WISE-710 Ubuntu 18.04

User Manual

V2.0

Contents

1. ARM Ubuntu18.04 Overview 4
   1.1. Cross-platform, consistent use experience on X86 and ARM platforms 4
   1.2. Rich Software Support 4
   1.3. Rapid Customization System 4
   1.4. Rapid Development and Maintenance 4
   1.5. Graphical System Settings ..... 錯誤! 尚未定義書籤。
   1.6. Long-term Support Version 4
   1.7. Advantech Provides Customized Industrial ARM Ubuntu Version 4
2. Advantech Ubuntu Software Support 6
   2.1. Built-in software 6
   2.2. Install the package online with the apt command 6
   2.3. Online Installation of Commonly Used Software 6
      2.3.1 The apt command is used as follows: 6
      2.3.2 Install Chromium browser 7
      2.3.3 Install database software mysql 7
      2.3.4 Install Apache web server 7
      2.3.5 Install PHP 7
      2.3.6 Install Python 7
      2.3.7 Install SSH (built-in) 7
      2.3.8 Install VNC 7
      2.3.9 Install Mono 7
3. Ubuntu16.04 System Burning and Boot 9
   3.1. Required hardware and software environment 9
   3.2. Debug port setting 9
   3.3. Preparation for Ubuntu Image Burning 10
   3.4. Start WISE-710 with SD card 10
   3.5. Start the Ubuntu system 11
   3.6. Start to Burn eMMC 11
4. WISE-710 Peripheral Test 13
   4.1. eMMC Flash Read & Write Test 13
   4.2. USB Read & Write Test 13
   4.3. SD Card Read & Write Test 14
   4.4. Serial Port Test 15
   4.5. LAN Port test 16
      4.5.1 Command Line Configuration 16
   4.6. Connecting WIFI Modules 18
      4.6.1 Supported Modules 18
      4.6.2 WIFI Configuration and Connection 18
   4.7. Connecting Bluetooth Module 20
      4.7.1 Supported Modules 20
      4.7.2 Bluetooth Module Configuration and Connection 20
   4.8. Connecting 4G/LTE Module 22

4.8.1 Supported Modules....22
4.8.2 4G/LTE Module Configuration and Connection....22
4.9. Time and Date Setting....23
4.10. DIO Test....24
4.11. CAN Test....25
5. System Configuration....26
5.1. Terminal Command Line....26
5.2. Add User & Password....26
5.3. Automatically Run Specific Programs at Startup....26
5.4. Backup & Deployment of Secondary Development Customized System....27
5.5. Backup Target System Image....27
5.6. Deploy to Other Devices....27

1. ARM Ubuntu18.04 Overview

1.1. Cross-platform, consistent use experience on X86 and ARM platforms

Ubuntu is one of the most popular Linux distributions. At present, many industrial applications are based on the Ubuntu platform. Running Ubuntu on ARM is easier, more convenient, faster, and more friend for industrial application development and maintenance. For existing X86 platform applications, it can be ported to the ARM Ubuntu platform more quickly and seamlessly, while reducing development difficulty, cost and risk.

1.2. Rich Software Support

Ubuntu officially provides a very rich ARM package. For the ssh, telnet, ftp, qt library, mysql database software, Java, , etc., which are commonly used in industrial control, off-the-shelf software packages are provided. When the customer needs these components, there is no need to cross-compile or port. Just like in x86, you can directly install it online through the apt command.

1.3. Rapid Customization System

The Ubuntu ARM version provides a very rich software package, so customers can quickly install the required software packages according to their own project requirements, quickly configure the relevant systems, and customize the required system. Moreover, developers who already have X86 Ubuntu experience don't have to spend time on ARM Linux development because the development techniques and methods are exactly the same.

1.4. Rapid Development and Maintenance

In addition to providing rich software support, Ubuntu is also very mature in development and has many resources. It supports multiple development environments such as gcc, qt, java, python, mono, php, etc., and provides related software development tools. Customers can choose familiar and appropriate development languages, environments and solutions as needed to accelerate software development.

1.5. Long-term Support Version

Ubuntu is developed and maintained by the commercial enterprise Canonical. Its stability and reliability are trustworthy. At the same time, Ubuntu provides a long-term support version - Ubuntu 16.04 LTS will provide 3 to 5 years of support and updates to meet the long-term support needs of industrial customers.

1.6. Advantech Provides Customized Industrial ARM Ubuntu Version

The ARM Ubuntu version provided by Advantech is based on ARM Ubuntu 18.04 and is adapted for WISE-710. To meet the common needs of industrial customers, the following aspects are customized:

1) Provide a variety of hardware test procedures and test instructions to facilitate customer

testing and verification of hardware, as well as learning how to use;

2) Provide sample programs and source code, such as serial communication, video playback, for customer reference during development;
3) Support wireless modules such as peripheral WIFI/4G, and provide built-in drivers to facilitate customers to establish wireless solutions;

Note:

You need to pay to use the Ubuntu system for commercial purposes. You first need to get the official Ubuntu license. Please contact Ubuntu official for further information. You can also contact Advantech PM, since Advantech has established communication and cooperation channels with Ubuntu.

2. Advantech Ubuntu Software Support

2.1. Built-in software

Image supports much software commonly used by industrial users by default.

- common Linux command
- Test demo
- LAN Utility
- ……

2.2. Install the package online with the apt command

If the required software is not included in the Ubuntu Image provided by Advantech, the ARM Ubuntu system also provides the APT (Advanced Package Tool) package management mechanism. In the case where the device is already connected to the network, the software can be queried and installed online through APT related commands. APT automatically handles dependencies and installs the required packages on the system.

Ubuntu offers a very rich ARM package. Most of the software required by customers can be installed directly through the apt command without cross-compilation from the source code, which is really convenient.

