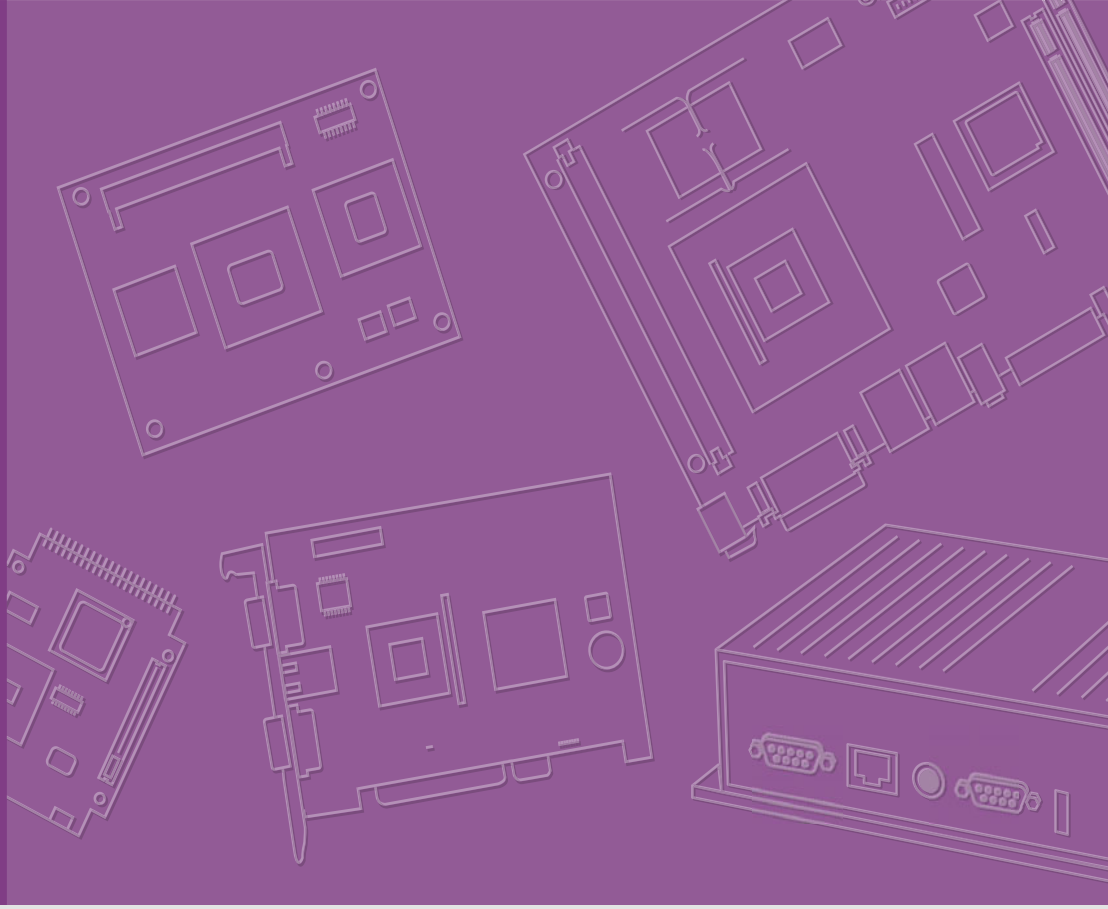


User Manual

UBC-330

Computing Box

ADVANTECH

Enabling an Intelligent Planet

Copyright

Acknowledgements

For more information about this and other Advantech products, please visit our website at:

http://www.Advantech.com/

Part No. 200K033020 Edition 1

Printed in Taiwan May 2015

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 merchandize 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.

- 1 x UBC-330 RISC Box
- 1 x (2 x 20) PIN Connector

Ordering Information

Model Number	Description
UBC-330NS-JLA1E	UBC-330 TI Sitara AM3352 Cortex A8 1GHz
1652006830	20 x 2P 2.54mm 180D 0156-1A40

Optional Accessories

Model Number	Description
1757003553	Adapter 100~240V 36W 12V 3A w/o PFC 9NA0361603
SQF-ISDS1-2G-86E	SQF SD C6 SLC 2G, 1CH (-40~85°C)
170203183C	Power Code 3P Europe (WS-010+WS-083)183cm
1700023307-01	A cable DC JACK/Plug-in 1*2P-5.0 10cm RSB-4220

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.
15. DO NOT LEAVE THIS EQUIPMENT IN AN ENVIRONMENT WHERE THE STORAGE TEMPERATURE MAY GO BELOW -20° C (-4° F) OR ABOVE 60° C (140° F). THIS COULD DAMAGE THE EQUIPMENT. THE EQUIPMENT SHOULD BE IN A CONTROLLED ENVIRONMENT.
16. CAUTION: DANGER OF EXPLOSION IF BATTERY IS INCORRECTLY REPLACED. REPLACE ONLY WITH THE SAME OR EQUIVALENT TYPE RECOMMENDED BY THE MANUFACTURER, DISCARD USED BATTERIES ACCORDING TO THE MANUFACTURER'S INSTRUCTIONS.

The sound pressure level at the operator's position according to IEC 704-1:1982 is no more than 70 dB (A).

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

Chapter 1

General Introduction

This chapter gives background information on UBC-330

1.1 Introduction

UBC-330 is a RISC complete box which integrated TI Sitara AM3352 Cortex-A8 processor. It offers 2x gigabit Ethernets, 6x serials, 4x GPI and 4x GPO. For the industrial application, the UART & GPIO feature a rugged ESD and isolation protection and it can protect system from unstable power damage. Otherwise, UBC-330 also has multiple power input and operation temperature support. It is an ideal solution for automation control such as smart grid, industrial and machinery automation applications.

1.2 Product Features

1.2.1 Key Features

- **CPU:**
  - TI Sitara AM3352
  - TI Sitara AM3352 Cortex A8 Single core 1GHz
- **System storage:**
  - DDR3 800 MHz
  - Capacity: on board 512MB
- **Giga LAN:**
  - Ethernet PHY : Realtek 8211
  - 2 x10/100/1000 Mbps

1.2.2 General

- **Kernel:** V 3.2.0
- **Flash:** On board 4GB EMMC
- 1 x USB, Support Host
- 2 x Giga LAN
- 1 x SD socket
- 1 x Reset button
- 1 x 40PIN connector for extend 8 x GPIO / 1 x CAN / 6 x COM etc features
- Support HW WDT (default WDT is 60s?Power on/off 1s)

1.2.3 Consumption

- **Idle:** 2.8W

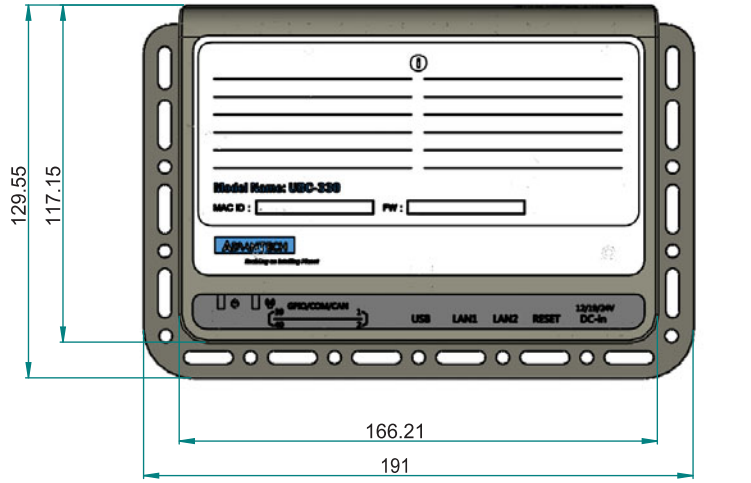
1.3 Mechanical Specification

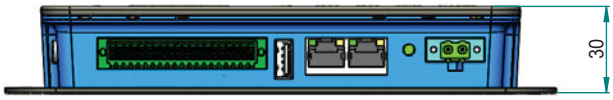
1.3.1 Weight

640g (Gross weight)

1.3.2 Dimension

- 129.55 x 191 mm (w/ metal plate)
- 117.15 x 166.21 mm(w/o metal plate)





1.4 Power Requirements

1.4.1 System Power

- Power input: DC 12 / 19 / 24 V

1.4.2 RTC Battery

- Power input: 3V / 210mAh

1.5 Environment Specification

1.5.1 Operating Temperature

- 0~55°C (32~131°F)

1.5.2 Relative Humidity

- 95% @ 40°C (Non condensing)

1.5.3 Storage Temperature

- -20~60°C (-4~104°F)

1.5.4 EMC

- CE / FCC / CCC / Class B

Chapter 2

HW Introduction

This chapter gives HW features and I/O testing introduction

2.1 Introduction

The following sections show the external connectors and pin assignments for applications.

