

# SBC8600B

## Single Board Computer



# User Manual

Version 1.0 – Dec. 12, 2012

---

## Copyright Statement:

- Devkit8600B 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

---

---

# Table of Contents

<b>CHAPTER 1 PRODUCT OVERVIEW .....</b>	<b>1</b>
1.1 INTRODUCTION .....	1
1.2 HARDWARE OVERVIEW .....	1
1.2.1 Mini8600B.....	1
1.2.2 Extension Board .....	3
1.3 MODULES SUITABLE FOR THE EXTENSION BOARD .....	5
<b>CHAPTER 2 HARDWARE SYSTEM .....</b>	<b>6</b>
2.1 CPU .....	6
2.1.1 Introduction to CPU .....	6
2.1.2 CPU Features.....	6
2.2 INTRODUCTION TO PERIPHERALS .....	8
2.2.1 NAND Flash H27U4G8F2DTR-BC .....	8
2.2.2 DDR H5TQ2G83CFR-H9C .....	8
2.2.3 Ethernet AR8035 .....	8
2.2.4 MAX3232.....	9
2.3 HARDWARE INTERFACES.....	10
2.3.1 Mini8600B.....	10
2.3.2 Extension Board .....	16
<b>CHAPTER 3 LINUX OPERATING SYSTEM .....</b>	<b>25</b>
3.1 INTRODUCTION .....	25
3.2 SOFTWARE RESOURCES.....	25
3.3 SOFTWARE FEATURES .....	26
3.4 SYSTEM DEVELOPMENT.....	27
3.4.1 Establishment of development environment .....	27
3.4.2 System Compilation .....	29

---

3.4.3 System Customization.....	31
3.5 INTRODUCTION OF DRIVER.....	34
3.5.1 NAND.....	34
3.5.2 SD/MMC .....	35
3.5.3 LCDC.....	36
3.5.4 Audio in/out.....	37
3.6 DRIVER DEVELOPMENT .....	38
3.6.1 GPIO_keys Driver.....	38
3.6.2 GPIO_leds Driver .....	43
3.7 SYSTEM UPDATE .....	46
3.7.1 Update of TF Card System Image .....	46
3.7.2 Update of NAND Flash.....	50
3.8 INSTRUCTIONS .....	53
3.8.1 Selecting Display Mode.....	53
3.8.2 Testing .....	55
3.8.3 Demo .....	70
3.9 The Development of Applications.....	73
<b>CHAPTER 4 WINDOWS EMBEDDED COMPACT 7 OPERATING SYSTEM .....</b>	<b>76</b>
4.1 INTRODUCTION.....	76
4.2 SOFTWARE RESOURCES.....	76
4.3 FEATURES.....	77
4.4 SYSTEM DEVELOPMENT.....	78
4.4.1 Installation of IDE (Integrated Development Environment).....	78
4.4.2 Extract BSP and project files to IDE.....	78
4.4.3 Sysgen & BSP Compilation.....	79
4.4.4 Introduction of Drivers .....	79
4.5 UPDATE OF SYSTEM IMAGE.....	80

---

4.5.1 Update of TF Card.....	81
4.5.2 Update of NAND Flash Image.....	86
4.6 INSTRUCTIONS FOR USE .....	87
4.6.1 How to use OpenGL ES demo.....	87
4.7 APPLICATION DEVELOPMENT.....	87
4.7.1 Application Interfaces and Examples .....	88
4.7.2 GPIO Application Interfaces and Examples .....	88
<b>APPENDIX.....</b>	<b>91</b>
APPENDIX I HARDWARE DIMENSION .....	91
APPENDIX II INSTALLATION OF UBUNTU.....	93
APPENDIX III INSTALLATION OF LINUX USB ETHERNET/RNDIS GADGET .....	109
APPENDIX IV FORMATING LINUX BOOT DISK.....	112
APPENDIX V SETUP OF TFTP SERVER.....	117
APPENDIX VI FAQ.....	119
<b>TECHNICAL SUPPORT AND WARRANTY .....</b>	<b>120</b>

---

# Chapter 1 Product Overview

## 1.1 Introduction

Measuring only 60mm by 27mm, the Mini8600B processor card is a small form-factor controller board based on TI's Sitara AM3359 ARM Cortex-A8 processor. The tiny module integrates 2\*256MBytes DDR3 SDRAM and 512Mbytes NAND Flash and uses two 0.4mm space 2\*40-pin board-to-board male expansion connectors to bring out many hardware peripheral signals and GPIOs from the CPU.

Embest has designed a single board computer SBC8600B which has an expansion board to carry the Mini8600B. The flexible design allows the fast and easy way of realizing and upgrading the controller's capabilities. In addition to those features offered by Mini8600B, the SBC8600B features 5 serial ports (including 2 RS232 and 3 TTL), 2 USB Host 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 for Linux 3.2.0, 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

#### Electric Features

- Working Temperature: 0 °C~ 70°C
- Working Humidity: 20% ~ 90%, Non-Condensing

- Dimensions: 60mm x 27mm
- Input Voltage: 3.3V

**Processor**

- 720-MHz 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

**Expansion Interfaces and Signals Routed to Pins**

- Two 0.4-pitch 2\*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 /1000Mb/s 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.

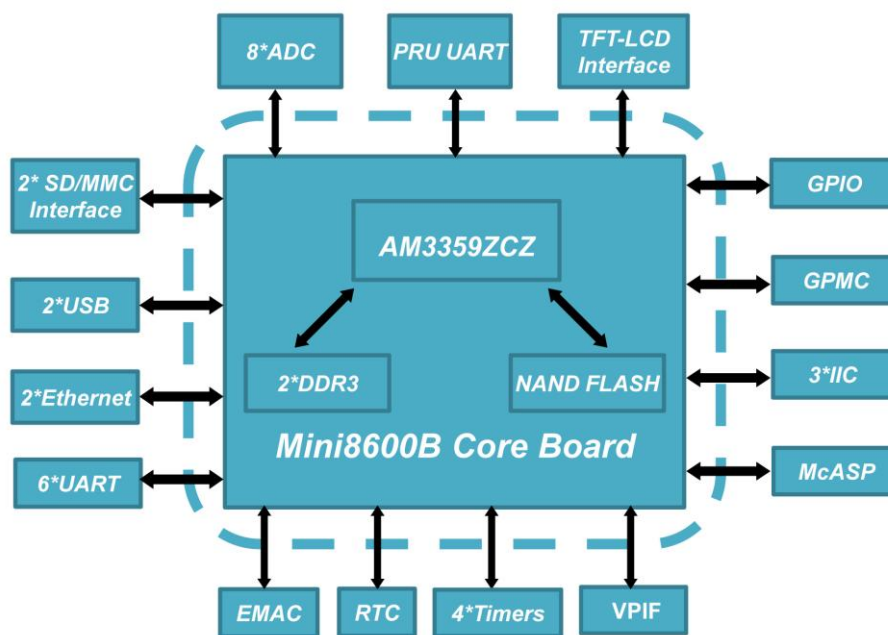


Figure 1-1 Mini8600 Structure Chart

### 1.2.2 Extension Board

**Electric Features**

- Working Temperature: 0 °C~ 70°C
- Working Humidity: 20% ~ 90%, Non-Condensing
- Dimesions: 95m x 95m
- Input Voltage: 12V/1.25A

**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

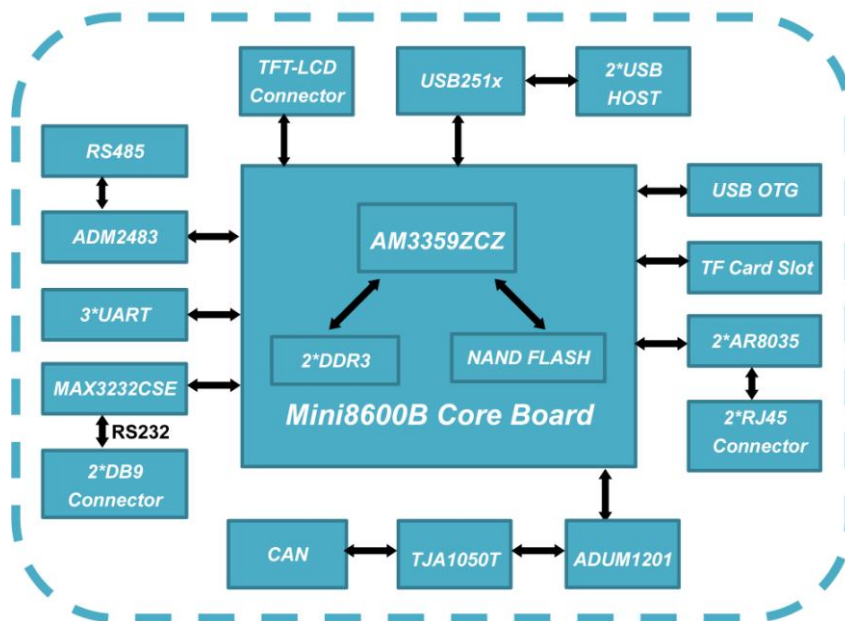


Figure 1-2 Mini8600 Structure Chart

### 1.3 Modules Suitable for the Extension Board

Table 1

Names	Linux	Android	WinCE	Relevant Materials
VGA8000	YES*	YES*	YES*	Available in DVD-ROM
WF8000-U	YES*	NO	NO	Available in DVD-ROM
CAM8100-U	YES*	NO	NO	Available in DVD-ROM
CDMA8000-U	YES*	NO	NO	<a href="#">Download</a>
WCDMA8000-U	YES*	NO	NO	<a href="#">Download</a>
LVDS8000	YES*	YES*	YES*	Available in DVD-ROM and on website

---

# Chapter 2 Hardware System

## 2.1 CPU

### 2.1.1 Introduction to CPU

The AM335x microprocessors, based on the ARM Cortex-A8, are enhanced with image, graphics processing, peripherals and industrial interface options such as EtherCAT and PROFIBUS. The device supports the following high-level operating systems (HLOSs) such as Linux, WinCE and Android.

The AM335x microprocessor contains these 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

AM3359 has two clock inputs, OSC1 and OCC0, and two clock outputs, LCKOUT1 and LCKOUT2.

OSC1 provides RTC with a 32.768KHz 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**

Resetting is controlled by PWRONRSTn signals from CPU. The device is reset when it is a low level signal.

### **General-Purpose Interfaces**

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

### **Programmable Real-Time Unit Subsystem**

The PRUSS of AM3359 consists of 2 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, as well as provide regular display and gaming effect. The subsystem features:

- Tile-Based Architecture Delivering 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 512M NAND Flash used on SBC8600B.

If you need more information about the NAND Flash, please refer to

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

### **2.2.2 DDR H5TQ2G83CFR-H9C**

H5TQ2G83DFR-H9C is a 256MB DDR3 SDRAM used on SBC8600B. There are two H5TQ2G83DFR-H9C on SBC8600B.

If you need to know more about the SDRAM, please refer to H5TQ2G83DFR.pdf under Disk-SBC8600B\HW design\datasheet\DDR\.

### **2.2.3 Ethernet AR8035**

AR8035 is a low-power and low-cost Ethernet PHY used on SBC8600B and integrated with a 10/100/1000Mb transceiver. It is a single-port tri-speed Ethernet PHY and supports MAC.TM RGMII interfaces.

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-through network cable, or connected to a computer with a crossover cable.

If you need know more about the Ethernet chip, please refer to AR8035.pdf under Disk-SBC8600B\HW design\datasheet\LAN\.

#### **2.2.4 MAX3232**

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

If you need to know more about the this chip, please refer to MAX3232CSE.pdf under Disk-SBC8600B\HW design\datasheet\Serial\.

## 2.3 Hardware Interfaces

### 2.3.1 Mini8600B

#### 2.3.1.1 CN1 Interface



Figure 2-1 Mini8600B CN1 Interface

Table 2 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

CN1		
PIN	Signal	Function
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
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



CN1		
PIN	Signal	Function
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
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

**2.3.1.2 CN2 Interface**



**Figure 2-2** SBC8600B CN2 Interface

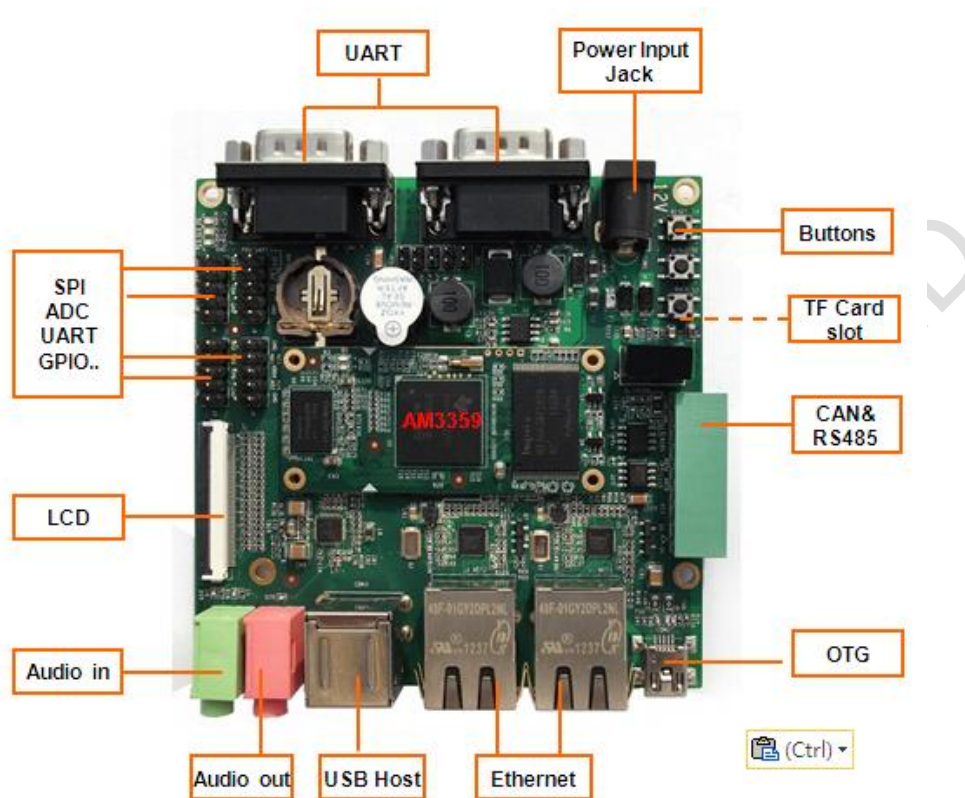
**Table 3** CN2 Interface

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

<b>CN2</b>		
<b>PIN</b>	<b>MODE1</b>	<b>Function</b>
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
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

CN2		
PIN	MODE1	Function
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_ADV_N_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
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

### 2.3.2.1 Power Jack

**Table 4** Power Jack

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

**2.3.2.2 TFT\_LCD Interface**
**Table 5 TFT\_LCD Interface**

<b>J3</b>		
<b>Pin</b>	<b>Singal</b>	<b>Description</b>
1	B0	GND
2	B1	GND
3	B2	GND
4	B3	LCD Pixel data bit 0
5	B4	LCD Pixel data bit 1
6	B5	LCD Pixel data bit 2
7	B6	LCD Pixel data bit 3
8	B7	LCD Pixel data bit 4
9	GND1	GND
10	G0	GND
11	G1	GND
12	G2	LCD Pixel data bit 5
13	G3	LCD Pixel data bit 6
14	G4	LCD Pixel data bit 7
15	G5	LCD Pixel data bit 8
16	G6	LCD Pixel data bit 9
17	G7	LCD Pixel data bit 10
18	GND2	GND
19	R0	GND
20	R1	GND
21	R2	GND
22	R3	LCD Pixel data bit 11
23	R4	LCD Pixel data bit 12
24	R5	LCD Pixel data bit 13
25	R6	LCD Pixel data bit 14
26	R7	LCD Pixel data bit 15
27	GND3	GND
28	DEN	AC bias control (STN) or pixel data enable (TFT)
29	HSYNC	LCD Horizontal Synchronization
30	VSYNC	LCD Vertical Synchronization

J3		
Pin	Singal	Description
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.

