

# MYD-YM62X

## Linux 软件评估指南



文件状态： [ ] 草稿 [✓] 正式发布	文件标识：	MYIR-MYD-YM62X-SW-EG-ZH-L6.1.46
	当前版本：	V1.0[文档]
	作 者：	Beste Wang
	创建日期：	2023-09-15
	最近更新：	2023-09-25

# 版本历史

版本	作者	参与者	日期	备注
V1.0[文档]	Beste Wang		20230915	初始版本, 适用于 MYD-YM62X



# 目 录

版 本 历 史.....	- 0 -
目 录.....	- 1 -
1. 概述.....	- 3 -
1.1. 硬件资源.....	- 3 -
1.2. 软件资源.....	- 4 -
1.3. 文档资源.....	- 4 -
1.4. 环境准备.....	- 4 -
2. 核心资源.....	- 5 -
2.1. CPU.....	- 5 -
2.2. 协处理器 M4F.....	- 10 -
2.3. Graphics.....	- 10 -
2.4. Memory.....	- 10 -
2.5. eMMC.....	- 15 -
2.6. RTC.....	- 20 -
2.7. Watchdog.....	- 21 -
3. 外围接口.....	- 25 -
3.1. GPIO.....	- 25 -
3.2. LED.....	- 27 -
3.3. Key(按键).....	- 28 -
3.4. RS485.....	- 29 -
3.5. CAN.....	- 30 -
3.6. USB.....	- 32 -
3.7. typeC 接口.....	- 34 -
3.8. Micro SD 卡.....	- 35 -
3.9. Display.....	- 38 -
3.10. Touch Panel.....	- 41 -



4. 网络与通信 .....	- 45 -
4.1. 网络设备 .....	- 45 -
4.2. 网络应用 .....	- 67 -
5. 图形系统 .....	- 83 -
5.1. GPU .....	- 84 -
5.2. Wayland+Weston+QT .....	- 84 -
6. 多媒体应用 .....	- 87 -
6.1. Camera .....	- 87 -
6.2. Audio .....	- 88 -
7. 系统工具 .....	- 90 -
7.1. 压缩解压工具 .....	- 90 -
7.2. 文件系统工具 .....	- 93 -
7.3. 磁盘管理工具 .....	- 95 -
7.4. 进程管理工具 .....	- 98 -
8. 开发支持 .....	- 102 -
8.1. 开发语言 .....	- 102 -
9. 参考资料 .....	- 107 -
附录一 联系我们 .....	- 108 -
附录二 售后服务与技术支持 .....	- 110 -



# 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

本文档使用于米尔电子的 MYD-YM62X 系列板卡，该板卡是米尔电子的嵌入式开发平台基于 TI 公司的高性能嵌入式 ARM 处理器开发的，其中该系列使用的核心芯片包括 Y M6254、YM6252 和 YM6231 三款型号。这一系列提供了不同型号，以满足不同性能需求。每个型号都配备了 Cortex-M4F 协处理器，提供强大性能和灵活性。

下表概述了 MYD-YM62X 系列板卡的三个型号及其主要特征：

表 1-1. 板卡型号介绍

核心芯片型号	处理器核心	M4F 协处理器	核心板	底板
AM6231	单核 ARMCortex-A53	是	MYC-YM6231	MYB-Y62X
AM6252	双核 ARMCortex-A53	是	MYC-YM6252	MYB-Y62X
AM6254	四核 ARMCortex-A53	是	MYC-YM6254	MYB-Y62X

关于硬件部分的详细配置参数请查看《MYD-YM62X 产品手册》。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 1-2. 选配模块

配件	接口方式	说明及链接
液晶屏	LVDS 接口	MY-LVDS070C <a href="https://www.myir.cn/shows/106/3.html">https://www.myir.cn/shows/106/3.html</a>
摄像头模块	CSI 接口	MY-CAM003M <a href="https://www.myir.cn/shows/113/11.html">https://www.myir.cn/shows/113/11.html</a>
4G 模块	PCIE 接口	EM05-CE <a href="https://www.quectel.com/product/lte-em05-series">https://www.quectel.com/product/lte-em05-series</a>
5G 模块	PCIE 接口	RM500Q <a href="https://www.quectel.com/product/5g-rm500q-ae">https://www.quectel.com/product/5g-rm500q-ae</a>



## 1.2. 软件资源

MYD-YM62X 系列开发板的 BSP 是基于 TI 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-YM62X SDK** 发布说明》。

开发板在出厂时已经烧录了镜像，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 中包含了除发布说明之外的入门指南，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-YM62X SDK2.0.0** 发布说明》表 2-4 中的说明。

## 1.4. 环境准备

在开始评估开发板软件之前，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-YM62X** 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, eMMC, RTC 等核心资源的参数进行读取与测试。

### 2.1. CPU

MYD-YM62X 系列开发板的核心芯片有 AM6231、AM6252、AM6254 等。AM62X 系列芯片采用多核异构芯片架构，内置高性能的 Cortex-A53 CPU 内核及独立安全控制从核心 Cortex M4F 的内核；支持安全启动且配套安全 OS，具有用户可编程 HSM 内核的专用安全控制器以及用于隔离式处理的专用安全 DMA 和 IPC 子系统；每个 A53 内核包含具有 SECDED ECC 功能的 32KB L1 DCache 和具有奇偶校验保护的 32KB L1 ICache；集成了 Ethernet/UART/SPI/IIC/MCASP/JTAG/MIPI CSI/GPMC/LVDS/等丰富的外设接口。

以下将通过一些命令测试 CPU 的状态或功能，将以 MYD-YM6254 为例。

#### 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。

```
root@myd-am62x:~# cat /proc/cpuinfo
processor : 0
BogoMIPS : 400.00
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 : 400.00
```



```
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 : 400.00
```

```
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 : 400.00
```

```
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-am62x:~# top
```

```
top - 03:41:26 up 3 min, 2 users, load average: 0.07, 0.21, 0.10
```

```
Tasks: 184 total, 1 running, 183 sleeping, 0 stopped, 0 zombie
```



```
%Cpu(s):  2.5 us,  3.8 sy,  0.0 ni, 93.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  1924.1 total,  1614.7 free,   137.0 used,   172.4 buff/cache
MiB Swap:    0.0 total,    0.0 free,    0.0 used.  1705.5 avail Mem
```

```
PID USER PR  NI  VIRT  RES  SHR S  %CPU  %MEM  TIME+ COMMAND
7855 2231 root    R    2792   0%   1   6% top
1997   1 root    S   11924   1%   2   0% /usr/bin/adbd
1966   1 root    S    6876   0%   0   0% sshd: /usr/sbin/sshd [listener]
0
1938   1 root    S    5044   0%   3   0% /usr/libexec/bluetooth/blueto
othd
1958 1957 www-data S    4308   0%   3   0% nginx: worker process
1957   1 root    S    3876   0%   1   0% nginx: master process /usr/s
bin/ng
1933   1 dbus    S    2980   0%   3   0% dbus-daemon --system
2231   1 root    S    2928   0%   0   0% -sh
2170 2153 root    S    2792   0%   2   0% {led} /bin/sh /usr/bin/led 1
1    0 root    S    2792   0%   2   0% init
6828 6826 root    S    2792   0%   1   0% {cantest} /bin/sh /usr/bin/ca
ntest
6831 6829 root    S    2792   0%   3   0% {cantest} /bin/sh /usr/bin/ca
ntest
2153   1 root    S    2792   0%   1   0% {main} /bin/sh /usr/bin/main
1904   1 root    S    2792   0%   3   0% /sbin/syslogd -n
1908   1 root    S    2792   0%   3   0% /sbin/klogd -n
6826   1 root    S    2792   0%   1   0% {cantest} /bin/sh /usr/bin/can
test
6829   1 root    S    2792   0%   3   0% {cantest} /bin/sh /usr/bin/can
test
6824 2153 root    S    2660   0%   0   0% sleep 600
7853 6828 root    S    2660   0%   0   0% sleep 1
7854 6831 root    S    2660   0%   3   0% sleep 1
```



- %usr: 表示用户空间程序的 cpu 使用率 (没有通过 nice 调度)
- %sys: 表示系统空间的 cpu 使用率, 主要是内核程序
- %nic: 表示用户空间且通过 nice 调度过的程序的 cpu 使用率
- %idle: 空闲 cpu
- %irq: cpu 处理硬中断的数量
- %sirq: cpu 处理软中断的数量

### 3) 获取 CPU 温度信息

CPU 内置温度传感器作为 CPU 温度采集, 可以很方便的获取 CPU 内部温度。

```
root@myd-am62x:~# cat /sys/class/thermal/thermal_zone0/temp
56300
```

上面显示数字为千分之一度, 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-am62x:~# echo "scale=5000; 4*a(1)" | bc -l -q &
root@myd-am62x:~#
```

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, 我们可以通过顶部命令检查 CPU 利用率的变化, 如下所示:

```
root@myd-am62x:~# top
top - 03:45:43 up 7 min,  2 users,  load average: 0.00, 0.09, 0.07
Tasks: 155 total,   1 running, 154 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.5 us,  3.8 sy,  0.0 ni, 92.4 id,  0.0 wa,  0.0 hi,  1.3 si,  0.0 st
MiB Mem :  1924.1 total,  1613.8 free,   137.7 used,   172.6 buff/cache
MiB Swap:    0.0 total,    0.0 free,    0.0 used.  1704.8 avail Mem

  PID USER   PR  NI   VIRT  RES  SHR S  %CPU  %MEM    TIME+  COMMAND
20125 2231 root    R   3260    0%   1  24% bc -l -q
  1997    1 root    S  11924    1%   2   0% /usr/bin/adbd
  1966    1 root    S   6876    0%   0   0% sshd: /usr/sbin/sshd [listener]
```

约 3 分钟后, PI 结果被计算出来。 在此器件 CPU 使用率很高, 没有发生异常, 说明 CPU 压力测试可以通过。通过增加精确度要求, 还可以进一步提高测试压力。



```
root@myd-am62x:~# 3.14159265358979323846264338327950288419716939937
5105820974944592307\
8164062862089986280348253421170679821480865132823066470938446095505
8\
2231725359408128481117450284102701938521105559644622948954930381964
4\
2881097566593344612847564823378678316527120190914564856692346034861
0\
4543266482133936072602491412737245870066063155881748815209209628292
5\
```



## 2.2. 协处理器 M4F

AM62X 系列由 ARM Cortex-A53 和 Arm Cortex-M4F 组成的多核异构处理器。可以同时运行软实时 Linux 系统和硬实时 RTOS 系统。

关于 M4 核的具体使用，请阅读应用笔记《MYD-YM62X M4 协处理器应用开发笔记》。

## 2.3. Graphics

MYD-YM62X 系列中，YM6254 和 YM6252 两款的芯片内部都有 GPU 模块，支持 Vulkan 1.2, OpenGL 3.x/2.0/1.1，支持 2D 3D 图形，

**注意：YM6231 则没有 GPU。**

具体操作历程参考后续 5.1 图形图像处理章节。

## 2.4. Memory

YM62X 设备内存有 1G/2G 的 DDR4 可供选择，根据不通板型具有不同的搭配可供使用。

### 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-am62x:~# cat /proc/meminfo
MemTotal:      950828 kB
MemFree:       591912 kB
MemAvailable:  680040 kB
Buffers:       9500 kB
Cached:        147800 kB
SwapCached:    0 kB
Active:        28512 kB
Inactive:      195140 kB
Active(anon):  1260 kB
```



```

Inactive(anon):      81408 kB
Active(file):        27252 kB
Inactive(file):      113732 kB
Unevictable:         0 kB
Mlocked:             0 kB
SwapTotal:           0 kB
SwapFree:            0 kB
Dirty:              112 kB
Writeback:           0 kB
AnonPages:           66472 kB
Mapped:              79856 kB
Shmem:               16316 kB
KReclaimable:        18192 kB
Slab:                43020 kB
SReclaimable:        18192 kB
SUnreclaim:          24828 kB
KernelStack:         2768 kB
PageTables:          2280 kB
SecPageTables:        0 kB
NFS_Unstable:         0 kB
Bounce:              0 kB
WritebackTmp:         0 kB
CommitLimit:         475412 kB
Committed_AS:        304988 kB
VmallocTotal: 133143592960 kB
VmallocUsed:          10756 kB
VmallocChunk:         0 kB
Percpu:              536 kB
HardwareCorrupted:    0 kB
AnonHugePages:       8192 kB
ShmemHugePages:       0 kB
ShmemPmdMapped:       0 kB

```



```
FileHugePages:      0 kB
FilePmdMapped:      0 kB
CmaTotal:           262144 kB
CmaFree:            245508 kB
HugePages_Total:     0
HugePages_Free:      0
HugePages_Rsvd:      0
HugePages_Surp:      0
Hugepagesize:        2048 kB
Hugetlb:             0 kB
```

- MemTotal: 系统总内存量，单位为千字节（KB）。
- MemFree: 可用内存量，表示当前未被使用的内存。
- MemAvailable: 可用内存的估算值，考虑了内核的内存管理策略。
- Buffers: 用于文件系统缓冲的内存量，即文件读写时使用的缓冲区。在这个示例中，缓冲区占用了 9,500 KB。
- Cached: 用于文件系统缓存的内存量，包括缓存的文件和目录数据。
- SwapCached: 已缓存到交换空间的内存量，通常为 0 表示没有交换空间被使用。
- Active: 活跃内存，当前正在被使用的内存量。这包括了活跃的匿名内存（Active(anon)）和活跃的文件内存（Active(file)）。
- Inactive: 不活跃内存，曾经被使用但现在不再使用的内存量。这包括了不活跃的匿名内存（Inactive(anon)）和不活跃的文件内存（Inactive(file)）。

## 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。以下通过 1G DDR 来展示。

```
root@myd-am62x:~# free -m
```

	total	used	free	shared	buff/cache	available
Mem:	928	190	638	16	99	654
Swap:	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压力上的测试。可使用系统工具 memtester 进行测试, 如指定内存大小 200MB, 测试次数为 10, 测试命令为 "memtester 200M 10"。

下列以使用 300MB 内存空间, 单次测试为例:

```
root@myd-am62x:~# memtester 300M 1
memtester version 4.5.1 (64-bit)
Copyright (C) 2001-2020 Charles Cazabon.
Licensed under the GNU General Public License version 2 (only).

pagesize is 4096
pagesizemask is 0xfffffffffff000
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. eMMC

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-YM62X 系列标配 8GB eMMC，本节将讲解在 Linux 系统下查看与操作 eMMC 的步骤与方法。

### 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-am62x:~# fdisk -l
Disk /dev/mmcblk0: 7.28 GiB, 7818182656 bytes, 15269888 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: 0xd31fb445

Device            Boot  Start      End  Sectors  Size Id Type
/dev/mmcblk0p1          204800  450559   245760  120M  c W95 FAT32 (LBA)
/dev/mmcblk0p2          450560 15269887 14819328  7.1G  83 Linux

Disk /dev/mmcblk0boot0: 4 MiB, 4194304 bytes, 8192 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

Disk /dev/mmcblk0boot1: 4 MiB, 4194304 bytes, 8192 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
```



## 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-am62x:~# df -h
Filesystem                Size      Used Available Use% Mounted on
/dev/root                  6.6G      1.3G      5.0G   21% /
devtmpfs                  335.2M      4.0K    335.2M    0% /dev
tmpfs                     464.3M         0    464.3M    0% /dev/shm
tmpfs                     185.7M     11.0M    174.7M    6% /run
tmpfs                      4.0M         0      4.0M    0% /sys/fs/cgroup
tmpfs                     464.3M      4.0K    464.3M    0% /tmp
tmpfs                     16.0M         0     16.0M    0% /media/ram
tmpfs                     50.0M      12.0K    50.0M    0% /var/volatile
tmpfs                     92.9M         0     92.9M    0% /run/user/1000
/dev/mmcblk0p1            119.7M      3.1M    116.6M    3% /run/media/mm
cblk0p1
```

- /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。

## 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，以下将介绍使用 iotest 命令进行的读写性能测试。

### ● 读写测试

使用以下命令进行读写测试：

```
root@myd-am62x:~# iotest -e -l -a -s 100M -r 1M -i 0 -i 1 -i 2 -b emmc.xl
s
iotest: Performance Test of File I/O
Version $Revision: 3.489 $
Compiled for 64 bit mode.
Build: linux
```

略



```

File size set to 102400 kB
Record Size 1024 kB
Command line used: iozone -e -l -a -s 100M -r 1M -i 0 -i 1 -i 2
Output is in kBytes/sec
Time Resolution = 0.000001 seconds.
Processor cache size set to 1024 kBytes.
Processor cache line size set to 32 bytes.
File stride size set to 17 * record size.

