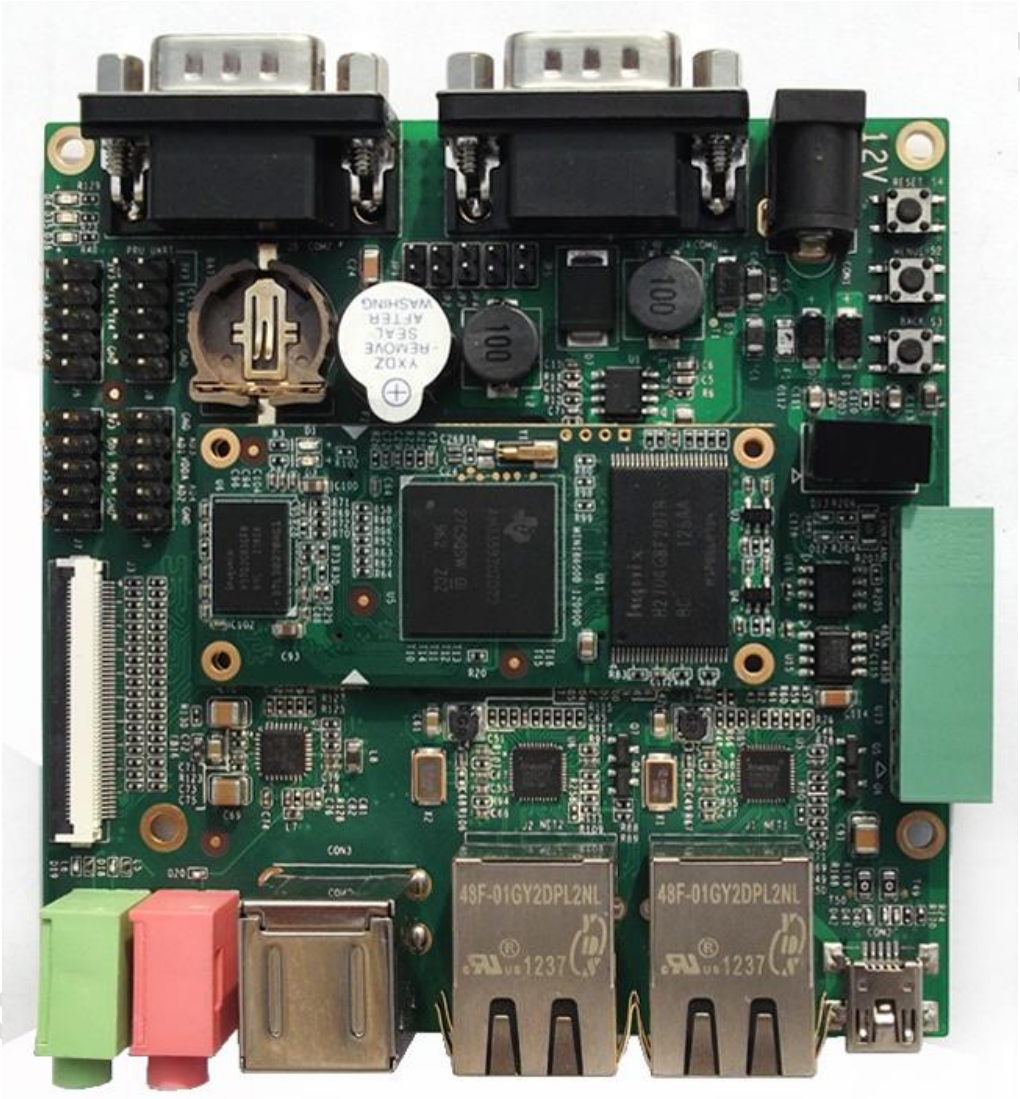
SBC8600B

Single Board Computer



User Manual

SBC8600B-UM-V2.0

July 16, 2017

Copyright Statement:

- SBC8600B 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 with the written permission issued by Embest Technology Co., Ltd.
- Microsoft, MS-DOS, Windows, Windows95, Windows98, Windows2000 and Windows Embedded Compact 7 are trademarks of Microsoft Corporation.

Revision History:

Version	Date	Description
1.0	2012-12-21	Initial Version
1.1	2014-5-1	Document Revision
2.0	2017-6-26	HW & SW Upgrade

Table of Contents

Chapter 0 ABBR

Mini86x0B: Mini8600B or Mini8610B, core board using MPU AM3358 or AM3352

SBC8600B: Extension Board of Mini86x0B

Chapter 1 Product Overview

1.1 Introduction

Mini86x0B is a small form-factor controller board based on TI's Sitara AM3354/AM3358 ARM Cortex-A8 processor, meauring only 60mm by 27mm; The tiny module integrates 2*256MBytes DDR3 SDRAM and 1*512Mbytes NAND Flash; It uses two 0.4mm pitch 40-pin board-to-board male extension connectors to bring out most hardware peripheral signals and GPIOs from the CPU.

SBC8600B is a single board computer designed by Embest, which has an extension board to carry the Mini86X0B core board. The flexible design allows the fast and easy way of realizing and upgrading the controller's capabilities. In additional to those features offered by Mini86X0B, the SBC8600B extension board has 5 serial ports (including 2 RS232 and 3 TTL), 2 USB Hosts and 1 USB OTG, 2 Ethernet ports, CAN, RS485, LCD, Touch screen, Audio, ADC and more other peripherals. The SBC8600B is a ready-to-run platform to support with Linux 4.1, Android 2.3 and WinCE 7 operating systems.

1.2 Hardware Overview

The following sections list out all the hardware features of the two parts of SBC8600B respectively.

1.2.1 Mini8600B/Mini8610B

Electric Features

- Working Temperature: 0 °C~ 70°C
- Working Humidity: 20% ~ 90%, Non-Condensing
- Dimensions: 60mm x 27mm

- Input Voltage: 3.3V

Processor

- 1GHz ARM Cortex™-A8 32-Bit RISC Microprocessor

  NEON™ SIMD Coprocessor

  32KB/32KB of L1 Instruction/Data Cache with Single-Error Detection (parity)

  256KB of L2 Cache with Error Correcting Code (ECC)
- SGX530 Graphics Engine
- Programmable Real-Time Unit Subsystem

Memories

- 512MB NAND Flash
- 2*256MB DDR3 SDRAM

Extension Interfaces and Signals Routed to Pins

- Two 0.4-pitch 40-pin DIP Interfaces
- A TFT LCD Interface (Support LCDs with 24-bpp parallel RGB interface)
- Two USB2.0 High-Speed OTG Interfaces
- Six UART Interfaces
- A SPI Interface
- Two 10/100 /1000Mbps Ethernet MAC(EMAC) with Management Data Input/Output(MDIO) Module
- A Multichannel Audio Serial Ports (McASP)
- 8-Channel 12bit ADC Interface
- Three IIC Signals
- Two 4-line SD/MMC card interfaces
- GPMC Signals

Note:
Some of the pins are multiplexed for UART、IIC、SPI、CAN. Please refer to the CPU datasheet

and schematics in the DVD-ROM for details.

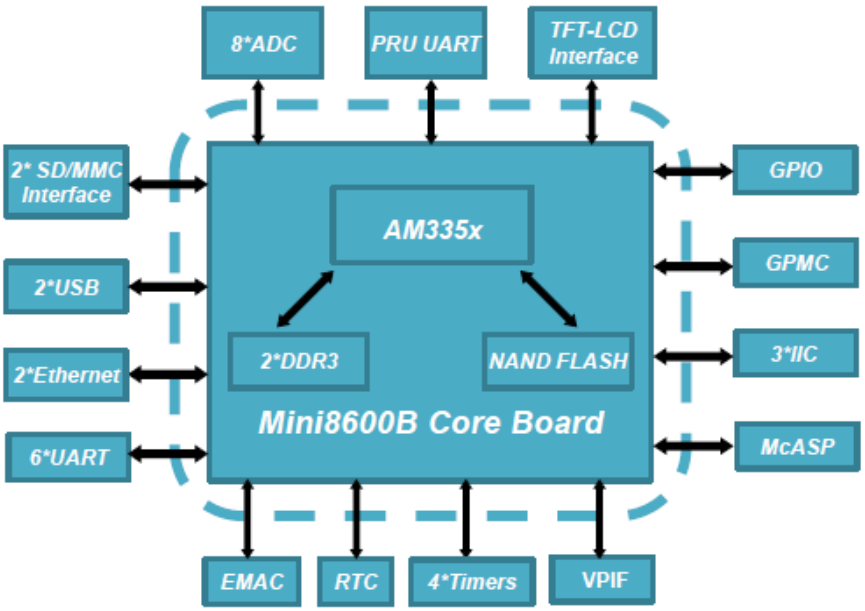


Figure 1-1 Mini86x0B Block Diagram

1.2.2 Extension Board

Electric Features

- Working Temperature: 0 °C~ 70°C
- Working Humidity: 20% ~ 90%, Non-Condensing
- Dimensions: 95m x 95m
- Input Voltage: 12V

Audio/Video Interfaces

- LCD/4-Line Resistive Touch-Screen Interface (50-pin FPC connector with 24-bit RGB output)
- An Audio Input Interface (3.5mm connector)
- An Dual-Channel Audio Output Interface (3.5mm connector)

Data Transfer Interface

- Two 10/100/1000Mbps Ethernet Interface (WinCE 7 support only one Ethernet interface)
- A CAN 2.0 Interface and a RS485 Interface (8-pin Phoenix Contact Connector)
- A USB 2.0 High-Speed OTG Ports with Integrated PHY (480Mbps, Mini USB Interface)
- Two USB 2.0 High-Speed HOST Ports with Integrated PHY (480Mbps,USB-A Interfaces)
- A TF Slot (SD/MMC compatible, 3.3V logic level）
- Serial Interfaces

  UART0, 3-Line RS232 Level, DB9 Debugging Serial Interface

  UART2, 3-Line RS232 Level, DB9 General-Purpose Serial Interface

  UART3, 3-Line TTL Level, DIP Interface

  UART4, 3-Line TTL Level, DIP Interface

  UART5, 3-Line TTL Level, DIP Interface
- GPIO Interfaces

Input Interfaces and others

- Two Customizable Buttons (MENU and BACK)
- A Reset Button
- A Buzzer
- A Power Indication LED
- Two Customizable LEDs

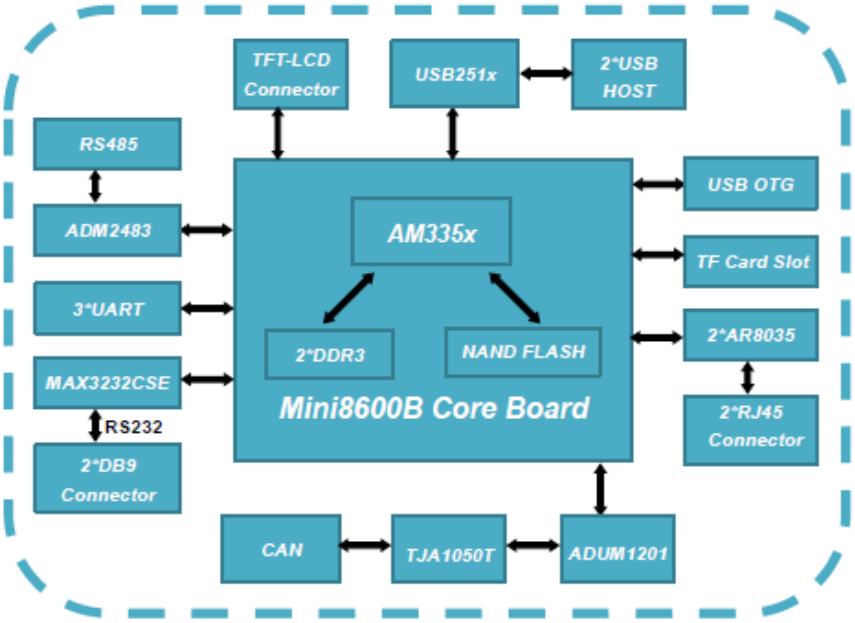


Figure 1-2 SBC8600B Block Diagram

1.3 Hardware Dimension

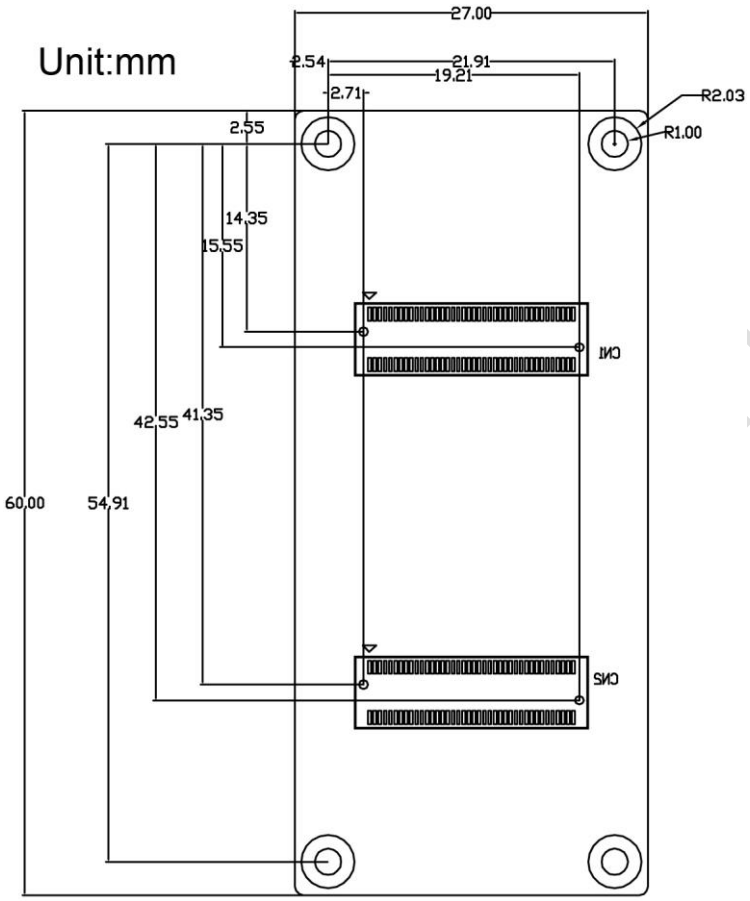


Figure 1-3 Mini86x0B Board Dimension

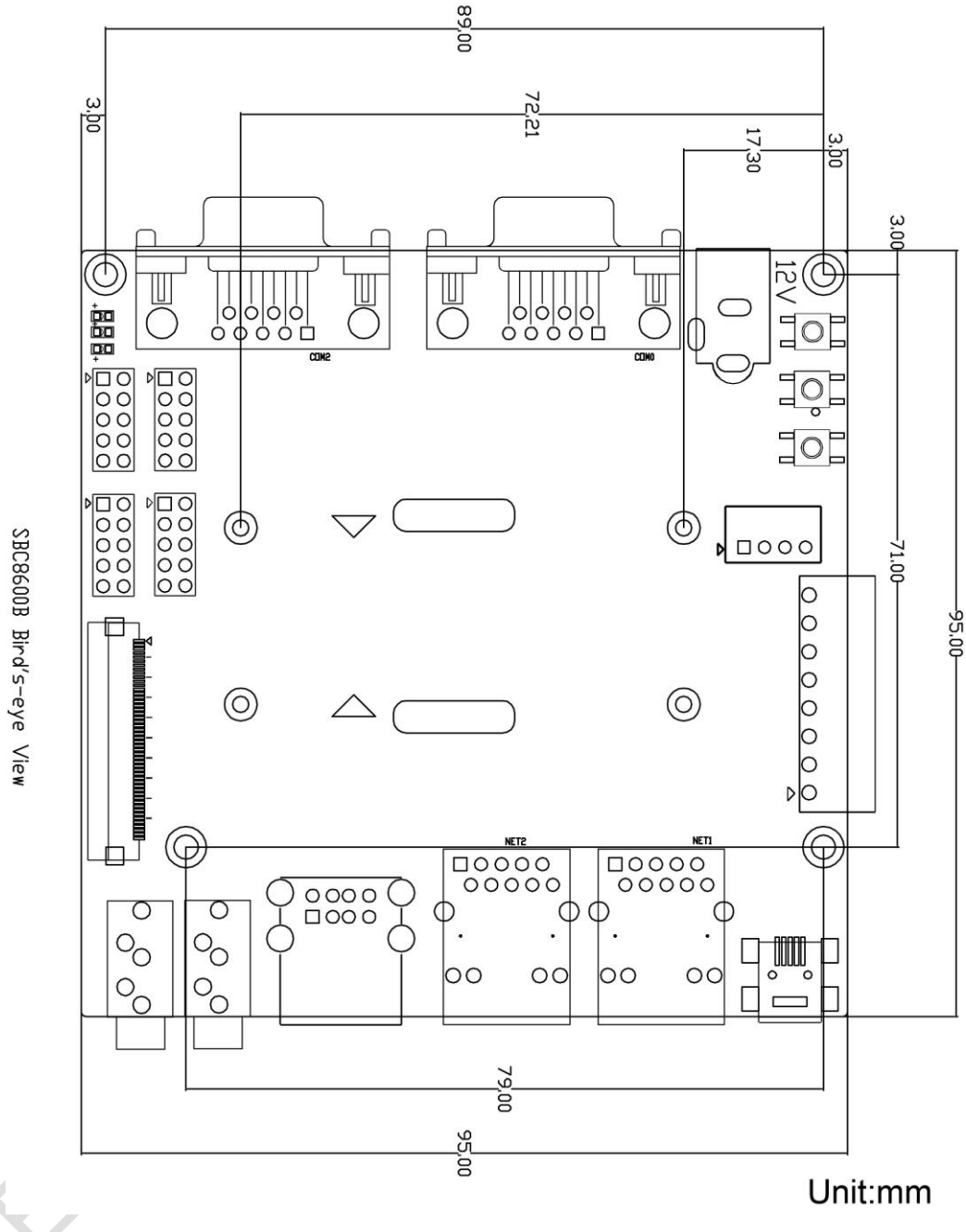


Figure 1-4 SBC8600B Extension Board Dimension

1.4 Modules Suitable for the Extension Board

Table 1-1

Names	Linux	Android	WinCE	Relevant Materials
VGA8000	NO*	YES*	YES*	Available in DVD-ROM
WF8000-U	NO*	NO	NO	Available in DVD-ROM
CAM8100-U	NO*	NO	NO	Available in DVD-ROM
CDMA8000-U	NO*	NO	NO	Download
WCDMA8000-U	NO*	NO	NO	Download
LVDS8000 Plus	NO*	YES*	YES*	Available in DVD-ROM and on website
LCD8000-97C	NO*	YES*	YES*	Available in DVD-ROM and on website

Chapter 2 Hardware System

2.1 CPU

2.1.1 Introduction to CPU

The AM335x is ARM Cortex-A8 based microprocessor, enhanced in the aspect of image, graphics processing, peripherals and industrial interface options such as Ether CAT and PROFIBUS; The device supports high-level operating systems (HLOSs) such as Linux, WinCE and Android.

The AM335x microprocessor contains following subsystems:

- Microprocessor unit (MPU) subsystem based on the ARM Cortex-A8 microprocessor.
- POWERVR SGX™ Graphics Accelerator subsystem for 3D graphics acceleration to support display and gaming effects.
- The Programmable Real-Time Unit and Industrial Communication Subsystem (PRU-ICSS) is separate from the ARM core, allowing independent operation and clocking for greater efficiency and flexibility.

2.1.2 CPU Features

Clock

AM335x has two clock inputs, OSC1 and OCC0, and two clock outputs, LCKOUT1 and LCKOUT2.

OSC1 provides RTC with a 32.768 KHz reference clock. And it is used to connect RTC_XTALIN terminal to RTC_XTALOUT terminal.

OSC0 provides reference clocks such as 19.2-MHz, 24-MHz, 25-MHz or 26-MHz for the clocks without RT function. It is also used to connect XTALIN terminal and XTALOUT terminal.

Reset

Reset is controlled by PWRONRSTn signals from CPU. The reset will be effective when active low.

General-Purpose Interfaces

There are 4 GPIO banks, each of which has 32 I/O pins, and the total pin number of GPIO can be 128 (4x32).

Programmable Real-Time Unit Subsystem

Note:
Not applicable to SBC8610B

The PRUSS of AM3358 consists of two programmable real-time units;a 12KB shared RAM with single-error detection (parity), tree 120B register bank that can be accessed by each PRU, an interrupt controller module used to process input events of the system, and the following peripherals:

- An UART with data flow control and maximum rate of 12Mbps
- Two MII Ethernet interface with support to industrial Ethernet such as EtherCAT™
- A MDIO interface
- An enhanced capture module (eCAP)

3D Graphics Engine

POWERVR® SGX graphics acceleration subsystem is used to improve 3D image processing to provide regular display and gaming effect support. The subsystem key features as below:

- Tile-Based Architecture with processing capacity up to 20 MPloy/sec
- Universal Scalable Shader Engine is a Multi-Threaded Engine Incorporating Pixel and Vertex Shader Functionality
- Advanced Shader Feature Set in Excess of Microsoft VS3.0, PS3.0 and OGL2.0
- Industry Standard API Support of Direct3D Mobile, OGL-ES 1.1 and 2.0, OpenVG 1.0, and OpenMax

2.2 Introduction to Peripherals

2.2.1 NAND Flash H27U4G8F2DTR-BC

H27U4G8F2DTR-BC is a 512MB NAND Flash used on SBC8600B; you can get more information from below access:

Disk-SBC8600B\HW design\datasheet\ NAND Flash\ H27U4G8F2DTR-BC.pdf

2.2.2 DDR H5TQ2G83CFR-H9C

H5TQ2G83DFR-H9C is a 256MB DDR3 SDRAM used on Mini86x0B. There are two H5TQ2G83DFR-H9C on Mini86x0B; you can get more information from below access:

Disk-SBC8600B\HW design\datasheet\DDR\ H5TQ2G83DFR.pdf

2.2.3 Ethernet AR8035

AR8035 is compliant with the IEEE 802.3az Energy Efficiency Ethernet Standard and the Atheros's proprietary SmartEEE standard, which allows traditional MAC/SoC devices incompatible with 802.3az to function as a complete 802.3az system.

SBC8600B can be connected to a hub with a straight-though network cable, or connected to a computer with a crossover cable, you can get more information of Ethernet chip from below access:

Disk-SBC8600B\HW design\datasheet\LAN\ AR8035.pdf

2.2.4 MAX3232

MAX3232 is used to convert TTL levels into RS232 levels so that the board can communicate with RS232 interfaces of PCs.

SBC8600B uses UART0 as debugging serial interface. The default voltage of UART0 is 1.8V, which needs to be boosted up to 3.3V for satisfying the external use. You can get more information about this device from below access:

SBC8600B\HW design\datasheet\ Serial\MAX3232CSE.pdf

2.3 Hardware Interfaces

2.3.1 Mini86x0B

CN1 Interface

Figure 2-1 Mini86x0B CN1 Interface

Table 2-1 CN1 Interface

CN1		
PIN	**Signal**	**Function**
1	GND	GND
2	VDDS_RTC	Supply voltage for RTC
3	CLK_OUT1	Clock out1
4	CLK_OUT2	Clock out2
5	MMC0_DAT0	MMC0 data bus
6	MMC0_DAT1	MMC0 data bus
7	MMC0_DAT2	MMC0 data bus
8	GLOBLE_RESETN	SYS_RESET IN/ OUTPUT
9	MMC0_DAT3	MMC0 data bus
10	AM335X_PWRON_RESETN	CPU PWRON Reset
11	GND	GND
12	GND	GND
13	AM355X_PRU_UART0_CTS	PRU UART0 Clear To Send
14	AM355X_PRU_UART0_RX	PRU UART0 receive data
15	AM355X_PRU_UART0_RTS	PRU UART0 request to send
16	AM355X_PRU_UART0_TX	PRU UART0 transmit data
17	AM355X_UART0_RX	UART0 receive data
18	AM355X_UART3_RX	UART3 receive data
19	AM355X_UART0_TX	UART0 transmit data
20	AM355X_UART3_TX	UART3 transmit data
21	AM355X_CAN0_RX	CAN0 receive data
22	AM355X_I2C0_SDA	I2C0 master serial data
23	AM355X_CAN0_TX	CAN0 transmit data
24	AM355X_I2C0_SCL	I2C0 master serial clock
25	AM355X_UART4_RX	UART4 receive data
26	AM355X_UART1_RX	UART1 receive data
27	AM355X_UART4_TX	UART4 transmit data
28	AM355X_UART1_TX	UART1 transmit data
29	GND	GND
30	GND	GND
31	MII1_COL	MII1 collision detect
32	AM355X_USB0_DRVVBUS	USB0 controller VBUS control output

CN1		
PIN	**Signal**	**Function**
33	MII1_TX_CLK	MII1 transmit clock
34	AM355X_USB1_DRVVBUS	USB1 controller VBUS control output
35	MII1_TX_EN	MII1 transmit enable
36	MII1_REF_CLK	MII1 reference clock
37	MII1_TXD3	MII1 transmit data
38	MII1_CRS	MII1 carrier sense
39	MII1_TXD2	MII1 transmit data
40	MII1_RX_ER	MII1 receive data error
41	MII1_TXD1	MII1 transmit data
42	MII1_RX_DV	MII1 receive data valid
43	MII1_TXD0	MII1 transmit data
44	MII1_RX_CLK	MII1 receive clock
45	MII_MDIO	MII MDIO DATA
46	MII1_RXD3	MII1 receive data
47	MII_MDC	MII MDIO CLK
48	MII1_RXD2	MII1 receive data
49	GND	GND
50	MII1_RXD1	MII1 receive data
51	AM355X_USB0_DM	USB0 DM-
52	MII1_RXD0	MII1 receive data
53	AM355X_USB0_DP	USB0 DP
54	MMC0_CMD	MMC0 Command Signal
55	GND	GND
56	USB0_VBUS	USB0 bus voltage
57	AM355X_USB1_DM	USB1 data-
58	AM355X_USB1_ID	USB1 ID
59	AM355X_USB1_DP	USB1 data+
60	AM355X_USB0_ID	USB0 ID
61	GND	GND
62	USB1_VBUS	USB1 bus voltage
63	GPMC_A0	GPMC address
64	GPMC_A7	GPMC address
65	GPMC_A5	GPMC address
66	GPMC_A11	GPMC address

CN1		
PIN	**Signal**	**Function**
67	GPMC_A4	GPMC address
68	GPMC_A10	GPMC address
69	GPMC_A3	GPMC address
70	GPMC_A9	GPMC address
71	GPMC_A2	GPMC address
72	GPMC_A8	GPMC address
73	GPMC_A6	GPMC address
74	GPMC_A1	GPMC address
75	GND	GND
76	GND	GND
77	VDD_3V3	Power
78	VDD_3V3	Power
79	VDD_3V3	Power
80	VDD_3V3	Power

CN2 Interface

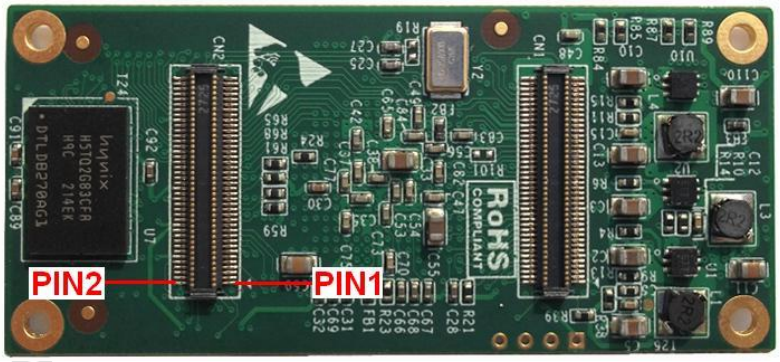


Figure 2-2 Mini86x0B CN2 Interface

Table 2-2 CN2 Interface

CN2		
PIN	**MODE1**	**Function**
1	GND	GND
2	GND	GND
3	MCASP0_AHCLKX	MCASP0 transmit master clock
4	MCASP0_ACLKX	MCASP0 transmit bit clock

