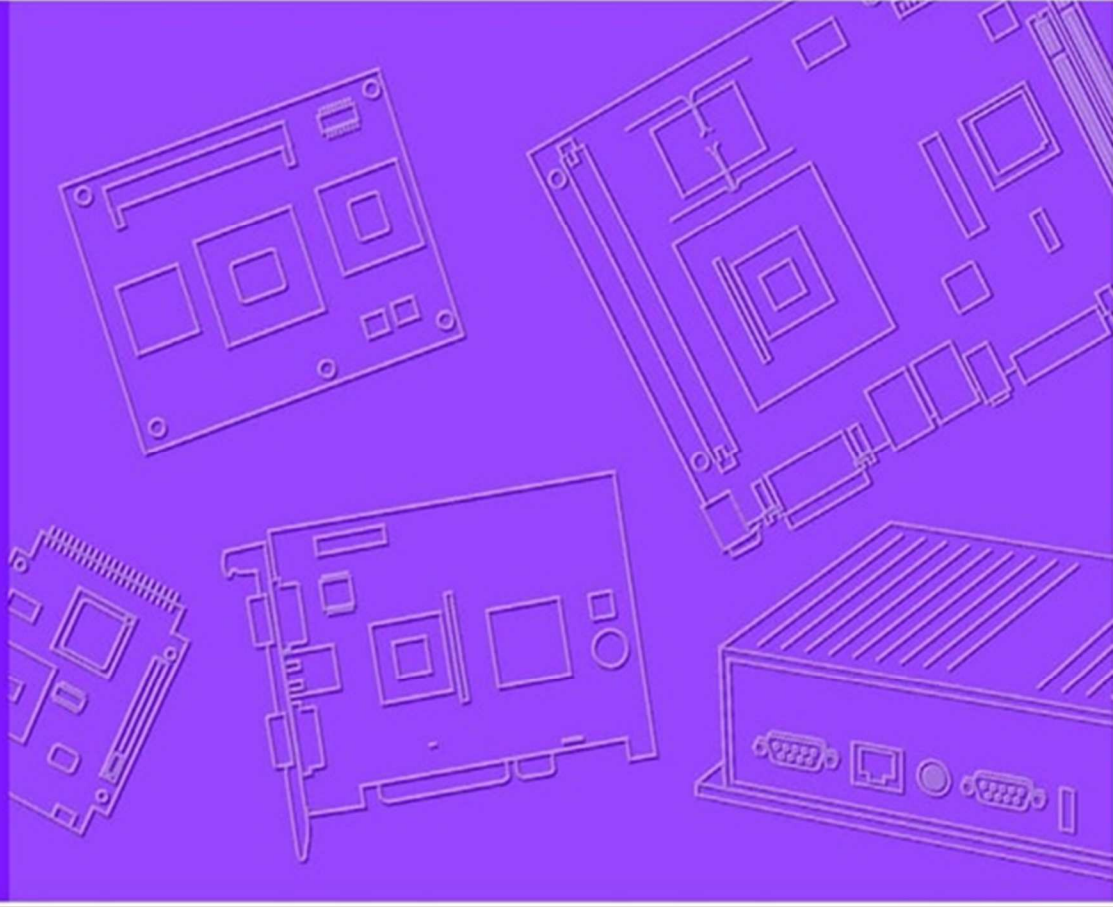


User Manual

RSB-4760

3.5" SBC with Qualcomm APQ-8016 Processor ARM® Cortex™ A53 Architecture

ADVANTECH

Enabling an Intelligent Planet

Table of Content

1. General Introduction

1.1. Introduction

RSB-4760 is a RISC 3.5" single board computer (SBC) powered by Qualcomm ARM® Cortex®-A53 APQ8016 processor that supports full HD display and intergrades on board wireless solution – Wi-Fi, BT and GPS. RSB-4760 also features in mini PCIe, M.2, and SIM card slots for expanding connectivity capability, like 3G, 4G/LTE modules. Equipped with complete Android, Linux and Win10 IoT core BSPs, this SBC enables customers to easily develop unique application on specific OS.

1.2. Specifications

1.2.1. Functional Specifications

Processor: ƒ Qualcomm APQ8016 CPU

- ARM Cortex™-A53 high performance processor, Quad core up to 1.2 GHz
- Supports 2 IPU, OpenGL ES 2.0 for 3D BitBLT for 2D and OpenVG™ 1.1
- Video decoder: MPEG-4 ASP, H.264 HP, H.263, MPEG-2 MP, MJPEG BP
- Video Encoder: MPEG-4 SP, H.264 BP, H.263, MJPEG BP
- Encode: 30 fps 720p (H.264 Baseline/MPEG-4)
  30 fps 1080p (MPEG-4/H.264/VP8/H.263)
- Decode: 30 fps 1080p (MPEG-4/H.264/H.263/DivX/MPEG2/VC1/Soreson/VP8)

System Memory Support

- DDR3 1066 MHz
- Capacity: on board DDR3 1/2 GB

Gigabit Ethernet

- Chipset: Microchip LAN7500
- 1 x10/100/1000 Mbps

Peripheral Interface

- 1 x HDMI
- 1 x USB OTG, 2 x USB Type A, and 2 x USB pin headers
- 1 x Line Out
- 1 x Mic In
- 1 x SD slot
- 1 x 4-wire RS-232/422/485 DB9 Connector
- 8 x GPIO via D-SUB 9 and 8 x GPIO via pin header (3.3V TTL level)
- 1 x 4 wires console via pin header
  (Configurable for general purpose UART or M.2 UART signal)
- 1 x miniPCIe slot
- 1 x M.2 slot
- 1 x SIM slot
- 1 x I2C
- 1 x Wi-Fi 802.11 b/g/n 2.4GHz
- 1 x Bluetooth 4.1
- 1 x GNSS

OS Support

Yocto Linux, Debian and Android

1.2.2. Mechanical Specifications

- Dimension: 146 x 102 mm (5.7”x4”)
- Height: 15.92 mm
- Reference Weight: 640 g (including whole package)

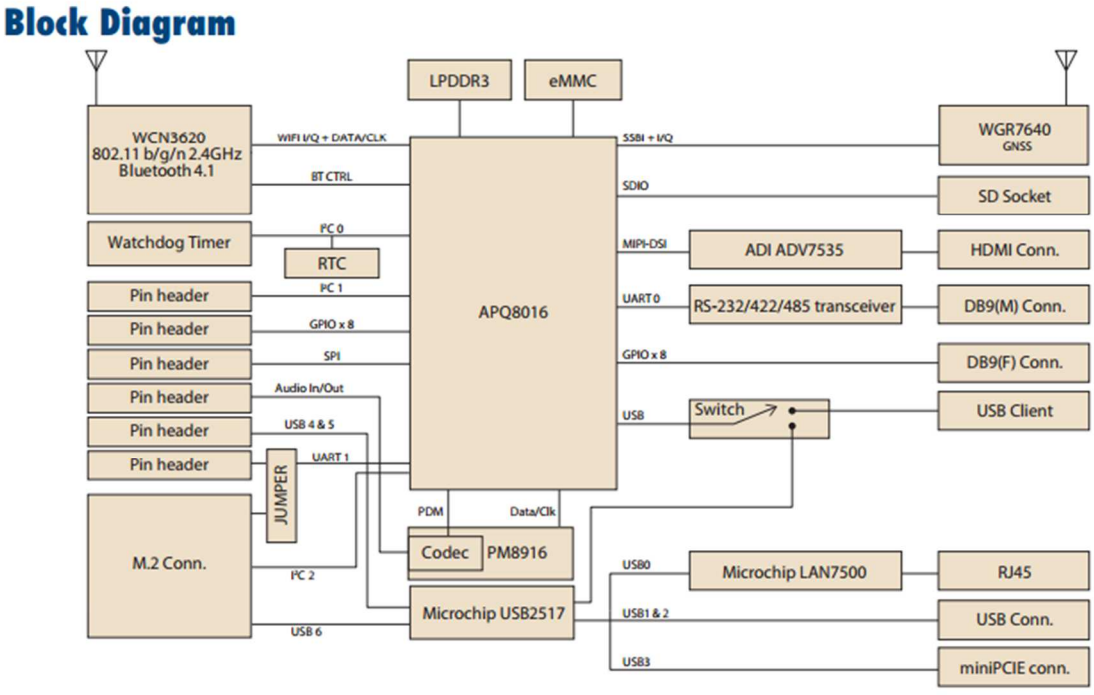
1.2.3. Electrical Specifications

- Power supply type: DC-in 9 - 36V
- Power consumption:
  - Kernel Idle mode: 3W
  - Max mode: 6W
- RTC Battery:
  - Typical voltage: 3.0 V

1.3. Environmental Specifications

- Operating temperature: 0 ~ 60° C (32 ~ 140° F)
- Operating humidity: 40° C @ 95% RH Non-condensing
- Storage temperature: -40 ~ 85° C (-40 ~ 185° F)
- Storage humidity: 60° C @ 95% RH Non-condensing

