

MYD-YF13X_Linux Software Evaluation Guide



File Status: [] Draft [√] Release	FILE ID:	MYIR-MYD-YF13X-SW-EG-EN-L5.15.67
	VERSION:	V1.0[DOC]
	AUTHOR:	Nene
	CREATED:	2023-05-23
	UPDATED:	2023-05-23



Revision History

VERSION	AUTHOR	PARTICIPANT	DATE	DESCRIPTION
V1.0	Nene		20230523	Initial Version: u-boot2021.1 L inux Kernel 5.15.67, Yocto 4. 1.



CONTENT

Revision History	- 2 -
CONTENT	- 3 -
1. Overview	- 8 -
1.1. Hardware Resources	- 8 -
1.2. Software Resources	- 8 -
1.3. Documents	- 9 -
1.4. Preparation	- 9 -
2. Core Components	- 10 -
2.1. CPU	- 10 -
1) View the CPU Information	- 10 -
2) View CPU Utilization	- 11 -
3) Gets CPU Temperature	- 12 -
4) CPU Stress Test	- 12 -
2.2. Memory	- 14 -
1) Check the Memory Information	- 14 -
2) Get memory usage	- 14 -
3) Memory Stress Test	- 15 -
2.3. eMMC	- 17 -
1).Check eMMC Capacity	- 17 -
2).View the eMMC Partition Information	- 18 -
3).eMMC Performance Test	- 18 -
2.4. NAND	- 19 -
1) . Check the Nand capacity and size	- 20 -
2) . View Nand partition information	- 20 -
3) . Nand performance testing	- 20 -
2.5. RTC	- 21 -
2.6. Watchdog	- 24 -
1) Stop the Watchdog	- 24 -
2) Watchdog testing in user space	- 24 -
3) Using the application to test the watchdog	- 25 -



2.7. Power Manager	- 27 -
1) View the Mode Supported by the Current Development Board	- 27 -
2) Set wake up source	- 27 -
3) Enter mem Power Management Mode	- 27 -
4) Wake up through debug serial port terminal	- 28 -
3. Peripheral Interface	- 29 -
3.1. GPIO	- 29 -
1) Export GPIO to User Space	- 29 -
2) Set/View the GPIO Direction	- 29 -
3.2. LED Lamp	- 31 -
1) View the LEDs	- 31 -
2) Test a LED	- 31 -
3.3. Key	- 33 -
1) Device Tree Node	- 33 -
2) Test Keys	- 33 -
3.4. RS232	- 35 -
1) Test Receiving Data from RS232 of Development Board	- 35 -
2) Test Sending Data to RS232 from the Development Board	- 36 -
3.5. RS485	- 38 -
1) Receive data from RS485 on development board	- 38 -
2) Send Data from RS485 on Development Board	- 39 -
3.6. CAN	- 41 -
1) Initializes the CAN network interface	- 41 -
2) Send and Receive Data	- 41 -
3) Statistics of can0 information	- 42 -
3.7. USB	- 43 -
1) Check the Kernel Message of USB	- 43 -
2) Mount and Read/Write the USB Flash Disk	- 44 -
3) Unmount USB Flash Disk	- 44 -
3.8. Micro SD card	- 45 -
1) Check SD Card Capacity	- 45 -
2) View the SD Card Partition Information	- 46 -



3) SD card Performance Test	- 47 -
3.9. ADC	- 49 -
1) View the SYSFS interface of the ADC	- 49 -
2) Get ADC Value	- 49 -
3.10. Display	- 49 -
1) Display scheme selection	- 50 -
3.11. Touch Panel	- 51 -
1) Touch screen connection	- 51 -
2) Touch test with evtest command	- 51 -
3) Print test information	- 53 -
4. Network Interface	- 56 -
4.1. Ethernet	- 56 -
1) Configure Ethernet IP addresses Manually and Temporarily	- 56 -
4.2. 4G	- 59 -
1). Check VID and PID	- 60 -
2).Check the kernel identification module	- 60 -
3).Use AT instruction for preliminary test	- 60 -
4). Use the ecm script to dial	- 61 -
5). Ping the Internet test	- 62 -
5. Network Applications	- 63 -
5.1. PING	- 63 -
1) Network Connection	- 63 -
2) PING Public Network	- 63 -
5.2. SSH	- 64 -
5.3. SCP	- 67 -
1) Copy Files from Remote to Local	- 67 -
2) Copy Files form Local to Remote	- 67 -
5.4. TFTP	- 69 -
1) Install TFTP Server Application	- 69 -
2) Configure TFTP Server	- 70 -
3) Restart TFTP Service	- 70 -
5.5. DHCP	- 71 -



5.6. IPTables	- 73 -
1) Configure iptables for Target Device	- 73 -
2) Ping the Target Device	- 73 -
3) Delete Rules for iptables	- 74 -
4) Ping the Target Device Again	- 74 -
5.7. Ethtool	- 75 -
5.8. iPerf3	- 78 -
1) Test Performance under TCP Mode	- 78 -
2) Test Performance under UDP Mode	- 80 -
6. Linux Graphics System	- 84 -
6.1. Wayland	- 86 -
1) Start the Weston Display Server	- 86 -
6.2. QT	- 86 -
1) Check Qt information	- 86 -
2) Introduction of QT Running Environment	- 87 -
3) Start Qt Applications	- 88 -
7. Multimedia	- 90 -
7.1. Camera	- 90 -
1) View the Device Topology of the USB Camera	- 90 -
7.2. Audio	- 91 -
7.3. Video	- 92 -
8. System Tools	- 93 -
8.1. Compress and Decompress Tools	- 93 -
1) tar Tool	- 93 -
2) gzip Tool	- 95 -
8.2. File System Tools	- 96 -
1) mount tool	- 96 -
2) mkfs tool	- 96 -
3) fsck tool	- 98 -
4) dumpe2fs tool	- 98 -
8.3. Disk Management Utils	- 101 -



1) fdisk Disk Partitioning Tool	- 101 -
2) dd tool	- 101 -
3) du tool	- 102 -
4) df tool	- 103 -
8.4. Process Management Utils	- 104 -
1) ps tool	- 104 -
2) top tool	- 106 -
3) vmstat: Virtual memory statistics tool	- 106 -
4) kill tool	- 109 -
9. Application Development	- 111 -
9.1. Development Language	- 111 -
1) SHELL	- 111 -
2) C/C++	- 112 -
3) Python	- 114 -
9.2. Database	- 116 -
9.3. Application Localization for Qt	- 118 -
1) Multiple Language	- 118 -
2) Fonts	- 120 -
3) Virtual Keyboard from Qt	- 121 -
10. Reference	- 123 -
Appendix A	- 124 -
Warranty & Technical Support Services	- 124 -



1. Overview

The Linux software evaluation guide describes the testing steps and evaluation methods for core and peripheral resources running open source Linux systems on MYIR Electronics' development boards. This paper can be used as a preliminary evaluation guide or as a test guide for general system developers.

1.1. Hardware Resources

This document applies to MYD-YF13X series boards of MYIR Electronics, which is a set of development platform based on ST company's high-performance embedded ARM processor STM32MP135DAF7 series. This development platform is composed of two parts, the core board MYC-YF13X and the bottom board MYB-YF13X. For detailed configuration parameters for the hardware part, please refer to the "MYD-YF13X Product Manual". The user will use some of the accessories during the evaluation test, see the list below.

Table 1-1. Optional Modules

Accessories	Interface	Description
Camera	USB	MY-CAM011B(2MegaPixels) https://www.myirtech.com/list.asp?id=534
LCD	RGB	MY-TFT070CV2 (7Inches LCD with Capacitive Touch Panel) https://www.myirtech.com/list.asp?id=477
RGB to HDMI interface module	RGB and HDMI ports	MY-RGB2HDMI (RGB LCD display output signal to HDMI output module) https://www.myirtech.com/list.asp?id=718

1.2. Software Resources

The BSP of MYD-YF13X series development board is based on the transplantation and modification of The Linux BSP of ST official open source community edition, and the system image is built by the Yocto project. All software resources of



Bootloader, Kernel and file system are open in the form of source code, please check the “MYD-YF13X SDK Release Notes” for details.

The development board has been programmed with “myir-image-full” image when it leaves the factory, so you only need to power it up and use it.

1.3. Documents

Depending on the stages of using the development board, the SDK contains different categories of documentation and manuals in addition to release notes, evaluation guides, development guides, application notes, and frequently asked questions. Please refer to Table 2-4 of “MYD-YF13X SDK Release Notes” for the detailed list of documents.

1.4. Preparation

Before starting to evaluate the development board software, you need to do some necessary preparation and configuration of the development board environment, including proper hardware wiring, configuring debugging serial ports, setting up and other steps. For detailed steps, please refer to “MYD-YF13X QSG” .

The following sections focus on how to evaluate and test the system's hardware resources and interfaces as well as software functions. Mainly with the help of some commonly used tools and commands under Linux, as well as our own examples of the application for testing. The software evaluation guide is divided into several parts, including: core components, peripheral interfaces, network applications, multimedia applications, development support applications, system tools and so on. The following chapters will make a comprehensive explanation for each part and describe in detail the specific evaluation methods and steps of each part of resources.



2. Core Components

In Linux, the PROC virtual file system is provided to query the parameters of various core resources and some common tools are provided to evaluate the performance of resources. The following will be specific to the CPU, memory, eMMC. The parameters of RTC and other core resources are also read and tested.

2.1. CPU

MYD-YF13X core chip is STM32MP135DAF7. The STM32MP135A is based on a high-performance single-core Arm®Cortex®-A7 32-bit RISC core operating at 1 GHz. The CPU core of the Cortex-A7 processor includes a 32 kbyte L1 instruction cache, a 32 kbyte L1 data cache, and a 128 kbyte secondary cache. The Cortex-A7 processor is a very power efficient application processor designed to deliver rich performance for high-end wearables as well as other low power embedded and consumer applications. It provides 20% more single-threaded performance than the Cortex-A5 and offers similar performance to the Cortex-A9.

1) View the CPU Information

Read the CPU provider and parameter information in the system, which can be obtained through the /proc/cpuinfo file.

```
root@myd-yf13x:~# cat /proc/cpuinfo
processor       : 0
model name     : ARMv7 Processor rev 5 (v7l)
BogoMIPS      : 48.00
Features       : half thumb fastmult vfp edsp thumbee neon vfpv3 tls vfpv4
                idiva idivt vfpd32 lpae evtstrm
CPU implementer : 0x41
CPU architecture: 7
CPU variant    : 0x0
CPU part       : 0xc07
CPU revision   : 5
```



Hardware : STM32 (Device Tree Support)
Revision : 0000
Serial : 000C00103232511438303631

- processor : The number of logical processing cores in a system can be either a physical core for multi-core processors or a virtual logical core using hyperthreading technology
- model name : The name and number of the CPU belongs to
- bogomips : A rough measure of the number of millions of instructions the CPU runs per second at kernel startup (Million Instructions Per Second)

2) View CPU Utilization

```
root@myd-yf13x:~# top
MMem: 77408K used, 374352K free, 8832K shrd, 5388K buff, 29848K cached
CPU:  0% usr  9% sys  0% nic 90% idle  0% io  0% irq  0% sirq
Load average: 0.10 0.20 0.09 1/93 665
```

PID	PPID	USER	STAT	VSZ	%VSZ	%CPU	COMMAND
665	656	root	R	2320	1%	9%	top
635	1	root	S	23180	5%	0%	/usr/sbin/iiod
572	1	root	S	16548	4%	0%	/lib/systemd/systemd-udev
598	1	root	S	14284	3%	0%	/lib/systemd/systemd-logind
651	649	root	S	9136	2%	0%	(sd-pam)
1	0	root	S	8008	2%	0%	{systemd} /sbin/init
649	1	root	S	7380	2%	0%	/lib/systemd/systemd --user
626	1	systemd-	S	6992	2%	0%	/lib/systemd/systemd-networkd
633	1	systemd-	S	6204	1%	0%	/lib/systemd/systemd-resolved
645	644	root	S	6024	1%	0%	systemd-userwork
646	644	root	S	6024	1%	0%	systemd-userwork
647	644	root	S	6024	1%	0%	systemd-userwork
644	1	root	S	5928	1%	0%	/lib/systemd/systemd-userdbd

- %usr: Represents the CPU utilization of the user space program (not scheduled by NICE)
- %sys: Represents the CPU utilization of system space, mainly kernel programs
- %nic: Represents the CPU usage of a program that has been scheduled through NICE in user space
- %idle: idle CPU
- %irq: The number of hard interrupts processed by the CPU
- %sirq: The number of soft interrupts processed by the CPU

3) Gets CPU Temperature

A temperature sensor is built into the CPU to collect the internal temperature of the CPU, which can be easily obtained through the hardware monitoring module in Linux system.

```
root@myd-yf13x:~# cat /sys/class/hwmon/hwmon0/temp1_input
39425
```

The number shown above is in the unit of one thousandth of degree centigrade, the result divided by 1000 is the current temperature.

4) CPU Stress Test

There are many ways to test the CPU pressure. For example, "bc" command can be used to calculate the PI, and we can test the stability of the CPU in the process of operation by this means.

```
root@myd-yf13x:~# echo "scale=5000; 4*a(1)" | bc -l -q &
root@myd-yf13x:~# 3.141592653589793238462643383279502884197169399375
1058209749445923078164062862089986280348253421170679821480865132823
0664709384460955058223172535940812848111745028410270193852110555964
462294895493038196
.....
[1]+  Done                  echo "scale=5000; 4*a(1)" | bc -l -q
[1]
```

The above command will calculate the PI in the background and accurate to 5000 decimal places. The calculation process needs a period of time. At this time, we can check the variation of CPU utilization by the top command, as follows:

```
root@myd-yf13x:~# top
MMem: 77352K used, 374408K free, 8840K shrd, 5488K buff, 30632K cached
CPU: 99% usr  0% sys  0% nic  0% idle  0% io  0% irq  0% sirq
Load average: 0.68 0.23 0.13 2/89 691
  PID  PPID  USER    STAT   VSZ %VSZ %CPU COMMAND
  687   656  root     R      2188  0% 100% bc -l -q
  691   656  root     R      2408  1%  0% top
    6     2  root    IW         0  0%  0% [kworker/0:0-eve]
```

After about 3 minutes, the result of PI was calculated. The CPU usage was very high and there were no exceptions, indicating that the CPU stress test had passed. By increasing the accuracy requirement, the test pressure can be further improved.



2.2. Memory

The STM32MP135DAF7 provides an external SDRAM interface that supports external memory up to 8Gbit capacity (1GB), 16-bit LPDDR2/LPDDR3 or DDR3/DDR3L, and operates at 533 MHz.

1) Check the Memory Information

The parameter information of memory in the system can be obtained by reading the `"/proc/meminfo"` file.

```
root@myd-yf13x:~# cat /proc/meminfo
```

```
MemTotal:      451760 kB
MemFree:       374408 kB
MemAvailable:  399876 kB
Buffers:       5488 kB
Cached:        30632 kB
SwapCached:    0 kB
Active:        12784 kB
Inactive:      31628 kB
...
```

- **MemTotal** : All available RAM sizes, physical memory minus reserved bits and kernel usage
- **MemFree** : LowFree + HighFree
- **Buffers** : The size used to cache the block device
- **Cached** : The buffer size of the file
- **SwapCached** : Memory that has been swapped out. Associated with I/O
- **Active** : Frequently (recently) used memory
- **Inactive** : Memory that has not been used much recently

2) Get memory usage



The “free” command can be used to read memory usage, with the “-m” parameter representing Mega Byte.

```
root@myd-yf13x:~# free -m
```

	total	used	free	shared	buff/cache	available
Mem:	451760	31844	374408		8840	45508
	399876					
Swap:	0	0	0			

- total : Total Memory
- used : The amount of memory used
- free : The amount of memory available

3) Memory Stress Test

The “memtester” tool under Linux system can be used to test the pressure of the existing memory of the system by giving the size and times of test memory. If the memory test is specified once and the memory under test is 300MByte, the test method is as follows.

```
root@myd-yf13x:~# memtester 300M 1
```

memtester version 4.5.1 (32-bit)
Copyright (C) 2001-2020 Charles Cazabon.
Licensed under the GNU General Public License version 2 (only).

```
pagesize is 4096
pagesizemask is 0xfffff000
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      : testing 49

```

```

                settiok      70

```

Done.



2.3. eMMC

eMMC is a data storage device that includes a MultiMediaCard (MMC) interface, a NAND Flash component. Its cost, small size, Flash technical independence, and high data throughput make it an ideal choice for embedded products. The STM32MP135 supports an 8-bit SDMMC interface (eMMC v5.1) with a maximum support of 128GB. This section explains how to view and operate eMMC in Linux.

