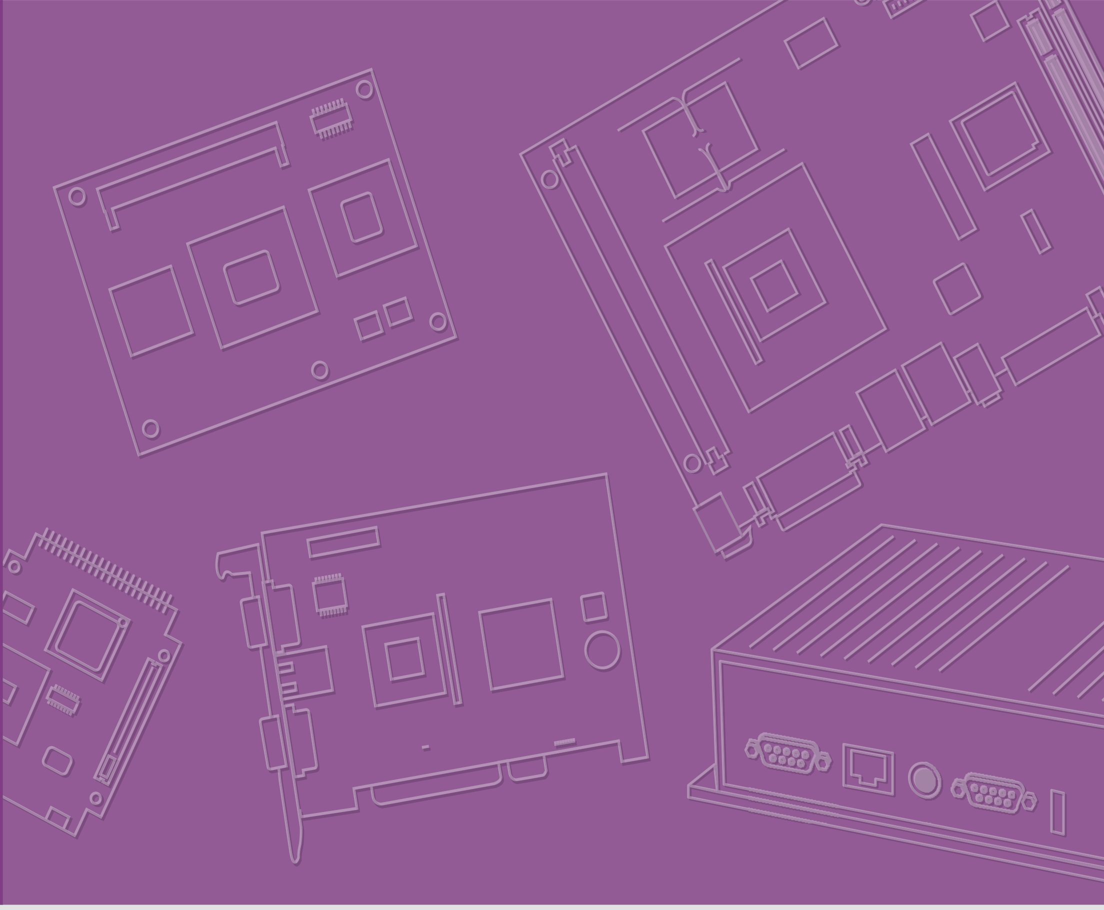
User Manual



ROM-5420

RISC-based SMARC Module with Freescale i.MX6 ARM® Cortex™ A9 Processor

Copyright

Acknowledgements

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.

Part No. 2006542010 Edition 1

Printed in Taiwan March 2014

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.

Warnings, Cautions and Notes

Warning! *Warnings indicate conditions that, if left uncorrected, 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.*

Safety Precaution - Static Electricity

Follow these simple precautions to protect yourself from harm and the products from damage.

- To avoid electrical shock, always disconnect the power from your PC chassis before you work on it. Don't touch any components on the CPU card or other cards while the PC is on.
- Take care to avoid electrostatic discharge (ESD) before making any configuration changes. Use a static-free workstation, grounding strap, or take other precautions. The ESD that may occur as you connect a jumper or install a card can damage sensitive electronic components.

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 ROM-5420
- 4 Screws for ROM-5420
- 1 China RoHs Notice

Optional Accessories

Part No.	**Description**
1960063090N001	Heat spreader for ROM-5420
1960063089N001	Semi-heat sink for ROM-5420
1930005215	Screw for heat spreader
1930004835	Screw for Semi-heat sink
9696ED2000E	Debug adapter board
1700022373-01	Debug port cable for ROM-5420

Development Board

Part No.	**Description**
ROM-DB5900-SWA1E	Development carrier board for SMARC Rev.1.0

For more information please refer to "Advantech Baseboard Check List" and "Evaluation Board Reference Schematic".

Ordering Information

Model Number Description

- **Commercial grade**

Part No.	**ROM-5420CD-MDA1E**
CPU	Freescale i.MX6 Dual 1 GHz
Memory	1 GB
eMMC	4 GB
Camera Input	1
Parallel RGB	Yes
HDMI	Yes
LVDS	Yes
PCIe	1
USB	1 Host, 1 OTG
Audio	Yes
GPIO	12
Serial	4
CAN	2
I2C	5
Operation Temp.	0 ~ 60° C

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

Chapter 1

Product Overview

This chapter briefly introduces ROM-5420 platform.

Sections include:

■ Introduction

■ Specification

1.1 Introduction

ROM-5420 adopts Freescale i.MX6 Dual Core Processor - ARM® Cortex™ A9 architecture as its SoC solution. The main features of this platform are followed by SMARC 1.0 standard, with a heatsink-less, compact, reliable & great power management. Therefore, ROM-5420 platform is suitable for following applications:

- HMI (Human Machine Interface)
- Portable devices
- Fleet management / Navigation
- Industrial data collector

And the main features of Freescale i.MX6 processors are shown as following:

- ARM Cortex™-A9 high performance processor, dual core 1 GHz
- Supports OpenGL ES 2.0 and OpenVG™ 1.1 hardware accelerators, full HD 1080p video codec
- Freescale Smart Speed™ Technology support low power consumption
- Capabilities of I/O expansion: UART(4), Single LVDS, Audio, USB Host, USB OTG, Gigabit Ethernet, SD, SATA, GPIO(12), I2C(5), SPI(4), I2S, CAN bus with 5 V level(2), PCIe signal inside and parallel RGB/HDMI support
- Supports SATA storage interface and CAN bus for vehicle application
- Supports Linux 3.0.35
- Support working temperature 0 ~ 60° C
- Specialized heat spreader and semi-heat sink design for ROM-5420

