MYS-ZU5EV Linux Evaluation Guide

File Status : [ ] Draft [√] Release	**FILE ID :**	MYIR-MYS-ZU5EV-SW-EG-EN-L5.4.0
	VERSION :	V1.10
	AUTHOR :	Fengyong
	CREATED :	2021-05-25
	UPDATED :	2021-05-25

Revision History

VERSION	AUTHOR	PARTICIPANT	DATE	DESCRIPTION
V1.10	Fengyong		20210525	Initial version for MYS-ZU5EV

CONTENT

1. Overview

Linux Software evaluation Guide is used to describe the test steps and evaluation methods for running core and peripheral resources under an open source Linux system on MYIR's board. This article can be used as a pre-assessment guide or as a test guide for common system development.

1.1. Hardware resources

MYS-ZU5EV Board is launched a Xilinx zynqMP ZU5EV as the main processor embedded development board of MYIR. MYS-ZU5EV Board uses Xilinx's zynqMP SoC platform to tightly integrate arm processors with FPGA architectures. With the high performance of 4 ARM Cortex A53 cpu cores and 2 ARM Cortex R5 cores, the product is designed to better meet a wide range of industrial needs. In addition, MYIR provides a wealth of software resources and documentation. For detailed configuration parameters for the hardware section, please refer to the "MYS-ZU5EV product Manual". At the same time, the user will use some accessories during the evaluation test, see the list below.

Table 1-1. Optional module

Accessories	Interface	Description and netlink
Camera	USB	MY-CAM002U http://www.myir-tech.com/product/my_cam002u.htm

1.2. Software resources

MYS-ZU5EV development board provides the development method: based on Petalinux development; Bootloader, Kernel, and various parts of the file system are all open as source code, please see “MYS-ZU5EV_SDK Release Notes”.

The development board has been burned at the factory according to the core board model mys-zu5ev-full image, you only need to power up to use. Explain: (1) For testing of vcu, please refer to the method of "VCU test" in the "8.4.8 VCU application example" section of “MYS-ZU5EV_Linux Development Guide”; (2) For testing of the mipi camera, please refer to the method in the "8.4.7 MIPI Camera application example" section of “MYS-ZU5EV_Linux Software Development Guide”.

1.3. Document resources

Depending on the stages of the user's use of the board, the SDK contains documentation for each stage, release instructions, evaluation guides, development guides, and other different categories of documentation and manuals. For a specific list of documents, see the instructions in Table 2-4 of the “MYS-ZU5EV_SDK Release Notes”.

1.4. Environmental preparation

Before you begin evaluating the board software, you need to do some necessary preparation and configuration of the board for some basic environment, including the correct hardware wire, configuration debug serial port, set up start-up and other steps. Detailed steps can be found in the “MYS-ZU5EV-QSG”.

The next section focuses on how to evaluate and test the hardware resources and interfaces of the system, as well as software features. Test primarily with some of the tools and commands commonly used under Linux, as well as applications you develop yourself. The software evaluation guide is divided into several parts to describe, including: core resources, peripheral resources, network applications, multimedia applications, development support applications, system tools and other categories. Later chapters provide a comprehensive explanation of each section and describe in detail the specific evaluation methods and steps for each part of the resource.

2. Core resources

In Linux system, a proc virtual file system is provided to query the parameters of each core resource and some common tools to evaluate the performance of the resource. The parameters of core resources such as CPU, memory, eMMC, RTC, etc. will be read and tested in detail below.

2.1. CPU

ZynqMP-zu5ev is the integration of 4 ARM Cortex A53 cpu cores and 2 ARM Cortex R5 cores into a programmable FPGA chip into an on-chip system.

ARM processors and various storage peripheral resources are often referred to as processing systems (Processor System, PS) and FPGA sections are referred to as programmable logic (Programmable Logic, PL). Arm Cortex-A53 is an application-grade processor that supports operations similar to linux operating systems. The Xilinx7 architecture used by FPGAs enables an industry-standard AXI interface that is efficiently coupled between ARM and FPGA, reducing the additional power consumption generated by discrete chips, enabling high-bandwidth, low-latency connectivity while reducing physical size and production costs.

The PL part of ZynqMP is ideal for high-speed logical operation and parallel data flow processing, and the PS part supports software control or operating system. This means that based on this platform, most systems can be designed to be divided into software and hardware based on functionality, allowing PS and PL to take advantage of each other to maximize performance across the system:

1) View the CPU information

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

```
root@mys_zu5ev # cat /proc/cpuinfo
processor       : 0
BogoMIPS        : 199.99
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
```

```
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 1
BogoMIPS        : 199.99
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 2
BogoMIPS        : 199.99
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 3
BogoMIPS        : 199.99
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4
```

- processor : The number of logical processing cores in the system can be physical or virtual using hyperthreaded technology for multicore processors

- BogoMIPS : A roughly measured CPU runs millions of instructions per second when the system kernel starts (Million Instructions Per Second)

2) View CPU usage

```
root@mys_zu5ev # top
Mem: 65424K used, 3966032K free, 56K shrd, 492K buff, 4528K cached
CPU:   0% usr   0% sys   0% nic  99% idle   0% io   0% irq   0% sirq
Load average: 0.00 0.08 0.05 1/74 184
  PID  PPID USER     STAT   VSZ %VSZ %CPU COMMAND
  177     1 root     S     3516   0%   0% -sh
  184   177 root     R     2560   0%   0% top
    1     0 root     S     2388   0%   0% init
  148     1 root     S     2388   0%   0% /sbin/syslogd -n
  151     1 root     S     2388   0%   0% /sbin/klogd -n
  127     2 root     SW       0   0%   0% [irq/38-fd4a0000]
   85     2 root     IW       0   0%   0% [kworker/2:1-eve]
  129     2 root     IW       0   0%   0% [kworker/0:3-eve]
  107     2 root     IW       0   0%   0% [kworker/2:2-rcu]
  124     2 root     IW       0   0%   0% [kworker/0:2-eve]
   31     2 root     IW       0   0%   0% [kworker/1:1-eve]
  117     2 root     SW       0   0%   0% [kpktgend_0]
  118     2 root     SW       0   0%   0% [kpktgend_1]
  119     2 root     SW       0   0%   0% [kpktgend_2]
  120     2 root     SW       0   0%   0% [kpktgend_3]
    2     0 root     SW       0   0%   0% [kthreadd]
    3     2 root     IW<      0   0%   0% [rcu_gp]
    4     2 root     IW<      0   0%   0% [rcu_par_gp]
    6     2 root     IW<      0   0%   0% [kworker/0:0H]
    7     2 root     IW       0   0%   0% [kworker/u8:0-ev]
```

- %usr : Represents cpu usage of the user space program (not dispatched via nice)
- %sys : The cpu usage that represents the system space is mainly kernel program

- %nic：Cpu usage for programs that represent user space and have been dispatched through nice
- %idle：Idle cpu
- %irq：The number of hard interrupts processed by CPU
- %sirq：The number of soft interrupts processed by CPU

3) CPU pressure test

There are many ways to test the pressure of CPU, and the circumference method can be calculated by bc command to test the stability of the CPU in the course of operation.

```
root@mys_zu5ev # echo "scale=5000; 4*a(1)" | bc -l -q &
[1] 716
```

The above command will calculate the PI in the background and be accurate to 5000 decimal places. The calculation process takes some time. At this point, we can check for changes in CPU utilization with the top command, as follows:

```
root@mys_zu5ev # top
top - 08:50:51 up 8 min,  1 user,  load average: 0.65, 0.27, 0.14
Tasks:  94 total,   2 running,  92 sleeping,   0 stopped,   0 zombie
%Cpu(s): 25.1 us,  0.0 sy,  0.0 ni, 74.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3937.1 total,   3859.4 free,     45.0 used,     32.7 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.   3801.9 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  716 root      20   0    2736   1604   1352 R  99.7   0.0   0:57.47 bc

  109 root      20   0       0      0      0 S   0.3   0.0   0:00.03 kpktgend_0

  719 root      20   0    3288   2044   1628 R   0.3   0.1   0:00.05 top

    1 root      20   0    1916    616    556 S   0.0   0.0   0:01.42 init
```

```
    2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
    3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
    7 root      20   0       0      0      0 I   0.0   0.0   0:00.00 kworker/u8:0
-events_unbound
    8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_
wq
    9 root      20   0       0      0      0 S   0.0   0.0   0:00.00 ksoftirqd/0
   10 root      20   0       0      0      0 I   0.0   0.0   0:00.01 rcu_sched
   11 root      rt   0       0      0      0 S   0.0   0.0   0:00.00 migration/0
   12 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0


. . .
```

After about 3 minutes, the PI results are calculated. During this period CPU usage rate reached 100%, no abnormality occurred, indicating that the CPU stress test passed. You can also continue to increase the exact value to further increase the test pressure.

```
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058
.....
74351362222477158915049530984448933309634087807693259939780541934144737744184263129860809988868741326047 20
```

4) CPU work frequency

This CPU supports fixed frequencies, and the four A53 cores are synchronous.

View CPU working frequency

```
root@mys_zu5ev # cat /sys/bus/cpu/devices/cpu0/cpufreq/cpuinfo_cur_freq
1199999
```

View CPU max work frequency

```
root@mys_zu5ev # cat /sys/bus/cpu/devices/cpu0/cpufreq/cpuinfo_max_freq
1199999
```

View all operating modes of the CPU

```
root@mys_zu5ev # cat /sys/bus/cpu/devices/cpu0/cpufreq/scaling_available_governors
userspace
```

userspace：user mode, in any case, controls the CPU running within the configured frequency range, and the user in the configuration adds the power-saving settings himself.

View the current working mode of the CPU

```
root@mys_zu5ev # cat /sys/bus/cpu/devices/cpu0/cpufreq/scaling_governor
userspace
```

2.2. Memory

MYS-ZU5EV memory defaults to 4GB. If you need to use 8G or other capacity of memory, you need to configure the parameters of 8G memory in vivado, the other configurations in vivado remain the same, generate new xsa files, and then compile the image file according to the "5.5.Petalinux Project building FPGA New Project" section of “MYS-ZU5EV_Linux Development Guide”, which is a petalin software that adapts 8G memory.

1) View memory information

Read the parameter information of the memory in the system and it can be obtained through the /proc/meminfo file.

```
root@mys_zu5ev # cat /proc/meminfo
MemTotal:        4031456 kB
MemFree:         3965528 kB
MemAvailable:    3897128 kB
Buffers:             496 kB
Cached:             4588 kB
SwapCached:            0 kB
Active:             5508 kB
Inactive:            620 kB
Active(anon):       1084 kB
Inactive(anon):       12 kB
Active(file):       4424 kB
Inactive(file):      608 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:                 0 kB
Writeback:             0 kB
AnonPages:          1040 kB
Mapped:             3256 kB
Shmem:                56 kB
```

```
KReclaimable:          9060  kB
Slab:                 28460  kB
SReclaimable:          9060  kB
SUnreclaim:           19400  kB
KernelStack:           1216  kB
PageTables:              76  kB
NFS_Unstable:             0  kB
Bounce:                   0  kB
WritebackTmp:             0  kB
CommitLimit:        3668624  kB
Committed_AS:          7784  kB
VmallocTotal:     262930368  kB
VmallocUsed:           1832  kB
VmallocChunk:             0  kB
Percpu:                 832  kB
AnonHugePages:            0  kB
ShmemHugePages:           0  kB
ShmemPmdMapped:           0  kB
FileHugePages:            0  kB
FilePmdMapped:            0  kB
CmaTotal:            262144  kB
CmaFree:             261888  kB
HugePages_Total:          0
HugePages_Free:           0
HugePages_Rsvd:           0
HugePages_Surp:           0
Hugepagesize:          2048  kB
Hugetlb:                  0  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.
- Active : Memory that is frequently (recently) used
- Inactive : Memory that has not been used very often recently

2) Get memory usage

You can use free command to read memory usage, and the -m parameter represents MByte.

```
root@mys_zu5ev # free -m
total        used        free      shared     buffers      cached
Mem:          3936          64        3872           0           0           4
-/+ buffers/cache:          59        3877
Swap:            0           0           0
```

- total : The total amount of memory
- used : The amount of memory used
- free : The amount of memory that can be used

3) Memory pressure test

Given the size and number of test memory, you can stress test the existing memory of the system. You can use the system tool memtester for testing, such as specifying a memory size of 300MB, 10 tests, and the test command "memtester 300M 10".
Take a single test using 300MB of memory space as an example:

```
root@mys_zu5ev # memtester 300M 1
memtester version 4.3.0 (32-bit)
Copyright (C) 2001-2012 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        : ok
Done.
```

2.3. eMMC

This section focuses on eMMC testing and is suitable for development boards with eMMC memory.

eMMC is a data storage device that includes a MultiMediaCard (MMC) interface.Its cost, small size, Flash technical independence, and high data throughput make it ideal for embedded products. The MYS-ZU5EV core board section is configured with 32GB eMMC, please refer to the MYS-ZU5EV Product Manual for detailed core board configuration parameters. This section explains the steps and methods for viewing and operating eMMC under Linux systems.

1) View EMMC capacity

The "fdisk –l" command allows you to query eMMC partition information and capacity.

