MYD-JX8MPQ Linux 软件评估指南

文件状态: [] 草稿 [√] 正式发布	**文件标识:**	MYIR-MYD-JX8MPQ-SW-EG-ZH-L5.10.9
	当前版本:	V1.0[DOC]
	作　　者:	Coin.Du
	创建日期:	2021-10-26
	最近更新:	2021-10-26

版 本 历 史

版本	作者	参与者	日期	备注
V1.0.0	Coin.Du		20211026	初始版本：uboot 2020.04， kernel 5.10.9，yocto 3.2.1

目 录

1. 概述

Linux 软件评估指南用于介绍在米尔的开发板上运行开源 Linux 系统下的核心资源与外设资源的测试步骤与评估方法。本文可作为前期评估指南使用，也可以作为通用系统开发的测试指导书使用。

1.1. 硬件资源

本文档适用于米尔电子的 MYD-JX8MPQ 系列板卡，它是基于 NXP 公司的高性能嵌入式 ARM 处理器 i.MX8M Plus 系列开发的一套开发板。关于硬件部分的详细配置参数请查看《MYD-JX8MPQ 产品手册》。同时用户在评估测试过程中会用到一些配件，参见下面的列表。

表 1-1.选配模块

配件	接口方式	说明及链接
摄像头	CSI 接口	MY-CAM003M（500 万像素） http://www.myir-tech.com/product/my_cam003m.htm
LVDS 屏	LVDS 接口	MY-LVDS070C（7 寸 LVDS） http://www.myir-tech.com/product/my-lvds070c.htm
WIFI/BT	插针	MY-WF005S http://www.myir-tech.com/product/MY-WF005S.htm
拓展板（RS485/SPI/CAN）	树莓派	MY-WIREDCOM http://www.myir-tech.com/product/MY-WIREDCOM.htm

1.2. 软件资源

MYD-JX8MPQ 系列开发板的 BSP 是基于 NXP 官方开源社区版 Linux BSP 移植与修改而来，系统镜像采用 Yocto 项目进行构建。Bootloader, Kernel 以及文件系统各部分软件资源全部以源码的形式开放，具体内容请查看《MYD-JX8MPQ SDK 发布说明》。

开发板在出厂时已经烧录了 myir-image-full 镜像，您只需要上电即可使用。

1.3. 文档资源

根据用户使用开发板的各个不同阶段，SDK 中包含了各阶段的文档，发布说明，评估指南，开发指南，应用笔记，常用问答等不同类别的文档和手册。具体的文档列表参见《MYD-JX8MPQ SDK 发布说明》表 2-4 中的说明。

1.4. 环境准备

在开始评估开发板之前，您需要对开发板做一些必要的准备和配置一些环境，包括正确硬件接线，配置调试串口，设置启动等步骤。详细的步骤可以参照《MYD-JX8MPQ 快速入门指南》。

接下来的部分重点介绍如何对系统的硬件资源和接口以及软件功能进行评估和测试。主要借助一些 Linux 下常用的工具和命令，以及自己开发的应用进行测试。软件评估指南分为多个部分来描述，包括：核心资源，外设资源，网络应用，多媒体应用，开发支持应用，系统工具等几大类。后面的章节会针对各个部分做全方位的讲解，并详细描述各部分资源的具体评估方法和步骤。

2. 核心资源

在 Linux 系统中，提供了 proc 虚拟文件系统来查询各项核心资源的参数以及一些通用工具来评估资源的性能。下面将具体对 CPU， memory， e·MMC，RTC 等核心资源的参数进行读取与测试。

2.1. CPU

MYD-JX8MPQ 核心芯片是 MIMX8ML8CVNKZAB/ MIMX8ML8DVNLZAB，是基于高性能四核 Arm®Cortex®-A53 64 位 RISC 核心，商业级频率最高 1.8GHZ，工业级 1.6GHZ。Cortex-A53 处理器的每个 CPU 核心包括一个 32 kbyte L1 指令缓存，一个 32 kbyte L1 数据缓存，一个 512kbyte 二级缓存。设备还嵌入了 Cortex®-M7 32 位 RISC 核心，最高可工作在 800MHz 频率。Cortex-M7 核心功能是浮点单元(FPU)单精度，支持 Arm®单精度数据处理指令和数据类型。Cortex -M7 支持一套完整的 DSP 指令和一个内存保护单元(MPU)增强应用程序安全性。

1) 查看 CPU 信息命令

读取系统中的 CPU 的提供商和参数信息，则可以通过/proc/cpuinfo 文件得到。

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

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

```
CPU part        : 0xd03
CPU revision    : 4

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

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

- processor ：系统中逻辑处理核的编号，对于多核处理器则可以是物理核、或者使用超线程技术虚拟的逻辑核。
- bogomips ：在系统内核启动时粗略测算的 CPU 每秒运行百万条指令数（Million Instructions Per Second）

2) 查看 CPU 使用率

```
root@myd-jx8mp:/sys/bus/cpu/devices/cpu0# top
top - 11:14:27 up 58 min,  2 users,  load average: 0.01, 0.01, 0.00
Tasks: 136 total,   1 running, 135 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.4 us,  2.8 sy,  0.0 ni, 95.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   2649.3 total,   2199.4 free,    338.3 used,    111.6 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.   2209.7 avail Mem
```

```
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME
+ COMMAND
   1790 root      20   0    4980   2352   1992 R   6.2   0.1   0:00.03 top

      1 root      20   0   10512   7256   5392 S   0.0   0.3   0:04.10 syste
md
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.01 kthrea
dd
```

- %us：表示用户空间程序的 cpu 使用率（没有通过 nice 调度）
- %sy：表示系统空间的 cpu 使用率，主要是内核程序。
- %ni：表示用户空间且通过 nice 调度过的程序的 cpu 使用率。
- %id：空闲 cpu

3）获取 CPU 温度信息

CPU 芯片内部有一块 TMU 来检测 CPU 温度，并且测温范围-40~125℃。

```
root@myd-jx8mp:~# cat /sys/class/thermal/thermal_zone0/temp
47000
```

- 显示数字为千分之一度，需除以 1000 就是当前温度值。

4）CPU 压力测试

CPU 的压力的测试方式有很多，可通过 bc 命令来计算圆周率方法来测试 CPU 在运算过程中的稳定性。

```
root@myd-jx8mp:~# echo "scale=5000; 4*a(1)" | bc -l -q &
[1] 2067
```

上述命令将在后台计算的 PI，并精确到小数点后 5000 位。 计算过程需要一段时间。 此时，我们可以通过 top 命令检查 CPU 利用率的变化，如下所示：

```
root@myd-jx8mp:~# top
top - 11:28:07 up  1:12,  2 users,  load average: 0.34, 0.08, 0.03
Tasks: 134 total,   2 running, 132 sleeping,   0 stopped,   0 zombie
%Cpu0  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

```
%Cpu1  :  0.0 us,  0.3 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu2  :  0.0 us,  0.3 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu3  :  0.0 us,  0.3 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   2649.3 total,   2197.8 free,    338.6 used,    112.8 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.   2209.3 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME
+ COMMAND

   2067 root      20   0    2764   1660   1400 R 100.0   0.1   0:24.23 bc


      1 root      20   0   10512   7256   5392 S   0.0   0.3   0:04.25 syste
md

      2 root      20   0       0      0      0 S   0.0   0.0   0:00.01 kthrea
dd

      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp


.....
```

约 1 分钟后，PI 结果被计算出来。 在此期间 CPU 使用率达到 100%，没有发生异常，说明 CPU 压力测试通过。还可以继续增加精确值，可进一步提高测试压力。

```
root@myd-jx8mp:~# 3.14159265358979323846264338327950288419716939937
5105820974944592307\
8164062862089986280348253421170679821480865132823066470938446095505
8\
2231725359408128481117450284102701938521105559644622948954930381964
4\
2881097566593344612847564823378678316527120190914564856692346034861
0\
```

```
45432664821339360726024914127372458700660631558817488152092096282925\
40917153643678925903600113305305488204665213841469519415116094330572\
70365759591953092186117381932611793105118548074462379962749567351885\
75272489122793818301194912983367336244065664308602139494639522473719\
07021798609437027705392171762931767523846748184676694051320005681271\
45263560827785771342757789609173637178721468440901224953430146549585\
37105079227968925892354201995611212902196086403441815981362977477130\
99605187072113499999983729780499510597317328160963185950244594553469\
08302642522308253344685035261931188171010003137838752886587533208381\
42061717766914730359825349042875546873115956286388235378759375195778\
18577805321712268066130019278766111959092164201989380952572010654858\
63278865936153381827968230301952035301852968995773622599413891249721\
77528347913151557485724245415069595082953311686172785588907509838175\
.....
```

5）CPU 工作频率

CPU 支持动态调频，也支持固定频率，并且 4 个 A53 核是同步操作。

- **查看 CPU 的工作频率**

```
root@myd-jx8mp:~# cat /sys/bus/cpu/devices/cpu0/cpufreq/cpuinfo_cur_freq
1200000
```

- **查看 CPU 最大工作频率**

```
root@myd-jx8mp:~# cat /sys/bus/cpu/devices/cpu0/cpufreq/cpuinfo_max_freq
1800000
```

- **查看 CPU 的所有工作模式**

```
root@myd-jx8mp:~# cat /sys/bus/cpu/devices/cpu0/cpufreq/scaling_available_governors
conservative ondemand userspace powersave performance schedutil
```

CPU 的这几种模式的介绍：

- Conservative: governor 会在 cpu loading 超过向上阈值是一点一点的增加 cpu freq。如果低于向下的阈值，则会减小频率。
- userspace：用户模式，任何情况下都会控制 CPU 运行在配置的频率范围内,配置中的用户自己添加的省电设置。在此情景模式下,降低 CPU 最大运行频率可以延长电池待机时间,但同时也会降低机器的唤醒速度,建议最好不使用该选项。
- ondemand:指的是平时以低速方式运行，当系统负载提高时候自动提高频率。以这种模式运行不会因为降频造成性能降低，同时也能节约电能和降低温度。
- powersave：省电模式，通常以最低频率运行。
- performance:高性能模式,高性能模式,按你设定范围的最高频率运行,即使系统负载非常低 cpu 的频率也为最高。性能很好,因为 CPU 本身不需要资源去调整频率,但是电量消耗较快,温度也高一些。
- Schedutil:基于调度程序调整 CPU 频率。

- **查看 CPU 的当前工作模式**

```
root@myd-jx8mp:~# cat /sys/bus/cpu/devices/cpu0/cpufreq/scaling_governor
ondemand
```

这里可以通过设置 CPU 的 performance 模式，让 CPU 一直处于最高频率。

```
root@myd-jx8mp~# echo "performance" > /sys/bus/cpu/devices/cpu0/cpufreq/scaling_governor
root@myd-jx8mp:~# cat /sys/bus/cpu/devices/cpu0/cpufreq/cpuinfo_cur_freq
1800000
```

2.2. M7

MYD-JX8MPQ 配备了一颗异构的 Cortex-M7 协处理器。可以同时运行 Linux 和 RTOS。本节主要介绍协处理器 M7 使用方法。

M7 在运行时可能会涉及到和 A53 核共用资源，这里列举出会冲突资源如下：ECSPI0/ECSPI2, FLEXCAN, GPIO1/GPIO5, GPT1, I2C3, I2S3, UART4, PWM4, SDMA1/SDMA2， 所以在一起使用时，需要将 A53 的这些资源关闭，这里就需要用到 myd-jx8mp-rpmsg.dtb 设备树。这里的 M7 使用 uart4 作为串口。

这里介绍如何使用 M7，以及编译 M7 程序的方法。

1）M7 使用方法

外接 2 个串口，一个接上开发板的 debug 串口，另一个接 M7 的 uart4 串口，设置好串口参数。

- **查看分区信息**

启动开发板按任意键进入 uboot 模式，查看 vfat 分区中存在的文件。

```
u-boot=> fatls mmc 2
 29209088    Image
     8208    imx8mp_m7_TCM_hello_world.bin
    19040    imx8mp_m7_TCM_rpmsg_lite_pingpong_rtos_linux_remote.bin
    18528    imx8mp_m7_TCM_rpmsg_lite_str_echo_rtos.bin
    40948    imx8mp_m7_TCM_sai_low_power_audio.bin
    62728    myd-jx8mp-atk-10.dtb
    61615    myd-jx8mp-base.dtb
    62728    myd-jx8mp-hontron-7.dtb
    62759    myd-jx8mp-lt8912.dtb
    62468    myd-jx8mp-m190etn01-19.dtb
  2113024    tee.bin
    62532    myd-jx8mp-rpmsg.dtb

12 file(s), 0 dir(s)
u-boot=>
```

- imxmp_m7_TCM_hello_world.bin : m7 程序

- imx8mp_m7_TCM_rpmsg_lite_pingpong_rtos_linux_remote.bin ： m7 程序
- imx8mp_m7_TCM_rpmsg_lite_str_echo_rtos.bin ： m7 程序
- imx8mp_m7_TCM_sai_low_power_audio.bin ： m7 程序
- myd-jx8mp-rpmsg.dtb ： m7 设备树

- **设置 M7 设备树**

kernel 加载的设备树由 fdt_file 变量决定，这里设置成 m7 专用设备树。

```
u-boot=> printenv fdt_file
fdt_file= myd-jx8mp-base.dtb
u-boot=> setenv fdt_file myd-jx8mp-rpmsg.dtb
u-boot=> save
Saving Environment to MMC... Writing to MMC(2)... OK
u-boot=> printenv fdt_file
fdt_file= myd-jx8mp-rpmsg.dtb
u-boot=>
```

- **设置 m7 启动参数**

M7 启动流程也是加载 m7 程序到内存，用 bootaux 命令启动， 正常 kernel 启动也一样，只是最后用 bootm 启动。

```
u-boot=> setenv m7_image imx8mp_m7_TCM_rpmsg_lite_str_echo_rtos.bin
u-boot=> setenv m7_boot_temp_addr 0x48000000
u-boot=> setenv m7_boot_addr 0x7E0000
u-boot=> setenv m7_run 'fatload mmc ${mmcdev}:${mmcpart} ${m7_boot_tem
p_addr} ${m7_image};cp.b ${m7_boot_temp_addr} ${m7_boot_addr} 0x20000; bo
otaux ${m7_boot_addr}'
u-boot=> setenv mmcboot "run m7_run;${mmcboot}"
u-boot=> save
Saving Environment to MMC... Writing to MMC(2)... OK
u-boot=>
```

- **测试 m7 程序**

此时已经设置好了 m7 启动，只需要重启开发板，那么在 A53 启动 kernel 同时，也会启动 m7 中的程序。启动之后 A53 的串口执行如下 2 句，既可以看到 M7 中串口出现对应的打印。

A53 串口：

```
myd-jx8mp login: root
root@myd-jx8mp:~# modprobe imx_rpmsg_tty
[    28.720911] imx_rpmsg_tty virtio0.rpmsg-virtual-tty-channel-1.-1.30: new channel: 0x400 -> 0x1e!
[    28.730126] Install rpmsg tty driver!
[    28.735080] imx-audio-rpmsg sound-rpmsg: assigned reserved memory node audio@0x81000000
root@myd-jx8mp:~# [    28.743278] imx-audio-rpmsg sound-rpmsg: snd_soc_register_card failed (-517)
[    28.752846] mx8-img-md: Registered mxc_isi.0.capture as /dev/video0
[    28.759460] mx8-img-md: Registered mxc_isi.1.capture as /dev/video1
[    28.765933] mx8-img-md: Can't find i2c client device for ov5640_mipi@3c
[    28.772828] mx8-img-md: Unregistered all entities
[    33.759401] can1-stby: disabling
[    33.762641] can2-stby: disabling
[    33.765906] VSD_3V3: disabling
[    33.769037] m2_keyb_pwr: disabling
root@myd-jx8mp:~# echo "hi m7!" > /dev/ttyRPMSG30
```

M7 串口：

```
Get Message From Master Side : "hello world!" [len : 12]
Get Message From Master Side : "hi m7!" [len : 6]
Get New Line From Master Side
```

2）M7demo 编译

可以从官网获取 M7 源码和 M7 工具链，也可以直接从 04-Source/m7 路径下载下面两个文件：

- m4 源码 SDK_2_10_0_EVK-MIMX8MP.tar.gz
- m4 工具链 gcc-arm-none-eabi-7-2017-q4-major-linux.tar.bz2

这里主要介绍如何编译 demo。

```
duxy@myir_server:~/tools/m7$ ls -l
total 116304
-rwxr--r-- 1 duxy duxy 99857645 3 月  15  2021 gcc-arm-none-eabi-7-2017-q4-major-linux.tar.bz2
-rwxr--r-- 1 duxy duxy 19233326 10 月 25 17:22 SDK_2_10_0_EVK-MIMX8MP.tar.gz
```

- **解压文件**

  解压 M7 源码和编译链

```
duxy@myir_server:~/tools/m7$ mkdir ~/toolchains
duxy@myir_server:~/tools/m7$ tar -jxf gcc-arm-none-eabi-7-2017-q4-major-linux.tar.bz2 -C ~/toolchains/
duxy@myir_server:~/tools/m7$ mkdir SDK_2_10_0_EVK-MIMX8MP
duxy@myir_server:~/tools/m7$ tar -zxf SDK_2_10_0_EVK-MIMX8MP.tar.gz -C SDK_2_10_0_EVK-MIMX8MP/
duxy@myir_server:~/tools/m7
```

- **加载编译链**

  这里需要 export ARMGCC_DIR 的值

```
duxy@myir_server:~/tools/m7$ export ARMGCC_DIR=~/toolchains/gcc-arm-none-eabi-7-2017-q4-major
duxy@myir_server:~/tools/m7$ env | grep ARMGCC
ARMGCC_DIR=/home/duxy/toolchains/gcc-arm-none-eabi-7-2017-q4-major
```

- **编译 M7demo**

  到编译目录，然后执行 clean.sh ，build_debug.sh，就可以编译出生成文件。

```
duxy@myir_server:~/tools/m7$ cd SDK_2_10_0_EVK-MIMX8MP/boards/evkmimx8mp/demo_apps/hello_world/armgcc/
duxy@myir_server:~/tools/m7/SDK_2_10_0_EVK-MIMX8MP/boards/evkmimx8mp/demo_apps/hello_world/armgcc$ ./clean.sh
duxy@myir_server:~/tools/m7/SDK_2_10_0_EVK-MIMX8MP/boards/evkmimx8mp/demo_apps/hello_world/armgcc$ ./build_debug.sh
-- TOOLCHAIN_DIR: /home/duxy/toolchains/gcc-arm-none-eabi-7-2017-q4-major
```

```
-- BUILD_TYPE: debug
-- TOOLCHAIN_DIR: /home/duxy/toolchains/gcc-arm-none-eabi-7-2017-q4-maj
or
-- BUILD_TYPE: debug
-- The ASM compiler identification is GNU
-- Found assembler: /home/duxy/toolchains/gcc-arm-none-eabi-7-2017-q4-maj
or/bin/arm-none-eabi-gcc
-- The C compiler identification is GNU 7.2.1
-- The CXX compiler identification is GNU 7.2.1
driver_clock component is included.
driver_common component is included.
device_MIMX8ML8_CMSIS component is included.
CMSIS_Include_core_cm component is included.
driver_rdc component is included.
driver_audiomix component is included.
utility_debug_console component is included.
component_serial_manager component is included.
component_serial_manager_uart component is included.
driver_iuart component is included.
component_lists component is included.
component_iuart_adapter component is included.
device_MIMX8ML8_startup component is included.
device_MIMX8ML8_system component is included.
utility_assert component is included.
utilities_misc_utilities component is included.
-- Configuring done
-- Generating done
-- Build files have been written to: /home/duxy/tools/m7/SDK_2_10_0_EVK-MI
MX8MP/boards/evkmimx8mp/demo_apps/hello_world/armgcc
Scanning dependencies of target hello_world.elf
[  4%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SD
K_2_10_0_EVK-MIMX8MP/boards/evkmimx8mp/demo_apps/hello_world/pin_mux.
c.obj
```

```
[ 13%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/boards/evkmimx8mp/demo_apps/hello_world/empty_rsc_table.c.obj
[ 18%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/components/lists/fsl_component_generic_list.c.obj
[ 18%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/devices/MIMX8ML8/drivers/fsl_clock.c.obj
[ 22%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/devices/MIMX8ML8/drivers/fsl_audiomix.c.obj
[ 27%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/components/uart/fsl_adapter_iuart.c.obj
[ 31%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/devices/MIMX8ML8/system_MIMX8ML8_cm7.c.obj
[ 36%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/devices/MIMX8ML8/utilities/debug_console/fsl_debug_console.c.obj
[ 40%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/devices/MIMX8ML8/drivers/fsl_common_arm.c.obj
[ 50%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/boards/evkmimx8mp/demo_apps/hello_world/clock_config.c.obj
[ 50%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/components/serial_manager/fsl_component_serial_manager.c.obj
[ 54%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/boards/evkmimx8mp/demo_apps/hello_world/hello_world.c.obj
[ 59%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/devices/MIMX8ML8/utilities/fsl_sbrk.c.obj
[ 63%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/devices/MIMX8ML8/drivers/fsl_uart.c.obj
[ 68%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/devices/MIMX8ML8/drivers/fsl_common.c.obj
```

```
[ 72%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/devices/MIMX8ML8/drivers/fsl_rdc.c.obj
[ 77%] Building ASM object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/devices/MIMX8ML8/gcc/startup_MIMX8ML8_cm7.S.obj
[ 86%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/boards/evkmimx8mp/demo_apps/hello_world/board.c.obj
[ 86%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/devices/MIMX8ML8/utilities/fsl_assert.c.obj
[ 90%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/devices/MIMX8ML8/utilities/str/fsl_str.c.obj
[ 95%] Building C object CMakeFiles/hello_world.elf.dir/home/duxy/tools/m7/SDK_2_10_0_EVK-MIMX8MP/components/serial_manager/fsl_component_serial_port_uart.c.obj
[100%] Linking C executable debug/hello_world.elf
Memory region           Used Size  Region Size  %age Used
    m_interrupts:        680 B        1 KB     66.41%
          m_text:      11660 B      127 KB      8.97%
          m_data:       2240 B      128 KB      1.71%
        m_data2:          0 GB       16 MB      0.00%
[100%] Built target hello_world.elf
```

2.3. Graphic

MYD-JX8MPQ 芯片内部有 GPU 模块，支持 2D，3D 加速，OpenGL ES1.1 ,2.0,3.0,3.1, OpenCL 1.2。

具体操作历程参考后续 6.1 图形图像处理章节。

2.4. Memory

MYD-JX8MPQ 内存有 2G/3G/4G 可选，系统会把内存分成设备内存(CMA)和系统内存(MEM)。设备内存是给驱动程序使用的一段连续空间，系统内存是给用户态分配空间。

1）查看内存信息

读取系统中的内存的参数信息，则可以通过/proc/meminfo 文件得到。

```
root@myd-jx8mp:~# cat /proc/meminfo
MemTotal:        2712880 kB
MemFree:         2284640 kB
MemAvailable:    2275504 kB
Buffers:            9816 kB
Cached:            57048 kB
SwapCached:            0 kB
Active:            26064 kB
Inactive:          70356 kB
Active(anon):        932 kB
Inactive(anon):    38788 kB
Active(file):      25132 kB
Inactive(file):    31568 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:                 4 kB
Writeback:             0 kB
AnonPages:         29564 kB
Mapped:            26652 kB
Shmem:             10168 kB
KReclaimable:      15024 kB
Slab:              39508 kB
SReclaimable:      15024 kB
```

```
SUnreclaim:            24484  kB
KernelStack:            2192  kB
PageTables:             2108  kB
NFS_Unstable:              0  kB
Bounce:                    0  kB
WritebackTmp:              0  kB
CommitLimit:         1356440  kB
Committed_AS:          85684  kB
VmallocTotal:     135290159040  kB
VmallocUsed:            8132  kB
VmallocChunk:              0  kB
Percpu:                 1248  kB
HardwareCorrupted:         0  kB
AnonHugePages:             0  kB
ShmemHugePages:            0  kB
ShmemPmdMapped:            0  kB
FileHugePages:             0  kB
FilePmdMapped:             0  kB
CmaTotal:             983040  kB
CmaFree:              718800  kB
HugePages_Total:           0
HugePages_Free:            0
HugePages_Rsvd:            0
HugePages_Surp:            0
Hugepagesize:           2048  kB
Hugetlb:                   0  kB
.............
```

