Chipsee

Industrial PC

Ubuntu 14.04 OS on iMX6Q User Manual

For iMX6Q Products

Content can change at anytime, check documentation website for latest information.
www.chipsee.com

Contents

1. Ubuntu 14.04 OS 4
1.1. Preparation 5
1.1.1. Hardware Requirements 5
1.1.2. Software Requirements 6
1.2. Debug 6
1.2.1. Serial Debug 6
1.2.2. SSH Debug 8
1.2.3. VNC Debug 10
1.3. Downloading images 13
1.3.1. Boot Switch Configuration 13
1.3.2. Prebuilt Files Package 13
1.3.3. Downloading Images by using MFGTool 14
1.3.3.1. Configuring MFGTool 14
1.3.3.1.1. Copy Image To Android Directory 15
1.3.3.1.2. Using MFGTool 16
1.3.4. Downloading Images by using the TF card 19
1.4. System Resource 20
1.4.1. TF Card/USB/SATA Disk 20
1.4.2. Network 20
1.4.2.1. Wired Ethernet 20
1.4.2.2. Wi-Fi 21
1.4.2.3. Remove and Install Network-manager Packages 23
1.4.2.4. Networking — Wired Ethernet 23
1.4.2.5. Networking — WIFI 24
1.4.3. Multimedia 24
1.4.3.1. Audio Test 25
1.4.4. HDMI 26
1.4.5. Serial Port 28
1.4.6. CAN Bus 29
1.4.7. GPIO 30

1.4.8. Buzzer 32
1.4.9. Modify Logo 33
1.4.9.1. Method 1 - Downloading images 33
1.4.9.2. Method 2 - Don’t Download Images 33
1.5. Development 35
1.5.1. Python 35
1.5.2. Qt Environment 36
1.6. Q&A 37
1.6.1. How to rotate the display 38
1.6.2. How to disable the Screensaver 38
1.6.3. Autostart Application after Boot 39
1.7. Disclaimer 41
1.8. Technical Support 41

Ubuntu 14.04 OS

Ubuntu 14.04 OS User Manual

This manual provides users with a fast guide of Chipsee Industrial Computer (Abbreviate as IPC) about Ubuntu 14.04 OS development. Through this manual, users can quickly understand the hardware resources; users can build a complete compilation of Linux development environment; users can debug Ubuntu 14.04 OS via serial and Internet.

Revision	Date	Author	Description
V1.1	2021-12-30	Randy	Revised
V1.0	2018-05-14	Madi	Initial Version

SUPPORTED BOARDS:

> *CS10600F070 CS10768F097 CS12800F101 CS10768F121 CS10768F121-U CS10768F150 CS12102F170 CS19108F215*

PREBUILT FILES PACKAGE:

Prebuilt files for the various industrial PCs can be found in the OS Downloads.
Below are the links to the prebuilt files for each industrial PC model.

- CS10600F070
- CS10768F097
- CS12800F101
- CS10768F121
- CS10768F121-U

- CS10768F150
- CS12102F170
- CS19108F215

System Features

Feature	Comment
Kernel	Kernel 3.14.52
Bootloader	Uboot 2015.04
System	Ubuntu 14.04 LTS
Python	Python 2.7.9 / Python 3.4.0
Qt	Need to be installed by user
GCC	4.8.2
Desktop	matchbox
user/password	[chipsee/chipsee]

Preparation

You will need to prepare the following items before you can start using the Prebuilt Files Package to re-flash the system.

Power Supply Unit (PSU) with the appropriate voltages, as follows:

- These products: CS10768F121, CS10768F121-U, CS10768F150, CS12102F170, and CS19108F215 requires a 15V to 36V power adapter.
- These products: CS10768F097 and CS12800F101 product needs a 12V to 36V power adapter.
- The CS10600F070 product needs a 6V to 36V power adapter.

You need to prepare the Power Adapter by yourself

Hardware Requirements

- Chipsee Industrial PC
- PSU according to the instructions above
- USB-to-serial or other serial cable for debugging
- USB A-A cable (used only if the hardware configured as OTG)
- Windows 7 PC
- Mini-B USB OTG Cable

- TF Card (at least 4GB) and card reader

Software Requirements

- Ubuntu 14.04 OS Prebuilt Files Package (from the link above)
- Xshell or other terminal emulation software
- VNC-Viewer
- Cross-toolchain
- MFGTools
- Useful tools for Qt development

> **Note**
>
> - If you want to re-flash the system, you need the Prebuilt image package.
> - You can use MFGTools on the Windows PC to download system images to the IPC.
> - You can use Xshell or other terminal emulation software to debug Chipsee Industrial PC products in Windows.
> - You can use VNC-Viewer to to remote control Chipsee Industrial PC over Ethernet.
> - The cross-toolchain can compile a program for Chipsee Industrial PC.

> **Note**
>
> In this documentation, all the commands are executed with `root` user privileges.

Debug

In this document, we use Xshell to debug the Chipsee Industrial Computer. You can also use other tools such as Putty, Minicom, SecureCRT or any terminal emulation software.

Serial Debug

You can refer to the RS232/RS485/CAN Connector section under the EPC/PPC-A9-070-C manual to understand the serial ports of the IPC. The debug serial port of Chipsee Industrial Computer is the first RS232 port. You can use it to debug directly, and the default user and password is [chipsee/chipsee]. You can use RS232_1_TXD, RS232_1_RXD, GND.

Follow these steps to perform serial debugging:

- Connect your Windows PC to the Chipsee IPC over a serial cable. Please reference the `How To Connect Board By Serial` manual to connect your PC and Chipsee Industrial Computer over a serial cable.
- **Open XShell and use the session properties as shown on the figure below.**

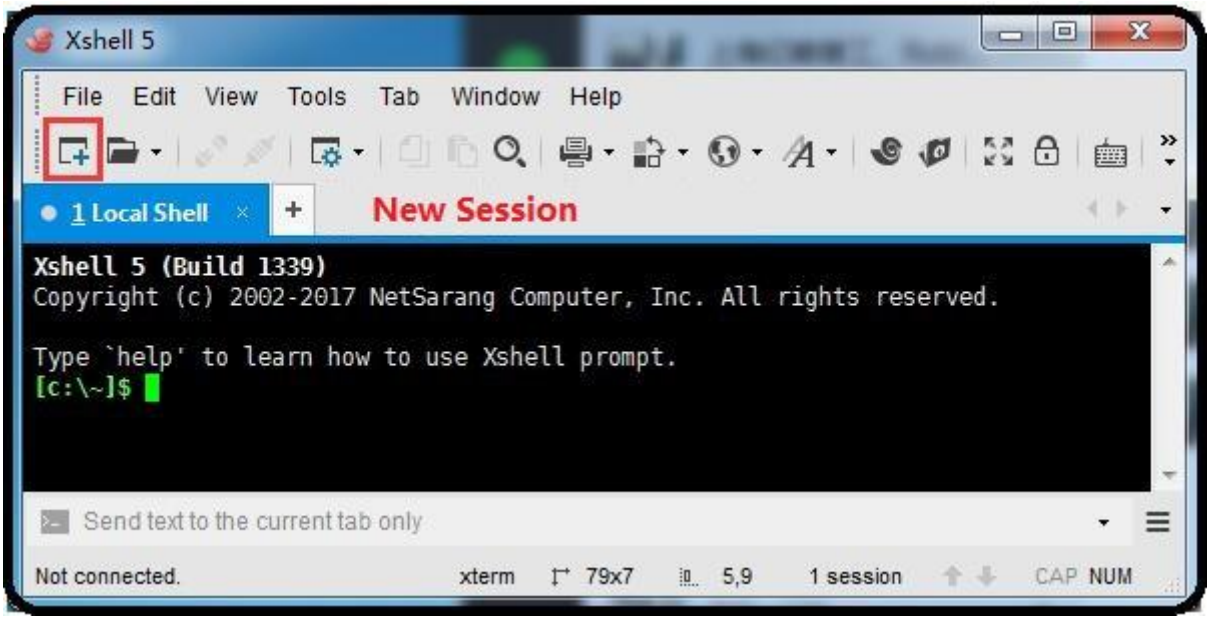

Figure 291: *Add Session*

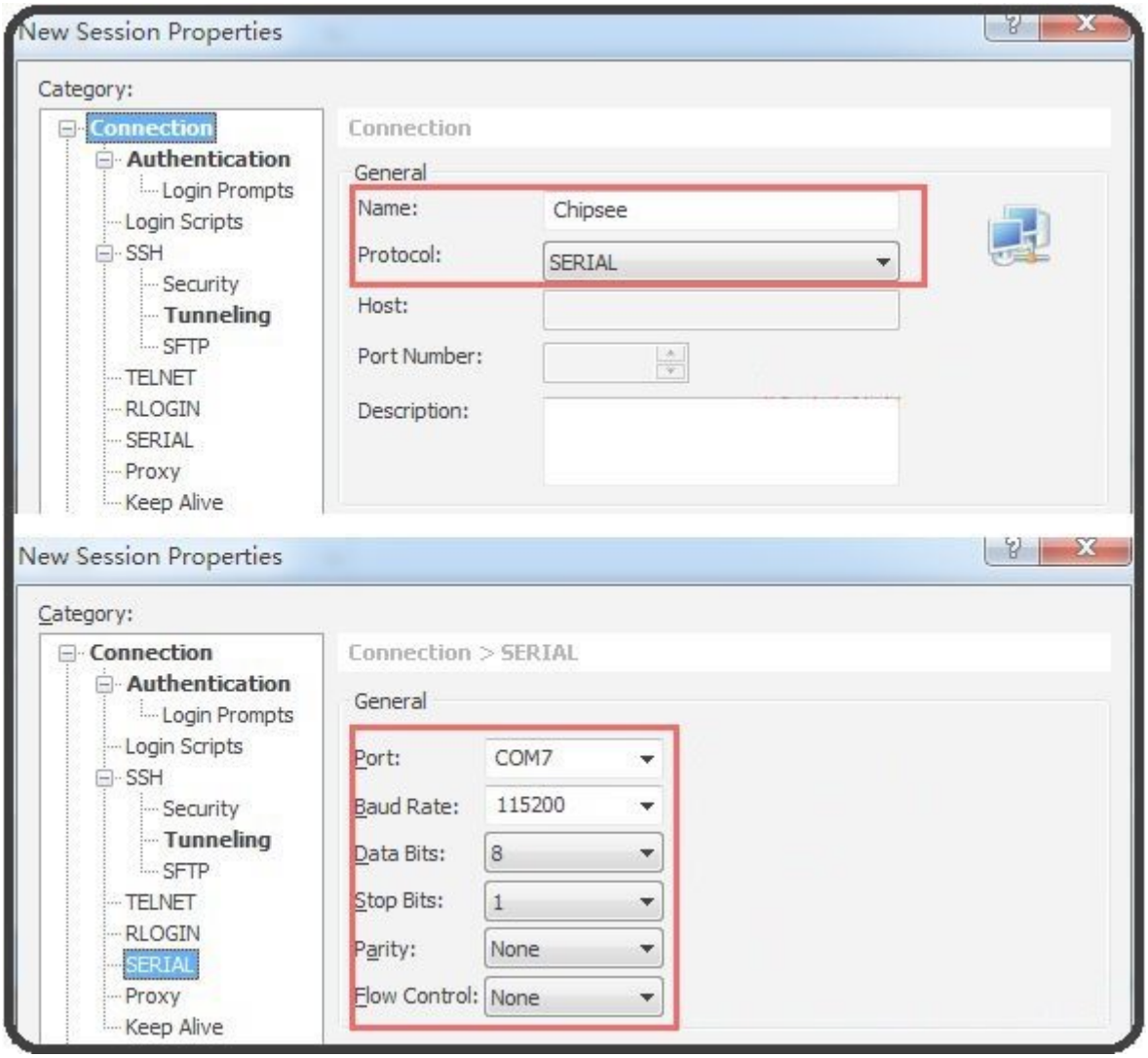

Figure 292: *Session Properties*

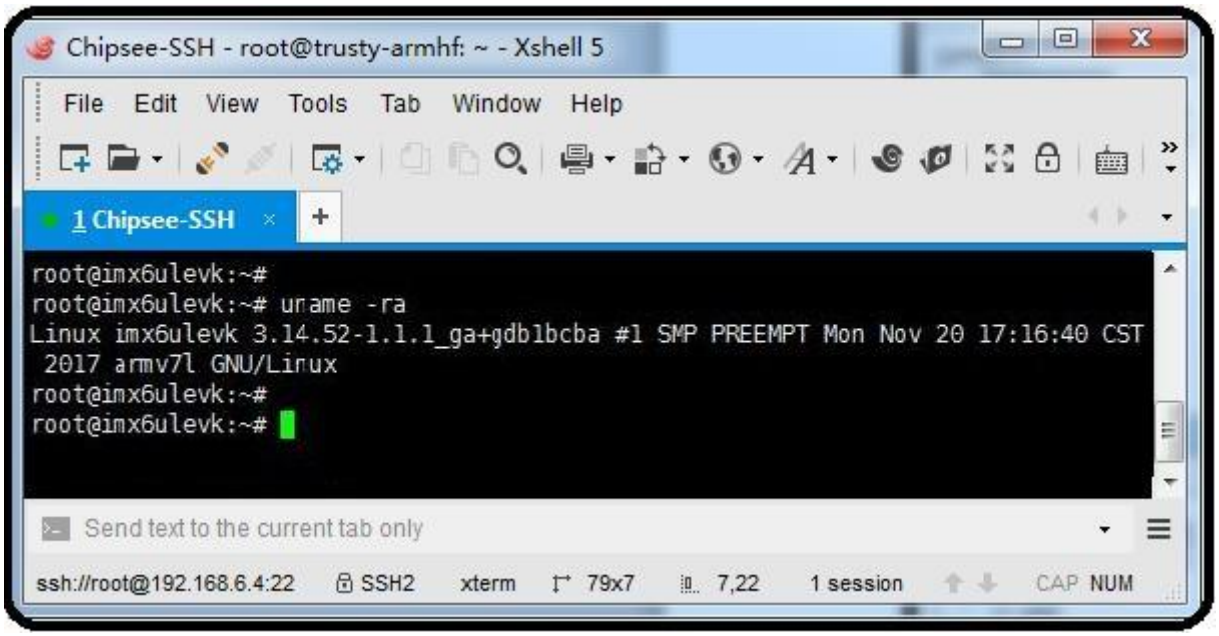

Figure 293: *Serial Debug*

SSH Debug

To perform SSH debugging on the Chipsee IPC, you must first connect the product to the Internet.