```
root@mys_zu5ev # fdisk -l
Disk /dev/mmcblk0: 29.1 GiB, 31272730624 bytes, 61079552 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000000

Device          Boot Start       End   Sectors  Size Id Type
/dev/mmcblk0p1       63  61079129  61079067 29.1G 83 Linux
```

- /dev/mmcblk0p1 : Used to store the root file system

2) View eMMC partition information

The df command allows you to query eMMC partition information, usage, mount directories, and more.

```
root@mys_zu5ev # df -h
Filesystem                Size      Used Available Use% Mounted on
/dev/root                28.5G    163.2M     26.9G   1% /
devtmpfs                  1.8G         0      1.8G   0% /dev
tmpfs                     1.9G         0      1.9G   0% /dev/shm
```

```
tmpfs                  1.9G     52.0K      1.9G   0% /tmp
tmpfs                  1.9G     28.0K      1.9G   0% /run
/dev/mmcblk1p1      1017.7M    397.4M    620.3M  39% /mnt/mmcblk1p1
/dev/mmcblk1p2        11.8G    454.7M     10.7G   4% /mnt/mmcblk1p2
/dev/mmcblk0p1        28.5G    163.2M     26.9G   1% /mnt/mmcblk0p1
```

- /dev/root : Root file system, mounted to the root directory.
- tmpfs : Memory virtual file system, mounted to different directories.
- devtmpfs : Used for system creation dev.
- /dev/mmcblk0p1 : Root file system, mounted on /mnt/mmcblk0p1.

3) eMMC performance testing

Performance testing mainly tests eMMC reading and writing speed of files under linux system, generally combined with time and dd dual commands to test.

● Write test

```
root@mys_zu5ev #time dd if=/dev/zero of=tempfile bs=1M count=100 conv=fdat
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 5.11405 s, 20.5 MB/s

real    0m5.117s
user    0m0.000s
sys     0m0.382s
```

When writing a file using the dd command, you need to add the conv-fdatasync parameter, which means that when dd write N times, the flush cache is synchronized to disk. Because writing to a disk is usually written first to the cache and returned before it is written to the disk. The write disk speed is tested here at 20.5MB/s.

● Read test

In embedded systems, it is often necessary to test the reading and writing performance of system files and ignore the influence of cache when reading files. At this point, you can specify the parameter “iflag=direct, nonblack”.

```
root@mys_zu5ev # time dd if=tempfile of=/dev/null bs=1M count=100 iflag=dire
ct,nonblock
100+0 records in
```

```
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 4.33127 s, 24.2 MB/s

real    0m4.334s
user    0m0.000s
sys     0m0.019s
```

It is known that the read speed is 24.2MB/s directly from disk.

2.4. QSPI

MYS-ZU5EV has a 32M qspi that can be used to store data, so notice one of its features. The area written by memory must be the area after erasing; Erasure is the smallest unit by block (0x10000).

1) Read and write under the file system

The following is a demonstration of the test steps under the file system, and here are some qspi read and write commands for reference

- hexdump : View partition content
- mtd_debug read : Used to read qspi data to a file
- mtd_debug wirte : Used to write file data to qspi
- mtd_debug erase : Erase qspi data

● View partition content

Use hexdump to view what QSPI stores:

```
root@mys_zu5ev # hexdump -C /dev/mtd0
00000000  00 00 00 14 00 00 00 14  00 00 00 14 00 00 00 14  |................|
```

● Write data

Write data to QSPI with the flashcp command and view the writes:

```
root@mys_zu5ev # flashcp -v qspi_test.txt /dev/mtd0
Erasing blocks: 1/1 (100%)
Writing data: 0k/0k (100%)
Verifying data: 0k/0k (100%)
[root@myir ~]# hexdump -C /dev/mtd0
00000000  6d 79 69 72 20 71 73 70  69 20 74 65 73 74 0a ff  |myir qspi test..|
```

● Read data

Read QSPI data through dd command:

```
root@mys_zu5ev # dd if=/dev/mtd0 of=read.txt bs=14 count=1
1+0 records in
1+0 records out
root@myir:~# cat read.txt
myir qspi test
```

2) Read and write qspi under uboot

The following shows the read qspi step in the uboot phase, where the memory operation instructions are typically mw and md.

- **Memory read and write command under uboot**

  - mw command

Entering instruction mw under uboot prompts the usage of mw and writes the command in memory:

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

Where b: 8 bits, w: 16 bits, l: 32 bits (default value), representing the number of bits written each time; address is the address to write to memory, value is the value to write, and count is the number to write from address. These are all hexadecimal numbers.

  - md command

Entering instruction md under uboot prompts the use of md, memory read command:

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

where b/w/l means the same, the default is to output in 32 bits.

- **Init QSPI**

Here you need to use the sf command of the SPI flash subsystem, enter sf under uboot, which prompts the use of the sf command. Power up press any key to enter the uboot shell interface and initialize qspi with the following command:

```
ZynqMP> sf probe
SF: Detected s25fl256s1 with page size 256 Bytes, erase size 64 KiB, total 32 MiB
```

- **Read the data after empty memory**

Read qspi content after emptying a piece of memory:

```
ZynqMP> mw.b 0x48000000 0xff 0x10
ZynqMP> md 0x48000000 0x10
48000000:   ffffffff   ffffffffffffffffffffffff      ................
48000010: 00000000 00000000 00002014 00000000      ........ ......
48000020: 00000000 00000000 00000000 00000000      ................
48000030: 00002011 00000000 00000001 00000000      . ............
ZynqMP> sf read 0x48000000 0x10 0x10
device 0 offset 0x10, size 0x10
SF: 16 bytes @ 0x10 Read: OK u-
ZynqMP>  md  0x48000000  0x10
u-boot=> md 0x48000000 0x10

48000000: 7269796d 70737120 65742069 ffff7473      myir qspi test..
48000010: 00000000 00000000 00002014 00000000      ........ ......
48000020: 00000000 00000000 00000000 00000000      ................
48000030: 00002011 00000000 00000001 00000000      . ............
```

Here, the sf read command reads 16 bytes from the QSPI offset address 0x10 to the memory 0x48000000 address. Since data has previously been written at the 0x10 address of the QSPI, you can see the "myir qspi test" written in the file system.

- **Write data after emptying memory**

After emptying the memory, write the memory back to 16 bytes of data, and then write a 16-byte 0x12 to the 0x100 offset of qspi:

```
ZynqMP>mw.b 0x48000000 0xff 0x10
ZynqMP>mw.b 0x48000000 0x12 0x10
ZynqMP>md 0x48000000 0x10

48000000: 12121212 12121212 12121212 12121212      ................
48000010: 00000000 00000000 00002014 00000000      ........ ......
48000020: 00000000 00000000 00000000 00000000      ................
48000030: 00002011 00000000 00000001 00000000      . ............
```

```
ZynqMP>sf write 0x48000000 0x100 0x10
device 0 offset 0x100, size 0x10
SF: 16 bytes @ 0x100 Written: OK
u-boot=>
```

- **Read again after emptying the memory**

Empty the memory content and read the contents written to qspi into memory again:

```
ZynqMP>mw.b 0x48000000 0xff 0x10
ZynqMP>sf read 0x48000000 0x100 0x10
device 0 offset 0x100, size 0x10
SF: 16 bytes @ 0x100 Written: OK
ZynqMP> mw.b 0x48000000 0xff 0x10
ZynqMP>sf read 0x48000000 0x100 0x10
device 0 offset 0x100, size 0x10
SF: 16 bytes @ 0x100 Read: OK
ZynqMP>md 0x48000000 0x10

48000000: 12121212 12121212 12121212 12121212    ................
48000010: 00000000 00000000 00002014 00000000    ........ ......
48000020: 00000000 00000000 00000000 00000000    ................
48000030: 00002011 00000000 00000001 00000000    . ............
```

You can see that the written data can be read out of the QSPI properly, and here you can also refer to the method read by the file system to view the qspi content.

2.5. RTC

This RTC (Real-time clock) is a clock that records the real time, and when the software system shuts down, the system retains the system time and continues to time, and the system is turned on again to synchronize the time into the software system.

MYS-ZU5EV has an internal RTC. RTC testing is usually performed in conjunction with the hwclock and date commands commonly used in Linux systems, and the following tests write system time to RTC, read RTC time and set it to system time and perform time-down-to-hold tests.

- **View the system RTC device**

```
root@mys_zu5ev # ls /dev/rtc* -al
lrwxrwxrwx    1 root     root             4 Apr  1 07:24 /dev/rtc -> rtc0
crw-------    1 root     root      252,   0 Apr  1 07:24 /dev/rtc0
root@mys_zu5ev #  cat /sys/class/rtc/rtc0/name
rtc_zynqmpffa60000.rtc
root@mys_zu5ev #
```

This RTC belongs to a Linux device and has its device node rtc0 under /dev for the user to operate on.

- **Set the system time.**

When setting the system time to Sat Jan 30 21:15:30 UTC 2021:

```
root@mys_zu5ev # date  013021152021.30
Sat Jan 30 21:15:30 UTC 2021
```

- **Write system time to RTC**

Write the system time set by the previous date command to the RTC device:

```
root@mys_zu5ev # hwclock -w
```

- **Read the RTC time and set it to system time**

```
root@mys_zu5ev # hwclock -r
2021-01-30 21:16:18.178677+00:00
```

- **Power down to maintain RTC time**

turn off the development board and disconnect the power, after about 2 minutes, power on again and start up. View RTC time and system time:

```
root@mys_zu5ev # hwclock -r
2021-01-30 21:43:10.955237+00:00
```

After rebooting, the RTC time and system time checked increased by about 2 minutes compared with the previous setting, indicating that the RTC was working normally. If the precision of the RTC needs to be tested in detail, the power off time can be extended to say 24 hours to test the difference between the RTC time and the standard time..

- **Synchronize system time with RTC time**

```
root@mys_zu5ev # hwclock -s
root@mys_zu5ev # date
Sat Jan 30 21:43:46 UTC 2021
```

If you add the "hwclock -s" command to the boot script, you can ensure that the system time and the RTC time are in sync each time you boot.

2.6. Watchdog

The Linux kernel contains the Watchdog subsystem. In the hardware design process, the Watchdog timer inside the chip can be used or the external Watchdog chip can be used to realize the function of the Watchdog, which is used to monitor the operation of the system. The system can reset automatically when abnormal conditions occur and the dog cannot be fed. MYS-ZU5EV uses the external watchdog chip CAT823TTDI-GT3. The dog feed pin of the chip is connected to the PS_MIO38 pin of the CPU. The watchdog also has a switch connected to the PS_MIO33 pin.

When using watchdog, you need to pay attention to some characteristics of watchdog. Generally, there are two points as follows:

This section demonstrates the use of Watchdog, simulates the Watchdog system reset function, and provides a modified example to set the Watchdog timeout.

1) Turn off the watchdog test

Put the watchdog switch PS_MIO33 pin zero and try to turn off the watchdog:

```
root@mys_zu5ev # echo 0 >/sys/class/leds/wdt_en/brightness
```

2) Watchdog feeds dog for testing

```
root@mys_zu5ev #echo 200 >/sys/class/myir-watchdog/wd_period_ms
root@mys_zu5ev # echo 1>/sys/class/leds/wdt_en/brightness
root@mys_zu5ev # echo 4000 >/sys/class/myir-watchdog/wd_period_ms
```

If feeding the dog times out, the board will be reset by the watchdog.

3. Basic peripheral interface

3.1. GPIO

The GPIO test is carried out through the file system sysfs interface, and the following article takes MIO6 as an example to illustrate the use of the zynqMP platform GPIO.

In the linux system of ZYNQMP platform, the Gpio number consists of Gpio Chip Id and MIO number. EMIO's Gpio number consists of Gpio Chip Id and 78 plus EMIO number, where ChipID can be viewed in one way:

```
root@mys_zu5ev #ls /sys/class/gpio
export          gpiochip338   unexport
cat /sys/class/gpio/gpiochip338/base
338
```

ChipID is 338, so MIO6's Gpio number is 338+6=344

1) Export GPIO

