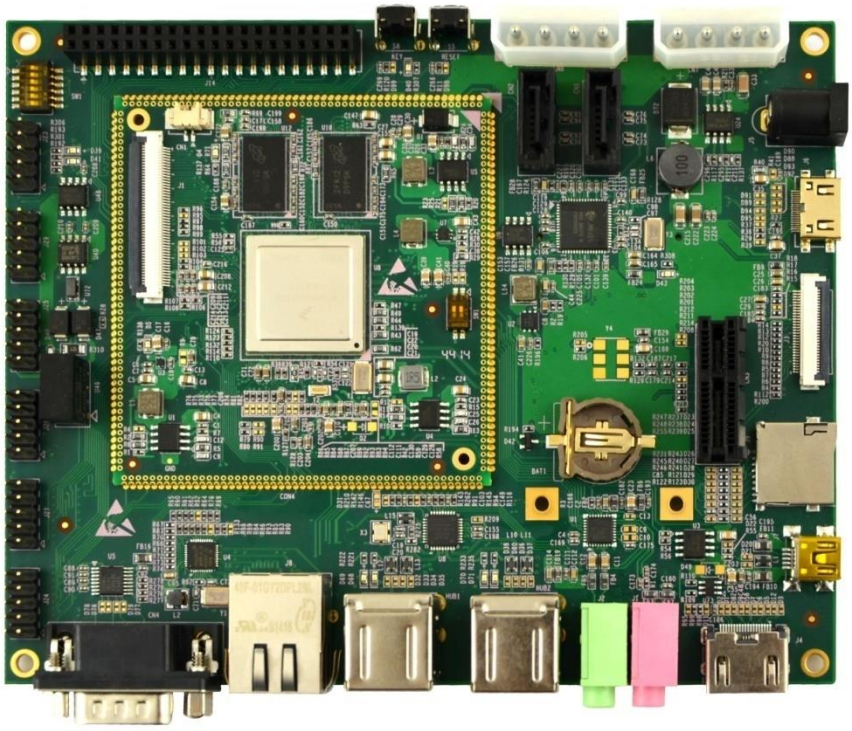
SBC9000
Single Board Computer



User Manual

Version 1.0 – Jan. 20th, 2015

Copyright Statement:

- SBC9000 and its related intellectual property are owned by Shenzhen Embest Technology Co., Ltd.
- Shenzhen Embest Technology has the copyright of this document and reserves all rights. Any part of the document should not be modified, distributed or duplicated in any approach and form without the written permission issued by Embest Technology Co., Ltd.
- The use of Microsoft, MS-DOS, Windows, Windows95, Windows98, Windows2000, Windows XP, and Windows Embedded Compact 7 is licensed by Microsoft.

Disclaimer:

- Shenzhen Embest Technology does not take warranty of any kind, either expressed or implied, as to the program source code, software and documents provided along with the products, and including, but not limited to, warranties of fitness for a particular purpose; The entire risk as to the quality or performance of the program is with the user of products.

Revision History:

Version	Date	Description
1.0	2015-1-20	Original Version

Table of Contents

Chapter 1 Product Overview

1.1 About SBC9000

SBC9000 is an embedded single board computer designed by Embest Technology based on i.MX 6Quad processor. It comes in a configuration of a CPU module Mini9000 + an Expansion board which provides 4 serial ports (one of them supports RS232 debugging), 2 isolated CAN2.0 interfaces, 1 gigabit Ethernet, 1 SPI, 3 I2C, 1 SDIO, 2 SATA, 1 PCIe, 1 Mini PCIe, 4 USB Host and 1 OTG, 1 LCD and 1 LVDS display interface, 1 HDMI, audio input/output, TF card slot, etc. SBC9000 is compatible with Linux 3.10.17 and Android 4.4.2, aiming to help developer in a wide range of areas such as industrial control, netbooks, all-in-one PCs, high-end mobile Internet devices, high-end PDAs, high-end portable medial players, game consoles and compact navigation devices.

1.2 Packing List

- SBC9000 (Mini9000 CPU module + expansion board）*1
- Cross Serial Cable
- 12V Power Adapter

1.3 Mini9000 CPU Module

1.3.1 Product Features

- **General Features**
  - Product Dimensions: 70 mm×68 mm
  - Operating Temperature: 0~70℃
  - Operating Humidity: 20% ~ 90% (Non-condensing)
  - Input Voltage: 5V
- **Processor**

- ARM Cortex™-A9 i.MX 6Quad Processor
- 32 KByte L1 Instruction Cache
- 32 KByte L1 Data Cache
- Private Timer and Watchdog
- Cortex-A9 NEON MPE (Media Processing Engine) Coprocessor
- 2D Graphics Processors

- **On-Board Memories:**
  - 4GByte eMMC
  - 4*256MB DDR3 SDRAM
- **On-Board Interfaces:**
  - TTL232×4 (4-bit×2 + 2-bit×2）
  - CAN2.0×2
  - RGMII×1
  - SPI×1
  - I2C×3
  - 8-bit SDIO×1
  - SATA×1
  - PCIe×1
  - USB Host×1
  - USB OTG×1
  - LVDS×1
  - HDMI×1
  - LCD×1
  - Audio×1
  - Camera×1
  - GPMC
  - Boot Configuration
  - GPIO

1.3.2 System Block Diagram

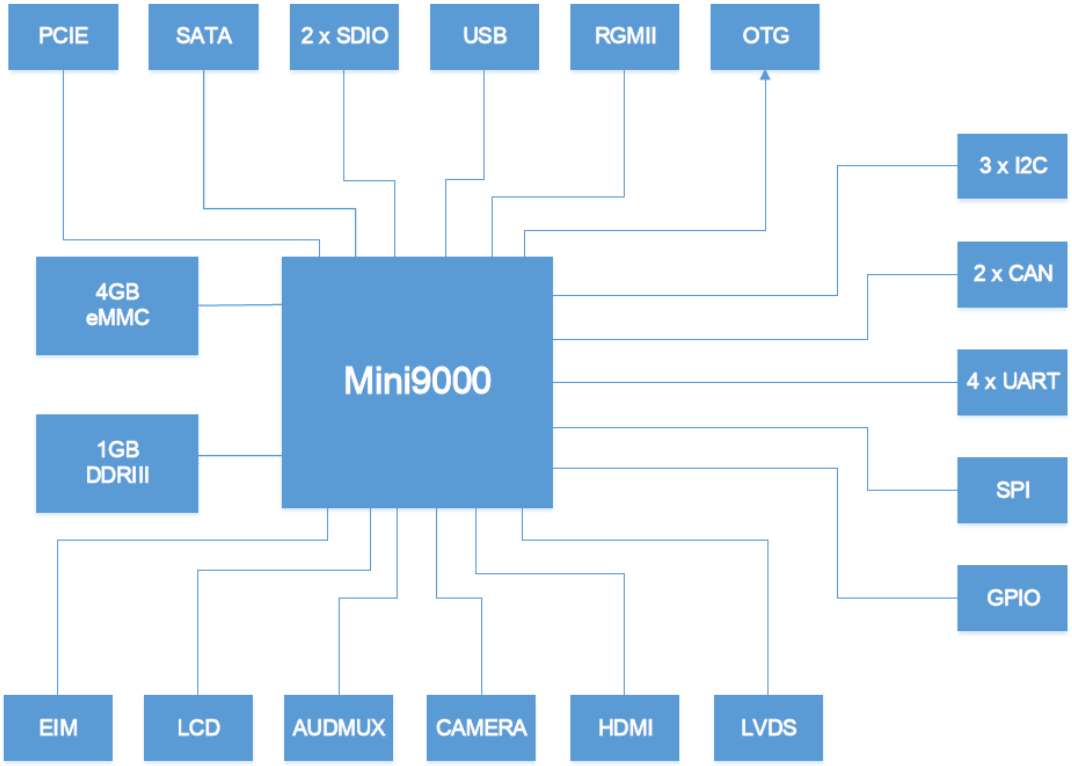


Figure 1-1 Mini9000 block diagram

1.4 Expansion Board

1.4.1 Product Features

- **General Features**
  - Product Dimensions: 150 mm×120 mm
  - Operating Temperature: 0~70℃
  - Operating Humidity: 20% ~ 90% (Non-condensing)
  - Input Voltage: 12V
- **Audio/Video Interfaces**
  - HDMI×1
  - LVDS×1
  - Audio Input (3.5mm)×1
  - Stereo Audio Output (3.5mm)×1
- **Data Transfer Interfaces**

- 10/100/1000Mbps Ethernet×1
- Isolated CAN 2.0×2
- SPI×1
- I2C×3
- SDIO×1
- SATA×2
- TF card slot×1
- PCIe×1
- mini PCIe×1
- USB host×4
- USB OTG×1
- Serial Port×4 (One of them supports DB9 debugging)

- **LEDs&Buttons**
  - User-Defined Button×1
  - Reset Button×1
  - LED Power Indicator×1
  - User-Defined LED×2

1.4.2 System Block Diagram

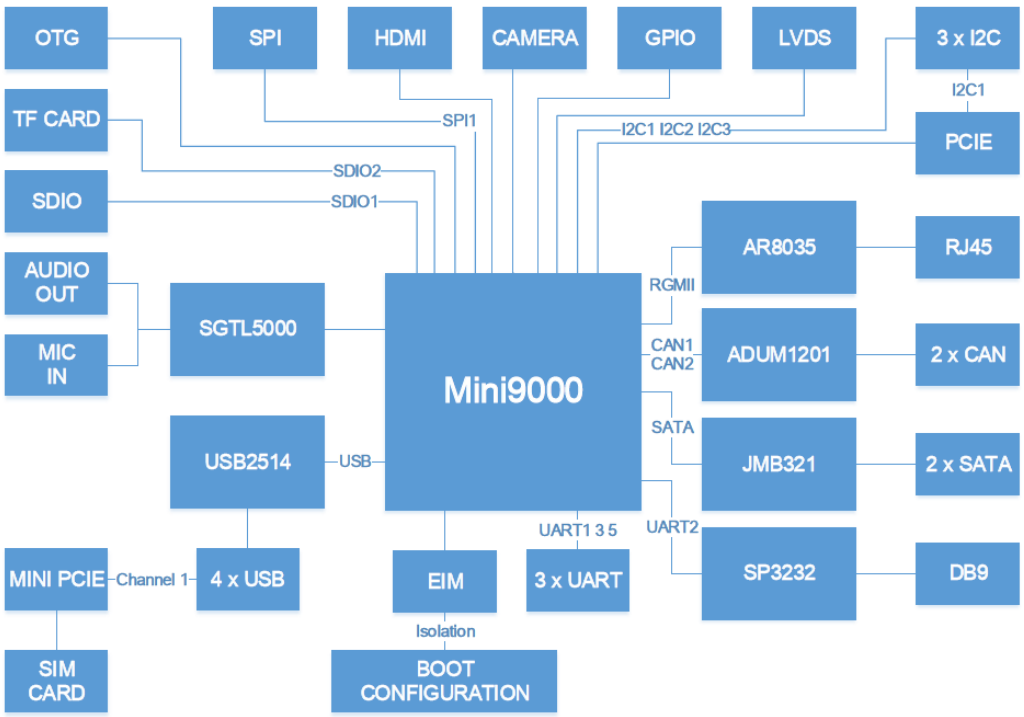


Figure 1-2 Expansion board block diagram

1.5 Mini9000 Dimensions

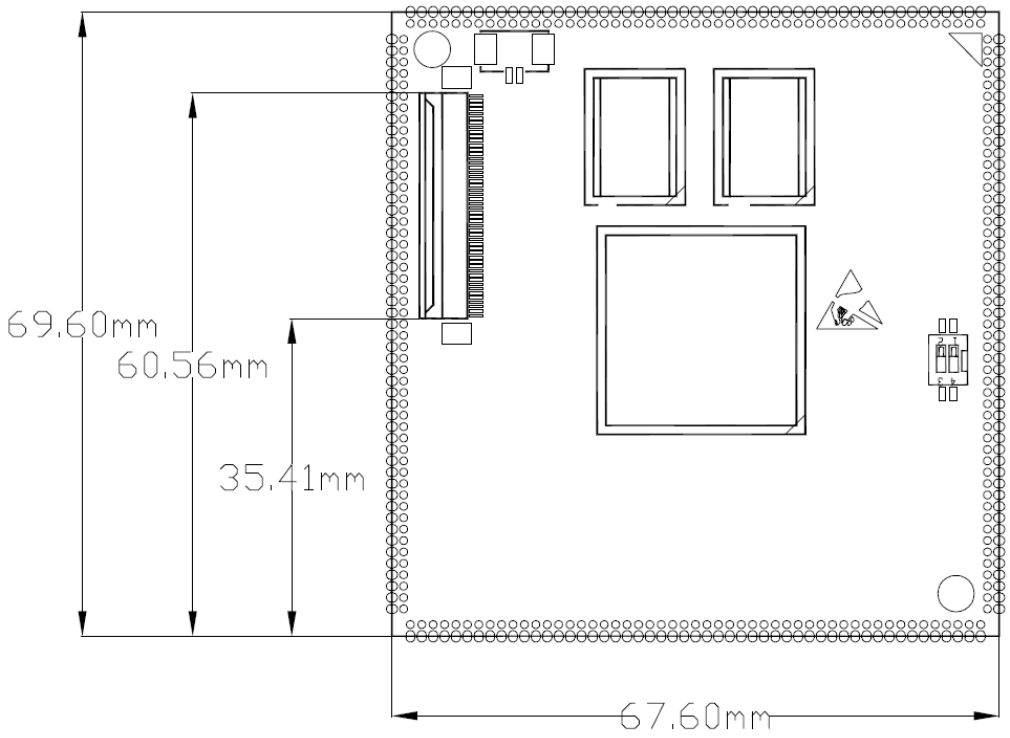


Figure 1-3 Mini9000 dimensions

1.6 Expansion Board Dimensions

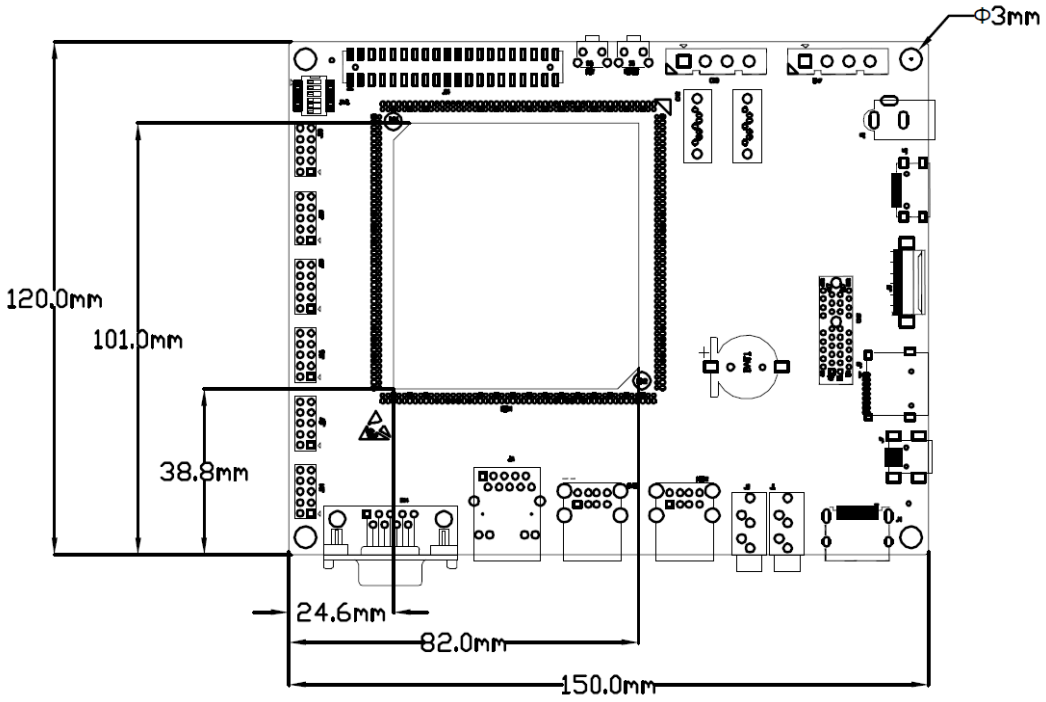


Figure 1-4 Expansion board dimensions

1.7 Associated Products

Table 1-1 Associated Products

Name	Description	Interface type	Linux	Android
WI-PI	WIFI module	USB2.0	Yes	Yes
CAM8100-U	2 megapixel USB Digital Camera	USB2.0	Yes	No
CAM8200-U	5 megapixel USB Digital Camera	USB 2.0	Yes	Yes
CAM8000-D	2 megapixel Digital Camera	30Pin FPC	Yes	Yes
CAM8100-D	5 megapixel Digital Camera	30Pin FPC	Yes	Yes
VGA8000	LCD to VGA module	50Pin FPC	Yes	Yes
LCD8000-97C	9.7-inch LVDS LCD, comes with Multi-Touch Capacitive Touch-Screen	Mini HDMI	Yes	Yes
LCD8000-43T	4.3-inch LCD, comes with a resistive touch screen	50Pin FPC	Yes	Yes
LCD8000-70T	7-inch LCD,comes with a resistive touch screen	50Pin FPC	Yes	Yes

Chapter 2 Introduction to Hardware

This chapter will give you a general understanding of the hardware system of SBC9000 by introducing CPU, peripheral ICs and pin definitions of the on-board interfaces.

2.1 CPU Introduction

i.MX 6Quad is an ARM™ Cortex-A9-based quad-core processor from Freescale. It runs at up to 1GHz, integrates 2D/3D graphics, 3D 1080p video processor and power management, and provides 64-bit DDR3/LVDDR3/LVDDR2-1066 interfaces as well as many other interfaces such as high-definition display and camera.

2.1.1 Clocks

The clock signals of i.MX 6Quad include a 32.768 KHz RTC clock and a 24 MHz external clock;

- **RTC Clock:** generated by an external crystal for low-frequency calculation;
- **External Clock:** used to generate main clock signal for PLL, CMM and other modules;

2.1.2 Reset Signal

Reset signal is determined by POR_B of CPU; low level validates resetting.

2.1.3 General Interfaces

General interfaces include 7 sets of GPIOs, each of which provides 32 dedicated GPIO pins (except GPIO7 which has 14 pins), and therefore the total pin number of GPIO can be up to 206.

2.1.4 Display Interface

- A parallel 24-bit RGB interface, supports 60Hz WUXGA output
- Two LVDS interfaces, support up to 165 Mpixels/sec output
- A HDMI 1.4 interface
- A MIPI/DSI interface with 1Gbps output rate

2.1.5 3D Graphics Acceleration System

i.MX 6Quad integrates GPU3Dv4 3D graphics processing unit which provides hardware acceleration for 3D graphics algorithms and allows desktop quality interactive graphics applications reach up to HD1080p resolution. The GPU3D supports OpenGL ES 2.0, including extensions, OpenGL ES 1.1, and OpenVG 1.1.

Additionally, i.MX 6Quad also has a GPUVGv2 vector graphics processing unit which provides hardware acceleration for 2D graphics algorithms.

2.2 Peripheral ICs around CPU

2.2.1 eMMC Flash MTFC4GMDEA-4M IT

MTFC4GMDEA-4M IT is an eMMC flash memory on SBC9000 with 4GB memory space. The flash supports high-speed DDR data transfer at a clock frequency of up to 52MHz, as well as three bit widths: 1-bit (default), 4-bit and 8-bit. The synchronous power management allows flash feature fast boot, automatically termination and sleep; meanwhile, MTFC4GMDEA-4M IT supports high-speed dual-data-transfer boot mode.

2.2.2 DDR MT41K128M16JT-125 IT

MT41K128M16JT-125 IT is a DDR3 SDRAM on Mini9000 with 256MB memory space. It is suited for high-capacity and high-bandwidth applications and supports differential clock input, differential data strobe, automatically refresh and asynchronous pin reset. Mini9000 integrates 4 memory chips in total, summing up to 1GB.

