux Kernel: 适用于MYD-JX8MX开发板的Linux 4.19.35内核，同时包含支持板载外设的驱动。
- device tree file：板子硬件配置相关。
- rootfs：文件系统。

3.1 Yocto编译完整镜像

本章节介绍使用yocto工具编译完整镜像方法。

解压存放在04-Source/目录下的yocto包：fsl-release-yocto.tar.gz，并指定imx-image-full 目标进行编译，操作如下：

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

(L4.19.35 将fsl-image-qt5-validation-imx 改名成了imx-image-full)

编译完成后，生成的镜像在如下目录：

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

一些关键的资源包说明如下：

文件名称	作用
boot-imx8mqevk-sd.bin-flash_evk	imx-uboot镜像，引导程序
Image	kernel
myd-fsl-imx8mq-evk.dtb	dtb 配置文件
imx-image-full-imx8mqevk-20200424001447.rootfs.tar.bz2	文件系统bz2格式
imx-image-full-imx8mqevk-20200424001447.rootfs.sdcard.bz2	完整镜像 imx-uboot+dtb+kernel+rootfs

除了能指定上述的imx-image-full 目标，还有如下参数供开发者选择：

镜像名称	目标
core-image-minimal	最小文件系统
imx-image-multimedia	镜像带gui，不带qt功能
imx-image-full	带qt的镜像

其他参数请参考《i.MX_Yocto_Project_User's_Guide.pdf》 5.2 Choosing an i.MX Yocto project image

3.2 Yocto编译U-Boot

I.MX8M的引导程序imx-boot是由u-boot，SPL，ARM trust firmware和HDMI firmware共同组成。

SPL，ARM trust firmware 和HDMI firmware 均已固件方式提供，所以只需要将编译好的u-boot打包。

选择编译DDR大小

DDR目前提供3种配置，1GDDR 2GDDR 2GDDR_MT3D1024。因此Uboot有3个分支

```
duxy@myir-server1:~/L4.19.35/L4.19.35/uboot-imx$ git branch
* imx_v2018.03_4.14.98_2.0.0_ga-1gddr
  imx_v2018.03_4.14.98_2.0.0_ga-2gddr
  imx_v2018.03_4.14.98_2.0.0_ga-2gddr-MT3D1024M
```

2019年6月后得2G分支为imx_v2018.03_4.14.98_2.0.0_ga-2gddr-MT3D1024M。

选择方式由

sources/meta-fsl-bsp-release/imx/meta-bsp/recipes-bsp/u-boot/u-boot-imx_2019.04.bb 文件决定,内容如下:

```
SRCBRANCH = "imx_v2018.03_4.14.98_2.0.0_ga-2gddr-MT3D1024M"
SRCREV = "767f79e2f522a947c5b07944bf07db5fe7ff3c98"
```

这里可以看到uboot使用得源码目录为：uboot-imx

分支名称：imx_v2018.03_4.14.98_2.0.0_ga-2gddr-MT3D1024M

git commit 值：767f79e2f522a947c5b07944bf07db5fe7ff3c98

选择分支和commit即可选择需要得配置。

查看分支命令如下：

```
duxy@myir-server1:~/L4.19.35/L4.19.35/uboot-imx$ git branch
* imx_v2018.03_4.14.98_2.0.0_ga-1gddr
  imx_v2018.03_4.14.98_2.0.0_ga-2gddr
  imx_v2018.03_4.14.98_2.0.0_ga-2gddr-MT3D1024M
```

切换分支命令为：

```
git checkout imx_v2018.03_4.14.98_2.0.0_ga-2gddr-MT3D1024M
```

查看git commit 命令为：

```
git log
```

选择完需要得配置，就可以通过yocto来编译imx-boot，操作如下：

```
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
```

uboot在编译时的源文件目录在build-xwayland/tmp/work/imx8mqevk-poky-linux/u-boot-imx/1_2019.04-r0/git

请到此目录确认branch 和commit是否已经修改过来。

生成镜像文件在如下位置：

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

3.3 Yocto编译Kernel

kernel是imx-boot启动后引导的程序，用yocto编译出kernel镜像操作如下：

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

编译时的源文件在build-xwayland/tmp/work/imx8mqevk-poky-linux/linux-imx/4.19.35-r0/git

生成镜像文件在如下位置：

build-xwayland/tmp/work/imx8mqevk-poky-linux/linux-imx/4.19.35-r0/build/arch/arm64/boot/Image

kernel 与 uboot类似，是由此文件决定分支，commit值：sources/meta-fsl-bsp-release/imx/meta-bsp/recipes-kernel/linux/linux-imx_4.19.35.bb

```
SRCBRANCH = "myir-master-4.19.35_1.1.1"
SRCREV = "a1f22dcf99e203e32f213ce73d1257c2ee661cec"
```

3.4 Yocto构建SDK工具

本章节介绍如何使用yocto工具构建sdk来编译开发板可以运行的程序。

下载编译链

在3.1章节中，已经解压了yocto源码包，可以选择如下方式来获取对应的工具链资源：

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

下载完成后，资源会存在/tmp/deploy/sdk 下。

```
.
├── fsl-imx-xwayland-glibc-x86_64-meta-toolchain-aarch64-toolchain-4.19-warrior.hos
t.manifest
├── fsl-imx-xwayland-glibc-x86_64-meta-toolchain-aarch64-toolchain-4.19-warrior.sh
├── fsl-imx-xwayland-glibc-x86_64-meta-toolchain-aarch64-toolchain-4.19-warrior.tar
get.manifest
└── fsl-imx-xwayland-glibc-x86_64-meta-toolchain-aarch64-toolchain-4.19-warrior.tes
tdata.json
```

安装工具链

执行命令

```
duxy@myir-server1:~/L4.19.35/L4.19.35/build-xwayland/tmp/deploy/sdk$ ./fsl-imx-xway
land-glibc-x86_64-meta-toolchain-aarch64-toolchain-4.19-warrior.sh
```

结果

```
duxy@myir-server1:~/L4.19.35/L4.19.35/build-xwayland/tmp/deploy/sdk$ ./fsl-imx-xway
land-glibc-x86_64-meta-toolchain-aarch64-toolchain-4.19-warrior.sh
NXP i.MX Release Distro SDK installer version 4.19-warrior
=====
Enter target directory for SDK (default: /opt/fsl-imx-xwayland/4.19-warrior): /home
/duxy/opt_warrior
You are about to install the SDK to "/home/duxy/opt_warrior". 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 en
vironment setup script e.g.
$ ./home/duxy/opt_warrior/environment-setup-aarch64-poky-linux
```