```
root@mys_zu5ev # echo 344 > /sys/class/gpio/export
```

After successful export, the gpio344 directory will be generated in the /sys/class/gpio/ directory.

2) set/view GPIO direction

- **set input**

```
root@mys_zu5ev # echo "in" > /sys/class/gpio/gpio344/direction
```

- **set output**

```
root@mys_zu5ev # echo "out" > /sys/class/gpio/gpio344/direction
```

- **view gpio direction**

```
root@mys_zu5ev # cat /sys/class/gpio/gpio344/direction
out
```

Returns in for input and out for output.

3) set/view GPIO value

- **set output to low**

```
root@mys_zu5ev # echo "0" > /sys/class/gpio/gpio344/value
```

- **set output to high**

```
root@mys_zu5ev # echo "1" > /sys/class/gpio/gpio344/value
```

- **view gpio value**

```
root@mys_zu5ev # cat /sys/class/gpio/gpio344/value
1
```

3.2. LED

This Linux system provides a separate subsystem to facilitate the operation of LED devices from user space. This subsystem provides the interface to the 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 development board. Let's test the LED by reading and writing sysfs. The following commands are all general commands and are also general methods of manipulating LED.

1) The directory for operating LED is /sys/class/leds

```
root@mys_zu5ev # ls /sys/class/leds/
led_sys@  mmc0::@  mmc1::@  rs485_de@  wdt_en@
```

The led_sys is a customizable led for the core board, which changes the duty-out ratio of the heartbeat by writing different values to /sys/class/leds/led_sys/brightness.

2) Test the LED with heartbeat lamp 1 as an example

- **Turn off the heartbeat light**

```
root@mys_zu5ev # echo none > /sys/class/leds/led_sys/trigger
```

Turn off the heartbeat light and then turn the led off and on separately.

- **Turn off LED**

```
root@mys_zu5ev # echo 1 > /sys/class/leds/led_sys/brightness
```

- **Turn on LED**

```
root@mys_zu5ev # echo 0 >/sys/class/leds/led_sys/brightness
```

- **Turn on LED trigger mode**

When the "Timer" starting mode is turned on, the LED flashes in a 1Hz cycle by default, and the duty cycle is 50% :

```
root@mys_zu5ev #  echo "timer" >/sys/class/leds/led_sys/trigger
```

- **Change the on/off duty cycle when the LED is flashing**

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

```
root@mys_zu5ev # echo "100" >/sys/class/leds/led_sys/delay_on
root@mys_zu5ev # echo "500" >/sys/class/leds/led_sys/delay_off
```

3.3. USB

In this section, the feasibility of USB Host driver is verified by relevant commands or hot-swappable and USB HUB, and the functions of reading and writing U disk and USB enumeration are realized.

1) View the printed information inserted into the USB

● View USB device information

Connect the USB disk to the USB Host interface of the development board. The kernel prompt message is as follows:

```
root@mys_zu5ev #
[  755.579811] usb 2-1.4: new SuperSpeed Gen 1 USB device number 3 using xhci-hcd
[  755.628681] usb 2-1.4: New USB device found, idVendor=1f75, idProduct=0918, bcdDevice= 3.10
[  755.637028] usb 2-1.4: New USB device strings: Mfr=2, Product=3, SerialNumber=4
[  755.644337] usb 2-1.4: Product: STORAGE DEVICE
[  755.648779] usb 2-1.4: Manufacturer: Device Storage
[  755.653651] usb 2-1.4: SerialNumber: 09294138
[  755.658659] usb-storage 2-1.4:1.0: USB Mass Storage device detected
[  755.665157] scsi host0: usb-storage 2-1.4:1.0
[  756.674979] scsi 0:0:0:0: Direct-Access     Specific STORAGE DEVICE   0009 PQ: 0 ANSI: 6
[  756.684483] sd 0:0:0:0: [sda] 61440000 512-byte logical blocks: (31.5 GB/29.3 GiB)
[  756.692636] sd 0:0:0:0: [sda] Write Protect is off
[  756.697992] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[  756.723524]  sda: sda1
[  756.727437] sd 0:0:0:0: [sda] Attached SCSI removable disk
[  756.889397] FAT-fs (sda1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
```

```
[  756.898896] vfat filesystem being mounted at /run/media/sda1 supports ti
mestamps until 2107 (0x10391447e)
```

From the above information, we can conclude that the device to be mounted is sda1.

2) U disk mount read and write

- **mount USB disk**

USB storage devices are automatically mounted in the provided boot image.

- **Read file**

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

```
root@mys_zu5ev # ls /run/media/sda1/
test.txt
root@mys_zu5ev # cat /run/media/sda1/test.txt
hello world!
```

- **Write file**

```
root@mys_zu5ev # touch test.txt
root@mys_zu5ev # echo "hello world !!!" > test.txt
root@mys_zu5ev # cp test.txt /run/media/sda1
root@mys_zu5ev # cat /run/media/sda1/test.txt
hello world !!!
```

After writing the file, you need to execute the sync command to ensure that the data is completely written to the USB disk before you can unload the USB disk device.

3) Umount U disk

- **Umount operation**

```
root@mys_zu5ev # umount /run/media/sda1/
```

3.4. Micro SD card

Micro SD Card, formerly known as Trans-Flash Card(TF Card), Micro SD Card is a very small flash memory Card. The Micro SD card is smaller than the standard SD card and is the smallest SD card in its type. Although the shape of the Micro SD card and the interface shape are different from the original SD card, the interface specification remains the same to ensure compatibility. If the Micro SD is inserted into a specific transfer card, it can be used as a standard SD card. SD card has become the most widely used memory card in the current consumer digital devices. It is a multi-functional memory card with large capacity, high performance, security and other characteristics. The back of the Micro SD card generally has 9 pins, including 4 data lines, supporting two data transmission widths of 4bit.

Zynqmp series of chips support 2-channel SD interface, and SD0 is used to connect Micro SD on the MYS-ZU5EV development board. The hardware specifications of this interface are as follows:

- is fully compliant with the SDIO 2.0 interface specification
- Support SDHC Class 10 microSD cards

This section will explain the steps and methods for viewing and manipulating TF cards under Linux.

4) View the TF card capacity

You can query the partition information and capacity of TF card by "fdisk -l " command:

```
root@mys_zu5ev # fdisk -l
……
Disk /dev/mmcblk1: 14.5 GiB, 15552479232 bytes, 30375936 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xdc3c8c4b
```

```
Device          Boot    Start       End   Sectors     Size  Id  Type
/dev/mmcblk1p1  *          63   2088449   2088387  1019.7M   c  W95 FAT32 (LB
A)
/dev/mmcblk1p2       2088450  30362849  28274400    13.5G  83  Linux
```

- /dev/mmcblk1p1：Use to store BOOT.bin、image.ub and other BOOT files
- /dev/mmcblk1p2：Used to store the root file system partition

5) Performance test of TF card

Performance test mainly tests the speed of file reading and writing of TF card under Linux system, generally combining time and dd double command to test. To mount the TF card partition that you want to test, take the first partition /dev/mmcblk1p1 as an example. The mount directory is /mnt/mmcblk1p1/.

● Write file testing

```
root@mys_zu5ev #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, 16.6057 s, 6.3 MB/s

real    0m16.609s
user    0m0.008s
sys     0m0.565s
```

When writing to a file using dd, the "conv=fdatasync" parameter is added to indicate that when dd is written N times, the flush cache is synchronized to disk. Because a write to disk is usually written to the cache and returned before it is written to disk. The write disk speed tested here is 6.3MB/s.

● Read file testing

Ignore the cache effect when reading files. This is where you can specify the parameter "iflag=direct, nonblack".

```
root@mys_zu5ev #  time dd if=tempfile of=/dev/null bs=1M count=100 iflag
=direct,nonblock
100+0 records in
```

```
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 4.45101 s, 23.6 MB/s

real    0m4.454s
user    0m0.000s
sys     0m0.027s
```

It can be seen that the speed of reading data directly from SD card is 23.6MB/s.

3.5. Display

MYS-ZU5EV currently supports the DP display. DP interface supports up to 4K 60FPS resolution output.

1) DP display testing

Since MYS-ZU5EV is displayed as DP, enter the Linux system and execute the following command:

```
root@mys_zu5ev:~#/etc/init.d/xserver-nodmstop
root@mys_zu5ev:~#/usr/share/examples/widgets/painting/deform/deform  -platform  linuxfb:fb=/dev/fb0
```

You can see the Qt interface on the DP monitor.

3.6. Ethernet

Net-tools, iproute2, systemd-networkd, network manager and connman are the most common tools for network configuration under Linux. All of these can be selected according to the actual needs of the system customization. Here are some common methods of manual temporary configuration and automatic permanent configuration of Ethernet.

1) Manually configure the Ethernet IP address temporarily

● manually configure the network using ifconfig from the Net-Tools toolkit

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

```
root@mys_zu5ev # ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0a:35:00:01:22
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
Interrupt:28 Base address:0xb000
```

Eth0 is the actual Ethernet device, which defaults to a fixed hardware MAC address

IP address 192.168.0.100 (192.168.0.100)

```
root@mys_zu5ev # ifconfig eth0 192.168.0.100 netmask 255.255.255.0 up
```

The above command manually configues the 192.168.0.100 IP address, 255.255.255.0 subnet mask, and the default broadcast address 192.168.0.255, and activates it with the UP parameter as shown below:

```
root@mys_zu5ev # ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0a:35:00:01:22
          inet addr:192.168.0.100  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
```

```
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
Interrupt:28 Base address:0xb000
```

- **manually configure the network using the IP command in the iproute2 toolkit**

The ifconfig command to manually set the IP address of the method can also use the IP addr and IP link for alternative, more information please view the instructions in https://wiki.linuxfoundation.org/networking/iproute2

```
root@mys_zu5ev # ip addr flush dev eth0
root@mys_zu5ev # ip addr add 192.168.0.101/24 brd + dev eth0
root@mys_zu5ev # ip link set eth0 up
```

If IP address has been configured before, the IP address configured with IP addr add will become secondary address. So, use IP addr flush to clear the previous address before configuration and activate. After completing the configuration, the eth0 information is viewed through the IP addr show command as follows:

```
root@mys_zu5ev # ip addr show eth0
3: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state
DOWN group default qlen 1000
    link/ether fc:69:47:33:a5:65 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.101/24 brd 192.168.1.255 scope global eth0
       valid_lft forever preferred_lft forever
```

2) Automatic permanent configuration of Ethernet IP address

The IP address configured by ifconfig command and IP command will be lost after power failure. If you need to make the IP address permanent, you need to modify the corresponding configuration file of network management tool.

- **Automatic permanent configuration and dynamic acquisition of IP address**

MYS-ZU5EV development board automatically obtains the IP address through DHCP by default. For details, you can view the configuration file /etc/network/interfaces.

```
root@mys_zu5ev # cat /etc/network/interfaces -- configuration file for ifup(8),
 ifdown(8)
# The loopback interface
auto lo
iface lo inet loopback
…
auto eth0
iface eth0 inet dhcp
iface eth1 inet dhcp
…
```

The above configuration file will configure the network card matching eth0. If the kernel boot parameter does not contain the nfsroot parameter, the IP address, gateway, DNS and other information will be automatically configured for the matching network card through DHCP. Please refer to the following links for detailed configuration parameters: https://wiki.debian.org/NetworkConfiguration.

- **Automatic permanent configuration of static IP address**

If you need to configure the permanent IP address for eth0, you need to modify the /etc/network/interfaces file as follows:

```
# Wired or wireless interfaces
auto eth0
#iface eth0 inet dhcp
iface eth0 inet static
        address 192.168.30.100
        netmask 255.255.255.0
gateway 192.168.30.1
        dns-nameservers 223.5.5.5 114.114.114.114
iface eth1 inet dhcp
```

After the configuration is completed, execute the following command to restart the network service. After restarting, you will find that the address of eth0 network card has been configured to 192.168.30.100, as shown below:

```
root@mys_zu5ev #/etc/init.d/networking restart
root@mys_zu5ev #ifconfig eth0
```

```
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:22:01
          inet addr:192.168.30.100  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::20a:35ff:fe00:2201/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6756 errors:0 dropped:0 overruns:0 frame:0
          TX packets:715 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:386149 (377.0 KiB)  TX bytes:105843 (103.3 KiB)
          Interrupt:30
```

3.7. CAN

This section uses cansend and candump commands commonly used in Linux system to test socket can communication. The two development boards used in this test are butt testing.

- **Initialize can network interface**

Refer to the following command to set the baud rate of two development boards to 500K respectively:

```
root@mys_zu5ev #ip link set can0 up type can bitrate 500000
[ 2056.319008] xilinx_can ff070000.can can0: bitrate error 0.0%
[ 2056.324869] IPv6: ADDRCONF(NETDEV_CHANGE): can0: link becomes ready
```

At this point, the can function is turned on.

- **send data**

Set one board to send, and use cansend to send 64 byte fixed format string to test:

```
root@mys_zu5ev #cansend can0 123#DEADBEEF
root@mys_zu5ev #cansend can0 123#DEADBEEF
```

- **receive data**

Set another board to receive. You can use candump to view the received data of can:

```
root@mys_zu5ev # candump can0 -L
(1621329829.902018) can0 123#DEADBEEF
(1621329829.902076) can0 123#DEADBEEF
```

- **Statistics can0 information**

After the can data is sent and received, the details and statistics of can devices are displayed. The value of "clock" represents the clock of can, the value of "drop" represents packet loss, the value of "overflow" represents overflow, and the value of "error" represents bus error.