2.2.3 AR8035 Ethernet PHY

AR8035 is a single port 10/100/1000 Mbps tri-speed Ethernet PHY featured with low power and low cost. AR8035 supports MAC.TM RGMII interface and IEEE 802.3az-2010, Energy Efficient Ethernet (EEE) standard through proprietary SmartEEE technology, improving energy efficiency in systems using legacy MAC devices without 802.3az support. SBC9000 can be either connected to a hub with a straight-through network cable, or to a PC with a cross-over network cable.

2.2.4 USB2514 Hub

USB2514 is an USB 2.0 4-port Hub controller enables SBC9000 to implement 4 USB hub signals that are connected to USB ports. One of the signals is also connected to a mini PCIe interface to implement multiplexing.

2.2.5 JMB321 SATA Port Multiplier IC

JMB321 is a SATA port multiplier chip which allows SBC9000 provide two SATA ports.

2.3 Interfaces/LEDs/Switches on Mini9000

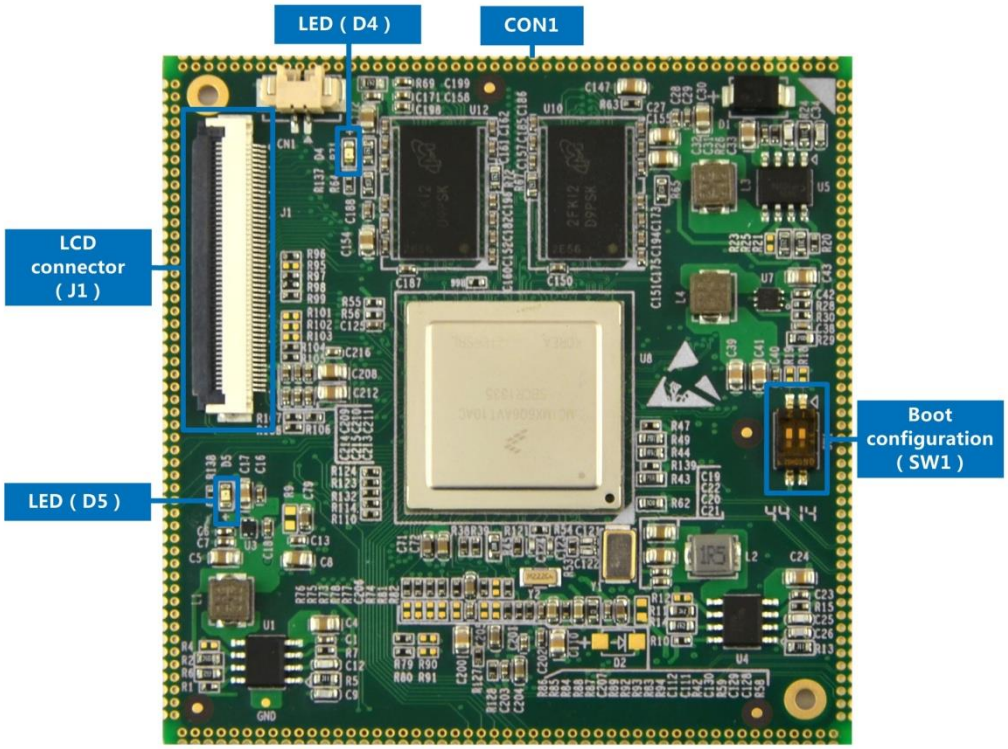


Figure 2-1 Interfaces/LEDs/Switches on Mini9000

2.3.1 CON1 Interface

Table 2-1 Pin definitions of CON1

Pins	Definitions	Descriptions
1	GPIO4_IO07	GPIO signal
2	GND	GND
3	OTG_VBUS	OTG bus, +5V
4	5VIN	Mini9000 power supply, +5V
5	5VIN	Mini9000 power supply, +5V
6	5VIN	Mini9000 power supply, +5V
7	GND	GND
8	VDD_RTC	Real-time clock power
9	2P5V	System power, +2.5V
10	2P5V	System power, +2.5V
11	GND	GND
12	3P3V	System power, +3.3V
13	3P3V	System power, +3.3V
14	3P3V	System power, +3.3V
15	GND	GND

Pins	Definitions	Descriptions
16	ON_OFF	System power on/off control signal
17	RESET_N_B	System reset control signal
18	GND	GND
19	CAN2_RXD	CAN2 receive data
20	CAN2_TXD	CAN2 transmit data
21	I2C2_SDA	I2C2 master serial data
22	I2C2_SCL	I2C2 master serial clock
23	GPIO4_IO11	GPIO signal
24	GPIO4_IO06	GPIO signal
25	EIM_WAIT	EIM ready/busy/wait signal
26	EIM_EB1	EIM byte enable
27	EIM_LBA	EIM address valid
28	EIM_CS0	EIM chip select
29	EIM_OE	EIM output enable
30	EIM_CS1	EIM chip select
31	EIM_EB0	EIM byte enable
32	EIM_RW	EIM memory write enable
33	EIM_D29	EIM MSB data bus signal
34	EIM_D28	EIM MSB data bus signal
35	EIM_CRE	Used as CRE/PS for Cellular Rammemory.
36	EIM_EB3	EIM byte enable
37	EIM_EB2	EIM byte enable
38	EIM_BCLK	EIM burst clock
39	EIM_A26	EIM MSB address bus signal
40	EIM_A23	EIM MSB address bus signal
41	EIM_A22	EIM MSB address bus signal
42	EIM_A18	EIM MSB address bus signal
43	EIM_A24	EIM MSB address bus signal
44	EIM_A21	EIM MSB address bus signal
45	EIM_A25	EIM MSB address bus signal
46	EIM_A16	EIM MSB address bus signal
47	EIM_A20	EIM MSB address bus signal
48	EIM_A19	EIM MSB address bus signal
49	EIM_A17	EIM MSB address bus signal
50	GND	GND
51	GPIO1_IO27	GPIO signal
52	EIM_DA11	EIM LSB multiplexed address/data bus signal
53	EIM_DA13	EIM LSB multiplexed address/data bus signal
54	EIM_DA14	EIM LSB multiplexed address/data bus signal
55	EIM_DA9	EIM LSB multiplexed address/data bus signal
56	EIM_DA12	EIM LSB multiplexed address/data bus signal
57	EIM_DA10	EIM LSB multiplexed address/data bus signal

Pins	Definitions	Descriptions
58	EIM_DA0	EIM LSB multiplexed address/data bus signal
59	EIM_DA8	EIM LSB multiplexed address/data bus signal
60	EIM_DA15	EIM LSB multiplexed address/data bus signal
61	EIM_DA7	EIM LSB multiplexed address/data bus signal
62	EIM_DA4	EIM LSB multiplexed address/data bus signal
63	EIM_DA3	EIM LSB multiplexed address/data bus signal
64	EIM_DA5	EIM LSB multiplexed address/data bus signal
65	EIM_DA2	EIM LSB multiplexed address/data bus signal
66	EIM_DA6	EIM LSB multiplexed address/data bus signal
67	EIM_DA1	EIM LSB multiplexed address/data bus signal
68	CSPI1_SS1	SPI1 chip select
69	CSPI1_CLK	SPI1 clock
70	CSPI1_MOSI	SPI1 master output salve input
71	CSPI1_MISO	SPI1 master input salve output
72	UART3_RXD	UART3 receive data
73	UART3_TXD	UART3 transmit data
74	UART2_RXD	UART2 receive data
75	UART2_TXD	UART2 transmit data
76	GND	GND
77	UART3_CTS	UART3 clear to send
78	UART3_RTS	UART3 request to send
79	UART2_RTS	UART2 request to send
80	UART2_CTS	UART2 clear to send
81	USB_H1_OC	Host 1 external input for VBUS
82	USB_OTG_OC	overcurrent detection
83	USB_HOST_DP	OTG External input for VBUS
84	USB_HOST_DN	overcurrent detection
85	USB_RSTn	USB host data+
86	SD1_WP	USB host data-
87	SD1_DATA3	USB host reset control signal
88	SD1_CLK	SD1 card write protect detect signal
89	SD1_DATA2	SD1 data 3
90	SD1_DATA1	SD1 clock
91	SD1_DATA6	SD1 data 2
92	SD1_DATA0	SD1 data 1
93	SD1_DATA7	SD1 data 6
94	GND	SD1 data 0
95	SD1_DATA5	SD1 data 7
96	SD1_DATA4	GND
97	SD1_CMD	SD1 data 5
98	SD1_CD	SD1 data 4
99	SD2_WP	SD1 command signal

Pins	Definitions	Descriptions
100	SD2_DATA2	SD1 card detect signal
101	SD2_DATA3	SD2 card write protect detect signal
102	SD2_DATA0	SD2 data 2
103	SD2_DATA5	SD2 data 3
104	GND	SD2 data 0
105	SD2_CD	SD2 data 5
106	SD2_DATA4	GND
107	SD2_CLK	SD2 card detect signal
108	SD2_DATA1	SD2 data 4
109	SD2_CMD	SD2 clock
110	SD2_DATA6	SD2 data 1
111	SD2_DATA7	SD2 command signal
112	GND	SD2 data 6
113	RGMII_REF_CLK	SD2 data 7
114	RGMII_MDIO	GND
115	RGMII_INT	RGMII reference clock
116	RGMII_MDC	RGMII information transferring control
117	RGMII_RXD3	RGMII interrupting signal
118	RGMII_TXD2	RGMII output clock
119	RGMII_RXDV	RGMII receive data
120	GND	RGMII transmit data
121	RGMII_TXD0	RGMII receive data valid
122	RGMII_TXCLK	GND
123	RGMII_nRST	RGMII transmit data
124	RGMII_TXD1	RGMII transmit clock
125	RGMII_RXD0	RGMII reset signal
126	RGMII_RXCLK	RGMII transmit data
127	RGMII_RXD2	RGMII receive data
128	RGMII_TXEN	RGMII receive clock
129	RGMII_TXD3	RGMII receive data
130	RGMII_RXD1	RGMII transmit enable
131	GPIO7_IO06	RGMII transmit data
132	GPIO7_IO04	RGMII receive data
133	GPIO7_IO05	GPIO signal
134	GPIO7_IO01	GPIO signal
135	UART1_RXD	GPIO signal
136	UART1_TXD	GPIO signal
137	USB_OTG_DN	UART1 receive data
138	USB_OTG_DP	UART1 transmit data
139	USB_OTG_ID	OTG data-
140	USB_OTG_PWR_EN	OTG data+
141	GND	OTG ID signal

Pins	Definitions	Descriptions
142	CSI0_DAT16	OTG power enable control signal
143	CSI0_DAT17	GND
144	CSI0_DAT15	CSI0 capture data bit 16
145	CSI0_DAT13	CSI0 capture data bit 17
146	CSI0_DAT14	CSI0 capture data bit 15
147	CSI0_DAT12	CSI0 capture data bit 13
148	CSI0_DAT19	CSI0 capture data bit 14
149	CSI0_DAT18	CSI0 capture data bit 12
150	GND	CSI0 capture data bit 19
151	CSI0_HSYNC	CSI0 capture data bit 18
152	CSI0_PIXCLK	GND
153	CSI0_VSYNC	CSI0 horizontal synchronization
154	CAM_MCLK	CSI0 pixel clock
155	CAM_RST	CSI0 vertical synchronization
156	CAM_EN	Camera clock
157	SATA_RXN	Camera reset control signal
158	SATA_RXP	Camera data enable control signal
159	SATA_TXP	SATA receive data-
160	SATA_TXN	SATA receive data+
161	PCIE_WAKEn	SATA transmit data+
162	PCIE_REFCLK_DP	SATA transmit data-
163	PCIE_REFCLK_DN	PCIe wake enable control signal
164	GND	PCIe reference clock+
165	PCIE_TXP	PCIe reference clock-
166	PCIE_TXM	GND
167	PCIE_RXP	PCIe transmit data+
168	PCIE_RXM	PCIe transmit data-
169	PRSNT2_N_X1	PCIe receive data+
170	I2C1_SDA	PCIe receive data-
171	I2C1_SCL	Add-in card presence detect signal
172	AUD3_RXD	I2C1 master serial clock
173	AUD3_TXFS	I2C1 master serial data
174	AUD3_TXD	Audio data receive signal
175	AUD3_TXC	Audio receive frame sync signal
176	GND	Audio data transmit signal
177	HDMI_HPD	Audio transmit clock signal
178	HDMI_CLKM	GND
179	HDMI_CLKP	HDMI hot plug and play detect
180	GND	HDMI data clock-
181	HDMI_D0M	HDMI data clock+
182	HDMI_D0P	GND
183	HDMI_D1M	HDMI data 0-

Pins	Definitions	Descriptions
184	HDMI_D1P	HDMI data 0+
185	HDMI_D2M	HDMI data 1-
186	HDMI_D2P	HDMI data 1+
187	I2C3_SDA	HDMI data 2-
188	I2C3_SCL	HDMI data 2+
189	UART5_TXD	I2C3 master serial data
190	UART5_RXD	I2C3 master serial clock
191	CAN1_RXD	UART5 transmit data
192	CAN1_TXD	UART5 receive data
193	PWM4	CAN1 receive data
194	LED_PWR_EN	CAN1 transmit data
195	Touch_Int	Pulse width modulation
196	LCD_PWR_EN	LED backlight enable
197	LVDS0_TX1_N	Touch interrupt signal
198	LVDS0_TX1_P	Touch reset signal
199	LVDS0_CLK_N	LVDS0 data1-
200	LVDS0_CLK_P	LVDS0 data1+
201	LVDS0_TX0_N	LVDS0 clock-
202	LVDS0_TX0_P	LVDS0 clock+
203	LVDS0_TX2_N	LVDS0 data0-
204	LVDS0_TX2_P	LVDS0 data0+

2.3.2 Boot Configuration Switch (SW1)

SBC9000 has two sets of boot configuration switches mounted on the Mini9000 CPU module and the expansion board respectively for selecting a boot mode.

Table 2-2 Boot configuration switch on Mini9000

Pins	1	2	Descriptions
Status	OFF	ON	Serial Downloader
	ON	OFF	Internal boot
	OFF	OFF	Boot from Fuses
	ON	ON	Reserved

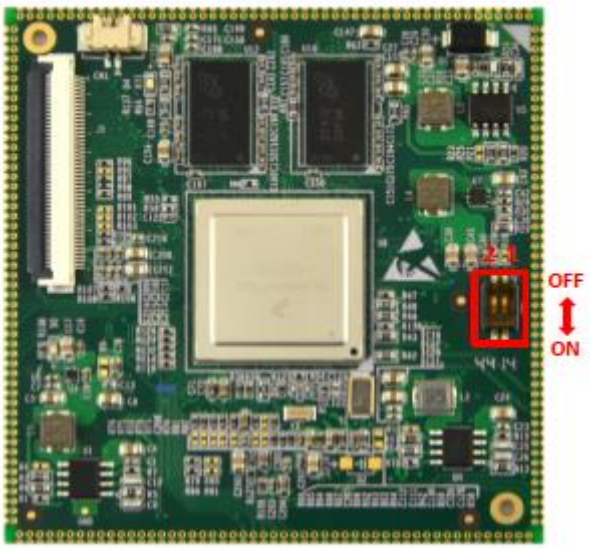


Figure 2-2 The boot configuration switch status description

2.3.3 LED Indicators

Table 2-3 LED Indicators

Pins	Descriptions
D5	Power indicator
D4	User-defined LED

2.3.4 LCD Interface (J1)

Table 2-4 LCD Interface

Pins	Definitions	Descriptions
1	B0	LCD Pixel data bit 0
2	B1	LCD Pixel data bit 1
3	B2	LCD Pixel data bit 2
4	B3	LCD Pixel data bit 3
5	B4	LCD Pixel data bit 4
6	B5	LCD Pixel data bit 5
7	B6	LCD Pixel data bit 6
8	B7	LCD Pixel data bit 7
9	GND1	GND
10	G0	LCD Pixel data bit 8
11	G1	LCD Pixel data bit 9
12	G2	LCD Pixel data bit 10
13	G3	LCD Pixel data bit 11
14	G4	LCD Pixel data bit 12

Pins	Definitions	Descriptions
15	G5	LCD Pixel data bit 13
16	G6	LCD Pixel data bit 14
17	G7	LCD Pixel data bit 15
18	GND2	GND
19	R0	LCD Pixel data bit 16
20	R1	LCD Pixel data bit 17
21	R2	LCD Pixel data bit 18
22	R3	LCD Pixel data bit 19
23	R4	LCD Pixel data bit 20
24	R5	LCD Pixel data bit 21
25	R6	LCD Pixel data bit 22
26	R7	LCD Pixel data bit 23
27	GND3	GND
28	DEN	AC bias control (STN) or pixel data enable (TFT)
29	HSYNC	LCD Horizontal Synchronization
30	VSYNC	LCD Vertical Synchronization
31	GND	GND
32	CLK	LCD Pixel Clock
33	GND4	GND
34	X+	X+ Position Input
35	X-	X- Position Input
36	Y+	Y+ Position Input
37	Y-	Y- Position Input
38	SPI_CLK	SPI serial clock
39	SPI_MOSI	SPI Master Output, Slave Input
40	SPI_MISO	SPI Master Input, Slave Output
41	SPI_CS	SPI Chip Select
42	IIC_CLK	IIC master serial clock
43	IIC_DAT	IIC serial bidirectional data
44	GND5	GND
45	VDD1	3.3V
46	VDD2	3.3V
47	VDD3	5V
48	VDD4	5V
49	RESET	Reset
50	PWREN	Backlight enable

2.4 Interfaces/LEDs/Switches on Expansion Board

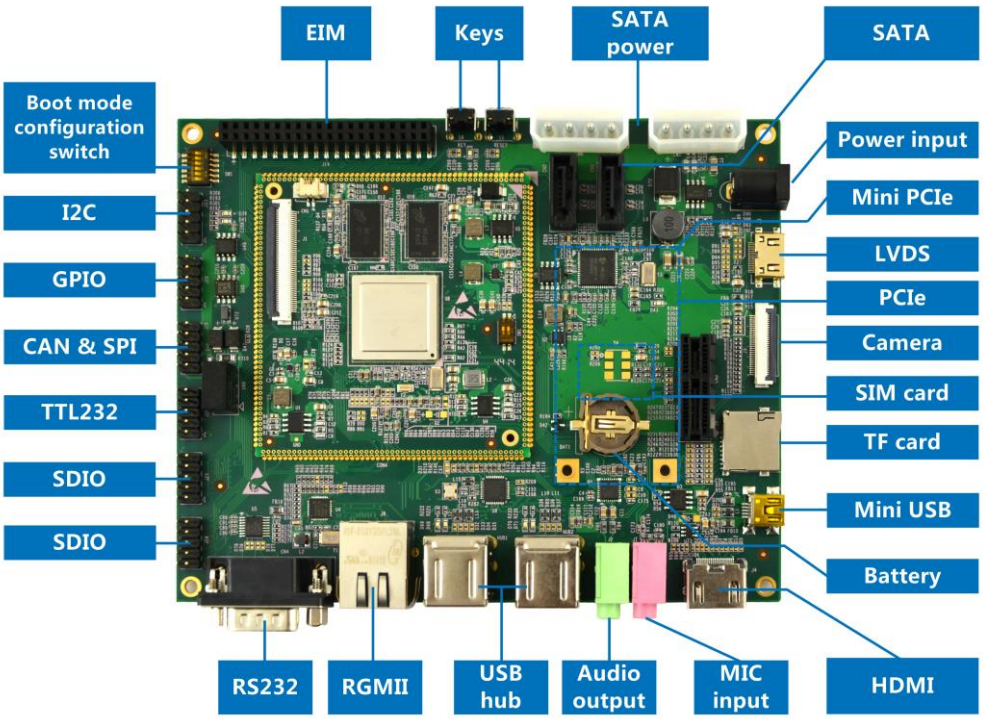


Figure 2-3 Interfaces/LEDs/switches on Expansion board

2.4.1 Power Jack (J5)

Table 2-5 Power jack

Pins	Definitions	Descriptions
1	GND	GND
2	+12V	Power supply (+12V)
3	NC	NC

2.4.2 Audio Input (J1)

Table 2-6 Audio Input

Pins	Definitions	Descriptions
1	GND	GND
2	MIC In	input
3	MIC In	input
4	MIC In	input
5	MIC In	input

