



**User Manual**

# **UBC-200 RISC Compact Box**

**Freescale ARM® Cortex™ A9  
i.MX6 Processor based**

**ADVANTECH**

*Enabling an Intelligent Planet*

---

## Copyright

The documentation and the software included with this product are copyrighted 2014 by Advantech Co., Ltd. All rights are reserved. Advantech Co., Ltd. reserves the right to make improvements in the products described in this manual at any time without notice. No part of this manual may be reproduced, copied, translated or transmitted in any form or by any means without the prior written permission of Advantech Co., Ltd. Information provided in this manual is intended to be accurate and reliable. However, Advantech Co., Ltd. assumes no responsibility for its use, nor for any infringements of the rights of third parties, which may result from its use.

## Acknowledgements

ARM® is trademarks of ARM Corporation.

Freescale® is trademarks of Freescale Corporation.

Microsoft Windows are registered trademarks of Microsoft Corp.

All other product names or trademarks are properties of their respective owners.

## Product Warranty (2 years)

Advantech warrants to you, the original purchaser, that each of its products will be free from defects in materials and workmanship for two years from the date of purchase.

This warranty does not apply to any products which have been repaired or altered by persons other than repair personnel authorized by Advantech, or which have been subject to misuse, abuse, accident or improper installation. Advantech assumes no liability under the terms of this warranty as a consequence of such events.

Because of Advantech's high quality-control standards and rigorous testing, most of our customers never need to use our repair service. If an Advantech product is defective, it will be repaired or replaced at no charge during the warranty period. For out-of-warranty repairs, you will be billed according to the cost of replacement materials, service time and freight. Please consult your dealer for more details.

If you think you have a defective product, follow these steps:

1. Collect all the information about the problem encountered. (For example, CPU speed, Advantech products used, other hardware and software used, etc.) Note anything abnormal and list any onscreen messages you get when the problem occurs.
2. Call your dealer and describe the problem. Please have your manual, product, and any helpful information readily available.
3. If your product is diagnosed as defective, obtain an RMA (return merchandise authorization) number from your dealer. This allows us to process your return more quickly.
4. Carefully pack the defective product, a fully-completed Repair and Replacement Order Card and a photocopy proof of purchase date (such as your sales receipt) in a shippable container. A product returned without proof of the purchase date is not eligible for warranty service.
5. Write the RMA number visibly on the outside of the package and ship it prepaid to your dealer.

# Declaration of Conformity

## FCC Class B

Note: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

## Technical Support and Assistance

1. Visit the Advantech website at <http://support.advantech.com> where you can find the latest information about the product.
2. Contact your distributor, sales representative, or Advantech's customer service center for technical support if you need additional assistance. Please have the following information ready before you call:
  - Product name and serial number
  - Description of your peripheral attachments
  - Description of your software (operating system, version, application software, etc.)
  - A complete description of the problem
  - The exact wording of any error messages

## Warnings, Cautions and Notes

**Warning!** Warnings indicate conditions, which if not observed, can cause personal injury!



**Caution!** Cautions are included to help you avoid damaging hardware or losing data. e.g.



*There is a danger of a new battery exploding if it is incorrectly installed. Do not attempt to recharge, force open, or heat the battery. Replace the battery only with the same or equivalent type recommended by the manufacturer. Discard used batteries according to the manufacturer's instructions.*

**Note!** Notes provide optional additional information.



## Packing List

Before setting up the system, check that the items listed below are included and in good condition. If any item does not accord with the table, please contact your dealer immediately.

- System
  - UBC-200 (P/N: UBC-200CD-MDA1E;UBC-200CQ-MEA1E)
- Accessories (Enclosed)
  - UBC-200 Wall Mount Bracket (P/N: 1960062939N001)
- Accessories (Optional)
  - DIN RAIL (P/N: 1960015198T011)
  - SQFlash SD Card SLC 2G, 2CH(-40~85°C) (P/N: SQF-ISDS1-2G-86E)
  - 802.11 b/g/n,AR9287,2T2R,Full size Mini PCIe (P/N: EWM-W142F01E)
  - Cellular, HSUPA/WCDMA/GPRS,Full Mini PCIe (P/N: EWM-C106FT01E )
  - Antenna Coaxial Cable SMA/MHF 10cm (P/N: 1750005583) (3G Cable)
  - Antenna SMA D(M) JACK to I-PEX MHF CABLE L:100mm (P/N: 1750006233 ) (3G Antenna)
  - WiFi RP-SMA short SMA Jack(9.5mm) to U.FL\_100mm(P/N: 1750007050-01) (WIFI Cable)
  - EMI Antenna 2DBI 2.4GHz SMA CONN for ARK-3384(P/N: 1750000318) (WIFI Antenna)
  - ADAPTER 100-240V 36W 12V 3A (P/N: 1757003553)
  - DC-Jack/Plug-in cable (P/N: 1700017968)

# Safety Instructions

1. Read these safety instructions carefully.
2. Keep this User Manual for later reference.
3. Disconnect this equipment from any AC outlet before cleaning. Use a damp cloth. Do not use liquid or spray detergents for cleaning.
4. For plug-in equipment, the power outlet socket must be located near the equipment and must be easily accessible.
5. Keep this equipment away from humidity.
6. Put this equipment on a reliable surface during installation. Dropping it or letting it fall may cause damage.
7. The openings on the enclosure are for air convection. Protect the equipment from overheating. **DO NOT COVER THE OPENINGS.**
8. Make sure the voltage of the power source is correct before connecting the equipment to the power outlet.
9. Position the power cord so that people cannot step on it. Do not place anything over the power cord.
10. All cautions and warnings on the equipment should be noted.
11. If the equipment is not used for a long time, disconnect it from the power source to avoid damage by transient overvoltage.
12. Never pour any liquid into an opening. This may cause fire or electrical shock.
13. Never open the equipment. For safety reasons, the equipment should be opened only by qualified service personnel.
14. If one of the following situations arises, get the equipment checked by service personnel:
  - The power cord or plug is damaged.
  - Liquid has penetrated into the equipment.
  - The equipment has been exposed to moisture.
  - The equipment does not work well, or you cannot get it to work according to the user's manual.
  - The equipment has been dropped and damaged.
  - The equipment has obvious signs of breakage.

**DISCLAIMER:** This set of instructions is given according to IEC 704-1. Advantech disclaims all responsibility for the accuracy of any statements contained herein.



# Contents

<b>Chapter 1</b>	<b>Product Overview .....</b>	<b>1</b>
1.1	Introduction .....	2
1.2	Features .....	3
1.3	Hardware Specifications .....	3
	Table 1.1: UBC-200 Specification .....	3
1.4	Block Diagram .....	4
	Figure 1.1 UBC-200 Block Diagram .....	4
<b>Chapter 2</b>	<b>UBC-200 H/W installation.....</b>	<b>5</b>
2.1	UBC-200 H/W Installation .....	6
2.1.1	I/O Description .....	6
	Figure 2.1 UBC-200 I/O Connector .....	6
	Figure 2.2 DC-in Connector .....	6
2.1.2	Wall Mount Bracket Assembly .....	7
	Figure 2.3 Wall Mount Bracket Assembly .....	7
2.1.3	DIN Rail Assembly .....	7
	Figure 2.4 DIN Rail Assembly .....	7
2.2	UBC-200 Mechanical .....	8
2.2.1	Top Cover .....	8
	Figure 2.5 UBC-200 Top Cover .....	8
2.2.2	Bottom Chassis .....	8
	Figure 2.6 UBC-200 Bottom Chassis .....	8
2.2.3	Wall Mount Bracket .....	9
	Figure 2.7 UBC-200 Wall Mount Bracket .....	9
2.3	Quick Start .....	9
2.3.1	Building Connection of Linux Console .....	9
2.4	Test Tools .....	14
2.4.1	eMMC Test .....	14
2.4.2	USB Test .....	14
2.4.3	SD Test .....	15
2.4.4	HDMI Test .....	16
	Figure 2.8 HDMI Test .....	16
2.4.5	Mini PCIe (3 G and Wifi) Test .....	16
2.4.6	OpenGL Test .....	17
2.4.7	LAN Test .....	18
2.4.8	Watchdog Timer Test .....	19
2.4.9	Photo Demo Test .....	20
	Figure 2.9 Photo Demo .....	20
<b>Chapter 3</b>	<b>Software Functionality .....</b>	<b>21</b>
3.1	Introduction .....	22
3.2	Package Content .....	22
3.2.1	Source Code Package .....	22
3.3	Set up Build Environment .....	25
3.3.1	setenv.sh .....	25
3.4	Build Instructions .....	26
3.4.1	Build u-boot Image .....	26
3.4.2	Build Linux Kernel Image .....	26
3.4.3	Build Log .....	26
3.5	Source Code Modification .....	27
3.5.1	Add a Driver to Kernel by menuconfig .....	27

	Figure 3.1 Linux Kernel Configuration .....	27
	Figure 3.2 Selecting Seiko Instruments S-35390A .....	28
3.5.2	Change UBC-200 Boot Logo .....	29
3.6	Create a Linux System Boot Media .....	29
3.6.1	Create a Linux System SD Card .....	29
3.6.2	Boot from Onboard Flash .....	29
3.7	Telnet Function .....	30
3.8	Linux Software AP and Testing on UBC-200 .....	34
3.8.1	“Hello World!” Application and Execution .....	34
3.8.2	Watchdog Timer Sample Code .....	35
3.8.3	Network Setup .....	36
3.8.4	Storage (SATA /eMMC/SD Card) .....	36
3.8.5	3G Sample Code .....	37
<b>Chapter 4</b>	<b>System Recovery .....</b>	<b>39</b>
4.1	System Recovery .....	40
<b>Chapter 5</b>	<b>Advantech Services .....</b>	<b>41</b>
5.1	RISC Design-in Services .....	42
5.2	Contact Information .....	44
5.3	Global Service Policy .....	45
5.3.1	Warranty Policy .....	45
5.3.2	Repair Process .....	45



# Chapter 1

## Product Overview

This chapter briefly introduces UBC-200 platform.

---

## 1.1 Introduction

In order to respond to multiple RISC platform requirements and the huge RISC market opportunities, a more efficient and low risk solution for the growing IoT market has been released - PC UBC-200. It's a RISC Box Computer solution with Freescale i.MX6 processor in an ARM® Cortex™ A9 architecture, with a complete 64-bit data bus, and Dual/Quad Core 1GHz speed SoC engine. UBC-200 is a high performance compact size micro computer which is ready-to-run, with simple I/O specified for the fulfillment of the features needed in the IoT market. With UBC-200 RISC Compact Box Computer, you get customized I/O interfaces and complete hardware and software solutions for IoT applications.

Advantech has been involved in RISC/ARM development for years and first started RISC-on-Module (ROM) product development in 2010, and after that we started developing ARM-based box PCs in 2012. In order to strengthen design-in services and speed up workflow processes, we provide you with several hardware and software design utilities which help you to easily check your design in detail. Advantech also brings you the necessary tools such as S/W driver list and function test plans that will help reduce your design effort and speed up application development.

UBC-200 is a complete hardware and software solution for developing your own innovation to conquer the IoT market. It includes system level software that developers need like BSP, SUSI API and multiple OS support. With UBC-200, you can develop your own software based on this highly-integrated HW platform and create more add-on value for your end product. With global customer service, trouble-shooting support and the deep market experience of Advantech, you can easily lead the market with your own innovation. Advantech also brings you optional accessories such as WIFI and 3G modules, antennas and extended storage such as SD cards or SATA drives for project evaluation and application development. UBC-200 has already integrated complete certified functions in Linux kernel 3.0.35 and Android 4.2. The test utilities and H/W development tools offered will facilitate the whole process of project development and make your creativity become real, easy and risk-free.

## 1.2 Features

UBC-200 adopts Freescale i.MX6 Dual Core Processor - in an ARM® Cortex™ A9 architecture as its SoC solution. It supports both Dual Core and Quad Core as its embedded SoC. UBC-200 is heatsink-less, compact, and reliable with great power management, and is a suitable solution for the following applications:

- Signage
- IoT
- Industrial data collection

Main features of Freescale i.MX6 processors:

- ARM Cortex™-A9 high performance processor, dual/quad core 1GHz
- Supports OpenGL ES 2.0 and OpenVG™ 1.1 hardware accelerators, full HD 1080p video codec
- Supports 3 IPU for 2D/3D/Image
- Freescale Smart Speed™ Technology supports low power consumption
- Capabilities of I/O expansion: USB Host, Gigabit Ethernet, SD, Mini PCI-E with USB/PCIe signal inside and HDMI support
- Supports Linux 3.0.35
- Supports Embedded Android 4.2
- Supports working temperatures 0 ~ 60°C (operating temperature)

## 1.3 Hardware Specifications

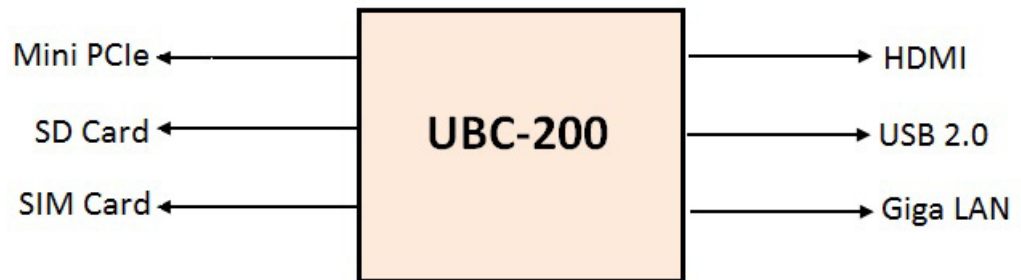
**Table 1.1: UBC-200 Specification**

Item	Description
<b>Kernel</b>	
CPU	Freescale i.MX6 Dual/Quad 1GHz (ARM Cortex A9)
2D/3D Accelerators	Supports OpenGL ES 2.0 and OpenVG™ 1.1 hardware accelerators
System RAM	1 GB for Dual Core; 2 GB for Quad Core
Onboard Flash	4 GB
RTC	Yes
Watchdog Timer	Yes
Reset	H/W reset & S/W reset
<b>I/O</b>	
Ethernet LAN	1 x 10/100/1000 Gigabit Ethernet (RJ-45)
USB Port	2 x USB 2.0 Type A, 1x USB 2.0 Pin Header
SD/MMC	1 x SD/MMC card slot
Mini PCI-E	1 x (Control by USB & PCIe interface)
SIM Card slot	1 x
<b>Multimedia</b>	
Graphic Chip	CPU internal LCD controller
HDMI	1 x
Brightness/Backlight Control	Yes