2.2 UBC-330 I/O View

2.2.1 UBC-330 front view

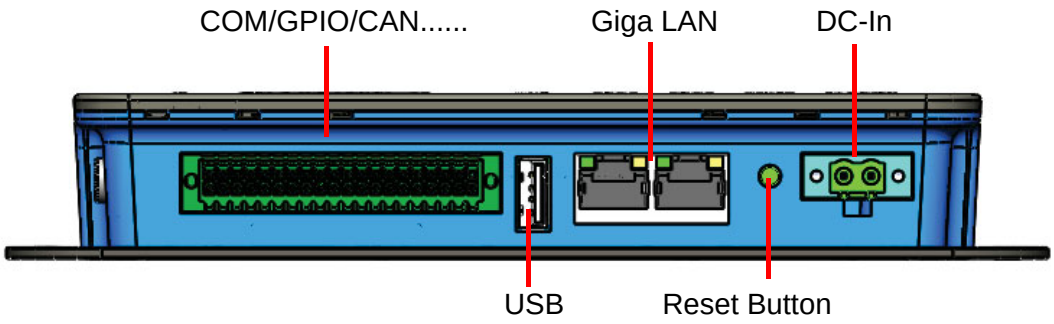


Figure 2.1 UBC-330 front view

2.2.2 UBC-330 LED Light

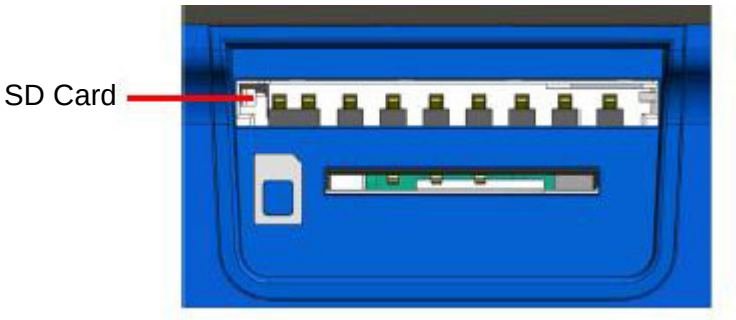


Figure 2.2 UBC-330 LED light

2.2.3 UBC-330 left view

Figure 2.3 UBC-330 left view

2.3 UBC-330 I/O Connector

2.3.1 Power input connector

UBC-330 comes with a DC-Jack header that carries 12 / 19 / 24V DC in external power input.

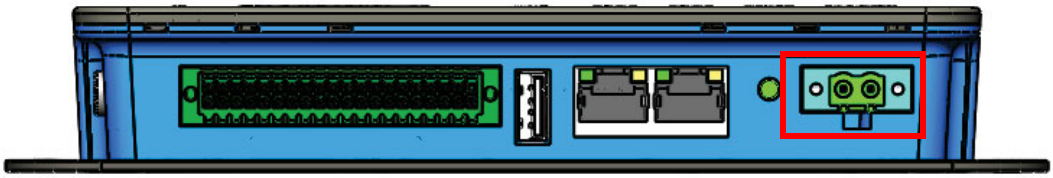


Figure 2.4 Power input connector

2.3.2 Reset Button

UBC-330 has a reset button on the front side. Press this button to activate the hardware reset function.

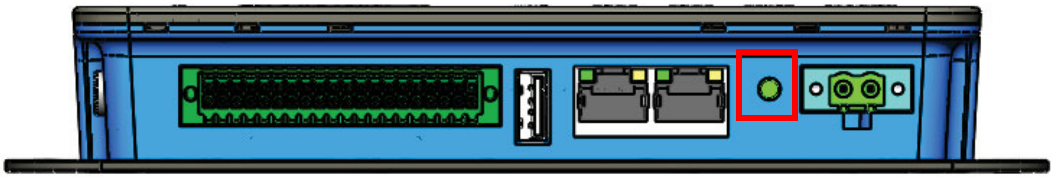


Figure 2.5 Reset Button

2.3.3 Ethernet Connector

UBC-330 provides two RJ45 LAN interface connector, which are fully compliant with IEEE 802.3u 10/100/1000 Base-T CSMA/CD standards. The Ethernet ports provide standard RJ-45 jack connector with LED indicators on the front side to show Active/ Link status and Speed status.

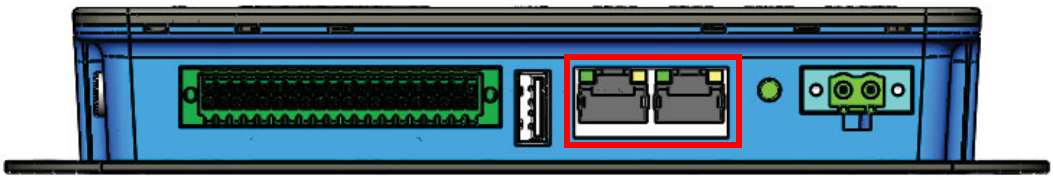


Figure 2.6 Ethernet Connector

Table 2.1: Ethernet PIN definition

Pin	Description
A1	MDI20+
A2	MDI20-
A3	MDI21+
A4	MDI21-
A5	GND
A6	GND
A7	MDI22+

Table 2.1: Ethernet PIN definition	
A8	MDI22-
A9	MDI23+
A10	MDI23-
A11	LAN2_100_LINK
A12	LAN2_1000_LINK
A13	+3.3V
A14	LAN2_ACT
B1	MDI10+
B2	MDI10-
B3	MDI11+
B4	MDI11-
B5	GND
B6	GND
B7	MDI12+
B8	MDI12-
B9	MDI13+
B10	MDI13-
B11	LAN1_100_LINK
B12	LAN1_1000_LINK
B13	+3.3V
B14	LAN1_ACT

2.3.4 USB Connector

UBC-330 has one standard USB2.0 Type A connector in the coastline.

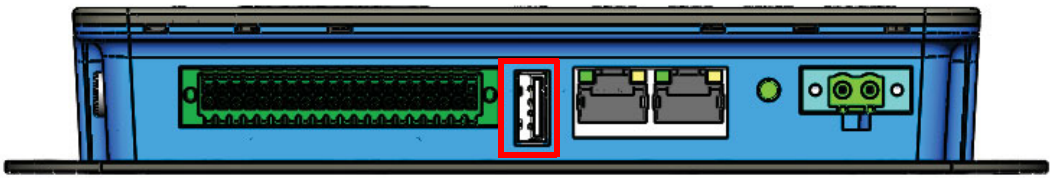


Figure 2.7 USB Connector

Table 2.2: USB PIN definition	
Pin	**Description**
1	+5V
2	USB Data-
3	USB Data+
4	GND

2.3.5 SD card socket

UBC-330 support SD card in Class2, 4, 6, 8, 10. Supported capacity is up to 4G (SDHC)

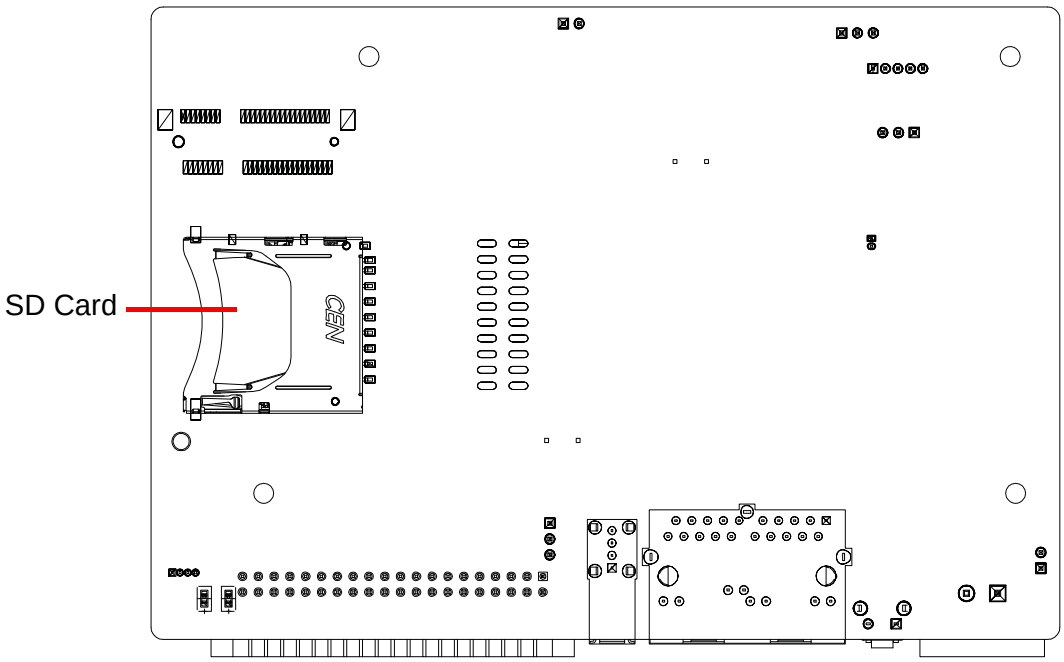


Figure 2.8 SD Card

Table 2.3: SD card PIN definition

Pin	Signal Name
1	DAT3
2	CMD
3	GND
4	+3.3V
5	CLK
6	GND
7	DAT0
8	DAT1
9	DAT2

2.3.6 COM / GPIO / CAN etc…connector

UBC-330 provides a 2X20 pin connector, which contains 2-wire UART with TX/RX, 2-wire UART Supporting RS232, one 5V CAN, one I2C bus and 8 GPIO w/isolation.