部分参数说明：

- MemTotal ：所有可用的 RAM 大小，物理内存减去预留位和内核使用
- MemFree ：LowFree + HighFree
- Buffers ：用来给块设备做缓存的大小
- Cached ：文件的缓冲区大小
- SwapCached ：已经被交换出来的内存。与 I/O 相关

- Active ：经常（最近）被使用的内存
- Inactive ：最近不常使用的内存。

2）获取内存使用率

可使用 free 命令来读取内存的使用情况，-m 参数代表单位为 MByte。

```
root@myd-jx8mp:~# free -m
               total        used        free      shared  buff/cache   available
Mem:            2649         338        2230           9          80        2221
Swap:              0           0           0           0           0           0
```

- total ： 内存总量。
- used ： 被使用的内存量。
- free ： 可使用的内存量。

3）内存压力测试

通过给定测试内存的大小和次数，可以对系统现有的内存进行项目上的测试。也方便做压力测试，如指定内存测试 20 次等。内核自带的 memtester 工具，测试方法如下。

```
root@myd-jx8mp:~# memtester 300M 1
memtester version 4.5.0 (64-bit)
Copyright (C) 2001-2020 Charles Cazabon.
Licensed under the GNU General Public License version 2 (only).

pagesize is 4096
pagesizemask is 0xfffffffffffff000
want 300MB (314572800 bytes)
got  300MB (314572800 bytes), trying mlock ...locked.
Loop 1/1:
  Stuck Address       : ok
  Random Value        : ok
  Compare XOR         : ok
  Compare SUB         : ok
```

```
  Compare MUL          : ok
  Compare DIV          : ok
  Compare OR           : ok
  Compare AND          : ok
  Sequential Increment: ok
  Solid Bits           : ok
  Block Sequential     : ok
  Checkerboard         : ok
  Bit Spread           : ok
  Bit Flip             : ok
  Walking Ones         : ok
  Walking Zeroes       : ok

Done.
```

2.5. e•MMC

eMMC 是一个数据存储设备，包括一个 MultiMediaCard (MMC)接口，一个 NAND Flash 组件。它的成本、体积小、Flash 技术独立性和高数据吞吐量使其成为嵌入式产品的理想选择。MYD-JX8MPQ 配备有一个 8G 容量的 eMMC。

1) 查看 emmc 容量

通过 fdisk -l 命令可以查询到 emmc 分区信息及容量。

```
root@myd-jx8mp:~# fdisk -l
Disk /dev/mmcblk2: 7.28 GiB, 7820083200 bytes, 15273600 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x7d83bf39

Device          Boot  Start      End  Sectors  Size Id Type
/dev/mmcblk2p1        20480   225279   204800  100M  c W95 FAT32 (LBA)
/dev/mmcblk2p2       225280 15273599 15048320  7.2G 83 Linux
```

- /dev/mmcblk2p1 ： 用来存放 kernel 和 dtb 文件
- /dev/mmcblk2p2 ： 用于存放文件系统

这里/dev/mmcblk2p1 起使在 20480 块开始，前面还保存着 bootloader 和分区表信息。

2) 查看 eMMC 分区信息

通过 df 命令可以查询到 eMMC 分区信息，使用情况，挂载目录等信息。

```
root@myd-jx8mp:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       6.7G  2.9G  3.5G  45% /
devtmpfs        844M  4.0K  844M   1% /dev
tmpfs           1.3G     0  1.3G   0% /dev/shm
tmpfs           530M  9.2M  521M   2% /run
tmpfs           4.0M     0  4.0M   0% /sys/fs/cgroup
```

```
tmpfs                1.3G   4.0K   1.3G    1% /tmp
tmpfs                1.3G   184K   1.3G    1% /var/volatile
/dev/mmcblk2p1   100M    31M    70M   31% /run/media/mmcblk2p1
tmpfs                265M   4.0K   265M    1% /run/user/0
```

- /dev/root ：根文件系统，挂载到根目录下。
- tmpfs ：内存虚拟文件系统，挂载到不同的目录下。
- devtmpfs ： 用于系统创建 dev
- /dev/mmcblk2p1：用来存放 kernel 和 dtb 文件，默认挂载在 /run/media/mmcblk2p1 目录

3）eMMC 的性能测试

性能测试主要测试 emmc 在 linux 系统下对文件的读写速度上的测试，测试一般结合 time 与 dd 双命令进行测试。

- **写文件测试**

```
root@myd-jx8mp:~# time dd if=/dev/zero of=tempfile bs=1M count=100 con
v=fdatasync
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 2.83643 s, 37.0 MB/s

real    0m2.842s
user    0m0.001s
sys     0m0.450s
```

使用 dd 命令写文件时，需要加 conv=fdatasync 参数，表示当 dd 写 N 次结束之后，会 flush cache 同步到磁盘。因为对磁盘的写一般是先写到缓存还没有写到磁盘就返回了。这里测试出写磁盘速度为 37.0MB/s。

- **读文件测试**

在嵌入式系统中，经常需要测试系统文件读写性能，读文件时忽略 cache 的影响。这时可以指定参数 iflag=direct，nonblock。

```
root@myd-jx8mp:~# time dd if=tempfile of=/dev/null bs=1M count=100 iflag
=direct,nonblock
100+0 records in
```

```
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.347186 s, 302 MB/s

real    0m0.351s
user    0m0.001s
sys     0m0.019s
```

测试出,读速度为 302MB/s。

2.6. QSPI

MYD-JX8MPQ 拥有一颗 32M 的 QSPI Flash，可以用来保存数据，这里请注意它的一个特性。写的区域必须是擦除后的区域；擦除是按 block（0x10000)作为最小单位。

1）文件系统下读写

下面将演示在文件系统下的测试步骤，下面是一些 QSPI Flash 读写的命令可以供参考

- hexdump :查看分区内容
- mtd_debug read：用来读取 qspi 数据到文件
- mtd_debug wirte：用来写入文件数据到 qspi
- mtd_debug erase ：擦除 qspi 数据

- **查看存储内容**

使用 hexdump 来查看 QSPI 存储的内容：

```
root@myd-jxmp:~# hexdump -C /dev/mtd0
00000000  77 51 49 30 39 50 48 75  6e 70 ff ff ff ff ff ff  |wQI09PHunp......|
00000010  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
02000000
```

- **写数据**

通过 mtd_debug wirte 命令向 QSPI 写数据，并查看写入内容：

```
root@myd-jx8mp:~# hexdump -C /dev/mtd0
00000000  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
02000000
root@myd-jx8mp:~# echo "myir qspi test" > qspi_test.txt
root@myd-jx8mp:~# mtd_debug write /dev/mtd0 0x10 14 qspi_test.txt
Copied 14 bytes from qspi_test.txt to address 0x00000010 in flash
root@myd-jx8mp:~# hexdump -C /dev/mtd0
00000000  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
00000010  6d 79 69 72 20 71 73 70  69 20 74 65 73 74 ff ff  |myir qspi tes
t..|
```

```
00000020  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
02000000
```

- **读数据**

通过 mtd_debug read 命令读取 QSPI 数据：

```
root@myd-jx8mp:~# mtd_debug read /dev/mtd0 0x10 14 read.txt
Copied 14 bytes from address 0x00000010 in flash to read.txt
root@myd-jx8mp:~# cat read.txt
myir qspi testroot@myd-jx8mp:~#
```

2) uboot 下读写 qspi

下面将演示在 uboot 阶段读取 qspi 步骤，uboot 下内存操作指令一般是 mw 和 md。

- **Uboot 内存读取命令**

  - **mw 命令**

uboot 下输入指令 mw,会提示 mw 的用法，内存写命令：

```
u-boot=> mw
mw - memory write (fill)
Usage:
mw [.b, .w, .l, .q] address value [count]
```

其中 b:8 位 w:16 位 l:32 位（默认值），代表每次写入的位数；address 是要写入内存的地址，value 是要写入的值，count 是从 address 开始要写入多少个数，这些都是 16 进制数。

  - **md 命令**

uboot 下输入指令 md,会提示 md 的用法，内存读命令：

```
u-boot=> md
md - memory display
Usage:
md [.b, .w, .l, .q] address [# of objects]
u-boot=>
```

其中 b/w/l 的意思同上，默认是以 32 位输出。

- **初始化 QSPI**

这里需要用到 SPI flash 子系统的 sf 命令，uboot 下输入 sf，会提示 sf 命令的用法。

上电按任意键进入 uboot shell 界面，执行以下命令初始化 qspi：

```
u-boot=>
u-boot=> sf probe
fsl_fspi_nor: fail to find cmd 0x1 from lut
SF: Detected gd25lq256 with page size 256 Bytes, erase size 4 KiB, total 32 MiB
```

- **清空内存后读取数据**

清空一段内存后读取 qspi 内容：

```
u-boot=> mw.b 0x48000000 0xff 0x10
u-boot=> md 0x48000000 0x10
48000000: ffffffff ffffffff ffffffff ffffffff    ................
48000010: 28000000 11000000 10000000 00000000    ...(............
48000020: 73000000 5c030000 00000000 00000000    ...s...\........
48000030: 00000000 00000000 01000000 00000000    ................
u-boot=> sf read 0x48000000 0x10 0x10
device 0 offset 0x10, size 0x10
SF: 16 bytes @ 0x10 Read: OK
u-boot=> md 0x48000000 0x10
48000000: 7269796d 70737120 65742069 ffff7473    myir qspi test..
48000010: 28000000 11000000 10000000 00000000    ...(............
48000020: 73000000 5c030000 00000000 00000000    ...s...\........
48000030: 00000000 00000000 01000000 00000000    ................
u-boot=>
```

这里用 sf read 命令从 QSPI 偏移地址 0x10 处读取 16 字节到内存 0x48000000 地址处。由于先前已经在 QSPI 的 0x10 地址写了数据，所以这里能看到读取了文件系统中写入的“myir qspi test“内容。

- **清空内存后写数据**

清空内存后，向 qspi 的 0x100 偏移处写入 16 个字节的 0x12：

```
u-boot=> mw.b 0x48000000 0xff 0x10
u-boot=> mw.b 0x48000000 0x12 0x10
u-boot=> md 0x48000000 0x10
```

```
48000000: 12121212 12121212 12121212 12121212    ................
48000010: 00000000 00000000 00000000 00000000    ................
48000020: 00000000 00000000 00000000 00000000    ................
48000030: 00000000 00000000 00000000 00000000    ................
u-boot=> sf write 0x48000000 0x100 0x10
device 0 offset 0x100, size 0x10
SF: 16 bytes @ 0x100 Written: OK
u-boot=>
```

- **清空内存后再次读取**

```
u-boot=> mw.b 0x48000000 0xff 0x10
u-boot=> sf read 0x48000000 0x100 0x10
device 0 offset 0x100, size 0x10
SF: 16 bytes @ 0x100 Read: OK
u-boot=> md 0x48000000 0x10
48000000: 12121212 12121212 12121212 12121212    ................
48000010: 28000000 11000000 10000000 00000000    ...(............
48000020: 73000000 5c030000 00000000 00000000    ...s...\........
48000030: 00000000 00000000 01000000 00000000    ................
u-boot=>
```

可以看到能正常从 QSPI 读出了写入的数据，这里也可以参考文件系统读的方法来查看 qspi 内容。

2.7. RTC

MYD-JX8MPQ 拥有内部 RTC(snvs),以及外部 RTC（RX8025），RTC 的功能是在系统开机后，能同步真实时间。软件使用外部 RTC(RX8025)，内部 RTC 被关闭。

1）查看系统 RTC 设备

```
root@myd-jx8mp:~# ls -l /dev/rtc*
lrwxrwxrwx 1 root root        4 Sep 20 10:24 /dev/rtc -> rtc0
crw------- 1 root root 251, 0 Sep 20 10:24 /dev/rtc0
```

rtc 属于 linux 设备，在/dev 下有其设备节点 rtc0 可供用户操作。

```
root@myd-jx8mp:~#  cat /sys/class/rtc/rtc0/name
rtc-rx8025 1-0032
```

2）设置系统时间

在将系统时间设置为 Wed Nov 18 09:07:30 UTC 2020：

```
root@myd-jx8mp:~# date  111809072020.30
Wed Nov 18 09:07:30 UTC 2020
```

3）将系统时间写入 RTC

将上一步 date 命令设置的系统时间写入到 RTC 设备

```
root@myd-jx8mp:~# hwclock -w
```

4）读取 RTC 时间并设置为系统时间

```
root@myd-jx8mp:~# hwclock -r
2020-11-18 09:08:04.693523+00:00
```

5）掉电保持 RTC 时间，

将设备按下 ON/OFF 键，不要断开电源，经过 5 分钟左右，按 ON/OFF 上电。查看 RTC 时间和系统时间。

```
root@myd-jx8mp:~# hwclock -r
2020-11-18 09:15:01.982115+00:00
```

6）将系统时间与 RTC 时间同步

```
root@myd-jx8mp:~# hwclock -s
```

```
root@myd-jx8mp:~# date
Wed Nov 18 09:15:26 UTC 2020
```

重新开机之后查看的 RTC 时间和系统时间比之前设置的时候增加了大约 5 分钟，说明 RTC 工作正常。如果需要详细测试 RTC 的精度，可以将断电时间延长如 24 小时，测试 RTC 时间与标准时间的差异。

2.8. Watchdog

MYD-JX8MPQ 芯片内部有 3 个看门狗，精度 0.5S,最长时间 128S。有一个看门狗供用户使用。本节演示 watchdog 的使用，测试看门狗的开启关闭和设置看门狗超时时间。

1） 关闭看门狗

向看门狗节点写入字符“V”即可关闭看门狗：

```
root@myd-jx8mp:~#  echo V > /dev/watchdog
```

2） 看门狗应用程序测试

```
root@myd-jx8mp:~# /unit_tests/Watchdog/wdt_driver_test.out

---- Running < /unit_tests/Watchdog/wdt_driver_test.out > test ----

Usage: wdt_driver_test <timeout> <sleep> <test>
    timeout: value in seconds to cause wdt timeout/reset
    sleep: value in seconds to service the wdt
    test: 0 - Service wdt with ioctl(), 1 - with write()
```

运行看门狗应用,超时时间为 4s，每间隔 1s 喂一次狗：

```
root@myd-jx8mp:~#  /unit_tests/Watchdog/wdt_driver_test.out  4 1 0

---- Running < /unit_tests/Watchdog/wdt_driver_test.out > test ----

Starting wdt_driver (timeout: 4, sleep: 1, test: ioctl)
Trying to set timeout value=4 seconds
The actual timeout was set to 4 seconds
Now reading back -- The timeout is 4 seconds
```

如果将上面的 1s 改的大约 4s，即超过了要求的 4s 喂狗时间，开发板会重启。

2.9. Power Manager

本章节演示 Linux 电源管理的 Suspend 功能，让开发板睡眠 ，通过外部事件唤醒。Linux 内核一般提供了三种 Suspend: Freeze、Standby 和 STR(Suspend to RAM)，在用户空间向”/sys/power/state”文件分别写入”freeze”、”standby”和”mem”，即可触发它们。MYD-JX8MPQ 支持 freeze 和 mem 2 种方式。

1） 查看当前开发板支持的模式

```
root@myd-jx8mp:~# cat /sys/power/state
freeze mem
```

2） 在用户空间写入的方法

```
root@myd-jx8mp:~# echo "freeze" > /sys/power/state
root@myd-jx8mp:~# echo "mem" > /sys/power/state
```

- **休眠到内存**

以 mem 为例测试，输入休眠命令后开发板休眠，此时调试串口无法再输入。

```
root@myd-jx8mp:~#  echo "mem" > /sys/power/state
[  149.205468] PM: suspend entry (deep)
[  149.212992] Filesystems sync: 0.003 seconds
[  149.220277] Freezing user space processes ... (elapsed 0.001 seconds) done.
[  149.228916] OOM killer disabled.
[  149.232163] Freezing remaining freezable tasks ... (elapsed 0.001 seconds)
done.
[  149.240660] printk: Suspending console(s) (use no_console_suspend to deb
ug)
```

- **通过按键（KEY USER）唤醒**

进入休眠之后，用户通过按“USER”按键即能正常唤醒：

```
[  149.865013] imx-dwmac 30bf0000.ethernet eth1: Link is Down
[  149.866670] imx-dwmac 30bf0000.ethernet eth1: Link is Up - 1Gbps/Full - f
low control off
[  149.866711] imx-dwmac 30bf0000.ethernet eth1: Link is Down
[  149.939555] PM: suspend devices took 0.692 seconds
```

```
[  149.941349] Disabling non-boot CPUs ...
[  149.941687] IRQ 14: no longer affine to CPU1
[  149.941796] CPU1: shutdown
[  149.942811] psci: CPU1 killed (polled 0 ms)
[  149.944215] CPU2: shutdown
[  149.944220] psci: CPU2 killed (polled 0 ms)
[  149.945570] CPU3: shutdown
[  149.945575] psci: CPU3 killed (polled 0 ms)
[  149.946609] Enabling non-boot CPUs ...
[  149.946986] Detected VIPT I-cache on CPU1
[  149.947007] GICv3: CPU1: found redistributor 1 region 0:0x00000000388a00
00
[  149.947040] CPU1: Booted secondary processor 0x0000000001 [0x410fd034]
[  149.947432] CPU1 is up
[  149.947763] Detected VIPT I-cache on CPU2
[  149.947774] GICv3: CPU2: found redistributor 2 region 0:0x00000000388c00
00
[  149.947789] CPU2: Booted secondary processor 0x0000000002 [0x410fd034]
[  149.948051] CPU2 is up
[  149.948372] Detected VIPT I-cache on CPU3
[  149.948383] GICv3: CPU3: found redistributor 3 region 0:0x00000000388e00
00
[  149.948399] CPU3: Booted secondary processor 0x0000000003 [0x410fd034]
[  149.948715] CPU3 is up
[  149.970446] imx-dwmac 30bf0000.ethernet eth1: configuring for phy/rgmii-i
d link mode
[  150.229163] imx-dwmac 30bf0000.ethernet eth1: No Safety Features suppor
t found
[  150.229579] usb usb1: root hub lost power or was reset
[  150.229583] usb usb3: root hub lost power or was reset
[  150.229664] usb usb2: root hub lost power or was reset
[  150.229668] usb usb4: root hub lost power or was reset
```

```
[  150.573789] usb 3-1: reset SuperSpeed Gen 1 USB device number 2 using xhci-hcd
[  150.721495] usb 1-1: reset high-speed USB device number 2 using xhci-hcd
[  151.229418] usb 1-1.5: reset high-speed USB device number 3 using xhci-hcd
[  151.330463] PM: resume devices took 1.380 seconds
[  151.507272] OOM killer enabled.
[  151.510413] Restarting tasks ... done.
[  151.516023] PM: suspend exit
```

此时调试串口可以重新输入。

3. 外围接口

3.1. GPIO

GPIO 的测试是通过文件系统 sysfs 接口来实现的，下面内容以 GPIO5_IO11 为例说明 GPIO 的使用过程。

MYD-JX8MPQ 的 GPIO 脚是以 GPIOX_Y 形式来定义的，由于 GPIOX_Y 转换成引脚编号公式为:（X-1）*32+Y，所以 GPIO5_IO11 编号为：（5-1）*32+11=139。

下面介绍 uboot 和 kernel 下操作 GPIO 方法。

1）Uboot 下操作 GPIO

- **进入 uboot 模式**

开机按任意键进入 uboot 模式：

```
Fastboot: Normal
Normal Boot
Hit any key to stop autoboot:  0
u-boot=>
u-boot=>
```

- **控制电压**

通过 uboot 设置 GPIO5_IO11 电压：

```
u-boot=> gpio set 139
gpio: pin 139 (gpio 139) value is 1
u-boot=> gpio clear 139
gpio: pin 139 (gpio 139) value is 0
```

使用万用表可以测量当 set 时，电压为 3.3V， clear 时电压为 0V。

2）文件系统 gpio 操作

在文件系统操作 GPIO 时，必须先检查想要操作的 GPIO 是否被其他驱动占用，如果被占用，后续操作会报 busy 的警告。下面以操作拓展脚的一个接口为例，使用万用表测量电压。(ECSPI2_MOSI_3V3, GPIO5_IO11,139)

- **导出 GPIO**

```
root@myd-jx8mp:~# echo 139 > /sys/class/gpio/export
```

导出成功后会在/sys/class/gpio/目录下生成 gpio139 这个目录。

- **设置/查看 GPIO 方向**

设置输出，输入以下命令:

```
root@myd-jx8mp:~# echo out > /sys/class/gpio/gpio139/direction
```

或者设置输入：

```
root@myd-jx8mp:~# echo in > /sys/class/gpio/gpio139/direction
```

查看 gpio 方向：

```
root@myd-jx8mp:~# cat /sys/class/gpio/gpio139/direction
out
```

返回 in 表示输入，返回 out 表示输出。

- **设置/查看 GPIO 的值**

设置输出低：

```
root@myd-jx8mp:~# echo "0" > /sys/class/gpio/gpio139/value
```

或者设置输出高：

```
root@myd-jx8mp:~# echo "1" > /sys/class/gpio/gpio139/value
```

查看 gpio 的值：

```
root@myd-jx8mp:~# cat /sys/class/gpio/gpio139/value
1
```

可以看到 GPIO5_IO11 输出高电平，然后可以用万用表测量 J25 扩展 IO 的 ECSPI2_MOSI_3V3 引脚，可以看到电压为 3.3V 左右。

3.2. LED

Linux 系统提供了一个独立的子系统以方便从用户空间操作 LED 设备，该子系统以文件的形式为 LED 设备提供操作接口。这些接口位于/sys/class/leds 目录下。在硬件资源列表中，我们已经列出了设备上所有的 LED。下面通过命令读写 sysfs 的方式对 LED 进行测试。下述命令均为通用命令，也是操控 LED 的通用方法。

MYD-JX8MPQ 硬件设计上没有预留软件可控的 LED 灯，这里以 MYS-8MMX 为例讲解。

1) 查看 LED 操作目录

操作 LED 的目录为/sys/class/gpios,该目录内容为：

```
root@mys-8mmx:~# ls  /sys/class/leds/
cpu  mmc0::  mmc1::  mmc2::  user
```

2) 以 user 为例测试 LED

- **读取 LED 状态**

读取 userLED 状态，其中 0 表示 LED 关闭，1~255 表示 LED 开启：

```
root@mys-8mmx:~# cat  /sys/class/leds/user/brightness
255
```

可知当前红灯 LED 为开启状态。

- **熄灭 LED**

```
root@mys-8mmx:~# echo 0 > /sys/class/leds/user/brightness
```

- **点亮 LED**

```
root@mys-8mmx:~# echo 1 > /sys/class/leds/user/brightness
```

3.3. Key(按键)

linux 的/dev/input/eventxx 设备可以用来方便地调试 鼠标、键盘、触摸板等输入设备。本节主要是测试 key。通过 evtest 命令来查看按键是否有反应。MYD-JX8MPQ 有三个按键，S2 是 on/off 按键，S3 系统复位按键，S4 是用户按键。

1）设备树配置信息

打开配套的设备树文件 myd-jx8mp-base.dts ，可以看到对应的 gpio-keys 节点：

```
gpio-keys {
        compatible = "gpio-keys";
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_gpio_key>;

        user {
                label = "User";
                gpios = <&gpio3 20 GPIO_ACTIVE_LOW>;
                gpio-key,wakeup;
                linux,code = <KEY_1>;
        };

};
```