1.2 Product Features

Compatible Module		SMARC v1.0
Processor System	CPU	Freescale i.MX6 Dual Core 1.0 GHz
Memory	Technology	DDR3 1066 MHz
	Capacity	On-board DDR3 1 GB
	Flash	4 GB eMMC for O.S. and 4 MB NOR Flash for Advantech boot loader
Graphic	Graphics Engine	3 GPUs. OpenGL ES 2.0 for 3D, BitBlt for 2D and OpenVG 1.1
	H/W Video Codec	Decoder: MPEG-4 ASP, H.264 HP, H.263, MPEG-2 MP, MJPEG BP Encoder: MPEG-4 SP, H.264 BP, H.263, MJPEG BP
	HDMI	1 HDMI TypeA
	LVDS	1 single 24-bit LVDS
	VGA	1 D-Sub 15 with female connector
Ethernet	Chipset	Freescale i.MX6 Dual integrated RGMII
	Speed	1 10/100/1000 Mbps, Max. 400Mbps
Watch Dog Timer		256 Level timer interval, from 0 ~ 128 sec, multi-option S/W watchdog timer
I/O	PCIe	1 PCIe, 1x LAN
	SATA	1 SATA II
	USB	1 USB 2.0, 1 USB2.0 OTG
	Audio	HD Audio with I2S
	SDIO	4-bit SDIO
	Serial Port	4 UART (2 x 2 wire, 2x 4 wire w/ 3.3 V)
	SPI	4 SPI
	CAN	2 CAN bus 2.0B
	GPIO	12 GPIO
	I2C	*5 I2C
O.S		Linux 3.0.35
Power	Supply Voltage	+3 ~ 5.25 V
Environment	Operation	0 ~ 60° C
	Operating Humidity	0% ~ 90% relative humidity, non-condensing
Physical Characteristics	Dimensions (WxD)	82 x 50 mm
Camera Input	Dimensions (WxD)	1 MIPI v1.0, 4x Lane

*4 I2C has been reserved for power management, camera, LCD display ID and HDMI support. You can use it if you do not need those features.

1.3 Mechanical Specifications

- **Dimensions:** SMARC form factor size, 82 mm (W) x 50 mm(D)
- **Height on Top:** Under 3.0 mm base on SPEC definition (without heatsink)
- **Height on Bottom:** Under 1.3 mm base on SPEC definition
- **Heat Spreader Dimensions:** 42 mm x 82 mm x 6 mm (L x W x H)
- **Semi-Heat sink Dimensions:** 42 mm x 82 mm x 13 mm (L x W x H)

1.4 Electrical Specifications

- **Power supply Voltage:**
  – Voltage requirements: +3 ~ 5.25 V
- **Power supply Current:**

+5 V	Kernel idle	Maximum mode
ROM-5420CD-MDA1E on Linux	1.25 W	3.4 W

1.5 Environmental Specifications

- **Operating temperature:** 0 ~ 60° C (32~140° F)
  The operating temperature refers to the environmental temperature for the model.
- **Operating Humidity:** 0% ~ 90% relative humidity, non-condensing
- **Storage temperature:** -40~85° C
- **Storage Humidity:**
  – Relative humidity: 95% @ 60° C

Chapter 2

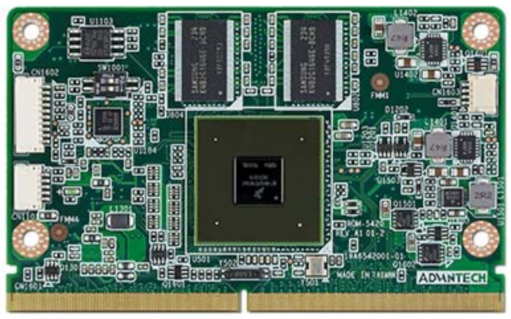
H/W Installation

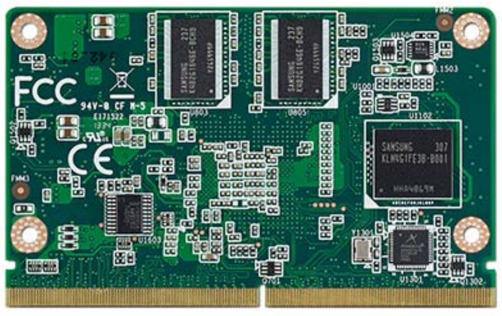
This chapter gives mechanical and connector information on the ROM-5420 CPU Computer on Module.

Sections include:

- Connector Information
- Mechanical Drawing

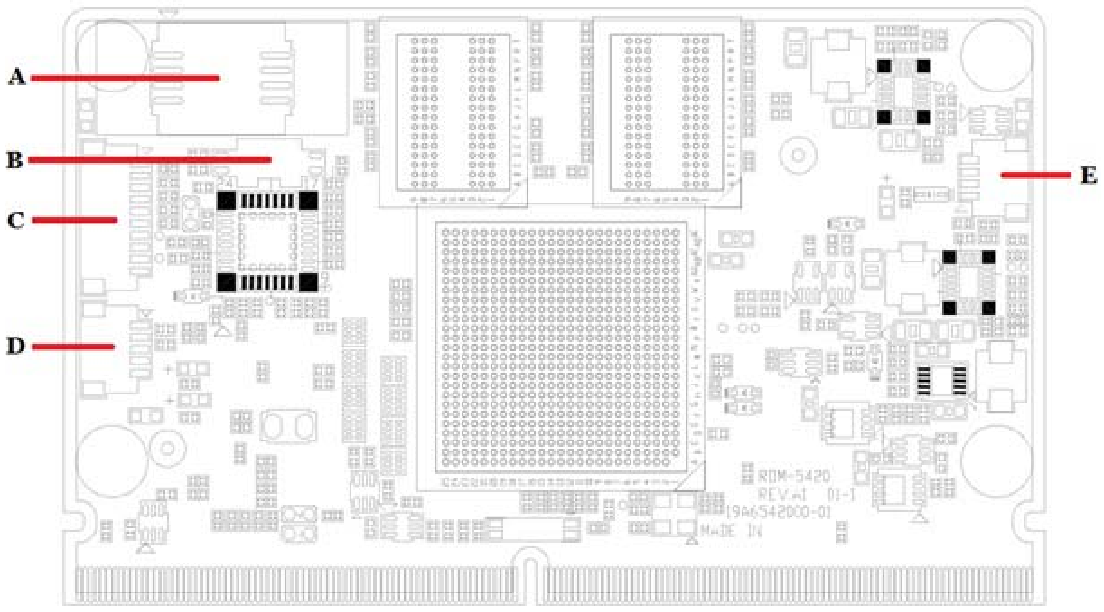
2.1 ROM-5420 Board Looks





2.2 Board Connectors

The board has four connectors that allow you to configure your system to your application.



2.2.1 Connector List

External IO Connector

Position	Description	
U1103	Flash ROM	A
SW1001	Boot selection	B
CN1602	JTAG connector	C
CN1101	MCU programming port	D
CN1603	Debug port	E

SW1001 (Boot selection)

Jumper	Mode	Jumper	Mode
1-ON 2-OFF	SPI-ROM (Default)	1-OFF 2-ON	SD (Reserved for recovery)

CN1101 (MCU programming port)

Pin	Signal	Pin	Signal
1	+3.3 V	2	SWDIO
3	SWCLK	4	MCU_JTAG_RESET
5	GND		

CN1602 (JTAG connector)

Pin	Signal	Pin	Signal
1	+3.3 V	2	JTAG_TRST#
3	JTAG_TMS	4	JTAG_TDO
5	JTAG_TDI	6	JTAG_TCK
7	JTAG_RTCK	8	GND
9	-	10	GND

CN1603 (Debug connector)

Pin	Signal	Pin	Signal
1	+3.3 V	2	UART1_TX
3	UART1_RX	4	GND

2.3 ROM-5420 Board Block Diagram

Below is the block diagram of ROM-5420.