2.4.3 Audio Output (J2)

Table 2-7 Audio Output

Pins	Definitions	Descriptions
1	GND	GND
2	Left	Left output
3	Right	Right output
4	Right	Right output
5	Left	Left output

2.4.4 Camera Interface (J3)

Table 2-8 Camera interface

Pins	Definitions	Descriptions
1	GND	GND
2	NC	NC
3	NC	NC
4	CSI0_DAT12	CSI0 capture data bit 12
5	CSI0_DAT13	CSI0 capture data bit 13
6	CSI0_DAT14	CSI0 capture data bit 14
7	CSI0_DAT15	CSI0 capture data bit 15
8	CSI0_DAT16	CSI0 capture data bit 16
9	CSI0_DAT17	CSI0 capture data bit 17
10	CSI0_DAT18	CSI0 capture data bit 18
11	CSI0_DAT19	CSI0 capture data bit 19
12	NC	NC
13	NC	NC
14	GND	GND
15	CSI0_PIXCLK	CSI0 pixel clock
16	GND	GND
17	CSI0_HSYNC	CSIO HSYNC
18	NC	NC
19	CSI0_VSYNC	CSIO VSYNC
20	VDD_NVCC	3.3V
21	CAM_MCLK	Camera clock
22	NC	NC
23	GND	GND
24	NC	NC
25	CAM_RST	CSI0 reset

Pins	Definitions	Descriptions
26	CAM_EN	CSI0 data enable
27	I2C1_SDA	I2C2 serial data
28	I2C1_SCL	I2C2 serial clock
29	GND	GND
30	3P3V	3.3V

2.4.5 EIM Interface (J14)

Table 2-9 EIM interface

Pins	Definitions	Descriptions
1	5VIN	5V
2	3P3V	3.3V
3	GND	GND
4	GND	GND
5	EIM_DA0	EIM LSB multiplexed address/data bus signal
6	EIM_DA1	EIM LSB multiplexed address/data bus signal
7	EIM_DA2	EIM LSB multiplexed address/data bus signal
8	EIM_DA3	EIM LSB multiplexed address/data bus signal
9	EIM_DA4	EIM LSB multiplexed address/data bus signal
10	EIM_DA5	EIM LSB multiplexed address/data bus signal
11	EIM_DA6	EIM LSB multiplexed address/data bus signal
12	EIM_DA7	EIM LSB multiplexed address/data bus signal
13	EIM_DA8	EIM LSB multiplexed address/data bus signal
14	EIM_DA9	EIM LSB multiplexed address/data bus signal
15	EIM_DA10	EIM LSB multiplexed address/data bus signal
16	EIM_DA11	EIM LSB multiplexed address/data bus signal
17	EIM_DA12	EIM LSB multiplexed address/data bus signal
18	EIM_DA13	EIM LSB multiplexed address/data bus signal
19	EIM_DA14	EIM LSB multiplexed address/data bus signal
20	EIM_DA15	EIM LSB multiplexed address/data bus signal
21	EIM_A16	EIM MSB address bus signal
22	EIM_A17	EIM MSB address bus signal
23	EIM_A18	EIM MSB address bus signal
24	EIM_A19	EIM MSB address bus signal
25	EIM_A20	EIM MSB address bus signal
26	EIM_A21	EIM MSB address bus signal
27	EIM_A22	EIM MSB address bus signal
28	EIM_A23	EIM MSB address bus signal
29	EIM_A24	EIM MSB address bus signal
30	EIM_A25	EIM MSB address bus signal

Pins	Definitions	Descriptions
31	EIM_WAIT	EIM ready/busy/wait signal
32	EIM_LBA	EIM address valid
33	EIM_EB0	EIM byte enable
34	EIM_EB1	EIM byte enable
35	EIM_RW	EIM memory write enable
36	EIM_CRE	Used as CRE/PS for Cellular Rammemory
37	EIM_BCLK	EIM burst clock
38	EIM_CS0	EIM chip select
39	EIM_CS1	EIM chip select
40	EIM_OE	EIM output enable

2.4.6 GPIO Interface (J29)

Table 2-10 GPIO interface

Pins	Definitions	Descriptions
1	5VIN	5V
2	3P3V	3.3V
3	GND	GND
4	GND	GND
5	EIM_A26	EIM MSB address bus signal
6	EIM_EB2	EIM byte enable
7	EIM_D28	EIM MSB data bus signal
8	EIM_EB3	EIM byte enable
9	EIM_D29	EIM MSB data bus signal
10	ON_OFF	System power on/off control signal

2.4.7 I2C Interface (J23)

Table 2-11 I2C interface

Pins	Definitions	Descriptions
1	3P3V	+5V
2	GND	GND
3	I2C3_SCL	I2C3 master serial clock
4	GPIO7_IO05	GPIO signal
5	I2C3_SDA	I2C3 master serial data
6	GPIO7_IO	GPIO signal
7	I2C1_SCL	I2C1 master serial clock
8	I2C2_SCL	I2C2 master serial clock
9	I2C1_SDA	I2C1 master serial data

Pins	Definitions	Descriptions
10	I2C2_SDA	I2C2 master serial data

2.4.8 CAN & SPI Interface (J25)

Table 2-12 CAN & SPI interface

Pins	Definitions	Descriptions
1	3P3V	3.3V
2	GND	GND
3	CAN2_L	CAN2_L
4	CSPI1_SS1	SPI1 chip select
5	CAN2_H	CAN2_H
6	CSPI1_CLK	SPI1 clock
7	CAN1_L	CAN1_L
8	CSPI1_MISO	SPI1 master input salve output
9	CAN1_H	CAN1_H
10	CSPI1_MOSI	SPI1 master output salve input

2.4.9 HDMI Interface (J4)

Table 2-13 HDMI interface

Pins	Definitions	Descriptions
1	HDMI_D2P	TMDS data 2+
2	GND	GND
3	HDMI_D2M	TMDS data 2 shield
4	HDMI_D1P	TMDS data 1+
5	GND	GND
6	HDMI_D1M	TMDS data 1-
7	HDMI_D0P	TMDS data 0+
8	GND	TMDS data 0 shield
9	HDMI_D0M	TMDS data 0-
10	HDMI_CLKP	TMDS data clock+
11	GND	GND
12	HDMI_CLKM	TMDS data clock-
13	NC	NC
14	NC	NC
15	BI2C2_SCL	IIC master serial clock
16	BI2C2_SDA	IIC serial bidirectional data
17	GND	GND
18	5V	5V

Pins	Definitions	Descriptions
19	HDMI_HPD_R	Hot plug and play detect

2.4.10 LVDS Interface (J6)

Table 2-14 LVDS interface

Pins	Definitions	Descriptions
1	LVDS_3P3V	+3.3V
2	LVDS0_TX2_P	LVDS0 Data2+
3	LVDS0_TX2_NN	LVDS0 Data2-
4	GND	GND
5	LVDS0_TX1_P	LVDS0 Data1+
6	LVDS0_TX1_N	LVDS0 Data1-
7	GND	GND
8	LVDS0_TX0_P	LVDS0 Data0+
9	LVDS0_TX0_N	LVDS0 Data-
10	GND	GND
11	LVDS0_CLK_PP	LVDS0_CLK+
12	LVDS0_CLK_N	LVDS0_CLK-
13	LCD_PWR_ENN	Touch Reset Signal
14	Touch_Int	Touch Interrupt Signal
15	I2C3_SCL	I2C3 Master Serial Clock
16	I2C3_SDA	I2C3 Master Serial Data
17	LED_PWR_EN	Backlight Enable
18	5VIN	+5V
19	PWM4	Pulse Width Modulation

2.4.11 Mini PCIe Interface (CN5)

Table 2-15 Mini PCIe interface

Pins	Definitions	Descriptions
1	GPIO1_IO27	GPIO Signal
2	MPCIE_3P3V	3.3V
3	NC	Not connected
4	GND	GND
5	NC	Not connected
6	NC	Not connected
7	NC	Not connected
8	UIM_PWR	UIM power supply
9	GND	GND

Pins	Definitions	Descriptions
10	UIM_DATA	UIM data
11	NC	Not connected
12	UIM_CLK	UIM clock
13	NC	Not connected
14	UIM_RESET	UIM reset control signal
15	GND	GND
16	NC	Not connected
17	NC	Not connected
18	GND	GND
19	NC	Not connected
20	W_DISABLE	Close wireless communications
21	GND	GND
22	PERST	Reset signal
23	NC	Not connected
24	MPCIE_3P3V	3.3V
25	NC	Not connected
26	GND	GND
27	GND	GND
28	NC	Not connected
29	GND	GND
30	NC	Not connected
31	NC	Not connected
32	NC	Not connected
33	NC	Not connected
34	GND	GND
35	GND	GND
36	DM1	USB data-
37	GND	GND
38	DP1	USB data+
39	MPCIE_3P3V	3.3V
40	GND	GND
41	MPCIE_3P3V	3.3V
42	LED_WWAN	Status indicated signal
43	GND	GND
44	NC	Not connected
45	NC	Not connected
46	NC	Not connected
47	NC	Not connected
48	NC	Not connected
49	NC	Not connected
50	GND	GND
51	NC	Not connected

Pins	Definitions	Descriptions
52	MPCIE_3P3V	3.3V

2.4.12 OTG Interface (J7)

Table 2-16 OTG interface

Pins	Definitions	Descriptions
1	OTG_VBUS	OTG bus, +5V
2	USB_OTG_DN	OTG data-
3	USB_OTG_DP	OTG data+
4	USB_OTG_ID	OTG ID signal
5	GND	GND

2.4.13 PCIe Interface (CN3)

Table 2-17 PCIe interface

Pins	Definitions	Descriptions
A1	GND	GND
A2	12VIN	12V
A3	12VIN	12V
A4	GND	GND
A5	NC	Not connected
A6	NC	Not connected
A7	NC	Not connected
A8	NC	Not connected
A9	3P3V	3.3V
A10	3P3V	3.3V
A11	RESET_N_B	System reset control signal
A12	GND	GND
A13	REFCLK+	PCIe reference clock+
A14	REFCLK+	PCIe reference clock-
A15	GND	GND
A16	PCIE_RXP	PCIe receive data+
A17	PCIE_RXN	PCIe receive data-
A18	GND	GND
B1	12VIN	12V
B2	12VIN	12V
B3	12VIN	12V
B4	GND	GND
B5	I2C1_SCL	I2C1 master serial clock

Pins	Definitions	Descriptions
B6	I2C1_SDA	I2C1 master serial data
B7	GND	GND
B8	3P3V	3.3V
B9	NC	Not connected
B10	3P3V	3.3V
B11	PCIE_WAKEn	PCIe wake enable control signal
B12	NC	Not connected
B13	GND	GND
B14	PCIE_TXP	PCIe transmit data+
B15	PCIE_TXN	PCIe transmit data-
B16	GND	GND
B17	PRSNT2_N_X1	Add-in card presence detect signal
B18	GND	GND

2.4.14 RGMII Interface (J8)

Table 2-18 RGMII interface

Pins	Definitions	Descriptions
1	TD1+	TD1+ output
2	TD1-	TD1- output
3	TD2+	TD2+ output
4	TD2-	TD2- output
5	TCT	2.5V Power for TD
6	RCT	2.5V Power for RD
7	RD1+	RD1+ input
8	RD1-	RD1- input
9	RD2+	RD2+ input
10	RD2-	RD2- input
11	GRLA	Green LED link signal
12	GRLC	Power supply for Green LED
13	YELC	Yellow LED action signal
14	YELA	Power supply for Yellow LED

2.4.15 SATA Interface (CN1/CN7/CN2/CN8)

Table 2-19 SATA data interface1 (CN1)

Pins	Definitions	Descriptions
1	GND	GND
2	SATA_TXP0	SATA transmit data 0+

Pins	Definitions	Descriptions
3	SATA_TXN0	SATA transmit data 0-
4	GND	GND
5	SATA_RXN0	SATA receive data 0-
6	SATA_RXP0	SATA receive data 0+
7	GND	GND

Table 2-20 SATA power interface1 (CN7)

Pins	Definitions	Descriptions
1	5VIN	5V
2	GND	GND
3	GND	GND
4	12VIN	12V

Table 2-21 SATA data interface2 (CN2)

Pins	Definitions	Descriptions
1	GND	GND
2	SATA_TXP1	SATA transmit data 1+
3	SATA_TXN1	SATA transmit data 1-
4	GND	GND
5	SATA_RXN1	SATA receive data 1-
6	SATA_RXP1	SATA receive data 1+
7	GND	GND

Table 2-22 SATA power interface2 (CN8)

Pins	Definitions	Descriptions
1	5VIN	5V
2	GND	GND
3	GND	GND
4	12VIN	12V

2.4.16 UART Interface (CN4/J28)

Table 2-23 DB9 interface (CN4)

Pins	Definitions	Descriptions
1	NC	Not connected
2	COM2_RXD	DB9 receive data
3	COM2_TXD	DB9 transmit data
4	NC	Not connected

Pins	Definitions	Descriptions
5	GND	GND
6	NC	Not connected
7	COM2_RTS	DB9 request to send
8	COM2_CTS	DB9 clear to send
9	NC	Not connected

Table 2-24 TTL232 interface (J28)

Pins	Definitions	Descriptions
1	3P3V	3.3V
2	GND	GND
3	UART1_RXD	UART1 receive data
4	UART3_RTS	UART3 request to send
5	UART1_TXD	UART1 transmit data
6	UART3_CTS	UART3 clear to send
7	UART5_TXD	UART5 transmit data
8	UART3_TXD	UART3 transmit data
9	UART5_RXD	UART5 receive data
10	UART3_RXD	UART3 receive data

2.4.17 USB HUB Interface (HUB1/HUB2)

Table 2-25 USB HUB interface1 (HUB1)

Pins	Definitions	Descriptions
1	USB_PWR1	USB power supply1
2	DM1	USB data 1-
3	DP1	USB data 1+
4	GND	GND
5	USB_PWR2	USB power supply2
6	DM2	USB data 2-
7	DP2	USB data 2+
8	GND	GND

Table 2-26 USB HUB interface2 (HUB2)

Pins	Definitions	Descriptions
1	USB_PWR3	USB power supply 3
2	DM3	USB data 3-
3	DP3	USB data 3+
4	GND	GND
5	USB_PWR4	USB power supply 4

Pins	Definitions	Descriptions
6	DM4	USB data 4-
7	DP4	USB data 4+
8	GND	GND

2.4.18 SDIO Interface (J9/J27/J24)

Table 2-27 TF card slot (J9)

Pins	Definitions	Descriptions
1	SD2_DATA2	SD2 data 2
2	SD2_DATA3	SD2 data 3
3	SD2_CMD	SD2 command signal
4	3P3V	3.3V
5	SD2_CLK	SD2 clock
6	GND	GND
7	SD2_DATA0	SD2 data 0
8	SD2_DATA1	SD2 data 1
9	SD2_CD	SD2 card detect signal
10	GND	GND

Table 2-28 SDIO interface1 (J27)

Pins	Definitions	Descriptions
1	SD1_CD	SD1 card detect signal
2	SD1_WP	SD1 card write protect detect signal
3	SD1_CMD	SD1 command signal
4	SD1_CLK	SD1 clock
5	3P3V	3.3V
6	GND	GND
7	SD1_DATA2	SD1 data 2
8	SD1_DATA3	SD1 data 3
9	SD1_DATA0	SD1 data 0
10	SD1_DATA1	SD1 data 1

Table 2-29 SDIO interface2 (J24)

Pins	Definitions	Descriptions
1	3P3V	3.3V
2	GND	GND
3	SD2_DATA7	SD2 data 7
4	SD1_DATA7	SD1 data 7
5	SD2_DATA6	SD2 data 6

Pins	Definitions	Descriptions
6	SD1_DATA6	SD1 data6
7	SD2_DATA5	SD2 data 5
8	SD1_DATA5	SD1 data 5
9	SD2_DATA4	SD2 data 4
10	SD1_DATA4	SD1 data 4

2.4.19 Boot Configuration Switch (SW1)

Table 2-30 Boot Configuration Switch

Pins	Definitions	Descriptions
1	BT_CFG1_6	BT_CFG1_6
2	BT_CFG1_5	BT_CFG1_5
3	BT_CFG2_6	BT_CFG2_6
4	BT_CFG2_5	BT_CFG2_5
5	BT_CFG2_4	BT_CFG2_4
6	BT_CFG2_3	BT_CFG2_3

Table 2-31 Boot Mode Pin Settings

Pins	1	2	3	4	5	6	Boot Mode
Status	ON	ON	ON	OFF	ON	ON	8-bit eMMC
	ON	OFF	X	ON	OFF	ON	4-bit TF Card
	ON	OFF	X	ON	OFF	OFF	4-bit SDIO1

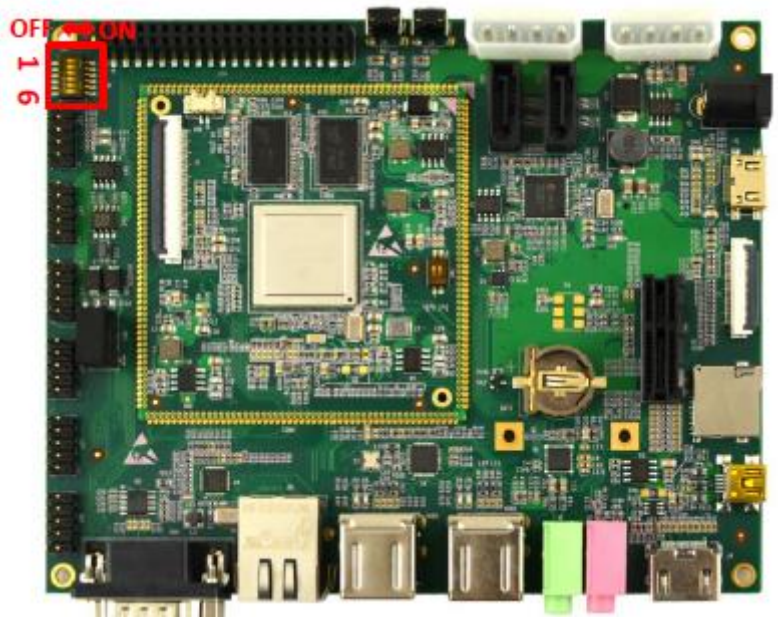


Figure 2-4 The boot configuration switch status description

2.4.20 Buttons

Table 2-32 Buttons

Pins	Descriptions
S5	Reset button
S8	User defined button

2.4.21 LEDs

Table 2-33 LEDs

Pins	Descriptions
D40	Power LED
D39	User defined LED
D41	User defined LED

Chapter 3 Preparations

Before you start to use SBC9000, please read the following sections to get yourself familiar with the system images, driver code and tools which might be involved during development process.

3.1 Software Introduction

The table shown below lists the versions of Linux and Android systems, as well as the device drivers that will be used later.

Table 3-1 OS and drivers

Categories		Notes
OS	Linux	Version 3.10.17
	Android	Version 4.4.2
Device Drivers	Serial	Serial interface driver
	RTC	Hardware clock driver
	Net	10/100/Gb IEEE1588 Ethernet
	Display	Three display ports (RGB, LVDS, and HDMI 1.4a)
	mmc/sd	One SD 3.0/SDXC card slot & eMMC
	USB	5 High speed USB ports (4xHost, 1xOTG)
	Audio	Analog audio (Audio out & Mic In) digital audio (HDMI)
	Camera	Camera interface (1xParallel)
	LED	User-defined LED driver

3.2 About Linux System

The following tables list the specific images and eMMC storage partitions required to build a Linux system.