The apt command can automatically find the Ubuntu software server through the source configuration file, and download the software from the service. Image has added Ubuntu's official image source by default, so you don't need to re-edit the settings.

However, if some software does not exist in the official Ubuntu source, but a third-party Ubuntu software source can be provided. The user can first modify the configuration of the software source, add third-party software source, and then install the software online.

The image source for the Ubuntu 16.04 system is in /etc/apt/source.list.

Step1: Edit the source.list file to add a new image source.

Step2: Run the apt-get update command to update the image source.

2.3. Online Installation of Commonly Used Software

2.3.1 The apt command is used as follows:

- Install software package:

  # sudo apt-get install packagename

- Remove software package:

  # sudo apt-get remove packagename

- Get a list of new packages:

 # sudo apt-get update

- Upgrade the system with available updates:

  # sudo apt-get upgrade

- Query the required packages:

  #apt-cache search packagename

- List more commands and options:

  # apt-get help

For more information on the use of apt, you can check the relevant information online for a deeper understanding.

The installation of some commonly used software packages for industrial users is listed below:

2.3.2 Install Chromium browser

 # apt-get install chromium-browser

2.3.3 Install database software mysql

 # apt-get install mysql-server

2.3.4 Install Apache web server

 # apt-get install apache2 apache2-dev

2.3.5 Install PHP

 # apt-get install php

2.3.6 Install Python

 # apt-get install python

2.3.7 Install SSH (built-in)

 # apt-get install openssh-server

2.3.8 Install VNC

 # apt-get install x11vnc

2.3.9 Install Mono

If installed in the default way:

sudo apt-get install mono-complete

The default version currently provided by Ubuntu is mono 4.0. If the customer wants to use a higher version of mono 5.2, you can modify the software source configuration file by the method provided before, add Mono official software source, or you can modify the software source by the following command:

#sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF

#echo "deb http://download.mono-project.com/repo/ubuntu xenial main" | sudo tee /etc/apt/sources.list.d/mono-official.list

#sudo apt-get update

#sudo apt-get install mono-complete

Up till now, mono 5.2 has been installed.

3. Ubuntu16.04 System Burning and Boot

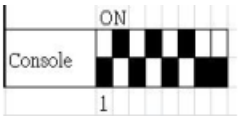
3.1. Required hardware and software environment

- WISE-710
- RS232 serial cable
- Install Ubuntu16.04 on X86 development host (So it is with WISE-710 Ubuntu16.0 device)
- WISE-710 Ubuntu Image
- SD card (8G or more)

3.2. Debug port setting

(1) Connect RS232 serial cable.

Set SW9 DIP switch of WISE-710 and switch COM1 to debug mode.

Console Mode	Bit 2,4,6 ON Bit 1,3,5,7,8 OFF	

Connect the RS-232 serial cable to COM port and the other end of the serial cable to your host.

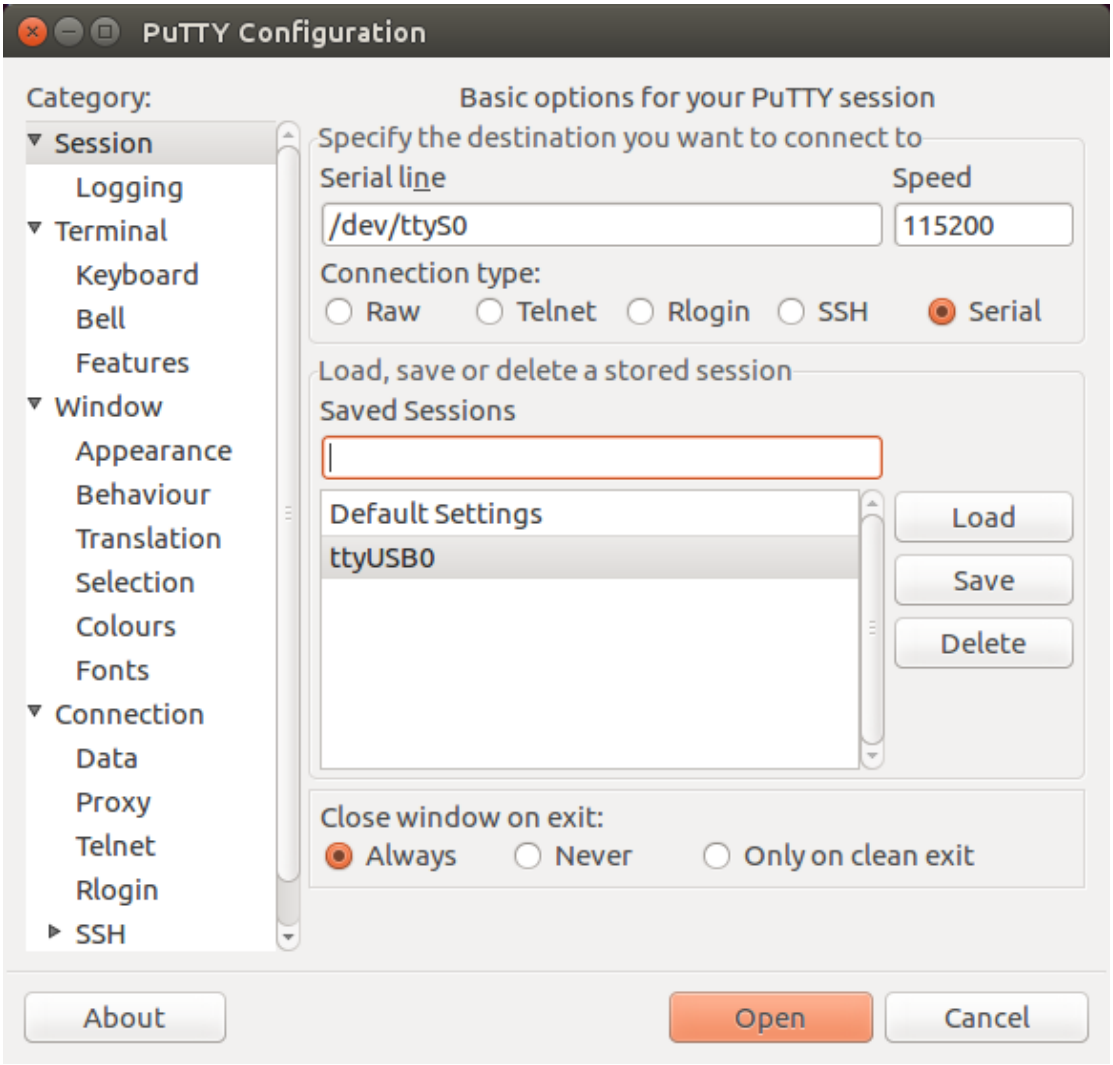
(2) Install the serial debugging tool putty under Linux.

