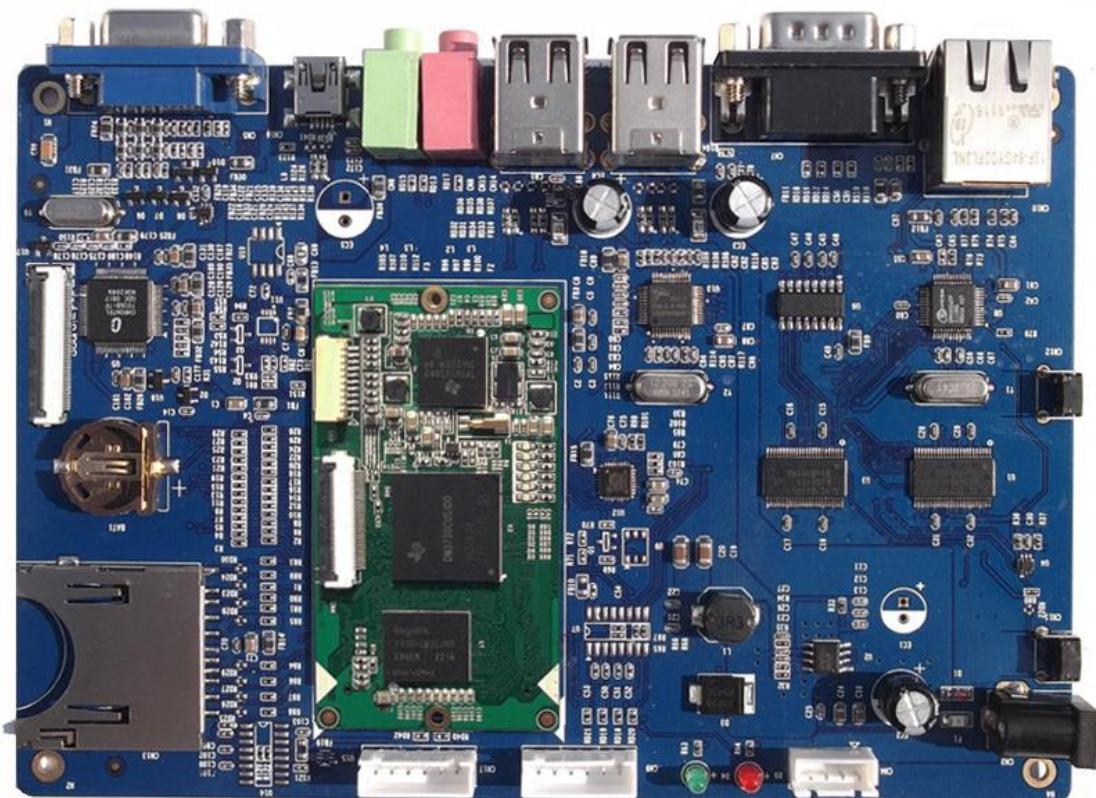


# **SBC8140**

# **Single Board Computer**



## **User Manual**

---

Version 1.0 – Mar. 20<sup>th</sup>, 2013

# Copyright Statement:

- SBC8140 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.
- The use of Microsoft, MS-DOS, Windows, Windows95, Windows98, Windows2000 and Windows embedded CE 6.0 are authorized by Microsoft.

# Disclaimer:

- Shenzhen Embest Techology does not take warranty of any kind, either expressed or implied, as to the program source code, software and documents in the CD/DVD-ROMs provided along with the products, and including, but not limited to, warranties of fitness for a particular purpose; The entire risk as to the quality or performance of the program is with the user of products; Should the program prove defective, the user of products assumes the cost of all necessary servicing, repair or correction.

# Revision History:

Version	Date	Note
1.0	2013-3-20	Original Version

# Table of Contents

<b>Chapter 1 Product Overview .....</b>	<b>1</b>
1.1 Introduction.....	1
1.2 Packing List .....	1
1.3 Product Features .....	2
1.3.1 Mini8510 Core Board .....	2
1.3.2 Expansion Board.....	4
1.4 Interfaces on SBC8140 .....	6
1.5 System Block Diagram .....	7
1.6 Hardware Dimensions (mm).....	8
1.6.1 Mini8510 Core Board .....	8
1.6.2 Expansion Board.....	9
1.7 Modules Supported by SBC8140.....	9
<b>Chapter 2 Introduction to Hardware .....</b>	<b>11</b>
2.1 CPU Introduction .....	11
2.1.1 Clock .....	11
2.1.2 Reset .....	11
2.1.3 General Interfaces.....	12
2.1.4 Display Subsystem.....	12
2.1.5 3D Graphics Acceleration System .....	12
2.2 Peripheral ICs around CPU.....	13
2.2.1 TPS65930 Power Management IC .....	13
2.2.2 H9DA4GH2GJAMCR Memory.....	14
2.2.3 DM9000 Ethernet Controller .....	14
2.2.4 FE1.1 USB Hub.....	14
2.2.5 TFP410 Flat Panel Display IC.....	14
2.2.6 MAX3232 Tranceiver.....	15
2.3 Hardware Interfaces and LEDs on Mini8510 .....	16
2.3.1 CN1 90pin DIP Interface (right row).....	16
2.3.2 CN2 90pin DIP Interface (left row) .....	19
2.3.3 CN3 JTAG Interface .....	22
2.3.4 CN4 Camera Interface .....	22

2.3.5 LED Indicators.....	23
2.4 Interfaces on Expansion Board .....	24
2.4.1 Power Jack.....	24
2.4.2 TFT_LCD Interface .....	24
2.4.3 Audio Output Interface .....	26
2.4.4 Audio Input Interface .....	26
2.4.5 Serial Interface .....	26
2.4.6 Ethernet Interface.....	27
2.4.7 USB OTG Interface .....	27
2.4.8 USB HOST Interface.....	27
2.4.9 SD Card Interface .....	28
2.4.10 LED Indicators.....	28
2.4.11 Buttons .....	28
<b>Chapter 3 Linux Operating System.....</b>	<b>29</b>
3.1 Structure of Embedded Linux System.....	29
3.2 Software Features .....	30
3.3 System Development Process .....	31
3.3.1 Building Development Environment.....	31
3.3.2 System Compilation .....	32
3.3.3 Customizing System .....	35
3.4 Introduction to Drivers .....	36
3.4.1 NAND Flash Driver.....	38
3.4.2 SD/MMC Driver .....	39
3.4.3 Display Subsystem Driver.....	40
3.4.4 Video Capture Driver.....	41
3.4.5 Audio Input/Output Driver.....	43
3.5 Driver Development.....	44
3.5.1 GPIO_Keys Driver.....	44
3.5.2 GPIO_LEDs Driver.....	49
3.6 System Update .....	52
3.6.1 Updating System in SD Card .....	53
3.6.2 Updating Syetem in NAND Flash.....	61
3.7 Display Mode Configurations .....	63

---

3.8 Tests and Demonstrations .....	65
3.8.1 Testing LED .....	66
3.8.2 Testing Touch-Screen.....	66
3.8.3 Testing RTC.....	67
3.8.4 Testing SD Card .....	68
3.8.5 Testing USB Device .....	68
3.8.6 Testing USB HOST .....	71
3.8.7 Testing Audio Function .....	71
3.8.8 Testing Network.....	73
3.8.9 Testing Carmera .....	74
3.8.10 Testing CDMA8000-U Module .....	75
3.8.11 Testing WCDMA8000-U Module .....	75
3.8.12 Demonstration of Android System .....	75
3.8.13 Demonstration of DVSDK System .....	77
3.9 Development of Applications .....	79
<b>Chapter 4 WinCE Operating System.....</b>	<b>82</b>
4.1 Software Resources .....	82
4.2 BSP Package.....	83
4.3 Process of System Development .....	84
4.3.1 Installing IDE .....	84
4.3.2 Uncompressing/Copying BSP and Exmaple Project .....	85
4.3.3 Compiling Sysgen and BSP .....	85
4.4 Introduction to Drivers .....	86
4.5 System Update .....	88
4.5.1 Updating System in SD Card .....	88
4.5.2 Updating Syetem in NAND Flash.....	91
4.6 Other Operations .....	92
4.6.1 OpenGL ES demo .....	92
4.6.2 CAM8000-A Module .....	93
4.6.3 CAM8000-D Module.....	94
4.7 GPIO API and Example Applications .....	96
<b>Appendix 1 – Installing Ubuntu Linux System .....</b>	<b>99</b>

---

<b>Appendix 2 – Installing Linux USB Ethernet/RNDIS Gadget Driver .....</b>	<b>111</b>
<b>Appendix 3 – Making Linux Boot Disk .....</b>	<b>114</b>
<b>Appendix 4 – Building TFTP Server.....</b>	<b>119</b>
<b>Appendix 5 – FAQ .....</b>	<b>121</b>
<b>Technical Support and Warranty.....</b>	<b>122</b>

# Chapter 1 Product Overview

## 1.1 Introduction

SBC8140 is a DM3730 processor based development kit designed by Ernest Technoogy.

This kit consists of a core board Mini8510 and an expansion board. Mini8510 is basically made up of a processor, MCP (includes DDR and NAND memory) and power management, forming a minimum system on TI DM3730. Expansion board can be connected to the core board with two 90-pin 1.27mm-pitch DIP connectors so as to provide a variety of extension interfaces such as network, audio input/output, USB, SD card, serial connecton, VGA, JTAG, camera, LCD and touch-screen.

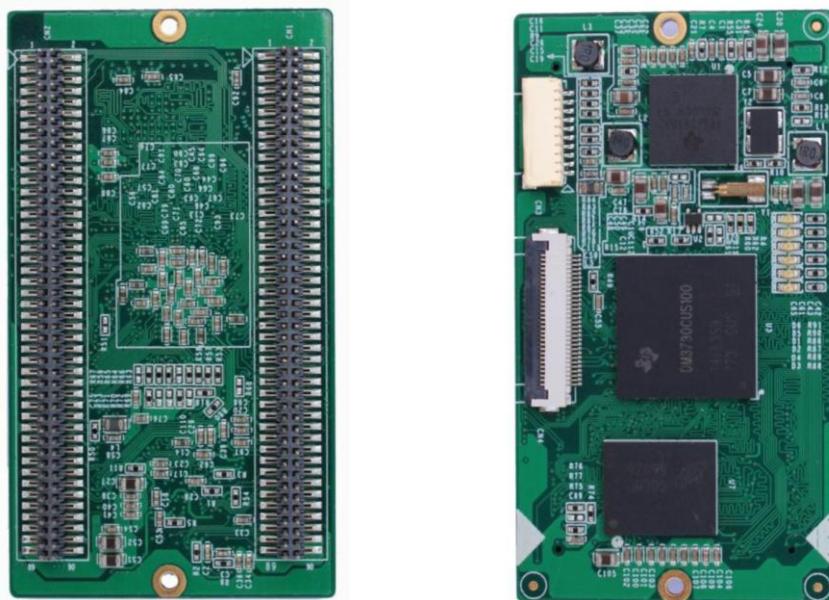
SBC8140 is designed to satisfy the different requirements of various fields including security, monitoring, access control communications, POS terminals, network terminals and advertising signs.

## 1.2 Packing List

- SBC8140 x 1
- Cross-over serial cable (DB9 to DB9) x 1
- 10-pin JTAG cable x 1
- JTAG8000 module x 1
- 5V/2A power adapter x 1
- DVD-ROM x 1
- LCD screen x 1 (optional item, available in 4.3-inch 480\*272 or 7-inch 800\*480)

## 1.3 Product Features

### 1.3.1 Mini8510 Core Board



**Figure 1-1** Back Side (left pic) and top side (right pic) of Mini8510 Core Board

- **Product Parameters:**

- Dimensions: 67mm x 37mm
- Operation Temperature: 0 ~ 70°C
- Operating Humidity: 20% ~ 90% (Non-condensing)
- Power Supply: 3.3V/0.17A

- **Processor:**

- TI DM3730 integrating 1GHz ARM Cortex™-A8 core
- Integrating 800-MHz TMS320C64x+™ DSP
- Integrating NEON™ SIMD co-processor
- Integrating POWERVR SGX™ graphic accelerator
- Integrating 32KB instruction buffer, 32KB data buffer, 256KB L2 cache, 64KB RAM and 32KB ROM

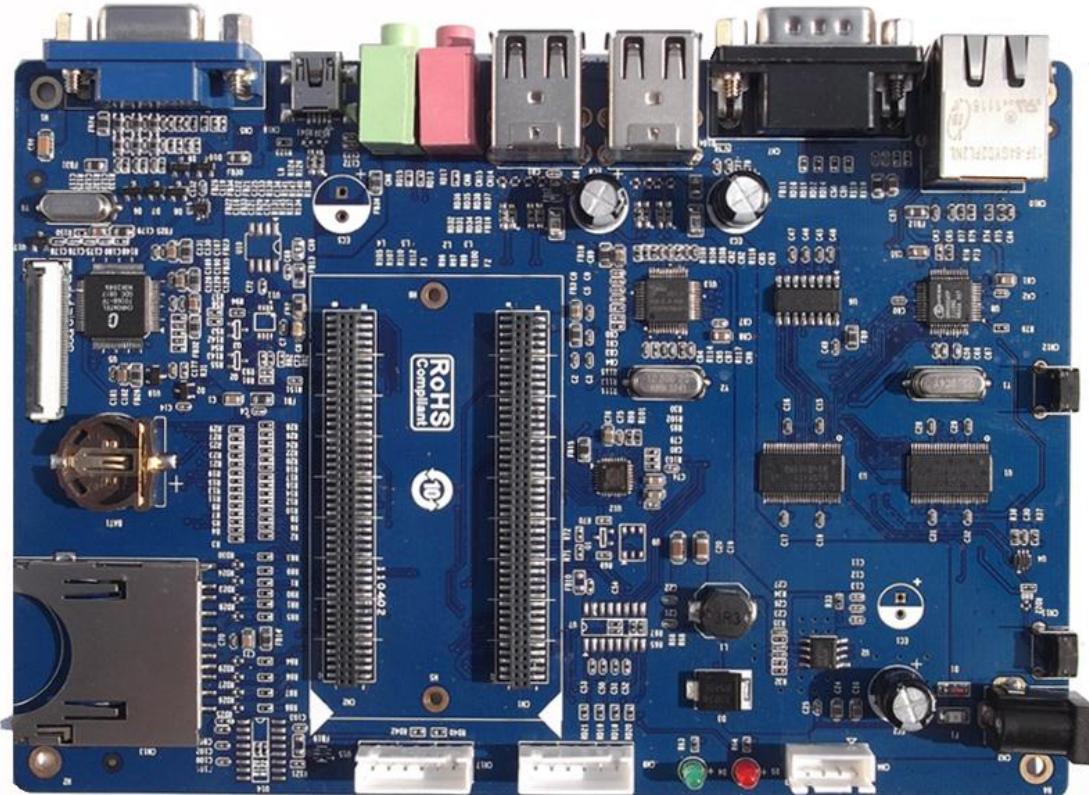
- **On-Board Memories:**

- 256MB 32bit DDR SDRAM
- 512MB 16bit NAND Flash

- **Interfaces and Signals**

- A camera interface (supports external CCD or CMOS camera)
- A JTAG interface
- Two 1.27mm-pitch 90-pin DIP connectors
- Six LED indicators (two power indicators and four user custom indicators)
- Two SPIs: SPI1 and SPI2
- GPMC bus (16-bit data, 10-bit address, four CS and some control signals)
- Three UARTs (5-wire, support hardware flow control)
- A ULPI (USB1 HS)
- Audio input and output
- A IIC bus (IIC3)
- Two McBSPs: McBSP1 and McBSP3 (McBSP3 is multiplexed on UART2)
- Two MMCs/SDs:MMC1 (8-wire) and MMC2 (4-wire)
- 24-bit DSS interface

### 1.3.2 Expansion Board



**Figure 1-2** Expansion board of SBC8140

- **Product Parameter:**

- Dimensions: 165mm x 115mm
  - Operation temperature: 0 ~ 70° C
  - Operating Humidity: 20% ~ 90% (Non-condensing)
  - Power Supply: 5V/2A

### ● **Audio/Video Interfaces:**

- LCD/touch-screen interface (24-bit RGB full-color output; 50-pin FPC connector)
  - A standard VGA interface, support 1024\*768 resolution by default
  - A audio input interface (3.5mm audio jack)
  - A dual-channel audio output interface (3.5mm audio jack)

- Data transfer interface:

- A 10/100Mbps Ethernet interface (RJ45 connector)
- A high-speed USB 2.0 OTG interface with PHY (480Mbps mini-USB interface)
- Four high-speed USB 2.0 HOST interfaces with PHY (480Mbps USB-A interface)
- A SD card slot (compatible with SD/MMC communication)
- Serial Interface

**Table 1-1 Serial Interfaces**

Interfaces	Descriptions
<b>UART1</b>	5-wire, RS232 voltage level, DB9 debugging serial interface
<b>UART2</b>	3-wire, TTL voltage level, 6-pin connector
<b>UART3</b>	5-wire, RS232 voltage level, 6-pin connector