### 2.3.2.3 Audio Output Interface

Table 6 Audio Output Interface

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

### 2.3.2.4 Audio Input Interface

Table 7 Audio Input Interface

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

### 2.3.2.5 USB HOST Interface

Table 8 USB HOST Interface

CON3		
Pin	Singal	Description
1	VBUS	+5V
2	DA-	USB Data-
3	DA+	USB Data+
4	GND	GND



### 2.3.2.6 USB OTG Interface

Table 9 USB OTG Interface

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

### 2.3.2.7 TF Card Interface

Table 10 TF Card Interface

TF1		
Pin	Singal	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

### 2.3.2.8 LAN Interface

Table 11 LAN Interface

J1,J2		
Pin	Singal	Description
1	TD1+	Transmit Data1+
2	TD1-	Transmit Data1-
3	TD2+	Transmit Data2+
4	TD2-	Transmit Data2-
5	TCT	Transmit common terminal

<b>J1,J2</b>		
<b>Pin</b>	<b>Singal</b>	<b>Description</b>
6	RCT	Receive common terminal
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

### 2.3.2.9 Serial Interface

Table 12 Serial Interface

<b>J4(UART0), J5(UART2)</b>		
<b>Pin</b>	<b>Singal</b>	<b>Description</b>
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

### 2.3.2.10 CAN&RS485 接口

Table 13 CAN&RS485 Interface

<b>U22</b>		
<b>Pin</b>	<b>Singal</b>	<b>Description</b>
1	+12V	+12V
2	GND	GND
3	GND2	Isolated GND
4	485B1	485B
5	485A1	485A

6	GND1	Isolated GND
7	CANL1	CANL
8	CANH	CANH

### 2.3.2.11 ADC Interface

**Table 14** ADC Interface

<b>J9</b>		
<b>Pin</b>	<b>Singal</b>	<b>Description</b>
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

### 2.3.2.12 SPI Interface

**Table 15** SPI Interface

<b>J8</b>		
<b>Pin</b>	<b>Singal</b>	<b>Description</b>
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

### 2.3.2.13 Extension Interface

**Table 16** Extension Interface

<b>J6</b>		
<b>Pin</b>	<b>Singal</b>	<b>Description</b>
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 17** Extension Interface

<b>J7</b>		
<b>Pin</b>	<b>Singal</b>	<b>Description</b>
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

### 2.3.2.14 Buttons

Table 18 Buttons

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

### 2.3.2.15 LED

Table 19 LED

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

# Chapter 3 Linux Operating System

## 3.1 Introduction

This chapter will introduce the Linux software system of SBC8600B by the following sections:

- Introducing the software resources provided along with SBC8600B;
- Introducing the software features.
- Introducing the creation of development environment, system development, driver principles and development.
- Details of system updating.
- Notes for the use of system.
- Introducing the development of upper layers.

**Note:**

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

## 3.2 Software Resources

This section provides an overview of software system components of SBC8600B. A basic software system consists of four parts: x-loader, u-boot, kernel and rootfs. The Figure 3-1 shows the structure of the system:

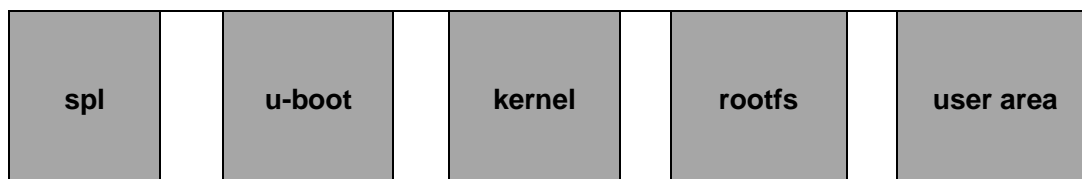


Figure 3-1

Features and functions of each part of the system are listed below:

- 1) spl is a 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 a second level bootstrap program. It is used for interacting with users and updating images and boot the kernel;
- 3) A Linux version with kernel 3.2.0 is employed here and it can be customized based on SBC8600B;
- 4) Rootfs employs open-source system ubifs. It is small in capacity and powerful, very suitable for embedded systems;

### 3.3 Software Features

Table 20

Name		Note
BIOS	spl	NAND
		MMC/SD
		FAT
	u-boot	NAND
		MMC/SD
		FAT
		NET
	Kernel	Linux-3.2.0
Device Driver	serial	Serial driver
	rtc	Hardware clock driver
	net	10/100M/1000M Ethernet driver
	can	Can bus driver
	flash	nand flash driver (supports nand boot)
	lcd	TFT LCD driver
	touch screen	4-line touch-screen controller driver
	mmc/sd	mmc/sd controller driver

	usb otg	usb otg 2.0 driver
	audio	Audio driver (supports audio recording and playback)
	keypad	Gpio keypad driver
	led	User custom led driver
Demo	Android	android 2.3.4 system
	TISDK	TISDK system

## 3.4 System Development

### 3.4.1 Establishment of 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.

#### 3.4.1.1 Installing Cross Compilation Tools

After you insert the DVD-ROM to your PC, Ubuntu will mount it automatically under the directory `/media/cdrom`. The cross compilation tools can be found under `/media/cdrom/linux/tools`.

The following instructions are executed at the Ubuntu terminal to decompress the cross compilation tools under the directory `$HOME`:

#### 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.

- `mkdir $HOME/tools`
- `cd /media/cdrom/linux/tools`
- `tar xvf arm-2009q1-203-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2 -C $HOME/tools`
- `tar xvf arm-eabi-4.4.0.tar.bz2 -C $HOME/tools`



Some of the other development tools used for source code compilation are saved under the same directory; the user can execute the following commands to copy them to local folder:


- `cp /media/cdrom/linux/tools/mkimage $HOME/tools`
- `cp /media/cdrom/linux/tools/mkfs.ubifs $HOME/tools`
- `cp /media/cdrom/linux/tools/ubinize $HOME/tools`
- `cp /media/cdrom/linux/tools/ubinize.cfg $HOME/tools`

### 3.4.1.2 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-4.4.0/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; command `echo $PATH` can be used to check the path.

### 3.4.1.3 Establishment of Android Development Environment

Apart from the cross compilation tools and environment variables, there are a few additional software packages need to be installed and configurations to be set in Ubuntu system before Android system source code could be compiled. Please visit <http://source.android.com/source/initializing.html> for more information.

## 3.4.2 System Compilation


### 3.4.2.1 Preparation

Source codes of all components of the system are saved under the directory linux/source in the disc; user has to decompress them in the Ubuntu system before executing development:

```
• mkdir $HOME/work
• cd $HOME/work
• tar xvf /media/cdrom/linux/source/u-boot-2011.09-bsp04.06.00.03.tar.bz2
• tar xvf /media/cdrom/linux/source/linux-3.2.0-bsp04.06.00.08.sdk.tar.bz2
• tar xvf
  /media/cdrom/linux/demo/android/source/linux-3.1.0-android.tar.bz2
• sudo tar xvf /media/cdrom/linux/source/rootfs.tar.bz2
• tar xvf
  /media/cdrom/linux/demo/android/source/rowboat-android-gingerbread-a
  m335xevm.tar.bz2
```

When the above steps are finished, the directories u-boot-2011.09-bsp04.06.00.03, Linux-3.2.0-bsp04.06.00.08.sdk, Linux-3.1.0-android, rootfs and rowboat-android-gingerbread-am335xevm 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.

### 3.4.2.2 Compilation of booting code

SBC8600B can boot up from TF card or NAND Flash, with the former as first boot-up device and the latter as the secondary.

We will introduce the generation of booting code image files for both the boot-up devices.

```
• cd u-boot-2011.09-psp04.06.00.03
• make distclean
• make sbc8600_config
• make
```

When the above steps are finished, two files named MLO and u-boot.img can be found under current directory.

### 3.4.2.3 Kernel compilation

The operations for Linux system are as follows:

```
• cd Linux-3.2.0-psp04.06.00.08.sdk
• make distclean
• make sbc8600_defconfig
• make ulmage
```

The operations for Android system are as follows:

```
• cd Linux-3.1.0-android
• make distclean
• make sbc8600_android_defconfig
• make ulmage
```

After above operations are executed, a ulmage file will be generated under the directory arch/arm/boot.

### 3.4.2.4 Generation of file system

#### 1) Ramdisk file

Please visit [http://www.elinux.org/DevKit8600\\_FAQ](http://www.elinux.org/DevKit8600_FAQ) for details of how to generate Ramdisk file.

#### 2) UBI file

```
• cd $HOME/work
• sudo $HOME/tools/mkfs.ubifs -r rootfs -m 2048 -e 126976 -c 812 -o ubifs.img
• sudo $HOME/tools/ubinize -o ubi.img -m 2048 -p 128KiB -s 512 -O 2048
  $HOME/tools/ubinize.cfg
```

After above operations are executed, a ubi.img file will be generated under the current directory.

### 3.4.2.5 Android system compilation

- 1) Execute the following instructions to start compilation of Android system;

```
• cd rowboat-android-gingerbread-am335xevm
• make TARGET_PRODUCT=am335xevm clean
• make TARGET_PRODUCT=am335xevm OMAPES=4.x
```

- 2) Modify Rules.make under hardware/ti/sgx/ ;

**Vi hardware/ti/sgx/Rules.make**


Replace “KERNEL\_INSTALL\_DIR=\$(HOME)/work/Linux-3.1.0-android” with “KERNEL\_INSTALL\_DIR=/home/user\_name/work/Linux-3.1.0-android”  
/home/user\_name is the directory where usernames are saved, in other words, it's the value of \$(HOME). To view the value, please enter whoami in the terminal window of Linux system.

- 3) Please enter the following instructions to start making ubi file system;

**source ./build\_ubi.sh**

ubi.img can be found under temp/.

#### Note:

 Before the compilation of Android file system, the Android kernel source code Linux-3.1.0-android needs to compile first, or errors might occur during the process.

### 3.4.3 System Customization

As Linux kernel has many kernel configuration options, users can add or remove drivers and some kernel features in the default configuration to meet specific requirement. The common process of system customization will be described with examples below.

### 3.4.3.1 Modification of kernel configuration


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

Linux-3.2.0-psp04.06.00.08.sdk/arch/arm/configs/sbc8600\_defconfig

Users can carry out system customization based on it:

- **cd Linux-3.2.0-psp04.06.00.08.sdk**
- **cp arch/arm/configs/sbc8600\_defconfig .config**
- **make menuconfig**

#### Note:

 If an error occurs when command 'make menuconfig' is executed, you might need to install 'ncurses' in the Ubuntu system; 'ncurses' 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**

Select the configuration below: The system customization will be described below by taking an example of usb mass storage device emulated with usb gadget:

- > Device Drivers
  - > USB support
    - > USB Gadget Support
      - > USB Gadget Drivers

```

--- USB Gadget Support
[ ] Debugging messages (DEVELOPMENT)
[ ] Debugging information files (DEVELOPMENT)
[ ] Debugging information files in debugfs (DEVELOPMENT)
(2) Maximum VBUS Power usage (2-500 mA)
(2) Number of storage pipeline buffers
<=> USB Peripheral Controller (Inventra HDRC USB Peripheral (TI, ADI, ...)) --->
<=> Select one gadget as builtin for one port
      Select USB port to bind builtin gadget (USB-0) --->
<M> USB Gadget Drivers
<M> Gadget Zero (DEVELOPMENT)
< > Audio Gadget (EXPERIMENTAL)
<M> Ethernet Gadget (with CDC Ethernet support)
[*] RNDIS support
[ ] Ethernet Emulation Model (EEM) support
< > Network Control Model (NCM) support
< > Gadget Filesystem (EXPERIMENTAL)
< > Function Filesystem (EXPERIMENTAL)
<M> File-backed Storage Gadget (DEPRECATED)
[*] File-backed Storage Gadget testing version
< > Mass Storage Gadget
< > Serial Gadget (with CDC ACM and CDC OBEX support)
< > MIDI Gadget (EXPERIMENTAL)
< > Printer Gadget
< > CDC Composite Device (Ethernet and ACM)
< > Multifunction Composite Gadget (EXPERIMENTAL)
< > HID Gadget
< > USB Webcam Gadget
    
```

**Figure 3-2**

Type <M> for “File-backed Storage Gadget”, and select Save when you exit, and compile the kernel again.

### 3.4.3.2 Compilation

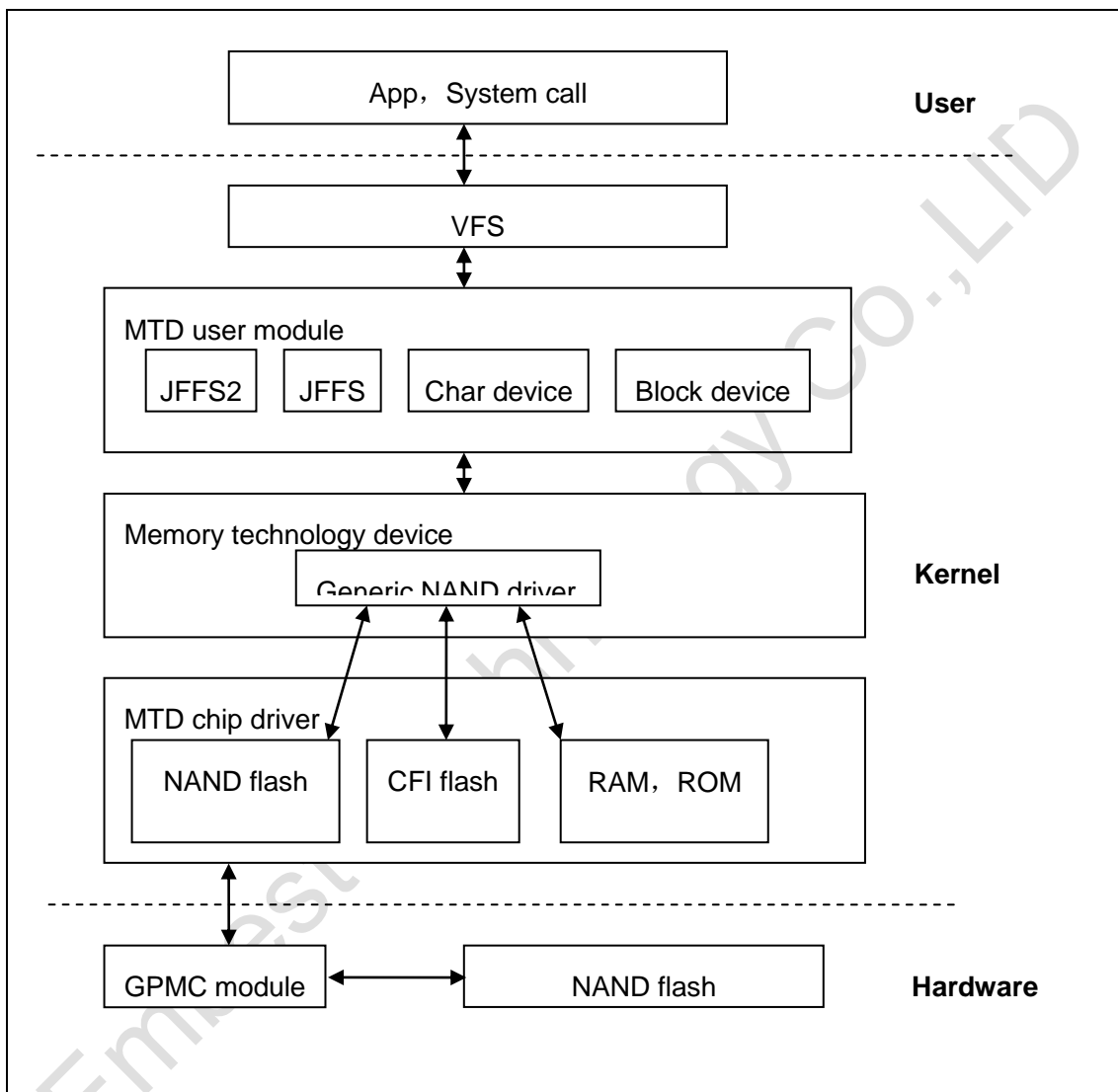
Save configuration, execute the following commands to recompile kernel:

- **make ulmage**
- **make modules**

After above operations are finished, a new kernel image ulmage will be generated under the directory arch/arm/boot, and a module file g\_file\_storage.ko can be found under the directory drivers/usb/gadget.