Continue the debugging by follow these steps:

- Get the IP address of the Chipsee IPC product.
- You can configure XShell or you can directly use the SSH tool in Linux OS. In this tutorial, we will use the XShell tool to perform SSH debugging.
- **Open XShell and add a new session and set it as shown on the figure below.**

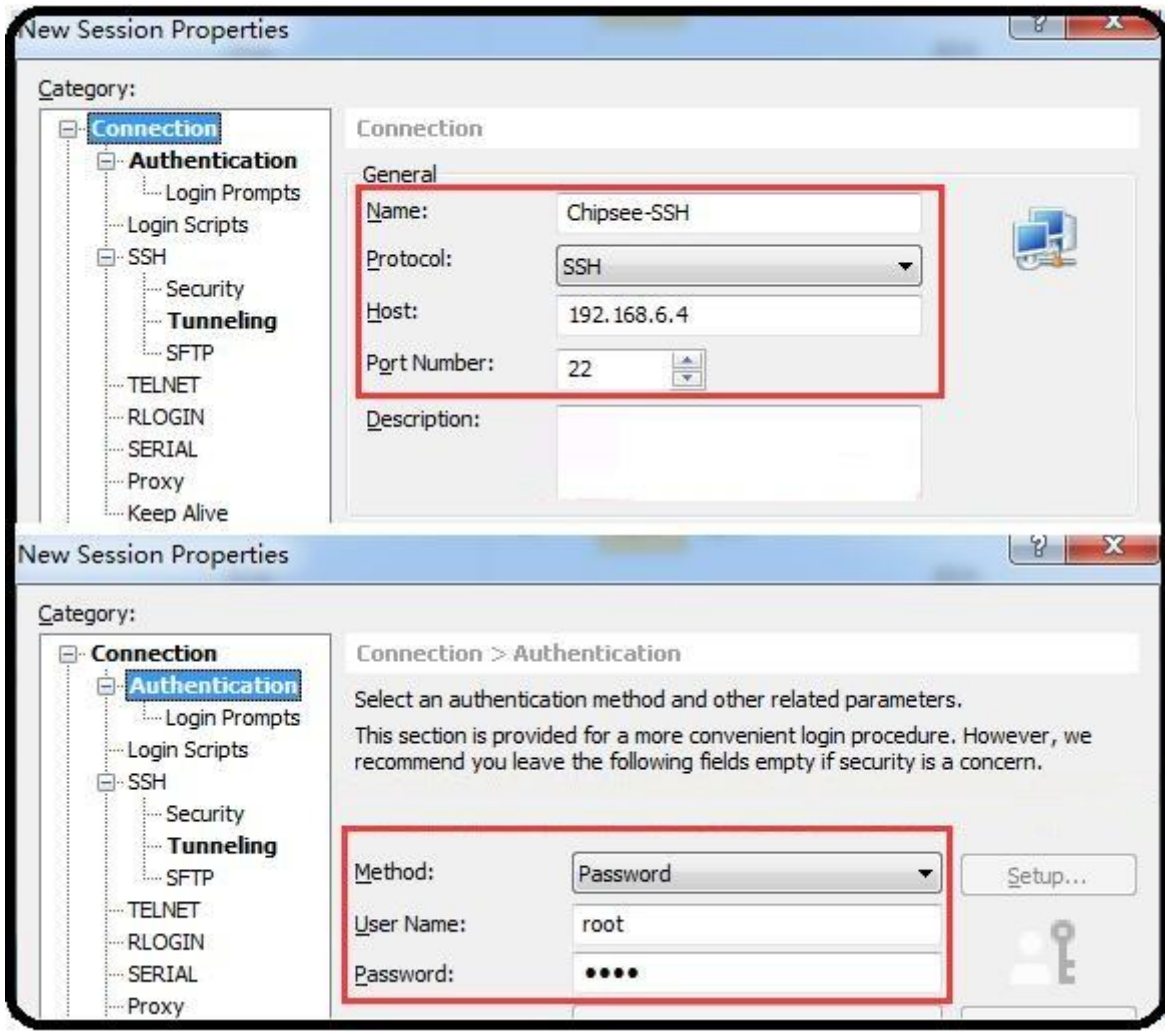

Figure 294: *SSH Setting*

- **Now we can perform SSH debugging using XShell.**

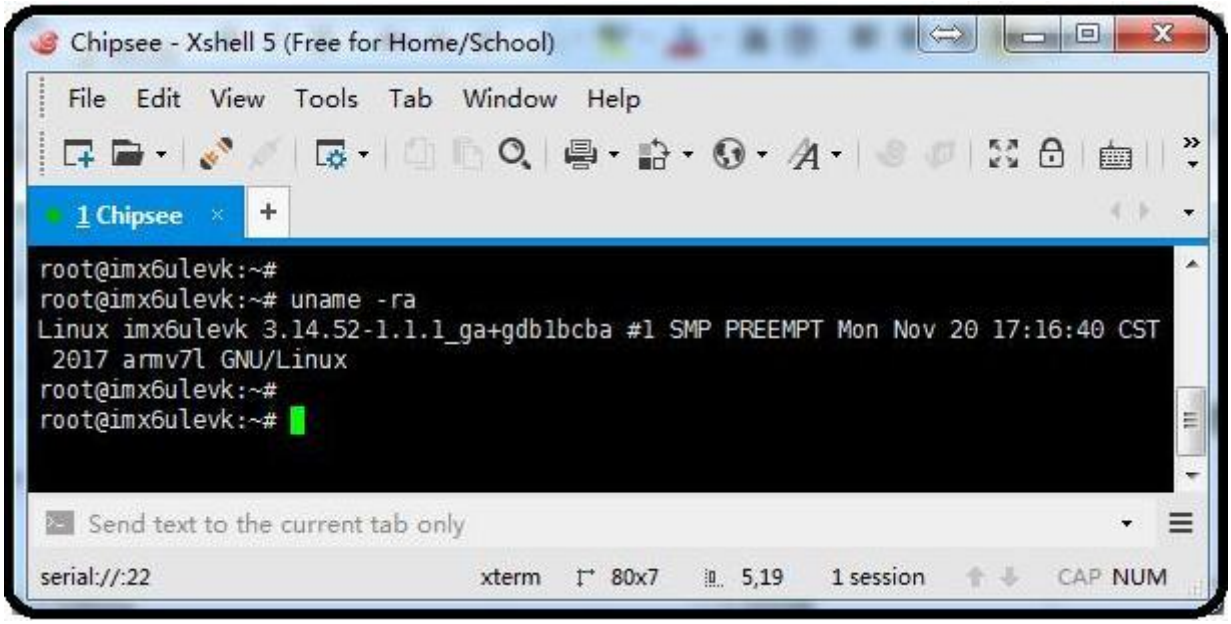

Figure 295: *SSH Debug*

VNC Debug

You can use the VNC-Viewer software in Windows to control Chipsee IPC over Ethernet.

Follow the steps below to perform VNC debug.

- Use XShell Serial or SSH to connect to the Chipsee IPC by logining as `chipsee` user.
- **Use the following command to install x11vnc:**

```
$ sudo apt-get update
$ sudo apt-get install x11vnc
```

- **Set the password for VNC-Viewer access. Save the password to default file:** *~/.vnc/passwd* **, as shown in the figure below.**

```
$ sudo x11vnc -storepasswd
```

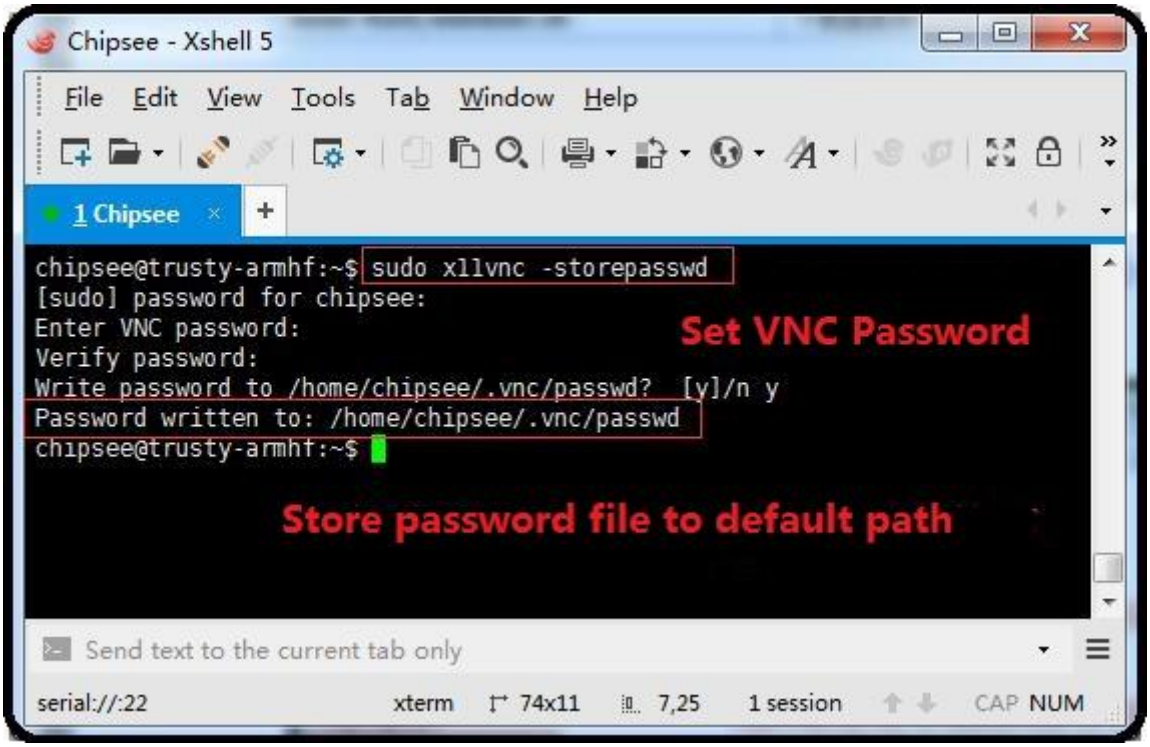

Figure 296: *VNC Password Setting*

- **Add the command below to** */etc/rc.local* **to enable the x11vnc execute after the system booted.**

```
x11vnc -display :0 -forever -bg -rfbauth /home/chipsee/.vnc/passwd -
rfbport 5900 -o /home/chipsee/.vnc/x11vnc.log
```