**Table 1.1: UBC-200 Specification**

Audio	HDMI
<b>Power</b>	
DC-input	9 ~ 24 V
Power Consumption	Normal Run: 2.5 W Full Run: 3.5 W
Power Control	1 x Reset button
Power Management	- Standard mode - Sleep mode (Controlled by Software)
<b>Mechanical and Environmental</b>	
Dimension	108 x 79 x 30 mm
Weight	330g
Operation Temperature	0 ~ 60° C (32 ~ 140° F)
Operating Humidity	5% ~ 95% Relative Humidity, non-condensing
Vibration	3.5 G, 1000 times
<b>Others</b>	
RoHS	Yes
Certification	CE/FCC Class B
O.S	Embedded Linux 3.0.35 Embedded Android 4.2

## 1.4 Block Diagram



**Figure 1.1 UBC-200 Block Diagram**

# Chapter **2**

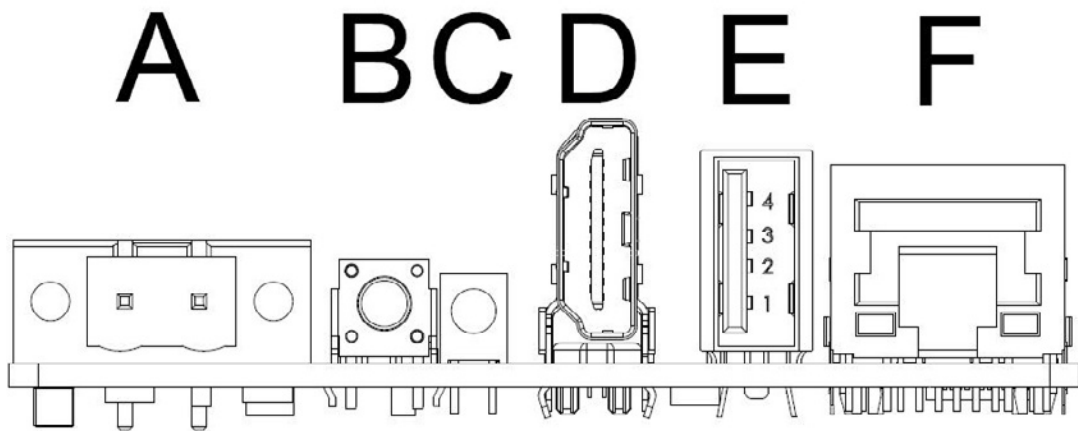
## UBC-200 H/W installation

This chapter introduces the startup procedures of the UBC-200 hardware, including assembly and device integration. It also includes mechanical drawings.

Be sure to read all safety precautions before you begin installation procedure.

## 2.1 UBC-200 H/W Installation

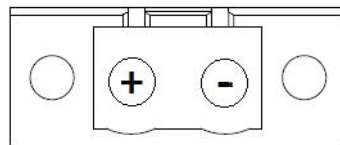
### 2.1.1 I/O Description



**Figure 2.1 UBC-200 I/O Connector**

#### A. DC-in Connector

Users can connect DC power cord to this connector to power on UBC-200. The pin define is shown below.



**Figure 2.2 DC-in Connector**

#### B. RESET Button

Press this button to reset UBC-200. System will reboot once the reset process is triggered.

#### C. Power LED

This LED is to identify the system status. When UBC-200 is powered on, the power LED lights up correspondingly.

#### D. HDMI Connector

UBC-200 supports HDMI output resolution up to 1920x1080 pixels.

#### E. USB 2.0 Connector

UBC-200 supports devices follow by USB 2.0 standard.

#### F. RJ45 Connector

UBC-200 supports 10/100/1000 Mbps Ethernet, users can simply connect LAN cable to this connector.

### 2.1.2 Wall Mount Bracket Assembly

There are 2 wall mount brackets in UBC-200 packing list. Users can assemble these brackets UBC-200's metal chassis as shown below:

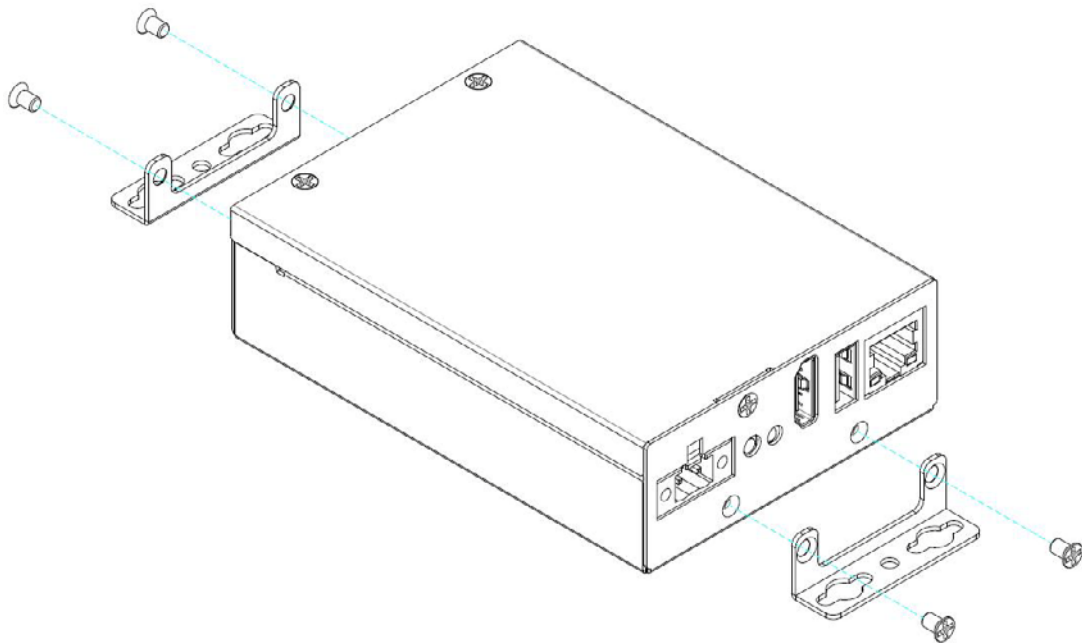


Figure 2.3 Wall Mount Bracket Assembly

### 2.1.3 DIN Rail Assembly

You can optionally purchase DIN rail (P/N: 1960015198T011) for holding UBC-200. Please contact an Advantech salesperson for more details. Users can assemble a DIN rail to UBC-200's metal chassis as shown below:

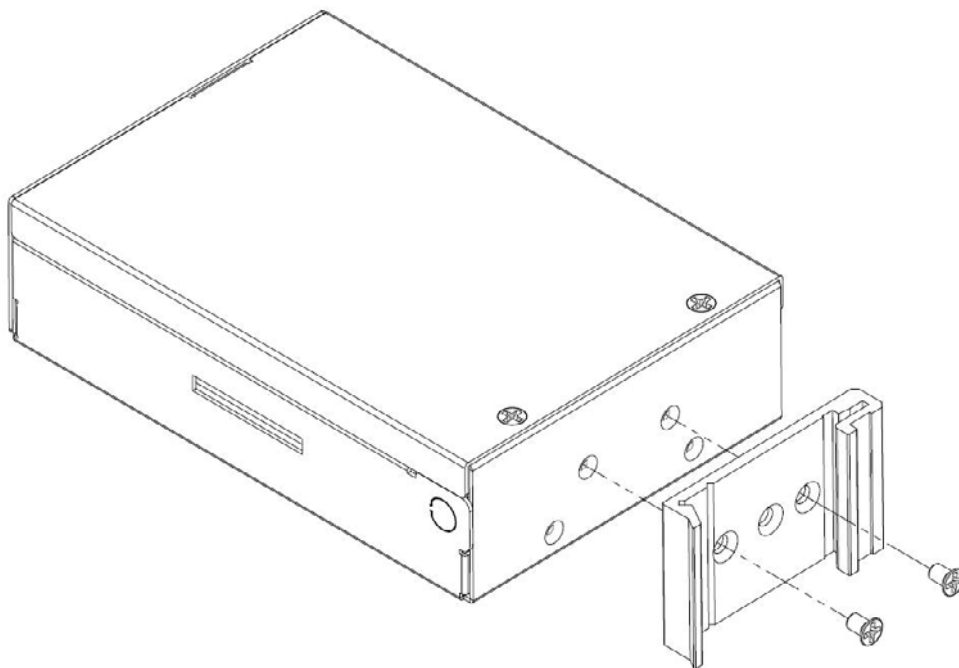


Figure 2.4 DIN Rail Assembly

## 2.2 UBC-200 Mechanical

### 2.2.1 Top Cover

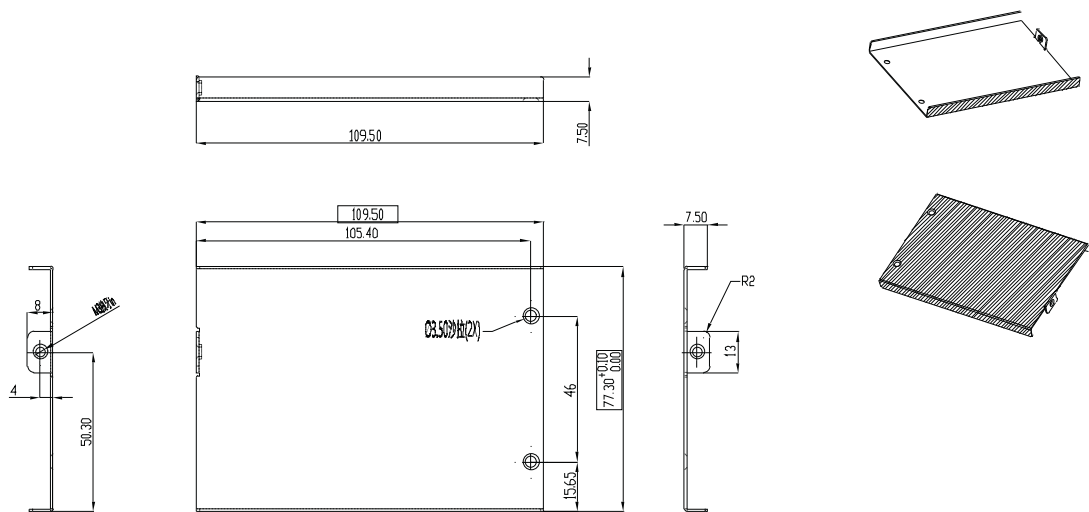


Figure 2.5 UBC-200 Top Cover

### 2.2.2 Bottom Chassis

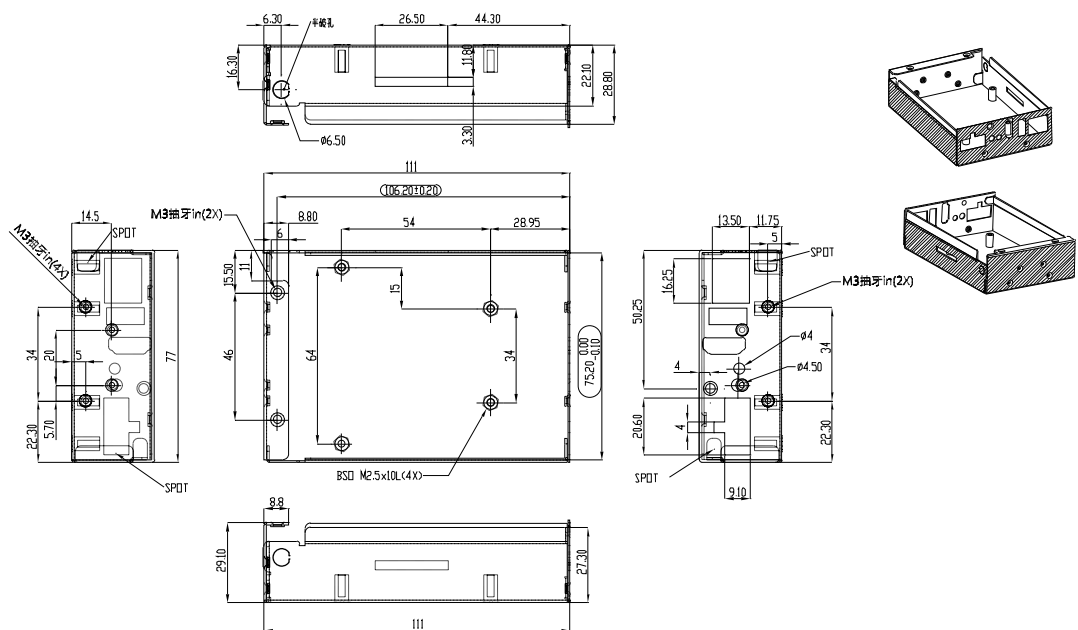


Figure 2.6 UBC-200 Bottom Chassis



## 2.2.3 Wall Mount Bracket

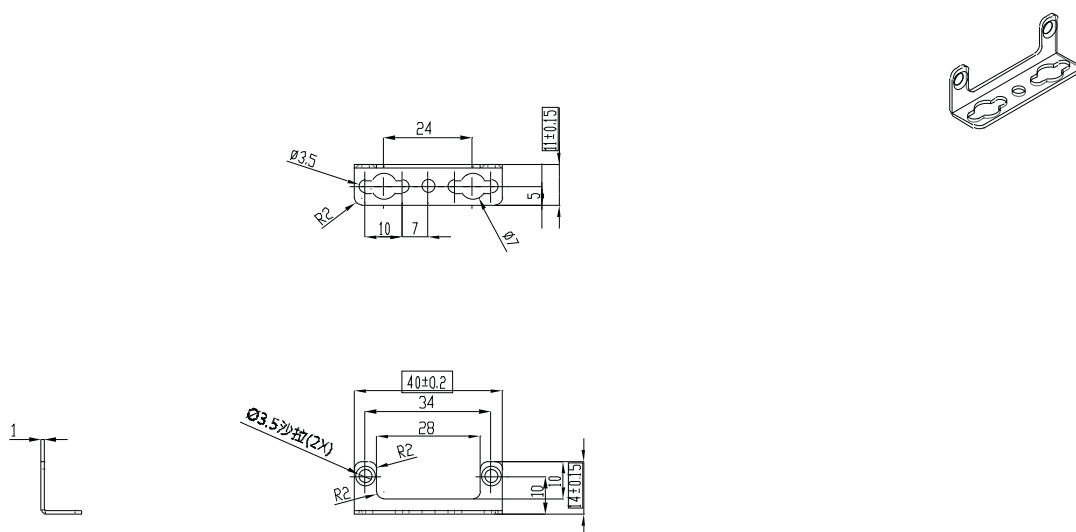


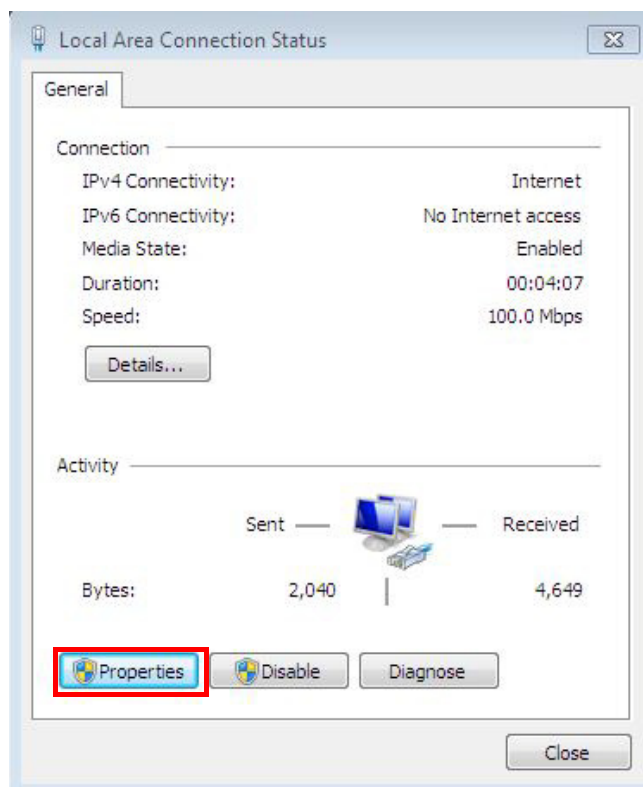
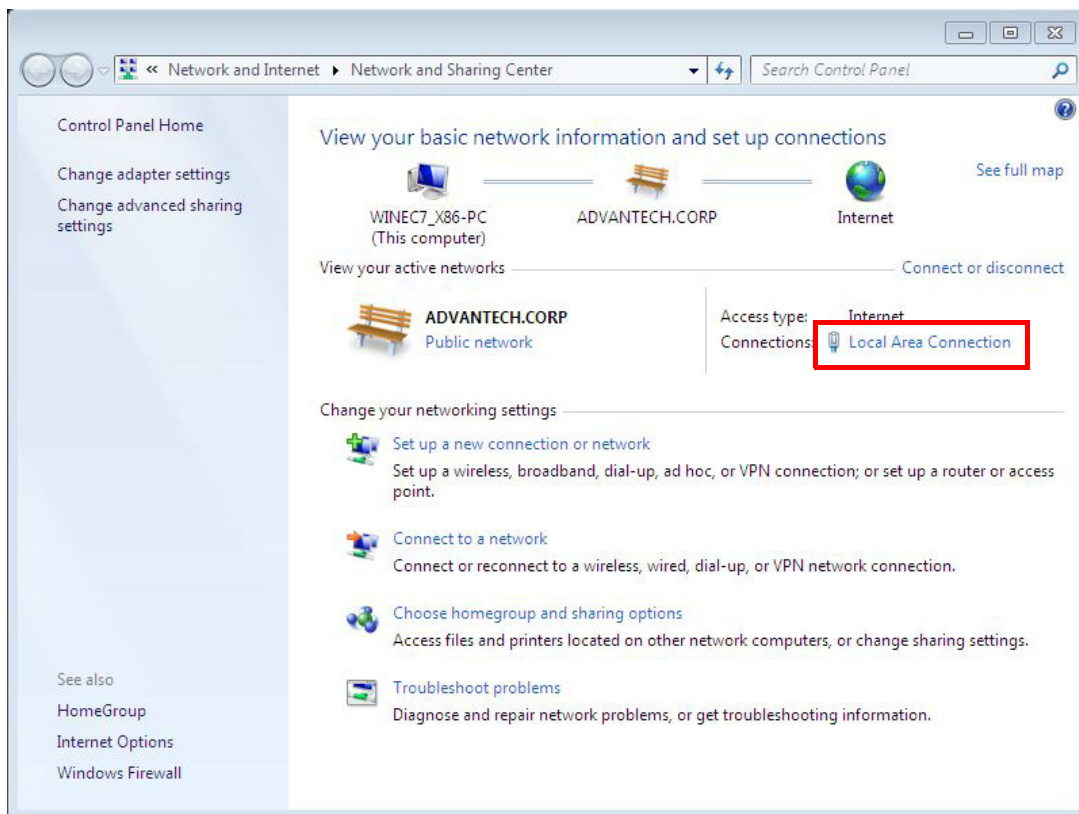
Figure 2.7 UBC-200 Wall Mount Bracket

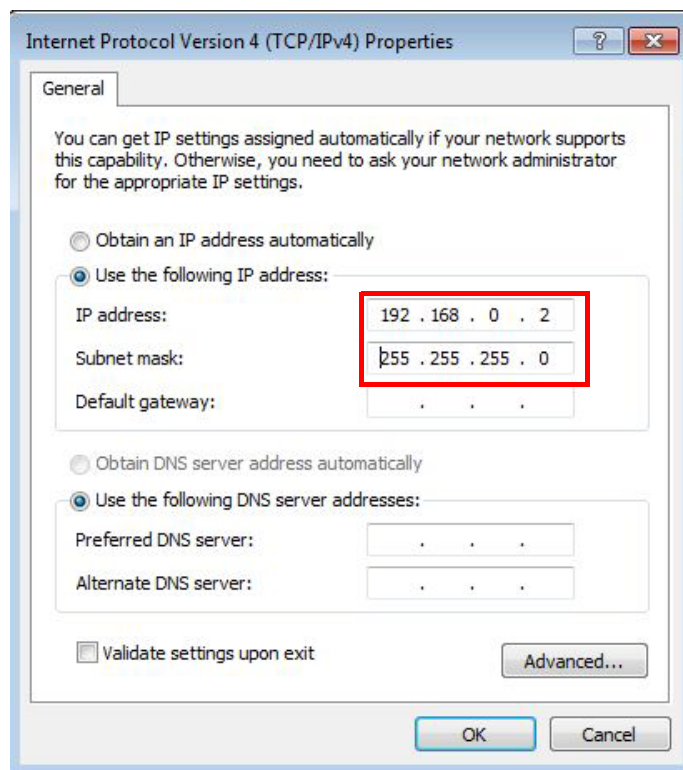
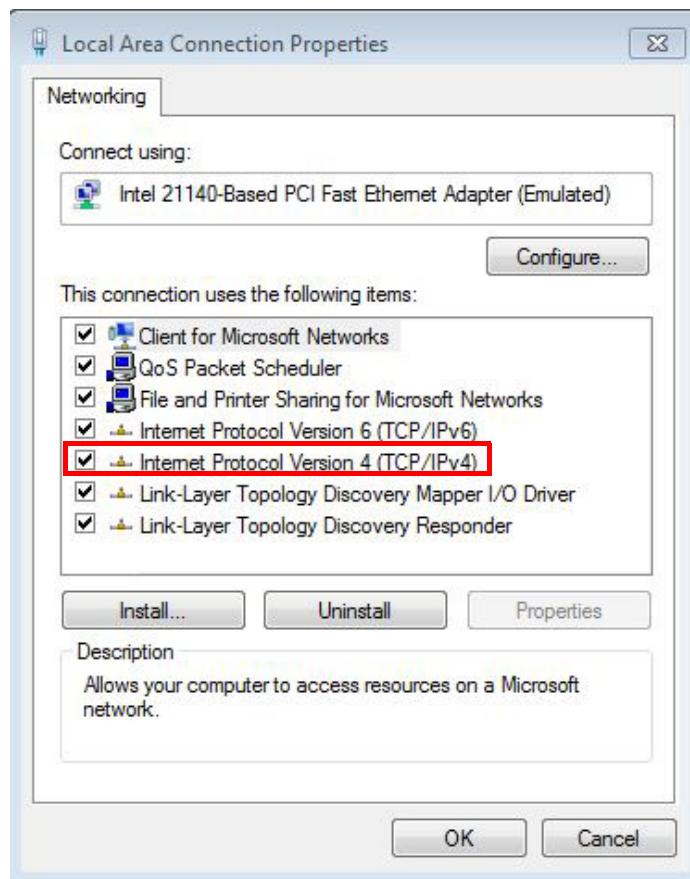
## 2.3 Quick Start

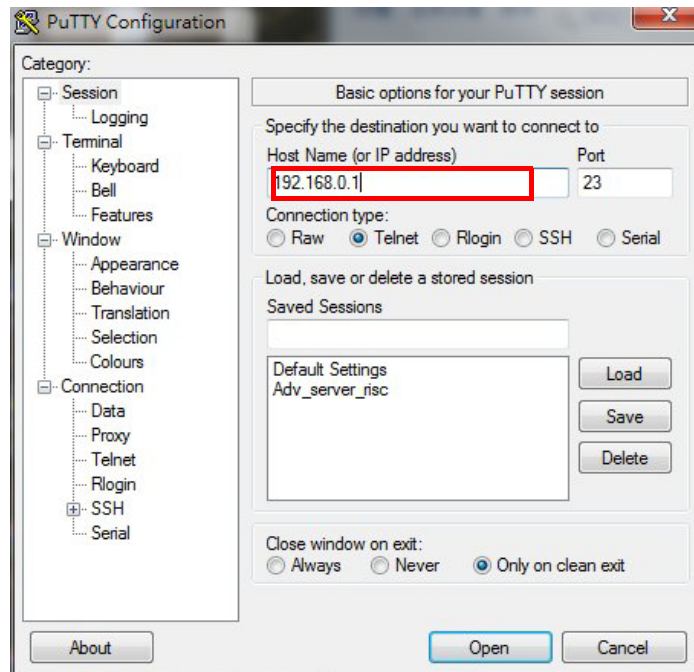
### 2.3.1 Building Connection of Linux Console

UBC-200 connects to a host PC (Windows) by using the telnet function. In order to communicate with a host PC, a telnet communication program such as PuTTY is a must. Below are instructions of how to set up a telnet console by using “PuTTY” on a Windows host:

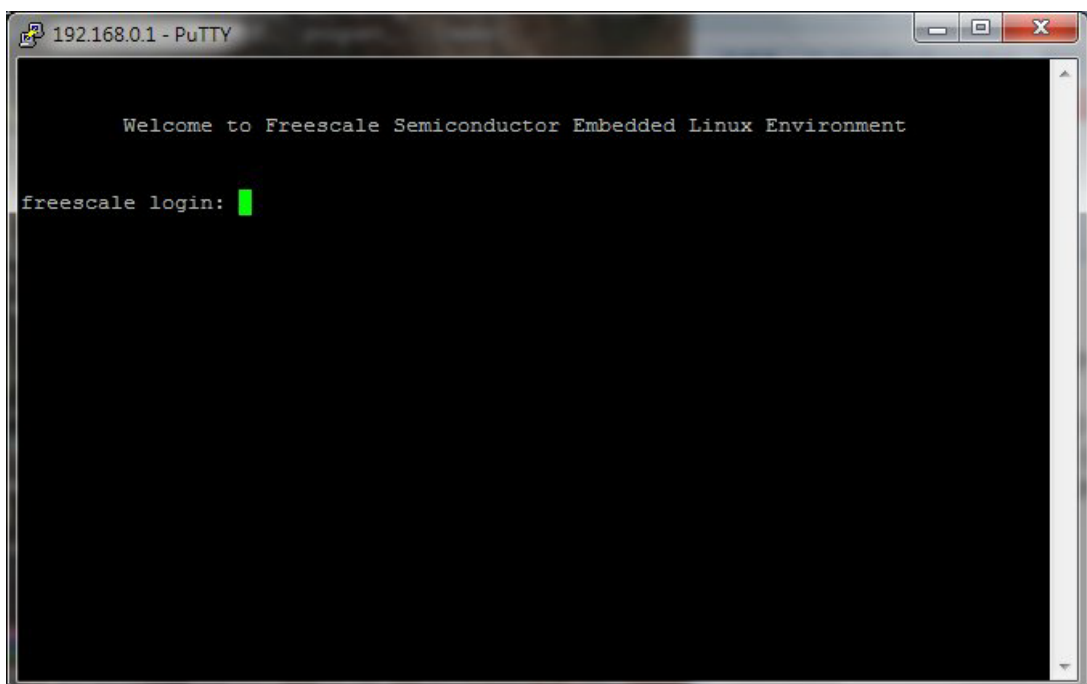
1. Connect UBC-200 to your Windows PC by using a LAN cable then insert the DC-in cable and power on UBC-200.
2. Open PuTTY on your Windows PC, and select the settings as shown below:

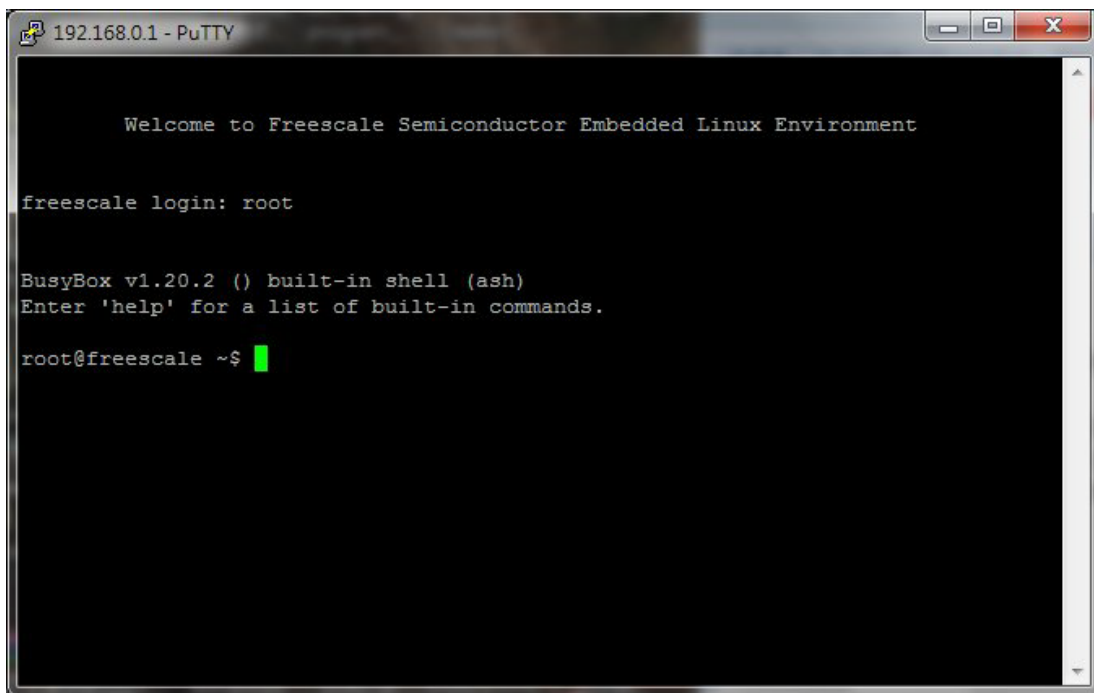






3. Now the Linux console prompt should be displayed on the terminal screen.





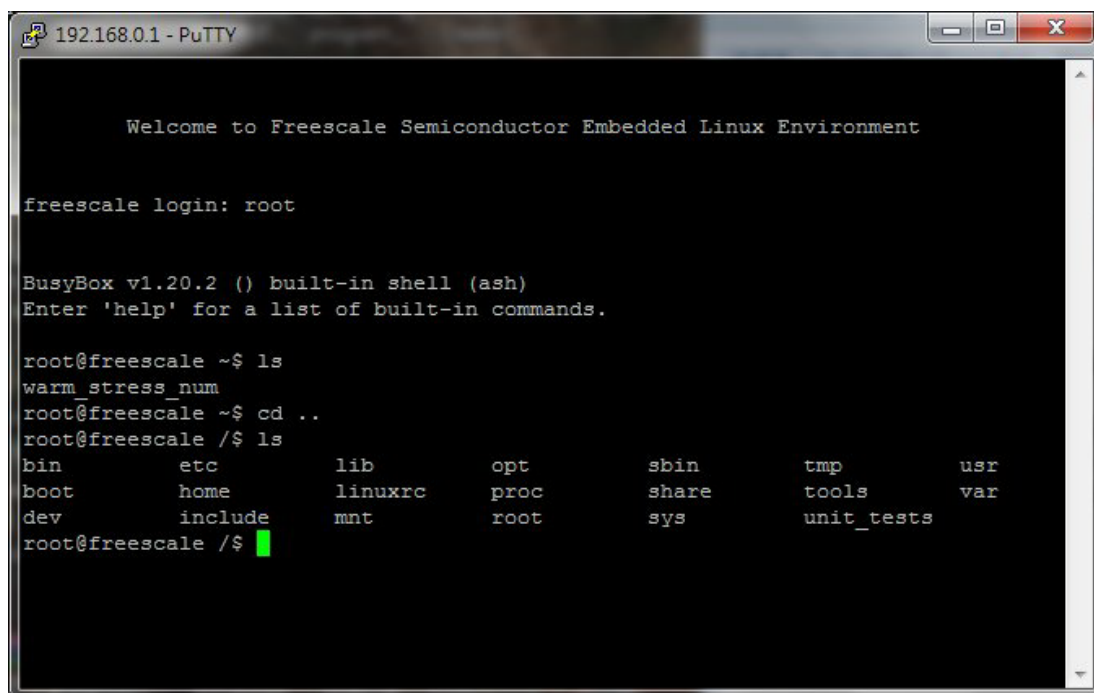
```
192.168.0.1 - PuTTY

Welcome to Freescale Semiconductor Embedded Linux Environment

freescale login: root

BusyBox v1.20.2 () built-in shell (ash)
Enter 'help' for a list of built-in commands.

root@freescale ~$
```



```
192.168.0.1 - PuTTY

Welcome to Freescale Semiconductor Embedded Linux Environment

freescale login: root

BusyBox v1.20.2 () built-in shell (ash)
Enter 'help' for a list of built-in commands.

root@freescale ~$ ls
warm_stress_num
root@freescale ~$ cd ..
root@freescale /$ ls
bin          etc          lib          opt          sbin         tmp          usr
boot        home        linuxrc     proc         share        tools        var
dev          include     mnt         root         sys          unit_tests
root@freescale /$
```

## 2.4 Test Tools

All test tools must be verified on UBC-200 system, please prepare required test fixtures before verifying each specified I/O. If you have any problems, please contact your Advantech contact window for help.

### 2.4.1 eMMC Test

1. Erase and check.

```
#dd if=/dev/zero of=/dev/mmcblk0 bs=1024 count=1 seek=25118
```

```
1+0 records in
```

```
1+0 records out
```

```
#hexdump -C /dev/mmcblk0 -s 25720832 -n 16
```

```
01887800 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
```

2. Write and check.


```
#echo -n "0123456789ABCDEF" | dd of=/dev/mmcblk0 bs=1024 count=1 seek=25118
```

```
0+1 records in
```

```
0+1 records out
```

```
#hexdump -C /dev/mmcblk0 -s 25720832 -n 16
```

```
01887800 30 31 32 33 34 35 36 37 38 39 41 42 43 44 45 46 |0123456789ABCDEF|
```

**Note!**  Please make sure parameter "seek" is equal to 25118 as indicated in red in above codes. If you create the file to a wrong sector, that may damage the system.

### 2.4.2 USB Test

1. Insert USB flash disk then assure it is in UBC-200 device list.
2. Erase and check.

```
#dd if=/dev/sdb of=/dev/mmcblk0 bs=1024 count=1 seek=25118
```

```
1+0 records in
```

```
1+0 records out
```

```
#hexdump -C /dev/mmcblk0 -s 25720832 -n 16
```

```
01887800 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
```

3. Write and check.


```
#echo -n "0123456789ABCDEF" | dd of=/dev/mmcblk0 bs=1024 count=1 seek=25118
```

```
0+1 records in
```

```
0+1 records out
```

```
#hexdump -C /dev/mmcblk0 -s 25720832 -n 16
```

```
01887800 30 31 32 33 34 35 36 37 38 39 41 42 43 44 45 46 |0123456789ABCDEF|
```

**Note!**  This operation may damage the data stored in the USB flash disk. Please make sure there is no critical data in the USB flash disk being used for this test.

### 2.4.3 SD Test

1. When booting from eMMC, you would see only below directories:

```
#ls /dev/mmcblk*
/dev/mmcblk0 /dev/mmcblk0boot0 /dev/mmcblk0boot1 /dev/mmcblk0p1
```

2. Insert SD card to SD card slot (SD1) and check your device again. You should be able to see more directories. /dev/mmcblk1 is the SD card storage.

```
#ls /dev/mmcblk*
/dev/mmcblk0 /dev/mmcblk0boot1 /dev/mmcblk1
/dev/mmcblk1p2/dev/mmcblk0boot0 /dev/mmcblk0p1 /dev/mmcblk1p1
```

3. Erase and check.

```
#dd if=/dev/zero of=/dev/mmcblk0 bs=1024 count=1 seek=25118
1+0 records in
1+0 records out
#hexdump -C /dev/mmcblk0 -s 25720832 -n 16
01887800 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
```

Step 4: Write and check

```
#echo -n "0123456789ABCDEF" | dd of=/dev/mmcblk0 bs=1024 count=1
seek=25118
0+1 records in
0+1 records out
#hexdump -C /dev/mmcblk0 -s 25720832 -n 16
01887800 30 31 32 33 34 35 36 37 38 39 41 42 43 44 45 46 |0123456789ABCDEF|
```

**Note!** *Please make sure parameter “seek” is equal to 25118 as indicated in red in above codes. If you create the file to a wrong sector, that may damage the system.*



## 2.4.4 HDMI Test

1. `#gplay /tools/Advantech.avi`
2. Then you can see the video demo on the default display screen.



Figure 2.8 HDMI Test

## 2.4.5 Mini PCIe (3 G and Wifi) Test

The command used to test 3G module is as follows, the supported module P/N is EWM-C106FT01E.

```
#3glink
Send AT commands...
#send (AT^M)
send (ATDT*99#^M)
expect (CONNECT)
AT^M^M
OK^M
ATDT*99#^M^M
CONNECT
-- got it
.....
```

The command used to test WiFi module is as following, the supported module P/N is EWM-W142F01E.

```
#ifconfig wlan0 up
#iwlist wlan0 scanning
#wpa_passphrase "WiFi name" password > /tmp/wpa.conf
#wpa_supplicant -Bdwext -iwlan0 -c/tmp/wpa.conf
#dhclient wlan0
```



## 2.4.6 OpenGL Test

Please follow the instructions below to test OpenGL on the UBC-200 platform:

1. Change path to /opt/viv\_samples/vdk.

```
#cd /opt/viv_samples/vdk
```

```
#ls tutorial*
```

```
tutorial1          tutorial2_es20    tutorial4          tutorial5_es20
tutorial1_es20    tutorial3          tutorial4_es20    tutorial6
tutorial2          tutorial3_es20    tutorial5          tutorial7
```

2. Run tutorial7 for OpenGL ES 1.1.

Using Vertex Buffer Objects (VBO) can substantially increase performance by reducing the bandwidth required to transmit geometry data. Information such as vertex, normal vector, color, and so on is sent once to locate device video memory and then bound and used as needed, rather than being read from system memory every time. This example illustrates how to create and use vertex buffer objects.

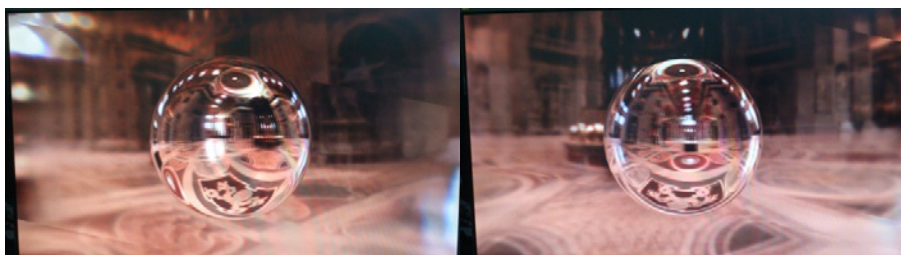
```
#./tutorial7
```



3. Run tutorial3\_es20 for OpenGL ES 2.0.

A ball made of a mirroring material and centered at the origin spins about its Y-axis and reflects the scene surrounding it.

```
#./tutorial3_es20
```



---

## 2.4.7 LAN Test

You can set dynamic ip by following instructions:

1. Open `/etc/rc.d/rc.conf`.
2. Modify `IPADDR0="dhcp"`.
3. `#reboot`.

Then you will see:

### **#ifconfig**

```
eth0      Link encap:Ethernet  HWaddr 00:04:9F:01:30:E0
          inet addr:172.17.21.96  Bcast:172.17.21.255  Mask:255.255.254.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:129 errors:0 dropped:18 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15016 (14.6 KiB)  TX bytes:656 (656.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  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:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

You can set static ip by following instructions:

1. Open `/etc/rc.d/rc.conf`.
2. Modify `IPADDR0="xxx.xx.xx.xxx"`.
3. Modify `NETMASK0="xxx.xx.xx.xxx"`.
4. **#reboot**
5. Open `/etc/rc.d/rc.local`
6. Add `#` to mark `ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up`

Here is a real case for your reference. The hosts(UBC-200) IP is 172.17.21.97; the target(A desktop computer) IP is 172.17.20.192

Desktop PC ip address:

```
#ifconfig eth0 172.17.20.192
#ifconfig eth0
```

```
eth0      Link encap:Ethernet  HWaddr 00:04:9F:01:30:E0
          inet addr:172.17.21.97  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2851 errors:0 dropped:271 overruns:0 frame:0
          TX packets:30 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:291407 (284.5 KiB)  TX bytes:2000 (1.9 KiB)
```

The target computer (Client) IP address is 172.17.20.192, so we can use the below command to see if we can get any response from the client.

```
#ping 172.17.20.192
```

```
PING 172.17.20.192 (172.17.20.192): 56 data bytes
64 bytes from 172.17.20.192: seq=0 ttl=128 time=7.417 ms
64 bytes from 172.17.20.192: seq=1 ttl=128 time=0.203 ms
64 bytes from 172.17.20.192: seq=2 ttl=128 time=0.300 ms

--- 172.17.20.192 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.203/2.640/7.417 ms
```

### 2.4.8 Watchdog Timer Test

1. Executing 'wdt\_driver\_test.out'

```
#!/unit_tests/wdt_driver_test.out
```

```
Usage: wdt_driver_test <timeout> <sleep> <test>
```

```
timeout: value in seconds to cause wdt timeout/reset
```

```
sleep: value in seconds to service the wdt
```

```
test: 0 - Service wdt with ioctl(), 1 - with write()
```

2. Please try below command to set timeout as 10 seconds, the system will reboot after that.

```
#!/unit_tests/wdt_driver_test.out 10 5 0
```

```
Starting wdt_driver (timeout: 10, sleep: 5, test: ioctl)
```

```
Trying to set timeout value=10 seconds
```

```
The actual timeout was set to 10 seconds
```

```
Now reading back -- The timeout is 10 seconds
```

Press [CTRL+C], you should be able to see the below results:

```
imx2-wdt imx2-wdt.0: Unexpected close: Expect reboot!
```

Then system will reboot in 10 seconds

---

## 2.4.9 Photo Demo Test

Execute the following commands to run the Photo demo application on UBC-200.