### 3.5 Introduction of Driver

#### 3.5.1 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 for management.

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

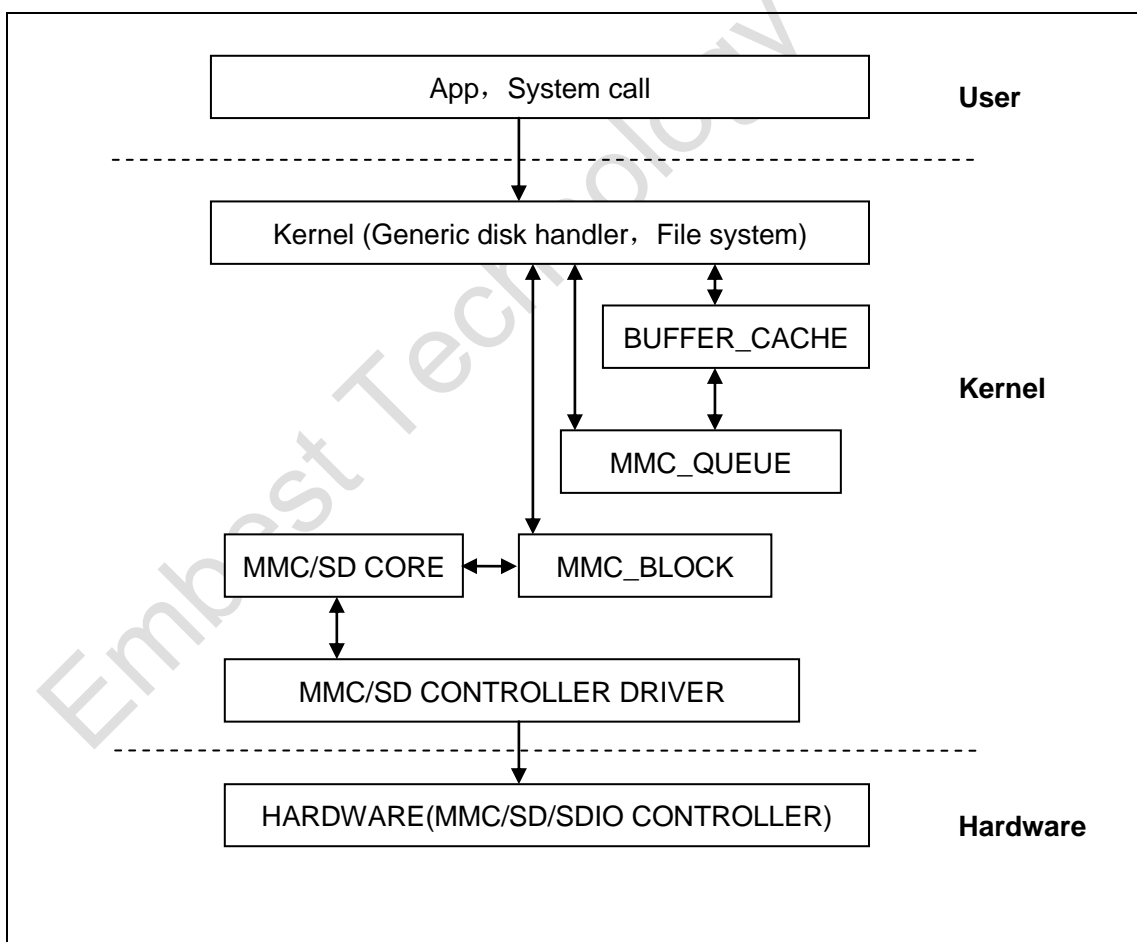
User->System Call->VFS->Block Device Driver->MTD->NAND Flash Driver->NAND Flash.

**Drivers and relevant documents:**

Linux-3.2.0-psp04.06.00.08.sdk/drivers/mtd/nand/

Linux-3.2.0-psp04.06.00.08.sdk/drivers/mtd/nand/omap2.c

**3.5.2 SD/MMC**



**Figure 3-4 Modular structure for SD/MMC**



SD/MMC card drivers for 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-3.2.0-psp04.06.00.08.sdk/drivers/mmc/

Linux-3.2.0-psp04.06.00.08.sdk/drivers/mmc/host/omap\_hsmmc.c

### 3.5.3 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.

- 1) Different interrupt configuration and status register
- 2) 2048\*2048 Higher display resolution of up to 2048\*2048
- 3) 24-bit active TFT grating per pixel

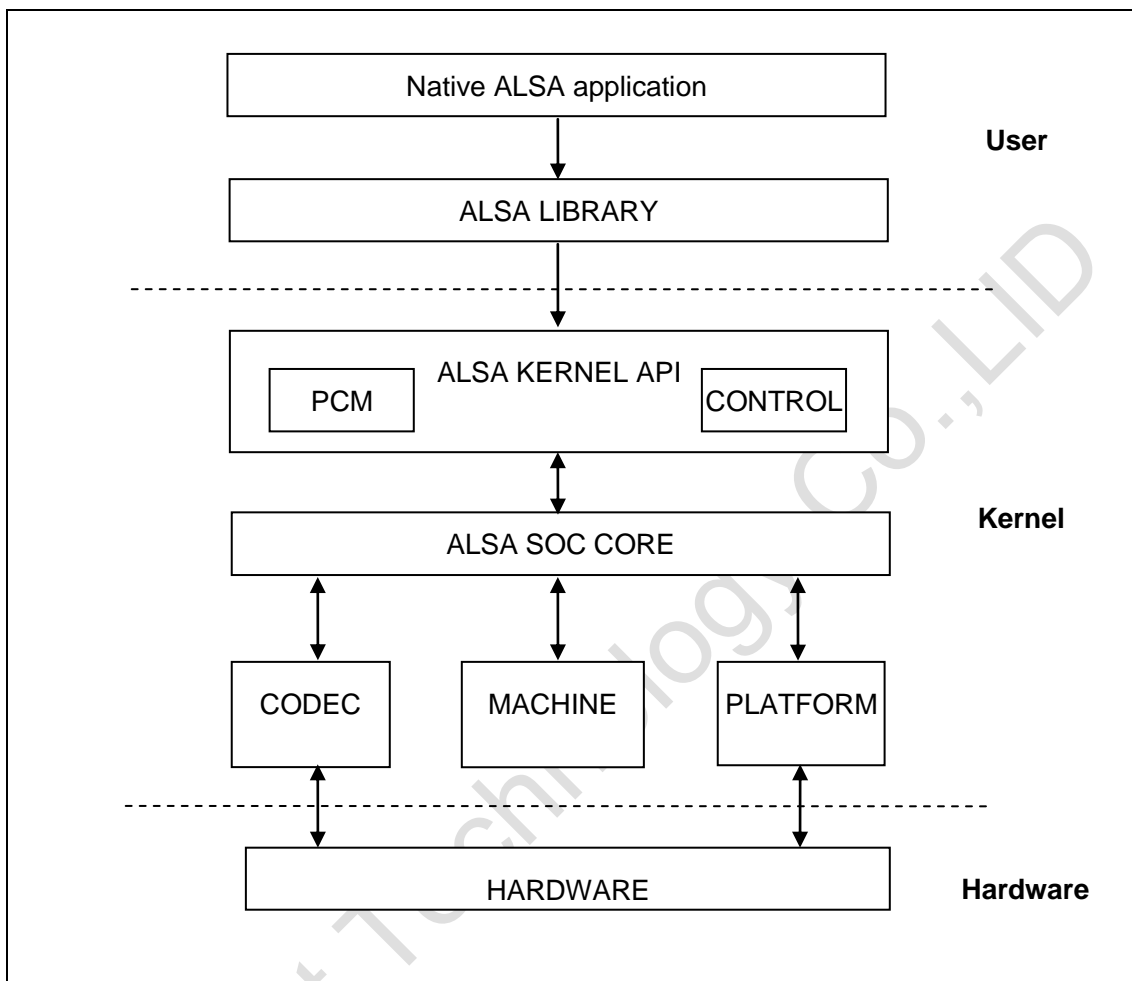
So da8xx-fb LCD driver can be used to improve the LCD\_VERSION2 code. By reading PID register, the update of LCDC version can be found.

**Drivers and relevant documents:**

Linux-3.2.0-psp04.06.00.08.sdk/drivers/video/

Linux-3.2.0-psp04.06.00.08.sdk/drivers/video/da8xx-fb.c

### 3.5.4 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:** The platform driver contains the audio dma engine and audio interface drivers (e.g. I2S, AC97, PCM) for 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-3.2.0-psp04.06.00.08.sdk/sound/soc/

Linux-3.2.0-psp04.06.00.08.sdk/sound/soc/davinci/davinci-evm.c

Linux-3.2.0-psp04.06.00.08.sdk/sound/soc/codecs/sgtl5000.c

## 3.6 Driver Development

### 3.6.1 GPIO\_keys Driver

#### 1) Device Definition

Linux-3.2.0-psp04.06.00.08.sdk/arch/arm/mach-omap2/board-am335xevm.c

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

```
static struct gpio_keys_button gpio_key_buttons[] = {
    {
        .code           = KEY_F1,
        .gpio           = GPIO_TO_PIN(0, 20),
        .active_low     = true,
        .desc           = "menu",
        .type           = EV_KEY,
        // .wakeup       = 1,
    },
    {
        .code           = KEY_ESC,
        .gpio           = GPIO_TO_PIN(2, 1),
        .active_low     = true,
        .desc           = "back",
        .type           = EV_KEY,
        // .wakeup       = 1,
    },
};

static struct gpio_keys_platform_data gpio_key_info = {
    .buttons           = gpio_key_buttons,
    .nbuttons          = ARRAY_SIZE(gpio_key_buttons),
};

static struct platform_device gpio_keys = {
```

```

        .name   = "gpio-keys",
        .id     = -1,
        .dev    = {
                .platform_data = &gpio_key_info,
        },
};

```

## 2) GPIO pinmux Configuration

Define the GPIO0.20 and GPIO2.1 as MODE7 (GPIO mode) and AM33XX\_PIN\_INPUT (configuration input).

Linux-3.2.0-psp04.06.00.08.sdk/arch/arm/mach-omap2/board-am335xevm.c

```

static struct pinmux_config gpio_keys_pin_mux[] = {
    {"xdma_event_intr1.gpio0_20",OMAP_MUX_MODE7 | AM33XX_PIN_INPUT},
    {"gpmc_clk.gpio2_1",OMAP_MUX_MODE7|AM33XX_PIN_INPUT},
    {NULL, 0},
};

```

## 3) Driver Design

Linux-3.2.0-psp04.06.00.08.sdk/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     = __devexit_p(gpio_keys_remove),
    .driver     = {
        .name    = "gpio-keys",
        .owner   = THIS_MODULE,
        .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)
{
    ...

    input = input_allocate_device();
    ...

    for (i = 0; i < pdata->nbuttons; i++) {
        struct gpio_keys_button *button = &pdata->buttons[i];
        struct gpio_button_data *bdata = &ddata->data[i];
        unsigned int type = button->type ?: EV_KEY;

        bdata->input = input;
        bdata->button = button;

        error = gpio_keys_setup_key(pdev, bdata, button);
        if (error)
            goto fail2;

        if (button->wakeup)
            wakeup = 1;

        input_set_capability(input, type, button->code);
    }
    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 fail3;
        }
    ...

```

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

```

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

    setup_timer(&bdata->timer, gpio_keys_timer, (unsigned long)bdata);
    INIT_WORK(&bdata->work, gpio_keys_work_func);

    error = gpio_request(button->gpio, desc);
    if (error < 0) {
        dev_err(dev, "failed to request GPIO %d, error %d\n",
                button->gpio, error);
        goto fail2;
    }

    error = gpio_direction_input(button->gpio);
    if (error < 0) {
        dev_err(dev, "failed to configure"
                " direction for GPIO %d, error %d\n",
                button->gpio, error);
        goto fail3;
    }

    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->timer_debounce = button->debounce_interval;
    }
}

```

```
    }

    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);
        goto fail3;
    }
    irqflags = IRQF_TRIGGER_RISING | IRQF_TRIGGER_FALLING;
    /*
     * 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 = request_threaded_irq(irq, NULL, gpio_keys_isr, irqflags, desc, bdata);
    if (error < 0) {
        dev_err(dev, "Unable to claim irq %d; error %d\n",
                irq, error);
        goto fail3;
    }

    return 0;

fail3:
    gpio_free(button->gpio);
fail2:
    return error;
}
```

**d) Interrupt processing**

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

```

static irqreturn_t gpio_keys_isr(int irq, void *dev_id)
{
    ...
    schedule_work(&bdata->work);
    ...
}

static void gpio_keys_work_func(struct work_struct *work)
{
    ...
    gpio_keys_report_event(bdata);
    ...
}

static void gpio_keys_report_event(struct gpio_button_data *bdata)
{
    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(button->gpio) ? 1 : 0) ^ button->active_low;

    input_event(input, type, button->code, !state);
    input_sync(input);
}

```

### 3.6.2 GPIO\_leds Driver

#### 1) Device Definition

Linux-3.2.0-psp04.06.00.08.sdk/arch/arm/mach-omap2/board-am335xevm.c

Configure GPIO1.30 as "sys\_led" ( system indicator ) and GPIO1.31 as "user\_led", both lighted up by high level signal.