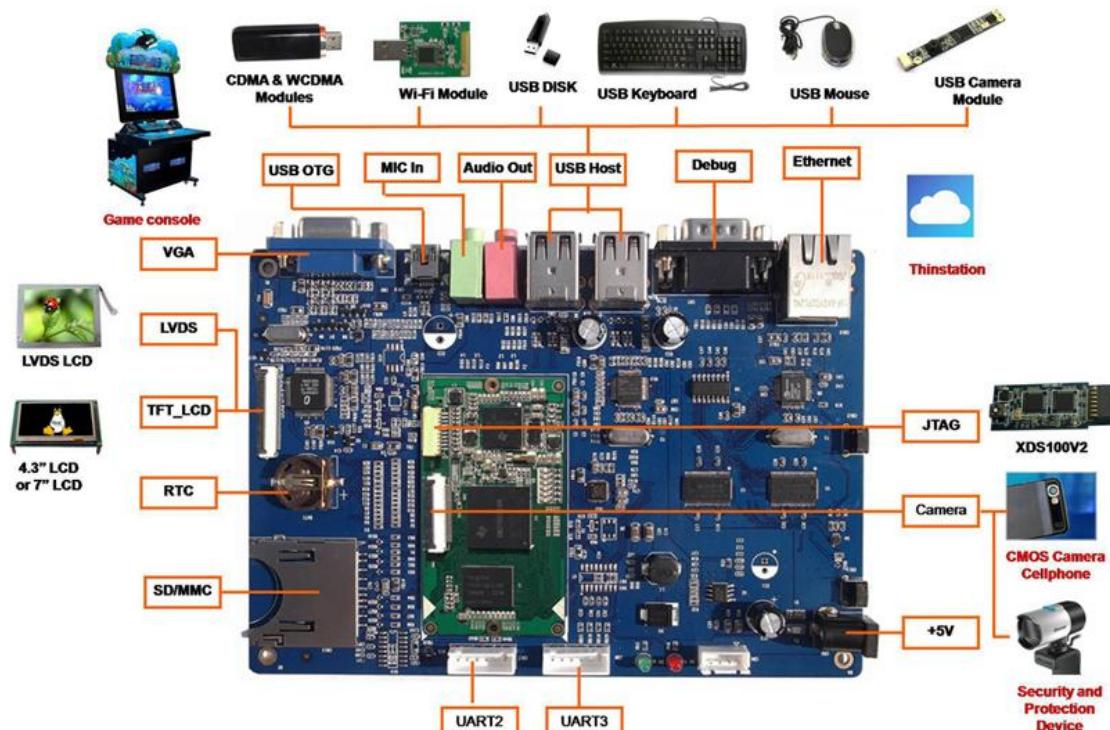
● **Input Interface:**

- A BOOT button
- A reset button

● **LED indicators**

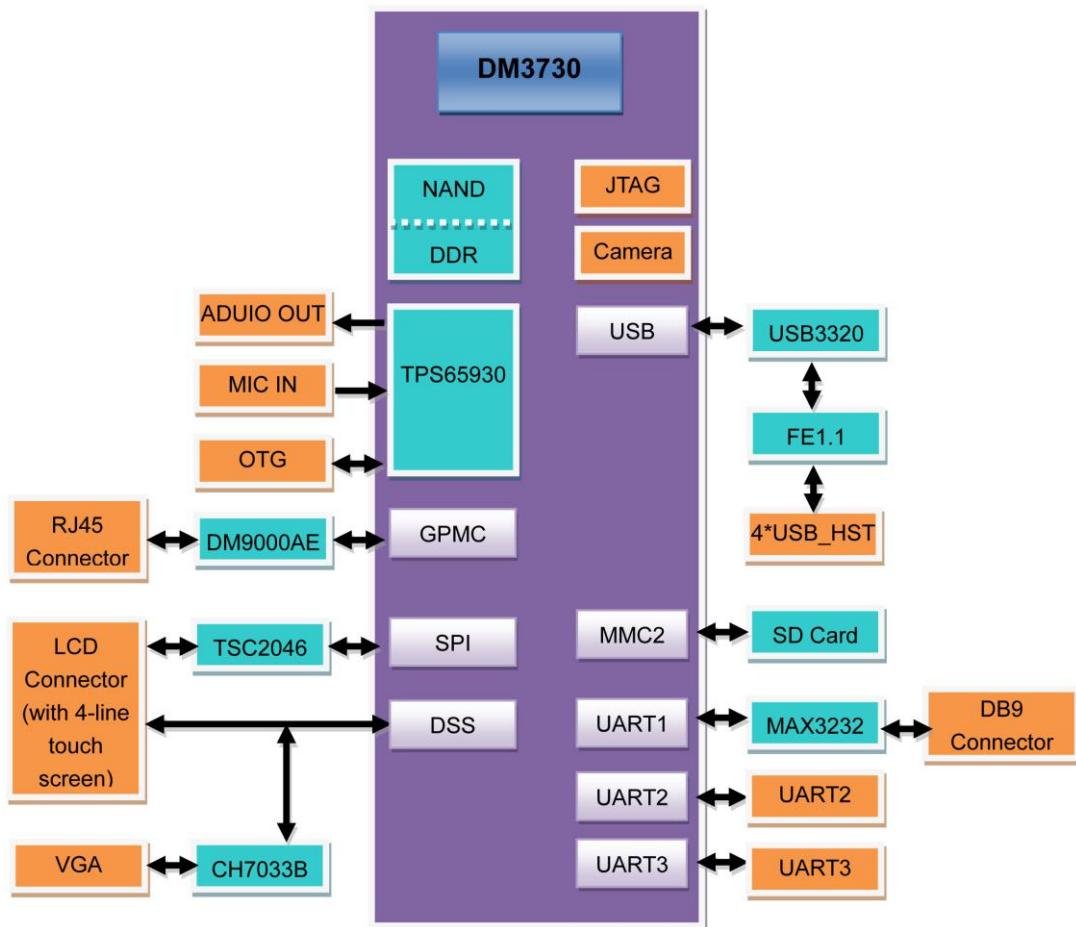
- A power indicator
- Two user custom indicators

## 1.4 Interfaces on SBC8140



**Figure 1-3 SBC8140 interfaces**

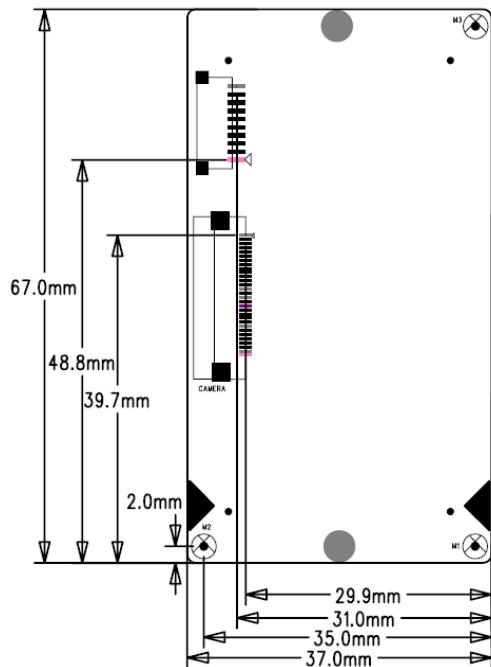
## 1.5 System Block Diagram



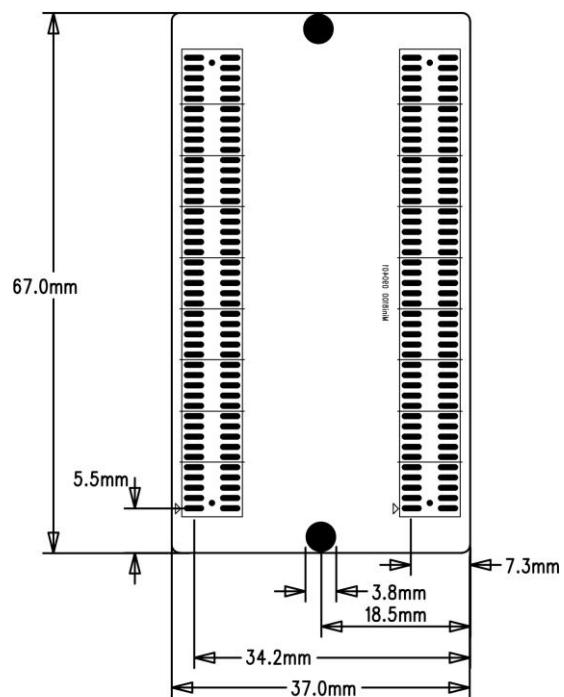
**Figure 1-4** SBC8140 system block diagram

## 1.6 Hardware Dimensions (mm)

### 1.6.1 Mini8510 Core Board

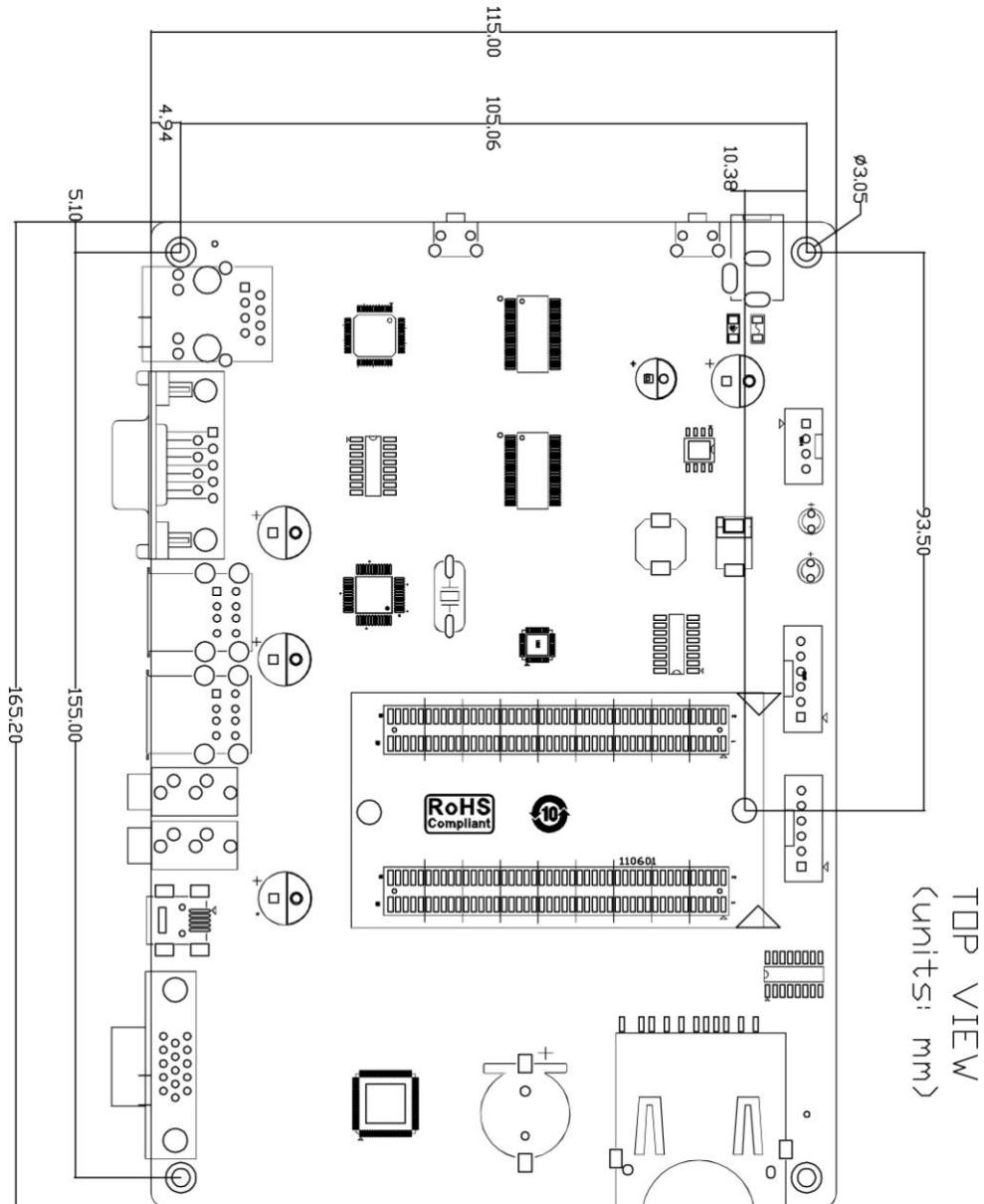


**Figure 1-5** Mini8510 dimensions (top side)



**Figure 1-6** Mini8510 dimensions (back side)

### 1.6.2 Expansion Board



**Figure 1-7 Expansion board dimension**

### 1.7 Modules Supported by SBC8140

**Table 1-2 Supported Modules**

Modules	Linux	Android	WinCE	Materials
WF8000-U/	Yes	No	No	Available in CD
CAM8100-U	Yes	No	No	Available in CD
CDMA8000-U	Yes	No	No	<a href="#">Click to download</a>

<b>Modules</b>	<b>Linux</b>	<b>Android</b>	<b>WinCE</b>	<b>Materials</b>
<b>WCDMA8000-U</b>	Yes	No	No	<a href="#">Click to Download</a>
<b>LVDS8000</b>	Yes	Yes	Yes	CD and Website

<b>Modules</b>	<b>Linux</b>	<b>Android</b>	<b>WinCE</b>	<b>Materials</b>
<b>WF8000-U</b>	Yes*	NO	Yes#	Provided with CD-ROM Separately
<b>CAM8000-A</b>	Yes*	Yes*	Yes*	Available in CD
<b>CAM8000-D</b>	Yes#	NO	Yes*	<a href="#">Click to download</a>
<b>CAM8100-U</b>	Yes*	Yes*	Yes#	Provided with CD-ROM Separately
<b>CDMA8000-U</b>	Yes*	No	Yes#	<a href="#">Click to download</a>
<b>WCDMA8000-U</b>	Yes*	No	Yes#	<a href="#">Click to download</a>
<b>LVDS8000</b>	Yes*	Yes*	Yes*	Available in CD and on website

# Chapter 2 Introduction to Hardware

This chapter will help you learn about the hardware composition of Mini8510 core board by briefly introduce CPU, peripheral ICs and pin definition of various interface on the product (Mini8510+Expansion board).

## 2.1 CPU Introduction

Mini8510 core board uses DM3730 – a TI's 45-nm high-performance processor with low power and enhanced digital media processing capability. The CPU has a 1GHz Cortex-A8 core and a 800MHz TMS320C64+ DSP core, and also integrates a 3D graphics processing unit, a imaging and video accelerator and USB 2.0, making it capable of 720p video coding and decoding.

### 2.1.1 Clock

The clock signals of DM3730 include sys\_32k, sys\_altclk, sys\_clkout1, sys\_clkout2, sys\_xtalout, sys\_xtalin and sys\_clkreq, among which:

- **sys\_32k:** the frequency is 32KHz, generated by power management chip TPS65930, and used for low-frequency calculation; low-power mode is enabled through sys\_32k pin.
- **sys\_xtalou** and **sys\_xtalin:** they are system input clocks with frequency at 26MHz and used to provide primary clocks for DLLs and other module.

### 2.1.2 Reset

Reset singal is determined by SYS\_NRESPWRON of CPU; low level validates reseting.

### **2.1.3 General Interfaces**

General interfaces include 6 sets of GPIOs, each of which provides 32 dedicated GPIO pins, and therefore the total pin number of GPIO can be up to 192 ( $6 \times 32$ ). These pins can be configured for different applications such as data input/output (driver), keypad interface and terminal control.

### **2.1.4 Display Subsystem**

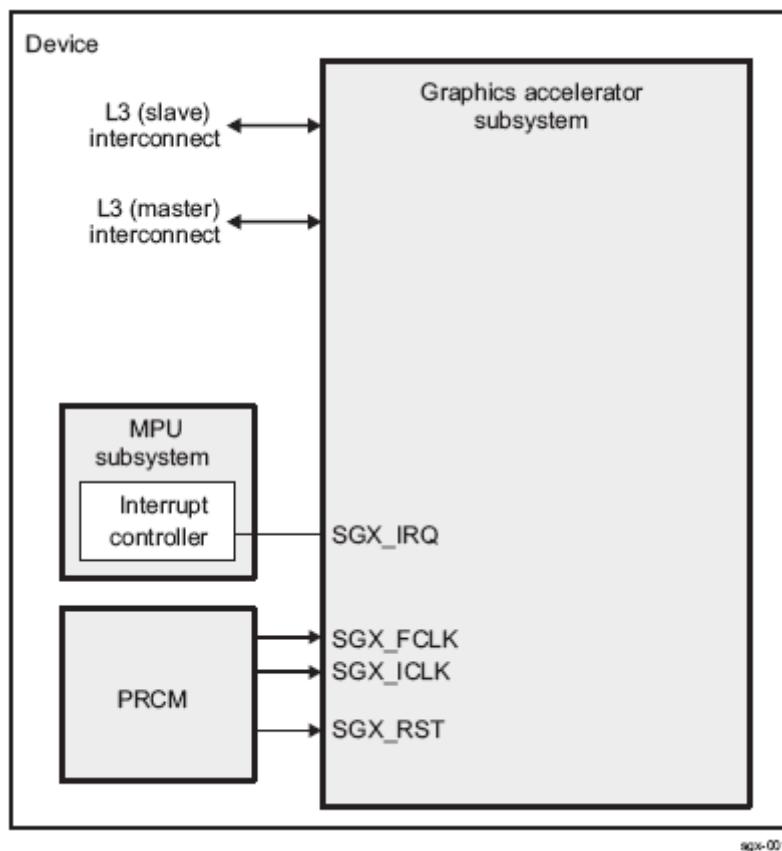
Display subsystem is used to provide LCD or TV interface with logic images which are stored in frame buffer (SDRAM or SRAM); it is made up of:

- Display control (DISPC) module
- Remote frame buffering interface (RFBI) module
- I/O module and DSI protocol engine of display serial interface (DSI)
- DSI PLL controller driver (DSI PLL and high-speed frequency divider)
- NTSC/PAL video codec

Display controller and DSI protocol engine are connected to the internal bus of L3 and L4, while RFBI and TV output codec module are connected to the internal bus of L.

### **2.1.5 3D Graphics Acceleration System**

2D/3D graphics acceleration system (SGX) can speed up 2D/3D graphic applications. SGX system is built on the POWERVR® SGX core from Imagination Technologies. It is a new-generation of programmable POWERVR graphics core. POWERVR SGX530 v1.2.5 has an adaptable architecture which makes it suited for a wide range of applications from main-stream mobile devices to high-end desk-top graphics processing. Its target applications are mainly feature phones, PDAs and some portable game consoles.



**Figure 2-1** SGX graphics acceleration system

The architecture of SGX graphics acceleration system realizes switching among multiple threads by adopting two-level scheduling and data partitioning, so that it is capable of processing pixels, vertexs, videos and general data.