编译开发板可执行文件

执行命令

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

使用file main 查看结果

```
#file main
main: ELF 64-bit LSB executable, ARM aarch64, version 1 (SYSV), dynamically linked,
interpreter /lib/ld-linux-aarch64.so.1, BuildID[sha1]=11ee951a2ce1e58097279db92ee4
aff0fcfe154, for GNU/Linux 3.14.0, not stripped
```

上述的编译链在03-Tools/Toolchain 已提供，另外还提供了QT的编译链工具。

关于 yocto 相关文档可以参考如下链接：

[Yocto Project Quick start](#)

[Bitback User Manual](#)

[Yocto Project Reference Manual](#)

[Yocto Project Development Manual](#)

[Yocto Project Complete Documentation Set](#)

4. Linux 应用开发

本章主要介绍MYD-JX8MX开发板底板外围硬件设备应用例程的使用。使用前，需要先安装Yocto提供的SDK工具链，如未安装请参考3.4小节，再编译所有例程代码，并拷贝至开发板目录下。

编译应用例程

加载工具链到当前终端后，可以查看gcc的版本信息，确认当前环境已正确加载。

```
source ~/opt_warrior/environment-setup-aarch64-poky-linux
aarch64-poky-linux-gcc (GCC) 8.3.0
Copyright (C) 2018 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.
```

编译示例代码，操作如下：

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

然后将整个example目录拷贝到U盘根目录，插到开发板上，以供后面测试使用。

4.1 GPIO-KEY测试

本例演示如何在Linux用户空间读取按键状态和键值。运行gpio_key程序后，按下或释放K3按键，串口会输出相应按键的状态信息。按下"Ctrl-C"可退出程序。

硬件连接

将4章节中存放example的U盘插入开发板。

软件测试

在开发板的控制终端上执行程序,然后按K3键会有code值和pressed/released打印，如下：

```
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测试

本例程演示使用Linux系统API操作开发板上的LED灯，D49,D50。运行程序后，D49 ,D50闪烁。按下"Ctrl-C"可结束程序。

硬件连接

将4章节中存放example的U盘插入开发板。

软件测试

在开发板的控制终端上执行程序，即可看到灯闪烁

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

4.3 USB-HOST测试

使用USB存储设备插入USB HOST(J6 或者J7)接口，调试串口会输出检测设备信息。同时，使用将此存储设备挂载至linux系统下对其读写。

硬件连接

将4章节中存放example的U盘插入开发板。

软件测试

当U盘在开发板上电之后插入，可以看到如下信息。

```
#  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.
```

软件默认会挂载对应设备，查看/proc/mounts挂载点，找到U盘挂载目录。

```
# cat /proc/mounts | grep sda
/dev/sda /run/media/sda vfat rw,relatime,gid=6,fmask=0007,dmask=0007,allow_utime=00
20,codepage=437,iocharset=iso8859-1,shortname=mixed,errors=remount-ro 0 0
```

到U盘挂载点写一个文件，然后把内容读出来，查看结果。

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

4.4 USB-DEVICE测试

本例程演示使用开发板的TypeC接口(J8)作为Device模式，将指定的文件或内存模拟为设备，连接到其它USB HOST设备。这里把文件作为存储设备提供给HOST设备。

硬件连接

使用usb-typeC线，一端接linux PC，另一端接开发板TypeC（J8）接口

软件测试

1. 开发板先创建一个6M的文件，命令如下：

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

2.然后格式化成vfat格式，命令如下：

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

3.最后将此文件模拟成存储设备,命令如下：

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

开发板显示设备为

```
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. Linux PC机上显示有USB设备接入，SerialNumber为"1234",容量是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 测试

本例使用Linux socket API，实现简单的C/S结构，两个程序通过TCP/IP协议栈通信。Linux Pc端使用pc_server 作为服务器端，开发板使用arm_client 作为客户端，进行通讯测试。

硬件连接

将4章节中存放example的U盘插入开发板，使用网线将 linux pc 和 开发板的网口连接起来。

软件测试

编译出来的example/network/pc_server 放到Linux Pc \${HOME}目录，设置一个静态IP为192.168.30.3，并且启动服务程序,操作如下：

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

开发板作为客户端，设置静态ip为 192.168.30.133后 运行arm_client，操作如下：

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

linux 服务器打印如下：

```
REC FROM: 192.168.30.133
```

4.6 Audio测试

本例演示用arecord从line in 录制声音，用aplay从 headphone 播出声音来验证audio功能。

硬件连接

需要使用两头3.5mm的音频AUX线，从电脑音频输出孔和开发板的LINE IN(J10)接口连接，HEADERPHONE(J9)连接耳机。

软件操作

在电脑中播放音频文件，执行arecord命令会先将LINE IN中的音频录制并保存为test.wav文件。运行一分钟后再按ctrl + c来停止。

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

执行aplay命令来播放上面录制好的音频文件。

```
# aplay test.wav
```

4.7 串口测试

本例程演示串口测试程序配置不同波特率收发数据情况，来验证对应的接口正确性。

开发板有3个串口引出来，分别是

串口	对应设备
J18	ttymxc0 默认串口
J19	ttymxc1
J21	ttymxc3

硬件连接

将4章节中存放example的U盘插入开发板，在J19上接另一条串口线。

软件测试

将设备ttymxc1 波特率配置成115200，并且发送“hello”字符串

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

使用串口工具CRT，配置参数如下：

- 波特率：115200
- 数据位: 8bit
- 校验方式：None
- 停止位：1bit
- 流控：Disable

此时可以看到串口有对应打印

```
hellohellohello
```

4.8 SPI测试

本例程展示SPI（J17）接口功能，使用程序进行收发数据测试。

硬件连接

将4章节中存放example的U盘插入开发板。

使用跳线帽将J17的SI和SO连在一起。

软件测试

执行example/spi中的spi_test程序进行测试，操作如下：

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

对应的运行结果

```
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测试

MYD-JX8MX开发板提供两个WiFi/BT模块，8274B-PR和6222B-PRB。

6222B-PRB wifi型号是RTL8822B，为新增wifi/bt模块，工作温度 0~70。

8274B wifi型号为QCA6714。

本章用Client模式连接热点，然后去ping外网来验证wifi功能。

硬件连接

测试前连接天线到U6，以免信号弱导致无法连接。

软件测试

Client模式是用于将WiFi模块作为客户端设备，主动连接于路由器或其它提供无线热点的设备。

- 查看wifi状态

系统中已经加入WiFi模块的驱动，启动后会自动加载相应驱动。驱动加载成功后会出现对应的wlp1s0网络设备，使用ifconfig命令来确认。