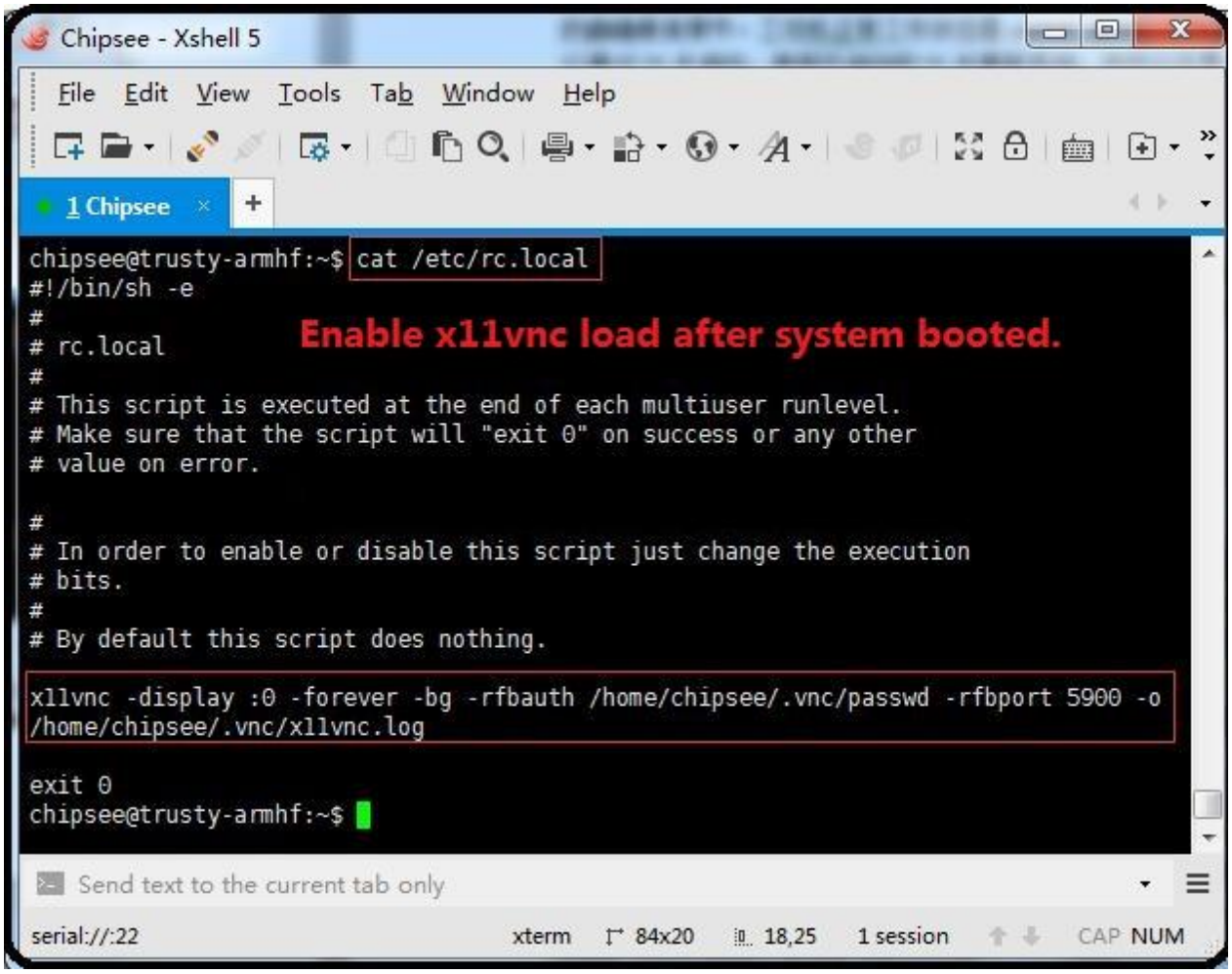

Figure 297: *x11vnc auto load*

- **Use VNC-Viewer in Windows to control it over Ethernet, as shown on the figure below.**

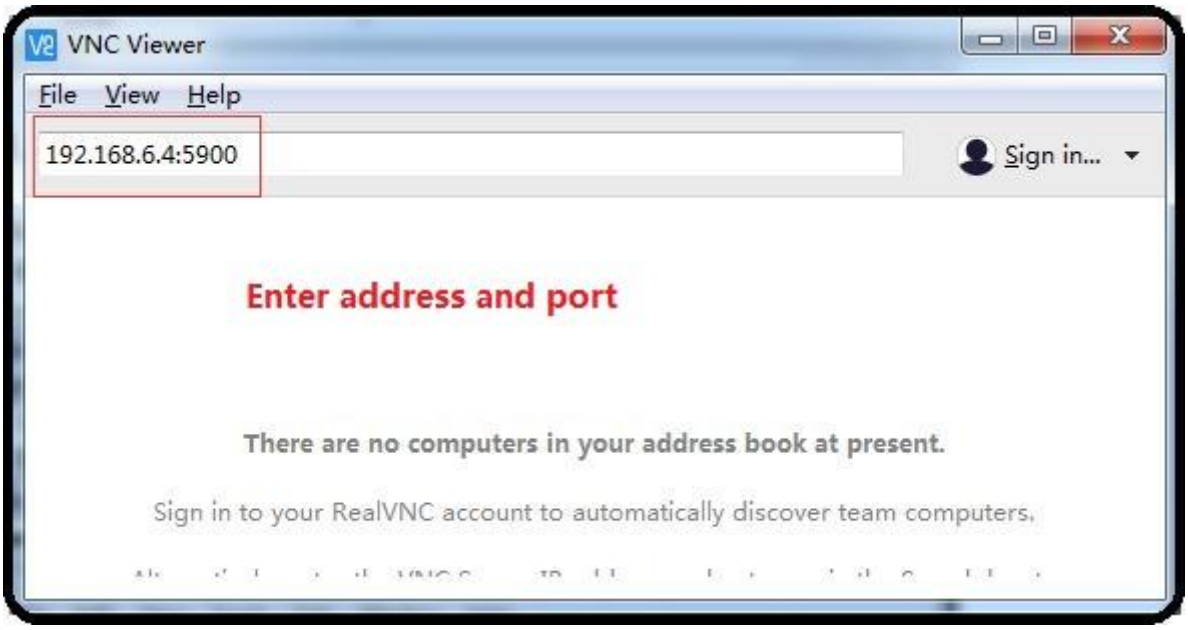

Figure 298: *VNC-Viewer Connect*

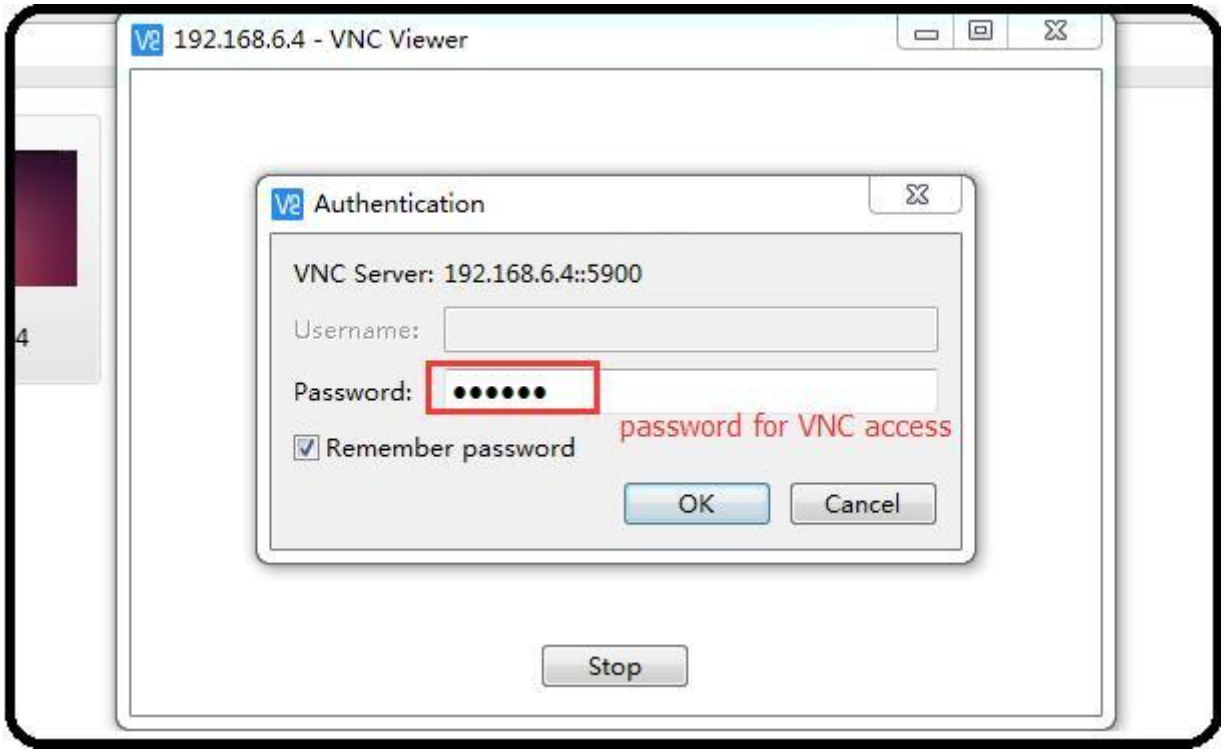

Figure 299: *Authentications*

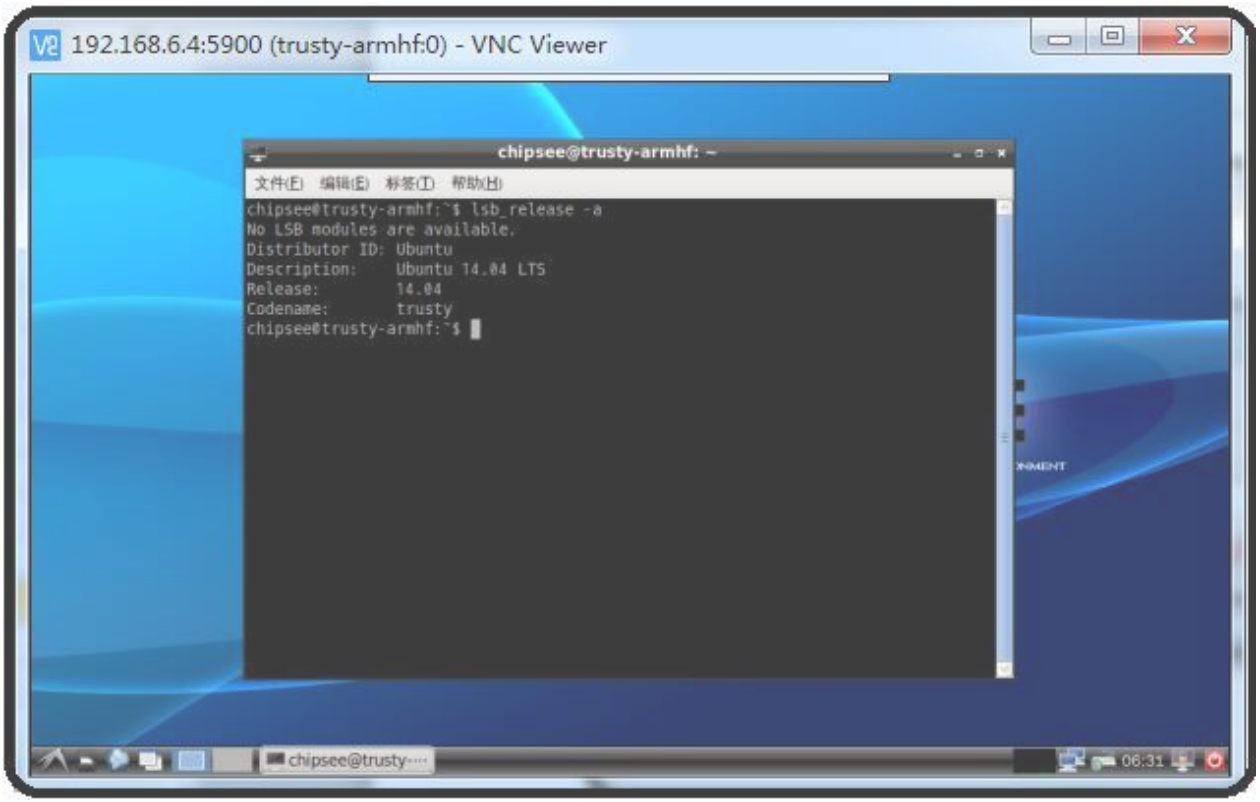

Figure 300: *VNC Desktop*

Downloading images

Boot Switch Configuration

CS-IMX6 has a boot configuration select switch, as shown on the figure below. You can use the boot select switch to change between three modes, namely

- TF Card
- eMMC Boot
- Download

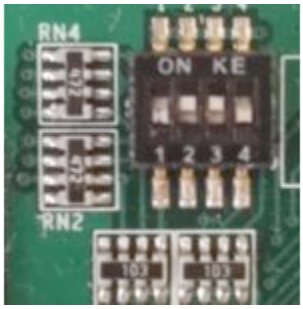

Figure 301: Boot Mode Setup

SW Mode	1	2	3	4
TF Card	1	0	0	0
eMMC	1	1	0	1
Download	0	1	1	0

Table 61 Boot Configuration Selection

> **Note**
>
> The user can use both the pre-built Ubuntu 14.04 image files and the MFGTools software to download new images to the system, boot system and perform necessary software and hardware test.

Prebuilt Files Package

You can get the Prebuilt Files Package for each model from links mentioned at the beginning of this documentation. You can also get the Prebuilt Files Package from the DVD in /Ubuntu 14.04/Prebuilds folder. However, it may be outdated so always compare the versions (the last number in the filename is the release date).

The prebuilt package has the following content:

Contents	Comment
boot/imx6q-eisd.dtb	TF Card boot dtb file
boot/u-boot-sd.imx	TF Card boot bootloader
boot/zImage	TF Card boot kernel file
boot/logo.bmp	TF Card boot logo file
filesystem/rootfs-emmc-flasher.tar.bz2	TF Card boot rootFS
mksdcard.sh	Shell tools to make bootable TF Card
README	Simple guidelines
S1.jpg	Boot Switch Config Figure
emmc-flash/emmc/rootfs.tar.bz2	RootFS in target eMMC
emmc-flash/emmc/u-boot-emmc.imx	Bootloader in target eMMC
emmc-flash/emmc/zImage	Kernel file in target eMMC
emmc-flash/emmc/zImage_framebuffer	Kernel file with frame-buffer
emmc-flash/emmc/imx6q-eisd.dtb	Dtb file in target eMMC
emmc-flash/emmc/imx6q-eisd.dtb_framebuffer	Dtb file with frame-buffer
emmc-flash/emmc/logo.bmp	Logo file in eMMC
emmc-flash/mkemmc.sh	Shell tool to download images to eMMC

Table 62 Prebuilt Files Package

Note

- The default `zImage` and `imx6q-sabresd.dtb` files support *'keep the logo from uboot to kernel'* but don't support framebuffer.
- We also provide `zImage_framebuffer` and `imx6q-eisd.dtb_framebuffer` file versions that support the framebuffer function but do not support the *'keep the logo from uboot kernel'* feature. If you need the framebufer, just rename these two files to `zImage` and `imx6q-eisd.dtb`.