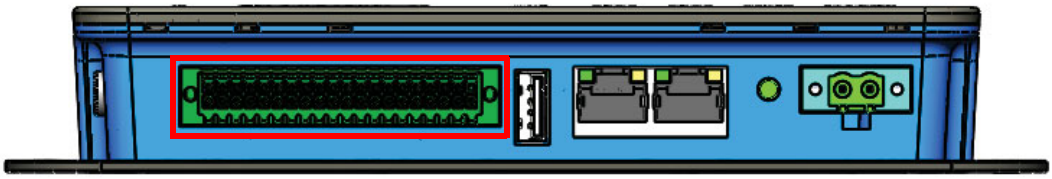


Figure 2.9 COM / GPIO / CAN etc…connector

Table 2.4: Other I/O PIN definition

Pin	Description
1	IDI0
2	IDO0
3	IDI1
4	IDO1
5	IDI2
6	IDO2
7	IDI3
8	IDO3
9	GND_iso
10	PCOM
11	NC
12	GND_iso
13	NC
14	NC
15	422_RXD-
16	NC
17	422_RXD+
18	I2C0_SCL
19	COM1_CTS
20	I2C0_SDA
21	COM1_TXD
22	GND
23	COM1_RTS
24	COM5_TX
25	COM1_RXD
26	COM5_RX
27	422-485_TXD+
28	CAN1_D+
29	422-485_TXD-
30	CAN1_D-
31	COM2_RX
32	COM0_TX
33	COM2_TX
34	COM0_RX

Table 2.4: Other I/O PIN definition	
35	COM4_TX
36	COM3_TX
37	COM4_RX
38	COM3_RX
39	GND
40	GND

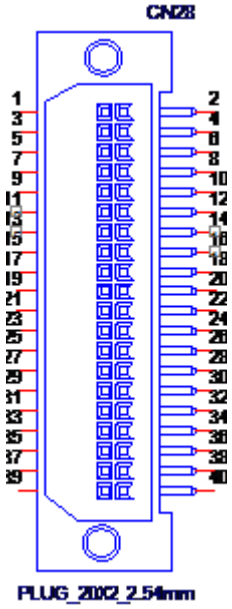
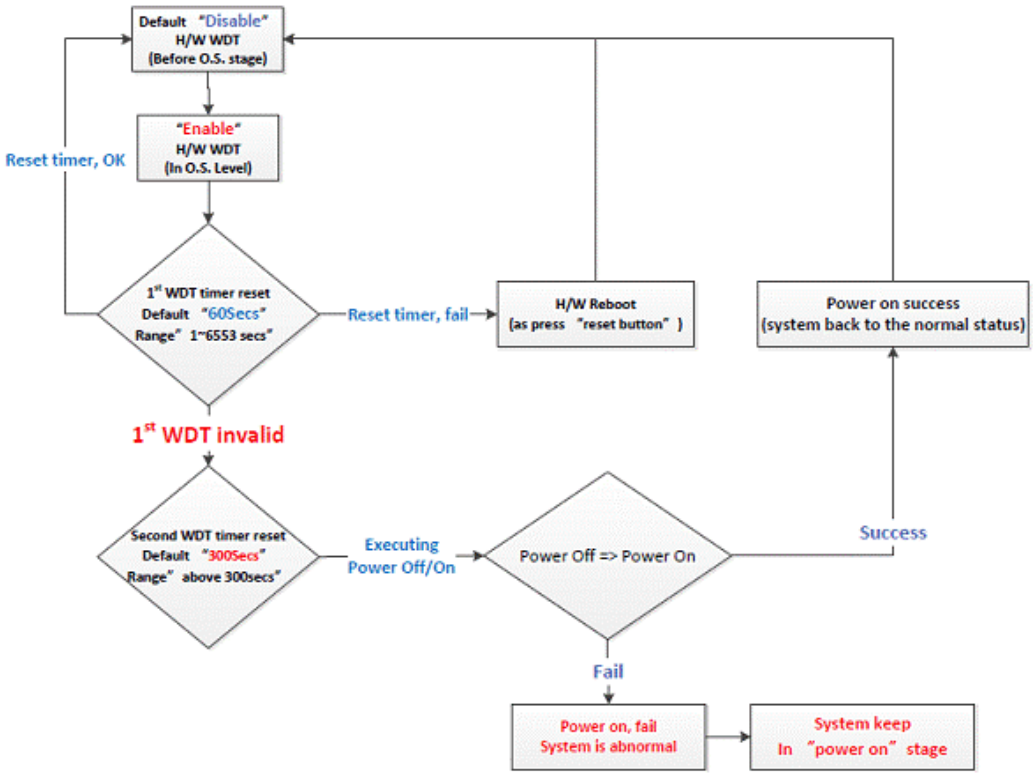


Figure 2.10 2X20 pin Connector

2.3.7 Watch Dog timer

UBC-330 has an external watchdog IC using TI msp430G2202, which will reset system when exception occurs. Please refer to below flow Diagram.



Advantech default WDT time is 60s. Anyway, Customer could change this date from 1s to 6553s that base on actual requierment .

The WDT test process is as below:

We can run the AutoRun_WTD program to test it.

#./AutoRun_WTD n

Here n is the feed time, that is to say, the test program will feed the watchdog every n seconds.

In AutoRun_WTD test program, first it will get the current timeout value, and then to set the timeout value to 10 seconds.

Program will feed to watchdog every n seconds, here the n is determined by the parameter, if the feed time is more than 10 seconds, the board will reboot after 10 seconds when run AutoRun_WTD program. If the watchdog time is less than 10, the board will not reboot because program will feed the watchdog within every 10 seconds.

To test the watchdog, run as follows, the board will reboot after 10 seconds.

```
root@am335x-adv:/unit_tests# ./AutoRun_WTD 15
```

Get the timeout value from driver: timeout = 60 seconds

Now, we set the timeout value to 10 seconds

Get the timeout value from driver again: timeout = 10 seconds

Setting succeeded and watchdog is enabled.

Feed the watchdog every 15 seconds.

Feed watchdog!

After reboot, user will see the follow the boot messages:

U-Boot SPL 2013.01.01-svn132 (Sep 28 2014 - 11:39:20)

musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, bulk combine, bulk split, HB-ISO Rx, HB-ISO Tx, SoftConn)

musb-hdrc: MHDRC RTL version 2.0

musb-hdrc: setup fifo_mode 4

musb-hdrc: 28/31 max ep, 16384/16384 memory

USB Peripheral mode controller at 47401000 using PIO, IRQ 0

musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, bulk combine, bulk split, HB-ISO Rx, HB-ISO Tx, SoftConn)

musb-hdrc: MHDRC RTL version 2.0

musb-hdrc: setup fifo_mod

2.4 Quick Start of UBC-330

2.4.1 Debug Port Connection

1. Connect "COM0_TX" and "COM0_RX" and "GND" to standard DB9 connector.
2. Connect DB9 to server PC's COM port.

2.4.2 Debug Port setting

1. Please download "Putty". http://putty.cs.utah.edu/download.html
2. Click and open your "putty" software?write-in some data are as below, then click "open" .
3. Connect your Adapater to UBC-330, then the terminal will display some information of bootlodader. Terminal interface will ask user for write-in "user name" and "password", user need write-in "root" as a supper user to log in the system and click "Enter".
   Above, user have already opened the UBC-330, For the rest I/O testing