1.4. Block Diagram

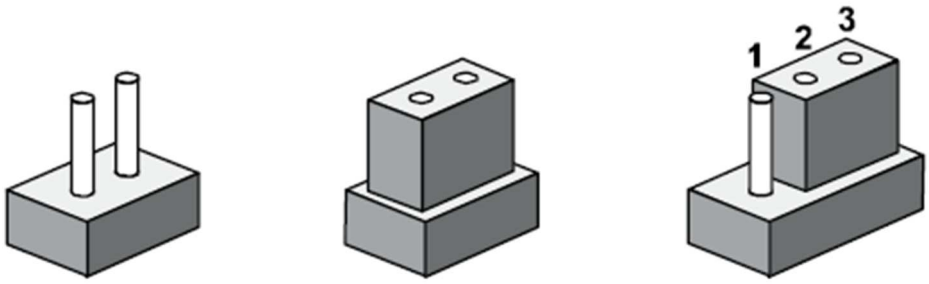


2. H/W Installation

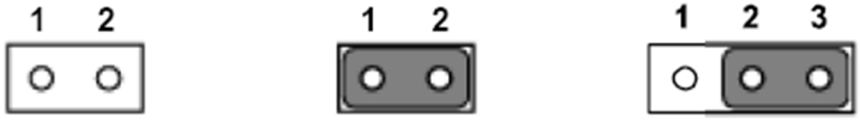
2.1. Jumpers

2.1.1. Jumper Description

Cards can configured by setting jumpers. A jumper is a metal bridge used to close an electric circuit. It consists of two metal pins and a small metal clip (often protected by a plastic cover) that slides over the pins to connect them. To close a jumper, you connect the pins with the clip. To open a jumper, you remove the clip. Sometimes a jumper will have three pins, labeled 1,2 and 3. In this case you would connect either pins 1 and 2 or 2 and 3



The jumper settings are schematically depicted in this manual as follows.



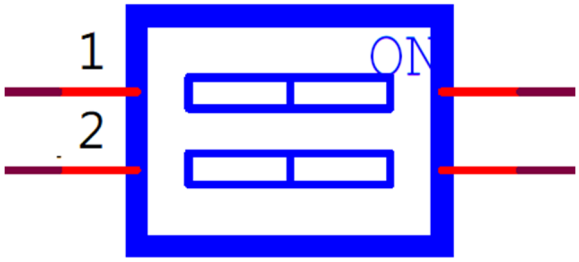
A pair of needle-nose pliers may be helpful when working with jumpers. If you have any doubts about the best hardware configuration for your application, contact your local distributor or sales representative before you make any changes.
Generally, you simply need a standard cable to make most connections.

2.1.2. Switch List

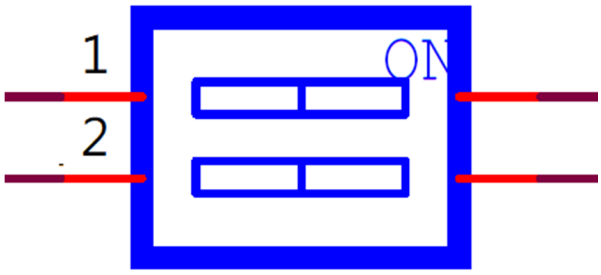
Table 2.1: Switch List	
SW1	Boot configuration
SW3	USB setting
SW4	RS-232/422/485 setting

2.1.3. Jumper Settings

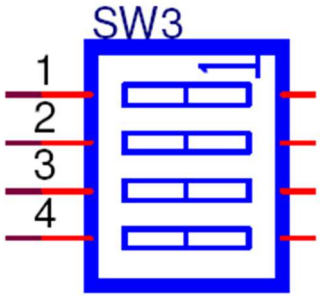
SW1	Boot device
Part number	1600000202
Description	DIP SW CHS-02TB(29) SMD 4P SPST P=1.27mm W=5.4mm
Setting	Function
1 OFF 2 ON	Boot from eMMC
1 ON 2 ON	Boot from USB



SW3	Boot device
Part number	1600000202
Description	DIP SW CHS-02TB(29) SMD 4P SPST P=1.27mm W=5.4mm
Setting	Function
1 ON	Force OTG mode
1 OFF	USB 2.0 Host mode



SW4	RS-232/422/485 selection
Part number	1600000084
Description	DIP SW CHS-02TB(29) SMD 4P SPST P=1.27mm W=5.4mm
Setting	Function
1 OFF 2 OFF	Loopback mode
1 OFF 2 ON	RS-232
1 ON 2 OFF	RS-485 Half Duplex
1 ON 2 ON	RS-422 Full Duplex



2.2. Connectors

2.2.1. Connector List

Table 2.2: Connector List	
BAT_COIN	RTC Battery
DCIN	DC Power Jack
GPIO	GPIO
COM	RS-232/422/485
LAN	Ethernet Connector
CN11	HDMI
USB	USB port 0/1
MICRO USB	USB OTG
UART	UART/Debug Port
USB_INIT	USB port 2/3
GPIO_INT	GPIO
MIC_IN	MIC in
LINE_OUT	Line Out
RF_GPS	GPS MHF connector
RF_WL_BT	WiFi/BT MHF connector
I2C	I2C
SPI	SPI
MINI_PCIE	MiniPCIe
CN4	SIM socket
SD	SD socket
M2	M.2

2.2.2. Connector Settings

2.2.2.1.RTC Battery connectors (CN1)

RSB-4760 supports a lithium 3V/210mAH CR2032 battery with wire via battery connector.

2.2.2.2.DC power Jack (DCIN)

RSB-4760 comes with a DC-Jack header that carries 9-36V DC external power input.

Pin	Description
1(Inner)	DC_In
2(Outer)	GND

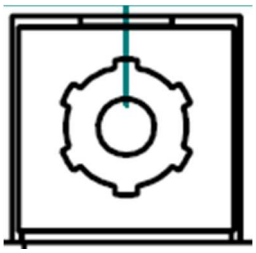


Figure 2.1 DC Power Jack

2.2.2.3.GPIO (GPIO)

RSB-4760 provides one D-Sub 9-pin connector for 8 GPIO

Pin	Description
1	GPIO1
2	GPIO2
3	GPIO3
4	GPIO4
5	+3.3V
6	GPIO5
7	GPIO6
8	GPIO7
9	GPIO8

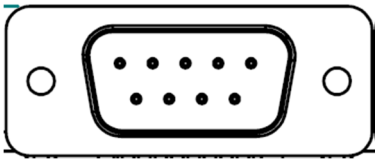


Figure 2.2 GPIO Connector

2.2.2.4.RS-232/422/485 (COM)

RSB-4760 provides one D-Sub 9-pin connector serial communication interface port. The port can support RS-232/422/485 mode communication.

Pin	Description		
1	N/C	RS-422 TX-	RS-485-
2	COM 2_RXD	RS-422 TX+	RS-485+
3	COM2_TXD	RS-422 RX+	
4	N/C	RS-422 RX-	
5	GND		
6	N/C		
7	COM2_RTS		
8	COM2_CTS		
9	N/C		

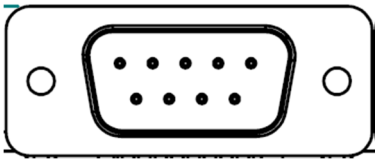


Figure 2.3 COM Port

2.2.2.5.Ethernet Connector (LAN)

RSB-4760 provides one RJ45 LAN interface connector; it is fully compliant with IEEE802.3u 10/100/1000 Base-T CSMA/CD standards. The Ethernet port provides standard RJ-45 jack connector with LED indicators on the front side to show Active/Link status and Speed status.

Pin	Description
1	MIDI0+
2	MIDI0-
3	MIDI1+
4	MIDI1-
5	GND
6	GND
7	MIDI2+
8	MIDI2-
9	MIDI3+
10	MIDI3-
11	VCC
12	ACT
13	Link100#
14	Link1000#

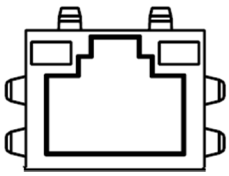


Figure 2.4 Ethernet Connector

2.2.2.6. HDMI (HDMI)

RSB-4760 provides one HDMI interface connector which provides all digital audio/video interfaces to transmit the uncompressed audio/video signals and is HDCP and CEC compliant Connect the HDMI audio/video device to this port. HDMI technology can support a maximum resolution of 1920 x 1080p but the actual resolution supported depends on the monitor being used.