## 2.2 Peripheral ICs around CPU

### 2.2.1 TPS65930 Power Management IC

The TPS65930 is a power-management IC for OMAP families. The device includes power-management, a USB high-speed transceiver, LED drivers, an analog-to-digital converter (ADC), a real-time clock (RTC), and embedded power control (EPC). In addition, the TPS65930 includes a full audio codec with two digital-to-analog converters (DACs) and two ADCs to implement dual voice channels, and a stereo downlink channel that can play all standard audio sample rates through a multiple format inter-integrated

sound (I2S™)/time division multiplexing (TDM) interface.

TPS65930 communicates with CPU through I2C protocol. It supplies 1.2V and 1.8V to keep CPU working properly. Additionally, TPS65930 is featured with Audio in, Audio out, OTG PHY, Keyboard, ADC and GPIO.

### **2.2.2 H9DA4GH2GJAMCR Memory**

H9DA4GH2GJAMCR is a two-in-one memory which combines 512MB NAND Flash and 256MB SDRAM DDR. The NAND Flash is accessed through GPMC bus, while SDRAM is accessed through Controller (SDRC).

### **2.2.3 DM9000 Ethernet Controller**

The DM9000 is a fully integrated fast Ethernet controller with a general processor interface, a 10/100M PHY and 4K DWORD SRAM. It supports 3.3V with 5V tolerance.

SBC8140 uses the 10/100M self-adaptive network interface of DM8000 which is a standard RJ45 interface with connection and data transfer indicators. The 10/100M Ethernet module integrated in DM9000 is compliant with IEEE 802.3 standard.

SBC8140 can be either connected to a hub with a straight-through network cable, or to a PC with a cross-over network cable.

### **2.2.4 FE1.1 USB Hub**

FE1.1 is a USB 2.0 high-speed 4-port hub solution. It uses USB3320 to provide 4 extended USB interface with support for high-speed (480MHz), full-speed (2MHz) and low-speed (1.5MHz) mode.

### **2.2.5 TFP410 Flat Panel Display IC**

The TFP410 is a Texas Instruments PanelBus flat panel display product, part of a comprehensive family of end-to-end DVI 1.0-compliant solutions, targeted at the PC and consumer electronics industry.

The TFP410 provides a universal interface to allow a glue-less connection to most commonly available graphics controllers. Some of the advantages of this universal interface include selectable bus widths, adjustable signal levels, and differential and single-ended clocking. The adjustable 1.1V to 1.8V digital interface provides a low-EMI, high-speed bus that connects seamlessly with 12-bit or 24-bit interfaces. The DVI interface supports flat panel display resolutions up to UXGA at 165 MHz in 24-bit true color pixel format.

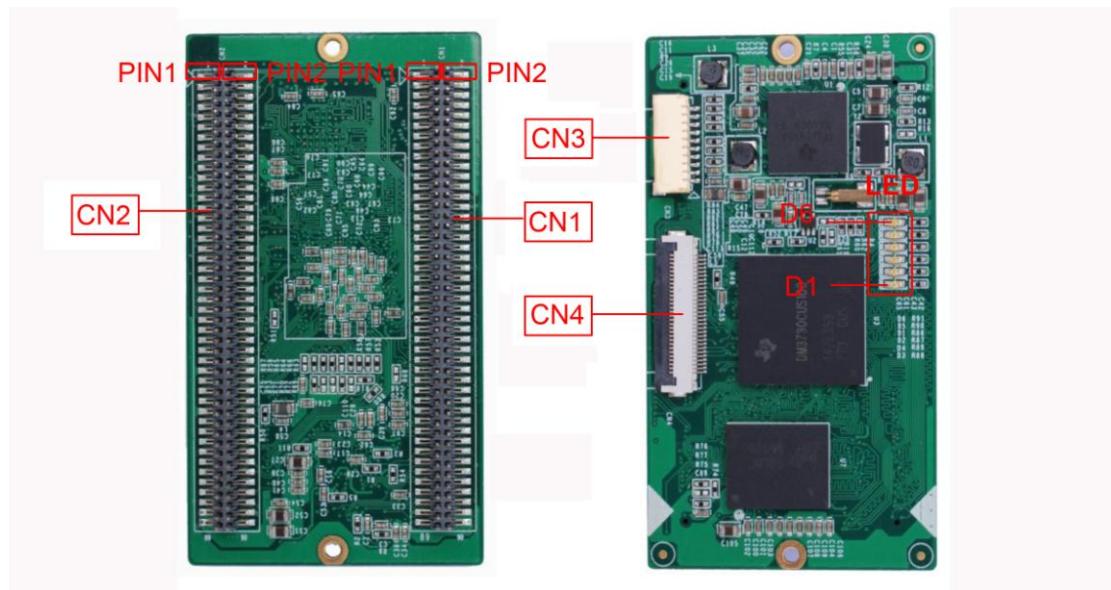
### **2.2.6 MAX3232 Tranceiver**

The MAX3232 transceiver has a proprietary low-dropout transmitter output stage enabling true RS-232 performance from a 3.0V to 5.5V supply with a dual charge pump. The devices require only four small 0.1 $\mu$ F external charge-pump capacitors. The MAX3232 is guaranteed to run at data rates of 120kbps while maintaining RS-232 output levels.

The MAX3232 has 2 receivers and 2 drivers. It features a 1 $\mu$ A shutdown mode that reduces power consumption and extends battery life in portable systems. Its receivers remain active in shutdown mode, allowing external devices such as modems to be monitored using only 1 $\mu$ A supply current. The MAX3232 is pin, package, and functionally compatible with the industry-standard MAX242 and MAX232, respectively. It is able to ensure  $\pm 5$ V transmission voltage which is the lowest requirement by RS-232 standard even when working at a hight data rate.

The MAX3232 guarantee a 120kbps data rate with worst-case loads. Typically, it can operate at a data rate of 235kbps. The transmitter can be paralleled to drive multiple receivers or mice.

## 2.3 Hardware Interfaces and LEDs on Mini8510



**Figure 2-2 Mini8510**

### 2.3.1 CN1 90pin DIP Interface (right row)

**Table 2-1 CN1 90pin DIP Interface (right row)**

Pins	Definitions	Descriptions
1	GND1	GND
2	G_D14	GPMC data bit 14
3	G_D13	GPMC data bit 13
4	G_D10	GPMC data bit 10
5	G_D8	GPMC data bit 8
6	G_D9	GPMC data bit 9
7	G_D5	GPMC data bit 5
8	G_D7	GPMC data bit 7
9	G_D3	GPMC data bit 3
10	G_D6	GPMC data bit 6
11	G_D12	GPMC data bit 12
12	G_D2	GPMC data bit 2
13	G_D11	GPMC data bit 11
14	G_D1	GPMC data bit 1
15	G_D4	GPMC data bit 4
16	G_D0	GPMC data bit 0
17	G_A2	GPMC address bit 2

<b>Pins</b>	<b>Definitions</b>	<b>Descriptions</b>
18	G_A3	GPMC address bit 3
19	G_A1	GPMC address bit 1
20	G_A6	GPMC address bit 6
21	G_A4	GPMC address bit 4
22	G_A7	GPMC address bit 7
23	G_A5	GPMC address bit 5
24	G_A8	GPMC address bit 8
25	G_A9	GPMC address bit 9
26	G_D15	GPMC data bit 15
27	G_A10	GPMC address bit 10
28	GND2	GND
29	SPI2_CS1/GPT8	SPI Enable 1PWM or event for GP timer 8
30	SPI2_CS10/GPT11	SPI Enable 0PWM or event for GP timer 11
31	SPI2_SIMO/GPT9	Slave data in, master data out PWM or event for GP timer 9
32	SPI2_CLK	SPI Clock
33	SPI2_SOMI GPT10	Slave data out, master data inPWM or event for GP timer 10
34	SPI1_CS3	SPI Enable 3
35	SPI1_CS0	SPI Enable 0
36	SPI1_SIMO	Slave data in, master data out
37	SPI1_SOMI	Slave data out, master data in
38	SPI1_CLK	SPI Clock
39	GND3	GND
40	GPIO0	GPIO0 /card detection 1
41	MMC2_D2/SPI3_CS1	MMC/SD Card Data bit 2SPI Enable 1
42	MMC2_D3/SPI3_CS0	MMC/SD Card Data bit 3SPI Enable 0
43	MMC2_D0/SPI3_SOMI	MMC/SD Card Data bit 0Slave data out, master data in
44	MMC2_D1	MMC/SD Card Data bit 1
45	MMC2_CMD/SPI3_SIMO	MMC/SD command signalSlave data in, master data out
46	MMC2_CLK/SPI3_CLK	MMC/SD Output ClockSPI Clock
47	BSP3_DR/UART2_RTS	Received serial dataUART2 Request To Send
48	BSP3_CLK/UART2_TX	Combined serial clockUART2 Transmit data
49	BSP3_FSX/UART2_RX	Combined frame synchronizationUART2 Receive data

<b>Pins</b>	<b>Definitions</b>	<b>Descriptions</b>
50	BSP3_DX/UART2_CTS	Transmitted serial dataUART2 Clear To Send
51	GND4	GND
52	UART1_CTS	UART1 Clear To Send
53	UART1_TX	UART1 Transmit data
54	UART1_RX	UART1 Receive data
55	UART1_RTS	UART1 Request To Send
56	USB1HS_STP	Dedicated for external transceiver Stop signal
57	USB1HS_D3	Dedicated for external transceiver Bidirectional data bus
58	USB1HS_D5	Dedicated for external transceiver Bidirectional data bus
59	USB1HS_6	Dedicated for external transceiver Bidirectional data bus
60	USB1HS_D7	Dedicated for external transceiver Bidirectional data bus
61	USB1HS_D1	Dedicated for external transceiver Bidirectional data bus
62	USB1HS_D2	Dedicated for external transceiver Bidirectional data bus
63	USB1HS_D4	Dedicated for external transceiver Bidirectional data bus
64	USB1HS_D0	Dedicated for external transceiver Bidirectional data bus
65	USB1HS_NXT	Dedicated for external transceiver Next signal from PHY
66	USB1HS_CLK	Dedicated for external transceiver 60-MHz clock
67	GND6	GND
68	USB1HS_DIR	Dedicated for external transceiver data form PHY
69	SYS_CLKOUT1	Configurable output clock1
70	LEDA	LED leg A
71	LEDB	LED leg B
72	ADCIN0	ADC input0 (Battery type)
73	NRESPWRON	Power On Reset
74	NRESWARM	Warm Boot Reset (open drain output)
75	SYSEN	System enable output
76	GND6	GND
77	REGEN	Enable signal for external LDO

Pins	Definitions	Descriptions
78	ADCIN1	ADC input1 (General-purpose ADC input)
79	KC0	Keypad column 0
80	KC1	Keypad column 0
81	KC2	Keypad column 0
82	KC3	Keypad column 0
83	AUDIO_IN	Analog microphone bias 1
84	AUDIO_OR	Predriver output right P for external class-D amplifier
85	AUXR	Auxiliary audio input right
86	AUDIO_OL	Predriver output left P for external class-D amplifier
87	GND7	GND
88	VBAT1	Power supply (3V - 4.2V 1.5A)
89	ON/OFF	Input; detect a control command to start or stop the system
90	VBAT2	Power supply (3V - 4.2V 1.5A)

### 2.3.2 CN2 90pin DIP Interface (left row)

**Table 2-2 CN2 90pin DIP Interface (left row)**

Pins	Definitions	Descriptions
1	G_NWE	GPMC Write Enable
2	G_NOE	GPMC Read Enable
3	G_NCS7/GPT8/G_DIR	GPMC Chip Select bit 7PWM / event for GP timer 8GPMC / IO direction control for use with external transceivers
4	G_NCS4/DMAREQ1	GPMC Chip Select bit 7PWM /DMA request 1
5	G_NCS6/DMAREQ3	GPMC Chip Select bit 7PWM / DMA request 3
6	G_NCS3DMAREQ0	GPMC Chip Select bit 7External DMA request 0
7	GND1	GND
8	G_WAIT0	External indication of wait
9	G_NBE0 / G_CLE	Lower Byte Enable. Also used for Command Latch Enable
10	G_ALE	Address Latch Enable
11	G_NBE1	Upper Byte Enable

<b>Pins</b>	<b>Definitions</b>	<b>Descriptions</b>
12	HDQ_SIO	Bidirectional HDQ 1-Wire control and data
13	MMC1_D0	MMC/SD Card Data bit 0
14	MMC1_D1	MMC/SD Card Data bit 1
15	MMC1_D2	MMC/SD Card Data bit 2
16	MMC1_D6/IO128	MMC/SD Card Data bit 6
17	MMC1_D5/IO127	MMC/SD Card Data bit 5
18	MMC1_D4/IO126	MMC/SD Card Data bit 4
19	MMC1_D7/IO129	MMC/SD Card Data bit 7
20	MMC1_D3	MMC/SD Card Data bit 3
21	GND2	GND
22	MMC1_CLK	MMC/SD Output Clock
23	MMC1_CMD	MMC/SD command signal
24	VMMC1	Power supply for SD/MMC1 (3.0 / 1.8V)
25	UART3_RX	UART3 Receive data
26	UART3_CTS	UART3 Clear To Send
27	UART3_TX	UART3 Transmit data
28	UART3_RTS	UART3 Request To Send
29	DSS_ACBIAS	AC bias control (STN) or pixel data enable (TFT) output
30	DSS_VSYNC	LCD Vertical Synchronization
31	GND3	GND
32	DSS_HSYNC	LCD Horizontal Synchronization
33	DSS_CLK	LCD Pixel Clock
34	DSS_D6	LCD Pixel Data bit 6
35	DSS_D8	LCD Pixel Data bit 8
36	DSS_D7	LCD Pixel Data bit 7
37	DSS_D9	LCD Pixel Data bit 9
38	DSS_D20	LCD Pixel Data bit 20
39	DSS_D17	LCD Pixel Data bit 17
40	DSS_D16	LCD Pixel Data bit 16
41	DSS_D18	LCD Pixel Data bit 18
42	DSS_D10	LCD Pixel Data bit 10
43	DSS_D5	LCD Pixel Data bit 5
44	DSS_D4	LCD Pixel Data bit 4
45	GND4	GND
46	DSS_D2	LCD Pixel Data bit 2
47	DSS_D3	LCD Pixel Data bit 3
48	DSS_D0	LCD Pixel Data bit 0
49	DSS_D15	LCD Pixel Data bit 15
50	DSS_D11	LCD Pixel Data bit 11

<b>Pins</b>	<b>Definitions</b>	<b>Descriptions</b>
51	DSS_D23	LCD Pixel Data bit 23
52	DSS_D22	LCD Pixel Data bit 22
53	DSS_D14	LCD Pixel Data bit 14
54	DSS_D19	LCD Pixel Data bit 19
55	DSS_D13	LCD Pixel Data bit 13
56	DSS_D21	LCD Pixel Data bit 21
57	DSS_D1	LCD Pixel Data bit 1
58	DSS_D12	LCD Pixel Data bit 12
59	GND5	GND
60	MCBSP1_FSR/IO157	Receive frame synchronization
61	MCBSP1_CLKR/IO156	Receive Clock
62	MCBSP1_FSX/IO161	Transmit frame synchronization
63	MCBSP1_CLKS/IO160	External clock input
64	MCBSP1_CLKX/IO162	Transmit clock
65	MCBSP1_DR/IO159	Received serial data
66	MCBSP1_DX/IO158	Transmitted serial data
67	GND6	GND
68	TV_OUTC	TV analog output S-VIDEO: TV_OUT2
69	TV_OUTY	TV analog output Composite: TV_OUT1
70	VDD33_1	Power supply for camera (3.3V 500mA )
71	IIC3_SCL	I2C Master Serial clock. Output is open drain
72	IIC3_SDA	I2C Serial Bidirectional Data. Output is open drain
73	IO25	General-purpose IO 183
74	IO27	General-purpose IO 183
75	BOOTJUMP	Boot configuration mode bit 5.
76	GND7	GND
77	VBUS	VBUS power rail (5V 10mA)
78	USB_DN	USB Data N
79	USB_ID	USB ID
80	USB_DP	USB Data P
81	PWM0	Pulse width driver 0
82	KR0	Keypad row 0
83	KR1	Keypad row 1
84	KR2	Keypad row 2
85	KR3	Keypad row 3
86	KR4	Keypad row 4
87	VDD18_1	Power supply from TPS65930 (VIO 1.8V)
88	GND8	GND

Pins	Definitions	Descriptions
89	VDD18_2	Power supply from TPS65930 (VIO 1.8V)
90	BKBAT	Backup battery

### 2.3.3 CN3 JTAG Interface

**Table 2-3 CN3 JTAGInterface**

Pins	Definitions	Descriptions
1	VDD18	1.8V output
2	TMS	Test mode select
3	TD1	Test data input
4	NTRST	Test system reset
5	TD0	Test data output
6	RTCK	Receive test clock
7	TCK	Test clock
8	EMU0	Test emulation 0
9	EMU1	Test Emulation 1
10	GND	GND

### 2.3.4 CN4 Camera Interface

**Table 2-4 CN4 Camera Interface**

Pins	Definitions	Descriptions
1	GND0	GND
2	D0	Digital image data bit 0
3	D1	Digital image data bit 1
4	D2	Digital image data bit 2
5	D3	Digital image data bit 3
6	D4	Digital image data bit 4
7	D5	Digital image data bit 5
8	D6	Digital image data bit 6
9	D7	Digital image data bit 7
10	D8	Digital image data bit 8
11	D9	Digital image data bit 9
12	D10	Digital image data bit 10
13	D11	Digital image data bit 11
14	GND1	GND

