Chipsee

Industrial PC

# Debian OS on AM335X User Manual

For AM335X Products

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

# Contents


1. Debian OS 3
  1.1. Preparation 4
    1.1.1. Hardware Requirements 4
    1.1.2. Software Requirements 4
  1.2. Getting Started and Tests 4
    1.2.1. DIP Switch Configuration 4
    1.2.2. Downloading Images 5
    1.2.3. Prebuilt Files Package 5
    1.2.4. How to make a bootable SD card 6
    1.2.5. How to flash Linux to eMMC 7
    1.2.6. Start Debian OS 8
    1.2.7. Tests 9
      1.2.7.1. Touch screen and buzzer test 9
      1.2.7.2. Audio test 10
      1.2.7.3. Serial test 10
      1.2.7.4. GPIO test 11
      1.2.7.5. Network 12
      1.2.7.6. Date and Time 12
      1.2.7.7. Backlight 13
      1.2.7.8. USB device 13
    1.2.8. Modify OS Start up Logo 13
  1.3. Debian OS debug 16
    1.3.1. View Debian system via the serial port 16
    1.3.2. Debug via NFS 17
  1.4. Debian App Development 18
    1.4.1. Preparation 19
    1.4.2. Example — Develop a `HelloWorld.java` Program 19
  1.5. Disclaimer 22
  1.6. Technical Support 22

# Debian OS

**Debian OS User Manual**

This manual provides users with a fast guide of Chipsee Industrial Computer (Abbreviated as IPC) about Debian OS development. Through this manual, users can quickly understand the hardware resources; users can debug Debian OS via serial and Internet.

| Revision | Date | Author | Description |
|---|---|---|---|
| V1.0 | 2021-12-09 | Randy | Initial Version |

**SUPPORTED BOARDS:**

> *CS80480T050 CS80600T080 CS10600T070 CS80480T070 CS10768T097*

**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.

- CS80480T050
- CS80600T080
- CS10600T070
- CS80480T070
- CS10768T097

System Features

| Feature | Comment |
| --- | --- |
| System | Debian 7.4 & Debian 8.4 |

# 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:
  > - Products with 5” display panel require 6V to 36V PSU
  > - Products with 7” to 10.1” display panel and larger require 6V to 42V PSU
- USB to serial cable for debugging Chipsee Industrial Embedded Computers (Chipsee IPC)
- TF Card to create a bootable storage for re-flashing the system. Use the prebuilt files link above to re-flash the system.

## Hardware Requirements

- Chipsee Industrial PC
- PSU according to the instructions above
- USB-to-serial or other serial cable for debugging
- TF Card (at least 4GB) and card reader
- USB A-A cable (used only if the hardware configured as OTG)
- Windows 7 PC

## Software Requirements

- Debian OS Prebuilt Files Package (from the link above)

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

# Getting Started and Tests

## DIP Switch Configuration

Set the boot DIP switch, as shown on the figure below, to boot the system from the external SD Card.