Table 3-2 Linux images

Images	Paths
U-boot image	u-boot.imx
Kernel image	uImage, imx6q-sbc9000.dtb

Images	Paths
Filesystem	fsl-image-fb-sbc9000.tar.bz2

Table 3-3 eMMC partitions

Partition types/indexes	Names	Start Offsets	Sizes	Filesystems	Contents
N/A	BOOT Loader	0	4MB	N/A	u-boot.imx
N/A	Kernel	4M	20MB	FAT	uImage, imx6q-sbc9000.dtb
Primary 1	Rootfs	20M	Total - Other	EXT3	fsl-image-fb-sbc9000.tar.bz2

- **Partition types/indexes:** Definitions are saved in MBR.
- **Names:** only useful under Android. You can ignore it when creating Linux partitions.
- **Start Offsets:** shows where partition starts with unit in MB.

3.3 About Android System

The following tables list the specific images and eMMC storage partitions required to build an Android system.

Table 3-4 Android image

Images	Paths
u-boot image	u-boot.bin
boot image	boot.img
Android system root image	system.img
Recovery root image	recovery.img

Table 3-5 Android Partitions

Partition types/indexes	Names	Start Offsets	Sizes	Filesystems	Contents
N/A	BOOT Loader	0	1MB	N/A	bootloader
Primary 1	Boot	8M	8MB	boot.img format, a kernel +	boot.img

Partition types/indexes	Names	Start Offsets	Sizes	Filesystems	Contents
				ramdisk	
Primary 2	Recovery	After Boot partition	8MB	boot.img format, a kernel + ramdisk	recovery.img
Logic 4 (Extended 3)	DATA	After Recovery partition	>1024MB	EXT4. Mount as /data	System application data
Logic 5 (Extended 3)	SYSTEM	After Recovery partition	512MB	EXT4. Mount as /system	Android system files under /system/
Logic 6 (Extended 3)	CACHE	After SYSTEM partition	512MB	EXT4. Mount as /cache	Android cache, for storage of OTA image
Logic 7 (Extended 3)	device	After CACHE partition	8MB	EXT4 Mount at /device	Storage of MAC addresses
Logic 8 (Extended 3)	Misc	After Vendor partition	8M	N/A	For restoring and saving bootloader
Primary 4	MEDIA	After Misc partition	Total - Other images	VFAT	Internal media partition under /mnt/sdcard

- **SYSTEM:** the system partition is used to store Android files.
- **DATA:** the data partition is used to store unpacked data of applications and system configuration database.

Normally the root filesystem is mounted from ramdisk. However it could be mounted from the RECOVERY partition if the system is recovery mode.

3.4 Setting up HyperTerminal

Use a cross-over serial cable to connect the debug interface “CN4” of SBC9000 to your PC’s serial port, then select **Start** > **Programs** > **Accessories** > **Communications** > **HyperTerminal** on desktop to set up a new HyperTerminal according to the parameters as show below.

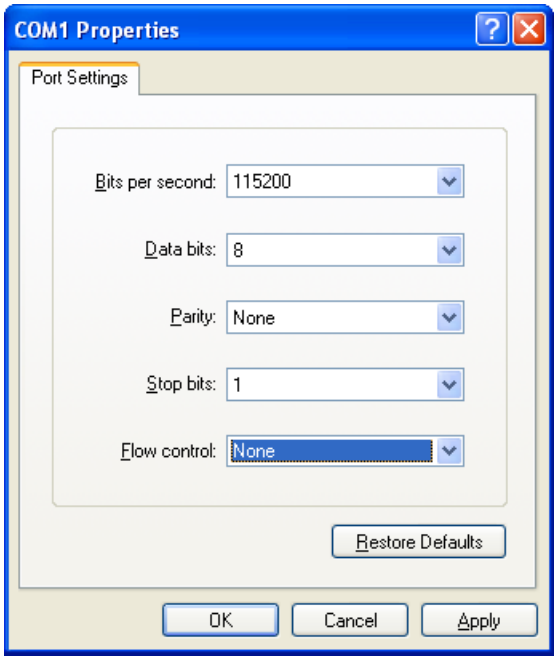


Figure 3-1 Setting up HyperTerminal

Chapter 4 Downloading and Running of System

Now you can download the available system images to SBC9000 and run the system.

Note:

- Each instruction has been put a bullet "•" before it to prevent confusion caused by the long instructions that occupy more than one line in the context.
- Please note that there are SPACES in the following instructions; Missing any SPACE will lead to failure when executing instructions.

4.1 Downloading/Running of Linux/Android

4.1.1 Using Mfgtools to Download Linux/Android

1） Download Linux and Android images for SBC9000 as well as programmer from Embest's website to the root directory (assuming C:\) of your PC;

2） Use a Mini USB cable to connect the USB OTG port (J7) of SBC9000 to an USB HOST port on your PC;

3） Set the DIP switch "SW1" on the Mini9000 to MfgTools mode according to the following table;

Table 4-1 Boot config switch settings

Switch	D1	D2
SW1	OFF	ON

4） Prepare image files

For Linux: Copy all of image files except fsl-image-fb-sbc9000.sdcard to the flash image tool **Mfgtools-Rel-4.1.0_130816_MX6DL_UPDATER\ Profiles\MX6DL Linux Update\OS Firmware\files\,** the files directory should have the following files:

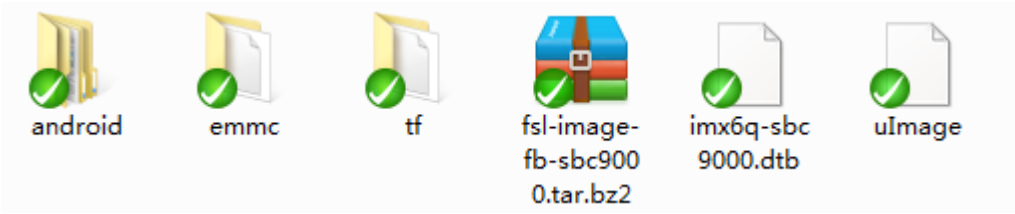


Figure 4-1 The files of files directory

For Android: Copy all of image files to the flash image tool **Mfgtools-Rel-4.1.0_130816_MX6DL_UPDATER\ Profiles\MX6DL Linux Update\OS Firmware\files\android\,** the android directory should have the following files:

Figure 4-2 The files of android directory

5） Run “MfgTool.exe” saved under “tools\Mfgtools-Rel-12.04.01_ER_MX6Q_UPDATER” and boot up SBC9000, then the software window will indicate that device has been found as shown blow;

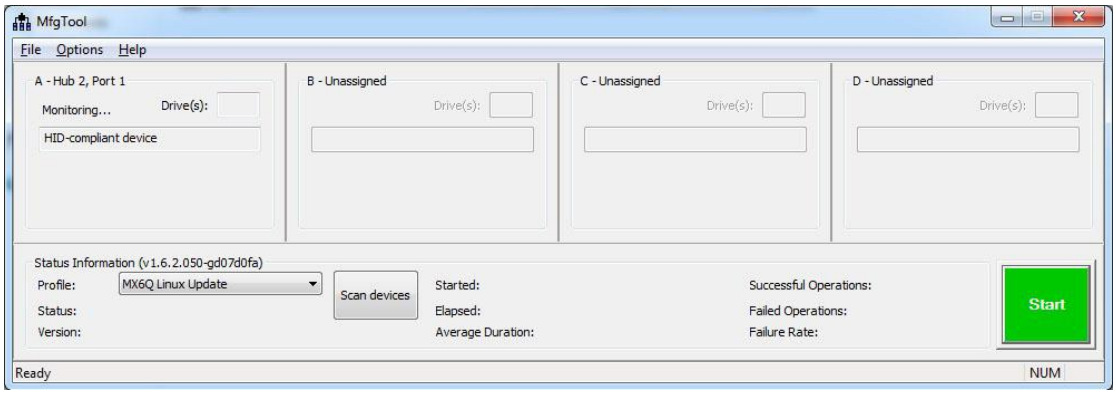


Figure 4-3 Mfgtool window

6） Select **Options** > **Configuration** on the menu bar to open the following window, then choose eMMC (default) or TF card as the target device to which system images (SBC9000 support Yocto and Android images) will be programmed;

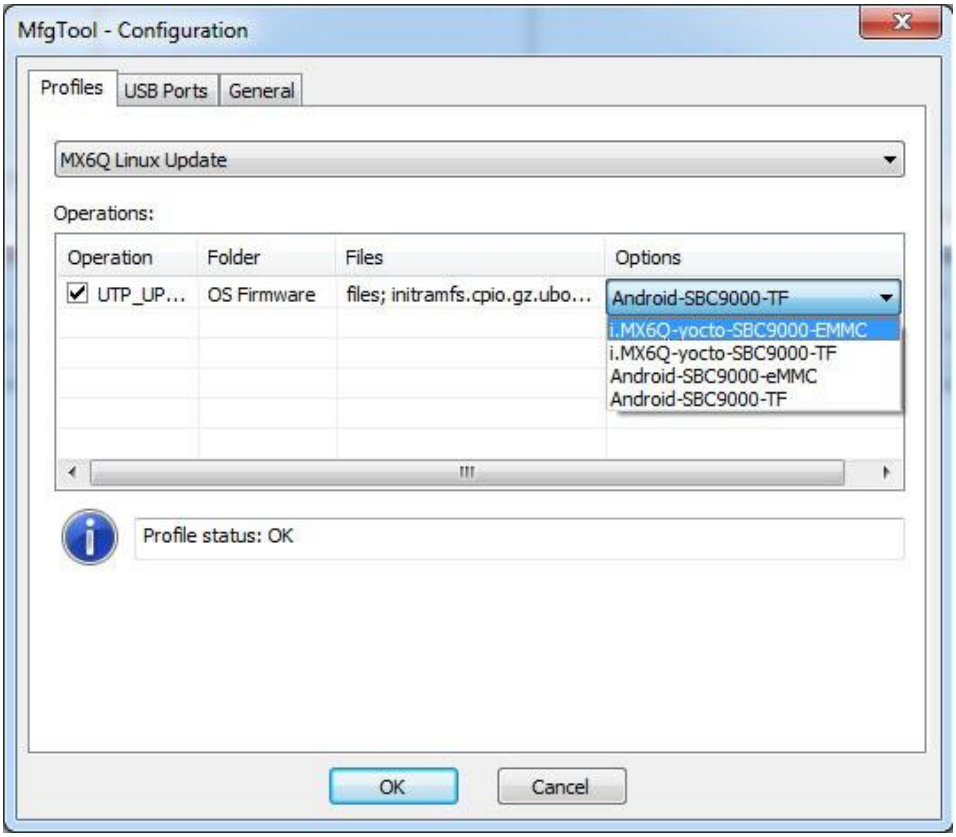


Figure 4-4 Mfgtool configurations

7） Click **Start** to start programming; When a pop-up window as shown below appears during programming, please click **Cancel**;

Figure 4-5 Start programming

8） When a green bar appears as shown below, please click **Stop** to finish programming;

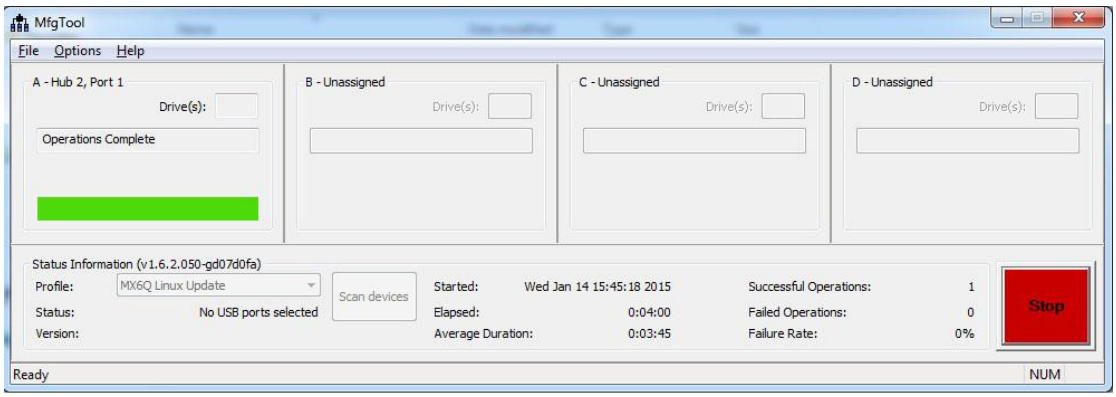


Figure 4-6 Programming finished

9） Power off SBC9000 and set it to eMMC or TF card booting mode by toggling the DIP switch SW1 according to the following table;

Table 4-2 eMMC booting mode

CPU Module	**Switch**	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**
	SW1	ON	OFF				
Expansion Board	**Switch**	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**
	SW1	ON	ON	OFF	ON	ON	ON

Table 4-3 TF card booting mode

CPU Module	**Switch**	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**
	SW1	ON	OFF				
Expansion Board	**Switch**	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**
	SW1	ON	OFF	OFF	ON	OFF	ON

10） After the boot mode is set, the system can be booted up when SBC9000 is powered on.

4.1.2 Using Linux Host to Download Linux to TF Card

1） Download Linux image file fsl-image-fb-sbc9000.sdcard for TF card from Embest's website or build new images by compiling Yocto projects (please refer to the section 5.1.2);

2） Execute the following instructions under Linux system to program images into a TF card;

- $ **sudo dd if= fsl-image-fb-sbc9000.sdcard of=/dev/sd<partition> bs=1M**
- $ **sync**

3） Power off SBC9000 and set the switch “SW1” to TF card booting mode according to the information in Table 4-3;

4） Insert the TF card onto SBC9000 and connect power supply to boot up Linux system.

4.2 Configuring Display Modes

Either Linux or Android for SBC9000 supports multiple display modes. Users can select one of the modes by configuring parameters under uboot. To enter uboot, please push any key on your keyboard when the prompt “Hit any key to stop autoboot” as shown blow appears in terminal window during system booting process.

Table 4-4 Enter uboot

```
U-Boot 2013.04-04992-g002bd44 (Sep 23 2014 - 14:48:51)

CPU:   Freescale i.MX6Q rev1.2 at 792 MHz
CPU:   Temperature 31 C, calibration data: 0x5654b67d
Reset cause: POR
Board: MX6Q-SBC9000
DRAM:  1 GiB
force_idle_bus: sda=0 scl=1 sda.gp=0xcb scl.gp=0x5
force_idle_bus: failed to clear bus, sda=0 scl=1
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
*** Warning - bad CRC, using default environment

wait_for_sr_state: Arbitration lost sr=93 cr=80 state=2020
i2c_init_transfer: failed for chip 0x4 retry=0
force_idle_bus: sda=0 scl=1 sda.gp=0xcb scl.gp=0x5
force_idle_bus: failed to clear bus, sda=0 scl=1
i2c_init_transfer: give up i2c_regs=021a8000
wait_for_sr_state: failed sr=a1 cr=80 state=2000
wait_for_sr_state: failed sr=a1 cr=80 state=2000
i2c_imx_stop:trigger stop failed
i2c_init_transfer: failed for chip 0x38 retry=0
force_idle_bus: sda=0 scl=1 sda.gp=0xcb scl.gp=0x5
force_idle_bus: failed to clear bus, sda=0 scl=1
i2c_init_transfer: give up i2c_regs=021a8000
```

```
wait_for_sr_state: Arbitration lost sr=93 cr=80 state=2020
i2c_init_transfer: failed for chip 0x48 retry=0
force_idle_bus: sda=0 scl=1 sda.gp=0xcb scl.gp=0x5
force_idle_bus: failed to clear bus, sda=0 scl=1
i2c_init_transfer: give up i2c_regs=021a8000
No panel detected: default to HDMI
unsupported panel HDMI
In:       serial
Out:      serial
Err:      serial
Net:      ----phy_id= 0x4dd072
FEC [PRIME]
Warning: failed to set MAC address

Normal Boot
Hit any key to stop autoboot:  0   (push any key on your keyboard to enter uboot)
MX6QSBC9000 U-Boot >
```

The following contents include instructions for different display modes;

- For 4.3” LCDs
  - MX6QSBC9000 U-Boot > **setenv dispmode video=mxcfb0:dev=lcd,4.3inch_LCD,if=RGB24 video=mxcfb1:dev=ldb,LDB-XGA,if=RGB666 fbmem=10M vmalloc=400M androidboot.console=ttymxc1 androidboot.hardware=freescale calibration**
  - MX6QSBC9000 U-Boot > **saveenv**
- For 7” LCDs
  - MX6QSBC9000 U-Boot > **setenv dispmode video=mxcfb0:dev=lcd,7inch_LCD,if=RGB24 video=mxcfb1:dev=ldb,LDB-XGA,if=RGB666 fbmem=10M vmalloc=400M androidboot.console=ttymxc1 calibration androidboot.hardware=freescale calibration**
  - MX6QSBC9000 U-Boot > **saveenv**
- For 9.7” LVDS displays
  - MX6QSBC9000 U-Boot > **setenv dispmode video=mxcfb0:dev=ldb,LDB-XGA,if=RGB666 video=mxcfb1:off fbmem=10M**

vmalloc=400M androidboot.console=ttymxc1 androidboot.hardware=freescale

- MX6QSBC9000 U-Boot > **saveenv**

- For HDMI displays (default)
  - MX6QSBC9000 U-Boot > **setenv dispmode**

    video=mxcfb0:dev=hdmi,1280x720M@60,if=RGB24

    video=mxcfb1:dev=ldb,LDB-XGA,if=RGB666 fbmem=40M vmalloc=400M

    androidboot.console=ttymxc1 androidboot.hardware=freescale
  - MX6QSBC9000 U-Boot > **saveenv**

Note:

📖 The default display resolution for HDMI is 1280x720. Users can change it to 1920x1080 or 640x480 for example, by replacing “dev=hdmi,1280x720M@60” in the instruction above with, for example “dev=hdmi, 1920x1080M@60”.

- For use of VGA8000
  - MX6Q SBC9000 U-Boot > **setenv dispmode**

    video=mxcfb0:dev=lcd,1024x768M@60,if=RGB24

    video=mxcfb1:dev=ldb,LDB-XGA,if=RGB666 fbmem=10M vmalloc=400M

    androidboot.console=ttymxc1 androidboot.hardware=freescale
  - MX6Q SBC9000 U-Boot > **saveenv**

Note:

📖 The default display resolution for VGA is 1024x768. Users can change it to 800x600, 1440x900 or 1280x1024 for example, by replacing “video=mxcfb0:dev=lcd,1024x768M@60” in the instruction above with, for example “video=mxcfb0:dev=lcd,800x600M@60”.

Chapter 5 Making Images

This Chapter will introduce how to make images by using BSP. The BSP for SBC9000 is a collection of binary, source code, and support files that can be used to create a u-boot bootloader, Linux kernel images, and Android filesystem.

Note:
🕮 All the following instructions are executed under Ubuntu system.

5.1 Making Linux Images

The following contests include two methods to make Linux system images: Direct Compilation method and Yocto method.

5.1.1 Direct Compilation

1） Please execute the following instructions to obtain cross-compiling tools;

- **$ cd ~**
- **$ git clone git://github.com/embest-tech/fsl-linaro-toolchain.git**

2） Download the latest Linux system source code from the “Download” tab on the page of SBC9000 at Embest’s website, or obtain the code from git by executing the following instructions;

- **$ git clone https://github.com/embest-tech/u-boot-imx.git**
- **$ git checkout -b embest_imx_v2013.04_3.10.17_1.0.0_ga origin/embest_imx_v2013.04_3.10.17_1.0.0_ga**
- **$ git clone https://github.com/embest-tech/linux-imx.git**
- **$ git checkout -b embest_imx_3.10.17_1.0.0_ga remotes/origin/embest_imx_3.10.17_1.0.0_ga**

3） Execute the following instructions to uncompress the source code downloaded;

- **$ cd ~**

- $ **tar xvf u-boot-imx.tar.bz2**
- $ **tar xvf linux-imx.tar.bz2**

4） Execute the following instructions to compile a boot image;

- $ **cd ~/u-boot-imx**
- $ **export ARCH=arm**
- $ **export CROSS_COMPILE=~/fsl-linaro-toolchain/bin/arm-fsl-linux-gnueabi-**
- $ **make distclean**
- $ **make mx6q_sbc9000_emmc_config**
- $ **make**

A new file named “u-boot.imx” can be found under the current directory after compilation is finished;

Note:

🕮 The instruction “**make mx6q_sbc9000_emmc_config**” is used for making a boot image for eMMC. If the system need to be booted from a TF card, please use “**make mx6q_sbc9000_tf_config**” instead.