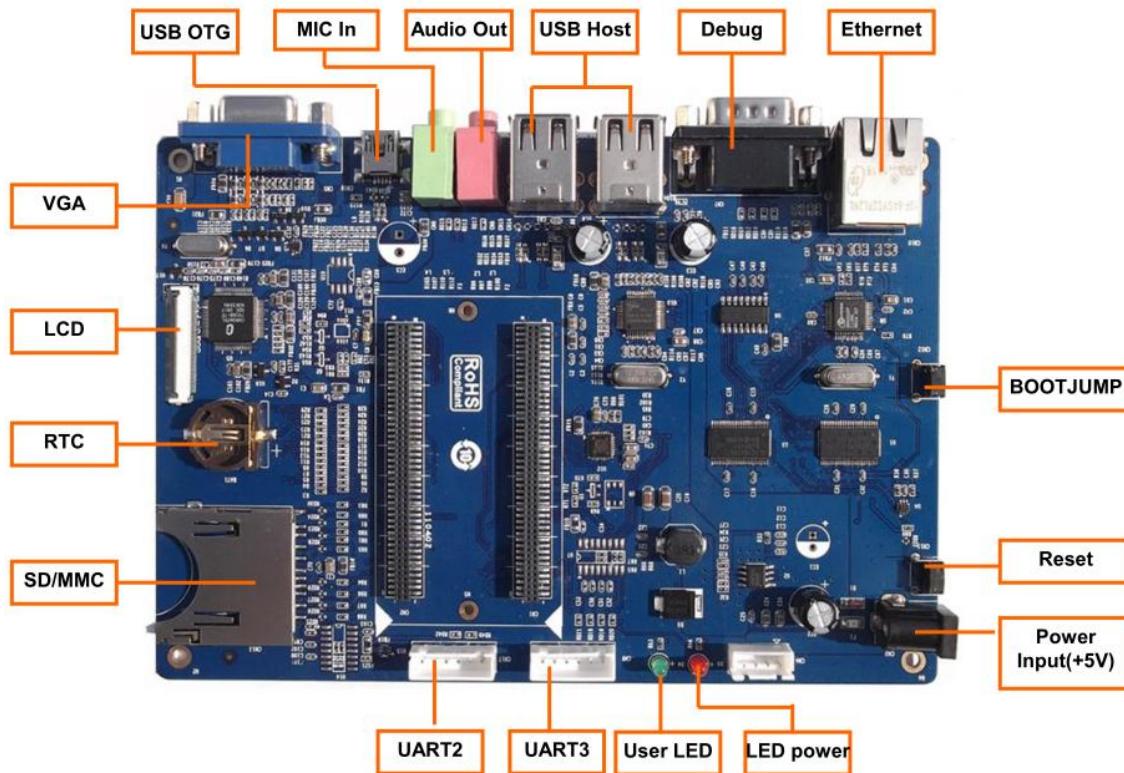
Pins	Definitions	Descriptions
15	PCLK	Pixel clock
16	GND2	GND
17	HS	Horizontal synchronization
18	VDD50	5V
19	VS	Vertical synchronization
20	VDD33	3.3V
21	XCLKA	Clock output a
22	XCLKB	Clock output b
23	GND3	GND
24	FLD	Field identification
25	WEN	Write Enable
26	STROBE	Flash strobe control signal
27	SDA	IIC master serial clock
28	SCL	IIC serial bidirectional data
29	GND4	GND
30	VDD18	1.8V

### 2.3.5 LED Indicators

**Table 2-5 LED Indicators**

LEDs	Definitions	Descriptions
D1	LED1	User custom LED
D2	LED2	User custom LED
D3	LED3	User custom LED
D4	LED4	User custom LED
D5	VDD18	Power indicator
D6	VBAT	Power indicator

## 2.4 Interfaces on Expansion Board



**Figure 2-3 Expansion Board**

### 2.4.1 Power Jack

**Table 2-6 Power Jack**

Pins	Definitions	Descriptions
1	GND	GND
2	+5V	Power supply (+5V) 2A (Type)

### 2.4.2 TFT\_LCD Interface

**Table 2-7 TFT\_LCD Interface**

Pins	Definitions	Descriptions
1	DSS_D0	LCD Pixel data bit 0
2	DSS_D1	LCD Pixel data bit 1
3	DSS_D2	LCD Pixel data bit 2
4	DSS_D3	LCD Pixel data bit 3

<b>Pins</b>	<b>Definitions</b>	<b>Descriptions</b>
5	DSS_D4	LCD Pixel data bit 4
6	DSS_D5	LCD Pixel data bit 5
7	DSS_D6	LCD Pixel data bit 6
8	DSS_D7	LCD Pixel data bit 7
9	GND	GND
10	DSS_D8	LCD Pixel data bit 8
11	DSS_D9	LCD Pixel data bit 9
12	DSS_D10	LCD Pixel data bit 10
13	DSS_D11	LCD Pixel data bit 11
14	DSS_D12	LCD Pixel data bit 12
15	DSS_D13	LCD Pixel data bit 13
16	DSS_D14	LCD Pixel data bit 14
17	DSS_D15	LCD Pixel data bit 15
18	GND	GND
19	DSS_D16	LCD Pixel data bit 16
20	DSS_D17	LCD Pixel data bit 17
21	DSS_D18	LCD Pixel data bit 18
22	DSS_D19	LCD Pixel data bit 19
23	DSS_D20	LCD Pixel data bit 20
24	DSS_D21	LCD Pixel data bit 21
25	DSS_D22	LCD Pixel data bit 22
26	DSS_D23	LCD Pixel data bit 23
27	GND	GND
28	DEN	AC bias control (STN) or pixel data enable (TFT)
29	HSYNC	LCD Horizontal Synchronization
30	VSYNC	LCD Vertical Synchronization
31	GND	GND
32	CLK	LCD Pixel Clock
33	GND	GND
34	X+	X+ Position Input
35	X-	X- Position Input
36	Y+	Y+ Position Input
37	Y-	Y- Position Input
38	SPI_CLK	SPI clock
39	SPI_MOSI	Slave data in, master data out
40	SPI_MISO	Slave data out, master data in
41	SPI_CS	SPI enable
42	IIC_CLK	IIC master serial clock
43	IIC_SDA	IIC serial bidirectional data
44	GND	GND

Pins	Definitions	Descriptions
45	VDD18	1.8V
46	VDD33	3.3V
47	VDD50	5V
48	VDD50	5V
49	RESET	Reset
50	PWREN	Power on enable

### 2.4.3 Audio Output Interface

**Table 2-8 Audio Output Interface**

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

### 2.4.4 Audio Input Interface

**Table 2-9 Audio Input Interface**

Pins	Definitions	Descriptions
1	GND	GND
2	NC	NC
3	MIC MAIN P	Right input
4	NC	NC
5	MIC MAIN N	Left input

### 2.4.5 Serial Interface

**Table 2-10 Serial Interface**

Pins	Definitions	Descriptions
1	NC	NC
2	RXD	Receive data
3	TXD	Transit data
4	NC	NC
5	GND	GND
6	NC	NC
7	RTS	Request To Send

Pins	Definitions	Descriptions
8	CTS	Clear To Send
9	NC	NC

#### 2.4.6 Ethernet Interface

**Table 2-11 Ethernet Interface**

Pins	Definitions	Descriptions
1	TX+	TX+ output
2	TX-	TX- output
3	RX+	RX+ input
4	VDD25	2.5V Power for TX/RX
5	VDD25	2.5V Power for TX/RX
6	RX-	RX- input
7	NC	NC
8	NC	NC
9	VDD	3.3V Power for LED
10	LED1	Speed LED
11	LED2	Link LED
12	VDD	3.3V Power for LED

#### 2.4.7 USB OTG Interface

**Table 2-12 USB OTG Interface**

Pins	Definitions	Descriptions
1	VBUS	+5V
2	DN	USB Data-
3	DP	USB Data+
4	ID	USB ID
5	GND	GND

#### 2.4.8 USB HOST Interface

**Table 2-13 USB HOST Interface**

Pins	Definitions	Descriptions
1	VBUS	+5V
2	DN	USB Data-
3	DP	USB Data+
4	ID	USB ID

### 2.4.9 SD Card Interface

Table 2-14 SD Interface

Pins	Definitions	Descriptions
1	CD/DAT3	Card detect/Card data 2
2	DCMD	Command Signal
3	VSS	GND
4	VDD	VDD
5	CLK	Clock
6	VSS	GND
7	TF_DAT0	Card data 0
8	TF_DAT1	Card data 1
9	TF_DAT2	Card data2
10	SW_2	SD write protect
11	SW_1	Card detect
12	GND	GND

### 2.4.10 LED Indicators

Table 2-15 LED Indicators

LEDs	Definitions	Descriptions
D4	LED_POWER	3.3V power indicator
D5	User LED	User custom LED

### 2.4.11 Buttons

Table 2-16 Buttons

Buttons	Definitions	Descriptions
CN12	BOOTJUMP	Boot system from TF card
CN11	Reset	Reset system

# Chapter 3 Linux Operating System

SBC8140 has a complete Linux system installed in its on-board NAND Flash by factory default. This chapter contains several sections to introduce Linux system of SBC8140 in detail, including structure of embedded Linux system, software features, system development process, driver introduction and development, and system update.

## Note:

- ❑ Same 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.
- ❑ Ubuntu Linux is used in this document. If you do not have any Linux system on your PC, please refer to Appendix 1 – Installing Ubuntu Linux.

## 3.1 Structure of Embedded Linux System

The following figure shows the structure of embedded Linux system:

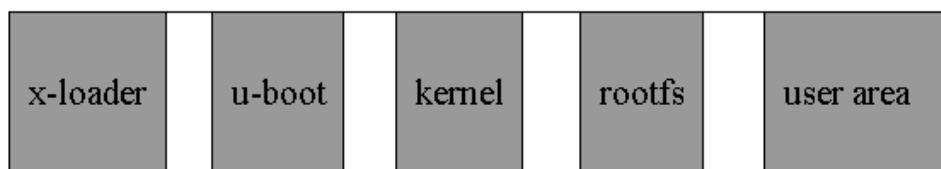


Figure 3-1 Embedded Linux system

- **x-loader:** First-level booting program; after the kit is powered on, the program is copied from the ROM in CPU to RAM and executed so as to initialize CPU and copy u-boot to RAM, and then u-boot takes control over the system,
- **u-boot:** Second-level booting program; it is used to interact with users and provide function such as updating image files and booting the core.
- **Kernel:** Core 2.6.32 version; customized for SBC8140.

- **Roofs:** Open-source ubifs filesystem; it is suited for embedded systems.

## 3.2 Software Features

**Table 3-1 Software Features**

<b>Software</b>		<b>Descriptions</b>	<b>Code Type</b>
<b>BIOS</b>	x-loader	NAND / ONENAND	Source code
		MMC/SD	Source code
		FAT	Source code
	u-boot	NAND / ONENAND	Source code
		MMC/SD	Source code
		FAT	Source code
		NET	Source code
<b>Kernel</b>	Linux-2.6.x	Supports ROM/CRAM/EXT2/EXT3/FAT/NFS/ JFFS2/UBIFS filesystems	Source code
<b>Device Driver</b>	serial	Serial interface driver	Source code
	rtc	Hardware clock driver	Source code
	net	10/100M Ethernet DM9000 driver	Source code
	flash	Nand flash driver (supports nand boot)	Source code
	lcd	TFT LCD driver	Source code
	touch screen	Touch-screen controller ads7846 driver	Source code
	mmc/sd	mmc/sd controller driver	Source code
	usb otg	USB OTG 2.0 driver (currently only support usb deivce mode)	Source code
	usb ehci	usb ehci driver	Source code
	VGA	Support VGA signal output	Source code
	audio	Sound card driver (support audio recording/playback)	Source code
	camera	Camera driver	Source code
	Key	Key driver	
	led	LEDs driver	Source code
<b>Demo</b>	Android	Android system v2.2	Source code
	DVSDK	DVSDK 4_00_00_22 system	Source code

## 3.3 System Development Process

This section will show you the whole process of developing software starting from building a development environment to making a customized system.

### 3.3.1 Building Development Environment

#### 1) **Installing Cross Compiling Tools;**

Put the DVD-ROM in your PC's DVD drive, Ubuntu will mount it under /media/cdrom/ automatically, and then execute the following instructions in the terminal window of Ubuntu to uncompress the cross compiling tools from /media/cdrom/linux/tools to \$HOME;

- `cd /media/cdrom/linux/tools`
- `tar xvf arm-eabi-4.4.0.tar.bz2 -C $HOME`
- `tar xvf arm-2007q3.tar.bz2 -C $HOME`

#### 2) **Copying More Tools;**

Continue executing the following instructions to copy tools required during source code compilation from /linux/tools to \$HOME/tools/;

- `mkdir $HOME/tools`
- `cp /media/cdrom/linux/tools/mkimage $HOME/tools`
- `cp /media/cdrom/linux/tools/signGP $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) **Adding Environment Variables;**

Execute the following instructions to add installed tools into environment variables;

- `export PATH=$HOME/arm-eabi-4.4.0/bin:$HOME /arm-2007q3/bin:$HOME/tools:`
- `$PATH`

**Note:**

- The instruction used to add environment variables can be put into the file .bashrc under user directory to allow the system load the variable automatically each time when it boots up.
- If you need to view the path, please use the instruction `echo $PATH`.

### 3.3.2 System Compilation

#### 1) Uncompress Source Code;

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

- `mkdir $HOME/work`
- `cd $HOME/work`
- `tar xvf /media/cdrom/linux/source/x-loader-03.00.02.07.tar.bz2`
- `tar xvf /media/cdrom/linux/source/u-boot-03.00.02.07.tar.bz2`
- `tar xvf /media/cdrom/linux/source/linux-2.6.32-sbc8140.tar.bz2`
- `sudo tar xvf /media/cdrom/linux/source/rootfs.tar.bz2`
- `tar xvf /media/cdrom/linux/demo/Android/source/rowboat-android-froyo-sbc8140.tar.bz2`

After all the instructions are executed, directories x-loader-03.00.02.07, u-boot-03.00.02.07, linux-2.6.32-sbc8140, rootfs and rowboat-android-froyo-sbc8140 are created under current directory.

#### 2) Compiling First-Level Booting Code;

- A. Execute the following instructions to compile first-level booting code for SD card boot-up mode;
  - `cd x-loader-03.00.02.07`
  - `make distclean`

- **make omap3sbc8140\_config**
- **make**
- **signGP x-load.bin**
- **mv x-load.bin.ift MLO**

After all the instructions are executed, a MLO file is generated under current directory.

**B.** Execute the following instructions to compile first-leve code for NAND

Flash boo-up mode;

- **cd x-loader-03.00.02.07**
- **vi include/configs/omap3sbc8140.h** → Note with the line // #define CONFIG MMC 1 in the file omap3sbc8140.h
- **cd x-loader-03.00.02.07**
- **make distclean**
- **make omap3sbc8140\_config**
- **make**
- **signGP x-load.bin**
- **mv x-load.bin.ift x-load.bin.ift\_for\_NAND**

After all the instructions are executed, a file named x-load.bin.ift\_for\_NAND is generated under current directory.

**3) Compiling Second-Level Code;**

Execute the following instructions to compile second-leve booting code;

- **cd u-boot-03.00.02.07**
- **make distclean**
- **make omap3\_sbc8140\_config**
- **make**

After all the instructions are executed, a file named u-boot.bin is generated under current directory.

**4) Compiling Kernel;**

**A.** The operations for Linux system are as follows:

- **cd linux-2.6.32-sbc8140**

- **make distclean**
- **make omap3\_sbc8140\_defconfig**
- **make ulimage**

B. The operations for Android system are as follows:

- **cd linux-2.6.32-sbc8140**
- **make distclean**
- **make omap3\_sbc8140\_android\_defconfig**
- **make ulimage**

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

## 5) Making Filesystem;

A. Making Ramdisk filesystem;

Please visit [http://www.elinux.org/DevKit8600\\_FAQ](http://www.elinux.org/DevKit8600_FAQ).

B. Execute the following instructions to generate UBI file;

- **cd \$HOME/work**
- **sudo \$HOME/tools/mkfs.ubifs -r rootfs -m 2048 -e 129024 -c 1996 -o ubifs.img**
- **sudo \$HOME/tools/ubinize -o ubi.img -m 2048 -p 128KiB -s 512**
- **\$HOME/tools/ubinize.cfg**

After all the instructions are executed, a file ubi.img is generated under current directory.

## 6) Android system compilation

A. Execute the following instructions to start compilation of Android system;

- **cd rowboat-android-froyo-SBC8140**
- **make**

B. 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 Androind kernel source code linux-2.6.32-sbc8140 needs to compile first, or errors might occur during the process.

### 3.3.3 Customizing System

There are many configurations available for users to add or remove drivers and features in Linux core so as to meet the requirement of customers. The following example shows the process of making a custom system.

#### 1) Entering Configuration Menu;

By default, the configuration file is saved under /linux-2.6.32-sbc8140/arch/arm/configs/omap3\_SBC8140\_defconfig/; please execute the following instructions to enter system configuration menu;

- **cd linux-2.6.32-sbc8140**
- **cp arch/arm/configs/omap3\_sbc8140\_defconfig .config**
- **make menuconfig**

**Notice:**

- ❑ If errors occur when executing **make menuconfig**, the possible cuase is that ncurses library is missing in Ubuntu system. Please execute **sudo apt-get install ncurses-dev** to install the library.

#### 2) Customizing Configurations;

Change configurations according to actual requirements, for example select Device Drivers > USB support > USB Gadget Support > USB Gadget Drivers and check **File-backed Storage Gadget** as shown below, and then exit and save changes.

```

--- USB Gadget Support
[ ] Debugging messages (DEVELOPMENT)
[ ] Debugging information files (DEVELOPMENT)
[ ] Debugging information files in debugfs (DEVELOPMENT)
(2) Maximum VBUS Power usage (2-500 mA)
USB Peripheral Controller (Invenra HDRC USB Peripheral (TI, ADI, ...)) --->
<H> USB Gadget Drivers
< > Gadget Zero (DEVELOPMENT)
< > Audio Gadget (EXPERIMENTAL)
<M> Ethernet Gadget (with CDC Ethernet support)
[*] RNDIS support
[ ] Ethernet Emulation Model (EEM) support
< > Gadget Filesystem (EXPERIMENTAL)
<M> File-backed Storage Gadget
[< > 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)

```

**Figure 3-2 USB Gadget Drivers submenu**

Set the option **File-backed Storage Gadget** to M, and then exit and save changes.