```
root@mys_zu5ev # ip -details -statistics link show can0
```

```
2: can0: <NOARP,UP,LOWER_UP,ECHO> mtu 16 qdisc pfifo_fast state UP mod
e DEFAULT group default qlen 10
    link/can  promiscuity 0 minmtu 0 maxmtu 0
    can <LOOPBACK> state ERROR-ACTIVE (berr-counter tx 0 rx 0) restart-ms
 0
          bitrate 499999 sample-point 0.875
          tq 250 prop-seg 3 phase-seg1 3 phase-seg2 1 sjw 1
          xilinx_can: tseg1 1..16 tseg2 1..8 sjw 1..4 brp 1..256 brp-inc 1
          clock 99999999
          re-started bus-errors arbit-lost error-warn error-pass bus-off
          0          0          0          0          0          0          num
txqueues 1 numrxqueues 1 gso_max_size 65536 gso_max_segs 65535
    RX: bytes  packets  errors  dropped overrun mcast
    20         3        0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    20         3        0       0       0       0
```

4. Network application

The image burned in the development board contains some common network applications by default, which is convenient for users to develop or debug.

4.1. PING

Ping is mainly used to test the connectivity of the network, and also can test the network delay and packet loss rate. After the Ethernet connection is configured in 4.1.1, Ping can be used to conduct simple test on the network connection.

1) Wiring and information output

Connect the development board to the switch or router through CAT6 network cable, and the console will display the connection information output by the kernel, as follows:

```
root@mys_zu5ev # [  208.032262] macb ff0e0000.ethernet eth0: link up (1000/Full)
```

2) Test Internet address

```
root@mys_zu5ev # ping www.baidu.com -I eth0
PING www.baidu.com (14.215.177.38): 56 data bytes
64 bytes from 14.215.177.38: seq=0 ttl=53 time=19.013 ms
64 bytes from 14.215.177.38: seq=1 ttl=53 time=18.602 ms
64 bytes from 14.215.177.38: seq=2 ttl=53 time=18.590 ms
64 bytes from 14.215.177.38: seq=3 ttl=53 time=18.973 ms
```

Note: Ping public network needs to ensure that DNS works normally.

The results above show that www.baidu.com after domain name resolution, the IP address is 14.215.177.38, icmp_seq represents the number of icmp packets. If the number is continuous, it means that there is no packet loss; Time represents the delay time of the response. Of course, the shorter the time, the better.

4.2. SSH

SSH is the abbreviation of secure shell, which is developed by IETF's network working group; SSH is a security protocol based on application layer, which is reliable and provides security for remote login session and other network services. Generally, dropbear or openssh is used to implement ssh server and client on Linux platform. Next, test the use of SSH client and server on Ethernet connection. Currently, openssh 7.6p1 is included by default (http://www.openssh.com/) Provide the client and service program.

First, configure the connection between the Ethernet interface of the development board and the ssh server, and the configured Ethernet card address is as follows:

```
root@mys_zu5ev # ifconfig
eth0      Link encap:Ethernet  HWaddr 26:BB:D9:BC:8C:DC
          inet addr:192.168.40.172  Bcast:192.168.40.255  Mask:255.255.255.0
          inet6 addr: fe80::24bb:d9ff:febc:8cdc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:383 errors:0 dropped:0 overruns:0 frame:0
          TX packets:95 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:22328 (21.8 KiB)  TX bytes:9174 (8.9 KiB)
          Interrupt:22
```

The IP address of ssh server is 192.168.40.172. Ping command can be used to test whether the connection between development board and ssh server is normal.

● SSH client test

The development board connects to the ssh server as a client, and logs in to the server with the SSH command on the development board. The commands and results are as follows:

```
root@mys_zu5ev # ssh fengy@192.168.40.2
The authenticity of host '192.168.40.2 (192.168.40.2)' 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 '192.168.40.2' (ECDSA) to the list of known host
s.
fengy@192.168.40.2's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-101-generic x86_64)

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

1677 个可升级软件包。
306 个安全更新。

New release '18.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu May 27 15:37:37 2021 from 192.168.40.37
fengy@myir-server1:~$
```

"fengy" is the user name on the server.

After successful login, you can automatically enter the console on the ssh server, and then the user can control the remote server within fengy user rights on the client. If you need to exit, you can directly execute the "exit" command in the current console.

● SSH server test

The development board is the SSH server, and other external devices are connected to the development board remotely.

The development board starts SSH service by default, so we can also use ssh command to log in to the current development board on other external devices (development board or PC) with SSH client

```
$ ssh root@192.168.40.172
```

In the above example, we login to the development board with the root account remotely, and enter the console to control the development board within the permission of the root user. If you need to exit, you can directly execute the "exit" command in the console.

OpenSSH is the main connection tool for remote login using SSH protocol. It encrypts all traffic to eliminate eavesdropping, connection hijacking and other attacks. In addition, OpenSSH also provides a series of large-scale security tunnel functions, a variety of authentication methods and complex and flexible configuration options. Users can modify the configuration file ssh_config and sshd_ config in the host /etc/ssh/ directory according to their own needs.
For example, if you want the SSH server to allow the root account to login remotely without password, you can modify /etc/ssh/sshd_config on the SSH server, add the following two lines of configuration.

```
PermitRootLogin  yes
PermitEmptyPasswords  yes
```

The above configuration has a large security risk and is generally used for remote deployment in the debugging phase. Considering the safety of the actual products, they are generally turned off.

4.3. SCP

SCP is the abbreviation of Secure Copy. It is a secure remote file copy command based on SSH protocol in Linux system. It is very practical in the system debugging stage.

We have already introduced the example of using SSH protocol and SSH client and server for remote login. Here we will introduce the example of remote file copy through SCP command

1) Copy files from remote to local

```
fengy@myir-server1:~$ scp BOOT.bin root@192.168.40.172:/root/
root@192.168.40.172's password:
BOOT.bin
                    100% 8790KB    8.6MB/s    00:01
```

You can see this file in the home directory of the development board, as follows:

```
root@mys_zu5ev # ls
BOOT.BIN
```

2) Copy files from local to remote

```
root@mys_zu5ev # scp BOOT.bin fengy@192.168.40.2:/home/fengy
fengy@192.168.40.2's password:
```

During the copy process, you need to follow the prompts to enter. After the verification is successful, the file is copied from the development board to the "$home" directory of the account specified on the server.

You can also copy the directory by adding the "-r" parameter. Please refer to the help of scp command for specific operations.

4.4. TFTP

Like FTP, TFTP uses client and server software to connect and transfer files between different devices. However, TFTP uses UDP protocol, which does not have login function. It is very simple, especially suitable for transmitting and backing up firmware, configuration file and other information on the device and server side. For example, the common u-boot supports TFTP protocol. It can load the Linux system on the server side and realize the function of network startup through the network.

The default mirror file contains the TFTP client program provided by busybox, and its command syntax is as follows:

```
root@mys_zu5ev # tftp --help
BusyBox v1.31.0 (2020-05-26 14:38:25 UTC) multi-call binary.

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

The detailed parameters are as follows:

- -g : get file
- -p : upload file
- -l : Local file
- -r : Remote files
- HOST: remote host IP address
- [PORT]: negligible, default is 69

TFTP server can choose tftp-hpa on Linux platform or tftpd32/64 on Windows platform (http://tftpd32.jounin.net/tftpd32_download.html). The following is an example of Ubuntu platform to illustrate the configuration of the TFTP server.

1) Install TFTP server

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

2) Configure TFTP service

Create the TFTP server working directory and open the TFTP service profile as follows:

```
$ 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

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

After the TFTP server is configured, a test file named test_file is placed in the <WORKDIR>/tftpboot/ directory configured above, and the TFTP client can be used to download and upload files on the development board.

```
root@mys_zu5ev # tftp -g -r test_file 192.168.0.2
```

The above command will download the test_file file of <WORKDIR>/tftpboot directory to the current directory of the development board.

```
root@mys_zu5ev # tftp -p -r test_file 192.168.0.2
```

The above command will put the test_file file in the current directory of the development board into the <WORKDIR>/tftpboot directory configured before the TFTP server.

4.5. DHCP

DHCP (Dynamic Host Configuration Protocol) is a local area network protocol. It means that the server controls a range of IP addresses, and the client can automatically obtain the IP address and subnet mask assigned by the server when logging in to the server.

This paper introduces the method of using udhcpc command to obtain IP address manually, which is convenient for users to use when debugging the network. Connect the development board and router with CAT6 network cable, manually assign IP address to eth0 network card with command, and observe the process of obtaining IP by DHCP.

- **Use udhcpc command to configure IP address**

```
root@mys_zu5ev # udhcpc -i eth0
udhcpc: started, v1.29.3
udhcpc: sending discover
udhcpc: sending select for 192.168.40.172
udhcpc: lease of 192.168.40.172 obtained, lease time 7200
deleting routers
adding dns 223.5.5.5
adding dns 201.104.111.114
```

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

```
root@mys_zu5ev # ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 26:BB:D9:BC:8C:DC
          inet addr:192.168.40.172  Bcast:192.168.40.255  Mask:255.255.255.0
          inet6 addr: fe80::24bb:d9ff:febc:8cdc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:17369 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11359 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10147310 (9.6 MiB)  TX bytes:9862414 (9.4 MiB)
          Interrupt:22
```

```
[root@myir ~]# cat /etc/resolv.conf
# Generated by Connection Manager
nameserver ::1
nameserver 127.0.0.1
search bad # eth0
nameserver 223.5.5.5 # eth0
nameserver 201.104.111.114 # eth0
```

4.6. Iptables

Iptables is a management tool for IPv4 packet filtering and NAT. It is used to set, maintain and check the IP packet filtering rule table in 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 matched packets.
The development board using Linux system usually uses iptables tool to configure firewall. Iptables processes all kinds of packets according to the methods defined by packet filtering rules, such as accept, reject and drop. Next, we use iptables to test the interception of ICMP packets and prohibit other external devices on the network from pinging them. For specific commands, please refer to: https://linux.die.net/man/8/iptables

1) Configuration development board iptables

On the development board, use iptables configuration to discard the ICMP packets and do not respond to Ping probes from other hosts. The command is as follows:

```
root@mys_zu5ev # iptables -A INPUT -p icmp --icmp-type 8 -j DROP
root@mys_zu5ev # iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -p icmp -m icmp --icmp-type 8 -j DROP
```

2) Ping test

Ping the development board on the development host and specifying the deadline as 10, the results are as follows:

```
PC$ ping 192.168.40.172 -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 development board after setting the firewall.

3) Delete the corresponding firewall rules

```
root@mys_zu5ev # iptables -F
root@mys_zu5ev # iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
```

4) Ping development board again

```
PC$ ping 192.168.40.172 -w 5
PING 192.168.40.172 (192.168.40.172) 56(84) bytes of data.
64 bytes from 192.168.40.172: icmp_seq=1 ttl=64 time=0.254 ms
64 bytes from 192.168.40.172: icmp_seq=2 ttl=64 time=0.219 ms
64 bytes from 192.168.40.172: icmp_seq=3 ttl=64 time=0.222 ms
64 bytes from 192.168.40.172: icmp_seq=4 ttl=64 time=0.226 ms
64 bytes from 192.168.40.172: icmp_seq=5 ttl=64 time=0.238 ms
64 bytes from 192.168.40.172: icmp_seq=6 ttl=64 time=0.236 ms

--- 192.168.40.172 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
```

After clearing the iptables rule, ping the development board from the development host again, and you can ping. The above example is just a simple demonstration. In fact, iptables can achieve very powerful functions with various rules, which will not be described in detail here.