5） Execute the following instructions to compile a kernel image;

- $ **export PATH=~/u-boot-imx/tools:$PATH**
- $ **cd ~/linux-imx**
- $ **export ARCH=arm**
- $ **export CROSS_COMPILE=~/fsl-linaro-toolchain/bin/arm-fsl-linux-gnueabi-**
- $ **make imx_v7_sbc9000_defconfig**
- $ **make uImage LOADADDR=0x10008000**
- $ **make imx6q-sbc9000.dtb**

After compilation is finished, two new files named “uImage” and “imx6q-sbc9000.dtb” can be found “arch/arm/boot/” and “arch/arm/boot/dts/” respectively.

Note:

- The tool “mkimage” used to make a kernel and ramfs is generated automatically and saved under tools/ when compiling the file “u-boot.bin”, and therefore the compilation of uboot should be done first before compiling a kernel image.
- Use the files “uImage”, “imx6q-sbc9000.dtb” built above to replace the files of the same names under Mfgtools-Rel-12.04.01_ER_MX6Q_UPDATER \Profiles\MX6Q Linux Update\OS Firmware\files\, and according to the boot mode you set(emmc or tf) to copy “u-boot.imx” to replace the files of the same name under emmc or tf directory that located in Mfgtools-Rel-12.04.01_ER_MX6Q_UPDATER \Profiles\MX6Q Linux Update\OS Firmware\files\,then verify the new Linux system by following the steps start from the step 2) in section 4.1.1.

5.1.2 Yocto Method

1） Execute the following instructions to obtain repo tool;

- $ **mkdir ~/bin**
- $ **curl https://raw.githubusercontent.com/android/tools_repo/master/repo > ~/bin/repo**
- $ **chmod a+x ~/bin/repo**
- $ **export PATH=~/bin:$PATH**

2） Execute the following instructions to build an environment for Yocto;

- $ **mkdir ~/fsl-arm-yocto-bsp**
- $ **cd ~/fsl-arm-yocto-bsp**
- $ **repo init --no-repo-verify --repo-url=git://github.com/android/tools_repo.git -u git://github.com/embest-tech/fsl-arm-yocto-bsp.git -b embest_imx-3.10.17-1.0.0_ga**
- $ **repo sync**

3） Execute the following instructions to build a Linux system with Yocto;

- $ **cd ~/fsl-arm-yocto-bsp**
- $ **MACHINE=sbc9000 source fsl-setup-release.sh -b build -e fb**
- $ **bitbake fsl-image-fb**

After compilation is finished, new files “u-boot.imx”, “uImage”, “uImage-imx6q-sbc9000.dtb” and “fsl-image-fb-sbc9000.tar.bz2” can be found under build/tmp/deploy/images/sbc9000. Please rename “uImage-imx6q-sbc9000.dtb” to

“imx6q-sbc9000.dtb”.

Note:

- Use the files “uImage”, “imx6q-sbc9000.dtb” and “fsl-image-fb-sbc9000.tar.bz2” built above to replace the files of the same names under Mfgtools-Rel-12.04.01_ER_MX6Q_UPDATER \Profiles\MX6Q Linux Update\OS Firmware\files\, and according to the boot mode you set(emmc or tf) to copy “u-boot.imx” to replace the files of the same name under emmc or tf directory that located in Mfgtools-Rel-12.04.01_ER_MX6Q_UPDATER \Profiles\MX6Q Linux Update\OS Firmware\files\,then verify the new Linux system by following the steps start from the step 2) in section 4.1.1.
- Yocto has different configurations for its images and Freescale provide some new features according to its SoC. Please refer to “Freescale_Yocto_Project_User's_Guide.pdf” for detailed information.

4） Rebuild a compilation environment;

If a “build” directory has been already created using the script “fsl-setup-release.sh”, there is no need to execute it again and just use the script “setup-environment” to reconfigure environment variables required by compilation after PC reboots or when a new Linux terminal window is opened.

- $ **cd ~/fsl-arm-yocto-bsp**
- $ **source setup-environment build**

5） Compile u-boot or kernel image separately;

- $ **bitbake u-boot-sbc9000** //compiling u-boot
- $ **bitbake linux-sbc9000** //compiling kernel

Note:

- The instruction “-c compile -f” is required to recompile Yocto Project if the source code has been modified:
  $ **bitbake -c compile -f u-boot-sbc9000** //recompiling u-boot
  $ **bitbake -c compile -f linux-sbc9000** //recompiling kernel

5.2 Making Android Images

1） Execute the following instructions to obtain repo tool;

- $ **mkdir ~/bin**
- $ **curl https://raw.githubusercontent.com/android/tools_repo/master/repo > ~/bin/repo**
- $ **chmod a+x ~/bin/repo**
- $ **export PATH=~/bin:$PATH**

2） Execute the following instructions to obtain Android source code;

- $ **mkdir ~/android-imx6-kk4.4.2-1.0.0**
- $ **cd ~/android-imx6-kk4.4.2-1.0.0**
- $ **repo init --no-repo-verify --repo-url=git://github.com/android/tools_repo.git -u git://github.com/embest-tech/imx-manifest.git -m embest_android_kk4.4.2_1.0.0**
- $ **repo sync**

Note:

Android source code can also be downloaded from the “Download” tab on the page of SBC9000 at Embest’s website. Please execute the following instructions to uncompress the code after it is downloaded;

$ **cd ~**

$ **tar xvf android-imx6-kk4.4.2-1.0.0-xxx.tar.bz2**（Replace “xxx” according to the file name downloaded）

3） Open the file “BoardConfig.mk” saved under “android-imx6-kk4.4.2-1.0.0/device/fsl/sbc9000_6solo/” with the notepad and modify the parameter “BUILD_TARGET_LOCATION” by referring to the following options to choose a boot mode;

- eMMC boot mode: BUILD_TARGET_LOCATION ？=emmc
- TF card boot mode: BUILD_TARGET_LOCATION ？=sdmmc

4） Execute the following instructions to compile Android images;

- $ **cd ~/android-imx6-kk4.4.2-1.0.0**
- $ **source build/envsetup.sh**

- $ **lunch sbc9000_6q-user**
- $ **make clean**
- $ **make**

The images generated can be found under "android-imx6-kk4.4.2-1.0.0/out/target/product/sbc9000_6q/"; the following table shows all the image names and new directories;

Table 5-1 Images and directories

Images/directories	Descriptions
root/	Root filesystem directory, mounted at the root directory
system/	Android system directory mounted at /system
data/	Android data section mounted at /data
recovery/	Root filesystem directory under Recovery mode; it is not used directly
boot.img	Composition image, includes kernel zImage, ramdisk and boot parameters
ramdisk.img	ramdisk image generated under Root/; it is not used directly
system.img	EXT4 image generated under System/; it can be written to the SYSTEM partition of SD/eMMC storage media by using the command "dd"
recovery.img	EXT4 image generated under Recovery/; it can be written to the RECOVERY partition of SD/eMMC storage media by using the command "dd"
u-boot.bin	Uboot image

Note:

- Android images need to be built under user mode;
- Please visit http://source.android.com/source/building.html for more information on building development environment.

5） Execute the following instructions to make boot.img separately;

- $ **source build/envsetup.sh**
- $ **lunch sbc9000_6q-user**
- $ **make bootimage**

The new file "boot.img" can be found under "android-imx6-kk4.4.2-1.0.0/out/target/product/sbc9000_6q/".

Note:

According to the boot mode you set, use the files “boot.img”, “recovery.img”, “system.img” and “u-boot.bin” built above to replace the files of the same names under emmc or tf directory that located in “Mfgtools-Rel-4.1.0_130816_MX6DL_UPDATER\Profiles\MX6DL Linux Update\OS Firmware\files\android”, then verify the new Android system by following the steps start from the step 2) in section 4.1.1.

5.3 Compiling Linux Upper-Layer Applications with Yocto

This section will use an example application “hello_world.c” to show you how to compile a Linux upper-layer application by using Yocto.

1） Please execute the following instructions to generate cross-compiling tools;

- $ **cd ~/fsl-arm-yocto-bsp**
- $ **MACHINE=sbc9000 source fsl-setup-release.sh -b build -e fb**
- $ **bitbake meta-toolchain //generate cross-compiling tools**
- $ **cd ~/fsl-arm-yocto-bsp/build/tmp/deploy/sdk/**

After the instructions above are executed, a script file for installing cross-compiling tools named “poky-eglibc-x86_64-meta-toolchain-cortexa9hf-vfp-neon-toolchain-1.5.2.sh” can be found under the current directory.

2） Execute the following instructions to install the cross-compiling tools; the default installation directory is “/opt/poky/1.5.2”;

- $**sudo ./poky-eglibc-x86_64-meta-toolchain-cortexa9hf-vfp-neon-toolchain-1.5.2.sh**
- $**source/opt/poky/1.5.2/environment-setup-cortexa9hf-vfp-neon-poky-linux-gnueabi**

3） Execute the following instruction to compile “hello_world.c”;

- $ **arm-poky-linux-gnueabi-gcc -o hello_world hello_world.c**

Note:

Upper-layer applications should use a hard-fload compiler with the same version as that used by Yocto projects.

Chapter 6 Tests

This chapter will introduce how to run a test on the different interfaces on SBC9000. All the tests will be conducted under Yocto system and all the instructions are executed in HyperTerminal unless otherwise specified.

Note:

🕮 Press “Ctrl+C” on your keyboard can exit testing programs.

6.1 LED Test

SBC9000 has three user-defined LEDs which are D4 (LED) on CPU module, D39 (LED2) and D41 (LED1) on Expansion board.

1） Execute the following instructions to turn off the LEDs;

- root@sbc9000:~# **echo 1 > /sys/class/leds/user_led1/brightness**
- root@sbc9000:~# **echo 1 > /sys/class/leds/user_led2/brightness**
- root@sbc9000:~# **echo 1 > /sys/class/leds/sys_led/brightness**

2） Execute the following instructions to turn on the LEDs;

- root@sbc9000:~# **echo 0 > /sys/class/leds/user_led1/brightness**
- root@sbc9000:~# **echo 0 > /sys/class/leds/user_led2/brightness**
- root@sbc9000:~# **echo 0 > /sys/class/leds/sys_led/brightness**

6.2 Button Test

There is a user-defined button (S8) on the Expansion board. Please execute the following instruction to run a test on the button with the tool “evtest”.

- root@sbc9000:~# **evtest /dev/input/event1**

Then push the button S8; The terminal window will print information as shown below;

Table 6-1 Button test information

```
Input driver version is 1.0.1evdev: (EVIOCGBIT): Suspicious buffer size
511, limiting output to 64 bytes. See
http://userweb.kernel.org/~dtor/eviocgbit-bug.html

Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys"
Supported events:
  Event type 0 (Sync)
  Event type 1 (Key)
    Event code 102 (Home)
Testing ... (interrupt to exit)
Event: time 278.544212, type 1 (Key), code 102 (Home), value 1
Event: time 278.544220, ---------- Report Sync ------------
Event: time 278.714484, type 1 (Key), code 102 (Home), value 0
Event: time 278.714487, ---------- Report Sync ------------
Event: time 279.080212, type 1 (Key), code 102 (Home), value 1
Event: time 279.080217, ---------- Report Sync ------------
Event: time 279.293936, type 1 (Key), code 102 (Home), value 0
Event: time 279.293940, ---------- Report Sync ------------
```

The “value 1” in the table above indicates the button was pushed, the “value 0” indicates the button was released.

6.3 Touchscreen Test

1） Execute the following instruction to run touchscreen calibration program after SBC9000 boots up, then tap the symbols “+” displayed on the screen to finish calibration;

- root@sbc9000:~# **ts_calibrate**

2） After calibration completed, execute the following instruction to run test programs and follow the prompts on the screen to finish test by drawing points and lines;

- root@sbc9000:~# **ts_test**

6.4 RTC Test

1） Execute the following instruction to set the system clock of SBC9000 to 5 pm, 11th of

March, 2014.

- root@sbc9000:~# **date 031117002014**

The terminal window will print information as shown below;

Table 6-2 Set system clock

Tue Mar 11 17:00:00 UTC 2014

2） Execute the following instruction to write system clock into RTC;

- root@sbc9000:~# **hwclock -w**

3） Execute the following instruction to read RTC;

- root@sbc9000:~# **hwclock**

The terminal window will print information as shown below;

Table 6-3 Read RTC

Tue Mar 11 17:01:01 2014 0.000000 seconds

The information above shows the system clock has been saved in hardware clock.

4） Reboot SBC9000 and execute the following instructions to restore system clock;

- root@sbc9000:~# **hwclock -s**
- root@sbc9000:~# **date**

The terminal window will print information as shown below;

Table 6-4 System clock

Tue Mar 11 17:03:33 UTC 2014

The information above shows the system clock has been restored from hard clock;

Note:

A coin battery with model code CR1220 is required on SBC9000's Expansion board to maintain a correct hardware clock after the system reboots. The battery is not supplied with the product and therefore needs to be purchased separately;

6.5 TF Card Test

1） Insert a TF card onto SBC900, and then the system will detect the card and display information as shown below;

Table 6-5 TF card information

mmc1: host does not support reading read-only switch. assuming write-enable. mmc1: new high speed SD card at address b7f5 mmcblk1: mmc0:b7f5 SD02G 1.83 GiB mmcblk1: p1

2） Execute the following instructions to mount the TF card to “/mnt” directory and view the contents in the card;

- root@sbc9000:~# **mount -t vfat /dev/mmcblk1p1 /mnt**
- root@sbc9000:~# **ls /mnt**

The terminal window will print information as shown below;

Table 6-6 TF card contents

Mlo	nand	ramdisk.gz	u-boot.bin	uImage

Note:

Please insert TF card as fast as you can, or the system might not detect it. If the card is not detected at the first time, please try again.

6.6 USB HOST Test

1） Insert a flash drive into the USB hub interface of SBC9000, and then system will detect the device and display the following information;

Table 6-7 USB device information

usb 2-1.4: new high speed USB device number 3 using fsl-ehci scsi1 : usb-storage 2-1.4:1.0 scsi 1:0:0:0: Direct-Access Generic- Multi-Card 1.00 PQ: 0 ANSI: 0 CCS

```
sd 1:0:0:0: [sda] 15394816 512-byte logical blocks: (7.88 GB/7.34 GiB)
sd 1:0:0:0: [sda] Write Protect is off
sd 1:0:0:0: [sda] No Caching mode page present
sd 1:0:0:0: [sda] Assuming drive cache: write through
sd 1:0:0:0: [sda] No Caching mode page present
sd 1:0:0:0: [sda] Assuming drive cache: write through
 sda: sda1 sda2
sd 1:0:0:0: [sda] No Caching mode page present
sd 1:0:0:0: [sda] Assuming drive cache: write through
sd 1:0:0:0: [sda] Attached SCSI removable disk
EXT2-fs (sda2): warning: mounting ext3 filesystem as ext2
EXT2-fs (sda2): warning: mounting unchecked fs, running e2fsck is
recommended
```

2） Execute the following instruction to mount the flash drive to “/mnt” directory;

- root@sbc9000:~# **mount -t vfat /dev/sda1 /mnt/**

3） Execute the following instruction to view the contests of the drive;

- root@sbc9000:~# **ls /mnt/**

The terminal window will print information as shown below;

Table 6-8 Flash drive contents

```
speed.avi      madplay      i believe.mp3  i believe.wav
```

6.7 USB Device Test

By default, the OTG interface (J7) of SBC9000 is working as USB Device under Linux system. A network communication can be built by connecting the OTG interface of SBC9000 to an USB host port on your PC with a MiniUSB cable. Please follow the steps listed below to finish USB Device test (under Windows 7).

1） Use an USB MINI B-to-USB A cable to connect SBC9000 to your PC and then install Linux USB Ethernet driver; (Please refer to Appendix 2 for detailed installation method)

2） Execute the following instructions to set and view the IP address of USB OTG port on

SBC9000 (the IP used here is only for reference; you can select any other IP as long as it is NOT in the same network segment as your PC's Ethernet port is);

- root@sbc9000:~# **ifconfig usb0 192.168.1.115**
- root@sbc9000:~# **ifconfig**

The terminal window will print information as shown below;

Table 6-9 IP config of USB port

```
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  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)

usb0      Link encap:Ethernet  HWaddr 1E:B1:B5:11:E5:46
          inet addr:192.168.1.115  Bcast:192.168.1.255
Mask:255.255.255.0
          inet6 addr: fe80::1cb1:b5ff:fe11:e546/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500
Metric:1
          RX packets:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1338 (1.3 KiB)  TX bytes:4994 (4.8 KiB)
```

3） Click "Control Panel" > "Network and Internet" > "Network and Sharing Center" to find a new Local Area Connection as shown below(Windows 7 for example here);

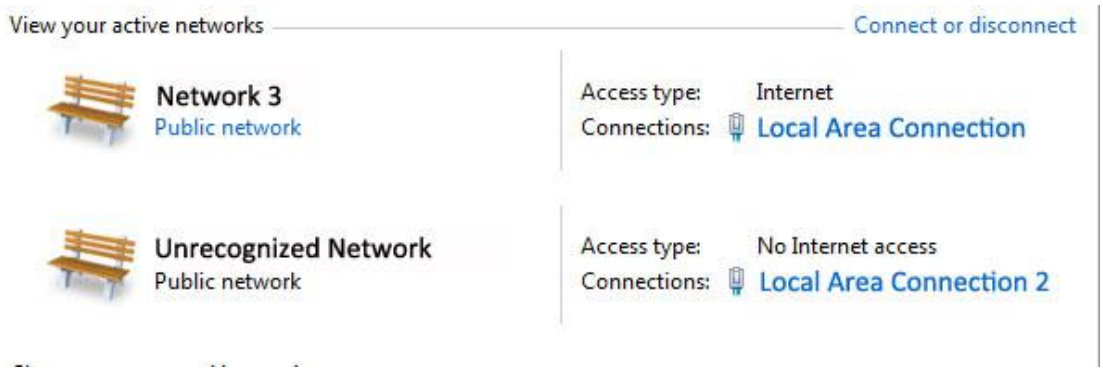


Figure 6-1 New Local Area Connection

4） Click the new connection “Local Area Connection 2” to open the “Local Connection Properties” window, and then select “Properties” > “Internet Protocol Version 4(TCP/IPv4” to open the following window;

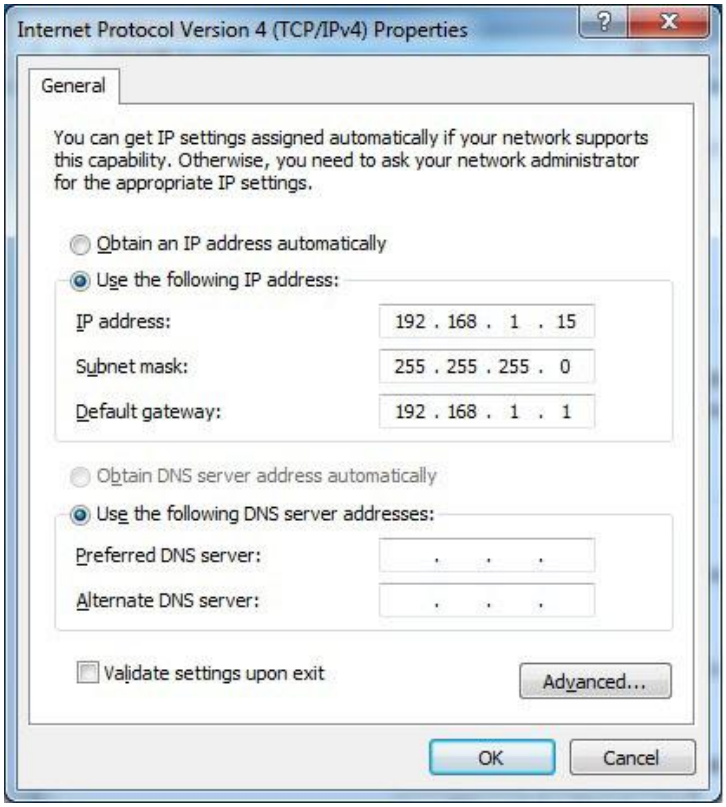


Figure 6-2 IP settings

Set an IP address that is in the same network segment as SBC9000’s USB OTG port is, then click “OK”.