```
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)
```

使用rfkill命令来查看wlan的情况，如果关闭（blocked），则打开（unblock）。rfkill查看命令

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

- 打开wlan

打开wlan，可以用rfkill unblock 0 或者rfkill unblock wlan。

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

- 设置wifi热点名称和密码

要连接的wifi名称需要按格式写到/etc/wpa_supplicant.conf 文件中，系统提供了wpa_passphrase命令来解析对应的wifi热点SSID和密码。下面使用wpa_passphrase生成对应WiFi热点SSID的密码，然后由wpa_supplicant命令实现WiFi模块与WiFi热点的连接。

```
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
```

- 杀掉wpa_supplicant进程

```
killall wpa_supplicant
```

- 连接wifi

设置完SSID的账号和密码之后，就可以连接wifi和获取IP地址。连接wifi命令如下：

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

获取ip地址命令如下：

```
udhcpc -i wlp1s0
```

上述2个命令一起执行效果如下：

```
#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
```

- 测试网络

ping百度来检测网络

```
# 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测试

MYD-JX8MX开发板提供一个WiFi/BT模块8274B-PR(U7)，wifi型号为QCA6714，蓝牙是串口模块，与ttymxc2相连，本章用开发板蓝牙连接PC端蓝牙来测试蓝牙功能。

硬件连接

测试前连接天线到U5，以免信号弱导致无法连接。

软件测试

- 绑定串口

使用hciattach命令 来绑定ttymxc2串口，命令如下：

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

请注意，如果使用Fn-link6222b模组，请用如下命令代替

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

- 打开wlan

打开wlan，可以用rfkill unblock 0 或者rfkill unblock wlan。测试时，请先关闭wlan，再打开。

```
#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
```

- 连接蓝牙

使用bluetoothctl工具来控制蓝牙进行扫描和连接。

功能	指令
扫描	scan on
查看扫描设备	devices
配对	pair target_mac
连接	connect target_mac

如下为连接PC端电脑操作：

```
# 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测试

MYD-JX8MX开发板提供一个支持4G模块的PCI-E插槽，此插槽使用USB数据线与4G模块通讯。当前仅支持移远EC20型号。

硬件连接

- 安装移远EC20模块到PCI-E插槽(U29)。
- 安装SMA天线到EC20模块
- 插sim卡到J25插槽
- 插U盘，U盘根目录存放04-source/ppp.tar.gz

软件测试

- 查看4G模块

系统中已经加入4G模块的驱动，启动后会自动加载相应驱动，驱动加载成功后会出现对应的/dev/ttyUSB*设备，如下：

```
# 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
```

- 拷贝4G需要文件到开发板

先将Upan的ppp.tar.gz包解压，然后将PPP内容复制到板子根目录解压ppp包，并且拷贝所有内容到板子根目录

```
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
```

- 拨号上网

执行命令：pppd call quectel-ppp &

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

连接时的log打印如下：

```
[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: EC20CEFDFKGR06A04M2G^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
s = 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_S
RVR"]
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

```

- 测试网络 ping百度来检测网络```

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

```

此时D23灯被点亮, 查看ppp0状态

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 测试

本例程演示RTC功能，通过向RTC写/读时间，以及断电，重启后查看系统时间同步情况来验证RTC功能。

硬件连接

J16安装纽扣电池

软件测试

进行RTC写/读时间测试

1.查看当前时间

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

2.将系统时间写到RTC中

```
hwclock -w
```

3.查看RTC时间

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

4.随意设置一个系统时间

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

5.将RTC时间更新到系统

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

进行RTC同步系统时间测试

上述过程已给RTC设置了时间，现在先断电，过1分钟后再上电，查看系统时间，可以看到时间为RTC对应时间。

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


4.13 M.2接口测试

M.2使用NVME协议，本例采用TOSHIBA RC100 的SSD硬盘测试,规格是nvme2280。

硬件连接

将SSD插入J13

软件测试

kernel已默认添加NVME协议，因此插SSD可以直接使用。

- 系统启动后插看是否识别，挂载 读写即可

```
#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
```

- 读写SSD卡 向SSD挂载点写hello字串到test.txt文件，然后读出来,操作如下：

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

4.14 HDMI测试

本例程对HDMI进行测试，使用gst-play播放视频文件来查看音视频输出。

硬件连接

- HDMI线连接开发板和电视机
- 插入U盘，U盘根目录存放音视频文件

软件测试

软件自带的gst-play可以用来播放视频，如下命令可以将文件对应的音视频呈现在电视机上：

```
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"
```

补充说明：HDMI的分辨率可以在uboot阶段，通过mmcargs将制式传上来，具体操作如下：

1. 连接好串口，并按住enter不放
2. 开发板上电
3. 在mmcargs中添加制式，并保存
4. 重启开发板

输出分辨率，加到bootargs传到kernel即可，默认是720P

```
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测试

开发板上的U37部件tc358775是一个将MIPI-DSI转成LVDS的器件，软件本身有DCSS 和LCDIF 2种输出。本例采用DCSS和LCDIF2种输出来分别测试，所接的屏为10.1寸屏，接J20fpc接口。

硬件连接

10.1寸屏连接到 J20fpc接口

软件测试

- 配置文件说明

在/run/media/mmcblk0p1/中有如下几个由.dtb结尾的配置文件

使用dts文件	功能
myb-fsl-imx8mq-evk.dtb	HDMI 使用DCSS
myb-fsl-imx8mq-evk-dcss-tc358775.dtb	10.1寸屏使用DCSS
myb-fsl-imx8mq-evk-lcdif-tc358775.dtb	10.1寸屏使用LCDIF
myb-fsl-imx8mq-evk-tc358775-dual-display.dtb	HDMI 使用DCSS，屏使用LCDIF

在Uboot阶段，fdt_file 环境默认采用myb-fsl-imx8mq-evk.dtb，本例设置成myb-fsl-imx8mq-evk-lcdif-tc358775.dtb进行测试。

进入uboot，设置fdt_file方法如下：

1.串口工具按住enter

2.开发板上电，进入uboot命令行模式

3.设置fdt_file,保存

4.重启开发板

uboot下设置dtb文件和保存操作如下：

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

到此，10.1寸屏就点亮。

4.16 MIPI-CSI 测试

开发板上的J26和J2分别式2个MIPI-CSI接口，使用myir配套的 ov5640摄像头进行预览和拍照测试，来验证 MIPI-CSI功能。

硬件连接

- 在J26上连接 ov5640摄像头。
- 将4章节中存放example的U盘插入开发板。

软件测试

软件自带的gst-play可以用来作预览，v4l2grab 作拍照。

开发板上有2个CSI接口，J26对应/dev/video0节点，J2对应 /dev/video1节点。

预览

使用J26来做预览，命令如下

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

拍照

U盘example/csi提供的v4l2grab 可以用来拍照，并将照片保存在U盘example/csi/目录中，操作如下：