Pin	Description
1	HDMI_TD2+
2	GND
3	HDMI_TD2-
4	HDMI_TD1+
5	GND
6	HDMI_TD1-
7	HDMI_TD0+
8	GND
9	HDMI_TD0-
10	HDMI_CLK+
11	GND
12	HDMI_CLK-
13	HDMI_CEC_A
14	GND
15	DDC_CLK_HDMI_A
16	DDC_DATA_HDMI_A
17	GND
18	+5V
19	HDMI_HPD

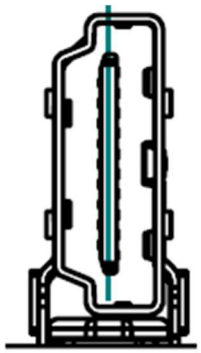


Figure 2.5 HDMI

2.2.2.7. USB Connector (USB)

RSB-4760 supports one standard USB2.0 Type A connector in the coastline.

Pin	Description
1	+5V
2	USB1_D-
3	USB1_D+
4	GND
5	+5V
6	USB2_D-
7	USB2_D+
8	GND

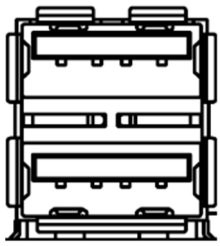


Figure 2.6 USB Type A Connector

2.2.2.8.USB OTG Connector (MICRO_USB)

RSB-4760 supports one USB OTG port in the coastline.

Pin	Description
1	+5V
2	USB1_D-
3	USB1_D+
4	ID
5	GND

Figure 2.7 USB OTG Connector

2.2.2.9.UART/Debug Port (UART)

RSB-4760 can communicate with a host server (Windows or Linux) by using ROM-ED20.

Pin	Description
1	+3.3V
2	GND
3	RXD
4	RTS
5	TXD
6	CTS

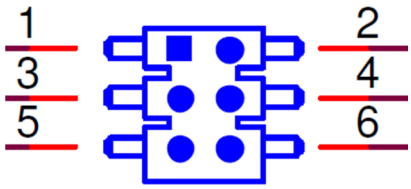


Figure 2.8 Debug Port

2.2.2.10. USB (Internal Pin Header) (SUB_INT)

RSB-4760 provides extra internal 2x USB2.0 pin headers.

Pin	Description
1	+5V
2	+5V
3	USB4_D-
4	USB3_D-
5	USB4_D+
6	USB3_D+
7	GND
8	GND
9	GND

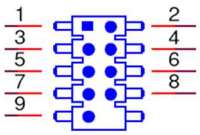


Figure 2.9 USB Internal Pin Header

2.2.2.11. MIC in (MIC_IN)

RSB-4760 offers MIC in, microphone can be connected to the MIC in pin header

Pin	Description
1	MIC IN
2	GND

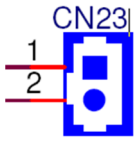


Figure 2.10 MIC in Internal Pin Header

2.2.2.12. Line out (LINE_OUT)

RSB-4760 offers Line-out stereo speakers; earphone can be connected to the lineout pin header

Pin	Description
1	LINEOUT_L
2	LINEOUT_R
3	GND

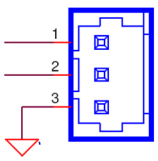


Figure 2.11 Line our Internal Pin Header

2.2.2.13. GPIO (CN31)

RSB-4760 provides internal GPIO interface by 2x11 pin headers.

Pin	Description
1	+3.3V
2	GND
3	GPIO_01
4	GPIO_02
5	GPIO_03
6	GPIO_04
7	GPIO_05
8	GPIO_06
9	GPIO_07
10	GPIO_08
11	GPIO_09
12	GPIO_10
13	GPIO_11
14	GPIO_12
15	GPIO_13
16	GPIO_14
17	GPIO_15
18	GPIO_16
19	GPIO_17
20	GPIO_18
21	GPIO_19
22	GPIO_20

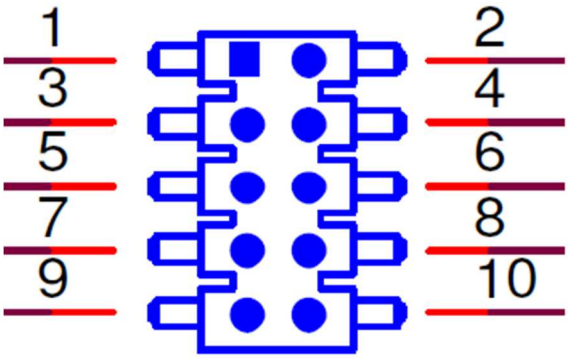


Figure 2.12 GPIO Internal Pin Header

2.2.2.14. I2C (I2C)

RSB-4760 provides 2 I2C pin headers.

Pin	Description
1	GND
2	I2C4_SDA
3	I2C4_SCL
4	+V3.3

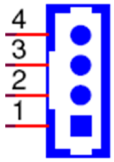


Figure 2.13 I2C Pin Headers

2.2.2.15. MiniPCIe (MINI_PCIE)

RSB-4760 supports full size miniPCIe slot both USB and PCIe interface. If the miniPCIe card is only half-sized, please purchase extending bracket (P/N: 1960047454N000)

Pin	Description	Pin	Description
1	NC	2	3.3V
3	NC	4	GND
5	NC	6	NC
7	NC	8	UIM_PWR
9	GND	10	UIM_DATA
11	NC	12	UIM_CLK
13	NC	14	UIM_RESET
15	GND	16	NC
Mechanical Key			
17	NC	18	GND
19	NC	20	3G_RF_OFF#
21	GND	22	WIFI_RESET#
23	NC	24	3.3V
25	NC	26	GND
27	GND	28	NC
29	GND	30	NC
31	NC	32	NC
33	NC	34	GND
35	GND	36	USD_D-
37	GND	38	UDB_D+
39	3.3V	40	GND
41	3.3V	42	WIMAX-3G_LED#
43	GND	44	WLAN_LED#
45	NC	46	NC
47	NC	48	NC
49	NC	50	GND
51	NC	52	3.3V

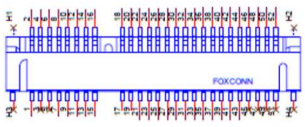


Figure 2.14 miniPCIe

2.2.2.16. SIM Socket (SIM)

RSB-4760 supports on board SIM socket is for 3G/4G integration. Please insert valid SIM card to dial to network.

Pin	Description	Pin	Description
C1	UIM_PWR	C2	UIM_RESET
C3	UIM_CLK		
C5	GND	C6	NC
C7	UIM_DATA	SW1	NC
SW2	NC		

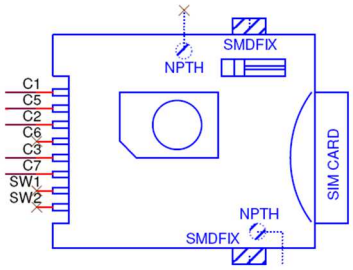


Figure 2.15 SIM Socket

2.2.2.17. SD Socket (SD1)

RSB-4760 supports SD/MMC card in Class2, 4, 6, 8, 10. Supported capacity is up to 32G(SDHC)

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

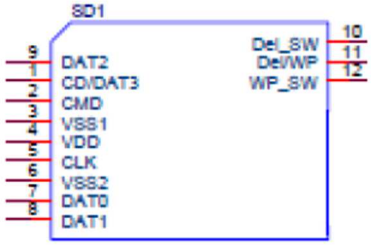


Figure 2.16 SD Slot

2.2.2.18. M.2 (CN3)

RSB-4760 supports M.2 2230 Key.E slot with I2C, UART and USB interface.

Pin	Description	Pin	Description
1	GND	2	+V3.3V
3	USB7_D+	4	+V3.3V
5	USB7_D-	6	M.2_WLAN_LED#
7	GND	8	NC
9	NC	10	NC
11	NC	12	NC
13	NC	14	NC
15	NC	16	M.2_BT_LED#
17	NC	18	GND
19	NC	20	NC
21	NC	22	UART1_RXD
23	NC		
Mechanical Key			
		32	UART1_TXD
33	GND	34	UART1_CTS#
35	NC	36	UART1_RTS#
37	NC	38	NC
39	GND	40	NC
41	NC	42	NC
43	NC	44	NC
45	GND	46	NC
47	NC	48	NC
49	NC	50	NC
51	GND	52	NC
53	NC	54	M.2_BT_X_OFF#
55	NC	56	M.2_WLAN_X_OFF#
57	GND	58	I2C2_SDA
59	NC	60	I2C2_SCL
61	NC	62	I2C1_ALERT#
63	GND	64	NC
65	NC	66	NC
67	NC	68	NC