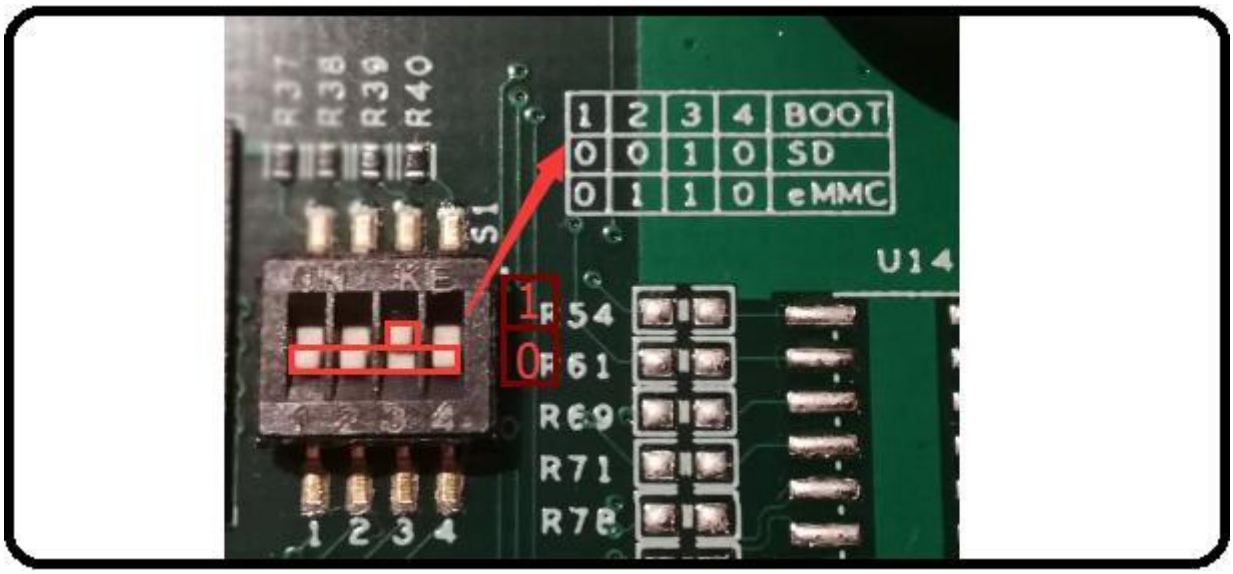


Figure 758: Boot Mode Setup

## Downloading Images

Chipsee IPC supports booting from an integrated eMMC or an external TF Card (also known as the micro SD card). Booting from the external TF Card allows flashing the system OS.

> **Note**
>
> The operator should use the prebuilt file we provided in the CD to test the hardware before re-flashing the system.

## 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 /Debian/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/imx6ulipc.dtb | TF Card boot dtb file |
| boot/u-boot.imx | TF Card boot bootloader |
| boot/zImage | TF Card boot kernel 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.gz | RootFS in target eMMC |

| Contents | Comment |
| --- | --- |
| emmc-flash/emmc/u-boot.imx | Bootloader in target eMMC |
| emmc-flash/emmc/zImage | Kernel file in target eMMC |
| emmc-flash/emmc/imx6ul-eisd.dtb | dtb file in target eMMC |
| emmc-flash/mkemmc.sh | Shell tools to download images |

Table 191 Prebuilt Files Package

**Note**

The default `zImage` and `imx6q-sabresd.dtb` files support *'keep the logo from uboot to kernel'* but do not support framebuffer. Chipsee provides `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`.

## How to make a bootable SD card

The Prebuilt Files Package has a shell tool that can help create a bootable SD card using a Linux platform (such as desktop PC or Virtual Machine running Ubuntu 14.04 distribution). Use the SD Card to download the bootable system image onto the Linux platform and follow the steps below to create a bootable SD 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-som-v3-csrrrrrtss-v3-debian-sd-yyyymmdd.tar.gz` to somewhere(such as $HOME).
5. **Extract the `prebuilt-som-v3-csrrrrrtss-v3-debian-sd-yyyymmdd.tar.gz`**

```
$ tar -xzvf prebuilt-som-v3-csrrrrrtss-v3-debian-sd-yyyymmdd.tar.gz
```

6. **Go to the folder**

```
$ cd ~/prebuilt-som-v3-csxxxxxtxx-v3-ezsdk-sd-yyyymmdd
```

7. **Use the following command to flash the Debian 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 DIP switch to uSD BOOT mode. (refer to DIP 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 the IPC and set the DIP switch to eMMC BOOT mode. (refer to DIP Switch Configuration above)

## How to flash Linux to eMMC

The Prebuilt Files Package has a shell tool that can help create a bootable SD card using a Linux platform (such as desktop PC or Virtual Machine running Ubuntu 14.04 distribution). Follow the steps below to create a bootable SD 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 file `prebuilt-som-v3-csrrrrrtss-v3-debian-emmc-yyyymmdd.tar.gz` to somewhere(such as $HOME).
5. **Extract the prebuilt file `prebuilt-som-v3-csrrrrrtss-v3-debian-emmc-yyyymmdd.tar.gz`**

```
$ tar -xzvf prebuilt-som-v3-csrrrrrtss-v3-debian-emmc-yyyymmdd.tar.gz
```

6. **Go to the folder** `prebuilt-som-v3-csrrrrrtss-v3-debian-emmc-yyyymmdd/`

```
$ cd ~/prebuilt-som-v3-csrrrrrtss-v3-debian-emmc-yyyymmdd
```

7. **Use the following command to flash the Debian 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 DIP switch to SD BOOT mode. (refer to DIP 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. Remove the SD card and Power OFF the IPC.
13. Set the DIP switch to eMMC BOOT mode (refer to DIP Switch Configuration above) and Power ON the IPC.