CN2		
PIN	**MODE1**	**Function**
5	MCASP0_FSX	MCASP0 transmit frame sync
6	MCASP0_AXR0	MCASP0 serial data(I/O)
7	MCASP0_AHCLKR	MCASP0 receiver master clock
8	MMC0_CLK	MMC0 clock
9	MCASP0_FSR	MCASP0 receive frame sync
10	MCASP0_AXR1	MCASP0 serial data(I/O)
11	GND	GND
12	GND	GND
13	VDDA_ADC	Supply voltage range for ADC
14	AM355X_ADC0	ADC0
15	AM355X_ADC1	ADC1
16	AM355X_ADC2	ADC2
17	AM355X_ADC3	ADC3
18	AM355X_ADC4	ADC4
19	AM355X_ADC5	ADC5
20	AM355X_ADC6	ADC6
21	AM355X_ADC7	ADC7
22	GND_ADC	GND ADC
23	GND	GND
24	GND	GND
25	LCD_DATA1	LCD data bus
26	LCD_DATA12	LCD data bus
27	LCD_DATA0	LCD data bus
28	LCD_DATA10	LCD data bus
29	LCD_DATA5	LCD data bus
30	LCD_DATA13	LCD data bus
31	LCD_DATA4	LCD data bus
32	LCD_DATA11	LCD data bus
33	LCD_DATA6	LCD data bus
34	LCD_DATA14	LCD data bus
35	LCD_DATA8	LCD data bus
36	LCD_VSYNC	LCD vertical sync
37	GND	GND
38	GND	GND
39	LCD_DATA9	LCD data bus

CN2		
PIN	**MODE1**	**Function**
40	LCD_PCLK	LCD pixel clock
41	LCD_DATA15	LCD data bus
42	GPMC_AD11	GPMC address & data
43	LCD_DATA3	LCD data bus
44	GPMC_AD15	GPMC address & data
45	LCD_DATA2	LCD data bus
46	GPMC_AD14	GPMC address & data
47	LCD_DATA7	LCD data bus
48	GPMC_WAIT0	GPMC wait0
49	LCD_HSYNC	LCD horizontal sync
50	GPMC_BEN1	GPMC byte enable 1
51	GND	GND
52	GND	GND
53	LCD_EN	LCD AC bias enable chip select
54	GPMC_WPN	GPMC write protect
55	GPMC_AD13	GPMC address & data
56	GPMC_CSN3	GPMC chip select
57	GPMC_AD9	GPMC address & data
58	GPMC_CSN2	GPMC chip select
59	GPMC_AD10	GPMC address & data
60	GPMC_CLK	GPMC clock
61	GPMC_AD8	GPMC address & data
62	GPMC_AD6	GPMC address & data
63	GPMC_AD12	GPMC address & data
64	GND	GND
65	GND	GND
66	GPMC_CSN1	GPMC chip select1
67	GPMC_ADVN_ALE	GPMC address valid/address latch enable
68	GPMC_AD5	GPMC address & data
69	GPMC_BEN0_CLE	GPMC byte enable 0/Command latch enable
70	GPMC_AD4	GPMC address & data
71	GPMC_OEN_REN	GPMC output /read enable
72	GPMC_AD1	GPMC address & data

CN2		
PIN	**MODE1**	**Function**
73	GPMC_AD2	GPMC address & data
74	GPMC_AD0	GPMC address & data
75	GPMC_AD3	GPMC address & data
76	GPMC_CSN0	GPMC chip select0
77	GPMC_AD7	GPMC address & data
78	GPMC_WEN	GPMC write enable
79	GND	GND
80	GND	GND

2.3.2 Extension Board

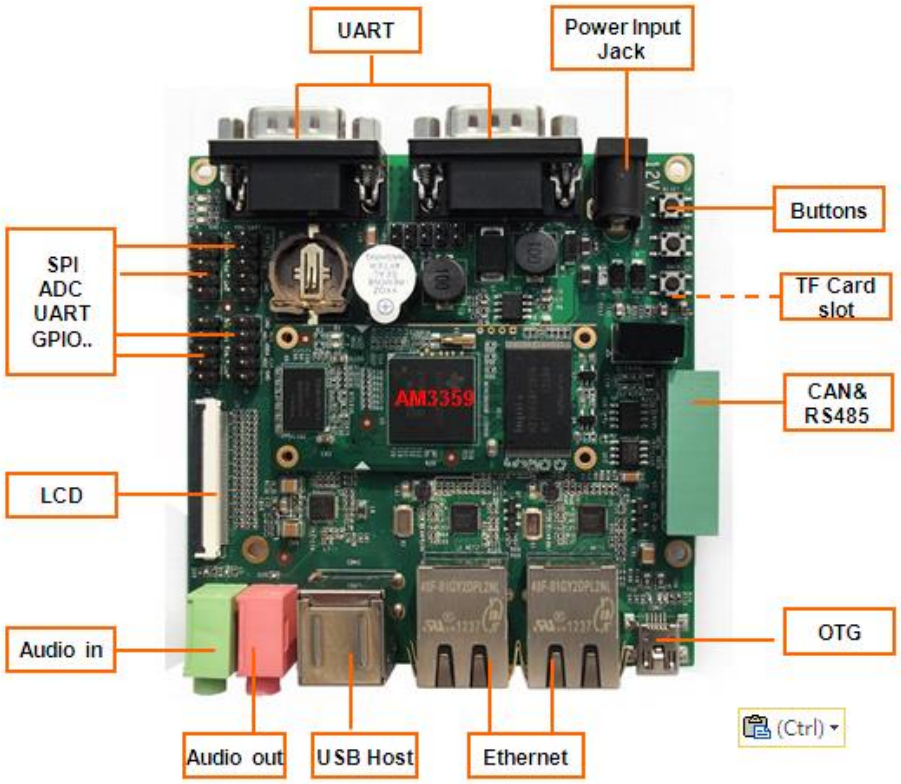


Figure 2-3 Extension board interfaces

The interface is on the bottom of the board

The interface is on the top of the board

1. Power Jack

Table 2-3 Power Jack

CON1		
Pin	**Signal**	**Description**
1	GND	GND
2	+12V	Power supply (+12V)
3	NC	NC

2. TFT_LCD Interface

Table 2-4 TFT_LCD Interface

J3		
Pin	**Signal**	**Description**
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
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

J3		
Pin	**Signal**	**Description**
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	NC	NC
39	NC	NC
40	NC	NC
41	NC	NC
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

Note:
Please do NOT disconnect the LCD flat cable when the board is powered on.

3. Audio Output Interface

Table 2-5 Audio Output Interface

HEADPHONE1		
Pin	**Signal**	**Description**
1	GND	GND
2	NC	NC
3	Right	Right output
4	NC	NC
5	Left	Left output

4. Audio Input Interface

Table 2-6 Audio Input Interface

MIC1		
Pin	**Signal**	**Description**
1	GND	GND
2	NC	NC
3	MIC IN	Input
4	NC	NC
5	MIC IN	Input

5. USB HOST Interface

Table 2-7 USB HOST Interface

CON3		
Pin	**Signal**	**Description**
1	VBUSA	+5V
2	DA-	USB Data-
3	DA+	USB Data+
4	GNDA	GND

6. USB OTG Interface

Table 2-8 USB OTG Interface

CON2		
Pin	**Signal**	**Description**
1	VB	+5V
2	D-	USB Data-
3	D+	USB Data+
4	ID	USB ID
5	G1	GND

7. TF Card Interface

Table 2-9 TF Card Interface

TF1		
Pin	**Signal**	**Description**
1	DAT2	Card data 2
2	CD/DAT3	Card data 3
3	CMD	Command Signal
4	VDD	VDD
5	CLOCK	Clock
6	VSS	VSS
7	DAT0	Card data 0
8	DAT1	Card data 1
9	CD	Card detect

8. LAN Interface

Table 2-10 LAN Interface

J1,J2		
Pin	**Signal**	**Description**
1	TD1+	Transmit Data1+
2	TD1-	Transmit Data1-
3	TD2+	Transmit Data2+
4	TD2-	Transmit Data2-
5	TCT	Transmit common terminal
6	RCT	Receive common terminal

J1,J2		
Pin	**Signal**	**Description**
7	RD1+	Receive Data1+
8	RD1-	Receive Data1-
9	RD2+	Receive Data2+
10	RD2-	Receive Data2-
11	GRLA	+2.5V
12	GRLC	LINK active LED
13	YELC	100M linked LED
14	YELA	+2.5V

9. Serial Interface

Table 2-11 Serial Interface

J4(UART0), J5(UART2)		
Pin	**Signal**	**Description**
1	NC	NC
2	RXD	Receive data
3	TXD	Transmit data
4	NC	NC
5	GND	GND
6	NC	NC
7	RTS	Request To Send
8	CTS	Clear To Send
9	NC	NC

10. CAN&RS485 Interface

Table 2-12 CAN&RS485 Interface

U22		
Pin	**Signal**	**Description**
1	+12V	+12V
2	GND	GND
3	GND2	Isolated GND
4	485B1	485B
5	485A1	485A
6	GND1	Isolated GND
7	CANL	CANL

8	CANH	CANH

11. ADC Interface

Table 2-13 ADC Interface

J9		
Pin	**Signal**	**Description**
1	GND	GND
2	GND	GND
3	ADC_CH1	ADC1
4	ADC_CH3	ADC3
5	VDDA_ADC	Power
6	VDDA_ADC	Power
7	ADC_CH2	ADC2
8	ADC_CH4	ADC4
9	GND	GND
10	GND	GND

12. SPI Interface

Table 2-14 SPI Interface

J8		
Pin	**Signal**	**Description**
1	+3.3V	3.3V
2	+3.3V	3.3V
3	SPI0_D1	SPI0 data1
4	SPI0_CLK	SPI0 clock
5	SPI0_CS0	SPI enable0
6	SPI0_D0	SPI data0
7	GND	GND
8	GND	GND
9	GND	GND
10	GND	GND

13. Extension Interface

Table 2-15 Extension Interface

J6

Pin	Signal	Description
1	VIO_3V3	+3.3V
2	VIO_3V3	+3.3V
3	UART3_TX_3V3	UART3 Transit data 3.3V level
4	UART4_TX_3V3	UART4 Transit data 3.3V level
5	UART3_RX_3V3	UART3 receive data 3.3V level
6	UART4_RX_3V3	UART4 receive data 3.3V level
7	GND	GND
8	GND	GND
9	GND	GND
10	GND	GND

Table 2-16 Extension Interface

J7		
Pin	**Signal**	**Description**
1	VIO_3V3	+3.3V
2	VIO_3V3	+3.3V
3	UART5_TX_3V3	UART5 Transit data 3.3V level
4	GPIO0_9	GPIO
5	UART5_RX_3V3	UART5 receive data 3.3V level
6	GPIO2_0	GPIO
7	GND	GND
8	GND	GND
9	GND	GND
10	GND	GND

14. Buttons

Table 2-17 Buttons

S1-3		
Pin	**Signal**	**Description**
S2	MENU	System menu key
S3	BACK	System back key
S4	Reset	System Reset key

15. LED

Table 2-18 LED

LEDs		
LED	**Definition**	**Description**
1	D4	Power Indicator
2	D35	User Custom LED
3	D36	User Custom LED

Chapter 3 Linux Operating System

This chapter will give you a general map of the Linux software resources contained in the DVD-ROM provided along with the product, as well as detailed introduction to the process of Linux system development, drivers development, system update, functionality tests and application development examples.

Note:
It is recommended referring to Appendix for details of Ubuntu Linux installation and learning about embedded Linux development technology before you get started.

3.1 Software Resources

The DVR-ROM provided along with the board contains demos, application examples, Linux source code and tools, helping you develop Linux applications and systems easily and quickly based on SBC8600B.

3.1.1 Locations of Resources

You can find software resources such as programs and codes contained in the DVD-ROM according to the information showed in the table below;

Table 3-1 Programs and Code

Categories	Locations
Demos	CD\android
	CD\linux\demo\tisdk
Applications	CD\linux\example\libsocketcan-0.0.9.tar.bz2
	CD\linux\example\uart_test.tar.bz2

	CD\linux\example\can_test.tar.bz2
Source Code	CD\linux\source\linux-4.1.tar.xz
	CD\linux\source\rootfs.tar.xz
	CD\linux\source\u-boot-2015.07.tar.xz
Tools	CD\linux\tools
Precompiled Images	CD\linux\image\

3.1.2 BSP

The following table lists types and formats of the files contained in BSP;

Table 3-2 Files in BSP

Categories	Names	Note	Formats
BIOS	spl	NAND	Source Code
		MMC/SD	Source Code
		FAT	Source Code
	u-boot	NAND	Source Code
		MMC/SD	Source Code
		FAT	Source Code
		NET	Source Code
Kernel	Linux-4.1	Supports ROM/CRAM/EXT4/FAT/NFS/ JFFS2/UBIFS and various file systems	Source Code
Device Driver	Serial	Serials driver	Source Code
	RTC	Hardware clock driver	Source Code
	NET	10/100M/1000M Ethernet driver	Source Code
	CAN	CAN bus driver	Source Code
	Flash	NAND flash driver (supports NAND boot)	Source Code
	SPI	SPI driver	Source code
	I2C	I2C driver	Source code
	LCD	TFT LCD driver	Source Code
	Touch screen	4-line touch-screen controller driver	Source Code
	ADC	4 channels ADC input	Source Code

	MMC/SD	MMC/SD controller driver	Source Code
	USB OTG	USB OTG 2.0 driver	Source Code
	Audio	Audio driver (supports audio recording and playback)	Source Code
	Keypad	GPIO keypad driver	Source Code
	LED	User custom Led driver	Source Code
Rootfs	Debian 8	Simplified, No GUI	Compressed Package
Demo	Android	Android 4.0.3 system	Source Code
	TISDK	TISDK system	Source Code

3.2 Structure of Embedded Linux System

SBC8600B has a Linux-4.1 system in on-board NAND flash by default. This system supports 4.3-inch touch-screen and consists of spl（MLO）, u-boot, kernel and rootfs. The following figure shows the structure of embedded Linux system.

Figure 3-1 Structure of Embedded Linux System

1) spl is the first level bootstrap program. After the system starts up, the ROM inside the CPU will copy spl to internal RAM and perform its routine work. Its main function is to initialize the CPU, copy u-boot into the memory and let u-boot lead the booting process;

2) u-boot is the second-level boot loader. It is used to interact with users, update images and boot the kernel;

3) The kernel used in this document is Linux 4.1 and has customizable for for SBC8600B;

4) Rootfs adopts open-source system ubifs. It is small in capacity and powerful, very suitable for embedded systems;

5) User area store the data specific to customer, need to be formatted before first using.

User area format instruction:

retriving device name:

```
root@arm:~# cat /proc/mtd
dev:    size    erasesize  name
mtd0: 00020000 00020000 "NAND.SPL"
mtd1: 00020000 00020000 "NAND.SPL.backup1"
mtd2: 00020000 00020000 "NAND.SPL.backup2"
mtd3: 00020000 00020000 "NAND.SPL.backup3"
mtd4: 00040000 00020000 "NAND.u-boot-spl-os"
mtd5: 00100000 00020000 "NAND.u-boot"
mtd6: 00020000 00020000 "NAND.u-boot-env"
mtd7: 00020000 00020000 "NAND.u-boot-env.backup1"
mtd8: 00400000 00020000 "NAND.logo"
mtd9: 00800000 00020000 "NAND.kernel"
mtd10: 19000000 00020000 "NAND.file-system"
mtd11: 06200000 00020000 "NAND.user"
```

User area device is /dev/mtd11

fomatting:

```
root@arm:~# flash_erase /dev/mtd11 0 0
root@arm:~# ubiattach /dev/ubi_ctrl -m 11
root@arm:~# ubimkvol /dev/ubi1 -N User –m
```

mounting：

ubiattach no need to do twice or more times after system bootup.

root@arm:~# **ubiattach /dev/ubi_ctrl -m 11**

root@arm:~# **mount -t ubifs /dev/ubi1_0 /mnt**

unmounting：

root@arm:~# **umount /mnt**

3.3 Building Development Environment

Before the software development based on SBC8600B, users have to establish a Linux cross development environment on PC. This section will take Ubuntu operating system as an example to introduce how to establish a cross development environment.

It's strongly recommended to install below packages for a new installed Ubuntu system before continue.

sudo apt-get update;

sudo apt-get install -y xz-utils ncurses-dev autoconf libtool automake texinfo bison flex libc6:i386 libncurses5:i386 libstdc++6:i386

Note:

- Each instruction has been put a bullets "•" before it to prevent confusion caused by the long instructions that occupy more than one line in the context.
- Please note the SPACES within each instruction; Missing of any SPACE will cause failure when executing instructions.

3.3.1 Installing Cross Compilation Tools

Insert the DVD-ROM in your PC's DVD-ROM drive, Ubuntu system will automatically mount it under /media/cdrom. Execute the following instructions in the terminal window of Ubuntu to uncompress cross-compiling tools from /media/cdrom/linux/tools to $HOME;

- **mkdir $HOME/tools**

- **cd /media/cdrom/linux/tools**
- **tar xvf gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux.tar.xz -C $HOME/tools**

3.3.2 Addition of environment variables

After all above tools are installed, it is necessary to use the following commands to add them in the temporary environment variables:

export PATH=$HOME/tools/arm-2009q1/bin:$HOME/tools/arm-eabi-4.4.3/bin:$HOME/tools:$PATH

Note:

- The instructions can be added in the .bashrc file located at the user directory, so that the addition of environment variables will be loaded automatically when the system is booting up;
- If you want to check the path, please use the instruction **echo $PATH**
- For Android system, a complete development environment needs additional installations and configurations apart from the steps listed above. Please visit http://source.android.com/source/initializing.html for more information

3.4 Prepare the Source Code

This section will introduce two ways to get the source code.

3.4.1 Get the Source Code from DVD

Execute the following instructions to uncompress source code from linux/source of the DVD-ROM to Ubuntu system;

- **mkdir $HOME/work**
- **cd $HOME/work**
  - **tar xvf /media/cdrom/linux/source/u-boot-2015.07.tar.xz**
  - **tar xvf /media/cdrom/linux/source/linux-4.1.tar.xz**
  - **mkdir rootfs**
  - **sudo tar xvf /media/cdrom/linux/source/rootfs.tar.xz -C rootfs**

- **tar xvf /media/cdrom/android/source/TI-Android-ICS-4.0.3-DevKit-EVM-SK-3.0.1.tar.bz2**

When the above steps are finished, the directories u-boot-2015.07, Linux-4.1, rootfs and TI-Android-ICS-4.0.3-DevKit-EVM-SK-3.0.1 will be created under current directory.

Note:

Please make sure the uncompressed source code is saved under the directory specified in the above instructions, or errors might occur in compilation process.

3.4.2 Get the source code by git and repo

1） Get bootloader source code by the following commands:

- **$ cd ~**
  - **$ git clone -b u-boot-2015.07 https://github.com/embest-tech/u-boot-am33x.git**
  - **$ cd u-boot-am33x/**

View current branch:

- **$ git branch**

```
* u-boot-2015.07
```

List all branches:

- **$ git branch -a**

```
* u-boot-2015.07
  remotes/origin/AM335XPSP_04.06.00.03
  remotes/origin/AM335XPSP_04.06.00.05
  remotes/origin/AM335XPSP_04.06.00.06
  remotes/origin/AM335XPSP_04.06.00.07
  remotes/origin/AM335XPSP_04.06.00.08
  remotes/origin/HEAD -> origin/master
  remotes/origin/TIOP_AM335XPSP_04.06.00.08
  remotes/origin/add-musb-to-AM335XPSP_04.06.00.03
  remotes/origin/amsdk-05.04.01.00
  remotes/origin/amsdk-05.06.00.00
```

```
remotes/origin/int_am335xpsp_04.06.00.01-v2011.09-for-sdk-05.03.00.00
remotes/origin/int_am335xpsp_04.06.00.01-v2011.09-for-sdk-05.03.01.00
remotes/origin/int_am335xpsp_04.06.00.01-v2011.09-for-sdk-05.04.00.00
remotes/origin/u-boot-2015.07
```

2） Get the linux kernel source code by the following instructions:

- **$ cd ~**
- **$ git clone -b linux-4.1 https://github.com/embest-tech/linux-am33x.git**
-
- **$ cd linux-am33x**

List all branches:

- **$ git branch –a**

```
* linux-4.1
```

Note:

📖 The Linux source codes of the CD are got from git, we will update the git website as soon as the source code update.

3.5 Compilation

3.5.1 Compiling Boot Loader

SBC8600B can boot up from TF card or NAND Flash, if the jumper JP5 is disconnect, the board will boot from nand flash, otherwise it will boot from TF card.

Please execute the following instructions to compile boot loader;

- **cd u-boot-2015.07**
- **make distclean**
- **make am335x_sbc8600_config**
- **make**

After all the instructions are executed, you can find booting images named MLO and u-boot.img under current directory.

3.5.2 Compiling Kernel

Execute the following instructions to compile kernel;

- **cd linux-4.1**
- **make distclean**
- **make embest_ti_sbc8600_defconfig**
- **make uImage**

After all the instructions are executed, you can find a kernel image named uImage under arch/arm/boot.

3.5.3 Generation Filesystem

Currently SBC8600B Linux provide two format filesystem: Ramdisk and UBI, the Ramdisk is used when the board boot from TF card, the UBI is used when the board boot from NAND Flash

3.6 Linux System Customization

In order to satisfy different requirements of customers, designers commonly need to make some custom modification based on the default configuration of Linux kernel. This chapter will introduce the process of system customization by showing you some examples.

3.6.1 Entering Configuration Menu

A default configuration file is provided in the factory kernel source codes:

Linux-4.1/arch/arm/configs/embest_ti_sbc8600_defconfig

Please execute the following instructions to enter the configuration menu:

- **cd linux-4.1**
- **cp arch/arm/configs/embest_ti_sbc8600_defconfig .config**
- **make menuconfig**

Note:

If an error occurs when command 'make menuconfig' is executed, you might need to install 'ncurse' in the Ubuntu system; 'ncurse' is a character graphic library required to generate configuration menu; please enter the following instruction to install the library:

sudo apt-get install ncurses-dev

3.6.2 Menu Options

Configure options according to customization requirements after entering configuration menu, for example, access **Device Drivers** > **USB support** > **USB Gadget Support** > **USB Gadget Drivers** as shown below;

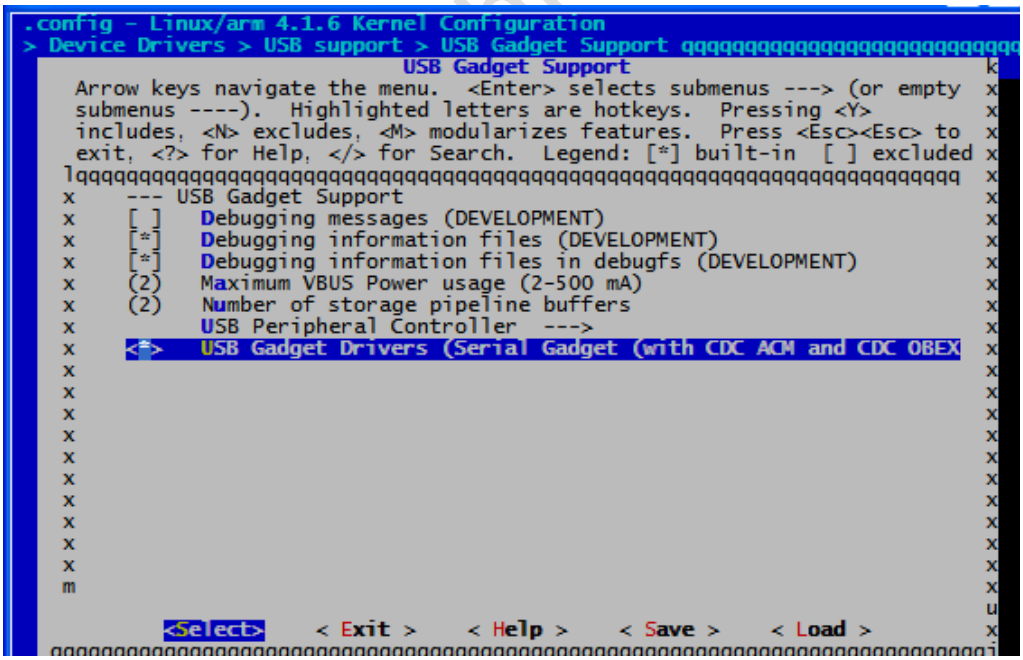


Figure 3-2 USB Gadget Drivers

Set USB Gadget Drivers（Serial Gadget.....） to **M**, and then exit and save changes.

3.6.3 Compiling Kernel

Please execute the following instructions to recompile kernel;

- **make dtbs zImage**

After all the instructions are executed, you can find a kernel image named **zImage**.

3.6.4 Testing Serial Gadget

Connect SBC8600B miniUSB port and PC USB port, the Gadget Serial v2.4 hardware device will be found in Windows Device Manager.
Install device driver CD\linux\tools\linux-cdc-acm.inf for the device above.

Run serial tool (such as Hyper Terminal) on PC, open COM port specific to Gadget Serial device. Then access /dev/ttyGS0 on SBC8600 will communicate with PC terminal.

3.7 Introduction to Drivers

This section will introduce all kinds of drivers required by a Linux system including NAND Flash driver, SD/MMC driver, display subsystem driver, video acquisition driver and audio input/output driver.

The table below shows the access path to find all the drivers;

Table 3-3 Locations of Source Code in BS

Categories	Names	Descriptions	Locations
BIOS	spl	NAND	drivers/mtd/nand/omap_gpmc.c
		MMC/SD	drivers/mmc/omap_hsmmc.c
		FAT	fs/
	u-boot	NAND	drivers/mtd/nand/omap_gpmc.c
		MMC/SD	drivers/mmc/omap_hsmmc.c

		FAT	fs/
		NET	drivers/net/cpsw.c
Kernel	Linux 4.1	ROM/CRAM/EXT4 /FAT/NFS/JFFS2/UBIFS filesystem	fs/
Devices	Serial	Serial driver	drivers/tty/serial/8250/8250_omap.c
	RTC	Hardware clock driver	drivers/rtc/rtc-omap.c
	NET	10/100M/1000M Ethernet driver	drivers/net/ethernet/ti/ti_cpsw.c
	CAN	CAN bus driver	drivers/net/can/c_can/c_can_platform.c
	Flash	NAND Flash driver (supporting NAND boot-up)	drivers/mtd/nand/omap2.c
	SPI	SPI driver	drivers/spi/spi-omap2-mcspi.c
	LCD	TFT LCD driver	drivers/gpu/drm/tilcdc/tilcdc_drv.c driver/video/of_display_timing.c
	Touch screen	4-wire touch screen controller driver	drivers/input/touchscreen/ti_tscadc.c
	ADC	4 channels ADC input	drivers/iio/adc/ti_am335x_adc.c
	MMC/SD	MMC/SD controller dirver	drivers/mmc/host/omap_hsmmc.c
	USB	USB controller dirver	drivers/usb/musb/musb_am335x.c
	Audio	Audio driver (supporting recording/playback)	sound/soc/codecs/sgtl5000.c
	Keypad	GPIO keypad driver	drivers/input/keyboard/gpio_keys.c
	LED	User LED driver	drivers/leds/leds-gpio.c

3.7.2 NAND

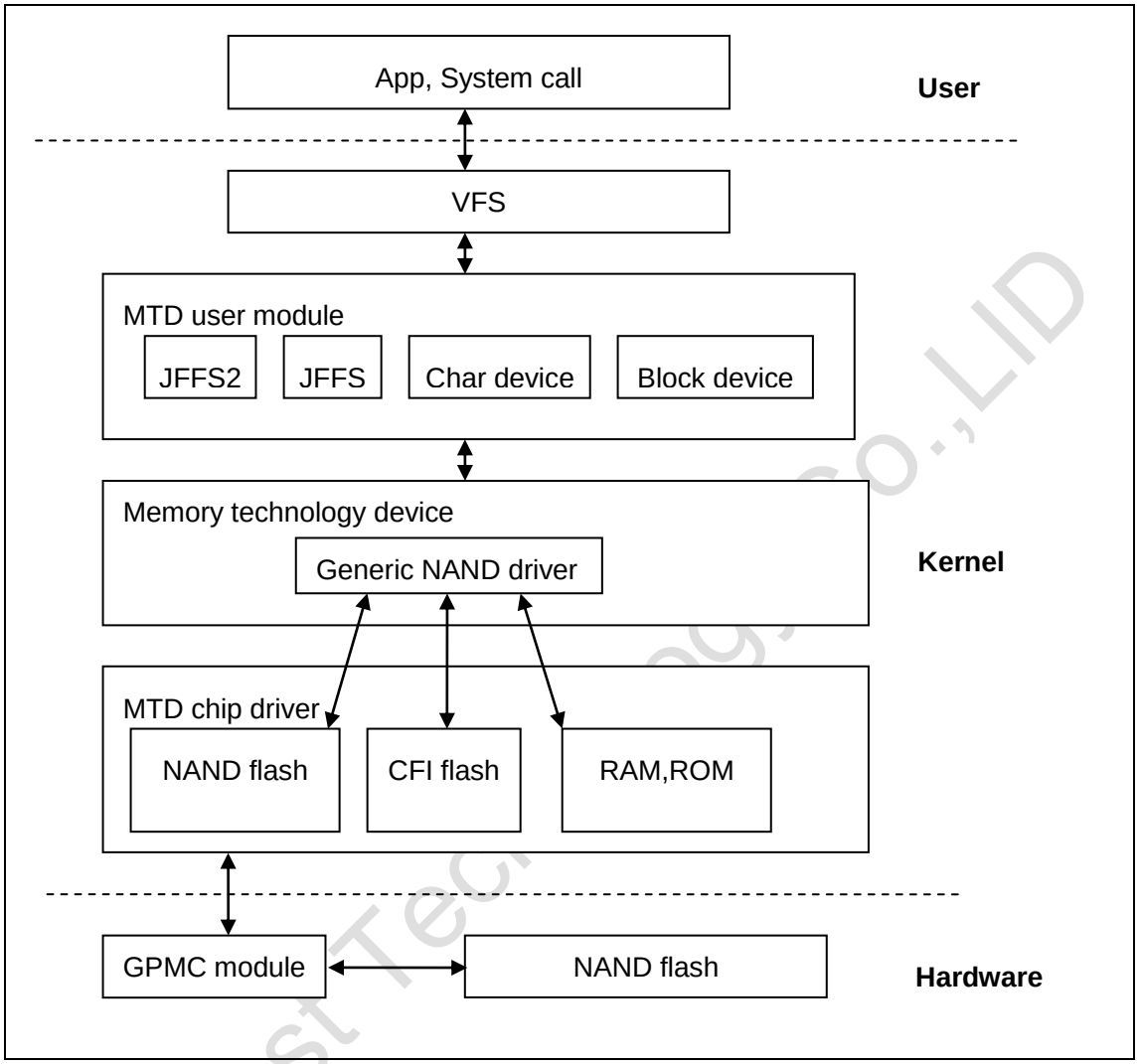


Figure 3-3 Modular structure for NAND

The solid-state memory used in embedded systems is typically a flash; it is NAND Flash in this system.