2）按键测试

- **查看对应的输入设备 event 信息**

```
root@myd-jx8mp:~# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:      30370000.snvs:snvs-powerkey
/dev/input/event1:      audio-hdmi HDMI Jack
/dev/input/event2:      gpio-keys
Select the device event number [0-2]:
```

由上可以知 gpio-keys 的对应设备事件为 event2。

- **evtest 测试按键**

执行下面命令，操作按键 S4，串口终端会打印出如下信息：

```
Select the device event number [0-2]: 2
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 2 (KEY_1)
Properties:
Testing ... (interrupt to exit)
Event: time 1605751345.1605751345, type 1 (EV_KEY), code 2 (KEY_1), value 1
Event: time 1605751345.1605751345, ---------- SYN_REPORT ------------
Event: time 1605751345.1605751345, type 1 (EV_KEY), code 2 (KEY_1), value 0
Event: time 1605751345.1605751345, ---------- SYN_REPORT ------------
```

每按一次 S4 当前终端会打印出当前事件码值，即按键正常。

3.4. USB

MYD-JX8MPQ 拥有 2 个 USB3.0 接口，一个为 Type_C 接口，另一个 USB3.0 口。

本节通过相关命令或热插拔去验证 USB Host 驱动的可行性，并实现读写 U 盘的功能、usb 枚举功能。

1）查看插入 usb 的打印信息

- **查看 USB 设备信息**

将 U 盘连接到开发板 USB Host 接口，内核提示信息如下：

```
root@myd-jx8mp:~# [60524.797576] usb 3-1.1: new SuperSpeed Gen 1 USB device number 3 using xhci-hcd
[60524.843205] usb-storage 3-1.1:1.0: USB Mass Storage device detected
[60524.850096] scsi host0: usb-storage 3-1.1:1.0
[60525.885850] scsi 0:0:0:0: Direct-Access     Kingston DataTraveler 3.0    PQ: 0 ANSI: 6
[60525.894896] sd 0:0:0:0: [sda] 60437492 512-byte logical blocks: (30.9 GB/28.8 GiB)
[60525.902721] sd 0:0:0:0: [sda] Write Protect is off
[60525.907901] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[60525.938595]  sda:
[60525.941974] sd 0:0:0:0: [sda] Attached SCSI removable disk
[60526.854112]  sda:
```

从上述信息可以得出需要挂载的设备为 sda。

2）U 盘挂载读写

U 盘挂载读写，由于文件系统已经加入了自动挂载功能，所以只要是 U 盘已经格式化 vfat，ext4，ntfs 格式（默认 kernel 不开启），就可以挂载。

- **查看挂载点**

```
root@myd-jx8mp:~# cat /proc/mounts | grep sda*
cgroup2 /sys/fs/cgroup/unified cgroup2 rw,nosuid,nodev,noexec,relatime,nsdelegate 0 0
```

```
/dev/sda /run/media/sda vfat rw,relatime,gid=6,fmask=0007,dmask=0007,allow_
utime=0020,codepage=437,iocharset=iso8859-1,shortname=mixed,errors=remou
nt-ro 0 0
```

- **写文件**

```
root@myd-jx8mp:~# echo "myir udisk test" > /run/media/sda/test_file.txt
```

- **读文件**

```
root@myd-jx8mp:~# cat /run/media/sda/test_file.txt
myir udisk test
```

写完文件后需要执行下 sync 命令，确保数据完全写入到 U 盘里面之后，才可以卸载设备。

3）关于断电文件保存情况

磁盘文件写入涉及缓存概念，分成全缓存，行缓存和无缓存，默认 io 里面是全缓存，只有当缓存数据满后才会写到磁盘文件。所以在写入文件中突然断电，会让文件写入内容丢失。但是当采用 reboot 时，系统会帮忙把缓冲区数据写入数据，命令行可用 sync 强行将数据写入磁盘。

3.5. Display

i.MX8M Plus 芯片拥有 1 路 HDMI，1 路 MIPI-DSI，1 路 LVDS 输出。MYD-JX8MPQ 将 LVDS 拆分成单路 LVDS J18，J19，以及双路 LVDS J20。

图像终端支持 wayland 和 xwayland（x11+wayland），不支持 FB 模式。

1）HDMI 显示

首先用 HDMI 高清线(或 HDMI 转 VGA 模块)连接 HDMI 显示器，然后使用高清线 HDMI 连接开发板和高清显示屏，开机默认是 HDMI 显示，观察 HDMI 能否正常输出，有无色差，抖动，偏移等异常现象。当正常显示后，快速插拔 HDMI 线缆多次，并检查每次是否都有正常输出。

接着输入以下命令可以查看当前 HDMI 显示的分辨率，分辨率采用最佳分辨率，最高支持 4K 分辨率：

```
root@myd-jx8mp:~# cat /sys/kernel/debug/dri/1/state
plane[31]: plane-0
        crtc=(null)
        fb=0
        crtc-pos=0x0+0+0
        src-pos=0.000000x0.000000+0.000000+0.000000
        rotation=1
        normalized-zpos=0
        color-encoding=ITU-R BT.601 YCbCr
        color-range=YCbCr limited range
plane[34]: plane-1
        crtc=(null)
        fb=0
        crtc-pos=0x0+0+0
        src-pos=0.000000x0.000000+0.000000+0.000000
        rotation=1
        normalized-zpos=0
        color-encoding=ITU-R BT.601 YCbCr
        color-range=YCbCr limited range
plane[37]: plane-2
```

```
        crtc=crtc-2
        fb=45
                allocated by = weston
                refcount=2
                format=XR24 little-endian (0x34325258)
                modifier=0x0
                size=1920x1080
                layers:
                        size[0]=1920x1080
                        pitch[0]=7680
                        offset[0]=0
                        obj[0]:
                                name=0
                                refcount=4
                                start=001007e9
                                size=8294400
                                imported=no
                                paddr=0x00000000d4b00000
                                vaddr=0000000075785086
        crtc-pos=1920x1080+0+0
        src-pos=1920.000000x1080.000000+0.000000+0.000000
        rotation=1
        normalized-zpos=0
        color-encoding=ITU-R BT.601 YCbCr
        color-range=YCbCr limited range
crtc[33]: crtc-0
        enable=0
        active=0
        self_refresh_active=0
        planes_changed=0
        mode_changed=0
        active_changed=0
        connectors_changed=0
```

```
        color_mgmt_changed=0
        plane_mask=0
        connector_mask=0
        encoder_mask=0
        mode: "": 0 0 0 0 0 0 0 0 0 0 0x0 0x0
crtc[36]: crtc-1
        enable=0
        active=0
        self_refresh_active=0
        planes_changed=0
        mode_changed=0
        active_changed=0
        connectors_changed=0
        color_mgmt_changed=0
        plane_mask=0
        connector_mask=0
        encoder_mask=0
        mode: "": 0 0 0 0 0 0 0 0 0 0 0x0 0x0
crtc[39]: crtc-2
        enable=1
        active=1
        self_refresh_active=0
        planes_changed=1
        mode_changed=0
        active_changed=0
        connectors_changed=0
        color_mgmt_changed=0
        plane_mask=4
        connector_mask=1
        encoder_mask=1
        mode: "1920x1080": 60 148500 1920 2008 2052 2200 1080 1084 1089
 1125 0x40 0x5
connector[41]: HDMI-A-1
```

```
        crtc=crtc-2
        self_refresh_aware=0
```

2）LVDS 显示

MYD-JX8MP 适配米尔的 LVDS 屏,型号为 MY-LVDS070C,屏幕先通过 40pin 柔性连接线接到底板的 J18 LVDS 接口接入 myir 1024x600 7 寸屏，然后再 uboot 下修改设备树名称，指定成 7 寸 LVDS 设备树。

```
u-boot=>
u-boot=> fatls mmc 2
 29209088    Image
     8208    imx8mp_m7_TCM_hello_world.bin
    19040    imx8mp_m7_TCM_rpmsg_lite_pingpong_rtos_linux_remote.bin
    18528    imx8mp_m7_TCM_rpmsg_lite_str_echo_rtos.bin
    40948    imx8mp_m7_TCM_sai_low_power_audio.bin
    62728    myd-jx8mp-atk-10.dtb
    61615    myd-jx8mp-base.dtb
    62728    myd-jx8mp-hontron-7.dtb
    62759    myd-jx8mp-lt8912.dtb
    62468    myd-jx8mp-m190etn01-19.dtb
  2113024    tee.bin
    12784    hello_world.bin
    62532    myd-jx8mp-rpmsg.dtb

13 file(s), 0 dir(s)

u-boot=> setenv fdt_file myd-jx8mp-hontron-7.dtb
u-boot=> save
Saving Environment to MMC... Writing to MMC(2)... OK
u-boot=>
```

开机后可以看到 LVDS 屏正常显示。

3）MIPI-DSI 显示

MIPI-DSI 测试，MYIR 制作了一款转接板（未量产，内部测试用），使用 LT8912 转接芯片，可以将 MIPI-DSI 转成 HDMI 输出。将转接板接入（J17）MIPI-DSI 接口，uboot 中设置对应得设备树，HDMI 接显示屏即可显示。

```
u-boot=>
u-boot=> fatls mmc 2
 29209088    Image
     8208    imx8mp_m7_TCM_hello_world.bin
    19040    imx8mp_m7_TCM_rpmsg_lite_pingpong_rtos_linux_remote.bin
    18528    imx8mp_m7_TCM_rpmsg_lite_str_echo_rtos.bin
    40948    imx8mp_m7_TCM_sai_low_power_audio.bin
    62728    myd-jx8mp-atk-10.dtb
    61615    myd-jx8mp-base.dtb
    62728    myd-jx8mp-hontron-7.dtb
    62759    myd-jx8mp-lt8912.dtb
    62468    myd-jx8mp-m190etn01-19.dtb
  2113024    tee.bin
    12784    hello_world.bin
    62532    myd-jx8mp-rpmsg.dtb

13 file(s), 0 dir(s)

u-boot=> setenv fdt_file myd-jx8mp-lt8912.dtb
u-boot=> save
Saving Environment to MMC... Writing to MMC(2)... OK
u-boot=>
```

3.6. Backlight

本例程主要是测试背光的亮度，通过操作 LVDS 在/sys 目录下的对应文件，以实现查询、调节背光亮度。

1）查看当前背光级别

- **进入背光参数调节目录**

```
root@myd-jx8mp:~# cd /sys/class/backlight/lvds_backlight/
root@myd-jx8mp:/sys/class/backlight/lvds_backlight# ls
actual_brightness bl_power brightness consumers device max_brightness
power scale subsystem suppliers type uevent
```

- **查看当前背光级别**

```
root@myd-jx8mp:/sys/class/backlight/lvds_backlight# cat brightness
80
```

- **查看背光最大级别**

```
root@myd-jx8mp:/sys/class/backlight/lvds_backlight# cat max_brightness
100
```

可知，背光最大级别是 100，范围为 0-100。

2）调节背光级别

如果需要调节背光级别，用 echo 向 brightness 写入对应亮度级别，但不能超过最大级别

- **关闭背光**

向 brightness 参数写入 ‘0’ 即会关闭背光：

```
root@myd-jx8mp:/sys/class/backlight/lvds_backlight# echo 0 > brightness
```

- **调整背光**

向 brightness 参数写入 0-100 之间的正整数：

```
root@myd-jx8mp:/sys/class/backlight/lvds_backlight# echo 60 > brightness
```

3.7. Touch Panel

触摸有电容触摸和电阻触摸，MYD-JX8MPQ 开发板硬件目前不支持电阻触摸，但是支持电容触摸，米尔科技提供 7 寸触摸的液晶屏配件。可根据实际需求自行购买配件。电容屏在使用中较为灵敏，很少出现问题。另外，电容屏不需要较准。因为根据电容屏的原理，电容屏在使用中是可以准确的识别出手指与屏幕接触的位置，具有很高的灵敏性。我们在使用中如果出现点击软件选不中的现象，一般只有一种情况：屏幕出现了问题。
MYD-JX8MPQ 电容触摸 IC 为 FT5x16。开机后可以用 evtest 命令测试触摸功能。

1）触摸屏连接

按照 3.5 节将 MY-LVDS070CV11 LVDS 屏连接到开发板。

2）测试过程

终端执行“evtest”进入测试界面。选择测试外设为触摸屏，这里默认电容触摸中断为输入中断 1，测试界面选择 ‘1’ 按下回车即可开始测试：

```
root@myd-jx8mp:~# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:      30370000.snvs:snvs-powerkey
/dev/input/event1:      generic ft5x06 (79)
/dev/input/event2:      gpio-keys
Select the device event number [0-2]: 1
Input driver version is 1.0.1
Input device ID: bus 0x18 vendor 0x0 product 0x0 version 0x0
Input device name: "generic ft5x06 (79)"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 330 (BTN_TOUCH)
  Event type 3 (EV_ABS)
    Event code 0 (ABS_X)
      Value      0
      Min        0
```

```
      Max       1023
    Event code 1 (ABS_Y)
      Value        0
      Min          0
      Max        599
    Event code 47 (ABS_MT_SLOT)
      Value        0
      Min          0
      Max          4
    Event code 53 (ABS_MT_POSITION_X)
      Value        0
      Min          0
      Max       1023
    Event code 54 (ABS_MT_POSITION_Y)
      Value        0
      Min          0
      Max        599
    Event code 57 (ABS_MT_TRACKING_ID)
      Value        0
      Min          0
      Max      65535
Properties:
  Property type 1 (INPUT_PROP_DIRECT)
Testing ... (interrupt to exit)
Event: time 1600596983.1600596983, type 3 (EV_ABS), code 57 (ABS_MT_TRAC
KING_ID), value 0
Event: time 1600596983.1600596983, type 3 (EV_ABS), code 53 (ABS_MT_POSI
TION_X), value 344
Event: time 1600596983.1600596983, type 3 (EV_ABS), code 54 (ABS_MT_POSI
TION_Y), value 395
Event: time 1600596983.1600596983, type 1 (EV_KEY), code 330 (BTN_TOUCH),
 value 1
```

```
Event: time 1600596983.1600596983, type 3 (EV_ABS), code 0 (ABS_X), value 344
Event: time 1600596983.1600596983, type 3 (EV_ABS), code 1 (ABS_Y), value 395
Event: time 1600596983.1600596983, ---------- SYN_REPORT ----------
Event: time 1600596983.1600596983, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 342
Event: time 1600596983.1600596983, type 3 (EV_ABS), code 0 (ABS_X), value 342
Event: time 1600596983.1600596983, ---------- SYN_REPORT ----------
Event: time 1600596983.1600596983, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 394
Event: time 1600596983.1600596983, type 3 (EV_ABS), code 1 (ABS_Y), value 394
Event: time 1600596983.1600596983, ---------- SYN_REPORT ----------
Event: time 1600596983.1600596983, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 395
Event: time 1600596983.1600596983, type 3 (EV_ABS), code 1 (ABS_Y), value 395
Event: time 1600596983.1600596983, ---------- SYN_REPORT ----------
Event: time 1600596983.1600596983, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID), value -1
Event: time 1600596983.1600596983, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 0
Event: time 1600596983.1600596983, ---------- SYN_REPORT ----------.....
```

可知，当每次点击 LVDS 屏时，屏幕会打印出键值坐标等，即触摸正常。

3.8. MIPI-CSI

MYD-JX8MPQ 有两路独立 MIPI-CSI 接口，适配 MY-CAM003M 摄像头，采用 OV5640 镜头，最高支持 500W 像素，这里采用此摄像头进行说明。

接开发板时请注意线序，需要用异面排线。

1）通过 i2cdetect 查看驱动是否加载

原理图上可以找到设备挂载在哪个 I2C 总线上，通过对应 I2C 原理图或者设备树文件可以找到对应的地址。OV5640 对应的地址是 0x3C。

```
root@myd-jx8mp:~# i2cdetect -y 2
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- UU -- -- -- -- --
20: -- -- -- -- -- -- -- -- UU -- -- -- -- -- -- --
30: -- -- UU -- -- -- -- -- -- -- -- -- UU -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- 6a -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

3C 里面 UU 代表驱动正常加载。

2）查看设备节点

```
root@myd-jx8mp:~# ls -l /dev/video*
crw-rw---- 1 root video 81, 0 Sep 20 10:16 /dev/video0
crw-rw---- 1 root video 81, 1 Sep 20 10:16 /dev/video1
```

3）查看设备支持的格式

```
root@myd-jx8mp:~# v4l2-ctl -D -d /dev/video0 --list-formats-ext
```

```
[  635.371814] ov5640 1-003c: ov5640_set_virtual_channel:481f 3c
[  635.378901] ov5640 1-003c: ov5640_set_virtual_channel:481f 1a
[  635.385684] ov5640 1-003c: ov5640_set_virtual_channel:0x4823 3c
[  635.392938] ov5640 1-003c: ov5640_set_virtual_channel:0x4823 10
[  635.399587] ov5640 1-003c: ov5640_set_virtual_channel:0x4827 32
[  635.406823] ov5640 1-003c: ov5640_set_virtual_channel:0x4827 16
[  635.413615] ov5640 1-003c: ov5640_set_virtual_channel:0x481b 3c
[  635.420871] ov5640 1-003c: ov5640_set_virtual_channel:0x481b c
Driver Info:
        Driver name      : mxc-isi-cap
        Card type        : mxc-isi-cap
        Bus info         : platform:32e00000.isi:cap_devic
        Driver version   : 5.10.9
        Capabilities     : 0x84201000
                Video Capture Multiplanar
                Streaming
                Extended Pix Format
                Device Capabilities
        Device Caps      : 0x04201000
                Video Capture Multiplanar
                Streaming
                Extended Pix Format
Media Driver Info:
        Driver name      : mxc-md
        Model            : FSL Capture Media Device
        Serial           :
```

```
        Bus info          :
        Media version     : 5.10.9
        Hardware revision: 0x00000000 (0)
        Driver version    : 5.10.9
Interface Info:
        ID                : 0x03000014
        Type              : V4L Video
Entity Info:
        ID                : 0x00000012 (18)
        Name              : mxc_isi.0.capture
        Function          : V4L2 I/O
        Pad 0x01000013    : 0: Sink
          Link 0x02000041: from remote pad 0x100000e of entity 'mxc_isi.0':
Data, Enabled
ioctl: VIDIOC_ENUM_FMT
        Type: Video Capture Multiplanar

        [0]: 'RGBP' (16-bit RGB 5-6-5)
                Size: Discrete 176x144
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 640x480
                        Interval: Discrete 0.067s (15.000 fps)
```

```
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 720x480
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 720x576
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1024x768
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1280x720
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1920x1080
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 2592x1944
                Interval: Discrete 0.125s (8.000 fps)
[1]: 'RGB3' (24-bit RGB 8-8-8)
        Size: Discrete 176x144
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 320x240
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 640x480
```

```
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 720x480
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 720x576
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1024x768
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1280x720
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1920x1080
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 2592x1944
                Interval: Discrete 0.125s (8.000 fps)
[2]: 'BGR3' (24-bit BGR 8-8-8)
        Size: Discrete 176x144
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 320x240
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
```

```
                Size: Discrete 640x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x576
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1024x768
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1920x1080
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 2592x1944
                        Interval: Discrete 0.125s (8.000 fps)
        [3]: 'YUYV' (YUYV 4:2:2)
                Size: Discrete 176x144
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.067s (15.000 fps)
```

```
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 640x480
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 720x480
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 720x576
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1024x768
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1280x720
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1920x1080
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 2592x1944
                Interval: Discrete 0.125s (8.000 fps)
[4]: 'YUV4' (32-bit A/XYUV 8-8-8-8)
        Size: Discrete 176x144
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 320x240
```

```
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 640x480
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 720x480
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 720x576
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1024x768
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1280x720
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1920x1080
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 2592x1944
                Interval: Discrete 0.125s (8.000 fps)
[5]: 'NV12' (Y/CbCr 4:2:0)
        Size: Discrete 176x144
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
```

```
                Size: Discrete 320x240
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 640x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x576
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1024x768
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1920x1080
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 2592x1944
                        Interval: Discrete 0.125s (8.000 fps)
        [6]: 'YM24' (Planar YUV 4:4:4 (N-C))
                Size: Discrete 176x144
                        Interval: Discrete 0.067s (15.000 fps)
```

```
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 320x240
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 640x480
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 720x480
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 720x576
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1024x768
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1280x720
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1920x1080
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 2592x1944
                Interval: Discrete 0.125s (8.000 fps)
[7]: 'XR24' (32-bit BGRX 8-8-8-8)
        Size: Discrete 176x144
```

```
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 320x240
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 640x480
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 720x480
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 720x576
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1024x768
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1280x720
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1920x1080
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 2592x1944
                Interval: Discrete 0.125s (8.000 fps)
[8]: 'AR24' (32-bit BGRA 8-8-8-8)
```

```
Size: Discrete 176x144
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
Size: Discrete 320x240
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
Size: Discrete 640x480
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
Size: Discrete 720x480
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
Size: Discrete 720x576
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
Size: Discrete 1024x768
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
Size: Discrete 1280x720
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
Size: Discrete 1920x1080
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
Size: Discrete 2592x1944
        Interval: Discrete 0.125s (8.000 fps)
```

- Driver name： 驱动名称
- Ioctl：VIDIOC_ENUM_FMT： 支持格式

4）进行预览

输入以下命令，进行窗口预览：

```
root@myd-jx8mp:~# gst-launch-1.0 v4l2src device=/dev/video0 ! 'video/x-raw, width=720,height=480,framerate=30/1' ! glimagesink
[  714.296901] ov5640 1-003c: ov5640_set_virtual_channel:481f 3c
[  714.306046] ov5640 1-003c: ov5640_set_virtual_channel:481f 1a
[  714.312538] ov5640 1-003c: ov5640_set_virtual_channel:0x4823 3c
[  714.319800] ov5640 1-003c: ov5640_set_virtual_channel:0x4823 10
[  714.326439] ov5640 1-003c: ov5640_set_virtual_channel:0x4827 32
[  714.333657] ov5640 1-003c: ov5640_set_virtual_channel:0x4827 16
[  714.340301] ov5640 1-003c: ov5640_set_virtual_channel:0x481b 3c
[  714.347534] ov5640 1-003c: ov5640_set_virtual_channel:0x481b c
[  714.434243] ov5640 2-003c: ov5640_write_reg: error: reg=3103, val=11
[  714.441719] ov5640 2-003c: ov5640_write_reg: error: reg=3808, val=2
[  714.448013] isi-capture 32e02000.isi:cap_device: Call subdev s_power fail!
[  715.102853] ov5640 1-003c: ov5640_set_virtual_channel:481f 3c
[  715.109953] ov5640 1-003c: ov5640_set_virtual_channel:481f 1a
[  715.116453] ov5640 1-003c: ov5640_set_virtual_channel:0x4823 3c
[  715.123798] ov5640 1-003c: ov5640_set_virtual_channel:0x4823 10
[  715.130451] ov5640 1-003c: ov5640_set_virtual_channel:0x4827 32
[  715.137657] ov5640 1-003c: ov5640_set_virtual_channel:0x4827 16
[  715.144313] ov5640 1-003c: ov5640_set_virtual_channel:0x481b 3c
[  715.151537] ov5640 1-003c: ov5640_set_virtual_channel:0x481b c
[  715.236668] ov5640 2-003c: ov5640_write_reg: error: reg=3103, val=11
```