**3) Execute the following instructions to compile core;**

- **make ulimage**
- **make modules**

After the instructions are executed, a core image file named ulimage and a module file g\_file\_storage.ko are generated under /arch/arm/boot/ and /drivers/usb/gadget/ respectively.

## 3.4 Introduction to Drivers

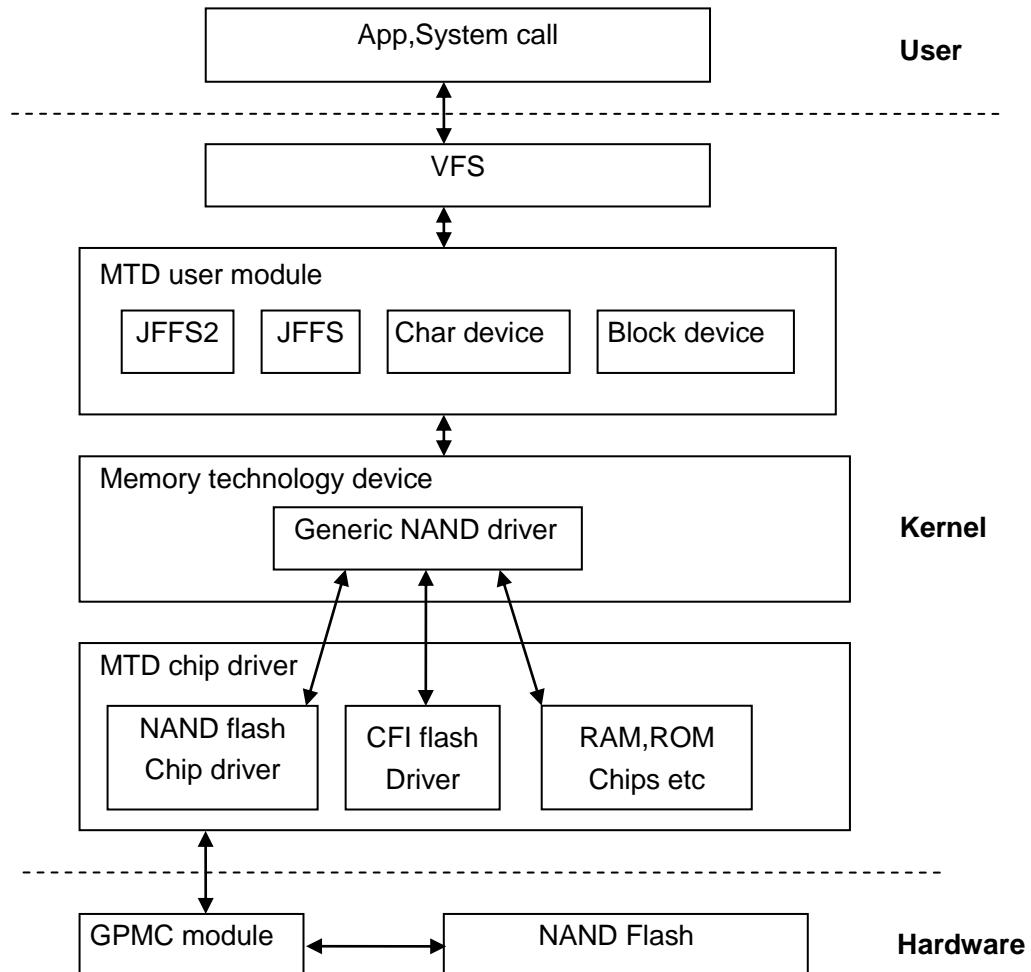
This section will introduce various drivers required in Linux system, including NAND Flash, SD/MMC, display subsystem, video capture and audio input/output drivers.

The following table contains the paths for all the drivers;

Software		Descriptions	Paths
BIOS	x-loader	ONENAND	x-loader-03.00.02.07/drivers/onenand.c
		NAND	x-loader-03.00.02.07/drivers/k9f1g08r0a.c
		MMC/SD	x-loader-03.00.02.07/cpu/omap3/mmc.c
		FAT	x-loader-03.00.02.07/fs/fat/
	u-boot	NAND	u-boot-03.00.02.07/drivers/mtd/nand/
		ONENAND	u-boot-03.00.02.07/drivers/mtd/onenand/
		MMC/SD	u-boot-03.00.02.07/drivers/mmc
		FAT	u-boot-03.00.02.07/fs/fat/

Software		Descriptions	Paths
		NET	u-boot-03.00.02.07/drivers/net/dm9000x.c
<b>Kernel</b>	Linux-2.6.x	Supports ROM/CRAM/EXT2/EXT 3/FAT/NFS/ JFFS2/UBIFS filesystems	linux-2.6.32-sbc8140/fs/
<b>Device Driver</b>	serial	Serial interface driver	linux-2.6.32-sbc8140/drivers/serial/8250.c
	rtc	Hardware clock driver	linux-2.6.32-sbc8140/drivers/rtc/rtc-tw1.c
	net	10/100M Ethernet DM9000 driver	linux-2.6.32-sbc8140/drivers/net/dm9000.c
	flash	Nand flash driver (supports nand boot)	linux-2.6.32-sbc8140/drivers/mtd/nand/omap2.c
	lcd	TFT LCD driver	linux-2.6.32-sbc8140/drivers/video/omap2/omapfb/omapfb-main.c
			linux-2.6.32-sbc8140/drivers/video/omap2/displays/panel-omap3-sbc8140.c
	touch screen	Touch-screen controller ads7846 driver	linux-2.6.32-sbc8140/drivers/input/touchscreen/ads7846.c
	mmc/sd	mmc/sd controller driver	linux-2.6.32-sbc8140/drivers/mmc/host/omap_hs mmc.c
	usb otg	USB OTG 2.0 driver (currently only support usb device mode)	linux-2.6.32-sbc8140/drivers/usb/otg/twl4030-us b.c
	usb ehci	usb ehci driver	linux-2.6.32-sbc8140/drivers/usb/host/ehci-hcd.c
	VGA	Support VGA signal output	linux-2.6.32-sbc8140/drivers/i2c/chips/ch7033.c
	audio	Sound card driver (support audio recording/playback)	linux-2.6.32-sbc8140/sound/soc/omap/omap3sb c8140.c
			linux-2.6.32-sbc8140/sound/soc/codecs/twl4030. c
	camera	Camera driver	Digital: linux-2.6.32-sbc8140/drivers/media/video/omap3 4xxcam.c
			Catalog: linux-2.6.32-sbc8140/drivers/media/video/tvp514 x-int.c
	Keypad	keypad driver	linux-2.6.32-sbc8140/drivers/input/keyboard/gpio _keys.c
	LED	LED driver	linux-2.6.32-sbc8140/drivers/leds/leds-gpio.c

### 3.4.1 NAND Flash Driver



**Figure 3-3 Working principle of NAND Flash**

NAND flash is used as a block device and has a filesystem built in it. The interaction between users and NAND Flash is realized by a specific filesystem. In order to eliminate the inconsistency between different flash memories, a MTD subsystem is placed between core's filesystem and flash driver, and therefore users need to go through the following path to access NAND Flash;

**User > System Call > VFS > Block Device Driver > MTD > NAND Flash Driver > NAND Flash**

**Table 3-2 Driver Reference Documents**

Reference	linux-2.6.32-sbc8140/drivers/mtd/nand/
-----------	--

linux-2.6.32-sbc8140/drivers/mtd/nand/omap2.c

### 3.4.2 SD/MMC Driver

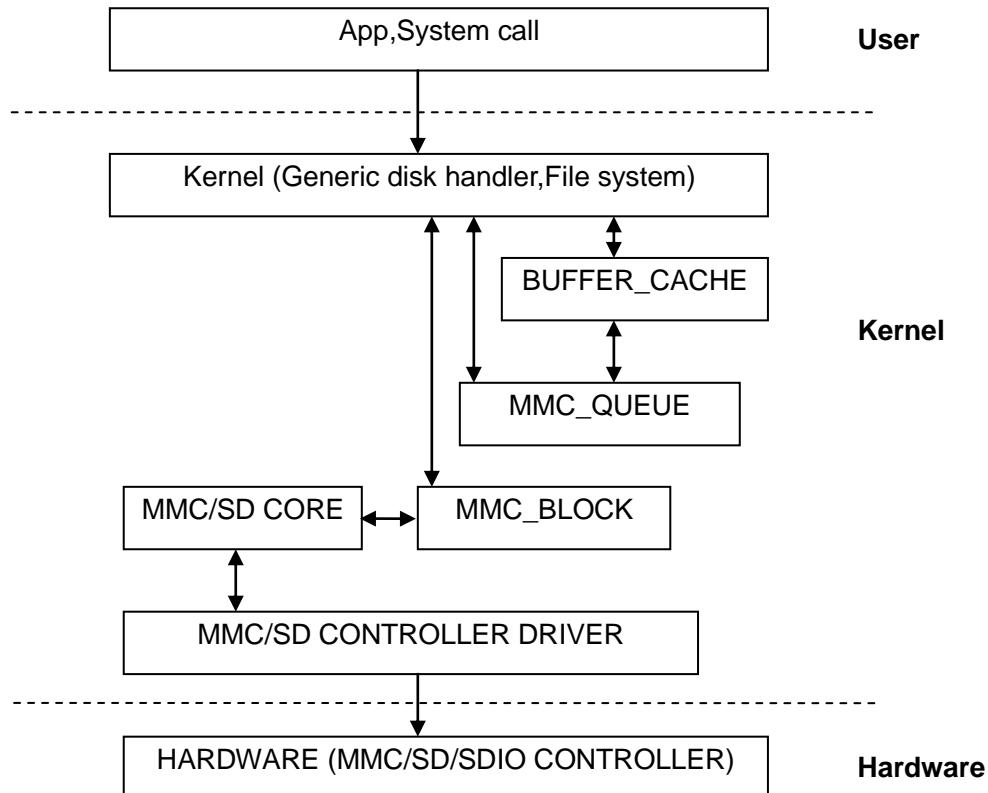


Figure 3-4 Working principle of SD/MMC

The SD/MMC card drivers under Linux system typically include four parts - SD/MMC core, mmc\_block, mmc\_queue and SD/MMC driver;

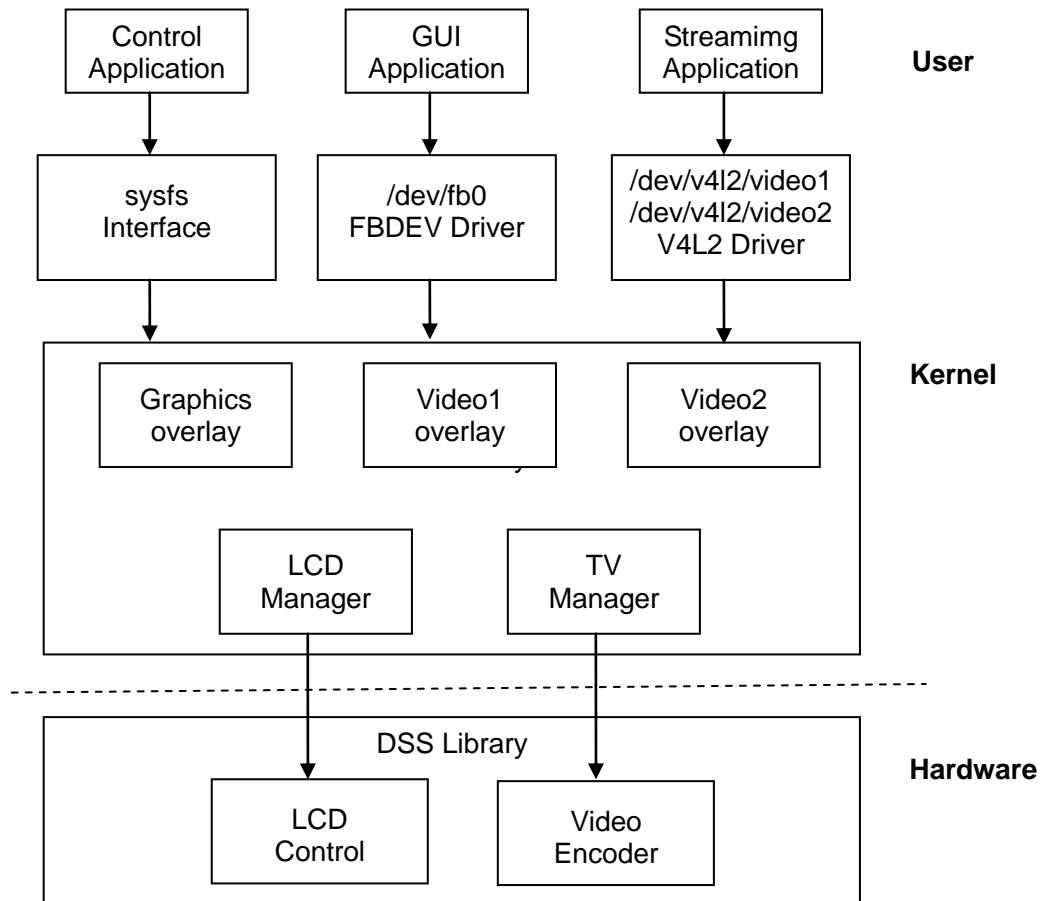
- SD/MMC core implements the structure independent core code in SD/MMC related operations;
- mmc\_block implements driver structure used when SD/MMC cards work as block devices;
- mmc\_queue implement management of request queue;
- SD/MMC driver implements controller drivers;

Table 3-3 Driver Reference Documents

Reference | linux-2.6.32-sbc8140/drivers/mmc/

	linux-2.6.32-sbc8140/drivers/mmc/host/omap_hsmmc.c
--	--

### 3.4.3 Display Subsystem Driver



**Figure 3-5** Working principle of display subsystem

The hardware of display subsystem includes a graphics channel, two video channels and two overlay management units; one of the units is responsible for digital interface, another for analog interface. Digital interface manages the LCD output, while analog one manages TV output.

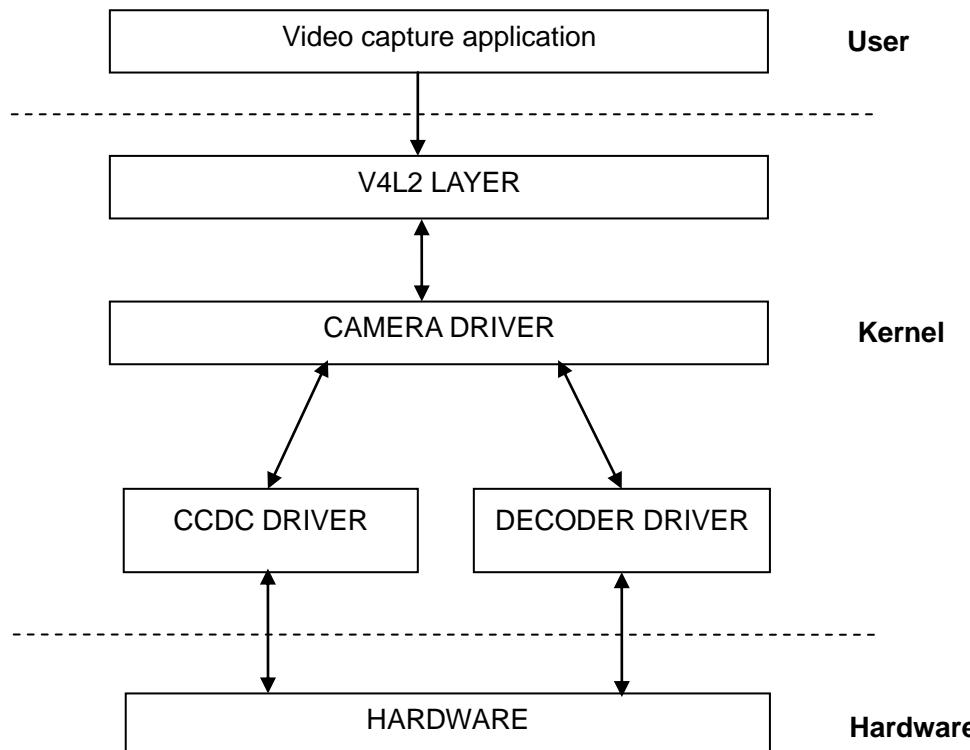
The main function of display driver is to provide interfaces for upper application layer and manage the hardware components of display subsystem.

**Table 3-4    Driver Reference Documents**

<b>Reference</b>	linux-2.6.32-sbc8140/drivers/video/omap2/ linux-2.6.32-sbc8140/drivers/video/omap2/omapfb/omapfb-main.c
------------------	--

```
linux-2.6.32-sbc8140/drivers/video/omap2/displays/panel-omap3-sbc8140.c
```

### 3.4.4 Video Capture Driver



**Figure 3-6** Working principle of video capture

- V4L2 Subsystem

V4L2 subsystem of Linux system works as the medium layer which helps access camera driver. The upper-layer applications of camera can access drivers through API of V4L2. The V4L2 subsystem of Linux 2.6 core is designed based on V4L2 standard.

- Video Buffer Library

Video Buffer Library is a part of V4L2. It provides assistance function to effectively manage video buffer by queuing method.

- Camera Driver

Camera driver allow externa codec capture video images. It is registers to layer V4L2 as a master device. Any codec driver that is added to layer V4L2 as slave devices will be associated with camera controller driver through new V4L2 master-slave interface. Currently the driver can support only one codec deivce associated with camera controller.

- **Codec Driver**

Codec driver needs to comply with V4L2 master-slave interface standard and should be registered to V4L2 as a slave device. Replacing codec can be accomplished by rewriting codec driver, without any change to carmra driver.

- **CCDC Library**

As a hardware module for data input, CCDC receives data from sensor/decoder. CCDC library provides API for configuring CCDC module and being called by camera driver.