```

static struct gpio_led gpio_leds[] = {
    {
        .name           = "sys_led",
        .default_trigger = "heartbeat",
        .gpio           = GPIO_TO_PIN(1, 30),
    },
    {

```



```

        .name          = "user_led",
        .gpio          = GPIO_TO_PIN(1, 31),
    },
};

static struct gpio_led_platform_data gpio_led_info = {
    .leds              = gpio_leds,
    .num_leds          = ARRAY_SIZE(gpio_leds),
};

static struct platform_device leds_gpio = {
    .name              = "leds-gpio",
    .id                 = -1,
    .dev                = {
        .platform_data = &gpio_led_info,
    },
};

```

## 2) GPIO pinmux Configuration

Linux-3.2.0-psp04.06.00.08.sdk/arch/arm/mach-omap2/board-am335xevm.c

Configure GPIO1.30 and GPIO1.31 as MODE7 (gpio mode) and AM33XX\_PIN\_OUTPUT (configuration output)

```

static struct pinmux_config gpio_led_pin_mux[] = {
    {"gpmc_csn1.gpio1_30",  OMAP_MUX_MODE7 | AM33XX_PIN_OUTPUT},
    {"gpmc_csn2.gpio1_31",  OMAP_MUX_MODE7 | AM33XX_PIN_OUTPUT},
    {NULL, 0},
};

```

## 3) Driver Design

Linux-3.2.0-psp04.06.00.08.sdk/drivers/leds/leds-gpio.c

### a) Call platform\_driver\_register to register gpio\_leds driver

```

static struct platform_driver gpio_led_driver = {
    .probe              = gpio_led_probe,
    .remove              = __devexit_p(gpio_led_remove),
    .driver               = {
        .name            = "leds-gpio",
        .owner            = THIS_MODULE,
        .of_match_table = of_gpio_leds_match,
    },
};

```

```

};

MODULE_ALIAS("platform:leds-gpio");

static int __init gpio_led_init(void)
{
    return platform_driver_register(&gpio_led_driver);
}

static void __exit gpio_led_exit(void)
{
    platform_driver_unregister(&gpio_led_driver);
}

module_init(gpio_led_init);
module_exit(gpio_led_exit);

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

```

**b) Apply for gpio and call led\_classdev\_register to led\_classdev driver**

```

static int __devinit gpio_led_probe(struct platform_device *pdev)
{
    ...

    if (pdata && pdata->num_leds) {
        priv = kzalloc(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]);
        kfree(priv);
        return ret;
    }
}
...
}

static int __devinit create_gpio_led(const struct gpio_led *template,
    struct gpio_led_data *led_dat, struct device *parent,
    int (*blink_set)(unsigned, unsigned long *, unsigned long *))
{
    ...

    ret = gpio_request(template->gpio, template->name);
    ...

    ret = gpio_direction_output(led_dat->gpio, led_dat->active_low ^ state);
    ...

    ret = led_classdev_register(parent, &led_dat->cdev);
    ...
}

```

- c) 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)
{
    ...

    gpio_set_value(led_dat->gpio, level);
}

```

## 3.7 System Update

### 3.7.1 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:

<http://www.embedinfo.com/english/download/SP27213.exe>

- 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:

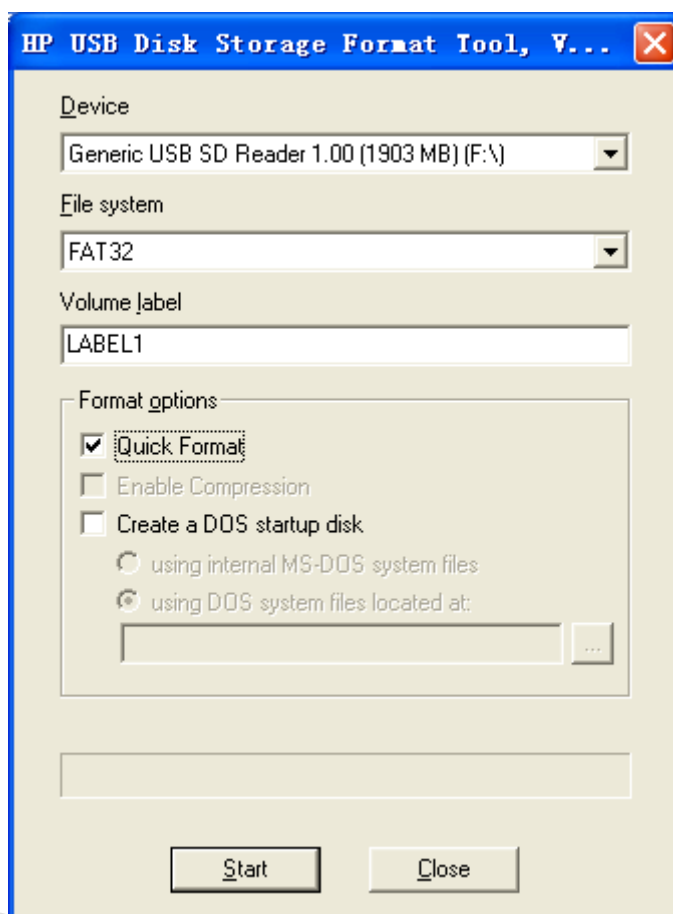



Figure 3-6

- c) Select “FAT32” file system
- d) Click “Start”
- e) When formatting is complete, click “OK”



**Note:**

 HP USB Disk Storage Format Tool will erase partitions of TF card. If you want to maintain the partitions, please use the formatting software of Windows.

## 2) Image update

Copy all files under the directory linux\image to the TF card, insert it on the board, and then power up the board. The information on serial interface will be shown as 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 3.8.1 Selecting Display Mode for more details
-  If there is already an image in NAND Flash, you need to short the jumper **JP5** on the board so as to make it boot up from TF card. Disconnect JP5 after successful boot-up of the system.

```
Booting from MMC...
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img

U-Boot 2011.09-svn55 (Dec 04 2012 - 09:29:02)

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
SD/MMC found on device 0
reading uEnv.txt

** Unable to read "uEnv.txt" from mmc 0:1 **
```

```
reading ulmage

3224184 bytes read
reading ramdisk.gz

12514633 bytes read
## Booting kernel from Legacy Image at 80007fc0 ...
   Image Name:   Linux-3.2.0
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    3224120 Bytes = 3.1 MiB
   Load Address: 80008000
   Entry Point:  80008000
   Verifying Checksum ... OK
   XIP Kernel Image ... OK
OK

Starting kernel ...



Uncompressing Linux... done, booting the kernel.
Linux version 3.2.0 (luofc@TIOP) (gcc version 4.3.3 (Sourcery G++ Lite 2009q1-203) )
#17 Fri Dec 7 10:04:07 CST 2012
.....
.....
RAMDISK: gzip image found at block 0
VFS: Mounted root (ext2 filesystem) on device 1:0.
Freeing init memory: 260K
INIT: version 2.86 booting
Starting udevdevd (741): /proc/741/oom_adj is deprecated, please use
/proc/741/oom_score_adj instead.
tar: removing leading '/' from member names

Remounting root file system...
mount: mounting /dev/root on / failed: Invalid argument
mount: mounting /dev/root on / failed: Invalid argument
root: mount: mounting rootfs on / failed: No such file or directory
Setting up IP spoofing protection: rp_filter.
Configuring network interfaces... udhcpc (v1.11.3) started
Sending discover...
udhcpc: sendto: Network is down
Sending discover...
udhcpc: sendto: Network is down
```



- a) Insert the TF card which contains the system images into the development board, and then connect power supply. Press any key on keyboard to enter the u-boot when a message "Hit any key to stop autoboot" appears:

**Note:**

-  You may short the jumper JP5 on the board to allow SBC8600B boot up from TF card and enter uboot to write the image in NAND Flash, and then disconnect JP5 to allow system boot up from NAND Flash.
-  Alternatively, you may leave **JP5** disconnected and insert TF card on the board to boot up from NAND Flash, and then write the image in NAND Flash through uboot.

```

U-Boot SPL 2011.09-svn55 (Nov 20 2012 - 10:37:42)
Texas Instruments Revision detection unimplemented
Booting from MMC...
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img

U-Boot SPL 2011.09-svn55 (Nov 20 2012 - 10:37:42)
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 (press any key to enter uboot)
    
```

- b) After entering the u-boot command line, type "run updatesys" to start update process of the system:



```
SBC8600# run updatesys

NAND erase.chip: device 0 whole chip
Erasing at 0x7fe0000 -- 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 u-boot.img

234896 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x80000, size 0x39590
 234896 bytes written: OK
reading ulmage

3224184 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x280000, size 0x313278
 3224184 bytes written: OK
reading ubi.img

14811136 bytes read
SW ECC selected

NAND write: device 0 offset 0x780000, size 0xe20000
Skip bad block 0x00ce0000
 14811136 bytes written: OK
```

Flashing LED on the board indicates that the update has been finished; please remove TF and 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 3.8.1 Selecting Display Mode.

## 3.8 Instructions

### 3.8.1 Selecting Display Mode

System supports a wide range of display mode. 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 2011.09-svn55 (Nov 20 2012 - 10:37:42)
Texas Instruments Revision detection unimplemented
Booting from MMC...
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img

U-Boot SPL 2011.09-svn55 (Nov 20 2012 - 10:37:42)
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 (press any key to enter uboot)
```

### **3.8.1.1 Using a 4.3" LCD Display**

Modify the parameter by executing the command as follows in the U-boot command mode.

```
SBC8600# setenv dispmode 4.3inch_LCD
```

```
SBC8600# saveenv
```

### **3.8.1.2 Using a 7" LCD Display**

Modify the parameter by executing the command as follows in the U-boot command mode.

```
SBC8600# setenv dispmode 7inch_LCD
```

```
SBC8600# saveenv
```

### **3.8.1.3 Using a VGA Display**

Modify the parameter by executing the command as follows in the U-boot command mode.

```
SBC8600# setenv dispmode VGA
```

```
SBC8600# saveenv
```

### **3.8.1.4 Using a LVDS Display**

Modify the parameter by executing the command as follows in the U-boot command mode.

```
SBC8600# setenv dispmode LVDS
```

```
SBC8600# saveenv
```

## 3.8.2 Testing

### 3.8.2.1 LED Testing

The D35 LED on the board is the system indicator, D36 is an user custom LED.

The following operations are accomplished in HyperTerminal:

- 4) Controlling system indicator

```
root@SBC8600:~# echo 1 > /sys/class/leds/sys_led/brightness
```

```
root@SBC8600:~# echo 0 > /sys/class/leds/sys_led/brightness
```

- 5) Controlling user custom LED

```
root@SBC8600:~# echo 1 > /sys/class/leds/user_led/brightness
```

```
root@SBC8600:~# echo 0 > /sys/class/leds/user_led/brightness
```

The LED will respond accordingly to the instructions.


### 3.8.2.2 KEYPAD Testing

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

```
root@SBC8600:~# 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.

### 3.8.2.3 Touch Screen Testing

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

- 1) Execute the following instruction to test touch-screen

```
root@SBC8600: # 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@SBC8600: # ts_test
```

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

### 3.8.2.4 Backlight Testing

The backlight brightness has a range from 0 to 100, in which 100 means highest brightness, 0 means lowest.

Execute the following instructions to test backlight brightness.

- 1) View the default brightness

```
root@SBC8600:~# cat /sys/class/backlight/pwm-backlight/brightness  
80
```

- 2) Set the brightness to 0

```
root@SBC8600:~# echo 0 > /sys/class/backlight/pwm-backlight/brightness
root@SBC8600:~# cat /sys/class/backlight/pwm-backlight/brightness
0
```

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

- 3) Set the brightness to 100

```
root@SBC8600:~# echo 100 > /sys/class/backlight/pwm-backlight/brightness
root@SBC8600:~# cat /sys/class/backlight/pwm-backlight/brightness
100
```

The screen is turned on.

### 3.8.2.5 RTC Testing

The development board contains hardware clock to save and synchronize the system time.

Test can be accomplished with the following steps:

- 1) **Set the system time as Mar 22 20:00:00 2012**

```
root@SBC8600: # date 032220002012
Thu Mar 22 20:00:00 UTC 2012
```

- 2) **Write the system clock into RTC**

```
root@SBC8600: # hwclock -w
```

- 3) **Read the RTC**

```
root@SBC8600: # hwclock
Thu Mar 22 20:00:10 2012 0.000000 seconds
```



We can see that the RTC clock has been set as Mar 22, 2012; the system clock will be saved in the hardware clock.

- 4) **Restart the system; enter the following commands to update the system clock**

```
root@SBC8600: # hwclock -s
root@SBC8600: # date
Thu Mar 22 20:01:30 2012 0.000000 seconds
```

We can see the system time is set as hardware time.

**Note:**

-  You may find the RTC stop running in the scenario where the board is powered off and then powered on again. This is caused by the bug of the CPU. Please refer to the errata provided by TI.
-  The development board does not have a RTC battery (model CR1220) by default; and users need to use a battery of their own.

### 3.8.2.6 TF Card Testing

- 1) After inserting TF card, the system will mount the TF card under the directory /media automatically:

```

root@SBC8600:~# df -h
Filesystem      Size      Used Available Use% Mounted on
rootfs          31.0M    19.7M   11.3M   64% /
/dev/root       31.0M    19.7M   11.3M   64% /
none            250.6M    684.0k  249.9M    0% /dev
tmpfs           250.6M     20.0k  250.6M    0% /var/volatile
tmpfs           250.6M      0     250.6M    0% /dev/shm
tmpfs           250.6M     3.0M   247.6M    1% /media/ram
/dev/mmcblk0p1  1.8G    101.8M    1.8G    5% /media/mmcblk0p1
    
```

- 2) Enter the following command to see the contents inside the TF card:

```

root@SBC8600:~# ls /media/mmcblk0p1
u-boot.img      mlo              ulmage
ramdisk.gz      ubi.img
    
```

- 3) Unmount TF card manually.

```

root@SBC8600:~# umount /media/mmcblk0p1
    
```


- 4) Mount TF card manually.

```

root@SBC8600:~# mount -t vfat /dev/mmcblk0p1 /mnt/cf
root@SBC8600:~# df -h
Filesystem      Size      Used Available Use% Mounted on
rootfs          31.0M    19.7M   11.3M   64% /
...
tmpfs           250.6M     3.0M   247.6M    1% /media/ram
/dev/mmcblk0p1  1.8G    101.8M    1.8G    5% /media/cf
root@SBC8600:~# ls /media/cf
    
```

u-boot.img	mlo	ulmage
ramdisk.gz	ubi.img	

**Note:**

 The system can mount the TF card automatically when you insert it on the board. However, we recommend mounting it manually because automatic mounting leads to slow writing speed.

### 3.8.2.7 USB DEVICE Testing

USB DEVICE testing is accomplished by using a cable to connect the miniUSB interface of the development board to the USB interface on PC; The development board is recognized by PC as a network device so that the two ends may communicate by Ping command.

- 1) After system boot-up, please use a USB cable to connect the development board to your PC. The Linux USB Ethernet driver needs to be installed on PC. Please refer to Appendix III Installation of Linux USB Ethernet/RNDIS Gadget for detailed information
- 2) Executing the following commands in the HyperTerminal:

```

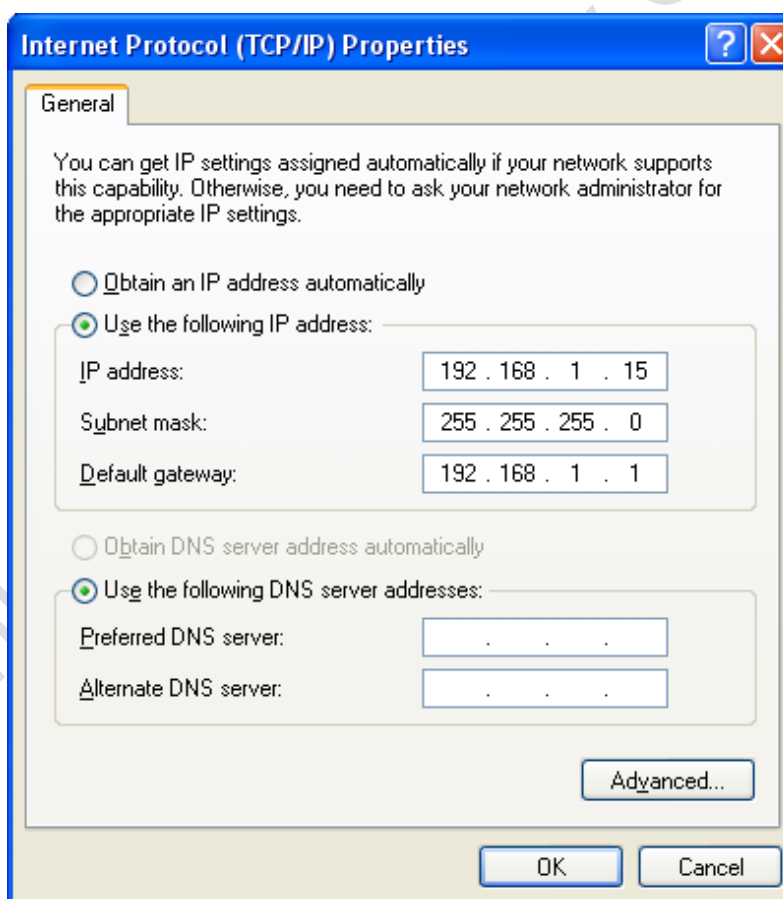
root@SBC8600:~# ifconfig usb0 192.168.1.115
root@SBC8600:~# ifconfig
lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            UP LOOPBACK RUNNING  MTU:16436  Metric:1
            RX packets:26 errors:0 dropped:0 overruns:0 frame:0
            TX packets:26 errors:0 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) After the development board is configured, please click My Computer > Network Neighborhood > Check Network Connection, a virtual network adapter will be added into the PC.
- 4) Right-click virtual network adapter on PC and select “Attribute”, and then double-click the “Internet Protocol (TCP/IP)” to configure the IP address of the virtual network adapter:




**Figure 3-7**

- 5) Use ping command in the HyperTerminal to test whether the settings of the development board are successful:

```

root@SBC8600:~# ping 192.168.1.15
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
    
```

6) The above information indicates a successful testing.

**Note:**  
 IP address of the network adapter configured in OTG should not be as same as that of Ethernet interface.

### 3.8.2.8 USB HOST Testing

1) After inserting USB flash disk on the board, the system will mount disk under the directory `/media` automatically;

```

root@SBC8600:~# df -h
Filesystem      Size      Used Available Use% Mounted on
rootfs          31.0M    19.7M    11.3M   64% /
/dev/root       31.0M    19.7M    11.3M   64% /
none           250.6M    684.0k   249.9M    0% /dev
tmpfs          250.6M     20.0k   250.6M    0% /var/volatile
tmpfs          250.6M      0      250.6M    0% /dev/shm
tmpfs          250.6M     3.0M   247.6M    1% /media/ram
/dev/sda1      99.2M     3.3M   95.9M    3% /media/sda1

root@SBC8600:~# ls /media/sda1/
MLO  u-boot.img  ulmage
    
```

2) Unmount USB disk manually;

```
root@SBC8600:~# cd /home/root
```

```
root@SBC8600:~# umount /media/sda1/
```

3) Type command `df`. The absence of directory `/media/sda1/` indicates that the USB disk is unmounted successfully;

```

root@SBC8600:~# df
Filesystem      1k-blocks    Used Available Use% Mounted on
rootfs          31729        20185    11544  64% /
/dev/root       31729        20185    11544  64% /
none            256624         684    255940   0% /dev
/dev/mmcbk0p1  1939712     104316   1835396   5% /media/mmcbk0p1
tmpfs           256624         20    256604   0% /var/volatile
tmpfs           256624         0    256624   0% /dev/shm
tmpfs           256624        3104    253520   1% /media/ram
    
```

4) Mount USB disk manually;

```

root@SBC8600:~# mount -t vfat /dev/sda1 /mnt/card/
root@SBC8600:~# df -h
Filesystem      Size      Used Available Use% Mounted on
rootfs          31.0M     19.7M     11.3M   64% /
/dev/root       31.0M     19.7M     11.3M   64% /
none            250.6M     684.0k    249.9M   0% /dev
tmpfs           250.6M     20.0k    250.6M   0% /var/volatile
tmpfs           250.6M         0    250.6M   0% /dev/shm
tmpfs           250.6M     3.0M     247.6M   1% /media/ram
/dev/sda1       99.2M     3.3M     95.9M   3% /media/card
    
```

### 3.8.2.9 AUDIO Testing

The board has audio input and output interfaces. Users can enter the following instructions to test alsa-utils audio player and recorder in the file system:

1) **Audio Recorder Testing**

Plug in a microphone to test the audio recorder.

```

root@SBC8600:~# arecord -t wav -c 1 -r 44100 -f S16_LE -v k
Recording WAVE 'k' : Signed 16 bit Little Endian, Rate 44100 Hz, Mono
Plug PCM: Route conversion PCM (sformat=S16_LE)
Transformation table:
0 <- 0*0.5 + 1*0.5
Its setup is:
stream      : CAPTURE
access     : RW_INTERLEAVED
format     : S16_LE
subformat  : STD
channels   : 1
    
```

```

rate      : 44100
exact rate : 44100 (44100/1)
msbits    : 16
buffer_size : 32768
period_size : 2048
period_time : 46439
tstamp_mode : NONE
period_step : 1
avail_min  : 2048
period_event : 0
start_threshold : 1
stop_threshold : 32768
silence_threshold: 0
silence_size : 0
boundary   : 1073741824
.....

```

## 2) Playback Testing

Plug in a headphone to listen to what you recorded.

```

root@SBC8600:~# aplay -t wav -c 2 -r 44100 -f S16_LE -v k
Playing WAVE 'k' : Signed 16 bit Little Endian, Rate 44100 Hz, Mono
Plug PCM: Route conversion PCM (sformat=S16_LE)
      Transformation table:
      0 <- 0
      1 <- 0

Its setup is:
stream      : PLAYBACK
access      : RW_INTERLEAVED
format      : S16_LE
subformat   : STD
channels     : 1
rate        : 44100
exact rate  : 44100 (44100/1)
msbits      : 16
buffer_size : 32768
period_size : 2048
period_time : 46439
tstamp_mode : NONE
period_step : 1
avail_min   : 2048
period_event : 0

```

```


start_threshold : 32768
stop_threshold : 32768
silence_threshold: 0
silence_size : 0
boundary : 1073741824
.....

```

### 3.8.2.10 Network Testing

- 1) There are two Ethernet interfaces, NET1 (J1) and NET2 (J2), associated with two device nodes, eth0 and eth1. Please use two network cables to 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..

```

[root@SBC8600/]# ifconfig eth0 192.192.192.200
[root@SBC8600/]# ifconfig
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)

[root@SBC8600/]# ping 192.192.192.170
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
[root@SBC8600/]# ifconfig eth1 192.168.168.116
[root@SBC8600/]# ifconfig
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)

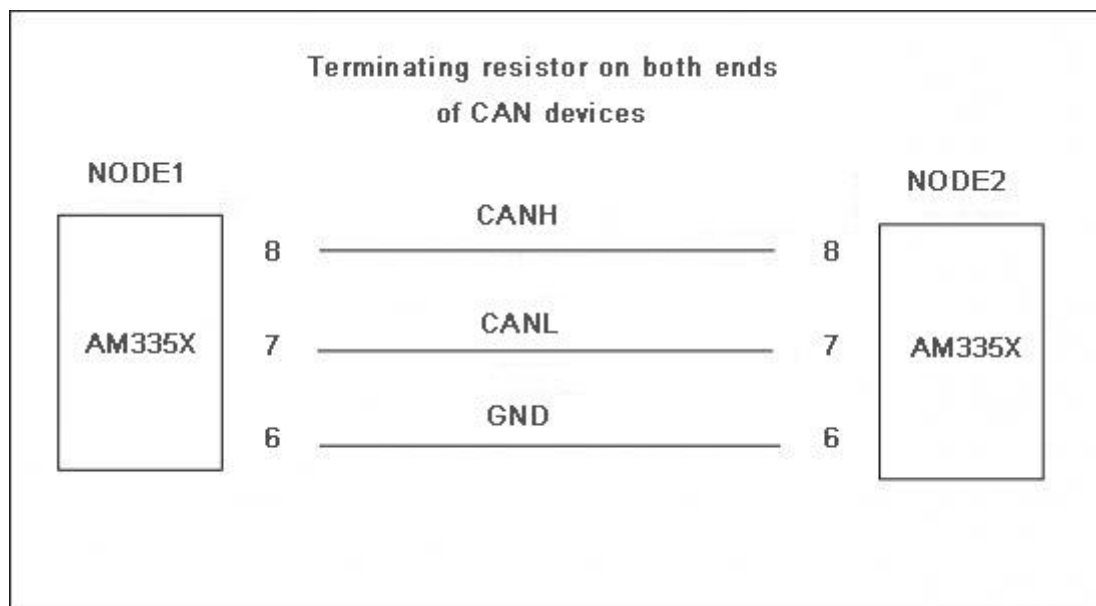
Lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

[root@SBC8600/]# ping 192.168.168.121
PING 192.168.168.121 (192.168.168.121): 56 data bytes
64 bytes from 192.168.168.121: seq=0 ttl=64 time=7.969 ms
64 bytes from 192.168.168.121: seq=1 ttl=64 time=0.319 ms
    
```

2) The above information indicates a successful network testing.

### 3.8.2.11 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 3-8**

**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@SBC8600:~# canconfig can0 bitrate 125000 ctrlmode triple-sampling on
```

```
root@SBC8600:~# canconfig can0 start
```

- 2) Transmit and receive data on the two devices respectively by typing the following instructions.

```
root@SBC8600:~# cansend can0 -i 0x10 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88
```

**Note:**

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

- 3) Receiving data package;

```
root@SBC8600:~# candump can0
```

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

- 4) Stop the CAN device

```
root@SBC8600:~# canconfig can0 stop
```

Users can test with different bit rates by using the instructions above. CAN device need to be stopped before it is reconfigured. The following list contains the bit rate that can be used to testing.

25KBPS (250000)

50KBPS (50000)

125KBPS (125000)


500KBPS (500000)

650KBPS (650000)

1MKBPS (1000000 )

You can also try other bit rates that have not been listed here.

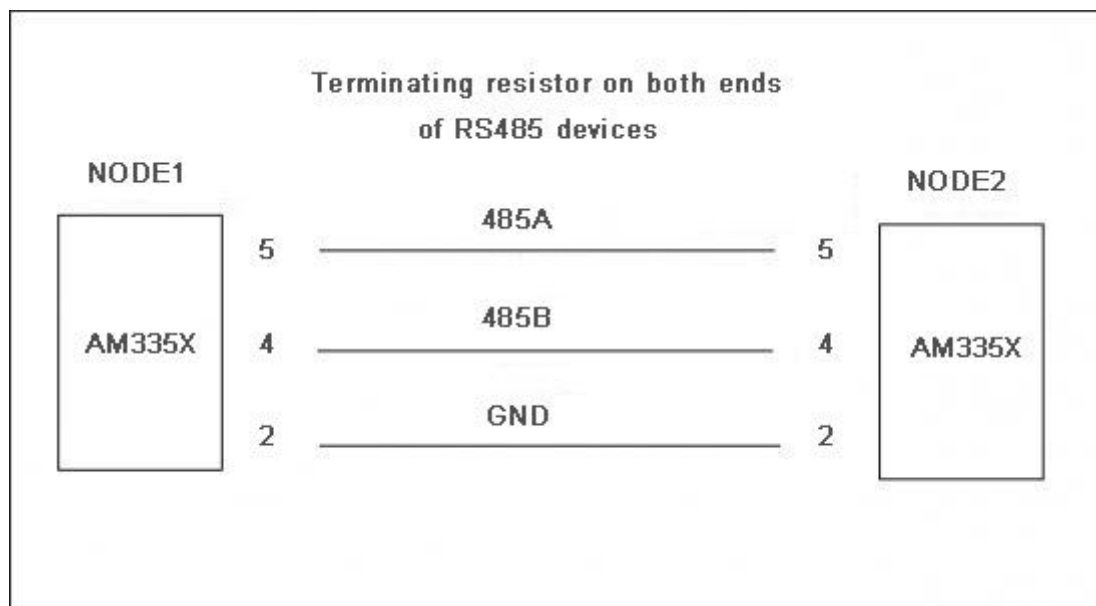
**Note:**

 Same bit rate has to be applied when testing over two development boards.

### 3.8.2.12 RS485 Testing

Please connect the RS485 interfaces on both SBC8600B and another device according to board schematic and the figure shown below:





**Figure 3-9**

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@SBC8600:~# cd /media/mmcblk0p1/
root@SBC8600:/media/mmcblk0p1# ./uart_test -d /dev/ttyO1 -b 115200
/dev/ttyO1 SEND: 1234567890
/dev/ttyO1 RECV 10 total
/dev/ttyO1 RECV: 1234567890
/dev/ttyO1 SEND: 1234567890
/dev/ttyO1 RECV 10 total
/dev/ttyO1 RECV: 1234567890
/dev/ttyO1 SEND: 1234567890
/dev/ttyO1 RECV 10 total
/dev/ttyO1 SEND: 1234567890
/dev/ttyO1 RECV 10 total
    
```

**3.8.2.13 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` into TF card, and then insert it on the board. Execute the following instructions in the terminal window;

```

root@SBC8600:~# cd /media/mmcblk0p1/
    
```

```
root@SBC8600:/media/mmcblk0p1# ./uart_test -d /dev/ttyO2 -b 115200
```

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

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

The same testing method can be applied on serial interface 3, 4 and 5 of J6 and J7 on SBC8600B

### 3.8.2.14 Buzzer Testing

- 1) Enable the buzzer;

```
root@SBC8600:~# echo 1 > /sys/class/misc/buzzer_ctl/state
```

- 2) Disable the buzzer

```
root@SBC8600:~# echo 0 > /sys/class/misc/buzzer_ctl/state
```

### 3.8.2.15 CDMA8000-U Module

CDMA8000-U is an optional module. You can download the relevant materials at

<http://www.timll.com/chinese/uploadFile/cdma8000.rar>

### 3.8.3 Demo

#### 3.8.3.1 Android System Demonstration

SBC8600B provides Android system demonstration, please follow the steps listed below:

- 1) Copy all files under the directory \linux\demo\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 ulmage

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 3.8.1 Selecting Display Mode.

### 3.8.3.2 TISDK System Demonstration

- 1) Format a TF card into two partitions by following the steps described in Appendix IV Formating Linux Boot Disk;

- 2) Insert DVD-ROM and TF card on PC, and then execute the following instructions;

```

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

```

- 3) After the above instructions are executed, please short jumper **JP5** and insert TF card on the board, and then plug in the power supply. The boot-up information is shown as below:

```

CCCCCCC
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 ulmage

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
```

- 4) TISDK file system is featured with some applications running on QT which allow users find and run example programs easily through a friendly graphic interface.
- 5) U-boot configuration  
The system image supports 4.3-inch display by default. If you are working with a display of other size, you need to modify the parameters in UBOOT. Please refer to 3.8.1 Selecting Display Mode for details.

### **3.9 The Development of Applications**

This section mainly introduces the development of application programs, and illustrates the general process of application programs development through examples.

#### **Development example of LED application program**

- 1) Composing Source Code

The following sentences are led\_acc.c source code: control the three LEDs on the development board to flash in a way 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) Cross compiling

```
arm-none-linux-gnueabi-gcc led_acc.c -o led_acc
```

## 3) Downloading and running

Downloading to the development board system through TF card, USB flash disk or network and enter the directory where the led\_acc file is saved, and then execute the following instructions to run led\_acc in the background.