4.7. Ethtool

Ethtool is a tool to view and modify the parameters of Ethernet devices. It plays a certain role in the network debugging stage. Let's use this command to view the information of Ethernet card and try to modify its parameters.

First, we can view the help information of the command through "ethtool –h":

```
root@mys_zu5ev # ethtool --help
ethtool version 5.2
Usage:
        ethtool DEVNAME Display standard information about device
        ethtool -s|--change DEVNAME     Change generic options
                [ speed %d ]
                [ duplex half|full ]
                [ port tp|aui|bnc|mii|fibre ]
                [ mdix auto|on|off ]
                [ autoneg on|off ]
                [ advertise %x ]
                [ phyad %d ]
                [ xcvr internal|external ]
                [ wol p|u|m|b|a|g|s|f|d... ]
                [ sopass %x:%x:%x:%x:%x:%x ]
                [ msglvl %d | msglvl type on|off ... ]
        ethtool -a|--show-pause DEVNAME Show pause options
.....................
```

To view the basic information of the Ethernet card on the development board:

```
root@mys_zu5ev # ethtool eth0
Settings for eth0:
        Supported ports: [ TP MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Supported pause frame use: Symmetric
        Supports auto-negotiation: Yes
```

```
        Supported FEC modes: Not reported
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Advertised pause frame use: No
        Advertised auto-negotiation: Yes
        Advertised FEC modes: Not reported
        Link partner advertised link modes:  10baseT/Half 10baseT/Full
                                             100baseT/Half 100baseT/Full
                                             1000baseT/Full
        Link partner advertised pause frame use: Symmetric Receive-only
        Link partner advertised auto-negotiation: Yes
        Link partner advertised FEC modes: Not reported
        Speed: 1000Mb/s
        Duplex: Full
        Port: MII
        PHYAD: 4
        Transceiver: internal
        Auto-negotiation: on
        Link detected: yes
```

Through the ethtool command, we can see that the current connection modes supported by the Ethernet card are 10 MB, 100 MB and Gigabit, half duplex and full duplex, the current connection status is negotiated Gigabit, full duplex, MII interface, PHY address is 6, etc.

We can also use the ethtool tool to set the parameters of Ethernet, which can play a certain role in Ethernet debugging and diagnosis. For example, we force the Ethernet to be 100 MB full duplex and turn off self negotiation. The command is as follows:

```
root@mys_zu5ev # ethtool -s eth0 speed 100 duplex full autoneg off
```

For more information about ethtool, please refer to: http://man7.org/linux/man-pages/man8/ethtool.8.html。

4.8. iperf3

Iperf3 is a tool to measure the maximum bandwidth of IP network. It supports adjusting test time, buffer size and protocol (TCP, UDP, SCTP under IPv4 and IPv6) and other parameters. According to roles, iperf3 can be divided into server mode or client mode. We can use it to test and view network bandwidth, TCP window value, retransmission probability, etc. in TCP mode, or test packet loss rate, delay and jitter under specified UDP bandwidth.

We take a Ubuntu 16.04 system, the server with Giga network card as the server of iperf3, and the tested development board as the client to test the performance of TCP and UDP respectively.

First, install iperf3 on the server, as follows:

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

Connect the server and development board directly through CAT6 network line and configure their IP addresses. For example, we set the server IP to 192.168.40.143, and the development board IP to 192.168.40.100, and use ping command test to ensure that they are connected.

Note: try not to connect router or switch as far as possible, so as to avoid the test result affected by transmission and forwarding of intermediate equipment.

1) Testing TCP Performance

- **Service end (192.168.40.143)**

The iperf3 on the server uses the -s parameter to represent the working mode on the server side.

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

- **Client (192.168.40.100)**

The iperf3 program running on the development board works in the client, TCP mode, and the parameters are described as follows:

- -c 192.168.40.143 : working on the client, connecting to the server 192.168.40.143
- -i 2 : the test result report interval is 2 seconds
- -t 10 : the total test time is 10 seconds

```
root@mys_zu5ev #  iperf3 -c 192.168.40.143 -i 2 -t 10
Connecting to host 192.168.40.143, port 5201
[  5] local 192.168.40.100 port 43446 connected to 192.168.40.143 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-2.00   sec   223 MBytes   933 Mbits/sec    0    714 KBytes

[  5]   2.00-4.00   sec   218 MBytes   915 Mbits/sec   49    667 KBytes

[  5]   4.00-6.00   sec   221 MBytes   927 Mbits/sec   29    556 KBytes

[  5]   6.00-8.00   sec   220 MBytes   922 Mbits/sec    0    591 KBytes

[  5]   8.00-10.00  sec   219 MBytes   917 Mbits/sec    0    597 KBytes

- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  1.07 GBytes   923 Mbits/sec   78             sender
[  5]   0.00-10.00  sec  1.07 GBytes   921 Mbits/sec                  receiver

iperf Done.
```

After 10 seconds, the client end the test and display the above test results, indicating that the TCP bandwidth is about 923 Mbps, no retransmission, and the TCP window value is 597 Kbytes

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

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

```
Accepted connection from 192.168.40.100, port 43444
[  5] local 192.168.40.143 port 5201 connected to 192.168.40.100 port 43446
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-2.00   sec   220 MBytes   921 Mbits/sec
[  5]   2.00-4.00   sec   219 MBytes   917 Mbits/sec
[  5]   4.00-6.00   sec   221 MBytes   925 Mbits/sec
[  5]   6.00-8.00   sec   219 MBytes   920 Mbits/sec
[  5]   8.00-10.00  sec   219 MBytes   917 Mbits/sec
[  5]  10.00-10.02  sec  1.38 MBytes   530 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  5]   0.00-10.02  sec  1.07 GBytes   921 Mbits/sec   78             sender
[  5]   0.00-10.02  sec  1.07 GBytes   919 Mbits/sec                  receiver
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

2) Test UDP performance

● Service (192.168.40.143)

The server continues to run iperf3 using the -s parameter to indicate that it works in server mode.

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

● Client (192.168.40.100)

On the development board, iperf3 works on the client in UDP mode, and the parameters are described as follows:

- -u : working in UDP mode
- -c 192.168.40.143 : working on the client, connecting to the server 192.168.40.143
- -i 2 : the test result report interval is 2 seconds

- -t 10 : the total test time is 10 seconds
- -b 100M : set the UDP transmission bandwidth to 100Mbps

```
root@mys_zu5ev # iperf3 -u -c 192.168.40.143 -i 2 -t 10 -b 100M
Connecting to host 192.168.40.143, port 5201
[  5] local 192.168.40.100 port 50499 connected to 192.168.40.143 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  5]   0.00-2.00   sec  23.8 MBytes   100 Mbits/sec  17263
[  5]   2.00-4.00   sec  23.8 MBytes   100 Mbits/sec  17264
[  5]   4.00-6.00   sec  23.8 MBytes   100 Mbits/sec  17265
[  5]   6.00-8.00   sec  23.8 MBytes   100 Mbits/sec  17265
[  5]   8.00-10.00  sec  23.8 MBytes   100 Mbits/sec  17265
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagra
ms
[  5]   0.00-10.00  sec   119 MBytes   100 Mbits/sec  0.000 ms   0/86322
(0%)  sender
[  5]   0.00-10.00  sec   119 MBytes  99.7 Mbits/sec  0.176 ms   243/86321
(0.28%)  receiver

iperf Done.
```

After 10 seconds, the client end the test and display the above test results. 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
```

```
root@myir:~# iperf3 -u -c 192.168.40.143 -i 2 -t 10 -b 100M
Connecting to host 192.168.40.143, port 5201
[  5] local 192.168.40.100 port 50499 connected to 192.168.40.143 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  5]   0.00-2.00   sec  23.8 MBytes   100 Mbits/sec  17263
[  5]   2.00-4.00   sec  23.8 MBytes   100 Mbits/sec  17264
[  5]   4.00-6.00   sec  23.8 MBytes   100 Mbits/sec  17265
```

```
[  5]   6.00-8.00   sec  23.8 MBytes   100 Mbits/sec  17265
[  5]   8.00-10.00  sec  23.8 MBytes   100 Mbits/sec  17265
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagra
ms
[  5]   0.00-10.00  sec   119 MBytes   100 Mbits/sec  0.000 ms   0/86322
(0%)  sender
[  5]   0.00-10.00  sec   119 MBytes  99.7 Mbits/sec  0.176 ms   243/86321
(0.28%)  receiver

iperf Done.
```

The client modifies the -b parameter and continues to increase the specified UDP bandwidth. When packet loss begins, the measured UDP bandwidth is the actual UDP bandwidth. There are many parameters that can be configured in the testing process of iperf3. Users can adjust the testing according to the actual application needs. For example, you can increase the value of -t parameter for long-time stress testing, or specify -P parameter for concurrent stress testing of multiple connections. For more information about iperf3 testing, please refer to : https://iperf.fr/iperf-doc.php#3doc.

5. Graphic system

This chapter mainly tests the graphics display of MYS-ZU5EV.

Linux graphics system is a more complex subsystem of Linux system, which has been constantly changing. It is located between the device driver related to the bottom display and the application of the upper user interface. It shields all kinds of bottom hardware differences, and provides a unified API for the upper user interface.

The earliest graphics system in Linux/unix environment is Xorg graphics system. The following is the structure diagram of a typical embedded graphics system.

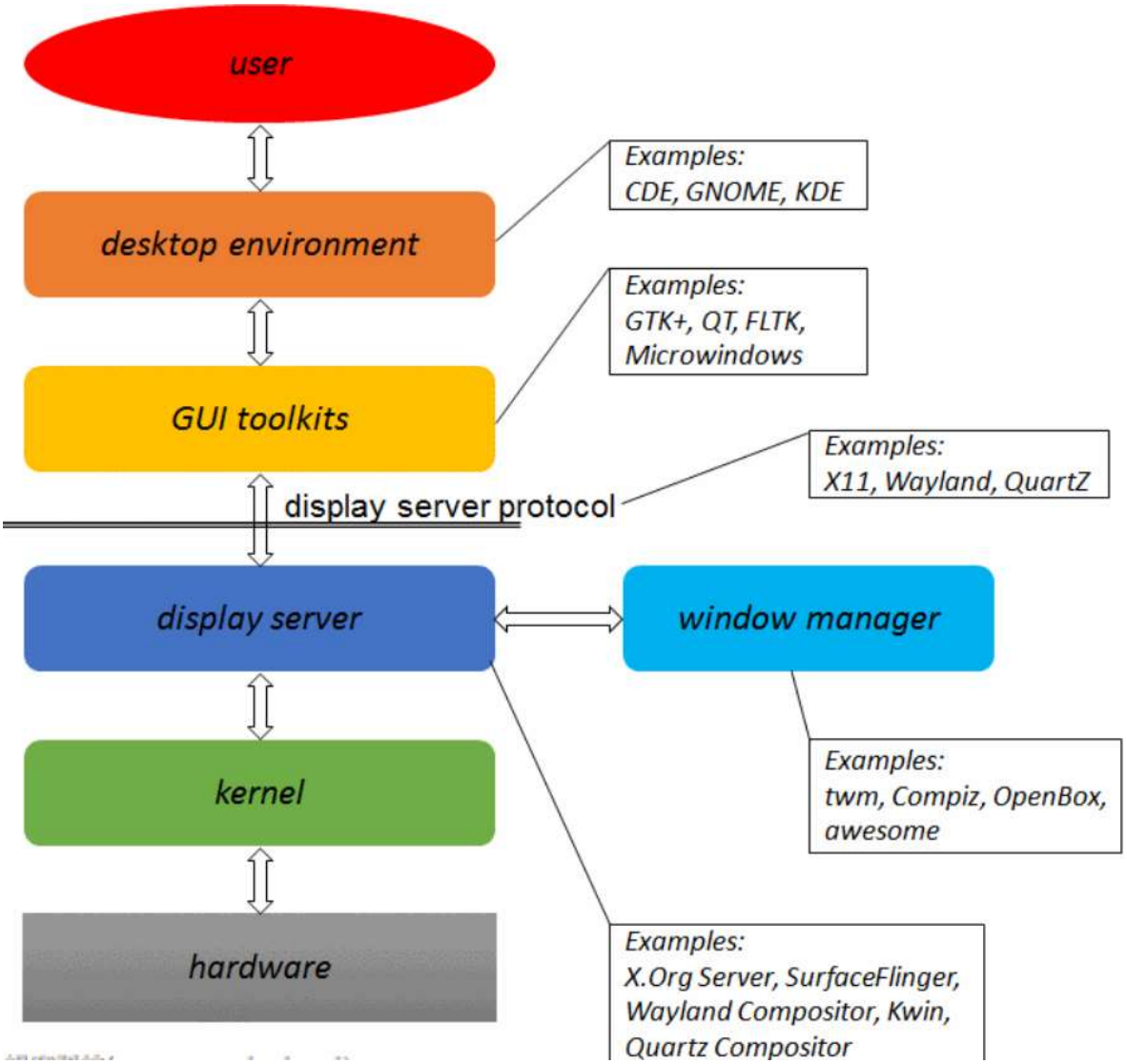

Figure 5-1. Embedded Linux Graphics Stack Overview

5.1. X11

This section mainly introduces X Window (X11). X only provides a basic framework for GUI environment construction, drawing and moving windows on the screen, and interacting with mouse and keyboard.
In mys-zu5ev-core image, a compact but suitable desktop environment of Matchbox desktop for embedded system is provided.