NAND Flash is used as a block device, on which the file system is established; interaction between user and NAND Flash is mainly realized by a specific file system. In order to realize compatibility with different Flash memories, an MTD subsystem is used to implement management between the file system and the specific flash driver.

Therefore, users need to access NAND Flash through the following process:

User->System Call->VFS->Block Device Driver->MTD->NAND Flash Driver->NAND Flash.

Drivers and relevant documents:

Linux-4.1/drivers/mtd/nand/

Linux-4.1/drivers/mtd/nand/omap2.c

3.7.3 SD/MMC

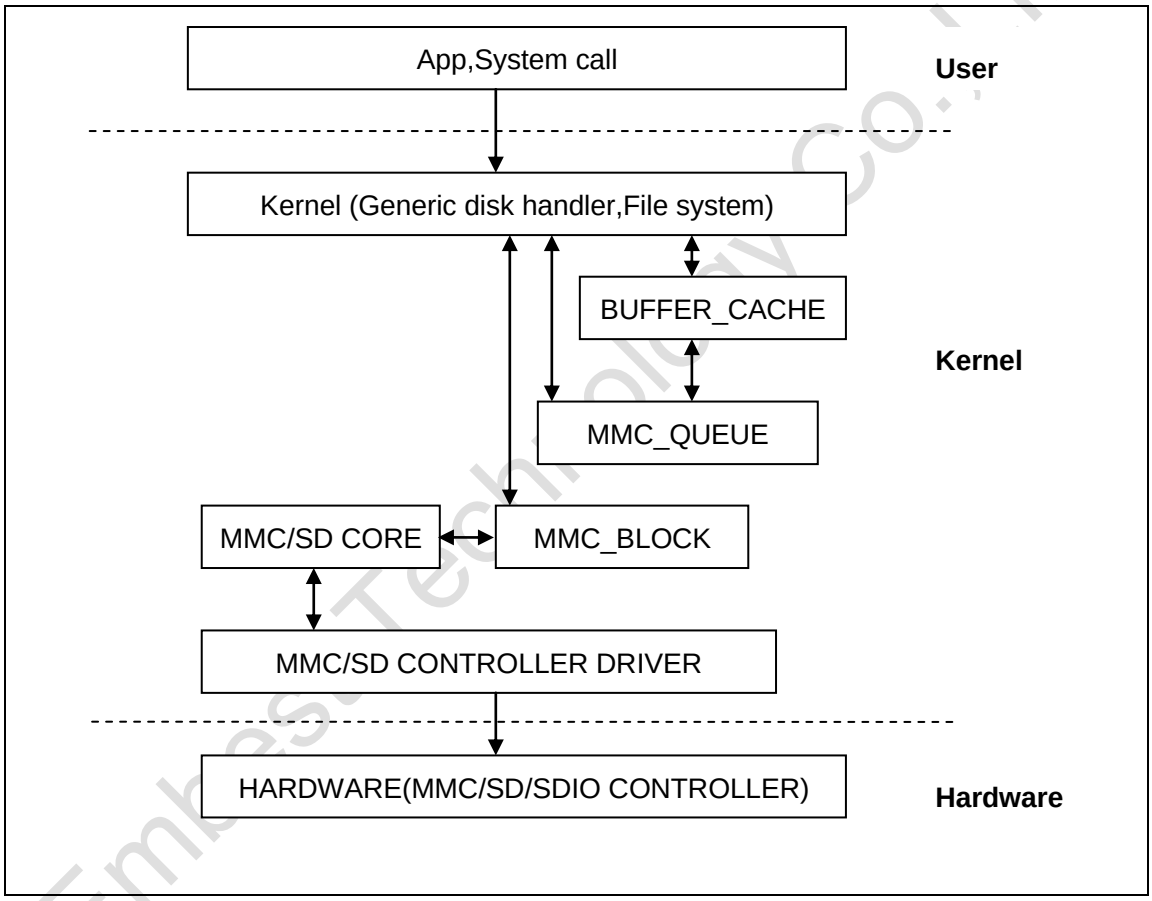


Figure 3-4 Modular structure for SD/MMC

SD/MMC card drivers in Linux mainly include SD/MMC core, mmc_block, mmc_queue and SD/MMC driver:

1） SD/MMC core realizes the codes unrelated to structure in the SD/MMC card operation.

2） mmc_block realizes driver structure when SD/MMC card is used as a block device.

3） mmc_queue realizes management of request queue.

4） SD/MMC driver realizes specific controller driver.

Drivers and relevant documents:

Linux-4.1/drivers/mmc/

Linux-4.1/drivers/mmc/host/omap_hsmmc.c

3.7.4 LCDC

The LCD controller (LCDC) of AM335x is the latest version integrated in OMAP-L138 SoC which has differences as follows comparing with OMAP-L138.

Different interrupt configuration and status register

2048*2048 Higher display resolution of up to 2048*2048

24-bit active TFT grating per pixel

So Linux LCD driver can be used to improve the LCD_VERSION2 code. By reading PID register, the update of LCDC version can be found.

Drivers and relevant documents:

Linux-4.1/drivers/video/

Linux-4.1/drivers/gpu/drm/tilcdc/tilcdc_panel.c

Linux-4.1/drivers/video/of_display_timing.c

3.7.5 Audio in/out

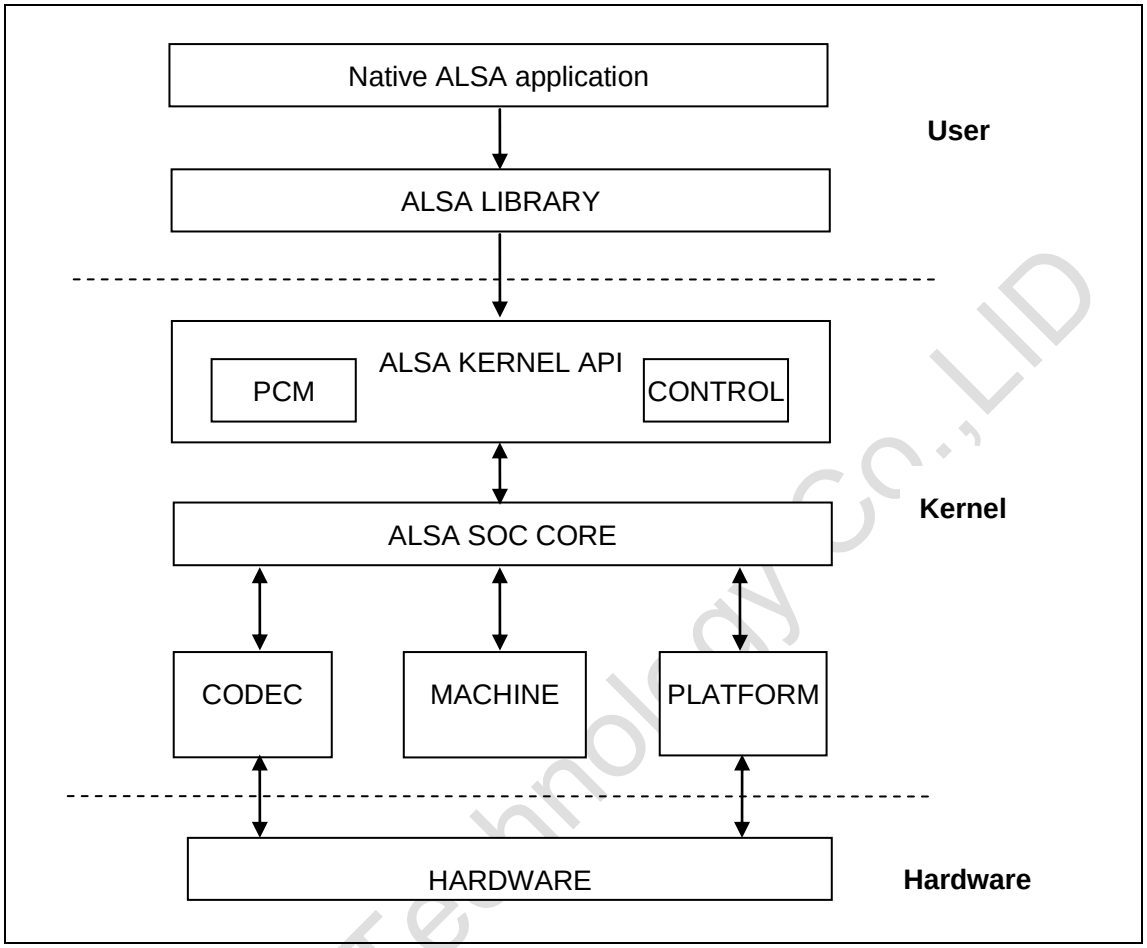


Figure 3-5 Modular structure for Audio

ASoC embedded audio system basically consists of three components:

1) Codec driver: The codec driver is platform independent and contains audio controls, audio interface capabilities, codec dapm definition and codec IO functions.

2) Platform driver: It contains the audio dma engine and audio interface drivers (e.g. I2S, AC97, PCM) of that platform.

3) Machine driver: The machine driver handles any machine specific controls and audio events i.e. turning on an amp at start of playback.

Drivers and relevant documents:

Linux-4.1/sound/soc/

Linux-4.1/sound/soc/davinci/davinci-evm.c

Linux-4.1/sound/soc/codecs/sgtl5000.c

3.8 Driver Development

3.8.1 GPIO_keys Driver

Device Definition

Linux-4.1/arch/arm/boot/dts/embest-SBC-SBC8600.dts

define gpio0.20 as “menu” key, return value as “KEY_F1”, triggered by low level; gpio2.1 as “back” key, return value as “KEY_ESC”, triggered by low level

```
gpio_keys {
    compatible = "gpio-keys";
    pinctrl-names = "default";
    pinctrl-0 = <&button_pins>;

    key@0 {
        label = "MENU";
        linux,code = <KEY_F1>;
        gpios = <&gpio0 20 GPIO_ACTIVE_LOW>;
        gpio-key,wakeup;
    };

    key@1 {
        label = "BACK";
        linux,code = <KEY_ESC>;
        gpios = <&gpio2 1 GPIO_ACTIVE_LOW>;
        gpio-key,wakeup;
    };
};
```

GPIO pinmux Configuration

Define the GPIO0.20 and GPIO2.1 as MODE7 (GPIO mode) and AM33XX_PIN_INPUT (configuration input).

Linux-4.1/arch/arm/boot/dts/embest-SBC-SBC8600.dts

```
        button_pins: pinmux_button_pins {
            pinctrl-single,pins = <
                0x1B4 (PIN_INPUT_PULLUP | MUX_MODE7)      /*
xdma_event_intr1.gpio0_20 */
                0x08C (PIN_INPUT_PULLUP | MUX_MODE7)      /* gpmc_clk.gpio2_1
*/
            >;
        };
```

Driver Design

Linux-4.1/drivers/input/keyboard/gpio_keys.c

a) Call platform_driver_register to register gpio_keys driver

```
static struct platform_driver gpio_keys_device_driver = {
        .probe          = gpio_keys_probe,
        .remove         = gpio_keys_remove,
        .driver         = {
                .name   = "gpio-keys",
                .pm     = &gpio_keys_pm_ops,
                  .of_match_table = gpio_keys_of_match,
        }
};

static int __init gpio_keys_init(void)
{
        return platform_driver_register(&gpio_keys_device_driver);
}

static void __exit gpio_keys_exit(void)
{
        platform_driver_unregister(&gpio_keys_device_driver);
}

late_initcall(gpio_keys_init);
module_exit(gpio_keys_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Phil Blundell <pb@handhelds.org>");
MODULE_DESCRIPTION("Keyboard driver for GPIOs");
MODULE_ALIAS("platform:gpio-keys");
```

b) Call input_register_device to register input driver

```
static int __devinit gpio_keys_probe(struct platform_device *pdev)
{
        if (!pdata) {
                pdata = gpio_keys_get_devtree_pdata(dev);
                if (IS_ERR(pdata))
                        return PTR_ERR(pdata);
        }
...
        input = devm_input_allocate_device(dev);
...
        for (i = 0; i < pdata->nbuttons; i++) {
                const struct gpio_keys_button *button = &pdata->buttons[i];
                struct gpio_button_data *bdata = &ddata->data[i];

                error = gpio_keys_setup_key(pdev, input, bdata, button);

                error = gpio_keys_setup_key(pdev, bdata, button);
                if (error)
                        return error;

                if (button->wakeup)
                        wakeup = 1;
        }
        error = sysfs_create_group(&pdev->dev.kobj, &gpio_keys_attr_group);
        if (error) {
                dev_err(dev, "Unable to export keys/switches, error: %d\n",
                        error);
                goto fail2;
        }

        error = input_register_device(input);
        if (error) {
                dev_err(dev, "Unable to register input device, error: %d\n",
                        error);
                goto err_remove_group;
        }
...
```

c) Apply for gpio and define the gpio as input, and register gpio interrupt.

```
static int gpio_keys_setup_key(struct platform_device *pdev,
                               struct input_dev *input,
                               struct gpio_button_data *bdata,
                               const struct gpio_keys_button *button)
{
    const char *desc = button->desc ? button->desc : "gpio_keys";
    struct device *dev = &pdev->dev;
    irq_handler_t isr;
    unsigned long irqflags;
    int irq;
    int error;

    bdata->input = input;
    bdata->button = button;
    spin_lock_init(&bdata->lock);

    if (gpio_is_valid(button->gpio)) {

        error = devm_gpio_request_one(&pdev->dev, button->gpio,
                                      GPIOF_IN, desc);
        if (error < 0) {
            dev_err(dev, "Failed to request GPIO %d, error %d\n",
                button->gpio, error);
            return error;
        }

        if (button->debounce_interval) {
            error = gpio_set_debounce(button->gpio,
                        button->debounce_interval * 1000);
            /* use timer if gpiolib doesn't provide debounce */
            if (error < 0)
                bdata->software_debounce =
                            button->debounce_interval;
        }

        if (button->irq) {
            bdata->irq = button->irq;
        } else {
            irq = gpio_to_irq(button->gpio);
            if (irq < 0) {
                error = irq;
```

```
                        dev_err(dev,
                                "Unable to get irq number for GPIO %d, error %d\n",
                                button->gpio, error);
                        return error;
                }
                bdata->irq = irq;
        }

        INIT_DELAYED_WORK(&bdata->work, gpio_keys_gpio_work_func);

        isr = gpio_keys_gpio_isr;
        irqflags = IRQF_TRIGGER_RISING | IRQF_TRIGGER_FALLING;

} else {
        if (!button->irq) {
                dev_err(dev, "No IRQ specified\n");
                return -EINVAL;
        }
        bdata->irq = button->irq;

        if (button->type && button->type != EV_KEY) {
                dev_err(dev, "Only EV_KEY allowed for IRQ buttons.\n");
                return -EINVAL;
        }

        bdata->release_delay = button->debounce_interval;
        setup_timer(&bdata->release_timer,
                    gpio_keys_irq_timer, (unsigned long)bdata);

        isr = gpio_keys_irq_isr;
        irqflags = 0;
}

input_set_capability(input, button->type ?: EV_KEY, button->code);

/*
 * Install custom action to cancel release timer and
 * workqueue item.
 */
error = devm_add_action(&pdev->dev, gpio_keys_quiesce_key, bdata);
if (error) {
```

```
            dev_err(&pdev->dev,
                    "failed to register quiesce action, error: %d\n",
                    error);
            return error;
    }

    /*
     * If platform has specified that the button can be disabled,
     * we don't want it to share the interrupt line.
     */
    if (!button->can_disable)
            irqflags |= IRQF_SHARED;

    error = devm_request_any_context_irq(&pdev->dev, bdata->irq,
                                         isr, irqflags, desc, bdata);
    if (error < 0) {
            dev_err(dev, "Unable to claim irq %d; error %d\n",
                    bdata->irq, error);
            return error;
    }

    return 0;
}
```

d) Interrupt processing.

When button is pressed, an interrupt is generated and key value is reported.

```
static irqreturn_t gpio_keys_gpio_isr(int irq, void *dev_id)
{
        …
        mod_delayed_work(system_wq,
                 &bdata->work,
                 msecs_to_jiffies(bdata->software_debounce));
        …
}

static void gpio_keys_gpio_work_func(struct work_struct *work)
{
       ...
    gpio_keys_gpio_report_event(bdata);
       ...
```

```
}
static void gpio_keys_gpio_report_event(struct gpio_button_data *bdata)
{
    const struct gpio_keys_button *button = bdata->button;
    struct input_dev *input = bdata->input;
    unsigned int type = button->type ?: EV_KEY;
    int state = (gpio_get_value_cansleep(button->gpio) ? 1 : 0) ^ button->active_low;

    if (type == EV_ABS) {
        if (state)
            input_event(input, type, button->code, button->value);
        } else {
            input_event(input, type, button->code, !!state);
        }
        input_sync(input);
}
```

3.8.2 GPIO_leds Driver

1） Device Definition

Linux-4.1/arch/arm/boot/dts/embest-SBC-SBC8600.dts

Configure GPIO1.30 as ”sys_led”（system indicator）and GPIO1.31 as “user_led”, both lighted up by high level signal.

```
    leds {
        compatible = "gpio-leds";
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&user_leds_default>;
        pinctrl-1 = <&user_leds_sleep>;

        led@0 {
            label = "sys";
            gpios = <&gpio1 30 GPIO_ACTIVE_HIGH>;
            linux,default-trigger = "heartbeat";
            default-state = "off";
        };

        led@1 {
            label = "D36";
```

```
                gpios = <&gpio1 31 GPIO_ACTIVE_HIGH>;
                linux,default-trigger = "none";
                default-state = "off";
            };
        };
```

2） GPIO pinmux Configuration

Linux-4.1/arch/arm/boot/dts/embest-SBC-SBC8600.dts

Configure GPIO1.30 and GPIO1.31 as MODE7 (gpio mode) and AM33XX_PIN_OUTPUT (configuration output)

```
user_leds_default: pinmux_user_leds_default {
    pinctrl-single,pins = <
        /* leds */
        0x080 (PIN_OUTPUT_PULLUP | MUX_MODE7)     /* gpmc_csn1.GPIO1_30
*/
        0x084 (PIN_OUTPUT_PULLUP | MUX_MODE7)     /* gpmc_csn2.GPIO1_31
*/
};
```

3） **Driver Design**

Linux-4.1/drivers/leds/leds-gpio.c

Call platform_driver_register to register gpio_leds driver

```
static struct platform_driver gpio_led_driver = {
    .probe          = gpio_led_probe,
    .remove         = gpio_led_remove,
    .driver         = {
        .name     = "leds-gpio",
        .of_match_table = of_gpio_leds_match,
        .pm = &gpio_led_pm_ops,
    },
};

module_platform_driver(gpio_led_driver);

MODULE_AUTHOR("Raphael Assenat <raph@8d.com>, Trent Piepho
<tpiepho@freescale.com>");
MODULE_DESCRIPTION("GPIO LED driver");
MODULE_LICENSE("GPL");
```

```
MODULE_ALIAS("platform:leds-gpio");
```

Apply for gpio and call led_classdev_register to led_classdev driver

```
static int gpio_led_probe(struct platform_device *pdev)
{
...
    if (pdata && pdata->num_leds) {
        priv = devm_kzalloc(&pdev->dev,
                sizeof_gpio_leds_priv(pdata->num_leds),
                    GFP_KERNEL);
        if (!priv)
            return -ENOMEM;

        priv->num_leds = pdata->num_leds;
        for (i = 0; i < priv->num_leds; i++) {
            ret = create_gpio_led(&pdata->leds[i],
                        &priv->leds[i],
                        &pdev->dev, pdata->gpio_blink_set);
            if (ret < 0) {
                /* On failure: unwind the led creations */
                for (i = i - 1; i >= 0; i--)
                    delete_gpio_led(&priv->leds[i]);
                return ret;
            }
        }
    } else {
        priv = gpio_leds_create(pdev);
        if (IS_ERR(priv))
            return PTR_ERR(priv);
    }

    platform_set_drvdata(pdev, priv);

    return 0;
}

static int create_gpio_led(const struct gpio_led *template,
```

```
        struct gpio_led_data *led_dat, struct device *parent,
        int (*blink_set)(struct gpio_desc *, int, unsigned long *,
                        unsigned long *))
{
...
            ret = devm_gpio_request_one(parent, template->gpio, flags,
template->name);
...
            ret = gpiod_direction_output(led_dat->gpiod, state);
...
            return led_classdev_register(parent, &led_dat->cdev);
}
```

Users may access the file named brightness under /sys/class/leds/xxx/brightness, and call gpio_led_set to configure LED status

```
static void gpio_led_set(struct led_classdev *led_cdev,
            enum led_brightness value)
{
…
                        gpiod_set_value(led_dat->gpiod, level);
}
```

3.9 System Update

SBC8600B can boot up from TF card or NAND Flash, if the jumper JP5 is disconnect, the board will boot from nand flash, otherwise it will boot from TF card.

This section will briefly introduce the process of system update on a TF card and a NAND Flash.

3.9.1 Update of TF Card System Image

1. Formatting TF Card

HP USB Disk Storage Format Tool 2.0.6 is recommended as the formatting tool:

Please download it from here:

a) Insert TF card into a card reader and then insert the reader into your PC.

b) Open HP USB Disk Storage Format Tool to show the following window:

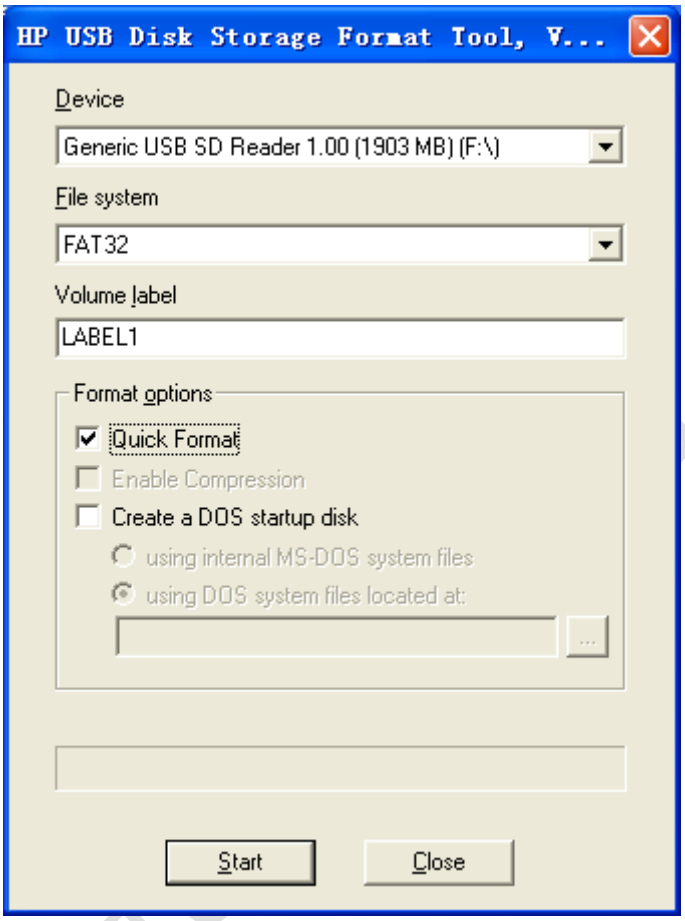


Figure 3-6 HP USB Disk Tool

c) Select “FAT32” file system

d) Click “Start”

e) When formatting is complete, click “OK”

Note:

- It is not recommended to use other versions of HP USB Disk Storage Format Tool.
- HP USB Disk Storage Format Tool will erase the partitions of TF card.