 # apt-get install putty

If it is a Windows environment, you can also download the Windows version of putty from the network to install.

(3) Use the putty tool to set the serial port number, the baud rate is 115200.

Please use the root user to open putty. The serial port number is the number of the serial port that debug first connects. For example, under Linux system, if the first serial port is connected (as shown above), it may be /dev/ttyS0; if you are using a USB to serial cable, it may be /dev/ttyUSB0.



Putty configuration

(4) Connect the power supply and start the WISE-710 from the SD card or onboard storage. From the bootloader stage, debugging information is output to the putty window.

Note:

If you are using the WISE-710 without a system installed, please refer to the following section to create a system SD card that can be booted, and you can burn the system to the onboard emmc via SD.

3.3. Preparation for Ubuntu Image Burning

You can get Ubuntu image from Advantech.

3.4. Start WISE-710 with SD card

After the WISE-710 is powered on, presence of an SD card will automatically be detected. If there is an SD card, it will be preferred to boot from the SD card, otherwise it will boot from eMMC. Insert SD card to start WISE-710 and enter the system.

Under Linux:

1) Unzip the Ubuntu Image archive under Linux.

```
root@ubuntu:/home/work/wise-710/release/2019-06-05-Release3-ubuntu-A101-3# ls
WISE-710-r3-ubuntu.20190605.tar.gz  WISE-710-r3-ubuntu.20190605.tar.gz.md5  WISE-710-r3-ubuntu.20190605.tar.gz.sha256
root@ubuntu:/home/work/wise-710/release/2019-06-05-Release3-ubuntu-A101-3# tar -zxvf WISE-710-r3-ubuntu.20190605.tar.gz
WISE-710-r3-ubuntu.20190605/
WISE-710-r3-ubuntu.20190605/scripts/
WISE-710-r3-ubuntu.20190605/scripts/mkinand-linux.sh
WISE-710-r3-ubuntu.20190605/scripts/Factory-final.sh
WISE-710-r3-ubuntu.20190605/scripts/mac_write_linux
WISE-710-r3-ubuntu.20190605/scripts/touch_fa.sh
WISE-710-r3-ubuntu.20190605/scripts/etp_write
WISE-710-r3-ubuntu.20190605/scripts/mksd_recovery-linux.sh
WISE-710-r3-ubuntu.20190605/scripts/Factory-linux.sh
WISE-710-r3-ubuntu.20190605/scripts/mkspi-advboot.sh
WISE-710-r3-ubuntu.20190605/scripts/touch_ecc.sh
WISE-710-r3-ubuntu.20190605/scripts/hostname_write.sh
WISE-710-r3-ubuntu.20190605/image/
WISE-710-r3-ubuntu.20190605/image/zImage
WISE-710-r3-ubuntu.20190605/image/SPL
WISE-710-r3-ubuntu.20190605/image/adv_logo_1024x600_32bpp.bmp
WISE-710-r3-ubuntu.20190605/image/u-boot.imx
WISE-710-r3-ubuntu.20190605/image/imx6dl-wise710-a1.dtb
WISE-710-r3-ubuntu.20190605/image/ubuntu16044.tar.gz
WISE-710-r3-ubuntu.20190605/image/u-boot_crc.bin
WISE-710-r3-ubuntu.20190605/image/8111g-cfg/
```

2) Get the ubuntu image file, insert the SD into the host and perform the dd burning operation.

```
root@ubuntu:/home/work/wise-710/release/2019-06-05-Release3-ubuntu-A101-3/WISE-710-r3-ubuntu.20190605/scripts# ls ../image/
8111g-cfg  adv_logo_1024x600_32bpp.bmp  imx6dl-wise710-a1.dtb  SPL  u-boot_crc.bin  u-boot_crc.bin.crc  u-boot.imx  ubuntu16044.tar.gz  zImage
root@ubuntu:/home/work/wise-710/release/2019-06-05-Release3-ubuntu-A101-3/WISE-710-r3-ubuntu.20190605/scripts# ls
etp_write         Factory-linux.sh   mac_write_linux   mksd_recovery-linux.sh  touch_ecc.sh
Factory-final.sh  hostname_write.sh  mkinand-linux.sh  mkspi-advboot.sh        touch_fa.sh
root@ubuntu:/home/work/wise-710/release/2019-06-05-Release3-ubuntu-A101-3/WISE-710-r3-ubuntu.20190605/scripts# ./mksd_recovery-linux.sh /dev/sdc ubuntu16044
All data on /dev/sdc now will be destroyed! Continue? [y/n]
y
partition start
DISK SIZE - 7742685184 bytes
partition done
partition done
mkfs.fat 3.0.28 (2015-05-16)
mkfs.fat: warning - lowercase labels might not work properly with DOS or Windows
mke2fs 1.42.13 (17-May-2015)
Creating filesystem with 1874944 4k blocks and 469568 inodes
Filesystem UUID: 519c8a4f-f5fa-4d2e-b52c-908fd91a5b48
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

dd [adv_boot & u-boot]
copy [zImage & dtb]
copy [rootfs]
[Copying iNAND upgrate tools...]
```

3) Wait for the burning to complete, then pull out the SD card and plug it into the device to start.