1).Check eMMC Capacity

The eMMC partition information and capacity can be queried through the “fdisk -l” command as follows.

```
root@myd-yf13x:~# fdisk -l
Disk /dev/mmcblk1: 3.59 GiB, 3850371072 bytes, 7520256 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: gpt
Disk identifier: ADF86F4-9408-4A45-90D8-198535757B7FF
```

Device	Start	End	Sectors	Size	Type
/dev/mmcblk1p1	1024	2047	1024	512K	Linux reserved
/dev/mmcblk1p2	2048	3071	1024	512K	Linux reserved
/dev/mmcblk1p3	3072	11263	8192	4M	unknown
/dev/mmcblk1p4	11264	19455	8192	4M	unknown
/dev/mmcblk1p5	19456	20479	1024	512K	Linux reserved
/dev/mmcblk1p6	20480	151551	131072	64M	Linux filesystem
/dev/mmcblk1p7	151552	184319	32768	16M	Linux filesystem
/dev/mmcblk1p8	184320	6475775	6291456	3G	Linux filesystem
/dev/mmcblk1p9	6475776	7519231	1043456	509.5M	Linux filesystem

- /dev/mmcblk1p3 : Used to store fip partitions
- /dev/mmcblk1p5 : Used to store the uboot env



- /dev/mmcblk1p6 : Used for kernel resource storage partitions
- /dev/mmcblk1p7 : Used to store third-party library files
- /dev/mmcblk1p8 : Used to store the root file system
- /dev/mmcblk1p9 : Can be used by users to use partitions

2).View the eMMC Partition Information

By “df” command, you can query eMMC partition information, usage, mount directory and other information.

```
root@myir-yf13x:/# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        187M   0  187M   0% /dev
/dev/mmcblk1p8  2.9G  806M  1.9G  30% /
tmpfs           221M  160K  221M   1% /dev/shm
tmpfs           89M   8.8M   80M  10% /run
tmpfs           4.0M    0   4.0M   0% /sys/fs/cgroup
tmpfs           221M   20K  221M   1% /tmp
/dev/mmcblk1p6   55M   12M   39M  24% /boot
/dev/mmcblk1p7   14M    24K   13M   1% /vendor
/dev/mmcblk1p9  472M   36M  406M   9% /usr/local
tmpfs           221M  100K  221M   1% /var/volatile
tmpfs           45M    0   45M   0% /run/user/0
```

- tmpfs : Memory virtual file system, mounted to a different directory
- devtmpfs : Used to create dev for the system
- /dev/mmcblk1p6 : Used for kernel resource storage partitions
- /dev/mmcblk1p8 : A separate partition that holds the library files used by the GPU, mounted in /vendor
- /dev/mmcblk1p9 : single partition and can be used for users to use, mounted in /usr/local

3).eMMC Performance Test



The performance test mainly tests the reading and writing speed of files by eMMC in Linux system. The test is generally combined with the dual command of “time” and “dd” .

● Write Performance

```
root@myd-yf13x:~# time dd if=/dev/zero of=tempfile bs=1M count=100 conv
=fdatasync
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 8.46865 s, 12.4 MB/s
real    0m 8.48s
user    0m 0.00s
sys     0m 1.55s
```

When you use the dd command to write a file, you need to add the “conv = fdatasync” parameter, which means that after using the dd command to write multiple times, it will be synchronized to the disk (flush-cached). Because the disk is usually written to the cache first and then returned before it is written to the disk. Here we test that the write disk speed is 12.4MB/s.

● Read Performance

In embedded system, it is often necessary to test the system's performance of reading. If you want to test reading files directly from disk, you need to ignore the impact of cache. In this case, you can specify the parameter “iflag = direct, Nonblock” .

```
root@myd-yf13x:~# dd if=tempfile of=/dev/null bs=1M count=100 iflag=direct,nonblock
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 3.28757 s, 31.9 MB/s
```

The test results show that reading speed without cache is 31.9MB/s.

2.4. NAND



Nand-flash memory is a kind of flash memory, which adopts nonlinear macro unit mode internally and has solid large capacity. The implementation of memory provides a cheap and effective solution. Now check the Nand partition size and information in the Linux file system.

1). Check the Nand capacity and size

```
root@myd-yf13x:~# cat /proc/mtd
dev:   size  erasesize  name
mtd0: 00080000 00020000 "fsbl1"
mtd1: 00080000 00020000 "fsbl2"
mtd2: 00080000 00020000 "metadata1"
mtd3: 00080000 00020000 "metadata2"
mtd4: 00400000 00020000 "fip-a1"
mtd5: 00400000 00020000 "fip-a2"
mtd6: 00400000 00020000 "fip-b1"
mtd7: 00400000 00020000 "fip-b2"
mtd8: 0ee00000 00020000 "UBI"
```

2). View Nand partition information

```
root@myd-yf13x:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
ubi0_3          146M   77M   69M   53% /
devtmpfs         62M     0   62M    0% /dev
tmpfs            94M     0   94M    0% /dev/shm
tmpfs            38M   8.6M   29M   23% /run
tmpfs            4.0M     0   4.0M    0% /sys/fs/cgroup
tmpfs            94M     0   94M    0% /tmp
/dev/ubi0_2      59M   12M   48M   20% /boot
tmpfs            94M   12K   94M    1% /var/volatile
tmpfs            19M     0   19M    0% /run/user/0
```

3). Nand performance testing

- Write a file



```
root@myd-yf13x:~# time dd if=/dev/zero of=tempfile bs=1M count=100 conv=
=fdatasync
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 2.84869 s, 36.8 MB/s
real    0m 2.86s
user    0m 0.00s
sys     0m 1.01s
```

When you use the dd command to write files, you need to add conv=fdatasync, which means that after the dd command is written for N times, flush cache is synchronized to disks. Because writes to disk are usually written to cache first and then returned before they're written to disk. The disk write speed tested here is 36.8MB/s.

- **Read the file**

In embedded systems, you often need to test the system file read and write performance, and ignore the impact of cache when reading files. Run the following command to clear the cache.

```
root@myd-yf13x:~# sync; echo 3 > /proc/sys/vm/drop_caches
[ 7744.684308] sh (511): drop_caches: 3
```

Then continue testing the file read speed

```
root@myd-yf13x:~# dd if=tempfile of=/dev/null bs=1M count=100
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 5.65183 s, 18.6 MB/s
```

The read speed directly read from the disk is 18.6MB/s.

2.5. RTC

Real-time clock (RTC) is a clock that records the Real time. When the software system is shut down, the system time is retained and the time is continued. After

the software system is restarted, the system time is synchronized to the software system.

The STM32MP135 chip contains the RTC clock. If the actual product does not have high requirements on RTC power consumption and the power failure time is required within one month, the internal RTC can be directly used; otherwise, a dedicated external RTC chip is required. The common hwclock and date commands used in Linux are used to test the RTC. In the following test, the system time is written to the RTC, the RTC time is read, set as the system time, and the time power-off maintenance test is performed.

- **View RTC Devices in Linux System**

```
root@myd-yf13x:~# ls /dev/rtc* -al
lrwxrwxrwx 1 root root      4 Apr 29 10:18 /dev/rtc -> rtc0
crw----- 1 root root 253, 0 Apr 29 10:18 /dev/rtc0
```

rtc is a linux device, and its device node rtc0 is available under /dev for users to operate.

- **Set System Time**

Set the system time to Mon May 1 00:00:00 UTC 2023 as follows.

```
root@myd-yf13x:~# date 050100002023.00
[ 97.013942] systemd-journald[526]: Time jumped backwards, rotating.
Mon May  1 00:00:00 UTC 2023
```

- **Write System Time into RTC**

Writes the system time set by the previous date command to the RTC device.

```
root@myir-yf13x:~# hwclock -w
```

- **Read Data from RTC and Set to System Time**

```
root@myd-yf13x:~# hwclock -r
2023-05-01 00:01:02.263476+00:00
```

- **Keep RTC Time When Power is Off**



Turn off the power and restart the device after about 2 minutes. Check the RTC time and system time again.

```
root@myd-yf13x:~# hwclock -r  
2023-05-01 00:03:55.021306+00:00
```

The above information indicates that the RTC time and system time viewed after rebooting have increased by about 2 minutes compared with the previous setting, indicating that the RTC is working normally. If the precision of the RTC needs to be tested in detail, the outage time can be extended by such as 24 hours to test the difference between the RTC time and the standard time.

- **Make the System Time and the RTC Time Synchronization**

```
root@myd-yf13x:~# hwclock -s  
root@myd-yf13x:~# date  
Mon May 1 00:04:50 UTC 2023
```

If you add the “hwclock -s” command to the startup script of Linux System, you can ensure that system time and RTC time are synchronized with each startup.



2.6. Watchdog

The Linux kernel contains the Watchdog subsystem. In the process of hardware design, the Watchdog timer inside the chip or the external watchdog chip can be used to realize the watchdog function, which is used to monitor the operation of the system. When the system fails to feed the dog due to abnormal conditions, the system can automatically reset. The MYD-YF13X chip has a watchdog inside. This chapter explains how to test the watchdog in linux.

1) Stop the Watchdog

The watchdog can be closed by writing the "V" character to the watchdog device by command.

```
root@myd-yf13x:~# echo V > /dev/watchdog0
[ 71.038670] watchdog: watchdog0: nowayout prevents watchdog being stopped!
[ 71.044297] watchdog: watchdog0: watchdog did not stop!
```

From the above information, the system cannot turn off the watchdog due to the nowayout attribute.

2) Watchdog testing in user space

Intentionally crash the kernel, test the watchdog reset function, and restart the system by default for 32S,as follows:

```
root@myd-yf13x:~# echo c > /proc/sysrq-trigger
[ 381.425435] sysrq: Trigger a crash
[ 381.427400] Kernel panic - not syncing: sysrq triggered crash
[ 381.433145] CPU: 0 PID: 656 Comm: sh Not tainted 5.15.67 #1
[ 381.438794] Hardware name: STM32 (Device Tree Support)
[ 381.443843] [<c010e8ec>] (unwind_backtrace) from [<c010c158>] (show_stack+0x10/0x14)
[ 381.451623] [<c010c158>] (show_stack) from [<c0c32f30>] (panic+0xfc/0x2f0)
```




```
[ 381.458490] [<c0c32f30>] (panic) from [<c06bfc1c>] (sysrq_reset_seq_param
_set+0x0/0x84)
[ 381.466462] [<c06bfc1c>] (sysrq_reset_seq_param_set) from [<00000002>]
(0x2)
[ 381.473532] ---[ end Kernel panic - not syncing: sysrq triggered crash ]---
. . .
[ 0.000000] Kernel command line: root=PARTUUID=491f6117-415d-4f53-88c
9-6e0de54deac6 rootwait rw console=ttySTM0,115200
[ 0.000000] Dentry cache hash table entries: 65536 (order: 6, 262144 bytes,
linear)
[ 0.000000] Inode-cache hash table entries: 32768 (order: 5, 131072 bytes,
linear)
[ 0.000000] mem auto-init: stack:off, heap alloc:off, heap free:off
[ 0.000000] Memory: 315700K/524288K available (12288K kernel code, 125
5K rdata, 3444K rodata, 1024K init, 186K bss, 77516K reserved, 131072K cm
a-reserved, 0K highmem)
[ 0.000000] SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=1, Nodes=
1
[ 0.000000] trace event string verifier disabled

root@myd-yf13x:~#
```

3) Using the application to test the watchdog

- **Set the Timeout Settings of the Watchdog**

To implement the timeout through IOCTL command:WDIOC_SETTIMEOUT, and you also need a parameter, that is:timeout. Examples of use:

```
ioctl(fd, WDIOC_SETTIMEOUT, &timeout);
```

The above is the reference code for setting the current timeout of the watchdog. "fd" is the file handle of the watchdog device.

- **Demo Application of the Watchdog**



Compile the production execution file(watchdog) and copy it to the development board. Execute the following command:

```
root@myd-yf13x:~# ./watchdog
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()
```

Run the watchdog app for a timeout of 4s and feed the dog every 1s:

```
root@myd-yf13x:~# ./watchdog 4 1 0
Starting wdt_driver (timeout: 4, sleep: 1, test: ioctl)
Trying to set timeout value=4 seconds
The actual timeout was set to 4 seconds
Now reading back -- The timeout is 4 seconds
```

If the above 1s is changed to greater than 4s, the development board will restart after the required 4s feeding time. For more information on ST watchdog peripherals, please refer to the following links:

https://wiki.st.com/stm32mpu/wiki/Watchdog_overview.



2.7. Power Manager

This section demonstrates the Suspend functionality of Linux power management, allowing development boards to sleep and wake up via external events.

The Linux kernel generally provides three types of suspend: Freeze, Standby, and STR(Suspend to RAM), which can be triggered by writing "freeze", "standby", and "mem" to the `/sys/power/state` files in user space. MYD-YF13X only supports suspend to ram mode, that is, "mem" mode.

1) View the Mode Supported by the Current Development Board

```
root@myd-yf13x:~# cat /sys/power/state
mem
```

2) Set wake up source

To test wake-up from sleep mode, you need to set the wake-up source first. Here, set the debug serial port (uart4) as the wake-up source:

```
root@myd-yf13x:~# echo enabled > /sys/devices/platform/soc/40010000.serial/
tty/ttySTM0/power/wakeup
```

3) Enter mem Power Management Mode

Write a string to `/sys/power/state` in user space to enter the corresponding power management mode. The sleep mode is as follows:

```
root@myd-yf13x:~# echo "mem" > /sys/power/state
[ 465.180354] PM: suspend entry (deep)
[ 465.197085] Filesystems sync: 0.014 seconds
[ 465.208816] Freezing user space processes ... (elapsed 0.001 seconds) done.
[ 465.216194] OOM killer disabled.
[ 465.219426] Freezing remaining freezable tasks ... (elapsed 0.001 seconds)
done.
[ 465.226782] printk: Suspending console(s) (use no_console_suspend to deb
ug)
```



4) Wake up through debug serial port terminal

Take “mem” mode as an example to test. After entering the sleep command, the development board sleeps. At this point, the debugging port cannot input again, and the heartbeat lamp stops flashing. At this time, because debug wake-up is set, the system can be successfully awakened by inputting characters at random on the keyboard, as follows:

```
[ 465.876499] stm32-dwmac 5800a000.eth1 eth0: configuring for phy/rgmii-id
link mode
[ 466.885622] stm32-dwmac 5800a000.eth1: Failed to reset the dma
[ 466.885649] stm32-dwmac 5800a000.eth1 eth0: stmmac_hw_setup: DMA en
gine initialization failed
[ 466.886230] stm32-dwmac 5800e000.eth2 eth1: configuring for phy/rgmii-id
link mode
[ 467.889448] stm32-dwmac 5800e000.eth2: Failed to reset the dma
[ 467.938198] Restarting tasks ... done.
[ 467.982982] PM: suspend exit.
[ 368.676566] PM: suspend exit
```

At this point, the debugging port can input again, and the heartbeat lamp starts flashing.



3. Peripheral Interface

3.1. GPIO

The testing of GPIO is implemented through the file system sysfs interface. The pin of PF14 on STM32MP135 is taken as an example to illustrate the use of GPIO.

The pin number of GPIO on STM32MP135 follows the formula below:

```
#define PIN_NO(port, line) (((port) - 'A') * 0x10 + (line))
```

Where port is the GPIO group, line is the corresponding pin of the GPIO group, ((port)-'A') represents the number difference between the current group and GPIOA. For example, the PIN_NO corresponding to PF14 is: $PIN_NO('F', 14) = (0x46 - 0x41) * 0x10 + 14 = (70 - 65) * 16 + 14 = 94$.

1) Export GPIO to User Space

```
root@myd-yf13x:~# echo 94 > /sys/class/gpio/export
```

After the successful exporting to user space, the directory PF14 will be generated under the directory */sys/class/gpio/*.

2) Set/View the GPIO Direction

- Set as Input

```
root@myd-yf13x:~# echo "in" > /sys/class/gpio/PF14/direction
```

- Set as Output

```
root@myd-yf13x:~# echo "out" > /sys/class/gpio/PF14/direction
```

- View the Direction of GPIO

```
root@myd-yf13x:~# cat /sys/class/gpio/PF14/direction
out
```

Return "in" for input and "out" for output.

3) Set/View the GPIO Value

- **Set Output Low:**

```
root@myd-yf13x:~# echo "0" > /sys/class/gpio/PF14/value
```

- **Set Output High:**

```
root@myd-yf13x:~# echo "1" > /sys/class/gpio/PF14/value
```

- **View the GPIO Value:**

```
root@myd-yf13x:~# cat /sys/class/gpio/PF14/value  
1
```

You can see that the output level of PF14 is high. You can use a multimeter to measure the PF14 pin of J7. You can see that the voltage is about 3.3V. Additionally, users who need to control GPIO in their application can refer to the Wiki:

https://wiki.st.com/stm32mpu/wiki/How_to_control_a_GPIO_in_userspace.



3.2. LED Lamp

The Linux system provides a separate subsystem to facilitate the operation of LED devices from user space. The subsystem provides an interface for LED devices in the form of files. These interfaces are located in the `/sys/class/leds` directory. In the list of hardware resources, we have listed all the LEDs on the device. Let's test a LED by reading and writing sysfs from the command. The following commands are generic commands and are general methods of controlling LEDs .