```
[  715.243241] ov5640 2-003c: ov5640_write_reg: error: reg=3808, val=2
[  715.249579] isi-capture 32e02000.isi:cap_device: Call subdev s_power fail!
Setting pipeline to PAUSED ...
[  715.935507] ov5640 1-003c: ov5640_set_virtual_channel:481f 3c
[  715.942575] ov5640 1-003c: ov5640_set_virtual_channel:481f 1a
[  715.949481] ov5640 1-003c: ov5640_set_virtual_channel:0x4823 3c
[  715.956719] ov5640 1-003c: ov5640_set_virtual_channel:0x4823 10
[  715.963396] ov5640 1-003c: ov5640_set_virtual_channel:0x4827 32
[  715.970638] ov5640 1-003c: ov5640_set_virtual_channel:0x4827 16
[  715.977585] ov5640 1-003c: ov5640_set_virtual_channel:0x481b 3c
[  715.984831] ov5640 1-003c: ov5640_set_virtual_channel:0x481b c
Pipeline is live and does not need PREROLL ...
Got context from element 'sink': gst.gl.GLDisplay=context, gst.gl.GLDisplay=(GstGLDisplay)"\(GstGLDisplayWayland\)\ gldisplaywayland0";
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
[  716.040581] bypass csc
[  716.042992] input fmt YUV4
[  716.045721] output fmt YUYV
[  716.512034] ov5640 1-003c: ov5640_set_virtual_channel:481f 1a
[  716.519481] ov5640 1-003c: ov5640_set_virtual_channel:481f 1a
[  716.525989] ov5640 1-003c: ov5640_set_virtual_channel:0x4823 10
[  716.533290] ov5640 1-003c: ov5640_set_virtual_channel:0x4823 10
[  716.539948] ov5640 1-003c: ov5640_set_virtual_channel:0x4827 16
[  716.547221] ov5640 1-003c: ov5640_set_virtual_channel:0x4827 16
```

```
[   716.553879] ov5640 1-003c: ov5640_set_virtual_channel:0x481b c
[   716.561040] ov5640 1-003c: ov5640_set_virtual_channel:0x481b c
```

- Device：指定设备节点
- Video/x-raw：显示视频格式
- Width：显示宽　需要和支持宽匹配
- Height：显示高　需要和支持高匹配
- Framerate：　帧率 需要和支持匹配
- Glimagesink：　显示终端

5）保存摄像头数据到本地

- **gst-launch 命令语法**

这里主要使用 gst-launch-1.0 工具就行预览与编码保存，在界面输入以下命令可以查看命令语法：

```
root@myd-jx8mp:~#  gst-launch-1.0 -help
Usage:
  gst-launch-1.0 [OPTION?] PIPELINE-DESCRIPTION
Help Options:
  -h, --help                    Show help options
  --help-all                    Show all help options
  --help-gst                    Show GStreamer Options
Application Options:
  -t, --tags                    Output tags (also known as metadata)
  -c, --toc                     Output TOC (chapters and editions)
  -v, --verbose                 Output status information and property no
tifications
  -q, --quiet                   Do not print any progress information
  -m, --messages                Output messages
```

```
  -X, --exclude=PROPERTY-NAME          Do not output status information for
the specified property if verbose output is enabled (can be used multiple tim
es)
  -f, --no-fault                       Do not install a fault handler
  -e, --eos-on-shutdown                Force EOS on sources before shutting th
e pipeline down
  --version                            Print version information and exit
```

- **H.264 格式进行编码保存**

输入以下命令进行 H.264 格式进行编码保存：

```
gst-launch-1.0  -e  v4l2src device=/dev/video0 io-mode=2 ! video/x-raw,forma
t=YUY2,width=1280,height=720,frmaerate=30/1 ! vpuenc_h264 ! queue ! h264
parse ! qtmux ! filesink location=uvc_h264.mp4
```

部分参数说明：

- v4l2src：采集视频的插件
- device：配置采集的设备节点，可读可写标志
- video/x-raw,format:配置采集的视频格式、长宽和帧率

- **HEVC 进行编码保存**

输入以下命令进行 HEVC 格式进行编码保存：

```
gst-launch-1.0  -e  v4l2src device=/dev/video0 io-mode=2 ! video/x-raw,forma
t=YUY2,width=1280,height=720,frmaerate=30/1 ! vpuenc_hevc ! queue ! h265p
arse ! filesink location=uvc.h265
```

- **预览+保存**

H.264 进行编码保存:

```
gst-launch-1.0  -e v4l2src device=/dev/video0 io-mode=2 ! video/x-raw,format
=YUY2,width=640,height=480,frmaerate=30/1 ! tee name=t !  queue ! waylan
dsink t. ! queue ! vpuenc_h264 ! queue ! h264parse ! qtmux ! filesink locatio
n=uvc_h264.mp4
```

HEVC 进行编码保存

```
gst-launch-1.0 -e v4l2src device=/dev/video0 io-mode=2 ! video/x-raw,format=YUY2,width=1280,height=720,frmaerate=30/1 ! tee name=t ! queue ! waylandsink t. ! queue ! vpuenc_hevc ! queue ! h265parse ! filesink location=uvc.h265
```

- **预览+H264 编码 UDP 传输**

```
gst-launch-1.0 v4l2src device=/dev/video0 io-mode=2 ! video/x-raw,format=YUY2,width=640,height=480,framerate=30/1 ! tee name=t ! queue ! waylandsink t. ! queue ! vpuenc_h264 ! rtph264pay ! udpsink host=192.168.30.178 port=1234
```

3.9. M.2

MYS-JX8MPQ 有一路 M.2 M-key 接口，现在引出来给 M.2 接口的 SSD 硬盘使用，接入 SSD 硬盘后可以读写 SSD 设备，与 U 盘接入 USB 口后测试方法一致。

这里要注意，新买的 SSD 卡没有对磁盘进行格式化，所以不会自动挂载，需要先用 fsck.fat/fsck.ext2/fsck.ext3,fsck.ext4 等命令进行格式化。

1）介绍如何查看 SSD

- **使用 df 命令查看磁盘**

```
root@myd-jx8mp:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       6.7G  2.9G  3.5G  46% /
devtmpfs        980M  4.0K  980M   1% /dev
tmpfs           1.5G     0  1.5G   0% /dev/shm
tmpfs           585M  9.3M  576M   2% /run
tmpfs           4.0M     0  4.0M   0% /sys/fs/cgroup
tmpfs           1.5G  4.0K  1.5G   1% /tmp
tmpfs           1.5G  236K  1.5G   1% /var/volatile
/dev/mmcblk2p1  100M   31M   70M  31% /run/media/mmcblk2p1
/dev/nvme0n1    110G  1.1G  103G   2% /run/media/nvme0n1
tmpfs           293M  4.0K  293M   1% /run/user/0
```

- /dev/nvme0n1　即使 SSD 硬盘，110G 大小

2）如何格式化 SSD 分区

这里采用 fsck.ext4 命令格式化成 ext4 格式。

```
root@myd-jx8mp:~#  mkfs.ext4 /dev/nvme0n1
mke2fs 1.45.3 (14-Jul-2019)
/dev/nvme0n1 contains a ext4 file system
```

```
        last mounted on /run/media/nvme0n1 on Fri Sep  4 23:35:35 2020
Proceed anyway? (y,N) y
Discarding device blocks: done
Creating filesystem with 29305206 4k blocks and 7331840 inodes
Filesystem UUID: 8886cb14-d475-4348-8218-4d5d640ac58e
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 265
4208,
        4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (131072 blocks): done
Writing superblocks and filesystem accounting information: done
root@myd-jx8mp:~#
```

3) 读写 SSD 方法

如果已经格式化成功，当系统上电时，会自动挂载此设备，这里演示查看，写文件，读文件。

- **查看挂载点**

```
root@myd-jx8mp:~#  cat /proc/mounts | grep nvme
/dev/nvme0n1 /run/media/nvme0n1 ext4 rw,relatime 0 0
```

- **写文件**

```
root@myd-jx8mp:~#  echo "myir nvme test" >> /run/media/nvme0n1/nvme_t
est.txt
```

- **读文件**

```
root@myd-jx8mp:~# cat /run/media/nvme0n1/nvme_test.txt
```

myir nvme test

4. 网络接口

MYD-JX8MPQ 开发板包含两个千兆以太网接口和 1 路扩展接口来接 MYIR 提供生产的 AP6212/BL6212 WIFI 蓝牙模块。下面分别对这两种网络设备的配置进行介绍。

4.1. Ethernet

Linux 下网络配置的工具很多，常见的有 net-tools, iproute2，systemd-networkd, network manager 以及 connman 等，这些都可以在系统定制的时候根据实际需要进行选择，这里介绍几种常用的以太网手动临时配置和自动永久配置方式。

1）手动临时配置以太网 IP 地址

- **使用 net-tools 工具包中的 ifconfig 对网络进行手动配置**

首先通过 ifconfig 命令查看网络设备信息如下：

```
root@myd-jx8mp:~#  ifconfig  eth0
eth0        Link  encap:Ethernet   HWaddr  96:b2:81:dd:88:58
            UP  BROADCAST  MULTICAST   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)
```

eth0 为实际的以太网设备，这里的 MAC 地址，代码中是按以下顺序读取，如果第一条没有设置就查找下面一条。

- uboot 传参 fec.macddr
- 设备树文件写入 MAC
- flash 或者 fuse 写入 mac
- uboot 在 FEC mac 寄存器写入 mac
- 随机 mac 地址

下面介绍给 eth0 手动配置 IP 地址 192.168.1.100 的方法，命令如下：

```
root@myd-jx8mp:~#    ifconfig  eth0  192.168.1.100  netmask  255.255.255.0  up
```

上面的命令手动配置 eth0 的 IP 地址为 192.168.1.100，子网掩码为 255.255.255.0，以及默认配置的广播地址 192.168.1.255，并通过 up 参数进行激活，如下所示：

```
root@myd-jx8mp:~# ifconfig  eth0
```

```
eth0      Link encap:Ethernet  HWaddr 96:b2:81:dd:88:58
          inet addr:192.168.1.100  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  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 iproute2 工具包中的 ip 命令对网络进行手动配置**

ifconfig 命令手动设置 IP 地址的方法也可以使用 ip addr 和 ip link 进行替代，更多的信息请查看 https://wiki.linuxfoundation.org/networking/iproute2 中的说明。

```
root@myd-jx8mp:~#  ip addr flush dev eth0
root@myd-jx8mp:~#  ip addr add 192.168.1.101/24 brd + dev eth0
root@myd-jx8mp:~#  ip link set eth0 up
```

如果之前已经配置过 IP 地址，再使用 ip addr add 配置的 IP 地址将会成为 Secondary 地址，所以这里先使用 ip addr flush 清除之前的地址之后再进行配置然后激活。完成配置之后，通过 ip addr show 命令查看 eth0 信息如下：

```
root@myd-jx8mp:~# ip addr show eth0
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state
DOWN group default qlen 1000
    link/ether 96:b2:81:dd:88:58 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.101/24 brd 192.168.1.255 scope global eth0
       valid_lft forever preferred_lft forever
```

2）自动永久配置以太网 IP 地址

通过 ifconfig 命令和 ip 命令配置的 IP 地址断电之后就会丢失，如果需要使 IP 地址永久生效，就需要修改网络管理工具相应的配置文件。

MYD-JX8MPQ 采用 systemd-networkd 管理网络，以太网开机默认采用自动获取 ip 方式，这里介绍如何配置静态 ip 地址。

- **查看/lib/systemd/network**

```
root@myd-jx8mp:/lib/systemd/network# ls -l
total 40
-rw-r--r-- 1 root root  75 Sep 20 10:17 10-eth0.network
```

```
-rw-r--r-- 1 root root 645 Mar  9  2018 80-container-host0.network
-rw-r--r-- 1 root root 718 Mar  9  2018 80-container-ve.network
-rw-r--r-- 1 root root 704 Mar  9  2018 80-container-vz.network
-rw-r--r-- 1 root root 672 Mar  9  2018 80-vm-vt.network
-rw-r--r-- 1 root root  78 Mar  9  2018 80-wifi-adhoc.network
-rw-r--r-- 1 root root 101 Mar  9  2018 80-wifi-ap.network.example
-rw-r--r-- 1 root root  64 Mar  9  2018 80-wifi-station.network.example
-rw-r--r-- 1 root root 124 Mar  9  2018 80-wired.network
-rw-r--r-- 1 root root 491 Mar  9  2018 99-default.link
```

- **查看 80-wired.network**

```
root@myd-jx8mp:/lib/systemd/network# vi 80-wired.network
[Match]
Name=en* eth*
KernelCommandLine=!nfsroot

[Network]
DHCP=yes

[DHCP]
UseMTU=yes
RouteMetric=10
ClientIdentifier=mac
```

这里可以看到以 eth 开头得网卡设备会以 DHCP 方式自动获取 IP。以上配置文件将会对 en*和 eth*匹配的网卡进行配置，如果内核启动参数中不包含 nfsroot 参数时，则通过 DHCP 自动为匹配到的网卡配置 IP 地址，网关，DNS 等信息。详细的配置参数参见以下链接：
https://www.freedesktop.org/software/systemd/man/systemd.network.html。

其中网卡文件格式是数字+名称.network，下面演示如何设置静态 ip。

- **创建/lib/systemd/network/10-eth0.network 文件**

```
[Match]
Name=eth1
[Network]
```

```
Address=192.168.40.100/24
Gateway=192.168.40.1
```

4.2. Wi-Fi

本节主要介绍 Linux 下 Wi-Fi 的配置和使用，通常 Wi-Fi 模块可以支持两种工作模式，分别是 STA 模式和 AP 模式，有些设备还支持 STA 和 AP 模式同时工作。STA 模式允许设备连接外部 Wi-Fi 热点，AP 模式将设备变成 Wi-Fi 热点，供其它设备连接。

MYD-JX8MPQ J21 可以接 MYIR 的 AP6212 Wi-Fi 和 Bluetooth 二合一模块，当前不支持 STA 和 AP 同时工作，AP6212/BL6212 Wi-Fi 模块对应的驱动为：

```
root@myd-jx8mp:~# lsmod
Module                  Size  Used by
brcmfmac              245760  0
brcmutil               20480  1 brcmfmac
```

驱动加载的过程中会将位于/lib/firmware/brcm 的 Wi-Fi 固件加载到模块内部。Wi-Fi 模块驱动加载成功之后生成 Wi-Fi 设备的网络节点 wlan0，如下所示：

```
root@myd-jx8mp:~# ifconfig wlan0
wlan0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether c0:84:7d:72:61:2a  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

1) STA 模式脚本连接 WiFi 热点

下面尝试手动连接附近的 Wi-Fi 热点“360WiFi-4FF7C3"，这是一个采用 WPA2 加密方式的 Wi-Fi 热点，密码为 12345678。

MYIR 已提供 ifup_wifi_sta 脚本，输入 SSID 和 PASSWD 即可连接。

```
root@myd-jx8mp:~# ifup_wifi_sta -ssid 360WiFi-4FF7C3 -passwd 12345678
SSID:360WiFi-4FF7C3 PASSWD:12345678 DRIVER:nl80211
udhcpc: no process found
wpa_supplicant: no process found
hostapd: no process found
udhcpd: no process found
find phy0 enable it
```

```
[ 2136.784415] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes read
y
udhcpc: started, v1.32.0
udhcpc: sending discover
udhcpc: sending select for 192.168.0.8
udhcpc: lease of 192.168.0.8 obtained, lease time 172800
RTNETLINK answers: File exists
/etc/udhcpc.d/50default: Adding DNS 192.168.0.1
root@myd-jx8mp:~#
```

2）STA 模式开机自动连接 WiFi 热点

前面讲述了手动连接 wifi 热点，每次启动都要手动连接或者手动执行脚本非常不方便，这里将介绍如何开机自动连接 wifi 热点。

通过上述使用 ifup_wifi_sta 连接 wifi 后，会在/etc/wpa_supplicant/目录下生成一个新文件

```
root@myd-jx8mp:~# ls -l /etc/wpa_supplicant/
total 4
-rw-r--r-- 1 root root 201 Sep 20 10:51 wpa_supplicant-wlan0.conf
```

只要使能对应的服务即可开机连接 WIFI。

```
root@myd-jx8mp:~# systemctl enable wpa_supplicant@wlan0.service
Created symlink /etc/systemd/system/multi-user.target.wants/wpa_supplicant@
wlan0.service -> /lib/systemd/system/wpa_supplicant@.service.
```

重启后可以查看到 WIFI 已经连接上，但是没有 ip 地址。

```
root@myd-jx8mp:~# wpa_cli -iwlan0 -p/var/run/wpa_supplicant status
bssid=2c:61:04:4f:f7:c3
freq=2462
ssid=360WiFi-4FF7C3
id=0
mode=station
pairwise_cipher=CCMP
group_cipher=CCMP
```

```
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
p2p_device_address=c2:84:7d:72:61:2a
address=c0:84:7d:72:61:2a
uuid=0adec649-ea6c-5abe-8e0a-bf30dd8f4ac8
root@myd-jx8mp:~# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::c284:7dff:fe72:612a  prefixlen 64  scopeid 0x20<link>
        ether c0:84:7d:72:61:2a  txqueuelen 1000  (Ethernet)
        RX packets 868  bytes 45444 (44.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 24  bytes 3410 (3.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

根据以太网卡配置静态 IP 那里可知，只需要将 wlan0 设置成自动获取 ip，就能达到开机自动获取 ip 目的，参考/lib/systemd/network/80-wired.network 创建 11-wlan0.network。

```
[Match]
Name=wlan0

[Network]
DHCP=yes

[DHCP]
RouteMetric=20
```

再次重启开发板，可以看到 WIFI 已经获取到 IP。

```
root@myd-jx8mp:~# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.12  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::c284:7dff:fe72:612a  prefixlen 64  scopeid 0x20<link>
        ether c0:84:7d:72:61:2a  txqueuelen 1000  (Ethernet)
```

```
        RX packets 53   bytes 5684 (5.5 KiB)
        RX errors 0   dropped 0   overruns 0   frame 0
        TX packets 37   bytes 6435 (6.2 KiB)
        TX errors 0   dropped 0 overruns 0   carrier 0   collisions 0

root@myd-jx8mp:~# ip route[    21.759877] imx-dwmac 30bf0000.ethernet eth
1: Link is Up - 100Mbps/Full - flow control rx/tx
[    21.768364] IPv6: ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready

default via 192.168.40.1 dev eth1 proto static
default via 192.168.0.1 dev eth0 proto dhcp src 192.168.0.10 metric 10
default via 192.168.0.1 dev wlan0 proto dhcp src 192.168.0.12 metric 20
192.168.0.0/24 dev wlan0 proto kernel scope link src 192.168.0.12
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.10
192.168.0.1 dev eth0 proto dhcp scope link src 192.168.0.10 metric 10
192.168.0.1 dev wlan0 proto dhcp scope link src 192.168.0.12 metric 20
192.168.40.0/24 dev eth1 proto kernel scope link src 192.168.40.100
```

3）AP 模式手动配置热点

Hostapd 是一个带加密功能的无线接入点程序，是 Linux 操作系统上构建无线接入点比较方便的工具，支持 IEEE 802.11 协议和 IEEE 802.1X/WPA/WPA2/EAP/RADIUS 加密。板子作为 WiFi 热点，需要为每一个接入该热点的终端（例如手机）分配 IP，路由等网络参数。通过 udhcpd 开启 wlan 的动态地址分配，然后用 hostapd 来开启 wifi 的 AP 模式，在用 iptables 将 wlan0 的数据通过 eth 转发，让连接到 wlan0 的设备通过以太网上网。如创建 SSID 为 MYIR_TEST，PASSWD 为 myir2020 的无线 wifi 热点。步骤如下：

- **RFKILL 网络管理，先打开 wlan**

```
root@myd-jx8mp:~#    rfkill unblock wlan
```

- **设置 wlan ip 地址**

```
root@myd-jx8mp:~#   ifconfig wlan0 up 192.168.10.1
```

在前面我们有说明当前 Wi-Fi 模块不支持 STA 和 AP 模式同时工作，所以这里需要清除 STA 模式的配置，如下：

```
root@myd-jx8mp:~#  killall udhcpc
root@myd-jx8mp:~#  killall wpa_supplicant
```

- **使用 wlan0 设备运行 DHCP 服务**

wlan0 工作在 AP 模式，当其它外部设备连接这个 AP 热点的时候，它需要通过 wlan0 为其它外部设备动态分配 IP 地址，所以需要使用 wlan0 来运行 DHCP 服务程序 udhcpd。udhcpd 对应的配置文件为/etc/udhcpd.conf，内容如下：

```
# the start and end of the IP lease block
start              192.168.10.10
end                 192.168.10.254
# the interface that udhcpd will use
interface          wlan0

opt       dns       8.8.8.8
option    subnet    255.255.255.0
opt       router    192.168.10.1
option    domain    local
option    lease     864000
```

当 AP 模式配置完成，外部设备连接到这个热点时，就会通过 wlan0 获取到上面地址池中的 IP 地址，地址范围为 192.168.10.10~192.168.10.254，子网掩码为 255.255.255.0，默认网关 192.168.10.1，DNS 为 8.8.8.8，以及 DHCP 租约时间 864000 秒。

udhcpd 配置文件准备好之后，执行下面的命令启动 udhcpd 服务之后，命令如下：

```
root@myd-jx8mp:~#  udhcpd /etc/udhcpd.conf
```

- **使用 wlan0 运行 hostapd 服务**

配置 AP 模式最关键的一步当然是启动 hostapd 服务。启动服务之前需要通过 /etc/hostapd.conf 配置 AP 模式的 ssid, password，加密算法，驱动类型，工作模式等，完整的参数配置说明参见：http://w1.fi/cgit/hostap/plain/hostapd/hostapd.conf, 面是针对当前硬件的*/etc/hostapd.conf*配置，内容如下：

```
# File: /etc/hostapd.conf
interface=wlan0
```

```
driver=nl80211
# mode Wi-Fi (a = IEEE 802.11a, b = IEEE 802.11b, g = IEEE 802.11g)
hw_mode=g
ssid=MYIR_TEST
channel=7
wmm_enabled=0
macaddr_acl=0
# Wi-Fi closed, need an authentication
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=myir2020
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

配置文件准备好之后，执行下面的命令启动 hostapd 服务：

```
root@myd-jx8mp:~# hostapd -B /etc/hostapd.conf
Configuration file: /etc/hostapd.conf
[  103.932995] [dhd-wlan0] wl_cfg80211_del_station : Disconnect STA : 66:66:3a:66:66:3a scb_val.val 3
Using interface wlan0 with hwaddr b0:02:47:d1:61:5d and ssid "MYIR_TEST"
[  103.958085] [dhd-wlan0] wl_cfg80211_set_channel : netdev_ifidx(3), chan_type(1) target channel(7)
[  103.967710] [dhd] CFG80211-ERROR) wl_cfg80211_parse_ies : No WPSIE in beacon
[  103.974872] [dhd] CFG80211-ERROR) wl_cfg80211_parse_ies : No WPSIE in beacon
[  103.987625] [dhd] _dhd_wlfc_mac_entry_update():1866, entry(32)
[  103.994491] [dhd-wlan0] wl_cfg80211_bcn_bringup_ap : Creating AP with sec=wpa2psk/mfpn/0x44
[  104.071671] [dhd-wlan0] wl_iw_event : Link UP with b0:02:47:d1:61:5d
[  104.071960] [dhd-wlan0] wl_notify_connect_status_ap : AP/GO Link up
```

```
[  104.078067] [dhd-wlan0] wl_ext_iapsta_event : [A] Link up w/o creating? (etype=16)
[  104.092051] [dhd-wlan0] wl_iw_event : Link UP with b0:02:47:d1:61:5d
[  104.098448] [dhd-wlan0] wl_ext_iapsta_event : [A] Link up w/o creating? (etype=16)
[  104.112600] [dhd] CFG80211-ERROR) wl_set_wsec_info_algos : wsec_info error (-23)
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
```

- **连接 wifi**

使用手机连接“MYIR_TEST”wifi 热点后，通过手机端可以看到 192.168.10.10 的 IP 地址，开发板 ping 手机可以连通。但是手机无法连接外网。