                                random    rand
om    bkwd    record    stride
                                kB    reflen    write    rewrite    read    reread    read    write
                                read    rewrite    read    fwrite    frewrite    fread    freread
                                102400    1024    36526    37075    43635    43882    43683
36825
iozone test complete.
Excel output is below:
"Writer report"
    "1024"
"102400"    35996

"Re-writer report"
    "1024"
"102400"    36536

"Reader report"
    "1024"
"102400"    44490

"Re-Reader report"
    "1024"
"102400"    44549

```



"Random read report"

"1024"

"102400" 44286

"Random write report"

"1024"

"102400" 36476

此命令中，-e 表示计算时间时将 flush (fsync,fflush) 包括进来，-l 表示对所有文件操作使用 VxFS VX\_DIRECT 。告诉 VxFS 文件系统所有对文件的操作将跨 过缓存直接在磁盘上进行。 ， -a 表示用来使用全自动模式。生成包括所有测试操作的报告，使用的块大小从 4k 到 16M，文件大小从 64k 到 512M。 ， -s 表示指定测试文件大小， -r 表示指定测试块 大小， -i 表示用来指定运行哪个测试。(0=write/rewrite, 1=read/re-read, 2=random-read/write, 3=Read-backwards, 4=Re-write-record, 5=stride-read, 6=fwrite/re-fwrite, 7=fread/Re-fread, 8=random mix, 9=pwrite/Re-pwrite, 10=pread/Re-pread, 11=pwritev/Re-pwritev, 12=preadv/Re-preadv)此处我们使用读和写以及随机读写选项测试， -b 表示创建一个兼容 Excel 的二进制格式的文件。

此命令的结果均为 Kbyte/s，以上述结果为例，结果如下：

表 2-1eMMC 读写速度结果

测试项	Writer	Re-writer	Reader	Re-Reader	Random read	Random write
测试结果 (Kb/s)	35996	36536	44490	44549	44286	36476

#### 4) eMMC 的寿命查看

eMMC 使用寿命是指嵌入式多媒体卡 (eMMC) 存储设备在正常使用条件下可靠运行的时间期限或总量。eMMC 使用寿命通常受到多种因素的影响，包括但不限于：使用频率，写入/读取数量，环境温度，闪存类型。一旦 eMMC 存储器接近寿命末期，可能会出现数据丢失或设备性能下降的问题，因此在管理和备份数据时需要考虑存储器的寿命。eMMC 使用寿命是指存储设备在一定条件下能够可靠运行的时间或数据写入/读取总量，它是在设备设计和使用过程中需要关注和管理的重要因素之一。

通过以下测试命令查看 eMMC 的大约剩余寿命：

- 查看 eMMC 寿命



```
root@myd-am62x:~# mmc extcsd read /dev/mmcblk0 | grep Life
eMMC Life Time Estimation A [EXT_CSD_DEVICE_LIFE_TIME_EST_TYP_A]: 0x01
eMMC Life Time Estimation B [EXT_CSD_DEVICE_LIFE_TIME_EST_TYP_B]: 0x01
```

其中关于这些参数的解释：

- eMMC Life Time Estimation A：设备寿命估计类型 A，针对 MLC 用户分区擦除块的寿命估计
- eMMC Life Time Estimation B：设备寿命估计类型 B，针对 SLC 引导分区擦除块的寿命估计

以上参数的值均以 10% 的步长参考，例如：0x02 表示使用了 10%-20% 的设备寿命。

或者还可以使用以下命令查看：

```
root@myd-am62x:~# cat /sys/devices/platform/bus@f0000/fa10000.mmc/mmc_host/mmc0/mmc0:0001/life_time
0x01 0x01
```

显示的两个值分别为类型 A 和类型 B 的寿命估计。

## ● 寿命估计类型 B 参数值的标准

标准值如下表 2-2 所示：

表 2-2 设计寿命估计类型 B 标准

Value	Description
0x00	Not defined
0x01	0%-10% device life time used
0x02	10%-20% device life time used
0x03	20%-30% device life time used
0x04	30%-40% device life time used
0x05	40%-50% device life time used
0x06	50%-60% device life time used
0x07	60%-70% device life time used
0x08	70%-80% device life time used
0x09	80%-90% device life time used
0x0A	90%-100% device life time used
0x0B	Exceeded its maximum estimated device life time
Others	Reserved



## 2.6. RTC

RTC (Real-time clock) 本身是一个时钟，用来记录真实时间，当软件系统关机后保留系统时间并继续进行计时，系统重新开启后在将时间同步进软件系统。下面测试将系统时间写入 RTC，读取 RTC 时间并设置为系统时间并进行时间掉电保持的测试。

- 查看系统 RTC 设备

```
root@myd-am62x:~# ls /dev/rtc* -al
lrwxrwxrwx    1 root    root          4 Jan  1  1970 /dev/rtc -> rtc0
crw-----    1 root    root      251,  0 Jan  1  1970 /dev/rtc0
```

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

- 设置系统时间

在将系统时间设置为 Tue Sep 8 09:07:30 UTC 2020。

```
root@myd-am62x:~# date 092010002023.30
Wed Sep 20 10:00:30 UTC 2023
```

在执行上面步骤时可能会遇到以下问题：

```
root@myd-am62x:~# date 092010002023.30
Wed Sep 20 10:00:30 UTC 2023
root@myd-am62x:~# [ 1055.511004] systemd-journald[153]: Time jumped backwards, rotating.
root@myd-am62x:~# date
Wed Sep 20 03:01:47 UTC 2023
```

此时可以使用以下命令关闭网络时间同步：

```
root@myd-am62x:~# timedatectl set-ntp false
```

重新使用 date 命令设置时间即可。

- 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-am62x:~# hwclock -w
```

- 读取 RTC 时间并设置为系统时间

```
root@myd-am62x:~# hwclock -r
```



Wed Sep 20 03:03:35 2023 0.000000 seconds

### ● 掉电保持 RTC 时间

将设备关机断开电源，经过 2 分钟左右，重新上电开机。查看 RTC 时间和系统时间。

```
root@myd-am62x:~# hwclock -r
```

Wed Sep 20 03:11:27 2023 0.000000 seconds

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 2 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。

### ● 将系统时间与 RTC 时间同步

```
root@myd-am62x:~# hwclock -s
```

```
root@myd-am62x:~# date
```

Wed Sep 20 03:11:52 UTC 2023

```
root@myd-am62x:~#
```

如果将 hwclock -s 命令添加到启动脚本中就可以保证每次启动都可以将系统时间和 RTC 时间保持同步了。

## 2.7. Watchdog

Linux 内核包含 Watchdog 子系统，硬件设计过程中一般可以利用芯片内部的看门狗定时器或者使用外部看门狗芯片来实现 Watchdog 的功能，用于监测系统的运行。当系统出现异常情况无法喂狗时系统将可以进行自动复位。

- magic close 特性：当用户想停止看门狗时，向 wdt 节点写字符“V”即可停止看门狗。MYD-YM62X 看门狗驱动不支持此种方式关闭看门狗。
- 32 位看门狗定时器，可编程终止值
- 灵活的看门狗时钟，带 16 位预分频器
- 强大的看门狗刷新控制，支持：
  - 基于窗口的看门狗刷新
  - 增强的刷新序列
- 支持溢出中断和非法操作中断



- 支持内部和外部复位请求

本节将演示 watchdog 的使用，模拟内核崩溃测试看门狗系统复位功能，并提供修改示例设置看门狗超时时间。

## 1) 应用程序测试看门狗

- 设置看门狗超时时间：

```
ioctl(wdt_fd, WDIOC_SETTIMEOUT, &time_out);
```

以上为设置看门狗当前超时时间的参考代码。其中 fd 为看门狗设备的文件句柄。

- 看门狗应用程序测试：

```
root@myd-am62x:~# ./watchdog_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()
```

注意在使用本测试程序时，timeout 超时时间不能小于等于内核设定的看门狗默认复位时间 10s，也就是必须设定 timeout 时间大于 10s，内核中对应的文件和位置如下：

```
beste@system1:~/myir-ti-linux$ vi drivers/watchdog/rti_wdt.c
// SPDX-License-Identifier: GPL-2.0
/*
 * Watchdog driver for the K3 RTI module
 *
 * (c) Copyright 2019-2020 Texas Instruments Inc.
 * All rights reserved.
 */

#include <linux/clock.h>
#include <linux/device.h>
#include <linux/err.h>
#include <linux/io.h>
#include <linux/kernel.h>
#include <linux/mod_devicetable.h>
```



```
#include <linux/module.h>
#include <linux/moduleparam.h>
#include <linux/platform_device.h>
#include <linux/pm_runtime.h>
#include <linux/types.h>
#include <linux/watchdog.h>
//此处设定看门狗默认复位时间
#define DEFAULT_HEARTBEAT 10

/* Max heartbeat is calculated at 32kHz source clock */
#define MAX_HEARTBEAT 1000
```

然后运行看门狗应用，超时时间为 11s，每间隔 1s 喂一次狗，

```
root@myd-am62x:~# ./watchdog_test 11 1
Starting wdt_driver (time_out: 11, sleep: 1)
options = 0x81a0,id = K3 RTI Watchdog
Trying to set time_out value=11 seconds
The actual time_out was set to 11 seconds
Now reading back -- The time_out is 11 seconds
food watchdog, count = 0
food watchdog, count = 1
food watchdog, count = 2
food watchdog, count = 3
food watchdog, count = 4
```

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

```
root@myd-am62x:~# ./watchdog_test 11 12
Starting wdt_driver (time_out: 11, sleep: 12)
options = 0x81a0,id = K3 RTI Watchdog
Trying to set time_out value=11 seconds
The actual time_out was set to 11 seconds
Now reading back -- The time_out is 11 seconds
food watchdog, count = 0
U-Boot SPL 2023.04-gf4e7b03f15 (Sep 15 2023 - 01:45:12 +0000)
```



SYSFW ABI: 3.1 (firmware rev 0x0009 '9.0.5--v09.00.05 (Kool Koala)')

SPL initial stack usage: 13376 bytes

Trying to boot from MMC2



## 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，以下内容将会以 NCA9555 芯片的 P10 引脚 540 为例说明 GPIO 的使用过程。

#### 1) 查看目前 gpio 的使用情况

使用以下命令可以查看目前正在被占用被使用的 gpio 情况。

```
root@myd-am62x:~# cat /sys/kernel/debug/gpio
gpiochip3: GPIOs 362-413, parent: platform/601000.gpio, 601000.gpio:
gpio-412 (                |vbus                ) out lo
gpio-413 (                |rts                ) out lo

gpiochip2: GPIOs 414-505, parent: platform/600000.gpio, 600000.gpio:
gpio-426 (                |rts                ) out lo
gpio-450 (                |GPIO Key USER1    ) in  hi ACTIVE LOW
gpio-485 (                |int                ) in  hi

gpiochip1: GPIOs 506-529, parent: platform/4201000.gpio, 4201000.gpio:
gpio-509 (                |csi1-io            ) out lo
gpio-510 (                |regulator-6        ) out lo
gpio-521 (                |regulator-4        ) out lo
gpio-522 (                |am62-sk:d53        ) out lo ACTIVE LOW
gpiochip0: GPIOs 530-545, parent: i2c/2-0020, can sleep:
gpio-531 (                |reset              ) out hi ACTIVE LOW
gpio-534 (                |am62-sk:d54        ) out lo ACTIVE LOW
gpio-535 (                |PHY reset          ) out hi ACTIVE LOW
gpio-536 (                |PHY reset          ) out hi ACTIVE LOW
```

gpiochip0: GPIOs 530-545 为 NCA9555 芯片，P10 即 530+10 得到 P10 对应的 gpio540 引脚值。



## 2) 导出 GPIO

```
root@myd-am62x:~# echo 540 > /sys/class/gpio/export
```

导出成功后会在/sys/class/gpio/目录下生成 gpio500 这个目录。

```
root@myd-am62x:~# ls /sys/class/gpio/
export      gpio540      gpiochip362  gpiochip414  gpiochip506  gpiochip530  unexport
```

## 3) 设置/查看 GPIO 方向

- 设置输入

```
root@myd-am62x:~# echo "in" > /sys/class/gpio/gpio540/direction
```

- 设置输出

```
root@myd-am62x:~# echo "out" > /sys/class/gpio/gpio540/direction
```

- 查看 gpio 方向

```
root@myd-am62x:~# cat /sys/class/gpio/gpio540/direction
out
```

返回 in 表示输入，返回 out 表示输出。

## 4) 设置/查看 GPIO 的值

- 设置输出低

```
root@myd-am62x:~# echo "0" > /sys/class/gpio/gpio540/value
```

- 设置输出高

```
root@myd-am62x:~# echo "1" > /sys/class/gpio/gpio540/value
```

- 查看 gpio 的值

```
root@myd-am62x:~# cat /sys/class/gpio/gpio540/value
1
```



## 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。

### 1) 操作 LED 的目录为 `/sys/class/leds`, 该目录内容为:

```
root@myd-am62x:/sys/class/leds# ls
am62-sk:d53  am62-sk:d54  mmc0::      mmc1::      mmc2::
```

### 2) 以 D53 灯为例测试 LED

- 读取 D53 灯 LED 状态

其中 1 表示 LED 开启, 0 表示 LED 关闭

```
root@myd-am62x:~# cat /sys/class/leds/am62-sk\:d53/brightness
1
```

可知当前蓝灯 LED 为开启状态。

- 熄灭 LED

```
root@myd-am62x:~# echo 0 > /sys/class/leds/am62-sk\:d53/brightness
```

- 点亮 LED

```
root@myd-am62x:~# echo 1 > /sys/class/leds/am62-sk\:d53/brightness
```

- 开启 LED 触发模式

开启 “timer” 出发模式后，LED 默认以 1Hz 周期闪烁，占空比为 50%。

```
root@myd-am62x:~# echo "timer" > /sys/class/leds/am62-sk\:d53/trigger
```

- 改变 LED 灯闪烁时的亮灭占空比

通过调整 led 下的 `delay_on` 和 `delay_off` 属性，可以调整 LED 闪烁周期和亮灭占空比，例如：

```
root@myd-am62x:~# echo "100" > /sys/class/leds/am62-sk\:d53/delay_on
root@myd-am62x:~# echo "500" > /sys/class/leds/am62-sk\:d53/delay_off
```