1) View the LEDs

The directory where leds are operated is `/sys/class/leds`. The contents of the directory are as follows:

```
root@myd-yf13x:~# ls /sys/class/leds/  
blue:heartbeat
```

2) Test a LED

Take the lights.

- **Get the Status of the LED**

Where 0 means the LED is on and 1 means the LED is off.

```
root@myd-yf13x:~# cat /sys/class/leds/blue:heartbeat/brightness  
0
```

The above results show that the current blue light LED is on.

- **Turn off the LED**

```
root@myir-yf13x:~# echo 1 > /sys/class/leds/red/brightness
```

- **Turn on the LED**

```
root@myir-yf13x:~# echo 0 > /sys/class/leds/red/brightness
```

- **Trigger the LED**



After starting the "timer" trigger mode, the LED flashes at a default cycle of 1Hz with a duty ratio of 50%.

```
root@myir-yf13x:~# echo "timer" > /sys/class/leds/red/trigger
```

- **Change the on-off duty ratio when the LED lights flash**

By adjusting the "delay_on" and "delay_off" properties under the LED, the LED flicker period and on-off duty cycle can be adjusted, for example:

```
root@myir-yf13x:~# echo "100" > /sys/class/leds/red/delay_on  
root@myir-yf13x:~# echo "500" > /sys/class/leds/red/delay_off
```




3.3. Key

The `"/dev/input/eventxx"` devices in Linux system can be used to easily debug input devices such as mice, keyboards, touchpads, etc. This section focuses on testing the key. We can use the `"hexdump"` command and the `"dmesg"` command to see if the keys work normally. MYD-YF13X has two buttons, S1 is the system reset button; S2 is a User KEY that has been configured in the device tree.

1) Device Tree Node

Open the supporting device tree file `/arch/arm/boot/dts/myb-stm32mp135f.dts`, and you can see the corresponding GPIO-Keys node:

```
Path:/
    gpio-keys {
        compatible = "gpio-keys";
        #size-cells = <0>;

        button-0 {
            label = "usr_button";
            linux,code = <KEY_ENTER>;
            interrupt-parent = <&gpioi>;
            interrupts = <1 IRQ_TYPE_EDGE_RISING>;
        };
    };
```

2) Test Keys

- View Information of the Input Device Event

```
root@myd-yf13x:~# 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=kbd event0
```



```
B: PROP=0
B: EV=3
B: KEY=10000000
```

The above results show that the corresponding device event for GPIO-Keys is "event0" .

- **Dump Information of event0**

Execute the following command and press S2. The serial port terminal will print out the following information:

```
root@myd-yf13x:~# hexdump /dev/input/event0
0000000 2729 644f d4c1 000b 0001 001c 0001 0000
0000010 2729 644f d4c1 000b 0000 0000 0000 0000
0000020 2729 644f e879 000b 0001 001c 0000 0000
0000030 2729 644f e879 000b 0000 0000 0000 0000
0000040 272a 644f 6723 0005 0001 001c 0001 0000
0000050 272a 644f 6723 0005 0000 0000 0000 0000
```

Each time S2 is pressed, the current terminal will print out the current event code value, that is, the key works normally.



3.4. RS232

This section uses the Linux API to configure the transceiver of RS232 interface on the development board. The Linux serial device driver node are generally named as `/dev/ttySTMn` ($n=0,1,2,3\dots$). N represents the serial port number in the system, "ttySTM" is the serial port device name defined by the kernel. In this section, J19 interface (namely UART5) on MYD-YF13X base board is taken as an example for testing. The numbered device node of UART5 is ttySTM2. Please follow the table below to prepare before testing:

Table3-1. RS232 Interface Configuration

	MYD-YF13X	Windows 10
Hardware Interface	RS232	USB-RS232 Module
Device Node	ttySTM2	COM12
Tools	uart_test	sscom

Connect the USB-RS232 converter to the USB Host port of the PC and connect the RXD and TXD of J19 to the TXD and RXD of the USB-RS232 converter, respectively.

1) Test Receiving Data from RS232 of Development Board

At this point, the development board works as receiver and the PC works as sender.

- **Send Data from Windows PC with SSCOM**

```
[22:46:53.902]send→◇12345678
[22:46:54.117]send→◇12345678
[22:46:54.415]send→◇12345678
[22:46:54.482]send→◇12345678
[22:46:54.661]send→◇12345678
[22:46:54.850]send→◇12345678
[22:46:55.058]send→◇12345678
[22:46:55.563]send→◇12345678
```

- **Receive Data from RS232 on Development Board**



Execute the following command on the development board to receive the data. After the execution of the command, the interrupt will enter a blocking state, waiting to receive data sent from the computer serial port.

```
root@myd-yf13x:~# ./uart_test -d /dev/ttySTM2 -b 115200
/dev/ttySTM2 RECV[8]: 12345678
/dev/ttySTM2 RECV[8]: 12345678
/dev/ttySTM2 RECV[8]: 12345678
/dev/ttySTM2 RECV[8]: 12345678
/dev/ttySTM2 RECV[8]: 12345678
/dev/ttySTM2 RECV[8]: 12345678
/dev/ttySTM2 RECV[8]: 12345678
/dev/ttySTM2 RECV[8]: 12345678
```

The data sent under Windows PC is consistent with the data received by the development board, that is, the RS232 received data of development board is correct.

2) Test Sending Data to RS232 from the Development Board

At this point, the development board works as sender and the PC works as receiver.

- **Send Data from the Development Board through RS232:**

```
root@myd-yf13x:~# ./uart_test -d /dev/ttySTM2 -b 115200 -m 1
/dev/ttySTM2 SEND: 1234567890
/dev/ttySTM2 SEND: 1234567890
/dev/ttySTM2 SEND: 1234567890
/dev/ttySTM2 SEND: 1234567890
/dev/ttySTM2 SEND: 1234567890
/dev/ttySTM2 SEND: 1234567890
/dev/ttySTM2 SEND: 1234567890
/dev/ttySTM2 SEND: 1234567890
```

When the above command is completed, the corresponding data will be sent automatically.



- **Receive Data from RS232 on Windows PC**

```
[11:49:30.366]Receive←◆1234567890  
[11:49:31.366]Receive←◆1234567890  
[11:49:32.366]Receive←◆1234567890  
[11:49:33.366]Receive←◆1234567890  
[11:49:34.367]Receive←◆1234567890  
[11:49:35.367]Receive←◆1234567890  
[11:49:36.367]Receive←◆1234567890  
[11:49:37.368]Receive←◆1234567890
```

SSCOM under Windows received data and development board sent data consistent, that is, development board RS232 sent data correctly.



3.5. RS485

This routine demonstrates how to use the Linux API to test the development board's data sending and receiving functions with RS485 device alias Serial3, so the device node is ttySTM3. Take RS485 of J19 interface as an example for testing. The hardware interface configuration is shown in the following table:

Table 3-2. RS485 Interface Configuration

	MYD-YF13X	Windows 10
Hardware Interface	RS485	USB-RS485
Device Node	ttySTM1	COM12
Tools	rs485_read、rs485_write	sscom

Connect the USB-RS485 converter to the USB Host port of the PC and connect the 485A and 485B of J19 to the 485A and 485B of the USB-RS485 converter, respectively.

1) Receive data from RS485 on development board

- **Send Data from SSCOM on Windows**

```
[20:01:05.903]Send→◇12345678
[20:01:06.734]Send→◇12345678
[20:01:12.231]Send→◇12345678
[20:01:12.752]Send→◇12345678
[20:01:13.171]Send→◇12345678
[20:01:13.426]Send→◇12345678
[20:01:13.699]Send→◇12345678
[20:01:13.940]Send→◇12345678
```

- **Set receiving mode**

```
root@myd-yf13x:~# echo 60 > /sys/class/gpio/export
root@myd-yf13x:~# echo out > /sys/class/gpio/PD12/direction
root@myd-yf13x:~# echo 0 > /sys/class/gpio/PD12/value
```

- **Receive Data on the Development Board**



```
root@myir:/home# ./rs485_read -d /dev/ttySTM1 -b 115200 -e 1
RECV[-163754450]: 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
RECV[-163754450]: 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
RECV[-163754450]: 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
RECV[-163754450]: 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
RECV[-163754450]: 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
RECV[-163754450]: 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
RECV[-163754450]: 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
RECV[-163754450]: 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
```

Send data under Windows and development board received data consistent, that is, development board RS485 received data correctly.

2) Send Data from RS485 on Development Board

- **Set send mode**

```
root@myd-yf13x:~# echo 60 > /sys/class/gpio/export
root@myd-yf13x:~# echo out > /sys/class/gpio/PD12/direction
root@myd-yf13x:~# echo 1 > /sys/class/gpio/PD12/value
```

- **Send Data on Development Board**

```
root@myir-yf13x:~# ./rs485_write -d /dev/ttySTM1 -b 115200 -e 1
SEND[08]: 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
SEND[08]: 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
SEND[08]: 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
SEND[08]: 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
SEND[08]: 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
SEND[08]: 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
SEND[08]: 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
SEND[08]: 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38
```

The application automatically sends the data when the above command completes.

- **Receive Data on Windows PC**



```
[20:27:27.972]Receive←◆31 32 33 34 35 36 37 38
[20:27:28.972]Receive←◆31 32 33 34 35 36 37 38
[20:27:29.971]Receive←◆31 32 33 34 35 36 37 38
[20:27:30.972]Receive←◆31 32 33 34 35 36 37 38
[20:27:31.971]Receive←◆31 32 33 34 35 36 37 38
[20:27:32.972]Receive←◆31 32 33 34 35 36 37 38
[20:27:33.972]Receive←◆31 32 33 34 35 36 37 38
[20:27:33.972]Receive←◆31 32 33 34 35 36 37 38
```

Under Windows received data and development board sent data consistent, that is, development board RS485 sent data correctly.



3.6. CAN

This section uses “cansend” and “candump” commands in Linux system for SocketCAN communication test. The test here uses two development boards for docking tests.

The CANH and CANL pins of the J19 are connected to CANH and CANL of the same type.

1) Initializes the CAN network interface

- **Set the CAN Baud Rate**

To use canfd, set the baud rate and enable the CAN network interface. Refer to the following commands to set the arbitration baud rate of the two development boards to 5KHz and the data baud rate to 4M respectively, and enable the canfd function: (both development boards need to do the following operations)

```
root@myd-yf13x:~# ifconfig can0 down
root@myir-yf13x:~# ip link set can0 up type can bitrate 500000 sample-point
0.75 dbitrate 4000000 dsample-point 0.8 fd on
[ 125.963519] m_can_platform 4400f000.can can0: bitrate error 3.8%
[ 125.984407] IPv6: ADDRCONF(NETDEV_CHANGE): can0: link becomes ready
root@myir-yf13x:~# ifconfig can0 up
```

At this point, you can use the CANFD function.

2) Send and Receive Data

- **Send Data**

Set one of the boards to send and use “cansend” to send a 64-byte fixed format string to test:

```
root@myd-yf13x:~# cansend can0 100#11.22.33.44
root@myd-yf13x:~# cansend can0 100#11.22.33.44
root@myd-yf13x:~# cansend can0 100#11.22.33.44
root@myd-yf13x:~# cansend can0 100#11.22.33.44
```



```
root@myd-yf13x:~# cansend can0 100#11.22.33.44
root@myd-yf13x:~# cansend can0 100#11.22.33.44
```

● Receive Data

Set another board to receive. Candump CAN be used to view CAN's receiving data:

```
root@myd-yf13x:~# candump can0 -L
(1712819690.336718) can0 100#11223344
(1712819693.324284) can0 100#11223344
(1712819693.735206) can0 100#11223344
(1712819694.134187) can0 100#11223344
(1712819694.526169) can0 100#11223344
(1712819695.004645) can0 100#11223344
```

3) Statistics of can0 information

After receiving and sending CAN data, it displays the details and statistics of CAN device. The value of "clock" stands for CAN clock, "drop" stands for lost packets, and "overrun" stands for bus overrun, "error" stands for bus error.

```
root@myd-yf13x:~# ip -details -statistics link show can0
2: can0: <NOARP,UP,LOWER_UP,ECHO> mtu 72 qdisc pfifo_fast state UP mode
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 500000 sample-point 0.750
        tq 20 prop-seg 37 phase-seg1 37 phase-seg2 25 sjw 1 brp 1
        m_can: tseg1 2..256 tseg2 2..128 sjw 1..128 brp 1..512 brp_inc 1
        dbitrates 3846153 dsample-point 0.769
        dtq 20 dprop-seg 4 dphase-seg1 5 dphase-seg2 3 dsjw 1 dbrp 1
        m_can: dtseg1 1..32 dtseg2 1..16 dsjw 1..16 dbrp 1..32 dbrp_inc 1
        clock 50000000
        re-started bus-errors arbit-lost error-warn error-pass bus-off
            0          0          0          0          0          0          nu
mtxqueues 1 gso_max_size 65536 gso_max_segs 65535 parentbus platform pa
rentdev 4400f000.can
```



RX:	bytes	packets	errors	dropped	missed	mcast
	0	0	0	0	0	0
TX:	bytes	packets	errors	dropped	carrier	collsns
	24	6	0	0	0	0

3.7. USB

This section verifies the feasibility of the USB Host driver through relevant commands or hot plug and USB HUB, and realizes the function of reading and writing USB flash disk and USB enumeration function.

1) Check the Kernel Message of USB

- View the USB Device Information

Connect the U disk to the USB Host interface of the development board, and the kernel log is as follows:

```
root@myd-yf13x:~#
[ 1431.459516] usb 1-1.1: new high-speed USB device number 3 using ehci-pl
atform
[ 1431.719887] usb-storage 1-1.1:1.0: USB Mass Storage device detected
[ 1431.743089] scsi host0: usb-storage 1-1.1:1.0
[ 1432.810835] scsi 0:0:0:0: Direct-Access    TU100    128GB thinkplus  0000
PQ: 0 ANSI: 4
[ 1432.837064] sd 0:0:0:0: [sda] 245760001 512-byte logical blocks: (126 GB/1
17 GiB)
[ 1432.845080] sd 0:0:0:0: Attached scsi generic sg0 type 0
[ 1432.863871] sd 0:0:0:0: [sda] Write Protect is off
[ 1432.890792] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, do
esn't support DPO or FUA
[ 1432.933455] sda: sda1
[ 1432.941359] sd 0:0:0:0: [sda] Attached SCSI removable disk
```

From the above information it can be concluded that the device to be assigned is sda1.



2) Mount and Read/Write the USB Flash Disk

- **Mount**

```
root@myd-yf13x:/# mount /dev/sda1/mnt/
```

- **Read USB Flash Disk**

You need to create a test.txt file on your USB drive in advance.

```
root@myd-yf13x:/# ls /mnt
test.txt
root@myd-yf13x:/# cat /mnt/test.txt
hello world!
```

- **Write USB Flash Disk**

```
root@myd-yf13x:/# touch test.txt
root@myd-yf13x:/# echo "hello world !!!" > test.txt
root@myd-yf13x:/# cp test.txt /mnt
root@myd-yf13x:/# cat /mnt/test.txt
hello world !!!
```

After the file is written, the sync command needs to be executed to ensure that the data is fully written to the USB disk before the device can be unmounted.

3) Unmount USB Flash Disk

- **Unmount Device**

```
root@myd-yf13x:/# umount /mnt
root@myd-yf13x:~# [ 1666.532207] usb 1-1.1: USB disconnect, device number
5
```

Unmount before you unplug the USB flash drive, otherwise the data in the USB flash disk may be corrupted.



3.8. Micro SD card

Micro SD card, formerly known as trans flash card (TF Card), is a very small flash memory card. Compared with the standard SD card, micro SD card is smaller in shape, and it is the smallest SD card in the type of SD card. Although the size and interface shape of micro SD card are different from the original SD card, the interface specification remains unchanged to ensure compatibility. If the micro SD card is inserted into a specific adapter card, it can be used as a standard SD card. The SD card has become the most widely used memory card in consumer digital devices. It has the characteristics of large capacity, high performance and security. There are usually 9 pins on the back of micro SD card, including 4 data lines, supporting 1 bit / 4 bit data transmission width.

The STM32MP135 supports two 8bit SDMMC interfaces. The MYD-YF135 development board uses SDMMC1 to connect to the MicroSD.