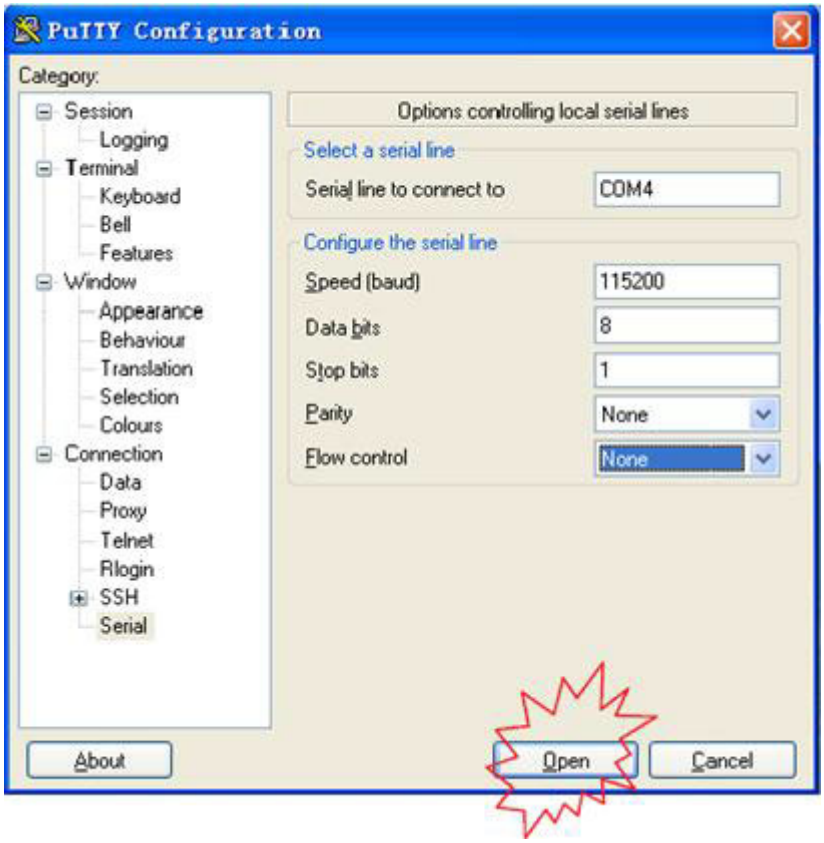


Figure 2.11 HyperTerminal Settings for Terminal Setup

2.5 Test Tools

All test tools must be verified on UBC-330, please prepare required test fixtures before verifying each specified I/O. If you have any problem to get the test fixture, please contact your Advantech contact window for help.

2.5.1 eMMC Test

Step1: Check the space of NAND flash

```
root@am335x-evm:~# fdisk -l /dev/mmcblk1
```

Disk /dev/mmcblk1: 3909 MB, 3909091328 bytes

4 heads, 16 sectors/track, 119296 cylinders, total 7634944 sectors

Units = sectors of 1 * 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk identifier: 0xb7e5e6db

Device	Boot	Start	End	Blocks	Id	System
/dev/mmcblk1p1		2048	22527	10240	83	Linux
/dev/mmcblk1p2		22528	63487	20480	83	Linux

Step2: Run program to read/write NAND flash.

```
root@am335x-adv:/unit_tests# ./AutoRun_eMMC.sh mmcblk1
```

```
============================================================
========Test Read/write and operation of filesystem for eMMC===============
============================================================
10240+0 records in
10240+0 records out
10240+0 records in
10240+0 records out
10240+0 records in
10240+0 records out
=================MMC Read/Write test pass================

=================MMC fdisk test pass====================

mke2fs 1.42.1 (17-Feb-2012)
5+0 records in
5+0 records out
===============MMC FS test passes=========================

mke2fs 1.42.1 (17-Feb-2012)
================MMC MKFS test pass=======================

==============all MMC function test PASS==================
```

2.5.2 USB Host Test

Step1: Plug a USB flash device into USB connector.
Step2: Mount the USB flash and check system can detect it.
Step3: Run program to read/write USB flash.

```
root@am335x-adv:/unit_tests# ./AutoRun_usb_host.sh
```

```
disk_to_test=sda
---create partition for sda
---format file system
partion_list=sda1
test /dev/sda1
pass
```

2.5.3 SD Test

Step 1: Insert SD card into SD1 slot.
Step 2: Mount the SD card device and check system can detect it.

```
root@am335x-adv:/unit_tests# fdisk -l /dev/mmcblk0
```

Disk /dev/mmcblk0: 3980 MB, 3980394496 bytes
255 heads, 63 sectors/track, 483 cylinders, total 7774208 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Device	Boot	Start	End	Blocks	Id	System
/dev/mmcblk0p1	*	63	144584	72261	c	W95 FAT32 (LBA)
/dev/mmcblk0p2		160650	7759394	3799372+	83	Linux

Step 3: Run program to read/write SD.

```
root@am335x-adv:/unit_tests# ./AutoRun_sd.sh mmcblk0
```

10240+0 records in
10240+0 records out
10240+0 records in
10240+0 records out
10240+0 records in
10240+0 records out
==========SD test pass==========

2.5.4 SPI Test

Step1: Power turns on and boots into OS
Step2: Run program to test SPI flash read/write.

```
root@am335x-adv:/unit_tests# ./AutoRun_spi.sh
```

####################---SPI Test for RSB 4220----####################
the spi falsh info:

31 0 4096 mtdblock0
==========
Test next block:/dev/mtd0 mtdblock0
==========
mtd.type = MTD_NORFLASH
mtd.flags = MTD_CAP_NORFLASH
mtd.size = 4194304 (4M)
mtd.erasesize = 4096 (4K)
mtd.writesize = 1
mtd.oobsize = 0
regions = 0

```
Erased 4194304 bytes from address 0x00000000 in flash
Copied 4194304 bytes from address 0x00000000 in flash to ./temp-1.img
0000000 ffff ffff ffff ffff ffff ffff ffff ffff
*
0400000

============================================================
 ################              SPI mtdblock0  Read 1 PASS !!!
============================================================
4096+0 records in
4096+0 records out
0000000 0000 0000 0000 0000 0000 0000 0000 0000
*
0400000
Copied 4194304 bytes from ./all-0.img to address 0x00000000 in flash
Copied 4194304 bytes from address 0x00000000 in flash to ./temp-0.img
0000000 0000 0000 0000 0000 0000 0000 0000 0000
*
0400000

============================================================
################## ----------SPI mtdblock0 Write 0 PASS !!!
============================================================

============================================================
####################----------> SPI Test all mtdblock0 PASS !!!
============================================================

==============================================
############ Finish SPI all blocks  Test PASS !!!
```

2.5.5 I2C Test

Step1: Power on UBC-330 and boot into OS

Step2: System should detect all devices with I2C interface controlled.

```
root@am335x-adv:/unit_tests# ./AutoRun_i2c.sh
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- UU -- --
30: UU UU UU UU UU UU UU UU -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: UU -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- UU -- -- -- --
=====I2C test Pass!=====
```

2.5.6 CAN Test

Step1: Connect one UBC-330 CAN Port CAN1_D+ /CAN1_D- and GND with another UBC-330.

Step2: Run program to transmit data between two UBC-330 CAN ports.

```
root@am335x-adv:/unit_tests# ./AutoRun_CAN.sh
Sun Sep 14 01:48:57 UTC 2014
interface = can0, family = 29, type = 3, proto = 1
wait for data
interface = can0, family = 29, type = 3, proto = 1
======CAN Pass=======
Sun Sep 14 01:49:01 UTC 2014
interface = can0, family = 29, type = 3, proto = 1
wait for data
interface = can0, family = 29, type = 3, proto = 1
======CAN Pass=======
```

2.5.7 GPIO Test

Step1: Power on UBC-330 and boot into OS

Step2: Short GPI0 to GPO0, GPI1 to GPO1, GPI2 to GPO2, GPI3 to GPO3.

Step3: Run program to test GPIO read/write.

```
root@am335x-adv:/unit_tests# ./AutoRun_gpio.sh
```

GPIO200 direction is:
in
GPIO201 direction is:
in
GPIO202 direction is:
in
GPIO203 direction is:
in
GPIO204 direction is:
out
GPIO205 direction is:
out
GPIO206 direction is:
out
GPIO207 direction is:
out
GPIO test PASS!

2.5.8 LAN Test

UBC-330 sets DHCP as defaul network portocal.

2.5.8.1 eth0 test

Step1:Connect UBC-330 eth0 port with a host computer.