2. Updating Image Files

Copy all the files under linux\image\ of the DVD-ROM to the TF card, insert TF card on the board and short the jumper JP5, then power on the board. The information in HyperTerminal window is shown below;

Note:

📖 The default display is a 4.3-inch LCD. If you are working with the LCDs of other size, please enter u-boot when the board is booting up to configure the display mode, and then type **boot** to continue boot-up process. Please refer to 3.10 Selecting Display Mode for more details

```
U-Boot SPL 2015.07-ga66ad5e-dirty (Jun 19 2017 - 14:48:34)
reading args
spl_load_image_fat_os: error reading image args, err - -1
reading u-boot.img
reading u-boot.img

U-Boot 2015.07-ga66ad5e-dirty (Jun 19 2017 - 14:48:34 +0800)

Watchdog enabled
I2C:   ready
DRAM:  512 MiB
NAND:  512 MiB
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
*** Warning - bad CRC, using default environment

Net:   <ethaddr> not set. Validating first E-fuse MAC
cpsw, usb_ether
Hit any key to stop autoboot:  0
WARNING: Could not determine device tree to use
switch to partitions #0, OK
mmc0 is current device
SD/MMC found on device 0
reading boot.scr
** Unable to read file boot.scr **
reading uEnv.txt
199 bytes read in 5 ms (38.1 KiB/s)
Loaded env from uEnv.txt
Importing environment from mmc0 ...
```

```
Running uenvcmd ...
reading /embest_SBC_SBC8600.dtb
36786 bytes read in 11 ms (3.2 MiB/s)
reading /zImage
3537760 bytes read in 260 ms (13 MiB/s)
reading ramdisk.img
12034853 bytes read in 861 ms (13.3 MiB/s)
Booting from ramdisk ...
Kernel image @ 0x82000000 [ 0x000000 - 0x35fb60 ]
## Loading init Ramdisk from Legacy Image at 88080000 ...
   Image Name:
   Created:          2017-04-24     9:02:19 UTC
   Image Type:      ARM Linux RAMDisk Image (gzip compressed)
   Data Size:       12034789 Bytes = 11.5 MiB
   Load Address: 23000000
   Entry Point:    23000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 88000000
   Booting using the fdt blob at 0x88000000
   Loading Ramdisk to 8f485000, end 8ffff2e5 ... OK
   Loading Device Tree to 8f479000, end 8f484fb1 ... OK

Starting kernel ...


[      0.000000] Booting Linux on physical CPU 0x0
[      0.000000] Initializing cgroup subsys cpuset
[      0.000000] Initializing cgroup subsys cpu
[      0.000000] Initializing cgroup subsys cpuacct
[      0.000000] Linux version 4.1.6 (chengpg@embest-tech) (gcc version 4.9.2
20140904 (prerelease) (crosstool-NG linaro-1.13.1-4.9-2014.09 - Linaro GCC
4.9-2014.09) ) #56 PREEMPT Mon Jun 19 18:01:17 CST 2017
[      0.000000] CPU: ARMv7 Processor [413fc082] revision 2 (ARMv7), cr=10c5387d
[      0.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction
cache
[      0.000000] Machine model: TI AM335x SBC8600
[      0.000000] cma: Reserved 24 MiB at 0x9e800000
[      0.000000] Memory policy: Data cache writeback
[      0.000000] CPU: All CPU(s) started in SVC mode.
[      0.000000] AM335X ES2.1 (sgx neon )
......
[      2.649525] RAMDISK: gzip image found at block 0
```

```
[       5.561390] EXT4-fs (ram0): mounting ext2 file system using the ext4 subsystem
[       5.570741] EXT4-fs (ram0): mounted filesystem without journal. Opts: (null)
[       5.578166] VFS: Mounted root (ext2 filesystem) on device 1:0.
[       5.584888] devtmpfs: mounted
[       5.588600] Freeing unused kernel memory: 268K (c090b000 - c094e000)
[       5.644967] EXT4-fs (ram0): re-mounted. Opts: (null)
Starting logging: OK
Populating /dev using udev: [       5.820982] udevd[84]: starting version 2.1.1
[       5.839500] random: udevd urandom read with 17 bits of entropy available
[       6.901556] FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some
data may be corrupt. Please run fsck.
done
Starting watchdog...
Initializing random number generator... done.
Starting system message bus: done
Starting network...
[       7.781337] net eth0: initializing cpsw version 1.12 (0)
[       7.867393] net eth0: phy found : id is : 0x4dd072
ssh-keygen: generating new host keys: ED25519
Starting sshd: OK
```

The above information indicates a successful boot-up of Linux from TF card.

3.9.2 Update of NAND Flash

NAND boot-up image update is accomplished base on u-boot. No matter whether NAND Flash has data or not, u-boot can be used to update NAND Flash images.

1. Preparation

a> Format the TF card as FAT or FAT32 file system by using HP USB Disk Storage Format Tool 2.0.6

b> Copy files **MLO, u-boot.img, zImage，ramdisk.img, rootfs.tar.xz, logo.bmp, uEnv.txt** in \linux\image of the DVD-ROM into TF card.

2. Update

a) Insert TF card which contains the system images into the development board and shourt the jumper **JP5**, and then connect power supply. The update process will auto starts.

```
U-Boot 2015.07-ga66ad5e-dirty (Jun 19 2017 - 14:48:34 +0800)

Watchdog enabled
I2C:   ready
DRAM:  512 MiB
NAND:  512 MiB
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
*** Warning - bad CRC, using default environment

Net:   <ethaddr> not set. Validating first E-fuse MAC
cpsw, usb_ether
Hit any key to stop autoboot:  0
WARNING: Could not determine device tree to use
switch to partitions #0, OK
mmc0 is current device
SD/MMC found on device 0
reading boot.scr
** Unable to read file boot.scr **
reading uEnv.txt
199 bytes read in 5 ms (38.1 KiB/s)
Loaded env from uEnv.txt
Importing environment from mmc0 ...
Running uenvcmd ...
reading /embest_SBC_SBC8600.dtb
36786 bytes read in 11 ms (3.2 MiB/s)
reading /zImage
3537760 bytes read in 260 ms (13 MiB/s)
reading ramdisk.img
12034853 bytes read in 861 ms (13.3 MiB/s)
Booting from ramdisk ...
Kernel image @ 0x82000000 [ 0x000000 - 0x35fb60 ]
## Loading init Ramdisk from Legacy Image at 88080000 ...
   Image Name:
   Created:          2017-04-24   9:02:19 UTC
   Image Type:    ARM Linux RAMDisk Image (gzip compressed)
   Data Size:     12034789 Bytes = 11.5 MiB
```

```
    Load Address: 23000000
    Entry Point:    23000000
    Verifying Checksum ... OK
## Flattened Device Tree blob at 88000000
    Booting using the fdt blob at 0x88000000
    Loading Ramdisk to 8f485000, end 8ffff2e5 ... OK
    Loading Device Tree to 8f479000, end 8f484fb1 ... OK

Starting kernel ...

[      0.000000] Booting Linux on physical CPU 0x0
[      0.000000] Initializing cgroup subsys cpuset
[      0.000000] Initializing cgroup subsys cpu
```

```
 ......
[      5.589121] VFS: Mounted root (ext2 filesystem) on device 1:0.
[      5.595856] devtmpfs: mounted
[      5.599582] Freeing unused kernel memory: 268K (c090b000 - c094e000)
[      5.655671] EXT4-fs (ram0): re-mounted. Opts: (null)
Starting logging: OK
Populating /dev using udev: [      5.829716] udevd[85]: starting version 2.1.1
[      5.835775] random: udevd urandom read with 17 bits of entropy available
[      6.892135] FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some
data may be corrupt. Please run fsck.
done
Starting watchdog...
Initializing random number generator... done.
Starting system message bus: done
Starting network...
[      7.772444] net eth0: initializing cpsw version 1.12 (0)
[      7.857384] net eth0: phy found : id is : 0x4dd072
ssh-keygen: generating new host keys: ED25519
Starting sshd: OK
running system update...
UPDATE FLAG    : TRUE
==============NANDFLASH UPDATE============
Welcome to ARM
arm login: Writing MLO ...
Erasing 128 Kibyte @ 0 -- 100 % complete
Writing at 0x00000000
Writing MLO ...
```

```
Erasing 128 Kibyte @ 0 -- 100 % complete
Writing at 0x00000000
Writing MLO ...
Erasing 128 Kibyte @ 0 -- 100 % complete
Writing at 0x00000000
Writing MLO ...
Erasing 128 Kibyte @ 0 -- 100 % complete
Writing at 0x00000000
Writing u-boot.img ...
Erasing 128 Kibyte @ e0000 -- 100 % complete
Writing at 0x00000000
Writing at 0x00020000
Writing at 0x00040000
Erasing 128 Kibyte @ 0 -- 100 % complete
Erasing 128 Kibyte @ 0 -- 100 % complete
Writing logo.bmp ...
Erasing 128 Kibyte @ 3e0000 -- 100 % complete
Writing at 0x00000000
Writing at 0x00020000
Writing at 0x00040000
Writing at 0x00060000
Writing embest_SBC_SBC8600.dtb ...
Erasing 128 Kibyte @ 20000 -- 100 % complete
Writing at 0x00000000
Writing zImage ...
Erasing 128 Kibyte @ 7e0000 -- 100 % complete
Writing at 0x00000000
.....
Writing at 0x00340000
Writing rootfs.tar.xz ...
ls: /dev/ubi[0-9]: No such file or directory
Erasing 12[     10.857820] cpsw 4a100000.ethernet eth0: Link is Up - 100Mbps/Full -
flow control rx/tx
Erasi[     18.699786] ubi0: attaching mtd10lete
Erasing 128 Kibyte @ f1e0000 -- 100 % complete
[     19.436363] ubi0: scanning is finished
[     19.446587] ubi0: empty MTD device detected
[     19.480742] ubi0: attached mtd10 (name "NAND.file-system", size 242 MiB)
[     19.490520] ubi0: PEB size: 131072 bytes (128 KiB), LEB size: 126976 bytes
[     19.498281] ubi0: min./max. I/O unit sizes: 2048/2048, sub-page size 512
[     19.505034] ubi0: VID header offset: 2048 (aligned 2048), data offset: 4096
```

```
[    19.516521] ubi0: good PEBs: 1936, bad PEBs: 0, corrupted PEBs: 0
[    19.530000] ubi0: user volume: 0, internal volumes: 1, max. volumes count: 128
[    19.537565] ubi0: max/mean erase counter: 0/0, WL threshold: 4096, image
sequence number: 3648342632
[    19.546764] ubi0: available PEBs: 1852, total reserved PEBs: 84, PEBs reserved
for bad PEB handling: 80
[    19.558880] ubi0: background thread "ubi_bgt0d" started, PID 578
UBI device number 0, total 1936 LEBs (245825536 bytes, 234.4 MiB), available 1852
LEBs (235159552 bytes, 224.3 MiB), LEB size 126976 bytes (124.0 KiB)
Set volume size to 235159552
Volume ID 0, size 1852 LEBs (235159552 bytes, 224.3 MiB), LEB size 126976 bytes
(124.0 KiB), dynamic, name "rootfs", alignment 1
[    19.750310] UBIFS (ubi0:0): default file-system created
[    19.776923] UBIFS (ubi0:0): background thread "ubifs_bgt0_0" started, PID 589
[    19.878742] UBIFS (ubi0:0): UBIFS: mounted UBI device 0, volume 0, name
"rootfs"
[    19.906996] UBIFS (ubi0:0): LEB size: 126976 bytes (124 KiB), min./max. I/O unit
sizes: 2048 bytes/2048 bytes
[    19.937002] UBIFS (ubi0:0): FS size: 233635840 bytes (222 MiB, 1840 LEBs),
journal size 11681792 bytes (11 MiB, 92 LEBs)
[    19.973269] UBIFS (ubi0:0): reserved for root: 4952683 bytes (4836 KiB)
[    19.987334] UBIFS (ubi0:0): media format: w4/r0 (latest is w4/r0), UUID
A1EB0AA1-88B6-43D1-8B86-0CD553781279, small LPT model
/media/mmcblk0p1/rootfs.tar.xz (1/1)
[    28.197424] random: nonblocking pool is initializeds          0:08            50 s
   100 %            18.0 MiB / 71.2 MiB = 0.252    1.5 MiB/s          0:48
/
[    76.686683] UBIFS (ubi0:0): un-mount UBI device 0
[    76.691654] UBIFS (ubi0:0): background thread "ubifs_bgt0_0" stops
UPDATE      : OK
UPDATE      : COMPLETED
```

Fast flashing LED on the board indicates that the update is running;

When the LEDs on the board blink as heart frequency and buzzer alarmed, the update has been finished.

Please remove TF card and the JP5 jumper cap, and then reboot the board.

3. U-boot configuration

The system image has a default setting for 4.3-inch LCD. You can change the settings in UBOOT according to the detailed instructions contained in 3.10 Selecting Display Mode.

3.10 Display Mode Configurations ABS

System supports a wide range of display mode (4.3/7.0-inch-LCD supported, VGA/LVDS/9.7-inch-LCD not supported). Users can change the display mode by modifying the U-Boot configure parameters.

How to enter the u-boot command mode:

Power on the board and press any key on PC's keyboard to enter u-boot when you see "Hit any key to stop autoboot" in your terminal window.

```
U-Boot SPL 2015.07-ga66ad5e-dirty (Jun 19 2017 - 14:48:34)
The Expected Linux image was not found. Please check your NAND configuration.
Trying to start u-boot now...

U-Boot 2015.07-ga66ad5e-dirty (Jun 19 2017 - 14:48:34 +0800)

Watchdog enabled
I2C:   ready
DRAM:  512 MiB
NAND:  512 MiB
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
*** Warning - bad CRC, using default environment

Net:   <ethaddr> not set. Validating first E-fuse MAC
cpsw, usb_ether
Hit any key to stop autoboot:  0
U-Boot#  0 (press any key to enter uboot)
```

1） Configuring for 4.3-inch LCD;

Execute the following instructions in u-boot mode to configure for 4.3-inch display mode;

- U-Boot# **setenv dispmode 4.3inch_LCD**

- U-Boot# **saveenv**

2） Configuring for 7-inch LCD;

Execute the following instructions in u-boot mode to configure for 7-inch display mode;

- U-Boot# **setenv dispmode 7inch_LCD**
- U-Boot# **saveenv**

3） Configuring for VGA;

Execute the following instructions in u-boot mode to configure for VGA display mode;

- SBC8600# **run clearenv**
- SBC8600# **setenv dispmode VGA**
- SBC8600# **saveenv**

4） Configuring for LVDS;

Execute the following instructions in u-boot mode to configure for LVDS display mode;

- SBC8600# **run clearenv**
- SBC8600# **setenv dispmode LVDS_800x600**
- SBC8600# **saveenv**

5） Configuring for LCD8000-97C

Execute the following instructions in u-boot mode to configure for 9.7inch LVDS display mode;

- SBC8600# **run clearenv**
- SBC8600# **setenv dispmode LVDS_1024x768**
- SBC8600# **saveenv**

Note:

Command to restore default U-Boot environment: run erase_env;reset

3.11 Test and Demonstration

This section will carry out many test on the devices on SBC8600B and also demonstration of Android and DVSDK systems.

Note:
The following tests are all implemented by entering instructions in a HyperTerminal window.

3.11.1 LED Testing

The D35 LED on the board is the system indicator; D36 is n user custom LED.

The following operations are accomplished in HyperTerminal:

Controlling system indicator

- root@arm:~# **echo none > /sys/class/leds/sys/trigger**
- root@arm:~# **echo 1 > /sys/class/leds/sys/brightness**
- root@arm:~# **echo 0 > /sys/class/leds/sys/brightness**

Restoring to heart blink function

- root@arm:~# **echo heartbeat > /sys/class/leds/sys/trigger**

Controlling user custom LED

- root@arm:~# **echo 1 > /sys/class/leds/D36/brightness**
- root@arm:~# **echo 0 > /sys/class/leds/D36/brightness**

The LED will respond accordingly to the instructions.

3.11.2 KEYPAD Testing

The board has two user custom keys, BACK and MENU. You can test them by executing the following instructions.

```
root@arm:~# evtest /dev/input/event1
    Input driver verevdev: (EVIOCGBIT): Suspicious buffer size 511
    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 1 (Esc)
        Event code 59 (F1)
    Testing ... (interrupt to exit)
    Event: time 1233046135.256046, type 1 (Key), code 1 (Esc), value 1
    Event: time 1233046135.256053, ---------- Report Sync ------------
    Event: time 1233046135.426967, type 1 (Key), code 1 (Esc), value 0
    Event: time 1233046135.426970, ---------- Report Sync ------------
    Event: time 1233046136.373255, type 1 (Key), code 59 (F1), value 1
    Event: time 1233046136.373260, ---------- Report Sync ------------
    Event: time 1233046136.548841, type 1 (Key), code 59 (F1), value 0
    Event: time 1233046136.548844, ---------- Report Sync ------------
```

Note:
Press Ctrl+C to quit the test. These combined keys can be used to quit any following test.

3.11.3 Touch Screen Testing

This test requires that Linux system boots up from NAND Flash.

1) Execute the following instruction to test touch-screen

- root@arm: # **ts_calibrate**

The information on LCD will guide you to click the icon "+" for 5 times to complete the calibration.

2) Calibration is complete, enter the following commands for Touch Panel Test

- root@arm: # **ts_test**

Select drawing dots or drawing lines from the prompt information to start testing.

3.11.4 Backlight Testing

The backlight brightness has a range from 0 to 8, in which 8 means highest brightness, 0 means lowest.

Execute the following instructions to test backlight brightness.

a> Execute the following instruction to view the default backlight brightness;

- root@arm:~# **cat /sys/class/backlight/backlight/brightness**

b> Execute the following instructions to set backlight brightness to 0 and check if the value is set;

- root@arm:~# **echo 0 > /sys/class/backlight/backlight/brightness**

Now backlight is turned off and the LCD shows a black screen.

c> Execute the following instructions to set brightness to level 8;

- root@arm:~# **echo 8 > /sys/class/backlight/backlight/brightness**

The screen is turned on.

3.11.5 ADC Testing

SBC8600B has 4 ADC input channels to measure external analog signal voltages.

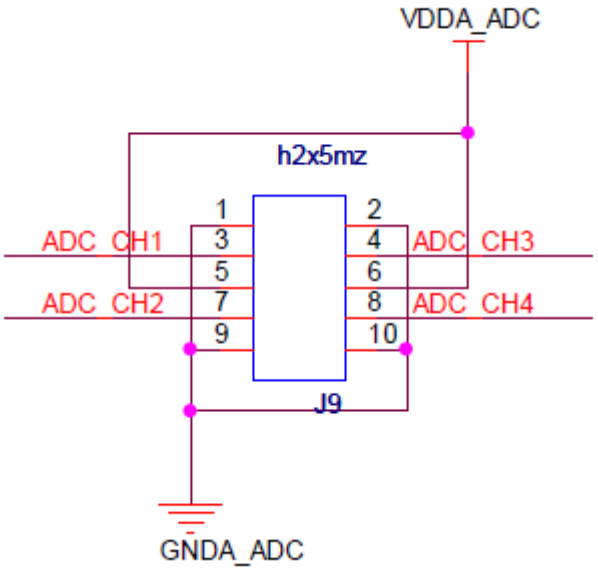


Figure 3-7 ADC Interface

Table 3-4 ADC channels

ADC Pin	System node
ADC_CH1	in_voltage4_raw
ADC_CH2	in_voltage5_raw
ADC_CH3	in_voltage6_raw
ADC_CH4	in_voltage7_raw

Connect the signal that to be measured to a ADC pin (ADC_CH2 eg)

Execute the following instruction to get ADC value

- root@arm:**~# cat /sys/bus/iio/devices/iio:device0/in_voltage5_raw**

Follow below formula to get adc input signal voltage

Voltage Input = VDDA_ADC * (value read) / 4096

here VDDA_ADC = 1.8V

3.11.6 RTC Testing

SBC8600B has a hardware clock which can store and recover system clock; please follow the steps listed below to test RTC;

1） Execute the following instruction to set system clock to 0:10 pm, Jun. 30th, 2017;

- root@arm: # **date –s “2017-06-30 12:10”**

The information in HyperTerminal window is shown below;

Setting System Clock

```
Fri Jun 30 12:10:00 UTC 2017
```

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

- root@arm: # **hwclock -w**

3） Execute the following instruction to read RTC;

- root@arm: # **hwclock**

The information in HyperTerminal window is shown below;

RTC Clock

```
Fri Jun 30 12:11:32 2017  0.000000 seconds
```

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

- root@arm: # **hwclock -s**
- root@arm: # **date**

The information in HyperTerminal window is shown below;

System Clock

```
Fri Jun 30 12:13:06 UTC 2017
```

The above information indicates that system clock has been recovered with hardware clock;

Note:
By default, SBC8600B has no CR1220 battery installed on board; You need to purchase it separately.

3.11.7 TF Card Testing

1） Insert a TF card on SBC8600B, system will mount the filesystem under /media/ automatically; please execute the following instructions to view the device name of TF card in the system;

- root@arm:~# **df -h**

The information in HyperTerminal window is shown below;

TF Card Device Name

```
Filesystem      Size  Used Avail Use% Mounted on
......
/dev/mmcblk0p1  3.7G   34M  3.6G   1% /media/mmcblk0p1
```

The characters in bold in the above table is the device name of the TF card.

2） Execute the following instruction to view the contents of the TF card;

- root@arm:/media# **ls mmcblk0p1/**

The information in HyperTerminal window is shown below;

Contents of TF Card

```
MLO              ramdisk.img          uEnv.txt
logo.bmp         u-boot.img
```

3） Execute the following instruction to unmount the TF card manually;

- root@arm:~# **umount /media/mmcblk0p1**

4） Execute the following instruction to mount the TF card manually and view the device name;

- root@arm:~# **mount -t vfat /dev/mmcblk0p1 /mnt/mnt**
- root@arm:~# **df –h**

The information in HyperTerminal window is shown below;

TF Card Device Name

```
Filesystem          Size      Used Available Use% Mounted on
......
/dev/mmcblk0p1      3.7G      34M       3.6G   1% /mnt
```

Continue to execute the following instruction to view the contents of the TF card;

- root@arm:~# **ls /media/mnt**

The information in HyperTerminal window is shown below;

Contents of TF Card

MLO	ramdisk.img	uEnv.txt
logo.bmp	u-boot.img	

3.11.8 USB DEVICE Testing

The test of USB device is accomplished by creating a network communication between miniUSB interface on the board and USB interface on PC;

1） After system boots up, connect SBC8600B to your PC with a Mini B-to-USB A cable, and then you need to install Linux USB Ethernet driver; please refer to **错误!未找到引用源。 错误!未找到引用源。**.

2） Execute the following instructions to set IP address of the USB interface of SBC8600B (the IP used below is only for reference; you can select a difference IP);

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

The information in HyperTerminal window is shown below;

```
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:26 errors:0 dropped:0 overruns:0 frame:0
          TX packets:26 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2316 (2.2 KiB)  TX bytes:2316 (2.2 KiB)

usb0      Link encap:Ethernet  HWaddr 5E:C5:F6:D4:2B:91
          inet addr:192.168.1.115  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:253 errors:0 dropped:0 overruns:0 frame:0
          TX packets:43 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:35277 (34.4 KiB)  TX bytes:10152 (9.9 KiB)
```

3） Right-click **My Network Places** on the desktop of your PC and select **Properties** to open **Network Connections** window; you can find a new **Local Area Connection** in the window as shown below;

Figure 3-8 New LAN Connection

4） Right-click the icon of new **Local Area Connection** and select **Properties**, and then double-click **Internet Protocol (TCP/IP)** to open the following window;

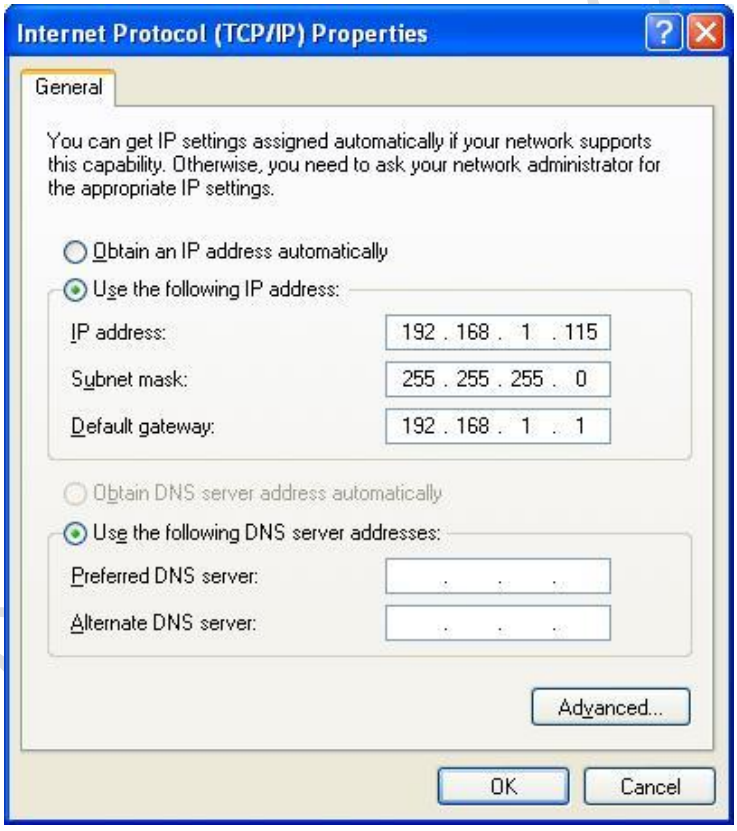


Figure 3-9 Setting IP address

Ensure the network segment where the IP address of PC's USB virtual network interface is set as same as that of SBC8600B's USB interface, and then click **OK**;

5） Use ping command in HyperTerminal window to test if the network works properly;

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

The information in HyperTerminal window is shown below;

Testing 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 above information indicates the network has been created successfully.

Note:
Please ensure the IP addresses used above and the Ethernet IP address of the board are set in different segments.

3.11.9 USB Host Testing

1） Insert a flash drive to the USB interface on SBC8600B, and then execute the following instruction to view the flash drive's device name assigned by the system;

- root@arm:~# **df –h**

The information in HyperTerminal window is shown below;

Table 1

Flash Drive Device Name

```
Filesystem                Size      Used Available Use% Mounted on
......
/dev/sda1                 3.7G     74M      3.6G     2% /media/usbhd-sda1
```

The characters in bold in the above information is device name of the flash drive; Continue to execute the following instruction to view the contents of the disk;

- root@arm:~# **ls /media/usbhd-sda1/**

The information in HyperTerminal window is shown below;

Contents of Flash Drive

MLO u-boot.img zImage

2） Execute the following instruction to unmount flash drive manually;

- root@arm:~# **umount /media/usbhd-sda1/**

3） Execute the following instructions to mount the flash drive manually and check the device name;

- root@arm:~# **mount -t vfat /dev/sda1 /mnt/**
- root@arm:~# **df -h**

Flash Drive Device Name

Filesystem	Size	Used	Available	Use%	Mounted on
.....					
/dev/sda1	**3.7G**	**74M**	**3.6G**	**2%**	**/mnt**

The characters in bold in the above information is device name of the flash drive assigned manually;

Note:
By manually unmounting flash drive and mounting it again, SBC8600B can write flash drive faster.