1) Check SD Card Capacity

The SD Card partition information and capacity can be queried through the "fdisk -l" command as follows.

```
root@myir-yf13x:~# fdisk -l
略
Disk /dev/mmcblk0: 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: gpt
Disk identifier: 432D1A65-71C3-41A5-BA22-86A16067C169
```

Device	Start	End	Sectors	Size	Type
/dev/mmcblk0p1	34	545	512	256K	Linux reserved
/dev/mmcblk0p2	546	1057	512	256K	Linux reserved
/dev/mmcblk0p3	1058	1569	512	256K	Linux reserved
/dev/mmcblk0p4	1570	2081	512	256K	Linux reserved



/dev/mmcblk0p5	2082	10273	8192	4M	unknown
/dev/mmcblk0p6	10274	18465	8192	4M	unknown
/dev/mmcblk0p7	18466	19489	1024	512K	Linux reserved
/dev/mmcblk0p8	19490	150561	131072	64M	Linux filesystem
/dev/mmcblk0p9	150562	183329	32768	16M	Linux filesystem
/dev/mmcblk0p10	183330	11197985	11014656	5.3G	Linux filesystem
/dev/mmcblk0p11	11197986	13385694	2187709	1G	Linux filesystem

- /dev/mmcblk0p1 : The first stage boot loader arm trusted firmware (TF-A) or u-boot secondary loader (SPL)
- /dev/mmcblk0p2 : Backup of the first stage boot loader (TF-A)
- /dev/mmcblk0p5 : The second stage bootloader (ssbl). Usually refers to the u-boot image
- /dev/mmcblk0p7 : Used to store uboot env
- /dev/mmcblk0p8 : Used to store kernel resource
- /dev/mmcblk0p9 : Used to store third-party library files
- /dev/mmcblk0p10 : Used to store the root file system
- /dev/mmcblk0p11 : Can be used by users to use partitions

2) View the SD Card Partition Information

By “df” command, you can query SD Card partition information, usage, mount directory and other information.

```
root@myd-yf13x:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        187M   0  187M   0% /dev
/dev/mmcblk0p10  4.9G  2.2G  2.5G  47% /
tmpfs           221M  172K  221M   1% /dev/shm
tmpfs           89M   8.8M   80M  10% /run
tmpfs           4.0M    0   4.0M   0% /sys/fs/cgroup
tmpfs           221M   20K  221M   1% /tmp
/dev/mmcblk0p8   55M   12M   39M  24% /boot
/dev/mmcblk0p9   14M   24K   13M   1% /vendor
```



```
tmpfs          221M  104K  221M   1% /var/volatile
/dev/mmcblk0p11 995M   36M  902M   4% /usr/local
tmpfs          45M    0   45M   0% /run/user/1000
tmpfs          45M    0   45M   0% /run/user/0
```

- tmpfs : Memory virtual file system, mounted to a different directory
- devtmpfs : Used to create dev for the system
- /dev/mmcblk0p8: Partition used to store kernel resources and start scripts, which is mounted in the /boot directory
- /dev/mmcblk0p10 : Used to store the root file system
/dev/mmcblk0p11: single partition and can be used for users to use, mounted in/usr/local

3) SD card Performance Test

The performance test mainly tests the reading and writing speed of files by TF card in Linux system. The test is generally combined with the dual command of “time” and “dd” .

● Write Performance

```
root@myd-yf13x:/mnt# time dd if=/dev/zero of=/usr/local/tempfile bs=1M co
unt=100 conv=fdatasync
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 12.8163 s, 8.2 MB/s
real    0m 12.82s
user    0m 0.00s
sys     0m 1.59s
```

When you use the dd command to write a file, you need to add the “conv = fdatasync” parameter, which means that after using the dd command to write multiple times, it will be synchronized to the disk(flush cache). Because the disk is usually written to the cache first and then returned before it is written to the disk. Here we test that the write disk speed is 8.2MB / s.

● Read Performance



In embedded system, it is often necessary to test the system's performance of reading and writing files. If you want to test reading files directly from disk, you need to ignore the impact of cache. In this case, you can specify the parameter "iflag = direct, Nonblock" .

```
root@myd-yf13x:/mnt# time dd if=/usr/local/tempfile of=/dev/null bs=1M count=100 iflag=direct,nonblock
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 4.44917 s, 23.6 MB/s
real    0m 4.45s
user    0m 0.00s
sys     0m 0.04s
```

The test results show that reading speed without cache is 23.6MB/s.



3.9. ADC

MYD-YF13X provides two ADC channels, both of which provide a default voltage value. ADC testing is implemented through the file system sysfs interface. The following examples illustrate ADC channels 2 and 8.

1) View the SYSFS interface of the ADC

Use the command to view the ADC read interface:

```
root@myd-yf13x:/mnt# grep -H "" /sys/bus/iio/devices/*/name | grep adc  
/sys/bus/iio/devices/iio:device0/name:48003000.adc:adc@0
```

2) Get ADC Value

- Read the channel 2 value

```
root@myd-yf13x:~# cat /sys/bus/iio/devices/iio:device0/in_voltage2_raw  
3151
```

Divide the above value by 1000 to get the measured voltage value.

- Read the channel 8 value

```
root@myd-yf13x:~# cat /sys/bus/iio/devices/iio:device0/in_voltage8_raw  
2043
```

Divide the above value by 1000 to get the measured voltage value.

3.10. Display

The MYD-YF13X development board supports HDMI and LCD display solutions:

- HDMI display: as there is no HDMI STM32MP135CPU relevant controller, so the HDMI is by displaying the conversion chip SII9022ACNU convert RGB to HDMI signal output, support the maximum resolution of 1280 x 720 @ 60 FPS.



- LCD display: The LCD test section will demonstrate the operation of drm device on Linux to realize LCD output display RGB color and color synthesis test. The LCD adopts RGB888 display mode and supports the MYiR 7 "capacitor screen.

The display can be switched by selecting the corresponding device tree file in uboot stage. The device tree information list of MYD-YF13X is as follows:

Table 3-3.Device tree information list

Device Tree	Description
myb-stm32mp135x-512m	Contains the most basic device tree and adds the LCD display description. The default device tree for factory image is this device tree.
myb-stm32mp135x-512m-hdmi	Contains the most basic device tree, and adds the HD MI display description.

This section will introduce the switching method of uboot display scheme and the test mode of display.

1) Display scheme selection

● HDMI display selection

If you need the HDMI display function, use the RGB to HDMI module listed in Table 1-1. After the connection is complete, start the development board, select 2 in boot mode during the uboot log phase, and press Enter.

Select the boot mode

```
1:      OpenSTLinux
2:      myb-stm32mp135x-512m-hdmi
3:      myb-stm32mp135x-512m
Enter choice: 2
```

● LCD display selection

MYD-YF13X ADAPTS to the LCD screen model of MY-TFT070CV2. The screen is a 7 "capacitive touch screen and a color active matrix thin film transistor (TFT) liquid crystal display (LCD). The screen is first connected to the J15 LCD interface on the baseplate by a 50pin flexible connector.



Select the boot mode

- 1: OpenSTLinux
- 2: myb-stm32mp135x-512m-hdmi
- 3: myb-stm32mp135x-512m

Enter choice: 3

3.11. Touch Panel

The general touch functions include capacitive touch and resistive touch. The hardware of MYD-YF13X series development board does not support resistance touch at present, but supports capacitive touch. Please refer to the LCD screen in Table 1-1 for the test accessories. Users can purchase accessories by themselves according to actual requirements. The capacitive screen is sensitive in use and seldom has problems. In addition, the capacitive touch screen does not need to be accurate. The following is a simple example of testing the touch function of capacitive screen through the evtest command.

1) Touch screen connection

Connect MY-TFT070CV2 touch LCD to the development board according to section 3.11.

2) Touch test with evtest command

The terminal executes "evtest" to enter the test interface. Select the test peripheral as the touch screen, where the default is input interrupt 0. Select "0" in the test interface and press enter to start the test:

```
root@myd-yf13x:~# 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 0

Min 0

Max 1023

Event code 1 (ABS_Y)

Value 0

Min 0

Max 599

Event code 47 (ABS_MT_SLOT)

Value 0

Min 0

Max 4

Event code 53 (ABS_MT_POSITION_X)

Value 0

Min 0

Max 1023

Event code 54 (ABS_MT_POSITION_Y)

Value 0

Min 0

Max 599

Event code 57 (ABS_MT_TRACKING_ID)

Value 0

Min 0

Max 65535

Properties:

Property type 1 (INPUT_PROP_DIRECT)

Testing ... (interrupt to exit)



It can be seen that after the touch screen is connected, it will be recognized as an input interface, and the event number of the touch screen is 1.

3) Print test information

Click the touch screen, and the terminal will print the corresponding information. For example, they will output the event information reported by the touch driver to console as follows:

```
Event: time 1651169395.095608, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID), value 0
Event: time 1651169395.095608, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 445
Event: time 1651169395.095608, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 154
Event: time 1651169395.095608, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 1
Event: time 1651169395.095608, type 3 (EV_ABS), code 0 (ABS_X), value 445
Event: time 1651169395.095608, type 3 (EV_ABS), code 1 (ABS_Y), value 154
Event: time 1651169395.095608, ----- SYN_REPORT -----
Event: time 1651169395.111154, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 447
Event: time 1651169395.111154, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 150
Event: time 1651169395.111154, type 3 (EV_ABS), code 0 (ABS_X), value 447
Event: time 1651169395.111154, type 3 (EV_ABS), code 1 (ABS_Y), value 150
Event: time 1651169395.111154, ----- SYN_REPORT -----
Event: time 1651169395.123595, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 445
Event: time 1651169395.123595, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 151
Event: time 1651169395.123595, type 3 (EV_ABS), code 0 (ABS_X), value 445
Event: time 1651169395.123595, type 3 (EV_ABS), code 1 (ABS_Y), value 151
Event: time 1651169395.123595, ----- SYN_REPORT -----
```



```
Event: time 1651169395.137485, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 446
Event: time 1651169395.137485, type 3 (EV_ABS), code 0 (ABS_X), value 446
Event: time 1651169395.137485, ----- SYN_REPORT -----
Event: time 1651169395.156492, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 445
Event: time 1651169395.156492, type 3 (EV_ABS), code 0 (ABS_X), value 445
Event: time 1651169395.156492, ----- SYN_REPORT -----
Event: time 1651169395.172975, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID), value -1
Event: time 1651169395.172975, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 0
Event: time 1651169395.172975, ----- SYN_REPORT -----)
```

From the above information, it mainly displays coordinate values and key values, the specific information is as follows:

- EV_SYN: Synchronous events
- EV_KEY: Key event, such as BTN_Touch means touch button
- EV_ABS: Absolute coordinates, such as those reported by touch screen
- BTN_TOUCH: Touch button
- ABS_MT_TRACKING_ID: Represents the beginning of the collection of information. And then another ABS_MT_TRACKING_ID indicates the end of the collection

A single touch message is carried in ABS and sent in certain order, such as:

- ABS_X: X absolute coordinates relative to the screen
- ABS_Y: Y absolute coordinates relative to the screen

But multi-touch information is carried in ABS_MT and sent in a certain, such as:

- ABS_MT_POSITION_X: Represents the center point x coordinate position of the screen contact surface
- ABS_MT_POSITION_Y: coordinate position representing the center point Y screen contact





4. Network Interface

The MYD-YF13X development board contains two Gigabit Ethernet interfaces. The following describes how to configure network peripheral interfaces.

4.1. Ethernet

Under Linux, there are many tools for network configuration, such as net-tools, Iproute2, Systemd-Networkd, Network Manager and Connman, etc. All of these can be selected according to actual needs during system customization. Here, several commonly used methods of Ethernet manual temporary configuration and automatic permanent configuration are introduced.

1) Configure Ethernet IP addresses Manually and Temporarily

- Use ifconfig to Configure the Network Manually

Firstly, check the network device information through ifconfig command as follows:

```
root@myd-yf13x:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 42:0F:18:CC:11:F9
          inet addr:192.168.40.119  Bcast:192.168.40.255  Mask:255.255.255.0
          inet6 addr: fe80::400f:18ff:fecc:11f9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:289 errors:0 dropped:32 overruns:0 frame:0
          TX packets:72 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:39356 (38.4 KiB)  TX bytes:11393 (11.1 KiB)
          Interrupt:44 Base address:0x8000
```

Eth0 is the actual Ethernet device and defaults to a random hardware MAC address. If the user needs to write the MAC address himself, refer to the official ST Wiki description:



[https://wiki.st.com/stm32mpu/wiki/How to update OTP with U-Boot#MAC address example](https://wiki.st.com/stm32mpu/wiki/How_to_update_OTP_with_U-Boot#MAC_address_example)

Eth0 is given the following command to manually configure the IP address 192.168.0.100:

```
# ifconfig eth0 192.168.0.100 netmask 255.255.255.0 up
```

The command above manually configured eth0 with the IP address of 192.168.0.100, the subnet mask of 255.255.255.0, and the default configured broadcast address of 192.168.0.255, and activated with the up parameter, as shown below:

```
root@myd-yf13x:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 42:0F:18:CC:11:F9
          inet addr:192.168.0.100  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::400f:18ff:fecc:11f9/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:840 errors:0 dropped:75 overruns:0 frame:0
          TX packets:118 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:113293 (110.6 KiB)  TX bytes:19267 (18.8 KiB)
          Interrupt:44 Base address:0x8000
```

- **Configure the Network Manually by iproute2**

The ifconfig command that sets the IP address manually can also be substituted using “ip addr” and “ip link”, for more information about iproute2, please refer to <https://wiki.linuxfoundation.org/networking/iproute2> for detail.

```
root@myd-yf13x:~# ip addr flush dev eth0
root@myd-yf13x:~# ip addr add 192.168.0.101/24 brd + dev eth0
root@myd-yf13x:~# ip link set eth0 up
```

If an IP address has been previously configured, the IP address configured using “ip addr add” will become the secondary address, so use “ip addr flush” to clear the previous address before configuring and activating it. Once configured, view the eth0 information from the IP Addr Show command as follows:



```
root@myd-yf13x:~# ip addr show eth0
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 42:0f:18:cc:11:f9 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) Configure Ethernet Automatically and Permanently

IP addresses configured through the “ifconfig” and “ip” commands are lost when power is lost, and if you need to make IP addresses permanent, you need to modify the corresponding configuration files of the network management tool.

- **Configure IP Addresses by systemd-networkd Dynamically**

MYD-YF13X default factory image adopts systemd-networkd for network configuration, and for details, you can check the configuration file 80-Wired.net Work in /lib/systemd/network/ directory.

```
root@myd-yf13x:~# cat /lib/systemd/network/80-wired.network
[Match]
Type=ether
Name=!veth*
KernelCommandLine=!nfsroot
KernelCommandLine=!ip

[Network]
DHCP=yes

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

The configuration file above will configure the network matching en* and eth*. If the kernel boot parameter does not contain the nfsroot parameter, then the DHCP will automatically configure the IP address, gateway, DNS and other

information for the matched network card. See the following connection for detailed configuration parameters:

<https://www.freedesktop.org/software/systemd/man/systemd.network.html>.

- **Configure Static IP Addresses by systemd-networkd Automatically and Permanently**

If you want to configure a permanent IP address for eth0, you can write another *50-static.network* file to place in */etc/systemd/network/* directory. The *50-static.network* contents are as follows:

```
[Match]
Name=eth0
[Network]
Address=192.168.30.100/24
Gateway=192.168.30.1
```

After configuring, restart *systemd-networkd.service* and you will see that the eth0 network address has been configured to 192.168.30.100, as shown below:

```
root@myd-yf13x:~# systemctl restart systemd-networkd.service
root@myd-yf13x:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 26:13:52:CE:15:40
          inet addr:192.168.30.100  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:238239 errors:0 dropped:752 overruns:0 frame:0
          TX packets:4930 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:17797368 (16.9 MiB)  TX bytes:809419 (790.4 KiB)
          Interrupt:51 Base address:0x8000
```

4.2. 4G



Linux devices can also be connected to the external 4G module for dial-up Internet access. MYD-YF13X development board uses the M5700 4G module. The following is a simple test of the 4G module.

1). Check VID and PID

```
root@myd-yf13x:~# lsusb
Bus 001 Device 003: ID 1782:4d11 Spreadtrum Communications Inc. M5700
Bus 001 Device 002: ID 1a40:0101 Terminus Technology Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

1782:4d11: Information about VID and PID of M5700.

2).Check the kernel identification module

If the kernel adds VID and PID configuration for this module, then /dev/ttyUSB* node will be generated:

```
root@myd-yf13x:~# ls -l /dev/ttyUSB*
crw-rw---- 1 root dialout 188, 0 Apr 28 18:30 /dev/ttyUSB0
crw-rw---- 1 root dialout 188, 1 Apr 28 18:30 /dev/ttyUSB1
crw-rw---- 1 root dialout 188, 2 Apr 28 18:30 /dev/ttyUSB2
crw-rw---- 1 root dialout 188, 3 Apr 28 18:30 /dev/ttyUSB3
crw-rw---- 1 root dialout 188, 4 Apr 28 18:30 /dev/ttyUSB4
crw-rw---- 1 root dialout 188, 5 Apr 28 18:30 /dev/ttyUSB5
crw-rw---- 1 root dialout 188, 6 Apr 28 18:30 /dev/ttyUSB6
crw-rw---- 1 root dialout 188, 7 Apr 28 18:30 /dev/ttyUSB7
```

3).Use AT instruction for preliminary test

AT command can be used to conveniently query signal strength, whether to insert a SIM card, whether the SIM card is currently searching for the operator, or to make phone calls to test the function of the current card. AT communication here also need to know which device is the communication port, here need to query module file, M5700 uses ttyUSB5 for AT communication. The example here is microcom, or minicom. For example, microcom /dev/ttyUSB5 enters the mode and ctrl+x exits.



```
root@myd-yf13x:~# microcom /dev/ttyUSB5
at
OK
```

- **Query signal quality**

```
at+csq
+CSQ: 12,99

OK
```

12, 99: 12 is signal quality, the number should be between 0 and 31 (99 for no signal), the higher the number, the better the signal quality.