## Start Debian OS

The first time you start Debian OS on the industrial PC will take a little time. But after the first time, Debian OS will start quickly. When the Debian OS starts up, you will see the Chipsee Logo on the LCD screen. It is a successful start if you see the Debian OS desktop such as the one shown in the figure below:

Figure 759: Debian OS start-up screen

## Tests

### Touch screen and buzzer test

Click on the screen, the mouse arrow stays in a position that triggers the buzzer sounds, indicating that touch and buzzer work properly.
After working for some time, the resistive touch screen may not be accurate. The user must run a touch screen calibration test.
Firstly delete the file `/etc/pointercal.xinput` using the command below.

```
$ sudo rm /etc/pointercal.xinput
```

Click on the Preferences–>Calibrate Touchscreen app to recalibrate.
Reboot the system. You will see the calibrate app upon boot up before you access the system. Just calibrate, the result will be saved.

The buzzer will sound when the screen is touched, if you want to disable it, you can do this:

- **On capacitive touchscreen:**

```
# echo 0 > /sys/devices/ocp.3/44e0b000.i2c/i2c-0/0-0038/buzopen
```

- **On resistive touchscreen:**

```
# echo 1 > /sys/devices/ocp.3/44e0d000.tscadc/tsc/buzopen
```

**where:**

- 0 = disable
- 1 = enable

### Audio test

Start the terminal, then use the `mplayer` command to play an audio file.

```
# mplayer FILENAME  //such as: mplayer ~/Music/test.mp3
```

### Serial test

There are four serial ports on the Chipsee IPC: 2 X RS232 and 2 X RS485. The COM1(RS232) is used as the debug serial port. Users can communicate with the OS via COM1. Refer to the table below for the available serial device nodes.

| Ports | Device Node |
|---|---|
| COM1(RS232, Debug) | /dev/ttyO0 |
| COM2(RS232) | /dev/ttyO1 |
| COM3(RS485) | /dev/ttyO2 |
| COM4(RS485) | /dev/ttyO4 |

Table 192 Serial Ports Nodes on the System

If you want to use COM1 as a normal serial port, you can re-configure the port by following these steps:

- Open and edit the file `/etc/inttab` with any text editor.
- At the end of the file, edit this line `T0:23:respawn:/sbin/getty –L ttyO0 115200 vt102` to

```
# T0:23:respawn:/sbin/getty –L ttyO0 115200 vt102
```

- The code-block above, comments off the last line making it possible to use all the four serial ports as normal serial ports.
- You can verify the changes by running a serial test.

- **Run a serial test:**
  - Install **SecureCRT** or **Putty** software on a Windows 7 PC and use it to perform the serial port testing.
  - Connect COM2 on the industrial PC to Windows 7 PC. Set BAUD as 9600.
  - Send an echo through the terminal as shown in the code block below.

```
echo "This is a test" > /dev/ttyO1
```

  - You will see the string on the **SecureCRT** or **Putty** software running on the Windows 7 PC.
  - Change ttyO1 to ttyO2/ttyO4 to test RS485.

### GPIO test

There are (4) four input and (4) four output pins. LOW is 0V, HIGH is 5V.
The GPIO input terminals connect to the GPIO output terminals, respectively. IN1-4 corresponds to OUT1-4.
As a result, if you set the GPIO_OUT area, you will see the GPIO_IN region change as well.
You can control the LED light on the industrial PC by setting the LED **ON** or **OFF**.

| GPIO | GPIO In System |
|---|---|
| OUT1 | gpio49 |
| OUT2 | gpio50 |
| OUT3 | gpio51 |
| OUT4 | gpio52 |
| IN1 | gpio53 |
| IN2 | gpio54 |
| IN3 | gpio55 |
| IN4 | gpio56 |
| USER_LED | gpio19 |

Table 193 GPIO Nodes on the System

You can read and write the GPIO by following the steps below. For this example, we are going to use **gpio49(OUT1)**.