```
root@myd-jx8mp:~# ping 192.168.10.10
PING 192.168.10.10 (192.168.10.10) 56(84) bytes of data.
64 bytes from 192.168.10.10: icmp_seq=1 ttl=64 time=289 ms
64 bytes from 192.168.10.10: icmp_seq=2 ttl=64 time=4.95 ms
64 bytes from 192.168.10.10: icmp_seq=3 ttl=64 time=486 ms
64 bytes from 192.168.10.10: icmp_seq=4 ttl=64 time=4.67 ms
```

- **设置网络转发**

当前只有 wifi 热点，还需 设置无线网卡共享 eth0 才能上网。用开发板连接路由器，并且能正常获取 ip 地址和 dns，再设置转发和伪装转发，此时可以看到手机可以上网。

```
root@myd-jx8mp:~# echo "1" > /proc/sys/net/ipv4/ip_forward
root@myd-jx8mp:~# iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -o eth0 -j MASQUERADE
```

4）AP 模式脚本配置热点

上述手动配置是为了讲解设置 WIFI AP 热点需要配置的文件和操作方法，MYIR 也提供了脚本 ifup_wifi_ap 来执行对应的命令。

```
root@myd-jx8mp:~# ifup_wifi_ap
```

4.3. Bluetooth

Linux 平台下通常采用 BlueZ（http://www.bluez.org/）进行蓝牙设备的配置和管理。Bluze 是一套比较完善的 Blutooth 配置和管理的工具集和协议栈，下面使用这些工具对 Bluetooth 进行配置和使用。

这里测试 AP6212 二合一模块中的蓝牙功能，它对应的驱动为 brcmfmac，brcmutil，如下所示：

```
root@myd-jx8mp:~# lsmod
Module                  Size  Used by
qmi_wwan               36864  0
qmi_wwan_q             40960  0
cdc_wdm                28672  2 qmi_wwan,qmi_wwan_q
usbnet                 45056  2 qmi_wwan,qmi_wwan_q
crct10dif_ce           20480  1
imx8_media_dev         20480  0
brcmfmac              245760  0
brcmutil               20480  1 brcmfmac
flexcan                32768  0
can_dev                36864  1 flexcan
```

驱动加载成功之后会生成 hci0 设备节点。下面具体介绍使用 BlueZ 工具集进行配置和连接附近蓝牙设备的过程：

1）绑定端口，此过程会自动打开 bluetoothd 服务

```
root@myd-jx8mp:~#  brcm_patchram_plus -d --enable_hci --no2bytes --tosleep 200000 --baudrate 3000000 --patchram /lib/firmware/bcmd/bcm43438a1.hcd /dev/ttymxc0 &
```

2）激活蓝牙，如果是 rfkill 管理，要先开启

```
root@myd-jx8mp:~# rfkill unblock bluetooth
root@myd-jx8mp:~# hciconfig hci0 up
```

3）扫描附件可获取的蓝牙设备

```
root@myd-jx8mp:~# hcitool scan
```

```
Scanning ...
        B0:FC:36:3B:CF:0E       MYIR-BT
        F8:A2:D6:26:F0:58       Z7VUR99EQQF62D2
```

4) 通过 bluetoothctl 命令来管理蓝牙的连接

bluetoothctl 是 BlueZ 提供的一套蓝牙管理工具，通过这个命令使能蓝牙控制器电源、代理、扫描、配对和连接。在使用 bluetoothctl 之前先确认系统的蓝牙服务已启动，或者确认/usr/libexec/bluetooth/bluetoothd 服务程序已启动。

```
root@myd-jx8mp:~# systemctl status bluetooth.service
* bluetooth.service - Bluetooth service
     Loaded: loaded (/lib/systemd/system/bluetooth.service; enabled; vendor p
reset: enabled)
     Active: active (running) since Sun 2020-09-20 11:40:22 UTC; 1min 39s ag
o
       Docs: man:bluetoothd(8)
   Main PID: 693 (bluetoothd)
     Status: "Running"
      Tasks: 1 (limit: 2350)
     Memory: 1.8M
     CGroup: /system.slice/bluetooth.service
             `-693 /usr/libexec/bluetooth/bluetoothd

Sep 20 11:40:22 myd-jx8mp systemd[1]: Starting Bluetooth service...
Sep 20 11:40:22 myd-jx8mp systemd[1]: Started Bluetooth service.
```

下面使用 bluetoothctl 进行具体测试蓝牙的扫描，配对和连接，步骤如下：

- **终端输入 bluetoothctl 进入蓝牙控制界面**

```
root@myd-jx8mp:~# bluetoothctl
Agent registered
[bluetooth]#
```

- **使能蓝牙控制电源**

```
[bluetooth]# power on
```

```
Changing power on succeeded
```

- **使能蓝牙代理**

使能蓝牙代理管理并查看代理管理是否成功，这个默认是使能的：

```
[bluetooth]# agent on
Agent is already registered
[bluetooth]# default-agent
Default agent request successfu
```

- **扫描附近可以连接的蓝牙设备**

```
[bluetooth]# scan on
Discovery started
[CHG] Controller B0:02:47:D1:61:5E Discovering: yes
[NEW] Device 4F:1B:E1:C3:C6:D0 4F-1B-E1-C3-C6-D0
[NEW] Device 40:01:47:9E:65:BC 40-01-47-9E-65-BC
[NEW] Device B4:0B:44:F3:49:5D B4-0B-44-F3-49-5D
[NEW] Device B0:FC:36:3B:CF:0E MYIR-BT
[NEW] Device 74:0A:E1:35:D0:FA Metro
[NEW] Device A4:4B:D5:71:6D:5F A4-4B-D5-71-6D-5F
[CHG] Device A4:4B:D5:71:6D:5F Name: A.Z 的 Redmi K30 5G
[CHG] Device A4:4B:D5:71:6D:5F Alias: A.Z 的 Redmi K30 5G
[CHG] Device B4:0B:44:F3:49:5D Name: 玩世不恭的弓长张
[CHG] Device B4:0B:44:F3:49:5D Alias: 玩世不恭的弓长张
```

由上面可知扫描到的当前设备为 74:0A:E1:35:D0:FA Metro。

- **进行设备配对**

```
[bluetooth]# pair 74:0A:E1:35:D0:FA
Attempting to pair with 74:0A:E1:35:D0:FA
[CHG] Device 74:0A:E1:35:D0:FA Connected: yes
Request confirmation
[agent] Confirm passkey 811621 (yes/no): yes
[CHG] Device 74:0A:E1:35:D0:FA Modalias: bluetooth:v010Fp107Ed1436
[CHG] Device 74:0A:E1:35:D0:FA UUIDs: 0000046a-0000-1000-8000-00805f9b34f
b
```

```
[CHG] Device 74:0A:E1:35:D0:FA UUIDs: 00001105-0000-1000-8000-00805f9b34f
b
[CHG] Device 74:0A:E1:35:D0:FA UUIDs: 0000110a-0000-1000-8000-00805f9b34f
b
[CHG] Device 74:0A:E1:35:D0:FA UUIDs: 0000110c-0000-1000-8000-00805f9b34f
b
[CHG] Device 74:0A:E1:35:D0:FA UUIDs: 00001112-0000-1000-8000-00805f9b34f
b
[CHG] Device 74:0A:E1:35:D0:FA UUIDs: 00001115-0000-1000-8000-00805f9b34f
b
[CHG] Device 74:0A:E1:35:D0:FA UUIDs: 00001116-0000-1000-8000-00805f9b34f
b
[CHG] Device 74:0A:E1:35:D0:FA UUIDs: 0000111f-0000-1000-8000-00805f9b34f
b
[CHG] Device 74:0A:E1:35:D0:FA UUIDs: 0000112f-0000-1000-8000-00805f9b34f
b
[CHG] Device 74:0A:E1:35:D0:FA UUIDs: 00001132-0000-1000-8000-00805f9b34f
b
[CHG] Device 74:0A:E1:35:D0:FA UUIDs: 00001200-0000-1000-8000-00805f9b34f
b
[CHG] Device 74:0A:E1:35:D0:FA UUIDs: 00001800-0000-1000-8000-00805f9b34f
b
[CHG] Device 74:0A:E1:35:D0:FA UUIDs: 00001801-0000-1000-8000-00805f9b34f
b
[CHG] Device 74:0A:E1:35:D0:FA UUIDs: 0000fdd1-0000-1000-8000-00805f9b34f
b
[CHG] Device 74:0A:E1:35:D0:FA UUIDs: 0000fe35-0000-1000-8000-00805f9b34f
b
[CHG] Device 74:0A:E1:35:D0:FA ServicesResolved: yes
[CHG] Device 74:0A:E1:35:D0:FA Paired: yes
Pairing successful
[CHG] Device 74:0A:E1:35:D0:FA ServicesResolved: no
[CHG] Device 74:0A:E1:35:D0:FA Connected: no
```

至此，表示已经与华为手机 Metro 蓝牙配对成功。

- **连接设备**

```
[bluetooth]# connect 74:23:44:CE:B6:7D
[bluetooth]# connect 74:0A:E1:35:D0:FA
Attempting to connect to 74:0A:E1:35:D0:FA
[CHG] Device 74:0A:E1:35:D0:FA Connected: yes
Connection successful
[CHG] Device 74:0A:E1:35:D0:FA ServicesResolved: yes
[Metro]#
```

至此，你将可以看到手机上面有显示蓝牙以及连接。

4.4. 4G/5G

LINUX 设备也可以外接 4G 或者 5G 模块来拨号上网，4G 模块使用较多的是移远的 EC20，5G 模块包括 rm500q 以及 RG801H。

拨号方式有 pppd，gobinet 以及 qmi_wwan 3 种方式，其中 pppd 比较通用，gobinet 没有用到，qmi_wwan 连接比较快，并且可以用来做 5G 模块连接。

这里就以 ppp 和 qmi_wwan2 种方式来进行拨号说明，使用模块为 EC20 CE FDKG，采用 USB 转接板。

1) 查看 VID 和 PID

将装好 EC20 模块的 USB 转接板接入开发板，使用 lsusb 查看 EC20 模块信息。

```
root@myd-jx8mp:~# lsusb
Bus 002 Device 003: ID 2c7c:0125 Quectel Wireless Solutions Co., Ltd. EC25 LTE modem
Bus 002 Device 002: ID 0424:2514 Standard Microsystems Corp. USB 2.0 Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
root@myd-jx8mp:~#
```

- 2c7c:0125 ： EC20 的 VID 和 PID 信息。

其中需要在${KERNEL_DIR}/drivers/usb/serial/option.c 中的数组 static const struct usb_device_id option_ids 中配置如下：

```
#define QUALCOMM_VENDOR_ID                  0x05C6
#define QUECTEL_PRODUCT_EC25                0x0125
static const struct usb_device_id option_ids[] = {
    --snip--
      { USB_DEVICE(QUECTEL_VENDOR_ID, QUECTEL_PRODUCT_EC25),
        .driver_info = RSVD(4) },
    --snip--
}
```

kernel 中需要打开这些配置：

```
+CONFIG_PPP=y
+CONFIG_PPP_BSDCOMP=y
+CONFIG_PPP_DEFLATE=y
```

```
+CONFIG_PPP_FILTER=y
+CONFIG_PPP_MPPE=y
+CONFIG_PPP_MULTILINK=y
+CONFIG_PPPOE=y
+CONFIG_PPP_ASYNC=y
+CONFIG_PPP_SYNC_TTY=y
+CONFIG_SLHC=y
```

2）查看 kernel 识别模块

如果 kernel 增加了此模块的 VID 和 PID 配置，那么会生成/dev/ttyUSB*的节点：

```
root@myd-jx8mp:~# ls -l /dev/ttyUSB*
crw-rw---- 1 root dialout 188, 0 Apr 13 05:24 /dev/ttyUSB0
crw-rw---- 1 root dialout 188, 1 Apr 13 05:24 /dev/ttyUSB1
crw-rw---- 1 root dialout 188, 2 Apr 13 05:24 /dev/ttyUSB2
crw-rw---- 1 root dialout 188, 3 Apr 13 05:24 /dev/ttyUSB3
```

3）使用 AT 指令进行初步测试

使用 AT 指令可以方便的来查询信号强度，是否插入 SIM 卡，SIM 卡当前是否搜索到运营商，也可以用 AT 来打电话测试下当前卡功能。 这里进行 AT 通讯还需要知道哪个设备是通讯口，这里需要查询模块文件，EC20 采用 ttyUSB2 进行 AT 通讯。这里采用 microcom 举例，也可以用 minicom。如：microcom /dev/ttyUSB2 进入模式 ctrl+x 退出。

- **查询信号质量**

```
root@myd-jx8mp:~# microcom /dev/ttyUSB2
at+csq
+CSQ: 31,99

OK
```

  - 31,99: 31 就是信号质量，数字越小代表信号越强。

- **查看能否操作**

```
at+cpin?
+CPIN: READY
```

```
OK
```

- +CPIN:READY : READY 代表就绪。

- **查看运营商**

```
at+cops?
+COPS: 0,0,"CHN-CT",100

OK
```

- CHN-CT,100: CHN-CT 代表电信， 100 代表采用 2G，3G，4G，还是 5G 根据模块手册查看。

如果上述 3 步都能正常，就可以进行拨号上网，这里还介绍下如何打电话和发短信，进一步验证。

- **打电话**

```
ATD177xxxx5673;
OK
```

- **发短信**

```
at+cmgf=1
OK
at+cscs="GSM"
OK
at+cmgs="177xxxx5673"
> hello
+CMGS: 28

OK
```

- At+cmgf=1： 设置文本消息模式
- At+cscs=”GSM”：设置 TE 使用 GSM 字符
- At+cmgs: “电话” 写入消息后 CTRL+Z 发送， ECS 退出发送

4）ppp 拨号测试

这里采用开发板自带的 pppd 拨号命令：

```
root@myd-jx8mp:~# pppd call quectel-dial
root@myd-jx8mp::~# [    63.175640] [dhd] CFG80211-ERROR) wl_cfg80211_net
dev_notifier_call : wdev null. Do nothing
[    63.184467] [dhd] CFG80211-ERROR) wl_cfg80211_netdev_notifier_call : wde
v null. Do nothing
[    63.311254] [dhd] CFG80211-ERROR) wl_cfg80211_netdev_notifier_call : wde
v null. Do nothing
[    63.319869] [dhd] CFG80211-ERROR) wl_cfg80211_netdev_notifier_call : wde
v null. Do nothing
ca[    63.640433] [dhd] CFG80211-ERROR) wl_cfg80211_netdev_notifier_call : wd
ev null. Do nothing
[    63.649881] [dhd] CFG80211-ERROR) wl_cfg80211_netdev_notifier_call : wde
v null. Do nothing
```

这个拨号需要等待一会，拨号 log 已经隐藏，用户可以查看对应 log:

```
root@myd-jx8mp:~# cat /var/log/quectel-dial.log
pppd options in effect:
debug               # (from /etc/ppp/peers/quectel-dial)
persist             # (from /etc/ppp/peers/quectel-dial)
logfile /var/log/quectel-dial.log                    # (from /etc/ppp/peers/quectel-d
ial)
maxfail 0                   # (from /etc/ppp/peers/quectel-dial)
dump                # (from /etc/ppp/peers/quectel-dial)
noauth              # (from /etc/ppp/peers/quectel-dial)
user card                   # (from /etc/ppp/peers/quectel-dial)
password ??????             # (from /etc/ppp/peers/quectel-dial)
remotename 3gppp                    # (from /etc/ppp/peers/quectel-dial)
/dev/ttyUSB3                # (from /etc/ppp/peers/quectel-dial)
115200              # (from /etc/ppp/peers/quectel-dial)
lock                # (from /etc/ppp/peers/quectel-dial)
```

```
connect chat -s -v -f /etc/ppp/chatscripts/quectel-chat-connect          # (fro
m /etc/ppp/peers/quectel-dial)
disconnect chat -s -v -f /etc/ppp/chatscripts/quectel-chat-disconnect
 # (from /etc/ppp/peers/quectel-dial)
nocrtscts                  # (from /etc/ppp/peers/quectel-dial)
local              # (from /etc/ppp/peers/quectel-dial)
lcp-echo-failure 12                # (from /etc/ppp/peers/quectel-dial)
lcp-echo-interval 5                # (from /etc/ppp/peers/quectel-dial)
hide-password              # (from /etc/ppp/peers/quectel-dial)
ipcp-accept-local                  # (from /etc/ppp/peers/quectel-dial)
ipcp-accept-remote                  # (from /etc/ppp/peers/quectel-dial)
ipparam 3gppp              # (from /etc/ppp/peers/quectel-dial)
noipdefault                # (from /etc/ppp/peers/quectel-dial)
ipcp-max-failure 10                # (from /etc/ppp/peers/quectel-dial)
defaultroute               # (from /etc/ppp/peers/quectel-dial)
usepeerdns                 # (from /etc/ppp/peers/quectel-dial)
noccp              # (from /etc/ppp/peers/quectel-dial)
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)
^M
OK
 -- got it

send (ATE0^M)
expect (OK)
^M
^M
```

```
OK
 -- got it

send (AT+CGATT=0^M)
expect (OK)
^M
^M
OK
 -- got it

send (AT+CGATT=1^M)
expect (OK)
^M
^M
OK
 -- got it

send (ATI;+CSUB;+CSQ;+CPIN?;+COPS?;+CGREG?;&D2^M)
expect (OK)
^M
^M
Quectel^M
EC20F^M
Revision: EC20CEFDKGR06A07M2G^M
^M
SubEdition: V03^M
^M
+CSQ: 31,99^M
^M
+CPIN: READY^M
^M
+COPS: 0,0,"CHN-CT",100^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 (ATDT*99#^M)
expect (CONNECT)
^M
^M
CONNECT
 -- got it

Script chat -s -v -f /etc/ppp/chatscripts/quectel-chat-connect finished (pid 71
9), status = 0x0
Serial connection established.
using channel 1
Using interface ppp0
Connect: ppp0 <--> /dev/ttyUSB3
rcvd [LCP ConfReq id=0x0 <mru 1380> <asyncmap 0x0> <auth chap MD5>
<magic 0x8f54852a> <pcomp> <accomp>]
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x80f3e48a> <pcomp>
<accomp>]
sent [LCP ConfAck id=0x0 <mru 1380> <asyncmap 0x0> <auth chap MD5>
<magic 0x8f54852a> <pcomp> <accomp>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x80f3e48a> <pcomp> <
accomp>]
```

```
sent [LCP EchoReq id=0x0 magic=0x80f3e48a]
rcvd [CHAP Challenge id=0x1 <4d359fbd8ebea60e4150f11375f05e4e>, name
= "HBTELECOM"]
sent [CHAP Response id=0x1 <aef3a25cdbbecb0e00c7fd4b3087fd6b>, name =
 "card"]
rcvd [LCP EchoRep id=0x0 magic=0x8f54852a]
rcvd [CHAP Success id=0x1 "Welcome to HBTELECOM."]
CHAP authentication succeeded: Welcome to HBTELECOM.
CHAP authentication succeeded
sent [IPCP ConfReq id=0x1 <compress VJ 0f 01> <addr 0.0.0.0> <ms-dns1 0.
0.0.0> <ms-dns2 0.0.0.0>]
rcvd [IPCP ConfReq id=0x0 <addr 172.16.160.166>]
sent [IPCP ConfAck id=0x0 <addr 172.16.160.166>]
rcvd [IPCP ConfRej id=0x1 <compress VJ 0f 01>]
sent [IPCP ConfReq id=0x2 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.
0>]
rcvd [IPCP ConfNak id=0x2 <addr 10.100.31.254> <ms-dns1 202.103.24.68> <
ms-dns2 202.103.44.150>]
sent [IPCP ConfReq id=0x3 <addr 10.100.31.254> <ms-dns1 202.103.24.68> <
ms-dns2 202.103.44.150>]
rcvd [IPCP ConfAck id=0x3 <addr 10.100.31.254> <ms-dns1 202.103.24.68> <
ms-dns2 202.103.44.150>]
local  IP address 10.100.31.254
remote IP address 172.16.160.166
primary   DNS address 202.103.24.68
secondary DNS address 202.103.44.150
Script /etc/ppp/ip-up started (pid 728)
Script /etc/ppp/ip-up finished (pid 728), status = 0x0
```

可以看到已经正常连接，能获取 IP。

输入以下命令查看”ppp”发现已经正常获取 IP：

```
root@myd-jx8mp:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 52:7c:0c:f7:1e:ff
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
```

```
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:82 errors:0 dropped:0 overruns:0 frame:0
          TX packets:82 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6220 (6.0 KiB)  TX bytes:6220 (6.0 KiB)

ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.100.31.254  P-t-P:172.16.160.166  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1380  Metric:
1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:220 (220.0 B)  TX bytes:662 (662.0 B)
```

- ppp0 ： ppp0 即为拨号网卡设备，ip 地址正常获取。

5) 采用 qmi_wwan 拨号

使用 quectel-CM 拨号，使用方法 quectel-CM -s ctnet & 。

```
root@myd-jx8mp:~# quectel-CM -s ctnet &

[1] 634

[09-20_11:45:34:450] Quectel_QConnectManager_Linux_V1.6.0.16

[09-20_11:45:34:451] Find /sys/bus/usb/devices/3-1.2 idVendor=0x2c7c idProdu
ct=0x800, bus=0x003, dev=0x003
```

```
[09-20_11:45:34:452] Auto find qmichannel = /dev/cdc-wdm0

[09-20_11:45:34:452] Auto find usbnet_adapter = rmnet_usb0

[09-20_11:45:34:452] netcard driver = qmi_wwan_q, driver version = V1.2.0.14

[09-20_11:45:34:452] qmap_mode = 1, qmap_version = 9, qmap_size = 31744,
 muxid = 0x81, qmap_netcard = rmnet_usb0.1

[09-20_11:45:34:452] Modem works in QMI mode

[09-20_11:45:34:464] cdc_wdm_fd = 7

root@myd-jx8mp:~# [09-20_11:45:35:466] QmiThreadSendQMITimeout pthread
_cond_timeout_np timeout

[09-20_11:45:36:745] Get clientWDS = 14

[09-20_11:45:36:777] Get clientDMS = 1

[09-20_11:45:36:809] Get clientNAS = 4

[09-20_11:45:36:841] Get clientUIM = 1

[09-20_11:45:36:873] Get clientWDA = 1

[09-20_11:45:36:905] requestBaseBandVersion RM500QCNAAR12A06M4G

[    50.887296] net rmnet_usb0: ul_data_aggregation_max_datagrams=11, ul_dat
a_aggregation_max_size=4096, dl_minimum_padding=0

[09-20_11:45:36:937] qmap_settings.rx_urb_size = 31744

[09-20_11:45:36:937] qmap_settings.ul_data_aggregation_max_datagrams  = 11

[09-20_11:45:36:937] qmap_settings.ul_data_aggregation_max_size       = 4096

[09-20_11:45:36:937] qmap_settings.dl_minimum_padding                 = 0
```

```
[09-20_11:45:37:065] requestGetSIMStatus SIMStatus: SIM_READY

[09-20_11:45:37:097] requestGetProfile[1] ///0

[09-20_11:45:37:129] requestRegistrationState2 MCC: 460, MNC: 0, PS: Attached, DataCap: 5G_SA