- **Check whether the card is recognized**

```
at+cops?
+COPS: 0,1,"UNICOM",7

OK
```

CPIN:READY :READY indicates ready

- **View carrier**

```
at+cops?
+COPS: 0,1,"UNICOM",7

OK
```

UNICOM,7: UNICOM stands for Unicom,7 stands for 2G, 3G, 4G, or 5G according to the module manual.

4). Use the ecm script to dial

```
root@myd-yf13x:~# ecm
AT+SYSNV=1,"usbmode",5

OK
```



```
AT+CGDCONT=1,"IP","CMNET"
```

```
OK
```

```
AT+CGACT=1,1
```

```
+CGACT: 1, 1, 10.250.181.217
```

```
OK
```

```
IP 10.250.181.217 available.
```

```
udhcpc: started, v1.35.0
```

```
udhcpc: broadcasting discover
```

```
udhcpc: broadcasting select for 10.250.181.217, server 192.168.1.1
```

```
udhcpc: lease of 10.250.181.217 obtained from 192.168.1.1, lease time 30840
```

```
RTNETLINK answers: File exists
```

```
/etc/udhcpc.d/50default: Adding DNS 218.104.111.122
```

5). Ping the Internet test

```
root@myd-yf13x:~# ping www.baidu.com -I eth2
```

```
PING www.a.shifen.com (112.80.248.75) from 172.22.188.79 eth2: 56(84) bytes  
of data.
```

```
64 bytes from 112.80.248.75 (112.80.248.75): icmp_seq=1 ttl=55 time=42.2 ms
```

```
64 bytes from 112.80.248.75 (112.80.248.75): icmp_seq=2 ttl=55 time=30.7 ms
```

```
64 bytes from 112.80.248.75 (112.80.248.75): icmp_seq=3 ttl=55 time=25.7 ms
```

```
64 bytes from 112.80.248.75 (112.80.248.75): icmp_seq=4 ttl=55 time=37.8 ms
```

```
^C
```

```
--- www.a.shifen.com ping statistics ---
```

```
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
```

```
rtt min/avg/max/mdev = 25.709/34.114/42.211/6.350 ms
```



5. Network Applications

The “myir-image-full” image of the device programmed from the factory contains some common network applications by default, which is convenient for users to develop or debug.

5.1. PING

PING is used primarily to test network connectivity. It can also test network latency and packet loss rates. Once the Ethernet connection is configured as in 4.1.1, PING can be used for simple testing of network connections.

1) Network Connection

When you connect a device to a switch or router via a CAT6 cable, the console displays the connection information that the kernel outputs, as follows:

```
root@myd-yf13x:~#  
[ 41.932837] stm32-dwmac 5800e000.eth2 eth1: Link is Up - 1Gbps/Full - flow control off  
[ 41.939390] IPv6: ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready
```

2) PING Public Network

```
root@myd-yf13x:~# 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=12.672 ms  
64 bytes from 112.80.248.76: seq=1 ttl=55 time=12.784 ms  
64 bytes from 112.80.248.76: seq=2 ttl=55 time=13.136 ms  
64 bytes from 112.80.248.76: seq=3 ttl=55 time=12.750 ms  
^C  
--- www.baidu.com ping statistics ---  
4 packets transmitted, 4 packets received, 0% packet loss  
round-trip min/avg/max = 12.672/12.835/13.136 ms
```



NOTE: Ping public network needs to ensure that DNS is working properly.

The above results show that the IP address of "www.baidu.com" after domain name resolution is 112.80.248.76, ICMP_SEQ stands for the number of ICMP packet, if the number is serial, no packet is lost; Time stands for the latency of the response, and the shorter the better, of course.

5.2. SSH

SSH, short for Secure Shell, is developed by the Network Working Group of the IETF. SSH is a secure protocol based on the application layer. It is reliable and provides security for remote login sessions and other network services. Generally, dropbear or OpenSSH is used on Linux to implement SSH servers and clients. Let's test the use of the SSH client and server on an Ethernet connection. The current factory default contains the client and service program provided by openssh v2020.81.

Configure the connection between the Ethernet interface on the development board and the SSH server. The IP address of the Ethernet card is as follows:

```
root@myd-yf13x:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 7A:F8:25:74:24:DF
          inet addr:192.168.40.220  Bcast:192.168.40.255  Mask:255.255.255.0
          inet6 addr: fe80::78f8:25ff:fe74:24df/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:428 errors:0 dropped:67 overruns:0 frame:0
          TX packets:99 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:65361 (63.8 KiB)  TX bytes:13172 (12.8 KiB)
          Interrupt:45 Base address:0x8000
```

The SSH server's IP address is 192.168.0.2. The following test can be conducted after the connection between the test device and SSH server is normal with the ping command.

- **SSH Client Test**



The device connects to the SSH server as an SSH client. The SSH command is used to log in the SSH server on the device. The commands and results are as follows:

```
root@myd-yf13x:~# ssh sur@192.168.40.242

Host '192.168.40.242' is not in the trusted hosts file.
(ssh-ed25519 fingerprint sha1!! 02:e2:70:b5:17:64:42:f3:26:9e:74:f3:9f:2f:03:fc:e4:06:d6:24)
Do you want to continue connecting? (y/n) y
sur@192.168.40.242's password:
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-148-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

37 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Sun May  7 15:07:23 2023 from 192.168.40.122
sur@myir:~$
```

sur indicates the user name on the server.

After a successful login, the user automatically enters the console console on the SSH server. Then, the user can control the remote server within the sur permission on the client. To exit, run the "exit" command on the current console.

● SSH Server Test

The device acts as an SSH server, and other devices are connected to the device remotely.



Since SSH service is also started on the device side by default, we can also use SSH commands on other devices with SSH clients to log in to the current device, with the following commands and results:

```
sur@myir:~$ ssh root@192.168.40.220
The authenticity of host '192.168.40.220 (192.168.40.220)' can't be established.
RSA key fingerprint is SHA256:OUHiUofsoGKS7A8ELAHBCfAqy6j2V6GqkfHKEYU
9fAE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.40.220' (RSA) to the list of known hosts.
root@myd-yf13x:~#
```

In the example above, we remotely log on to the device as root and go to the console to perform root user privileges on the device. If you need to exit, simply execute the "exit" command from the console.

OpenSSH is the primary connection tool for remote login using the SSH protocol. It encrypts all traffic to eliminate eavesdropping, connection hijacking, and other attacks. In addition, OpenSSH provides a range of large secure tunneling capabilities, multiple authentication methods, and complex and flexible configuration options. Users can modify the configuration files "ssh_config" and "sshd_config" in the "/etc/ssh/" directory as they wish.

For example, if you want the SSH server to allow the root account to log in remotely without a password, you can modify "/etc/ssh/sshd_config" on the SSH server device to add the following two lines of configuration.

```
PermitRootLogin yes
PermitEmptyPasswords yes
```

The above configuration poses a significant security risk and is typically used for remote deployment during the debug phase. In real products, it's usually turned off for security reasons.



5.3. SCP

SCP, short for Secure Copy, is a Secure remote file Copy command based on the SSH protocol on Linux systems and is very useful during system debugging.

In 4.2.2, we have introduced the example of using SSH protocol and SSH client and server for remote login. Here, we introduce the example of remote copy of files through SCP command:

1) Copy Files from Remote to Local

```
PC $ scp test root@192.168.40.220:/home/root
The authenticity of host '192.168.40.101 (192.168.40.220)' can't be established.
ECDSA key fingerprint is SHA256:C6fXFGctxoRu2eBuCPe04fEcSRzI82WJ1Rbbqji
kp4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.40.101' (ECDSA) to the list of known ho
sts.
test                                100%  17MB  2.9MB/s  14:28 ETA
```

Go to the development board home directory to see this file, as shown below:

```
root@myd-yf13x:~# ls
myb-stm32mp135x-256m.dtb  myb-stm32mp135x-512m.dtb  rs485_read  rs48
5_write  tempfile  test  ulmage  watchdog_test
```

2) Copy Files form Local to Remote

```
root@myd-yf13x:~# scp test sur@192.168.40.242:~/
The authenticity of host '192.168.40.106 (192.168.40.242)' 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 '192.168.40.106' (ECDSA) to the list of known ho
sts.
sur@192.168.40.242's password:
```

test	100%	17MB	2.2MB/s	14:28 ETA
------	------	------	---------	-----------

During the copying process, please follow the prompts to input. After successful verification, the file is copied from the device to the \$HOME directory of the specified account on the server.

You can also copy directories by adding the "-r" parameter, as the SCP command helps.



5.4. TFTP

Like FTP, TFTP uses client and server software to connect and transfer files between two devices, but the difference is that TFTP uses UDP protocol, which does not have the login function. It is very simple, especially suitable for transferring and backing up firmware, configuration files and other information on the device and server side. For example, TFTP protocol is supported in the common U-boot, which can load the server-side Linux system through the network and realize the function of network startup.

The default image file contains the TFTP client program provided by BusyBox, with the following command syntax:

```
root@myd-yf13x:~# tftp --help
BusyBox v1.35.0 () multi-call binary.

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

The detailed parameters are described below:

- -g : Get
- -p : Upload/Put
- -l : Local files
- -r : Remote files
- HOST: Remote host name or IP address
- [PORT]: Optional remote host port, default is 69

Users can choose tftp-hpa on Linux, can also choose tftpd32/64 (http://tftpd32.jounin.net/tftpd32_download.html) on Windows as the TFTP server application. Now, we choose “tftpd-hpa” on Ubuntu as an example to show the TFTP server configuration.

1) Install TFTP Server Application

Install the TFTP server application on the Ubuntu server as follows:



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

2) Configure TFTP Server

Create the TFTP server working directory and open the TFTP service configuration file as follows, please replace <WORKDIR> with yourselves work directory:

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

Modify or add the following fields:

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

3) Restart TFTP Service

Restart the TFTP service on the Ubuntu server as follows:

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

Once the TFTP server is configured, place a test file "*zImage*" in the configured "*<WORKDIR>/tftpboot/*" directory, and you can download and upload files using the TFTP client on the target device.

```
root@myd-yf13x:~# tftp -g -r zImage -l zImage 192.168.0.2
```

The above command downloads the "*zImage*" from the TFTP server "*<WORKDIR>/tftpboot/*" directory to the current directory of the target device.

```
root@myd-yf13x:~# tftp -p -l config -r config_01 192.168.0.2
```

The above command will upload the "*config*" file from the current directory on the target device to the directory previously configured on the TFTP server, under "*<WORKDIR>/tftpboot/*" and rename it to "*config_01*".



5.5. DHCP

DHCP (Dynamic Host Configuration Protocol) is a network protocol for local area networks. The IP address range is controlled by the server. Clients can automatically obtain the IP address and subnet mask assigned by the server when they log in to the server.

DHCP also has two roles: server and client. This topic describes how to use the `dhclient` and `udhcpc` commands to manually obtain IP addresses for network debugging.

Connect the development board to the router using the CAT6 network cable, manually assign an IP address to the `eth0` network adapter, and observe the ip acquisition process by `dhcpc`.

According to the above test log, the following four processes are observed: DHCP Discover, DHCP Offer, DHCP Request, and DHCP ACK. Press `Ctrl+c` to end the DHCP test.

- **Configure the IP Address Dynamically by `udhcpc`**

```
root@myd-yf13x:~# udhcpc -i eth0
udhcpc: started, v1.35.0
udhcpc: broadcasting discover
udhcpc: broadcasting select for 192.168.40.220, server 192.168.40.1
udhcpc: lease of 192.168.40.220 obtained from 192.168.40.1, lease time 7200
/etc/udhcpc.d/50default: Adding DNS 192.168.40.1
```

Either way, you can configure `eth0` with the IP address, gateway, subnet mask, DNS, and other information as follows:

```
root@myd-yf13x:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 7A:F8:25:74:24:DF
          inet addr:192.168.40.220  Bcast:192.168.40.255  Mask:255.255.255.0
          inet6 addr: fe80::78f8:25ff:fe74:24df/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:73960 errors:0 dropped:779 overruns:0 frame:0
```



```
TX packets:57773 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:50012427 (47.6 MiB) TX bytes:2842661 (2.7 MiB)
Interrupt:45 Base address:0x8000
```

```
root@myd-yf13x:~# cat /etc/resolv.conf
# This is /run/systemd/resolve/resolv.conf managed by man:systemd-resolved
(8).
# Do not edit.
#
# This file might be symlinked as /etc/resolv.conf. If you're looking at
# /etc/resolv.conf and seeing this text, you have followed the symlink.
#
# This is a dynamic resolv.conf file for connecting local clients directly to
# all known uplink DNS servers. This file lists all configured search domains.
#
# Third party programs should typically not access this file directly, but only
# through the symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a
# different way, replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported modes
of
# operation for /etc/resolv.conf.
nameserver 192.168.40.1
search.
```




5.6. IPTables

Iptables is an administrative tool for IPv4 packet filtering and NAT. It is used to set up, maintain, and check the IP packet filtering rule table in the Linux kernel. Several different tables can be defined. Each table contains many built-in chains and can also contain user-defined chains. Each chain is a list of rules that can match a set of packets. Each rule specifies how to handle matching packets.

Devices using Linux systems typically use the “iptables” tool to configure firewalls. “iptables” handles packets based on methods defined by packet filtering rules, such as accept, reject, and drop. Let's use “iptables” to test intercepting ICMP packets, preventing other devices on the network from ping them. Specific commands to use see: <https://linux.die.net/man/8/iptables>.

1) Configure iptables for Target Device

Use the “iptables” configuration on the target device to discard the input ICMP package and not respond to ping probes from other hosts, with the following command:

```
root@myd-yf13x:~# iptables -A INPUT -p icmp --icmp-type 8 -j DROP
root@myd-yf13x:~# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -p icmp -m icmp --icmp-type 8 -j DROP
```

2) Ping the Target Device

Ping the target device on the development host and specifying a deadline of 10, the result is shown as follows:

```
PC$ ping 192.168.0.60 -w 10
PING 192.168.0.60 (192.168.0.60) 56(84) bytes of data.

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



The above results show that the development host cannot ping the target device after setting the firewall.

3) Delete Rules for iptables

```
root@myd-yf13x:~# iptables -F
root@myd-yf13x:~# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
```

4) Ping the Target Device Again

```
PC$ ping 192.168.0.60 -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
```

Once the “iptables” rules are cleared, ping the target device again from the development host is ready to ping. The above example is just a simple demonstration, but iptables with various rules can be very powerful, and I won't go into the details here.



5.7. Ethtool

Ethtool is a tool for viewing and modifying Ethernet device parameters, which is useful during network debugging. Use this command to look at the Ethernet card information and try to modify its parameters.

We look at the help information for the command through “ethtool -h” .

```
root@myd-yf13x:~# ethtool --help
ethtool version 5.16
Usage:
    ethtool [ FLAGS ] DEVNAME      Display standard information about
device
    ethtool [ FLAGS ] -s|--change DEVNAME  Change generic options
    [ speed %d ]
    [ lanes %d ]
    [ duplex half|full ]
    [ port tp|aui|bnc|mii|fibre|da ]
    [ mdix auto|on|off ]
    [ autoneg on|off ]
    [ advertise %x:%x | mode on|off ... [--] ]
    [ phyad %d ]
    [ xcvr internal|external ]
    [ wol %d[%d] | p|u|m|b|a|g|s|f|d... ]
    [ sopass %x:%x:%x:%x:%x:%x ]
    [ msglvl %d[%d] | type on|off ... [--] ]
    [ master-slave preferred-master|preferred-slave|forced-master|fo
rced-slave ]
    ethtool [ FLAGS ] -a|--show-pause DEVNAME      Show pause optio
ns
    ethtool [ FLAGS ] -A|--pause DEVNAME      Set pause options
```

View the basic information of the current device's Ethernet as follows.

```
root@myd-yf13x:~# 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 Receive-only
Supports auto-negotiation: Yes
Supported FEC modes: Not reported
Advertised link modes:  10baseT/Half 10baseT/Full
                        100baseT/Half 100baseT/Full
                        1000baseT/Full
Advertised pause frame use: Symmetric Receive-only
Advertised auto-negotiation: Yes
Advertised FEC modes: Not reported
Speed: 1000Mb/s
Duplex: Full
Auto-negotiation: on
Port: Twisted Pair
PHYAD: 0
Transceiver: internal
MDI-X: Unknown
Supports Wake-on: ug
Wake-on: d
Current message level: 0x0000003f (63)
                        drv probe link timer ifdown ifup

Link detected: yes
```

From the “ethtool” command, you can see that the current Ethernet supports 10, 100, and gigabit half-duplex and full-duplex modes. The current connection status is negotiated gigabit, full-duplex mode, MII interface, and so on.