```
./led_acc &
```

Embest Technology Co., Ltd



# Chapter 4 Windows Embedded Compact 7 Operating System

## 4.1 Introduction

This section mainly introduces SBC8600B system and application development of Windows Embedded Compact 7, as well as software resources in DVD-ROM, software features, and installation of development environment, and how to compile project and build BSP (board support package)

## 4.2 Software Resources

### **BSP (Board Support Package)**

CD\WINCE700\BSP\SBC8600.rar

CD\WINCE700\BSP\COMMON\_TI\_V1.rar

CD\WINCE700\BSP\3rdParty.rar

CD\WINCE700\BSP\PowerVR.rar

### **Windows Embedded Compact 7 sample project**

CD\WINCE700\project\SBC8600

### **Example applications**

CD\WINCE700\app\

### **Pre-compiled image**

CD\WINCE700\image\

MLO

First bootloader for TF card boot

xldrnanand.nb0	First bootloader for NAND flash boot
Ebootsd.nb0	Second bootloader for TF card boot
Ebootnd.nb0	Second bootloader for NAND flash boot
Nk.bin	WinCE runtime image

### 4.3 Features

Resources in BSP

Table 21

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

## 4.4 System Development

### 4.4.1 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>.

### 4.4.2 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.

### 4.4.3 Sysgen & BSP Compilation

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

- 1) Open the existing project file SBC8600.sln under [C:\WINCE700\OSDesigns\SBC8600]
- 2) Select [Build-> Build Solution] in VS2008 to start the process of sysgen and BSP compilation.
- 3) Copy the files MLO, EBOOTSD.nb0 and NK.bin under [C:\WINCE700\OSDesigns\SBC8600\SBC8600\ReIDir\SBC8600\_ARMV7\_Release] to the TF card after compilation is done.
- 4) Insert TF card on SBC8600B and power it on.

### 4.4.4 Introduction of Drivers

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

**Table 22**

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.

## 4.5 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.

### 4.5.1 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 <http://www.embedinfo.com/english/download/SP27213.exe>

- 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 appear.

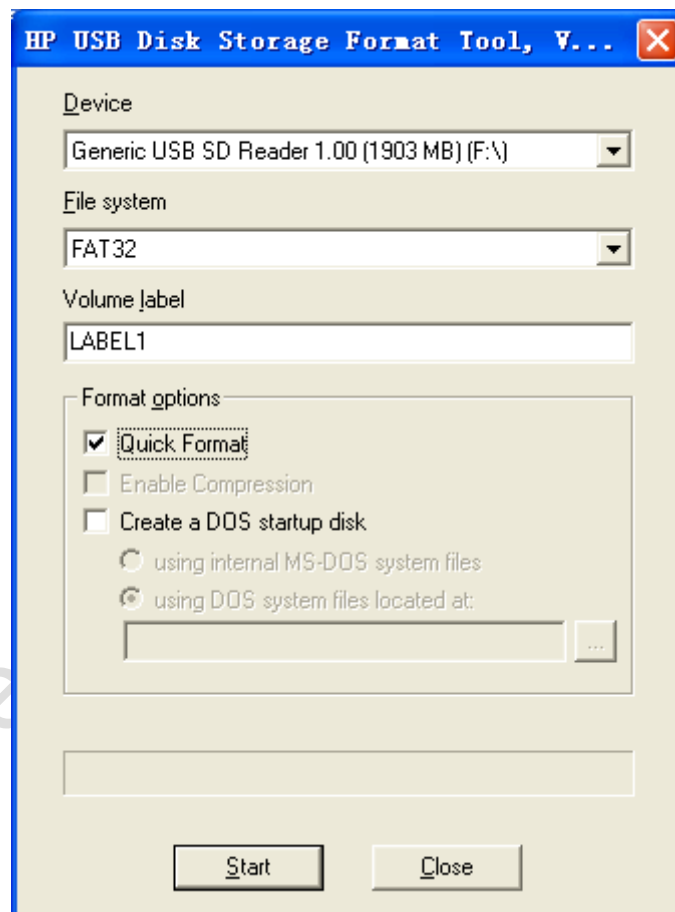



Figure 4-1

- c) Select "FAT32" file system
- d) Click "Start"
- e) Click "OK" when it's complete

**Note:**

 HP USB Disk Storage Format Tool will erase partitions of TF card. If you want to maintain the partitions, please use the formatting software of Windows.

**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 Frequency is 24 MHz---

```

## 4.5.2 Update of NAND Flash Image

### 1) Formatting TF card

Please refer to the contents of 4.5.1 Update of TF Card.

### 2) Copy runtime image

Copy **MLO**, **EBOOTND.nb0**, **NK.bin**, **XLDRNAND.nb0** and **EBOOTSD.nb0** image files under CDWINCE700\image to the TF card.

### 3) 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:

- Type [8] to enter the Flash menu;
- Type [9]->[4]->[A], [9]->[3]->[B] and [9]->[2]->[C] to write XLDR, EBOOT and NK images;
- 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.

## 4.6 Instructions for Use

### 4.6.1 How to use OpenGL ES demo

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

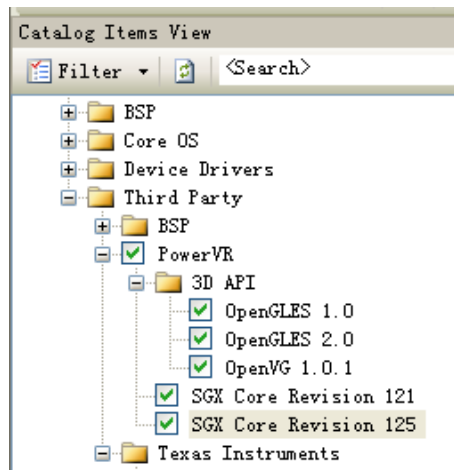


Figure 4-2

- 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.

## 4.7 Application Development

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

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

### 4.7.2 GPIO Application Interfaces and Examples

GPIO application interfaces and examples:

**Table 23**

<b>IOCTL code</b>	<b>Description</b>
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:

```
DeviceloControl (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 24

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

Table 4-4

3) Output of GPIO pins

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

a) High level output:

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

b) Low level output

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

c) Read the pin state

```
DeviceloControl (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;
DeviceloControl (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**

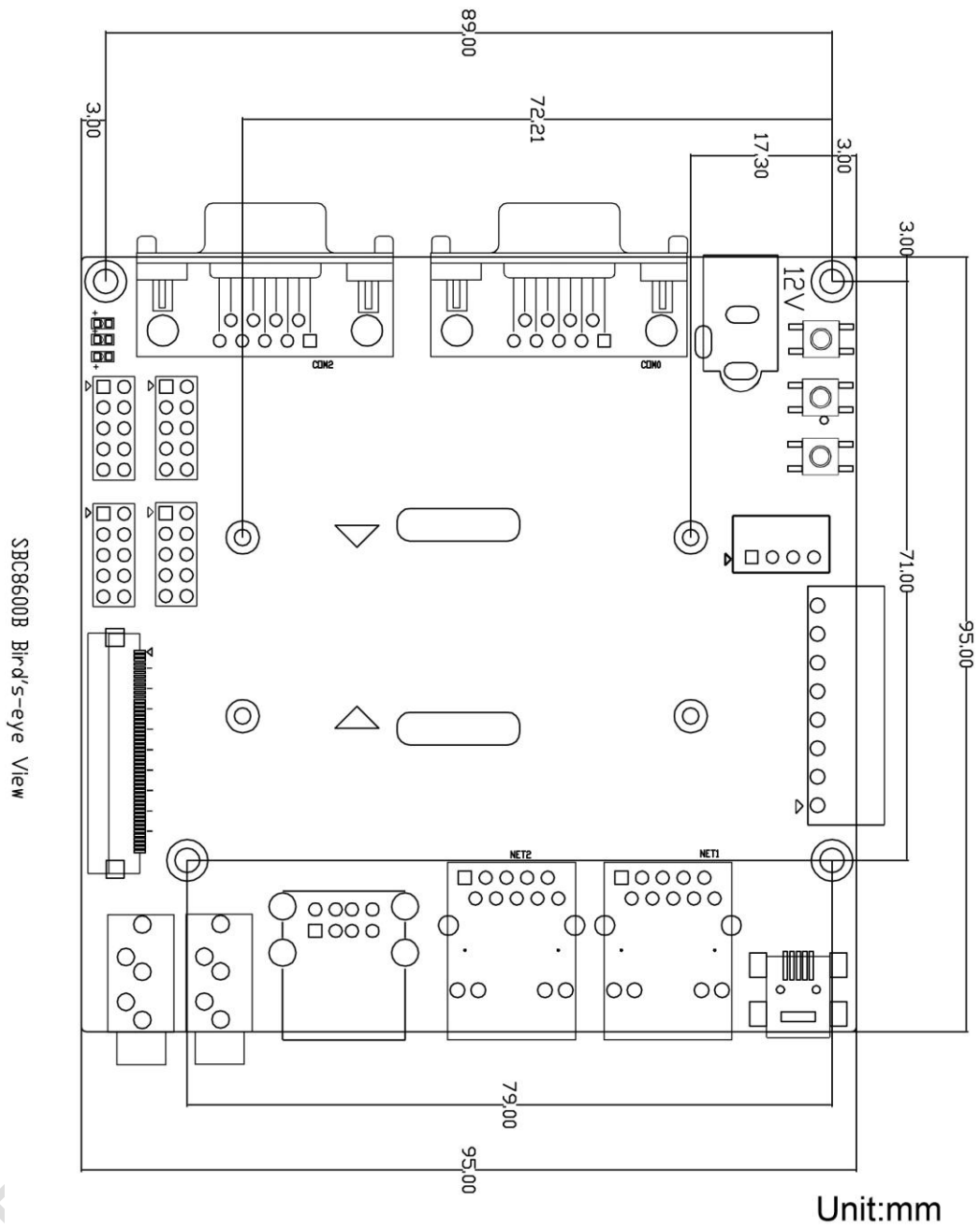
**CloseHandle (hFile);**

**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/.







**Figure 2** SBC8600B hardware dimension

---

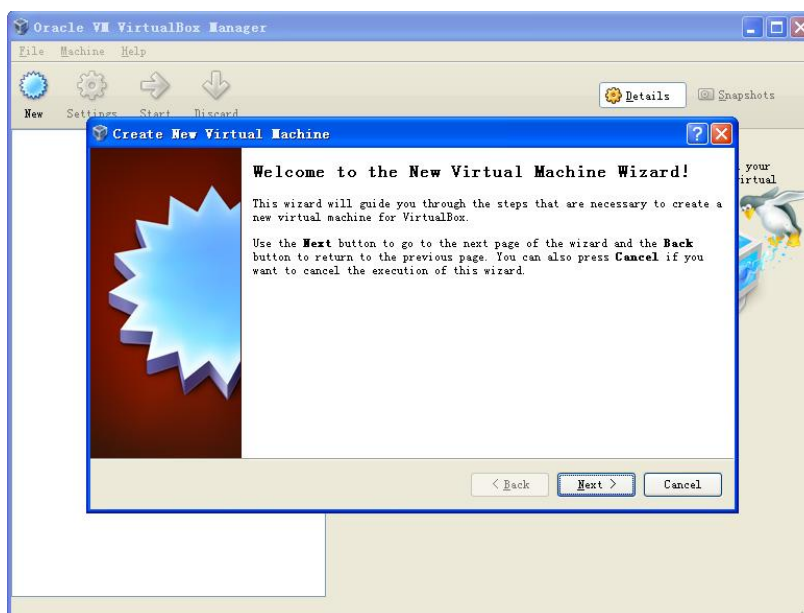
## **Appendix II Installation of Ubuntu**

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

### **Installing VirtualBox**

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

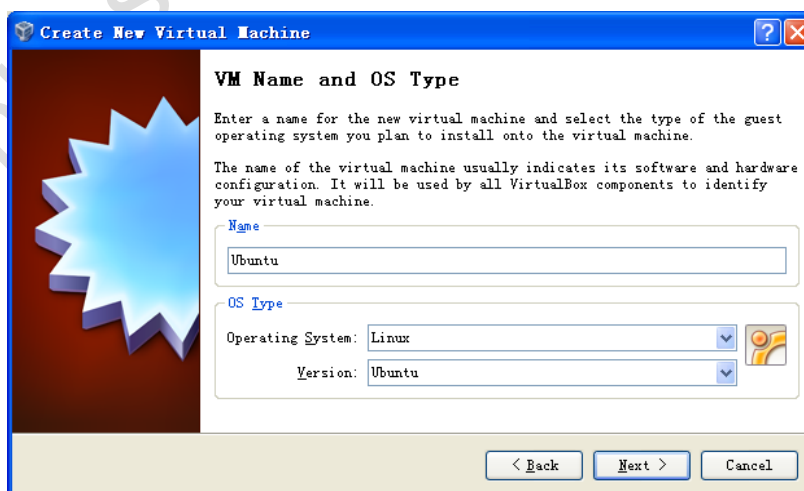
- 1) The installation process is simple and will not be introduced. Please start VirtualBox from the **Start** menu of Windows, and then click **New** in VirtualBox window. A pop-up window **Create New Virtual Machine** will be shown as below;



**Figure 3** 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 4** 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**;

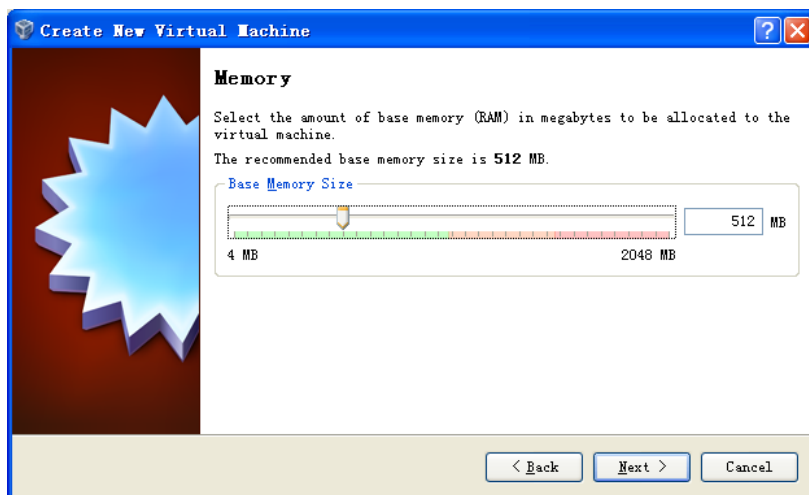


Figure 5 Memory allocation

**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**;

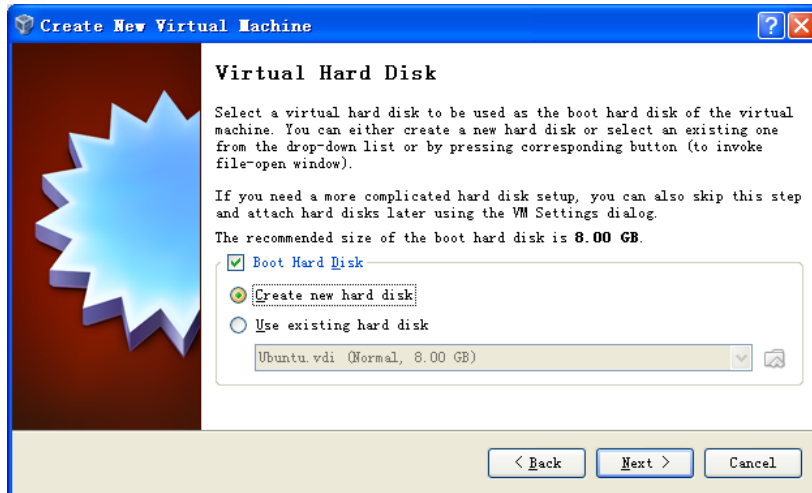


Figure 6 Create new hard disk

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

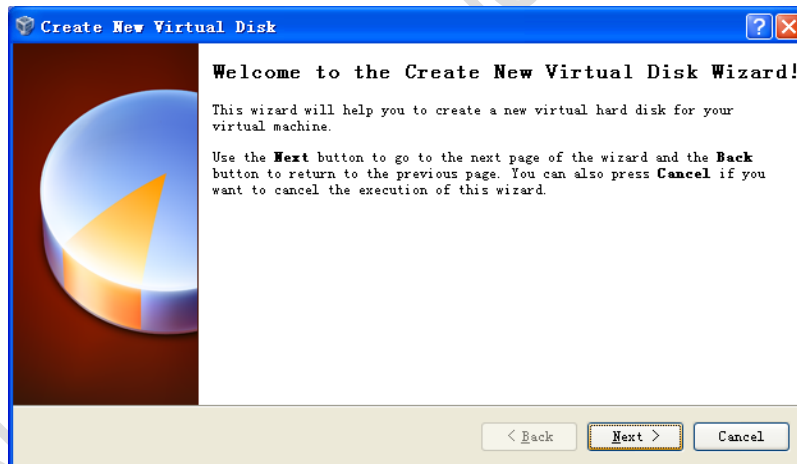


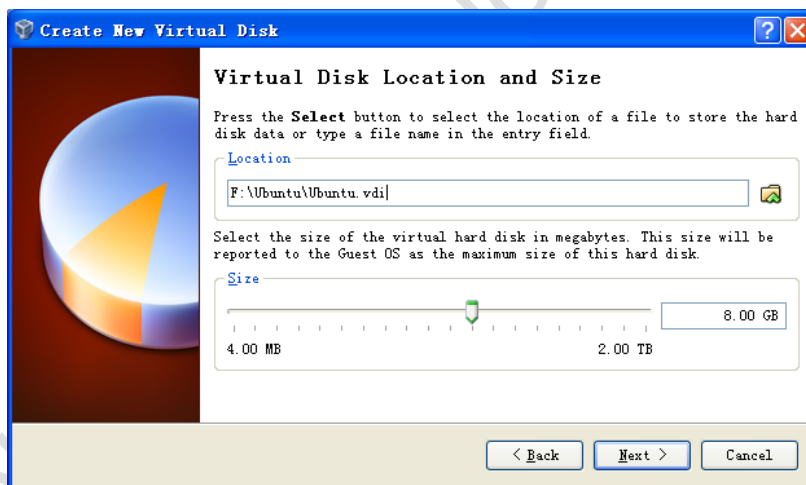
Figure 7 Wizard of new virtual disk creation