3.11.10 AUDIO Testing

SBC8600B has input/output interfaces which support audio recording and playback. The filesystem is integrated with alsa-utils audio recording and playback tool. You can test it by following the steps below;

1） Insert a microphone into the 3.5mm audio input interface on SBC8600B, and

then execute the following instruction to start audio recording;

- root@arm:~# **arecord -t wav -c 1 -r 44100 -f S16_LE -v k**

The information in HyperTerminal window is shown below;

Start Recording

```
Recording WAVE 'k' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Plug PCM: Hardware PCM card 0 'omap3evm' 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
    stop_threshold      : 22052
    silence_threshold: 0
    silence_size : 0
    boundary        : 1445199872
    appl_ptr        : 0
    hw_ptr          : 0
Slave: Hardware PCM card 0 'AM335x-SGTL5000' device 0 subdevice 0.....
```

2） Insert an earphone into the 3.5mm audio output interface, and then execute the following instruction to play the recorded audio;

- root@arm:~# **aplay -t wav -c 2 -r 44100 -f S16_LE -v k**

The information in HyperTerminal window is shown below;

Start Playback

```
Recording WAVE 'k' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

```
Plug PCM: Hardware PCM card 0 'omap3evm' 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
    stop_threshold      : 22052
    silence_threshold: 0
    silence_size : 0
    boundary        : 1445199872
    appl_ptr        : 0
    hw_ptr          : 0
```

3） Following instruction to adjust volume;

- root@arm:~# **amixer set Headphone 100**

Where the volume parameter 1(100 eg) give value in range [0..127].

3.11.11 Network Testing

SBC8600B has two Ethernet interfaces, NET1 (J1) and NET2 (J2) ; Corresponding device nodes are eth0 and eth1. Please use two network cables connect the interfaces to a network and ensure that the IP addresses of the interfaces are set in different network segments.

Note:

The IP addresses of the two network interfaces need to be set in different network segments, or the testing would be failed.

1） Execute the following instructions to set the IP address of NET1

- root@arm:~# **ifconfig eth0 192.192.192.200**
- root@arm:~# **ifconfig**

The information in HyperTerminal window is shown below;

```
eth0      Link encap:Ethernet  HWaddr D4:94:A1:8D:EB:25
          inet addr:192.192.192.200 Bcast:192.192.192.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:137 errors:0 dropped:4 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:13792 (13.4 KiB)  TX bytes:0 (0.0 B)
          Interrupt:40

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)
```

2） Execute the following instruction to test the network communication

- root@arm:~# **ping 192.192.192.170**

The information in HyperTerminal window is shown below;

```
PING 192.192.192.170 (192.192.192.170): 56 data bytes
64 bytes from 192.192.192.170: seq=0 ttl=128 time=4.486 ms
64 bytes from 192.192.192.170: seq=1 ttl=128 time=0.336 ms
```

The above information indicates the network is working properly.

3） Execute the following instructions to set the IP address of NET2

- root@arm:~# **ifconfig eth1 192.168.168.116**
- root@arm:~# **ifconfig**

The information in HyperTerminal window is shown below;

```
eth1      Link encap:Ethernet  HWaddr 00:17:EA:96:34:D5
          net addr:192.168.168.116  Bcast:192.168.168.255
Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  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:1000
          RX bytes:0 (0.0 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)
```

4） Execute the following instruction to test the network communication

- root@arm:~# **ping 192.168.168.121**

The information in HyperTerminal window is shown below;

```
PING 192.192.192.170 (192.192.192.170): 56 data bytes
64 bytes from 192.192.192.170: seq=0 ttl=128 time=4.486 ms
64 bytes from 192.192.192.170: seq=1 ttl=128 time=0.336 ms
```

The above information indicates the network is working properly.

3.11.12 CAN Testing

SBC8600B can be working as a CAN device. Please connect the CAN interfaces on your SBC8600B and another CAN device according to the board schematic and the figure shown below:

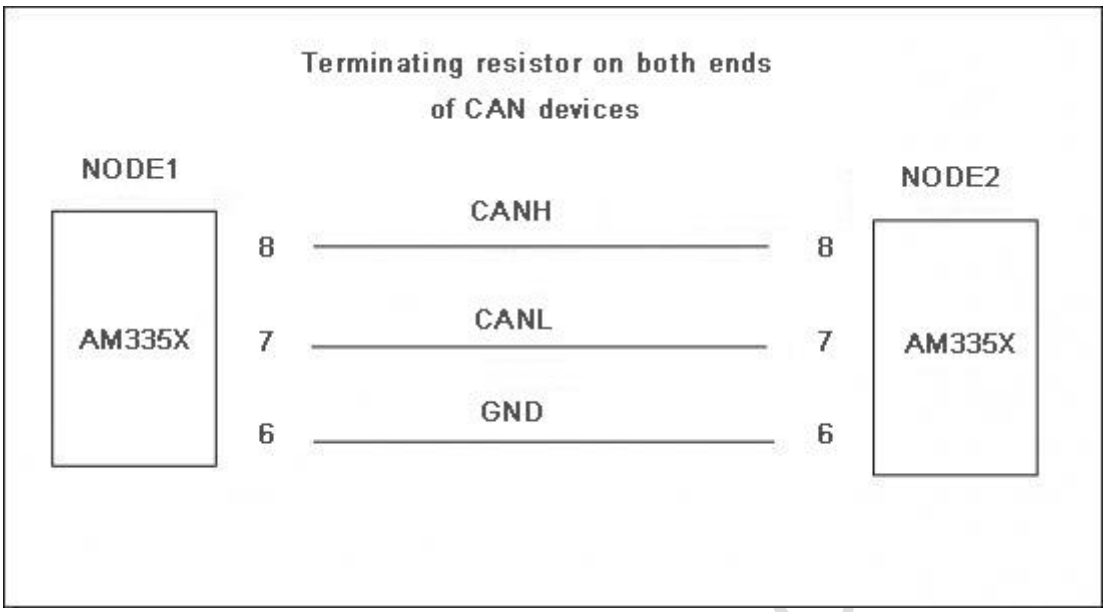


Figure 3-10 CAN Connection

Follow the steps listed below to complete CAN testing

1) Set the communication bit rate to 125Kbps for both SBC8600B and the other CAN device, and enable CAN devices.

- root@arm:~# **ifconfig can0 down**

2) Making sure the device is closed, otherwise an error will occur.

- root@arm:~# **/bin/ip link set can0 type can bitrate 125000**
- root@arm:~# **/bin/ip link set can0 type can restart-ms 100**

3) Essential step to set recovery time.

- root@arm:~# **canconfig can0 start**

4) Transmit data on the one device by typing the following instructions.

- root@arm:~# **cansend can0 "5A1#1122334455667788"**

Note:
The instruction sends data only once. Type it again to send another date package.
The receiving device needs to remain in receiving status so that the received information can be shown in the terminal window.

5) Receiving data package on another device connected with the previous one

The terminal window will print the information of the received data package

- root@arm:~# **candump can0**

6) Stop the CAN device

- root@arm:~# **ipconfig can0 down**

You can run the test with different baudrates. The following table contains several baudrates which have been proved. However, you can also try other baudrates if you want.

Table 3-5 Proved Baudrates

No.	Proved Baudrates
1	25Kbps(250000)
2	50Kbps(50000)
3	125Kbps(125000)
4	500Kbps(500000)
5	650Kbps(650000)
6	1Mbps(1000000)

Note:

- You need to disable CAN bus BEFORE you can set a new baud rate. The two end devices of CAN communication must set identical baud rate.
- For source code, please refer to Open-Source software can-utils.

3.11.13 RS485 Testing

Please connect SBC8600B to another device enabled with RS485 bus according to the figure shown below and the schematic under /HW design/schematic/ in the DVD-ROM;

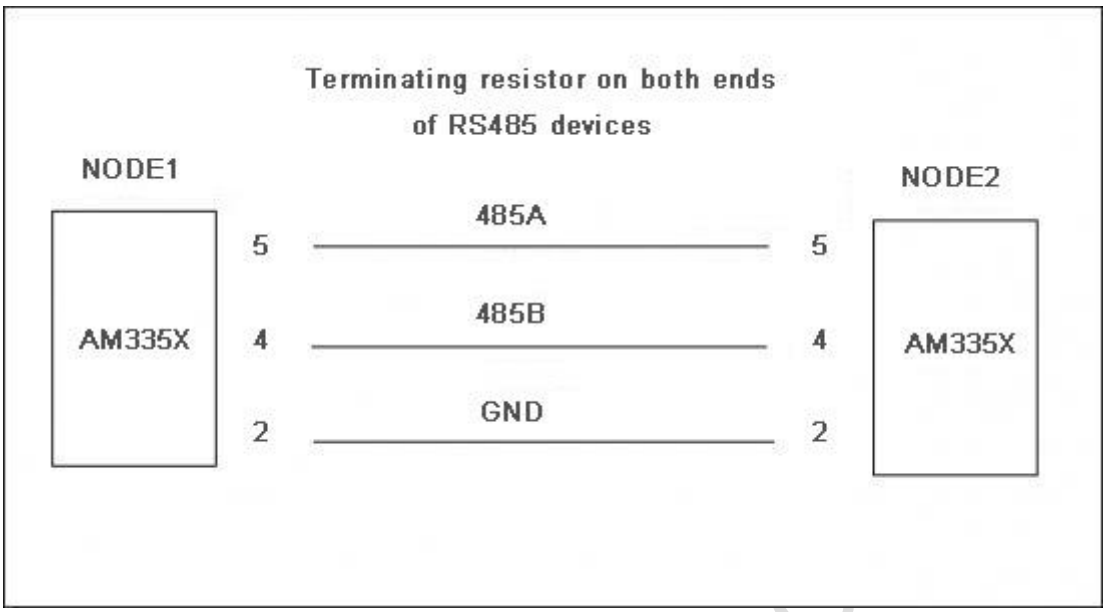


Figure 3-11 RS485 Bus Connections

RS485 interface works under half-duplex mode, which means each of two ends can only send or receive data at a time. Copy the file uart_test under linux\example\uart_test into TF card, and then insert the card on SBC8600B and execute the following instructions;

- root@arm:~# **/embest/uart_test -d /dev/ttyS1 -b 115200**

The information in HyperTerminal window should be shown as below;

```
/dev/ttyS1 SEND: 1234567890
/dev/ttyS1 RECV 10 total
/dev/ttyS1 RECV: 1234567890
/dev/ttyS1 SEND: 1234567890
/dev/ttyS1 RECV 10 total
/dev/ttyS1 RECV: 1234567890
/dev/ttyS1 SEND: 1234567890
/dev/ttyS1 RECV 10 total
/dev/ttyS1 SEND: 1234567890
/dev/ttyS1 RECV 10 total
```

3.11.14 Serial Interface Testing

Short the pins RX3V3 and TX3V3 of J5 on the board and copy the file uart_test under linux\example\uart_test to TF card, and then instert it on the board. Execute the following instructions in the terminal window;

- root@arm:~ # **/embest/uart_test -d /dev/ttyS2 -b 115200**

The following information in the terminal window indicates a successful testing.

```
dev/ttyO2 SEND: 1234567890
/dev/ttyS2 RECV 10 total
/dev/ttyS2 RECV: 1234567890
/dev/ttyS2 SEND: 1234567890
/dev/ttyS2 RECV 10 total
/dev/ttyS2 RECV: 1234567890
/dev/ttyS2 SEND: 1234567890
/dev/ttyS2 RECV 10 total
/dev/ttyS2 RECV: 1234567890
/dev/ttyS2 SEND: 1234567890
/dev/ttyS2 RECV 10 total
/dev/ttyS2 RECV: 1234567890
/dev/ttyS2 SEND: 1234567890
/dev/ttyS2 RECV 10 total
/dev/ttyS2 RECV: 1234567890
```

The same testing method can be applied on UART3, UART4 and UART5 of J6 and J7 on SBC8600B

3.11.15 Buzzer Testing

Enable the buzzer;

- root@arm:~# **echo 1 > /sys/class/leds/buzzer/brightness**

Disable the buzzer

- root@arm:~# **echo 0 > /sys/class/ leds/buzzer/brightness**

3.11.16 Suspend & Resume Testing

Execute the following instruction to enter suspend state

- root@arm:~# **echo mem > /sys/power/state**

```
[ 4351.715262] PM: Syncing filesystems ... done.
[ 4351.729920] Freezing user space processes ... (elapsed 0.001 seconds) done.
[ 4351.738777] Freezing remaining freezable tasks ... (elapsed 0.001 seconds)
done.
    [ 4351.747471] Suspending console(s) (use no_console_suspend to debug)
```

When system suspends finished, as above messages showing, use Uart Input,Touch Screen or Button S2 to trigger system resuming.

```
[ 4352.047646] PM: suspend of devices complete after 293.057 msecs
[ 4352.049277] PM: late suspend of devices complete after 1.602 msecs
[ 4352.051137] PM: noirq suspend of devices complete after 1.833 msecs
[ 4352.051147] PM: Successfully put all powerdomains to target state
[ 4352.051147] PM: Wakeup source UART
[ 4352.068706] PM: noirq resume of devices complete after 17.431 msecs
[ 4352.070122] PM: early resume of devices complete after 1.071 msecs
[ 4352.070950] net eth0: initializing cpsw version 1.12 (0)
[ 4352.075081] sgtl5000 0-000a: Failed to get mclock: -2
[ 4352.145051] net eth0: phy found : id is : 0x4dd072
[ 4352.147181] net eth1: initializing cpsw version 1.12 (0)
[ 4352.150627] sgtl5000 0-000a: Failed to get mclock: -2
[ 4352.225065] net eth1: phy found : id is : 0x4dd072
[ 4352.344861] PM: resume of devices complete after 274.700 msecs
[ 4352.422824] Restarting tasks ...
[ 4352.426558] usb 2-1: USB disconnect, device number 2
[ 4352.454743] done
```

3.11.17 GPIO Testing

Below steps show to test GPIO0_19 and GPIO2_0 of J7

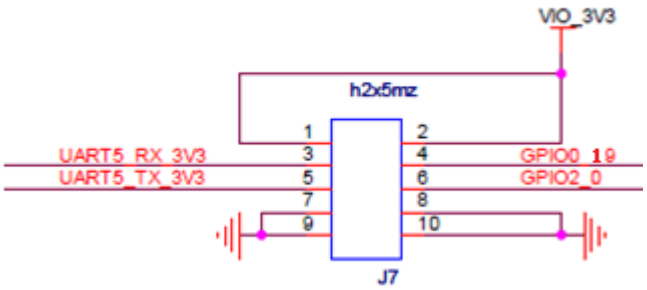


Figure 3-12 J7 Schematic

1) Export GPIO directory, once only

Export GPIO0_19, will create directory /sys/class/gpio/gpio19

Execute the following instructions to install a client;

- **root@arm:~# echo $((32 * 0 + 19)) > /sys/class/gpio/export**

Export GPIO2_0 , will create directory /sys/class/gpio/gpio64

- **root@arm:~# echo $((32 * 2 + 0)) > /sys/class/gpio/export**

2) Output level to GPIO0_19

Set GPIO0_19 to output mode

- **root@arm:~# echo out > /sys/class/gpio/gpio19/direction**

Output high level

- **root@arm:~# echo 1 > /sys/class/gpio/gpio19/direction**

Output low level

- **root@arm:~# echo 0 > /sys/class/gpio/gpio19/direction**

3) Set GPIO0_19 to input mode

- **root@arm:~# echo in > /sys/class/gpio/gpio19/direction**

Then read input level of GPIO0_19

- **root@arm:~# cat /sys/class/gpio/gpio19/value**

Note:

🕮 To test GPIO2_0, replace gpio19 in the path above with gpio64.

3.11.18 Debian Configuration

SBC8600B provides Debian 8 system.

1. Network Configuration

Table 3-6 Net Config

Config File	/etc/network/interfaces
DHCP	iface eth0 inet dhcp
Static	iface eth0 inet static hwaddress ether d6:6e:0d:6e:2e:6d address 192.168.1.210 netmask 255.255.255.0

Manual DHCP：

- root@arm:~# **dhclient -v eth0**

Manual config of static IP：

- root@arm:~# **ifconfig eth0 192.168.1.210**

Note

🕮 DHCP is close by default, due to notable increment of system startup time when no valid network connection exists. Please modify the config file to open DHCP.

2. Time zone Configuration

System use UTC time by default. Follow below steps to change Timezone to ‘Shanghai’.

- root@arm:~# **echo "Asia/Shanghai" > /etc/timezone**
- root@arm:~# **ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime**

3. Auto starting application

Follow below steps to make the application /home/userProgram auto starting and restart system when all of them finished.

A creating a service file

- root@arm:~# **vi /etc/systemd/system/user.service**

```
[Unit]
Description=user program
Documentation=user
After=network.target remote-fs.target

[Service]
Type=forking
PIDFile=/run/user.pid
ExecStart=/home/userPrgram
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s QUIT $MAINPID
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

B Enabling the service

- root@arm:~# **systemctl enable user.service; sync**

The command will create a symlink

etc/systemd/system/multi-user.target.wants/user.service

linked from /etc/systemd/system/user.service.

C Restart system

4. Disable LCD cursor blinking

LCD cursor will blink by default. To disable the blinking, follow below instruction.

- root@arm:~# **echo 0 > /sys/class/graphics/fbcon/cursor_blink**

Go back to blinking state.

- root@arm:~# **echo 1 > /sys/class/graphics/fbcon/cursor_blink**

3.11.19 TISDK System Demonstration

1. Program image info TF card

The TISDK sytem can be started with a TF card, and there are two methods to program image into the TF card.

Method A - Program image into the TF card through commands

Format a TF card as two partitions (please refer to Appendix 3) and connect it to your PC with a TF card reader, and then execute the following instructions in Ubuntu Linux system;

- **cp /media/cdrom/linux/demo/tisdk/image/MLO /media/LABEL1**
- **cp /media/cdrom/linux/demo/tisdk/image/u-boot.img /media/LABEL1**
- **cp /media/cdrom/linux/demo/tisdk/image/zImage /media/LABEL1/**
- **rm -rf /media/LABEL2/***
- **sudo tar xvf**
- **/media/cdrom/linux/demo/tisdk/image/tisdk-rootfs-am335x-evm.tar.gz -C /media/LABEL2**
- **sync**
- **umount /media/LABEL1**
- **umount /media/LABEL2**

Method B - Program image into the TF card through Win32DiskImager.exe

1) Connect a TF card to your PC by using a MciroSD card adapter or a USB flash card reader

2) Uncompress the **win32diskimager-v0.7-binary.zip** under CD/linux/tools/, then run **Win32DiskImager.exe** on your pc windows system.

It might take about 2 minutes before the tool's interface appears on your desktop as shown blow

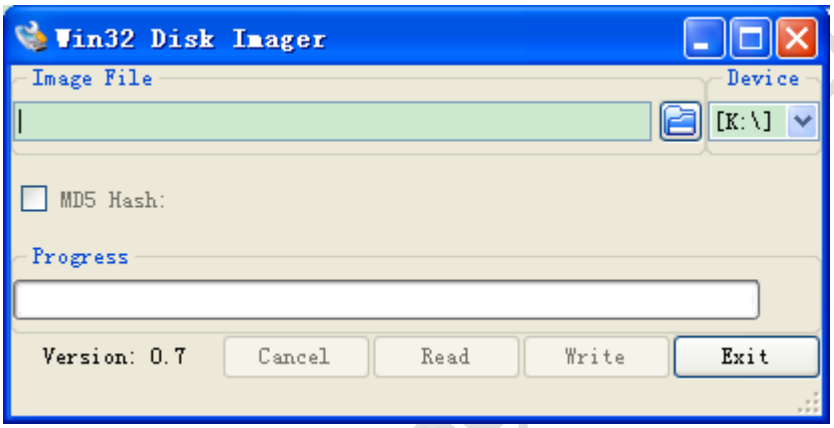
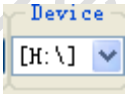


Figure 3-13 Win32 Disk Imager Tool

3) Click the button  to select the image files CD/linux/demo/tisdk/image/am335x-tisdk.img, and then select the TF card in the drop-down menu 

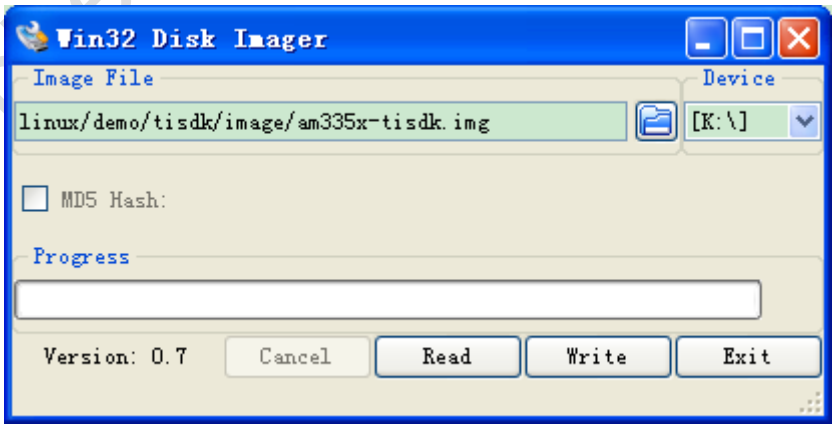


Figure 3-14

Click “Write” and then “Yes” when a warning window pops up. The programming will take a while. The following window will appear when it is finished.

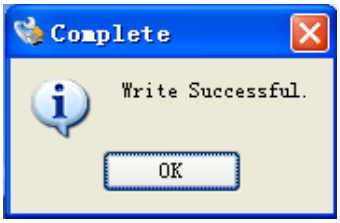


Figure 3-15 Programming Finished

2. Startup TISDK

1） Insert the TF card on SBC8600B and short the jumper JP5, then power it on; the information in HyperTerminal window is shown below;

```
CCCCCCCC
U-Boot SPL 2011.09-svn55 (Dec 04 2012 - 09:33:23)
Texas Instruments Revision detection unimplemented
Booting from MMC...
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img



U-Boot 2011.09-svn55 (Dec 04 2012 - 09:33:23)

I2C:   ready
DRAM:  512 MiB
WARNING: Caches not enabled
Did not find a recognized configuration, assuming General purpose EVM in Profile 0
with Daughter board
NAND:  HW ECC Hamming Code selected
512 MiB
MMC:   OMAP SD/MMC: 0
*** Warning - bad CRC, using default environment

Net:   cpsw
Hit any key to stop autoboot:  0
```

```
Booting from dvsdk ...
reading zImage

3175384 bytes read
## Booting kernel from Legacy Image at 80007fc0 ...
    Image Name:    Linux-3.2.0
    Image Type:    ARM Linux Kernel Image (uncompressed)
    Data Size:     3175320 Bytes = 3 MiB
    Load Address: 80008000
    Entry Point:   80008000
    Verifying Checksum ... OK
    XIP Kernel Image ... OK
OK
Starting kernel ...
……                    //Omitted part
Arago Project http://arago-project.org am335x-evm ttyO0

Arago 2011.09 am335x-evm ttyO0

am335x-evm login: root      //Type root to log in
```

Type user name **root** to log in the system when you see prompt information **am335x-evm login** in HyperTerminal window; TISDK filesystem includes some pre-installed applications which can be run under QT graphical interface.

Note:

- By default, the system support 4.3-inch screen. If you need to select another display mode, please refer to 3.10 Display Mode Configurations.
- The profile file sbc8600_tisdk.h is located in u-boot-2015.07/include/configs/, if you want to geraruation TISDK u-boot, the following instructions should be Executed
- make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- sbc8600_tisdk_config
- make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-

3.12 Development of Applications

This section will introduce the common process of development applications through several examples.

3.12.1 Development of LED Applications

Example Application

1） Compose source code led_acc.c to instruct the two LEDs on SBC8600B to blink in the mode of accumulator;

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/ioctl.h>
#include <fcntl.h>

#define LED1 "/sys/class/leds/sys_led/brightness"
#define LED2 "/sys/class/leds/user_led/brightness"

int main(int argc, char *argv[])
{
        int f_led1, f_led2;
        unsigned char i = 0;
        unsigned char dat1, dat2;
        if((f_led1 = open(LED1, O_RDWR)) < 0){
                printf("error in open %s",LED1);
                return -1;
        }
        if((f_led2 = open(LED2, O_RDWR)) < 0){
                printf("error in open %s",LED2);
                return -1;
        }
        for(;;){
                i++;
                dat1 = i&0x1 ? '1':'0';
                dat2 = (i&0x2)>>1 ? '1':'0';
                write(f_led1, &dat1, sizeof(dat1));
                write(f_led2, &dat2, sizeof(dat2));
                usleep(300000);
        }
}
```

2） Execute the following instruction in Ubuntu Linux system to implement cross

compilation;

- **arm-linux-gcc led_acc.c -o led_acc**

3） Download the compiled files to SBC8600B and enter the directory where the file led_acc is saved, and then execute the following instruction to run LED application;

- **./led_acc &**

3.12.2 Development of CAN Applications

Example Application

1） Defining Data to Be Sent;

The syntax for CAN frame definition is “<can_id>#{R|data}”; CAN_ID could be a 3-bit (standard frame) or 8-bit (extended frame) hexadecimal format; Data could be a 0 to 8-bit hexadecimal format (can be separated with separators “.”).

Often used syntax:

```
char *cmd_str = “123#1122334455667788”;
char *cmd_str = “123#11.22.33.44.55.66.77.88”
char *cmd_str = “12345678#112233”
```

2） Creating Sockets;

Before CAN network is ready to function, you need to create a socket first. SocketCAN introduces a new protocol family, so it is necessary to include PF_CAN as a parameter when calling the function socket(). Currently there are two CAN protocols available, one is Raw Socket protocol, the other is BCM (Broadcast Manager).

For example:

```
s = socket(PF_CAN, SOCK_RAW, CAN_RAW);
```

After successfully creating a socket, you typically nened to use the function bind() to create a binding between the socket and a CAN interface. After binding (CAN_RAW) or

connecting (CAN_BCM) socket, you can use read()/write() on the socket. The basic CAN frame and socket address struct are defined in can.h under /include/linux/.

For example:

```
/**
 * struct can_frame - basic CAN frame structure
 * @can_id:   the CAN ID of the frame and CAN_*_FLAG flags, see above.
 * @can_dlc: the data length field of the CAN frame
 * @data:     the CAN frame payload.
 */
struct can_frame {
        canid_t can_id;   /* 32 bit CAN_ID + EFF/RTR/ERR flags */
        __u8    can_dlc; /* data length code: 0 .. 8 */
        __u8    data[8] __attribute__((aligned(8)));
};
```

The valid data of struct is included in data[] array with 64-bit byte alignment, so users can easily transmit their custom struct and union with data[]. There is no default byte sequence on CAN bus. Calling read() over the socket CAN_RAW would return a can_frame struct to user space. sockaddr_can struct includes an index which is bound to specific interfaces.

For example:

```
/**
 * struct sockaddr_can - the sockaddr structure for CAN sockets
 * @can_family:   address family number AF_CAN.
 * @can_ifindex: CAN network interface index.
 * @can_addr:     protocol specific address information
 */
struct sockaddr_can {
        sa_family_t can_family;
        int             can_ifindex;
        union {
                /* transport protocol class address information (e.g. ISOTP)
*/
                struct { canid_t rx_id, tx_id; } tp;
                /* reserved for future CAN protocols address information */
        } can_addr;
```

```
};
```

3） Specifying Interface Index and create binding;

In order to band sockets with all the CAN interfaces, the function ioctl() needs to be called when specifying an interface index; Interface index has to 0 so that sockets can receive CAN frames on all the enabled CAN interfaces.

For example:

```
int s;
struct sockaddr_can addr;
struct ifreq ifr;

s = socket(PF_CAN, SOCK_RAW, CAN_RAW);

strcpy(ifr.ifr_name, "can0" );
ioctl(s, SIOCGIFINDEX, &ifr);

addr.can_family = AF_CAN;
addr.can_ifindex = ifr.ifr_ifindex;

bind(s, (struct sockaddr *)&addr, sizeof(addr));
```

4） Reading CAN frames from Sockets;

```
struct can_frame frame;

nbytes = read(s, &frame, sizeof(struct can_frame));

if (nbytes < 0) {
perror("can raw socket read");
return 1;
}

/* paranoid check ... */
if (nbytes < sizeof(struct can_frame)) {
fprintf(stderr, "read: incomplete CAN frame\n");
return 1;
}
```

5） Writing CAN frames on Sockets;

```
struct can_frame frame;

nbytes = read(s, &frame, sizeof(struct can_frame));

if (nbytes < 0) {
perror("can raw socket read");
return 1;
}

/* paranoid check ... */
if (nbytes < sizeof(struct can_frame)) {
fprintf(stderr, "read: incomplete CAN frame\n");
return 1;
}
```

6） Enabling CAN Interfaces;

```
int can_do_start(const char *name)
{
        return set_link(name, IF_UP, NULL);
}
```

7） Disabling CAN Interfaces;

```
int can_do_stop(const char *name)
{
        return set_link(name, IF_DOWN, NULL);
}
```

Example Applications

1） CAN applications are saved under can_test directory which contains three source code files - can_test.c, lib.c and libsocketcan.c; lib.c defines character conversion function; libsocketcan.c defines CAN interface function; The following tables provide part of source code;

lib.c

```
int parse_canframe(char *cs, struct can_frame *cf);
/*Transfers a valid ASCII string decribing a CAN frame into struct can_frame*/
```

libsocketcan.c

```
int can_do_start(const char *name) /*start the can interface*/
int can_do_stop(const char *name) /*stop the can interface*/
```

can_test.c

```
#define MAX_CANFRAME          "12345678#01.23.45.67.89.AB.CD.EF"
#define MAX_LONG_CANFRAME "12345678    [8] 10101010 10101010
10101010 10101010 10101010 10101010 10101010 10101010     '........'"
static int s = -1;
char buf[sizeof(MAX_LONG_CANFRAME)+1]="";
char *cmd_str = "111#1122334455667788";//Define the message to be sent

int main(void)
{
        struct sockaddr_can addr;
        static struct ifreq ifr;
        const char* name = argv[1];

        if ((argc < 2) || !strcmp(argv[1], "--help"))
                help();

        if (argc < 3)
                cmd_show_interface(name);

        cmd_stop(argc, argv, name);    // can stop

        while (argc-- > 0) {
                if (!strcmp(argv[0], "bitrate"))
                        cmd_bitrate(argc, argv, name);
                if (!strcmp(argv[0], "ctrlmode"))
                        cmd_ctrlmode(argc, argv, name);
                if (!strcmp(argv[0], "start"))
                        cmd_start(argc, argv, name);
                if (!strcmp(argv[0], "stop"))
                        cmd_stop(argc, argv, name);
                argv++;
```