**Table 3-5    Driver Reference Documents**

<b>Reference</b>	linux-2.6.32-sbc8140/drivers/media/video/
	linux-2.6.32-sbc8140/drivers/media/video/omap34xxcam.c
	linux-2.6.32-sbc8140/drivers/media/video/tvp514x-int.c

### 3.4.5 Audio Input/Output Driver

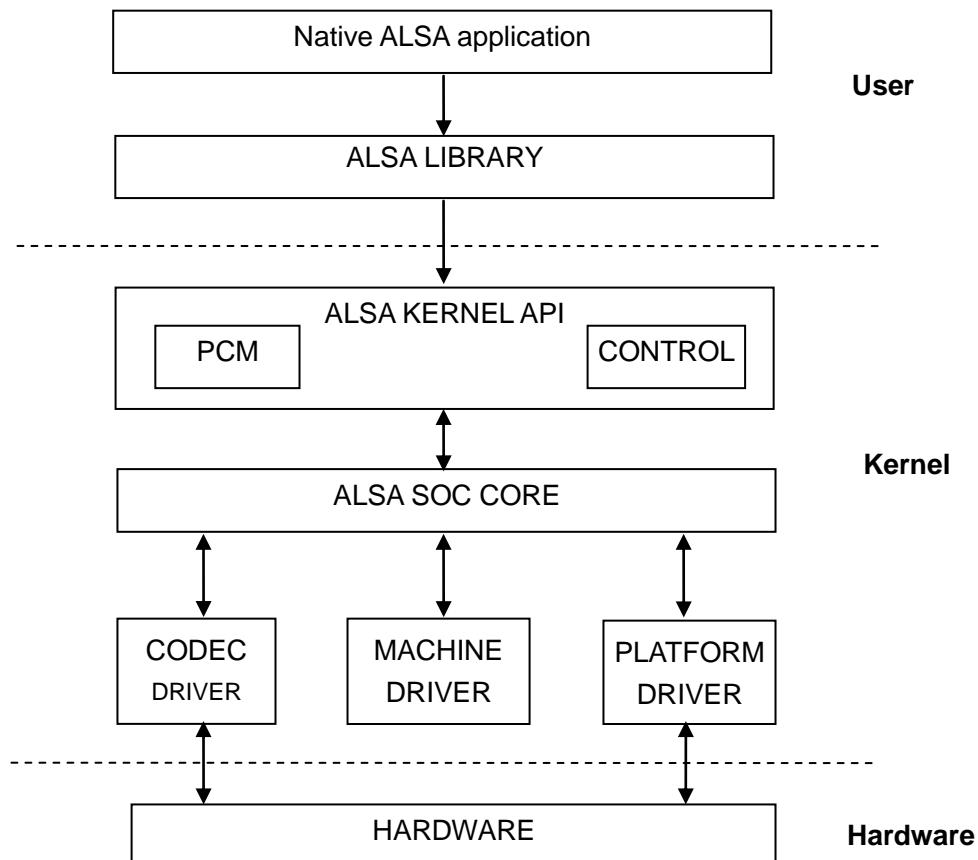


Figure 3-7 Working principle of audio input/output

ASoC embedded audio system is comprised of the following parts;

- Codec Driver;  
Codec driver is platform independent and contains audio controls, audio interface capabilities, codec DAPM definition and codec IO functions;
- Platform Driver;  
Platform driver contains the audio DMA engine and audio interface drivers (e.g. I2S, AC97, PCM) for that platform;
- Machine Driver;  
Machine driver handles any machine specific controls and audio events. i.e. turning on an amp at start of playback;

**Table 3-6 Driver Reference Documents**

<b>Reference</b>	linux-2.6.32-sbc8140/sound/soc/
	linux-2.6.32-sbc8140/sound/soc/omap/omap3sbc8140.c
	linux-2.6.32-sbc8140/sound/soc/codecs/twl4030.c

## 3.5 Driver Development

This section will introduce how to develop drivers with two examples, GPIO\_Keys and GPIO\_LEDs drivers development.

### 3.5.1 GPIO\_Keys Driver

#### 1) Device Definition;

Source file board-omap3sbc8140.c is saved under /linux-2.6.32-sbc8140/arch/arm/mach-omap2/;

**Table 3-7 Device Definition Source Code**

```
static struct gpio_keys_button gpio_buttons[] = {
{
    .code          = KEY_F1,
    .gpio          = 26,
    .desc          = "menu",
    .active_low    = true,
},
{
    .code          = KEY_ESC,
    .gpio          = 29,
    .desc          = "back",
    .active_low    = true,
},
};

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

static struct platform_device keys_gpio = {
```

```

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

Set gpio 26 as **menu** key, returning key value **KEY\_F1**, triggered by low voltage level.

## **2) GPIO pinmux Configuration;**

File sbc8140.h is saved under /u-boot-03.00.02.07/board/timll/sbc8140/;

**Table 3-8 GPIO pinmux Configuration**

```

/*
 * IEN - Input Enable
 * IDIS - Input Disable
 * PTD - Pull type Down
 * PTU - Pull type Up
 * DIS - Pull type selection is inactive
 * EN - Pull type selection is active
 * M0 - Mode 0
 * The commented string gives the final mux configuration for that pin
*/
MUX_VAL(CP(ETK_D12_ES2),          (IEN | PTU | DIS | M4)) /*GPIO_26*/
MUX_VAL(CP(ETK_D15_ES2),          (IEN | PTU | DIS | M4)) /*GPIO_29*/
```

Set GPIO 26 and GPIO 29 as M4 (gpio mode) and IEN (allow input).

## **3) Driver Design;**

Source file gpio\_keys.c is saved under /linux-2.6.32-sbc8140/drivers/input/keyboard/;

### **A. Call platform\_driver\_register to register gpio\_keys driver;**

**Table 3-9 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,
```

```

#ifndef CONFIG_PM
    .pm      = &gpio_keys_pm_ops,
#endif
}

};

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);
}

module_init(gpio_keys_init);
module_exit(gpio_keys_exit);

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

```

**B. Call input\_register\_device to register input driver;**

**Table 3-10 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(dev, bdata, button);
        if (error)
            goto fail2;
    }
}

```

```
    if (button->wakeup)
        wakeup = 1;

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

error = input_register_device(input);
...
```

C. Apply for gpio, set gpio as input, and register gpio interrupt;

**Table 3-11 Register GPIO Interrupt**

```
static int __devinit gpio_keys_setup_key(struct device *dev,
                                         struct gpio_button_data *bdata,
                                         struct gpio_keys_button *button)
{
    char *desc = button->desc ? button->desc : "gpio_keys";
    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;
    }

    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;
    }

    error = request_irq(irq, gpio_keys_isr,
                         IRQF_SHARED |
                         IRQF_TRIGGER_RISING | IRQF_TRIGGER_FALLING,
                         desc, bdata);
    if (error) {
        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; an interrupt is generated when pressing a button, and the a key value will be returned;

**Table 3-12 Interrupt Processing**

```
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);
```

### 3.5.2 GPIO\_LEDs Driver

#### 1) Device Definitions;

Souce file board omap3sbc8140.c is saved under /linux-2.6.32-sbc8140/arch/arm/mach-omap2/;

Table 3-13 Deivce Definition Source Code

```
static struct gpio_led gpio_leds[] = {
{
    .name          = "led0",
    .default_trigger = "heartbeat",
    .gpio          = 136,
    .active_low     = true,
},
{
    .name          = "led1",
    .gpio          = 137,      /* gets replaced */
    .active_low     = true,
},
{
    .name          = "led2",
    .gpio          = 138,      /* gets replaced */
    .active_low     = true,
},
{
    .name          = "led3",
    .gpio          = 139,      /* gets replaced */
    .active_low     = true,
},
};
```

Associate GPIO 136 with led0 (system breath LED), GPIO137 with led1, GPIO138 with led2, GPIO139 with led3; they are all valid upon low voltage level.

## **2) GPIO pinmux Configurations;**

File sbc8140.h is saved under /u-boot-03.00.02.07/board/timll/sbc8140/;

**Table 3-14 GPIO pinmux Configurations**

```

/*
 * IEN - Input Enable
 * IDIS - Input Disable
 * PTD - Pull type Down
 * PTU - Pull type Up
 * DIS - Pull type selection is inactive
 * EN - Pull type selection is active
 * M0 - Mode 0
 * The commented string gives the final mux configuration for that pin
 */

MUX_VAL(CP(MMC2_DAT4),           (IDIS | PTD | DIS | M4)) /*GPIO_136*/\n
MUX_VAL(CP(MMC2_DAT5),           (IDIS | PTD | DIS | M4)) /*GPIO_137*/\n
MUX_VAL(CP(MMC2_DAT6),           (IDIS | PTD | DIS | M4)) /*GPIO_138*/\n
MUX_VAL(CP(MMC2_DAT7),           (IDIS | PTU | DIS | M4)) /*GPIO_139*/

```

Set GPIO136, GPIO137, GPIO138 and GPIO139 as M4 (gpio mode) and IDIS (input not allowed)

## **3) Driver Design;**

Source file leds-gpio.c is saved under /linux-2.6.32-sbc8140/drivers/leds/;

### **A. Call platform\_driver\_register to register gpio\_leds driver;**

**Table 3-15 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,
    },
};

```

```
static int __init gpio_led_init(void)
{
    int ret;

#ifdef CONFIG_LEDS_GPIO_PLATFORM
    ret = platform_driver_register(&gpio_led_driver);
    if (ret)
        return ret;
#endif
#ifdef CONFIG_LEDS_GPIO_OF
    ret = of_register_platform_driver(&of_gpio_leds_driver);
#endif
#ifdef CONFIG_LEDS_GPIO_PLATFORM
    if (ret)
        platform_driver_unregister(&gpio_led_driver);
#endif

    return ret;
}

static void __exit gpio_led_exit(void)
{
#ifdef CONFIG_LEDS_GPIO_PLATFORM
    platform_driver_unregister(&gpio_led_driver);
#endif
#ifdef CONFIG_LEDS_GPIO_OF
    of_unregister_platform_driver(&of_gpio_leds_driver);
#endif
}

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 register led\_classdev driver;

**Table 3-16 Register led\_classdev Driver**

```

static int __devinit gpio_led_probe(struct platform_device *pdev)
{
...
leds_data = kzalloc(sizeof(struct gpio_led_data) * pdata->num_leds,
GFP_KERNEL);
...
for (i = 0; i < pdata->num_leds; i++) {
    ret = create_gpio_led(&pdata->leds[i], &leds_data[i],
    &pdev->dev, pdata->gpio_blink_set);
    if (ret < 0)
        goto err;
}
...
}

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. Call `gpio_led_set` function to control LEDs' status by accessing `/sys/class/leds/xxx/brightness`;

**Table 3-17 Control LED Status**

```

static void gpio_led_set(struct led_classdev *led_cdev,
    enum led_brightness value)
{
...
    gpio_set_value(led_dat->gpio, level);
}

```

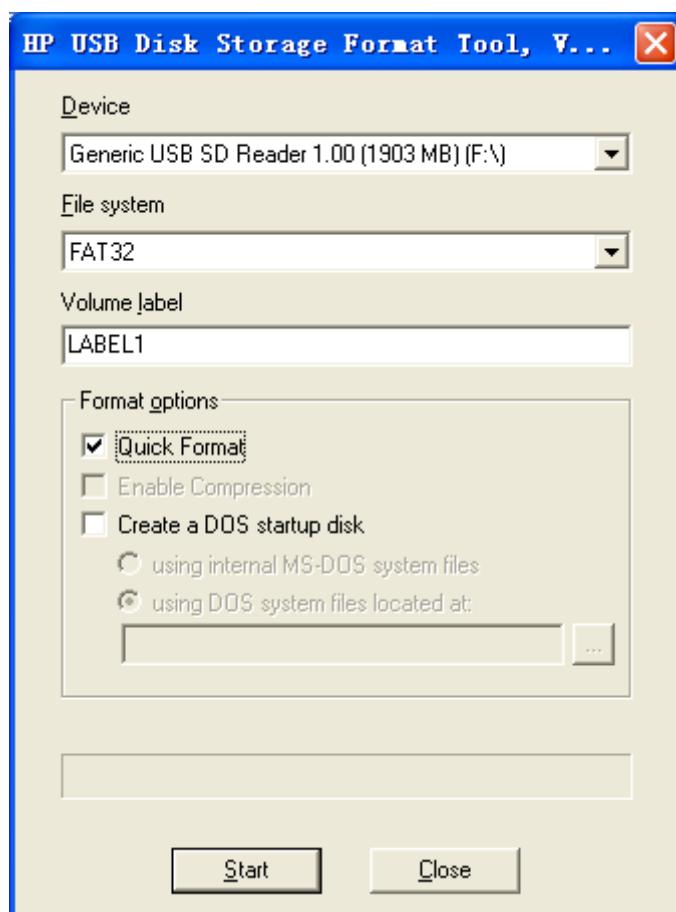
## 3.6 System Update

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

### 3.6.1 Updating System in SD Card

#### 1) **Formatting SD Card;**

You can download HP USB Disk Storage Format Tool 2.0.6 from <http://www.embedinfo.com/english/download/SP27213.exe>, and use it to format SD card; the figure shown below is the tool's interface;



**Figure 3-8** Format SD card

Select **FAT32** in the **File system** drop-down menu, and then click **Start** to format SD card.

**Note:**

- ❑ HP USB Disk Storage Format Tool will erase the partitions of TF card. If partitions need to be retained, please use the format function of Windows system.

**2) Updating Image Files;**

Copy all the files under X:\linux\image\ to a SD card (X is label of your DVD drive), and then insert it on SBC8140 and power on the system; the information on serial interface is shown below;

**Table 3-18 Updating Image Files**

```
Texas Instruments X-Loader 1.47 (Mar 1 2013 - 17:05:22)
Starting X-loader on MMC
Reading boot sector

231872 Bytes Read from MMC
Starting OS Bootloader from MMC...
Starting OS Bootloader...

U-Boot 2010.06-rc1-svn84 (Mar 04 2013 - 12:00:27)

OMAP3630-GP ES2.1, CPU-OPP2 L3-133MHz
OMAP3 SBC8140 board + LPDDR/NAND
I2C: ready
DRAM: 256 MiB
NAND: 512 MiB
*** Warning - bad CRC or NAND, using default environment

In:    serial
Out:   serial
Err:   serial
Die ID #3d1400029e3800000168682f07003018
Net:   dm9000
Hit any key to stop autoboot: 0
mmc1 is available
reading boot.scr

** Unable to read "boot.scr" from mmc 0:1 **
reading ulimage
```

```
2548700 bytes read
reading ramdisk.gz

15345565 bytes read
Booting from ramdisk ...
## Booting kernel from Legacy Image at 81000000 ...
    Image Name:  Linux-2.6.32
    Image Type:  ARM Linux Kernel Image (uncompressed)
    Data Size:   2548636 Bytes = 2.4 MiB
    Load Address: 80008000
    Entry Point: 80008000
    Verifying Checksum ... OK
    Loading Kernel Image ... OK
OK

Starting kernel ...

Uncompressing
Linux.....done, booting the kernel.

Linux version 2.6.32 (tanjian@TIOP) (gcc version 4.4.0 (GCC) ) #5 Sat Mar 2 16:14:46 CST
2013
CPU: ARMv7 Processor [413fc082] revision 2 (ARMv7), cr=10c53c7f
CPU: VIPT nonaliasing data cache, VIPT nonaliasing instruction cache
Machine: OMAP3 SBC8140 Board
Memory policy: ECC disabled, Data cache writeback
OMAP3630/DM3730 ES1.0 (l2cache iva sgx neon isp 192mhz_clk )
SRAM: Mapped pa 0x40200000 to va 0xfe400000 size: 0x100000
Reserving 12582912 bytes SDRAM for VRAM
Built 1 zonelists in Zone order, mobility grouping on. Total pages: 65024
Kernel command line: console=ttyS0,115200n8 mpurate=1000 vram=12M
omapdss.def_disp=lcd omapfb.mode=lcd:4.3inch_LCD root=/dev/ram0 rw
ramdisk_size=65536 initrd=0x81600000,64M rootfstype=ext2
PID hash table entries: 1024 (order: 0, 4096 bytes)
Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)
Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)
Memory: 256MB = 256MB total
Memory: 176768KB available (4388K code, 378K data, 164K init, 0K highmem)
Hierarchical RCU implementation.
NR_IRQS:402
Clocking rate (Crystal/Core/MPU): 26.0/266/600 MHz
Reprogramming SDRC clock to 266000000 Hz
dpll3_m2_clk rate change failed: -22
```

```
GPMC revision 5.0
IRQ: Found an INTC at 0xfa200000 (revision 4.0) with 96 interrupts
Total of 96 interrupts on 1 active controller
OMAP GPIO hardware version 2.5
OMAP clockevent source: GPTIMER12 at 32768 Hz
Console: colour dummy device 80x30
Calibrating delay loop... 480.01 BogoMIPS (lpj=1875968)
Mount-cache hash table entries: 512
CPU: Testing write buffer coherency: ok
regulator: core version 0.5
NET: Registered protocol family 16
Found NAND on CS0
Registering NAND on CS0
Target VDD1 OPP = 4, VDD2 OPP = 2
OMAP DMA hardware revision 5.0
bio: create slab <bio-0> at 0
SCSI subsystem initialized
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
i2c_omap i2c_omap.1: bus 1 rev4.0 at 2600 kHz
twl4030: PIH (irq 7) chaining IRQs 368..375
twl4030: power (irq 373) chaining IRQs 376..383
twl4030: gpio (irq 368) chaining IRQs 384..401
regulator: VUSB1V5: 1500 mV normal standby
regulator: VUSB1V8: 1800 mV normal standby
regulator: VUSB3V1: 3100 mV normal standby
twl4030_usb twl4030_usb: Initialized TWL4030 USB module
regulator: VMMC1: 1850 <-> 3150 mV normal standby
regulator: VDAC: 1800 mV normal standby
regulator: VPLL2: 1800 mV normal standby
regulator: VMMC2: 1850 <-> 3150 mV normal standby
regulator: VSIM: 1800 <-> 3000 mV normal standby
i2c_omap i2c_omap.2: bus 2 rev4.0 at 400 kHz
i2c_omap i2c_omap.3: bus 3 rev4.0 at 400 kHz
Switching to clocksource 32k_counter
musb_hdrc: version 6.0, musb-dma, otg (peripheral+host), debug=0
musb_hdrc: USB OTG mode controller at fa0ab000 using DMA, IRQ 92
NET: Registered protocol family 2
IP route cache hash table entries: 2048 (order: 1, 8192 bytes)
TCP established hash table entries: 8192 (order: 4, 65536 bytes)
TCP bind hash table entries: 8192 (order: 3, 32768 bytes)
TCP: Hash tables configured (established 8192 bind 8192)
```