[09-20_11:45:37:161] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED

[09-20_11:45:37:161] ifconfig rmnet_usb0 down

[09-20_11:45:37:170] ifconfig rmnet_usb0.1 0.0.0.0

SIOCSIFFLAGS: Network is down

[09-20_11:45:37:176] ifconfig rmnet_usb0.1 down

[09-20_11:45:37:833] requestSetupDataCall WdsConnectionIPv4Handle: 0x3b0d6530

[    51.911181] net rmnet_usb0: link_state 0x0 -> 0x1

[09-20_11:45:37:966] ifconfig rmnet_usb0 up

[09-20_11:45:37:972] ifconfig rmnet_usb0.1 up

[09-20_11:45:37:980] busybox udhcpc -f -n -q -t 5 -i rmnet_usb0.1

udhcpc: started, v1.32.0

udhcpc: sending discover

udhcpc: sending select for 10.30.168.127

udhcpc: lease of 10.30.168.127 obtained, lease time 7200

RTNETLINK answers: File exists
```

```
[09-20_11:45:38:135] /etc/udhcpc.d/50default: Adding DNS 211.137.64.163

[09-20_11:45:38:135] /etc/udhcpc.d/50default: Adding DNS 211.137.58.20
```

- -s ctnet 电信
- -s cmnet 移动
- -s 3gnet 联通

ifconfig 可以看到能正常获取 IP:

```
root@myd-jx8mp:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 1a:c2:0b:54:b2:9d
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:82 errors:0 dropped:0 overruns:0 frame:0
          TX packets:82 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6220 (6.0 KiB)  TX bytes:6220 (6.0 KiB)

wwan0     Link encap:Ethernet  HWaddr a6:28:32:50:90:df
          inet addr:10.100.46.11  Bcast:10.100.46.15  Mask:255.255.255.248
          UP BROADCAST RUNNING NOARP MULTICAST  MTU:1380  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)
```

5. 网络应用

设备出厂烧录的 myir-image-full 镜像默认包含了一些常见的网络应用程序，方便用户进行开发或调试。

5.1. PING

PING 主要用来测试网络的连通性，也可以测试网络延迟以及丢包率。按照 4.1.1 中配置好以太网连接之后就可以使用 PING 对网络连接进行简单的测试。

1）接线与信息输出

通过 CAT6 网线将设备连接到交换机或路由器，控制台会显示内核输出的连接信息，如下：

```
[ 2983.858754] fec 30be0000.ethernet eth0: Link is Up - 1Gbps/Full - flow control rx/tx
```

2）测试外网网址

```
root@myd-jx8mp:~# ping www.baidu.com -I eth0
PING www.baidu.com (14.215.177.38) from 192.168.40.100 eth0: 56(84) bytes of data.
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=1 ttl=54 time=19.9 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=2 ttl=54 time=19.6 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=3 ttl=54 time=19.6 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=4 ttl=54 time=19.6 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=5 ttl=54 time=19.8 ms
```

注：ping 公网需要确保 DNS 正常工作。

上面结果显示 www.baidu.com 经过域名解析之后的 IP 地址为 14.215.177.38，icmp_seq 代表 icmp 包的编号，如果编号连续说明没有丢包；time 代表响应的延迟时间，当然这个时间越短越好。除了对以太网进行测试，ping 命令也可以用于测试 Wi-Fi。

5.2. SSH

SSH 为 Secure Shell 的缩写，由 IETF 的网络小组（Network Working Group）所制定；SSH 为建立在应用层基础上的安全协议， 是较可靠，专为远程登录会话和其他网络服务提供安全性的协议。通常 Linux 平台下使用 dropbear 或 OpenSSH 来实现 SSH 的服务端和客户端。下面在以太网连接上分别测试 SSH 客户端和服务端的使用。当前出厂默认包含 openssh 7.6p1（http://www.openssh.com/）提供的客户端和服务程序。

首先按照 4.1 节配置好以太网接口到 SSH 服务器的连接，配置后的以太网卡地址如下：

```
root@myd-jx8mp:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 36:4a:23:f7:45:c2
          inet addr:192.168.40.100  Bcast:192.168.40.255  Mask:255.255.255.0
          inet6 addr: fe80::344a:23ff:fef7:45c2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9078 errors:0 dropped:0 overruns:0 frame:0
          TX packets:83 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8015784 (7.6 MiB)  TX bytes:10787 (10.5 KiB)
```

SSH 服务器的 IP 地址为 192.168.40.2，用 ping 命令可测试开发板和 SSH 服务器之间的连接是否正常。

- **SSH 客户端测试**

开发板作为客户端连接 SSH 服务器，在开发板上使用 ssh 命令登陆服务器，命令和结果如下：

```
root@myd-jx8mp:~# ssh wujl@192.168.40.2
Host '192.168.40.2' is not in the trusted hosts file.
(ecdsa-sha2-nistp256 fingerprint sha1!! 17:59:83:29:04:cc:97:63:b8:82:52:73:3c:47:
70:80:8a:5d:2a:37)
Do you want to continue connecting? (y/n) y
wujl@192.168.40.2's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-101-generic x86_64)
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
```

```
 * Support:        https://ubuntu.com/advantage

779 个可升级软件包。
573 个安全更新。
New release '18.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
Last login: Fri Nov 20 13:53:53 2020 from 192.168.40.143
```

其中 wujl 为服务器上的用户名。

登录成功之后，自动进入 SSH 服务器上的 console 控制台，用户就可以在客户端对远程服务器执行 wujl 用户权限内的控制。如果需要退出，直接在当前控制台执行"exit"命令即可。

- **SSH 服务端测试**

开发板作为 SSH 服务端，其它外部设备远程连接到此台开发板。

开发板端默认启动了 SSH 服务，因此我们也可以在其它具有 SSH 客户端的外部设备（开发板或者 PC）上使用 ssh 命令登陆到当前的开发板上，命令和结果如下：

```
duxy@myir_server:~$ ssh root@192.168.40.100
The authenticity of host '192.168.40.100 (192.168.40.100)' can't be established.
RSA key fingerprint is SHA256:6T7Bvwtc+MU8nlbdoU9YH0qonwuvWPdb8D+F+ae5ktA.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.40.100' (RSA) to the list of known hosts.
```

上面的示例中，我们从远程以 root 账户登录到了此开发板上，并进入 console 控制台，可以对开发板执行 root 用户权限内的控制。如果需要退出，直接在控制台执行"exit"命令即可。

OpenSSH 是使用 SSH 协议远程登录的主要连接工具。它加密所有流量以消除窃听、连接劫持和其他攻击。此外，OpenSSH 还提供一系列大型安全隧道功能、多种身份验证方法和复杂灵活的配置选项。用户可以根据自身需要修改位于/etc/ssh/目录下的配置文件 ssh_config 和 sshd_config。

例如，如果希望 SSH 服务端允许 root 账户不用密码远程登录，则可以修改 SSH 服务端设备上的/etc/ssh/sshd_config，如在主机上添加下面两行配置：

```
PermitRootLogin yes
PermitEmptyPasswords yes
```

上面的配置有比较大的安全风险，一般用于调试阶段远程部署。实际产品中考虑到安全性，一般都是关掉的。

5.3. SCP

SCP 是 Secure Copy 的缩写，它是 linux 系统下基于 SSH 协议的安全的远程文件拷贝命令，在系统调试阶段非常实用。

在 4.2.2 中我们已经介绍过使用 SSH 协议以及 SSH 客户端和服务端进行远程登录的示例，这里再介绍通过 SCP 命令进行文件远程拷贝的示例：

1）从远程拷贝文件到本地

```
duxy@myir_server:~$ scp gpiotest root@192.168.40.100:/home/root
gpiotest
 100%    13KB   13.1KB/s    00:00
duxy@myir_server:~$
```

进入开发板 home/root 目录可以看到此文件，如下：

```
root@myd-jx8mp:~# ls -l gpiotest
-rwxr-xr-x 1 root root 13448 Nov 11 05:59 gpiotest
root@myd-jx8mp:~# pwd
/home/root
```

2）从本地拷贝文件到远程

```
root@myd-jx8mp:~# scp gpiotest wujl@192.168.40.2:~/
wujl@192.168.40.2's password:
gpiotest
 100%    13KB   13.1KB/s    00:00
```

进入服务器当前目录，可以看到此文件，如下：

```
duxy@myir_server:~$ ls -l gpiotest
-rwxr-xr-x 1 duxy duxy 13448 11 月 20 14:10 gpiotest
```

拷贝的过程中需要按照提示输入，验证成功之后文件从设备上拷贝到服务器上指定账户的$HOME 目录。

通过添加”-r“参数，还可以进行目录的拷贝，具体操作请参照 scp 命令的帮助。

5.4. FTP

FTP(File Transfer Protocol)是一种进行文件传输的网络协议。该协议遵循客户机-服务器通信模型。要使用 FTP 传输文件，用户需要运行一个 FTP 客户端程序，并启动与运行 FTP 服务器软件的远程计算机的连接。建立连接后，客户端可以选择发送和/或接收文件的副本。FTP 服务器在 TCP 端口 21 上侦听来自 FTP 客户端的连接请求。当接收到请求时，服务器使用这个端口来控制连接，并打开一个单独的端口来传输文件数据。

默认出厂的 myir-image-full 镜像包含有 FTP 客户端程序 ftp 和服务端程序 proftpd(具体移植过程参见《MYD-JX8MPQ Linux 软件开发指南》7.3 节应用移植)启动开发板，设备端 FTP 根目录位于/var/lib/ftp，下面测试在局域网内使用 ftp 命令匿名登陆到目标设备并从开发主机（IP 地址为 192.168.40.143)向目标设备（IP 地址为 192.168.40.100）传输文件的示例。

1）开发主机使用 FTP 登录目标设备

开发主机为 Windows10 系统的 PC 机，打开 cmd 命令窗口，用 FTP 登陆到开发板，用户名为 ftp，密码任意。

```
C:\Users\10539>ftp 192.168.40.100
连接到 192.168.40.100。
220 ProFTPD Server (ProFTPD Default Installation) [192.168.40.100]
500 OPTS UTF8 not understood
用户(192.168.40.100:(none)): ftp
331 Anonymous login ok, send your complete email address as your passwor
d
密码:
230 Anonymous access granted, restrictions apply
ftp> pwd
257 "/" is the current directory
ftp>
```

此时当前目录已经是开发板/var/lib/ftp，然后可以下载和上传文件。

2）目标设备上创建测试文件

```
root@myd-jx8mp:~# cd /var/lib/ftp/
root@myd-jx8mp:/var/lib/ftp# touch test
```

```
root@myd-jx8mp:/var/lib/ftp# ls
test
```

3）开发主机 FTP 查看当前目录

```
ftp> ls
200 PORT command successful
150 Opening ASCII mode data connection for file list
test
226 Transfer complete
ftp: 收到 9 字节，用时 0.00 秒 9000.00 千字节/秒。
```

4）下载文件

FTP 命令行支持使用 mget 子命令下载多个文件，但是每个文件中间需要用户确认，如下所示：

```
ftp> mget aaa test
200 Type set to A
mget aaa? y
200 PORT command successful
150 Opening ASCII mode data connection for aaa
226 Transfer complete
mget test? y
200 PORT command successful
150 Opening ASCII mode data connection for test
226 Transfer complete
ftp>
ftp> bye
221 Goodbye.
```

输入命令“bye”即退出 ftp，然后 windows10 主机磁盘 C:\Users\10539 目录可以找到从开发板下载的文件，如下：

```
C:\Users\10539>dir -l test aaa
 驱动器 C 中的卷是 Windows-SSD
 卷的序列号是 94C5-2171
 C:\Users\10539 的目录
```

```
 C:\Users\10539 的目录
2020/11/20   14:33                    0 test
 C:\Users\10539 的目录
2020/11/20   14:33                    0 aaa
               2 个文件                  0 字节
               0 个目录 60,599,259,136 可用字节
```

5）上传文件

FTP 需要普通用户或者 root 用户登陆才能上传文件，这里使用 root 用户登陆,用户名为“root”，没有密码，直接按“Enter”键，如需要密码用户可以在目标开发板直接 passwd 设置 root 密码。

- **设置 root 密码**

在开发板上输入下面命令设置 root 密码，这里我设置 root 密码为“123”：

```
root@myd-jx8mp:/var/lib/ftp# passwd root
New password:
Retype new password:
passwd: password updated successfully
```

- **修改配置并重启服务**

如果需要开放 root 账户登录 FTP 服务器，需要先修改/etc/proftpd.conf 文件，在文件中增加一行配置 "RootLogin on"。

```
systemctl restart proftpd
```

- **FTP 连接到目标板**

```
C:\Users\10539>ftp 192.168.40.100
连接到 192.168.40.100。
220 ProFTPD Server (ProFTPD Default Installation) [192.168.40.100]
500 OPTS UTF8 not understood
用户(192.168.40.100:(none)): root
331 Password required for root
密码:
230 User root logged in
ftp> pwd
257 "/home/root" is the current directory
```

- **上传文件**

FTP 文件传输支持 ASCII 模式和 Binary 模式，上面下载文件默认采用的是 ASCII 模式，如果是传输二进制文件，如目标设备上的可执行程序，尽量采用 Binary 模式。如下所示：

```
ftp> binary
200 Type set to I
ftp> put myapp
local: myapp remote: myapp
200 PORT command successful.
150 Opening BINARY mode data connection for 'myapp'.
226 Transfer complete.
1024 bytes sent in 0.02 secs (63.7308 kB/s)
ftp>bye
```

FTP 可追溯到 20 世纪 70 年代初，编写时没有任何安全考虑。它不对任何内容使用加密。登录凭据（如用户名和密码）以及您下载或上传的数据以明文传输。因此它不太适合在 Internet 上传输敏感信息，然而考虑到其良好的兼容性和稳定性，在局域网内使用还是很方便的。

5.5. TFTP

与 FTP 一样，TFTP 使用客户端和服务器软件在两台设备之间进行连接和传输文件，但不同的是 TFTP 使用的是 UDP 协议，不具备登录功能，它非常简洁，特别适合在设备和服务器端传输和备份固件，配置文件等信息。例如常见的 u-boot 中就支持 TFTP 协议，可以通过网络加载服务器端的 Linux 系统并实现网络启动的功能。

默认的镜像文件包含 busybox 提供的 tftp 客户端程序，其命令语法如下：

```
root@myd-jx8mp:~# tftp --help
BusyBox v1.31.0 (2020-09-05 00:06:04 UTC) multi-call binary.

Usage: tftp [OPTIONS] HOST [PORT]
```

详细参数说明如下：

- -g： 获取文件
- -p： 上传文件
- -l ： 本地文件
- -r ： 远程文件
- HOST: 远程主机 IP 地址
- [PORT]: 可忽略，默认为 69

TFTP 服务端可以选择 Linux 平台下的 tftp-hpa,也可以选择 windows 平台下的 tftpd32/64(http://tftpd32.jounin.net/tftpd32_download.html)。下面以 ubuntu 平台为例说明 tftp 服务端的配置。

1）安装 TFTP 服务端

```
$ sudo apt-get install tftp-hpa tftpd-hpa
```

2）配置 TFTP 服务

创建 TFTP 服务器工作目录,并打开 TFTP 服务配置文件,如下:

```
$ mkdir -p <WORKDIR>/tftpboot
$ chmod -R 777 <WORKDIR>/tftpboot
$ sudo vi /etc/default/tftpd-hpa
```

修改或添加以下字段：

```
TFTP_DIRECTORY="<WORKDIR>/tftpboot"
TFTP_OPTIONS="-l -c -s"
```

3）重启 TFTP 服务

```
$ sudo service tftpd-hpa restart
```

配置好 tftp 服务端之后，将一个测试文件 zImage 放置到上面配置的<WORKDIR>/tftpboot/目录，就可以在目标设备上使用 tftp 客户端进行文件的下载和上传了。

```
root@myd-jx8mp:~# tftp -g -r zImage -l zImage 192.168.0.2
```

上面的命令会把 tftp 服务端<WORKDIR>/tftpboot 目录下的 zImage 下载到目标设备当前目录下。

```
root@myd-jx8mp:~# tftp -p -l config -r config_01 192.168.0.2
```

上面的命令会把目标设备上当前目录下的 config 文件上传到 tftp 服务端之前配置的<WORKDIR>/tftpboot 目录下，并重新命名为 config_01。

5.6. DHCP

DHCP（动态主机配置协议）是一个局域网的网络协议。指的是由服务器控制一段 IP 地址范围，客户机登录服务器时就可以自动获得服务器分配的 IP 地址和子网掩码。

DHCP 也包含服务器端和客户端两种角色，在 4.1 中我们已经测试过使用 DHCP 客户端模式(udhcpc)自动获取 IP 地址；在 4.2 中配置 WiFi 的 AP 模式时，我们也测试了 DHCP 服务端模式(udhcpd)给连接的 WiFi 设备分配 IP 地址。这里再介绍一下使用 dhclient 命令和 udhcpc 命令手动获取 IP 地址的方法，方便用户在调试网络时使用。

用 CAT6 网线连接开发板和路由器，使用命令手动为 eth0 网卡分配 IP 地址，观察 dhcp 获取 ip 的过程。

使用 udhcpc 命令配置 IP 地址

```
root@myd-jx8mp:~# udhcpc -i eth0
udhcpc: started, v1.31.0
udhcpc: sending discover
udhcpc: sending select for 192.168.40.153
udhcpc: lease of 192.168.40.153 obtained, lease time 7200
/etc/udhcpc.d/50default: Adding DNS 223.5.5.5
/etc/udhcpc.d/50default: Adding DNS 201.104.111.114
```

这里使用开发板终端作为 dhcp 客户端，动态获取 IP。

5.7. IPTables

iptables 是一个用于 IPv4 包过滤和 NAT 的管理工具。它用于设置、维护和检查 Linux 内核中的 IP 包过滤规则表。可以定义几个不同的表。每个表包含许多内置链，也可以包含用户定义的链。每个链是一个规则列表，它可以匹配一组数据包。每个规则指定如何处理匹配的数据包。

使用 Linux 系统的设备通常使用 iptables 工具来配置防火墙。iptables 就根据包过滤规则所定义的方法来处理各种数据包，如放行（accept）、拒绝（reject）和丢弃（drop）等。下面使用 iptables 来测试拦截 icmp 包，禁止网络上的其它设备对其进行 ping 探测。具体命令使用参见：https://linux.die.net/man/8/iptables。

1）配置目标设备 iptables

在目标设备上使用 iptables 配置丢弃输入的 icmp 包，不回应其他主机的 ping 探测，命令如下：

```
root@myd-jx8mp:~# iptables -A INPUT -p icmp --icmp-type 8 -j DROP
root@myd-jx8mp:~# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -p icmp -m icmp --icmp-type 8 -j DROP
```

2）测试 ping 目标设备

在开发主机上 ping 目标设备，并指定 deadline 为 10，结果如下：

```
PC$ ping 192.168.40.100 -w 10
PING 192.168.40.100 (192.168.40.100) 56(84) bytes of data.

--- 192.168.40.100 ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, time 9197ms
```

以上结果表明，设置防火墙后开发主机无法 ping 通目标设备。

3）删掉对应的防火墙规则

```
root@myd-jx8mp:~# iptables -F
root@myd-jx8mp:~# iptables -S
```

```
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
```

4) 再次测试 ping 目标设备

```
PC$ ping 192.168.40.100 -w 5
PING 192.168.40.100 (192.168.40.100) 56(84) bytes of data.
64 bytes from 192.168.40.100: icmp_seq=1 ttl=64 time=86.5 ms
64 bytes from 192.168.40.100: icmp_seq=2 ttl=64 time=193 ms
64 bytes from 192.168.40.100: icmp_seq=3 ttl=64 time=167 ms
64 bytes from 192.168.40.100: icmp_seq=4 ttl=64 time=74.1 ms
64 bytes from 192.168.40.100: icmp_seq=5 ttl=64 time=109 ms

--- 192.168.40.100 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 74.178/126.267/193.950/46.644 ms
```

清除 iptables 规则之后，再次从开发主机 ping 目标设备，就可以 ping 通了。上述示例只是一个简单的演示，实际上 iptables 配合各种规则可以实现非常强大的功能，这里就不详细介绍了。

5.8. Ethtool

ethtool 是一个查看和修改以太网设备参数的工具，在网络调试阶段具有一定的作用，下面使用该命令查看一下以太网卡的信息，并尝试修改其参数。

首先，我们通过 ethtool -h 查看该命令的帮助信息：

```
root@myd-jx8mp:~# ethtool --help
ethtool version 5.8
Usage:
        ethtool [ --debug MASK ][ --json ] DEVNAME       Display standard i
nformation about device
        ethtool [ --debug MASK ][ --json ] -s|--change DEVNAME  Change ge
neric options
                [ speed %d ]
                [ duplex half|full ]
                [ port tp|aui|bnc|mii|fibre|da ]
.................
```

查看当前设备以太网卡的基本信息：

```
root@myd-jx8mp:~# ethtool eth1
Settings for eth1:
        Supported ports: [ TP     MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Supported pause frame use: Symmetric Receive-only
        Supports auto-negotiation: Yes
        Supported FEC modes: Not reported
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Advertised pause frame use: Symmetric Receive-only
        Advertised auto-negotiation: Yes
        Advertised FEC modes: Not reported
        Link partner advertised link modes:  10baseT/Half 10baseT/Full
```

```
                                        100baseT/Half 100baseT/Full
                                        1000baseT/Full
        Link partner advertised pause frame use: No
        Link partner advertised auto-negotiation: Yes
        Link partner advertised FEC modes: Not reported
        Speed: 1000Mb/s
        Duplex: Full
        Auto-negotiation: on
        Port: MII
        PHYAD: 0
        Transceiver: external
        Supports Wake-on: ug
        Wake-on: d
        Current message level: 0x0000003f (63)
                               drv probe link timer ifdown ifup
        Link detected: yes
```

通过 ethtool 命令可以查看到当前以太网卡支持的连接模式为十兆，百兆和千兆半双工与全双工六种模式，当前连接状态为协商的千兆，全双工模式，使用 MII 接口，PHY 地址为 4 等等。

我们还可以使用 ethtool 工具对以太网的参数进行设置，这些在进行以太网调试和诊断的时候有一定的作用，例如我们强制将以太网设置为百兆全双工，并且关闭自协商，命令如下：

```
root@myd-jx8mp:~# ethtool -s eth1 speed 100 duplex full autoneg off
```

关于 ethtool 的更多说明请参考：http://man7.org/linux/man-pages/man8/ethtool.8.html。

5.9. iPerf3

iPerf3 是在 IP 网络上主动测量最大可实现带宽的工具。 它支持调节测试时间、缓冲区大小和协议(TCP、UDP、带有 IPv 的 SCTP)等各种参数。 iPerf3 按角色可以分为服务端模式或客户端模式，我们可以用它来测试和查看 TCP 模式下的网络带宽，TCP 窗口值，重传的概率等，也可以测试指定 UDP 带宽下丢包率，延迟和抖动情况。

我们以一台 windows 主机，带千兆网卡的服务器作为 iperf3 的服务端，被测试的设备作为客户端分别测试设备以太网卡 TCP 和 UDP 的性能。

首先在服务器上安装 iperf3，如下：

```
PC $ sudo apt-get install iperf3
```

将服务器和设备通过 CAT6 网线直连，并配置好各自的 IP 地址。例如我们设置服务器 ip 为 192.168.40.143，设备 IP 为 192.168.40.100，并使用 ping 命令测试确保它们之间是连通的。

注意：尽量不要连接路由器或交换机，以免测试结果受到中间设备的影响。

1）测试 TCP 性能

服务端（192.168.40.143），服务器上 iperf3 使用-s 参数表示工作在服务端模式：

```
PC $ $ iperf3 -s -i 2
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

客户端（192.168.40.100）：

设备上 iperf3 工作在客户端，TCP 模式，其中参数说明如下：

- -c 192.168.40.143：工作在客户端，连接服务端 192.168.40.143
- -i 2：测试结果报告时间间隔为 2 秒
- -t 10：总测试时长为 10 秒

```
root@myd-jx8mp:~# iperf3 -c 192.168.40.143 -i 2 -t 10
Connecting to host 192.168.40.143, port 5201
[ 5] local 192.168.40.100 port 49692 connected to 192.168.40.143 port 5201
[ ID] Interval Transfer Bitrate Retr Cwnd
[ 5] 0.00-2.00 sec 218 MBytes 914 Mbits/sec 242 625 KBytes
```

```
[  5]   2.00-4.00   sec   219 MBytes   918 Mbits/sec   14    551 KBytes

[  5]   4.00-6.00   sec   221 MBytes   927 Mbits/sec    0    609 KBytes

[  5]   6.00-8.00   sec   220 MBytes   923 Mbits/sec    0    631 KBytes

[  5]   8.00-10.00  sec   220 MBytes   923 Mbits/sec  369    554 KBytes

- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  1.07 GBytes   921 Mbits/sec  625             sender
[  5]   0.00-10.00  sec  1.07 GBytes   918 Mbits/sec                  receiver

iperf Done.
```

客户端经过 10 秒之后测试结束并显示上面的测试结果，表明 TCP 带宽为 921Mbps 左右，没有重传，测试时 TCP 窗口值为 554KBytes.