Step2:Config UBC-330 eth0 IP as 192.168.1.2.meanwhile,config the host computer IP as 192.168.1.1

```
root@am335x-adv:~# ifconfig eth0 192.168.1.2
root@am335x-adv:~# ifconfig eth0
```

eth0 Link encap:Ethernet HWaddr 78:A5:04:DD:E1:0A
inet addr:192.168.1.2 Bcast:192.168.1.255 Mask:255.255.255.0
UP BROADCAST RUNNING ALLMULTI MULTICAST MTU:1500 Metric:1
RX packets:160 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:23334 (22.7 KiB) TX bytes:0 (0.0 B)

Step3: we can use below command to see if we can get any response from the host

```
root@am335x-adv:~# ping 192.168.1.1
```

PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: seq=0 ttl=64 time=0.384 ms
64 bytes from 192.168.1.1: seq=1 ttl=64 time=0.159 ms
64 bytes from 192.168.1.1: seq=2 ttl=64 time=0.110 ms

64 bytes from 192.168.1.1: seq=3 ttl=64 time=0.102 ms
64 bytes from 192.168.1.1: seq=4 ttl=64 time=0.208 ms
64 bytes from 192.168.1.1: seq=5 ttl=64 time=0.135 ms
64 bytes from 192.168.1.1: seq=6 ttl=64 time=0.186 ms
64 bytes from 192.168.1.1: seq=7 ttl=64 time=0.151 ms
64 bytes from 192.168.1.1: seq=8 ttl=64 time=0.091 ms
64 bytes from 192.168.1.1: seq=9 ttl=64 time=0.203 ms
64 bytes from 192.168.1.1: seq=10 ttl=64 time=0.111 ms
64 bytes from 192.168.1.1: seq=11 ttl=64 time=0.105 ms
64 bytes from 192.168.1.1: seq=12 ttl=64 time=0.098 ms
64 bytes from 192.168.1.1: seq=13 ttl=64 time=0.091 ms
64 bytes from 192.168.1.1: seq=14 ttl=64 time=0.187 ms
64 bytes from 192.168.1.1: seq=15 ttl=64 time=0.123 ms

2.5.8.2 eth1 test

Step1:connect UBC-330 eth1port with a host computer.
Step2:config UBC-330 eth1 IP as 192.168.1.3.

```
root@am335x-adv:~# ifconfig eth1 192.168.1.3
root@am335x-adv:~# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 78:A5:04:DD:E1:0C
          inet addr:192.168.1.3  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:41 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5035 (4.9 KiB)  TX bytes:0 (0.0 B)
```

Step3: we can use below command to see if we can get any response from the host.

```
root@am335x-adv:~# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: seq=0 ttl=64 time=0.373 ms
64 bytes from 192.168.1.1: seq=1 ttl=64 time=0.208 ms
64 bytes from 192.168.1.1: seq=2 ttl=64 time=0.234 ms
64 bytes from 192.168.1.1: seq=3 ttl=64 time=0.115 ms
64 bytes from 192.168.1.1: seq=4 ttl=64 time=0.122 ms
64 bytes from 192.168.1.1: seq=5 ttl=64 time=0.107 ms
```

2.5.9 RS232 Test

There are 6 UART supported by RSB-4220. /dev/ttyo0 is reserved for RSB-4220 debug port (RSB-4220 CN5), the rest UART ports could be applied by user.

2.5.9.1 UART1 to UART5 RS232 test

Short TX with RX of UART1 to UART5.

```
root@am335x-adv:/unit_tests# ./AutoRun_uart232
```

```
=====test rs232!=====
rs232 number: 5
/dev/ttyO1 PASS!
/dev/ttyO2 PASS!
/dev/ttyO3 PASS!
/dev/ttyO4 PASS!
/dev/ttyO5 PASS!


+--------------------+
| [RS232] Test Pass! |
```

Chapter 3

SW Introduction

This chapter gives SW information such as BSP , Image and so on.

3.1 Introduction

UBC-330 platform is an embedded system with Linux kernel 3.2.0 inside. It contains all system-required shell commands and drivers ready for UBC-330 platform. We do not offer IDE developing environment in UBC-330 BSP, users can evaluate and develop under Ubuntu 12.04 LTS environment.

There are three major boot components for Linux, "u-boot.img", "uImage" and "File System". The "u-boot.img" is for initializing peripheral hardware parameters; the "uImage" 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 configuration and development of UBC-330 to you, so that you can develop your own application(s) efficiently.

UBC-330 application development is only in Linux host PC and you cannot develop your application on Windows/Android host PC. For now the official supported host version is Ubuntu 12.04 LTS, host PC in any other Linux version may have compatibility issue. In this case, we strongly recommend to have Ubuntu 12.04 LTS installed to your host PC before start UBC-330 evaluation/development.

3.2 Package Content

We would offer you two different kinds of Linux package for UBC-330. One is pre-built system image for system recovery another is source code package (BSP).

3.2.1 Pre-built System Image

You are able to find the pre-built image FA30LIVxxxx_yyyy-mm-dd.tar.gz from UBC-330 evaluation kit DVD image downloaded from Advantech website. UBC-330 supports booting from SD card so you can extract the image to SD card then dump the image file to onboard eMMC to complete system recovery. For more detail, please refer to section 3.6 Create a Linux System Boot media.

3.2.2 Surce Code Package

UBC-330 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-330 source code package is composed of six main folders: "cross_compiler", "document", "image", "package", "scripts", and "source".

Note! *UBC-330 source code package (BSP) is Advantech's Intellectual Property. If you need to access this package, please contact your Advantech support window.*

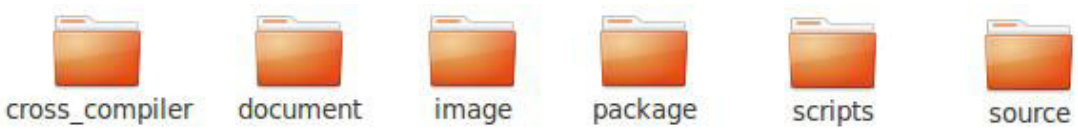


Figure 3.1 Source code package structure

The description of UBC-330 BSP package contents:

- "cross_compiler" This folder contains source code for cross compiler.
- "document" This folder contains user guide.
- "image" This folder contains the uImage, u-boot.img
- "image/rootfs" This folder contains Linux root file system
- "package" This folder contains source code provided by TI without any modification
- "scripts" 'This folder contains scripts for configure system and compile images automatically.
- "source" 'This folder contains source code owned by Advantech

3.2.2.1 cross_compiler

You can use the cross compiler tool chain to compile the uImage and related applications. (gcc version is 4.7.3 20130226)

3.2.3 document

User guide of how to setup up the environment of development

3.2.3.1 image

This folder includes uImage & u-boot.img.

3.2.3.2 image/rootfs

Linux adopts Hierarchical File System (HFS), image/rootfs is the Linux file system in highest level of the tree structure. image/rootfs is just like the trunk of the tree. Its sub-directories are the branches and the files in these directories are the leaves of the tree. image/rootfs contains all subdirectories and files used in the file system, that's why it is called the root of the whole file system.

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!

- 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.

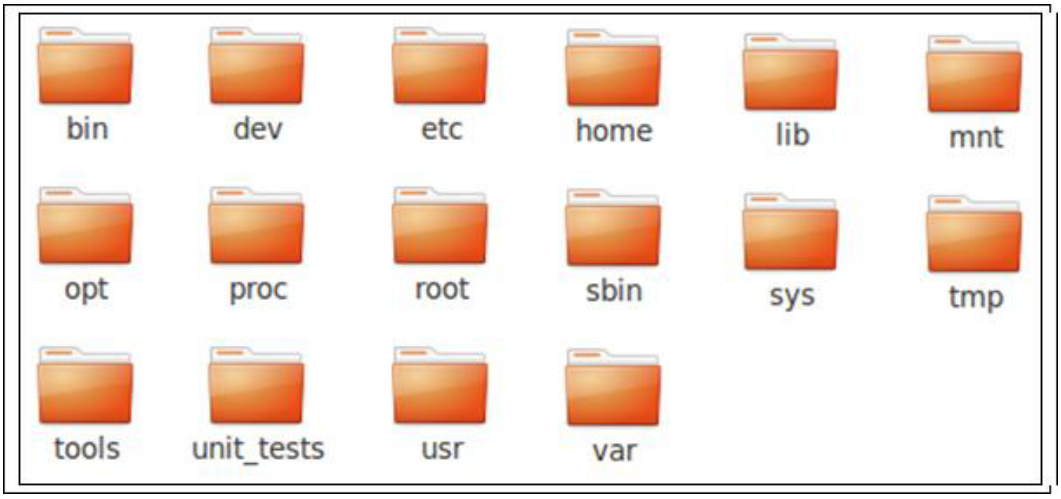


Figure 3.2 image\rootfs

3.2.3.3 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 "uImage" and copy the "uImage" to "image" folder after building.
- mksd-linux.sh A script to setup up a bootable SD card if users build their images
- mkinand-linux.sh A script to go to SD card Linux O.S. then burn O.S to eMMC flash

3.2.3.4 source

This folder contains sub-directories "linux-3.2.0-psp04.06.00.11" and "u-boot-2013.01.01-psp06.00.00.00". They are the source codes of the Linux kernel and U-boot.