- **Use this command to export gpio.**

```
# echo 49 > /sys/class/gpio/export
```

- **Use this command to check if the directory `/sys/class/gpio/gpio49/` exist before writing to it**

```
# find /sys/class/gpio/gpio49/
```

- **Use this command to write gpio**

```
# echo 1 > /sys/class/gpio/gpio49/value
```

- **Use this command to read gpio**

```
# cat /sys/class/gpio/gpio49/value
```

### Network

**Use this command to view the network information on the industrial PC.**

```
# ifconfig –a
```

### Date and Time

- **Check the system time**

```
# date
```

- **Set the system time**

```
# date –s "2014-03-15 10:30:30"
```

- **Check RTC**

```
# hwclock
```

- **Write RTC**

```
# hwclock -w
```

- **Modify the time zone to a different timezone, such as China**

```
# ln -sf /usr/share/zoneinfo/Asia/Hong_Kong /etc/localtime
```

### Backlight

Modify this file `/sys/class/backlight/backlight` to adjust the screen brightness.
Brightness ranges from 0 to 100 where 0 means no backlight, and 100 is the MAX brightness value.
For example, you can adjust the screen brightness using this command:

```
# echo 50 > /sys/class/backlight/backlight.10/Brightness
```

### USB device

The Debian OS supports USB-WiFi (RTL8723). If you want to get USB-Wifi module to work, you need to edit file: `/etc/network/interfaces` as shown in the code-block below.

```
# Wifi Example
allow-hotplug wlanX
auto wlanX
iface wlanX inet dhcp
wpa-ssid "Chipsee"   # Router' name
wpa-psk "1234567890" # Passwd

**wlanX** usual should be ``wlan0``, you can use the command ``ifconfig –a`` to
make sure of that.
```

## Modify OS Start up Logo

Chipsee® provides a software to change the OS boot up logo. The software `ChipSee_LOGO_MOD_EN.exe` is provided on the CD for a product.
To change the logo, follow these steps:

1. **Open the software:** `Chipsee_LOGO_MOD_EN.exe` **in Windows**

Figure 760: Chipsee OS Boot-up Logo Modify Software

2. **Click the first Browse button. Select the picture file you want to use as the logo.**

Figure 761: Choose the Logo you want

3. **Click the second Browse button. Select the `u-boot.img` file you want to use.**