5） Execute the following instruction to verify the network connection;

- root@sbc9000:~# **ping 192.168.1.15**

The terminal window will print information as shown below;

Table 6-10 Ping information

PING 192.168.1.15 (192.168.1.15): 56 data bytes
64 bytes from 192.168.1.15: seq=0 ttl=128 time=0.885 ms
64 bytes from 192.168.1.15: seq=1 ttl=128 time=0.550 ms

The information shown above indicates the network connection is working properly.

6.8 Audio Test

SBC9000 has audio input and output ports on the board. The system has built-in alsa-utils audio playback and recording tool. Please connect a headphone to SBC9000 and follow the steps list below to finish test.

1） Execute the following instructions to start audio recording;

- root@sbc9000:~# **cd /tmp**
- root@sbc9000:/tmp# **arecord -t wav -c 1 -r 44100 -f S16_LE -v k**

The terminal window will print information as shown below;

Table 6-11 Audio recording is ready

```
Recording WAVE 'k' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Plug PCM: Hardware PCM card 0 'k14' device 0 subdevice 0
Its setup is:
   stream          : CAPTURE
   access          : RW_INTERLEAVED
   format          : S16_LE
   subformat       : STD
   channels        : 2
   rate            : 44100
   exact rate      : 44100 (44100/1)
   msbits          : 16
   buffer_size     : 22052
   period_size     : 5513
   period_time     : 125011
   tstamp_mode     : NONE
   period_step     : 1
   avail_min       : 5513
   period_event : 0
   start_threshold    : 1
   st  op_threshold      : 22052
   silence_threshold: 0
   silence_size : 0
   boundary        : 1445199872
   appl_ptr        : 0
   hw_ptr          : 0
```

Now you can start audio recording with the headphone.

2） Execute the following instruction to play the audio you recorded;

- root@sbc9000:/tmp# **aplay -t wav -c 2 -r 44100 -f S16_LE -v k**

The terminal window will print information as shown below;

Table 6-12 Audio playback

```
Playing WAVE 'k' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Plug PCM: Hardware PCM card 0 'k14' device 0 subdevice 0
Its setup is:
   stream          : PLAYBACK
   access          : RW_INTERLEAVED
   format          : S16_LE
   subformat       : STD
   channels        : 2
   rate            : 44100
   exact rate      : 44100 (44100/1)
   msbits          : 16
   buffer_size     : 22052
   period_size     : 5513
   period_time     : 125011
   tstamp_mode     : NONE
   period_step     : 1
   avail_min       : 5513
   period_event : 0
   start_threshold     : 22052
   stop_threshold      : 22052
   silence_threshold: 0
   silence_size : 0
   boundary        : 1445199872
   appl_ptr        : 0
   hw_ptr          : 0
```

Now you can hear the audio playback for the headphone.

6.9 HDMI Audio Test

1） Use a HDMI cable to connect SBC9000 to a HDMI display;

2） Connect the power and set the display mode to HDMI according the method described in section 错误!未找到引用源。;

3） Execute the following instruction to test audio output from HDMI interface;

- root@sbc9000:~# **aplay Windows.wav -D plughw:1,0**

6.10 Ethernet Test

1） Use a RJ45 cable to connect SBC9000 to your PC and execute the following instruction to set IP address of SBC9000; (the IP address of SBC9000 should be set to the same network segment as your PC)

- root@sbc9000:~# **ifconfig eth0 192.168.8.52**

The terminal window will print information as shown below;

Table 6-13 Set IP address

```
eth0: Freescale FEC PHY driver [Generic PHY] (mii_bus:phy_addr=1:04,
irq=-1)
root@sbc9000:~# PHY: 1:04 - Link is Up - 100/Full
```

2） Execute the following instruction to view the network configurations;

- root@sbc9000:~# **ifconfig**

Table 6-14 View IP address

```
eth0      Link encap:Ethernet  HWaddr 1E:ED:19:27:1A:B3
          inet addr:192.168.8.52  Bcast:192.168.8.255
Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500
Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:366 (366.0 B)  TX bytes:0 (0.0 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)
```

3） Execute the following instruction to verify the connectivity of the network;

- root@sbc9000:~# **ping 192.168.8.1**

The terminal window will print information as shown below;

Table 6-15 Verify connectivity

```
PING 192.168.8.1 (192.168.8.1): 56 data bytes
64 bytes from 192.168.8.1: seq=0 ttl=64 time=0.378 ms
64 bytes from 192.168.8.1: seq=1 ttl=64 time=0.285 ms
64 bytes from 192.168.8.1: seq=2 ttl=64 time=0.222 ms
64 bytes from 192.168.8.1: seq=3 ttl=64 time=9.928 ms
64 bytes from 192.168.8.1: seq=4 ttl=64 time=0.217 ms
64 bytes from 192.168.8.1: seq=5 ttl=64 time=0.289 ms

--- 192.168.8.1 ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 0.217/1.886/9.928 ms
```

6.11 CAN Test

There are two CAN interfaces extended from J25 on SBC9000 and named CAN1 and CAN2. The interfaces are respectively associated to device nodes can0 and can1 under Linux system. This test will use can1 as the sender and can0 as the receiver. Please refer to the figure shown below to finish hardware connections between two pieces of SBC9000.

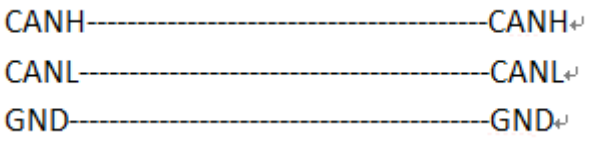


Figure 6-3 CAN connections

The pin definitions of J25 are shown below;

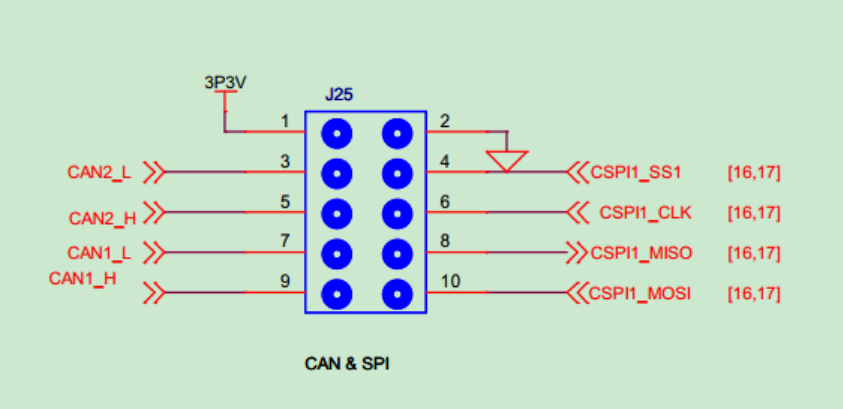


Figure 6-4 J25 pin definitions

1） Please execute the following instructions to set baudrate to 125KBPS and enable CAN devices;

- root@sbc9000:~# **ip link set can0 type can bitrate 125000 triple-sampling on**
- root@sbc9000:~# **ip link set can1 type can bitrate 125000 triple-sampling on**
- root@sbc9000:~# **ip link set can0 up**
- root@sbc9000:~# **ip link set can1 up**

2） Execute the following instruction to set can0 as receiver;

- root@sbc9000:~# **candump can0 &**

3） Execute the following instructions to send data from can1;

- root@sbc9000:~# **cansend can1 123#1122334455667788**

The terminal window will print information as shown below;

Table 6-16 Data received by can0

can0 123 [8] 11 22 33 44 55 66 77 88

4） Execute the following instructions to disable links between CAN devices;

- root@sbc9000:~# **ip link set can0 down**
- root@sbc9000:~# **ip link set can1 down**

Note:

- The instruction “cansend” only send data for once each time when it is executed.
- It is not necessary to set baudrate of CAN devices to 125000, but it has to be communicating at the same speed for both sides. The baudrate should be changed when the CAN links are disabled.

6.12 Serial Interface Test

Apart from debug serial interface, SBC9000 has three available serial interfaces UART1 (ttymxc0), UART3 (ttymxc2) and UART5 (ttymxc4). The following test will take UART3 as an example and suited for the rest of serial interfaces.

1） Execute the following instruction to run serial interface test program;

- root@sbc9000:~# **uart_test -d /dev/ttymxc2 -b 115200**

2） Short the transmission pin (TX) and reception pin (RX) of UART3 to build a transmission loop. The terminal window will print information as shown below;

Table 6-17 Serial interface test

```
/dev/ttymxc2 SEND: 1234567890
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 1
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 2
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 3
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 4
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 5
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 6
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 7
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 8
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 9
/dev/ttymxc2 RECV 1 total
```

```
/dev/ttymxc2 RECV: 0
```

6.13 Mini-PCIe Test

Connect a MC2716 (CDMA2000 module, purchased separately) that has been equipped with a China Telecom SIM card to the Mini-PCIe slot (supports hot plugging) on SBC9000, and follow the steps listed below to finish test.

1） Execute the following instruction to load drivers for MC2716;

- root@sbc9000:~# **modprobe usbserial vendor=0x19d2 product=0xffed**

The terminal window will print information as shown below;

Table 6-18 Load drivers

```
usbcore: registered new interface driver usbserial
USB Serial support registered for generic
usbserial_generic 2-1.1:1.0: generic converter detected
usb 2-1.1: generic converter now attached to ttyUSB0
usbserial_generic 2-1.1:1.1: generic converter detected
usb 2-1.1: generic converter now attached to ttyUSB1
usbserial_generic 2-1.1:1.2: generic converter detected
usb 2-1.1: generic converter now attached to ttyUSB2
usbserial_generic 2-1.1:1.3: generic converter detected
usb 2-1.1: generic converter now attached to ttyUSB3
usbcore: registered new interface driver usbserial_generic
usbserial: USB Serial Driver core
```

2） Execute the following instruction to start dialing;

- root@sbc9000:~# **pppd connect 'chat -v "" "AT" "" "AT" "" "AT&C1" "" "AT" "" "ATDT#777 CONNECT"' user CARD password CARD /dev/ttyUSB0 115200 updetach nocrtscts nocdtrcts multilink usepeerdns defaultroute debug**

The terminal window will print information as shown below;

Table 6-19 Start dialing

```
Serial connection established.
using channel 2
Starting negotiation on /dev/ttyUSB0
rcvd [LCP ConfReq id=0x1 <asyncmap 0x0> <auth chap MD5> <magic
```

```
0xd6709950> <pcomp> <accomp>]
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x6734606d>
<pcomp> <accomp> <mrru 1500> <endpoint [MAC:1e:ed:19:27:1a:b3]>]
sent [LCP ConfAck id=0x1 <asyncmap 0x0> <auth chap MD5> <magic
0xd6709950> <pcomp> <accomp>]
rcvd [LCP ConfRej id=0x1 <mrru 1500> <endpoint
[MAC:1e:ed:19:27:1a:b3]>]
sent [LCP ConfReq id=0x2 <asyncmap 0x0> <magic 0x6734606d>
<pcomp> <accomp>]
rcvd [LCP ConfAck id=0x2 <asyncmap 0x0> <magic 0x6734606d>
<pcomp> <accomp>]
rcvd [CHAP Challenge id=0x1
<911cc74daa1650438af632f437aefc73ad064933>, name = ""]
sent [CHAP Response id=0x1 <b3ccdba9f56ad3a146d0fa3478bc5e41>,
name = "CARD"]
rcvd [CHAP Success id=0x1 ""]
CHAP authentication succeeded
CHAP authentication succeeded
Using interface ppp0
sent [CCP ConfReq id=0x1 <deflate 15> <deflate(old#) 15> <bsd v1 15>]
sent [IPCP ConfReq id=0x1 <compress VJ 0f 01> <addr 0.0.0.0> <ms-dns1
0.0.0.0> <ms-dns3 0.0.0.0>]
rcvd [IPCP ConfReq id=0x1 <compress VJ 0f 00> <addr 115.168.82.165>]
sent [IPCP ConfAck id=0x1 <compress VJ 0f 00> <addr 115.168.82.165>]
rcvd [LCP ProtRej id=0x1 80 fd 01 01 00 0f 1a 04 78 00 18 04 78 00 15 03
2f]
Protocol-Reject for 'Compression Control Protocol' (0x80fd) received
rcvd [IPCP ConfNak id=0x1 <addr 14.27.146.31> <ms-dns1
202.96.128.86> <ms-dns3 202.96.134.133>]
sent [IPCP ConfReq id=0x2 <compress VJ 0f 01> <addr 14.27.146.31>
<ms-dns1 202.96.128.86> <ms-dns3 202.96.134.133>]
rcvd [IPCP ConfAck id=0x2 <compress VJ 0f 01> <addr 14.27.146.31>
<ms-dns1 202.96.128.86> <ms-dns3 202.96.134.133>]
local  IP address 14.27.146.31
remote IP address 115.168.82.165
primary     DNS address 202.96.128.86        //DNS address obtained
secondary DNS address 202.96.134.133
```

3） Execute the following instruction to add DNS address to the file “resolv.conf”; The DNS address can be found in the information of terminal window after dialing succeeds.

- root@sbc9000:~# **echo nameserver 202.96.128.86 > /etc/resolv.conf**

4） Execute the following instruction to verify if MC2716 connects to Internet properly;

- root@sbc9000:~# **ping www.baidu.com**

The terminal window will print information as shown below;

Table 6-20 Ping information

```
PING www.baidu.com (220.181.6.19): 56 data bytes
64 bytes from 220.181.6.19: seq=0 ttl=56 time=91.491 ms
64 bytes from 220.181.6.19: seq=1 ttl=56 time=101.196 ms
64 bytes from 220.181.6.19: seq=2 ttl=56 time=95.459 ms
64 bytes from 220.181.6.19: seq=3 ttl=56 time=96.893 ms

--- www.baidu.com ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 91.491/96.259/101.196 ms
```

Note:

🕮 The first port of USB HUB would be disabled when Mini-PCIe interface is in use.

6.14 PCI-E Test

6.14.1 Test 1

Connect a “PCI-E to USB” adapter to the PCI-E slot on SBC9000 and power on the board, then insert a flash drive to the USB interface of the adapter. The system will detect the flash drive automatically.

6.14.2 Test 2

Connect an AR5B93 WLAN module (needs to purchased separately) to the PCI-E slot on SBC9000 and power on the board.

1） Execute the following instructions to load the module;

- root@sbc9000:~# **wpa_passphrase GK-TIOP-TEST Embest8877 >/etc/wpa_supplicant.conf**
- root@sbc9000:~# **wpa_supplicant -B -P /var/run/wpa_supplicant.wlan0.pid -i wlan0 -c /etc/wpa_supplicant.conf -D wext**

Table 6-21 Load the module

```
P /var/run/wpa_supplicant.wlan0.pid -i wlan0
Successfully initialized wpa_supplicant
rfkill: Cannot open RFKILL control device
IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
root@sbc9000:~# wlan0: authenticate with 94:0c:6d:17:0a:bc
wlan0: send auth to 94:0c:6d:17:0a:bc (try 1/3)
wlan0: authenticated
ath9k 0000:01:00.0 wlan0: disabling HT as WMM/QoS is not supported by
the AP
ath9k 0000:01:00.0 wlan0: disabling VHT as WMM/QoS is not supported by
the AP
wlan0: associate with 94:0c:6d:17:0a:bc (try 1/3)
wlan0: RX AssocResp from 94:0c:6d:17:0a:bc (capab=0x431 status=0
aid=4)
IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
wlan0: associated
```

2） Execute the following instruction to obtain a dynamic IP address automatically;

- root@DevKit8600:~# **udhcpc -i wlan0**

The terminal window will print information as shown below;

Table 6-22 Obtain IP

```
udhcpc (v1.21.1) started
Sending discover...
Sending select for 192.168.8.169...
Lease of 192.168.8.169 obtained, lease time 86400
/etc/udhcpc.d/50default: Adding DNS 192.192.192.248
/etc/udhcpc.d/50default: Adding DNS 202.96.134.133
/etc/udhcpc.d/50default: Adding DNS 202.96.128.86
```

3） Execute the following instruction to verify the Internet connectivity;

- root@sbc9000:~# **ping www.baidu.com**

The terminal window will print information as shown below;

Table 6-23 Ping information

```
PING www.baidu.com (180.97.33.107): 56 data bytes
64 bytes from 180.97.33.107: seq=0 ttl=54 time=35.663 ms
64 bytes from 180.97.33.107: seq=1 ttl=54 time=35.835 ms
64 bytes from 180.97.33.107: seq=2 ttl=54 time=38.501 ms
```



```
64 bytes from 180.97.33.107: seq=3 ttl=54 time=35.776 ms
^C
--- www.baidu.com ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 35.663/36.443/38.501 ms
```

Note:

PCI-E interface does NOT support hot plugging.

6.15 Backlight Test

The brightness of the backlight is ranged from 0 to 7. 0 indicates turning off the backlight, while 7 means the highest brightness.

6.15.1 LCD Backlight Test

1） Execute the following instruction to view the current brightness value;

- root@sbc9000:~# **cat /sys/class/backlight/backlight-lcd.27/brightness**

2） Execute the following instruction to turn off LCD backlight;

- root@sbc9000:~# **echo 0 > /sys/class/backlight/backlight-lcd.27/brightness**

3） Execute the following instruction to set the backlight to the highest brightness;

- root@sbc9000:~# **echo 7 > /sys/class/backlight/backlight-lcd.27/brightness**

6.15.2 Capacitive Touchscreen Backlight Test

1） Execute the following instruction to view the current brightness value;

- root@sbc9000:~# **cat /sys/class/backlight/backlight-ldb.28/brightness**

2） Execute the following instruction to turn off backlight;

- root@sbc9000:~# **echo 0 > /sys/class/backlight/backlight-ldb.28/brightness**

3） Execute the following instruction to set the backlight to the highest brightness;

- root@sbc9000:~# **echo 7 > /sys/class/backlight/backlight-ldb.28/brightness**

6.16 SATA Test

1） Connect a SATA hard drive (needs to be purchased separately) to the SATA slot on SBC9000 and power on the drive, then execute the following instruction to load driver for SATA after the system boots up;

- root@ SBC9000 ~$ **modprobeahci_imx**

2） After the driver is loaded successfully, the system will mount the SATA hard drive automatically; Please execute the following instruction to view the directory where the hard drive is mounted to;

- root@ SBC9000 ~$ **df -h**

Table 6-24 Directory list

Filesystem	Size	Used	Available	Use%	Mounted on
/dev/root	3.4G	484.0M	2.8G	14%	/
devtmpfs	340.0M	4.0K	340.0M	0%	/dev
tmpfs	500.1M	192.0K	499.9M	0%	/run
tmpfs	500.1M	52.0K	500.1M	0%	/var/volatile
/dev/mmcblk0p1 /media/mmcblk0p1	19.8M	5.8M	14.0M	29%	
/dev/sda5	**74.5G**	**480.8M**	**74G**	**1%**	**/media/sda5**

The above information shows that the SATA hard drive has been mounted to “/media/sda5”.

3） Execute the following instruction to view the contents of the hard drive;

- root@ SBC9000 ~$ **ls /media/sda5**

Table 6-25 Contents of hard drive

can.txt

Appendix 1 – Installing Ubuntu System

As we all know, an appropriate development environment is required for software development. The DVD-ROM attached with product has contained a development environment which needs to be installed under Linux system. If you are working on a PC running Windows, you have to create a Linux system first, and then you can install the environment. Here we recommend using VirtualBox – a virtual machine software to accommodate Ubuntu Linux system under Windows. The following sections will introduce the installation processes of VirtualBox and Ubuntu system.

Installing VirtualBox

You can access http://www.virtualbox.org/wiki/Downloads to download the latest version of VirtualBox. VirtualBox requires 512MB memory space at least. A PC with memory space of more than 1GB would be preferred.