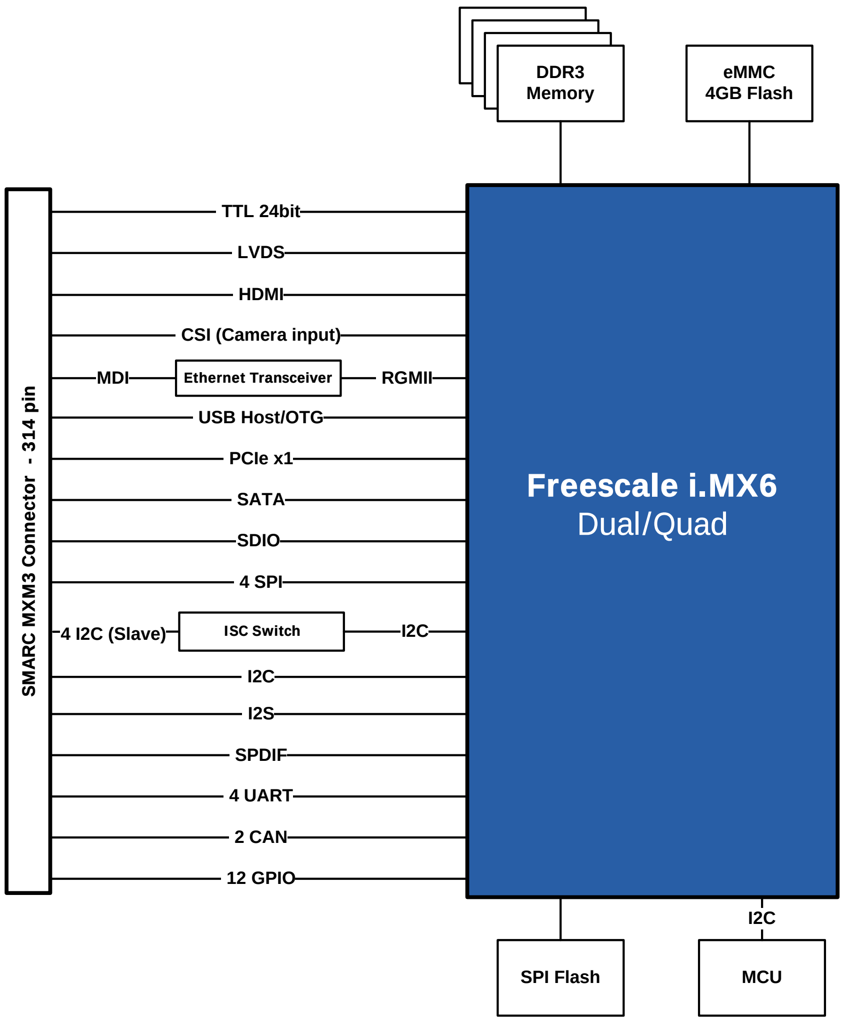


Figure 2.1 ROM-5420 Block Diagram

Chapter 3

Software Functionality

This chapter details the software programs on the ROM-5420 plat-form.

3.1 Test Tools

All test tools must be verified on ROM-5420 Evaluation kit, 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.

3.1.1 eMMC Test

Step1: 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 00  |................|
```

Step2: Write and check

3.1.2 SATA Test

Step1: Erase and check

```
#dd if=/dev/zero of=/dev/sda bs=1024 count=1 seek=25118
1+0 records in
1+0 records out

#hexdump -C /dev/sda -s 25720832 -n 16
01887800  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

Step2: Write and check

```
#echo -n "0123456789ABCDEF" | dd of=/dev/sda bs=1024 count=1 seek=25118
0+1 records in
0+1 records out

#hexdump -C /dev/sda -s 25720832 -n 16
01887800  30 31 32 33 34 35 36 37  38 39 41 42 43 44 45 46  |0123456789ABCDEF|
```

3.1.3 USB Test

Step 1: Insert USB flash disk then assure it is in ROM-5420 device list
Step 2: Erase and check

```
#dd if=/dev/sdb of=/dev/sdb bs=1024 count=1 seek=25118
1+0 records in
1+0 records out

#hexdump -C /dev/sdb -s 25720832 -n 16
01887800  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

Step 3: Write and check

```
#echo  -n  "0123456789ABCDEF"  |  dd  of=/dev/sdb  bs=1024  count=1
seek=25118
0+1 records in
0+1 records out

#hexdump -C /dev/sdb -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 USB flash disk. Please make sure there is no critical data in the USB flash disk being used for this test.*

3.1.4 SD Test

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

```
#ls /dev/mmcblk*

/dev/mmcblk0  /dev/mmcblk0boot0  /dev/mmcblk0boot1  /dev/mmcblk0p1
```

Step 2: Insert SD card to SD card slot 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
```

Step 3: Erase and check

```
#dd if=/dev/zero of=/dev/mmcblk1 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 00  |................|
```

Step 4: Write and check

```
#echo  -n  "0123456789ABCDEF"  |  dd  of=/dev/mmcblk1  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.*

3.1.5 GPIO Test

3.1.5.1 ROM-5420 GPIO Default Setting

ROM-DB5900	Linux OS /sys/class/gpio	Jumper Setting in ROM-DB5900
GPIO0	gpio1	CN24(1-2)
*GPIO1	-	CN26(2-3)CAM1_PWR#
GPIO2	gpio4	CN27(1-2)
*GPIO3	-	CN28(2-3)CAM1_RST#
GPIO4	gpio205	CN30(1-2)
GPIO5	gpio21	CN31(1-2)
GPIO6	gpio198	CN32(1-2)
GPIO7	gpio199	CN35(1-2)
GPIO8	gpio193	CN34(1-2)
GPIO9	gpio192	CN26(1-2)
GPIO10	gpio178	
GPIO11	gpio177	

* GPIO1 & GPIO3 has been reserved for camera input in ROM-5420. You can use the two GPIO ports if you do not need the camera feature.

```
#cd /sys/class/gpio
```

You can use “ls” to list all GPIO devices, and you should also see GPIO ports in above table.

3.1.5.2 Example of Testing GPIO

A. Set gpio177 GPI (in)

```
#echo in > ./gpio177/direction
#cat ./gpio177/direction
in
```

B. Set gpio178 GPO (out)

```
#echo out > ./gpio178/direction
#cat ./gpio178/direction
out
```

C. Set gpio177 GPO value “0”

```
#echo 0 > ./gpio177/value
```

D. Get gpio178 value

```
#cat ./gpio178/value
0
```

As you can see in below procedure A and B, we set gpio 177 as GPI and gpio 178 as GPO so once we send data out from gpio 178, it should be able to receive the same data from gpio 177.

3.1.6 LVDS/HDMI/VGA Test

3.1.6.1 Testing through gplay (for default single display)

Step 1: `#gplay /tools/Advantech.avi`

```
#gplay /tools/Advantech.avi
```

Step 2: Then you can see the video demo on the default display screen.

3.1.6.2 Testing through gst-launch (for multi-display)

If you'd like to do multiple display such as dual LVDS, VGA and HDMI output , you should set parameter in uboot first. Please refer to section 3.7.5.3 for more detail. Once the display method is set up, please follow below instruction run gst-launch to play video.