### 3.3. Key(按键)

Linux 的/dev/input/eventx 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 hexdump 命令以及 dmesg 命令来查看按键是否有反应。MYD-YM62X 有三个按键，S1 是 MCU 复位按键；S2 是 Soc 复位按键；S3 是用户按键，已经在设备树配置。

#### 1) 按键测试

- 查看对应的输入设备 event 信息

```
root@myd-am62x:~# ls /proc/bus/input/
devices  handlers
root@myd-am62x:~# cat /proc/bus/input/devices
I: Bus=0019 Vendor=0001 Product=0001 Version=0100
N: Name="gpio-keys"
P: Phys=gpio-keys/input0
S: Sysfs=/devices/platform/gpio-keys/input/input0
U: Uniq=
H: Handlers=event0
B: PROP=0
B: EV=100003
B: KEY=1 0 0 0 0
```

由上可以知 gpio-keys 的对应设备事件为 event0。

- dump 按键信息

执行下面命令，操作按键 S3，串口终端会打印出如下信息：

```
root@myd-am62x:~# hexdump /dev/input/event0
00000000 6f8d 650a 0000 0000 a838 0007 0000 0000
00000100 0001 0100 0001 0000 6f8d 650a 0000 0000
00000200 a838 0007 0000 0000 0000 0000 0000 0000
00000300 6f8d 650a 0000 0000 ae56 0009 0000 0000
00000400 0001 0100 0000 0000 6f8d 650a 0000 0000
00000500 ae56 0009 0000 0000 0000 0000 0000 0000
```



每按一次 S3 当前终端会打印出当前事件码值，即按键正常。

## 3.4. RS485

本例程演示如何使用 Linux API 测试开发板 RS485 发送和接收数据功能，RS485\_1 设备为 J13 的 8, 9 口，为 UART6，设备节点为 ttyS1；RS485\_2 设备为 J13 的 11, 12 口，为 UART4，设备节点为 ttyS0。以下将以 RS485\_1 和 RS485\_2 相互连接来测试，硬件接口配置如下表：

表 3-1.RS485 接口配置

	MYD-YM62X	MYD-YM62X
接口	RS485_1	RS485_2
设备节点	ttyS1	ttyS0
测试软件	tty_test	tty_test

将 RS485\_1 的 485A1 和 485B2 分别与 RS485\_2 的 485A2 和 485B2 连接。

### 1) 开发板下 RS485-1 的接收和发送

首先输入以下命令查看 tty\_test 的使用说明：

```
root@myd-am62x:~# ./tty_test -h
Usage: ./tty_test [options]

Version 1.0
Options:
-d | --device name    tty device name: /dev/ttyO0
-m | --mode mode      tty mode. 0: RS232, 1: RS485 default mode: 0
-f | --flow           flow control
-b | --baudrate baudrate  set baudrate, default baudrate:9600
-l | --loop           operate circularly, default not operate circularly!
-w | --write frame    frame string. such as: 0123456789
-h | --help           Print this message
root@myd-am62x:~#
```

执行以下命令来让 RS485\_2 处于后台等待接收的状态：

```
root@myd-am62x:~# ./tty_test -m 1 -b 115200 -l -d /dev/ttyS0 &
```



[1] 694

执行以下命令让 RS485\_1 发送数据，使用 -l 循环发送，可以看到，如果正确发送以及正确接收，RS485\_2 会反馈 RECV 接收到的数据。确认正确后按下 Ctrl+C 停止。

```
root@myd-am62x:~# ./tty_test -m 1 -b 115200 -l -d /dev/ttyS0 -w 1234567890
SEND:1234567890
RECV:1234567890, total:10
SEND:1234567890
RECV:1234567890, total:10
SEND:1234567890
RECV:1234567890, total:10 ^C
```

执行 fg 命令让后台接收的 RS485\_2 返回前台运行，然后按下 Ctrl+C 停止。

```
root@myd-am62x:~# fg
./tty_test -m 1 -l -d /dev/ttyS0 -b 2400
^C
```

## 3.5. CAN

本节采用 Linux 系统常用的 cansend、candump 命令进行 SocketCAN 的通讯测试。开发板本身具有一个标准 CAN 和一个 M 核的 M\_CAN，以下会介绍使用同类型开发板的标准 CAN 接口对接测试的测试过程。

首先将 J13 座子的 4,5 口也就是 CANL2 和 CANH2 连接同类型开发板的对应接口。

### 1) 初始化 CAN 网络接口

- 设置 CAN 波特率

使用 canfd 需要先设置波特率，并开启 CAN 网络接口。参考下面命令分别将两块板子的仲裁波特率设置为 1M，数据波特率设备为 4M，并开启 canfd 功能。

```
root@myd-am62x:~# ip link set can0 down
root@myd-am62x:~# ip link set can0 up type can bitrate 1000000 dbitrates 4000000 fd on
```

至此，就可以使用 canfd 功能了。



## 2) 收发数据

设置其中一块开发板的 CAN0 为接收端，使用 candump 命令等待接受数据：

```
root@myd-am62x:~# candump can0 -L
```

设置另一块开发板的 CAN0 为发送端，使用 cansend 命令发送数据测试：

```
root@myd-am62x:~# cansend can0 00f##0.a1.a2.a3.a4.a5.a6.a7.a8.a9.10.11.12.1
3.14.15.16.17.18.19.20.21.23.24.25.26.27.28.29.30.31.32.33.34.35.36.37.38.39.40.41.
42.43.44.45.46.47.48.49.50.51.52.53.54.55.56.57.58.59.60.61.62.63.64
root@myd-am62x:~# cansend can0 00f##0.a1.a2.a3.a4.a5.a6.a7.a8.a9.10.11.12.1
3.14.15.16.17.18.19.20.21.23.24.25.26.27.28.29.30.31.32.33.34.35.36.37.38.39.40.41.
42.43.44.45.46.47.48.49.50.51.52.53.54.55.56.57.58.59.60.61.62.63.64
root@myd-am62x:~# cansend can0 00f##0.a1.a2.a3.a4.a5.a6.a7.a8.a9.10.11.12.1
3.14.15.16.17.18.19.20.21.23.24.25.26.27.28.29.30.31.32.33.34.35.36.37.38.39.40.41.
42.43.44.45.46.47.48.49.50.51.52.53.54.55.56.57.58.59.60.61.62.63.64
```

观察接收端开发板打印，可以看到成功接受到的数据：

```
root@myd-am62x:~# candump can0 -L
(0000000400.823368) can0 00F##4A1A2A3A4A5A6A7A8A9101112131415161718
1920212324252627282930313233343536373839404142434445464748495051525
3545556575859606162636400
(0000000401.913890) can0 00F##4A1A2A3A4A5A6A7A8A9101112131415161718
1920212324252627282930313233343536373839404142434445464748495051525
3545556575859606162636400
(0000000402.217901) can0 00F##4A1A2A3A4A5A6A7A8A9101112131415161718
1920212324252627282930313233343536373839404142434445464748495051525
3545556575859606162636400
```

## 3) 配置 can0 接口

CAN 数据收发之后显示 CAN 设备的详情和收发统计信息，其中 “clock” 的值代表 can 的时钟， “ drop” 的值代表丢包， “ overrun” 的值代表溢出， “ error” 代表总线错误。

```
root@myd-am62x:~# ip -details -statistics link show can0
4: can0: <NOARP,UP,LOWER_UP,ECHO> mtu 72 qdisc pfifo_fast state UP mod
e DEFAULT group default qlen 10
```



```
link/can promiscuity 0 minmtu 0 maxmtu 0
can <FD> state ERROR-ACTIVE (berr-counter tx 0 rx 0) restart-ms 0
    bitrate 1000000 sample-point 0.750
    tq 12 prop-seg 29 phase-seg1 30 phase-seg2 20 sjw 1 brp 1
    m_can: tseg1 2..256 tseg2 2..128 sjw 1..128 brp 1..512 brp_inc 1
    dbitrate 4000000 dsample-point 0.750
    dtq 12 dprop-seg 7 dphase-seg1 7 dphase-seg2 5 dsjw 1 dbrp 1
    m_can: dtseg1 1..32 dtseg2 1..16 dsjw 1..16 dbrp 1..32 dbrp_inc 1
    clock 80000000
    re-started bus-errors arbit-lost error-warn error-pass bus-off
        0          0          0          0          0          0          nu
mtxqueues 1 numrxqueues 1 gso_max_size 65536 gso_max_segs 65535 parent
bus platform parentdev 20701000.can
RX:  bytes packets errors dropped missed mcast
     201         6      0      0      0      0
TX:  bytes packets errors dropped carrier collsns
     192         3      0      0      0      0
root@myd-am62x:~#
```

#### 4) M\_CAN 的测试

M\_CAN 测试可以将 J13 座子的 1,2 口连接标准 CAN 的 4,5 口做通讯测试，也就是 CANL1 连接 CANL2，CANH1 连接 CANH2。

具体的测试过程请阅读应用笔记《MYD-YM62X M4 协处理器应用开发笔记》相关章节的内容。

### 3.6. USB

本节通过相关命令或热插拔、USB HUB 验证 USB Host 驱动的可行性，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(J16), 内核提示信息如下:

```
root@myd-am62x:~# [ 6229.728605] usb 2-1.1: new high-speed USB device n
umber 4 using xhci-hcd
[ 6229.941933] usb-storage 2-1.1:1.0: USB Mass Storage device detected
[ 6229.949370] scsi host0: usb-storage 2-1.1:1.0
[ 6230.977538] scsi 0:0:0:0: Direct-Access      aigo      U330          2.00
PQ: 0 ANSI: 4
[ 6230.993399] sd 0:0:0:0: [sda] 61440000 512-byte logical blocks: (31.5 GB/29.
3 GiB)
[ 6231.005361] sd 0:0:0:0: [sda] Write Protect is off
[ 6231.010560] sd 0:0:0:0: [sda] No Caching mode page found
[ 6231.016029] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 6231.026129]  sda: sda1
[ 6231.029480] sd 0:0:0:0: [sda] Attached SCSI removable disk
```

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

### ● 挂载设备

```
root@myd-am62x:~# mount /dev/sda1 /mnt/
```

### ● 读文件

需要提前在 U 盘上建立一个 test.txt 文件。

```
root@myd-am62x:~# ls /mnt
test.txt
root@myd-am62x:~# cat /mnt/test.txt
hello world!
```

### ● 写文件

```
root@myd-am62x:~# touch test.txt
root@myd-am62x:~# echo "hello world !!!" > test.txt
root@myd-am62x:~# cp test.txt /mnt
root@myd-am62x:~# cat /mnt/test.txt
hello world !!!
```



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) 卸载 U 盘

- 卸载设备命令

```
root@myd-am62x:~# umount /mnt
```

## 3.7. typeC 接口

本节通过使用 typeC 转 USB 的 OTG 接线连接 U 盘测试 typeC 口的 host 功能，以及通过 typeC 转 USB 公口线连接 PC 电脑模拟 U 盘测试 typeC 口的 device 功能。

### 1) 测试 host 功能

- 将 OTG 线接入开发板并接入 U 盘

```
root@myd-am62x:~# [ 281.493596] sd 0:0:0:0: [sda] 61440000 512-byte logic
al blocks: (31.5 GB/29.3 GiB)
[ 281.506449] sd 0:0:0:0: [sda] Write Protect is off
[ 281.519666] sd 0:0:0:0: [sda] No Caching mode page found
[ 281.525166] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 281.534353] sda: sda1
[ 281.537612] sd 0:0:0:0: [sda] Attached SCSI removable disk
root@myd-am62x:~# cat /sys/kernel/debug/usb/31000000.usb/mode
host
```

- 挂载 U 盘并读写测试

```
root@myd-am62x:~# mount /dev/sda /mnt
root@myd-am62x:~# cd /mnt
root@MYD-YM62X:/mnt# echo "this is a test file" > test.txt
root@MYD-YM62X:/mnt# cat test.txt
this is a test file
root@MYD-YM62X:/mnt#
```



## 2) 测试 devic 功能

首先使用 typeC 线连接开发板和主机 PC，查看 mode 确实已经自动切换为 device 模式：

```
root@myd-am62x:~# cat /sys/kernel/debug/usb/31000000.usb/mode
device
```

加载 g\_mass\_storage 驱动让开发板模拟为 USB 设备：

```
root@myd-am62x:~# modprobe g_mass_storage stall=0 file=/dev/mmcbk0p1
removable=1 iSerialNumber=1234
[ 127.144992] Mass Storage Function, version: 2009/09/11
[ 127.150337] LUN: removable file: (no medium)
[ 127.155077] LUN: removable file: /dev/mmcbk0p1
[ 127.159642] Number of LUNs=1
[ 127.162778] g_mass_storage gadget.0: Mass Storage Gadget, version: 2009/
09/11
[ 127.169936] g_mass_storage gadget.0: g_mass_storage ready
root@myd-am62x:~#
```

可以看到，已经将/dev/mmcbk0p1 模拟为 U 盘，查看 PC 显示：

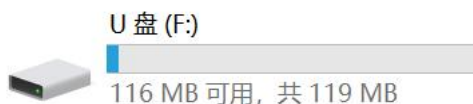


图 3-1 模拟挂载为 U 盘

## 3.8. Micro SD 卡

Micro SD Card，原名 Trans-flash Card(TF 卡)，Micro SD 卡是一种极细小的快闪存储器卡。Micro SD 卡相比标准 SD 卡，外形上更加小巧，是 SD 卡类型中尺寸最小的一种 SD 卡。尽管 Micro SD 卡的外形大小及接口形状与原来的 SD 卡有所不同，但接口规范保持不变，确保了兼容性。若将 Micro SD 插入特定的转接卡中，可当作标准 SD 卡来使用，SD 卡已成为目前消费数码设备中应用最广泛的一种存储卡，具有大容量、高性能、安全等多种特点的多功能存储卡。Micro SD 卡背面一般有 9 个引脚，包含 4 根数据线，支持 1bit/4bit 两种数据传输宽度。



本节将讲解在 Linux 系统下查看与操作 TF 卡的步骤与方法。

## 1) 查看 TF 卡容量

通过 fdisk -l 命令可以查询到 TF 卡分区信息及容量。

```
root@myd-am62x:~# fdisk -l
略
Disk /dev/mmcblk1: 29.72 GiB, 31914983424 bytes, 62333952 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: 0x798ed1d5

Device            Boot  Start      End  Sectors  Size Id Type
/dev/mmcblk1p1  *           2048   264191   262144  128M  c W95 FAT32 (LBA)
/dev/mmcblk1p2           264192  5933789  5669598   2.7G  83 Linux
[ 1422.906481] EXT4-fs (mmcblk1p2): mounted filesystem with ordered data
mode. Quota mode: none.
```

➤ /dev/mmcblk1: TF 卡对应设备节点

## 2) 查看 TF 卡分区信息

通过 df 命令可以查询到 TF 卡分区信息，使用情况，挂载目录等信息。

```
root@myd-am62x:~# df -h
Filesystem              Size      Used Available Use% Mounted on
略
/dev/mmcblk1p1          127.7M    127.7M      0    100% /run/media/boot-m
mmcblk1p1
/dev/mmcblk1p2           2.5G      1.5G    926.8M    62% /run/media/root-mmc
blk1p2
```