1） The installation process is simple and will not be introduced. Please start VirtualBox from the **Start** menu of Windows, and then click **New** in VirtualBox window. A pop-up window **Create New Virtual Machine** will be shown as below;

Figure 1 Create new virtual machine

Click **Next** to create a new virtual machine.

2） Enter a name for the new virtual machine and select operating system type as shown below;

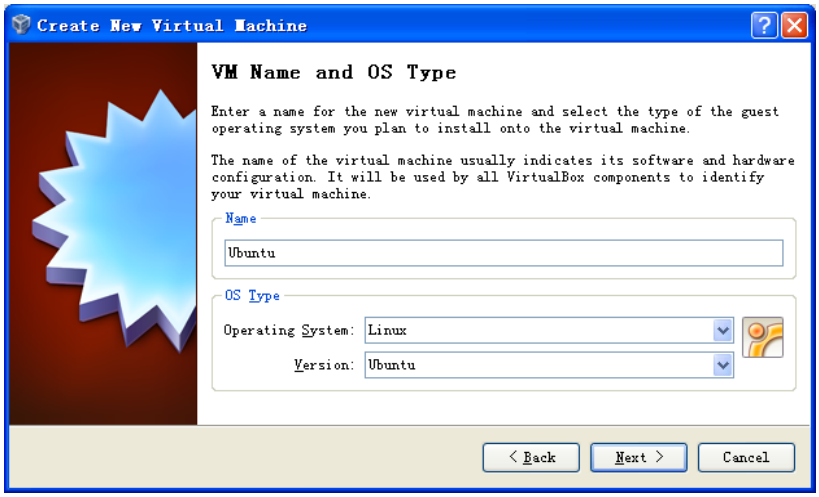


Figure 2 Name and OS type of virtual machine

Enter a name in the **Name** field, e.g. Ubuntu, and select **Linux** in the **Operating System** drop-down menu, and then click **Next**.

3） Allocate memory to virtual machine and then click **Next**;

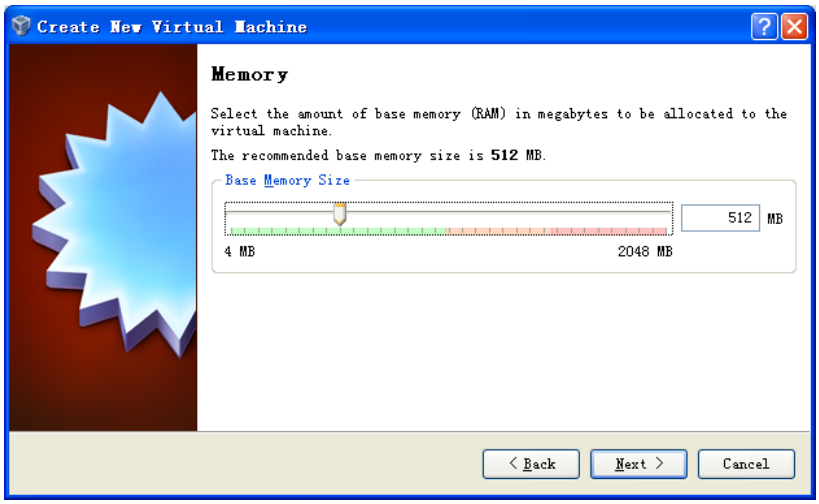


Figure 3 Memory allocation

Note:

🕮 If the memory of your PC is only 1GB or lower, please keep the default setting;

If the memory of your PC is higher than 1GB, you can allocate 1/4 or fewer to virtual machine, for example, 512MB out of 2GB memory could be allocated to virtual machine.

4） If this is the first time you install VirtualBox, please select **Create new hard disk** in the following window, and then click **Next**;

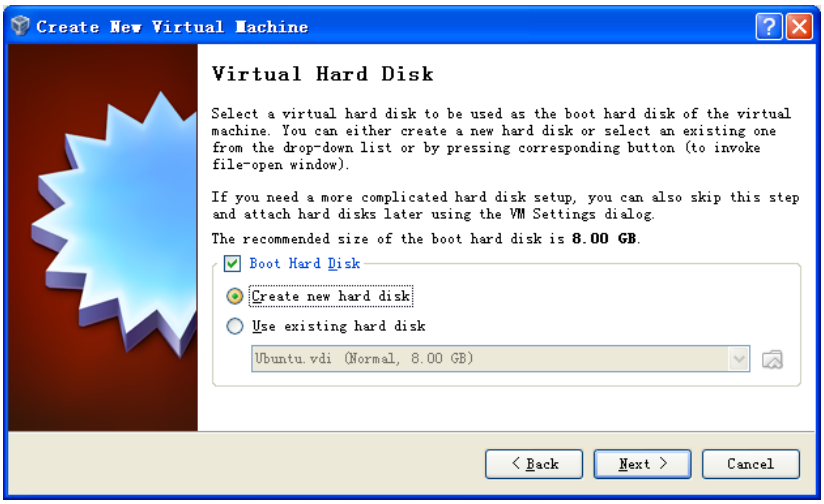


Figure 4 Create new hard disk

5） Click **Next** in the following window;

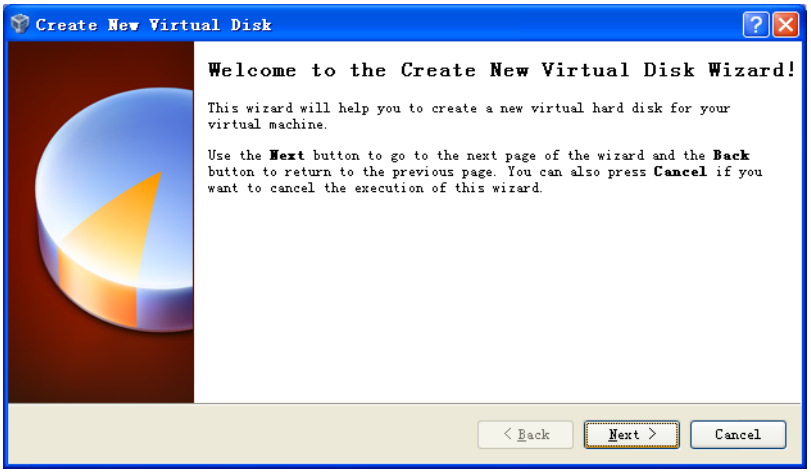


Figure 5 Wizard of new virtual disk creation

6） Selecting **Fixed-size storage** in the following window and click **Next**;

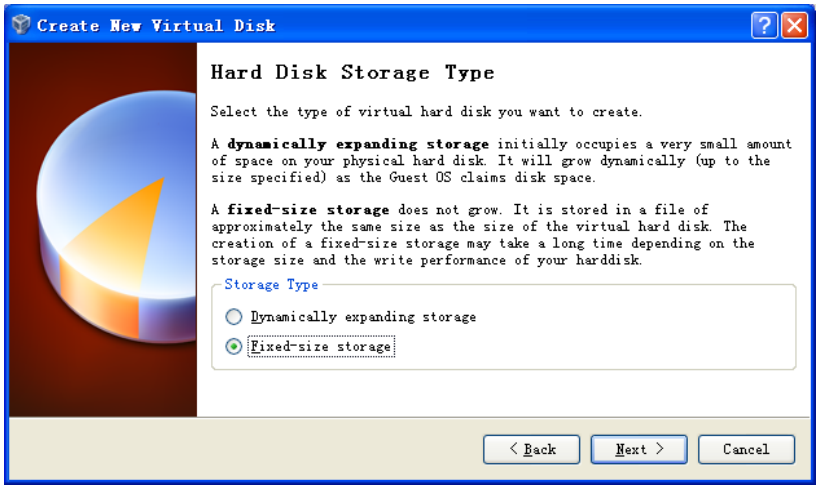


Figure 6 Select the second option

7） Define where the hard disk data is stored and the default space of the virtual disk (8G at least), and then click **Next**;

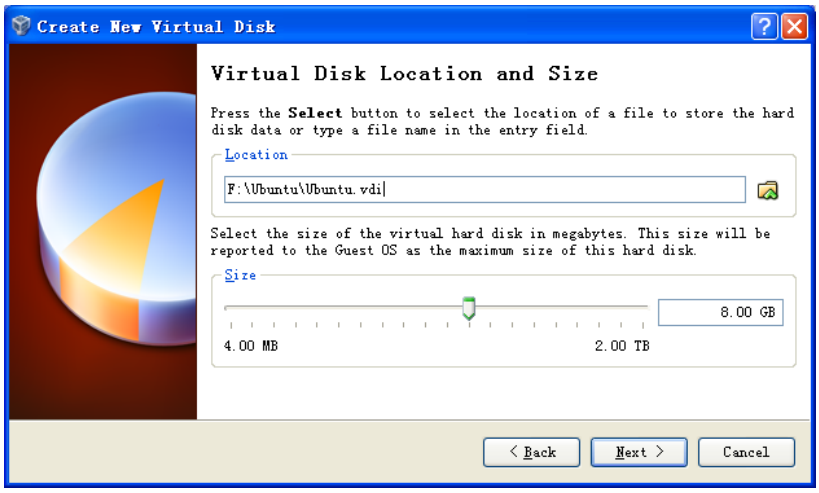


Figure 7 Virtual disk configuration

8） Click **Finish** in the following window;

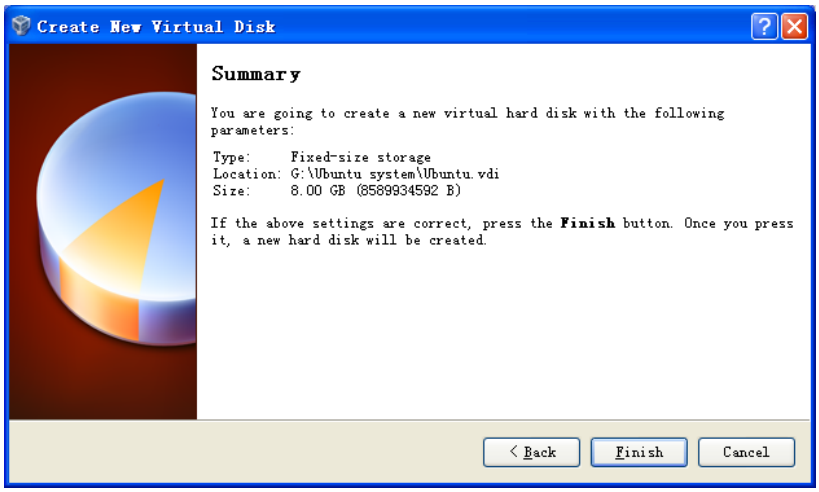


Figure 8 Virtual disk summary

9） PC is creating a new virtual disk;

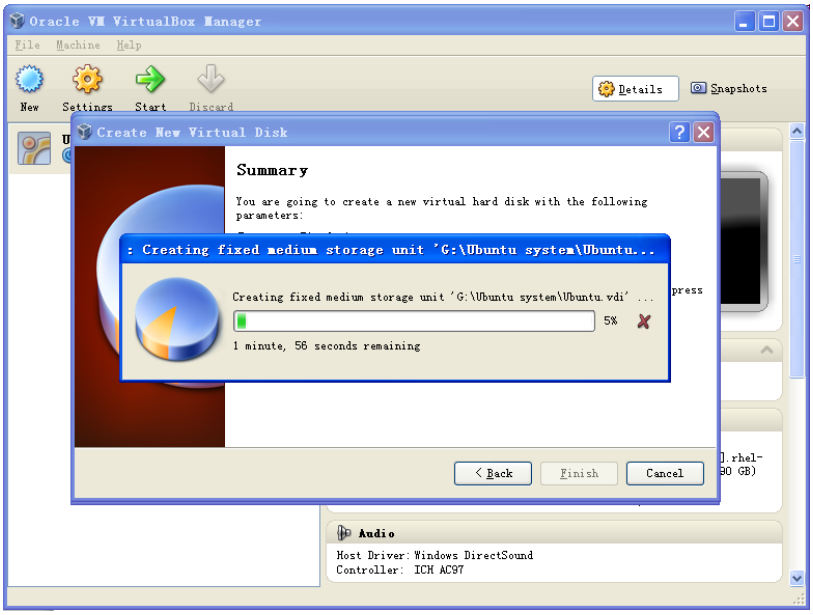


Figure 9 Virtual disk creation in process

10） A window with summary of the newly created virtual machine will be shown as below when the creation process is done. Please click **Finish** to complete the whole process.

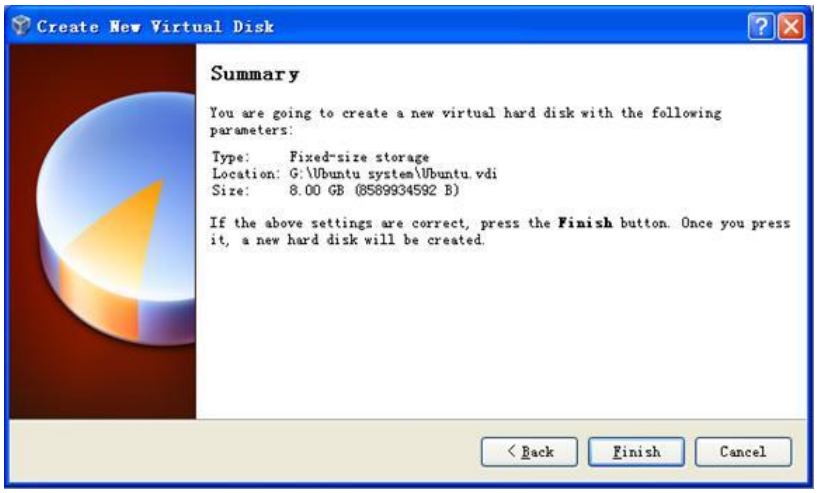


Figure 10 Virtual machine is ready

Getting Started to Install Ubuntu

After virtualBox is installed, we can start the installation of Ubuntu Linux system now. Please access http://www.Ubuntu.com/download/Ubuntu/download to download the ISO image file of Ubuntu, and then follow the steps.

1） Start VirtualBox from the **Start** menu and click **Setting** on the VirtualBox window. A **Settings** window will be shown as below;

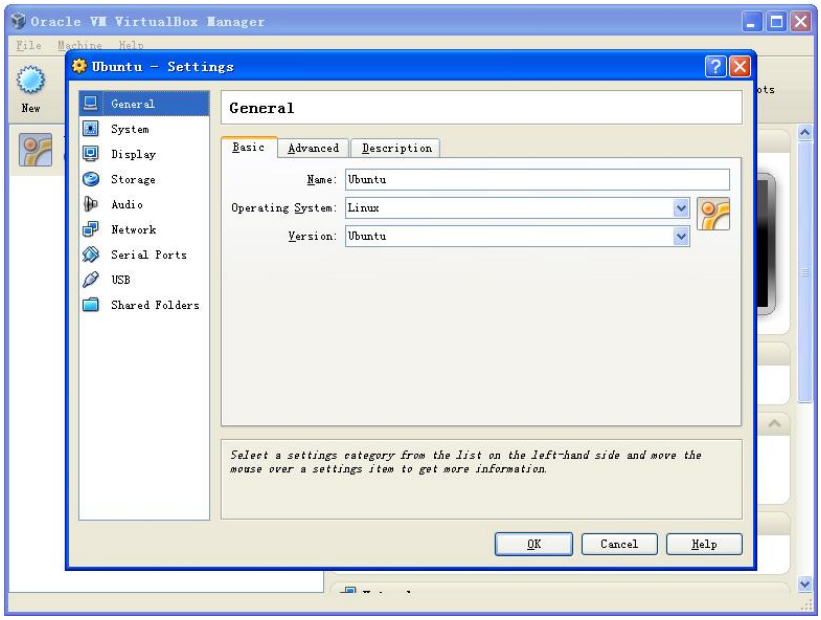


Figure 11 Setting window

2） Select **Storage** on the left in the **Setting** window and click the CD-like icon next to the option **Empty** under IDC controller in the right part of the window, and then find the ISO file you downloaded;

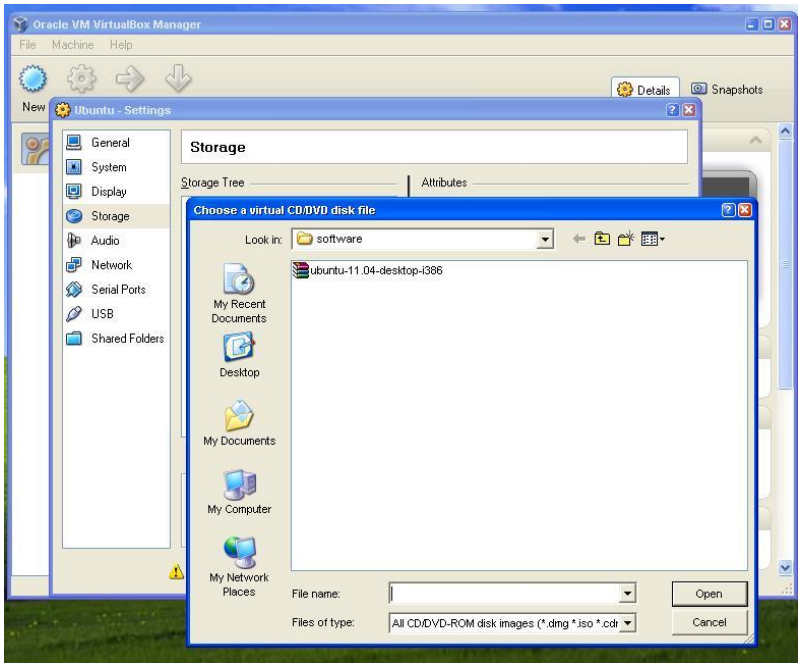


Figure 12 Find ISO file

3） Select the ISO file you added in and click **OK** as shown below;

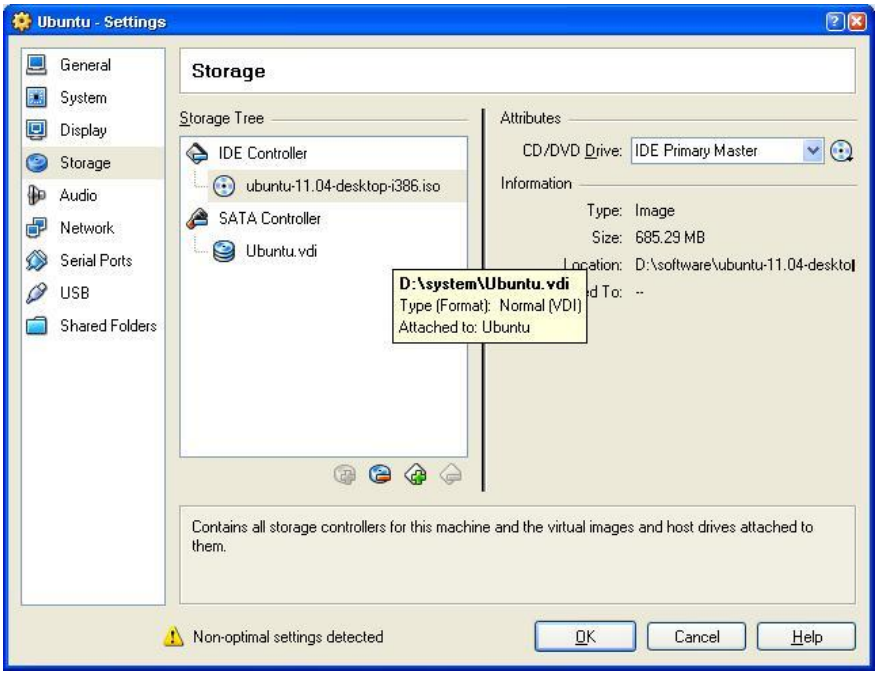


Figure 13 Select ISO file

4） Click **Start** on the VirtualBox window, the installation program of Ubuntu will be initiating as shown below;

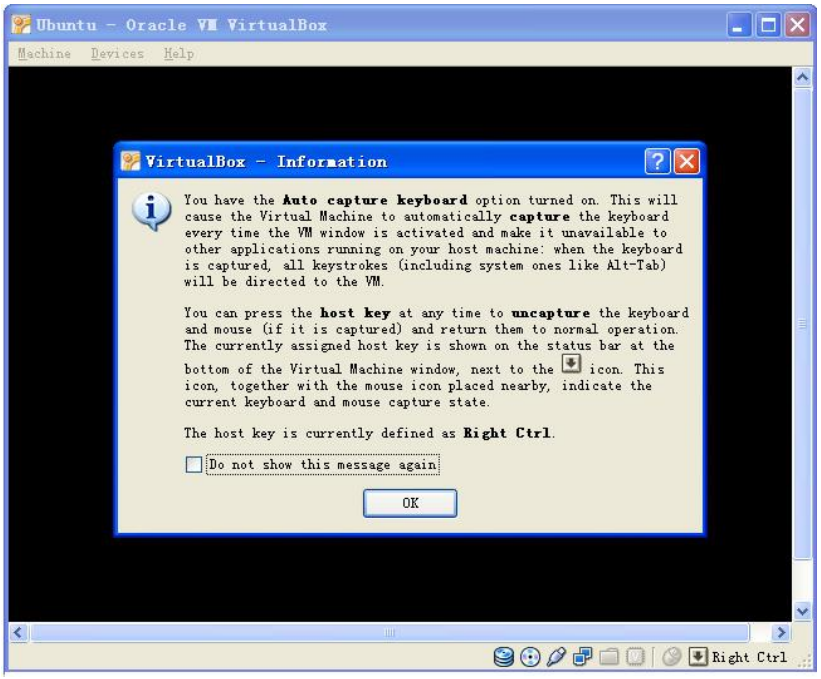


Figure 14 Ubuntu initiating window

Some prompt windows will interrupt in during the initiating process. You just need to click **OK** all the way to the end of the process.