Note: If SD boot card is made in EMMC system of WISE-710 device, please execute the following command: ./mksd_recovery-linux.sh / dev / mmcblk1 ubuntu16044

3.5. Start the Ubuntu system

After the system is launched:
Super user: root
Password: 123456

3.6. Start to Burn eMMC

- The system is started by SD card (/mk_inand/scripts directory)

```
root@tpc71wn21pa:/mk_inand/scripts# ls
etp_write         flash_erase        mkinand-linux.sh  touch_ecc.sh
Factory-final.sh  hostname_write.sh  mkspi-advboot.sh  touch_fa.sh
Factory-linux.sh  mac_write_linux    mtd_debug
root@tpc71wn21pa:/mk_inand/scripts# ./mkinand-linux.sh /dev/mmcblk0 ubuntu16044
partition start
blk_update_request: I/O error, dev mmcblk0rpmb, sector 0
blk_update_request: I/O error, dev mmcblk0rpmb, sector 0
Warning: Error fsyncing/closing /dev/mmcblk0rpmb: Input/output error
blk_update_request: I/O error, dev mmcblk0rpmb, sector 0
Warning: Error fsyncing/closing /dev/mmcblk0rpmb: Input/output error
partition done
```

Note: If the mkfs.vfat command is not found, you can use **[*# apt-get install dosfstools*]**

- After the burning is complete, sync with sync, then shut down and remove the SD card.
  Note: SD: /dev/mmcblk1　　eMMc：/dev/mmcblk0
- WISE-710 will start from eMMC.

4. WISE-710 Peripheral Test

4.1. eMMC Flash Read & Write Test

Step1: After the device boots from the SD card, run the following command to erase and check eMMC Flash.

(When booting from the SD card, the eMMC Flash node identified in the system is mmcblk1)

```
root@wise710a1:~# dd if=/dev/zero of=/dev/mmcblk0 bs=1024 count=1 seek=1
1+0 records in
1+0 records out
root@wise710a1:~# hexdump -C /dev/mmcblk0 -s 1024 -n 16
00000400 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
```

Step 2: Run the following command to write and check the eMMC Flash.

```
root@wise710a1:~# echo -n "0123456789ABCDEF" | dd of=/dev/mmcblk0 bs=1024 count=1
seek=1
0+1 records in
0+1 records out
root@wise710a1:~# hexdump -C /dev/mmcblk0 -s 1024 -n 16
00000400 30 31 32 33 34 35 36 37 38 39 41 42 43 44 45 46 |0123456789ABCDEF|
```

4.2. USB Read & Write Test

Step 1: Insert the USB storage device and view the WISE-710 device list to get the device node.

Step 2: Run the following command to erase and check the USB storage device.

```
root@wise710a1:~#dd if=/dev/zero of=/dev/sda bs=1024 count=1 seek=1
1+0 records in
1+0 records out
root@wise710a1:~# hexdump -C /dev/sda -s 1024 -n 16
01887800 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
```

Step 3: Run the following command to write and check the USB storage device.

```
root@wise710a1:~# echo -n "0123456789ABCDEF" | dd of=/dev/sda bs=1024 count=1 seek=1
0+1 records in
0+1 records out
root@wise710a1:~# hexdump -C /dev/sda -s 1024 -n 16
00000400 30 31 32 33 34 35 36 37 38 39 41 42 43 44 45 46 |0123456789ABCDEF|
```

Note!

- NXP i.MX6D/Q has the limitation on USB device collection, we rec-ommend the follow brands: Logitech K120 / Lenovo K5819 LXH - EKB-10YA / RAPOO 1800.Pro / Dell MS111-P / Microsoft Wired Keyboard 200 (Model:1406) and so on.
- This operation **may damage the data stored** in USB flash disk. Please make sure there is no critical data in the USB flash disk being used for this test. If your U Disk size is small, the **seek value need to be small**.

4.3. SD Card Read & Write Test

Step 1: When the device boots from the internal eMMC Flash (the SD card is not inserted at startup), the following information can be viewed from the system.

```
root@wise710a1:~# ls /dev/mmcblk* -l
brw-rw---- 1 root disk 179,    8 Feb    5 17:01 /dev/mmcblk0
brw-rw---- 1 root disk 179, 16 Feb    5 17:01 /dev/mmcblk0boot0
brw-rw---- 1 root disk 179, 24 Feb    5 17:01 /dev/mmcblk0boot1
brw-rw---- 1 root disk 179,    9 Feb    5 17:01 /dev/mmcblk0p1
brw-rw---- 1 root disk 179, 10 Feb    5 17:01 /dev/mmcblk0p2
brw-rw---- 1 root disk 179, 32 Feb    5 17:01 /dev/mmcblk0rpmb
```

Step 2: Insert the SD card into the WISE-710 and re-check the device information. /dev/mmcblk1 represents the current SD card device (in this example, the SD card has two partitions).

```
root@wise710a1:~# ls -l /dev/mmcblk*
brw-rw---- 1 root disk 179,    8 Feb    5 17:08 /dev/mmcblk0
brw-rw---- 1 root disk 179, 16 Feb    5 17:08 /dev/mmcblk0boot0
brw-rw---- 1 root disk 179, 24 Feb    5 17:08 /dev/mmcblk0boot1
brw-rw---- 1 root disk 179,    9 Feb    5 17:08 /dev/mmcblk0p1
brw-rw---- 1 root disk 179, 10 Feb    5 17:08 /dev/mmcblk0p2
brw-rw---- 1 root disk 179, 32 Feb    5 17:08 /dev/mmcblk0rpmb
brw-rw---- 1 root disk 179,    0 Feb    5 17:08 /dev/mmcblk1
brw-rw---- 1 root disk 179,    1 Feb    5 17:08 /dev/mmcblk1p1
brw-rw---- 1 root disk 179,    2 Feb    5 17:08 /dev/mmcblk1p2
```