```
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: width = 640 height = 480
1: width = 720 height = 480
2: width = 1280 height = 720
3: width = 1920 height = 1080
4: width = 2592 height = 1944
5: width = 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
```

双路预览

J26 和J2 同时接入ov5640，可以用gst-play进行双路预览，操作如下：

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

4.17 NXP提供例子

NXP提供了大量例子，包括GPU，VPU等，路径如下

```
/opt$ tree -L 1
.
├── imx-gpu-sdk
├── ltp
└── viv_samples
```

imx-gpu-sdk

```
/opt/imx-gpu-sdk$ tree -L 1
.
├── Console
├── GLES2
├── GLES3
├── OpenCL
├── OpenVG
├── OpenVX
├── Vulkan
└── Window
```

QT相关

```
/usr/share/examples$ tree -L 1
.
├── corelib
├── dbus
├── examples.pro
├── gui
├── network
├── opengl
├── qml
├── qmltest
├── qpa
├── qt3d
├── qtconcurrent
├── qtestlib
├── quick
├── README
├── sql
├── widgets
└── xml
```

vpu解码相关

```
/unit_tests/VPU/hantro$ tree -L 1
.
├── ax170dec
├── g2dec
├── hx170dec
├── jx170dec
├── m2x170dec
├── mx170dec
├── vp6dec
├── vp8x170dec
└── vx170dec
```

5.QT应用开发

Qt是一个跨平台的图形应用开发框架，被应用在不同尺寸设备和平台上，同时提供不同版权版本供用户选择。 MYD-JX8MX使用Qt 5.9.4版本进行应用开发。在Qt应用开发中，推荐使用QtCreator集成开发环境，可以在Linux PC下开发Qt应用，自动化地交叉编译为开发板的ARM架构。

本章使用Yocto构建的SDK工具作为交叉编译系统，配合QtCreator快速开发图形类应用程序。开始本章前，请先完成第三章的Yocto构建过程。或者使用光盘中提供的预编译好的SDK工具包。本章开始前，请安装好应用SDK开发工具。

5.1 安装Qt编译链和QtCreator

Qt编译链安装

Qt编译链在03-Tools/Toolchain/fsl-imx-xwayland-glibc-x86_64-meta-toolchain-qt5-aarch64-toolchain-4.9.88-2.0.0.sh

安装编译链如下：

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

结果为：

```
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 en
vironment setup script e.g.
$ . /home/duxy/opt(Qt)/environment-setup-aarch64-poky-linux
```

QtCreator安装

QtCreator安装包是一个二进制程序，直接执行就可以完成安装。

工具链可以用yocto下载，也可以在03-Tools/Qt中找到 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
```

一直点下一步，按默认安装，或者选着其他目录安装，本例选择安装再用户目录下。

如下命令即可启动qt程序

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


5.2配置QtCreate

想要利用qtcreator编译出板子可用的程序需要进行编译链的配置

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

需要重新配置项：

- 配置一个GCC和G++ 编译链
- 配置一个QTversion
- 配置一个QTdebug
- 新增一个device设备
- 创建一个kit，把上述4项添加到一起组成编译QT配置