Step1: Turn ON the HDMI display, please type as below

```
#gst-launch    playbin2    uri=file:///tools/Advantech.avi    video-
sink="mfw_v4lsink device=/dev/video16"&
```

Step2: Turn ON VGA display at the same time, please type..

```
#gst-launch    playbin2    uri=file:///tools/Advantech.avi    video-
sink="mfw_v4lsink device=/dev/video18"&
```

You can see display independent both show Advantech.avi at the same time.

If you'd like to set the output audio as HDMI out or speaker out, please add the parameter of plughw:

A. Plughw:0 → Output the audio through audio jack (AUDIO1)

```
#gst-launch playbin2 uri=file:///tools/Advantech.avi video-sink="mfw_v4lsink device=/dev/video17" audio-sink="alsasink device=plughw:0"
```

B. Plughw:1 → Output the audio through HDMI.

```
#gst-launch playbin2 uri=file:///tools/Advantech.avi video-sink="mfw_v4lsink device=/dev/video17" audio-sink="alsasink device=plughw:1"
```

If you'd like to change display monitor, please refer to below table:

video16	HDMI
video17	HDMI overlay
video18	VGA
video19	VGA overlay
video20	LVDS 0
video21	LVDS 1

3.1.7 I2C Test

There are three i2c bus in ROM-5420.

```
#ls /sys/class/i2c-dev
i2c-0  i2c-1  i2c-2  i2c-3  i2c-4  i2c-5  i2c-6
```

```
#i2cdetect -l
i2c-0   i2c         imx-i2c                 I2C adapter
i2c-1   i2c         imx-i2c                 I2C adapter
i2c-2   i2c         imx-i2c                 I2C adapter
i2c-3   i2c         i2c-1-mux (chan_id 0)   I2C adapter
i2c-4   i2c         i2c-1-mux (chan_id 1)   I2C adapter
i2c-5   i2c         i2c-1-mux (chan_id 2)   I2C adapter
i2c-6   i2c         i2c-1-mux (chan_id 3)   I2C adapter
```

Please try below command to know if there is any device connected to i2c bus 1.

```
#i2cdetect -y 1

     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- UU -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- UU -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: UU -- -- -- -- -- -- -- UU -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: UU -- -- -- -- -- -- -- --
```

The 0x50 is the HDMI address. So you can try below command to know I2C bus is work or not.

```
#i2cdump -f -y 1 0x50
```

No size specified (using byte-data access)

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 00 ff ff ff ff ff ff 00 04 69 fd 22 03 01 01 01    ........?i?"????
10: 16 17 01 03 80 30 1b 78 2a 2a c5 a4 56 4f 9e 28    ?????0?x**??VO?(
20: 0f 50 54 b7 ef 00 d1 c0 81 40 81 80 95 00 b3 00    ?PT??.???@???.?.
30: 71 4f 81 c0 81 00 02 3a 80 18 71 38 2d 40 58 2c    qO???.?:??q8-@X,
40: 45 00 dc 0c 11 00 00 1e 00 00 00 ff 00 44 35 4c    E.???..?.....D5L
50: 4d 51 53 30 39 36 30 34 38 0a 00 00 00 fd 00 32    MQS096048?...?.2
60: 4b 18 53 11 00 0a 20 20 20 20 20 20 00 00 00 fc    K?S?.?      ...?
70: 00 41 53 55 53 20 56 53 32 32 38 0a 20 20 01 f6    .ASUS VS228?  ??
80: 02 03 22 71 4f 01 02 03 11 12 13 04 14 05 0e 0f    ??"qO???????????
90: 1d 1e 1f 10 23 09 17 07 83 01 00 00 65 03 0c 00    ????#?????..e??.
a0: 10 00 8c 0a d0 8a 20 e0 2d 10 10 3e 96 00 dc 0c    ?.???? ?-??>?.??
b0: 11 00 00 18 01 1d 00 72 51 d0 1e 20 6e 28 55 00    ?..???.rQ?? n(U.
c0: dc 0c 11 00 00 1e 01 1d 00 bc 52 d0 1e 20 b8 28    ???..???.?R?? ?(
d0: 55 40 dc 0c 11 00 00 1e 8c 0a d0 90 20 40 31 20    U@???..????? @1
e0: 0c 40 55 00 dc 0c 11 00 00 18 00 00 00 00 00 00    ?@U.???..?......
f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 b5    ...............?
```

If there is nothing connected to HDMI port, the result should be as below:

```
#i2cdump -f -y 5 0x50
```

No size specified (using byte-data access)

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
10: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
20: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
30: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
40: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
50: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
60: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
70: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
80: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
90: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
a0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
b0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
c0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
d0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
e0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
f0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
```

3.1.8 Mini PCIe (Wifi) Test

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

```
#ifconfig wlan0 up
#wpa_passphrase "$ESSID" "$PASS" > /tmp/wpa.conf
(note: put $ESSID and $PASS with exact essid and pass phrase)
#wpa_supplicant -BDwext -iwlan0 -c/tmp/wpa.conf
#dhclient wlan0
```

3.1.9 CAN Test

Step 1: Check CAN network device

```
#dmesg | grep can
vcan: Virtual CAN interface driver
flexcan netdevice driver
flexcan imx6q-flexcan.0: device registered (reg_base=c09b8000, irq=142)
flexcan imx6q-flexcan.1: device registered (reg_base=c09e8000, irq=143)
can: controller area network core (rev 20090105 abi 8)
can: raw protocol (rev 20090105)
can: broadcast manager protocol (rev 20090105 t)
```

Step 2: Activate CAN device

```
#ip link set can0 up type can bitrate 125000
flexcan imx6q-flexcan.0: writing ctrl=0x0e312005

#ip link set can1 up type can bitrate 125000
flexcan imx6q-flexcan.1: writing ctrl=0x0e312005
```

Note! *Bitrate is supported from 1 to 1M.*

Step 3: Send and Receive CAN frames

Receive CAN frames:

```
#cantest can0 &
```

Send CAN frames

```
#cantest can1 12345678#123412341234
read 16 bytes
12345678  [6] 12 34 12 34 12 34
```

3.1.10 Audio Out and MIC In Test

MIC IN command is as follows:

```
#arecord -t wav -c 1 -r 44100 -d 5 2.wav
```

Audio out command is as follows:

```
#aplay 2.wav
```

3.1.11 OpenGL Test

Please follow below instructions to test OpenGL on ROM-5420 platform:

Step 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
```

Step 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 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
```

Step 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
```

3.1.12 LAN Test

ROM-5420 sets DHCP as defaul network portocal.