5） Click **Install Ubuntu** to start installation when the following window appears;

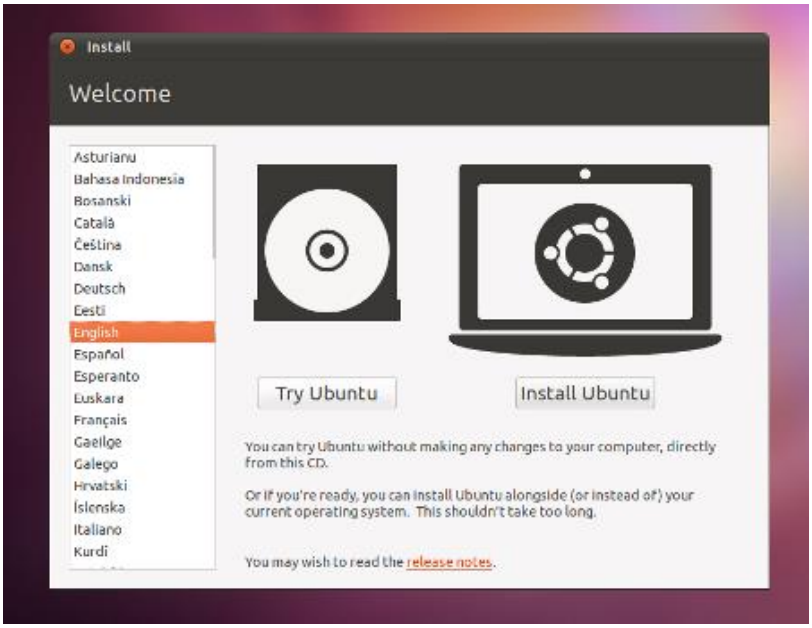


Figure 15 Ubuntu installation window

6） Click **Forward** to continue the process;

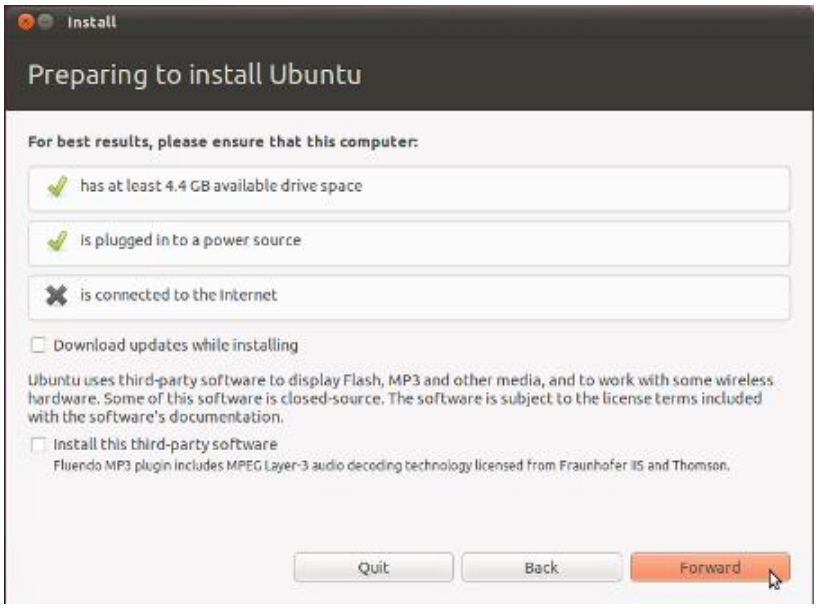


Figure 16 Information before installation

7） Select Erase disk and install Ubuntu and click Forward;

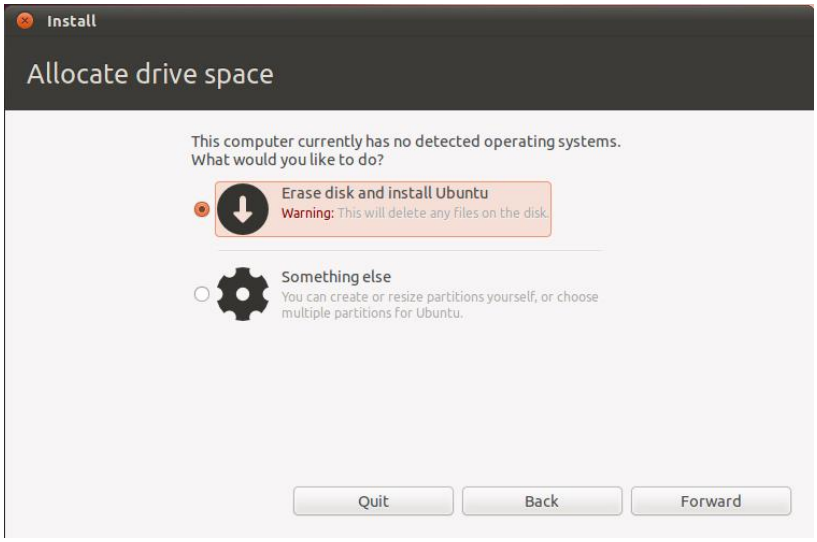


Figure 17 Options before installation

Note:

🕮 Selecting this option will not lead to any content loss on your hard drive.

8） Click **Install Now** in the following window to start installation;

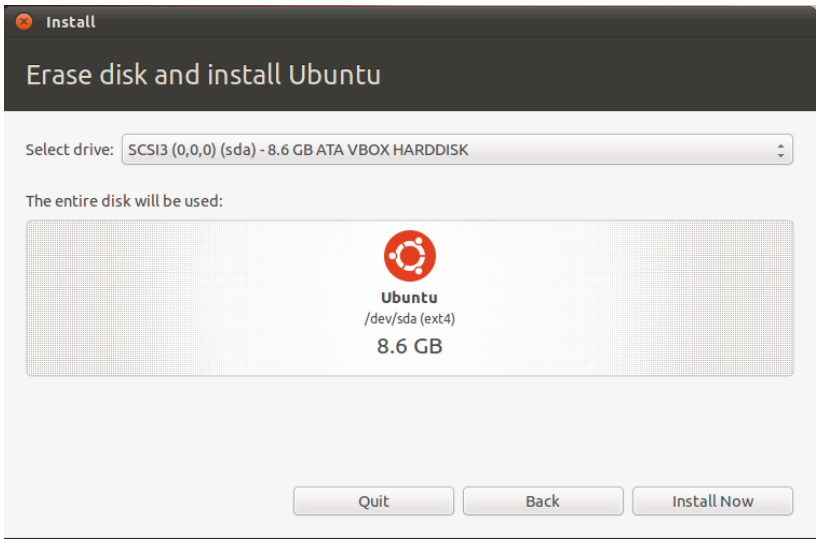


Figure 18 Confirm installation

9） Some simple questions need to be answered during the installation process. Please enter appropriate information and click **Forward**. The following window is the last question that will appear during the process;

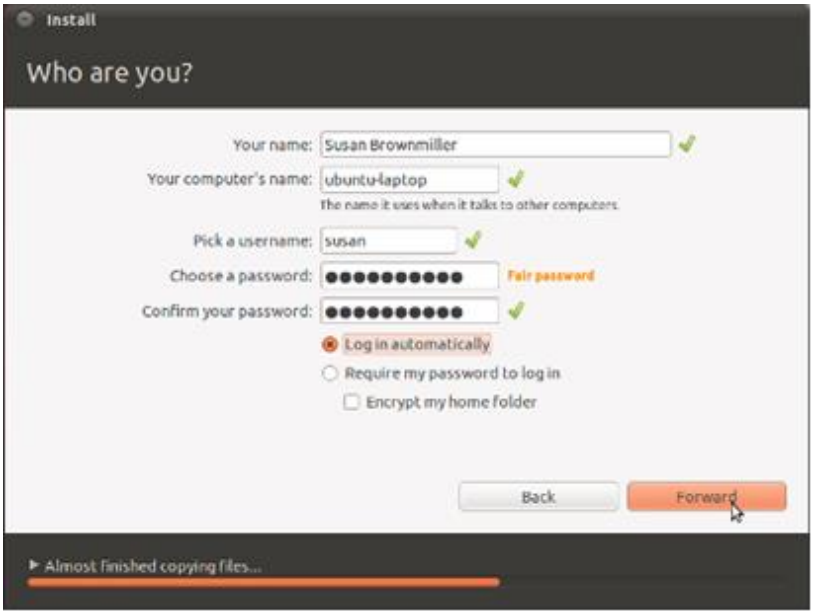


Figure 19 Enter appropriate information

After all the required information is properly entered in to the fields, select **Log in automatically** and click **Forward**.

10） The installation of Ubuntu may take 15 minutes to about 1 hour depending on your PC's performance. A prompt window will be shown as below after

installation is done. Please select **Restart Now** to restart Ubuntu system.

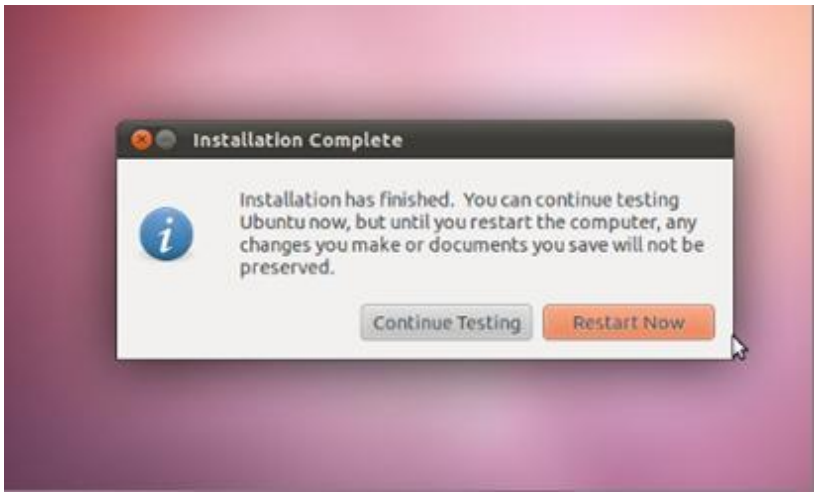


Figure 20 Restart Ubuntu

11） Ubuntu system is ready for use after restarting. Normally the ISO file shown in Figure 13 will be ejected automatically by VirtualBox after restarting Ubuntu. If it doesn't, you could eject the ISO file manually in the **Setting** window of VirtualBox. The following window shows how it looks after the ISO file is ejected.

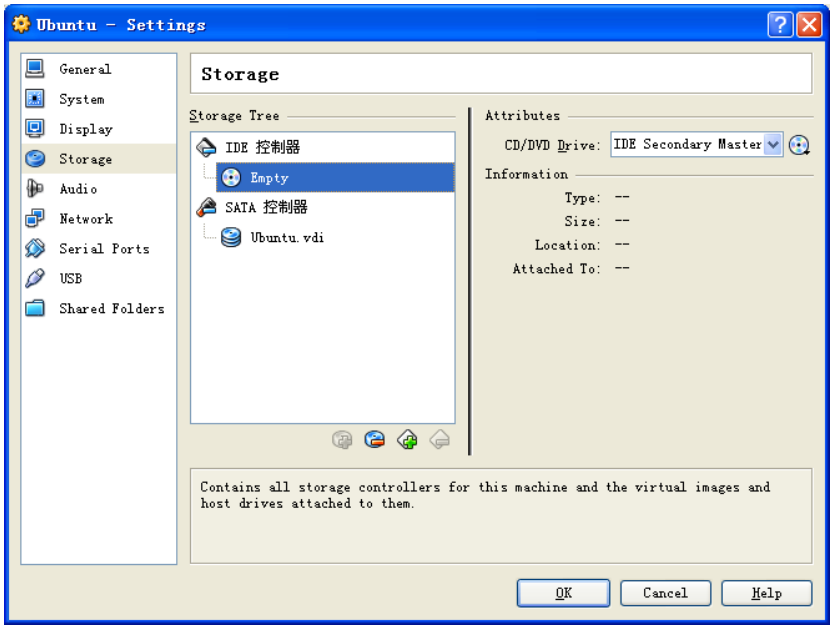


Figure 21 ISO file ejected

Appendix 2 - Installing Linux USB Ethernet/RNDIS Gadget Driver

1） Download “Associated Tools for Linux” from the “Download” tab on the page of SBC9000 at Embest’s website.

2） Connect the USB OTG port of SBC9000 to your PC with a Mini USB cable; a bubble “Installing device driver software” will appear on the system tray as shown below if you have not installed Linux USB Ethernet/RNDIS Gadget driver;

Figure 22 Searching driver

3） Click the bubble to open the following window, then click “Skip obtaining driver software from Windows Updates”;

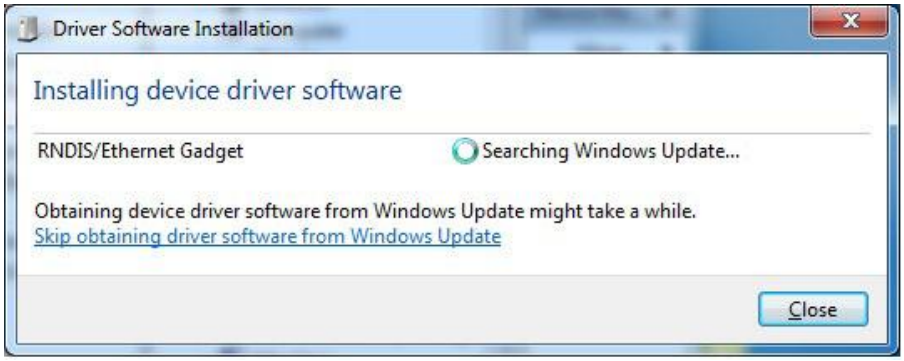


Figure 23 Install driver automatically

4） Click “Yes” in the following window;

Figure 24 Skip auto installation

5） Click “Brose my computer for driver software” in the following window;

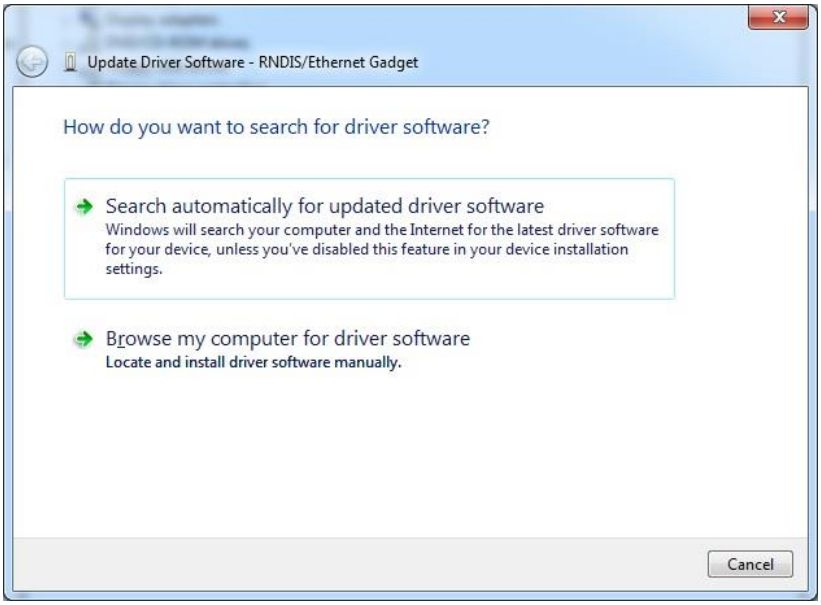


Figure 25 Find driver manually

6） Click “Browse” in the following window to specify the directory “tools\usb driver” in the “Associated Tools for Linux” you downloaded at the beginning, then click “Next” to start installation;

Figure 26 Find driver

7） The window shown below appears after installation indicates that the driver has been installed successfully.

Figure 27 Installation completes

Technical Support and Warranty

Technical Support

Embest Technology provides its product with one-year free technical support including:

- Providing software and hardware resources related to the embedded products of Embest Technology;
- Helping customers properly compile and run the source code provided by Embest Technology;
- Providing technical support service if the embedded hardware products do not function properly under the circumstances that customers operate according to the instructions in the documents provided by Embest Technology;
- Helping customers troubleshoot the products.

The following conditions will not be covered by our technical support service. We will take appropriate measures accordingly:

- Customers encounter issues related to software or hardware during their development process;
- Customers encounter issues caused by any unauthorized alter to the embedded operating system;
- Customers encounter issues related to their own applications;
- Customers encounter issues caused by any unauthorized alter to the source code provided by Embest Technology;

Warranty Conditions

1） 12-month free warranty on the PCB under normal conditions of use since the sales of the product;

2） The following conditions are not covered by free services; Embest Technology will charge accordingly:

- Customers fail to provide valid purchase vouchers or the product identification tag is damaged, unreadable, altered or inconsistent with the products.
- Products are damaged caused by operations inconsistent with the user manual;
- Products are damaged in appearance or function caused by natural disasters (flood, fire, earthquake, lightning strike or typhoon) or natural aging of components or other force majeure;
- Products are damaged in appearance or function caused by power failure, external forces, water, animals or foreign materials;
- Products malfunction caused by disassembly or alter of components by customers or, products disassembled or repaired by persons or organizations unauthorized by Embest Technology, or altered in factory specifications, or configured or expanded with the components that are not provided or recognized by Embest Technology and the resulted damage in appearance or function;
- Product failures caused by the software or system installed by customers or inappropriate settings of software or computer viruses;
- Products purchased from unauthorized sales;
- Warranty (including verbal and written) that is not made by Embest Technology and not included in the scope of our warranty should be fulfilled by the party who committed. Embest Technology has no any responsibility;

3） Within the period of warranty, the freight for sending products from customers to Embest Technology should be paid by customers; the freight from Embest to customers should be paid by us. The freight in any direction occurs after warranty period should be paid by customers.

4） Please contact technical support if there is any repair request.

Note:

📖 Embest Technology will not take any responsibility on the products sent back without the permission of the company.

Contact Information

Technical Support

Telephone Number: +86-755-25635626-872/875/897

Email Address: support@embest-tech.com

Sales Information

Telephone Number: +86-755-25635626- 863/865/866/867/868

Fax Number: +86-755-25616057

Email Address: globalsales@embest-tech.com

Company Information

Company Website: http://www.embest-tech.com
Company Address: Tower B 4/F, Shanshui Building, Nanshan Yungu Innovation Industry Park, Liuxian Ave. No. 1183, Nanshan District, Shenzhen, Guangdong, China (518055)