```
#cd /tools
```

```
#./fbv Advantech.jpg
```

Then you can see the photo demo on the default display screen.



Figure 2.9 Photo Demo

# Chapter 3

## Software Functionality

This chapter details the Linux operating system on the UBC-200 platform.

## 3.1 Introduction

UBC-200 platform is an embedded system with Linux kernel 3.0.35 inside. It contains all system-required shell commands and drivers ready. We do not offer IDE developing environment in the standard BSP, users can evaluate and develop under Ubuntu 10.04LTS environment.

There are three major boot components for Linux, “u-boot.bin”, “ulmage” and “File System”. The “u-boot.bin” is for initializing peripheral hardware parameters; the “ulmage” is the Linux kernel image and the “File System” is for Linux O.S. used.

It will not be able to boot into Linux environment successfully if one of above three files is missing from booting media (SD card or onboard flash)

The purpose of this chapter is to introduce software development of UBC-200 to you, so that you can develop your own application(s) efficiently.

UBC-200 is designed for supporting Linux host only so you may fail developing your AP on Windows/Android host PC. For now the official supported host version is Ubuntu 10.04 LTS, host PC in any other version may have compatibility issue. In this case, we strongly recommend to have Ubuntu 10.04 LTS installed to your host PC before start UBC-200 evaluation/development.

## 3.2 Package Content

### 3.2.1 Source Code Package

UBC-200 source code package (BSP) contains cross compiler, Linux source code, Uboot source code, root file system and some scripts used in OS development. Some of above components are developed by Advantech and the others are developed by open source community. UBC-200 source code package is composed of six main folders: “cross\_compiler”, “document”, “image”, “package”, “scripts”, and “source”.

**Note!** *UBC-200 source code package (BSP) is Advantech’s Intellectual Property. If you need to access this package, please contact your Advantech support window.*



The description of U200LBVxxxx package contents:

- **“cross\_compiler”** → This folder contains source code for cross compiler.
- **“document”** → This folder contains user guide.
- **“image”** → This folder contains the ulmage, and the script for making Linux system media automatically.
- **“image/rootfs”** → This folder contains Linux root file system
- **“package”** → This folder contains source code provided by Freescale without any modification
- **“scripts”** → This folder contains scripts for configure system and compile images automatically.
- **“source”** → This folder contains source code owned by Advantech

### 3.2.1.1 cross\_compiler

You can use the cross compiler toolchain to compile the ulmage and related applications. (gcc version is 4.6.2 20110630)

Toolchain directory structure is as follow:

```
|-- bin // toolchain with prefix, such as arm-none-linux-gnueabi-gcc etc.
|-- lib // library files used for toolchain itself, not for application
|-- arm-fsl-linux-gnueabi
    |-- bin // toolchain without prefix, such as gcc.
    |-- debug-root // all debug tools
    |-- multi-libs // all libraries and headers.
    |-- armv5 // library for armv5 (i.mx 2xx). only support soft float point
    |-- armv6 // library for armv6 (i.mx 3xx), soft fpu version
    |-- armv7-a // library for armv7-a (i.mx5xx and i.mx6xx), hardware fpu version
|-- lib //default library. It can be used for armv4t and above.
    |-- usr
    |-- include //header files for the application development
    |-- lib //three-part library and static built library Freescale
```

### 3.2.1.2 document

User guide of how to setup up the environment of development

### 3.2.1.3 image

This folder includes ulmage & u-boot.

### 3.2.1.4 image/rootfs

Linux adopts Hierarchical File System (HFS), image/rootfs is the Linux file system in highest level of the tree structure.

The main folders in “rootfs” are listed as follows:

- bin → Common programs, shared by the system, the system administrator and the users.
- dev → Contains references to all the CPU peripheral hardware, which are represented as files with special properties.
- etc → Most important system configuration files are in /etc, this directory contains data similar to those in the Control Panel in Windows
- home → Home directories of the common users.
- lib → Library files, includes files for all kinds of programs needed by the system and the users.
- mnt → Standard mount point for external file systems.
- opt → Typically contains extra and third party software.
- proc → A virtual file system containing information about system resources. More information about the meaning of the files in proc is obtained by entering the command man proc in a terminal window. The file proc.txt discusses the virtual file system in detail.
- root → The administrative user's home directory. Mind the difference between /, the root directory and /root, the home directory of the root user.
- sbin → Programs for use by the system and the system administrator.
- sys → Linux sys file system

- tmp → Temporary space for use by the system, cleaned upon reboot, so doesn't use this for saving any work!
- unit\_tests → unit test tools are provided by Freescale i.MX6 product
- usr → Programs, libraries, documentation etc. for all user-related programs.
- var → Storage for all variable files and temporary files created by users, such as log files, the mail queue, the print spooler area, space for temporary storage of files downloaded from the Internet.
- tools → just for sample test.

### 3.2.1.5 scripts

Some scripts provided by Advantech will help you configure system or build the images more quickly. Please check them as follows:

- setenv.sh → A script to setup the developing environment quickly.
- cfg\_uboot.sh → A script to configure the u-boot building setup quickly.
- mk\_uboot.sh → A script to build the u-boot and copy the “u-boot” to “image” folder after building.
- cfg\_kernel.sh → A script to configure the kernel building setup quickly.
- mk\_kernel.sh → A script to build the “ulmage” and copy the “ulmage” to “image” folder after building.
- mksd-linux.sh → A script to setup up a bootable SD card if users build their images

### 3.2.1.6 source

This folder contains sub-directories “linux-3.0.35” and “u-boot-2009.08”. They are the source codes of the Linux kernel and U-boot.

The main sub-directories under “linux-3.0.35” are listed as follows:

- arch → The items related to hardware platform, most of them are for CPU.
- block → The setting information for block.
- crypto → The encryption technology that kernel supports.
- Documentation → The documentation for kernel.
- drivers → The drivers for hardware.
- firmware → Some of firmware data for old hardware.
- fs → The file system the kernel supports.
- include → The header definition for the other programs used.
- init → The initial functions for kernel.
- ipc → Define the communication for each program of Linux O.S.
- kernel → Define the Kernel process, status, schedule, signal.
- lib → Some of libraries.
- mm → The data related the memory.
- net → The data related the network.
- security → The security setting.
- sound → The module related audio.
- virt → The data related the virtual machine.

There are also various README files in ./source/linux-3.0.35/Documentation, you can find the kernel-specified installations and notes for drivers. You can refer to ./source/linux-3.0.35/Documentation/00-INDEX for a list of the purpose of each README/note.



## 3.3 Set up Build Environment

All instructions in this guide are based on Ubuntu 10.04 LTS developing environment. Please install the Ubuntu 10.04 LTS at your PC/NB in advance.

When you obtain the UBC-200 Linux source code package, please refer to following instructions to extract to your developing environment:

1. Copy "U200LBVxxxx.tar.bz2" package to your desktop. "xxxx" is the version of the BSP source code.
2. Start your "Terminal" on Ubuntu 10.04 LTS.
3. **\$sudo su** (Change to "root" authority)
4. Input user password
5. **#cd Desktop/**
6. **#tar xvf U200LBVxxxx.tar.bz2** (Unzip file)

Advantech offer you a script to setup the developing environment quickly. You can refer following steps to setup your developing environment:

1. Open "Terminal" on Ubuntu 10.04 LTS.
2. **\$sudo su** (Change to "root" authority)
3. Input user password
4. Change directory to BSP's scripts folder
5. **#. setenv.sh** (To configure the developing environment automatically)
6. Then you can start to code the source code, build images, or compile applications.

### 3.3.1 setenv.sh

This script is used to configure the developing environment quickly. It will configure the folder paths for system, and you can also add/modify the setenv.sh by yourself if you have added/changed the folders and paths.

The major part of setenv.sh is shown as follows:

```
export SRCROOT=${PWD}/..
export CC_PATH=${SRCROOT}/cross_compiler/fsl-linaro-toolchain
export CROSS_COMPILE=${CC_PATH}/bin/arm-none-linux-gnueabi-
export CC=${CROSS_COMPILE}gcc
export STRIP=${CROSS_COMPILE}strip
export ARCH=arm
export KROOT=${SRCROOT}/source/linux-3.0.35
export ADVBOOT_SOURCE=${SRCROOT}/source/u-boot-2009.08
export UBOOT_SOURCE=${SRCROOT}/source/u-boot-2009.08
export ROOTFS=${SRCROOT}/image/rootfs
export LOG=${SRCROOT}/Build.log
export PATH=${CC_PATH}/bin:${UBOOT_SOURCE}/tools:$PATH
```

---

**Note!** You have to wrap "setenv.sh" once you open a new "Terminal" utility every time.



(i.e. #source setenv.sh)

**Note!** It is suggested to change to "root" authority to use the source code.



## 3.4 Build Instructions

This section will guide you how to build the u-boot & Linux kernel.

### 3.4.1 Build u-boot Image

Advantech has written a script to build the u-boot quickly. You can build u-boot image by follow below steps:

1. Open "Terminal" on Ubuntu 10.04 LTS
2. **\$sudo su** (Change to "root" authority)
3. Input user password
4. Change directory to BSP's scripts folder
5. **#. setenv.sh** (To configure the developing environment automatically)
6. **#. /cfg\_uboot.sh mx6q\_rom-7420\_1G\_config** (To set the u-boot configuration automatically)
7. **#. /mk\_uboot.sh** (Start to build the u-boot)
8. Then you can see u-boot\_crc.bin and u-boot\_crc.bin.crc are being built and located in ../image.

### 3.4.2 Build Linux Kernel Image

Advantech offer you a script to build the "ulmage" quickly. You can build ulmage by follow below steps:

1. Open "Terminal" on Ubuntu 10.04 LTS.
2. **\$sudo su** (Change to "root" authority)
3. Input user password.
4. Change directory to BSP's scripts folder.
5. **#. setenv.sh** (To configure the developing environment automatically)
6. **#. /cfg\_kernel.sh imx6\_rom7420\_defconfig** (To set the ulmage configuration automatically)
7. **#. /mk\_kernel.sh** (Start to build the ulmage)
8. Then you can see ulmage is being built and located in ../image.

### 3.4.3 Build Log

You can find the build log from the directory of UBC-200 BSP. If you got any error message when building Linux kernel, it is suggested to look into the log file to learn more detail about it.

## 3.5 Source Code Modification

This section will guide you how to use the Linux source code. You will see some examples of using BSP source code in this section.

### 3.5.1 Add a Driver to Kernel by menuconfig

You can add a driver to kernel by menuconfig. Here is an example to guide you how to add a RTC driver (Seiko Instruments S-35390A) to Linux kernel. Please refer to the following steps:

1. Open "Terminal" on Ubuntu 10.04 LTS.
2. **\$sudo su** (Change to "root" authority)
3. Input user password.
4. Change directory to BSP's scripts folder
5. **#. setenv.sh** (To configure the developing environment automatically)
6. **#!/cfg\_kernel.sh menuconfig**
7. Then you will see a GUI screen (Linux Kernel Configuration) as below:

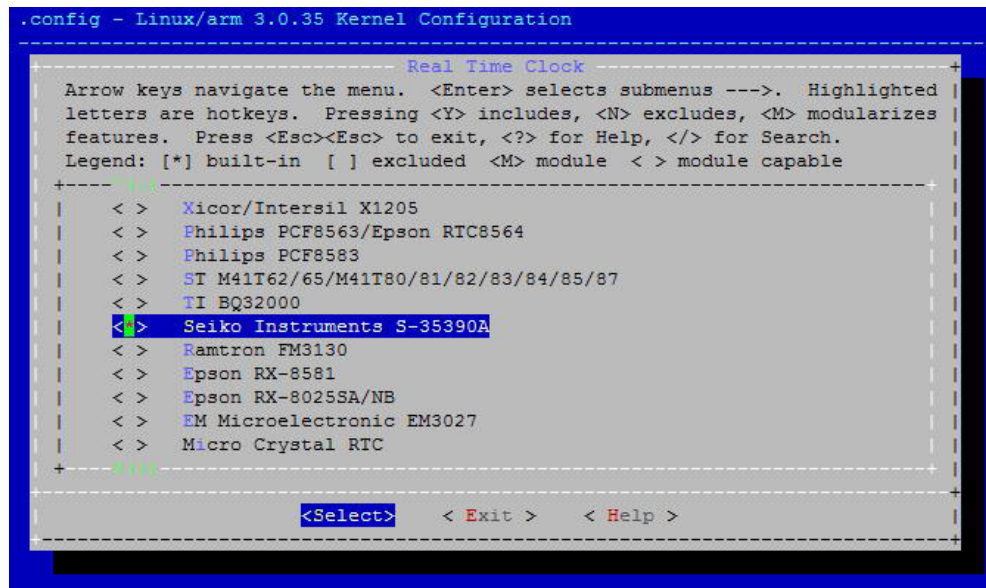
```

.config - Linux/arm 3.0.35 Kernel Configuration
-----
Linux/arm 3.0.35 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted
letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes
features. Press <Esc><Esc> to exit, <?> for Help, </> for Search.
Legend: [*] built-in [ ] excluded <M> module < > module capable
-----
[ ] Patch physical to virtual translations at runtime (EXPERIMENTAL)
[*] General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
System Type --->
Bus support --->
Kernel Features --->
Boot options --->
CPU Power Management --->
Floating point emulation --->
Userspace binary formats --->
+-----+
<Select> < Exit > < Help >

```

Figure 3.1 Linux Kernel Configuration

- Select “Device Drivers”→”Real Time Clock”, you will see an option “Seiko Instruments S-35390A” on the list. Choose this option then exit and save your configuration.



**Figure 3.2 Selecting Seiko Instruments S-35390A**

- Change directory to “source/linux-3.0.35/arch/arm/mach-mx6”, edit the “board-mx6q\_ROM-7420.h” and “board-mx6q\_advantech.c”. Please add below codes to source/linux-3.0.35/arch/arm/mach-mx6/board-mx6q\_ROM-7420.h:

```

/* I2C */
static struct i2c_board_info mxc_i2c0_board_info[] __initdata = {
    {
        I2C_BOARD_INFO("sgt15000", 0x0a),
    },
    {
        I2C_BOARD_INFO("s35390a", 0x30),
    },
};
;

```

Please add below codes to