```
#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)
```

If you would like to config IP manually, please use the command below:

#ifconfig eth0 xxx.xxx.xxx.xxx up

Here is a real case for your reference.The hosts(ROM-5420) IP is 172.17.21.97; the target(A desktop computer) IP is 172.17.20.192

```
#ifconfig eth0 172.17.21.97 up
#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 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
```

3.1.13 RS232 Test

As you can see below, there are 5 UART supported by ROM-5420. /dev/ttymxc0 is reserved for ROM-5420 debug port (ROM-5420 CN1), the rest UART ports could be applied by user.

#setserial -g /dev/ttymxc*

```
/dev/ttymxc0, UART: undefined, Port: 0x0000, IRQ: 58
```

```
/dev/ttymxc1, UART: undefined, Port: 0x0000, IRQ: 59
/dev/ttymxc2, UART: undefined, Port: 0x0000, IRQ: 60
/dev/ttymxc3, UART: undefined, Port: 0x0000, IRQ: 61
/dev/ttymxc4, UART: undefined, Port: 0x0000, IRQ: 62
```

3.1.13.1 UART2 Testing

```
#stty -F /dev/ttymxc1 -echo
#cat /dev/ttymxc1 &
#echo hello > /dev/ttymxc1
  Hello
```

3.1.13.2 UART3 Testing

```
#stty -F /dev/ttymxc2 -echo
#cat /dev/ttymxc2 &
#echo hello > /dev/ttymxc2
  Hello
```

3.1.13.3 /UART3 Testing

```
#stty -F /dev/ttymxc3 -echo
#cat /dev/ttymxc3 &
#echo hello > /dev/ttymxc3
  Hello
```

3.1.13.4 UART4 Testing

```
#stty -F /dev/ttymxc4 -echo
#cat /dev/ttymxc4 &
#echo hello > /dev/ttymxc4
  Hello
```

3.1.14 Watchdog Timer Test

Step 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()
```

Step 2: Please try below command to set timeout as 10 seconds, system will reboot after then.

```
#/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] then you should be able to see below result:

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

Then system will reboot in 10 seconds

3.1.15 Audio Test

Execute the following commands to run the Audio demo application on ROM-5420.

```
#cd /unit_tests
#aplay audio8k16S.wav
```

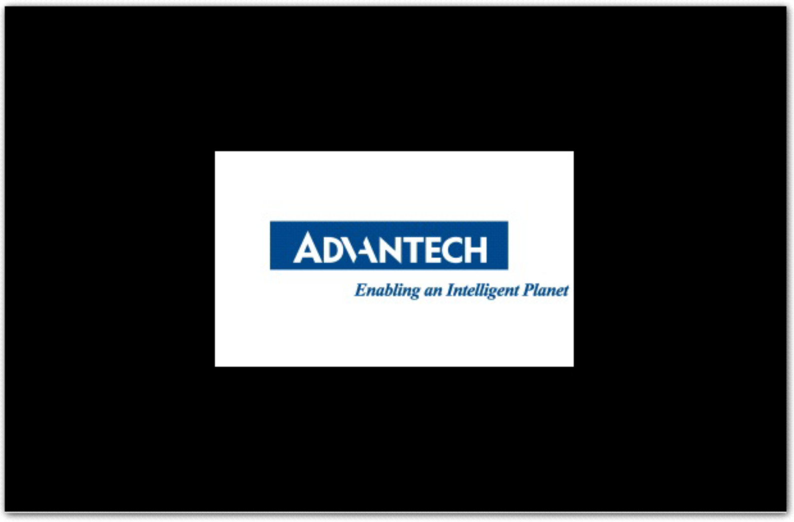
Then you can hear the music from speaker/head-sets.

3.1.16 Photo Demo Test

Execute the following commands to run the Photo demo application on ROM-5420.

```
#cd /tools
#./fbv Advantech.jpg
```

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



3.1.17 Camera Input Test

Execute the following commands to run the camera demo application on ROM-5420.

3.1.17.1 Preview

```
 #gst-launch mfw_v4lsrc ! mfw_v4lsink
```

3.1.17.2 Capture

```
#gst-launch  mfw_v4lsrc  num-buffers=1  !   jpegenc  !  filesink  loca-
tion=/tools/snapshot.jpg
```

3.1.17.3 PView picture

```
#VSALPHA=1 gst-launch filesrc location=/tools/snapshot.jpg ! jpegdec
! imagefreeze ! mfw_isink
```

3.1.17.4 PRecord video

```
#gst-launch  mfw_v4lsrc  !  queue  !  vpuenc  codec=6  !  matroskamux  !
filesink location=/tools/output.avi sync=false
```

3.1.17.5 PPlay video

```
#gst-launch     playbin2     uri=file:///tools/output.avi     video-
sink="mfw_v4lsink device=/dev/video17"
```

3.1.18 Battery Test

Execute the following commands to run the battery application on ROM-5420.

Read mV

```
 # i2cset -f -y 4 0x30 0x82 0x05
 # RAWDATA=`i2cget -f -y 4 0x30 0x80 w`
 # echo $((RAWDATA % 256 * 256 + RAWDATA / 256))mV
```

ROM-5420 platform is an embedded system with Linux kernel 3.0.35 inside. It contains all system-required shell commands and drivers ready for ROM-5420 platform. We do not offer IDE developing environment in ROM-5420 BSP, users can evaluate and develop under Ubuntu 10.04LTS environment and the BSP for ROM-5420 supports console mode only.

There are three major boot components for Linux, “u-boot.bin”, “uImage” and “File System”. The “u-boot.bin” 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, SATA HDD or onboard flash)

The purpose of this chapter is to introduce software development of ROM-5420 to you, so that you can develop your own application(s) efficiently.

ROM-5420 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 ROM-5420 evaluation/development.

3.2 Package Content

We would offer you two different kinds of Linux package for ROM-5420. One is pre-built system image for system recovery another is source code package (BSP).

3.2.1 Source Code Package

ROM-5420 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. ROM-5420 source code package is composed of six main folders: “cross_compiler”, “document”, “image”, “package”, “scripts”, and “source”.

Note! *ROM-5420 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 5420LBVxxxx package contents:

■ **“cross_compiler”**	→	This folder contains source code for cross compiler.
■ **“document”**	→	This folder contains user guide.
■ **“image”**	→	This folder contains the uImage, 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 uImage 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 uImage & 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 “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

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.

Linux is a clone of the operating system UNIX. It has all the features you would expect in a modern fully-fledged UNIX, including true multitasking, virtual memory, shared libraries, demand loading, shared copy-on-write executables, proper memory management, and multitask networking including IPv4 and IPv6.

Linux is easily portable to most general-purpose 32- or 64-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-3.0.35” 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 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-3.0.35/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 ROM-5420 Linux source code package, please refer to following instructions to extract to your developing environment:

1. Copy "5420LBVxxxx.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 5420LBVxxxx.tar.bz2`** (Unzip file)
7. Then you can see folder "*7420LBVxxxx*" on desktop.

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-5420_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 “uImage” quickly. You can build uImage 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. **`6)#./cfg_kernel.sh imx6_rom5420_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 “./7420LBVxxxx”. 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.

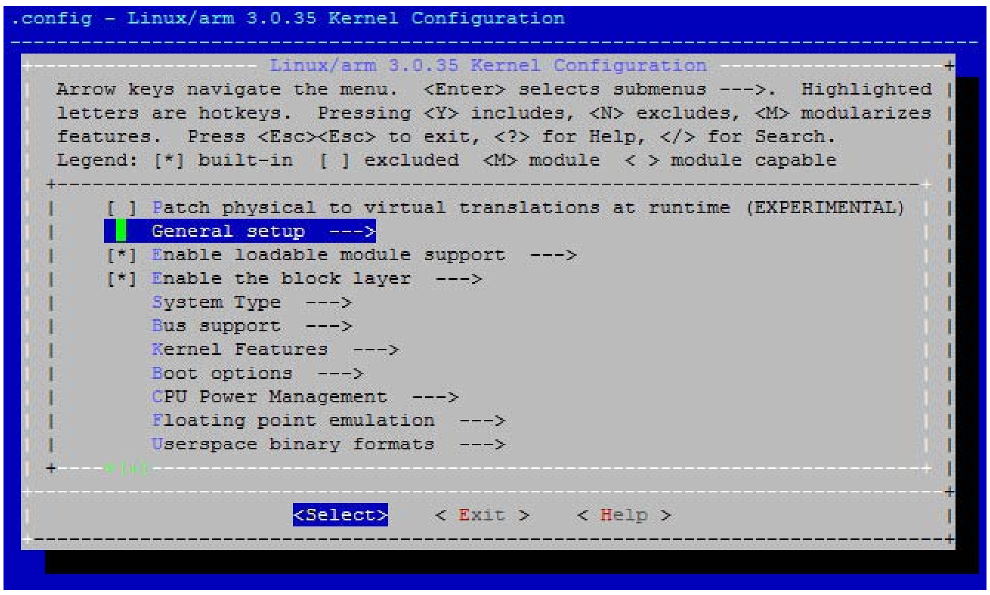
3.5 Source Code Modification

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:



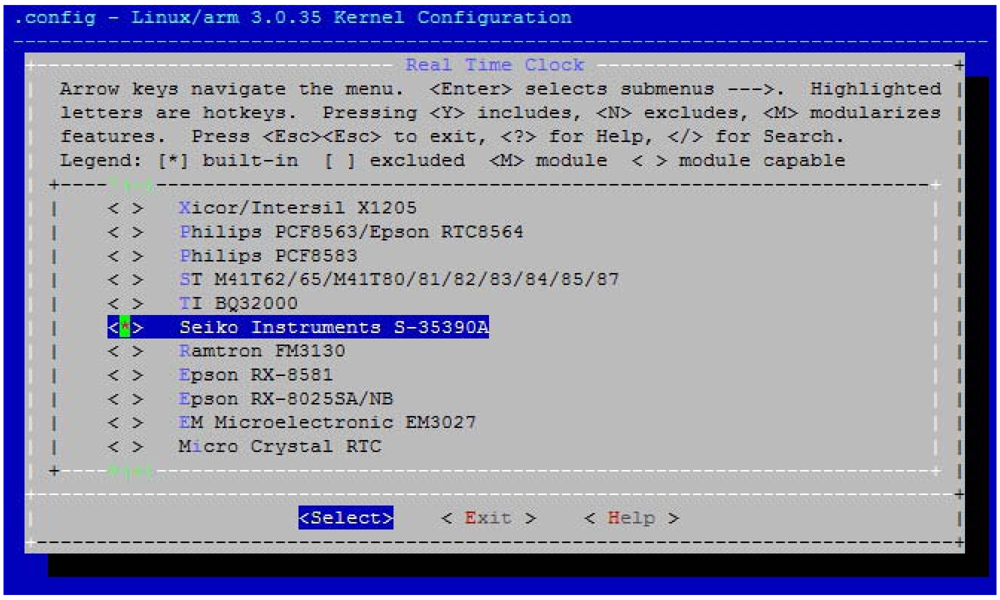
```
.config - Linux/arm 3.0.35 Kernel Configuration
 ------------------- Linux/arm 3.0.35 Kernel Configuration -------------------
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted
  letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
  features.  Press <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  --->
     v(+)