```
        }
        cmd_start(argc, argv, name); // can start
        if (s != -1) {
                return 0;
        }
        s = socket(PF_CAN, SOCK_RAW, CAN_RAW);
        if(s < 0) {
                perror("socket");
                return 1;
        }
        memset(&ifr.ifr_name, 0, sizeof(ifr.ifr_name));
        strcpy(ifr.ifr_name, "can0");
        if(ioctl(s, SIOCGIFINDEX, &ifr) < 0) {
                perror("SIOCGIFINDEX");
                exit(1);
        }

        addr.can_family = AF_CAN;
        addr.can_ifindex = ifr.ifr_ifindex;

        if(bind(s, (struct sockaddr *)&addr, sizeof(addr)) < 0) {
                perror("bind");
                return 1;
        }

        funct_select();
}
Sending message：
int can_send(char *buf)
{
        int nbytes;
        struct can_frame frame;

        if(parse_canframe(buf, &frame)) {
                fprintf(stderr, "\nWrong CAN-frame format!\n\n");
                fprintf(stderr, "Try: <can_id>#{R|data}\n");
                fprintf(stderr, "can_id can have 3 (SFF) or 8 (EFF) hex
chars\n");
                fprintf(stderr, "data has 0 to 8 hex-values that can
(optionally)");
                fprintf(stderr, " be seperated by '.'\n\n");
```

```
                fprintf(stderr, "e.g. 5A1#11.2233.44556677.88 /
123#DEADBEEF / ");
                fprintf(stderr, "5AA# /\n      1F334455#1122334455667788 /
123#R ");
                fprintf(stderr, "for remote transmission request.\n\n");
                return 1;
        }

        if((nbytes = write(s, &frame, sizeof(frame)))
             != sizeof(frame)) {
                perror("write");
                return 1;
        }

        return 0;
}
Receive Information：
int can_recv(char *buf)
{
        int ret, nbytes;
        fd_set rdfs;
        struct can_frame frame;

        FD_ZERO(&rdfs);
        FD_SET(s, &rdfs);

        if((ret = select(s+1, &rdfs, NULL, NULL, NULL)) < 0) {
                perror("select");
                exit(1);
        }
        if(FD_ISSET(s, &rdfs)) {
                nbytes = read(s, &frame, sizeof(struct can_frame));
                if(nbytes < 0) {
                        perror("read");
                        return 1;
                }
                if(nbytes < sizeof(struct can_frame)) {
                        fprintf(stderr, "read: incomplete CAN frame\n");
                        return 1;
                }
                sprint_long_canframe(buf, &frame, 0);
```

```
                printf("%s\n", buf);
        }
}
```

2） Execute the following instructions in Ubuntu system to copy can_test.tar.bz2 from \linux\example\ to \work and then compile the file;

- **cd $HOME/work**
- **tar xvf /media/cdrom/linux/example/can_test.tar.bz2**
- **cd can_test**
- **make**

3） Download the compiled file to SBC8600B, and then enter the directory where can_test is saved and execute the following instruction to run CAN application;

- root@arm:~# **./can_test can0 bitrate 125000 ctrlmode triple-sampling on**

Note:

CAN applications need to be running on two CAN-enabled devices; Please refer to 3.11.12 CAN Testing for details on how to connect CAN devices.

The information in HyperTerminal window is shown below;

```
can0 state: STOPPED
can0 bitrate: 125000, sample-point: 0.875
can0 ctrlmode: loopback[OFF], listen-only[OFF],
tripple-sampling[ON],one-shot[OFF], bd_can d_can: can0: setting CAN BT =
0x518
err-reporting[OFF]
can0 state: ERROR-ACTIVE

Select 1 : Send a message
Select 2 : Receive messages
>
```

The TRIPLE-SAMPLING in the above information is one of three CAN controller modes, the other two modes are LOOPBACK and LISTEN-ONLY; ERROR-ACTIVE is one of five statuses of CAN controller, the other four are ERROR-WARNING, ERROR-PASSIVE, BUS-OFF and STOPPED. Please visit http://blog.csdn.net/zhangxiaopeng0829/article/details/7646639 for more information about SocketCAN.

4） Type **1** at the transmitting end and press **Enter** key to send data, and then type **2** at the receiving end and press **Enter** key to receive data; The information in HyperTerminal window is shown below;

```
111   [8] 11 22 33 44 55 66 77 88
111   [8] 11 22 33 44 55 66 77 88
111   [8] 11 22 33 44 55 66 77 88
```

3.12.3 Development of Serial Interface Applications

The following table lists the header files required for operations on serial interfaces;

Table 3-7 Header Files Related to Serial Interfaces

Header Files	Descriptions
#include <stdio.h>	Standard input/output definition
#include <stdlib.h>	Standard function library definition
#include <unistd.h>	UNIX standard function definition
#include <sys/types.h>	
#include <sys/stat.h>	
#include <fcntl.h>	File control definition
#include <termios.h>	PPSIX terminal control definition

1） Opening Serial Interfaces;

The file for serial interfaces under Linux is saved in /dev; By using standard “Open" function, serial interface can be opened;

For example:

```
int fd;
fd = open( "/dev/ttyS0", O_RDWR); /*open serial interface by write/read
method*/
if (-1 == fd){
      perror(" error warning");/* can not open serial interface 1*/
}
```

2） Configuring Serial Interfaces;

Configuration of serial interfaces includes seting up baudrates, check bits, stop bits and values of struct members;

For example:

Source Code of Baudrate Setting

```
struct   termios opt;
tcgetattr(fd, &opt);
cfsetispeed(&opt,B115200);       /*set to 15200Bps*/
cfsetospeed(&opt,B115200);
tcsetattr(fd,TCANOW,&opt);
```

Table 3-8 Check Bits and Stop Bits

No Check	Odd Check
8-bit	7-bit
Option.c_cflag &= ~PARENB; Option.c_cflag &= ~CSTOPB; Option.c_cflag &= ~CSIZE; Option.c_cflag |= ~CS8;	Option.c_cflag |= ~PARENB; Option.c_cflag &= ~PARODD; Option.c_cflag &= ~CSTOPB; Option.c_cflag &= ~CSIZE; Option.c_cflag |= ~CS7;
Even Check	Space Check
7-bit	7it
Option.c_cflag &= ~PARENB; Option.c_cflag |= ~PARODD; Option.c_cflag &= ~CSTOPB; Option.c_cflag &= ~CSIZE; Option.c_cflag |= ~CS7;	Option.c_cflag &= ~PARENB; Option.c_cflag &= ~CSTOPB; Option.c_cflag &= &~CSIZE; Option.c_cflag |= CS8;

Values of Struct Members

```
struct termio {
        unsigned short   c_iflag;    /* input mode flag */
        unsigned short   c_oflag;          /* output mode flag */
        unsigned short   c_cflag;          /* control mode flag*/
        unsigned short   c_lflag;          /* local mode flags */
        unsigned char    c_line;                /* line discipline */
        unsigned char    c_cc[NCC];      /* control characters */
};
```

3） Configuring Check Functions;

Check Function Configuration

```
/**
*@brief     set serial interface data bits, stop bits and check bits
*@param    fd        type    int    opened file handle of serial interface
*@param    databits type    int data bits     value is 7 or 8
*@param    stopbits type    int stop bits     value is 1 or 2
*@param    parity    type    int    check    value is N,E,O,,S
*/
int set_Parity(int fd,int databits,int stopbits,int parity)
{
        struct termios options;
        if (tcgetattr( fd,&options)   !=   0) {
                perror("SetupSerial 1");
                return(FALSE);
        }
        options.c_cflag &= ~CSIZE;
        switch (databits) /*set data bits*/
        {
        case 7:
                options.c_cflag |= CS7;
                break;
        case 8:
                options.c_cflag |= CS8;
                break;
        default:
                fprintf(stderr,"Unsupported data size\n"); return (FALSE);
        }
switch (parity)
{
```

```
        case 'n':
        case 'N':
                options.c_cflag &= ~PARENB;     /* Clear parity enable */
                options.c_iflag &= ~INPCK;       /* Enable parity checking */
                break;
        case 'o':
        case 'O':
                options.c_cflag |= (PARODD | PARENB); /* set to odd check*/
                options.c_iflag |= INPCK;                   /* Disnable parity checking
*/
                break;
        case 'e':
        case 'E':
                options.c_cflag |= PARENB;       /* Enable parity */
                options.c_cflag &= ~PARODD;     /* change to even check*/
                options.c_iflag |= INPCK;          /* Disnable parity checking */
                break;
        case 'S':
        case 's':   /*as no parity*/
             options.c_cflag &= ~PARENB;
                options.c_cflag &= ~CSTOPB;break;
        default:
                fprintf(stderr,"Unsupported parity\n");
                return (FALSE);
        }
/* set stop bits*/
switch (stopbits)
{
        case 1:
                options.c_cflag &= ~CSTOPB;
                break;
        case 2:
                options.c_cflag |= CSTOPB;
           break;
        default:
                 fprintf(stderr,"Unsupported stop bits\n");
                 return (FALSE);
}
/* Set input parity option */
if (parity != 'n')
        options.c_iflag |= INPCK;
```

```
tcflush(fd,TCIFLUSH);
options.c_cc[VTIME] = 150; /*set timeout to 15 seconds*/
options.c_cc[VMIN] = 0; /* Update the options and do it NOW */
if (tcsetattr(fd,TCSANOW,&options) != 0)
{
        perror("SetupSerial 3");
        return (FALSE);
}
return (TRUE);
}
```

Note:
If it is not terminal development and serial interfaces are only used to transmit data, but not to process, Raw Mode can be used to realize communication. For example:
options.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG); /*Input*/
options.c_oflag &= ~OPOST; /*Output*/

4） Write/Read Serial Interafces;

After configuration is done, serial interface can be written and read as files; The read function can be used to read serial interfaces; If serial interfaces are working under Raw Mode, the characters length returned by the function is the character length received by serial interfaces; Asynchronous read can be realized by using the function fcntl or select; The following table contains the source code example of write/read operations;

Write on Serial Interfaces

```
char   buffer[1024];int      Length;int      nByte;   nByte = write(fd, buffer ,Length)
```

Read on Serial Interfaces

```
char   buff[1024];int      Len;int   readByte = read(fd,buff,Len);
```

5） The code used to close serial interfaces is as same as that used to close files;

Close Serial Interfaces

```
close(fd);
```

Example Applications

1） The following table contains main source code involved in programs for serial interfaces;

Program Example

```
int main(int argc, char *argv[])
{
        int   fd, next_option, havearg = 0;
        char *device;
        int i=0,j=0;
        int nread;              /* Read the counts of data */
        char buff[512];         /* Recvice data buffer */
        pid_t pid;
        char *xmit = "1234567890"; /* Default send data */
        int speed ;
        const char *const short_options = "hd:s:b:";

        const struct option long_options[] = {
                { "help",     0, NULL, 'h'},
                { "device", 1, NULL, 'd'},
                { "string", 1, NULL, 's'},
                { "baudrate", 1, NULL, 'b'},
                { NULL,       0, NULL, 0   }
        };

        program_name = argv[0];

        do {
                next_option = getopt_long (argc, argv, short_options,
long_options, NULL);
                switch (next_option) {
                        case 'h':
                                print_usage (stdout, 0);
                        case 'd':
                                device = optarg;
                                havearg = 1;
                                break;
```

```
                case 'b':
                        speed = atoi(optarg);
                        break;
                case 's':
                        xmit = optarg;
                        havearg = 1;
                        break;
                case -1:
                        if (havearg)   break;
                case '?':
                        print_usage (stderr, 1);
                default:
                        abort ();
        }
}while(next_option != -1);

sleep(1);
fd = OpenDev(device);
if (fd > 0) {
        set_speed(fd, speed);
} else {
        fprintf(stderr, "Error opening %s: %s\n", device, strerror(errno));
        exit(1);
}
if (set_Parity(fd,8,1,'N')== FALSE) {
        fprintf(stderr, "Set Parity Error\n");
        close(fd);
        exit(1);
}
pid = fork();
if (pid < 0) {
        fprintf(stderr, "Error in fork!\n");
} else if (pid == 0){
        while(1) {
                printf("%s SEND: %s\n",device, xmit);
                write(fd, xmit, strlen(xmit));          //cyclic write
                sleep(1);
                i++;
        }
        exit(0);
} else {
```

```
            while(1) {
                nread = read(fd, buff, sizeof(buff));//cyclic read
                if (nread > 0) {
                    buff[nread] = '\0';
                    printf("%s RECV %d total\n", device, nread);
                    printf("%s RECV: %s\n", device, buff);
                }
            }
        }
        close(fd);
        exit(0);
}
```

2） Execute the following instructions in Ubuntu system to copy uart_test.tar.bz2 from \linux\example\ to \work\ and then compile the file;

- **cd $HOME/work**
- **tar xvf /media/cdrom/linux/example/uart_test.tar.bz2**
- **cd uart_test**
- **make**

Note:
For details about how to test serial interfaces, please refer to 3.11.14 Serial Interface Testing.

Chapter 4 Android Operating System

This chapter will give you a general map of the Android software resources contained in the DVD-ROM provided along with the product, as well as a introduction to the process of Android develop, system update, functionality tests.

Note:
It is recommended refering to Appendix for details of Ubuntu Linux installation and learning about embedded Android development technology before you get started.

4.1 Development

4.1.1 Get the Source Code

Get the android source code by the following instructions:

1) Execute the following commands to get the repo tool:

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

2) Execute the following commands to initialize **repo** source code

- **$ mkdir ~/TI-Android-ICS-4.0.3-DevKit-EVM-SK-3.0.1**
- **$ cd ~/TI-Android-ICS-4.0.3-DevKit-EVM-SK-3.0.1**
- **$ repo init --repo-url=git://github.com/android/tools_repo.git -u https://github.com/embest-tech/rowboat-manifest.git -m TIOP-TI-Android-ICS-4.0.3-DevKit-EVM-SK-3.0.1.xml**

3) Execute the following command to synchronize the repo source code:

- **$ cd ~/TI-Android-ICS-4.0.3-DevKit-EVM-SK-3.0.1**

- $ repo sync

4.1.2 Compiling source code

1) Execute the following instructions to compile Android bootloader;

- cd TI-Android-ICS-4.0.3-DevKit-EVM-SK-3.0.1/u-boot
- make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- sbc8600_android_config
- make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-

After all the instructions are executed, you can find an image file named MLO and u-boot.img under current directory.

2) Execute the following instructions to generate the kernel and UBI filesystem required by Android;

- cd TI-Android-ICS-4.0.3-DevKit-EVM-SK-3.0.1/
- export PATH=$HOME/TI-Android-ICS-4.0.3-DevKit-EVM-SK-3.0.1/prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin:$PATH
- export PATH=$HOME/tools:$PATH
- make TARGET_PRODUCT=sbc8600 sgx_clean kernel_clean clean
- make TARGET_PRODUCT=sbc8600 OMAPES=4.x
- source ./build_ubi.sh sbc8600

After all the instructions are executed, you can find a **uImage** file under TI-Android-ICS-4.0.3-DevKit-EVM-SK-3.0.1/kernel/arch/arm/boot/, a **ubi.img** file under TI-Android-ICS-4.0.3-DevKit-EVM-SK-3.0.1/temp/

4.2 Demonstration of Android System

SBC8600B provides Android system demonstration, please follow below steps:

1） Copy all files under the directory \android\image of the DVD-ROM to a TF card;

2） Insert the TF card on the board and short jumper JP5, and then power on the board. The debugging tool will show the following information:

```
CCCCCCCC
U-Boot SPL 2011.09-svn55 (Dec 04 2012 - 09:36:25)
Texas Instruments Revision detection unimplemented
Booting from MMC...
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img



U-Boot 2011.09-svn55 (Nov 22 2012 - 11:35:28)

I2C:   ready
DRAM:  512 MiB
WARNING: Caches not enabled
Did not find a recognized configuration, assuming General purpose EVM in Profile 0
with Daughter board
NAND:  HW ECC Hamming Code selected
512 MiB
MMC:   OMAP SD/MMC: 0
*** Warning - bad CRC, using default environment


NAND erase.chip: device 0 whole chip
Skipping bad block at  0x03620000
Erasing at 0x1ffe0000 -- 100% complete.
OK
reading MLO

36079 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x0, size 0x8cef
 36079 bytes written: OK
reading flash-uboot.img

234620 bytes read
HW ECC BCH8 Selected
```

```
NAND write: device 0 offset 0x80000, size 0x3947c
  234620 bytes written: OK
reading uImage

2719416 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x280000, size 0x297eb8
  2719416 bytes written: OK
reading ubi.img

72744960 bytes read
SW ECC selected

NAND write: device 0 offset 0x780000, size 0x4560000
72744960 bytes written: OK
```

3） When the writing process is complete, on-board LED will be flashing. Please remove TF card and the jumper cap.

4） Power on the board again to load Android operating system;

5） U-boot configuration

The system image has a default setting for 4.3-inch LCD. You can change the settings in UBOOT according to the detailed instructions contained in 3.10 Display Mode Configurations.

Note:

🕮 If you want t the Android operation system is displayed on a VGA monitor, you need to execute the following command in uboot to disable the touch screen. Calibration program.
SBC8600# setenv calibration 0

Chapter 5 WinCE Operating System

This chapter briefly introduce the software resources contained in the DVD-ROM provided along with the board, and also give the details introduction about development of WinCE system, system update and APIs.

5.1 Software Resources

The DVD-ROM contains a number of software resources such as BSP, example projects, example applications and precompiled images, which allow you conduct the development of WinCE system and applications easily and quickly based on SBC8600.

5.1.1 Locations of Software Resources

You can find software in the DVR-ROM according to the location information listed in the table below;

Table 5-1 WinCE Software Resources

Categories	Locations
BSP	CD\WINCE700\BSP\SBC8600.rar
	CD\WINCE700\BSP\COMMON_TI_V1.rar
	CD\WINCE700\BSP\3rdParty.rar
	CD\WINCE700\BSP\PowerVR.rar
Example Projects	CD\WINCE700\project\SBC8600
Example Applications	CD\WINCE700\app\
Precompiled Images	CD\WINCE700\Image\

5.1.2 Precomiled Images and BSP

There are five precompiled images can be found in the DVD-ROM – MLO, xldnand.nb0, Ebootsd.nb0, Ebootnd.nb0 and NK.bin. Please see the following table to learn about these images;

Table 5-2 Precompile Images

Images	Descriptions
MLO	First boot loader for TF card boot-up
xldrnand.nb0	First boot loader for NAND Flash boot-up
Ebootsd.nb0	Second boot loader for TF card boot-up
Ebootnd.nb0	Second boot loader for NAND Flash boot-up
NK.bin	WinCE runtime image

The following table lists the files and their types contained in BSP;

Table 5-3 BSP

Catalog	**Item**	**Source code / binary**
X-Loader (First boot loader)	NAND	Source
	SD	Source
EBOOT (Second boot loader)	NAND	Source
	SD	source
OAL	Boot parameter	Source
	KILT(EMAC)	Source
	Serial debug	Source
	REBOOT	Source
	Watchdog	Source
	RTC	Source
	Kernel profiler	Source
	System timer	Source
	Interrupt controller	Source
	MMU	Source
Driver	NLED driver	Source

	GPIO/I2C/SPI/MCASP driver	Source
	Serial port driver	Source
	Audio driver	Source
	NAND driver	Source
	Display driver	Source
	TOUCH driver	Source
	SD/MMC/SDIO driver	Source
	EMAC driver	Source
	USB OTG driver	Source
	GPIO keyboard driver	Source
	DMA driver	Source
	Backlight driver	Source
	Battery driver	Source
	RPU driver	Source
SDK	powerVR DDK & SDK	Binary & Source

5.2 System Development

5.2.1 Installation of IDE (Integrated Development Environment)

Please install the software listed below under windows XP

1) Visual Studio 2008

2) Visual Studio 2008 SP1

3) Windows Embedded Compact 7

4) Windows Embedded Compact 7 Updates

5) ActiveSync 4.5

Note:

🕮 The DVD-ROM doesn't contain the IDE for Windows Embedded Compact 7. Please download it from http://www.microsoft.com/download/en/default.aspx.

5.2.2 Extract BSP and project files to IDE

Please follow the steps listed below:

1) Uncompress [CD\WINCE700\BSP\SBC8600.rar] to [C:\WINCE700\PLATFORM]

2) Uncompress [CD\WINCE700\BSP\COMMON_TI_V1.rar] to [C:\WINCE700\PLATFORM\COMMON\SRC\SOC]

3) Uncompress [CD\WINCE700\BSP\3rdParty.rar] to [C:\WINCE700]

4) Uncompress [CD\WINCE700\BSP\powerVR.rar] to [C:\WINCE700\public]

5) Copy [CD\WINCE700\project\SBC8600] to [C:\WINCE700\OSDesigns]

Note:

📖 The default installation directory of Windows Embedded Compact 7 is [C:\WINCE700] hereafter.

5.2.3 Sysgen & BSP Compilation

Please follow the steps listed below to build Sysgen and BSP:

Open the existing project file SBC8600.sln under [C:\WINCE700\OSDesigns\SBC8600]

Select [Build-> Build Solution] in VS2008 to start the process of sysgen and BSP compilation.

Copy the files MLO, EBOOTSD.nb0 and NK.bin under [C:\WINCE700\OSDesigns\SBC8600\SBC8600\RelDir\SBC8600_ARMV7_Release] to the TF card after compilation is done.

Insert TF card into SBC8600B and short the jumper JP5, and then power it on.

5.2.4 Introduction of Drivers

This table lists out all the drivers and the directories under which they are saved:

Table 5-4 Source Location

NLED driver	BSP\SBC8600\SRC\DRIVERS\NLED
GPIO	BSP\SBC8600\SRC\DRIVERS\GPIO BSP\COMMON_TI_V1\COMMON_TI_AMXX\GPIO
I2C	BSP\COMMON_TI_V1\COMMON_TI_AMXX\OAL\OALI2C
SPI	BSP\COMMON_TI_V1\COMMON_TI_AMXX\SPI BSP\SBC8600\SRC\DRIVERS\MCSPI
MCASP driver	BSP\COMMON_TI_V1\COMMON_TI_AMXX\MCASP
Serial port driver	BSP\COMMON_TI_V1\COMMON_TI_AMXX\SERIAL BSP\SBC8600\SRC\DRIVERS\UART
Audio driver	BSP\SBC8600\SRC\DRIVERS\WAVEDEV2
NAND driver	BSP\SBC8600\SRC\DRIVERS\BLOCK BSP\COMMON_TI_V1\COMMON_TI_AMXX\BLOCK
Display driver	BSP\COMMON_TI_V1\COMMON_TI_AMXX\DSS_Netra BSP\SBC8600\SRC\DRIVERS\DISPLAY
TOUCH driver	BSP\SBC8600\SRC\DRIVERS\TOUCH
SD/MMC/SDIO driver	BSP\SBC8600\SRC\DRIVERS\SDHC BSP\COMMON_TI_V1\COMMON_TI_AMXX\SDHC BSP\COMMON_TI_V1\COMMON_TI\SDHC
EMAC driver	BSP\COMMON_TI_V1\AM33X\CPSW3Gminiport BSP\SBC8600\SRC\DRIVERS\EMAC
USB OTG driver	BSP\SBC8600\SRC\DRIVERS\USB BSP\COMMON_TI_V1\AM33X\USB
GPIO keyboard driver	BSP\SBC8600\SRC\DRIVERS\KEYPAD
Backlight driver	BSP\SBC8600\SRC\DRIVERS\BACKLIGHT
Battery driver	BSP\SBC8600\SRC\DRIVERS\BATTERY
PRU driver	BSP\COMMON_TI_V1\AM33X\PRU BSP\SBC8600\SRC\DRIVERS\PRU
DMA driver	BSP\SBC8600\SRC\DRIVERS\EDMA BSP\COMMON_TI_V1\COMMON_TI_AMXX\EDMA

If users want to see more examples of driver development under Windows Embedded Compact 7, please refer to the reference document provided with PB7.0.

You can find the document on your PC by clicking:

Start->

All Programs->

Microsoft Visual Studio 2008->

Microsoft Visual Studio 2008 Document->

Content(C)->

Windows Embedded Compact 7->

Device Driver.

5.3 Update of System Image

SBC8600B can boot up from TF card and NAND Flash; this section will introduce two different ways of system update respectively.

5.3.1 Update of TF Card

1) Formatting TF card

HP USB Disk Storage Format Tool 2.0.6 is recommended as the formatting tool;

You can download it from here

a) Insert TF card into a card reader and then insert the reader into PC.

b) Open the HP USB Disk Storage Format Tool, the following window will appears.

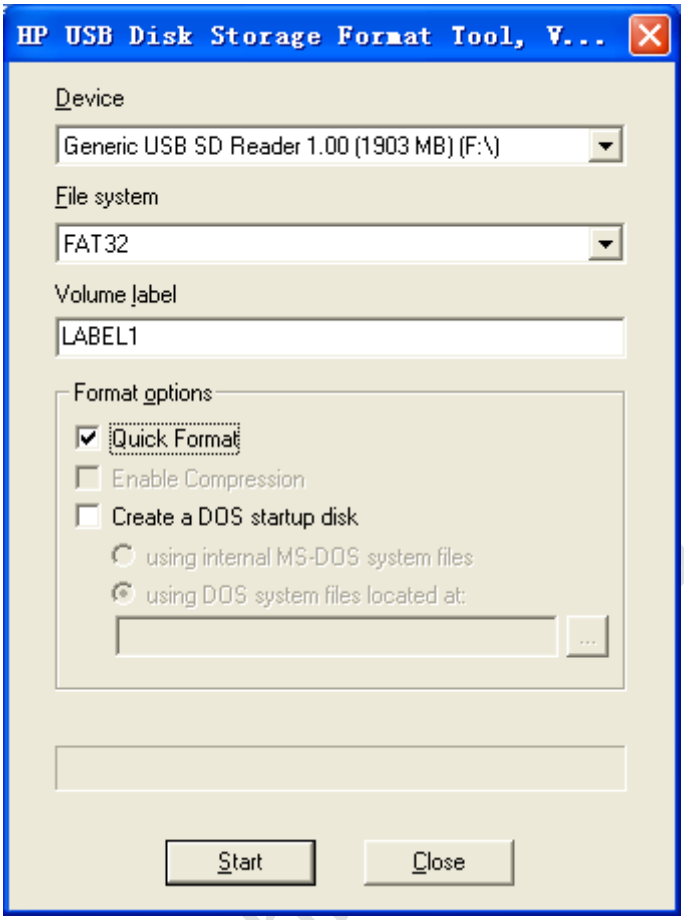


Figure 5-1 Format Tool

c) Select “FAT32” file system

d) Click “Start”

e) Click “OK” when it’s complete

Note:

- It is not recommended to use other versions of HP USB Disk Storage Format Tool.
- HP USB Disk Storage Format Tool will erase the partitions of TF card.

2) Copy runtime image

Copy **MLO, EBOOTSD.nb0 and NK.bin** image files under CD\WINCE700\image to the TF card;

3) System Boot-up

Insert TF card and short jumper **JP5**, reboot the system from TF card and press Space in a few seconds to enter to the EBOOT menu as shown below:

a) Enter EBOOT Menu

```
CCCCCCCC
Texas Instruments Windows CE SD X-Loader33X
Built Jul 27 2012 at 11:25:59
Version BSP_WINCE_ARM_A8 02.30.00.03
open ebootsd.nb0 file
Init HW: controller RST
SDCARD: requested speed 1000000, actual speed 1000000
SDCARD: requested speed 25000000, actual speed 19200000
read ebootsd.nb0 file

jumping to ebootsd image

Microsoft Windows CE Bootloader Common Library Version 1.4 Built Jul 27 2012
11:23:05
I2C EEPROM returned wrong magic value 0xffffffff
INFO:OALLogSetZones: dpCurSettings.ulZoneMask: 0x8409

Texas Instruments Windows CE EBOOT for AM33x, Built Jul 27 2012 at 11:25:53
EBOOT Version 0.0.1, BSP BSP_WINCE_ARM_A8 02.30.00.03
AHCLKX pinmux:0
AHCLKX CTRL:0x8001
pin function:0x0
pin dir:0x8000000

TI AM33X

ecc type:3
System ready!
Preparing for download...
INFO: Predownload....
Checking bootloader blocks are marked as reserved (Num = 18)