1) Start matchbox-desktop

By default, the desktop of matchbox desktop starts automatically when it is turned on. The following example will first turn off matchbox and then start.

```
root@mys_zu5ev #/etc/init.d/xserver-nodm restart
Stopping XServer
xinit: connection to X server lost

waiting for X server to shut down dbus-daemon[922]: Reloaded configuration
(II) Server terminated successfully (0). Closing log file.

xinit: unexpected signal 15
Starting Xserver
```

5.2. QT

QT is a cross platform C++ GUI application development framework. It can be used to develop GUI programs as well as non GUI programs, such as console tools and servers. QT is an object-oriented framework. With special code generation extensions and some macros, QT is easy to extend and allows real component programming.
The development board will burn the system with QT runtime library when it leaves the factory, and a large number of demos are attached.

1) Get QT information

First, check the QT version supported by the current system, as follows:

```
root@mys_zu5ev  #ls  /usr/lib/*Qt5Core*
/usr/lib/libQt5Core.so.5  /usr/lib/libQt5Core.so.5.13   /usr/lib/libQt5Core.so.5.13.2
```

As can be seen from the above, the version of QT is 5.13.2, and the attached QT routines are located in the /usr/share/examples directory.

2) Introduction of QT operation environment

When running QT application program, according to different software and hardware requirements, the running environment of QT, such as platform plug-in, display parameters, input device and cursor pointer, can be configured appropriately.

- **Platform plug in configuration**

In embedded Linux system, we can use several platform plug-ins: EGLFS, linuxfb, DirectFB or XCB. However, the availability of these plug-ins depends on the characteristics of the actual hardware platform and the configuration of QT. XCB plug-in is used by default on my-zu5ev development board. Also, for quick testing, use -platform command line arguments with the same syntax. For example, in the user console, you can use the following command to configure.

```
root@mys_zu5ev  #cd  /usr/share/examples/
root@mys_zu5ev  #  export  DISPLAY=:0.0
root@mys_zu5ev  #export  QT_QPA_PLATFORM  =  xcb
root@mys_zu5ev  #./widgets/touch/pinchzoom/pinchzoom
```

Or use the -platform xcb instruction platform plug-in as follows:

```
root@mys_zu5ev #./widgets/touch/pinchzoom/pinchzoom -platform xcb
```

Note: starting from QT 5.0, QT no longer has its own window system (QWS) implementation, so the -qws parameter used in earlier versions of QT is no longer supported.

- **Display parameter configuration**

QT applications can obtain screen display related parameters through QScreen class or QDesktopWidget, so as to write screen matching applications. Generally, there is no problem to obtain the screen resolution and color depth through QScreen or QDesktopWidget, but sometimes the physical size obtained is not necessarily correct due to the display driver. At this time, you can configure and adjust the following parameters to make the elements displayed on the actual interface suitable for the size of the display screen

The plug-in writes the frame buffer directly through the fbdev subsystem of Linux. Only software rendered content is supported. Please note, In some settings, display performance may be limited.

However, since fbdev has been abandoned in Linux kernel, DRM cache support has been provided since QT 5.9. To use it, set QT_QPA_FB_DRM environment variable is set to a non-zero value. After setting, as long as the system supports dummy cache, /dev/fb0 will not access the old frame cache device. It is presented through DRM API settings, similar to eglfs_kmsEGLFS. After double buffering and page flipping, the output also provides proper vertical synchronization for the content rendered by the software.

Note: when using dummy buffers, none of the options described below apply because properties such as physical and logical screen sizes are automatically queried. The linuxfb plug-in allows you to use QT_QPA_PLATFORM environment variable or the -platform command line option specify additional settings. For example, QT_QPA_PLATFORM= linuxfb:fb=/dev/fb1 , specifying that /dev/fb1 must use a framebuffer device instead of default fb0. To specify multiple settings, separate with colon (:)

Table 5-1. QT Linuxfb related environment variables

Environment variable	Description
fb=/dev/fbN	Specifies the frame buffer device. On multiple display settings, this setting allows you to run the application on different displays. Currently, multiple framebuffers cannot be used from a single QT application.
size=<width>x<height>	Specifies the screen size in pixels. The plug-in tries to query the physical and logical display size from the frame buffer device. However, this query may not always lead to the correct results. You may need to specify these values explicitly.
mmsize=<width>x<height>	Specifies the physical width and height in millimeters.
offset=<width>x<height>	Specifies the upper left corner offset of the screen in pixels. The default location is (0,0)
nographicsmodeswitch	Specifies that the virtual terminal is not switched to graphics mode (KD_ GRAPHICS) 。 In general, enabling graphics mode will disable flashing cursors and screen white space. However, after setting this parameter, these two functions will also be skipped.
tty=/dev/ttyN	Override the virtual console. Not set only in no graphics mode switch, It can be used when the system is running.

Starting from QT 5.9, EGLFS and linuxfb are synchronized in terms of window sizing strategy: using two platform plug-ins, the first top-level window is forced to cover the whole screen. If it is not necessary to do so, please insert QT_QPA_FB_FORCE_FULLSCREEN environment variable is set to 0 to recover behavior from earlier QT versions.

- **Input peripheral configuration**

If you want to enable tslib support, QT_QPA_EGLFS_TSLIB (for eglfs) or QT_ QPA_ FB_ TSLIB (for linuxfb) needs to be enabled, environment variable is set to 1. Reference for the specific use of tslib:
https://github.com/libts/tslib/blob/master/README.md
Note: tslib input processing program is often used for resistive touch, which generates mouse events and only supports single touch. Screen calibration is also required for initial use.

3) Start QT program

Generally, the touch device does not need to display the mouse pointer. The user can set the environment variable to hide the mouse pointer before executing the application.

If you are using the eglfs platform plug-in, the settings are as follows:

```
export QT_QPA_EGLFS_HIDECURSOR=1
```

If you are using the linuxfb platform plug-in, the settings are as follows:

```
export QT_QPA_FB_HIDECURSOR=1
```

The mys-zu5ev-full image of MYS-ZU5EV development board is provided with demo program. The xcb platform plug-in and evdevtouch touch processing program are used. The corresponding startup command program is as follows:

```
root@mys_zu5ev # export QT_QPA_FB_HIDECURSOR=1
root@mys_zu5ev #./widgets/touch/pinchzoom/pinchzoom -platform xcb&
```

You can directly exit the process by kill

```
root@mys_zu5ev # killall pinchzoom
```

6. Multimedia application

This chapter mainly tests the multimedia application of MYS-ZU5EV. Users need to select the corresponding chapter according to the main chip model of the development board for evaluation and testing. The development boards for zynqMP series processors are tested in chapter order.

6.1. Camera

This section uses GStreamer, v4l2-utils, media-ctl commands to evaluate and test the MYIR USB camera (MY-CAM002U). The main test USB camera preview, capture frame (take a picture).

1) View USB camera topology and resolution

● View the basic information of the device

Insert the USB camera and generate the character device /dev/media1:

```
root@mys_zu5ev # ls -l /dev/media1
crw-rw---- 1 root video 252, 1 Nov 27 08:43 /dev/media1
```

● View device topology

Use "media-ctl -d /dev/media1 -p" to list the topology as follows:

```
root@mys_zu5ev # media-ctl -d /dev/media1 -p
Media controller API version 5.4.31

Media device information
------------------------
driver uvcvideo
model USB 2.0 HD Camera : USB 2.0 HD
serial
bus info usb-5800d000.usbh-ehci-1.1
hw revision 0x0
driver version 5.4.31
```

```
Device topology
- entity 1: USB 2.0 HD Camera : USB 2.0 HD  (1 pad, 1 link)
            type Node subtype V4L flags 1
            device node name /dev/video1
        pad0: Sink
<- "Processing 2":1 [ENABLED,IMMUTABLE]

- entity 4: USB 2.0 HD Camera : USB 2.0 HD  (0 pad, 0 link)
            type Node subtype V4L flags 0
            device node name /dev/video2

- entity 8: Processing 2 (2 pads, 3 links)
            type V4L2 subdev subtype Unknown flags 0
        pad0: Sink
<- "Camera 1":0 [ENABLED,IMMUTABLE]
        pad1: Source
                -> "USB 2.0 HD Camera : USB 2.0 HD ":0 [ENABLED,IMMUTA
BLE]
                -> "Extension 6":0 [ENABLED,IMMUTABLE]

- entity 11: Extension 6 (2 pads, 1 link)
             type V4L2 subdev subtype Unknown flags 0
        pad0: Sink
<- "Processing 2":1 [ENABLED,IMMUTABLE]
        pad1: Source

- entity 14: Camera 1 (1 pad, 1 link)
             type V4L2 subdev subtype Unknown flags 0
        pad0: Source
                -> "Processing 2":0 [ENABLED,IMMUTABLE]
```

- **Use v4l2-ctl to see the resolution supported by /dev/video2**

```
root@mys_zu5ev # v4l2-ctl -D -d /dev/video2  --list-formats-ext
```

```
Driver Info:
        Driver name       : uvcvideo
        Card type         : USB 2.0 HD Camera : USB 2.0 HD
        Bus info          : usb-5800d000.usbh-ehci-1.1
        Driver version    : 5.4.31
        Capabilities      : 0x84a00001
                Video Capture
                Metadata Capture
                Streaming
                Extended Pix Format
                Device Capabilities
        Device Caps       : 0x04200001
                Video Capture
                Streaming
                Extended Pix Format
Media Driver Info:
        Driver name       : uvcvideo
        Model             : USB 2.0 HD Camera : USB 2.0 HD
        Serial            :
        Bus info          : usb-5800d000.usbh-ehci-1.1
        Media version     : 5.4.31
        Hardware revision: 0x00000000 (0)
        Driver version    : 5.4.31
Interface Info:
        ID                : 0x03000002
        Type              : V4L Video
Entity Info:
        ID                : 0x00000001 (1)
        Name              : USB 2.0 HD Camera : USB 2.0 HD
        Function          : V4L2 I/O
        Flags             : default
        Pad 0x01000007    : 0: Sink
```

```
        Link 0x02000010: from remote pad 0x100000a of entity 'Processing
2': Data, Enabled, Immutable
ioctl: VIDIOC_ENUM_FMT
        Type: Video Capture

        [0]: 'MJPG' (Motion-JPEG, compressed)
                Size: Discrete 1280x720
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 640x480
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 160x120
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 352x288
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 176x144
                        Interval: Discrete 0.040s (25.000 fps)
        [1]: 'YUYV' (YUYV 4:2:2)
                Size: Discrete 1280x720
                        Interval: Discrete 0.067s (15.000 fps)
                Size: Discrete 640x480
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 160x120
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 352x288
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 176x144
                        Interval: Discrete 0.040s (25.000 fps)
```

From the above log, we can clearly see the frame rate supported by the corresponding resolution. For example, the frame rate supported by 320x240 is 25fps.

2) Camera window Preview

● Use the gstream tool to preview

Use the gst-launch command in gstream to preview the camera. The terminal executes the following commands:

```
root@mys_zu5ev #gst-launch-1.0 -v v4l2src device=/dev/video2 ! video/x-raw,
width=640,height=320,framerate=25/1 ! videoconvert ! autovideosink

Setting pipeline to PAUSED ...
Pipeline is live and does not need PREROLL ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock

^Chandling interrupt.
Interrupt: Stopping pipeline ...
EOS on shutdown enabled -- Forcing EOS on the pipeline
Waiting for EOS...
Got EOS from element "pipeline0".
EOS received - stopping pipeline...
Execution ended after 0:03:12.080302175
Setting pipeline to PAUSED ...
Setting pipeline to READY ...
Setting pipeline to NULL ...
Freeing pipeline ...
```

After execution, the screen will generate a preview window with width of 640, height of 320 and frame rate of 25 frames per second.