- /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



### 3) TF 卡的性能测试

性能测试主要测试 TF 卡在 linux 系统下对文件的读写速度，此处使用 iotest 命令进行测试。挂载需要测试的 TF 卡分区，这里以最后一个分区/dev/mmcblk1p2 为例，挂载目录为/mnt。

#### ● 读写测试

以下使用命令 iotest，只测试读写以及重读和重写速度，具体参数含义请参考 2.5 节 eMMC 中 eMMC 的性能测试部分。

```
root@MYD-YM62X:/mnt# iotest -e -l -a -s 100M -r 1M -i 0 -i 1
iotest: Performance Test of File I/O
Version $Revision: 3.489 $
Compiled for 64 bit mode.
Build: linux

Contributors:William Norcott, Don Capps, Isom Crawford, Kirby Collins
Al Slater, Scott Rhine, Mike Wisner, Ken Goss
Steve Landherr, Brad Smith, Mark Kelly, Dr. Alain CYR,
Randy Dunlap, Mark Montague, Dan Million, Gavin Brebner,
Jean-Marc Zucconi, Jeff Blomberg, Benny Halevy, Dave Boone,
Erik Habbinga, Kris Strecker, Walter Wong, Joshua Root,
Fabrice Bacchella, Zhenghua Xue, Qin Li, Darren Sawyer,
Vangel Bojaxhi, Ben England, Vikentsi Lapa,
Alexey Skidanov, Sudhir Kumar.

Run began: Fri Jan 1 00:03:17 2066

Include fsync in write timing
O_DIRECT feature enabled
Auto Mode
File size set to 102400 kB
```



```
Record Size 1024 kB
Command line used: iotest -e -l -a -s 100M -r 1M -i 0 -i 1
Output is in kBytes/sec
Time Resolution = 0.000001 seconds.
Processor cache size set to 1024 kBytes.
Processor cache line size set to 32 bytes.
File stride size set to 17 * record size.

kB      reflen    write    rewrite    read    reread
102400   1024      7188     7248      22411   22542
random   random    bkwd     record    stride
read     write     read     rewrite    read    fwrite    frewrite    fread    freread

iotest test complete.
```

根据结果得到该 Sd 卡的读写速度数据，读速度为 7188Kb/s，写速度为 22411Kb/s，重写速度为 7248Kb/s，重读速度为 22542Kb/s。

## 3.9. Display

本节将是对显示部分进行测试。MYD-YM62X 具有 1 路 hdmi 显示，1 路 lvds 输出，开发板将这 1 路 lvds 输出分为两个单路 LVDS J5 和 J6，以及双路 LVDS J8。

### 1) HDMI 显示

首先使用 HDMI 连接底板的 J29 接口和显示器，启动后观察 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。

查看当前显示的分辨率，比如当前默认显示的分辨率为 1920\*1080。

```
root@myd-am62x:~# cat /sys/kernel/debug/dri/0/state
plane[31]: plane-0
    crtc=crtc-0
    fb=55
        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=3
        start=001017bb
        size=8294400
        imported=no
        dma_addr=0x00000000b0700000
        vaddr=0000000032003f42

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 full range
plane[41]: 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 full range
crtc[38]: crtc-0
    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=1
connector_mask=1
encoder_mask=1
mode: "1920x1080": 60 148500 1920 2008 2052 2200 1080 1082 1087
1125 0x40 0x5
connector[40]: HDMI-A-1
crtc=crtc-0
self_refresh_aware=0
max_requested_bpc=0
root@myd-am62x:~#

```

## 2) LVDS 显示

LVDS 显示适配的是 MY-LVDS070C 这款屏幕，是 7 寸的分辨率为 1024\*600 分辨率的电容触摸屏，屏幕先通过 40pin 柔性连接，以下单路 LVDS 显示将以 J6 接口为例。

首先需要将/run/media/mmcblk0p1/uEnv.txt 文件，修改为以下内容。

```

root@myd-am62x:~# vi /run/media/mmcblk0p1/uEnv.txt

# This uEnv.txt file can contain additional environment settings that you
# want to set in U-Boot at boot time. This can be simple variables such
# as the serverip or custom variables. The format of this file is:
#   variable=value
# NOTE: This file will be evaluated after the bootcmd is run and the
#       bootcmd must be set to load this file if it exists (this is the
#       default on all newer U-Boot images. This also means that some
#       variables such as bootdelay cannot be changed by this file since

```



```
# it is not evaluated until the bootcmd is run.
#To enable the corresponding feature, please remove the comment on the li
ne below the description of the functionality for "name_overlays".
#Attention: Only one of the following features can be enabled at a time.
#HDMI audio playback.
#name_overlays=myir/myd-y62x-hdmi-audio.dtbo
#Dual-channel LVDS display
#name_overlays=myir/myd-y62x-lvds-dual.dtbo
#Single-channel LVDS display with J5 interface
#name_overlays=myir/myd-y62x-lvds.dtbo
#Single-channel LVDS display with J6 interface
name_overlays=myir/myd-y62x-lvds-j6.dtbo
```

将 name\_overlays=myir/myd-y62x-lvds-j6.dtbo 一行取消注释以开启 J6 的单路 LVDS 接口的使用，然后输入 sync 同步保存，重启开发板。

启动后可以看到屏幕正常显示。

其他接口显示参考上述操作，取消其他功能的注释即可，但只能同时开启一个。

## 3.10. Touch Panel

MYD-YM62X 系列开发板支持电容触摸，米尔电子提供 7 寸触摸的液晶屏配件，见表 1-1。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。触摸屏能点击，就表示它没问题。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。下面通过 evtest 命令屏的触摸功能做简单测试。

- 1) 按照 3.9 节将 MY-LVDS070C 触摸液晶显示器连接到开发板 J5 或者 J6，并注意按照 3.9 节内容修改 uEnv 文件，并且需要连接底板上的跳线帽，如果是 J5 接口连接 JP3，J6 接口则连接 JP2。
- 2) 终端执行“evtest”进入测试界面。

选择测试设备为触摸屏设备，这里默认为输入中断 1，测试界面选择 ‘1’ 按下回车即可开始测试：



```
root@myd-am62x:~# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:      gpio-keys
/dev/input/event1:      generic ft5x06 (79)
Select the device event number [0-1]: 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      50
      Min        0
      Max      1023
    Event code 1 (ABS_Y)
      Value      22
      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)

可知，接上 LVDS 后会识别为一个输入中断 1 设备。

3) 点击触摸屏，终端会打印对应的信息

```

Event: time 1695266775.747141, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID), value 17
Event: time 1695266775.747141, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 252
Event: time 1695266775.747141, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 337
Event: time 1695266775.747141, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 1
Event: time 1695266775.747141, type 3 (EV_ABS), code 0 (ABS_X), value 252
Event: time 1695266775.747141, type 3 (EV_ABS), code 1 (ABS_Y), value 337
Event: time 1695266775.747141, ----- SYN_REPORT -----
Event: time 1695266775.805983, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 249
Event: time 1695266775.805983, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 339
Event: time 1695266775.805983, type 3 (EV_ABS), code 0 (ABS_X), value 249
Event: time 1695266775.805983, type 3 (EV_ABS), code 1 (ABS_Y), value 339
Event: time 1695266775.805983, ----- SYN_REPORT -----
Event: time 1695266775.823965, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 338
Event: time 1695266775.823965, type 3 (EV_ABS), code 1 (ABS_Y), value 338

```



```
Event: time 1695266775.823965, ----- SYN_REPORT -----  
Event: time 1695266775.860706, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID), value -1  
Event: time 1695266775.860706, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 0  
Event: time 1695266775.860706, ----- SYN_REPORT -----
```

由上面可知，主要显示坐标值、键值，具体信息如下：

- EV\_SYN：同步事件
- EV\_KEY：按键事件，如 BTN\_TOUCH 表示是触摸按键
- EV\_ABS：绝对坐标，如触摸屏上报的坐标
- BTN\_TOUCH：触摸按键
- ABS\_MT\_TRACKING\_ID 表示采集信息开始，后面一个 ABS\_MT\_TRACKING\_ID 表示采集信息结束

单点触摸信息是以 ABS 承载并按一定顺序发送，如：

- ABS\_X：是相对于屏幕绝对坐标 X
- ABS\_Y：是相对于屏幕绝对坐标 Y

而多点触摸信息则是以 ABS\_MT 承载并按一定顺序发送，如：

- ABS\_MT\_POSITION\_X：表示屏幕接触面的中心点 x 坐标位置。
- ABS\_MT\_POSITION\_Y：表示屏幕接触面的中心点 Y 坐标位置



## 4. 网络与通信

本章内容主要用于测试和评估网络设备，如以太网，Wi-Fi，移动网络（可选）的连接，配置以及使用。很多情况下 Wi-Fi 与 Bluetooth 为二合一的模块，所以本章将一起介绍 Bluetooth 的配置与使用。

### 4.1. 网络设备

MYD\_YM62X 开发板包含两个千兆以太网接口和一个 Wi-Fi 和 Bluetooth 二合一的模块 L297B。下面分别对这两种网络设备的配置进行介绍。

#### 4.1.1. 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

首先通过通过 ifconfig 命令查看网络设备信息如下：

```
root@myd-am62x:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 1C:63:49:1A:6A:1F
          inet addr:192.168.30.216  Bcast:192.168.30.255  Mask:255.255.255.0
          inet6 addr: fe80::1e63:49ff:fe1a:6a1f/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1111  errors:0  dropped:204  overruns:0  frame:0
          TX packets:78  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:109508 (106.9 KiB)  TX bytes:8031 (7.8 KiB)
```

eth0 为实际的以太网设备，默认采用的是随机硬件 MAC 地址

下面介绍给 eth0 手动配置 IP 地址 192.168.0.100 的方法，命令如下：

```
root@myd-am62x:~# ifconfig eth0 192.168.0.100 netmask 255.255.255.0 up
```



上面的命令手动配置 eth0 的 IP 地址为 192.168.0.100, 子网掩码为 255.255.255.0, 以及默认配置的广播地址 192.168.0.255, 并通过 up 参数进行激活, 如下所示:

```
root@myd-am62x:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 1C:63:49:1A:6A:1F
          inet addr:192.168.0.100  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::1e63:49ff:fe1a:6a1f/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1123 errors:0 dropped:204 overruns:0 frame:0
          TX packets:90 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:113598 (110.9 KiB)  TX bytes:9840 (9.6 KiB)
```

- 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-am62x:~# ip addr flush dev eth0
root@myd-am62x:~# ip addr add 192.168.0.101/24 brd + dev eth0
root@myd-am62x:~# 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-am62x:~# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
P group default qlen 1000
    link/ether 1c:63:49:1a:6a:1f brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.101/24 brd 192.168.0.255 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。

嵌入式 Linux 系统使用 busybox 或者 ifupdown 工具进行网络管理的时候, 是根据系统文件/etc/network/interfaces 来对网络进行自动配置的, 所以这种情况下就需要预



先写好/etc/network/interfaces 文件或者修改/etc/network/interfaces 文件给 eth0 配置默认的静态 IP 地址，也可以配置通过 DHCP 动态获取 IP 地址。

- 使用 ifupdown 管理工具自动永久配置静态 IP 地址

使用 ifupdown 或者 busybox 自动永久配置静态 IP 地址的/etc/network/interfaces 文件如下：

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
    address 192.168.0.100
    netmask 255.255.255.0
    gateway 192.168.0.1
    dns-domain example.com
    dns-nameservers 192.168.0.11
```

这样每次开机 eth0 就默认配置 IP 地址为 192.168.0.100,子网 掩码为 255.255.255.0,网关地址为 192.168.0.1, dns 服务器为 192.168.0.11。

- 使用 ifupdown 管理工具配置动态获取 IP 地址

使用 ifupdown 或者 busybox 配置动态获取 IP 地址的/etc/network/interfaces 文件如下：

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
    pre-up /etc/network/nfs_check
    wait-delay 15
```

这样每次开机的时候 eth0 默认会向局域网内的 DHCP 服务器申请动态 IP 地址并进行 IP 地址，网关以及 DNS 的配置。



## 4.1.2. 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。

开发板板载 L297B Wi-Fi 和 Bluetooth 二合一模块，当前不支持 STA 和 AP 同时工作，L297B Wi-Fi 模块对应的驱动为 moal 和 mlan：

```
root@myd-am62x:~# lsmod
Module                Size  Used by    Tainted: G
...
moal                  753664  0
mlan                  528384  1 moal
```

驱动加载的过程中会将位于 /lib/firmware/nxp 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 mlan0，如下所示：

```
root@myd-am62x:~# ifconfig mlan0
mlan0      Link encap:Ethernet  HWaddr 5C:C5:63:7D:3B:8C
           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)
```

### 1) STA 模式手动连接 WiFi 热点

下面尝试手动连接附近的 Wi-Fi 热点“MYIR”，这是一个采用 WPA2 加密方式的 Wi-Fi 热点，密码为 myir@1601。

- 确保 mlan0 网络设备处于激活状态

```
root@myd-am62x:~# ifconfig mlan0 up
```

- 扫描附近 WiFi 热点

扫描附近的 wifi 热点，得到附近 Wi-Fi 热点列表如下：

```
root@myd-am62x:~# iw dev mlan0 scan | grep SSID
SSID: 360WiFi-4FF7C3
SSID: HP-Print-3C-LaserJet Pro MFP
```



```
SSID: MYIR
SSID: HUAWEI_B316_CA1C
SSID: MYIR
```

### ● wpa\_passphrase 设置 wifi 名字和密码

```
root@myd-am62x:~# wpa_passphrase MYIR myir@1601 >> /etc/wpa_supplicant.conf
root@myd-am62x:~# cat /etc/wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
    key_mgmt=NONE
}
network={
    ssid="MYIR"
    #psk="myir@1601"
    psk=6a85bda077eebd8473c9b585b74c823199c6cbb66a4f1d9e40c9b94e9605aa0
}
```

从一个 SSID 的 ASCII 密码生成一个 WPA PSK 进行加密操作。

### ● 关掉 wpa\_supplicant 进程

使用 wpa\_supplicant 连接、配置 WIFI 之前，需要先关掉 wpa\_supplicant 进程。

```
root@myd-am62x:~# killall wpa_supplicant
```

### ● 初始化 wpa\_supplicant

wpa\_supplicant 是一个连接、配置 WIFI 的工具，它的主要工作是通过 socket 与驱动交互并上报数据给用户层，而用户层也可以通过 socket 发送命令给 wpa\_supplicant 调用驱动来对 WiFi 芯片操作。它通常在后台运行，如下所示：

```
root@myd-am62x:~# wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant.conf
```

➤ -B : 在后台运行守护进程



- -c : 配置信息的路径
- -i : 监听的 wifi 接口

### ● 获取 WiFi IP 地址

```
root@myd-am62x:~# udhcpc -i wlan0
udhcpc: started, v1.31.1
udhcpc: sending discover
udhcpc: sending select for 192.168.40.107
udhcpc: lease of 192.168.40.107 obtained, lease time 7200
/etc/udhcpc.d/50default: Adding DNS 223.5.5.5
/etc/udhcpc.d/50default: Adding DNS 201.104.111.114

root@myd-am62x:~# ping www.baidu.com
PING www.baidu.com (112.80.248.75) 56(84) bytes of data.
64 bytes from 112.80.248.75 (112.80.248.75): icmp_seq=1 ttl=56 time=28.8 ms
64 bytes from 112.80.248.75 (112.80.248.75): icmp_seq=2 ttl=56 time=37.6 ms
64 bytes from 112.80.248.75 (112.80.248.75): icmp_seq=3 ttl=56 time=38.9 ms
```

### ● 使用 shell 脚本配置 Wi-Fi STA 模式

我们将手动配置 Wi-Fi 的过程整理成一个脚本 ifup\_wifi\_sta 提供给用户参考，用户只需要准备好/etc/wpa\_supplicant.conf 配置文件，然后执行这个脚本就可以配置好 STA 模式了，脚本内容如下：