Downloading Images by using MFGTool

The MFGTools can be used to download images into a target device. It is a quick and easy tool for downloading images.

Before downloading images with the MFGTools, set the boot switch to download mode. (refer to Boot Switch Configuration above)

Configuring MFGTool

To configure MFGTool, follow these steps:

- Untar `Mfgtools-K31452-Vx.x.tar.gz` file.

- Open the extracted folder `Mfgtools-K31452-Vx.x` and edit `cfg.ini` file.
- In the `cfg.ini` file, ensure the `name` and `display` variables are set to `eMMC-Ubuntu` and `1024600` respectively, as shown on the figure below.

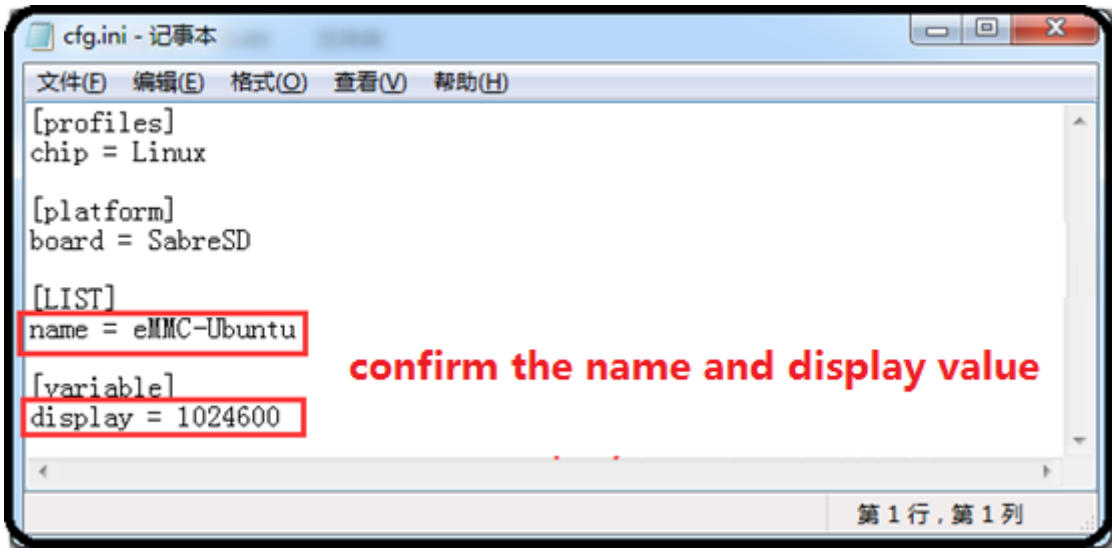

Figure 302: *Cfg.ini file*

Note

You can get the supported display from *Mfgtools-K31452-V1.0\Profiles\Linux\OS Firmware\firmware* directory. Modify config `UICfg.ini` file. This file has only one line: `PortMgrDlg=1` that indicates you can download the images to one board at the same time. The max value is 4.

COPY IMAGE TO ANDROID DIRECTORY

Follow these steps to copy image to Linux directory:

- Copy the images from *prebuilt-xxx/emmc-flash/emmc/* to *Mfgtools-K31452-V1.0\Profiles\Linux\OS Firmware\files\ubuntu* directory.

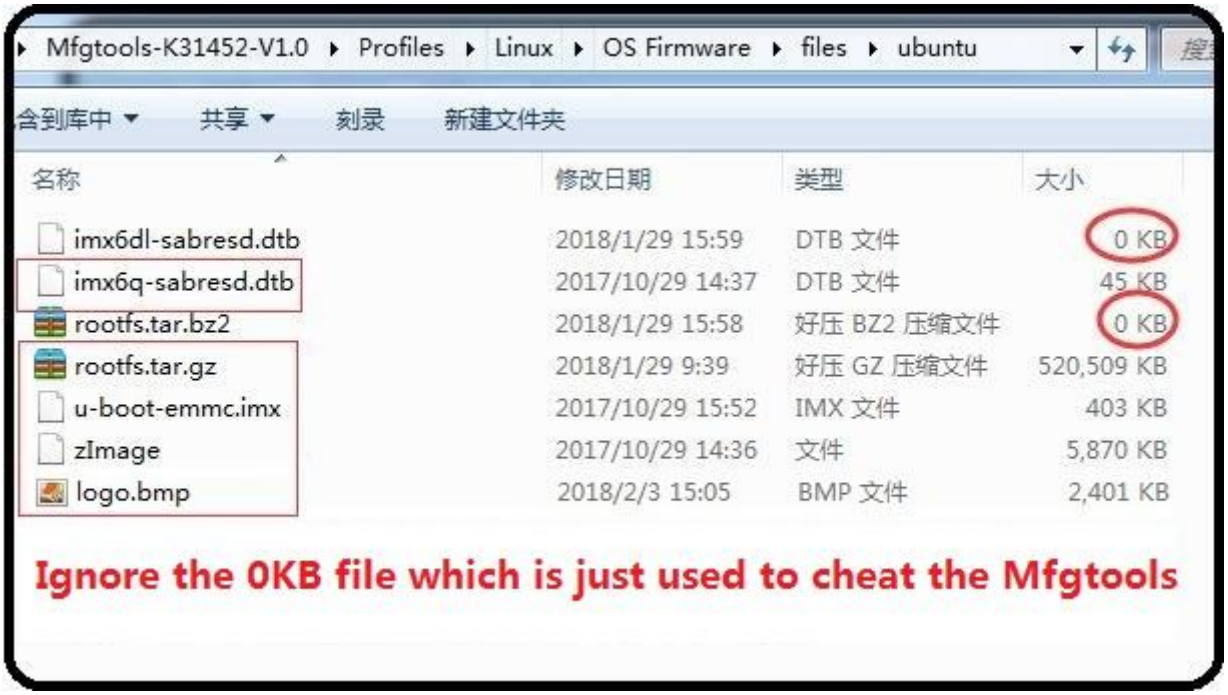

Figure 303: *Prepare Images*

USING MFGTOOL

1. Connect a USB OTG cable from a Windows PC to the USB OTG port on the IPC.
2. **Change the boot select configuration to 0 1 1 0, as shown on the figure below.**

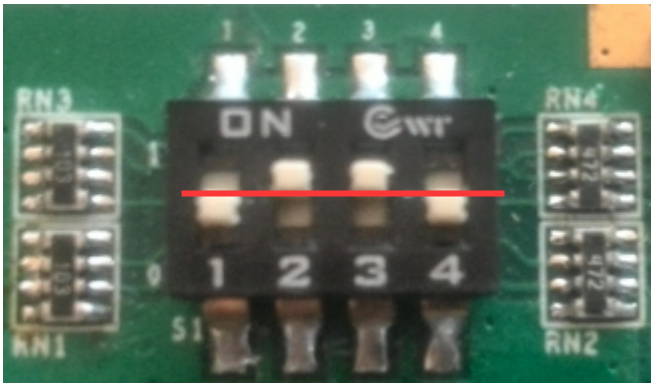

Figure 304: *Boot Switch Config*

3. Connect a 12V-2A power adapter to the IPC and power ON.
4. **On your Windows PC, open the `Mfgtools-Rel-XXX_XXXXXX_MX6Q_UPDATER_VXX` directory and run the `MfgTool2.exe` file, as shown on the figure below.**

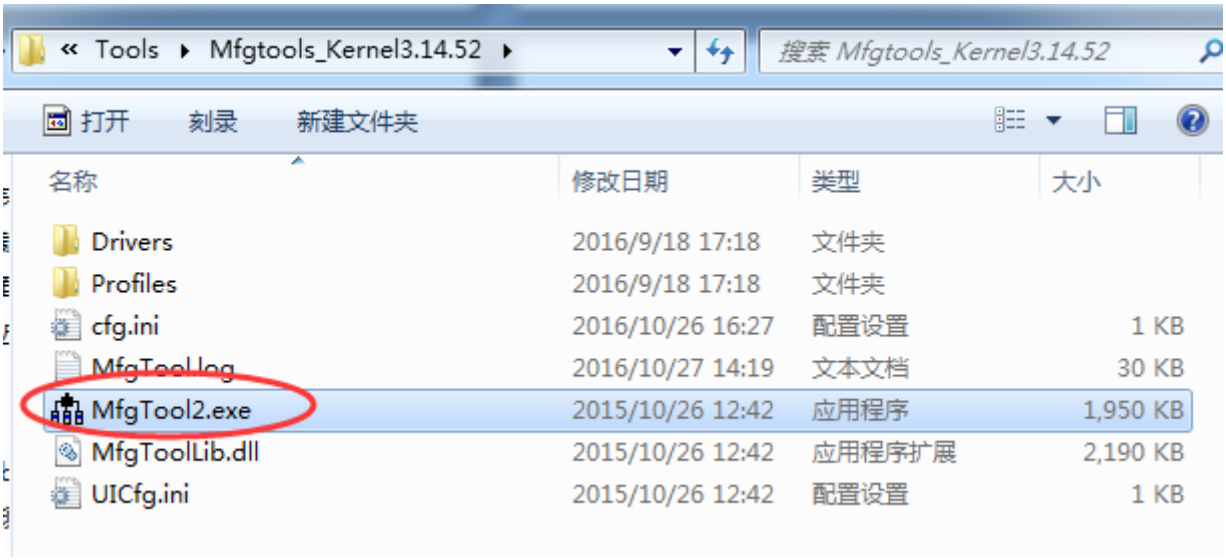

Figure 305: *Run* **MfgTools2.exe** *file*

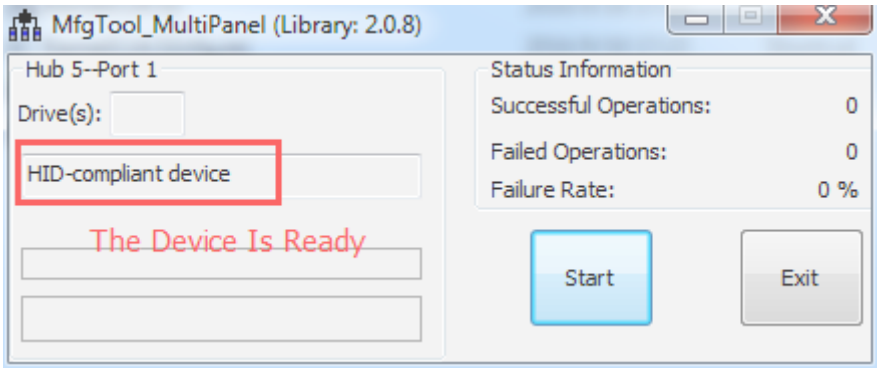

Figure 306: *Prepare to start*

Note

If you get a message saying *No Device Connected*, check the USB-OTG cable to ensure it is ready.

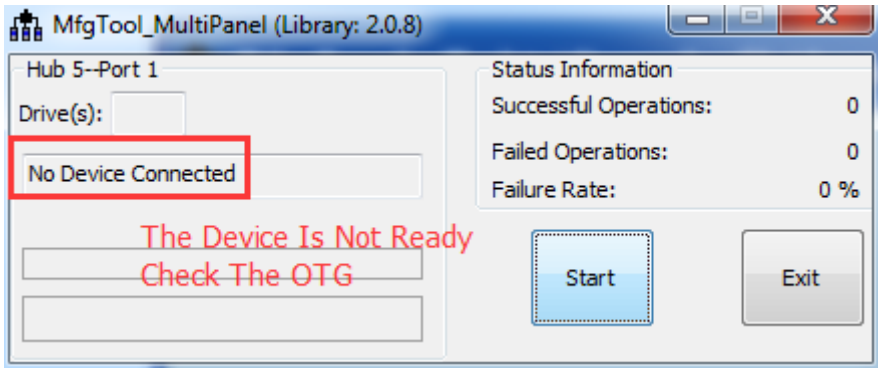

Figure 307: *The USB-OTG cable is not connected correctly.*

5. **Click on Start button to download the Image.**

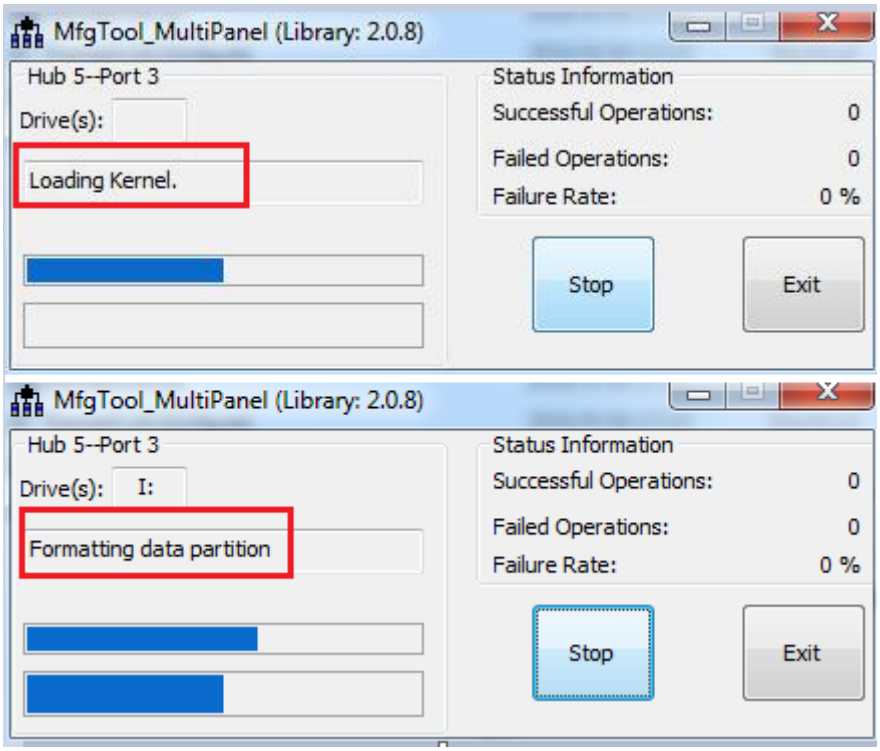

Figure 308: *Downloading Images*

Note

If you are using a Window 7 PC, you will receive a prompt that asks you to format the disk. Please ignore or cancel it.