69	GND	70	NC
71	NC	72	+3.3V
73	NC	74	+3.3V
75	GND		

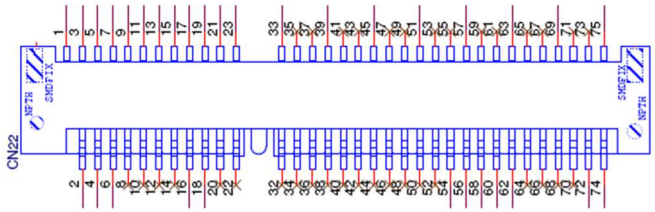
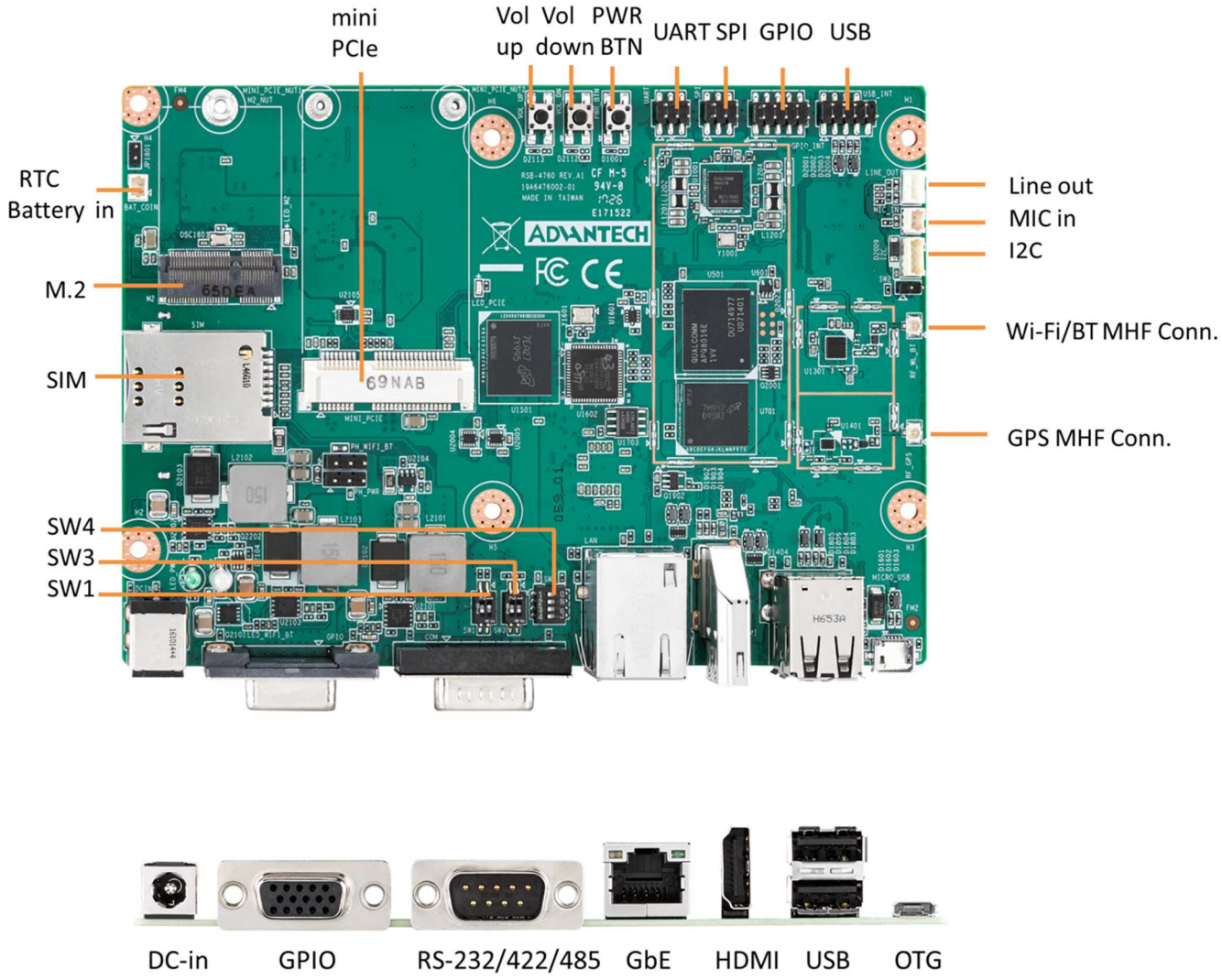


Figure 2.17 M.2 Connector

2.3. Mechanical

2.3.1. Jumper and Connector Location



2.3.2. Board Dimensions

2.3.2.1.Board Drawing

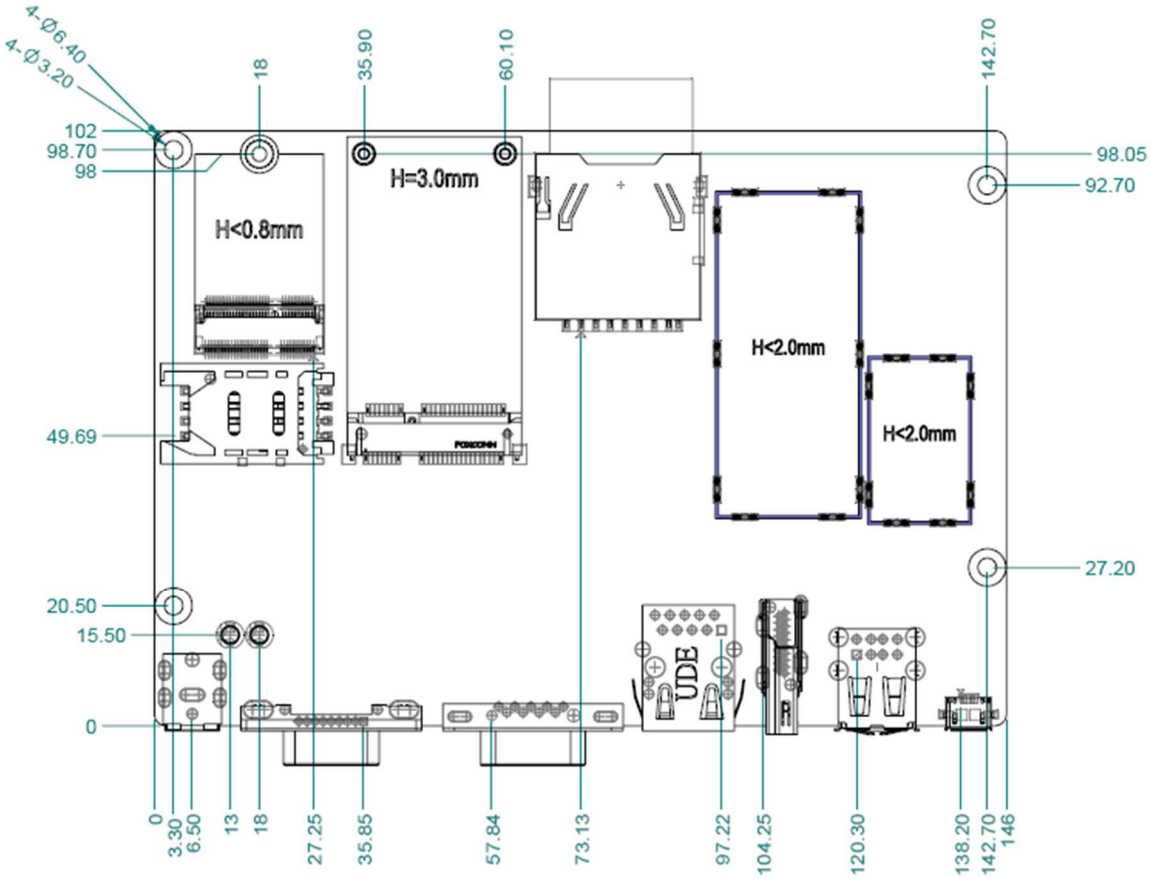


Figure 2.18 Board Dimension Layout (Top Side)