```

source/linux-3.0.35/arch/arm/mach-mx6/board-mx6q_advantech.c
i2c_register_board_info(0, mxc_i2c0_board_info,
    ARRAY_SIZE(mxc_i2c0_board_info));

```

- Please refer to former Chapter 3.4.2 to rebuild the kernel with RTC driver (Seiko Instruments S-35390A) after completing above steps

**Note!** *If you cannot find the driver for your device from the list, please contact your hardware vender.*



### 3.5.2 Change UBC-200 Boot Logo

By default, UBC-200 shows a boot logo when booting up. You can replace the logo to whatever you want by following below steps:

1. Install “netpbm” by typing `$sudo apt-get install netpbm`
2. Prepare your boot logo. For example: bootlogo.png (Under folder Desktop/bootlogo)

**Note!** *This picture should be in PNG format and less than 224 colors. It is suggested to have the image resolution equal to your LCD panel size.*



3. Open "Terminal" on Ubuntu 10.04 LTS.
4. `$sudo su` (Change to “root” authority)
5. Input user password.
6. `#cd Desktop/bootlogo` (Go into the folder that bootlogo.png located)
7. `#pngtopnm bootlogo.png | ppmquant 224 | pnmtoplainpnm > logo_linux_clut224.ppm`
8. copy logo\_linux\_clut224.ppm to the directory source/linux-3.0.35/drivers/video/logo/
9. Then you can refer Chapter 3.4.2 to rebuild the kernel with your own boot logo.

## 3.6 Create a Linux System Boot Media

UBC-200 supports boot from SD card, SATA device and onboard flash. This section will guide you how to build a image for UBC-200 Linux system boot media.

### 3.6.1 Create a Linux System SD Card

#### 3.6.1.1 From Source Code Package

When you receive the UBC-200 Linux source code package, you can refer following steps to create a Linux system SD card for booting up from it.

1. Open "Terminal" on Ubuntu 10.04 LTS.
2. `$sudo su` (Change to “root” authority)
3. Input your password.
4. Insert one SD card to your developing computer
5. Check the SD card location, like: /dev/sdf
6. Change directory to BSP's scripts folder
7. `#./mkzd-linux.sh /dev/sdf`
8. Type “y” (Start to copy files, wait until it shows [Done])

Then insert the Linux system SD card to UBC-200 SD card slot (SD1), it will boot up with Linux environment.

#### 3.6.2 Boot from Onboard Flash

If you already have a Linux system SD card, you can refer to the following steps to copy the content to onboard flash and then boot from onboard flash. Advantech also provides you a script “mkinand-linux.sh” to speed up the process of installing system image to onboard flash.

1. Refer to Chapter 3.6.1 to make a Linux system SD card
2. Insert this Linux system SD card to UBC-200 and connect serial console.
3. On UBC-200 platform, type `#root` (Login)

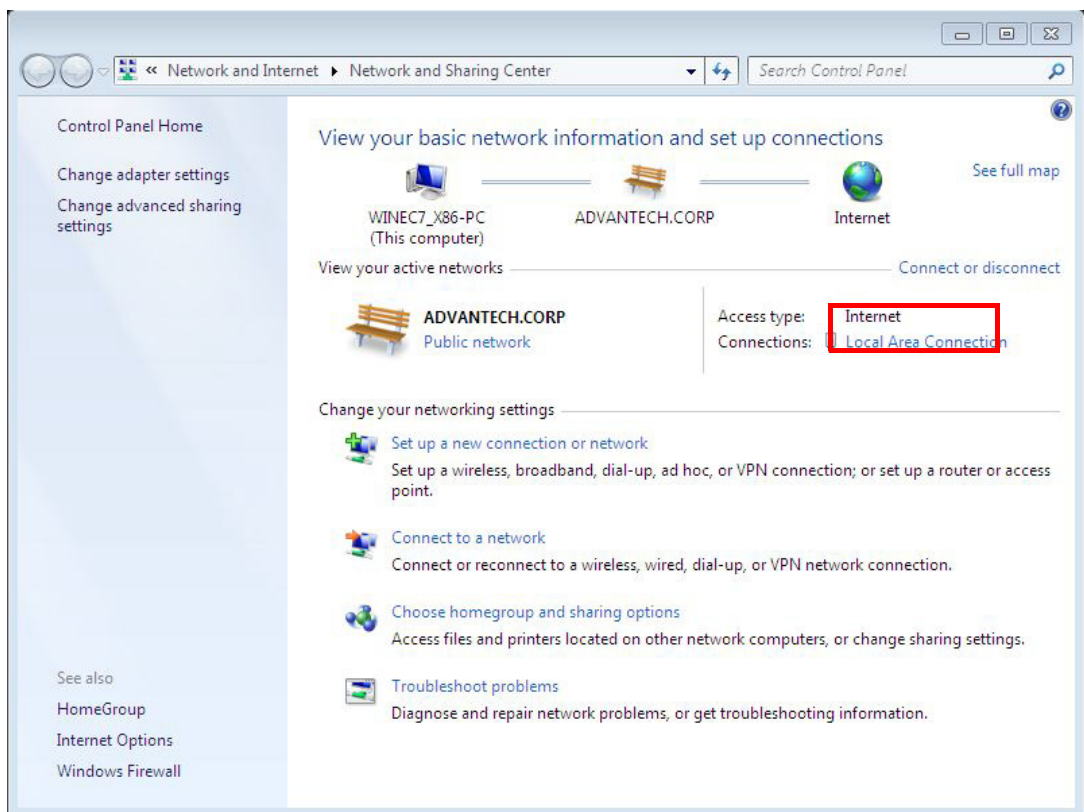
4. On UBC-200 platform, type `#cd /mk_inand`
5. On UBC-200 platform, type `#!/mkinand-linux.sh /dev/mmcblk0`
6. On UBC-200 platform, type “y” (Start to copy files, wait until it shows [Done])
7. Power off and remove this SD card.

Then you can boot from onboard flash without SD card.

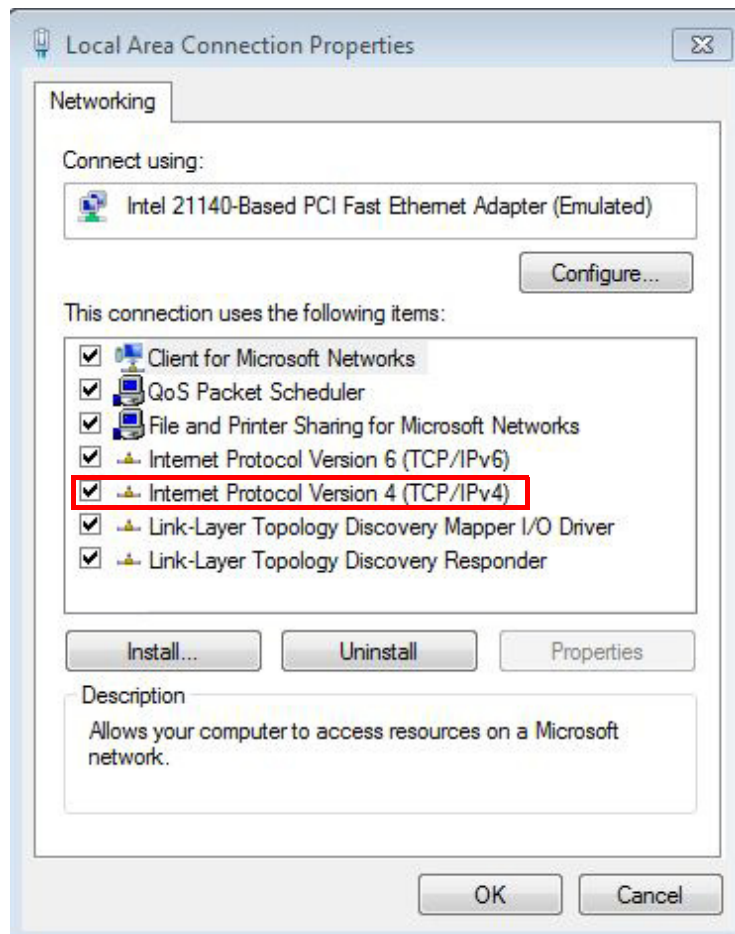
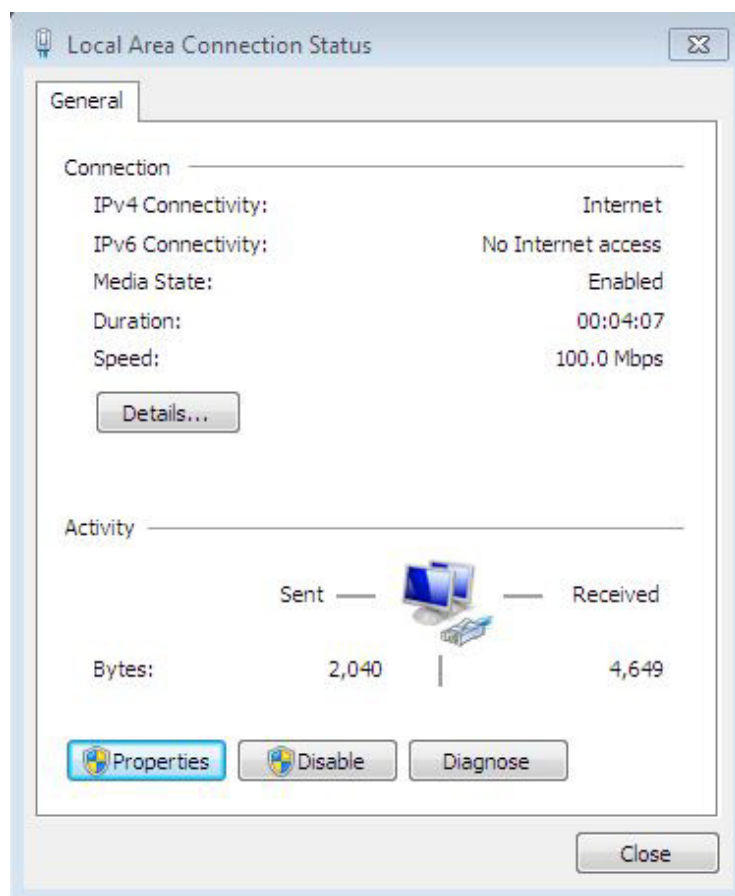
## 3.7 Telnet Function

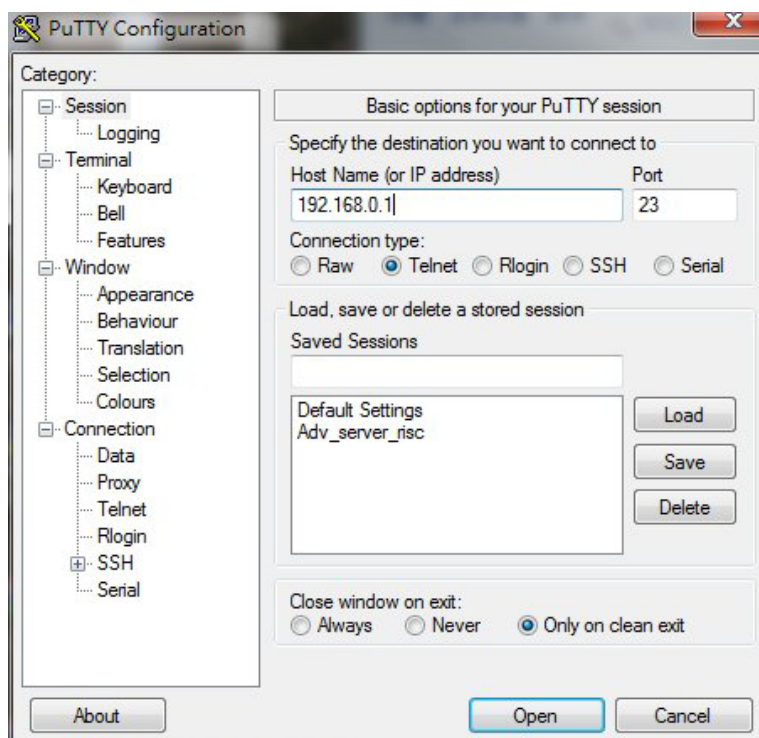
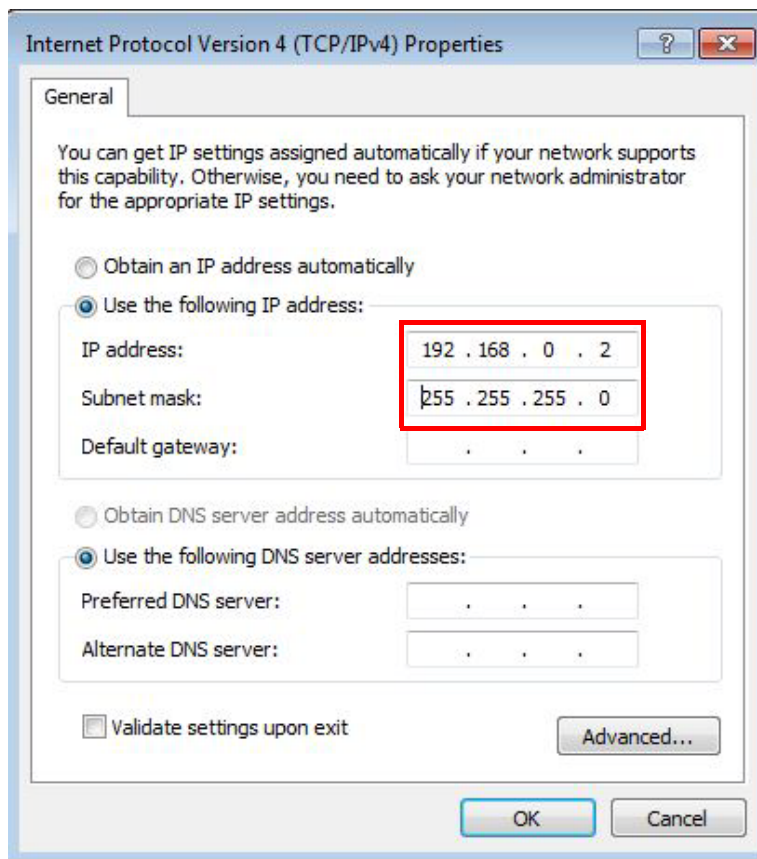
UBC-200 connects to a host PC (Windows) by using telnet function. In order to communicate with host PC, telnet communication program such as PuTTY is a must. Below is the detail instruction of how to set up a telnet console “PuTTY” on a Windows host:

1. Connect UBC-200 to your Windows PC by using LAN cable then insert DC-in and power on UBC-200.
2. Open PuTTY on your Windows PC, and select the settings as shown below.



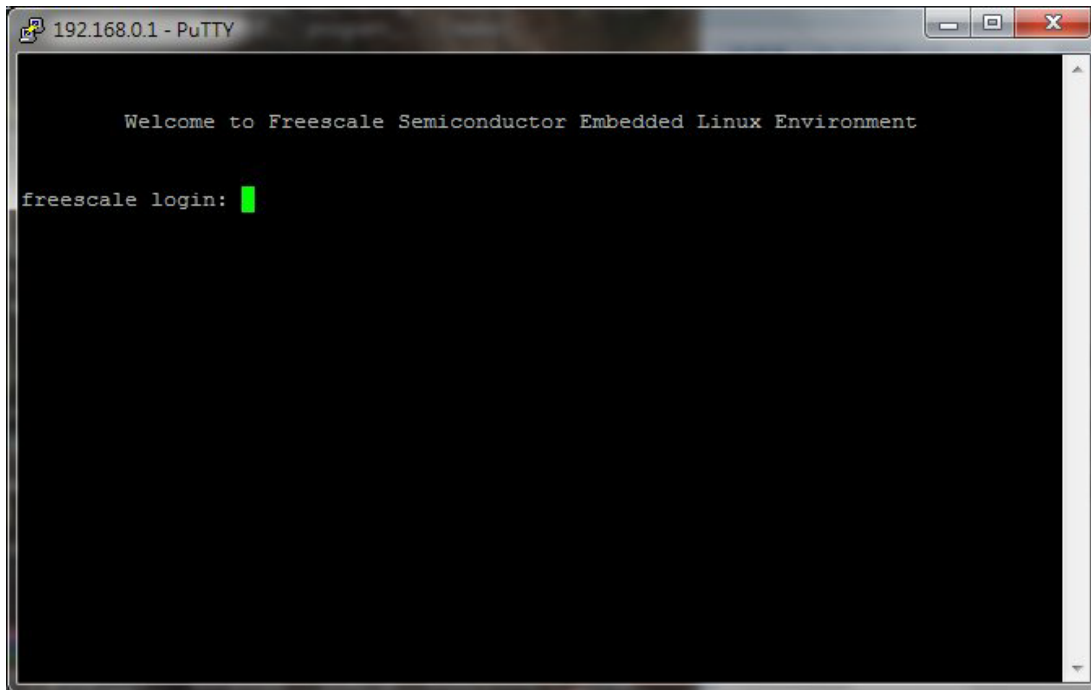






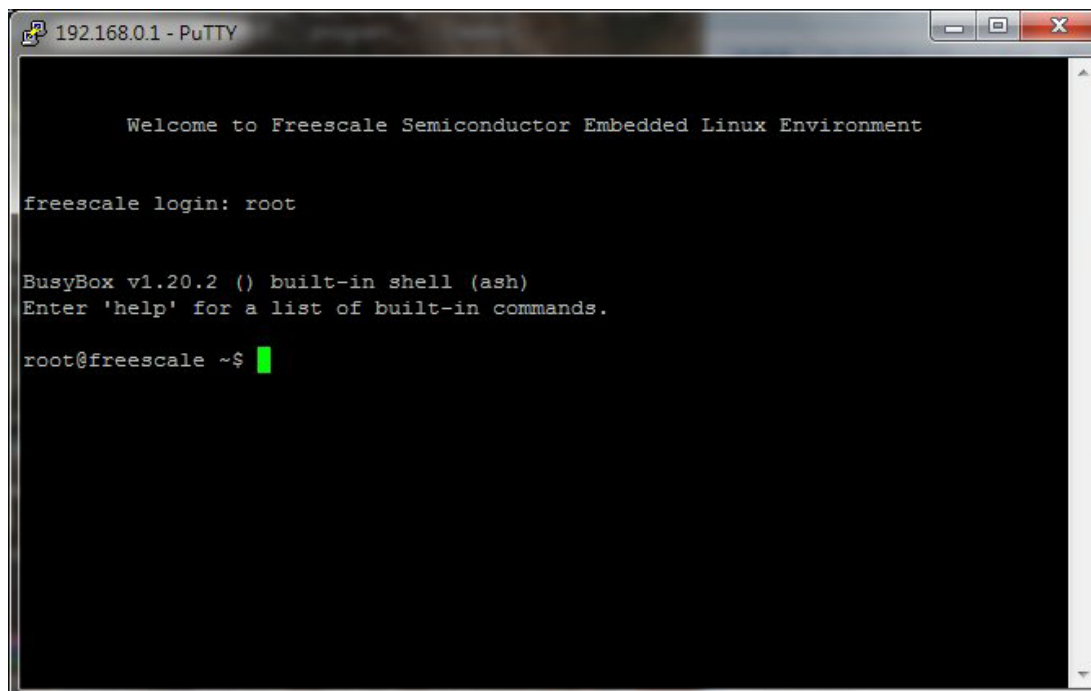


3) Now the Linux console prompt should be displayed on the terminal screen.



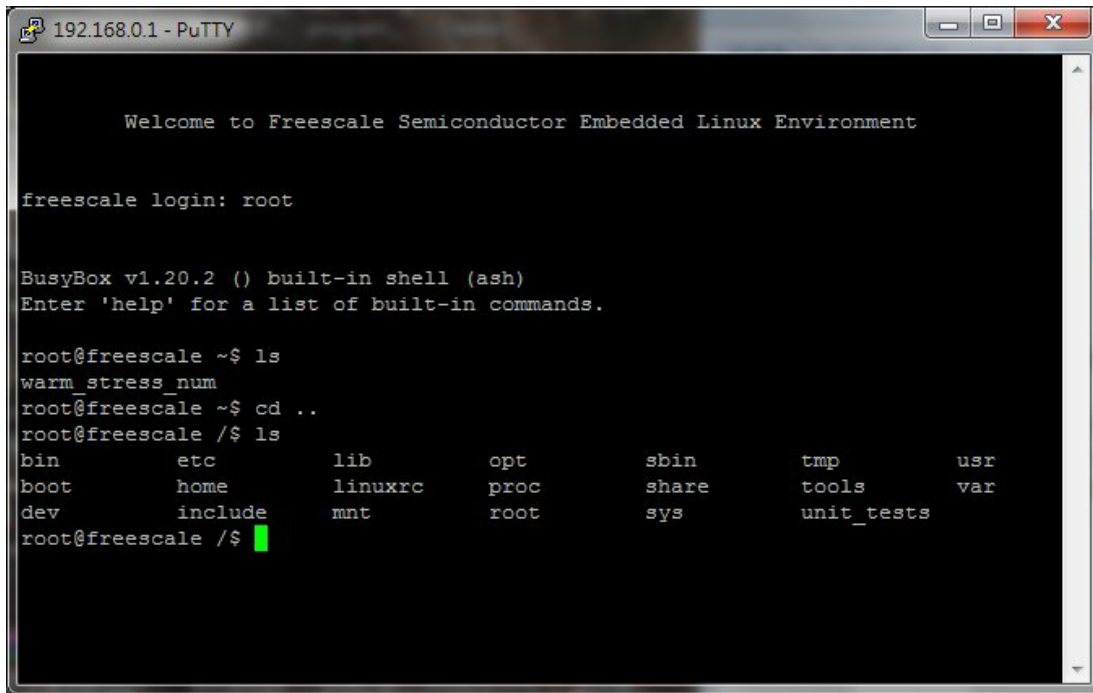
A terminal window titled "192.168.0.1 - PuTTY" with a black background and white text. The text reads: "Welcome to Freescale Semiconductor Embedded Linux Environment" followed by "freescale login:" and a green cursor.