We can also use the “ethtool” to set Ethernet parameters, which are useful for debugging and diagnosing Ethernet, such as forcing Ethernet to be set to 100 megabytes full duplex and turning off self-negotiation, as shown below:



```
root@myd-yf13x:~# ethtool -s eth0 speed 100 duplex full autoneg off
[ 3012.546094] stm32-dwmac 5800a000.eth1 eth0: Link is Down
[ 3015.692806] stm32-dwmac 5800a000.eth1 eth0: Link is Up - 100Mbps/Full
- flow control off
```

For more information about “ethtool” , please refer to the man page of “ethtool” at: <http://man7.org/linux/man-pages/man8/ethtool.8.html> .



5.8. iPerf3

iPerf3 is a tool that actively measures the maximum realized bandwidth over an IP network. It supports tuning various parameters such as test time, buffer size, and protocols (TCP, UDP, SCTP with IPv4 and IPV6). The iPerf3 can be divided into server mode or client mode by role, and we can use it to test and view network bandwidth in TCP mode, TCP window values, probability of retransmission, etc., as well as to test packet loss rates, latency, and jitter under specified UDP bandwidth.

We used an Ubuntu 16.04 system, a computer with gigabit card as the server of iPerf3, and the device under test as the client to test the performance on TCP and UDP for the device's Ethernet card.

Install iPerf3 on the server, as follows:

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

Connect the server and device directly via CAT6 and configure their respective IP addresses. For example, we set the server IP to 192.168.0.2 and the device IP to 192.168.0.60, and use the ping command to test to make sure they are connected.

Note: Try not to connect to routers or switches to prevent test results from being affected by intermediate devices.

1) Test Performance under TCP Mode

- **Server Side (192.168.0.2)**

Iperf3 on the server uses the -s parameter to indicate that it is operating in the server mode.

```
PC $ $ iperf3 -s -i 2
```

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

- **Client Side (192.168.0.60)**

Iperf3 works on the device as client, under TCP mode, where the parameters are described below:

- -c 192.168.0.2 : Work on the client side, connect to the server side 192.168.0.2
- -i 2 : Test results are reported at an interval of 2 seconds
- -t 10 : The total test time is 10 seconds

```
## 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
[ 5]  0.00-10.00   sec    1.09 GBytes    939 Mbits/sec
iperf Done.
```

After 10 seconds, the client finished the test and displayed the above test results, indicating that the TCP bandwidth was about 940Mbps, no retransmission, and the TCP window value was 560KBytes during the test.

At the same time, the server also displays the test results as follows, and then continues to listen on port 5201 waiting for the client to connect:

```
$ 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) Test Performance under UDP Mode

● Server Side (192.168.0.2)

Continue running iPerf3 on the server and use the -s parameter to indicate that you are working in the server mode.

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

● Client Side (192.168.0.60)

Iperf3 works on the device as client, under UDP mode, where the parameters are described below:

- -u : Work in UDP mode
- -c 192.168.0.2 : Work on the client side, connect to the server side 192.168.0.2
- -i 2 : Test results are reported at an interval of 2 seconds
- -t 10 : The total test time is 10 seconds
- -b 100M : Set the UDP transmission bandwidth as 100Mbps.

```
# 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 Mbites/sec	17263
------	-----------	-----------------	----------------	-------

[5]	2.00-4.00	sec 23.8 MBytes	100 Mbites/sec	17266
------	-----------	-----------------	----------------	-------

[5]	4.00-6.00	sec 23.8 MBytes	100 Mbites/sec	17266
------	-----------	-----------------	----------------	-------

[5]	6.00-8.00	sec 23.8 MBytes	100 Mbites/sec	17263
------	-----------	-----------------	----------------	-------

[5]	8.00-10.00	sec 23.8 MBytes	100 Mbites/sec	17268
------	------------	-----------------	----------------	-------

[ID]	Interval	Transfer	Bitrate	Jitter	Lost/Total Datagrams
-------	----------	----------	---------	--------	----------------------

[5]	0.00-10.00	sec 119 MBytes	100 Mbites/sec	0.000 ms	0/86326 (0%) sender
------	------------	----------------	----------------	----------	---------------------

[5]	0.00-10.00	sec 119 MBytes	100 Mbites/sec	0.068 ms	0/86326 (0%) receiver
------	------------	----------------	----------------	----------	-----------------------

iperf Done.

The client completes the test after 10 seconds and displays the above test results, showing that UDP did not lose packets when specifying a bandwidth of 100Mbps.

At the same time, the server also displays the test results as follows, and then continues to listen on port 5201 waiting for the client to connect:

\$ \$ 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 Mbites/sec	0.079 ms	0/16918 (0%)
------	-----------	-----------------	-----------------	----------	--------------

[5]	2.00-4.00	sec 23.8 MBytes	100 Mbites/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 Mbits/sec 0.074 ms 0/17267 (0%)
[ 5] 10.00-10.04 sec 486 KBytes 100 Mbits/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 Mbits/sec 0.068 ms 0/86326 (0%)
-----
Server listening on 5201
-----
```

The client adjust the “-b” parameter and continues to increase the specified UDP bandwidth. When packet loss occurs, the UDP bandwidth measured is the actual UDP bandwidth. If the specified bandwidth reaches the Ethernet card's highest bandwidth of 1Gbps and still no packets are lost, the bandwidth returned by the iPerf3 test is the actual UDP bandwidth. For example, the following example specifies a bandwidth of 1000Mbps, still without packet loss, but the actual bandwidth is around 600Mbps.

```
# 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 Mbits/sec 16462
[ 4] 2.00-4.00 sec 143 MBytes 599 Mbits/sec 18293
[ 4] 4.00-6.00 sec 143 MBytes 599 Mbits/sec 18295
[ 4] 6.00-8.00 sec 143 MBytes 600 Mbits/sec 18300
[ 4] 8.00-10.00 sec 143 MBytes 599 Mbits/sec 18294
-----
[ ID] Interval      Transfer      Bandwidth      Jitter      Lost/Total Datagrams
[ 4] 0.00-10.00 sec 700 MBytes 587 Mbits/sec 0.080 ms 0/89643 (0%)
[ 4] Sent 89643 datagrams
iperf Done.
```

Iperf3 also has a number of parameters that can be configured during the test so that users can tailor the test to their actual application needs. For example, you



can increase the value of the “-t” parameter for long time stress test, or specify the “-P” parameter for multiple concurrent stress tests. For more information about the IPerf3 test, please refer to [https://iperf.fr/iperdoc. Ph #3doc](https://iperf.fr/iperdoc.Ph#3doc).

6. Linux Graphics System

Linux graphics system is a more complex subsystem of Linux system, has been in constant change. It sits between the underlying display related device drivers and the upper user interface applications, masking various underlying hardware differences while providing a unified API for the upper user interface.

The earliest Linux/Unix environment graphics system is Xorg graphics system, with the development of software and hardware, especially the development of embedded system, Xorg is too heavy, and then some more concise, easy to develop and maintain the graphics system, such as Wayland (<https://wayland.freedesktop.org/>). The following is the structure of a typical embedded graphics system.

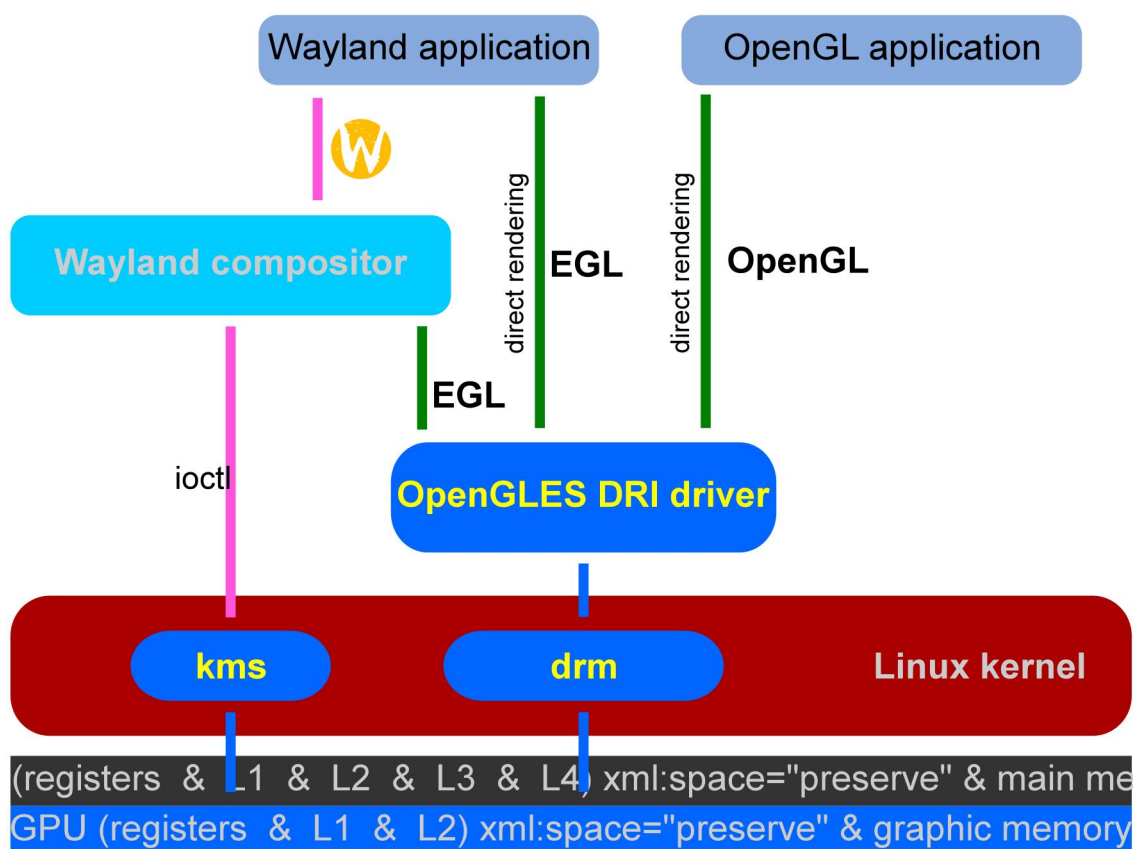


Figure 5-1. Embedded Linux Graphics Stack Overview

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



As you can see from the figure above, Wayland is not a necessary component for graphical interface development, for example MEasy HMI V2.x, which is included in our factory image, runs Qt5 directly on linuxfb with the specified platform plugin. The following sections test and explain some key aspects of the graphics system.



6.1. Wayland

This section focuses on Wayland/Weston. The Wayland/Weston framework is dedicated to the management of the display, including the composition of its content, and support for its related input device events (touch screen, mouse, keyboard...). And its Settings (background wallpaper, resolution, multi-screen...):

- Wayland is a protocol, it specifies the display server and the communication between the client application layer. Wayland is intended as an alternative to a simpler X window system that is easier to develop and maintain.
- Weston is a reference implementation of the Wayland composite application, which is a display server that uses the Wayland protocol. Weston is a minimal and fast combination that is suitable for many embedded and mobile use cases.

1) Start the Weston Display Server

```
root@myd-yf13x:~# systemctl restart weston-launch
```

6.2. QT

QT is a cross-platform C++ graphical user interface application development framework. It can be used to develop both GUI programs and non-GUI programs, such as console tools and servers. Qt is an object-oriented framework that uses special code-generation extensions and macros. Qt is easy to extend and allows for true component programming.

The development board burns systems with the Qt runtime library at delivery and provides a rich HMI demonstration system. See "MYD-YF13X_QT and MEasy HMI2.0 software development guide" for details.

1) Check Qt information

Take a look at the current QT version supported on the system, as follows:

```
root@myd-yf13x:~# qmake -v
QMake version 3.1
```

Using Qt version 5.15.3 in /usr/lib

2) Introduction of QT Running Environment

When running Qt applications, the Qt operating environment, such as platform plug-ins, display parameters, input devices and cursor pointers, can be appropriately configured according to different hardware and software requirements.

● Platform Plugins Configuration

On embedded Linux systems, you can use multiple platform plug-ins: EGLFS, LinuxFB, DirectFB, or Wayland. However, the availability of these plug-ins depends on the characteristics of the actual hardware platform and how Qt is configured. EGLFS is the default plugin on many motherboards. In the actual test, EGLFS is not suitable, so Linuxfb is used. Use the QT_QPA_PLATFORM environment variable to request another plug-in. Also, for quick tests, use the -platform command-line argument with the same syntax. For example, you can run the following command on the user console:

```
root@myir-yf13x:~# export QT_QPA_PLATFORM=linuxfb
root@myir-yf13x:~# /home/mxapp2
```

Or use the “-platform linuxfb” instruction platform plug-in, as shown below:

```
root@myir-yf13x:~# /home/mxapp2 -platform linuxfb
```

NOTE: Starting with Qt 5.0, Qt no longer has its own Windows System (QWS) implementation and therefore no longer supports the -QWS parameter used on earlier versions of Qt.

● Input Devices Configuration

When there is no window system (such as XWindow or Weston) on an embedded Linux device, the mouse, button, or touch device retrieves input device information, such as libinput or TSlib, by reading evdev directly or using other intermediate libraries. The EGLFS and LinuxFB platform plug-ins contain both types of input. The most straightforward aspect of Qt5 input device configuration is to look at the code of the platform plug-in used by Qt5, such as the code of the EGLFS plug-in input device configuration section:



linuxfb platform plug-in default is to use EvdevTouch input processor, this way is often used to deal with capacitive touch, capacitive touch driver reported event coordinates exactly correspond to the actual screen area coordinates, there is no need to do additional processing, if the reverse, you can make some adjustments through the environment variables, The EvdevTouch input handler supports the following additional parameters:

Table 5-2. Parameters of Environment Variable for EvdevTouch Input Handler

Parameters	Description
/dev/input/...	Specifies the name of the input device. If unspecified, Qt looks for a suitable device either via libudev or by traversing the available nodes.
rotate	On some touch screens the coordinates must be rotated by setting rotate to 90, 180, or 270.
invertx/inverty	Specifies the parameters to invert the X or Y coordinates in the input events.

For example, if QT_QPA_EVDEV_TOUCHSCREEN_PARAMETERS are passed with the following values to the platform plug-in before starting the application, the touch device is explicitly specified as “/dev/input/event1”, whose coordinates are flipped 180 degrees. This is useful when the orientation of the actual screen and the touch screen do not match.

```
export QT_QPA_EVDEV_TOUCHSCREEN_PARAMETERS=/dev/input/event5:rotate=180
```

If you want to enable TSLIB support, you need to set the QT_QPA_EGLFS_TSLIB (for EGLFS) or QT_QPA_FB_TSLIB (for linuxfb) environment variable to 1. Refer to TSLIB for specific usage at

<https://github.com/libts/tslib/blob/master/README.md>.

NOTE: The TSLIB input handler, often used for resistive touches, generates mouse events and supports only single touch, and requires screen calibration for initial use.

3) Start Qt Applications



Devices that typically use touch do not need to display the mouse pointer, and users can set environment variables to hide the mouse pointer before executing the application.

You are using a “linuxfb” platform plug-in, the Settings are as follows:

```
export QT_QPA_FB_HIDECURSOR=1
```

“myir-image-full” image of MYD-YF13X development board factory has MEasy HMI demo program, using EGLFS platform plug-in and EvdevTouch handler, corresponding boot command program is as follows:

```
root@myir-yf13x:~# /home/mxapp2 -platform linuxfb &
root@myir-yf13x:~# qt.qpa.input: xkbcommon not available, not performing k
ey mapping
qml: index=0
qml: currentIndex=0
qml: index=0
stop !
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
```

Mxapp2 is started by default. If you need to run your own Qt application, you need to stop MXAPP2 before starting other applications.

You can directly exit a process by using the kill command

```
root@myir-yf13x:~# killall mxapp2
```



7. Multimedia

7.1. Camera

This section uses the system's built-in gstreamer, v4l2-utils, media-ctl command to evaluate and test the MildvP ov2659 camera (model MY-CAM011B). Main test camera preview, capture frame (take photos).

1) View the Device Topology of the USB Camera

- **Query the ip address of the development board (This IP address must be able to access the Internet).**

```
root@myir-yf13x:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr AE:7D:A5:7E:C4:A2
          inet addr:192.168.40.146  Bcast:192.168.40.255  Mask:255.255.255.0
          inet6 addr: fe80::ac7d:a5ff:fe7e:c4a2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1669 errors:0 dropped:102 overruns:0 frame:0
          TX packets:440 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:243972 (238.2 KiB)  TX bytes:58552 (57.1 KiB)
          Interrupt:44 Base address:0x8000
```

- **Execute preview command**

```
root@myir-yf13x:~# uvc_stream -d /dev/video0 -y -P 123456 -r 640x480
Using V4L2 device: /dev/video0
Format: YUYV
JPEG quality: 40
Resolution: 640 x 480 @ 5 fps
TCP port: 8080 user: uvc_user pass: *****
```

- **Open the preview page**

Open the web page and type:

192.168.40.146:8080/stream.mjpeg

The entered ip address must be the same as the previous ip address. When you enter this page, you will be prompted to enter your user name and password.

- **Execute a photo command**