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



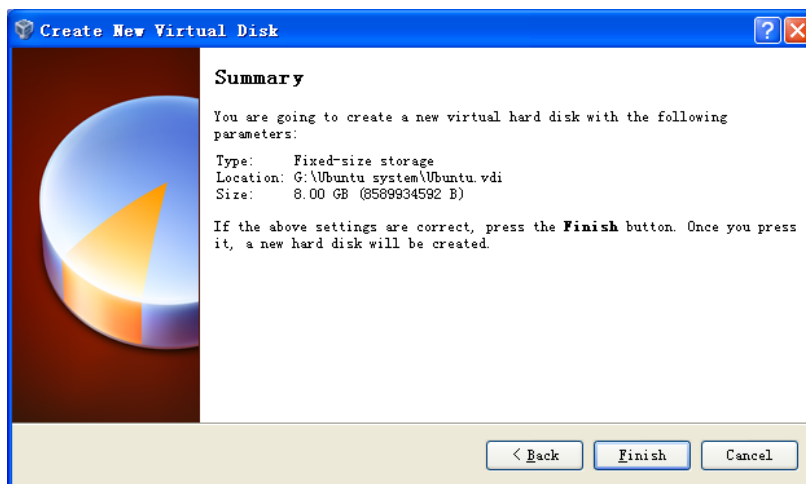
**Figure 8** Select the second option

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



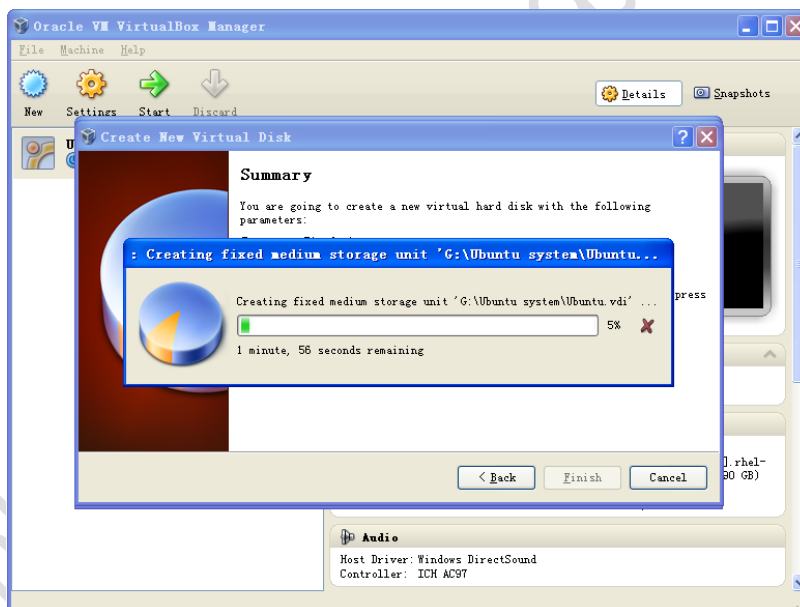
**Figure 9** Virtual disk configuration

- 8) Click Finish in the following window;



**Figure 10** Virtual disk summary

- 9) PC is creating a new virtual disk;



**Figure 11** 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.

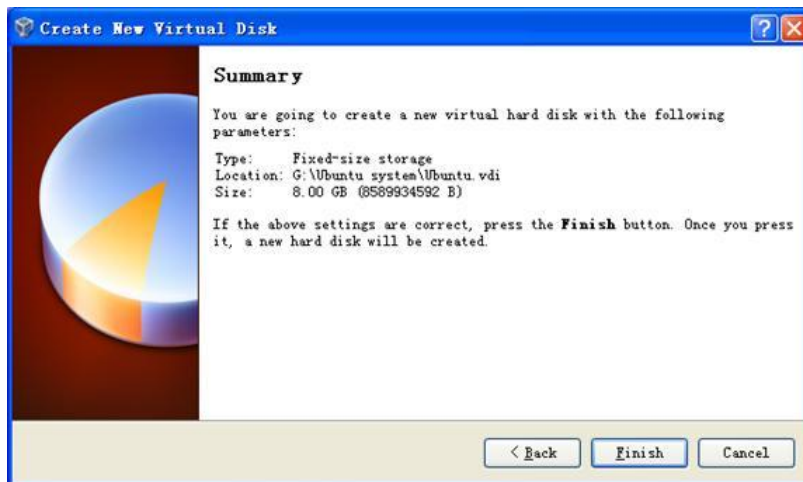


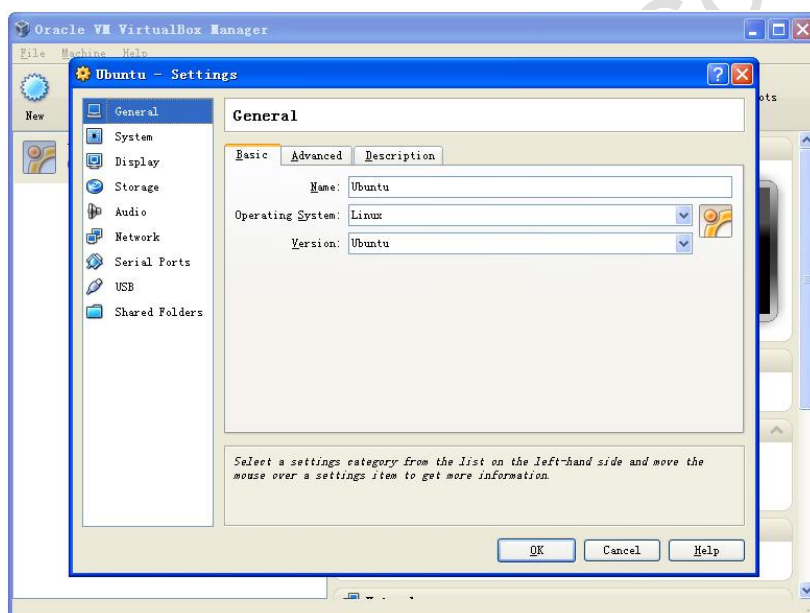
Figure 12 Virtual machine is ready



## Installing Ubuntu Linux System

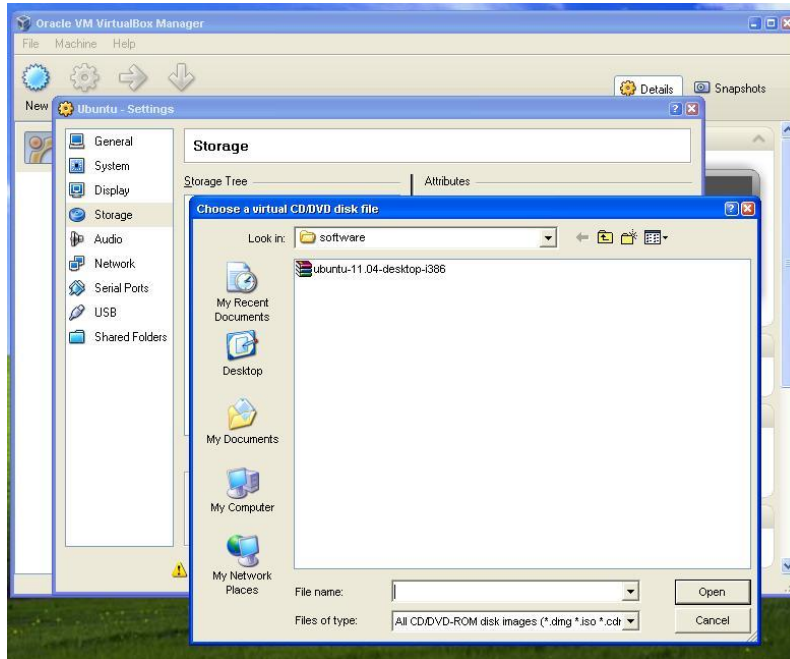
After virtualBox is installed, we can start the installation of Ubuntu Linux system now. Please access <http://www.Ubuntu.com/download/Ubuntu/download> to download the ISO image file of Ubuntu, and then follow the steps.

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



**Figure 13** Setting window

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



**Figure 14** Find ISO file

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

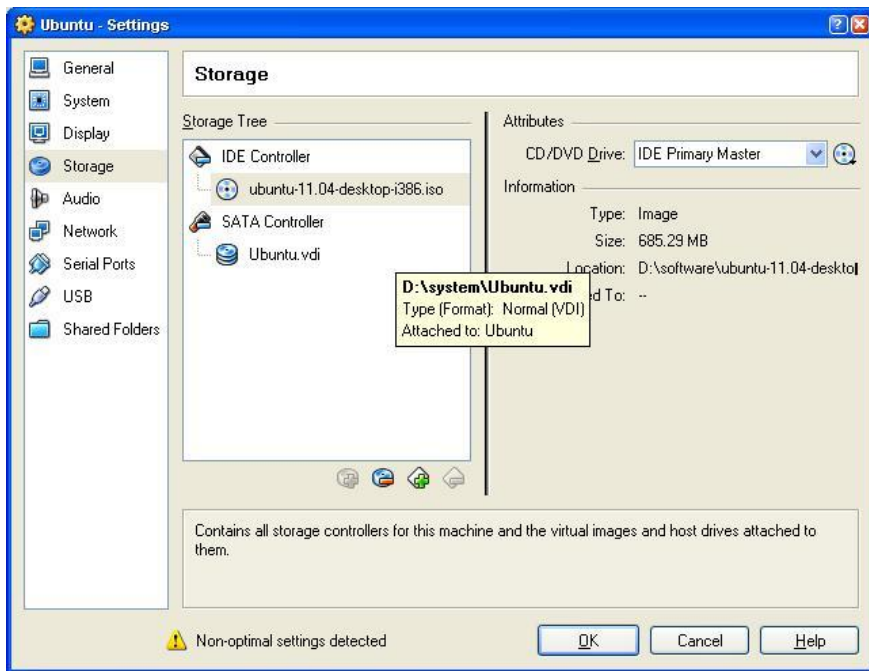
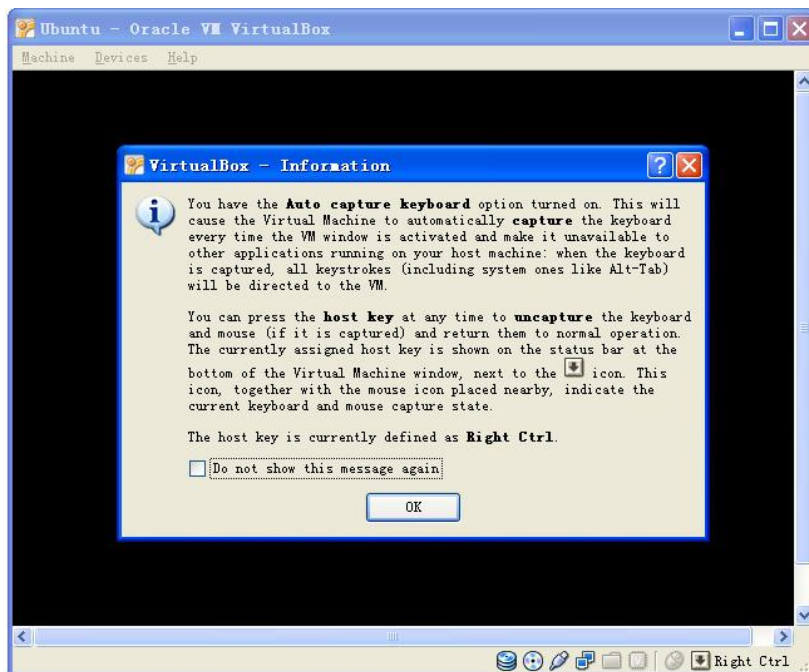