Step 3: Run the following command to erase and check the SD card.

```
root@wise710a1:~# dd if=/dev/zero of=/dev/mmcblk1 bs=1024 count=1 seek=1
1+0 records in
1+0 records out
root@wise710a1:~# hexdump -C /dev/mmcblk1 -s 1024 -n 16
01887800 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
```

Step 4: Write and check SD card.

```
root@wise710a1:~# echo -n "0123456789ABCDEF" | dd of=/dev/mmcblk1 bs=1024 count=1 seek=25118
0+1 records in
0+1 records out
root@wise710a1:~# hexdump -C /dev/mmcblk1 -s 1024 -n 16
01887800 30 31 32 33 34 35 36 37 38 39 41 42 43 44 45 46 |0123456789ABCDEF|
```

4.4. Serial Port Test

As shown in the table below, the WISE-710 has three serial ports. COM1 has three modes: debug mode, RS232 mode and RS485 mode. COM2 and COM3 only support RS485 mode.

HW	SW	DEVICE
COM1	Debug port	/dev/ttymxc0
COM1	232 / 485	/dev/ttyUSB0
COM2	485	/dev/ttyUSB2
COM3	485	/dev/ttyUSB3

Test COM1 rs-232 loopback (baud rate 9600):

```
root@wise710a1:~# cd /usr/Advantech/Serial_test
root@wise710a1:/usr/Advantech/Serial_test# ./st -rsavo -m 232 -b 9600 /dev/ttyUSB0
```

Test COM1 rs-232 read (baud rate 9600):

```
root@wise710a1:~# cd /usr/Advantech/Serial_test
root@wise710a1:/usr/Advantech/Serial_test# ./st -ravo -m 232 -b 9600 /dev/ttyUSB0
```

Test COM1 rs-232 write (baud rate 9600):

```
root@wise710a1:~# cd /usr/Advantech/Serial_test
root@wise710a1:/usr/Advantech/Serial_test# ./st -savo -m 232 -b 9600 /dev/ttyUSB0
```

Test COM2 rs-485 read (baud rate 115200):

```
root@wise710a1:~# cd /usr/Advantech/Serial_test
root@wise710a1:/usr/Advantech/Serial_test# ./st -ravo -m 485 -b 115200 /dev/ttyUSB2
```

Test COM3 rs-485 write (baud rate 115200):

```
root@wise710a1:~# cd /usr/Advantech/Serial_test
root@wise710a1:/usr/Advantech/Serial_test# ./st -savo -m 485 -b 115200 /dev/ttyUSB3
```

Note: COM1's debug, RS232, and RS485 modes can be switched through SW9 and software. For the specific operation of SW9, please refer to the WISE-710 hardware operation manual. Note that

after changing SW9, you need to restart the COM1 mode twice to make it take effect.

4.5. LAN Port test

4.5.1 Command Line Configuration

Check the current IP.

```
root@wise710a1:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr c4:00:ad:2b:72:00
          inet addr:172.21.73.179  Bcast:172.21.73.255  Mask:255.255.255.0
          inet6 addr:  fe80::c600:adff:fe2b:7200/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:819 errors:0 dropped:0 overruns:0 frame:0
          TX packets:41 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:66038 (64.4 KiB)  TX bytes:8198 (8.0 KiB)
```

Open Eth0 network

```
root@wise710a1:~# nmcli connection up eth0
```

Close Eth0 network

```
root@wise710a1:~# nmcli connection down eth0
```

View the connection NAME & UUID of the current network

```
root@wise710a1:~# nmcli c
NAME                UUID                                  TYPE            DEVICE
eth0    ba093436-fba3-46d9-991d-97ec56064bce  802-3-ethernet  --
eth1    b515183e-dc86-4486-81d0-9936fd1c0125  802-3-ethernet  --
```

Delete current Eth0 link

```
root@wise710a1:~# nmcli connection delete eth0
```

Or

```
root@wise710a1:~# nmcli connection delete ba093436-fba3-46d9-991d-97ec56064bce
```

Set Eth0 to dynamic IP mode

```
root@wise710a1:~# nmcli connection add con-name "eth0" type ethernet ifname eth0
```

Set Eth0 to static IP mode

```
root@wise710a1:~# nmcli connection add con-name "eth0" ifname eth0 autoconnect yes
type ethernet ip4 172.21.73.179/24 gw4 172.21.73.253
```

```
root@wise710a1:~# nmcli connection down "eth0"
root@wise710a1:~# nmcli connection mod "eth0" ipv4.dns 172.21.128.10
root@wise710a1:~# nmcli connection up "eth0"
```

View the current network status of Eth0

```
root@wise710a1:~# nmcli device show eth0
GENERAL.DEVICE:                         eth0
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                           C4:00:AD:2B:72:00
GENERAL.MTU:                            1500
GENERAL.STATE:                          100 (connected)
GENERAL.CONNECTION:                       eth0
GENERAL.CON-PATH:
/org/freedesktop/NetworkManager/ActiveConnection/3
WIRED-PROPERTIES.CARRIER:                 on
IP4.ADDRESS[1]:                         172.21.73.179/24
IP4.GATEWAY:                              172.21.73.253
IP4.DNS[1]:                             172.21.128.10
IP6.ADDRESS[1]:                         fe80::c600:adff:fe2b:7200/64
IP6.GATEWAY:
IP6.ROUTE[1]:                           dst = ff00::/8, nh = ::, mt = 256
```