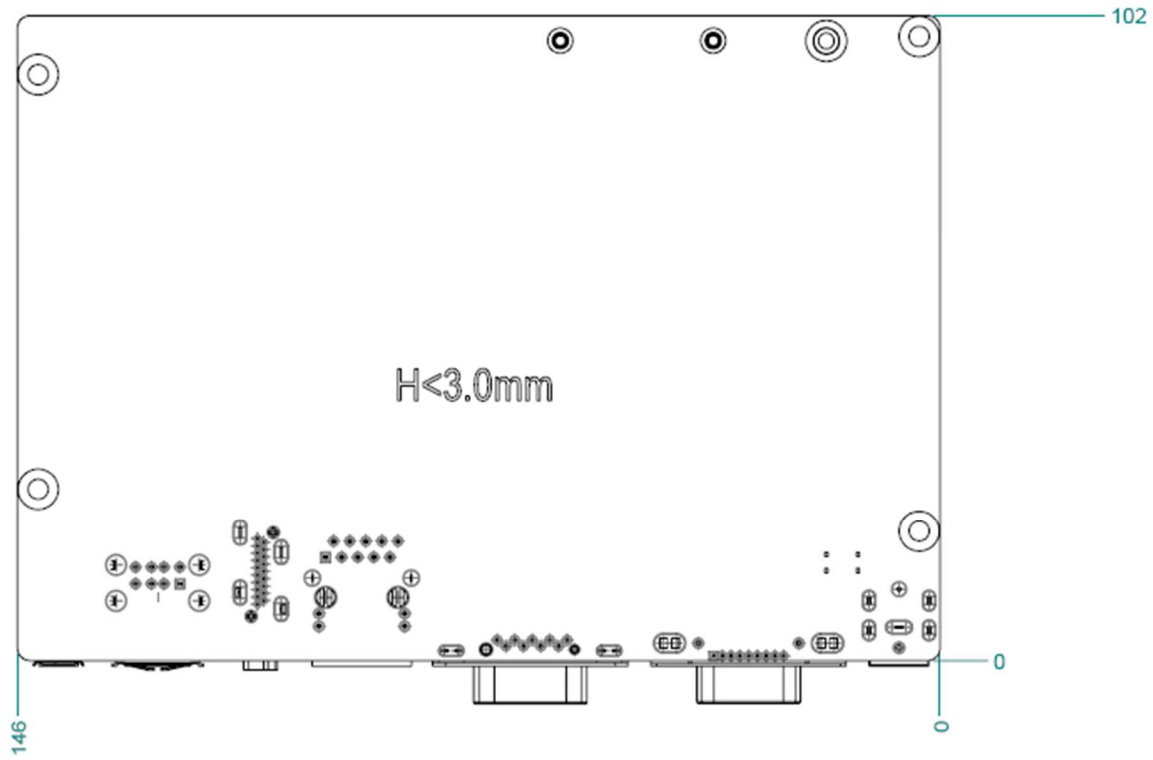


Figure 2.19 Board Dimension Layout (Bottom Side)

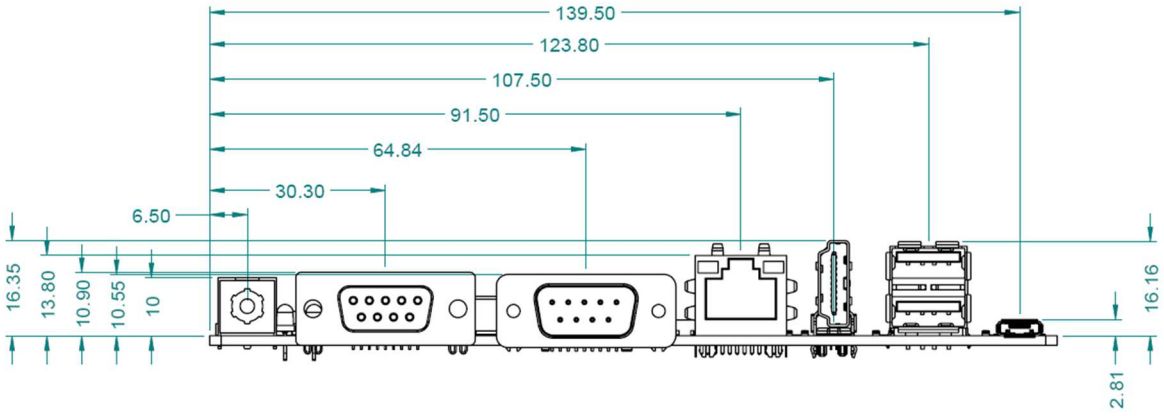


Figure 2.20 Board Dimension Layout (Coastline)

2.4. Quick Start of RSB-4760

2.4.1. Debug Port Connection

1. Connect debug port cable to the RSB-4760 debug connector.
2. Connect the ROM-ED20 extension board via cable.
3. Connector the other sides of the extension cable to the USB-to-RS-232 cable then connect to your PC.

2.4.2. Debug Port Setting

RSB-4760 can communicate with a host server (Windows or Linux) by using serial cables. Common serial communication programs such as Hyper Terminal, Tera Term or PuTTY can be used in this case. The example below describes the serial terminal setup using Hyper Terminal on a Windows host:

1. Connect RSB-4760 with your Windows PC by using a serial cable.
2. Open Hyper Terminal on your Windows PC, and select the settings as shown in Figure 2.25.
3. After the bootloader is programmed on SD card, insert power adapter connector to DC jack on RSB-4760 to power up the board. The bootloader prompt is displayed on the terminal screen.

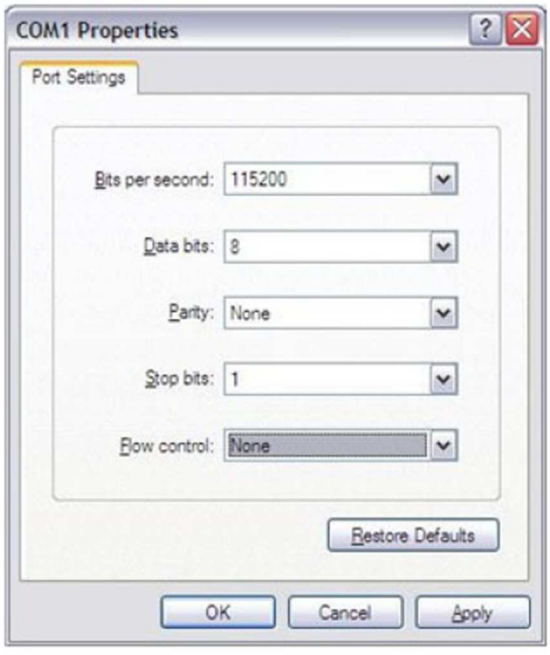


Figure 2.21 Hyper Terminal Settings for Terminal Setup

3.1. Test Tools

3.1.1. Display Test

On RSB-4760, it only supports one HDMI display.

3.1.1.1 Resolution Adjustment

[X Windows]

You can change the HDMI resolution by this command in kernel.

```
$ abootimg -u /dev/disk/by-partlabel/boot -c "cmdline=root=/dev/mmcblk0p10
rw rootwait console=ttyMSM0,115200n8
drm_kms_helper.edid_firmware=HDMI-A-1:edid/1920x1080.bin"
```

Currently, you can set the following resolutions.

```
1920x1080
1600x1200
1680x1050
1280x1024
1024x768
800x600
```

[Framebuffer Console]

If you want to change resolution for console mode, you can use this command for any resolution values.

```
$ abootimg -u /dev/disk/by-partlabel/boot -c "cmdline =
root=/dev/mmcblk0p10 rw rootwait console=ttyMSM0,115200n8
video=HDMI-A-1:640x480@60"
```

3.1.2. Video Playback

[AVI]

```
$ gst-play-1.0 test.avi
```

[MP4]

```
$ GST_GL_PLATFORM=egl gst-launch-1.0 filesrc location=/home/root/test.mp4 !
qtdemux name=m m.video_0 ! h264parse ! v4l2video0dec ! Glimagesink
```

3.1.3. Audio Test

3.1.3.1.Audio Playback

Refer to /tools/audio_playback.sh

```
$ amixer -c 0 cset iface=MIXER,name='RX1 MIX1 INP1' 'RX1'
$ amixer -c 0 cset iface=MIXER,name='RX2 MIX1 INP1' 'RX2'
$ amixer -c 0 cset iface=MIXER,name='RDAC2 MUX' 'RX2'
$ amixer -c 0 cset iface=MIXER,name='HPHL' 1
$ amixer -c 0 cset iface=MIXER,name='HPHR' 1
$ amixer -c 0 cset iface=MIXER,name='RX1 Digital Volume' 127
$ amixer -c 0 cset iface=MIXER,name='RX2 Digital Volume' 127
$ aplay -D plughw:0,1 test.wav
```

If you want to play audio on HDMI audio channel, you can see the DHMI audio is at card 0, device 0.

```
$ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: DB410c [DB410c], device 0: ADV7533 adv7511-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: DB410c [DB410c], device 1: WCD msm8916_wcd_i2s_rx1-1 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

Then, you can play as below.