```
#!/usr/bin/env sh

SSID=
PASSWD=
WLAN=wlan0
WPA_FILE=/etc/wpa_supplicant/wpa_supplicant-${WLAN}.conf
DRIVER_NAME=nl80211

usage(){
```



```

    echo "Usage: ./ifup_wifi_sta [-ssid wifi_sta_name] [-passwd wifi_sta_passwd]
[-driver nl80211 or wext]"
}

clean_stage(){
    killall udhcpc
    killall wpa_supplicant

    killall hostapd
    killall udhcpd
    sleep 1
}

enable_wifi(){
    T_HCI="phy0"
    RFKILL_SYS_PATH="/sys/class/rfkill/"
    dir=`ls ${RFKILL_SYS_PATH}`
    for i in ${dir}
    do
        if [ ${T_HCI} == `cat ${RFKILL_SYS_PATH}${i}/name` ];then
            echo 0 > ${RFKILL_SYS_PATH}${i}/state
            echo "find ${T_HCI} enable it"
            sleep 1
            echo 1 > ${RFKILL_SYS_PATH}${i}/state
        fi
    done
}

parse_input_info(){
    while [ $# -gt 0 ];do
        case $1 in

```



```

        -ssid)
            SSID="$2"
            shift
        ;;
        -passwd)
            PASSWD="$2"
            if [ ${#PASSWD} -lt 8 ];then
                echo "passwd should be 8...64"
                exit
            fi
            shift
        ;;
        -driver)
            DRIVER_NAME="$2"
            shift
        ;;
        -h)
            usage
            exit
        ;;
    esac
    shift $(( $# > 0 ? 1:0))
done

    echo "SSID:${SSID} PASSWD:${PASSWD} DRIVER:${DRIVER_NAME}"
}

connect_wifi(){
    if [ -n "${SSID}" ];then
        if [ ! -d ${WPA_FILE%/*} ];then
            mkdir -p ${WPA_FILE%/*}
        fi
        echo "ctrl_interface=/var/run/wpa_supplicant" > ${WPA_FILE}
    fi
}

```



```

    echo "ctrl_interface_group=0" >> ${WPA_FILE}
    echo "update_config=1" >> ${WPA_FILE}
    echo "" >> ${WPA_FILE}

    wpa_passphrase ${SSID} ${PASSWD} >> ${WPA_FILE}

fi

    wpa_supplicant -B -i ${WLAN} -c ${WPA_FILE} -D ${DRIVER_NAME} >/dev/
null 2>&1
}

obtain_dns(){
    time=10
    while [ $time -gt 0 ];do
        state=`wpa_cli -i${WLAN} -p/var/run/wpa_supplicant status | grep wpa_
state | awk -F[=] '{print $2}'`
        if [ "${state}" = "COMPLETED" ];then
            udhcpc -i ${WLAN}
            exit
        fi
        let time-=1
        sleep 1
    done
    echo "connect wifi error"
}

parse_input_info $@
clean_stage

```



```
enable_wifi  
connect_wifi  
obtain_dns
```

使用上面的脚本会根据输入的 ssid 和 passwd 参数生成/etc/wpa\_supplicant.conf 配置文件，并连接指定的 WiFi 热点，通过 DHCP 配置 wlan0 的 IP 地址信息，例如：

```
root@myd-am62x:~# /usr/bin/ifup_wifi_sta -ssid MYIR -passwd myir@1601
```

执行以上命令之后，wlan0 将会以 STA 模式连接到外部的 MYIR 热点。

## 2) 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 操作系统上构件无线接入点的一个比较方便的工具，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。如创建 SSID 为 MYIR\_TEST，PASSWORD 为 myir2020 的无线 wifi 热点。下面先以手动方式进行配置：

### ● 配置 wlan0 的 IP 地址

当使用 AP 模式时，需要为激活 wlan0 并配置一个静态 IP 地址，这里配置一个默认的 IP 地址：192.168.10.1。

```
root@myd-am62x:~# ifconfig wlan0 192.168.10.1 up
```

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-am62x:~# killall udhcpc  
root@myd-am62x:~# killall udhcpd  
root@myd-am62x:~# killall wpa_supplicant  
root@myd-am62x:~# killall hostapd
```

### ● 使用 wlan0 设备运行 DHCP 服务

wlan0 工作在 AP 模式，当其它设备连接这个 AP 热点的时候，它需要通过 wlan0 为其它设备动态分配 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，将内容修改为如下：

```
# 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        uap0
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，其它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。

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-am62x:~# 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>,下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 服务之后，就可以正常使用了上面配置的热点了。

```
root@myd-am62x:~# hostapd -B /etc/hostapd.conf
```

如果以太网卡 eth0 已经连接 Internet，那么通过下面的配置进行 IP 转发，那连接到 MYIR\_TEST 的设备也可以连接 Internet 了。

```
root@myd-am62x:~# echo "1" > /proc/sys/net/ipv4/ip_forward
root@myd-am62x:~# iptables -t nat -A POSTROUTING -s 192.168.10.1/24 -o
eth0 -j MASQUERADE
```

### ● 使用 shell 脚本配置 Wi-Fi AP 模式

我们将手动配置 Wi-Fi 的过程整理成一个脚本 ifup\_wifi\_ap 提供给用户参考，用户只需要准备好/etc/dhcp/dhcpd.conf 和/etc/hostapd.conf 两个配置文件，然后执行这个脚本就可以配置好 AP 模式了，脚本内容如下：

```
#!/usr/bin/env sh
ETH=eth0
WLAN=uap0
WLAN_IP=192.168.10.1
DHCP_FILE=/etc/myir_udhcpd.conf
HOSTAPD_FILE=/etc/myir_hostapd.conf
clean_stage(){
    killall udhcpc
    killall wpa_supplicant

    killall hostapd
    killall udhcpd
    sleep 1
}
enable_wifi(){
    T_HCI="phy0"
```



```

RFKILL_SYS_PATH="/sys/class/rfkill/"
dir=`ls ${RFKILL_SYS_PATH}`
for i in ${dir}
do
    if [ ${T_HCI} == `cat ${RFKILL_SYS_PATH}${i}/name` ];then
        echo 0 > ${RFKILL_SYS_PATH}${i}/state
        echo "find ${T_HCI} enable it"
        sleep 1
        echo 1 > ${RFKILL_SYS_PATH}${i}/state
    fi
done
}
enable_sta_mode(){

    ifconfig ${WLAN} up ${WLAN_IP}

    echo "1" > /proc/sys/net/ipv4/ip_forward
    iptables -t nat -A POSTROUTING -s ${WLAN_IP}/24 -o ${ETH} -j MASQUE
RADE

    sleep 1

    udhcpd ${DHCP_FILE}
    hostapd -B ${HOSTAPD_FILE}
}
clean_stage
enable_wifi
enable_sta_mode

```

在 Linux 系统控制台下执行/usr/bin/ifup\_wifi\_ap, 即可配置好 AP 模式。执行成功之后, 用户可以使用其它设备连接 MYIR\_TEST 热点进行测试



### 4.1.3. 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。BlueZ 是一套比较完善的 Bluetooth 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 L297b 二合一模块中的蓝牙功能，它对应的驱动为 hci\_uart 模块和 btbcm 模块，首先使用 hciattach 建立串口和蓝牙协议层的数据连接通道：

```
root@myd-am62x:~# hciattach /dev/ttyS3 any 115200 flow
[ 4074.747618] Bluetooth: HCI UART driver ver 2.3
[ 4074.752231] Bluetooth: HCI UART protocol H4 registered
[ 4074.758096] Bluetooth: HCI UART protocol LL registered
[ 4074.764054] Bluetooth: HCI UART protocol Broadcom registered
[ 4074.770479] Bluetooth: HCI UART protocol QCA registered
[ 4074.776194] Bluetooth: HCI UART protocol Marvell registered
Device setup complete
```

通过 lsmod 命令查看到驱动结果如下所示：

```
root@myd-am62x:~# lsmod | grep hci
hci_uart          77824  1
btqca             20480  1 hci_uart
btbcm             24576  1 hci_uart
bluetooth         507904  24 btqca,hci_uart,btbcm
```

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) 激活蓝牙设备 hci0

```
root@myd-am62x:~# hciconfig hci0 up
```

如果出现下面的提示：

```
root@myd-am62x:~# hciconfig hci0 up
Can't init device hci0: Operation not possible due to RF-kill (132)
```

则说明蓝牙的开关没有打开，可以通过 rfkill 查看当前蓝牙的开关状态，并做相应的处理。

- 查看无线设备开关状态



```
root@myd-am62x:~# rfkill list
0: phy0: wlan
    Soft blocked: no
    Hard blocked: no
1: hci0: bluetooth
    Soft blocked: yes
    Hard blocked: no
```

以上结果表明蓝牙设备的 rfkill ID 为 0，当前是处于 soft blocked 的状态，所以无法惊醒激活和后续的操作。

- **打开蓝牙开关**

```
root@myd-am62x:~# echo 1 > /sys/class/rfkill/rfkill0/state
```

通过以上命令改变蓝牙的状态之后，再次执行 hciconfig hci0 up 就没有问题了。

## 2) 扫描附近可获取的蓝牙设备

```
root@myd-am62x:~# hcitool scan
Scanning ...
    54:14:F3:CA:70:37      BESTE
    C0:3C:59:46:38:49      USER-WU
    84:5C:F3:24:9C:89      DESKTOP-LJRC636
```

## 3) 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。

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-am62x:~# bluetoothctl
Agent registered
[CHG] Controller 5C:C5:63:01:CC:0E Pairable: yes
[bluetooth]#
```

- **使能蓝牙控制电源**



```
[bluetooth]# power on  
Changing power on succeeded
```

- 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 successful
```

- 扫描附近可以连接的蓝牙设备

```
[bluetooth]# scan on  
Discovery started  
[CHG] Controller 5C:C5:63:01:CC:0E Discovering: yes  
[NEW] Device F0:A6:54:99:AA:F6 QDXCXY  
[NEW] Device 54:14:F3:CA:70:37 BESTE  
[bluetooth]# scan off  
[CHG] Device B0:FC:36:3B:CF:0E TxPower is nil  
[CHG] Device B0:FC:36:3B:CF:0E RSSI is nil  
[CHG] Device 54:14:F3:CA:70:37 RSSI is nil  
[CHG] Controller C0:84:7D:B6:34:69 Discovering: no  
Discovery stopped
```

如下通过上面扫描到的设备 54:14:F3:CA:70:37 BESTE 来进行后续测试。

- 进行设备配对

```
[bluetooth]# pair 54:14:F3:CA:70:37  
Attempting to pair with 54:14:F3:CA:70:37  
[CHG] Device 54:14:F3:CA:70:37 Connected: yes  
Request confirmation  
[agent] Confirm passkey 711813 (yes/no): yes  
[CHG] Device 54:14:F3:CA:70:37 Bonded: yes  
[CHG] Device 54:14:F3:CA:70:37 Modalias: bluetooth:v0006p0001d0A00
```



```
[CHG] Device 54:14:F3:CA:70:37 UUIDs: 00001000-0000-1000-8000-00805f9b34f
b
[CHG] Device 54:14:F3:CA:70:37 UUIDs: 0000110a-0000-1000-8000-00805f9b34f
b
[CHG] Device 54:14:F3:CA:70:37 UUIDs: 0000110b-0000-1000-8000-00805f9b34f
b
[CHG] Device 54:14:F3:CA:70:37 ServicesResolved: yes
[CHG] Device 54:14:F3:CA:70:37 Paired: yes
Pairing successful
[CHG] Device 54:14:F3:CA:70:37 ServicesResolved: no
[CHG] Device 54:14:F3:CA:70:37 Connected: no
```

至此，表示已经与小米手机蓝牙配对成功

#### ● 连接设备

```
[小米]# [11501.839856] input: 小米 (AVRCP) as /devices/virtual/input/input2
[CHG] Device 54:14:F3:CA:70:37 ServicesResolved: yes
[DEL] Device B0:FC:36:3B:CF:0E MYIR-BT
[小米]#
```

至此，你将可以看到手机上面有显示蓝牙以及连接。

### 4.1.4. 4G/5G

LINUX 设备也可以外接 4G 或者 5G 模块来拨号上网，4G 模块使用的是 EM05-CE，5G 模块包括 RM500Q 以及 RG801H。

MYD-YM62X 提供一个 M.2 Key B (J21)接口来接外部 5G 模块，已适配的 5G 模块型号为 RM500Q，插入板子后可直接使用，不支持热插拔。

#### 1) 查看 VID 和 PID

将装好 RM500Q 模块通过 M.2 接口接入开发板，然后启动开发板。使用 lsusb 查看 RM500Q 模块信息。

```
root@myd-am62x:~# lsusb
Bus 004 Device 002: ID 2c7c:0800
.....
```

➤ 2c7c:0800 : RM500Q 的 VID 和 PID 信息。



## 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-am62x:~# ls -l /dev/ttyUSB*
crw-rw---- 1 root dialout 188, 0 Dec  9 10:18 /dev/ttyUSB0
crw-rw---- 1 root dialout 188, 1 Dec  9 10:18 /dev/ttyUSB1
crw-rw---- 1 root dialout 188, 2 Dec  9 10:18 /dev/ttyUSB2
crw-rw---- 1 root dialout 188, 3 Dec  9 10:18 /dev/ttyUSB3
```

## 3) 使用 AT 指令进行初步测试

使用 AT 指令可以方便得查询信号强度，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 micro com 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-am62x:~# microcom /dev/ttyUSB2
at+csq
+CSQ: 22,99

OK
```

➤ 22,99: 22 就是信号质量，数字越大代表信号越强。

### ● 查看能否操作

```
at+cpin?
+CPIN: READY

OK
```

➤ +CPIN:READY : READY 代表就绪。

由于 MYD-YM62X 有两个 SIM 卡槽，如果发现上述测试步骤反馈失败，可以尝试切换模块的 SIM 卡槽选择：

```
AT+QUIMSLLOT=1/2 // 切换到 SIM1/2
AT+QUIMSLLOT? // 查看当前所使用的 SIM 卡
2
```



## ● 查看运营商

```
at+cops?
+COPS: 0,0,"CHN-UNICOM",2
OK
```

➤ "CHN-UNICOM",2: CHN-UNICOM 代表联通, 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

```
ATD134xxxx5673;
OK
```

## ● 发短信

```
at+cmgf=1
OK
at+cscs="GSM"
OK
at+cmgs="134xxxx5673"
> hello[ 63.568240] audit: type=1006 audit(1607509801.160:3): pid=741 uid=
0 old-auid=4294967295 auid=0 tty=(none) old-ses=4294967295 ses=2 res=1
+CMGS: 76
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, ESC 退出发送