1.配置GCC,G++

打开qtcreator之后，先打开一个例程，让"项目"变成可配置，然后点击"项目"->Manage kits，如下图所示。

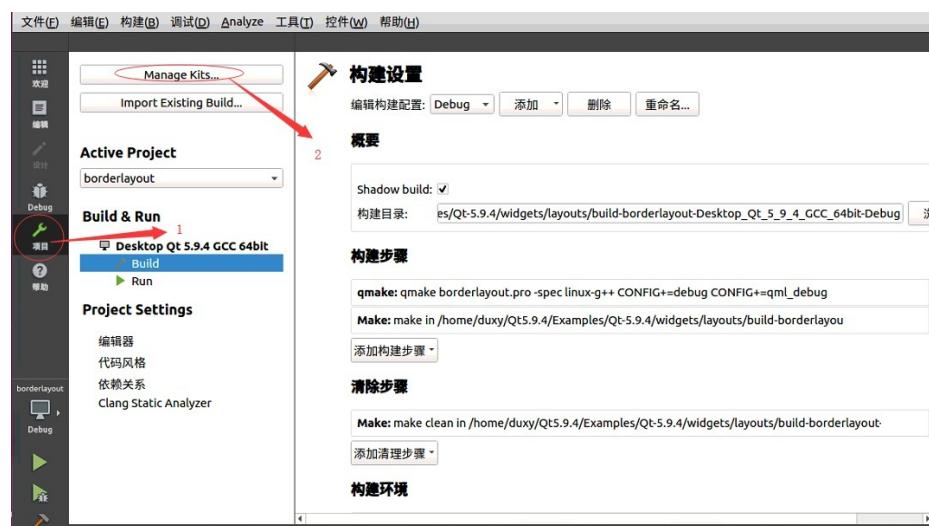


图5-2-1 qtcreate界面

接着在"选项"->编译器->添加"->GCC->C/C++。过程如下图：



图5-2-2 选项界面

接着配置C和C++，配置时需要填“名称”，“编译器路径”，“abi”中选择arm架构

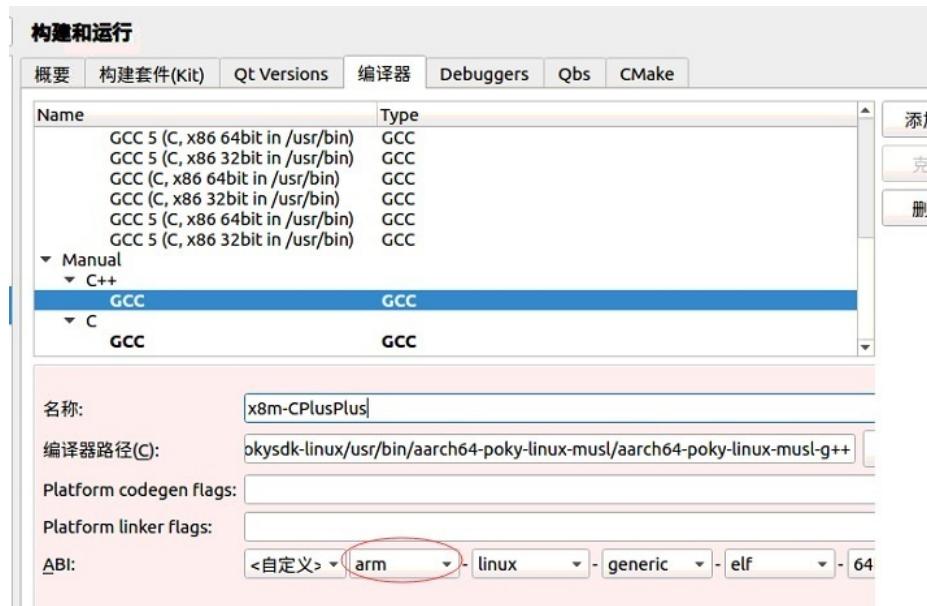


图5-2-3 C++配置

C++的编译器路径为：

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

C的编译器路径为：

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

2. 配置Qtversion

“选项”-> Qt Version -> “添加”，在弹出框中选择如下文件即可

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

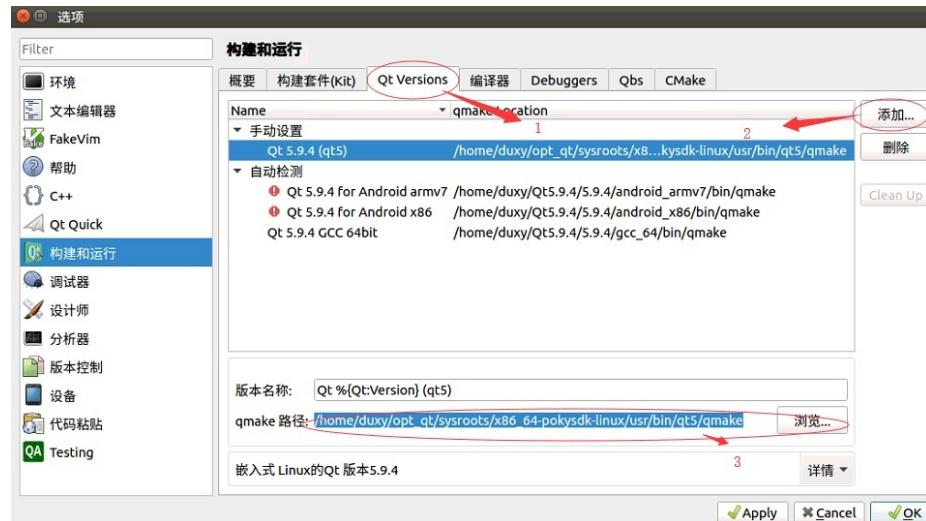


图5-2-4 QTversion配置

3.配置QT debug "选项"->"Debuggers"->"add" , 填写 name 和path。

path路径 :

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

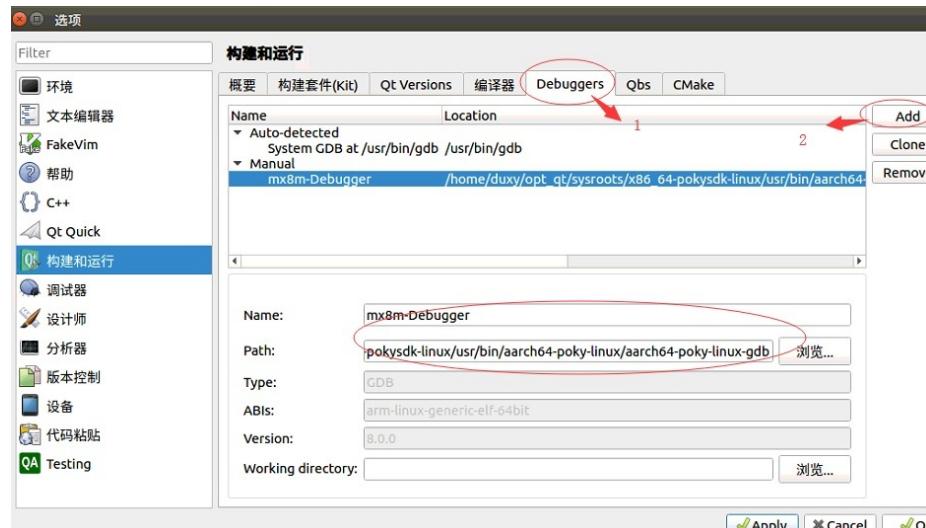


图5-2-5 QTdebug配置

4.新增device设备 "设备"->"添加"->"通用Linux设备" 然后配置名称 , ip地址 , 登陆用户名 , 账号和密码。这里只需要记住配置名称 , ip地址 , 登陆用户名 , 账号密码可以随意填写。