The Ping test shows that the IP of WISE-710 is 172.21.173.179. The IP of the target machine is 172.21.173.29

```
root@wise710a1:~# ifconfig eth0
eth0          Link encap:Ethernet    HWaddr c4:00:ad:2b:72:00
              inet addr:172.21.73.179    Bcast:172.21.73.255    Mask:255.255.255.0
              inet6 addr:    fe80::c600:adff:fe2b:7200/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST    MTU:1500    Metric:1
              RX packets:21354 errors:0 dropped:0 overruns:0 frame:0
              TX packets:240 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:1642835 (1.5 MiB)    TX bytes:40223 (39.2 KiB)
root@wise710a1:~# ping 172.21.73.29 -c 5
PING 172.21.73.29 (172.21.73.29) 56(84) bytes of data.
64 bytes from 172.21.73.29:    icmp_seq=1 ttl=64 time=0.517 ms
64 bytes from 172.21.73.29:    icmp_seq=2 ttl=64 time=0.420 ms
64 bytes from 172.21.73.29:    icmp_seq=3 ttl=64 time=0.430 ms
64 bytes from 172.21.73.29:    icmp_seq=4 ttl=64 time=0.431 ms
64 bytes from 172.21.73.29:    icmp_seq=5 ttl=64 time=0.431 ms
--- 172.21.73.29 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3998ms
```

```
rtt min/avg/max/mdev = 0.420/0.445/0.517/0.044 ms
root@wise710a1:~#
```

Note!

The target computer (Client) firewall need close.

4.6. Connecting WIFI Modules

4.6.1 Supported Modules

WiFi model: 968AD00259 (RTL8188EE) REYAX RYWDB00 (RS9116)

4.6.2 WIFI Configuration and Connection

a) **REYAX RYWDB00 (RS9116) Command Line Configuration**

Setp1: Enable wireless network

```
root@wise710a1:~# cd /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116
```

Open common_insert.sh file

Make sure DRIVER_MODE and COEX_MODE are as follows:

- DRIVER_MODE = 1
- COEX_MODE = 3

```
root@wise710a1:    /lib/modules/
4.1.15/kernel/drivers/net/wireless/RS9116~# ./wlan_enable.sh
```

Setp2: use WiFi Client Mode

```
root@wise710a1:    /lib/modules/
4.1.15/kernel/drivers/net/wireless/RS9116~# ./onebox_util rpine0 create_vap
wlx88da1a763770 sta sw_bmiss

root@wise710a1:    /lib/modules/
4.1.15/kernel/drivers/net/wireless/RS9116~# ./wpa_supplicant -i wlx88da1a763770 -D
nl80211 -c sta_settings.conf    –ddddt    &

root@wise710a1:    /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~# dhclient –r
wlx88da1a763770

root@wise710a1:    /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~# dhclient –
v wlx88da1a763770
```

If you need to modify the user name and password for connecting to the network, please modify it in the sta_settings.conf file.

Setp3: Using WiFi AP Mode

```
root@wise710a1:  /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~# ./onebox_util rpine0 create_vap wlx88da1a763771 ap

root@wise710a1:  /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~# brctl addbr br0

root@wise710a1:  /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~# ifconfig br0 192.168.2.236 netmask 255.255.255.0

root@wise710a1:  /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~# route add default gw 192.168.2.254

root@wise710a1:  /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~# brctl addif br0 eth0

root@wise710a1:  /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~# ifconfig wlx88da1a763771 192.168.1.1

root@wise710a1:  /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~# ./hostapd hostapd_open.conf –ddddt &

root@wise710a1:  /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~# brctl addif  br0  wlx88da1a763771
```

If you need to modify the account name and password of the WiFi AP hotspot, please modify it in the hostapd_open.conf file.

Step3: Disable Wireless network

```
root@wise710a1:  /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~#  。/ wlan_disable.sh
```

b) 968AD00259（RTL8188EE）command line configuration

Setp1: Enable wireless network

```
root@wise710a1:~# nmcli radio wifi on
```

Setp2: Scan WiFi network

```
root@wise710a1:~# nmcli device wifi
```

Setp3: Connect to WPA2/PSK to encrypt the network

```
root@wise710a1:~# nmcli device wifi connect <WIFI_SSID> <WIFI_PASSWD> name wlan0 ifname wlan0
```

Setp4: Connect to a passwordless network

```
root@wise710a1:~# nmcli device wifi connect <WIFI_SSID> “” name wlan0 ifname wlan0
```

Setp5: Close the current connection

```
root@wise710a1:~# nmcli connection down wlan0
```

Setp6: Delete the current connection

```
root@wise710a1:~# nmcli connection delete wlan0
```

Setp7: Disable wireless network

```
root@wise710a1:~# nmcli radio wifi off
```

4.7. Connecting Bluetooth Module

4.7.1 Supported Modules

Bluetooth model: REYAX RYWDB00 (RS9116)

REYAX RYWDB00 (RS9116) is a module that supports both WLAN and Bluetooth functions.

4.7.2 Bluetooth Module Configuration and Connection

a) REYAX RYWDB00 (RS9116) Command Line Configuration

Setp1: Enable Bluetooth function:

```
root@wise710a1:~# cd /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116
```

Open common_insert.sh file

Make sure DRIVER_MODE and COEX_MODE are as follows:

- DRIVER_MODE = 1
- COEX_MODE = 14(For WLAN Access Point, BT Classic and BT LE)

- COEX_MODE = 13(For WLAN Station, BT Classic and BT LE)

If you use only Bluetooth function

```
root@wise710a1:  /lib/modules/
4.1.15/kernel/drivers/net/wireless/RS9116~# ./bt_enable.sh
```

If you use the Bluetooth and wireless functions of REYAX RS9116 at the same time

```
root@wise710a1:  /lib/modules/
4.1.15/kernel/drivers/net/wireless/RS9116~# ./wlan_bt_insert.sh
```

For the configuration of wireless functions, please refer to Section 4.6.2 The configuration of the wireless functions of REYAX RS9116.