```
$ aplay -D plughw:0,0 test.wav
```

3.1.3.2.Audio Recording

Refer to /tools/audio_recording.sh

```
$ amixer -c 0 cset iface=MIXER,name='DEC1 MUX' 'ADC2'
$ amixer -c 0 cset iface=MIXER,name='ADC2 Volume' 70
$ amixer -c 0 cset iface=MIXER,name='ADC2 MUX' 'INP2'
$ arecord -D plughw:0,2 -r 16000 -f S16_LE ./f-16000.wav
```

3.1.4. GPIO Test

3.1.4.1. Export GPIO

```
$ cd /sys/class/gpio
# GPIO 1~8 (DB9)
$ echo 8 > export
$ echo 9 > export
$ echo 12 > export
$ echo 13 > export
$ echo 24 > export
$ echo 25 > export
$ echo 26 > export
$ echo 27 > export

# GPIO 9~16 (GPIO_INT)
$ echo 28 > export
$ echo 33 > export
$ echo 34 > export
$ echo 35 > export
$ echo 36 > export
$ echo 69 > export
$ echo 73 > export
$ echo 108 > export
```

3.1.4.2. Loopback Test

1. Connect two GPIO pins respectively. For example, we connect GPIO 1 & 2 together.
2. Set directions

```
$ echo out > gpio1/direction
$ echo in > gpio2/direction
```

3. Get values (default should be 0)

```
$ cat gpio2/value
```

4. Set values & verify

```
$ echo 1 > gpio1/value
$ cat gpio2/value
```

The values of GPIO 1 & 2 should be the same.

3.1.5. I2C Test

3.1.5.1. I2C Mapping

I2C1	blsp_i2c6: i2c@78ba000	0x30: External RTC
I2C2	cci_i2c0: i2c@1b0c000	M.2
I2C3	blsp_i2c4: i2c@78b8000	0x39: AD7535 (HDMI)
I2C4	blsp_i2c3: i2c@78b7000	Pin Header

3.1.4.2 Test

List all I2C buses

```
$ i2cdetect -l
i2c-0   i2c             QUP I2C adapter                         I2C
adapter
i2c-2   i2c             QUP I2C adapter                         I2C
adapter
i2c-3   i2c             QUP I2C adapter                         I2C
adapter
```

Note: I2C2 (i2c-1) now is not available.

Check ADV7535 status.

```
$ i2cdetect -r -y 2
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- 38 UU -- -- UU -- -- UU
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

You can get values by:

```
$ i2cget -f -y 2 0x39 0 w
0x0014
```

Dump EDID by i2cdump:

```
$ i2cdump -f -y 2 0x39
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: 14 00 00 00 00 00 00 00 00 00 41 0e bc 18 01 13    ?.........A?????
10: 25 37 00 00 00 00 20 00 46 62 04 a8 00 00 1c 84    %7.... .Fb??..??
20: 1c bf 04 a8 1e 70 02 1e 00 00 04 a8 08 12 1b ac    ?????p??..??????
30: 00 00 00 00 00 00 00 00 00 00 00 80 00 10 40 00    ............?.?@.
40: 00 10 f0 7e 10 70 70 70 70 00 80 00 00 00 00 00    .??~?pppp.?.....
50: 00 00 02 0d 6d 02 00 00 00 00 00 00 00 00 00 00    ..??m?..........
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
70: 01 0a 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ??..............
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
90: 00 00 00 00 84 80 20 00 03 02 e0 18 38 61 1a 00    ....?? .????8a?.
a0: 00 00 a0 a0 08 04 00 00 00 00 00 40 00 00 40 16    ..????.....@..@?
b0: 00 00 00 00 00 00 00 00 00 00 70 00 00 00 00 00    ..........p.....
c0: 00 00 00 00 00 10 16 00 02 03 00 00 02 00 01 04    .....??.??..?.??
d0: 30 ff 80 80 80 00 00 00 00 00 00 00 00 82 01       0.???.........??
e0: 80 78 01 00 40 80 fd 00 00 00 52 46 00 00 00 00    ?x?.@??...RF....
f0: 95 04 ff 00 00 00 00 00 00 10 7d aa 1c 00 b0 00    ??.......?}??.?.
```

3.1.6. USB Test

3.1.6.1. USB Port Mapping

USB Port 1	SMSC_LAN7500
USB Port 2	USB 0
USB Port 3	USB 1
USB Port 4	Mini PCIe (3G)
USB Port 5	USB_INT (pin 1,3,5)
USB Port 6	USB_INT (pin 2,4,6)
USB Port 7	M.2

3.1.6.2. Test

1. Insert USB disk
2. Run:

```
# Generate random file
$ dd if=/dev/urandom of=data bs=1 count=1024
# Backup
$ dd if=/dev/sda of=backup bs=1 count=1024 skip=4096
# Write data
$ dd if=data of=/dev/sda bs=1 seek=4096
```

3. Read & verify

```
$ dd if=/dev/sda of=data1 bs=1 count=1024 skip=4096
$ diff data data1
```

4. Restore

```
$ dd if=backup of=/dev/sda bs=1 seek=4096
```

3.1.7. RTC Test

Set system time and write to RTC.
The date format is MMDDhhmm[[CC]YY][.ss]

```
$ date 021710452016 && hwclock -w & date
```

Set incorrect time and read time from RTC to verify

```
$ date 010100002000 && hwclock -r & date
```

Restore the RTC time to system time

```
$ hwclock -s && date
```

3.1.7.1.Switch to external RTC

1. Check RTC1 info

```
$ udevadm info -a -p /sys/class/rtc/rtc1
  looking at device
'/devices/platform/soc/78ba000.i2c/i2c-0/0-0030/rtc/rtc1':
    KERNEL=="rtc1"
    SUBSYSTEM=="rtc"
    DRIVER==""
```

```
ATTR{hctosys}=="0"
ATTR{max_user_freq}=="64"
ATTR{name}=="rtc-s35390a"
ATTR{wakealarm}==""
```

2. Add udev rule

```
$ vi /etc/udev/rules.d/99-rtc1.rules
```

KERNEL=="rtc1", SUBSYSTEM=="rtc", DRIVER=="", ATTR{name}=="rtc-s35390a ", SYMLINK="rtc", MODE="0666"

3. Run the rule

```
$ udevadm test /sys/class/rtc/rtc1
```

...

creating link '/dev/rtc' to '/dev/rtc1'

atomically replace '/dev/rtc'

…

4. Confirm

```
$ ls -al /dev/rtc*
lrwxrwxrwx 1 root root      4 May 19  2017 /dev/rtc -> rtc1
crw------- 1 root root 254, 0 May 19  2017 /dev/rtc0
crw------- 1 root root 254, 1 Jan  1 02:10 /dev/rtc1
```

3.1.8. MMC (eMMC/SD) Test

3.1.8.1.Read/Write Operations

It’s simple to verify read/write operations for MMC devices.

You can use many tools to do this, e.g. dd, vi, cp, etc.

eMMC: /dev/mmcblk0

SD: /dev/mmcblk1

3.1.8.2.Write Protect

We support write protect feature for SD card.

Once a SD card with write proctection is inserted, you can see “RO” message in kernel log.

```
[ 8086.082015] mmc1: new high speed SDHC card at address e624
[ 8086.084105] mmcblk1: mmc1:e624 SU04G 3.69 GiB (ro)
[ 8086.113996]  mmcblk1: p1 p2 p3 p4 p5 p6
```

3.1.9. Ethernet Test

3.1.9.1. Interface

Run the following commands to test

```
$ ifconfig
$ ping 8.8.8.8
```

3.1.9.2.Change MAC Address

You can change the MAC address in EEPROM by running:

```
$ /tools/update-mac_smsc75xx.sh
update-mac_smsc75xx.sh {Interface} {Mac Address}
Ex: update-mac_smsc75xx.sh ${ETH_IF} 06 05 04 03 02 01
```

3.1.9 RS-232/485/422 Test

To configure the mode of RS232/422/485 transceiver, you have to export GPIO 99 & 100 first.

	Loopback	RS-232	RS-485	RS-422
CRTL_MODE0 (GPOIO 99)	0	1	0	1
CRTL_MODE1 (GPOIO 100)	0	0	1	1

For example, you can set GPIO 99 as 1 and GPIO 100 as 0 to enable RS-232 mode.

[RS-232]

```
$ stty -F /dev/ttyMSM1 -echo -onlcr 115200 crtscts
$ cat /dev/ttyMSM1 &
$ echo "Serial Port Test" > /dev/ttyMSM1
```

[RS-485/422]

Send/Receive packages in loopback or connect mode

```
$ st-fsl /dev/ttyMSM1 -b 115200 -m 485 -g 20 -f none -c n81 -rop &
```

```
$ st-fsl /dev/ttyMSM1 -b 115200 -m 485 -g 5 -f none -c n81 -soa
```

PS. Change to `-m 422` for RS-422 test

3.1.10. Watchdog Test

We build in a demo program for watchdog.

```
$ susidemo4
*********************************************
**               SUSI4.0 demo              **
*********************************************

Main (demo version : 4.0.14490.0)

0) Terminate this program
1) Watch Dog
2) HWM
3) GPIO
4) VGA
5) I2C
6) Information

Enter your choice:
```

Then, choose **(1) Watch Dog**

```
Watchdog 1

0) Back to Main menu
1) Select watchdog timer: [0]
2) Start
3) Trigger
4) Stop

Enter your choice:
```

Choose **(2)** to start

```
Start Watchdog:
```

```
0) Back to Watch Dog menu
1) Delay time (0 to 64000): 0
2) Reset time (0 to 128000): 0
3) Run

Enter your choice:
```

Set Delay time to 10000 (10 sec) and Reset time to 5000 (5 sec), then device will reboot in 5 seconds.

Set Delay time to 10000 (10 sec) and Reset time to 11000 (11 sec), then you will see the following message printed out every 10 seconds. That means watchdog is reset.

```
IMX6D_WDT_SignalRoutine
```

To leave this test mode, you can press ENTER and choose **4) Stop** to disable watchdog function.