Figure 15 Select ISO file

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



**Figure 16** Ubuntu initiating window

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

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

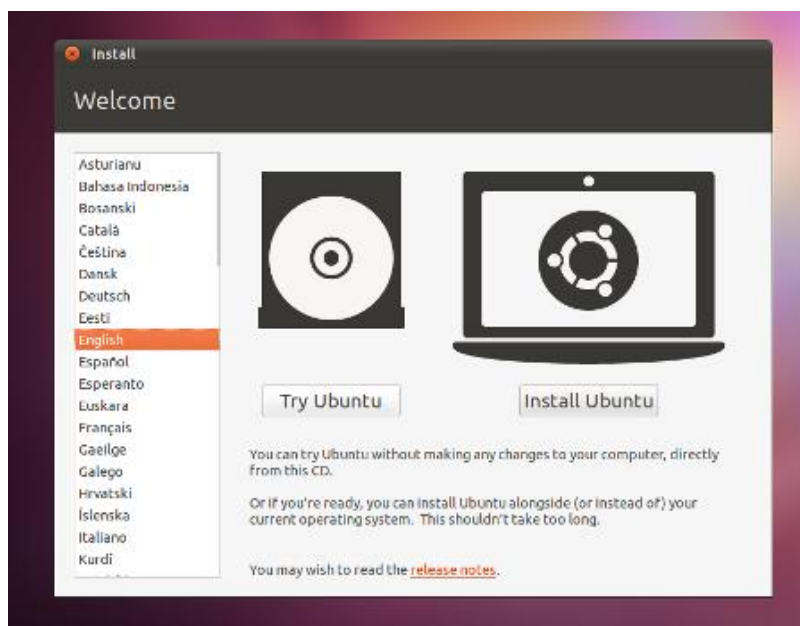


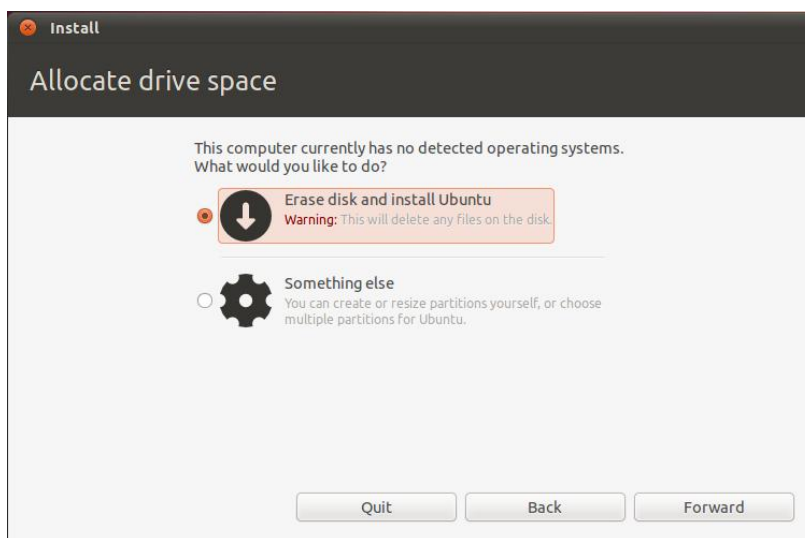
Figure 17 Ubuntu installation window

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




Figure 18 Information before installation

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

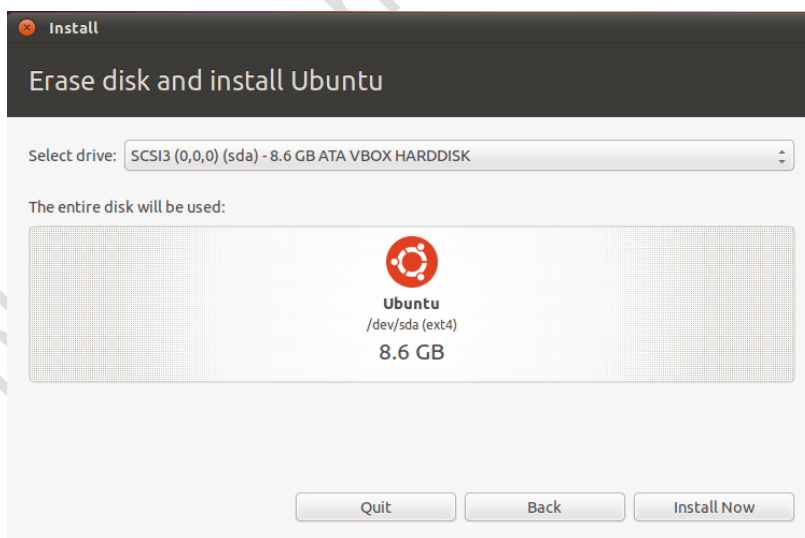


**Figure 19** Options before installation

**Note:**

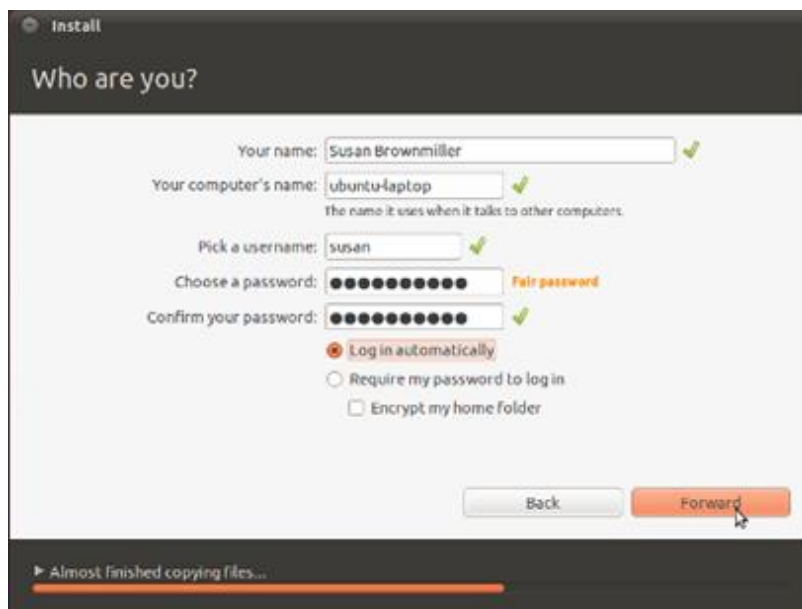
 Selecting this option will not lead to any content loss on your hard drive.

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



**Figure 20** 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 21** Enter appropriate information

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

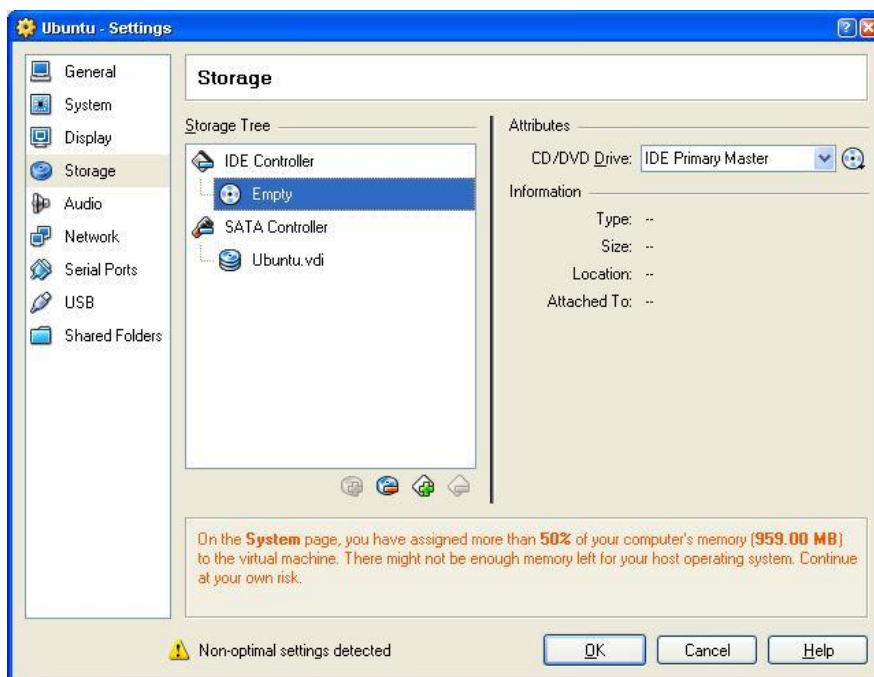
- 10) The installation of Ubuntu may take 15 minutes to about 1 hour depending on your PC's performance. A prompt window will be shown as below after installation is done. Please select **Restart Now** to restart Ubuntu system.



**Figure 22** Restart Ubuntu



- 11) Ubuntu system is ready for use after restarting. Normally the ISO file shown in Figure 15 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 23** ISO file ejected

## Appendix III Installation of Linux USB Ethernet/RNDIS Gadget

- 1) If you don't install driver of Linux USB Ethernet/RNDIS Gadget, PC will find the new hardware and give you a hint on the screen, please select "From list or designated location", then click "Next".



Figure 24

- 2) Designate a path for the usb driver, and the usb driver directory is [linux\tools] of the DVD-ROM, then click "Next"

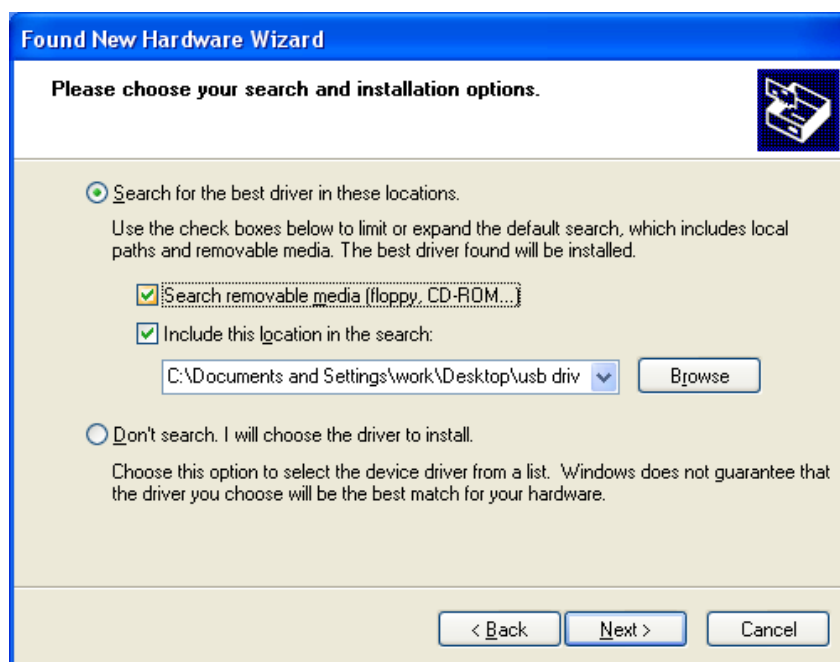


Figure 25

3) When the following appears, select “Continue”;



Figure 26

- 4) Please wait until the installation is completed;

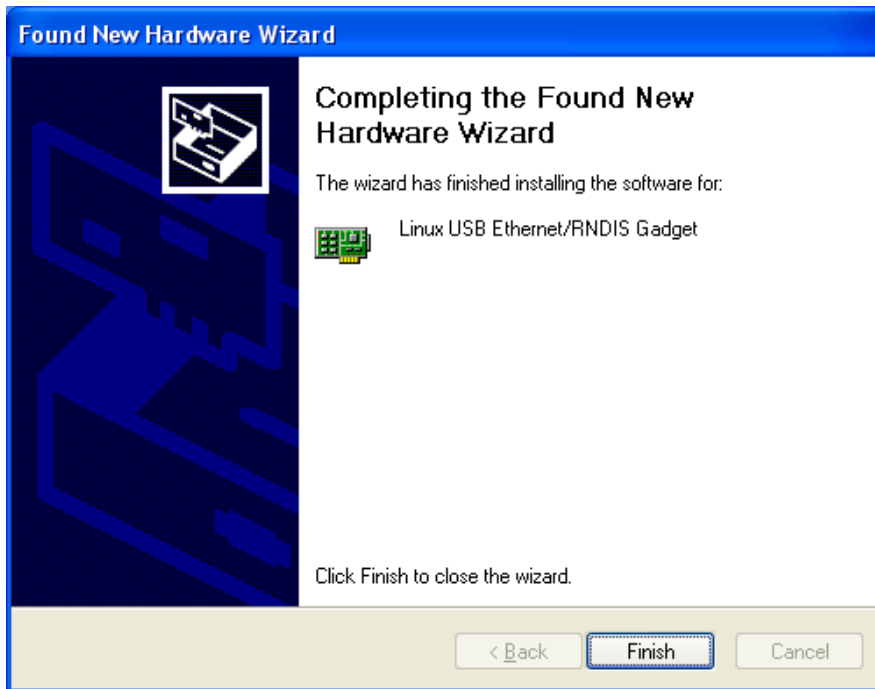


Figure 27

## Appendix IV Formatting Linux Boot Disk

How to create a dual-partition card for SBC8600B to boot Linux from first partition and have root file system at second partition.

### 1. Introduction

This guide is meant for those looking to create a **dual-partition** card, booting from a FAT partition that can be read by the OMAP3 ROM bootloader and Linux/Windows, then utilizing an ext3 partition for the Linux root file system.

### 2. Details

**Note:** Text marked with [] shows user input.

#### 1) Determine which device the TF Card Reader is on your system

Plug the TF Card into the TF Card Reader and then plug the TF Card Reader into your system. After doing that, do the following to determine which device it is on your system.

```

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

```

In this case, it shows up as /dev/sdc (note sdc inside the square brackets above).

#### 2) Check to see if the automounter has mounted the TF card

Note there may be more than one partition (only one shown in the example below).

```

$ [df -h]

```

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

Note the "Mounted on" field in the above and use that name in the umount commands below.

**3) If so, unmount it**

```
$ [umount /media/disk]
```

**4) Start fdisk**

Be sure to choose the whole device (/dev/sdc), not a single partition (/dev/sdc1).

```
$ [sudo fdisk /dev/sdc]
```

**5) Print the partition record**

So you know your starting point. **Make sure to write down the number of bytes on the card (in this example, 2021654528).**

```
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)
```

**6) Delete any partitions that are there already**

```
Command (m for help): [d]
Selected partition 1
```

**7) Set the Geometry of the TF Card**

If the print out above does not show 255 heads, 63 sectors/track, then do the following expert mode steps to redo the TF Card:

**a) Go into expert mode.**

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

**b) Set the number of heads to 255.**

```
Expert Command (m for help): [h]
Number of heads (1-256, default xxx): [255]
```

**c) Set the number of sectors to 63.**

```
Expert Command (m for help): [s]
Number of sectors (1-63, default xxx): [63]
```

**d) Now Calculate the number of Cylinders for your TF Card.**

```
#cylinders = FLOOR (the number of Bytes on the TF Card (from above) / 255 / 63 / 512 )
```

**e) Set the number of cylinders to the number calculated.**

```
Expert Command (m for help): [c]
Number of cylinders (1-256, default xxx): [enter the number you calculated]...
```

**f) Return to Normal mode.**

```
Expert Command (m for help): [r]
```

**8) Print the partition record to check your work**

```
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
```

**9) Create the FAT32 partition for booting and transferring files from Windows**

```
Command (m for help): [n]
Command action
  e   extended
  p   primary partition (1-4)
[p]
Partition number (1-4): [1]
First cylinder (1-245, default 1): [(press Enter)]
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-61, default 61): [+5]

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

10) Mark it as bootable

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

11) Create the Linux partition for the root file system

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

12) Print to Check Your Work

```
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
```

13) Save the new partition records on the TF Card

This is an important step. All the work up to now has been temporary.

```
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.

#### 14) Format the partitions

The two partitions are given the volume names LABEL1 and LABEL2 by these commands. You can substitute your own volume labels.

```


$ [sudo mkfs.msdos -F 32 /dev/sdc1 -n LABEL1]
mkfs.msdos 2.11 (12 Mar 2005)

$ [sudo mkfs.ext3 -L LABEL2 /dev/sdc2]
mke2fs 1.40-WIP (14-Nov-2006)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
195072 inodes, 389576 blocks
19478 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=402653184
12 block groups
32768 blocks per group, 32768 fragments per group
16256 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information:

```

#### Note:

 After formatting and dividing into FAT and EXT3 under Ubuntu system, the FAT needs reformatting under windows system, otherwise, start-up with TF card can be realized.

## Appendix V Setup of TFTP Server

### 1) Installing client

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

### 2) Installing inet

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

### 3) Configuring server

**Firstly**, create tftpboot under root directory, and set the properties as “any user can write and read”

```
$>cd /
$>sudo mkdir tftpboot
$>sudo chmod 777 tftpboot
```

**Secondly**, add a line in /etc/inetd.conf as shown below:

```
$>sudo vi /etc/inetd.conf //把下面的语句添加的此文件里
tftpd dgram udp wait root /usr/sbin/in.tftpd /usr/sbin/in.tftpd -s /tftpboot
```

**Then**, reload inetd process:

```
$>sudo /etc/init.d/inetd reload
```

**Finally**, enter directory /etc/xinetd.d/, and create a new file **tftp** and add some lines in it;

```
$>cd /etc/xinetd.d/ //进入目录 /etc/xinetd.d/
$>sudo touch tftp //新建文件 tftp
$>sudo vi tftp //编辑文件 tftp,把下面内容加入 tftp 文件中
service tftp
{
    disable = no
    socket_type = dgram
    protocol = udp
    wait = yes
    user = root
    server = /usr/sbin/in.tftpd
    server_args = -s /tftpboot -c
    per_source = 11
```

```
cps          = 100 2
}
```

- 4) Reboot the server:

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

- 5) Test the server

Test the server by creating a new file under /tftpboot

```
$>touch abc
```

Enter another folder

```
$>tftp 192.168.1.15 (192.168.1.15 为本机 IP)
$>tftp> get abc
```

A successful download indicates that the server is working properly.

## Appendix VI FAQ

Please visit [http://www.elinux.org/SBC8600\\_FAQ](http://www.elinux.org/SBC8600_FAQ)

Embest Technology Co., Ltd

# 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:
  - A. Customers fail to provide valid purchase vouchers or the product identification tag is damaged, unreadable, altered or inconsistent with the products.
  - B. Products are damaged caused by operations inconsistent with the user manual;
  - C. 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;
  - D. Products are damaged in appearance or function caused by power failure, external forces, water, animals or foreign materials;
  - E. 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;
  - F. Product failures caused by the software or system installed by customers or inappropriate settings of software or computer viruses;
  - G. Products purchased from unauthorized sales;
  - H. 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

**Hotline:** +86-755-25635626-872/875/897

**Fax:** +86-755-25635626-666

**Pre-sales:** [sales@embedinfo.com](mailto:sales@embedinfo.com)

**After-sales:** [support@embedinfo.com](mailto:support@embedinfo.com)

**Website:** <http://www.armkits.com> or <http://www.embest-tech.com>

**Address:** Tower B 4/F, Shanshui Building, Nanshan Yungu Innovation Industry Park,  
Liuxian Ave. No. 1183, Taoyuan St., Nanshan District, Shenzhen, China (518055)