                     <Select>    < Exit >    < Help >
```

Figure 3.1 Linux Kernel Configuration

8. 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.



```
.config - Linux/arm 3.0.35 Kernel Configuration
 ------------------------------------------------------------------------------
  +-------------------------------- Real Time Clock ---------------------------------+
  |  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              |
  | +----^(-)---------------------------------------------------------------------+  |
  | |    < >   Xicor/Intersil X1205                                              |  |
  | |    < >   Philips PCF8563/Epson RTC8564                                     |  |
  | |    < >   Philips PCF8583                                                   |  |
  | |    < >   ST M41T62/65/M41T80/81/82/83/84/85/87                             |  |
  | |    < >   TI BQ32000                                                        |  |
  | |    <*>   Seiko Instruments S-35390A                                        |  |
  | |    < >   Ramtron FM3130                                                    |  |
  | |    < >   Epson RX-8581                                                     |  |
  | |    < >   Epson RX-8025SA/NB                                                |  |
  | |    < >   EM Microelectronic EM3027                                         |  |
  | |    < >   Micro Crystal RTC                                                 |  |
  | +----v(+)---------------------------------------------------------------------+  |
  +----------------------------------------------------------------------------------+
  |                    <Select>    < Exit >    < Help >                              |
  +----------------------------------------------------------------------------------+
```

Figure 3.2 Selecting Seiko Instruments S-35390A

9. Change directory to “source/linux-3.0.35/arch/arm/mach-mx6”, edit the “board-mx6q_ROM-5420.h” and “board-mx6q_advantech.c”.
Please add below codes to source/linux-3.0.35/arch/arm/mach-mx6/board-mx6q_ROM-5420.h:

```
static struct i2c_board_info mxc_i2c0_board_info[] __initdata = {
        {
                I2C_BOARD_INFO("sgtl5000", 0x0a),
        },
        {
                I2C_BOARD_INFO("s35390a", 0x30),
        },
        {
                I2C_BOARD_INFO("24c02", 0x57),
        },

};
```

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));
```

10. Please refer to former Chapter 3.5.1 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 ROM-5420 Boot Logo

By default, ROM-5420 shows a boot logo when booting up. You can replace the logo to whatever your want by following below steps:

1. You have to download "netpbm" corresponding to your OS version from internet first,
2. Install "netpbm" by typing `$sudo apt-get install netpbm`.
3. 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.*

4. Open "Terminal" on Ubuntu 10.04 LTS..
5. `$sudo su` (Change to "root" authority)
6. Input user password.
7. `#cd Desktop/bootlogo` (Go into the folder that bootlogo.png located)
8. `#pngtopnm bootlogo.png | ppmquant 224 | pnmtoplainpnm > logo_linux_clut224.ppm`
9. `#cp logo_linux_clut224.ppm \ ../7420LBVxxxx/source/linux-3.0.35/drivers/video/logo/`
10. Then you can refer Chapter 3.3.1 to rebuild the kernel with your own boot logo.

3.6 Create a Linux System Boot Media

ROM-5420 supports boot from SD card, SATA device and onboard flash. This section will guide you how to build a image for ROM-5420 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 ROM-5420 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. `#./mksd-linux.sh /dev/sdf`
8. Type "y" (Start to copy files, wait until it shows [Done])

Then insert the Linux system SD card to ROM-5420 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 3.5.1 to make a Linux system SD card

2. Insert this Linux system SD card to ROM-DB7500 and connect serial console.
3. On ROM-5420 platform, type **`#root`** (Login)
4. On ROM-5420 platform, type **`#cd /mk_inand`**
5. On ROM-5420 platform, type **`#./mkinand-linux.sh /dev/mmcblk0`**
6. On ROM-5420 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.

3.6.3 Boot from SATA

If you already have a Linux system SD card, you can refer to the following steps to copy the content to SATA drive, you can refer following steps to boot form SATA devices. The script "mksd-linux.sh" will be helpful to install system image to SATA device.