BOOT_CFG_SIGNATURE is different, read -1, expect 1111705159
WARN: Boot config wasn't found, using defaults
INFO: SW3 boot setting: 0x04
IsValidMBR: MBR sector = 0x480 (valid MBR)
OpenPartition: Partition Exists=0x1 for part 0x20.
```

```
>>> Forcing cold boot (non-persistent registry and other data will be wiped) <<<
e0311800 56e4 -> 0 18 31 e0 e4 56
e0311800 57e4 -> 0 18 31 e0 e4 57
Hit space to enter configuration menu [56] 5...(press SPACE to enter EBOOT menu)
```

b) Type [2]->[2] to set the board to boot up from TF card

```
---------------------------------------------------------------------------
  Main Menu
---------------------------------------------------------------------------
  [1] Show Current Settings
  [2] Select Boot Device
  [3] Select KITL (Debug) Device
  [4] Network Settings
  [5] SDCard Settings
  [6] Set Device ID
  [7] Save Settings
  [8] Flash Management
  [9] Enable/Disable OAL Retail Messages
  [a] Select Display Resolution
  [b] Select OPP Mode
  [0] Exit and Continue

Selection: 2


---------------------------------------------------------------------------
  Select Boot Device
---------------------------------------------------------------------------
  [1] Internal EMAC
  [2] NK from SDCard FILE
  [3] NK from NAND
  [0] Exit and Continue

Selection (actual Internal EMAC): 2
  Boot device set to NK from SDCard FILE
```

c) Type [a] to enter “Select Display Resolution” menu and select LCD\LVDS as the output

```
---------------------------------------------------------------------------
  Main Menu
---------------------------------------------------------------------------
  [1] Show Current Settings
  [2] Select Boot Device
  [3] Select KITL (Debug) Device
  [4] Network Settings
  [5] SDCard Settings
  [6] Set Device ID
  [7] Save Settings
  [8] Flash Management
  [9] Enable/Disable OAL Retail Messages
  [a] Select Display Resolution
  [b] Select OPP Mode
  [0] Exit and Continue

Selection: a


---------------------------------------------------------------------------
  Select Display Resolution
---------------------------------------------------------------------------
  [1] LCD   480x272 60Hz                    //For 4.3-inch LCD
  [2] DVI   640x480 60Hz(N/A)
  [3] DVI   640x480 72Hz(N/A)
  [4] LCD   800x480 60Hz                    //For 7-inch LCD
  [5] DVI   800x600 60Hz(N/A)               //For LVDS
  [6] DVI   800x600 56Hz(N/A)
  [7] VGA 1024x768 60Hz                     //For VGA
  [8] DVI 1280x720 60Hz(N/A)
[0] Exit and Continue Selection (actual LCD   480x272 60Hz): 4
```

d) Type [0] to continue the boot-up process

```
---------------------------------------------------------------------------
  Main Menu
---------------------------------------------------------------------------
  [1] Show Current Settings
  [2] Select Boot Device
  [3] Select KITL (Debug) Device
  [4] Network Settings
  [5] SDCard Settings
  [6] Set Device ID
  [7] Save Settings
```

```
    [8] Flash Management
    [9] Enable/Disable OAL Retail Messages
    [a] Select Display Resolution
    [b] Select OPP Mode
    [0] Exit and Continue

Selection: 0
  mode = 3
  LcdPdd_LCD_GetMode:3
  mode = 3
  LcdPdd_LCD_Initialize:3
  OEMPreDownload: Filename nk.bin
  Init HW: controller RST
  SDCARD: requested speed 1000000, actual speed 1000000
  SDCARD: requested speed 25000000, actual speed 19200000

  BL_IMAGE_TYPE_BIN

  +OEMMultiBinNotify(0x8feb24d8 -> 1)
  Download file information:
  ------------------------------------------------------------
  [0]: Address=0x80002000   Length=0x03c9e9bc   Save=0x80002000
  ------------------------------------------------------------
  Download file type: 1
  +OEMIsFlashAddr(0x80002000) g_eboot.type 1
  ..........................................................................................................
  ..........................................................................................................
  ..........................rom_offset=0x0.
  ..ImageStart = 0x80002000, ImageLength = 0x3c9e9bc, LaunchAddr = 0x8000b6a0

  Completed file(s):
  ------------------------------------------------------------------------------
  +OEMIsFlashAddr(0x80002000) g_eboot.type 1
  [0]: Address=0x80002000   Length=0x3c9e9bc   Name="" Target=RAM
  ROMHDR at Address 80002044h
  Launch Windows CE image by jumping to 0x8000b6a0...

  Windows CE Kernel for ARM (Thumb Enabled)
  CPU CP15 Control Register = 0xc5387f
  CPU CP15 Auxiliary Control Register = 0x42
  I2C EEPROM returned wrong magic value 0xffffffff
```

```
+OALTimerInit(1, 24000, 200)
--- High Performance Frequecy is 24 MHz---
```

5.3.2 Update of NAND Flash Image

a. Formatting TF card

Please refer to the contents of Update of TF Card.

b. Copy runtime image

Copy **MLO, EBOOTND.nb0, NK.bin, XLDRNAND.nb0 and EBOOTSD.nb0** image files under CD\WINCE700\image to the TF card.

c. Update of NAND Flash image files

Insert TF card and short jumper JP5, reboot the system from TF card and press Space in a few seconds to enter to the EBOOT menu, and then follow the steps listed below:

1) Type [8] to enter the Flash menu;
2) Type [9]->[4]->[A], [9]->[3]->[B] and [9]->[2]->[C] to write XLDR, EBOOT and NK images;
3) Type [0] to return to the main menu, and then type [2] and [3] to select boot-up from NAND Flash; Type [A] to select LCD/DVI display mode; Type [7] and [y] to save the boot-up settings;

Remove TF card and the jumper cap, reboot the system. The system will boot from NAND Flash.

5.4 Instructions for Use

5.4.1 How to use openGL ES demo

1) Check PowerVR in the Catalog Items View of VS2008 as shown below;

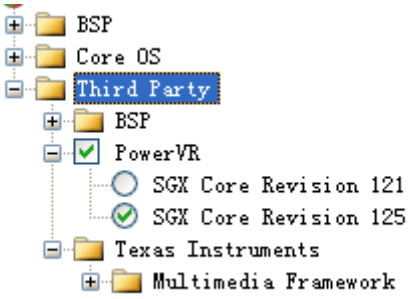


Figure 5-2 Catalog Item

2) Select [Build-> Build Solution] in the menu bar of VS2008, and then replace the nk.bin in the TF card with the newly generated nk.bin after sysgen and BSP compilation is complete.

3) Copy C:\WINCE700\PUBLIC\PowerVR\oak\target\Rev125\ARMV4I\retail*.exe to the windows embedded compact 7 system of SBC8600B, and then double-click the demo to start testing.

5.5 Application Development

This chapter introduces how to develop Windows Embedded Compact 7 applications for SBC8600B.

5.5.1 Application Interfaces and Examples

API used for development of SBC8600B applications is the standard application interface defined by Windows Embedded Compact 7. SBC8600B has extended the interface definition of GPIO based on the standard API. You can find the applications used

to control the status of GPIO pins under \WINCE700\app\GPIOAppDemo of the DVD-ROM.

Pleaes refer to the help documents for MSDN Windows Embedded Compact 7 API to learn about the definitions of Windows Embedded Compact 7 standard API.

5.5.2 GPIO Application Interfaces and Examples

GPIO application interfaces and examples:

Table 5-5 GPIO IOCTL

IOCTL code	Description
IOCTL_GPIO_SETBIT	Set GPIO pin as 1
IOCTL_GPIO_CLRBIT	Set GPIO pin as 0
IOCTL_GPIO_GETBIT	Read GPIO pin
IOCTL_GPIO_SETMODE	Set the working mode of GPIO pin
IOCTL_GPIO_GETMODE	Read the working mode of GPIO pin
IOCTL_GPIO_GETIRQ	Read the corresponding IRQ of GPIO pin

Please follow the steps listed below:

1) Enable GPIO device

```
HANDLE hFile = CreateFile (_T ("GIO1:"), (GENERIC_READ|GENERIC_WRITE),
(FILE_SHARE_READ|FILE_SHARE_WRITE), 0, OPEN_EXISTING, 0, 0);
```

2) Configure GPIO operating mode

```
DWORD id = 48, mode = GPIO_DIR_OUTPUT;
```

Configure GPIO operating mode

```
DWORD pInBuffer [2];
pInBuffer [0] = id;
pInBuffer [1] = mode;
DeviceIoControl (hFile, IOCTL_GPIO_SETMODE, pInBuffer, sizeof (pInBuffer),
NULL, 0, NULL, NULL);
```

Read mode of GPIO:

```
DeviceIoControl (hFile, IOCTL_GPIO_GETMODE, &id, sizeof(DWORD), &mode,
sizeof(DWORD), NULL, NULL);
```

"id" refers to the pin code of GPIO, "mode" refers to the mode definition of GPIO, including:

Table 5-6 GPIO Mode

Mode Definition	Description
GPIO_DIR_OUTPUT	Output mode
GPIO_DIR_INPUT	Input mode
GPIO_INT_LOW_HIGH	Rising edge trigger mode
GPIO_INT_HIGH_LOW	Falling edge trigger mode
GPIO_INT_LOW	low level trigger mode
GPIO_INT_HIGH	high level trigger mode
GPIO_DEBOUNCE_ENABLE	Jumping trigger enable

3) Output of GPIO pins

```
DWORD id = 48, pinState = 0;
```

a) High level output:

```
DeviceIoControl (hFile, IOCTL_GPIO_SETBIT, &id, sizeof (DWORD), NULL, 0,
NULL, NULL);
```

b) Low level output

```
DeviceIoControl (hFile, IOCTL_GPIO_CLRBIT, &id, sizeof (DWORD), NULL, 0,
NULL, NULL);
```

c) Read the pin state

```
DeviceIoControl (hFile, IOCTL_GPIO_GETBIT, &id, sizeof (DWORD),
&pinState, sizeof (DWORD), NULL, NULL);
```

"id" refers to the pin code of GPIO, "pin" return the pin state.

4) Other Operations

Read the corresponding IRP number of GPIO pin:

```
DWORD id = 0, irq = 0;
DeviceIoControl (hFile, IOCTL_GPIO_GETIRQ, &id, sizeof (DWORD), &irq, sizeof
(DWORD), NULL, NULL);
```

"id" refers to pin code of GPIO, "irq returns IRQ number.

5) Disable GPIO device