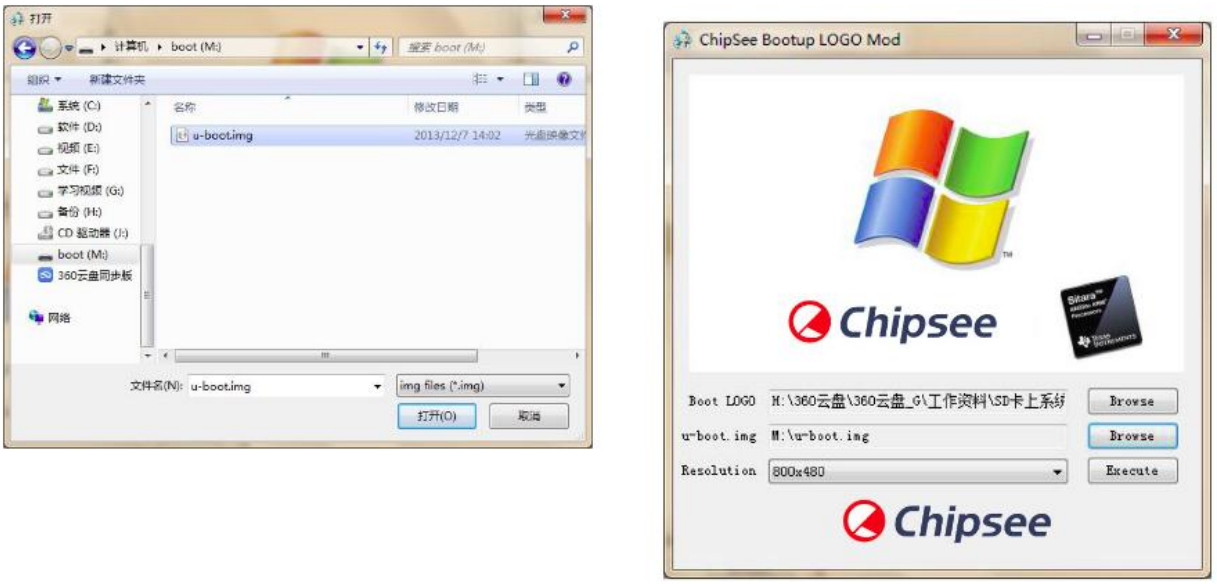


Figure 762: Choose the u-boot.img file

4. **Choose the correct resolution for your product, then click Execute.**

Figure 763: Change the Logo successful

5. Insert the SD card into the IPC. Power ON the IPC and the Logo will be replaced.

# Debian OS debug

In this section, we will discover how to view the Debian system via the serial port on a Windows 7 PC.
Also, we will discover how to debug using NFS on a Ubuntu Linux PC.

## View Debian system via the serial port

Install the **SecureCRT** or **Putty** software on a Windows 7 PC to view the Debian system via the serial ports.
Follow these steps to view Debian system via the serial port:

- Connect COM1 on the industrial PC board to Windows 7 PC.

- Open the **SecureCRT** or **Putty** software on the Windows 7 PC.
- Power ON the industrial PC. You will see the serial output information as shown on the figure below.
- When the system is fully booted, you can communicate with it by logging in with these details: user= debian or root and password= chipsee or root.