7. System tools

The default image contains some commonly used system tools to facilitate users to view and manage various resources of the system in system debugging or actually deployed products, and can also be invoked in SHELL scripts or other applications. These tools may not fully meet the user's system customization needs, at this time, the system developers need to make appropriate adjustments according to the actual situation.

7.1. Compression and decompression tool

This section mainly tests the decompression tool of the system. Compression can compress multiple files into a compressed package, which is convenient for file transmission. Decompression can restore the compressed file to its original size, which is convenient to use. In this section, we will take tar, gzip, gunzip and other tools in the file system as examples.

1) tar

Now we use tar tool in Linux. It can not only package files, but also compress, view, add and decompress them. Here is the package operation.

- **Grammatical format**

Enter the following command to view the tar syntax format:

```
root@mys_zu5ev # tar --help
BusyBox v1.31.0 (2020-05-26 14:38:25 UTC) multi-call binary.

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

Create, extract, or list files from a tar file

        c       Create
        x       Extract
```

```
t     List
-f FILE    Name of TARFILE ('-' for stdin/out)
-C DIR    Change to DIR before operation
-v    Verbose
-O    Extract to stdout
-m    Don't restore mtime
-o    Don't restore user:group
-k    Don't replace existing files
-Z    (De)compress using compress
-z    (De)compress using gzip
-J    (De)compress using xz
-j    (De)compress using bzip2
-a    (De)compress using lzma
-h    Follow symlinks
-T FILE    File with names to include
-X FILE    File with glob patterns to exclude.
```

The detailed parameters are as follows:

- -c : The parameter instruction of creating a compressed file (the meaning of create);
- -x : Untie the parameter instruction of a compressed file!
- -t : Check the files in tarfile! In particular, only one c/x/t can exist in the release of parameters! Can't exist at the same time! Because it is impossible to compress and decompress at the same time.
- -z : Do you have gzip attribute at the same time? That is, do you need gzip compression?
- -j : Do you have bzip2 attribute at the same time? That is, do you need bzip2 compression?
- -v : Compression process shows the file! This is often used, but not recommended in the background running process!
- -f : When using file name, please note that you need to receive file name immediately after f, do not add parameters! For example, using "tar -zcvfP tfile sfile" is the wrong way to write it. You should write it as "tar -zcvPf tfile sfile".

- -p : Use the original attributes of the original file (the attributes will not change according to the user)
- -P : Can use absolute path to compress!
- -N : It is newer than the following date (yyyy/mm/dd) before it will be packaged into the new file!
- --exclude FILE : in the process of compression, do not package the FILE!

● Using tar compression

Create a new test.txt file and pack the file into .gz format by entering the following command:

```
root@mys_zu5ev # tar -czf test.tar.gz test.txt
root@mys_zu5ev # ls
test.tar.gz      test.txt
```

So add the z parameter above, and use .tar.gz or .tgz to represent the gzip command compressed tar file.

● Use tar to extract

Unpack the tar.gz file

```
root@mys_zu5ev # tar -xvf test.tar.gz
test.txt
root@mys_zu5ev # ls
test.tar.gztest.txt
```

2) Gzip compression tool

Gzip is a command which is often used in Linux system to compress and decompress files. It is convenient and easy to use.

● Syntax format

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

```
root@mys_zu5ev # gzip --help
BusyBox v1.31.1 () multi-call binary.
Usage: gzip [-cfkdt] [FILE]...
```

- **Compress files with gzip**

```
root@mys_zu5ev # gzip test.txt
root@mys_zu5ev # ls
test.txt.gz
```

- **Decompress with gunzip tool**

Unzip the file with gunzip. The example is as follows:

```
root@mys_zu5ev # gunzip test.txt.gz
root@mys_zu5ev # ls
test.txt
```

7.2. File system tools

This section will introduce several common file system management tools. The file system tools of the system are mount, mkfs, fsck and dumpe2fs.

1) mount tool

Mount is a command under Linux. It can attach a partition to a folder in Linux, so as to connect the partition with the directory. Therefore, as long as we access the folder, it is equivalent to accessing the partition. Its application syntax format is as follows:

```
root@mys_zu5ev # 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 the fourth partition for mounting SD card is as follows:

```
root@mys_zu5ev # mount /dev/mmcblk1p4 /mnt/
[10677.124115] EXT4-fs (mmcblk1p4): mounted filesystem with ordered data mode. Opts: (null)
[10677.130984] ext4 filesystem being mounted at /mnt supports timestamps until 2038 (0x7fffffff)
```

2) mkfs format tool

After the hard disk partition, the next step is to establish the Linux file system. Similar to formatted hard disk under windows. The establishment of a file system on a hard disk partition will flush out the data on the partition and cannot be recovered. Therefore, before establishing the file system, make sure that the data

on the partition is no longer used. The command to establish the file system is mkfs, and the application syntax format is as follows:

```
root@mys_zu5ev # 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 SD card is as follows:

```
root@mys_zu5ev # mkfs -t ext3 -V -c /dev/mmcblk1p7
mkfs from util-linux 2.32.1
mkfs.ext3 -c /dev/mmcblk1p7
mke2fs 1.44.3 (10-July-2018)
/dev/mmcblk1p7 contains a ext4 file system labelled 'userfs'
        created on Thu Aug 13 04:32:02 2020
Proceed anyway? (y,N) y
Creating filesystem with 330532 1k blocks and 82656 inodes
Filesystem UUID: 46bd5bb8-1882-4a68-901a-e11b4e71924b
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729, 204801, 221185
Checking for bad blocks (read-only test): done
Allocating group tables: done
```

```
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

3) fsck file repair tool

Fsck command is mainly used to check the correctness of the file system. When there is an error in the file system, fsck command can be used to try to repair it. And repair the Linux disk. For example:

```
root@mys_zu5ev # fsck -a /dev/mmcblk1p7
fsck from util-linux 2.32.1
/dev/mmcblk1p7: clean, 11/82656 files, 20726/330532 blocks
```

4) dumpe2fs

Print the information of super block and blocks group of the existing file system on a specific device. Enter the following command to view the application syntax on the development board:

```
root@mys_zu5ev # dumpe2fs -h
dumpe2fs 1.45.4 (23-Sep-2019)
Usage: dumpe2fs [-bfghimxV] [-o superblock=<num>] [-o blocksize=<num>]
device
```

View the detailed properties of the formatted file system. For example, enter a command to view the detailed information of a disk:

```
root@mys_zu5ev # dumpe2fs /dev/mmcblk0p2
dumpe2fs 1.44.3 (10-July-2018)
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          46bd5bb8-1882-4a68-901a-e11b4e71924b
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
```

If you check the number of inodes on a disk, inodes will also consume the hard disk space, so when the hard disk is formatted, the operating system will

automatically divide the hard disk into two areas. One is the data area, which stores the file data; The other is the inode table, which stores the information contained in the inode.

```
root@mys_zu5ev # dumpe2fs /dev/mmcblk0p2 | grep -i "inode size"
dumpe2fs 1.44.3 (10-July-2018)
Inode size:                  128
```

Looking at the number of blocks in a disk, when the operating system reads the hard disk, it will not read each sector one by one, which is too inefficient. Instead, it reads multiple sectors continuously at one time, that is, it reads a "block" at one time. This "block" composed of multiple sectors is the smallest unit of file access.

```
root@mys_zu5ev # dumpe2fs /dev/mmcblk0p2 | grep -i "block size"
dumpe2fs 1.44.3 (10-July-2018)
Block size:                  1024
```

7.3. Disk management tools

This section will introduce several common disk management tools. fdisk, dd, mkfs, du, df and other disk management tools are included in the system. Through these commands, you can monitor the usual disk usage.

1) fdisk tool

Fdisk disk partition tool has corresponding applications in DOS, windows and Linux. In Linux system, fdisk is a menu based command. To partition a hard disk with fdisk, you can directly add the hard disk to be partitioned as a parameter after the fdisk command

```
root@mys_zu5ev # fdisk -h
Usage:
 fdisk [options] <disk>         change partition table
 fdisk [options] -l [<disk>] list partition table(s)
```

Partition EMMC:

```
root@mys_zu5ev # fdisk /dev/mmcblk1p1
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 copy command

"The dd command is used to copy the specified input file to the specified output file. And in the process of copying can be format conversion. The difference between the dd command and the cp command is that the dd command can be executed on a floppy disk without creating a file system, and the data copied to the floppy disk is actually a mirror file. It is similar to the function of diskcopy command in DOS. The format of dd command is: dd [<if=input file name/ device

name>] [<of=output file name/device name>] [BS=block byte size] [count=block number]

An example of creating a 2M file is as follows:

```
root@mys_zu5ev # time dd  if=/dev/zero of=ffmpeg1 bs=2M count=1 conv=f
sync
1+0 records in
1+0 records out
2097152 bytes (2.1 MB, 2.0 MiB) copied, 0.195515 s, 10.7 MB/s
real    0m 0.20s
user    0m 0.00s
sys     0m 0.05s
```

3) du disk usage statistics tool

The du command is used to display the usage of disk space. This command displays the file system data block occupied by each level subdirectory of the specified directory level by level. The general syntax of du is as follows:

```
root@mys_zu5ev # du --help
Usage: du [OPTION]... [FILE]...
  or:  du [OPTION]... --files0-from=F
Summarize disk usage of the set of FILEs, recursively for directories.
```

Description of some parameters:

- -a: Displays the size of all directories or files
- -h: Taking K, M and G as units to improve the readability of information
- -k: Output in KB
- -m: Output in MB

Count the file size generated by dd command:

```
root@mys_zu5ev # du ffmpeg1
2048    ffmpeg1
root@mys_zu5ev # du -h ffmpeg1
2.0M    ffmpeg1
```

4) df disk statistics tool

It is used to display the disk usage statistics of the file system on the Linux system. The general usage is as follows:

```
root@mys_zu5ev # df --help
Usage: df [OPTION]... [FILE]...
Show information about the file system on which each FILE resides,
or all file systems by default.
```

Description of some parameters:

- -h : You can display in the appropriate units based on the size used
- -i: View the number of inodes and the usage of inodes under the partition
- -T: Print out file system type

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

```
root@mys_zu5ev # df -i
Filesystem      Inodes IUsed   IFree IUse% Mounted on
devtmpfs         37543   477   37066    2% /dev
/dev/mmcblk1p6 289600 25068 264532    9% /
tmpfs            54499    17   54482    1% /dev/shm
tmpfs            54499   704   53795    2% /run
tmpfs            54499    14   54485    1% /sys/fs/cgroup
tmpfs            54499    22   54477    1% /tmp
/dev/mmcblk1p4   16384    24   16360    1% /boot
/dev/mmcblk1p5    4096    34    4062    1% /vendor
tmpfs            54499    26   54473    1% /var/volatile
/dev/mmcblk1p7   65536   438   65098    1% /usr/local
tmpfs            54499    10   54489    1% /run/user/0
```

Inode is divided by the system when we format. Inode is related to the size of the disk partition. When our inode usage has reached 100%, we can't write data to disk even if we still have disk space left. For other application examples, please refer to Section 2.5.

7.4. 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.