```
root@myir-yf13x:~# v4l2grab -d /dev/video0 -W 640 -H 480 -o 1.jpeg
Unable to set frame interval.
```

At this time, a 1.jpeg file will be generated in the current directory. After moving this file to a windows computer, you can punch in with the picture viewing software.

7.2. Audio

This is the test play audio. There are two ways to play the audio directly through the HMI2.0 app and manually through the aplay tool.

1) Play Music on MEasy HMI 2.0

For HMI multimedia music playing, please refer to the HMI manual "MEasy HMI2.0 Development Manual".

2) Adjust the volume

```
root@myir-yf13x:~# amixer cset name='PCM Playback Volume' 180
numid=1,iface=MIXER,name='PCM Playback Volume'
; type=INTEGER,access=rw---R--,values=2,min=0,max=192,step=0
: values=192,192
amixer: Control default element TLV read error: No such device or address
```

From the above information, you can see that the minimum volume is 0 and the maximum volume is 192. Set the volume to 180

3) Play Music by Command

- **Play wav**

```
root@myir-yf13x:~# aplay 01+Singalongsong.wav
```

Playing WAVE '01+Singalongsong.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo

7.3. Video

This section mainly tests the two ways to play the video, one is direct HMI click to play the video, the other is to play the video by command.

1) Play Videos on MEasy HMI 2.0

For HMI multimedia video playback, please refer to the HMI manual "MYD-YF13X_QT and MEasy HMI2.0 software development guide".



8. System Tools

Some commonly used system tools are included in the default image of the factory, which is convenient for users to view and manage various resources of the system in system debugging or actual deployed products, and can also be called in SHELL scripts or other applications. These tools may not fully meet the needs of the user's system customization, at which point the system developer needs to make appropriate adjustments based on the actual situation.

8.1. Compress and Decompress Tools

This section mainly tests the compression and decompression tools of the system. Compression is to compress multiple files into a compression package, then it will be convenient for file transfer. And decompression will make the compressed files back to the original size for easy use. This section illustrates the file system with tools such as "tar", "gzip", "gunzip", and so on.

1) tar Tool

The "tar" tool, now commonly used in Linux, not only packages files, but also compresses, views, adds, and unzips them. Here is the package operation.

- **Syntax Format**

Use the --help parameter to see the "tar" syntax format as follows:

```
root@myir-yf13x:~# tar --help
Usage: tar [OPTION...] [FILE]...
GNU 'tar' saves many files together into a single tape or disk archive, and can
restore individual files from the archive.
```

Examples:

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

```
tar -xf archive.tar          # Extract all files from archive.tar.
```

The detailed parameters are described below:

- -c : Create a compressed file parameter instruction
- -x : Unpacking a compressed file parameter instruction
- -t : Check out the files in a tar file! Note in particular that c/x/t can only exist in one parameter. Do not exist at the same time! Because it is impossible to compress and uncompress at the same time.
- -z : Do you have gzip properties at the same time? Is gzip compression required?
- -j : Do you have a bzip2 attribute at the same time? Is bzip2 compression necessary?
- -v : Show files during compression! This is often used, but is not recommended when running in the background
- -f : Use file name, please note, the file name must be followed -f parameter immediately
- -p : Use the original properties of the original file (properties do not change depending on the user)
- -P : Can use absolute path to compress
- --exclude FILE: During compression, do not pack a FILE

● Use tar to Compress

Create a new test.txt file and type the following command to package the file in ".gz" format:

```
root@myir-yf13x:~# tar -czf test.tar.gz test.txt
root@myir-yf13x:~# ls
OpenAMP_TTY_echo.elf  rs485_write  test.tar.gz  uart_test
rs485_read            test.c       test.txt    wifi.conf
```

With the z parameter, the ".tar.gz" or ".tgz" represents a gzip-compressed tar file.

● Use tar to Decompress

Unpack the file which has been compressed as tar.gz.



```
root@myir-yf13x:~# tar -xvf test.tar.gz
test.txt
root@myir-yf13x:~# ls
OpenAMP_TTY_echo.elf  rs485_write  test.tar.gz  uart_test
rs485_read            test.c       test.txt    wifi.conf
```

2) gzip Tool

“gzip” is a file compression and decompression command frequently used in Linux systems, both convenient and easy to use.

- **Syntax Format**

Enter the following command at the development board terminal to view the “gzip” syntax:

```
root@myir-yf13x:~# gzip --help
BusyBox v1.31.1 () multi-call binary.
Usage: gzip [-cfkdt] [FILE]...
```

- **Use gzip to Compress**

```
root@myir-yf13x:~# gzip test.txt
root@myir-yf13x:~# ls
OpenAMP_TTY_echo.elf  rs485_write  test.tar.gz  uart_test
rs485_read            test.c       test.txt.gz  wifi.conf
```

- **Use gzip to Decompress**

Unzip the file with “gunzip”, as shown below:

```
root@myir-yf13x:~# gunzip test.txt.gz
root@myir-yf13x:~# ls
OpenAMP_TTY_echo.elf  rs485_write  test.tar.gz  uart_test
rs485_read            test.c       test.txt    wifi.conf
```



8.2. File System Tools

The main test system file system tools, this section will introduce several common file system management tools. System comes with file system tools "mount" , "mkfs" , "fsck" , "dumpe2fs" .

1) mount tool

"mount" is a command under Linux that connects a partition to a folder on Linux, associating the partition with that directory, so simply accessing the folder is equivalent to accessing the partition, in the following syntax format:

```
root@myir-yf13x:~# mount -h
```

Usage:

```
mount [-lhV]
```

```
mount -a [options]
```

```
mount [options] [--source] <source> | [--target] <directory>
```

```
mount [options] <source> <directory>
```

```
mount <operation> <mountpoint> [<target>]
```

Mount a filesystem.

An example of mounting the first partition of an SD card is shown below:

```
root@myir-yf13x:~# mount /dev/mmcblk0p1 /mnt/
```

2) mkfs tool

After partitioning the hard disk, the next step is to set up the Linux file system. Similar to a formatted hard disk in Windows. Setting up a file system on a hard disk partition washes out the data on the partition and is not recoverable, so make sure that the data on the partition is no longer used before setting up the file system. The command to set up the file system is mkfs, in the following syntax format:

```
root@myir-yf13x:~# mkfs -h
```


**Usage:**

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

Make a Linux filesystem.

Options:

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

For more details see mkfs(8).

An example of formatting the seventh partition of an SD card is shown below:

```
root@myir-yf13x:~# mkfs -t ext3 -V -c /dev/mmcblk0p1
mkfs from util-linux 2.37.4
mkfs.ext3 -c /dev/mmcblk0p1
mke2fs 1.46.5 (30-Dec-2021)
/dev/mmcblk0p1 contains a vfat file system labelled 'Volumn'
Proceed anyway? (y,N) y
[ 108.937869] mmc_erase: erase error -110, status 0x0
Discarding device blocks: done
Creating filesystem with 20480 1k blocks and 5112 inodes
Filesystem UUID: fd6238ad-340e-447b-b40d-2115472289ee
Superblock backups stored on blocks:
    8193

Checking for bad blocks (read-only test): done
```



```
Allocating group tables: done
Writing inode tables: done
Creating journal (1024 blocks): done
Writing superblocks and filesystem accounting information: done
```

3) fsck tool

The “fsck” command is mainly used to check the correctness of the file system. When the file system fails, the fsck instruction can be used to try to fix it. And fix the Linux disk. For example:

```
root@myir-yf13x:~# fsck -a /dev/mmcbk0p1
fsck from util-linux 2.37.4
/dev/mmcbk0p1: clean, 11/5112 files, 2490/20480 blocks
```

4) dumpe2fs tool

Prints information about super blocks and blocks groups of existing file systems on a particular device. The development board enters the following command to see the application syntax:

```
root@myir-yf13x:~# dumpe2fs -h
dumpe2fs 1.46.5 (30-Dec-2021)
Usage: dumpe2fs [-bfghimxV] [-o superblock=<num>] [-o blocksize=<num>]
device
```

View detailed properties of the file system formatted, for example, by entering a command to view the details of a disk:

```
root@myir-yf13x:~# dumpe2fs /dev/mmcbk0p1
dumpe2fs 1.46.5 (30-Dec-2021)
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          fd6238ad-340e-447b-b40d-2115472289ee
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype sp
arse_super large_file
```



```
Filesystem flags:      unsigned_directory_hash
Default mount options: user_xattr acl
Filesystem state:     clean
Errors behavior:      Continue
Filesystem OS type:   Linux
Inode count:          5112
Block count:          20480
Reserved block count: 1024
Overhead clusters:    2471
Free blocks:          17990
Free inodes:          5101
First block:          1
```

.....

Group 0: (Blocks 1-8192)

Primary superblock at 1, Group descriptors at 2-2

Reserved GDT blocks at 3-81

Block bitmap at 82 (+81)

Inode bitmap at 83 (+82)

Inode table at 84-509 (+83)

6640 free blocks, 1693 free inodes, 2 directories

Free blocks: 1553-8192

Free inodes: 12-1704

Check the number of inodes on a disk. Inodes also consume disk space, so when the disk is formatted, the operating system automatically divides the disk into two areas. One is the data area, where file data is stored; The other is the inode section (inode table), which holds the information contained in the inode.

```
root@myir-yf13x:~# dumpe2fs /dev/mmcblk0p1 | grep -i "inode size"
dumpe2fs 1.46.5 (30-Dec-2021)
Inode size:          256
```

Look at the number of blocks on a disk. When the operating system reads the hard disk, it will not read sectors one by one, which is too inefficient. Instead, it



will read multiple sectors in a row at one time, that is, read a "block" at one time. This "block" of multiple sectors is the smallest unit of file access.

```
root@myir-yf13x:~# dumpe2fs /dev/mmcblk0p1 | grep -i "block size"
dumpe2fs 1.46.5 (30-Dec-2021)
Block size:                1024
```



8.3. Disk Management Utils

This section mainly test system disk management tools, including several common disk management tools. The system comes with disk management tools "fdisk" , "dd" , "du" , "df" , "cfdisk" and so on, These commands allow you to monitor your daily disk usage.

1) fdisk Disk Partitioning Tool

The "fdisk" disk partitioning tool has applications for DOS, Windows, and Linux. In Linux, "fdisk" is a menu-based command. To partition a hard disk with "fdisk" , you can add the hard disk to partition directly after the fdisk command as a parameter. The syntax is as follows:

```
root@myir-yf13x:~# fdisk -h
Usage:
fdisk [options] <disk>      change partition table
fdisk [options] -l [<disk>] list partition table(s)
```

Partition the eMMC as follows:

```
root@myir-yf13x:~# fdisk /dev/mmcb1k1p9
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 tool

The "dd" command is used to copy the specified input file to the specified output file. And the format can be converted during replication. The difference between the "dd" command and the "cp" command is that the "dd" command can be done on a floppy disk that has not been created, and the data



copied to the floppy disk is actually a mirror file. Similar to the “diskcopy” command in DOS. The format of the “dd” command is:

```
dd [<if=input file name/device name>] [<of=output file name/device name>]  
[bs=block size(byte)] [count=block count]
```

An example of creating a file of 2M size is shown below:

```
root@myir-yf13x:~# time dd if=/dev/zero of=temp bs=1M count=100 conv=  
fsync  
100+0 records in  
100+0 records out  
104857600 bytes (105 MB, 100 MiB) copied, 9.09095 s, 11.5 MB/s  
real    0m 9.09s  
user    0m 0.00s  
sys     0m 1.16s
```

3) du tool

The du command is used to display disk space usage. This command shows, level by level, how each level of subdirectory of the specified directory occupies a block of file system data. “du” is generally used in the following syntax:

```
root@myir-yf13x:~# du --help  
Usage: du [OPTION]... [FILE]...  
or: du [OPTION]... --files0-from=F  
Summarize disk usage of the set of FILEs, recursively for directories.
```

Partial parameter are as follows:

- -a: Displays the size of all directories or files
- -h: Take K,M and G Bytes as units to improve the readability of information
- -k: Output in Kilo Bytes
- -m: Output in Mega Bytes

Statistics the file size generated by the dd command:

```
root@myir-yf13x:~# du temp  
102400    temp
```



```
root@myir-yf13x:~# du -h temp
100M    temp
```

4) df tool

Used to display disk usage statistics for the current file system on a Linux system, commonly used as follows:

```
root@myir-yf13x:~# df --help
Usage: df [OPTION]... [FILE]...
Show information about the file system on which each FILE resides,
or all file systems by default.
```

Partial parameter are as follows:

- -h: The appropriate units can be displayed based on the size used
- -i: View the number of inodes and inode usage under the partition
- -T: Print out the file system type

To see the number of inodes and inode usage under the partition, use the following command:

```
root@myir-yf13x:~# df -i
Filesystem      Inodes IUsed   IFree IUse% Mounted on
devtmpfs        47667  486   47181    2% /dev
/dev/mmcblk1p8  723840 24114  699726    4% /
tmpfs            56483    3   56480    1% /dev/shm
tmpfs           819200   714  818486    1% /run
tmpfs           1024    13    1011    2% /sys/fs/cgroup
tmpfs          1048576   22 1048554    1% /tmp
/dev/mmcblk1p6   16384    22   16362    1% /boot
/dev/mmcblk1p7   4096    22    4074    1% /vendor
tmpfs           56483    27   56456    1% /var/volatile
tmpfs           11296    14   11282    1% /run/user/0
```

The inode is defined by the system during formatting, and the inode depends on the size of the disk partition. When we reach 100 percent inode usage, we cannot

write data to disk even though we still have disk space left. Please refer to Section 2.5 for other application examples.

8.4. Process Management Utils

Process is also an important concept in the operating system, it is a process of the execution of a program, the program is a static description of the process, the system run every program is in its process run. All processes in Linux are interconnected, and all processes have a parent except for initializing the process. Instead of being created, new processes are copied or copied from previous processes. All processes in Linux are derived from an "init" process with process number 1. Linux consists of three different types of processes, each with its own characteristics and attributes:

- The Interaction Process: A process started by a Shell can run both in the foreground and in the background.
- The Batch Process: This process has no connection to the terminal and is a sequence of processes. This process is committed to a process waiting for execution in queue order.
- Monitor Processes (daemons): Daemons are always active and usually run in the background, and daemons are usually started automatically by the system at the beginning through script activation or root.

For Linux system, process management is an important link, and process management is usually achieved through process management tools. There are several commonly used process management commands in Linux system: "ps" , "top" , "vmstat" , "kill" and so on.

1) ps tool

"ps" is a command to show the status of current processes. The general syntax is as follows:

```
root@myir-yf13x:~# ps --help
Usage:
ps [options]
```




Try 'ps --help <simple|list|output|threads|misc|all>'
or 'ps --help <s|l|o|t|m|a>'
for additional help text.
For more details see ps(1).

Partial parameter are as follows:

- -u: Organize a user-centric display of process status information.
- -a: A process that is not terminal dependent.
- -x: The process associated with the terminal.

Usually the above commands are combined with: aux

- -e: Show all processes; The equivalent of ax.
- -f: Displays full format program information.