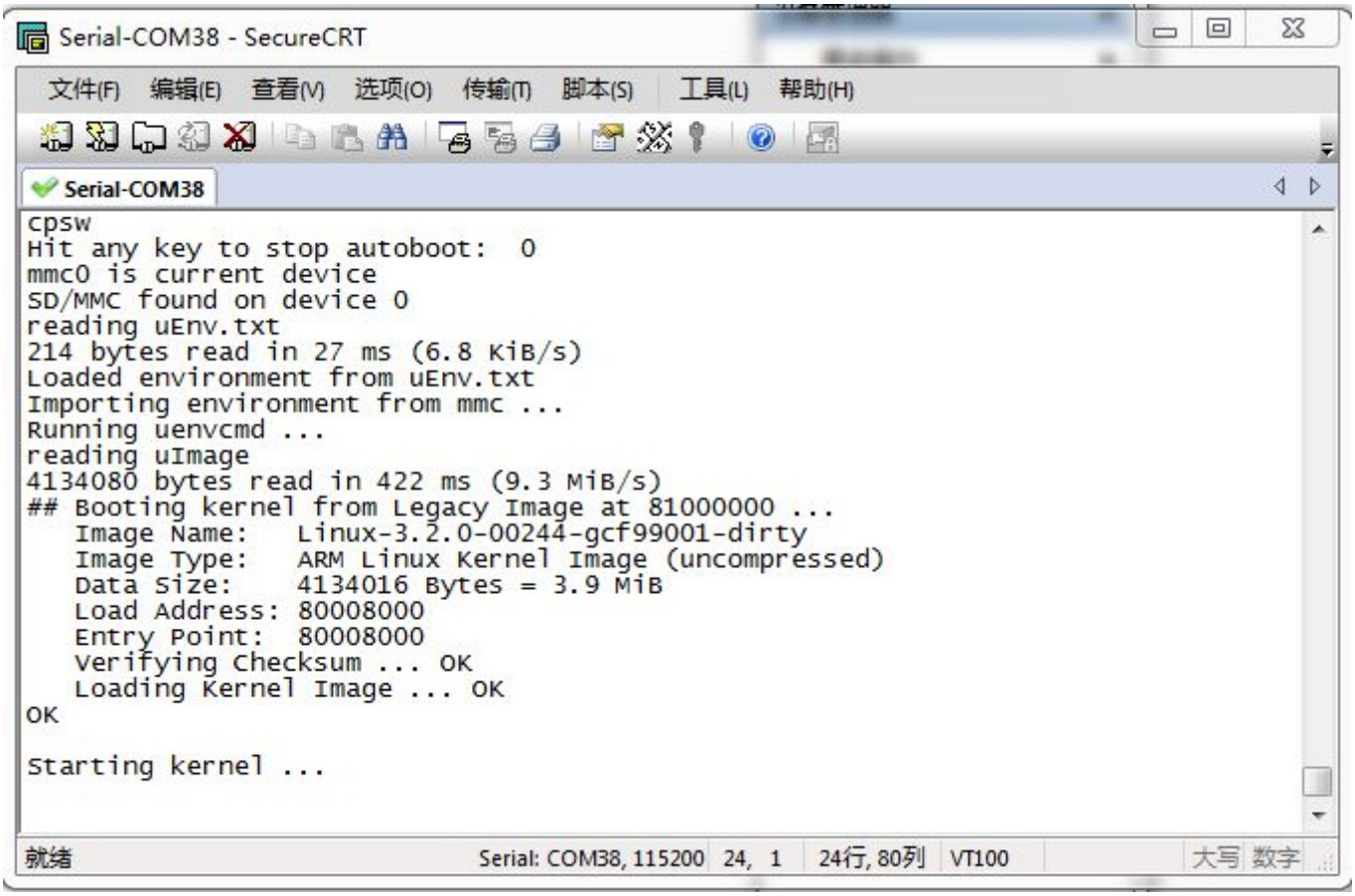


Figure 764: Serial output information

## Debug via NFS

1. **Install NFS on Ubuntu Linux PC.**

```
$ sudo apt-get install nfs-kernel-server
```

2. **Configure the file `/etc/exports`, by adding this line at the end of the file.**

```
/qtprojects *(rw, sync, insecure, no_subtree_check)
```

> **Note**
>
> - `/qtprojects`: the shared folder in Ubuntu system
> - `*`: allows all other PC to get access to this system
> - `rw`: means this folder can be read and write by NFS client

> - `sync` : synchronous write memory and hard disk
> - `insecure` : sent message through the port above 1024
> - `no_subtree_check` : no check the parent directory permissions

3. **Restart NFS service.**

```
$ sudo /etc/init.d/portmap restart
$ sudo /etc/init.d/nfs-kernel-server restart
```

4. **Test**

```
$ showmount -e
```

or mount the shared folder to `/mnt` :

```
$ sudo mount -t nfs –o nolock localhost:/qtprojects /mnt
```

Use the command `df` to check out the result, then umount.

```
$ df -h
$ sudo umount /mnt
```

5. **Mount NFS on the industrial PC running Debian OS.**

Create the `nfsdir` directory

```
# mkdir /nfsdir
```

Mount the folder `/qtprojects` on the Ubuntu Linux PC to `/nfsdir` on the industrial PC.

```
# mount –t nfs :/qtprojects /nfsdir
```

If you have an executable program like **SerialTest** under folder `/qtprojects` , you can run it directly on the industrial PC.

```
# /nfsdir/SerialTest
```

# Debian App Development

In this section, we will set up the Java environment and show you how to a make simple Java application.

## Preparation

**Download and install a Java jdk.**

```
# sudo apt-get install openjdk-6-jdk
```

## Example — Develop a `HelloWorld.java` Program

1. **Open and edit, with any text editor, a simple java `HelloWorld.java` program.**

```
 1 import java.awt.Color;
 2 import java.awt.Font;
 3 import java.awt.Toolkit;
 4 import javax.swing.JFrame;
 5 import javax.swing.JTextField;
 6
 7 public class HelloWorld extends JFrame{
 8     public HelloWorld(){
 9          JTextField text = new JTextField("Hello, world!");
10          text.setFont(new Font("Times New Roman",Font.BOLD,60));
11          text.setForeground(Color.BLACK);
12          this.getContentPane().add(text);
13      }
14     public static void main(String argv[]){
15          HelloWorld win = new HelloWorld();
16          Toolkit tk  = Toolkit.getDefaultToolkit();
17          int winWidth = 512;
18          int winHeight = 300;
19          int Width  = tk.getScreenSize().width;
20          int Height = tk.getScreenSize().height;
21          win.setSize(winWidth, winHeight);
22          win.setLocation((Width-winWidth)/2, (Height-winHeight)/2);
23          win.setVisible(true);
24          win.setDefaultCloseOperation(EXIT_ON_CLOSE);
25      }
26 }
```

2. **Compile the source.**

```
# javac HelloWorld.java
```

**This will be very slow in Debian OS, we suggest do it in your PC, you need install jdk-1.6 first.**

3. **Run the program.**

```
# java HelloWorld
```

You will see something similar as the figure below.