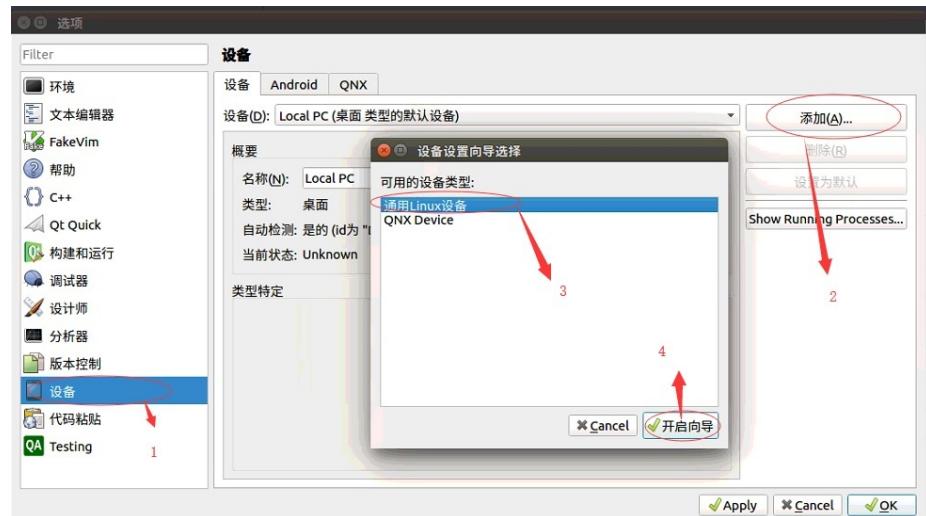


图5-2-6 device配置01

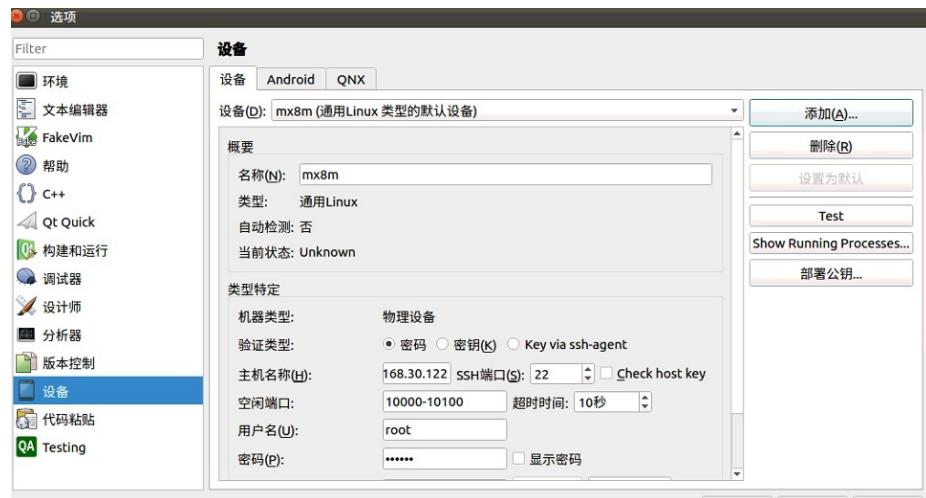


图5-2-7 device配置02

5. 创建kit

"构建和运行"->"构建套件 (kit)" ->"添加"。 这里需要把上诉的4步配置的文件组合到一起，并且填写sysroot:

```
/home/duxxy/opt_qt/sysroots
```