## 4) ppp 拨号测试

这里采用开发板自带得 pppd 拨号命令:



```
root@myd-am62x:~# pppd call quectel-dial
root@myd-am62x:~# [ 33.774572] [dhd] CFG80211-ERROR) wl_cfg80211_net
dev_notifier_call : wdev null. Do nothing
[ 33.783789] [dhd] CFG80211-ERROR) wl_cfg80211_netdev_notifier_call : wde
v null. Do nothing
[ 37.259047] [dhd] CFG80211-ERROR) wl_cfg80211_netdev_notifier_call : wde
v null. Do nothing
[ 37.267814] [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-am62x:~# cat /var/log/quectel-dial.log
pppd options in effect:
debug          # (from /etc/ppp/peers/quectel-dial)
略
noauth         # (from /etc/ppp/peers/quectel-dial)
user card      # (from /etc/ppp/peers/quectel-dial)
password ????? # (from /etc/ppp/peers/quectel-dial)
略
not replacing default route to eth0 [192.168.40.1]
local IP address 10.241.79.143
remote IP address 10.64.64.64
primary DNS address 218.104.111.114
secondary DNS address 218.104.111.122
Script /etc/ppp/ip-up started (pid 749)
Script /etc/ppp/ip-up finished (pid 749), status = 0x0
```

可以看到已经正常连接，能获取 IP

输入以下命令查看“ppp0”发现已经正常获取 IP，并且能正常 ping 百度：

```
root@myd-am62x:~# ifconfig ppp0
ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.241.79.143  P-t-P:10.64.64.64  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:
1
```



```
RX packets:7 errors:0 dropped:0 overruns:0 frame:0
TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:3
RX bytes:130 (130.0 B) TX bytes:414 (414.0 B)
```

```
root@MYD-JYM62X:~# ping www.baidu.com
PING www.baidu.com (14.215.177.39) 56(84) bytes of data.
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=1 ttl=54 time=19.6 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=2 ttl=54 time=19.4 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=3 ttl=54 time=19.5 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=4 ttl=54 time=19.6 ms
```

➤ 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 &，其中“-s ctnet” 电信；“-s cmnet” 移动；“-s 3gnet” 联通：

```
root@myd-am62x:~# quectel-CM &
854
root@myd-am62x:~# [09-07_09:59:16:978] Quectel_QConnectManager_Linux_V1.6.0.16
[09-07_09:59:16:980] Find /sys/bus/usb/devices/2-1.3 idVendor=0x2c7c idProduct=0x125, bus=0x002, dev=0x003
[09-07_09:59:16:981] Auto find qmichannel = /dev/cdc-wdm0
[09-07_09:59:16:981] Auto find usbnet_adapter = wwan0
[09-07_09:59:16:981] netcard driver = qmi_wwan_q, driver version = 6.1.46-g6b301669c273-dirty
[09-07_09:59:16:981] ioctl(0x89f3, qmap_settings) failed: No such file or directory, rc=-1
[09-07_09:59:16:982] Modem works in QMI mode
[09-07_09:59:17:000] cdc_wdm_fd = 7
[09-07_09:59:17:106] Get clientWDS = 7
[09-07_09:59:17:137] Get clientDMS = 1
[09-07_09:59:17:170] Get clientNAS = 2
```



```
[09-07_09:59:17:201] Get clientUIM = 1
[09-07_09:59:17:234] Get clientWDA = 1
[09-07_09:59:17:266] requestBaseBandVersion EM05CEFCR06A04M1G_ND
[09-07_09:59:17:395] requestGetSIMStatus SIMStatus: SIM_READY
[09-07_09:59:17:427] requestGetProfile[1] 3gnet///0
[09-07_09:59:17:458] requestRegistrationState2 MCC: 460, MNC: 11, PS: Attach
ed, DataCap: LTE
[09-07_09:59:17:491] requestQueryDataCall IPv4ConnectionStatus: DISCONNECT
ED
[09-07_09:59:17:491] ifconfig wwan0 0.0.0.0
[09-07_09:59:17:508] ifconfig wwan0 down
[09-07_09:59:17:586] requestSetupDataCall WdsConnectionIPv4Handle: 0x8719e
440
[09-07_09:59:17:715] ifconfig wwan0 up
[ 36.505425] systemd-journald[177]: Time jumped backwards, rotating.
[09-07_09:59:17:224] busybox udhcpc -f -n -q -t 5 -i wwan0
udhcpc: started, v1.35.0
udhcpc: broadcasting discover
udhcpc: broadcasting select for 10.73.143.43, server 10.73.143.44
udhcpc: lease of 10.73.143.43 obtained from 10.73.143.44, lease time 7200
[09-07_09:59:17:515] /etc/udhcpc.d/50default: Adding DNS 202.103.44.150
[09-07_09:59:17:515] /etc/udhcpc.d/50default: Adding DNS 202.103.24.68
```

ifconfig 可以看到能正常获取 IP:

```
root@myd-am62x:~# ifconfig wwan0
wwan0      Link encap:Ethernet  HWaddr 0e:84:fc:87:fd:cb
            inet addr:10.42.46.46  Bcast:10.42.46.47  Mask:255.255.255.252
            UP BROADCAST RUNNING NOARP 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)
```

Ping 外网:



```
root@myd-am62x:~# ping www.baidu.com
PING www.baidu.com (14.215.177.39) 56(84) bytes of data.
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=1 ttl=54 time=20.1 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=2 ttl=54 time=19.7 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=3 ttl=54 time=19.5 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=4 ttl=54 time=19.4 ms
```

## 4.2. 网络应用

设备出厂烧录的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。

### 4.2.1. 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

通过网线将设备连接到交换机或路由器，例如接到开发板 eth0 网口。

```
root@myd-am62x:~# udhcpc -i eth0
udhcpc: started, v1.33.0
udhcpc: sending discover
udhcpc: sending select for 192.168.30.131
udhcpc: lease of 192.168.30.131 obtained, lease time 59353
deleting routers
adding dns 192.168.30.1
root@myd-am62x:~#
```

#### 2) 测试外网网址

```
root@myd-am62x:~# ping www.baidu.com -I eth0
PING www.baidu.com (112.80.248.76): 56 data bytes
64 bytes from 112.80.248.76: seq=0 ttl=55 time=13.029 ms
64 bytes from 112.80.248.76: seq=1 ttl=55 time=12.909 ms
```



```
64 bytes from 112.80.248.76: seq=2 ttl=55 time=13.239 ms
64 bytes from 112.80.248.76: seq=3 ttl=55 time=12.884 ms
64 bytes from 112.80.248.76: seq=4 ttl=55 time=12.812 ms
^C
--- www.baidu.com ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 12.812/12.974/13.239 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 112.80.248.76, 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。

## 4.2.2. 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 (<http://www.openssh.com/>) 提供的客户端和服务程序。

首先按照 4.1.1 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-am62x:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 1C:63:49:1A:6A:1F
          inet addr:192.168.40.216  Bcast:192.168.30.255  Mask:255.255.255.0
          inet6 addr: fe80::1e63:49ff:fe1a:6a1f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13590 errors:0 dropped:2181 overruns:0 frame:0
          TX packets:364 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1630491 (1.5 MiB)  TX bytes:27886 (27.2 KiB)
```

SSH 服务器的 IP 地址为 192.168.40.21，用 ping 命令测试设备和 SSH 服务器之间的连接正常之后即可进行下面的测试。

### ● SSH 客户端测试



设备作为 SSH 客户端连接 SSH 服务器，在设备上使用 ssh 命令登陆 SSH 服务器，命令和结果如下：

```
root@myd-am62x:~# ssh beste@192.168.40.21

Host '192.168.1.13' is not in the trusted hosts file.
(ecdsa-sha2-nistp256 fingerprint sha1!! 99:7b:56:89:e4:26:b5:2a:9a:80:27:85:a3:92:14:02:f0:ff:13:c1)
Do you want to continue connecting? (y/n) y
beste@192.168.40.21's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-79-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
*** System restart required ***
Last login: Wed Sep 20 10:52:24 2023 from 192.168.40.41
beste@system1:~$
```

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 my-linux 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

设备作为 SSH 服务端，其它设备远程连接到设备。

由于设备端默认也启动了 SSH 服务，因此我们也可以在其它具有 SSH 客户端的设备上使用 ssh 命令登陆到当前的设备，命令和结果如下：

```
beste@system1:~$ ssh root@192.168.40.216
```



上面的示例中，我们从远程以 root 账户登录到了设备上，并进入 console 控制台，可以对设备执行 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，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。

### 4.2.3. 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

```
PC $ scp test.txt root@192.168.40.216:~
The authenticity of host '192.168.40.216 (192.168.40.216)' can't be established.
ECDSA key fingerprint is SHA256:C6fXFGctxoRu2eBuCPe04fEcfsRzl82WJ1Rbbqjji
kp4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.40.216' (ECDSA) to the list of known ho
sts.
test.txt                                100% 2303KB   1.1MB/s   00:02
```

进入开发板 home 目录可以看到此文件，如下：

```
root@myd-am62x:~# ls
test.txt
```



## 2) 从本地拷贝文件到远程

```
# scp test.txt beste@192.168.40.216:~/
The authenticity of host '192.168.40.216 (192.168.40.216)' can't be established.
ECDSA key fingerprint is SHA256:KCTJPoHzafew1fRJmTx2hV4BNymgaZ1WDg2o
vdtqtCw.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.40.216' (ECDSA) to the list of known ho
sts.
myir@192.168.40.216's password:
test.txt
```

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。

### 4.2.4. TFTP

TFTP 使用客户端和服务端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-am62x:~# tftp --help
BusyBox v1.29.3 () 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 tftpd 32/64([http://tftpd32.jounin.net/tftpd32\\_download.html](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 test.txt 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-am62x:~# tftp -g -r test.txt -l test.txt 192.168.40.21
```

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

```
root@myd-am62x:~# tftp -p -l Ym62x-test -r Ym62x-test.txt 192.168.0.2
```

上面的命令会把目标设备上当前目录下的 Ym62x-test 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 Ym62x-test.txt。

## 4.2.5. 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 4.1.1 中我们已经测试过使用 DHCP 客户端模式自动获取 IP 地址; 在 4.1.2 中配置 WiFi 的 AP 模式时, 我们也测试了 DHCP 服



务端模式给连接的 WiFi 设备分配 IP 地址。这里再介绍一下使用 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-am62x:~# udhcpc -i eth0
udhcpc: started, v1.31.1
udhcpc: sending discover
udhcpc: sending select for 192.168.30.199
[ 3038.408887] IPv4: martian source 255.255.255.255 from 192.168.8.1, on dev eth0
[ 3038.414720] II header: 00000000: ff ff ff ff ff ff 20 da 22 fc ca 1b 08 00
udhcpc: lease of 192.168.30.199 obtained, lease time 3600
/etc/udhcpc.d/50default: Adding DNS 223.5.5.5
/etc/udhcpc.d/50default: Adding DNS 201.104.111.114
```

然后就可以为 eth0 配置好 IP 地址，以及网关，子网掩码，DNS 等信息如下：

```
root@myd-am62x:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 26:13:52:CE:15:40
          inet addr:192.168.30.199  Bcast:192.168.30.255  Mask:255.255.255.0
          inet6 addr: fe80::2413:52ff:fece:1540/64  Scope:Link
          inet6 addr: fd20:da22:fcca:1b00:2413:52ff:fece:1540/64  Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:20745 errors:0 dropped:752 overruns:0 frame:0
          TX packets:633 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1529726 (1.4 MiB)  TX bytes:93845 (91.6 KiB)
          Interrupt:51 Base address:0x8000

root@myd-am62x:~# cat /etc/resolv.conf
nameserver 223.5.5.5
nameserver 201.104.111.114
```

## 4.2.6. 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-am62x:~# iptables -A INPUT -p icmp --icmp-type 8 -j DROP
root@myd-am62x:~# 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 192.168.40.216 -w 10
PING 192.168.40.216 (192.168.40.216) 56(84) bytes of data.

--- 192.168.40.216 ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, time 9064ms
```

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-am62x:~# iptables -F
root@myd-am62x:~# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
```



-P OUTPUT ACCEPT

#### 4) 再次测试 ping 目标设备

```
PC$ ping 192.168.40.216 -w 5
PING 192.168.0.60 (192.168.0.60) 56(84) bytes of data.
64 bytes from 192.168.0.60: icmp_seq=1 ttl=64 time=0.254 ms
64 bytes from 192.168.0.60: icmp_seq=2 ttl=64 time=0.219 ms
64 bytes from 192.168.0.60: icmp_seq=3 ttl=64 time=0.222 ms
64 bytes from 192.168.0.60: icmp_seq=4 ttl=64 time=0.226 ms
64 bytes from 192.168.0.60: icmp_seq=5 ttl=64 time=0.238 ms
64 bytes from 192.168.0.60: icmp_seq=6 ttl=64 time=0.236 ms

--- 192.168.0.60 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4996ms
rtt min/avg/max/mdev = 0.219/0.232/0.254/0.019 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。

#### 4.2.7. 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-am62x:~# ethtool --help
ethtool version 4.5
Usage:
    ethtool DEVNAME Display standard information about device
    ethtool -s|--change DEVNAME      Change generic options
        [ speed %d ]
        [ duplex half|full ]
        [ port tp|aui|bnc|mii|fibre ]
        [ mdix auto|on|off ]
        [ autoneg on|off ]
```



```
[ advertise %x ]
[ phyad %d ]
[ xcvr internal|external ]
[ wol p|u|m|b|a|g|s|f|d... ]
[ sopass %x:%x:%x:%x:%x:%x ]
[ msglvl %d | msglvl type on|off ... ]
ethtool -a|--show-pause DEVNAME Show pause options
```

### 查看当前设备以太网卡的基本信息

```
root@myd-am62x:~# ethtool eth0
Settings for eth0:
    Supported ports: [ TP      MII ]
    Supported link modes:   10baseT/Half 10baseT/Full
                           100baseT/Half 100baseT/Full
                           1000baseT/Full
    Supported pause frame use: Symmetric
    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
    Advertised auto-negotiation: Yes
    Advertised FEC modes: Not reported
    Speed: 1000Mb/s
    Duplex: Full
    Auto-negotiation: on
    Port: Twisted Pair
    PHYAD: 5
    Transceiver: internal
    MDI-X: Unknown
    Supports Wake-on: d
```



Wake-on: d

Current message level: 0x000020f7 (8439)

drv probe link ifdown ifup rx\_err tx\_err hw

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 6 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-am62x:~# ethtool -s eth0 speed 100 duplex full autoneg off
root@myd-am62x:~# [66620.951859] am65-cpsw-nuss 8000000.ethernet eth0:
Link is Down
[66625.041328] am65-cpsw-nuss 8000000.ethernet eth0: Link is Up - 100Mbps
/Full - flow control off
```

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

#### 4.2.8. 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(IPV4 和 IPV6 下的 TCP、UDP、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 ubuntu 16.04 系统，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 192.168.0.2，设备 IP 为 192.168.0.60，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 (192.168.0.2)

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 (192.168.0.60)

设备上 iperf3 工作在客户端，TCP 模式，其中参数说明如下：

- -c 192.168.0.2 : 工作在客户端，连接服务端 192.168.0.2
- -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-am62x:~# iperf3 -c 192.168.0.2 -i 2 -t 10
```

```
Connecting to host 192.168.0.2, port 5201
```

```
[ 5] local 192.168.0.60 port 38994 connected to 192.168.0.2 port 5201
```