Figure 765: Hello World Program

4. **Adding the Java program to Quick Start.**

- **Make a directory**

```
# mkdir /usr/lib/java/
```

- **Copy `HelloWorld.class` to `/usr/lib/java/` directory**

```
# cp HelloWorld.class /usr/lib/java/
```

- **Open and edit script `/usr/bin/HelloWorld.sh` with any text editor.**

```
1 #!/bin/bash
2 cd /usr/lib/java/
3 java HelloWorld
```

- **Change the permissions of the script.**

```
# sudo chmod a+x HelloWorld.sh
```

- **Edit file `/usr/share/applications/javatest.desktop`.**

```
[Desktop Entry]
Name=HelloWorld
Comment=Simple test for Java
Exec=/usr/bin/HelloWorld.sh
Icon=/usr/share/pixmaps/chipsee.png
```

```
Terminal=false
Type=Application
Categories=GTK;Utility;GNOME;
```

- **This is the result:**

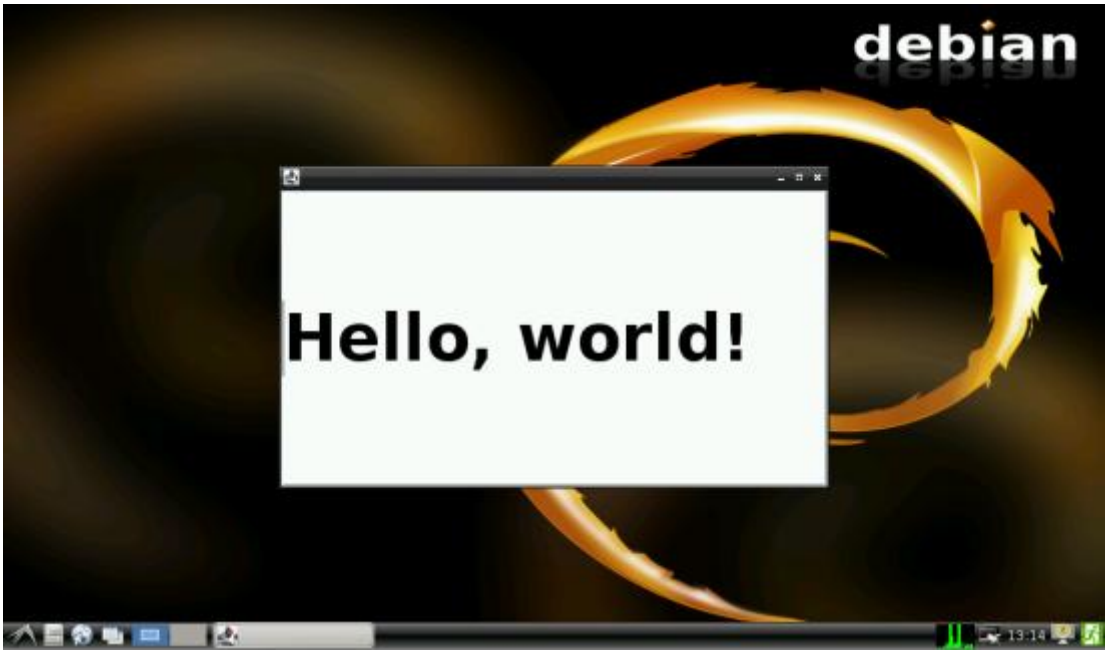


Figure 766: Hello World Program

5. **Auto-Launch Java app**

Add script `89javatest` in directory `/etc/X11/Xsession.d/`.

```
#!/bin/bash
cd /usr/lib/java/
java HelloWorld
```

Now when you Reboot the system, the `HelloWorld` app will automatically launch.

# 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.