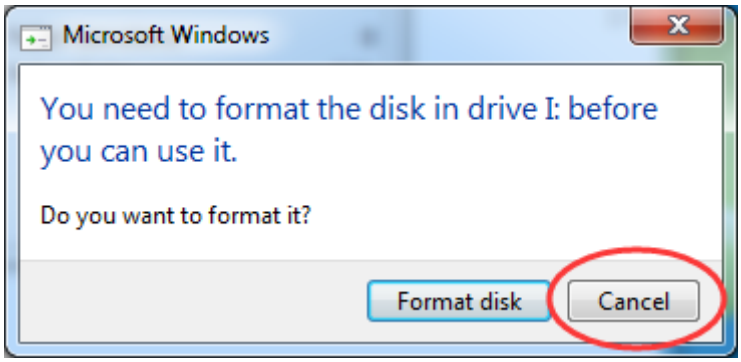

Figure 309: *Cancel format disk*

6. **When the process is complete, you click the Stop button to stop downloading Image and exit.**

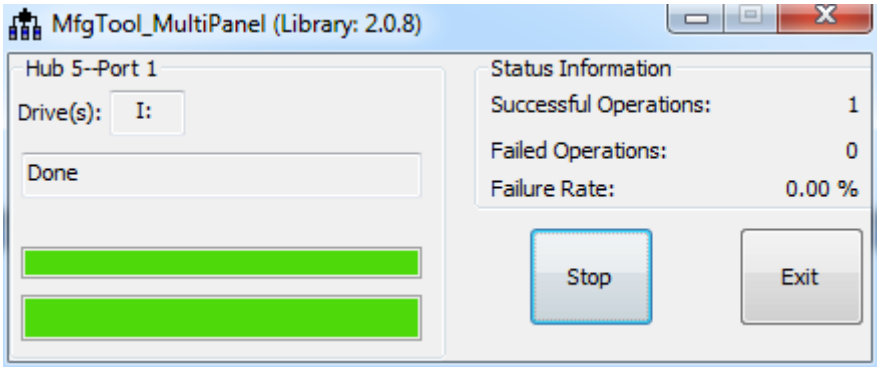

Figure 310: *Download Image is finished*

Downloading Images by using the TF card

Follow the steps below to download images onto the eMMC by using the TF Card:

1. Copy the Prebuilt Files Package to a Linux environment (such as Ubuntu 14.04).
2. Insert the SD card into your computer. If you are using virtual machines, please ensure the SD card is mounted to the Linux operating system.
3. **Confirm the SD card mount point, `/dev/sdX` (e.g., `/dev/sdc` or `/dev/sdb`, be sure to use the right one). In a Linux system, you can use the command below to find out what *X* is.**

   ```
   $ sudo fdisk –l
   ```

4. Copy the `prebuilt-imxv1-csXXXXXfXXXvX-android6-emmc-YYYYMMDD.tar.gz` to somewhere(such as $HOME) on the Ubuntu PC.
5. **Extract the `prebuilt-imxv1-csXXXXXfXXXvX-android6-emmc-YYYYMMDD.tar.gz`**

   ```
   $ tar -xzvf prebuilt-imxv1-csXXXXXfXXXvX-android6-emmc-YYYYMMDD.tar.gz
   ```

6. **Go to the folder**

   ```
   $ cd prebuilt-imxv1-csXXXXXfXXXvX-android6-emmc-YYYYMMDD
   ```

7. **Use the following command to flash the Ubuntu 14.04 OS to the SD card**

   ```
   $ sudo ./mksdcard.sh --device /dev/sd<?>
   ```

   > **Note**
   >
   > • sd<?> means the SD card mount point, (e.g., `/dev/sdc` or `/dev/sdb`) in Ubuntu system.
   >
   > • The recommended SD card should be Sandisk Class4 level SD card or above.

8. The bootable SD Card is now ready. Power OFF the industrial PC and insert the SD Card.
9. Set the switch S1 to TF card boot mode. (refer to Boot Switch Configuration above)
10. Connect the industrial PC to PC via COM1. Power ON the IPC.
11. After 20 minutes, if the LED on industrial PC stays lit, flashing is completed. Using COM1, you can also find this message **>>>>>>> eMMC Flashing Completed <<<<<<<** which indicates that the system image was downloaded correctly to the eMMC.
12. Power OFF and set the switch S1 to eMMC boot mode. (refer to Boot Switch Configuration above)

System Resource

TF Card/USB/SATA Disk

The TF Card and USB Storage supports hot-plug but the SATA Disk does not support hot-plug. These devices will be automatically mounted on */media/chipsee/* , as shown in the figure.

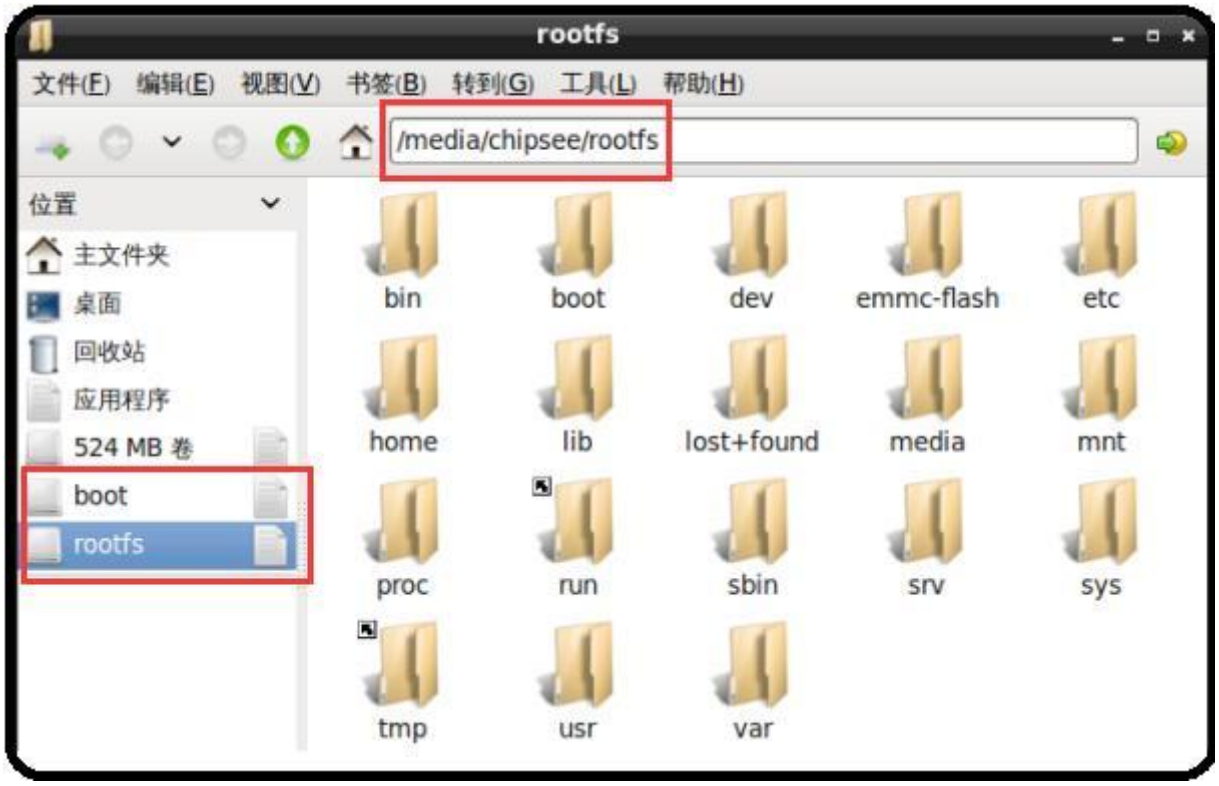

Figure 311: *TF Card*

> **Note**
>
> The TF card and USB Storage do not support NTFS format. Please format it to FAT32 first before plugging into IPC.

Network

This system uses a networking service to control Ethernet and uses `wpa_supplicant` to control the WIFI network.

Wired Ethernet

You can get the IP address from the following application, as shown on the figure below.

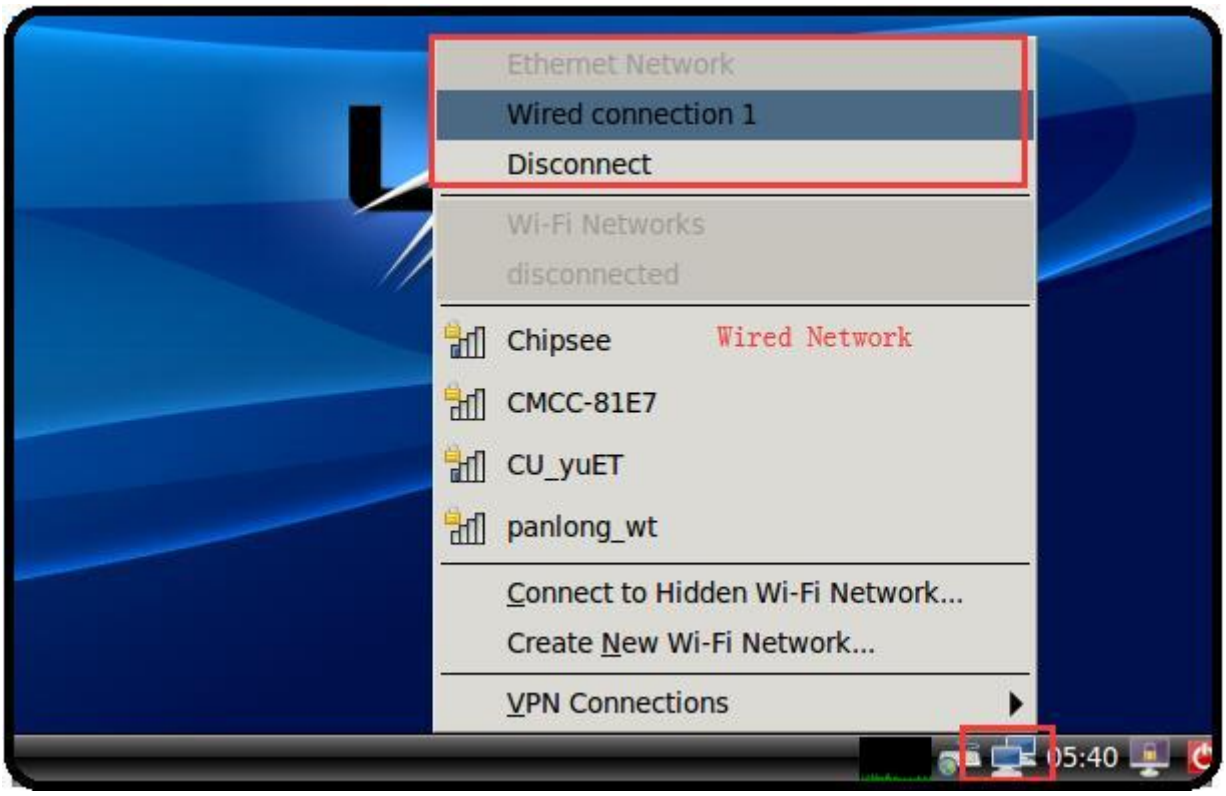

Figure 312: *Wired Connection*

Wi-Fi

- **Disconnect wired connection before you use Wi-il. We will connect to the *Chipsee* network. Fill in the password, as shown on the figure below.**

Figure 313: *Wi-Fi Password*

- **Next, you will get the dialog which will request you to set the password for the new keyring. Just leave it blank or set a password for yourself, as shown on the figure below. We advise you to leave it blank, in order for the WiFi to connect automatically during the next boot.**

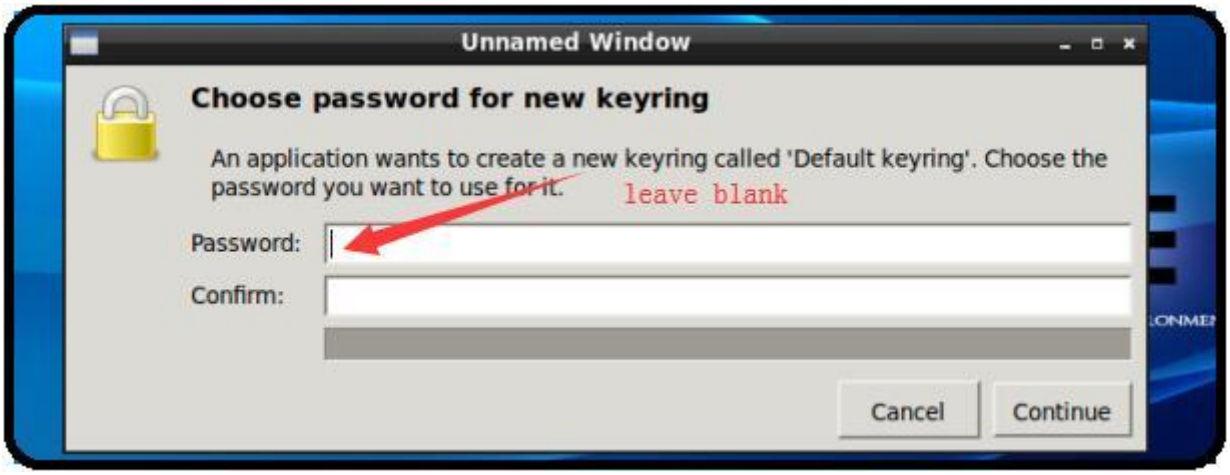

Figure 314: *Keyring setting*

If you set the keyring and want to reset it, do the following:

- **Open Preferences->Passwords and Keys, as shown on the figure below:**

Figure 315: *Passwords and Keys*

- **Right click Default keyring tab to change the Password, and set it to blank, as shown on the figure below.**

Figure 316: *Change the keyring password*

Remove and Install Network-manager Packages

If you want to set a static IP, you can use the Networking Service to manage your network. Before that, you need to remove the Network-manager Package and reboot the IPC board. You can use this command to remove the packages:

```
$ sudo apt-get remove --purge network-manager
$ sudo apt-get autoremove --purge network-manager
```

If you want to reinstall it, use this commands:

```
$ sudo apt-get install network-manager
```

Networking — Wired Ethernet

You can get the `interfaces` file from */etc/network/* directory, this is the config file for the Networking service.
The following are some examples on how to set the network.