同时服务端也显示测试结果如下，然后继续监听 5201 端口等待客户端连接：

```
PC$  iperf3 -s -i 2
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.40.100, port 49690
[  5] local 192.168.40.143 port 5201 connected to 192.168.40.100 port 49692
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-2.00   sec   215 MBytes   901 Mbits/sec
[  5]   2.00-4.00   sec   219 MBytes   918 Mbits/sec
[  5]   4.00-6.00   sec   221 MBytes   925 Mbits/sec
[  5]   6.00-8.00   sec   221 MBytes   927 Mbits/sec
[  5]   8.00-10.00  sec   219 MBytes   920 Mbits/sec
[  5]  10.00-10.00  sec   283 KBytes   920 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
```

```
[  5]   0.00-10.00  sec  1.07 GBytes   921 Mbits/sec  625             sender
[  5]   0.00-10.00  sec  1.07 GBytes   918 Mbits/sec                  receiver
-----------------------------------------------------------
Server listening on 5201
```

2）测试 UDP 性能

服务端（192.168.40.143 ）

服务器上继续运行 iperf3 使用-s 参数表示工作在服务端模式。

```
PC $ $ iperf3 -s -i 2
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

客户端（192.168.40.100）

设备上 iperf3 工作在客户端，UDP 模式，其中参数说明如下：

- -u：工作在 UDP 模式
- -c 192.168.40.143：工作在客户端，连接服务端 192.168.40.143
- -i 2：测试结果报告时间间隔为 2 秒
- -t 10：总测试时长为 10 秒
- -b 100M：设定 UDP 传输带宽为 100Mbps.

```
root@myd-jx8mp:~# iperf3 -u -c 192.168.40.143 -i 2 -t 10 -b 100M
Connecting to host 192.168.40.143, port 5201
[  5] local 192.168.40.100 port 40126 connected to 192.168.40.143 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  5]   0.00-2.00   sec  23.8 MBytes   100 Mbits/sec  17259
[  5]   2.00-4.00   sec  23.8 MBytes   100 Mbits/sec  17265
[  5]   4.00-6.00   sec  23.8 MBytes   100 Mbits/sec  17265
[  5]   6.00-8.00   sec  23.8 MBytes   100 Mbits/sec  17265
[  5]   8.00-10.00  sec  23.8 MBytes   100 Mbits/sec  17265
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagra
ms
[  5]   0.00-10.00  sec   119 MBytes   100 Mbits/sec  0.000 ms  0/86319
(0%)  sender
```

```
[  5]   0.00-10.00  sec   119 MBytes  99.4 Mbits/sec  0.186 ms  466/86313
(0.54%)  receiver

iperf Done.
```

客户端经过 10 秒之后测试结束并显示上面的测试结果，表明 UDP 在指定带宽为 100Mbps 时没有丢包。

同时服务端也显示测试结果如下，然后继续监听 5201 端口等待客户端连接：

```
$ $ iperf3 -s -i 2
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Server listening on 5201
Accepted connection from 192.168.40.100, port 49694
[  5] local 192.168.40.143 port 5201 connected to 192.168.40.100 port 40126
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datag
rams
[  5]   0.00-2.00   sec  23.8 MBytes  99.8 Mbits/sec  0.230 ms  0/17240
(0%)
[  5]   2.00-4.00   sec  23.8 MBytes  99.6 Mbits/sec  0.161 ms  67/17267 (0.
39%)
[  5]   4.00-6.01   sec  23.6 MBytes  98.7 Mbits/sec  0.926 ms  124/17208
(0.72%)
[  5]   6.01-8.00   sec  23.7 MBytes  99.8 Mbits/sec  0.171 ms  136/17330
(0.78%)
[  5]   8.00-10.00  sec  23.6 MBytes  99.2 Mbits/sec  0.186 ms  139/17258
(0.81%)
[  5]  10.00-10.00  sec  14.1 KBytes  61.1 Mbits/sec  0.186 ms  0/10 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datag
rams
[  5]   0.00-10.00  sec   119 MBytes   100 Mbits/sec  0.186 ms  466/86313
(0.54%)
-----------------------------------------------------------
```

```
Server listening on 5201
-----------------------------------------------------------
```

客户端修改-b 参数，继续增大指定的 UDP 带宽，发送端能达到的最大速率即是最大带宽，丢包率取决于服务器端 CPU 性能，网卡 buffer 大小。

```
root@myd-jx8mp:~# iperf3 -u -c 192.168.40.143 -i 2 -t 10 -b 1000M
Connecting to host 192.168.40.143, port 5201
[  5] local 192.168.40.100 port 45388 connected to 192.168.40.143 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  5]   0.00-2.00   sec   238 MBytes  1000 Mbits/sec  172623
[  5]   2.00-4.00   sec   238 MBytes  1000 Mbits/sec  172643
[  5]   4.00-6.00   sec   238 MBytes  1.00 Gbits/sec  172673
[  5]   6.00-8.00   sec   238 MBytes  1000 Mbits/sec  172602
[  5]   8.00-10.00  sec   238 MBytes  1.00 Gbits/sec  172707
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagra
ms
[  5]   0.00-10.00  sec  1.16 GBytes  1000 Mbits/sec  0.000 ms  0/863248
(0%)  sender
[  5]   0.00-10.00  sec   979 MBytes   821 Mbits/sec  0.014 ms  153823/862
946 (18%)  receiver

iperf Done.
```

iperf3 在测试的过程中还有很多参数可以配置，用户可以根据实际应用需要进行有针对性的调整测试。比如可以增大-t 参数的值进行长时间压力测试，或者指定-P 参数进行多个连接并发的压力测试等。关于 iperf3 测试的更多信息请参考：https://iperf.fr/iperf-doc.php#3doc。

6. 图形系统

6.1. GPU

MYD-JX8MPQ 拥有 2 个 GPU 核，一个用来做 3D 数据处理，另一个用来做 2D 和 3D 加速。3D GPU 核支持：

- OpenGL ES 1.1 ， 2.0 ,3.0,3.1
- Open CL 1.2
- 2D GPU 核支持
- 多图层混合

1) OpenGL ES2.0

myir-image-full 镜像中已有打包好的测试镜像，目录在/opt/imx-gpu-sdk/GLES2 这里有如下例子供测试：

```
root@myd-jx8mp:/opt/imx-gpu-sdk/GLES2# ls
Bloom                  DFSimpleUI100      InputEvents          S01_SimpleTriangle
 S06_Texturing                    Stats
Blur                   DFSimpleUI101      LineBuilder101       S02_ColoredTriangle
S07_EnvironmentMapping           T3DStressTest
DFGraphicsBasic2D      EightLayerBlend    ModelLoaderBasics    S03_Transform
  S08_EnvironmentMappingRefraction    TextureCompression
DFNativeBatch2D        FractalShader      ModelViewer          S04_Projection
 S09_VIV_direct_texture           VIVDirectTextureMultiSampling
```

用 S08_EnvironmentMappingRefraction 进行测试：

```
root@myd-jx8mp:/opt/imx-gpu-sdk/GLES2# ./S08_EnvironmentMappingRefracti
on/GLES2.S08_EnvironmentMappingRefraction_Wayland
```

除此之外，还有 glmark 自带的 glmark2-es2-wayland 程序，以及 /user/share/example/opengl 下的 QT 例子可以测试。

2）OpenVG

myir-image-full 镜像中已有打包好的测试镜像，目录在/opt/imx-gpu-sdk/OpenVG，这里有如下例子供测试：

```
root@myd-jx8mp:/opt/imx-gpu-sdk/OpenVG# ls
BitmapFont  CoverFlow  DFGraphicsBasic2D  Example1  Example2  Example3
 SimpleBench  VGStressTest
```

运行 Example3 里面的例子：

```
root@myd-jx8mp::/opt/imx-gpu-sdk/OpenVG# ./Example3/OpenVG.Example3_Wayland
```

6.2. Wayland + Weston + QT

MYD-JX8MPQ 在发布时，只支持 wayland 和 Xwayland，不支持 fb 模式，wayland 采用 weston 图形桌面，可以在 weston 桌面上运行 QT 程序。

1）使用 pstree 查看进程服务

```
root@myd-jx8mp:~# pstree
systemd-+-agetty
        |-atd
        |-avahi-daemon---avahi-daemon
        |-check_upgrade.s---sleep
        |-crond
        |-dbus-daemon
        |-login---sh---pstree
        |-ninfod
        |-ntpd---{ntpd}
        |-ofonod
        |-proftpd
        |-psplash-systemd
        |-rdisc
        |-rpc.statd
        |-rpcbind
        |-syslogd
        |-systemd---(sd-pam)
        |-systemd-journal
        |-systemd-logind
        |-systemd-network
        |-systemd-resolve
        |-systemd-timesyn---{systemd-timesyn}
        |-systemd-udevd
        |-systemd-userdbd---3*[systemd-userwor]
        |-tee-supplicant
        `-weston-+-(sd-pam)
```

```
                |-weston-desktop-
                |-weston-keyboard
                `-{weston}
```

- weston-desktop　　weston 桌面
- weston-keyboard　　weston 键盘支持

2）查看系统环境

```
root@myd-jx8mp:/#  env
SHELL=/bin/sh
EDITOR=vi
PWD=/
LOGNAME=root
XDG_SESSION_TYPE=tty
HOME=/home/root
LANG=C
QT_QPA_PLATFORM=wayland
XDG_SESSION_CLASS=user
TERM=xterm
USER=root
SHLVL=1
XDG_SESSION_ID=c2
XDG_RUNTIME_DIR=/run/user/0
PS1=\u@\h:\w\$
HUSHLOGIN=FALSE
PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/0/bus
MAIL=/var/spool/mail/root
_=/usr/bin/env
OLDPWD=/lib/systemd/system
```

- XDG_RUNTIME_DIR：　运行环境
- QT_QPA_PLATFORM：　运行平台

3）运行 QT 例程

QT 的例子都在/user/share/example 目录运行一个例程：

```
root@myd-jx8mp:/#    /usr/share/examples/widgets/touch/pinchzoom/pinchzoo
m
```

可以看到屏幕上显示 QT 画面。

如果想开机运行 QT 程序，需要在脚本中增加 XDG_RUNTIME_DIR 和 QT_QPA_PLATFORM 才行，如下：

```
root@myd-jx8mp:/#    cat  qt_demo.sh
#!/bin/sh
export  XDG_RUNTIME_DIR=/run/user/0
export  QT_QPA_PLATFORM=wayland
/usr/share/examples/widgets/touch/pinchzoom/pinchzoom
```

7. 多媒体应用

7.1. Camera

本节采用系统自带 gstreamer，v4l2-utils，media-ctl 命令对米尔开发板做测试。主要测试 CSI 摄像头的预览、抓帧（拍照）。

1）查看 ov5640 摄像头分辨率

- **查看设备基本信息**

通过 J6 接入 CSI 摄像头，会生成字符设备/dev/video0：

```
root@myd-jx8mp:/# ls /dev/video0
/dev/video0
```

- **使用 v4l2-ctl 查看/dev/video0 所支持的分辨率**

```
root@myd-jx8mp:/# v4l2-ctl -D -d /dev/video0  --list-formats-ext
Driver Info:
        Driver name      : mx6s-csi
        Card type        : i.MX6S_CSI
        Bus info         : platform:32e20000.csi1_bridge
        Driver version   : 5.10.9
        Capabilities     : 0x84200001
                Video Capture
                Streaming
                Extended Pix Format
                Device Capabilities
        Device Caps      : 0x04200001
                Video Capture
                Streaming
                Extended Pix Format
Frame rate set to 30.000 fps
ioctl: VIDIOC_ENUM_FMT
        Type: Video Capture
```

```
        [0]: 'YUYV' (YUYV 4:2:2)
                Size: Discrete 640x480
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x480
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1920x1080
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 2592x1944
                        Interval: Discrete 0.067s (15.000 fps)
```

从以上 log 可清楚的看到对应的分辨率下所支持的帧率，如分辨率 320x240 所支持的帧率为 30fps。

2）摄像头窗口预览

- **使用 gstream 工具来进行预览**

使用 gstream 中的 gst-launch 命令进行摄像头的预览，终端执行以下命令：

```
root@myd-jx8mp:/# gst-launch-1.0 v4l2src ! "video/x-raw, width=640, Height=480, framerate=(fraction)30/1" ! queue ! glimagesink
Setting pipeline to PAUSED ...
Pipeline is live and does not need PREROLL ...
Got context from element 'sink': gst.gl.GLDisplay=context, gst.gl.GLDisplay=(GstGLDisplay)"\(GstGLDisplayWayland\)\ gldisplaywayland0";
Setting pipeline to PLAYING ...
New clock: GstSystemClock
[  809.430069] ov5640_mipi 1-003c: s_stream: 1
```

执行完后，屏幕会生成一个宽为 640，高为 480，帧速率为每秒 30 帧的预览窗口。

- **使用 gstream 工具来进行拍照**

使用 gstream 中的 gst-launch 命令进行摄像头的拍照，并保存为一个 jpeg 图片，终端执行以下命令：

```
root@myd-jx8mp:/# gst-launch-1.0  v4l2src device=/dev/video0 num-buffers=1 ! jpegenc ! filesink location=/home/root/test.jpg
Setting pipeline to PAUSED ...
Pipeline is live and does not need PREROLL ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
[ 2817.405874] ov5640_mipi 1-003c: s_stream: 1
Got EOS from element "pipeline0".[ 2819.911036] ov5640_mipi 1-003c: s_stream: 0
Execution ended after 0:00:02.897659250
Setting pipeline to PAUSED ...
Setting pipeline to READY ...
Setting pipeline to NULL ...
Freeing pipeline ...
```

由于当前摄像头只支持”YUYV”格式的像素，所以这里需要转码成 jpeg 格式图像。参数说明如下：

- Jpegenc：转换图像为 JPEG 格式

7.2. VPU

MYD-JX8MPQ 拥有一路 VPU，可以用来对视频进行硬件编/解码。

VPU 支持 解码格式如下：

- 1080P HEVC
- H.265
- VP9
- H.264
- VP8

支持编码格式如下：

- H.264
- H.265

1）解码测试

这里通过 gst 命令来查看文件音视频信息，播放到屏幕来说明 VPU 的解码功能。

- **提供视频源**

视频源，可以采用网络视频源，可以通过 tftp 方式拷贝，亦或者 nfs 挂载方式获取视频源，这里采用把视频源放置到 U 盘，U 盘接开发板方式。

```
root@myd-jx8mp:/# [55137.663104] usb 2-1.1: new high-speed USB device number 3 using ci_hdrc
[55137.790235] usb-storage 2-1.1:1.0: USB Mass Storage device detected
[55137.797566] scsi host0: usb-storage 2-1.1:1.0
[55138.826610] scsi 0:0:0:0: Direct-Access     Generic  Flash Disk       8.07 PQ: 0 ANSI: 4
[55138.839316] sd 0:0:0:0: [sda] 31129600 512-byte logical blocks: (15.9 GB/14.8 GiB)
[55138.848168] sd 0:0:0:0: [sda] Write Protect is off
[55138.853917] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[55138.894108]  sda: sda1
[55138.900910] sd 0:0:0:0: [sda] Attached SCSI removable disk

root@myd-jx8mp:/#
root@myd-jx8mp:/#
```

```
root@myd-jx8mp:/# cd /run/media/sda1/
root@myd-jx8mp:/run/media/sda1# ls -l
total 26568
drwxrwx--- 2 root disk       8192 Jul 30 10:56 System Volume Information
-rwxrwx--- 1 root disk 27191347 Jul 17  2018 qq.mp4
root@myd-jx8mp:/run/media/sda1#
```

- **查看视频格式**

这里采用 gst-discoverer-1.0 来查看 qq.mp4 的音视频格式。

```
root@myd-jx8mp:/run/media/sda1# gst-discoverer-1.0 qq.mp4
Analyzing file:///run/media/sda1/qq.mp4

====== AIUR: 4.5.4 build on Sep  5 2020 04:45:35. ======
Mime:
    video/x-h264, parsed=(boolean)true, alignment=(string)au, stream-format=
(string)avc, width=(int)540, height=(int)960, framerate=(fraction)30/1, codec_dat
a=(buffer)0164001fffe100102764001fac56c0881e7be6a02020204001000428ee3cb
0fdf8f800
------------------------
--snip--
Done discovering file:///run/media/sda1/qq.mp4

Topology:
  container: Quicktime
    audio: MPEG-4 AAC
    video: H.264 (High Profile)

--snip--
```

- audio: MPEG-4 AAC　　音频格式 AAC
- video: H.264　　视频格式 H.264
- container:Quicktime　　使用过滤器容器 quciktime

- **播放视频**

gst 系列有通用命令 gst-play-1.0 可以播放音视频，但是操作空间较小，另一种就是 gst-launcher-1.0，功能灵活。

gst-play-1.0 播放视频：

```
root@myd-jx8mp:/run/media/sda1# gst-play-1.0 qq.mp4
```

gst-luancher-1.0 播放视频：

```
gst-launch-1.0 filesrc location=qq.mp4 typefind=true ! video/quicktime ! aiurd
emux ! queue ! vpudec ! autovideosink
```

- filesrc： 指定使用文件，location 指定文件路径
- typefind： 表示文件类型已知
- video/quicktime： 采用 quicktime 容器
- aiurdmux： 音视频分析器 demux
- queue： 使用队列
- vpudec： 采用解码器 VPUDEC
- Autovideosink：默认选择输出源，可选 waylandsink 和 glimagesink

2）编码测试

MYD-JX8MPQ 支持 vpuenc_h264 和 vpuenc_hevc 2 种编码模块。

根据上面例子已经知道 qq.mp4 是 H.264 格式，VPU 对于硬件编码只支持 H.264 和 VP8，那么本章节举例说明如何将 H.264 转换成 hevc 格式。

```
gst-launch-1.0 -e filesrc location=qq.mp4 typefind=true ! video/quicktime ! ai
urdemux ! vpudec ! vpuenc_hevc ! h265parse ! filesink location=test.h265
```

- -e：异常中断加结束符
- vpuenc_hevc: 硬件编码器
- matroskamux: 音视频混合器
- filesink:按文件输出，location 代表保存路径

```
root@myd-jx8mp:~# gst-discoverer-1.0 test.h265
Analyzing file:///home/root/test.h265
====== VPUDEC: 4.5.4 build on Apr 2 2021 10:49:57. ======
        wrapper: 3.0.0 (VPUWRAPPER_ARM64_LINUX Build on Apr 2 2021 09:
54:48)
        vpulib: 1.1.1
        firmware: 1.1.1.0
Done discovering file:///home/root/test.h265
```

```
Topology:
  video: H.265 (Main Profile)

Properties:
  Duration: 0:01:00.257000000
  Seekable: yes
  Live: no
  Tags:
       video codec: H.265 (Main Profile)
```

然后使用 gst-play-1.0 test.h265 进行播放。

8. 系统工具

8.1. 压缩解压工具

本节主要测试系统的解压缩工具。压缩是可以把多个文件压缩成一个压缩包可以把多个文件压缩成一个压缩包，方便进行文件的传输。而解压可以把经过压缩的压缩文件还原成原始大小方便使用。本节将在文件系统以 tar、gzip、gunzip 等工具为例进行说明。

1）tar 工具

- **语法格式**

现在我们在 Linux 中使用的 tar 工具，它不仅可以对文件打包，还可以对其进行压缩，查看，添加以及解压等一系列操作。这里是将打包操作。输入以下命令查看 tar 语法格式：

```
root@myd-jx8mp:~# tar --help
Usage: tar [OPTION...] [FILE]...
GNU 'tar' saves many files together into a single tape or disk archive, and ca
n
restore individual files from the archive.