Linux is OS that is including true multitasking, virtual memory, shared libraries, demand loading, shared copy-on-write executables, proper memory management, and multitask networking.

Linux is easily portable to most general-purpose 32-bit architectures as long as they have a paged memory management unit (PMMU) and a port of the GNU C compiler (gcc) (part of The GNU Compiler Collection, GCC). Linux has also been ported to a number of architectures without a PMMU, although functionality is then obviously somewhat limited. Linux has also been ported to itself.

The main sub-directories under "linux-3.2.0" are listed as following:

- 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 plenty of documentations or materials available on Internet and also could be obtained from books and magazines, you can easily find the answers for both Linux-specific and general UNIX questions.

There are also various README files in ./source/ linux-3.2.0-psp04.06.00.11/Documentation, you can find the kernel-specified installations and notes for drivers. You can refer to ./source/ linux-3.2.0-psp04.06.00.11/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 12.04 LTS developing environment. Please install the Ubuntu 12.04 LTS at your PC/NB in advance.

When you obtain the UBC-330 Linux source code package, please refer to following instructions to extract to your developing environment:

1) Copy "335XLBVxxxx_yyyy-mm-dd.bin" package to /root/.
2) Start your "Terminal" on Ubuntu 12.04 LTS.
3) **$sudo su** (Change to "root" authority)
4) Input user password
5) **#cd /root/**
6) **#chmod a+x** 335XLBVxxxx_yyyy-mm-dd.bin
7) **#.**/335XLBVxxxx_yyyy-mm-dd.bin
8) **Input "yes"**
9) Then you can see folder "335XLBVxxxx_yyyy-mm-dd" on /root/.

Note! *xxxx is the version number, yyyy is the year, mm is mouth, dd is the day. For example: 335XLBV1010_2014-10-01.*

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 12.04 LTS.
2) **$sudo su** (Change to "root" authority)
3) Input user password
4) **#cd /root/335XLBVxxxx_yyyy-mm-dd/scripts/**
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.

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

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 12.04 LTS..
2) **$sudo su** (Change to "root" authority)
3) Input user password.
4) **#cd /root/335XLBVxxxx_yyyy-mm-dd/scripts/**
5) **#. setenv.sh** (To configure the developing environment automatically)
6) **#./cfg_uboot am335x_rsb4220** (To set the u-boot configuration automatically)
7) **#./mk_uboot.sh** (Start to build the u-boot)
8) Then you can see u-boot.img is being built and located in ../image.

3.4.2 Build Linux Kernel Image

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

1) Open "Terminal" on Ubuntu 12.04 LTS.
2) **$sudo su** (Change to "root" authority)
3) Input user password.
4) **#cd /root/335XLBVxxxx_yyyy-mm-dd/scripts/**
5) **#. setenv.sh** (To configure the developing environment automatically)
6) **#./cfg_kernel.sh am335x_rsb4220_defconfig**
(To set the uImage configuration automatically)
7) **#./mk_kernel.sh** (Start to build the uImage)
8) Then you can see uImage is being built and located in ../image.

3.4.3 Build Log

You can find the build log from folder "./335XLBVxxxx_yyyy-mm-dd/". 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 Kernel 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.

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 12.04 LTS.
2) **$sudo su** (Change to "root" authority)
3) Input user password.
4) **#cd /root/335XLBVxxxx_yyyy-mm-dd/scripts/**
5) **#. setenv.sh** (To configure the developing environment automatically)
6) **#./cfg_kernel.sh am335x_rsb4220_defconfig**
7) **#./cfg_kernel.sh menuconfig**
8) Then you will see a GUI screen (Linux Kernel Configuration) as below:

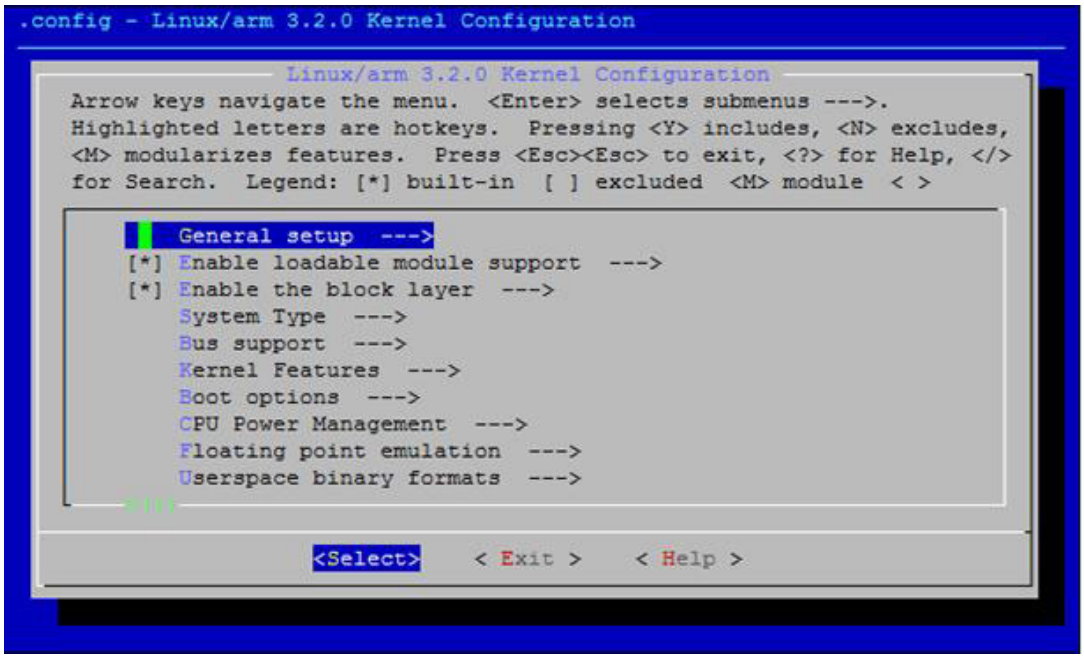


Figure 3.3 Linux Kernel Configuration

9) 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.

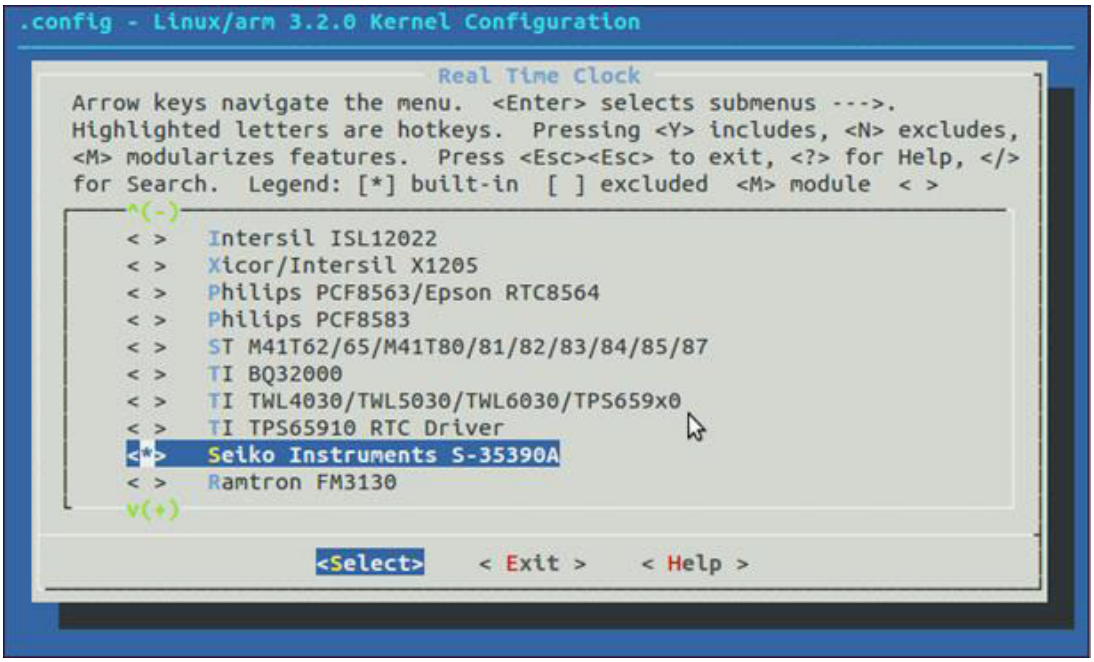


Figure 3.4 Selecting Seiko Instruments S-35390A

10) Change directory to "source/ linux-3.2.0-psp04.06.00.11/arch/arm/mach-omap2", edit the "board-rsb4220.h" and "board-advantech.c".