1. Refer to Chapter 3.5.1 to make a Linux system SD card
2. Insert this Linux system SD card to ROM-DB7500 and open serial console.
3. On ROM-5420 platform, type **`#root`** (Login)
4. On ROM-5420 platform, type **`#cd /mk_inand`**
5. On ROM-5420 platform, type **`#./mksd-linux.sh /dev/sda`** (Check the SATA location like: /dev/sda)
6. On ROM-5420 platform, type "y "(Start to copy files, waiting a few minutes until it shows [Done])
7. Power off and remove this SD card.

Now you can boot from SATA without SD card.

3.7 Debug Message

ROM-5420 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 required. Below are the detailed instructions for how to set up serial console, a "HyperTerminal" on a Windows host:

1. Connect ROM-5420 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 3.6.
3. Press "POWER" key to power up the board. The bootloader prompt is displayed on the terminal screen.

3.8 Linux Software AP and Testing on ROM-5420

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 ROM-5420 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. 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 ROM-5420, then open serial console.
16. On ROM-5420 platform, type **#root** (Login)
17. On ROM-5420 platform, type **#cd /tool**
18. On ROM-5420 platform, type **#./helloworld**
19. Now you should be able to see “Hello World!” shown on ROM-5420.

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;
     }
t    imeout = 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_KEEPALIVE, 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 GPIO Setting

Please see GPIO initial code listed below. Below code is to assign the starting value to GPIO variable.

```
/* Enable GPIO */
gpio_request(SABRESD_GPIO0, "gpio-0");
gpio_request(SABRESD_GPIO1, "gpio-1");
gpio_request(SABRESD_GPIO2, "gpio-2");
gpio_request(SABRESD_GPIO3, "gpio-3");
gpio_request(SABRESD_GPIO4, "gpio-4");
gpio_request(SABRESD_GPIO5, "gpio-5");
gpio_request(SABRESD_GPIO6, "gpio-6");
gpio_request(SABRESD_GPIO7, "gpio-7");
gpio_request(SABRESD_GPIO8, "gpio-8");
gpio_request(SABRESD_GPIO9, "gpio-9");
gpio_request(SABRESD_GPIO10, "gpio-10");
gpio_request(SABRESD_GPIO11, "gpio-11");
```

3.8.4 RS232 Initial Code

The RS232 initial code as below. It shows you how to initial COM2 ports.

```
int open_port(void)
{
     int fd;
     fd=open("/dev/ttymxc1",O_RDWR|O_NOCTTY|O_NDELAY);
     if(fd == -1){
     perror("open error");
     }
     return(fd);
}
```

3.8.5 Display Output Setting

3.8.5.1 LVDS Settings

Please set environment in u-boot as below:

```
setenv bootargs_mmc 'setenv bootargs ${bootargs} root=/dev/
mmcblk1p1 rootwait rw video=mxcfb0:dev=ldb,LDB-XGA,if=RGB24'
```

LDB-XGA is an example for the resolution of your LVDS panel. You can input the actual resolution of your LVDS panel here, such as 800x480, 1024x768, etc. The system will accomplish the corresponding parameters automatically.

If the panel has problem to be activated, you may need to check the panel datasheet to configure the panel related parameters. The LVDS video mode database is stored in linux-3.0.35/drivers/video/mxc/ldb.c. You can add a new one for your LVDS panel.

```
static struct fb_videomode ldb_modedb[] = {
     {
     "LDB-XGA", 60, 1024, 768, 15385,
        220, 40,
        21, 7,
        60, 10,
        0,
        FB_VMODE_NONINTERLACED,
        FB_MODE_IS_DETAILED,},
    }
```

The definition of fb_videomode in linux-3.0.35/include/linux/fb.h:

The name field is optional. If you input this value, it can be used in U-Boot environment settings.

The refresh field is the screen refresh frame rate, such as 60Hz, 70Hz. The resolution can be filled in the xres & yres fields.

The pixel clock (pixclock) is equaled to *10^{12}/(Total horizontal line * Total vertical line * DCLK*). For example, the total horizontal line is 1344 DCLK, and total vertical number is 806 horizontal lines. The DCLK frequency is 60 MHz. Therefore, we can get 10^{12}/(1344*806*60) = 15385.

The *margin* values can be seen as front porch & back porch.

The *sync_len* means pulse width.

The *sync* value indicates the sync polarity (low or high).

```
struct fb_videomode {
     const char *name;      (optional)
     u32 refresh;            (optional)
     u32 xres;
     u32 yres;
     u32 pixclock;
     u32 left_margin;
     u32 right_margin;
     u32 upper_margin;
     u32 lower_margin;
     u32 hsync_len;
     u32 vsync_len;
     u32 sync;
     u32 vmode;
     u32 flag;
};
```

3.8.5.2 Single Display Settings

HDMI out, please set in u-boot as below:

```
setenv bootargs_mmc 'setenv bootargs ${bootargs} root=/dev/mmcblk1p1 rootwait rw video=mxcfb0:dev=hdmi,1920x1080M@60,if=RGB24'
```

VGA out, please set in u-boot as below:

```
setenv bootargs_mmc 'setenv bootargs ${bootargs} root=/dev/mmcblk1p1 rootwait rw video=mxcfb0:dev=lcd,1920x1080M@60,if=RGB24'
```

LVDS 1(Single) out, please set in u-boot as below:

```
setenv bootargs_mmc 'setenv bootargs ${bootargs} root=/dev/
mmcblk1p1 rootwait rw video=mxcfb0:dev=ldb,1920x1080M@60,if=RGB24
ldb=sin0'
```

LVDS 2(Split) out, please set in u-boot as below:

```
setenv bootargs_mmc 'setenv bootargs ${bootargs} root=/dev/
mmcblk1p1 rootwait rw video=mxcfb0:dev=ldb,1920x1080M@60,if=RGB24
ldb=sin1'
```

3.8.5.3 Multi Display Settings

When you want to display dual LVDS, VGA and HDMI output , please set parameter in U-boot as follows. This is the default settings in U-boot.

```
setenv bootargs_mmc 'setenv bootargs ${bootargs} root=/dev/mmcblk1p1
rootwait
setenv bootargs_base 'setenv bootargs console=ttymxc0,115200
enable_wait_mode=off video_mode=extension'
```

For display interface clock, there are several options (Independently for each port) listed below:

1. Derived from the IPU internal clock (Master Mode)\
2. Provided by an external source (Slave Mode)
3. The transfer rate supported

When a single port is active, the pixel clock rate is up to 264 MHz

When both LVDS ports are active, you have to follow below condition:

1. Each pixel clock rate may be up to 220 MHz**
2. The sum of pixel clock rates is up to 240 MHz

Note! *Specified pixel clocks frequencies are applicable for internal clocks, but may be limited by IO buffers speed capability. Final numbers are subjected to AC characterization.*

3.8.6 Network Setup

Default: IP get form DHCP.