- **Set wired Ethernet to use DHCP in obtaining IP. Edit the `interfaces` file by adding these lines**

```
### ethX demo
### For ethX uncomment follow two lines.
allow-hotplug  eth0
auto  eth0
## ethX dhcp demo
iface  eth0  inet  dhcp
```

- **Set wired Ethernet to use Static IP. Edit the `interfaces` file by adding these lines**

```
### ethX demo
### For ethX uncomment follow two lines.
allow-hotplug  eth0
auto  eth0
## ethX static demo
iface  eth0  inet  static
pre-up  ifconfig  eth0  hw  ether  00:22:44:66:88:AA  //Set MAC
address  192.168.6.98
netmask  255.255.255.0
gateway  192.168.6.1
dns-nameservers  8.8.8.8      // set DNS
```

Networking — WIFI

You can get the `interfaces` file from */etc/network/* directory, this is the config file for the Networking service.
The following are some examples on how to set the network.

- **Enable Wi-Fi and set it to use DHCP to obtain IP. Edit the `interfaces` file by adding these lines**
    - **Use the following command to set the SSID and Password of Wi-Fi, and generate */etc/wpa_supplicant.conf* .**

```
# wpa_passphrase  "your ssid"  " your password "  >  /etc/
wpa_supplicant.conf
```

    - **Modify */etc/network/interfaces* , like this:.**

```
auto  wlan0
iface  wlan0  inet  dhcp
wireless_mode  managed
wireless_essid  any
wpa-driver  nl80211
wpa-conf  /etc/wpa_supplicant.conf
```

- **Enable Wi-Fi and set it to use a Static IP. Edit the `interfaces` file by adding these lines**

```
iface wlan0 inet static

address  192.168.1.98
netmask  255.255.255.0
gateway  192.168.1.1
dns-nameservers  8.8.8.8

wireless_mode  managed
wireless_essid  any
wpa-driver  nl80211
wpa-conf  /etc/wpa_supplicant.conf
```

> **Note**
>
> This system uses `wpa_cli` and `wpa_supplicant` to manage Wi-Fi that supports `nl80211`. There is no wireless tools and you can't use iwconfig and iwlist.

Multimedia

This system supports NXP Gstreamer-imx Multimedia library and its various plugins.

```
root@imx6qsabresd:~# gst-inspect-1.0 | grep imx
overlaysink.imx:  overlaysink: IMX Video (video compositor) sink
imxmp3enc.imx:  imxmp3enc: imx mp3 audio encoder
beep.imx: ac3: [Invalid UTF-8] `Z\xc7s
beep.imx: 3ca: [Invalid UTF-8] `Z\xc7s
beep.imx:  beepdec: Beep universal decoder
imxcompositor.imx:  imxcompositor_ipu: IMX ipu Video Compositor
imxcompositor.imx:  imxcompositor_g2d: IMX g2d Video Compositor
vpu.imx:  vpuenc_h264: VPU-based AVC/H264 video encoder
vpu.imx:  vpuenc_mpeg4: VPU-based MPEG4 video encoder
vpu.imx:  vpuenc_h263: VPU-based H263 video encoder
vpu.imx:  vpuenc_jpeg: VPU-based JPEG video encoder
vpu.imx:  vpudec: VPU-based video decoder
imxvideoconvert.imx:  imxvideoconvert_ipu: IMX ipu Video Converter
imxvideoconvert.imx:  imxvideoconvert_g2d: IMX g2d Video Converter
aiur.imx: webm: [Invalid UTF-8] \xa0!\xf0u
aiur.imx:  aiurdemux: Aiur universal demuxer
imxv4l2.imx:  imxv4l2sink: IMX Video (video4linux2) sink
imxv4l2.imx:  imxv4l2src: IMX Video (video4linux2) Source
root@imx6qsabresd:~#
```

Figure 317: *GStreamer Plugins*

Audio Test

You can use the command below to record music. The `-d` parameter means *interrupt after # seconds*. In this example, `-d` is equal to 18 seconds.

```
$ sudo  arecord  -N  -M  -r  44100  -f  S16_LE  -c  2  -d  18  test.wav
```

You can use the command below to playback the recorded sound above.

```
$ sudo  aplay  -N  -M  test.wav
```

You can also use the **LXMusic** to playback audio.

Figure 318: *LXMusic*

Set output as ALSA, as shown on the figure below.

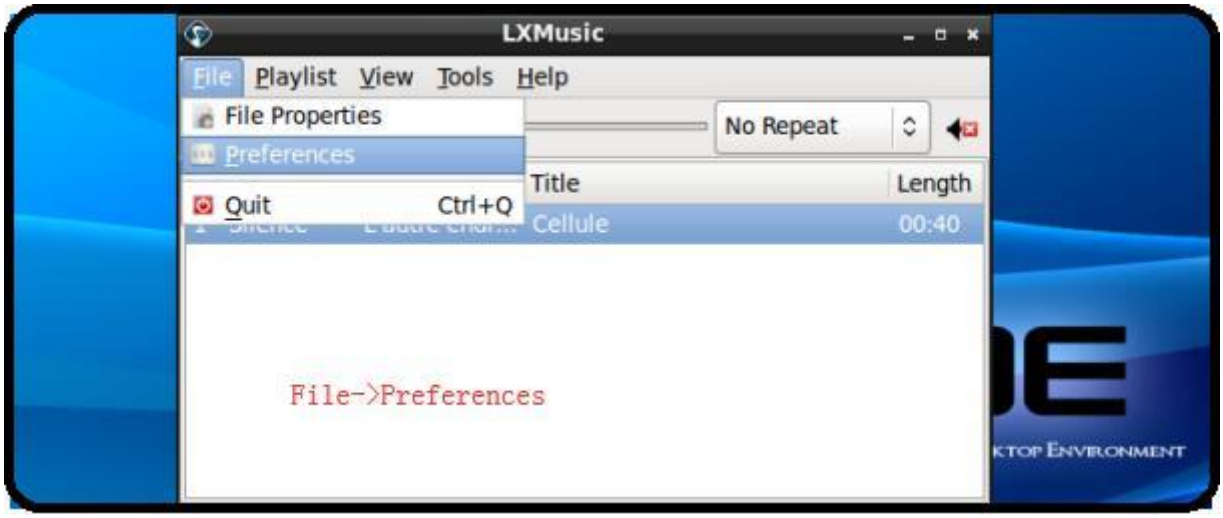

Figure 319: *Set Audio Plugin*

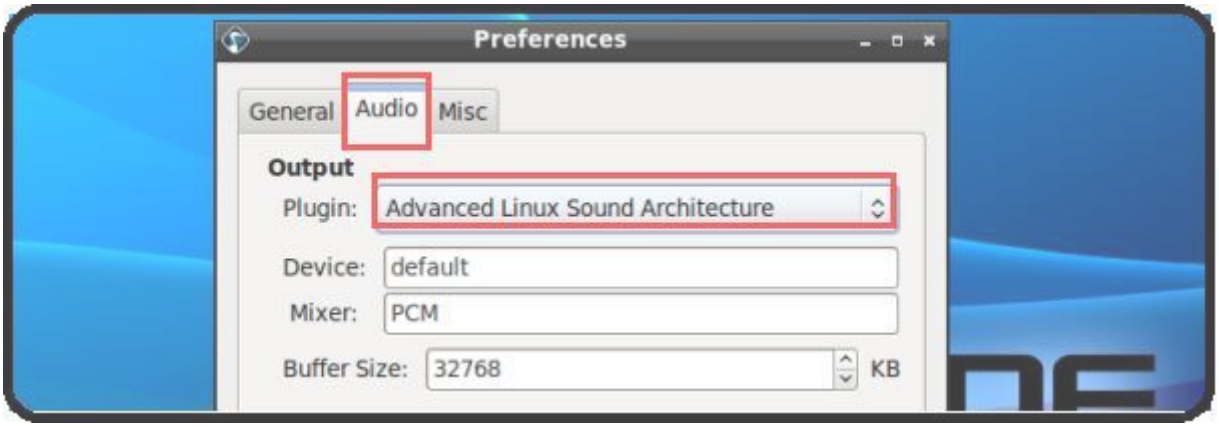

Figure 320: *Set Audio Plugin*

HDMI

You can follow the steps below to display the IPC output onto an external display via HDMI.

- Power OFF IPC. Connect the external display to the IPC using an HDMI cable.
- Refer to the Serial Debug section to set serial debug.
- **Power ON IPC. In XShell, hit any key to stop auto boot and input the uboot command mode, as shown on the figure below.**

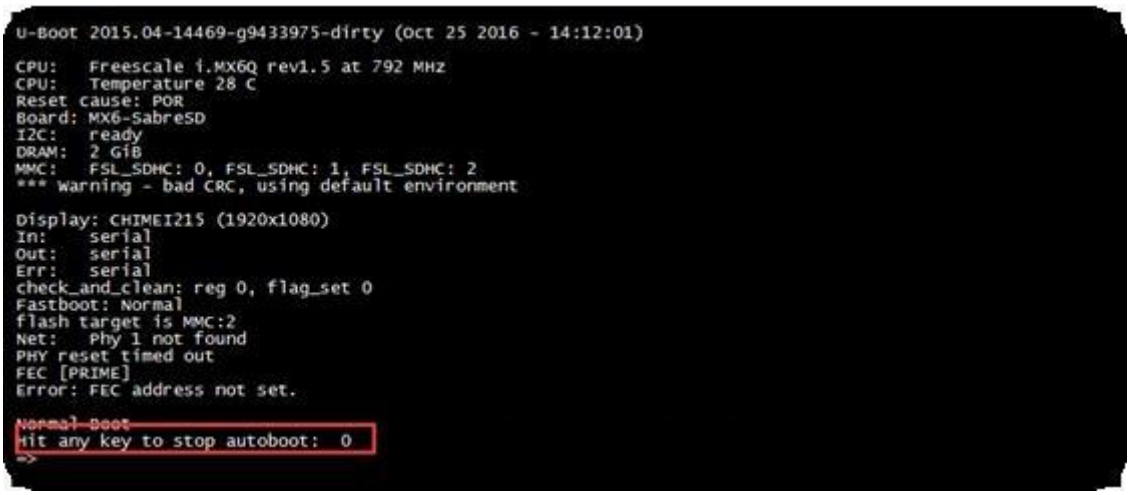

Figure 321: *Uboot*

> **Note**
>
> HDMI does not support hot-plug. The sound comes from the HDMI monitor, neither the speaker nor the headset on board.

- **Use the following command to set different resolution**
  - **For 1080p**

```
=> setenv displayargs video=mxcfb0:dev=hdmi,1920x1080M@60
video=mxcfb1:dev=off video=mxcfb2:off
=> saveenv
=> boot
```

  - **For 720p**

```
=> setenv displayargs video=mxcfb0:dev=hdmi,1280x720M@60
video=mxcfb1:dev=off video=mxcfb2:off
=> saveenv
=> boot
```

  - **For 480p**

```
=> setenv displayargs video=mxcfb0:dev=hdmi,800x480M@60
video=mxcfb1:dev=off video=mxcfb2:off
=> saveenv
=> boot
```

Figure 322: *HDMI Output Setting*

- Reboot the IPC.
- **Use the following command to reset the output from LDB.**

```
=> setenv displayargs video=mxcfb0:dev=ldb video=mxcfb1:dev=off
video=mxcfb2:off
=> saveenv
=> boot
```

Serial Port

There are five serial ports on the Chipsee IPC: 2 x RS232 and 3 x RS485 (can be customised). Refer to the table below for the available serial device nodes.