```
192.168.0.1 - PuTTY
Welcome to Freescale Semiconductor Embedded Linux Environment
freescale login: █
```



A terminal window titled "192.168.0.1 - PuTTY" with a black background and white text. The text reads: "Welcome to Freescale Semiconductor Embedded Linux Environment" followed by "freescale login: root", "BusyBox v1.20.2 () built-in shell (ash)", "Enter 'help' for a list of built-in commands.", and "root@freescale ~\$" with a green cursor.

```
192.168.0.1 - PuTTY
Welcome to Freescale Semiconductor Embedded Linux Environment
freescale login: root
BusyBox v1.20.2 () built-in shell (ash)
Enter 'help' for a list of built-in commands.
root@freescale ~$ █
```



```
192.168.0.1 - PuTTY
Welcome to Freescale Semiconductor Embedded Linux Environment

freescall login: root

BusyBox v1.20.2 () built-in shell (ash)
Enter 'help' for a list of built-in commands.

root@freescall ~$ ls
warm_stress_num
root@freescall ~$ cd ..
root@freescall /$ ls
bin          etc          lib          opt          sbin         tmp          usr
boot        home        linuxrc     proc         share       tools       var
dev         include     mnt         root         sys         unit_tests
root@freescall /$
```

## 3.8 Linux Software AP and Testing on UBC-200

This section will guide you how to develop your own application under Linux environment. First of all, an example “Hello World” will be shown. And then you will see some pre-installed test programs on UBC-200 will be introduced in this section.

### 3.8.1 “Hello World!” Application and Execution

This section will guide you how to write a sample application “Hello World”. You can refer to following steps:

1. Open "Terminal" on Ubuntu 10.04 LTS.
2. **\$sudo su** (Change to “root” authority)
3. Type user password.
4. Change directory to BSP's scripts folder
5. **#. setenv.sh** (To configure the developing environment automatically)
6. **#cd ../source**
7. **#mkdir helloworld** (Create your own work directory on the Desktop)
8. **#cd helloworld** (Enter the work directory)
9. **#gedit helloworld.c** (Create a new C source file)

Edit the helloworld.c with the following source code:

```
#include <stdio.h>
void main()
{
    printf("Hello World!\n");
}
```

10. Save the file and exit.
11. **#\$CC -o helloworld helloworld.c** (To compile helloworld.c)
12. Then you can see “helloworld” in current directory.

13. Insert the Linux system SD card to your developing computer.
14. **#cp helloworld /media/rootfs/tool** (/media/rootfs is the mounted point of your Linux system SD card)
15. Remove this SD card and insert it to UBC-200, then open serial console.
16. On UBC-200 platform, type **#root** (Login)
17. On UBC-200 platform, type **#cd /tool**
18. On UBC-200 platform, type **#./helloworld**
19. Now you should be able to see "Hello World!" shown on UBC-200.

### 3.8.2 Watchdog Timer Sample Code

WatchDog Timer (WDT) sample code is as below:

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <linux/watchdog.h>
#include <sys/ioctl.h>
#include <unistd.h>

void help_info(void);
int main(int argc, const char *argv[])
{
    int fd, timeout, sleep_sec, test;
    int count=1;
    if (argc < 2) {
        help_info();
        return 1;
    }
    timeout = atoi(argv[1]);
    sleep_sec = atoi(argv[2]);
    if (sleep_sec <= 0) {
        sleep_sec = 1;
        printf("correct 0 or negative sleep time to %d seconds\n",
            sleep_sec);
    }
    test = atoi(argv[3]);
    printf("Starting wdt_driver (timeout: %d, sleep: %d, test: %s)\n",
        timeout, sleep_sec, (test == 0) ? "ioctl" : "write");
    fd = open("/dev/watchdog", O_WRONLY);
    if (fd == -1) {
        perror("watchdog");
        exit(1);
    }
    printf("Trying to set timeout value=%d seconds\n", timeout);
    ioctl(fd, WDIOC_SETTIMEOUT, &timeout);
    printf("The actual timeout was set to %d seconds\n", timeout);
```

```

ioctl(fd, WDIOC_GETTIMEOUT, &timeout);
printf("Now reading back -- The timeout is %d seconds\n", timeout);
while (1) {
    printf("WDT Time out counter:%d\n",count);
    if ((test !=0) && (test ==count)) {
        printf("Ping Watchdog (reset wdt)\n");
        ioctl(fd, WDIOC_KEEPAALIVE, 0);
        test=0;
        count=0;
    }
    sleep(sleep_sec);
    count+=sleep_sec;
}
return 0;
}

void help_info(void)
{
    printf("Usage: wdt_driver_test <timeout> <sleep> <trigger>\n");
    printf("    timeout: value in seconds to cause wdt timeout/reset\n");
    printf("    sleep: value in seconds to display wdt timeout\n");
    printf("    trigger: value in seconds to ping the wdt\n");
}

```

If you would like to change the WDT time, please modify:

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

### 3.8.3 Network Setup

If you would like to use DHCP mode instead, please goes to `/etc/rc.d/rc.conf`, and set `IPADDR0` to "dhcp".

If you prefer to use static IP as default, you should goes to `/etc/rc.d/rc.conf`, and set `IPADDR0` to the IP to be used. And the `NETMASK0` should be changed to corresponding netmask as well.

### 3.8.4 Storage (SATA /eMMC/SD Card)

The storages devices are named as follows:

Device	Name
eMMC	/dev/mmcblk0
SD card	/dev/mmcblk1

### 3.8.5 3G Sample Code

The code of 3glink, we have tried this command in 3 G test (Section 1.8).

```
#!/bin/bash
```

```
echo "Send AT commands..."
```

```
pppd connect 'chat -v -s -t 10 "" "AT" "" "ATDT*99#" "CONNECT" ""'  
user username password password /dev/ttyUSB2 460800 nodetach crtscts  
debug usepeerdns defaultroute &
```



# Chapter 4

## System Recovery

This chapter introduces how to recover Linux operating system if it is damaged accidentally.

---

## 4.1 System Recovery

This section provides detail procedures of restoring the eMMC image. You can do system recovery following these steps if you destroy onboard flash image by accident.

1. Open "Terminal" on Ubuntu 10.04 LTS.
2. **\$sudo su** (Change to "root" authority).
3. Input your password.
4. Insert one SD card to your developing computer.
5. Check the SD card location, like /dev/sdf.
6. Change directory to BSP's scripts folder.
7. **#!/mksd-linux.sh /dev/sdf**
8. Type "y" (Start to copy files, wait until it shows [Done]).
9. Insert the SD card to UBC-200 SD card slot.
10. On UBC-200 platform, type **#root** (Login)
11. On UBC-200 platform, type **#cd /mk\_inand**
12. On UBC-200 platform, type **#!/mkinand-linux.sh /dev/mmcblk0**
13. On UBC-200 platform, type "y "  
(Start to copy files, wait until it shows [Done])
14. Power off and remove this SD card.