Please add below codes to source/ linux-3.2.0-psp04.06.00.11/arch/arm/mach-mx6/board-rsb4220.h:

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

Please add below codes to
source/ linux-3.2.0-psp04.06.00.11/arch/arm/mach-omap2/board-advantech.c

```
omap_register_i2c_bus(1, 100, am335x_i2c0_boardinfo,
                      ARRAY_SIZE(am335x_i2c0_boardinfo));
```

11) 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.*

3.6 Create a Linux System Boot Media

UBC-330 supports boot from SD card or onboard flash. This section will guide you how to build an image for UBC-330 Linux system boot media.

3.6.1 Storage Information (eMMC/SD card)

The storages devices name as following:

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

3.6.2 Create a Linux System SD Card

3.6.2.1 From Pre-built System Image

You are able to find the pre-built image from Advantech website. Please follow below steps to create a SD card for boot up.

1) Copy "4420LIVxxxx_yyyy-mm-dd.tar.gz" package to your /root/.
2) Open "Terminal" on Ubuntu 12.04 LTS..
3) **$sudo su** (Change to "root" authority)
4) Input your password.
5) **#cd /root/**
6) **#tar xzvf 4220LIVxxxx_yyyy-mm-dd.tar.gz (Unzip files)**
7) Insert one SD card to your developing computer
8) Check the SD card location, like /dev/sdb
9) **#cd ./4220LIVxxxx_yyyy-mm-dd/scripts**
10) **#./mksd-linux.sh /dev/sdb**
11) Type "y" (Start to copy files, wait until it shows [Done])

Then insert the Linux system SD card to UBC-330, it will boot up with Linux environment.

3.6.3 From Source Code Package

When you receive the UBC-330 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 12.04 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/sdb
6) **#cd /root/335XLBVxxxx_yyyy-mm-dd/scripts**
7) **#./mksd-linux.sh /dev/sdb**
8) Type "y" (Start to copy files, wait until it shows [Done])

Then insert the Linux system SD card to UBC-330 SD card slot (SD1), it will boot up with Linux environment.

3.6.4 Boot from Onboard Flash

If you've already had a Linux system SD card, you can refer following steps to copy the content to onboard flash and then boot from onboard flash. Advantech also provide 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-330 and connect serial console.
3) On UBC-330 platform, type **#root** (Login)
4) On UBC-330 platform, type **#cd /mk_inand**
5) On UBC-330 platform, type **#./mkinand-linux.sh /dev/mmcblk1**
6) Power off and remove this SD card.

Then you can boot from onboard flash without SD card.

3.7 Debug Message

UBC-330 can connect to a host PC (Linux or Windows) by using console cable and debug port adapter. In order to communicate with host PC, serial communication program such as HyperTerminal, Tera Term or PuTTY is must required. Below is the detail instruction of how to set up serial console, a "HyperTerminal" on a Windows host:

1. Connect UBC-330 to your Windows PC by using serial cable, debug port adapter and console cable.
2. Open HyperTerminal on your Windows PC, and select the settings as shown in Figure 3.5.
3. Power up the board. The bootloader prompt is displayed on the terminal screen.

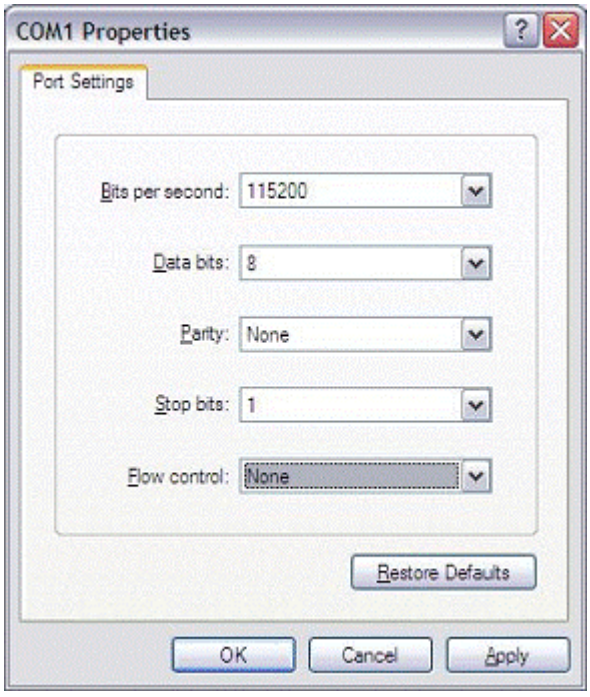


Figure 3.5 HyperTerminal Settings for Serial Console Setup

3.8 Linux System Configuration and Use

3.8.1 Service Configuration

UBC-330 has built five common network services: tftp service, ftp service, ssh service, telnet service and http service.

3.8.1.1 Tftp Server

When boot up the UBC-330, the tftp service is already started by default and the tftp server's working directory is /tftpboot. You need execute "chmod 777 /tftpboot" on UBC-330 to let the tftp server work. Then, user can tftp to UBC-330 by tftp client in host PC. Use command to get and put file like this:

```
hostPC$ tftp  TARGET_SYSTEM_IP
tftp>get file1
tftp>put file2
```

Note! *Command "get file1" is to download file1 from tftp server. File "file1" must exist under the directory /tftpboot on UBC-330;*

Command "put file2" is to upload file2 to tftp server. If put file2 success, file2 will be put to directory /tftpboot on UBC-330.

The service start command is:

```
root@am335x-adv?/ # /etc/init.d/tftpd start
```

And the stop is:

```
root@am335x-adv?/ # /etc/init.d/tftpd stop
```

3.8.1.2 ftp server

The ftp server on UBC-330 is vsftpd and you should manually start it using flowing command:

```
root@am335x-adv?/ # /etc/init.d/vsftpd start
```

While, the stoping command is:

```
root@am335x-adv?/ # /etc/init.d/vsftpd stop
```

Note! *After start the ftp server. You had to manually add user ftp:*

```
root@am335x-adv?/ # adduser ftp
root@am335x-adv?/ # chown root?root /home/ftp/
```

Then you can ftp the UBC-330 using user ftp.

3.8.1.3 ssh server

When boot up the UBC-330, the ssh service is already started by default. You can run the following command on your host PC to login the UBC-330:

```
hostPC$ sudo ssh -l  root  TARGET_SYSTEM_IP
```

The service start command is:

```
root@am335x-adv?/ # /etc/init.d/dropbear start
```

And the stop is:

```
root@am335x-adv?/ # /etc/init.d/dropbear stop
```

3.8.1.4 telnet Server

When boot up the UBC-330, the telnet service is already started by default. You can run the following command on your host PC to login the UBC-330:

```
hostPC$ sudo telnet TARGET_SYSTEM_IP
```

The service start command is:

```
root@am335x-adv?/ # /etc/init.d/telnetd start
```

And the stop is:

```
root@am335x-adv?/ # /etc/init.d/telnetd stop
```

3.8.1.5 http Server

We support an embedded web server name lighttpd and the matrix gui is based on it.