```
TCP reno registered
UDP hash table entries: 256 (order: 0, 4096 bytes)
UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
NET: Registered protocol family 1
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
RPC: Registered tcp NFSv4.1 backchannel transport module.
Trying to unpack rootfs image as initramfs...
rootfs image is not initramfs (no cpio magic); looks like an initrd
Freeing initrd memory: 65536K
omap-iommu omap-iommu.0: isp registered
NetWinder Floating Point Emulator V0.97 (double precision)
ashmem: initialized
VFS: Disk quotas dquot_6.5.2
Dquot-cache hash table entries: 1024 (order 0, 4096 bytes)
msgmni has been set to 473
alg: No test for stdrng (krng)
io scheduler noop registered
io scheduler deadline registered
io scheduler cfq registered (default)
OMAP DSS rev 2.0
OMAP DISPC rev 3.0
OMAP VENC rev 2
Serial: 8250/16550 driver, 4 ports, IRQ sharing enabled
serial8250.0: ttyS0 at MMIO 0x4806a000 (irq = 72) is a ST16654
console [ttyS0] enabled
serial8250.1: ttyS1 at MMIO 0x4806c000 (irq = 73) is a ST16654
serial8250.2: ttyS2 at MMIO 0x49020000 (irq = 74) is a ST16654
serial8250.3: ttyS3 at MMIO 0x49042000 (irq = 80) is a ST16654
brd: module loaded
loop: module loaded
omap2-nand driver initializing
NAND device: Manufacturer ID: 0xad, Chip ID: 0xbc (Hynix NAND 512MiB 1,8V 16-bit)
cmdlinepart partition parsing not available
Creating 5 MTD partitions on "omap2-nand":
0x000000000000-0x000000080000 : "X-Loader"
0x000000080000-0x000000260000 : "U-Boot"
0x000000260000-0x000000280000 : "U-Boot Env"
0x000000280000-0x000000680000 : "Kernel"
0x000000680000-0x000020000000 : "File System"
PPP generic driver version 2.4.2
PPP Deflate Compression module registered
PPP BSD Compression module registered
```

```
PPP MPPE Compression module registered
NET: Registered protocol family 24
PPPoL2TP kernel driver, V1.0
dm9000 Ethernet Driver, V1.31
eth0: dm9000a at d0862000,d0866400 IRQ 185 MAC: 00:11:22:33:44:55 (chip)
usbcore: registered new interface driver asix
usbcore: registered new interface driver cdc_ether
usbcore: registered new interface driver cdc_eem
usbcore: registered new interface driver dm9601
usbcore: registered new interface driver smsc95xx
usbcore: registered new interface driver gl620a
usbcore: registered new interface driver net1080
usbcore: registered new interface driver plusb
usbcore: registered new interface driver rndis_host
usbcore: registered new interface driver cdc_subset
usbcore: registered new interface driver zaurus
usbcore: registered new interface driver MOSCHIP usb-ethernet driver
ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
ehci-omap ehci-omap.0: OMAP-EHCI Host Controller
ehci-omap ehci-omap.0: new USB bus registered, assigned bus number 1
ehci-omap ehci-omap.0: irq 77, io mem 0x48064800
ehci-omap ehci-omap.0: USB 2.0 started, EHCI 1.00
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 3 ports detected
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
g_ether gadget: using random self ethernet address
g_ether gadget: using random host ethernet address
usb0: MAC 1e:8b:da:88:c8:d7
usb0: HOST MAC d2:49:09:b6:08:e4
g_ether gadget: Ethernet Gadget, version: Memorial Day 2008
g_ether gadget: g_ether ready
musb_hdrc musb_hdrc: MUSB HDRC host driver
musb_hdrc musb_hdrc: new USB bus registered, assigned bus number 2
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 1 port detected
mice: PS/2 mouse device common for all mice
input: gpio-keys as /devices/platform/gpio-keys/input/input0
input: TWL4030 Keypad as
/devices/platform/i2c_omap.1/i2c-1/1-004a/twl4030_keypad/input/input1
ads7846 spi1.0: touchscreen, irq 187
input: ADS7846 Touchscreen as /devices/platform/omap2_mcspi.1/spi1.0/input/input2
```

```
using rtc device, twl_rtc, for alarms
twl_rtc twl_rtc: rtc core: registered twl_rtc as rtc0
twl_rtc twl_rtc: Power up reset detected.
twl_rtc twl_rtc: Enabling TWL-RTC.
i2c /dev entries driver
ch7033 id:5e
Linux video capture interface: v2.00
tvp514x 2-005d: Registered to v4l2 master omap34xxcam!!
omap-iommu omap-iommu.0: isp: version 1.1
usbcore: registered new interface driver uvcvideo
USB Video Class driver (v0.1.0)
OMAP Watchdog Timer Rev 0x31: initial timeout 60 sec
Registered led device: led0
Registered led device: led1
Registered led device: led2
Registered led device: led3
Registered led device: led4
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
Advanced Linux Sound Architecture Driver Version 1.0.21.
usb 1-1: new high speed USB device using ehci-omap and address 2
No device for DAI omap-mcbsp-dai-0
No device for DAI omap-mcbsp-dai-1
No device for DAI omap-mcbsp-dai-2
No device for DAI omap-mcbsp-dai-3
No device for DAI omap-mcbsp-dai-4
OMAP3 SBC8140 SoC init
asoc: twl4030 <-> omap-mcbsp-dai-0 mapping ok
ALSA device list:
#0: omap3sbc8140 (twl4030)
TCP cubic registered
NET: Registered protocol family 17
NET: Registered protocol family 15
Power Management for TI OMAP3.
Unable to set L3 frequency (400000000)
Switched to new clocking rate (Crystal/Core/MPU): 26.0/266/1000 MHz
IVA2 clocking rate: 800 MHz
SmartReflex driver initialized
VFP support v0.3: implementor 41 architecture 3 part 30 variant c rev 3
Console: switching to colour frame buffer device 60x34
regulator_init_complete: incomplete constraints, leaving VDVI on
regulator_init_complete: incomplete constraints, leaving VDAC on
hub 1-1:1.0: USB hub found
```

```
regulator_init_complete: incomplete constraints, leaving VMMC1 on
hub 1-1:1.0: 4 ports detected
twl_rtc twl_rtc: setting system clock to 2000-01-01 00:00:00 UTC (946684800)
mmc0: host does not support reading read-only switch. assuming write-enable.
mmc0: new high speed SD card at address 1234
mmcblk0: mmc0:1234 SA02G 1.85 GiB
mmcblk0: p1
tvp514x 2-005d: chip id mismatch msb:0x87 lsb:0x87
tvp514x 2-005d: Unable to detect decoder
tvp514x 2-005d: chip id mismatch msb:0x87 lsb:0x87
tvp514x 2-005d: Unable to detect decoder
tvp514x 2-005d: chip id mismatch msb:0x87 lsb:0x87
tvp514x 2-005d: Unable to detect decoder
tvp514x 2-005d: chip id mismatch msb:0x87 lsb:0x87
tvp514x 2-005d: Unable to detect decoder
omapdss DPI error: display already enabled
omap_vout omap_vout: 'lcd' Display already enabled
omapdss DPI error: display already enabled
omap_vout omap_vout: 'lcd' Display already enabled
omap_vout omap_vout: Buffer Size = 3686400
omap_vout omap_vout: : registered and initialized video device 0
omap_vout omap_vout: Buffer Size = 3686400
omap_vout omap_vout: : registered and initialized video device 1
RAMDISK: gzip image found at block 0
EXT2-fs (ram0): warning: mounting unchecked fs, running e2fsck is recommended
VFS: Mounted root (ext2 filesystem) on device 1:0.
Freeing init memory: 164K
INIT: version 2.86 booting
Starting udevtar: 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
root: mount: mounting usbfs on /proc/bus/usb 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
Sending discover...
udhcpc: sendto: Network is down
```

```
No lease, failing
done.

Tue Jan 27 08:47:00 UTC 2009

INIT: Entering runlevel: 5
Starting syslogd/klogd: done

.
.
.
|   |
|   |-----|-----||-----|-----| | | | | |
|   |   |__|_--'`-'|_-'|   |   |   |
|   |   |   |   |---||--'`|_-'|_-'|_-'|_-'|
'-----|-----|-----|-----|-----|-----|-----'
          |
          |

The Angstrom Distribution SBC8140 ttyS0

Angstrom 2008.1-test-20090127 SBC8140 ttyS0

SBC8140 login: root (enter root to log in)
```

The above information indicates that Linux system has booted up successfully from SD card.

**Note:**

- By default SBC8140 boots from NAND Flash; pressing and holding BOOT button before connecting power can make it boot from SD card.
- By default, the system support 4.3-inch screen. If you need to select another display mode, please refer to 3.7 Display Mode Configurations to change the display mode and type boot under u-boot mode to continue boot-up process.

### 3.6.2 Updating System in NAND Flash

Updating image files in NAND Flash requires the help of u-boot. No matter whether there is data in NAND Flash, image files can be updated by running u-boot from SD card.

- 1) Use HP USB Disk Storage Format Tool 2.0.6 to format SD card to FAT or FAT32 filesystem;

- 2) Copy the files MLO, u-boot.bin, x-load.bin.ift\_for\_NAND, flash-uboot.bin, ulimage and ubi.img from \linux\image of DVD-ROM to SD card;
- 3) Insert SD card on SBC8140 and power it on; when the information on serial interface shows countdown in seconds, press any key on your PC's keyboard to enter u-boot mode;

**Table 3-19 Enter u-boot Mode**

```
Texas Instruments X-Loader 1.47 (Mar 1 2013 - 17:05:22)
```

```
Starting X-loader on MMC
```

```
Reading boot sector
```

```
231872 Bytes Read from MMC
```

```
Starting OS Bootloader from MMC...
```

```
Starting OS Bootloader...
```

```
U-Boot 2010.06-rc1-svn84 (Mar 04 2013 - 12:00:27)
```

```
OMAP3630-GP ES2.1, CPU-OPP2 L3-133MHz
```

```
OMAP3 SBC8140 board + LPDDR/NAND
```

```
I2C: ready
```

```
DRAM: 256 MiB
```

```
NAND: 512 MiB
```

```
*** Warning - bad CRC or NAND, using default environment
```

```
In: serial
```

```
Out: serial
```

```
Err: serial
```

```
Die ID #3d1400029e3800000168682f07003018
```

```
Net: dm9000
```

```
Hit any key to stop autoboot: 0 (press any key to enter u-boot mode)
```

- 4) Type **run updatesys** and press **Enter** key to start system update; the information on serial interface is shown below;

**Table 3-20 Updating System**

```
OMAP3 SBC8140 # run updatesys
```

```
NAND erase: device 0 whole chip
```

```
Erasing at 0x1ffe0000 -- 100% complete.
```

```
OK
mmc1 is available
reading x-load.bin.ift_for_NAND

11648 bytes read
HW ECC selected

NAND write: device 0 offset 0x0, size 0x2d80
12288 bytes written: OK
reading flash-uboot.bin

231872 bytes read
SW ECC selected

NAND write: device 0 offset 0x80000, size 0x389c0
233472 bytes written: OK
reading ulimage

2548700 bytes read
SW ECC selected

NAND write: device 0 offset 0x280000, size 0x26e3dc
2549760 bytes written: OK
reading ubi.img

12320768 bytes read
SW ECC selected

NAND write: device 0 offset 0x680000, size 0xbc0000
12320768 bytes written: OK
```

When the LEDs on the kit start to blink, the update is completed; please remove SD card and reboot the system;

### 3.7 Display Mode Configurations

The system supports multiple display modes. Users can select an appropriate mode by configuring booting parameters. The following contents show how to configure for 4.3-inch and 7-inch LCDs, VGA mode and LVDS mode.

You need to enter u-boot mode to realize configurations. Please reboot the kit and press any key on your PC's keyboard when the system prompts you with a countdown in seconds as shown below;

**Table 3-21 Booting Information**

```
Texas Instruments X-Loader 1.47 (Mar 1 2013 - 17:05:22)
Starting X-loader on MMC
Reading boot sector

231872 Bytes Read from MMC
Starting OS Bootloader from MMC...
Starting OS Bootloader...

U-Boot 2010.06-rc1-svn84 (Mar 04 2013 - 12:00:27)

OMAP3630-GP ES2.1, CPU-OPP2 L3-133MHz
OMAP3 SBC8140 board + LPDDR/NAND
I2C: ready
DRAM: 256 MiB
NAND: 512 MiB
*** Warning - bad CRC or NAND, using default environment

In: serial
Out: serial
Err: serial
Die ID #3d1400029e3800000168682f07003018
Net: dm9000
Hit any key to stop autoboot: 0 (press any key to enter u-boot mode)
```

**1) Configuring for 4.3-inch LCD;**

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

```
OMAP3 SBC8140 # setenv defaultdisplay lcd
```

```
OMAP3 SBC8140 # setenv dispmode 4.3inch_LCD
```

```
OMAP3 SBC8140 # saveenv
```

2) Configuring for 7-inch LCD;

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

```
OMAP3 SBC8140 # setenv defaultdisplay lcd
```

```
OMAP3 SBC8140 # setenv dispmode 7inch_LCD
```

```
OMAP3 SBC8140 # saveenv
```

3) Configuring for VGA;

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

```
OMAP3 SBC8140 # setenv defaultdisplay lcd
```

```
OMAP3 SBC8140 # setenv dispmode VGA
```

```
OMAP3 SBC8140 # saveenv
```

4) Configuring for LVDS;

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

```
OMAP3 SBC8140 # setenv defaultdisplay lcd
```

```
OMAP3 SBC8140 # setenv dispmode LVDS
```

```
OMAP3 SBC8140 # saveenv
```

## 3.8 Tests and Demonstrations

This section will carry out many tests on the devices on SBC8140 and also demonstrations of Android and DVSDK systems.

**Note:**

- BOOK The following tests are all implemented by entering instructions in a HyperTerminal window.

### 3.8.1 Testing LED

- 1) Execute the following instructions to test LED0;

```
root@SBC8140:# echo 1 > /sys/class/leds/led0/brightness
```

```
root@SBC8140:# echo 0 > /sys/class/leds/led0/brightness
```

- 2) Execute the following instructions to test LED1;

```
root@SBC8140:# echo 1 > /sys/class/leds/led1/brightness
```

```
root@SBC8140:# echo 0 > /sys/class/leds/led1/brightness
```

- 3) Execute the following instructions to test LED2;

```
root@SBC8140:# echo 1 > /sys/class/leds/led2/brightness
```

```
root@SBC8140:# echo 0 > /sys/class/leds/led2/brightness
```

- 4) Execute the following instructions to test LED3;

```
root@SBC8140:# echo 1 > /sys/class/leds/led3/brightness
```

```
root@SBC8140:# echo 0 > /sys/class/leds/led3/brightness
```

When executing each instruction, the corresponding LED will be turned on or turned off.

### 3.8.2 Testing Touch-Screen

Boot up SBC8140 from NAND Flash and start testing;

- 1) Execute the following instruction to calibrate touch-screen;

```
root@SBC8140: # ts_calibrate
```

Touch all the “+” marks on the screen by following the screen prompt information to finish calibration;

- 2) Execute the following instruction to test touch-screen;

```
root@SBC8140: # ts_test
```

Draw points and lines on the screen as you see the prompt information to proceed with testing;

### 3.8.3 Testing RTC

SBC8140 has a hardware clock which can store and recover system clock; please carry out the test of RTC through the following steps;

- 1) Execute the following instruction to set system clock to 8 pm, Aug. 8<sup>th</sup>, 2011;

```
root@SBC8140 : # date 080820002011
```

The information in HyperTerminal window is shown below;

**Table 3-22 Setting System Clock**

Mon Aug 8 20:00:00 UTC 2011
-----------------------------

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

```
root@SBC8140: # hwclock -w
```

- 3) Execute the following instruction to read RTC;

```
root@SBC8140: # hwclock
```

The information in HyperTerminal window is shown below;

**Table 3-23 RTC Clock**

Mon Aug 8 20:01:01 2011 0.000000 seconds
--

The above information indicates that system clock has been stored in hardware clock;

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

```
root@SBC8140: # hwclock -s
```

```
root@SBC8140: # date
```

The information in HyperTerminal window is shown below;

**Table 3-24 System Clock**

Mon Aug 8 20:01:01 2011 0.000000 seconds
--

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

**Note:**

■ SBC8140 has no CR2032 battery by default and you need to purchase it separately.

### 3.8.4 Testing SD Card

- 1) Insert SD card on SBC8140, system will mount it under /media/ automatically; please execute the following instructions to view the name of SD device in the system;

```
root@SBC8140:~# cd /media/
```

```
root@SBC8140:/media# ls
```

The information in HyperTerminal window is shown below;

**Table 3-25 SD Device Name**

card	hdd	mmcblk0p1	ram	union
cf	mmc1	net	realroot	

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

```
root@SBC8140:/media# ls mmcblk0p1/
```

The information in HyperTerminal window is shown below;

**Table 3-26 Contents of SD Card**

flash-u-boot.bin	u-boot.bin	x-load.bin.ift_for_NAND
mlo	ulimage	
ramdisk.gz	ubi.img	

### 3.8.5 Testing USB Device

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

- 1) After system boots up, connect SBC8140 to your PC with a Mini B-to-USB A cable, and then you need to install Linux USB Ethernet driver; please refer to Appendix 2 – Installing Linux USB Ethernet/RNDIS Gadget.
- 2) Execute the following instructions to set the IP addresses of SBC8140 (the

IP used below is only for reference; you can select a different IP);