Use bluetoothctl to power on the Bluetooth module and make the Bluetooth module discoverable by surrounding Bluetooth devices.

```
root@wise710a1:  /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~#
bluetoothctl
[bluetooth]# poweron
[bluetooth]# discoverableon
[bluetooth]# exit
```

Use hcitool to scan the surrounding Bluetooth devices:

```
root@wise710a1:  /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~# hcitool
scan
```

Use rfcomm to connect to the scanned Bluetooth device, where MAC-Address is the MAC address of the device scanned in the previous step.

```
root@wise710a1:  /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~# rfcomm
connect hci0 MAC-Address
```

Use sdptool to scan Bluetooth devices and transmission channels around the OPUSH Bluetooth transmission protocol.

```
root@wise710a1:  /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~# sdptool
search OPUSH
```

Use obex_test for file transfer, MAC-Address is the MAC address of the Bluetooth device to be connected, and Path is the OPUSH transmission protocol channel of the Bluetooth device found in the previous step.

```
root@wise710a1:  /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~# obex_test
–b MAC-Address Path
> c
>x
      Enter the absolute path of the file to be sent by this machine
>q
```

Use obex_test to receive files. Add OPUSH transmission protocol function to REYAX Bluetooth device.

```
root@wise710a1:  /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~# sdptool add OPUSH
```

View the channels of OPUSH transmission function of REYAX Bluetooth device

```
root@wise710a1:  /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~# sdptool brower local
```

Use obex_test to receive the file, where MAC-Address is the MAC address of the connected Bluetooth device, and Path is the channel where the OPUSH function of REYAX Bluetooth module is used, which can be obtained from the previous operation.

```
root@wise710a1:  /lib/modules/ 4.1.15/kernel/drivers/net/wireless/RS9116~# obex_test –b MAC-Address Path
> s
      Wait for transfer to complete
>q
```

The file will be saved in the root directory when receiving is complete.

Step2 Disable Bluetooth function.

```
root@wise710a1:  /lib/modules/
4.1.15/kernel/drivers/net/wireless/RS9116~# ./bt_disable.sh
```

If you turn off both Bluetooth and wireless function at the same time.

```
root@wise710a1:  /lib/modules/
4.1.15/kernel/drivers/net/wireless/RS9116~# ./remove_all.sh
```

4.8. Connecting 4G/LTE Module

4.8.1 Supported Modules

4G/LTE model: EWM-C117FL01E series, EG-25G series.

4.8.2 4G/LTE Module Configuration and Connection

a) EG-25G Command Line Configuration

Setp1: Enable Mobile Data Network

```
root@wise710a1:~# cd /usr/sbin
root@wise710a1:/usr/sbin~# ./wan,sh unicom ttyUSB7 up
```

Setp2: Disconnect mobile data network

```
root@wise710a1:/usr/sbin~# ./wan,sh unicom ttyUSB7 down
```

Note: The first parameter indicates operator: China Mobile - cmnet
China Unicom - unicom
China Telecom - telecom

The second parameter is the port number of the mobile module
The third parameter is enable / disable module

b) EWM-C117FL01E Command Line Configuration

Setp1: Enable Mobile Data Network

```
root@wise710a1:~# nmcli radio wwan on
```

Setp2: Connect Mobile Data Network

```
root@wise710a1:~# nmcli connection add con-name "ppp" type gsm ifname ttyUSB1 apn 3gnet user uninet password "111111"
```

Or

```
root@wise710a1:~# nmcli connection add con-name "usb" type ethernet ifname usb0
```

Setp3: Disconnect mobile data network

```
root@wise710a1:~# nmcli connection down "ppp"
```

Setp4: Delete mobile data network

```
root@wise710a1:~# nmcli connection delete "ppp"
```

Setp5: Disable mobile data network

```
root@wise710a1:~# nmcli radio wwan off
```

4.9. Time and Date Setting

Set system time (2019/01/01 13:25:00):

```
root@wise710a1:~# date -s “2019/01/01 13:25:00”
```

Synchronize time from the NTP server:

```
root@wise710a1:~# ntpdate <NTPSERVERIP>
```

Reset RTC hardware clock time (use current system time):

```
root@wise710a1:~# hwclock -w
```

Reset system time (use RTC hardware clock time):

```
root@wise710a1:~# hwclock -s
```

Set system time zone (use Shanghai time):

```
root@wise710a1:~# cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
root@wise710a1:~# sync
```

4.10. DIO Test

As you can see below, there are 4 DI/DO supported by WISE-710 internal.

HW	Default value	System node	Software node ID
DO1	low	/sys/class/gpio/gpio1/value	1
DO2	low	/sys/class/gpio/gpio2/value	2
DO3	low	/sys/class/gpio/gpio3/value	3
DO4	low	/sys/class/gpio/gpio4/value	4
DI1	-	/sys/class/gpio/gpio5/value	5
DI2	-	/sys/class/gpio/gpio6/value	6
DI3	-	/sys/class/gpio/gpio7/value	7
DI4	-	/sys/class/gpio/gpio8/value	8

Please use Advantech EAPI api & example to test DIO.

Set DO1 output value to high:

```
root@wise710a1:~# cd /usr/Advantech/EAPI_test
root@wise710a1:    /usr/Advantech/EAPI_test# ./testdl_gpio 5 1 1
GPIOSetLevel Id:    1
Level:    1
```

Set DO2 output value to low:

```
root@wise710a1:~# cd /usr/Advantech/EAPI_test
root@wise710a1:    /usr/Advantech/EAPI_test# ./testdl_gpio 5 2 0
GPIOSetLevel Id:    2
Level:    0
```

Get DI1 output value:

```
root@wise710a1:~# cd /usr/Advantech/EAPI_test
root@wise710a1:/usr/Advantech/EAPI_test# ./testdl_gpio 4 5
GPIOGetLevel Id:    5
level:    0
```

Get DO1 output value:

```
root@wise710a1:~# cd /usr/Advantech/EAPI_test
root@wise710a1:/usr/Advantech/EAPI_test# ./testdl_gpio 4 1
GPIOGetLevel Id:    1
```

```
level:    1
```

4.11. CAN Test

As you can see below, there are 1 flexCAN supported by WISE-710 internal.