The default serial port configuration is 2 x RS232, 2 x RS485, 1 x RS485 which is shared with Bluetooth.
Contact us if you need help with changing the default serial port configuration

Ports	Device Node
COM1(RS232, Debug)	/dev/ttymxc0
COM2(RS485)	/dev/ttymxc1
COM3(RS232)	/dev/ttymxc2
COM4(RS485)	/dev/ttymxc3
COM5(RS485)	/dev/ttymxc4

Table 63 Serial Ports Nodes on the System

> **Note**
>
> If you use COM2(RS485), you can't use Bluetooth because COM2(RS485) share pin with Bluetooth.

You can install **cutecom** to test the serial port:

```
$ sudo apt-get install cutecom
```

Only users with root permissions can use the serial port

```
$ sudo cutecom
```

CAN Bus

Chipsee Industrial PC is equipped with two CAN busses (CAN1 and CAN2). Two devices can be interconnected. You can test the CAN buses by using the **HT application** but you must add one 120Ω resistor between CAN_H and CAN_L on one of the two Boards, as shown on the figure below.

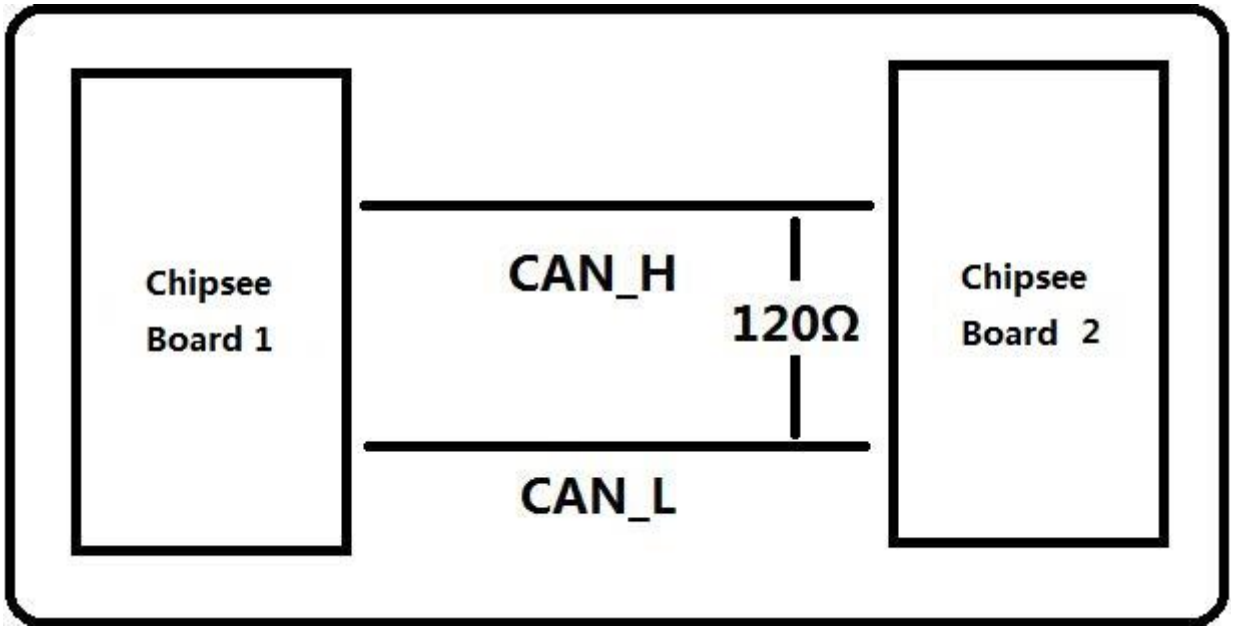

Figure 323: *CAN Connect*

> **Note**
>
> The Chipsee IPC does not mount the 120Ω matched resistor on all CAN signals by default.

Here are a few examples to test CAN by using CAN units

- **Install** *can-utils*

  ```
  $ sudo apt install can-utils
  ```

- **Set the bit-rate to 50Kbits/sec with triple sampling using the following command (use ROOT user):**

```
$ sudo ip link set can0 type can bitrate 50000 triple-sampling on
```

OR

```
$sudo canconfig can0 bitrate 50000 ctrlmode triple-sampling on
```

- **Bring up the device using the command:**

```
$ sudo ip link set can0 up
```

OR

```
$ sudo canconfig can0 start
```

- **Transfer packets**
  - Transmit 8 bytes with standard packet id number as 0x10

```
$ sudo cansend can0 -i 0x10 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88
```

  - Transmit 8 bytes with extended packet id number as 0x800

```
$ sudo cansend can0 -i 0x800 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 - e
```

  - Transmit 20 8 bytes with extended packet id number as 0xFFFFF

```
$ sudo cansend can0 -i 0xFFFFF 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 -e
--loop=20
```

- **Receive data from CAN bus**

```
$ sudo candump can0
```

- **Bring down the device**

```
$ sudo ip link set can0 down
```

GPIO

There are 8 GPIOs, 4 Output, and 4 Input, they are all isolated. You can control the output or input pin voltage by feeding the VDD_ISO suite voltage. The pin voltage should be from 5V to 24V. Refer to the tables below for a detailed port definition:

Pin Number	GPIO Number
11	205
12	106
13	29
14	30
15	28
16	204
17	94
18	95

Table 64 CS80480F070 – V1.0 P11 Port

Pin Number	GPIO Number
21	106
22	29
23	30
24	28
27	95
28	94
29	87
30	130

Table 65 CS10600F070 – V1.0 P21 Port

Pin Number	GPIO Number
21	29
22	106
23	28
24	30
27	130
28	87
29	94
30	95

Table 66 CS10600F070 – V2.0 P21 Port

Pin Number	GPIO Number
3	106
4	30
6	95
7	87
8	29
9	28
11	94
12	130

Table 67 CS12800F010 – V1.0 P28 Port

> **Note**
>
> You need `ROOT` permissions to control GPIO.

Set *gpio106 Output* to high or low using this command

```
# echo  106  > /sys/class/gpio/export                //export gpio106
# echo  out  > /sys/class/gpio/gpio106/direction     //set gpio106 Output
# echo  1  > /sys/class/gpio/gpio106/value           //Set gpio106 high
# echo  0  > /sys/class/gpio/gpio106/value           //Set gpio106 low
```

Set *gpio30 Input* using this command

```
# echo  30  > /sys/class/gpio/export                  //export  gpio30
# echo  in  > /sys/class/gpio/gpio30/direction        //Set  gpio30 input
```

Un-export *gpio30* using this command

```
# echo  30  > /sys/class/gpio/unexport                 //un-export  gpio30
```

Buzzer

The buzzer is one GPIO, which has the GPIO Number as 80.

You can test the buzzer with the following commands.

```
# echo  80  > /sys/class/gpio/export                 //export gpio80
# echo  out  > /sys/class/gpio/gpio80/direction      //set gpio80 output
# echo  1  > /sys/class/gpio/gpio80/value            //Open Buzzer
# echo  0  > /sys/class/gpio/gpio80/value            //Close Buzzer
```

You also can use the HT application to test the buzzer.

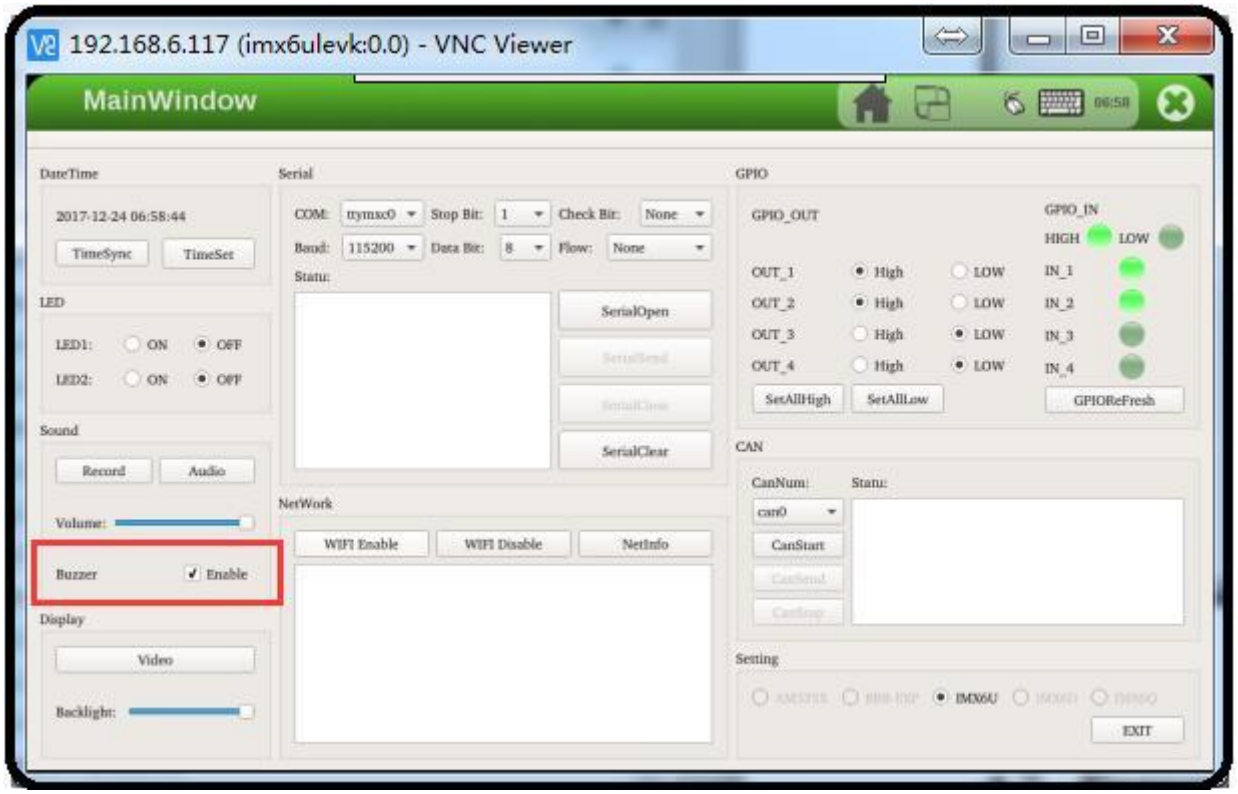

Figure 324: *Buzzer*

Modify Logo

This system supports changing the logo by yourself. There are two ways:

- Replace the logo file in prebuilt images packages, and download images.
- Change the logo without downloading images.

> **Note**
>
> Logo file is one 32bpp, format is bmp.

Method 1 - Downloading images

Replace the *prebuilt-xxx/emmc-flash/emmc/logo.bmp* and reference Downloading Images by using MFGTool to flash the image.

Method 2 - Don't Download Images

We will use **MFGTools** and the **Logoflasher** apps to change the logo.

Use MFGTools to Change LOGO

- Replace the `logo.bmp` file in *Mfgtools-K31452-V1.0\Profiles\Linux\OS Firmware\files\ubuntu* with your customised logo file.
- **Open and edit the** *Mfgtools-K31452-V1.0\cfg.ini* **file and set the** `name` **variable to** `eMMC-Ubuntu-Logo` **as shown below.**

Figure 325: *Change name*

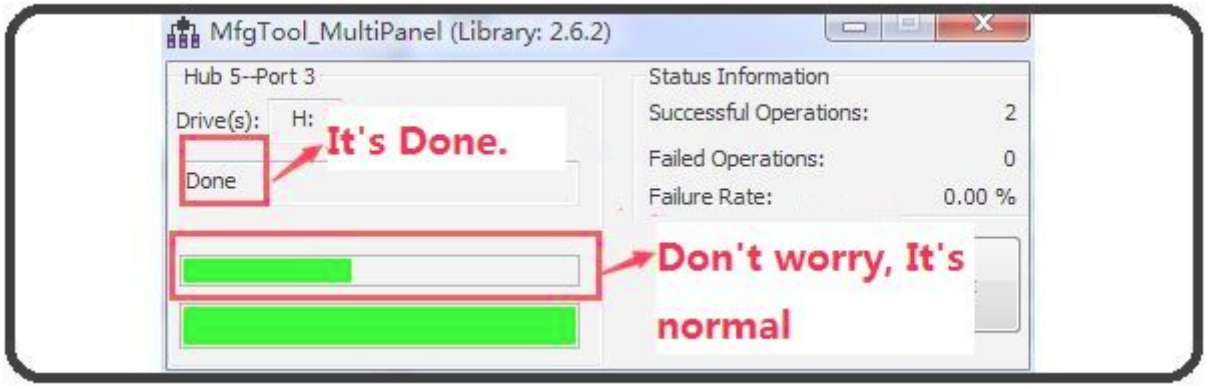

Figure 326: *Logo Modify with MFGTool*

Use Logoflasher to Change Logo

You can get the Logoflasher file and use these tools to make one bootable TF card. Follow the steps below to change logo

- **Use the following commands to make bootable TF card.**

```
$ sudo  tar  zxvf  prebuilt-imx6qdl-bootfile-update-xxx.tar.gz
$ sudo  cd  prebuilt-imx6qdl-bootfile-update-xxx
$ sudo  ./mksdcard.sh  --device  /dev/sdX --display 1024600     //
resolution
```

- Put your custom logo file in the first partition `boot-flash` directory on the TF Card.

- Set boot mode to **TF card**. You can reference Boot Switch Configuration.
- Power ON the IPC. If you see this message, **>>>>>>> eMMC Flashing Completed <<<<<<<**, you are done:

Development

In this chapter, you will learn how to set up the Python3 and QT development environment, and develop the first QT application on Chipsee IPC boards.