The service start command is:

```
root@am335x-adv:/ # /etc/init.d/lighttpd start
```

And the stop is:

```
root@am335x-adv:/ # /etc/init.d/lighttpd stop
```

3.8.2 Network configuration

You can also do the network configuration by console using telnet. Run the following command on UBC-330:

Get IP by DHCP:

```
advantech# /etc/init.d/dhcpc eth0 start
advantech# /etc/init.d/dhcpc eth1 start
```

If you want to reserve the setting after rebooting the device, set as below

```
advantech#  echo  "/etc/init.d/dhcpc  eth0  start"  >  /etc/adv.d/
netcfg.eth0
advantech#  echo  "/etc/init.d/dhcpc  eth1  start"  >  /etc/adv.d/
netcfg.eth1
```

Set static IP:

Stop the DHCP process

```
advantech# /etc/init.d/dhcpc eth0 stop
advantech# /etc/init.d/dhcpc eth1 stop
```

Set the static IP as below.

```
advantech# /sbin/ifconfig eth0 172.21.73.191 netmask 255.255.255.0
advantech# /sbin/route add default gw 172.21.73.253 eth0
advantech# echo 'nameserver 172.21.128.251' >> /etc/resolv.conf
advantech# /sbin/ifconfig eth1 192.168.3.102 netmask 255.255.255.0
advantech# /sbin/route add default gw … eth1
advantech# echo 'nameserver 172.21.128.251' >> /etc/resolv.conf
```

If you want to reserve the setting after rebooting the device, set as below

```
advantech#  echo  "/sbin/ifconfig  eth0  172.21.73.191  netmask
255.255.255.0; /sbin/route add default gw 172.21.73.253 eth0;echo
'nameserver  172.21.128.251'  >  /etc/resolv.conf;"  >  /etc/adv.d/
netcfg.eth0
advantech#  echo  "/sbin/ifconfig  eth1  192.168.3.102  netmask
255.255.255.0; /sbin/route add default gw ... eth1;echo 'nameserver
172.21.128.251'  >  /etc/resolv.conf;" > /etc/adv.d/netcfg.eth1
```

Note! *The IP address in above should be replaced according to user's the requirement.*

For examples: IP 172.21.128.251 in above is the DNS server's IP, user should replace it with the correct DNS IP address.

3.8.3 Date/Time Configuration*

You can use data, hwclock shell command to modify the system time.

3.8.4 About System

This is a way to get version info under console. You can use "version" command to achieve this as flowing:

```
root@am335x-adv：~# version
Bsp version:     UBC-330 V1.000
Device name?    UBC-330
Release date：    2014-10-09
Kernel version：   3.2.0
```

3.8.5 Add a Startup items when boot

1. Remove a Startup items?

   update-rc.d [-n] [-f] [-r <root>] <basename> remove

   basename is your service script name

   eg. update-rc.d -f matrix-gui-2.0 remove

2. Add a Startup items?

   Firstly,You must ensure that the service script is exists,then run the flowing command:

   update-rc.d [-n] [-r <root>] [-s] <basename> start|stop NN runlvl [runlvl] [...] .

   start|stop : when system start /shutdown the basename will run automatically

   NN: 0~99

   runlvl: UBC-330 runlevel is 5(default);

   eg. update-rc.d networking start 40 5 .

   then you can find the S40networking in rc5.d directory;

3.8.6 Package online install

3.8.6.1 OPKG Package Manager

Opkg is a lightweight package management system. It is written in C and resembles apt/dpkg/yum in operation. It is intended for use on embedded Linux devices and is used in this capacity in the OpenEmbedded and OpenWrt projects.

Advantech Embedded Linux for UBC-330 has built-in OPKG package manager, with this tool you can install most of the required software online, and manage them, such as uninstall, upgrades and so on.

3.8.6.2 Installation New Software package

If you want to install a software which is not exist in the current OS, you should follow the steps below

Step 1: Update the online software source:

```
advantech# opkg update
```

Step 2: Search whether the software source server has the software you need.

```
advantech# opkg list | grep package
```

Note! *Package is the keywords of the software name, for example, you want to search an ftp server, and the package should be 'ftp'.*

Step 3: Find the full name of the software you need in the search result list. And install it by following command:

advantech# opkg install packagename

3.8.6.3 More about OPKG

More about use and development of OPKG, Please refer to the project website of OPKG:

https://code.google.com/p/opkg/

3.9 Development Guide and Reference

3.9.1 Development of C/C++ Programs

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

1. Open "Terminal" on Ubuntu 12.04 LTS and Change to "root":

```
$ sudo su
```

Type user password.

2. Create the develop environment using flowing command:

```
#source    /usr/local/cross_compiler/linux-devkit/environment-setup
# cd /root//4220LBVxxxx_yyyy_mm_dd/source
# mkdir helloworld
# cd helloworld
# gedit hellowrold.c
```

3. Edit the helloworld.c with the following source code:

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

Save the file and exit.

4. Compile helloworld.c using flowing command:

```
# $CC -o helloworld helloworld.c
```

Then you can see "helloworld" in current directory.

5. Run the executable file helloworld on the UBC-330.

Insert the Linux system SD card to your developing computer.

cp helloworld /media/rootfs/tool

Note. /media/rootfs is the mounted point of your Linux system SD card

Remove this SD card and insert it to UBC-330, then open serial console.

On UBC-330 platform, type #root (Login)

On UBC-330 platform, type #cd /tool

On UBC-330 platform, type #./helloworld

Now you should be able to see "Hello World!" shown on UBC-330.

3.9.2 Demo program source code

3.9.2.1 Serial Port Programming

Please refer to <BSP_PATH>/source/demo/uart

It is an example of sending and receiving data via the serial port.

Receiving data:

```
# ./uart_ctrl read /dev/ttyO1
```

Sending data:

```
# ./uart_ctrl write /dev/ttyO2
```

Before using your program of serial port, please ensure that your serial port is in 232/422/485 mode.

User can reference the uart demo source code to develop the uart application.

3.9.2.2 Watchdog Programming

UBC-330 support hardware watchdog, the watchdog API is follow posix standards. The valid timeout value is from 1 to 6553 seconds, if the timeout value to set is not in this scope, driver will set timeout value to default value (60 seconds).

Sample C code:

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <signal.h>
#include <linux/watchdog.h>
int fd;
int main(int argc, const char *argv[]) {
        int timeout = 10;
/*open watchdog device, the watchdog device node is /dev/watchdog */
fd=open("/dev/watchdog",O_WRONLY);

/*set timeout to 10 seconds*/
/*when set timeout value, the watchdog driver will enable the watchdog automatically.*/
ioctl (fd, WDIOC_SETTIMEOUT, &timeout);
while (1)
{
                /*feed the watchdog every 5 seconds*/
                /*when call this funtion to feed watchdog, the watchdog will reset its internal timer so it doesn't trigger the board reset. If do not feed the watchdog more than 10 seconds, the watchdog will timeout and the board will reset.*/
        ioctl( fd, WDIOC_KEEPALIVE, NULL );
        sleep (5);
}
close (fd);
}
```

Here are some other APIs for watchdog.

Disable the watchdog timer sample code:

/*if user want to disable the watchdog before timeout, call the following ioctl function*/

int i_dis = WDIOS_DISABLECARD;

ioctl(fd, WDIOC_SETOPTIONS, &i_dis);

Enable the watchdog timer sample code:

/*if user want to enable the watchdog again before timeout when it is disabled, call the following ioctl function. */

int i_en = WDIOS_ENABLECARD;

ioctl(fd, WDIOC_SETOPTIONS, & i_en);

Get the current timeout value:

/*get the current timeout value the driver used*/

int timeout = 0;

ioctl (fd, WDIOC_GETTIMEOUT, &timeout);

Please refer to <BSP_PATH>/source/demo/watchdog folder to get more information.

3.9.2.3 GPIO Programming

UBC-330 has 8 gpios. Please refer to <BSP_PATH>/source/demo/gpio

Usage:

./gpio 200 out 1

Note.

"200" means gpio0, and so 200-207 corresponds to gpio0-gpio7

"out" means output

"1" is the value set to the corresponding gpio port

3.9.2.4 Can Programming

Please refer to <BSP_PATH>/source/demo/can_test

Note! *Can sending data sample c code,please refer to can_write.c*

Can receiving data sample c code,please refer to can_read.c

3.9.2.5 Brightness Programming

Please refer to <BSP_PATH>/source/demo/brightness

Brightness driver provide the sys interface, so we can set and get brightness value throught the sys file:

/sys/class/backlight/pwm-backlight/brightness

You can set brightness using flowing command:

```
# echo "20" > /sys/class/backlight/pwm-backlight/brightness
```

Note! *The value should be between 1-100.*

You can get current brightness value using flowing command:

```
# cat /sys/class/backlight/pwm-backlight/brightness
```

Chapter 4

Services

4.1 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 of 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.

4.2 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

On the other hand, you can reach our service team through below website; 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

4.3 Global Service Policy

4.3.1 Warranty Policy

Below is the warranty policy of Advantech products:

4.3.1.1 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.

4.3.1.2 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.

4.3.1.3 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 lighting strike, flood, earthquake, etc.
- Product updates/upgrades and tests upon the request of customers who are without warranty.

4.3.2 Repair Process

4.3.2.1 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 ADVANTECH'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.

4.3.2.2 Returning the Product for Repair

It's possible customers can save time and meet end-user requirements by returning defective products to an y 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 s end cards without accessories (manuals, cables, etc.). Remove any unnecessary components from the card, 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. We strongly recommend adding 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 unrepaired 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.

4.3.2.3 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 lighting 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.

4.3.2.4 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.

4.3.2.5 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.

4.3.2.6 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