# Chapter 5

## Advantech Services

This chapter introduces Advantech design in serviceability, technical support and warranty policy for UBC-200.

---

## 5.1 RISC Design-in Services

With the spread of industrial computing, a whole range of new applications have been developed, resulting in a fundamental change in the IPC industry. In the past, System Integrators (SI) were used to completing projects without outside assistance but now such working models have moved on. Due to diverse market demands and intense competition, cooperation for (both upstream and downstream) vertical integration has become a much more effective way to create competitive advantages. As a result, ARM-based CPU modules were born out of this trend. Concentrating all necessary components on the CPU module and placing other parts on the carrier board in response to market requirements for specialization, provides greater flexibility while retaining its low power consumption credentials.

Advantech has been involved in the industrial computer industry for many years and found that customers usually have the following questions when implementing modular designs.

### **General I/O design capability**

Although customers possess the ability for vertical integration and have enough know-how and core competitiveness in the professional application field, the lack of expertise and experience in general power and I/O design causes many challenges for them, especially integrating CPU modules into their carrier board.

### **The acquisition of information**

Even if the individual client is able to obtain sufficient information to make the right decision for the specialized vertical application, some customers encounter difficult problems dealing with platform design in general and communicating with CPU or chipset manufacturers, thereby increasing carrier board design difficulties and risk as well as seriously impacting on Time-to-market and lost market opportunities.

### **Software development and modification**

Compared to x86 architectures, RISC architectures use simpler instruction sets, therefore the software support for x86 platforms cannot be used on RISC platforms. System integrators need to develop software for their system and do the hardware and software integration themselves. Unlike x86 platforms, RISC platforms have less support for Board Support Packages (BSP) and drivers as well. Even though driver support is provided, SIs still have to make a lot of effort to integrate it into the system core. Moreover, the BSP provided by CPU manufacturers are usually for carrier board design, so it's difficult for SIs to have an environment for software development.

In view of this, Advantech proposed the concept of Streamlined Design-in Support Services for RISC-based Computer On Modules (COM). With a dedicated professional design-in services team, Advantech actively participates in carrier board design and problem solving. Our services not only enable customers to effectively distribute their resources but also reduce R&D manpower cost and hardware investment.

By virtue of a close interactive relationship with leading original manufacturers of CPUs and chipsets such as ARM, TI and Freescale, Advantech helps solve communication and technical support difficulties, and that can reduce the uncertainties of product development too. Advantech's professional software team also focuses on providing a complete Board Support Package and assists customers to build up a software development environment for their RISC platforms.

Advantech RISC design-in services helps customers overcome their problems to achieve the most important goal of faster time to market through a streamlined RISC Design-in services.

Along with our multi-stage development process which includes: planning, design, integration, and validation, Advantech's RISC design-in service provides comprehensive support to the following different phases:

### **Planning stage**

Before deciding to adopt Advantech RISC COM, customers must go through a complete survey process, including product features, specification, and compatibility testing with software. So, Advantech offers a RISC Customer Solution Board (CSB) as an evaluation tool for carrier boards which are simultaneously designed when developing RISC COMs. In the planning stage, customers can use this evaluation board to assess RISC modules and test peripheral hardware. What's more, Advantech provides standard software Board Support Package (BSP) for RISC COM, so that customers can define their product's specifications as well as verifying I/O and performance at the same time. We not only offer hardware planning and technology consulting, but also software evaluation and peripheral module recommendations (such as WiFi, 3G, BT). Resolving customer concerns is Advantech's main target at this stage. Since we all know that product evaluation is the key task in the planning period, especially for performance and specification, so we try to help our customers conduct all the necessary tests for their RISC COM.

### **Design stage**

When a product moves into the design stage, Advantech will supply a design guide of the carrier board for reference. The carrier board design guide provides pin definitions of the COM connector with limitations and recommendations for carrier board design, so customers can have a clear guideline to follow during their carrier board development. Regarding different form factors, Advantech offers a complete pin-out check list for different form factors such as Q7, ULP and RTX2.0, so that customers can examine the carrier board signals and layout design accordingly. In addition, our team is able to assist customers to review the placement/layout and schematics to ensure the carrier board design meets their full requirements. For software development, Advantech RISC software team can assist customers to establish an environment for software development and evaluate the amount of time and resources needed. If customers outsource software development to a 3rd party, Advantech can also cooperate with the 3rd party and provide proficient consulting services. With Advantech's professional support, the design process becomes much easier and product quality will be improved to meet their targets.

### **Integration stage**

This phase comprises HW/SW integration, application development, and peripheral module implementation. Due to the lack of knowledge and experience on platforms, customers need to spend a certain amount of time on analyzing integration problems. In addition, peripheral module implementation has a lot to do with driver designs on carrier boards, RISC platforms usually have less support for ready-made drivers on the carrier board, therefore the customer has to learn from trial and error and finally get the best solution with the least effort. Advantech's team has years of experience in customer support and HW/SW development knowledge. Consequently, we can support customers with professional advice and information as well as shortening development time and enabling more effective product integration.

### Validation stage

After customer's ES sample is completed, the next step is a series of verification steps. In addition to verifying a product's functionality, the related test of the product's efficiency is also an important part at this stage especially for RISC platforms.

As a supportive role, Advantech primarily helps customers solve their problems in the testing process and will give suggestions and tips as well. Through an efficient verification process backed by our technical support, customers are able to optimize their applications with less fuss. Furthermore, Advantech's team can provide professional consulting services about further testing and equipment usage, so customers can find the right tools to efficiently identify and solve problems to further enhance their products quality and performance.

## 5.2 Contact Information

Below is the contact information for Advantech customer service

Region/Country	Contact Information
America	1-888-576-9688
Brazil	0800-770-5355
Mexico	01-800-467-2415
Europe (Toll Free)	00800-2426-8080
Singapore & SAP	65-64421000
Malaysia	1800-88-1809
Australia (Toll Free)	1300-308-531
China (Toll Free)	800-810-0345 800-810-8389 Sales@advantech.com.cn
India (Toll Free)	1-800-425-5071
Japan (Toll Free)	0800-500-1055
Korea (Toll Free)	080-363-9494 080-363-9495
Taiwan (Toll Free)	0800-777-111
Russia (Toll Free)	8-800-555-01-50

You can also reach our service team through the website below; our technical support engineer will provide quick response once the form is filled out:

[http://www.advantech.com.tw/contact/default.aspx?page=contact\\_form2&subject=Technical+Support](http://www.advantech.com.tw/contact/default.aspx?page=contact_form2&subject=Technical+Support)

## 5.3 Global Service Policy

### 5.3.1 Warranty Policy

Below is the warranty policy of Advantech products:

#### 5.3.1.1 Warranty Period

Advantech branded off-the-shelf products and 3rd party off-the-shelf products used to assemble Advantech Configure to Order products are entitled to a 2 years complete and prompt global warranty service. Product defect in design, materials, and workmanship, are covered from the date of shipment.

All customized products will by default carry a 15 months regional warranty service. The actual product warranty terms and conditions may vary based on sales contract.

All 3rd party products purchased separately will be covered by the original manufacturer's warranty and time period, and shall not exceed one year of coverage through Advantech.

#### 5.3.1.2 Repairs under Warranty

It is possible to obtain a replacement (Cross-Shipment) during the first 30 days of the purchase, thru your original ADVANTECH supplier to arrange DOA replacement if the products were purchased directly from ADVANTECH and the product is DOA (Dead-on-Arrival). The DOA Cross-Shipment excludes any shipping damage, customized and/or build-to-order products.

For those products which are not DOA, the return fee to an authorized ADVANTECH repair facility will be at the customers' expense. The shipping fee for reconstructive products from ADVANTECH back to customers' sites will be at ADVANTECH's expense.

#### 5.3.1.3 Exclusions from Warranty

The product is excluded from warranty if

- The product has been found to be defective after expiry of the warranty period.
- Warranty has been voided by removal or alternation of product or part identification labels.
- The product has been misused, abused, or subjected to unauthorized disassembly/modification; placed in an unsuitable physical or operating environment; improperly maintained by the customer; or failure caused which ADVANTECH is not responsible whether by accident or other cause. Such conditions will be determined by ADVANTECH at its sole unfettered discretion.
- The product is damaged beyond repair due to a natural disaster such as a lightning strike, flood, earthquake, etc.
- Product updates/upgrades and tests upon the request of customers who are without warranty.

### 5.3.2 Repair Process

#### 5.3.2.1 Obtaining an RMA Number

All returns from customers must be authorized with an ADVANTECH RMA (Return Merchandise Authorization) number. Any returns of defective units or parts without valid RMA numbers will not be accepted; they will be returned to the customer at the customer's cost without prior notice.

An RMA number is only an authorization for returning a product; it is not an approval for repair or replacement. When requesting an RMA number, please access ADVAN-

TECH's RMA web site: <http://erma.ADVANTECH.com.tw> with an authorized user ID and password.

You must fill out basic product and customer information and describe the problems encountered in detail in "Problem Description". Vague entries such as "does not work" and "failure" are not acceptable.

If you are uncertain about the cause of the problem, please contact ADVANTECH's Application Engineers (AE). They may be able to find a solution that does not require sending the product for repair.

The serial number of the whole set is required if only a key defective part is returned for repair. Otherwise, the case will be regarded as out-of-warranty.

#### **5.3.2.2 Returning the Product for Repair**

It is possible customers can save time and meet end-user requirements by returning defective products to any authorized ADVANTECH repair facility without an extra cross-region charge. It is required to contact the local repair center before offering global repair service.

It is recommended to send products without accessories (manuals, cables, etc.). Remove any unnecessary components from the product, such as CPU, DRAM, and CF Card. If you send all these parts back (because you believe they may be part of the problem), please note clearly that they are included. Otherwise, ADVANTECH is not responsible for any items not listed. Make sure the "Problem Description" is enclosed.

European Customers that are located outside European Community are requested to use UPS as the forwarding company. It is strongly recommended to add a packing list to all shipments. Please prepare a shipment invoice according to the following guidelines to decrease goods clearance time:

1. Give a low value to the product on the invoice, or additional charges will be levied by customs that will be borne by the sender.
2. Add information "Invoice for customs purposes only with no commercial value" on the shipment invoice.
3. Show RMA numbers, product serial numbers and warranty status on the shipment invoice.
4. Add information about Country of origin of goods

In addition, please attach an invoice with RMA number to the carton, then write the RMA number on the outside of the carton and attach the packing slip to save handling time. Please also address the parts directly to the Service Department and mark the package "Attn. RMA Service Department".

All products must be returned in properly packed ESD material or anti-static bags. ADVANTECH reserves the right to return un-repaired items at the customer's cost if inappropriately packed.

Besides that, "Door-to-Door" transportation such as speed post is recommended for delivery, otherwise, the sender should bear additional charges such as clearance fees if Air-Cargo is adopted.

Should DOA cases fail, ADVANTECH will take full responsibility for the product and transportation charges. If the items are not DOA, but fail within warranty, the sender will bear the freight charges. For out-of-warranty cases, customers must cover the cost and take care of both outward and inward transportation.

#### **5.3.2.3 Service Charges**

The product is excluded from warranty if :

- The product is repaired after expiry of the warranty period.
- The product is tested or calibrated after expiry of the warranty period, and a No Problem Found (NPF) result is obtained.

- The product, though repaired within the warranty period, has been misused, abused, or subjected to unauthorized disassembly/modification; placed in an unsuitable physical or operating environment; improperly maintained by the customer; or failure caused which ADVANTECH is not responsible whether by accident or other cause. Such conditions will be determined by ADVANTECH at its sole unfettered discretion.
- The product is damaged beyond repair due to a natural disaster such as a lightning strike, flood, earthquake, etc.
- Product updates and tests upon the request of customers who are without warranty.

If a product has been repaired by ADVANTECH, and within three months after such a repair the product requires another repair for the same problem, ADVANTECH will do this repair free of charge. However, such free repairs do not apply to products which have been misused, abused, or subjected to unauthorized disassembly/modification; placed in an unsuitable physical or operating environment; improperly maintained by the customer; or failure caused which ADVANTECH is not responsible whether by accident or other cause.

Please contact your nearest regional service center for detail service quotation.

Before we start out-of-warranty repairs, we will send you a pro forma invoice (P/I) with the repair charges. When you remit the funds, please reference the P/I number listed under "Our Ref". ADVANTECH reserves the right to deny repair services to customers that do not return the DOA unit or sign the P/I. Meanwhile, ADVANTECH will scrap defective products without prior notice if customers do not return the signed P/I within 3 months.

#### **5.3.2.4 Repair Report**

ADVANTECH returns each product with a "Repair Report" which shows the result of the repair. A "Repair Analysis Report" is also provided to customers upon request. If the defect is not caused by ADVANTECH design or manufacturing, customers will be charged US\$60 or US\$120 for in-warranty or out-of-warranty repair analysis reports respectively.

#### **5.3.2.5 Custody of Products Submitted for Repair**

ADVANTECH will retain custody of a product submitted for repair for one month while it is waiting for return of a signed P/I or payment (A/R). If the customer fails to respond within such period, ADVANTECH will close the case automatically. ADVANTECH will take reasonable measures to stay in proper contact with the customer during this one month period.

#### **5.3.2.6 Shipping Back to Customer**

The forwarding company for RMA returns from ADVANTECH to customers is selected by ADVANTECH. Per customer requirement, other express services can be adopted, such as UPS, FedEx and etc. The customer must bear the extra costs of such alternative shipment. If you require any special arrangements, please indicate this when shipping the product to us.

**ADVANTECH**

*Enabling an Intelligent Planet*

**[www.advantech.com](http://www.advantech.com)**

Please verify specifications before quoting. This guide is intended for reference purposes only.

All product specifications are subject to change without notice.

No part of this publication may be reproduced in any form or by any means, electronic, photocopying, recording or otherwise, without prior written permission of the publisher.

All brand and product names are trademarks or registered trademarks of their respective companies.

© Advantech Co., Ltd. 2014