HW	DEVICE	MODE
flexCAN0	can0	socket can

Setting: Open flexCAN device (125000 baud rate, loopback off)

```
root@wise710a1:~# ip link set can0 down
root@wise710a1:~# ip link set can0 up type can bitrate 125000 loopback off
root@wise710a1:~# ip link set can0 up
root@wise710a1:~# ifconfig can0
can0          Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
              UP RUNNING NOARP  MTU:16  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:10
              RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
              Interrupt:31
```

Check can0 status:

```
root@wise710a1:~# ip -details link show can0
3:  can0:  <NOARP,UP,LOWER_UP,ECHO> mtu 16 qdisc pfifo_fast state UNKNOWN mode
DEFAULT group default qlen 10
    link/can  promiscuity 0
    can state ERROR-ACTIVE (berr-counter tx 0 rx 0) restart-ms 0
          bitrate 125000 sample-point 0.875
          tq 500 prop-seg 6 phase-seg1 7 phase-seg2 2 sjw 1
          flexcan:  tseg1 4..16 tseg2 2..8 sjw 1..4 brp 1..256 brp-inc 1
          clock 30000000
```

Send message (“123#11") to socket can0:

```
root@wise710a1:~# cansend can0 123#11
```

Recv message from socket can0:

```
root@wise710a1:~# candump can0
```

Note: The Linux system provides Socket CAN interface, which makes CAN bus communication

similar to Ethernet communication, and application development Interface more general and flexible. For more details, please refer to
https://gitorious.org/linux-can/can-utils
https://github.com/linux-can/can-utils/
https://www.kernel.org/doc/Documentation/networking/can.txt

5. System Configuration

5.1. Terminal Command Line

Many system operations and configurations under Linux are performed from the command line. There are two ways to start the command line:
Method 1: Start a terminal by selecting Applications-> Accessories-> Terminal
Method 2: Shortcut: Ctrl + Alt + T
On the command line, you can run various shell commands, scripts and commonly used commands

- View directory: ls
- Create directory: mkdir (Directory name)
- Change directory: cd (directory/location)
- Copy file/directory: cp (source file or directory name) (destination directory or file name)
- Delete file/directory: rm (file name or directory name)
- Rename file/directory: mv (file name or directory name)
- Search file/directory: locate (file name or directory name)
- pwd shows current directory
- ifconfig shows the network of the system

The command line is the most basic operation under Linux. It is an essential skill for every Linux developer and user. Therefore, it will not be described in detail here.

5.2. Add User & Password

Step1: Create new user.
 # useradd -d /home/test -g root -m test //Create a new test user and specify the user group as the root user group, and a login directory will be automatically established
passwd test //Set a password for the test user

Step2: Add workgroup to existing users
usermod -G root test //Set root user group for test user
gpasswd -a test root //Set root user group for the test user

5.3. Automatically Run Specific Programs at Startup

By default, the system's auto-startup file is under /etc/rc.local. The client writes the specific

program to be run in the form of a script. 1. Run the script in the rc.local file.
For example:
(1) The client needs to run the demo program after booting. First, create a sh script file and write the running demo into the script.

```
# vi demo.sh
cd  demo/  &&  ./demo
```

(2) After the execution script is written, write the command to run the script in /etc/rc.local.

```
# vi  /etc/rc.local
/root/demo.sh
```

5.4. Backup & Deployment of Secondary Development Customized System

For specific applications, developers definitely need to conduct secondary development and customization on the system we provide to meet the needs of specific application solutions. This chapter gives the corresponding methods to guide customers how to back up the system and deploy other machines in batches after completing secondary development customization on WISE-710.
When the user installs the system on emmc and integrates his own application, after the debugging is completed, the system needs to be backed up for batch deployment, and the backup script program provided by us can be used for simple backup and deployment.
Note: It is to back up the system on emmc to SD, and then burn to other devices' emmc through SD or remote mode, so the secondary development custom system is on emmc.

5.5. Backup Target System Image

1) Prepare an SD in advance and burn the original ARM Ubuntu system we provided, and boot from the SD card.
   Note: This system uses the system image we provided to burn to SD.
2) After startup, the system will go to the (/ mk_inand / scripts /) directory and execute the backup_emmc_to_sdcard.sh script to back up the system. At this time, the system will be backed up to the / mk_inand / image / update folder

```
root@wise710a1:~# cd  /mk_inand/scripts
root@wise710a1:~/mk_inand/scripts#./backup_emmc_to_sdcard.sh /dev/mmcblk0
```

3) Start synchronization when backup is complete
4) Then power off and remove the SD card.

5.6. Deploy to Other Devices

At this time, the SD card / mk_inand / image / update folder has backed up the latest system which can be burned to other WISE-710 devices.

5.7. Upgrade WISE-710

5.7.1 Upgrade WISE-710 with SD card

1） First select an SD card with FAT32 format and copy the files in the backup update folder to this

SD card (the capacity of the SD card is 8G and above).

2） Insert this SD card into the device that needs to be upgraded.

3） WISE-710 will automatically upgrade the system to the latest operating system on the SD card.

5.7.2 Remote Upgrade WISE-710

1） Download the files use client (which need customer provided) in the update folder to the eMMC Recovery partition of WISE-710.

2） Reboot the system.

3） WISE-710 will automatically upgrade the system to the latest operating system on the Recovery partition.