Python

In this example, we will develop one Python3 GUI application.

- **First, you must install the Tkinter package using this command:**

```
$ sudo apt-get install python3-tk
```

- **Create a hello_world.py file and use the following code:**

```
 1 #!/usr/bin/env python3
 2 # -*- coding: UTF-8 -*-
 3
 4 import tkinter as tk
 5
 6 rt = tk.Tk()
 7 rt.resizable(False,False)
 8 rt.title("ChipseePython")
 9
10 rt.update()
11 curWidth = rt.winfo_reqwidth()
12 curHeight = rt.winfo_height()
13 scnWidth,scnHeight = rt.maxsize()
14
15 tmpcnf = '%dx%d+%d+%d'%(curWidth,curHeight,
16 (scnWidth-curWidth)/2,(scnHeight-curHeight)/2)
17 rt.geometry(tmpcnf)
18
19 tim=tk.Label(rt,text="Hello Chipsee",font=("Arial",
14,"bold"),bg='yellow',justify='left')
20 tim.pack(expand="yes",fill="both")
21
22 rt.mainloop()
```

- **Save the file. Run it using this command.**

```
$ python3 hello_world.py
```

Figure 327: *Python App*

Qt Environment

There is no Qt environment and build environment in this system, you need to install Qt and set a build environment first. Then we will develop one Qt application.

- **Use the following command to prepare and set the Qt Environment.**

```
$ sudo apt-get update
$ sudo apt-get install build-essential git libudev-dev
$ sudo apt-get install qt5-default // or qt4-default if you want to use
qt4
$ sudo apt-get clean
```

- **We use hardwaretest_serial to demonstrate this development exercise. To perform this demo, we need to install `qtserialport` support first using this commands:**

```
$ cd ~
$ git clone git://code.qt.io/qt/qtserialport.git
$ cd qtserialport
$ git checkout 5.3    // for qt4 is “git checkout qt4-dev”
$ cd ../
$ mkdir qtserialport-build
$ cd qtserialport-build
$ qmake ../ qtserialport/qtserialport.pro
$ make
$ sudo make install
```

- Use SSH or USB Storage to put `hardwaretest_serial_ok_20170223.tar.gz` file onto Chipsee IPC board.

Now we are in Chipsee IPC Debian system console.

- **Use the following command to build the Qt application:**

```
$ tar zxvf hardwaretest_serial_ok_20170223.tar.gz
$ cd hardwaretest_serial
$ qmake
$ make
```

- **Modify the permission of the serial ports device node**

```
$ sudo chmod 666 /dev/ttymxc*
```

- **Run the `hardwaretest_serial` app using this command:**

```
$ cd hardwaretest_serial
$ export DISPLAY=:0
$ ./hardwaretest_serial
```

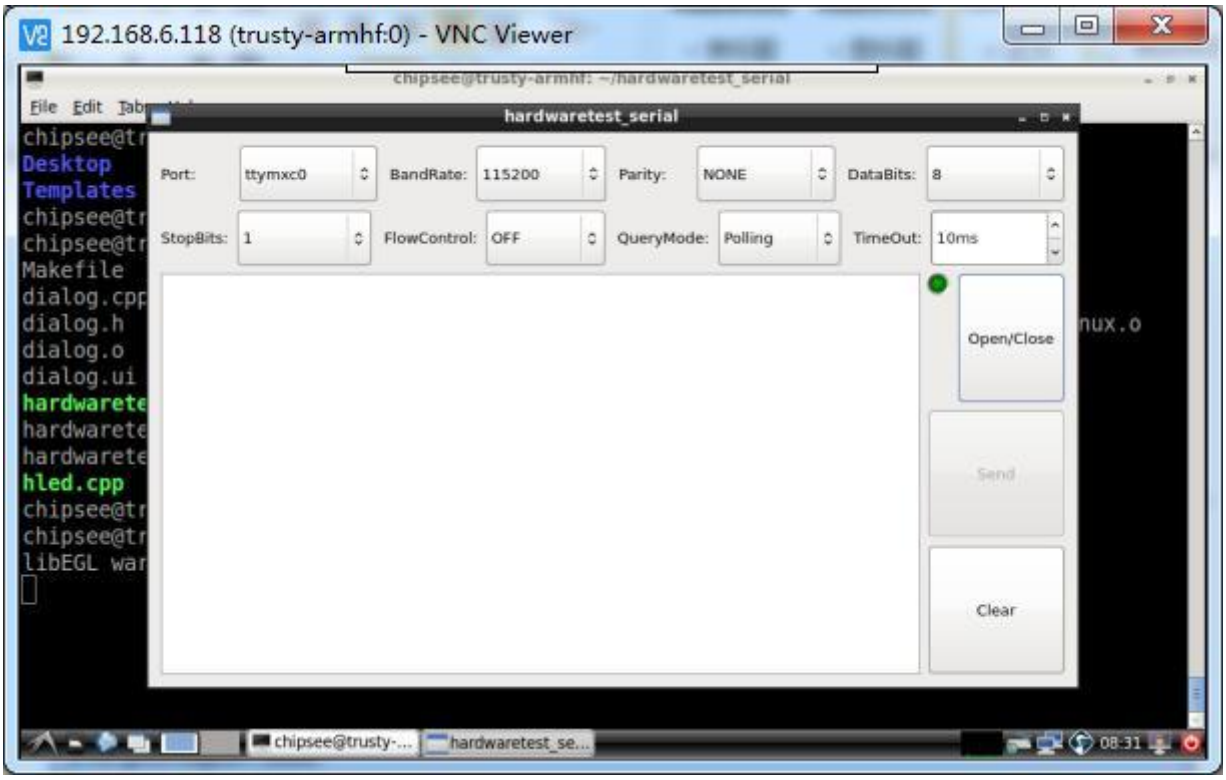

Figure 328: *hardwaretest_serial App*

Q&A

In this chapter, you can learn how to set up the QT development environment, and develop the first QT application on Chipsee IPC boards.

How to rotate the display

Modify */etc/X11/xorg.conf* and */usr/share/X11/xorg.conf.d/10-evdev.conf* to rotate the display and touchscreen. If the files do not exist, please create a new one.

- */etc/X11/xorg.conf*

```
Section "Device"
        Identifier      "Builtin Default fbdev Device 0"
        Driver          "fbdev"
#       Option          "Rotate" "CW"   // 90°
#       Option          "Rotate" "UD"   // 180°
#       Option          "Rotate" "CCW"   // 270°
EndSection
```

- */usr/share/X11/xorg.conf.d/10-evdev.conf*

```
Section "InputClass"
        Identifier "evdev touchscreen catchall"
        MatchIsTouchscreen "on"
        MatchDevicePath "/dev/input/event*"
#90°
#       Option "SwapAxes" "True"         //Swap X Axes and Y Axes
#       Option "InvertY" "True"           //Invert Y Axes
#180°
#       Option " InvertX" "True"          // Invert X Axes
#       Option "InvertY" "True"           //Invert Y Axes
#270°
#       Option "SwapAxes" "True"         //Swap X Axes and Y Axes
#       Option "InvertX" "True"           //Invert X Axes

        Driver "evdev"
EndSection
```

How to disable the Screensaver

Open the Screensaver Setting dialog, as shown on the figure below.

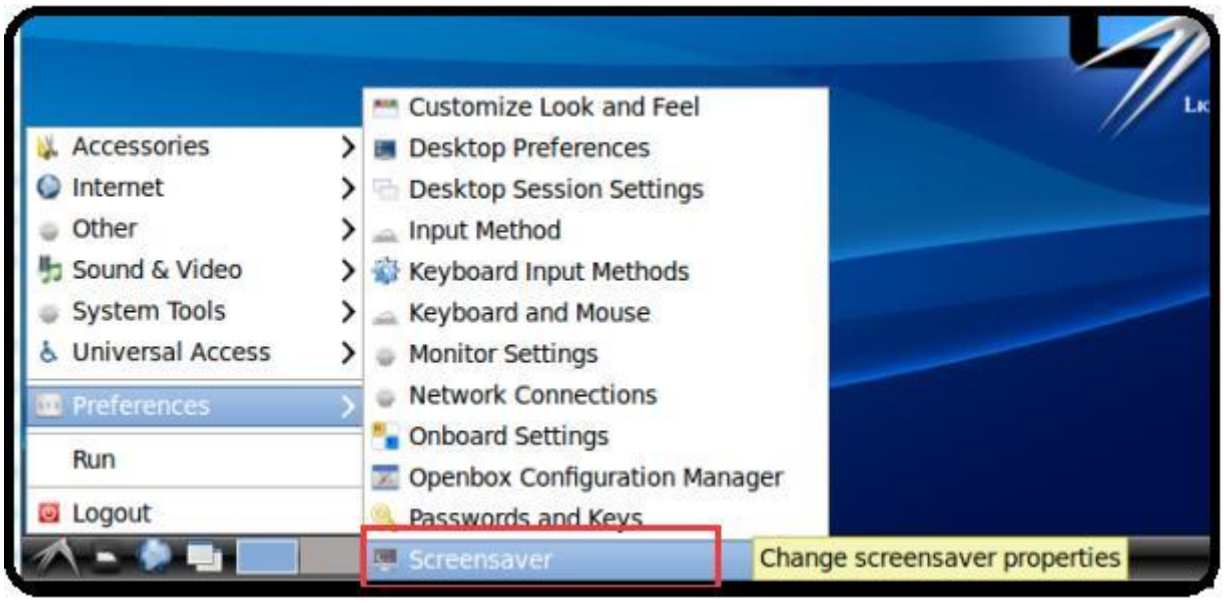

Figure 329: *Screensaver*

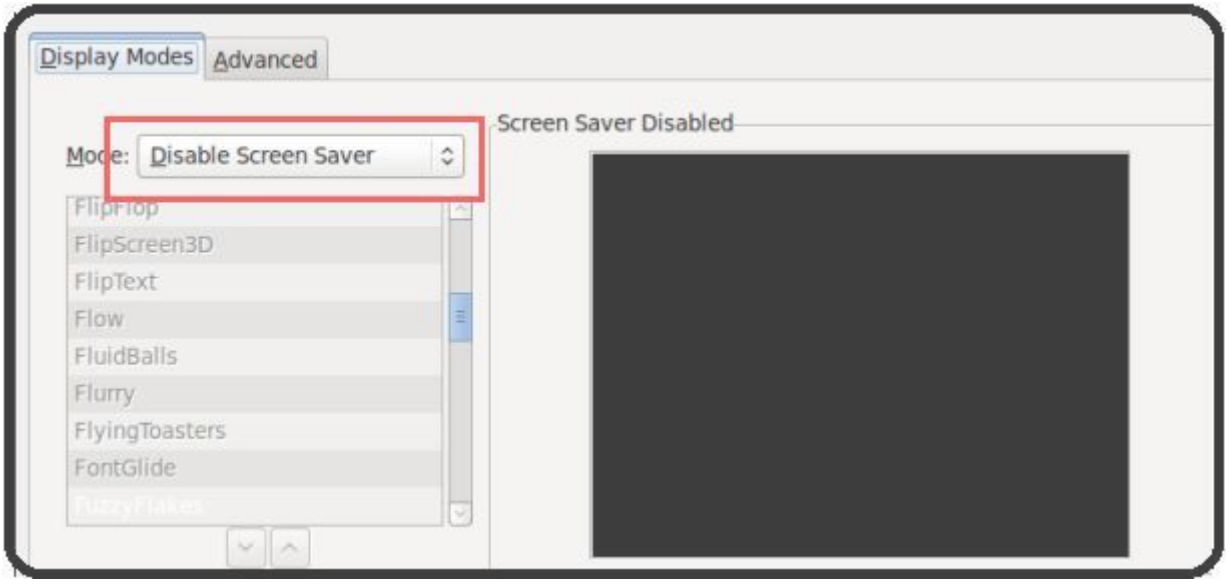

Figure 330: *Disable Screen Saver*

Autostart Application after Boot

We will autostart the Python `hello_world.py` app from Python.

- Change the mode for `hello_world.py` and copy it to */usr/local/bin*

```
$ sudo chmod a+x hello_world.py
$ sudo cp test.py /usr/local/bin/
```

- Put `hello_world.py` in LXDE autostart file, using this command:

```
Autostart File /home/chipsee/.config/lxsession/LXDE/autostart
Add follow to the end of autostart file.
```

```
@test.py
```

- Reboot the IPC to apply changes.

Disclaimer

This document is provided strictly for informational purposes. Its contents are subject to change without notice. Chipsee assumes no responsibility for any errors that may occur in this document. Furthermore, Chipsee reserves the right to alter the hardware, software, and/or specifications set forth herein at any time without prior notice and undertakes no obligation to update the information contained in this document.

While every effort has been made to ensure the accuracy of the information contained herein, this document is not guaranteed to be error-free. Further, it does not offer any warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document.

Despite our best efforts to maintain the accuracy of the information in this document, we assume no responsibility for errors or omissions, nor for damages resulting from the use of the information herein. Please note that Chipsee products are not authorized for use as critical components in life support devices or systems.

Technical Support

If you encounter any difficulties or have questions related to this document, we encourage you to refer to our other documentation for potential solutions. If you cannot find the solution you're looking for, feel free to contact us. Please email Chipsee Technical Support at **support@chipsee.com**, providing all relevant information. We value your queries and suggestions and are committed to providing you with the assistance you require.