5) ps tool

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

```
root@mys_zu5ev # ps --help
BusyBox v1.29.3 (2021-03-31 10:39:30 CST) multi-call binary.

Usage: ps [-o COL1,COL2=HEADER]

Show list of processes
```

```
-o COL1,COL2=HEADER        Select columns for display
```

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 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@mys_zu5ev # ps aux
PID   USER     COMMAND
    1 root     init
    2 root     [kthreadd]
    3 root     [rcu_gp]
    4 root     [rcu_par_gp]
    5 root     [kworker/0:0-eve]
    6 root     [kworker/0:0H]
    7 root     [kworker/u8:0-ev]
    8 root     [mm_percpu_wq]
    9 root     [ksoftirqd/0]
   10 root     [rcu_sched]
   11 root     [migration/0]
   12 root     [cpuhp/0]
   13 root     [cpuhp/1]
   14 root     [migration/1]
   15 root     [ksoftirqd/1]
   16 root     [kworker/1:0-pm]
   17 root     [kworker/1:0H-kb]
   18 root     [cpuhp/2]
```

```
 19 root      [migration/2]
 20 root      [ksoftirqd/2]
 21 root      [kworker/2:0-eve]
 22 root      [kworker/2:0H-kb]
 23 root      [cpuhp/3]
 24 root      [migration/3]
 25 root      [ksoftirqd/3]
 26 root      [kworker/3:0-eve]
 27 root      [kworker/3:0H-kb]
 28 root      [kdevtmpfs]
 29 root      [netns]
 30 root      [kauditd]
 31 root      [kworker/1:1-eve]
 32 root      [oom_reaper]
 33 root      [writeback]
 34 root      [kcompactd0]
 35 root      [khugepaged]
 37 root      [kworker/3:1-eve]
 38 root      [kworker/0:1-eve]
 39 root      [kworker/u8:1]
 85 root      [kworker/2:1-eve]
 89 root      [kblockd]
 90 root      [blkcg_punt_bio]
 91 root      [edac-poller]
 92 root      [watchdogd]
 93 root      [rpciod]
 94 root      [kworker/u9:0]
 95 root      [xprtiod]
 96 root      [cfg80211]
 97 root      [kswapd0]
 98 root      [ecryptfs-kthrea]
 99 root      [nfsiod]
102 root      [spi0]
```

```
104 root      [sdhci]
105 root      [irq/32-mmc0]
106 root      [sdhci]
107 root      [kworker/1:2-rcu]
108 root      [irq/33-mmc1]
109 root      [kworker/1:3-pm]
116 root      [ion_system_heap]
117 root      [kpktgend_0]
118 root      [kpktgend_1]
119 root      [kpktgend_2]
120 root      [kpktgend_3]
121 root      [ipv6_addrconf]
122 root      [krfcommd]
123 root      [mmc_complete]
124 root      [kworker/0:2-eve]
125 root      [kworker/1:1H-kb]
126 root      [kworker/0:1H-mm]
127 root      [irq/38-fd4a0000]
128 root      [mmc_complete]
129 root      [kworker/0:3+eve]
130 root      [kworker/0:4-eve]
136 root      [kworker/3:1H-kb]
137 root      [jbd2/mmcblk0p1-]
138 root      [ext4-rsv-conver]
139 root      [kworker/0:5-pm]
151 root      [kworker/1:2H]
153 root      [kworker/2:1H-kb]
156 root      /sbin/syslogd  -n
160 root      /sbin/klogd  -n
171 root      [jbd2/mmcblk1p2-]
172 root      [ext4-rsv-conver]
193 dbus       dbus-daemon  --system
206 root      /usr/sbin/connmand  -n
```

```
  209 root       [kworker/3:2H]
  210 root       [kworker/2:2H]
  219 root       /usr/sbin/sshd -D
  221 root       -sh
  226 root       ps aux
```

A brief explanation of some of the taskbars above :

- VSZ : Virtual memory size
- RSS : Resident size
- STAT : There are several process states
  - R : runing
  - S : interuptable sleeping
  - D : uninteruptable sleeping
  - T : stopped
  - Z: zombie process
  - +: Foreground process
  - N: Low priority process
  - l: Multithreaded process

1) 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@mys_zu5ev # top -help
top: invalid option -- 'h'
BusyBox v1.29.3 (2021-03-31 10:39:30 CST) multi-call binary.

Usage: top [-b] [-nCOUNT] [-dSECONDS]
```

Dynamically viewing system processes.

```
root@mys_zu5ev # top
Mem: 65704K used, 3965760K free, 80K shrd, 2304K buff, 9748K cached
CPU:   0% usr   0% sys   0% nic  99% idle   0% io   0% irq   0% sirq
Load average: 0.00 0.01 0.02 1/81 231
```

```
  PID   PPID  USER      STAT   VSZ  %VSZ  %CPU  COMMAND
  219      1  root      S      6116   0%    0%  /usr/sbin/sshd -D
  206      1  root      S      4836   0%    0%  /usr/sbin/connmand -n
  221      1  root      S      3800   0%    0%  -sh
  231    221  root      R      3224   0%    0%  top
    1      0  root      S      3088   0%    0%  init
  156      1  root      S      3088   0%    0%  /sbin/syslogd -n
  160      1  root      S      3088   0%    0%  /sbin/klogd -n
  193      1  dbus      S      2936   0%    0%  dbus-daemon --system
  127      2  root      SW        0   0%    0%  [irq/38-fd4a0000]
   31      2  root      IW        0   0%    0%  [kworker/1:1-eve]
  129      2  root      IW        0   0%    0%  [kworker/0:3-eve]
  117      2  root      SW        0   0%    0%  [kpktgend_0]
  118      2  root      SW        0   0%    0%  [kpktgend_1]
  119      2  root      SW        0   0%    0%  [kpktgend_2]
  120      2  root      SW        0   0%    0%  [kpktgend_3]
  107      2  root      IW        0   0%    0%  [kworker/1:2-rcu]
  124      2  root      IW        0   0%    0%  [kworker/0:2-eve]
  227      2  root      IW        0   0%    0%  [kworker/3:2-eve]
  228      2  root      IW        0   0%    0%  [kworker/3:0-eve]
    2      0  root      SW        0   0%    0%  [kthreadd]
```

2) 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@mys_zu5ev # 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@mys_zu5ev # 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 200640 13556 131852    0    0    39    11  148  227  6  3 90  0  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:~# vmstat -s
```

```
    435992 K total memory
     90008 K used memory
     82380 K active memory
     88924 K inactive memory
    200528 K free memory
     13556 K buffer memory
    131900 K swap cache
         0 K total swap
         0 K used swap
         0 K free swap
     20273 non-nice user cpu ticks
      4412 nice user cpu ticks
     19691 system cpu ticks
    630164 idle cpu ticks
       241 IO-wait cpu ticks
         0 IRQ cpu ticks
        22 softirq cpu ticks
         0 stolen cpu ticks
    127150 pages paged in
     36195 pages paged out
         0 pages swapped in
         0 pages swapped out
    969112 interrupts
   1492562 CPU context switches
1581090644 boot time
     10649 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.

3) 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" andthe pipe command to determine the PID to kill the process.

```
root@mys_zu5ev # ps -ef | grep mxapp2
root        1045       1 15 15:51 ?     00:00:02 /home/mxapp2 -platform eglfs
root        1174    1117  0 15:51 ttySTM0   00:00:00 grep mxapp2
```

Then type the following command to terminate the process :

```
root@mys_zu5ev # kill 1045
```

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

8. Development support

This chapter mainly introduces some basic information about the secondary development of the current SDK. The current SDK provides reference images of two configurations. Please refer to "MYS-ZU5EV_SDK release notes".

8.1. Development Language

● 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. The following example is the script xserver-nodm for booting and booting of the Matchbox-desktop in the mys-zu5ev-full image. The content is as follows:

```
#!/bin/sh
#
### BEGIN INIT INFO
# Provides: xserver
# Required-Start: $local_fs $remote_fs dbus
# Required-Stop: $local_fs $remote_fs
# Default-Start:       5
# Default-Stop:        0 1 2 3 6
### END INIT INFO

killproc() {                # kill the named process(es)
        pid=`/bin/pidof $1`
```

```
        [ "$pid" != "" ] && kill $pid
}

read CMDLINE < /proc/cmdline
for x in $CMDLINE; do
        case $x in
        x11=false)
            echo "X Server disabled"
            exit 0;
                ;;
        esac
done

case "$1" in
  start)
        . /etc/profile

        #default for USER
        . /etc/default/xserver-nodm
        echo "Starting Xserver"
        if [ "$USER" != "root" ]; then
            # setting for rootless X
            chmod o+w /var/log
            chmod g+r /dev/tty[0-3]
            # hidraw device is probably needed
            if [ -e /dev/hidraw0 ]; then
                chmod o+rw /dev/hidraw*
            fi
        fi

        # Using su rather than sudo as latest 1.8.1 cause failure [YOCTO #121
1]
        su -l -c '/etc/xserver-nodm/Xserver &' $USER
```

```
        # Wait for the desktop to say its finished loading
        # before loading the rest of the system
        # dbus-wait org.matchbox_project.desktop Loaded
    ;;

    stop)
            echo "Stopping XServer"
            killproc xinit
            sleep 1
            chvt 1 &
    ;;

    restart)
          $0 stop
            $0 start
    ;;

    *)
            echo "usage: $0 { start | stop | restart }"
    ;;
esac

exit 0
```

● C/C++

C/C++ is the most commonly used programming language for the development of underlying applications on Linux platform, and it is also the most efficient language next to assembly. Using C/C++ for development usually adopts the way of cross development, that is, developing on the development host side, compiling and generating binary execution files running on the target machine, and then deploying to the target machine to run.

In this way, you need to install the SDK based on petalinux first. Please refer to "MYS-ZU5EV linux development guide" for the installation steps. After installation, you need to configure the SDK environment as follows:

```
PC$ $CC --version
aarch64-xilinx-linux-gcc (GCC) 9.2.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PU
RPOSE.
```

In this section, we write a simple example of Hello world to demonstrate the application development. The following is the demo program hello-CC.c written in C on the development host side:

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

C++ demo program hello-CXX.cpp

```
//file: hello-CXX.cpp
#include <iostream>
using namespace std;
int main(int argc,char *argv[])
{
          cout << "hello world!";
             return 0;
}
```

Next, compile the application. Here, C language is compiled with $CC, and C++ is compiled with $CXX. Because the corresponding header file and link are needed when compiling, $CC/$CXX contains the corresponding system library and configuration information. If the header file cannot be found when compiling directly, you can add the parameter -v to view the detailed link process.

```
$CC -v hello-CC.c -o hello-CC
or
$CXX -v hello-CXX.cpp -o hello-CXX
```

```
.............
```

Then the scp command is used to copy the generated execution file to the target machine for execution

```
root@mys_zu5ev # ./hello-CC
hello world!
or
root@mys_zu5ev # ./hello-CXX
hello world!
```

For more complex examples and development methods, please refer to the application migration section of "MYS-ZU5EV linux development guide".

- **Python**

Python is an interpretive, object-oriented, dynamic data type high-level programming language. Python was invented by Guido van Rossum at the end of 1989, and its first public release was in 1991. Like Perl, Python source code also follows GPL (GNU General Public License) protocol. This section mainly tests the use of python from two aspects: python command line and script.

1) View the python3 versions supported by the system

```
root@mys_zu5ev # python3 –version
Python 3.7.6
```

2) Python command line test

Start python, enter the following text information in the python prompt, and then press enter to see the running effect:

```
root@mys_zu5ev # python3
Python 3.7.6 (default, Feb 25 2020, 10:39:28)
[GCC 9.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print ("Hello, Python!")
```

In Python version 3.7.6, the output results of the above examples are as follows:

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

Exit Python command line and execute exit() to exit python

```
>>> exit()
root@mys_zu5ev #
```

3) Writing scripts to test Python

Write a simple Python script program, all Python files will take .Py as the extension.

```
root@mys_zu5ev # vi test.py
root@mys_zu5ev # cat test.py
#!/usr/bin/python3
print ("Hello, Python!")
```

Execute the script file. The python3 interpreter is in the /usr/bin directory. Use the following command to execute the script.

```
root@mys_zu5ev # chmod +x test.py
root@mys_zu5ev # ./test.py
Hello, Python!
```

Call the python3 interpreter through script parameters to start executing the script until the script is finished. When the script is executed, the interpreter is no longer valid. The current system does not support pip command and pip installation of python package. If customers need pip tools, they need to migrate them.

8.2. Database

Database is a warehouse that organizes, stores and manages data according to data structure. There are many types of databases, including Access, Oracle, Mysql, SQL server, SQLite, etc.

- **System SQLite**

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite can read and write ordinary disk files directly. A complete SQL database with multiple tables, indexes, triggers, and views is contained in a single disk file. This is a lightweight database, is to comply with the acid relational database management system, its design goal is embedded, and has been used in many embedded products, it occupies very low resources, in the embedded system, may only need a few hundred K of memory is enough. This database runs faster than Mysql and PostgreSQL. MYS-ZU5EV uses SQLite3. Here is a simple test.

Start SQLite3, create a new database <testDB.db>, and enter the following command in the terminal interface to enter the operation interface.

```
root@mys_zu5ev # sqlite3 testDB.db
SQLite version 3.25.3 2018-11-05 20:37:38
Enter ".help" for usage hints.
```

The above command will create a file testDB.db in the current directory. This file will be used by the SQLite engine as a database. Note that the SQLite3 command provides a “sqlite>” prompt after the database file is successfully created.

After the database is created, you can use SQLite's .databases command to check whether it is in the database list, as follows:

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

Exit the sqlite prompt with the .quit command as follows:

```
sqlite> .quit
root@mys_zu5ev #
```

If you want to know more about SQLlite, please refer to the official website https://www.sqlite.org/docs.html.

9. Reference

https://xilinx-wiki.atlassian.net/wiki/home

https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug1144-petalinux-tools-reference-guide.pdf

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

https://www.x.org/wiki/

https://wiki.debian.org/NetworkConfiguration

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