Usually the above commands are combined with: ef

- -H: Shows the number of processes at the process level
- -F: Display more program information

Usually the above commands are combined with: eHF

An example of displaying information for all processes is shown below:

```
root@myir-yf13x:~# ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.2	1.3	8132	5876	?	Ss	21:39	0:04	/sbin/init
root	2	0.0	0.0	0	0	?	S	21:39	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	I<	21:39	0:00	[rcu_gp]
root	4	0.0	0.0	0	0	?	I<	21:39	0:00	[rcu_par_gp]
root	5	0.0	0.0	0	0	?	I<	21:39	0:00	[netns]
root	8	0.0	0.0	0	0	?	I	21:39	0:01	[kworker/u2:0- events_unbound]
root	9	0.0	0.0	0	0	?	I<	21:39	0:00	[mm_percpu_wq]
root	10	0.0	0.0	0	0	?	S	21:39	0:00	[rcu_tasks_kthre]



A brief explanation of some of the taskbars above:

- VSZ: Virtual memory size
- RSS: Resident size
- STAT: There are several process states
 - R: Running
 - S: Interruptable sleeping
 - D: Uninterruptable sleeping
 - T: Stopped
 - Z: Zombie process
 - +: Foreground process
 - N: Low priority process
 - I: Multithreaded process

2) top tool

The “top” command puts quite a bit of overall system performance information on one screen. The display can also be changed in an interactive manner. Dynamic continuous monitoring of the running state of the process, the “top” syntax is generally as follows:

```
root@myir-yf13x:~# top -help
procps-ng 3.3.17-dirty
Usage:
top -hv | -bcEeHiOSs1 -d secs -n max -u|U user -p pid(s) -o field -w [cols]
```

See section 2.1 for an example of dynamically viewing system processes.

3) vmstat: Virtual memory statistics tool

This command looks at the usage status of the memory space and gives you a snapshot of the performance of the entire system. “vmstat” runs in two modes: sample mode and average mode. If no parameters are specified, “vmstat” statistics run in average mode and “vmstat” displays the average of all statistics since system startup. The common syntax and parameters are as follows:

```
root@myir-yf13x:~# vmstat -h
Usage:
```



```
vmstat [options] [delay [count]]
```

Options:

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

-h, --help           display this help and exit
-V, --version        output version information and exit
```

“vmstat” runs in average mode, showing the average of all statistics since system startup:

```
root@myir-yf13x:~# vmstat
procs -----memory----- ---swap-- -----io---- -system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi   bo    in   cs us sy id wa
st
 0  0       0 159352 12520 214560    0    0   80   76  256  408  3  4
92  1  0
```

A brief explanation of some of the taskbars above:

- r: Number of processes currently running. Instead of waiting for I/O, these processes have ready numbers to run. Ideally, the number of processes that can run is equal to the number of CPU available
- b: The number of blocked processes waiting for I/O to complete
- forks: The number of times a new process was created
- in: The number of system interrupts



- cs: The number of times a context switch occurred in the system
- us: Percentage of total CPU time consumed by user processes
- sy: The percentage of total CPU time consumed by the system code, which includes time consumed in the System, IRQ, and Softirq states
- wa: The percentage of total CPU consumed waiting for I/O
- id: The percentage of total CPU time consumed by the system idle

Statistical system various data details are as follows:

```
root@myir-yf13x:~# vmstat -s
451864 K total memory
65432 K used memory
30732 K active memory
219400 K inactive memory
159352 K free memory
12520 K buffer memory
214560 K swap cache
0 K total swap
0 K used swap
0 K free swap
1864 non-nice user cpu ticks
2825 nice user cpu ticks
7588 system cpu ticks
163145 idle cpu ticks
1317 IO-wait cpu ticks
0 IRQ cpu ticks
122 softirq cpu ticks
0 stolen cpu ticks
139309 pages paged in
132883 pages paged out
0 pages swapped in
0 pages swapped out
450684 interrupts
```



718821 CPU context switches
1651181992 boot time
1599 forks

A brief explanation of some of the information above:

- total memory: Total system memory
- used memory: Used memory
- CPU ticks: This shows the CPU time since the system started, where "tick" is a unit of time
- forks: Roughly speaking, this represents the number of new processes that have been created since system startup

"vmstat" provides a great deal of information about the performance of Linux systems. It is one of the core tools for investigating system problems.

4) kill tool

Sends the specified signal to the appropriate process. Not specifying the model sends SIGTERM (15) to terminate the specified process. If the program cannot be terminated with the available "-KILL" parameter, the signal it sends is SIGKILL(9), which forces the process to be terminated. The process number can be viewed using the "ps" command or the "jobs" command. A root user will affect the user's process, and a non-root user can only affect his or her own process. The general syntax of the kill command is as follows:

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

Partial parameter are as follows:

- -s: Specifies the signal to send
- -p: simulate transmitting signal
- -l: Specifies the name list of signals
- pid: The ID number of the process to abort
- Signal: According to the signal

First use "ps -ef" and the pipe command to determine the PID to kill the process.



```
root@myir-yf13x:~# ps -ef | grep mxapp2
root      1601   1050   0 22:10 ttySTM0  00:00:00 grep mxapp2
```

Then type the following command to terminate the process:

```
root@myir-yf13x:~# kill 1050
```

The “killall” command terminates all processes within the same process group, allowing you to specify the name of the process to be terminated instead of the PID process number.

```
root@myir-yf13x:~# killall mxapp2
root@myir-yf13x:~#
```



9. Application Development

This chapter mainly introduces some basic information for secondary development of the current SDK. The current SDK provides two configuration reference images, one is "myir-image-core", which is mainly for non-GUI applications; the other is "myir-image-full", which adds some GUI applications on the basis of "myir-image-core", such as Qt runtime libraries, graphics processing related libraries and demo applications. For information about these two images, please refer to the "MYD-YF13X SDK2.0.0 Release notes".

9.1. Development Language

1) SHELL

Shell is a program written in C language, which is a bridge for users to use Linux. Shell is both a command language and a programming language. There are many types of common Linux shells:

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

Bash, which is also supported by the current SDK as default, is highlighted here. The following example is the script "start.sh" in myir-image-full image when MEasy HMI 2.0 boots up. The contents are as follows:

```
#!/bin/sh -e

echo "start myir HMI 2.0..."

export QT_QPA_EGLFS_ALWAYS_SET_MODE="1"
export QT_QPA_EGLFS_KMS_ATOMIC='1'
```



```
export QT_QPA_EGLFS_KMS_CONFIG='/usr/share/qt5/cursor.json'
#export QT_QPA_PLATFORM='eglfs'
psplash-drm-quit

strA=$(cat /sys/firmware/devicetree/base/compatible)
strB="ya157c"

result=$(echo $strA | grep "${strB}")
if [[ "$result" != "" ]]
then
export QT_QPA_PLATFORM='eglfs'
/home/mxapp2 -platform eglfs &
else
/home/mxapp2 -platform linuxfb &
fi

exit 0
```

This script first sets the environment variable MEasy HMI V2.0, and then calls the /home/mxapp2 executable to start MEasy HMI V2.0 according to different kernel device tree information with different parameters. The "&" at the end of the parameter means the program will enter the background to run.

2) C/C++

C/C++ is the most commonly used programming language for low-level application development under the Linux platform, and is also the most efficient language after assembly. Development using C/C++ usually adopts the mode of cross-development, that is, development is carried out on the development host side, binary execution files are compiled and generated running on the target machine, and then deployed to run on the target machine. In this way, the SDK built based on Yocto needs to be installed first. Please refer to "MYD-YF13X Yocto Software Development Guide" for installation steps. After installation, the SDK environment needs to be configured as follows:



```
PC $:source /opt/st/myir-yf13x/4.0.4-snapshot/environment-setup-cortexa7t2hf-
neon-vfpv4-ostl-linux-gnueabi
PC $:arm-ostl-linux-gnueabi-gcc -march=armv7ve -mthumb -mfpu=neon-vfpv4
-mfloat-abi=hard -mcpu=cortex-a7 --sysroot=/opt/st/myir-yf13x/4.0.4-snapsho
t/sysroots/cortexa7t2hf-neon-vfpv4-ostl-linux-gnueabi
```

This section demonstrates application development by writing a simple Hello World instance, and the following demo program “hello.c” is written on the development host side:

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

Next, the application is compiled with “\$CC” , because the corresponding header file and link are needed when compiling. “\$CC” contains the corresponding system library and configuration information. If you directly compile with “arm-openstlinux_eglfs-linux-gnueabi-gcc” , you will not find the header files.

```
$CC -v hello.c -o hello
Using built-in specs.
COLLECT_GCC=arm-ostl-linux-gnueabi-gcc
COLLECT_LTO_WRAPPER=/opt/st/myir-yf13x/4.0.4-snapshot/sysroots/x86_64-ostl
_sdk-linux/usr/libexec/arm-ostl-linux-gnueabi/gcc/arm-ostl-linux-gnueabi/11.3.0/
lto-wrapper
Target: arm-ostl-linux-gnueabi
.....
```

The generated execution file is then copied to the target machine through the “scp” command and executed as follows:

```
root@myir-yf13x:~# ./hello
hello world!
```



For more complex examples and development methods, refer to the “MYD-YF13X Yocto Software Development Guide” for instructions on application migration.

3) Python

Python is a high-level programming language with interpreted, object-oriented, dynamic data types. Python was invented by Guido van Rossum in late 1989, and the first public release was released in 1991. Like the Perl language, the Python source code is also licensed under the GPL(GNU General Public License). This section focuses on testing the use of Python, both from the Python command line and from the script.

- **Check the Supported Version of Python**

```
root@myir-yf13x:~# python3.10 -V
Python 3.10.4
```

- **Python Interactive Mode**

Start Python and enter the following text at the Python prompt, then press Enter to see how it works:

```
root@myir-yf13x:~# python3
Python 3.10.4 (main, Mar 23 2022, 20:25:24) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print ("Hello, Python!")
```

In Python version 3.8.2, the output from the above instance is as follows:

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

Run “exit()” function or press “ctrl+D” to exit Python:

```
>>> exit()
root@myir-yf13x:~#
```

- **Write a Python Script for Testing**

Write a simple Python script, and all Python files will have the.py extension.



```
root@myir-yf13x:~# vi test.py
root@myir-yf13x:~# cat test.py
#!/usr/bin/python3
print ("Hello, Python!")
```

The Python3 interpreter executes the script in the */usr/bin* directory using the following command.

```
root@myir-yf13x:~# chmod +x test.py
root@myir-yf13x:~# ./test.py
Hello, Python!
```

The Python3 interpreter is called with script parameters to start executing the script until it is finished. When the script has finished executing, the interpreter is no longer valid. The current system does not support “pip” commands , and clients need to port the “pip” tool if they need it.

9.2. Database

A Database is a warehouse that organizes, stores, and manages data in terms of data structures. There are many types of databases, commonly used databases are Access, Oracle, Mysql, SQL Server, SQLite and so on.

- **System SQLite**

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to a normal disk file. A complete SQL database containing multiple tables, indexes, triggers, and views is contained in a single disk file. This is a lightweight database, an ACID-compliant relational database management system, designed for embedded use and currently used in many embedded products. Its footprint is very low, and in embedded devices, it may only need a few hundred K of memory. This database runs faster than MySQL or PostgreSQL, so here's a quick test.

Start sqlite3 and create a new database <testdb.db>. Enter the following command into the terminal interface.

```
root@myir-yf13x:~# sqlite3 testDB.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite>
```

The above command creates a file "*testDB.db*", in the current directory. This file will be used as a database by the SQLite engine. Notice that the sqlite3 command provides a "sqlite>" prompt after successfully creating the database file.

With the database created, you can use the ".databases" command of SQLite to check if it is in the database list, as shown below:

```
sqlite> .databases
main: /home/root/testDB.db
sqlite>
```

Exit the SQLite prompt with the ".quit" command, as shown below:

```
sqlite> .quit  
root@myir-yf13x:~#
```

If you want to learn more about SQLite related information, please refer to the website <https://www.sqlite.org/docs.html>.



9.3. Application Localization for Qt

This section discusses localization-related issue and localization test with Qt applications. Localization refers to that a program or software, on the basis of supporting internationalization, gives the language information of a specific area of a program so that it can adapt to the use of the population of a specific area in the processing of information input and output. Some locale variables used by the program are allowed to be configured dynamically at program execution time. This chapter is mainly for the localization of QT applications, taking MEasy HMI 2.0 as an example for illustration.

1) Multiple Language

This section mainly uses MEasy HMI 2.0 as an example to illustrate the practical application of multiple languages in QT projects.

- **Open mxapp2 Demo Project**

MEasy HMI 2.0 corresponding project source code is "*Mxapp2.tar.gz*", copy to the environment built by the above document, and use QT Creator to open the project.

- **Generate *.ts File for Qt Project**

Enter the source directory of the MXAPP2 project through the terminal, and execute the following command to generate the translation file.

```
PC$ lupdate mxapp2.pro
Info: creating stash file /home/qinlh/download/MXAPP/.qmake.stash
Updating 'languages/language_zh.ts'...
    Found 202 source text(s) (0 new and 202 already existing)
Updating 'languages/language_en.ts'...
    Found 202 source text(s) (0 new and 202 already existing)
```

The project has completed the translation work before the release, here will not regenerate the "*language_zh.ts*" and "*language_en.ts*" files. The "*MXAPP2*" application display Chinese by loading the *.qm files generated from



language_zh.ts, and display English by loading the *.qm files generated from language_en.ts.

● Translation

Open the language_en.ts file, for the translated source string is <source> node inside the content, the translated target string is <translation> inside the content, the user can modify according to the needs.

```
<message>
  <location filename="../Album.qml" line="50"/>
  <source>返回</source>
  <translation type="unfinished">Return</translation>
</message>
```

● Generate *.qm Files for Qt

*.ts file needs to be manually generated for translation template file after modification. The following command can be used to generate translation template file.

```
PC$ lrelease language_en.ts -qm language_en.qm
Updating 'language_en.qm'...
  Generated 183 translation(s) (2 finished and 181 unfinished)
Ignored 21 untranslated source text(s)
PC$ lrelease language_zh.ts -qm language_zh.qm
Updating 'language_en.qm'...
  Generated 183 translation(s) (2 finished and 181 unfinished)
Ignored 21 untranslated source text(s)
```

● Apply Translation Files

The use of the translation file needs to be called through the application code, the specific call method refers to the load Language member function in the source code "*translator.cpp*".



2) Fonts

● Install Fonts for Qt Applications

Font files can be placed directly into the development board file system /usr/lib/ directory.

```
# ls /usr/lib/fonts/msyh.ttc
/usr/lib/fonts/msyh.ttc
```

Or add directly to the QT project inside.

```
PC $ tree
└── fontawesome-webfont.ttf
```

● Use Fonts in Qt Applications

The font file needs to be called through the application code, In addition to using "*msyh.ttc*" to display text, MXAPP2 uses a special font called "*fontawesome-webfont.ttf*" to display icons. Regarding the specific method for loading fonts , please refer to IconFontInit () function in "main.cpp" .

```
void iconFontInit()
{
    int fontId_digital = QFontDatabase::addApplicationFont(":/fonts/DIGITAL/DS-DIGIB.TTF");
    int fontId_fws = QFontDatabase::addApplicationFont(":/fonts/fontawesome-webfont.ttf");
    QString fontName_fws = QFontDatabase::applicationFontFamilies(fontId_fws).at(0);
    QFont iconFont_fws;
    iconFont_fws.setFamily(fontName_fws);
    QApplication::setFont(iconFont_fws);
    iconFont_fws.setPixelSize(20);
}
```




3) Virtual Keyboard from Qt

This chapter mainly takes MEasy HMI 2.0 as an example to illustrate the practical application of soft keyboard in QT project. Before reading the following contents, please refer to Chapter 3.1 Environment Construction of "MYD-YF13X_QT and MEasy HMI2.0 software development guide", and setup the compilation environment for MEasy HMI.

Since version 5.7, QT Virtual Keyboard component has been added to the official source code. QT 5.15 version has been used by MYIR in the published MEasy HMI 2.0 application, and the Virtual Keyboard component has been configured.

- **Embedded Virtual Keyboard into QML**

The soft keyboard provided by Qt can only be used in QML code, and the location of the soft keyboard pop-up and the size of the soft keyboard need to be defined before the call, as shown in the following code.

```
InputPanel {
id: inputPanel
x: adaptive_width/8
y: adaptive_height/1.06
z:99
anchors.left: parent.left
anchors.right: parent.right

states: State {
name: "visible"
when: inputPanel.active
PropertyChanges {
target: inputPanel
y: adaptive_height/1.06 - inputPanel.height
}
}
}
```

- **Trigger Virtual Keyboard**



The soft keyboard is triggered by QML TextField, TextEdit or other editable components. The user only needs to add this component to the UI component to call up the soft keyboard on the UI and use it. If it is a QT system, you can use the Qt VirtualKeyboard input that comes with Qt.

```
TextField{
id: netmask_input
InputMethodHints: Qt.ImhFormattedNumbersOnly
onAccepted: digitsField.focus = true
font.family: "Microsoft YaHei"
color: "white"
}
```

- **Use Virtual Keyboard**

For the use of soft keyboard, please refer to Chapter 2.6, "MYD-YF13X_QT and MEasy HMI2.0 software development guide", for the UI interface of system setting. The user clicks on the editable interface component and the soft keyboard pops up.



10. Reference

- **STM32MPU Development Zone**

https://wiki.st.com/stm32mpu/wiki/Development_zone

- **Yocto Project BSP Development Guide**

<https://docs.yoctoproject.org/4.0.9/bsp-guide/index.html>

- **Yocto Project Linux Kernel Development Guide**

<https://docs.yoctoproject.org/4.0.9/kernel-dev/index.html>

- **Linux Kernel Watchdog Information**

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

- **Qt for Embedded Linux**

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

- **Wayland Architecture**

<https://wayland.freedesktop.org/architecture.html>

- **Systemd Network Configuration**

<https://www.freedesktop.org/software/systemd/man/systemd.network.html>



Appendix A

Warranty & Technical Support Services

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

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

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

Service Guarantee

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

Price



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

Delivery Time

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

Technical Support

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

After-sale Service

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

Technical support service

MYIR offers technical support for the hardware and software materials which have provided to customers;

- To help customers compile and run the source code we offer;
- To help customers solve problems occurred during operations if users follow the user manual documents;
- To judge whether the failure exists;



- To provide free software upgrading service.

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

- Hardware or software problems occurred during customers' own development;
- Problems occurred when customers compile or run the OS which is tailored by themselves;
- Problems occurred during customers' own applications development;
- Problems occurred during the modification of MYiR's software source code.

After-sales maintenance service

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

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

Warm tips



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

Maintenance period and charges

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

Shipping cost



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

Products Life Cycle

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

Value-added Services

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

MYIR Electronics Limited

Room 04, 6th Floor, Building No.2, Fada Road,
Yunli Inteiligent Park, Bantian, Longgang District.

Support Email: support@myirtech.com

Sales Email: sales@myirtech.com

Phone: +86-755-22984836

Fax: +86-755-25532724

Website: www.myirtech.com