图5-2-8 kit配置

最终完成后如下

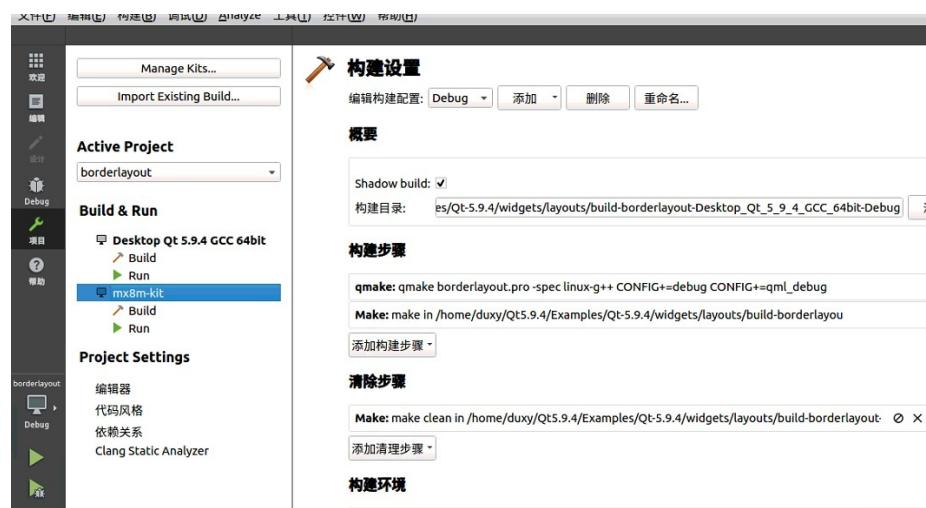


图5-2-9 kit配置完成

5.3测试QT应用

本例编译一个QT例子，并在开发板上运行。

QT例子源码在04-Source/borderlayout.tar.gz。

1.解压文件

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

2.qtcreator 编译例程

运行qtcreator

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

"欢迎"->open project ->选择上述解压的borderlayout.pro文件->"项目"->选择mx8m-kit->编译

接着在"选项"->编译器->"添加"->GCC->C/C++。过程如下图：

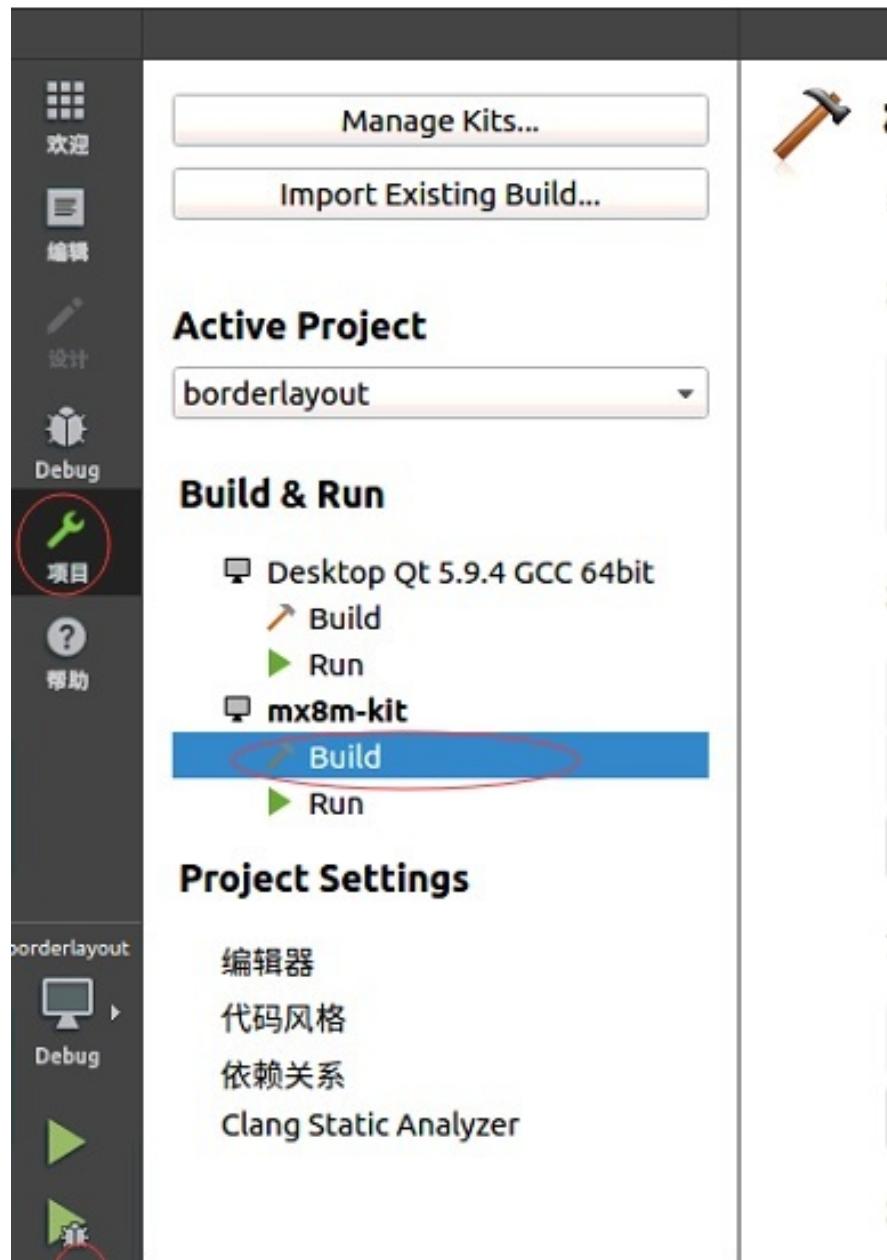


图5-3-1 编译qt

编译完成后，再同级目录下存在build-borderlayout-mx8m_kit-Debug目录。

目录里面的的borderlayout是可执行文件，把此文件拷贝到U盘根目录后，插到开发板，并上电。

3.运行qt例程

开发板上电，执行U盘下的文件

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

可以看到如下界面

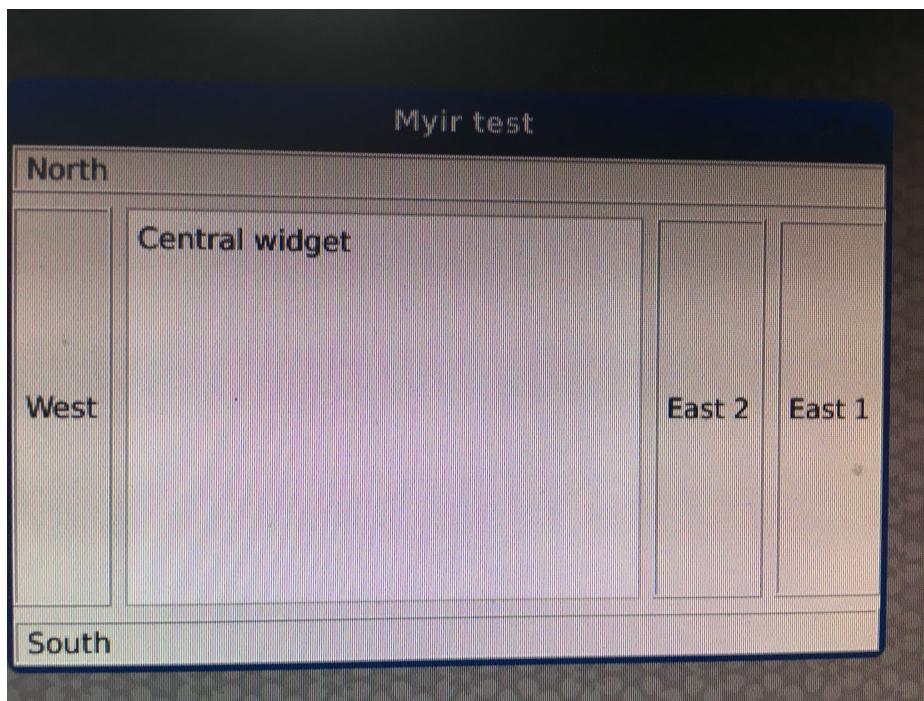


图5-3-2 运行qt

6.系统更新

MYD-JX8MX系列开发板提供了两种更新Linux系统的方法，UUU工具更新和SD卡更新。

UUU更新：

将芯片切换至Download模式，使用UUU工具烧写文件到开发板的EMMC内。需要PC和开发板相连接，适用的调试阶段。

SD卡更新：

设置SD卡启动，启动后从SD卡烧写文件到开发板的EMMC内。适用于量产阶段。

6.1 UUU 更新系统

UUU 是一款支持NXP 芯片在windows和linux端烧录工具， 可以将镜像烧录到开发板的EMMC或者SD卡中。

win10下烧录流程

将开发板设置成下载模式，在windows 的cmd 命令行中输入烧录命令，开发板上电即开始烧录。

1.PC端准备工作

烧录镜像需要如下文件，在yocto编译后即可生成

imx-uboot 引导程序，完整镜像的sdcard包。

文件	编译目录名称
imx-uboot	imx-boot-imx8mqevk-sd.bin-flash_evk
sdcard	imx-image-full-imx8mqevk-20200424001447.rootfs.sdcard.bz2

编译出来的imx-image-full-imx8mqevk-20200424001447.rootfs.sdcard.bz2 需要先解压缩:

```
bzip2 -d imx-image-full-imx8mqevk-20200424001447.rootfs.sdcard.bz2
```

将03-Tools/UUU/UUU4_19_35_MYD-JX8MQ6.zip 放到windows下D盘根目录，并把 imx-boot-imx8mqevk-sd.bin-flash_evk 替换到对应的uboot目录8E1D，8E2D，8E2D_MT53。

imx-image-full-imx8mqevk-20200424001447.rootfs.sdcard 替换 imx-image-full-imx8mqevk.rootfs.sdcard文件。

名称	对应版本
8E1D	1GDDR uboot
8E2D	2GDDR uboot 201906前使用
8E2D_MT53	2GDDR uboot 201906后使用

2.开发板准备工作

开发板断电，将开发板拨码开关SW1调整成EMMC启动，SW2调整成下载模式

SW1 boot dev (1~4)	启动设备
0010	EMMC方式启动
1100	SD卡启动

SW2 boot mode (1~2)	启动方式
00	fuse模式（优先内部启动）
10	下载模式

01	内部模式
11	保留

如果现在要烧EMMC，需要调整成 0010 10

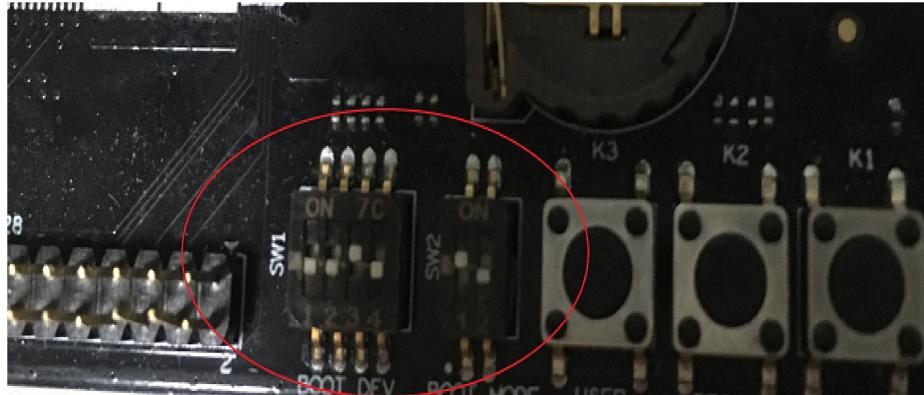


图6-1 设置EMMC下载模式

将typec一端接PC，另一端接板子typeC接口，串口线一端接PC，另一端接板子J18串口（这里串口只起到查看log作用，不接也行）

3. 开始烧录

上电前，在window cmd命令行中执行 如下命令：

```
D:\UUU4_19_35_MYD-JX8MQ6>uuu.exe uuu_8E2D_MT53.auto
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.2.39-0-gdcc404f

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

图6-2 运行UUU

开发板上电