3.1.11. SPI Test

Write/Read/Verify data in the start/end 4 byte of flash

Note: Make sure the content of SPI Nor flash is erased.

```
$ echo -n $'\x06\x05\x04\x03\x02\x01' > test
$ dd if=test of=/dev/mtd0
$ hexdump -C /dev/mtd0 -n 64
```

3.1.12. 3G Test

1. Insert 3G modules and boot up
2. To verify with AT command, you can run:

```
$ stty -F /dev/ttyUSB1 -echo
$ cat /dev/ttyUSB1 &
$ echo AT+CSQ > /dev/ttyUSB1
```

3. To establish data connection, you can refer to our test scripts.

[/tools/ewm-c106.sh]

- PPP connection

- /etc/ppp/peers/3glink
- /etc/chatscripts/3g.chat

[/tools/telit3g.sh]
- ECM mode

3.1.13. WiFi Test

Refer to /tools/thermal.sh

```
#!/bin/sh

SSID="WISE-work"
WPA_KEY="advantech"

rfkill unblock all
killall wpa_supplicant
rm /etc/resolv.conf
ifconfig wlan0 up

wpa_passphrase ${SSID} ${WPA_KEY} > /tmp/wpa.conf
wpa_supplicant -BDwext -iwlan0 -c/tmp/wpa.conf
udhcpc -b -i wlan0
```

3.1.14. Bluetooth Test

Use hcitools for test

```
$ hciconfig hci0 up
$ bluetoothctl
$ discoverable on
$ pairable on
$ scan on
[NEW] 98:0D:2E:83:DE:80 myphone
$ scan off
$ pair 98:0D:2E:83:DE:80
$ connect 98:0D:2E:83:DE:80
```

3.1.15. LED Test

1. Check if LED triggers are expected.

[WiFi]

```
$ cat /sys/class/leds/apq8016-sbc:green:wlan/trigger
none kbd-scrollock kbd-numlock kbd-capslock kbd-kanalock kbd-shiftlock
kbd-altgrlock kbd-ctrllock kbd-altlock kbd-shiftllock kbd-shiftrlock
kbd-ctrlllock kbd-ctrlrlock mmc0 mmc1 timer oneshot heartbeat backlight gpio
cpu0 cpu1 cpu2 cpu3 default-on hci0-power hci0-tx_rx rfkill0 rfkill1 [phy0rx]
phy0tx phy0assoc phy0radio
```

[Bluetooth]

```
$ cat /sys/class/leds/apq8016-sbc:yellow:bt/trigger
none kbd-scrollock kbd-numlock kbd-capslock kbd-kanalock kbd-shiftlock
kbd-altgrlock kbd-ctrllock kbd-altlock kbd-shiftllock kbd-shiftrlock
kbd-ctrlllock kbd-ctrlrlock mmc0 mmc1 timer oneshot heartbeat backlight gpio
cpu0 cpu1 cpu2 cpu3 default-on hci0-power [hci0-tx_rx] rfkill0 rfkill1 phy0rx
phy0tx phy0assoc phy0radio
```

2. Verify LED behavior

For example, we set LED trigger as phy0rx for WiFi. That means LED should be blink when data is receiving via WiFi.

3.1.16. M.2 Test

We take M.2 UART function as example. Here, we use AzureWave AW-NB136NF (M.2 Uart BT module).

Because the serial port conflicts with debug console (UART0). So, you have to disable serail-getty service, before you test.

```
$ systemctl mask serial-getty@ttyMSM0.service
$ reboot
$ hciattach /dev/ttyMSM0 bcm43xx 115200 flow
$ hciconfig hci1 up
# You can see a BT hci now. Follow the BT test steps.
```

After test, you can recover by the command below.

```
$ systemctl unmask serial-getty@ttyMSM0.service
$ reboot
```

3.1.17. GPS Test

To start GPS hardware, follow the instructions.

1. Set correct date/time before starting GPS
2. Run

```
$ systemctl start gpsd.socket
$ systemctl start gpsd
$ systemctl start qdsp-start
$ systemctl start gnss-gpsd
$ gpsmon
```

To stop GPS hardware, follow the instructions:

1. close gpsmon
2. systemctl stop gnss-gpsd

To restart GPS hardware, follow the instructions:

1. systemctl start gnss-gpsd
2. close gpsmon

4. Software Functionality

4.1. Package Content

An official build release consists of the following contents. We take V1.050 for RSB-4760 as example.

Filename	***Usage***
4760LIV1050_sd_install.img.gz	Installation SD card image
4760LIV1050_2017-05-19_misc.tgz	Kernel, DTS & kernel modules
4760LIV1050_2017-05-19.tgz	Image binaries (boot image & rootfs)
4760LIV1050_2017-05-19.csv	Official build information
4760LBV1050_2017-05-19_sdk.tgz	SDK installer
4760LBV1050_2017-05-19.tgz	BSP source code, including bootloader binaries
*.md5, log.tgz & *.log	MD5 checksum & Log files

4.2. Setup Build Environment

Currently, we adopt Docker as build environment.

You can get the latest version of advrisc/u14.04-410clbv1 Docker image for developing Qualcomm APQ8016 projects.

If you don't know much about Docker, please refer to IoTGateway/Docker for details.

4.2.1. Conventions

${BSP_HOME} : home directory of the BSP

${BDIR} : build directory (e.g. build/)

${MACHINE} : available target boards list below

- rsb-4760
- epc-r4761

${DISTRO} : Linux distribution

- rpb
- rpb-wayland

${RPB-IMAGES} : meta-rpb provides the following images

- rpb-console-image
- rpb-desktop-image
- rpb-minimal-image
- rpb-qt5-image
- rpb-weston-image

4.2.2. Board Support Package (BSP)

You have two methods to put BSP into Docker container.

1. Download BSP from GitHub

The oe-rpb-manifest @GitHub is our manifest repository for Qualcomm APQ8016 projects. You can use **repo init** & **repo sync** to get the entire BSP we put on GitHub.

Create your own BSP folder first.

```
$ mkdir ${BSP_HOME}
$ cd ${BSP_HOME}
```

To get the latest version of each meta-layers, you can use default.xml.

```
$ repo init -u https://github.com/ADVANTECH-Corp/oe-rpb-manifest.git -b morty
```

To get an official release version, you can assign a specific xml, e.g. 410cLBV1050.xml.

```
$ repo init -u https://github.com/ADVANTECH-Corp/oe-rpb-manifest.git -b morty
-m 410cLBV1050.xml
```

Finally, pull down the BSP by running

```
$ repo sync
```

2. Copy BSP tarball into Container

You can use `docker cp` to do this.

```
// Copy BSP source code into container

$ docker cp 410cLBV1050_2017-05-19.tgz <your container id>:/home/adv
```

Or Use *docker run* with **-v** options to enable data volume. Then, you are able to put the BSP into the data volume folder.

For example, you can put the BSP into /home/root/workspace folder.

```
$ docker run -it --name docker_test -v /home/root/workspace:/home/adv/BSP
advrisc/u14.04-410clbv1:20170605/bin/bash
```

4.2.2.1.BSP Content

The descriptions of some important folders in BSP are listed below:

bitbake/ : Yocto build command

bootloader/ : Bootloader binaries including of CDT, SBL, RPM, LK, etc.

layers/ : Sources for meta-layers

- **meta-96boards/ :** meta layer of 96Boards definitions
- **meta-advantech/ :** meta layer by Advantech
- **meta-qcom/ :** meta layer for APQ8016 configurations
- **meta-rpb/ :** meta layer for RPB distro

setup-environment : to set up build environment for Yocto

4.2.2.2.Naming Rule

The tarball/prebuilt image name consists of the model name followed by "LB" or "LI" plus version number and released date.

For example, **BSP file name: 410cLBV1050_2017-05-19.tgz**

Which "410c" means Dragonboard**410c** which a demo board for APQ8016 we use

"LB" is acronym of **L**inux **B**SP,

"V1050" stands for **V**ersion **1**.**050**.

Another example, **Yocto image name: 4760LIV1040_2017-04-20.tgz**

which "4760" stands for **RSB-4760**

"LI" is acronym for prebuilt **L**inux **I**mage.