Manual: Set IP by below command:

```
#ifconfig eth0 192.168.0.1 up
```

ifconfig is to configure network interfaces, the manual page is as below.

```
SYNOPSIS
       ifconfig [-v] [-a] [-s] [interface]
       ifconfig [-v] interface [aftype] options | address ...
OPTIONS
       -a     display all interfaces which are currently available, even if
              down
       -s     display a short list (like netstat -i)
```

```
-v      be more verbose for some error conditions
   interface
      The name of the interface. This is usually a driver name fol-
      lowed by a unit number, for example eth0 for the first Ethernet
      interface. If your kernel supports alias interfaces, you can
      specify them with eth0:0 for the first alias of eth0. You can
      use them to assign a second address. To delete an alias inter-
      face use ifconfig eth0:0 down. Note: for every scope (i.e. same
      net with address/netmask combination) all aliases are deleted,
      if you delete the first (primary).
   [aftype]
   up     This flag causes the interface to be activated. It is
          implicitly specified if an address is assigned to the inter
          face.

   down      This flag causes the driver for this interface to be shut down.
   address  The IP address to be assigned to this interface.
   netmask [addr]
      Set the IP network mask for this interface. This value defaults
      to the usual class A, B or C network mask (as derived from the
      interface IP address), but it can be set to any value.
   broadcast [addr]
      If the address argument is given, set the protocol broadcast
      address for this interface. Otherwise, set (or clear)
   the IFF_BROADCAST flag for the interface.
     del addr/prefixlen
       Remove an IPv6 address from an interface.
```

3.8.7 Storage (SATA /eMMC/SD Card)

The storages devices are named as follows:

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

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.

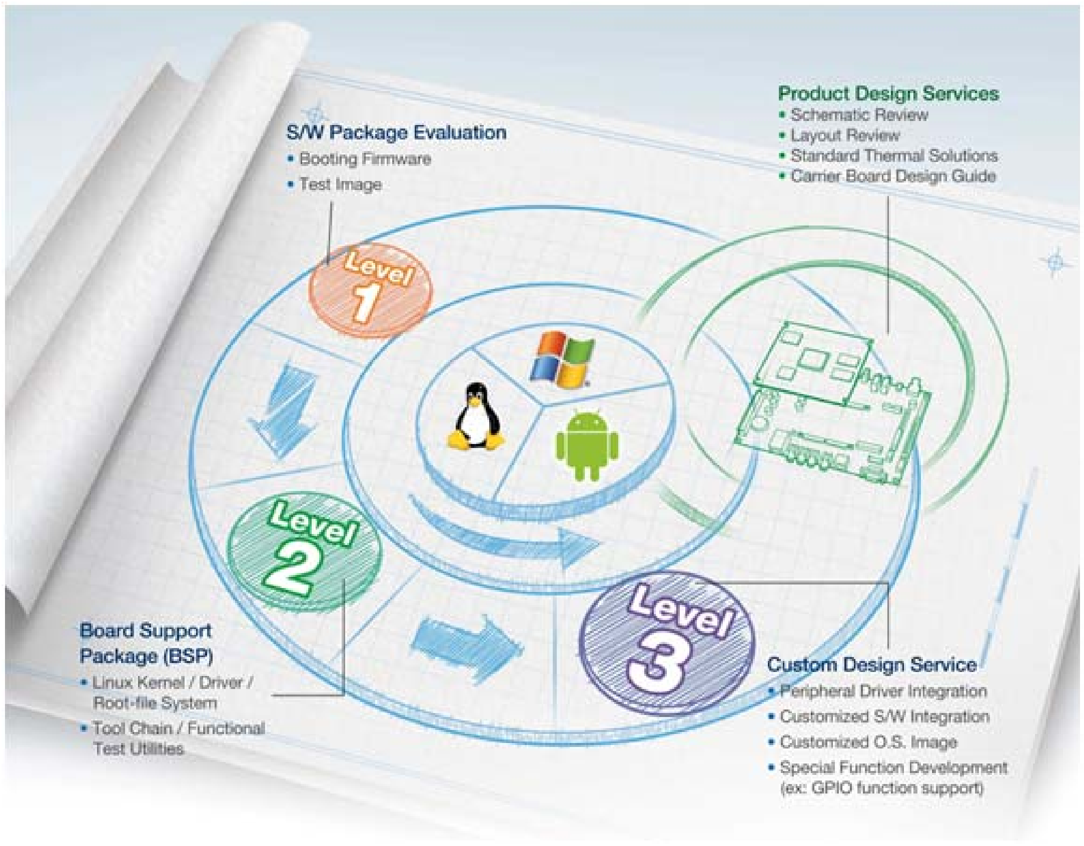
1. Copy “5420LIVxxxx.tar.bz2” package to your desktop.
2. Open "Terminal" on Ubuntu 10.04 LTS.
3. **`$sudo su`** (Change to “root” authority)
4. Input your password.
5. **`#cd Desktop/`**
6. **`#tar xvf 7420LIVxxxx.tar.bz2`** (Unzip files)
7. Insert one SD card to your developing computer
8. Check the SD card location, like /dev/sdf
9. **`#cd ./5420LBVxxxx_prebuilt_image`**
10. **`#dd if=5420LIVxxxx.img of=/dev/sdf`**
11. Please wait until dump disk is done
12. Connect console cable to debug port (CN1) and open serial console program on Ubuntu 10.04 LTS, set baudrate to 115200. For detail console setting, please refer to section 3.6.
13. On ROM-5420 platform, type **`#root`** (Login)
14. On ROM-5420 platform, type **`#cd /mk_inand`**
15. On ROM-5420 platform, type **`#./mkinand-linux.sh /dev/mmcblk0`**
16. On ROM-5420 platform, type “y “
    (Start to copy files, wait until it shows [Done])
17. Power off and remove this SD card.

Chapter 5

Advantech Services

This chapter introduces Advantech design in serviceability, technical support and warranty policy for ROM-5420 evaluation kit.

5.1 RISC Design-in Services


S/W Package Evaluation
• Booting Firmware
• Test Image
Product Design Services
• Schematic Review
• Layout Review
• Standard Thermal Solutions
• Carrier Board Design Guide
Level 1
Level 2
Level 3
Board Support Package (BSP)
• Linux Kernel / Driver / Root-file System
• Tool Chain / Functional Test Utilities
Custom Design Service
• Peripheral Driver Integration
• Customized S/W Integration
• Customized O.S. Image
• Special Function Development (ex: GPIO function support)

Advantech RISC Design-in Services help customers to reduce the time and work involved with designing new carrier boards. We handle the complexities of technical research and greatly minimize the development risk associated with carrier boards.

Easy Development

Advantech has support firmware, root file-system, BSP or other develop tools for customers. It helps customers to easy develop their carrier board and differentiate their embedded products and applications.

- Full Range of RISC Product Offerings
- Comprehensive Document Support

Design Assistance Service

Advantech provides check list for engineer for easy check their schematics and also review service based on customer carrier board schematics. Those services are preventative, and help to catch design errors before they happen. It helps to save a lot of time and costs with regard to developing carrier boards.

- Schematic Review
- Placement and Layout Review
- Debugging Assistance Services
- General/Special Reference Design Database.

Thermal Solution Services

In order to provide quicker and more flexible solutions for customer's thermal designs. Advantech provides thermal solution services including modularized thermal solutions and customized thermal solutions.

- Standard Thermal Solutions
- Customized Thermal Solutions

Embedded Software Services

Supports driver, software integration or customized firmware, root file-system and Linux image. Customer can save lot of time and focus on their core development.

- Embedded Linux/ Android OS
- Advantech boot loader Customization

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	(618)529-4525

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.emacinc.com/support

5.3 Technical Support and Assistance

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

http://www.emacinc.com/

For technical support and service, please visit our support website at:

<http://www.emacinc.com/>

1. Visit the disrtributor's web site at www.emacinc.com/support 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