```
CloseHandle (hFile);
```

Note:

- Definition of GPIO pin: 0~127 MPU Bank0~3 GPIO pin.
- GPIO pins 0~127 must be configured as GPIO in bsp_padcfg.h located at SBC8600/SRC/inc/.

Appendix 1 Installing Ubuntu Linux System

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.

Setting up 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 more than 1GB memory space would be preferred.

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

Figure A-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;

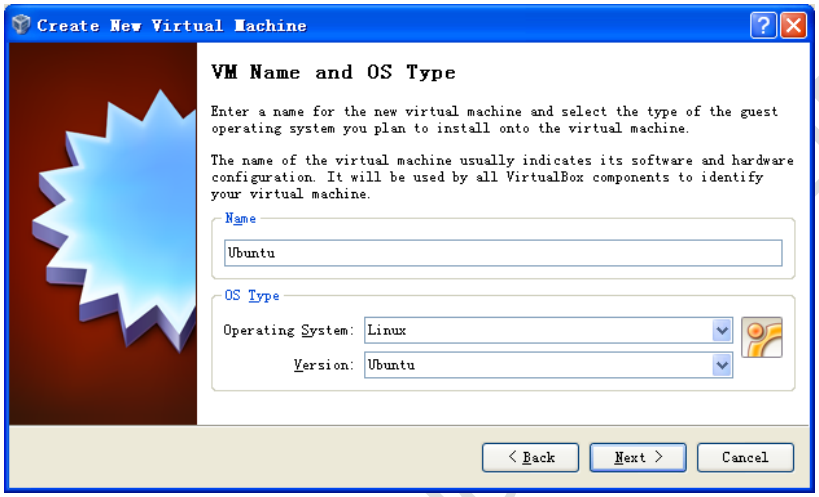


Figure A-2 Name and OS Type of Virtual Machine

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;

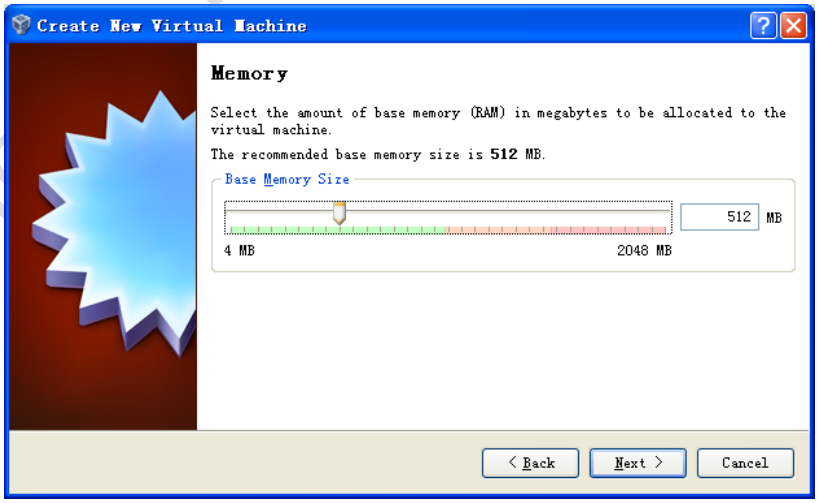


Figure A-3 Memory Allocation

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;

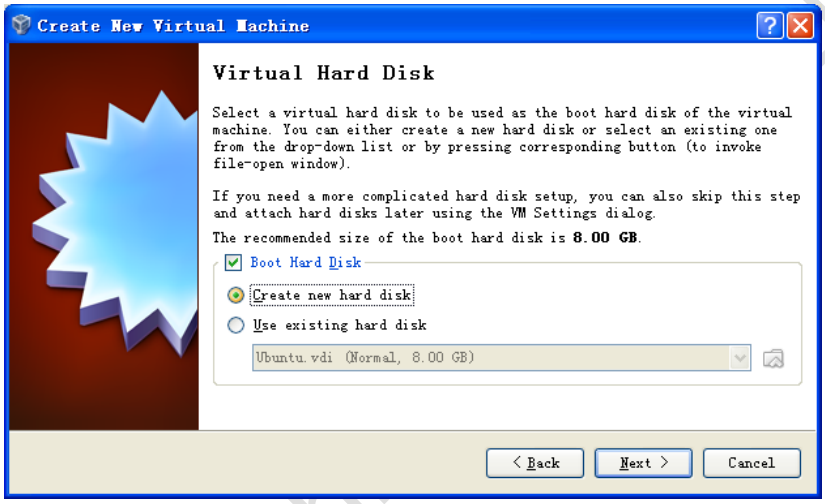


Figure A-4 Create New Hard Disk

5) Click Next in the following window;

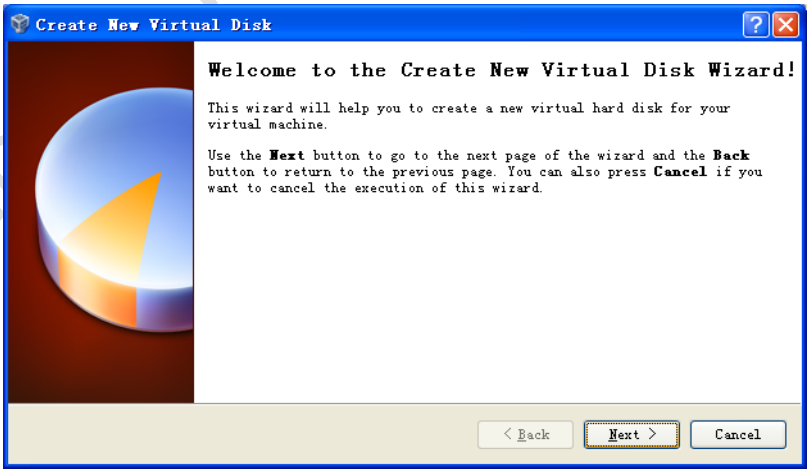


Figure A-5 Wizard of New Virtual Disk Creation

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

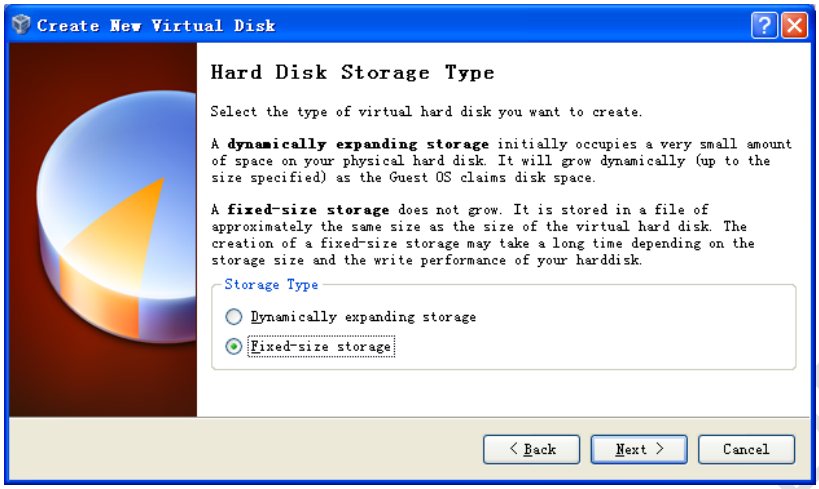


Figure A-6 Select the Second Option

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

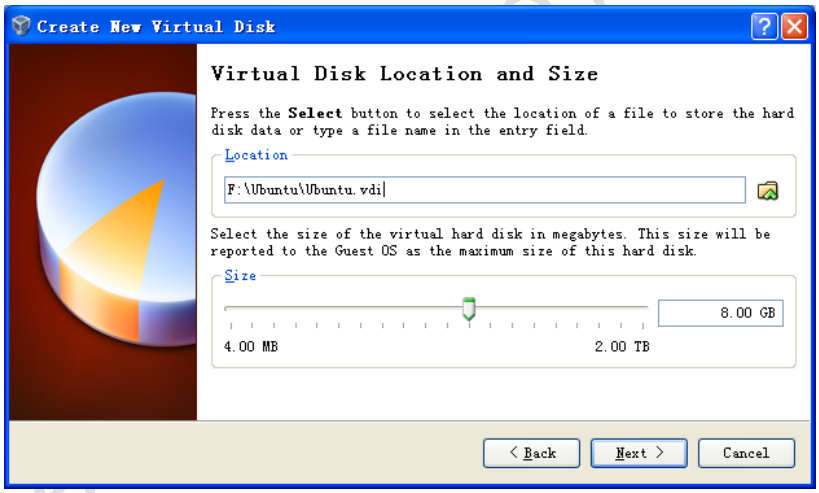


Figure A-7 Virtual Disk Configuration

8) Click Finish in the following window;

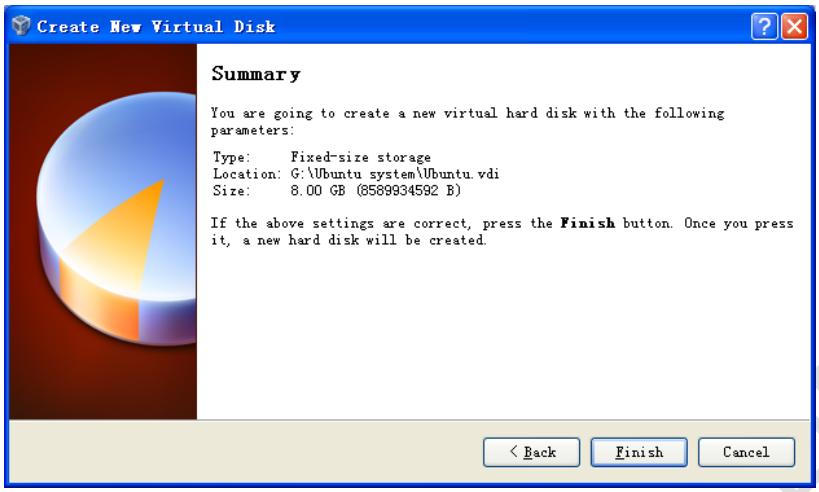


Figure A-8 Virtual Disk Summary

9) PC is creating a new virtual disk;

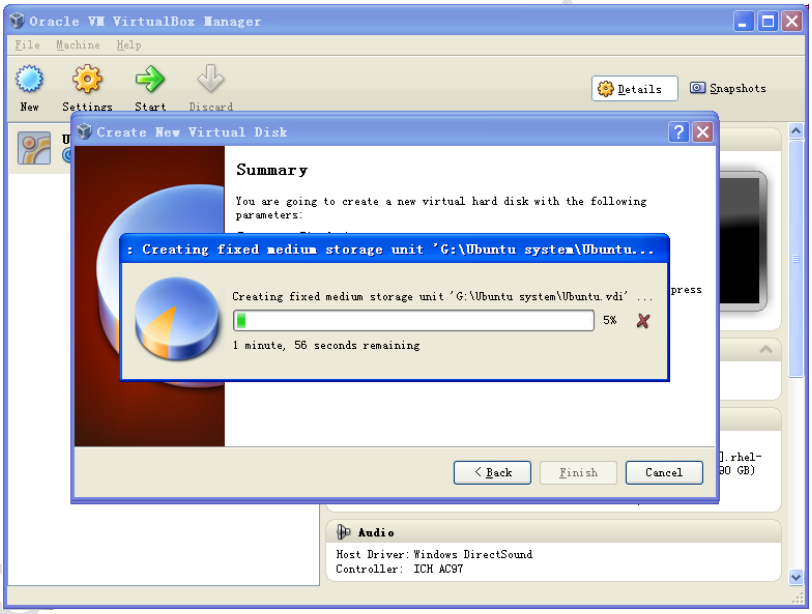


Figure A-9 Virtual Disk Creation in Process

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.

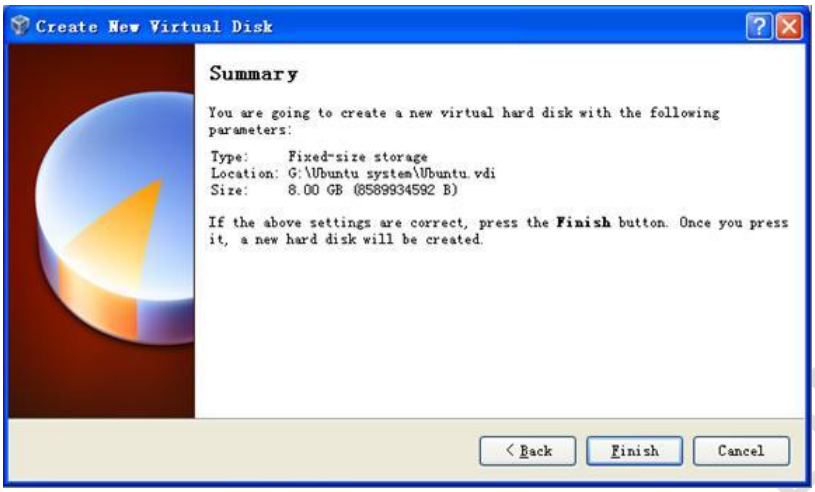


Figure A-10 Virtual Machine is Ready

Installing Ubuntu Linux System

After VirtualBox is set up, 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 listed below;

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

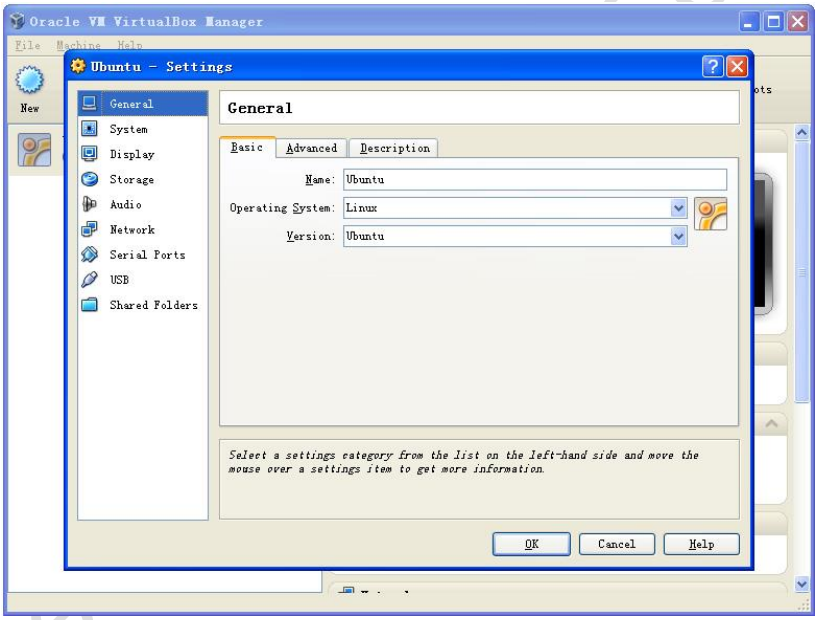


Figure A-11 Settings

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;

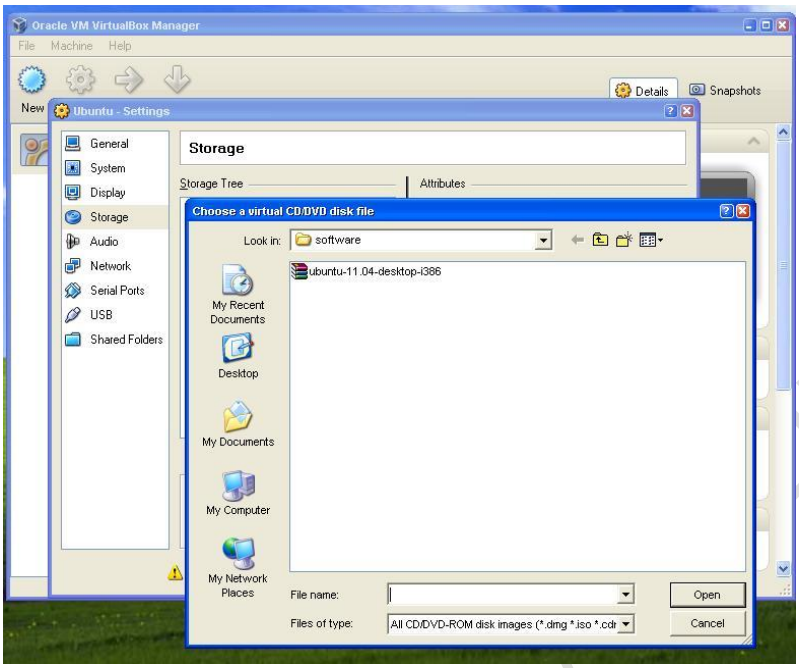


Figure A-12 Find ISO File

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

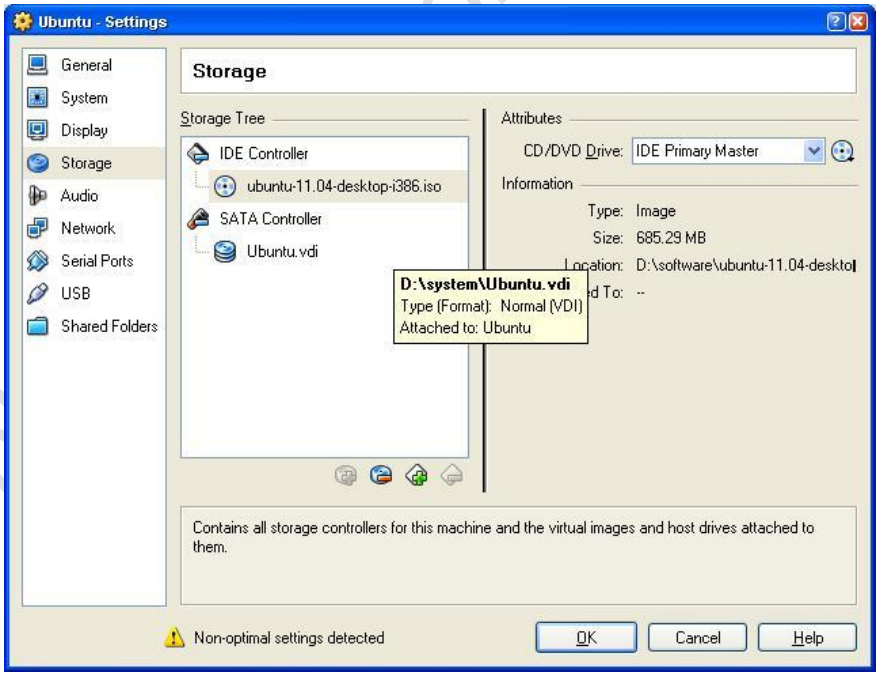


Figure A-13 Select ISO File

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

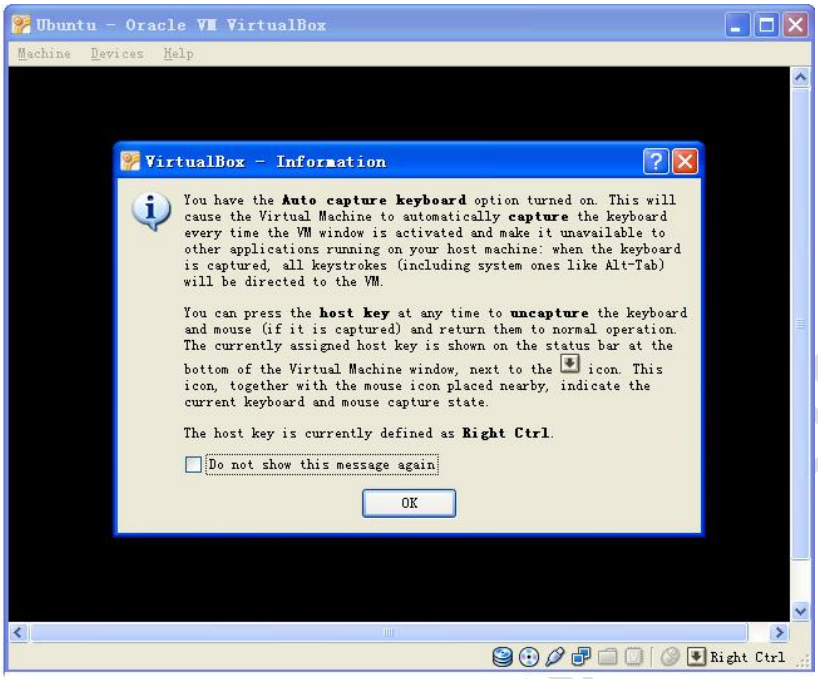


Figure A-14 Ubuntu Initiating Window

Some prompt information 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;

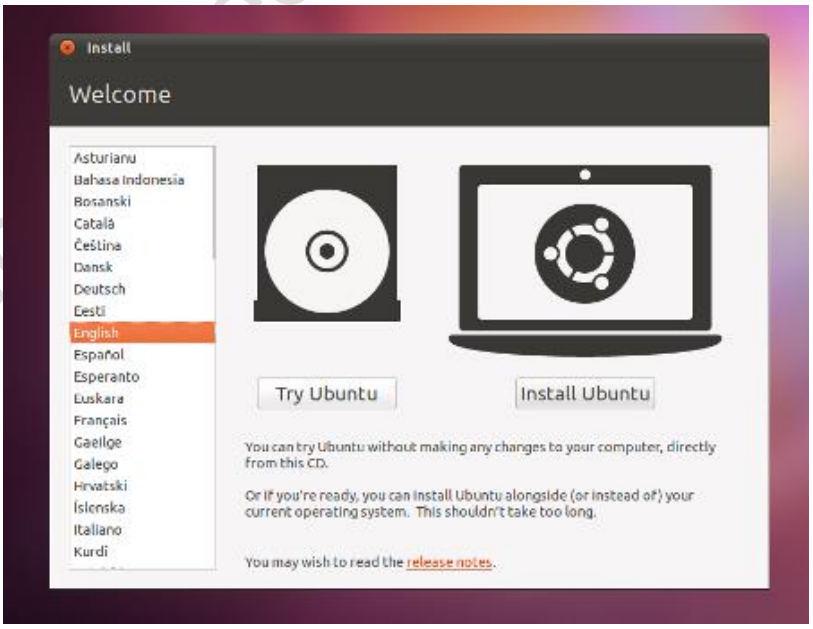


Figure A-15 Ubuntu Installation Window

6) Click Forward to continue the process;

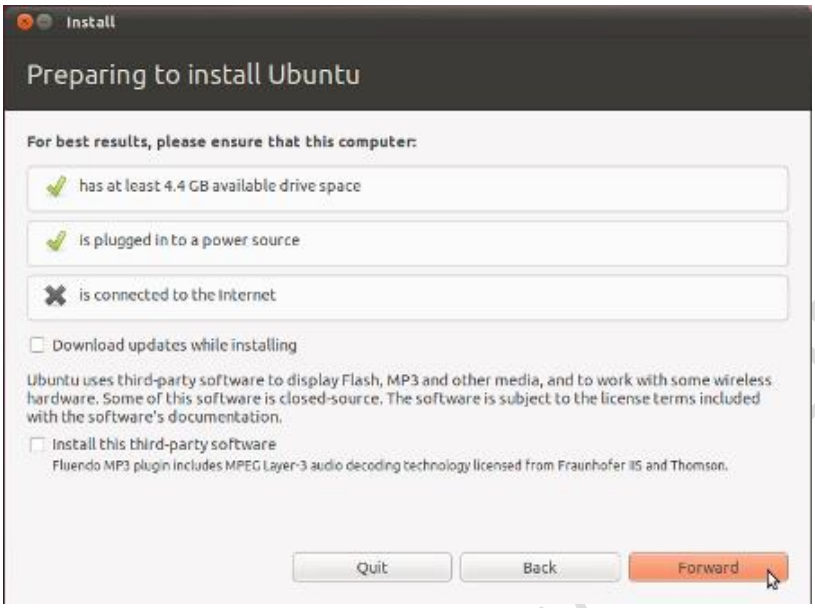


Figure A-16 Information before Installation

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

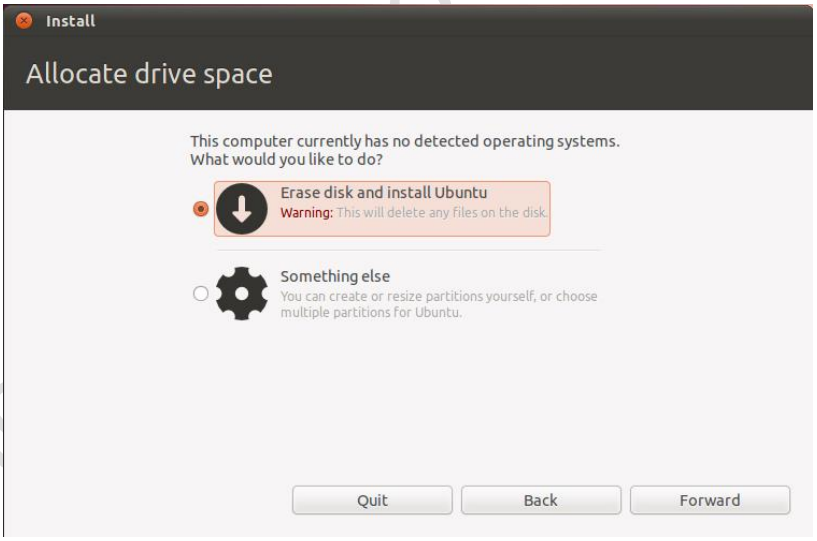


Figure A-17 Options before Installation

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;

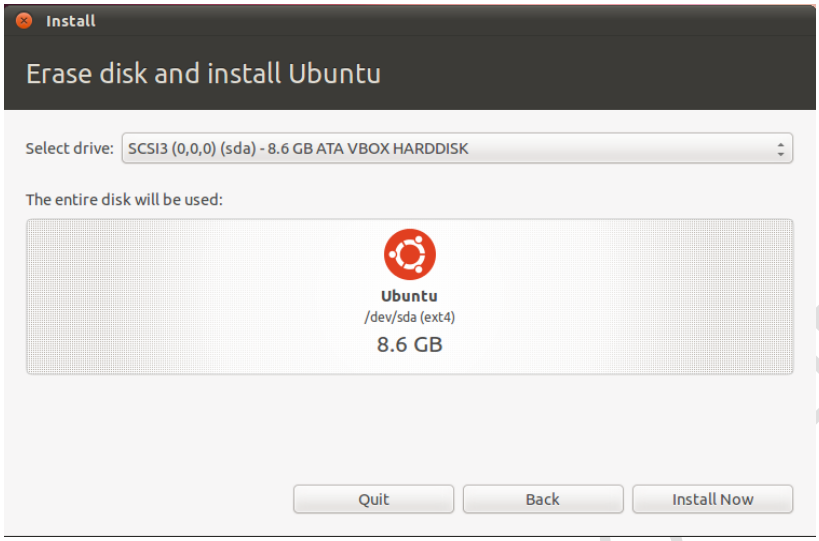


Figure A-18 Confirm Installation

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;

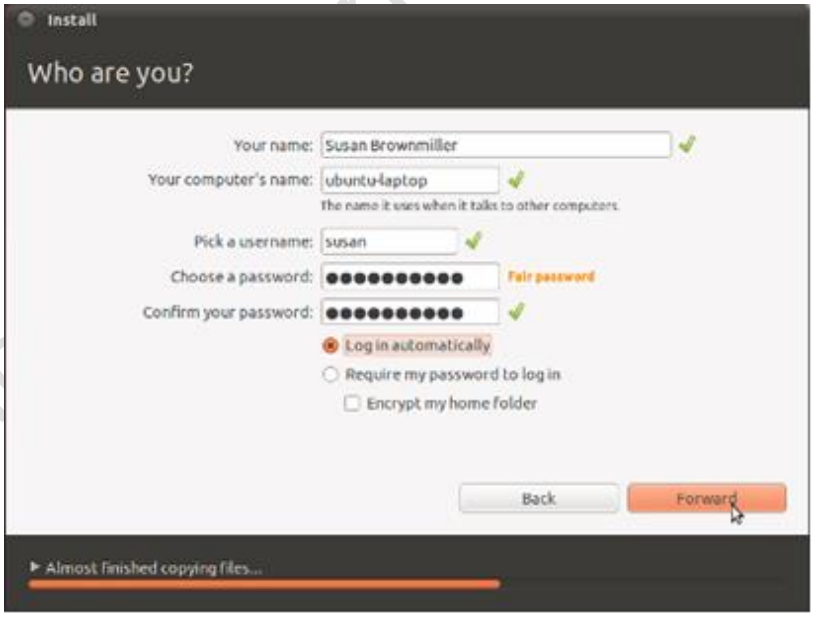


Figure A-19 Enter Appropriate Information

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.

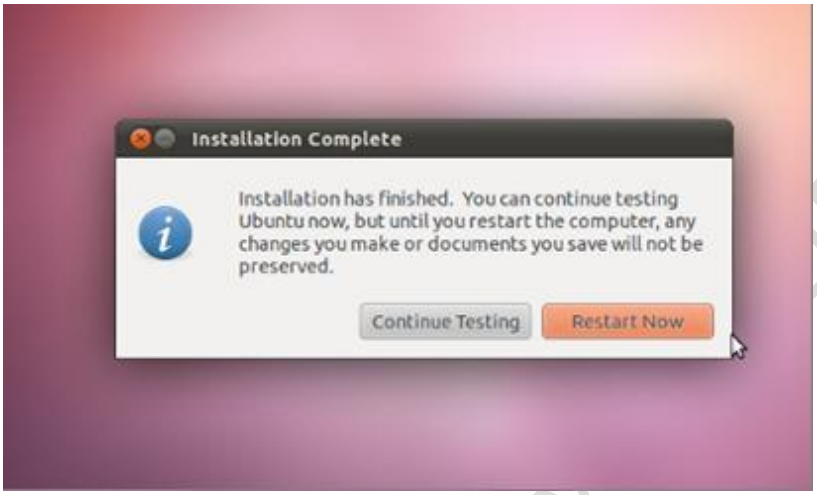


Figure A-20 Restart Ubuntu

11) Ubuntu system is ready for use after restarting. Normally the ISO file shown in **错误! 未找到引用源。** 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.

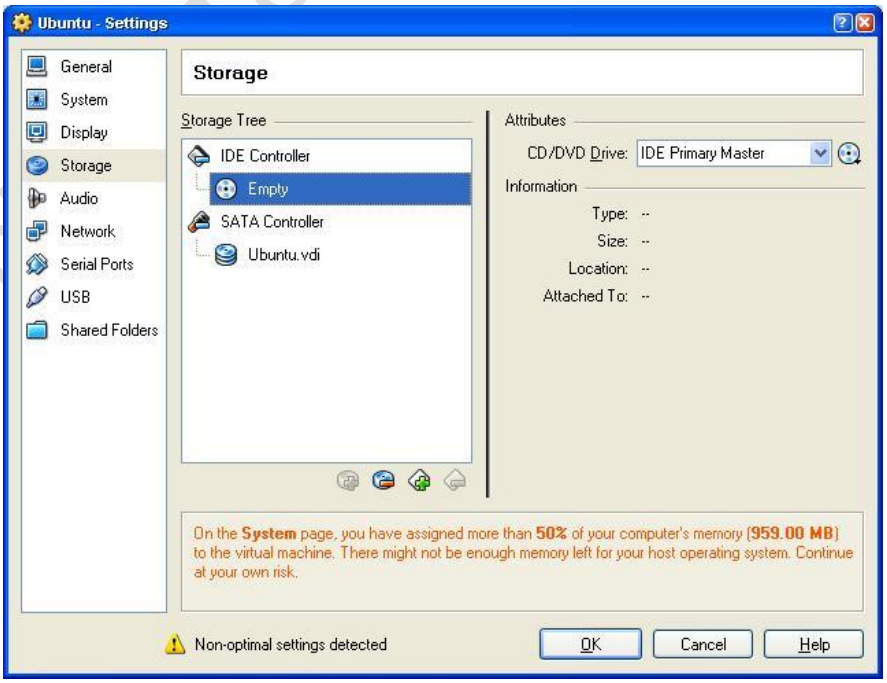


Figure A-21 ISO File Ejected

Appendix 2 Installing Linux USB Ethernet/RNDIS Gadget

1) Please select Install from a list or specific location (Advanced) in the following Found New Hardware Wizard window, and then click Next;

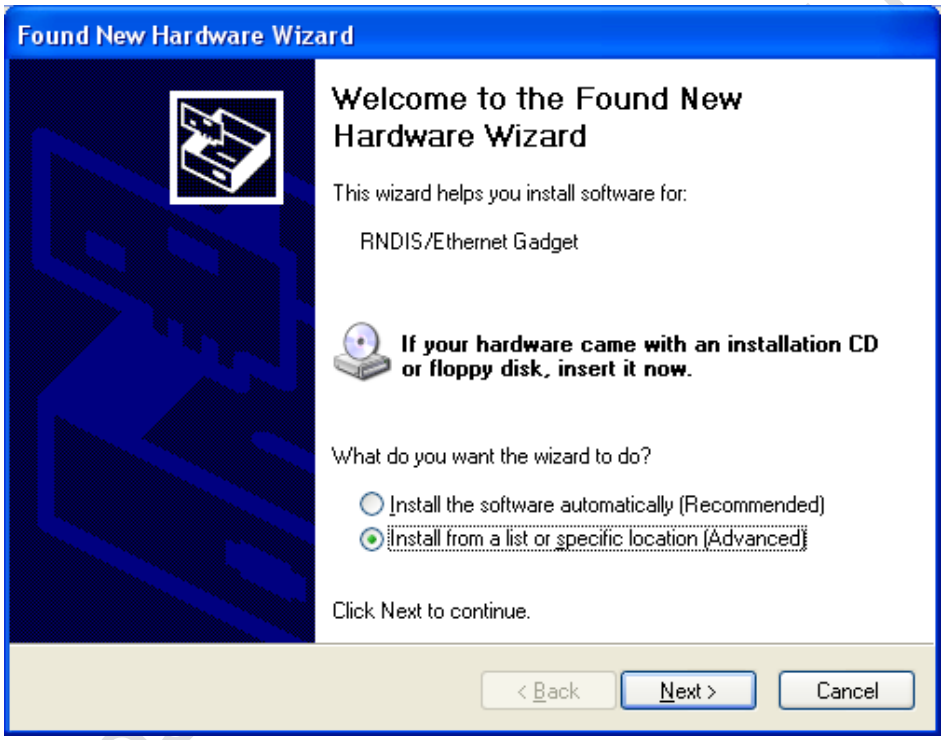


Figure A-22 Found New Hardware

2) Specify the path of USB driver as X:\linux\tools\ (X is label of DVD drive), and then click Next;

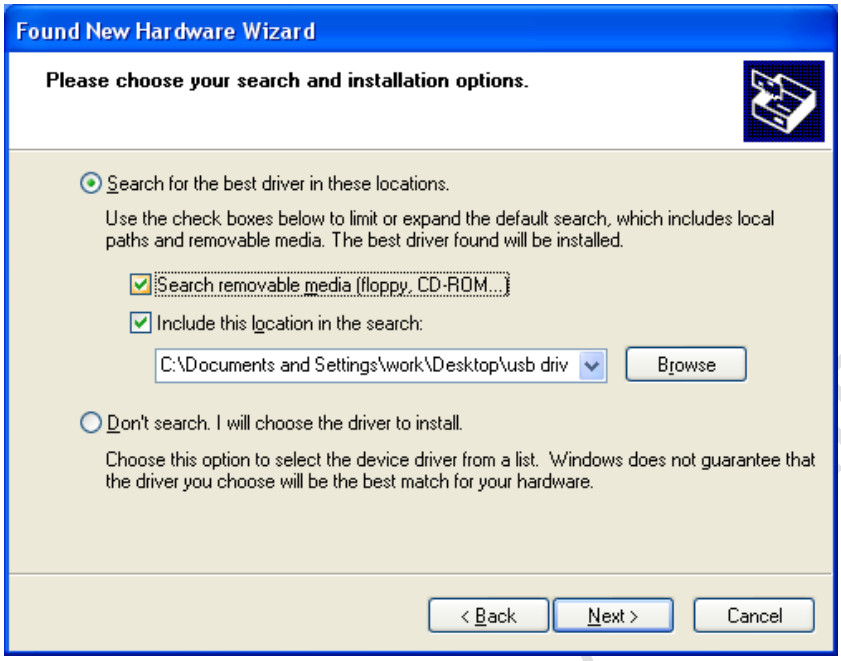


Figure A-23 Driver Location

3) Click Continue Anyway in the following window;

Figure A-24 Select Continue Anyway

4) The driver is now installed successfully when you see the following window; click Finish to exit installation wizard;

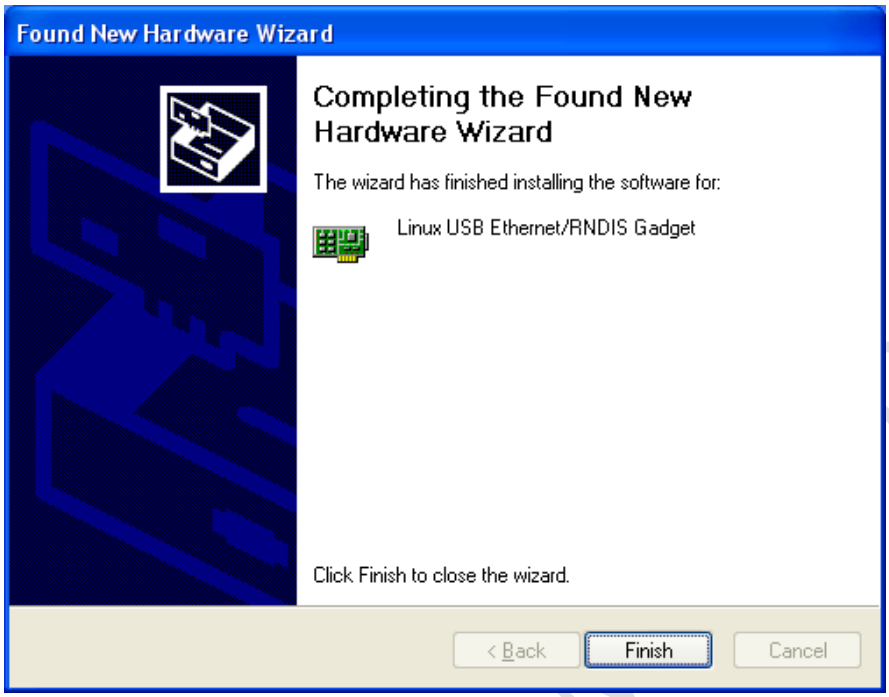


Figure A-25 Click Finish

Appendix 3 Making Linux Boot Disk

The following contents will show you how to create a dual-partition flash disk for booting up Linux system from the first partition, while saving root filesystem in the second one;

1) Insert a TF card into a TF card reader and then connect the reader to your PC; execute the following instruction in Ubuntu system to view the device name of the TF card;

- $ **dmesg | tail**

Device Information

```
...
[ 6854.215650] sd 7:0:0:0: [sdc] Mode Sense: 0b 00 00 08
[ 6854.215653] sd 7:0:0:0: [sdc] Assuming drive cache: write through
[ 6854.215659]  sdc: sdc1
[ 6854.218079] sd 7:0:0:0: [sdc] Attached SCSI removable disk
[ 6854.218135] sd 7:0:0:0: Attached scsi generic sg2 type 0
...
```

The above information shows the TF card device name is **/dev/sdc**;

2) Execute the following instruction to view the path where Ubuntu mount the device automatically;

- $ **df -h**

Device Path

```
. Filesystem            Size  Used Avail Use% Mounted on
...
/dev/sdc1               400M   94M  307M  24% /media/disk
```

At the end of the line starting with **/dev/sdc1** you can see the device path is **/media/disk**;

Note:

🕮 If TF card has two or more partitions, there would be multiple paths such as /dev/sdc1, /dev/sdc2 and /dev/sdc3 corresponding to the partitions.

3) Execute the following instruction to unmount the device;

- $ **umount /media/disk**

4) Execute fdisk instruction;

- $ **sudo fdisk /dev/sdc**

Please make sure you type the device path for the whole device, not one of the partitions such as /dev/sdc1 or /dev/sdc2;

5) After executing the above instruction, type p to print partition records of the device as shown below;

Partition Records

```
Command (m for help): [ p ]

Disk /dev/sdc: 2021 MB, 2021654528 bytes
255 heads, 63 sectors/track, 245 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1   *           1         246     1974240+   c  W95 FAT32 (LBA)
Partition 1 has different physical/logical endings:
     phys=(244, 254, 63) logical=(245, 200, 19)
```

Write down the total bytes shown in the above information, for example 2021654528 bytes, and then type d to delete all the partitions;

6) If you do not find information 255 heads and 63 sectors/track in the above table,

please go though the following steps to recover TF card;

A） Type characters as shown in the following table to set Heads and Sectors;

Set Heads and Sectors

Command (m for help): **[x]** (type x to enter expert mode)
Expert Command (m for help): **[h]** (type h to set heads)
Number of heads (1-256, default xxx): **[255]** (set heads to 255)
Expert Command (m for help): **[s]** (type s to set sectors)
Number of sectors (1-63, default xxx): **[63]** (set sector to 63)

B） Use the following equation to calculate the number of Cylinders;

Cylinders = the total bytes written down previously / 255 / 63 / 512

Type characters as shown in the following table to set Cylinders;

Set Cylinders

Expert Command (m for help): **[c]** (type c to set cylinders)
Number of cylinders (1-256, default xxx): (enter the numer of cylinders calculated above)...
Expert Command (m for help): **[r]** (type r to go back to normal mode)

7) Type p to check the parameters set just now as shown below;

Check Parameters

Command (m for help): **[p]**
63 sectors/track, 245 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device Boot Start End Blocks Id System

8) Create a FAT32 partition and transfer files from Windows according to the operations in the follow table;

Create FAT32 Boot Partition

Command (m for help): **[n]** (type n to start creating patition)
Command action

```
    e    extended
    p    primary partition (1-4)
[ p ] (type p to create primary partition)
Partition number (1-4): [ 1 ] (set the partition number to 1)
First cylinder (1-245, default 1): [ ] (press Enter key on your keyboard)
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-61, default 61): [ +5 ]  （enter +5）

Command (m for help): [ t ]  （type t）
Selected partition 1
Hex code (type L to list codes): [ c ] (type c to set partition type)
Changed system type of partition 1 to c (W95 FAT32 (LBA))
```

9) Type a and 1 to set TF card to bootable mode;

Set Bootable Mode

```
Command (m for help): [ a ]
Partition number (1-4): [ 1 ]
```

10) Type characters as shown in the following table to create a partition for root filesystem;

Create Root Filesystem Partition

```
Command (m for help): [ n ] (type n to create a partition)
Command action
    e    extended
    p    primary partition (1-4)
[ p ] (type p to select primary partition)
Partition number (1-4): [ 2 ] (set partition number to 2)
First cylinder (7-61, default 7): [ ] (press Enter key on your keyboard)
Using default value 52
Last cylinder or +size or +sizeM or +sizeK (7-61, default 61): [ ] (press Enter key)
Using default value 245
```

11) Type p to check the created partitions as shown below;

Check Partitions

```
Command (m for help): [ p ]

Disk /dev/sdc: 2021 MB, 2021654528 bytes
255 heads, 63 sectors/track, 245 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1   *           1           6      409626    c  W95 FAT32 (LBA)
/dev/sdc2               7          61     1558305   83  Linux
```

12) Type w to save new partition records as shown below;

Save Partition Records

```
Command (m for help): [ w ]
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table.
The new table will be used at the next reboot.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
```

13) Execute the following instructions to format new partitions;

- $ **[sudo mkfs.msdos -F 32 /dev/sdc1 -n LABEL1]**
- $ **[sudo mkfs.ext3 -L LABEL2 /dev/sdc2]**

Note:

- The drive labels LABEL1 and LABEL2 in the above instructions are just for referece, you can use your own labels if you want;
- After FAT and EXT3 partitions are formatted, the FAT partition needs to be formatted again under Windowns system to avoid failure when booting from TF card.

Appendix 4 Building TFTP Server

Please go through the following steps to create a TFTP server under Ubuntu Linux system;

1) Execute the following instructions to install a client;

- $>**sudo apt-get install tftp-hpa**
- $>**sudo apt-get install tftpd-hpa**

2) Execute the following instructions to install inet;

- $>**sudo apt-get install xinetd**
- $>**sudo apt-get install netkit-inetd**

3) Execute the following instructions to configure the server;

A） Create a tftpoot under root directory and change the properties to readable/writable by users;

- $>**cd /**
- $>**sudo mkdir tftpboot**
- $>**sudo chmod 777 tftpboot**

B） Execute the following instruction and add the line **tftpd dgram udp wait root /usr/sbin/in.tftpd /usr/sbin/in.tftpd -s /tftpboot**;

- $>**sudo vi /etc/inetd.conf**

C） Reload inetd process;

- $>**sudo /etc/init.d/inetd reload**

D） Execute the following instructions to create a file named **tftp**, and then add the codes contained in the following table to the file;

- $>**cd /etc/xinetd.d/**

- $**>sudo touch tftp**
- $**>sudo vi tftp**

Add Codes

```
Command (m for help): [ p ]
Disk /dev/sdc: 2021 MB, 2021654528 bytes
255 heads, 63 sectors/track, 245 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1   *           1           6      409626    c  W95 FAT32 (LBA)
/dev/sdc2               7          61     1558305   83  Linux
```

4) Execute the following instructions to restart the service;

- $**>sudo /etc/init.d/xinetd restart**
- $**>sudo in.tftpd -l /tftpboot**

5) Test TFTP server by the following operations;

A）Create a file under /tftpboot/;

- $**>touch abc**

B）Enter another directory and download the file you created just now (192.168.1.15 is the IP address of PC);

- $**>tftp 192.168.1.15**
- $**>tftp> get abc**

If the file is downloaded successfully, the server is working properly;

Appendix 5 FAQ

Please visit http://www.elinux.org/SBC8600_FAQ

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 circumstance 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

Tel: +86-755-33190868-872/875/897
Email: support@embest-tech.com

Sales Information

Tel: +86-755-33190868-863/865/866/867/868
Fax: +86-755-25616057
Email: globalsales@embest-tech.com

Company Information

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