Examples:
  tar -cf archive.tar foo bar  # Create archive.tar from files foo and bar.
  tar -tvf archive.tar         # List all files in archive.tar verbosely.
  tar -xf archive.tar          # Extract all files from archive.tar.
```

详细参数说明如下：

- -c ： 建立一个压缩文件的参数指令(create 的意思)；
- -x ： 解开一个压缩文件的参数指令！
- -t ： 查看 tarfile 里面的文件！特别注意，在参数的下达中， c/x/t 仅能存在一个！不可同一时候存在！由于不可能同一时候压缩与解压缩。
- -z ： 是否同一时候具有 gzip 的属性？亦即是否须要用 gzip 压缩？
- -j ： 是否同一时候具有 bzip2 的属性？亦即是否须要用 bzip2 压缩？
- -v ： 压缩的过程中显示文件！这个经常使用，但不建议用在背景运行过程！

- -f ：使用档名，请留意，在 f 之后要马上接档名，不要再加参数！比如使用『tar -zcvfP tfile sfile』就是错误的写法，要写成『 tar -zcvPf tfile sfile』才对。
- -p ：使用原文件的原来属性（属性不会根据使用者而变）
- -P ：能够使用绝对路径来压缩！
- -N ：比后面接的日期(yyyy/mm/dd)还要新的才会被打包进新建的文件里！
- --exclude FILE：在压缩的过程中，不要将 FILE 打包！

- **使用 tar 压缩**

新建 test.txt 文件，并输入以下命令将文件打包成.gz 格式：

```
root@myd-jx8mp:~# tar -czf test.tar.gz test.txt
root@myd-jx8mp:~# ls
OpenAMP_TTY_echo.elf  rs485_write  test.tar.gz  uart_test
rs485_read            test.c       test.txt     wifi.conf
```

所以上面加 z 参数，则以 .tar.gz 或 .tgz 来代表 gzip 压缩过的 tar file 。

- **使用 tar 解压**

把打包成 tar.gz 格式文件解压

```
root@myd-jx8mp:~# tar -xvf test.tar.gz
test.txt
root@myd-jx8mp:~# ls
OpenAMP_TTY_echo.elf  rs485_write  test.tar.gz  uart_test
rs485_read            test.c       test.txt     wifi.conf
```

2）gzip 压缩工具

- **语法格式**

gzip 是在 Linux 系统中经常使用的一个对文件进行压缩和解压缩的命令，既方便又好用。在开发板终端输入以下命令查看 gzip 语法：

```
root@myd-jx8mp:~# gzip --help
Usage: gzip [OPTION]... [FILE]...
```

- **用 gzip 把文件压缩**

```
root@myd-jx8mp:~# gzip test.txt
root@myd-jx8mp:~# ls
OpenAMP_TTY_echo.elf  rs485_write  test.tar.gz  uart_test
```

```
rs485_read                test.c          test.txt.gz    wifi.conf
```

- **用 gunzip 把文件解压**

```
root@myd-jx8mp:~# gunzip test.txt.gz
root@myd-jx8mp:~# ls
OpenAMP_TTY_echo.elf   rs485_write   test.tar.gz   uart_test
rs485_read             test.c        test.txt      wifi.conf
```

8.2. 文件系统工具

主要测试系统的文件系统工具，本节将介绍几种常见的文件系统管理工具。系统自带的文件系统工具 mount、mkfs、fsck、dumpe2fs。

1）mount 挂载工具

mount 是 Linux 下的一个命令，它可以将分区挂接到 Linux 的一个文件夹下，从而将分区和该目录联系起来，因此我们只要访问这个文件夹，就相当于访问该分区了，其应用语法格式如下：

```
root@myd-jx8mp:~# mount -h

Usage:
 mount [-lhV]
 mount -a [options]
 mount [options] [--source] <source> | [--target] <directory>
 mount [options] <source> <directory>
 mount <operation> <mountpoint> [<target>]

Mount a filesystem.
```

- **挂载 U 盘**

```
root@myd-jx8mp:~# mount /dev/sda1 /mnt/
```

2）mkfs 格式工具

硬盘分区后，下一步的工作是 Linux 文件系统的建立。类似于 Windows 下的格式化硬盘。在硬盘分区上建立文件系统会冲掉分区上的数据，而且不可恢复，因此在建立文件系统之前要确认分区上的数据不再使用。建立文件系统的命令是 mkfs,应用语法格式如下：

```
root@myd-jx8mp:~# mkfs -h

Usage:
 mkfs [options] [-t <type>] [fs-options] <device> [<size>]

Make a Linux filesystem.
```

```
Options:
 -t, --type=<type>   filesystem type; when unspecified, ext2 is used
      fs-options     parameters for the real filesystem builder
      <device>       path to the device to be used
      <size>         number of blocks to be used on the device
 -V, --verbose       explain what is being done;
                       specifying -V more than once will cause a dry-run
 -h, --help          display this help
 -V, --version       display version

For more details see mkfs(8).
```

- **格式化 U 盘**

```
root@myd-jx8mp:~# umount /mnt
root@myd-jx8mp:~# umount /run/media/sda1/

root@myd-jx8mp:~# mkfs -t ext3 -V -c /dev/sda1
mkfs from util-linux 2.34
mkfs.ext3 -c /dev/sda1
mke2fs 1.45.3 (14-Jul-2019)
/dev/sda1 contains a vfat file system
Proceed anyway? (y,N) y
Creating filesystem with 3890688 4k blocks and 972944 inodes
Filesystem UUID: 97810d2b-76aa-44a4-9409-2c70de71eca0
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 265
4208

Checking for bad blocks (read-only test):
[ 6873.703223] audit: type=1006 audit(1599269401.268:4): pid=947 uid=0 old-
auid=4294967295 auid=0 tty=(none) old-ses=4294967295 ses=3 res=1
done
Allocating group tables: done
```

```
Writing inode tables:
done
Creating journal (16384 blocks):
done
Writing superblocks and filesystem accounting information: done
```

3）fsck 文件修复工具

fsck 命令主要用于检查文件系统的正确性，当文件系统发生错误时，可用 fsck 指令尝试加以修复。并对 Linux 磁盘进行修复。例如：

```
root@myd-jx8mp:~# fsck -a /dev/sda1
fsck from util-linux 2.34
/dev/sda1: clean, 11/972944 files, 86964/3890688 blocks
```

4）dumpe2fs

打印特定设备上现存的文件系统的超级块(super block)和块群(blocks group)的信息。开发板输入以下命令查看应用语法：

```
root@myd-jx8mp:~# dumpe2fs -h
dumpe2fs 1.45.3 (14-Jul-2019)
Usage: dumpe2fs [-bfghimxV] [-o superblock=<num>] [-o blocksize=<num>]
device
```

查看文件系统格式化后的详细属性，例如，输入命令查看某个磁盘的详细信息：

```
root@myd-jx8mp:~# dumpe2fs -h
dumpe2fs 1.45.3 (14-Jul-2019)
Usage: dumpe2fs [-bfghimxV] [-o superblock=<num>] [-o blocksize=<num>]
device
root@myd-jx8mp:~#
root@myd-jx8mp:~# dumpe2fs -h^C
root@myd-jx8mp ~# dumpe2fs /dev/sda1
dumpe2fs 1.45.3 (14-Jul-2019)
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          97810d2b-76aa-44a4-9409-2c70de71eca0
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
```

```
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype sp
arse_super large_file
Filesystem flags:         unsigned_directory_hash
```

查看某磁盘的 inode 数量，inode 也会消耗硬盘空间，所以硬盘格式化的时候，操作系统自动将硬盘分成两个区域。一个是数据区，存放文件数据；另一个是 inode 区（inode table），存放 inode 所包含的信息。

```
root@myd-jx8mp:~# dumpe2fs /dev/sda1 | grep -i "inode size"
dumpe2fs 1.45.3 (14-Jul-2019)
Inode size:               256
```

查看某磁盘的 block 数量，操作系统读取硬盘的时候，不会一个个扇区地读取，这样效率太低，而是一次性连续读取多个扇区，即一次性读取一个"块"（block）。这种由多个扇区组成的"块"，是文件存取的最小单位。

```
root@myd-jx8mp:~# dumpe2fs /dev/sda1 | grep -i "block size"
dumpe2fs 1.45.3 (14-Jul-2019)
Block size:               4096
```

8.3. 磁盘管理工具

主要测试系统的磁盘管理工具，本节将介绍几种常见的磁盘管理工具。系统自带的磁盘管理工具 fdisk、dd、mkfs、du、df、cfdisk、fsck 。通过这些命令可以监控平时的磁盘使用情况。

1）fdisk 磁盘分区工具

fdisk 磁盘分区工具在 DOS、Windows 和 Linux 中都有相应的应用程序。在 Linux 系统中，fdisk 是基于菜单的命令。用 fdisk 对硬盘进行分区，可以在 fdisk 命令后面直接加上要分区的硬盘作为参数，其应用语法格式如下：

```
root@myd-jx8mp:~# fdisk -h
Usage:
 fdisk [options] <disk>         change partition table
 fdisk [options] -l [<disk>] list partition table(s)
```

对 eMMC 进行分区：

```
root@myd-jx8mp:~# fdisk /dev/mmcblk2p2

Welcome to fdisk (util-linux 2.34).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

The old ext4 signature will be removed by a write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x43403b02.

Command (m for help):
```

2）dd 拷贝命令

dd 命令用于将指定的输入文件拷贝到指定的输出文件上。并且在复制过程中可以进行格式转换。dd 命令与 cp 命令的区别在于：dd 命令可以在没有创建文件系统的软盘上进行，拷贝到软盘的数据实际上是镜像文件。类似于 DOS 中的 diskcopy 命令的作用。

dd 命令的格式为： dd [<if=输入文件名/设备名>] [<of=输出文件名/设备名>] [bs=块字节大小] [count＝块数]。

创建一个大小为 2M 的文件。

```
root@myd-jx8mp:~# time dd if=/dev/zero of=ffmpeg1 bs=2M count=1 conv=fsync
1+0 records in
1+0 records out
2097152 bytes (2.1 MB, 2.0 MiB) copied, 0.119542 s, 17.5 MB/s

real    0m0.129s
user    0m0.000s
sys     0m0.019s
```

3) du 磁盘用量统计工具

du 命令用于显示磁盘空间的使用情况。该命令逐级显示指定目录的每一级子目录占用文件系统数据块的情况。du 一般使用语法如下：

```
root@myd-jx8mp:~# du --help
Usage: du [OPTION]... [FILE]...
  or:  du [OPTION]... --files0-from=F
Summarize disk usage of the set of FILEs, recursively for directories.
```

部分参数说明：

- -a:显示所有目录或文件的大小
- -h:以 K,M,G 为单位，提高信息可读性
- -k:以 KB 为单位输出
- -m:以 MB 为单位输出

统计 dd 命令生成的文件大小：

```
root@myd-jx8mp:~# du ffmpeg1
2048    ffmpeg1
root@myd-jx8mp:~# du -h ffmpeg1
2.0M    ffmpeg1
```

4) df 磁盘统计工具

用于显示目前在 Linux 系统上的文件系统的磁盘使用情况统计，一般用法如下：

```
root@myd-jx8mp:~# df --help
```

```
Usage: df [OPTION]... [FILE]...
Show information about the file system on which each FILE resides,
or all file systems by default.
```

部分参数说明：

- -h：可以根据所使用大小使用适当的单位显示
- -i:查看分区下 inode 的数量和 inode 的使用情况
- -T:打印出文件系统类型

查看分区下 inode 的数量和 inode 的使用情况，使用如下命令：

```
root@myd-jx8mp:~# df -i
Filesystem      Inodes IUsed   IFree IUse% Mounted on
/dev/root       497488 48320  449168    10% /
devtmpfs        166144   489  165655     1% /dev
tmpfs           248792     1  248791     1% /dev/shm
tmpfs           248792   706  248086     1% /run
tmpfs           248792    13  248779     1% /sys/fs/cgroup
tmpfs           248792    16  248776     1% /tmp
tmpfs           248792    28  248764     1% /var/volatile
tmpfs           248792     9  248783     1% /run/user/0
/dev/mmcblk1p1       0     0       0      - /run/media/mmcblk1p1
/dev/mmcblk2p1       0     0       0      - /run/media/mmcblk2p1
/dev/mmcblk2p2  546176 41910  504266     8% /run/media/mmcblk2p2
```

inode 是我们在格式化的时候系统给我们划分好的 ,inode 与磁盘分区大小有关。当我们的 inode 使用已经达到百分百时，即使我们的磁盘空间还是有剩余，我们也是写不了数据到磁盘的。其他应用例子请参考 2.5 节。

8.4. 进程管理工具

进程也是操作系统中的一个重要概念，它是一个程序的一次执行过程，程序是进程的一种静态描述，系统中运行的每一个程序都是在它的进程中运行的。Linux 系统中所有的进程是相互联系的，除了初始化进程外，所有进程都有一个父进程。新的进程不是被创建，而是被复制，或是从以前的进程复制而来。Linux 中所有的进程都是由一个进程号为 1 的 init 进程衍生而来的。Linux 系统包括 3 种不同类型的进程，每种进程都有自己的特点和属性：

- 交互进程：由一个 Shell 启动的进程，既可以在前台运行，又可以在后台运行。
- 批处理进程：这种进程和终端没有联系，是一个进程序列。这种进程被提交到等待队列顺序执行的进程。
- 监控进程(守护进程)：守护进程总是活跃的，一般是在后台运行，守护进程一般是由系统在开始时通过脚本自动激活启动或 root 启动。

对于 linux 系统来说，进程的管理是重要的一环，对于进程的管理通常是通过进程管理工具实现的，Linux 系统中比较常用的进程管理命令有以下几种：ps ，top， vmstat kill。

1）ps 显示当前进程工具

- **语法格式**

显示当前系统进程的运行情况，一般语法如下：

```
root@myd-jx8mp:~# ps --help

Usage:
 ps [options]

 Try 'ps --help <simple|list|output|threads|misc|all>'
  or 'ps --help <s|l|o|t|m|a>'
 for additional help text.

For more details see ps(1).
```

部分参数组合说明：

- -u：以用户为中心组织进程状态信息显示；
- -a：与终端无关的进程；

- -x：与终端有关的进程；　（线程，就是轻量级进程；）
- 通常上面命令组合使用：aux。
- --e：显示所有进程；相当于 ax；
- -f：显示完整格式程序信息；
- 通常以上命令组合使用：ef
- -H：以进程层级显示进程数的
- -F：显示更多的程序信息

通常组合使用命令：eHF。

- **显示所有进程的信息情况**

```
root@myd-jx8mp:~# ps -aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.3  92272  7176 ?        Ss   Sep04   0:03 /sbin/init
root           2  0.0  0.0      0     0 ?        S    Sep04   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   Sep04   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   Sep04   0:00 [rcu_par_gp]
root           8  0.0  0.0      0     0 ?        I<   Sep04   0:00 [mm_percpu_wq]
root           9  0.0  0.0      0     0 ?        S    Sep04   0:00 [ksoftirqd/0]
root          10  0.0  0.0      0     0 ?        I    Sep04   0:00 [rcu_preempt]
root          11  0.0  0.0      0     0 ?        S    Sep04   0:00 [migration/0]
root          12  0.0  0.0      0     0 ?        S    Sep04   0:00 [cpuhp/0]
root          13  0.0  0.0      0     0 ?        S    Sep04   0:00 [cpuhp/1]
root          14  0.0  0.0      0     0 ?        S    Sep04   0:00 [migration/1]
```

对其上面一些任务栏进行简单解释说明：

- VSZ：vittual memory size 虚拟内存集
- RSS：resident size，常驻内存集
- STAT：进程状态有以下几个

- R：runing
- S：interuptable sleeping 可中断睡眠
- D：uninteruptable sleeping 不可中断睡眠
- T：stopped
- Z: zombie 僵尸进程
- +:前台进程
- N:低优先级进程
- l:多线程进程

2）top 显示 linux 进程

- **语法格式**

top 命令将相当多的系统整体性能信息放在一个屏幕上。显示内容还能以交互的方式进行改变。动态的持续监控进程的运行状态，top 语法一般如下：

```
root@myd-jx8mp:~#  top -help
  procps-ng 3.3.15
Usage:
  top -hv | -bcEHiOSs1 -d secs -n max -u|U user -p pid(s) -o field -w [cols]
```

- **动态查看系统进程**

```
root@myd-jx8mp:~# top
top - 02:03:28 up  2:28,  1 user,  load average: 0.01, 0.35, 0.99
Tasks: 135 total,   1 running, 134 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.4 us,  1.4 sy,  0.0 ni, 97.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   1943.7 total,   1487.8 free,    244.2 used,    211.6 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.   1597.4 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME
+ COMMAND

    982 root      20   0    3444   2024   1616 R   5.9   0.1   0:00.02 top
```

```
      1 root      20   0   92272   7176   5064 S   0.0   0.4   0:03.49 systemd

      2 root      20   0       0      0      0 S   0.0   0.0   0:00.03 kthreadd

      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
```

3）vmstat 虚拟内存的统计工具

- **语法格式**

这个命令可查看内存空间的使用状态，能获取整个系统的性能粗略信息，vmstat 运行于两种模式：采样模式和平均模式。如果不指定参数，则 vmstat 统计运行于平均模式下，vmstat 显示从系统启动以来所有统计数据的均值。常用语法以及参数如下：

```
root@myd-jx8mp:~# vmstat  -h

Usage:
 vmstat [options] [delay [count]]

Options:
 -a, --active           active/inactive memory
 -f, --forks            number of forks since boot
 -m, --slabs            slabinfo
 -n, --one-header       do not redisplay header
 -s, --stats            event counter statistics
 -d, --disk             disk statistics
 -D, --disk-sum         summarize disk statistics
 -p, --partition <dev>  partition specific statistics
 -S, --unit <char>      define display unit
 -w, --wide             wide output
 -t, --timestamp        show timestamp

 -h, --help     display this help and exit
```

```
 -V, --version   output version information and exit

For more details see vmstat(8).
```

- **vmstat 运行于平均模式**

显示从系统启动以来所有统计数据的均值：

```
root@myd-jx8mp:~# vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b    swpd    free    buff  cache   si   so    bi    bo   in   cs us sy id wa
 st
 0  0       0 1524140   12552 204240    0    0   441    13   80  103  0  0
 95  5   0
```

对其上面一些任务栏进行简单解释说明

- r：当前可运行的进程数。这些进程没有等待 I/O,而是已经准备号运行。理想状态，可运行进程数与可用 cpu 数量相等
- b：等待 I/O 完成的被阻塞进程数
- forks：创建新进程的次数
- in：系统发生中断的次数
- cs：系统发生上下文切换的次数
- us：用户进程消耗的总 CPU 时间百分比
- sy：系统代码消耗的总 CPU 时间的百分比，其中包括消耗在 system、irq 和 softirq 状态的时间
- wa：等待 I/O 消耗的总 CPU 的百分比
- id：系统空闲消耗的总 CPU 时间的百分比

- **统计系统各种数据详细信息**

```
root@myd-jx8mp:~# vmstat  -s
        1990336 K total memory
         249516 K used memory
         103972 K active memory
         132832 K inactive memory
        1524012 K free memory
          12568 K buffer memory
         204240 K swap cache
```

```
            0 K total swap
            0 K used swap
            0 K free swap
        13051 non-nice user cpu ticks
            1 nice user cpu ticks
         4360 system cpu ticks
      3403234 idle cpu ticks
       166345 IO-wait cpu ticks
         1782 IRQ cpu ticks
         1637 softirq cpu ticks
            0 stolen cpu ticks
     15733523 pages paged in
       472350 pages paged out
            0 pages swapped in
            0 pages swapped out
      2847443 interrupts
      3669954 CPU context switches
   1599262527 boot time
          990 forks
```

对其上面一些任务栏进行简单解释说明

- total memory：系统总内存
- used memory：已使用内存
- CPU ticks：显示的是自系统启动的 CPU 时间，这里的“tick”是一个时间单位
- forks：大体上表示的是从系统启动开始已经创建的新进程的数量

vmstat 提供了关于 linux 系统性能的众多信息。在调查系统问题时，它是核心工具之一。

4) kill 进程终止工具

- **语法格式**

发送指定的信号到相应进程。不指定型号将发送 SIGTERM（15）终止指定进程。如果任无法终止该程序可用“-KILL” 参数，其发送的信号为 SIGKILL(9) ，将强制结束进程，使用 ps 命令或者 jobs 命令可以查看进程号。root 用户将影响用户的进程，非 root 用户只能影响自己的进程。kill 命令一般语法如下：

```
kill [ -s signal | -p ] [ -a ] pid ...
kill -l [ signal ]
```

部分参数组合说明：

- -s：指定发送的信号
- -p：模拟发送信号
- -l：指定信号的名称列表
- pid：要中止进程的 ID 号
- Signal：表示信号

首先使用 ps -ef 与管道命令确定要杀死进程的 PID，

```
root@myd-jx8mp:~# ps -ef | grep wpa_supplicant
root         564         1   0 Sep04 ?          00:00:00 /usr/sbin/wpa_supplicant
 -u
root         989       850   0 02:05 ttymxc1    00:00:00 grep wpa_supplicant
```

然后输入以下命令终止进程：

```
root@myd-jx8mp ~# kill 564
```

killall 命令终止同一进程组内的所有进程，允许指定要终止的进程的名称而非 PID 进程号：

```
root@myd-jx8mp:~# killall wpa_supplicant
root@myd-jx8mp:~#
```

9. 开发支持

本章主要介绍针对当前 SDK 进行二次开发的一些基本信息，当前 SDK 提供两种配置的参考镜像，一种是 myir-image-core，主要针对无 GUI 的应用；另外一种是 myir-image-full，在 myir-image-core 的基础上增加一些需要 GUI 的应用。关于这两种镜像的信息请参考《MYD-JX8MPQ SDK 发布说明》。

9.1. 开发语言

1) SHELL

Shell 是一个用 C 语言编写的程序，它是用户使用 Linux 的桥梁。Shell 既是一种命令语言，又是一种程序设计语言。常见的 Linux 的 Shell 种类众多，常见的有：

- Bourne Shell（/usr/bin/sh 或/bin/sh）
- Bourne Again Shell（/bin/bash）
- C Shell（/usr/bin/csh）
- K Shell（/usr/bin/ksh）
- Shell for Root（/sbin/sh）

MYD-JX8MPQ 支持 bourne shell 和 Bourne Again Shell 2 种：

```
root@myd-jx8mp:~# echo "echo 'myir test'" > shell_demo.sh
root@myd-jx8mp:~# sh shell_demo.sh
myir test
root@myd-jx8mp:~# bash shell_demo.sh
myir test
```

2) C/C++

C/C++是 Linux 平台下进行底层应用开发最为常用的编程语言，也是仅次于汇编的最为高效的语言。使用 C/C++进行开发通常采用的是交叉开发的方式，即在开发主机端进行开发，编译生成目标机器上运行的二进制执行文件，然后部署到目标机器上运行。

采用这种方式，首先需要安装基于 Yocto 构建的 SDK，安装步骤请参《MYD-JX8MPQ Linux 软件开发指南》

本节通过编写一个简单的 Hello World 实例来演示应用程序的开发，以下为在开发主机端编写的演示程序 hello.c：

```
1 #include<stdio.h>
2 int main(int argc,char *argv[])
3 {
4       printf("hello world!\n");
5       return 0;
6 }
```

接着编译应用程序，这里用$CC 编译，因为编译的时候需要对应的头文件和链接，$CC 包含有对应的系统库和配置信息，如果直接用 aarch64-poky-linux-gcc 来编译会出现找不到头文件的情况，这个时候可以加入参数-v 来查看详细的链接过程。

```
duxy@myir-server1:~/test$ source ~/opt_meta_10_9/environment-setup-aarch64-poky-linux
duxy@myir-server1:~/test$ cat main.c
#include <stdio.h>


int main(int argc, char *argv[])
{
        print("myir test!\n");
        return 0;
}
duxy@myir-server1:~/test$ $CC main.c -o main
duxy@myir-server1:~/test$ file main
main: ELF 64-bit LSB shared object, ARM aarch64, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-aarch64.so.1, BuildID[sha1]=b24bfdeaefb3ca10ebf845be5dbc95bbd2aa5c1d, for GNU/Linux 3.14.0, not stripped
```

然后通过 scp 命令把生成的执行文件拷贝到目标机器上执行，结果如下：

```
root@myd-jx8mp:~# ./main
myir test!
```

更复杂的示例和开发方式请参考《MYD-JX8MPQ Linux 软件开发指南》应用移植部分的说明。

3) Python

Python 是一种解释型、面向对象、动态数据类型的高级程序设计语言。Python 由 Guido van Rossum 于 1989 年底发明，第一个公开发行版发行于 1991 年。像 Perl 语言一样, Python 源代码同样遵循 GPL(GNU General Public License) 协议。本节主要测试 python 的使用，从 python 命令行和脚本两个方面来说明。

- **查看系统支持的 phthon 版本**

```
root@myd-jx8mp:~# python
python                  python2                 python2.7               python3
                python3.8               python3.8-config-lib
python-config           python2-config          python2.7-config        python3-c
onfig           python3.8-config
```

- **python 命令行测试**

启动 python，并在 python 提示符中输入以下文本信息，然后按 Enter 键查看运行效果：

```
root@myd-jx8mp:~# python3
Python 3.8.5 (default, Jul 20 2020, 13:26:22)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("myir test")
```

在 Python 3.8.5 版本中,以上实例输出结果如下：

```
myir test
>>>
```

退出 Python 命令行，执行 exit()即退出 Python：

```
>>> exit()
root@myd-jx8mp:~#
```

3) 编写脚本测试 Python

编写一个简单的 Python 脚本程序，所有 Python 文件将以 .py 为扩展名：

```
root@myd-jx8mp:~# vi test.py
root@myd-jx8mp:~# cat test.py
#!/usr/bin/env python3
print("myir test")
```

执行脚本文件，解释器在/usr/bin/env 中以 python3 运行目录中，使用以下命令执行脚本：

```
root@myd-jx8mp:~# chmod a+x test.py
root@myd-jx8mp:~# ./test.py
myir test
```

通过脚本参数调用 Python3 解释器开始执行脚本，直到脚本执行完毕。当脚本执行完成后，解释器不再有效。当前系统支持 pip 和 pip3，下载包时会遇到认证不过情况，此时按如下操作：

```
pip install numpy -i http://pypi.douban.com/simple/ --trusted-host pypi.douban.com
```

10. 参考资料

- **Linux kernel 开源社区**
  https://www.kernel.org/
- **NXP 开发社区**
  https://community.nxp.com/
- **i.MX 8 系列处理器介绍**
  https://www.nxp.com/products/processors-and-microcontrollers/arm-processors/i-mx-applications-processors/i-mx-8-processors:IMX8-SERIES
- **MCU SDK 资源下载**
  https://mcuxpresso.nxp.com/en/welcome

附录一 联系我们

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

武汉分公司

负责区域：武汉

电话：027-59621648

传真：Na

邮编：430000

地址：湖北省武汉市洪山区关南园一路 20 号当代科技园 7 号楼 1903 号

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：上海市浦东新区金吉路 778 号浦发江程广场 1 号楼 805 室

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：北京市大兴区荣华中路 8 号院力宝广场 10 号楼 901 室

销售联系方式

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

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（自我司收到物品之日起，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。