[ ID]	Interval	Transfer	Bitrate	Retr	Cwnd
[ 5]	0.00-2.00 sec	225 MBytes	940 Mbits/sec	0	529 KBytes
[ 5]	2.00-4.00 sec	224 MBytes	940 Mbits/sec	0	529 KBytes
[ 5]	4.00-6.00 sec	224 MBytes	941 Mbits/sec	0	560 KBytes
[ 5]	6.00-8.00 sec	224 MBytes	940 Mbits/sec	0	560 KBytes
[ 5]	8.00-10.00 sec	225 MBytes	940 Mbits/sec	0	560 KBytes

```
-----
[ ID] Interval      Transfer    Bitrate      Retr
[ 5]  0.00-10.00 sec  1.10 GBytes  940 Mbits/sec  0          sender
[ 5]  0.00-10.00 sec  1.09 GBytes  939 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 940Mbps 左右，没有重传，测试时 TCP 窗口值为 560KBytes。

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

Accepted connection from 192.168.0.60, port 38992

[ 5] local 192.168.0.2 port 5201 connected to 192.168.0.60 port 38994

[ ID]	Interval	Transfer	Bandwidth
-------	----------	----------	-----------

[ 5]	0.00-2.00 sec	218 MBytes	916 Mbits/sec
------	---------------	------------	---------------

[ 5]	2.00-4.00 sec	224 MBytes	941 Mbits/sec
------	---------------	------------	---------------

[ 5]	4.00-6.00 sec	224 MBytes	941 Mbits/sec
------	---------------	------------	---------------

[ 5]	6.00-8.00 sec	224 MBytes	940 Mbits/sec
------	---------------	------------	---------------

[ 5]	8.00-10.00 sec	224 MBytes	940 Mbits/sec
------	----------------	------------	---------------

[ 5]	10.00-10.04 sec	5.02 MBytes	938 Mbits/sec
------	-----------------	-------------	---------------

[ ID]	Interval	Transfer	Bandwidth	Retr
-------	----------	----------	-----------	------

[ 5]	0.00-10.04 sec	1.10 GBytes	937 Mbits/sec	0
------	----------------	-------------	---------------	---

[ 5]	0.00-10.04 sec	1.09 GBytes	936 Mbits/sec	
------	----------------	-------------	---------------	--

sender

receiver

Server listening on 5201

## 2) 测试 UDP 性能

### ● 服务端 (192.168.0.2)

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 (192.168.0.60)

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 192.168.0.2 : 工作在客户端，连接服务端 192.168.0.2
- -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-am62x:~# iperf3 -u -c 192.168.0.2 -i 2 -t 10 -b 100M
Connecting to host 192.168.0.2, port 5201
[ 5] local 192.168.0.60 port 42492 connected to 192.168.0.2 port 5201
[ ID] Interval           Transfer     Bitrate        Total Datagrams
[ 5]  0.00-2.00   sec    23.8 MBytes   100 Mbits/sec   17263
[ 5]  2.00-4.00   sec    23.8 MBytes   100 Mbits/sec   17266
[ 5]  4.00-6.00   sec    23.8 MBytes   100 Mbits/sec   17266
[ 5]  6.00-8.00   sec    23.8 MBytes   100 Mbits/sec   17263
[ 5]  8.00-10.00  sec    23.8 MBytes   100 Mbits/sec   17268
-----
[ ID] Interval           Transfer     Bitrate        Jitter    Lost/Total Datagrams
ms
[ 5]  0.00-10.00  sec    119 MBytes   100 Mbits/sec   0.000 ms   0/86326
(0%) sender
[ 5]  0.00-10.00  sec    119 MBytes   100 Mbits/sec   0.068 ms   0/86326
(0%) 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 100 Mbps 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
-----
Accepted connection from 192.168.30.206, port 38372
[ 5] local 192.168.30.2 port 5201 connected to 192.168.30.206 port 42492
[ ID] Interval           Transfer     Bandwidth      Jitter    Lost/Total Datagrams
[ 5]  0.00-2.00   sec    23.4 MBytes   98.0 Mbits/sec   0.079 ms   0/16918 (0%)
[ 5]  2.00-4.00   sec    23.8 MBytes   100 Mbits/sec   0.077 ms   0/17264 (0%)
```



```
[ 5] 4.00-6.00 sec 23.8 MBytes 100 Mbites/sec 0.065 ms 0/17268 (0%)
[ 5] 6.00-8.00 sec 23.8 MBytes 100 Mbites/sec 0.084 ms 0/17265 (0%)
[ 5] 8.00-10.00 sec 23.8 MBytes 100 Mbites/sec 0.074 ms 0/17267 (0%)
[ 5] 10.00-10.04 sec 486 KBytes 100 Mbites/sec 0.068 ms 0/344 (0%)
-----
[ ID] Interval      Transfer      Bandwidth      Jitter      Lost/Total Datagrams
[ 5] 0.00-10.04 sec 119 MBytes 99.6 Mbites/sec 0.068 ms 0/86326 (0%)
-----
Server listening on 5201
-----
```

客户端修改-b 参数，继续增大指定的 UDP 带宽，当开始出现丢包时测到的 UDP 带宽即为实际的 UDP 带宽。如果指定带宽达到以太网卡最高的 1Gbps，仍然没有丢包，那么 iperf3 测试返回的 bandwidth 即为实际的 UDP 带宽。例如下面的例子指定带宽为 1000Mbps，仍然没有丢包，但实际带宽其实为 600Mbps 左右。

```
root@myd-am62x:~# iperf3 -u -c 192.168.0.2 -i 2 -t 10 -b 1000M
Connecting to host 192.168.0.2, port 5201
[ 4] local 192.168.0.60 port 58904 connected to 192.168.0.2 port 5201
[ ID] Interval      Transfer      Bandwidth      Total Datagrams
[ 4] 0.00-2.00 sec 129 MBytes 539 Mbites/sec 16462
[ 4] 2.00-4.00 sec 143 MBytes 599 Mbites/sec 18293
[ 4] 4.00-6.00 sec 143 MBytes 599 Mbites/sec 18295
[ 4] 6.00-8.00 sec 143 MBytes 600 Mbites/sec 18300
[ 4] 8.00-10.00 sec 143 MBytes 599 Mbites/sec 18294
-----
[ ID] Interval      Transfer      Bandwidth      Jitter      Lost/Total Datagrams
[ 4] 0.00-10.00 sec 700 MBytes 587 Mbites/sec 0.080 ms 0/89643 (0%)
[ 4] Sent 89643 datagrams
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>



## 5. 图形系统

Linux 图形系统是 Linux 系统中比较复杂的一个子系统，一直处于不停的变革当中。它位于底层显示相关的设备驱动和上层用户界面应用之间，屏蔽了形形色色的底层硬件差异，同时又向上层用户界面提供统一的 API。

Linux/Unix 环境下最早的图形系统是 Xorg 图形系统，随着软硬件的发展，特别是嵌入式系统的发展，Xorg 显得过于笨重，随之又出现了一些更简洁，易于开发和维护的图形系统，比如 Wayland (<https://wayland.freedesktop.org/>)。下面是一个典型的嵌入式系统下图形系统的结构图。

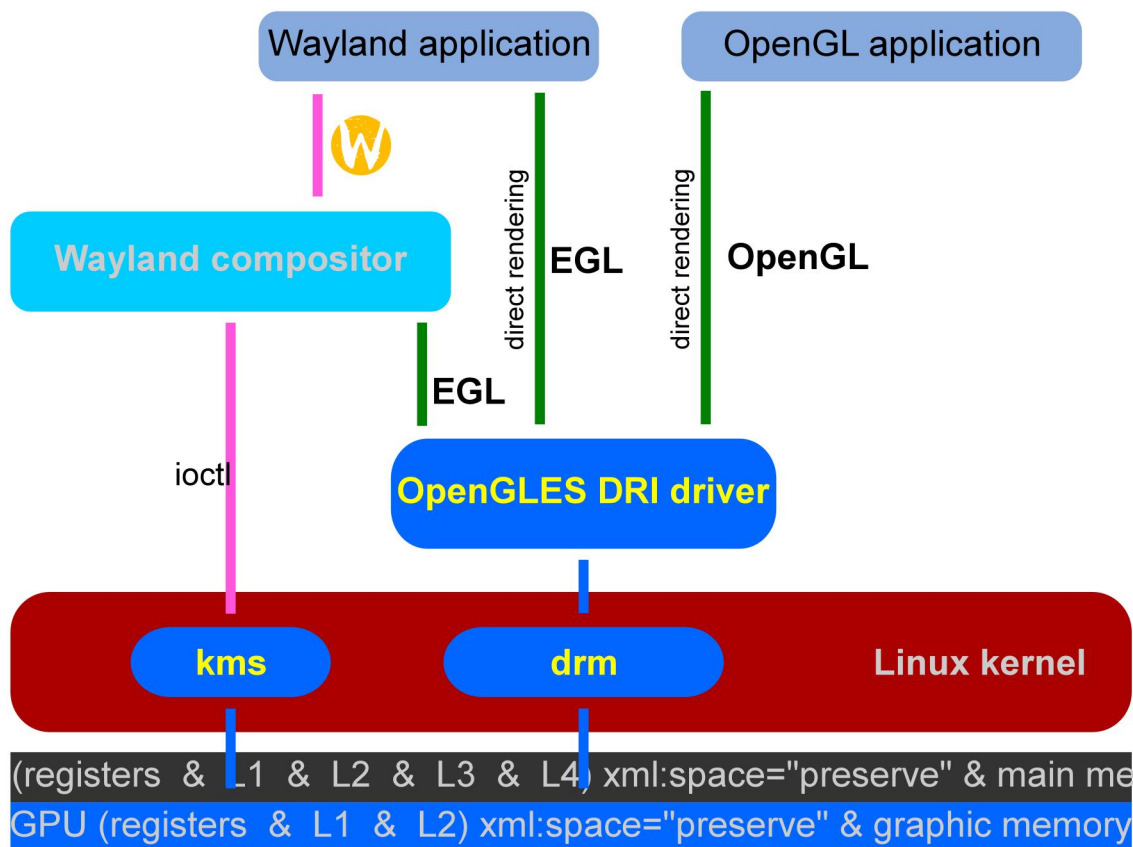


图 5-1. Embedded Linux Graphics Stack Overview

(by Shmuel Csaba Otto Traian; GFDL 1.2+ and CC-BY-SA 3.0+; created 2013-11-04)

上图可以看出，Wayland 并非进行图形界面开发的必需组件，例如我们出厂镜像中包含的 MEasy HMI V2.0 就可以通过指定平台插件直接运行 Qt5。以下章节将针对图形系统中部分关键内容进行测试和说明。



## 5.1. GPU

MYD-YM62X 系列开发板中，YM6254 和 YM6252 包含一个 GPU 核，支持 OpenGL 3.x/2.0/1.1，支持 2D 3D 图形。

以下使用开发板提供的位于 /user/share/example/opengl 下的 QT 例子测试：

```
root@myd-am62x:/usr/share/examples/opengl# ls
2dpainting          computegles31      cube                hellogl
es3                 legacy              paintedwindow       qopenglwindo
w                   threadedqopenglwidget
README              contextinfo         hellogl2            hellowi
ndow                opengl.pro          qopenglwidget       textures
```

使用 helloworld 进行测试：

```
root@myd-am62x:/usr/share/examples/opengl# ./helloworld/helloworld
```

## 5.2. Wayland+Weston+QT

QT 是一种跨平台 C++ 图形用户界面应用程序开发框架。它既可以开发 GUI 程序，也可用于开发非 GUI 程序，比如控制台工具和服务器。Qt 是面向对象的框架，使用特殊的代码生成扩展以及一些宏，Qt 很容易扩展，并且允许真正地组件编程。

Wayland 采用 Weston 图形桌面，可以在 weston 桌面上运行 QT 程序，开发板会在出厂的时候烧提供了一个丰富的 HMI 演示系统。

### 1) 运行 QT 例程

QT 的例子都在 /user/share/example 目录，运行一个例程：

```
root@myd-am62x:~# /usr/share/examples/widgets/touch/pinchzoom/pinchzoom
```

可以看到屏幕上显示 QT 画面。

如果想开机运行 QT 程序，需要在脚本中增加如下内容：

```
root@myd-am62x:~# cat qt_demo.sh
#!/bin/sh
export QT_WAYLAND_SHELL_INTEGRATION=xdg-shell
export QTWEBENGINE_DISABLE_SANDBOX=1
```



```
export QT_QPA_EGLFS_ALWAYS_SET_MODE=1
export WAYLAND_DISPLAY=/run/wayland-0
export XDG_RUNTIME_DIR=/run/user/0
/usr/share/examples/widgets/touch/pinchzoom/pinchzoom
```

## 2) 运行 HMI

MYD-YM62X 开发板出厂自带一个演示使用的 qt 例程，默认为开机启动，运行脚本为/home/root/hmi.sh

```
root@myd-am62x:~# cat hmi.sh
#!/bin/sh
export QT_WAYLAND_SHELL_INTEGRATION=xdg-shell
export QTWEBENGINE_DISABLE_SANDBOX=1
export QT_QPA_EGLFS_ALWAYS_SET_MODE=1
export WAYLAND_DISPLAY=/run/wayland-0
export XDG_RUNTIME_DIR=/run/user/0
flag=`cat /proc/cpuinfo | grep processor | wc -l`
echo "flag ${flag}"

AutoDisabled()
{
file="/etc/systemd/system/graphical.target.wants/weston.service"
pam_name="PAMName=weston-autologin"

if [ ! -e "$file" ]; then
    touch "$file"
fi

if grep -q "$pam_name" "$file"; then
    if ! grep -q "#$pam_name" "$file"; then
        sed -i "s/$pam_name/#$pam_name/" "$file"
    fi
else
```



```
    echo "#$pam_name" >> "$file"
fi
}

if [ ${flag} -gt 1 ];then
    /home/root/mxapp2
else
    time=1
    while [ $time -lt 10 ];
    do
        if [ -L /dev/input/touchscreen0 ];then
            break
        else
            sleep 3
            echo ${time}
            time=$((time + 1))
        fi
    done
    systemctl stop weston
    AutoDisabled
    /home/root/lvgl_ethercat_demo
fi

root@myd-am62x:~#
```

MYD-YM6231 由于性能不足，运行 weston 时会非常卡顿，使用 full 镜像启动时会通过脚本来运行显示 LVGL 的例程，并且会将自动启动 Weston 的参数关闭，而不会显示 weston 和 hmi。



## 6. 多媒体应用

### 6.1. Camera

本节采用系统自带 gstreamer, v4l2-utils, media-ctl 对 csi 摄像头进行评估测试。主要测试摄像头的预览和拍照。

#### 1) 摄像头窗口预览、拍照测试

- 查看设备基本信息

插入 usb 摄像头, 会生成字符设备/dev/video0

```
root@myd-am62x:~# ls -l /dev/video0
crw-rw---- 1 root video 81, 0 Jan 1 1970 /dev/video0
```

- 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-am62x:~# gst-launch-1.0 v4l2src device=/dev/video0 io-mode=2 !
video/x-raw,format=YUY2,width=640,height=480,framerate=30/1 ! waylandsink
Setting pipeline to PAUSED ...
Pipeline is live and does not need PREROLL ...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
Redistribute latency...
0:00:00.0 / 99:99:99.
```

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 工具来进行预览

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-am62x:~# gst-launch-1.0 v4l2src device=/dev/video0 io-mode=2 num-
buffers=10 ! video/x-raw,format=YUY2,width=640,height=480,framerate=30/
1 ! jpegenc ! filesink location=pic.jpeg
Setting pipeline to PAUSED ...
Pipeline is live and does not need PREROLL ...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
Redistribute latency...
Got EOS from element "pipeline0".
Execution ended after 0:00:01.480332314
Setting pipeline to NULL ...
Freeing pipeline ...
可以将 pic.jpeg
```

## 6.2. Audio

这节是测试播放音频。有两种方式，一种是 HMI2.0 应用直接播放，一种是通过 gstream 工具手动播放音频。

### 1) HMI2.0 音乐播放

HMI 多媒体音乐播放请参考 HMI 手册 《MEasy HMI2.0 开发手册》。

### 2) 手动播放音乐

- 播放 wav 格式的音乐

```
root@MYD-YM62X:~/audio# aplay att16k.wav
Press 'k' to see a list of keyboard shortcuts.
Now playing /home/root/audio/att16k.wav
Redistribute latency...
0:00:14.9 / 0:00:14.9
Reached end of play list.
```



- 播放 mp3 格式音乐

```
root@myd-am62x:~# ls
Born a Stranger.mp3  The promise.mp3
```

终端执行下面命令播放 MP3 格式音乐：

```
root@myd-am62x:~# aplay The\ promise.mp3
Press 'k' to see a list of keyboard shortcuts.
Now playing /usr/share/myir/Music/The promise.mp3
Redistribute latency...
0:02:54.0 / 0:03:15.8
```



## 7. 系统工具

默认映像中包含了一些常用的系统工具，便于用户在系统调试或实际部署的产品中查看和管理系统的各种资源，也可以在 SHELL 脚本或其他应用程序中调用。这些工具可能不完全满足用户的系统定制需求，此时系统开发人员需要根据实际情况做出适当的调整。

### 7.1. 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

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。

- 语法格式

输入以下命令查看 tar 语法格式：

```
root@myd-am62x:~# tar --help
BusyBox v1.33.0 (2023-05-19 10:43:45 CST) multi-call binary.