4.2.2.3.Pre-built Images

In **LIV** tarball file, you can get binary images. For example,

boot-Image--4.4-r0-rsb-4760-20170426102349-52.img : Boot Image

rpb-desktop-image-rsb-4760-20170426102349-52.rootfs.img : Root filesystem

In BSP tarball file, you can see a zip file in bootloader/ folder. It is bootloader binary images.

advantech_bootloader_emmc_linux-72/

cdt_1.1_MT52L256M32D1PF.bin : DDR CDT parameters

emmc_appsboot.mbn : Little Kernel (LK) responsible to load Linux kernel

flashall : Shell script to flash bootloader binaries

gpt_both0.bin : GPT partition table

hyp.mbn : Trust Zone / QHEE

rpm.mbn : Resource and Power Management (RPM)

sbl1.mbn : Secondary Boot Loader (SBL)

tz.mbn : Trust Zone / QSEE

4.2.3. Build Instructions

4.2.3.1.Create New Build Environment

To create one new build environment, perform following commands in terminal console:

```
$ cd ${BSP_HOME}
$ MACHINE=${MACHINE} DISTRO=${DISTRO} source setup-environment ${BDIR}
```

You need to read and accept the EULA. Press "Y"

4.2.3.2.Load Existed Build Environment

To continue an existed build environment, perform following commands in terminal console:

```
$ cd ${BSP_HOME}
$ source setup-environment ${BDIR}
```

4.2.3.3.Build Images

To build the desktop image, run

```
$ bitbake rpb-desktop-image
```

4.2.3.4.Build Toolchain Installer

To build SDK toolchain installer, perform following commands in terminal console:

```
$ bitbake ${RPB-IMAGES} -c populate_sdk
```

4.2.3.5.Build Bootloader

Refer to Debian_16.09 " How to get and customize the bootloader "

```
$ git clone
git://codeaurora.org/platform/prebuilts/gcc/linux-x86/arm/arm-eabi-4.8.git
-b LA.BR.1.1.3.c4-01000-8x16.0

$ git clone http://git.linaro.org/landing-teams/working/qualcomm/lk.git -b
debian-qcom-dragonboard410c-LA.BR.1.2.4-00310-8x16.0-linaro2

$ cd lk

$ make -j4 msm8916 EMMC_BOOT=1 TOOLCHAIN_PREFIX=<path to arm-eabi-4.8
tree>/bin/arm-eabi-
```

4.2.3.6.Build Linux Kernel

To build kernel only, run

```
$ bitbake linux-linaro-qcomlt
```

4.2.4. Flash Pre-built Images

There are 3 different ways to flash images into target eMMC.

4.2.4.1.USB Download Tools

1. Install DragonBoardUpdateTool 32 bit, 64 bit

2. Switch USB Download Mode (SW1: 1,1), power on and then connect with USB Cable

3. Open Command Prompt on PC

```
> cd C:\Program Files\Qualcomm\DragonBoardUpdateTool
```

4. Check COM port

```
> emmcdl.exe -l
```

5. Flash entire system

```
> emmcdl.exe -p [Your COM Port] -f mbns\8916\prog_emmc_firehose_8916_ddr.mbn
-x rawprogram0.xml
```

if it success, it will show

```
Status: 0 The operation completed successfully.
```

6. Then, flash CDT.bin

```
> emmcdl.exe -p [Your COM Port ] -f mbns\8916\prog_emmc_firehose_8916_ddr.mbn
-x rawprogram2.xml
```

7. Finally, switch to SW1:(0,1) and boot from eMMC normally.

4.2.4.2.Fastboot Tool

1. Check device id

```
$ sudo fastboot devices
```

2. Flash bootloader binaries

```
$ sudo ./flashall [Devices ID]
```

3. Erase boot / rootfs partitions

```
$ sudo fastboot erase boot

$ sudo fastboot erase rootfs
```

4. Flash images to boot & rootfs partitions

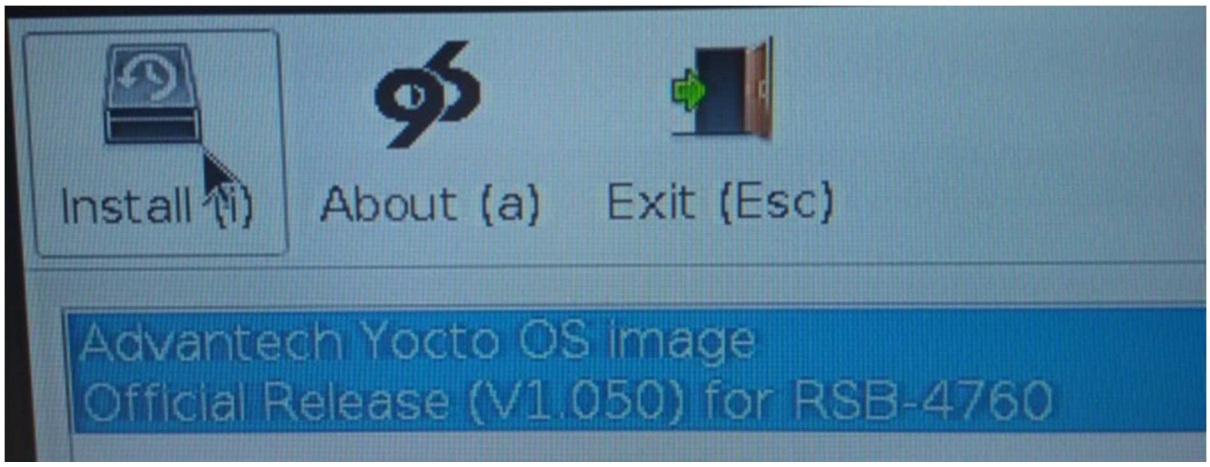
```
$ sudo fastboot flash boot <Your boot image>

$ sudo fastboot flash rootfs <Your boot image>
```

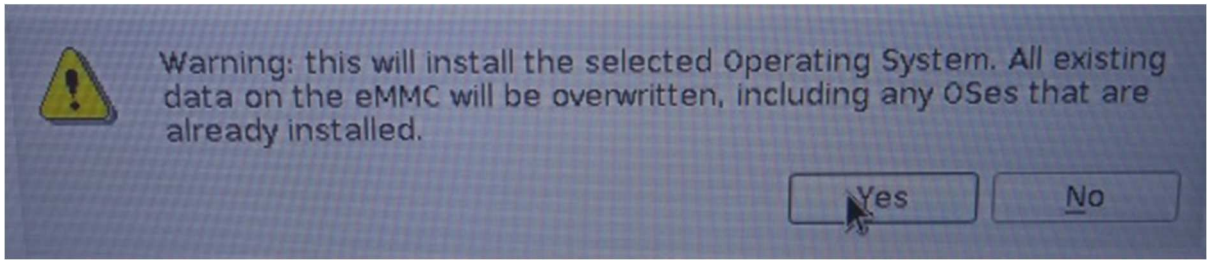
4.2.4.3.Installation SD Card

1. Download specific **sd_install.img.gz** into SD card.

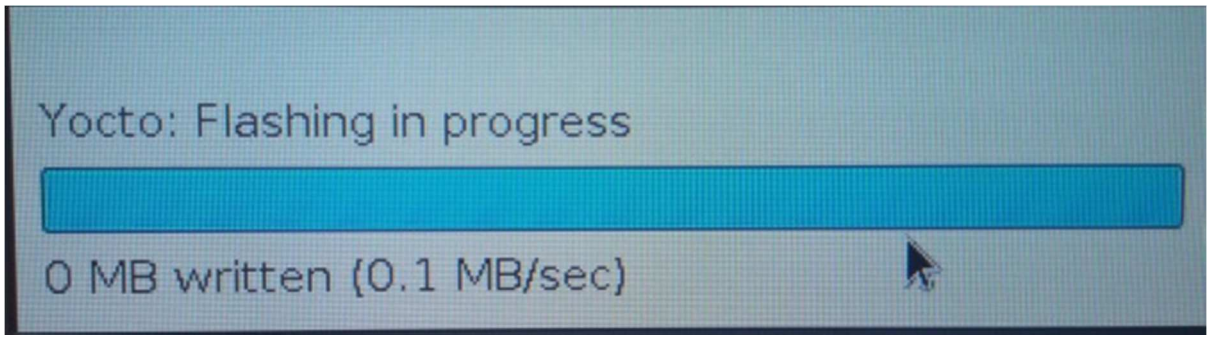
2. Boot from installation SD Card, and Click "Install"



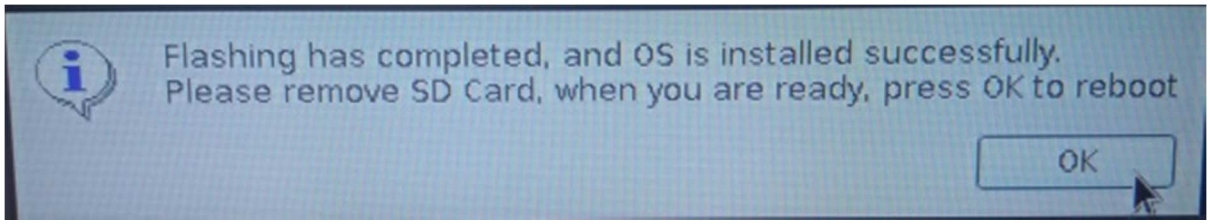
3. Flash all partitions



4. Installing



5. Finish and Reboot


Flashing has completed, and OS is installed successfully.
Please remove SD Card, when you are ready, press OK to reboot
OK