```
D:\UUU4_19_35_MYD-JX8MQ6\uuu.exe uuu_SED_MT53.auto
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.2.39-0-gdcc404f

Success 0 Failure 0
1:4 4/ 7 [ ] FB: flash -raw2sparse all imx-image-full-imx8mqevk.rootfs.sdcard
```

图6-3 UUU开始烧录

烧录完成

图6-4 UUU烧录完成

板子断电，将启动模式改成内部启动（01）后上电，板子可以正常运行

linux 烧录镜像方法

将开发板设置成下载模式sw1(0010) sw2(10),通过typeC线与PC相连。PC端使用lsusb查看节点

```
Bus 002 Device 006: ID 1fc9:012b NXP Semiconductors
```

把此配置写到 /etc/udev/rules.d/51-android.rules

```
SUBSYSTEM=="usb", ATTR{idVendor}=="1fc9", MODE="0777"
```

将uuu 放到/usr/bin 并添加执行权限

解压uuu包。

烧录uboot

```
sudo uuu -b emmc imx-boot-imx8mqevk-sd.bin-flash_evk
```

烧录完成镜像

```
sudo uuu -b emmc_all imx-boot-imx8mqevk-sd.bin-flash_evk imx-image-full-imx8mqevk.romfs.sdcard
```

具体参考《i.MX_Linux_User's_Guide.pdf》 4.2 Universal update utility

6.2 SD卡启动更新

使用mfg工具烧录镜像，必须一个板子配一台电脑，不适合量产。这里介绍使用SD卡进行烧录方式。SD卡烧录步骤为：

- 1.将SD卡烧录镜像烧录到SD卡中
- 2.SD卡插入开发板，并且设置开发板启动模式，SD卡启动
- 3.上电，开始烧录，界面有QT显示的简单进度显示，烧录完成后灯闪烁，进度条显示SUCCESS。

这里介绍制作SD卡烧录包方法

需要准备文件：

准备烧录到EMMC中的 imx-boot，kernel，dtb 和rootfs。

根据EMMC大小，将03-Tools/MksdcardTool/Mksdcard_MYD-JX8MQ6-8E1D-130-E.tar.gz 或者 Mksdcard_MYD-JX8MQ6-8E2D-130-E.tar.gz 在系统下解压.

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

内容如下

```
.
├── factory
├── firmware
├── lib
├── mfgimages  #烧录EMMC 文件目录
├── mkfs.ext3
├── mkfs.ext4
├── mkfs.ext4.real
├── myir_mkscard.sh      #制作升级包脚本
├── ppp
├── README.txt
└── rootfs
```

把所需要烧录的镜像替换到mfgimages后，即可执行

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

生成最后的JX8MX-Update-20190602144250-2gddr.rootfs.scard 文件

用dd if 推到SD卡，或者用03-Tools/MksdcardTool/Win32DiskImager-1.0.0-binary.zip 烧录到SD卡中都行。

将SD卡插到板子，设置SD卡启动后，上电即可运行程序，利用burn_emmc.sh脚本烧录mfgimages下的镜像到EMMC

附录一 联系方式

销售联系方式

- 网址：www.myir-tech.com
- 邮箱：sales.cn@myirtech.com

深圳总部

- 负责区域：广东 / 四川 / 重庆 / 湖南 / 广西 / 云南 / 贵州 / 海南 / 香港 / 澳门
- 电话：0755-25622735 0755-22929657
- 传真：0755-25532724
- 邮编：518020
- 地址：深圳市龙岗区坂田街道发达路云里智能园2栋6楼04室

上海办事处

- 负责区域：上海 / 湖北 / 江苏 / 浙江 / 安徽 / 福建 / 江西
- 电话：021-60317628 15901764611
- 传真：021-60317630
- 邮编：200062
- 地址：上海市普陀区中江路106号北岸长风I座1402

北京办事处

- 负责区域：北京 / 天津 / 陕西 / 辽宁 / 山东 / 河南 / 河北 / 黑龙江 / 吉林 / 山西 / 甘肃 / 内蒙古 / 宁夏
- 电话：010-84675491 13269791724
- 传真：010-84675491
- 邮编：102218
- 地址：北京市昌平区东小口镇中滩村润枫欣尚2号楼1009

技术支持联系方式

- 电话：027-59621648
- 邮箱：support.cn@myirtech.com

如果您通过邮件获取帮助时，请使用以下格式书写邮件标题：

[公司名称/个人--开发板型号]问题概述

这样可以使我们更快速跟进您的问题，以便相应开发组可以处理您的问题。

附录二 售后服务与技术支持

凡是通过米尔科技直接购买或经米尔科技授权的正规代理商处购买的米尔科技全系列产品，均可享受以下权益：

- 1、6个月免费保修服务周期
- 2、终身免费技术支持服务
- 3、终身维修服务
- 4、免费享有所购买产品配套的软件升级服务
- 5、免费享有所购买产品配套的软件源代码，以及米尔科技开发的部分软件源代码
- 6、可直接从米尔科技购买主要芯片样品，简单、方便、快速；免去从代理商处购买时，漫长的等待周期
- 7、自购买之日起，即成为米尔科技永久客户，享有再次购买米尔科技任何一款软硬件产品的优惠政策
- 8、OEM/ODM服务

如有以下情况之一，则不享有免费保修服务：

- 1、超过免费保修服务周期
- 2、无产品序列号或无产品有效购买单据
- 3、进液、受潮、发霉或腐蚀
- 4、受撞击、挤压、摔落、刮伤等非产品本身质量问题引起的故障和损坏
- 5、擅自改造硬件、错误上电、错误操作造成的故障和损坏
- 6、由不可抗拒自然因素引起的故障和损坏

产品返修：用户在使用过程中由于产品故障、损坏或其他异常现象，在寄回维修之前，请先致电米尔科技客服部，与工程师进行沟通以确认问题，避免故障判断错误造成不必要的运费损失及周期的耽误。

维修周期：收到返修产品后，我们将即日安排工程师进行检测，我们将在最短的时间内维修或更换并寄回。一般的故障维修周期为3个工作日（自收到物品之日起，不计运输过程时间），由于特殊故障导致无法短期内维修的产品，我们会与用户另行沟通并确认维修周期。

维修费用：在免费保修期内的产品，由于产品质量问题引起的故障，不收任何维修费用；不属于免费保修范围内的故障或损坏，在检测确认问题后，我们将与客户沟通并确认维修费用，我们仅收取元器件材料费，不收取维修服务费；超过保修期限的产品，根据实际损坏的程度来确定收取的元器件材料费和维修服务费。

运输费用：产品正常保修时，用户寄回的运费由用户承担，维修后寄回给用户的费用由我司承担。非正常保修产品来回运费均由用户承担。