Usage: tar c|x|t [-ahvokO] [-f TARFILE] [-C DIR] [-T FILE] [-X FILE] [OPTION]...
[FILE]...

Create, extract, or list files from a tar file

      c      Create
      x      Extract
      t      List
      -f FILE Name of TARFILE ('-' for stdin/out)
      -C DIR  Change to DIR before operation
```



```
-v      Verbose
-O      Extract to stdout
-o      Don't restore user:group
-k      Don't replace existing files
-a      (De)compress based on extension
-h      Follow symlinks
-T FILE File with names to include
-X FILE File with glob patterns to exclude
--exclude PATTERN      Glob pattern to exclude
--overwrite            Replace existing files
--strip-components NUM  NUM of leading components to strip
--no-recursion          Don't descend in directories
--numeric-owner         Use numeric user:group
--no-same-permissions  Don't restore access permissions
--to-command COMMAND   Pipe files to COMMAND 567.001398]
```

delay timeout.

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。
- -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-am62x:~# tar -cvf test.tar.gz test.txt
root@myd-am62x:~# 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-am62x:~# tar -xvf test.tar.gz
test.txt
root@myd-am62x:~# 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

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。

- **语法格式**

在开发板终端输入以下命令查看 gzip 语法:

```
root@myd-am62x:~# gzip --help
BusyBox v1.31.1 () multi-call binary.
Usage: gzip [-cfkdt] [FILE]...
```

- **用 gzip 把文件压缩**

```
root@myd-am62x:~# gzip test.txt
root@myd-am62x:~# ls
OpenAMP_TTY_echo.elf  rs485_write  test.tar.gz  uart_test
rs485_read            test.c       test.txt.gz  wifi.conf
```

- **用 gunzip 工具解压**

用 gunzip 把文件解压, 示例如下:

```
root@myd-am62x:~# gunzip test.txt.gz
root@myd-am62x:~# ls
```



```
OpenAMP_TTY_echo.elf rs485_write test.tar.gz uart_test
rs485_read test.c test.txt wifi.conf
```

## 7.2. 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-am62x:~# mount -h
mount: invalid option -- 'h'
BusyBox v1.33.0 (2023-05-19 10:43:45 CST) multi-call binary.

Usage: mount [OPTIONS] [-o OPT] DEVICE NODE

Mount a filesystem. Filesystem autodetection requires /proc.
```

挂载 sd 卡第四分区示例如下：

```
root@myd-am62x:~# mount /dev/mmcb1k1p4 /mnt/
[10677.124115] EXT4-fs (mmcb1k1p4): mounted filesystem with ordered data
mode. Opts: (null)
[10677.130984] ext4 filesystem being mounted at /mnt supports timestamps
until 2038 (0x7fffffff)
```

### 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-am62x:~# mkfs.
```



mkfs.ext2 mkfs.ext3 mkfs.ext4 mkfs.fat mkfs.msdos mkfs.vfat

格式化 sd 卡第 7 个分区示例如下:

```
root@myd-am62x:~# mkfs.ext3 -c /dev/mmcblk1p7
mkfs from util-linux 2.32.1
mkfs.ext3 -c /dev/mmcblk1p7
mke2fs 1.44.3 (10-July-2018)
/dev/mmcblk1p7 contains a ext4 file system labelled 'userfs'
        created on Thu Aug 13 04:32:02 2020
Proceed anyway? (y,N) y
Creating filesystem with 330532 1k blocks and 82656 inodes
Filesystem UUID: 46bd5bb8-1882-4a68-901a-e11b4e71924b
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729, 204801, 221185
Checking for bad blocks (read-only test): done
Allocating group tables: done
Writing inode tables: done
Creating journal (8192 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-am62x:~# fsck -a /dev/mmcblk1p7
fsck from util-linux 2.32.1
/dev/mmcblk1p7: clean, 11/82656 files, 20726/330532 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-am62x:~# dumpe2fs -h
dumpe2fs 1.45.4 (23-Sep-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-am62x:~# dumpe2fs /dev/mmcb1k0p50
dumpe2fs 1.45.6 (20-Mar-2020)
Filesystem volume name:   <none>
Last mounted on:          /mnt
Filesystem UUID:          d4815f27-a633-453f-9b2a-14f8d14aab9d
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype ext
ent 64bit flex_bg sparse_super large_file huge_file dir_nlink extra_isize metadat
a_csum
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-am62x:~# dumpe2fs /dev/mmcb1k0p50| grep -i "inode size"
dumpe2fs 1.45.6 (20-Mar-2020)
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-am62x:~# dumpe2fs /dev/mmcb1k0p50| grep -i "block size"
dumpe2fs 1.45.6 (20-Mar-2020)
Block size:                4096
```

## 7.3. 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-am62x:~# 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-am62x:~# fdisk /dev/mmcbk2p2
Welcome to fdisk (util-linux 2.32.1).
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 0xd2dcd6ef.
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 的文件示例如下：

```
root@myd-am62x:~# 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, 0.195515 s, 10.7 MB/s
real    0m 0.20s
```



```
user    0m 0.00s
sys     0m 0.05s
```

### 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-am62x:~# 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-am62x:~# du ffmpeg1
2048    ffmpeg1
root@myd-am62x:~# 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-am62x:~# 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
- 



## 7.4. 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 v mstat kill 。

### 1) ps 显示当前进程工具

显示当前系统进程的运行情况，一般语法如下：

```
root@myd-am62x:~# ps --help
Usage:
ps [options]

Try 'ps --help <simple|list|output|threads|misc|all>'
or 'ps --help <s||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示例如下：

```
root@myd-am62x:~# ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.6	1.1	23804	4820	?	Ss	06:57	0:07	/sbin/init
root	2	0.0	0.0	0	0	?	S	06:57	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	I<	06:57	0:00	[rcu_gp]
root	4	0.0	0.0	0	0	?	I<	06:57	0:00	[rcu_par_gp]
root	8	0.0	0.0	0	0	?	I<	06:57	0:00	[mm_percpu_wq]
root	9	0.0	0.0	0	0	?	S	06:57	0:00	[ksoftirqd/0]
root	10	0.0	0.0	0	0	?	I	06:57	0:00	[rcu_preempt]
root	11	0.0	0.0	0	0	?	S	06:57	0:00	[migration/0]
root	12	0.0	0.0	0	0	?	S	06:57	0:00	[cpuhp/0]
root	13	0.0	0.0	0	0	?	S	06:57	0:00	[cpuhp/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: interruptable sleeping 可中断睡眠

■ D: uninterruptable 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



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-am62x:~# top -help
procs-ng 3.3.16
Usage:
top -hv | -bcEHiOSs1 -d secs -n max -u|U user -p pid(s) -o field -w [cols]
```

动态查看系统进程示例如下：

```
root@myd-am62x:~# top
top - 07:19:53 up 22 min, 1 user, load average: 0.00, 0.03, 0.05
Tasks: 109 total, 1 running, 108 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 1.3 sy, 0.0 ni, 98.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 425.8 total, 255.1 free, 56.6 used, 114.1 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 351.6 avail Mem
  PID USER  PR  NI  VIRT  RES  SHR  S  %CPU  %MEM  TIME+  COMMAND
  593 root   20   0   2348  1596  1056 S   1.3   0.4   0:25.27 apps.plugin
  553 root   20   0 129684 14196  2076 S   0.7   3.3   0:13.86 netdata
  594 root   20   0   2692  2140  1608 S   0.7   0.5   0:07.95 bash
```

### 3) kill 进程终止工具

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-am62x:~# ps -ef | grep mxapp2
root      1045      1 15 15:51 ?        00:00:02 /home/mxapp2 -platform eglfs
root      1174    1117   0 15:51 ttySTM0  00:00:00 grep mxapp2
```

然后输入以下命令终止进程：

```
root@myd-am62x:~# kill 1045
```

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-am62x:~# killall mxapp2
root@myd-am62x:~#
```



## 8. 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，SDK 的相关信息请参考《MYD-YM62X SDK 发布说明》。

### 8.1. 开发语言

#### 8.1.1. 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)

这里重点介绍，也是当前 SDK 默认支持的 bash。下面示例为镜像中 MEasy HMI 2.0 开机自启动为/etc/init.d/S99hmi，内容如下：

```
#!/bin/sh
export LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib:/out_qt_5.15.2/lib
export QT_QPA_PLATFORM=eglfs
export QT_QPA_EGLFS_INTEGRATION=none
export QT_QPA_FONTDIR=/usr/share/fonts
export QT_QPA_PLATFORM_PLUGIN_PATH=/out_qt_5.15.2/plugins
export QML_IMPORT_PATH=/out_qt_5.15.2/qml/
export QML2_IMPORT_PATH=/out_qt_5.15.2/qml/
export QML_IMPORT_PATH=/out_qt_5.15.2/qml
export QMLSCENE_DEVICE=softwarecontext
sleep 1
/out_qt_5.15.2/mxapp2/mxapp/mxapp2 -platform eglfs &
```



这个脚本首先设置 MEasy HMI 的环境变量，然后根据实用不同的参数调用/home/mxapp2 可执行程序启动 MEasy HMI，参数最后的"&"代表程序将进入后台运行。



### 8.1.2. 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 SDK，安装步骤请参《MYD-YM62X\_Linux 软件开发指南》，安装完成后需要配置一下 SDK 环境，如下：

```
PC $:source /home/my-linux/Yocto/SDK/environment-setup-cortexa7t2hf-neon-vfpv4-openstlinux_eglfs-linux-gnueabi
PC $:arm-openstlinux_eglfs-linux-gnueabi-gcc -march=armv7ve -mthumb -mfp
u=neon-vfpv4 -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=/home/my-linux/Y
octo/SDK/sysroots/cortexa7t2hf-neon-vfpv4-openstlinux_eglfs-linux-gnueabi
```

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：

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

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，\$C 包含有对应的系统库和配置信息，如果直接用 arm-openstlinux\_eglfs-linux-gnueabi-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。

```
$CC -v hello.c -o hello
Using built-in specs.
COLLECT_GCC=arm-openstlinux_eglfs-linux-gnueabi-gcc
COLLECT_LTO_WRAPPER=/home/my-linux/Yocto/SDK/sysroots/x86_64-openstlin
ux_eglfs_sdk-linux/usr/libexec/arm-openstlinux_eglfs-linux-gnueabi/gcc/arm-ope
nstlinux_eglfs-linux-gnueabi/8.2.0/lto-wrapper
Target: arm-openstlinux_eglfs-linux-gnueabi
.....
```

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-am62x:~# ./hello
hello world!
```

更复杂的示例和开发方式请参考《MYD-YM62X\_Linux 软件开发指南》应用移植部分的说明。

### 8.1.3. 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，第一个公开发行人版发行于 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。

#### 1) 查看系统支持的 python 版本

```
root@myd-am62x:~# python3.10 -V
Python 3.10.9
```

#### 2) 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-am62x:~# python3
Python 3.10.9 (main, Dec 6 2022, 18:44:57) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

在 Python 3.10.9 版本中,以上实例输出结果如下:

```
Hello, Python!
>>>
```

退出 Python 命令行，执行 exit()即退出 Python:

```
>>> exit()
root@myd-am62x:~#
```

#### 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-am62x:~# vi test.py
root@myd-am62x:~# cat test.py
```



```
#!/usr/bin/python3  
print ("Hello, Python!")
```

执行脚本文件，Python3 解释器在/usr/bin 目录中，使用以下命令执行脚本。

```
root@myd-am62x:~# chmod +x test.py  
root@myd-am62x:~# ./test.py  
Hello, Python!
```

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。当前系统不支持 pip 命令与 pip 安装 python 包，如果客户需要 pip 工具需要自行移植。



## 9. 参考资料

- Yocto 官方网站

<https://Yocto.org/>

- Linux 内核看门狗介绍

<https://www.kernel.org/doc/html/latest/watchdog/index.html>

- 嵌入式 Linux 下的 Qt

<https://doc.qt.io/qt-5/embedded-linux.html>

- TI AM62X Wiki

[https://software-dl.ti.com/processor-sdk-linux-rt/esd/AM62X/09\\_00\\_00\\_03/exports/docs/devices/AM62X/index.html](https://software-dl.ti.com/processor-sdk-linux-rt/esd/AM62X/09_00_00_03/exports/docs/devices/AM62X/index.html)



# 附录一 联系我们

## 深圳总部

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

负责区域：广东、广西、海南、重庆、云南、贵州、四川、西藏、香港、澳门

传真：0755-25532724

电话：0755-25622735

## 武汉研发中心

地址：武汉东湖新技术开发区关南园一路 20 号当代科技园 4 号楼 1601 号

电话：027-59621648

## 华东地区

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

负责区域：上海、福建、浙江、江苏、安徽、山东

传真：021-62087085

电话：021-62087019

## 华北地区

地址：北京市大兴区荣华中路 8 号院力宝广场 10 号楼 901 室

负责区域：辽宁、吉林、黑龙江、北京、天津、河北、山西、内蒙古、湖北、湖南、江西、河南、陕西、甘肃、宁夏、青海、新疆

传真：010-64125474

电话：010-84675491

## 销售联系方式

网址：[www.myir.cn](http://www.myir.cn)

邮箱：[sales.cn@myir.cn](mailto:sales.cn@myir.cn)

## 技术支持联系方式

邮箱：[support.cn@myir.cn](mailto:support.cn@myir.cn)

武汉研发中心电话：027-59621648

深圳总部技术电话：0755-22316235



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

凡是通过米尔电子直接购买或经米尔电子授权的正规代理商处购买的米尔电子全系列产  
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，以及米尔电子开发的部分软件源代码
- 6、可直接从米尔电子购买主要芯片样品，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，即成为米尔电子永久客户，享有再次购买米尔电子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，请先致电米尔电子  
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。