```
root@SBC8140:~# ifconfig usb0 192.168.1.115
```

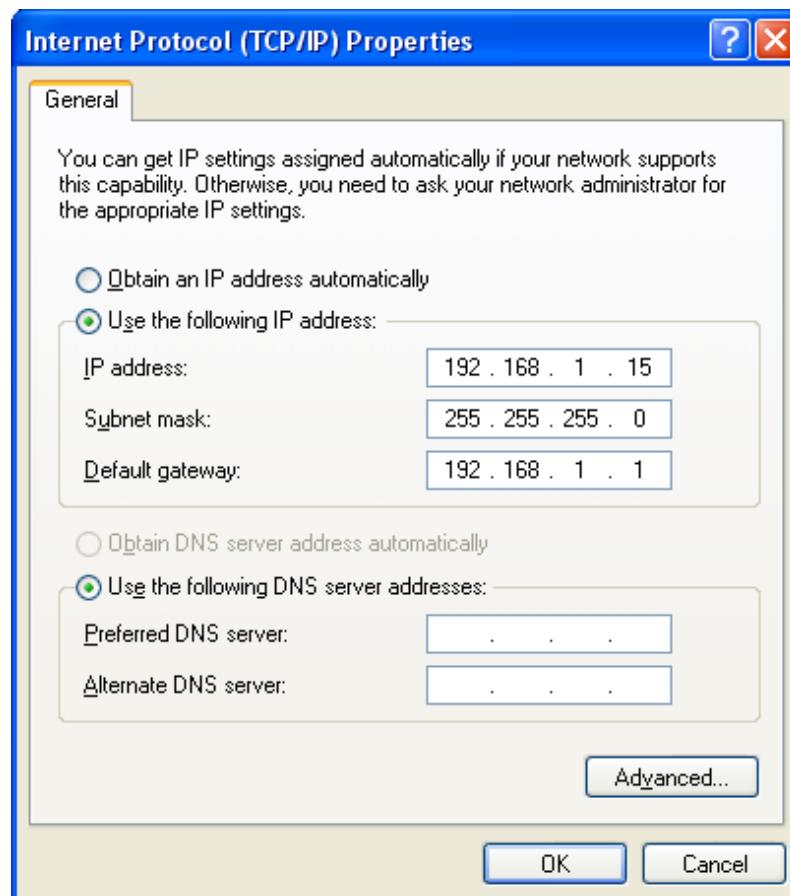
```
root@SBC8140:~# ifconfig
```

The information in HyperTerminal window is shown below;

**Table 3-27 Setting IP Address**

lo	Link encap:Local Loopback inet addr:127.0.0.1 Mask:255.0.0.0 UP LOOPBACK RUNNING MTU:16436 Metric:1 RX packets:26 errors:0 dropped:0 overruns:0 frame:0 TX packets:26 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:0 RX bytes:2316 (2.2 KiB) TX bytes:2316 (2.2 KiB)
usb0	Link encap:Ethernet HWaddr 5E:C5:F6:D4:2B:91 inet addr:192.168.1.115 Bcast:192.168.1.255 Mask:255.255.255.0 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:253 errors:0 dropped:0 overruns:0 frame:0 TX packets:43 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:35277 (34.4 KiB) TX bytes:10152 (9.9 KiB)

- 3) Right-click **My Network Places** on the desktop of your PC and select **Properties** to open **Network Connections** window; you can find a new **Local Area Connection** in the window;
- 4) Right-click the icon of new **Local Area Connection** and select **Properties**, and then double-click **Internet Protocol (TCP/IP)** to open the following window;



**Figure 3-9** Setting IP address

Setting the IP address of USB virtual network interface as the same network segment as that SBC8140's IP was set in, and then click **OK**;

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

```
root@SBC8140:~# ping 192.168.1.15
```

The information in HyperTerminal window is shown below;

**Table 3-28 Testing Information**

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

The above information indicates the network has been created successfully.

### 3.8.6 Testing USB HOST

- 1) Insert a flash disk into the USB interface on SBC8140, system will display the following information;

**Table 3-29 Information on Serial Interface**

```
root@SBC8140:/# usb 1-1.4: new high speed USB device using ehci-omap and address 3
scsi0 : usb-storage 1-1.4:1.0
scsi 0:0:0:0: Direct-Access      SanDisk  Flash Memory      0.1  PQ: 0 ANSI: 2
sd 0:0:0:0: [sda] 2001888 512-byte logical blocks: (1.02 GB/977 MiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] Assuming drive cache: write through
sda: sda1
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] Attached SCSI removable disk
```

- 2) The system will mount the flash disk under /media/ automatically; please execute the following instructions to view contents of the disk;

```
root@SBC8140:/media# ls /media/sda1/
```

The information in HyperTerminal window is shown below;

**Table 3-30 Contents of Flash Disk**

flash-uboot.bin	u-boot.bin	x-load.bin.ift_for_NAND
mlo	ulimage	
ramdisk.gz	ubi.img	

### 3.8.7 Testing Audio Function

SBC8140 has input/output interfaces which support audio recording and playback. The filesystem is integrated with als-audio recording and playback tool. You can test it by following the steps below;

- 1) Insert a microphone into the 3.5mm audio input interface (the green one) on SBC8140, and then execute the following instruction to start audio recording;

```
root@SBC8140:~# arecord -t wav -c 1 -r 44100 -f S16_LE -v k
```

The information in HyperTerminal window is shown below;

**Table 3-31 Start Recording**

```
Recording WAVE 'k' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Plug PCM: Hardware PCM card 0 'omap3evm' device 0 subdevice 0
Its setup is:
stream      : CAPTURE
access       : RW_INTERLEAVED
format       : S16_LE
subformat    : STD
channels     : 2
rate         : 44100
exact rate   : 44100 (44100/1)
msbits       : 16
buffer_size  : 22052
period_size  : 5513
period_time  : 125011
tstamp_mode  : NONE
period_step  : 1
avail_min    : 5513
period_event : 0
start_threshold  : 1
stop_threshold : 22052
silence_threshold: 0
silence_size  : 0
boundary     : 1445199872
appl_ptr     : 0
hw_ptr       : 0
```

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

```
root@SBC8140:~# aplay -t wav -c 2 -r 44100 -f S16_LE -v k
```

The information in HyperTerminal window is shown below;

**Table 3-32 Start Playback**

```
Playing WAVE 'k' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Plug PCM: Hardware PCM card 0 'omap3evm' device 0 subdevice 0
Its setup is:
stream      : PLAYBACK
access       : RW_INTERLEAVED
format       : S16_LE
```

```
subformat      : STD
channels       : 2
rate           : 44100
exact rate     : 44100 (44100/1)
msbits         : 16
buffer_size    : 22052
period_size    : 5513
period_time    : 125011
tstamp_mode    : NONE
period_step    : 1
avail_min      : 5513
period_event   : 0
start_threshold: 22052
stop_threshold : 22052
silence_threshold: 0
silence_size   : 0
boundary        : 1445199872
appl_ptr        : 0
hw_ptr          : 0
```

### 3.8.8 Testing Network

- 1) Setting the IP address of SBC8140 as the same network segment as that your PC is set in, for example, execute the following instructions;

```
root@SBC8140:~# ifconfig eth0 192.192.192.203
```

```
root@SBC8140:~# ifconfig
```

The information in HyperTerminal window is shown below;

**Table 3-33 Setting IP Address**

eth0	Link encap:Ethernet HWaddr 00:11:22:33:44:55
	inet addr:192.192.192.203 Bcast:192.192.192.255 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)
	Interrupt:185 Base address:0x2000

- 2) Execute the following instruction to test the network communication between

SBC8140 and PC;

```
root@SBC8140:~# ping 192.192.192.170
```

The information in HyperTerminal window is shown below;

**Table 3-34 Testing Network**

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

The above information indicates the network is working properly.

### 3.8.9 Testing Camera

Connect camera module (needs to be purchased separately), CCD camera and LCD screen to SBC8140, and then execute the following instructions;

```
root@SBC8140:~# saMmapLoopback
```

The information in HyperTerminal window is shown below;

**Table 3-35 Testing Camera**

```
tvp514x 2-005d: tvp5146m2 found at 0xba (OMAP I2C adapter)
```

```
Capture: Opened Channel
```

```
Capture: Current Input: COMPOSITE
```

```
Capture: Current standard: PAL
```

```
Capture: Capable of streaming
```

```
Capture: Number of requested buffers = 3
```

```
Capture: Init done successfully
```

```
Display: Opened Channel
```

```
Display: Capable of streaming
```

```
Display: Number of requested buffers = 3
```

```
Display: Init done successfully
```

```
Display: Stream on...
```

```
Capture: Stream on...
```

The images captured by CCD camera can be seen on LCD screen.

### 3.8.10 Testing CDMA8000-U Module

Please download the user manual of the module from <http://www.timll.com/chinese/uploadFile/cdma8000.rar> and follow the instructions in the manual to do the test.

**Note:**

 CDMA8000-U is an optional module. You need to purchase it separately.

### 3.8.11 Testing WCDMA8000-U Module

Please download the user manual of the module from <http://www.timll.com/chinese/uploadFile/WCDMA8000.zip> and follow the instructions in the manual to do the test.

**Notice:**

 WCDMA8000-U is an optional module. You need to purchase it separately.

### 3.8.12 Demonstration of Android System

Copy all the files under X:\linux\demo\Android\image (X is label of your DVD drive) to SD card and insert it on SBC8140, and then power on the kit while pressing and holding BOOT button (Button CN11); The information in HyperTerminal window is shown below;

**Table 3-36 Programming Information**

60
Texas Instruments X-Loader 1.47 (Apr 23 2012 - 09:09:16)
Starting X-loader on MMC
Reading boot sector
1154092 Bytes Read from MMC
Starting OS Bootloader from MMC...

```
Starting OS Bootloader...

U-Boot 2010.06-rc1-svn (Apr 17 2012 - 10:28:23)

OMAP34xx/35xx-GP ES2.1, CPU-OPP2 L3-165MHz
OMAP3 SBC8140 board + LPDDR/NAND
I2C:    ready
DRAM:   256 MiB
NAND:   512 MiB
*** Warning - bad CRC or NAND, using default environment

In:     serial
Out:    serial
Err:    serial
SBC8140 xM Rev A
Die ID #259000029e38000001683b060d023028

NAND erase: device 0 whole chip
Skipping bad block at 0x08660000
Erasing at 0x1ffe0000 -- 100% complete.
OK
mmc1 is available
reading x-load.bin.ift_for_NAND

11668 bytes read
HW ECC selected

NAND write: device 0 offset 0x0, size 0x2d94
12288 bytes written: OK
reading flash-uboot.bin

1152640 bytes read
SW ECC selected

NAND write: device 0 offset 0x80000, size 0x119680
1153024 bytes written: OK
reading ulimage

2573772 bytes read
SW ECC selected

NAND write: device 0 offset 0x280000, size 0x2745cc
```

```
2574336 bytes written: OK  
reading ubi.img  
79036416 bytes read  
SW ECC selected  
  
NAND write: device 0 offset 0x680000, size 0x4b60000  
79036416 bytes written: OK
```

When the LEDs on the kit start to blink, the programming process is completed; please remove SD card and reboot the kit to enter Android operating system.

**Note:**

- BOOK By default, the system support 4.3-inch screen. If you need to select another display mode, please refer to 3.7 Display Mode Configurations.

### 3.8.13 Demonstration of DVSDK System

DVSDK (Digital Video Software Development Kit) is software released by TI to build connections between ARM processors and DSPs.

Applications are running on ARM end which process IO interfaces and applications. ARM uses VISA APIs interface provided by Codec Engine to process videos, images and audio signals; and then Codec Engine communicates with the Codec Engine server created by DSP using DSP/BIOS Link as well as xDIAS and xDM protocols. DSP will process these signals and put the results in a memory space shared with ARM, allowing access to the results from ARM end.

- 1) Format a SD card as two partitions (please refer to Appendix 3 – Making Linux Boot Disk) and connect it to your PC with a SD card reader, and then execute the following instructions in Ubuntu Linux system;

```
cp /media/cdrom/linux/demo/dv sdk/image/MLO /media/LABEL1  
cp /media/cdrom/linux/demo/dv sdk/image/u-boot.bin /media/LABEL1  
cp /media/cdrom/linux/demo/dv sdk/image/uImage /media/LABEL1/uImage  
rm -rf /media/LABEL2/*
```

```
sudo tar xvf /media/cdrom/linux/demo/dv sdk/image/dv sdk-dm37x-evm-rootfs.tar.
```

```
bz2 -C /media/LABEL2
```

```
sync
```

```
umount /media/LABEL1
```

```
umount /media/LABEL2
```

- 2) Insert SD card on SBC8140 and power on the kit; the information in HyperTerminal window is shown below;

Table 3-37 System Login

```
2548012 bytes read
## Booting kernel from Legacy Image at 80300000 ...
Image Name: Linux-2.6.32
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 2547948 Bytes = 2.4 MiB
Load Address: 80008000
Entry Point: 80008000
Verifying Checksum ... OK
Loading Kernel Image ... OK
OK

Starting kernel ...
..... //中间部分省略
Arago Project http://arago-project.org dm37x-evm ttyS2

Arago 2010.07 dm37x-evm ttyS2

dm37x-evm login:root (enter root here to log in)
```

Enter user name **root** to log in the system when you see prompt information

**dm37x-evm login** in HyperTerminal window;

- 3) The filesystem of DVSDK has been pre-installed some applications; the following contents will take GStreamer pipelines as the example to demonstrate H.264 decoding; please execute the following instructions;

```
root@dm37x-evm:cd /usr/share/ti/gst/omap3530
```

```
root@dm37x-evm:/usr/share/ti/gst/omap3530# ./loadmodules.sh
```

```
root@dm37x-evm:/usr/share/ti/gst/omap3530# gst-launch filesrc location=/usr/shar  
e(ti/data/videos/davincieffect_480p30.264 !! typefind ! TIViddec2 ! TIDmaiVideoS  
ink -v
```

The information in HyperTerminal window is shown below;

**Table 3-38 Running Application**

```
Setting pipeline to PAUSED ...
/GstPipeline:pipeline0/GstTypeFindElement:typefindelement0.GstPad:src: caps =
video/x-h264

Pipeline is PREROLLING ...
/GstPipeline:pipeline0/GstTIViddec2:tividdec20.GstPad:sink: caps = video/x-h264
/GstPipeline:pipeline0/GstTIViddec2:tividdec20.GstPad:src: caps = video/x-raw-yuv,
format=(fourcc)UYVY, framerate=(fraction)30000/1001, width=(int)720, height=(int)576
/GstPipeline:pipeline0/GstTIViddec2:tividdec20.GstPad:src: caps = video/x-raw-yuv,
format=(fourcc)UYVY, framerate=(fraction)30000/1001, width=(int)720, height=(int)480
/GstPipeline:pipeline0/GstTIDmaiVideoSink:tidmaivideosink0.GstPad:sink: caps =
video/x-raw-yuv, format=(fourcc)UYVY, framerate=(fraction)30000/1001, width=(int)720,
height=(int)480

Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
```

A video clip will be played on LCD screen.

**Note:**

- ❑ For the details of DVSDK, please visit TI's website or view TMS320DM3730\_Software\_Developers\_Guide.pdf under X:\linux\demo\dv sdk\source\ (X is label of your DVD drive).
- ❑ By default, the system support 4.3-inch screen. If you need another display mode, please refer to 3.7 Display Mode Configurations.

## 3.9 Development of Applications

This section will introduce the common process of development applications through an example of LED application.

- 1) Compose source code led\_acc.c to instruct the three LEDs on SBC8140 to

blink in the mode of accumulator;

**Table 3-39 Source Code of LED Application**

```
#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/led1/brightness"
#define LED2 "/sys/class/leds/led2/brightness"
#define LED3 "/sys/class/leds/led3/brightness"

int main(int argc, char *argv[])
{
    int f_led1, f_led2, f_led3;
    unsigned char i = 0;
    unsigned char dat1, dat2, dat3;
    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;
    }
    if((f_led3 = open(LED3, O_RDWR)) < 0){
        printf("error in open %s",LED3);
        return -1;
    }
    for(;){
        i++;
        dat1 = i&0x1 ? '1':'0';
        dat2 = (i&0x2)>>1 ? '1':'0';
        dat3 = (i&0x4)>>2 ? '1':'0';
        write(f_led1, &dat1, sizeof(dat1));
        write(f_led2, &dat2, sizeof(dat2));
        write(f_led3, &dat3, sizeof(dat3));
        usleep(300000);
    }
}
```

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

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

- 3) Download the compiled files to SBC8140 and enter the directory where the file led\_acc is saved, and execute the following instruction to run LED application;

```
./led_acc &
```

# Chapter 4 WinCE Operating System

This chapter will mainly cover the system and application development based on SBC8140 under Windows Embedded CE 6.0 R3, as well as the software resources and features in DVD-ROM, building development environment, and how to compile projects and BSP (Board Support Package).

## 4.1 Software Resources

The DVD-ROM provided along with the kit contains abundant software resources; the following tables will help find them in the DVD-ROM (X is label of your DVD drive).

**Table 4-1 BSP Package**

<b>BSP</b>	X:\WINCE600\bsp\mini8510.rar
	X:\WINCE600\bsp\COMMON_TI_V1.rar
	X:\WINCE600\bsp\dv sdk_wince_01_11_00_02.rar
	X:\WINCE600\SGX\wince_gfx_sgx_01_01_00_patch_01_setup.exe

**Table 4-2 Windows Embedded CE 6.0 R3 Example Project**

<b>Example Project</b>	X:\WINCE600\prj\mini8510.rar
------------------------	------------------------------

**Table 4-3 Example Application**

<b>Example Application</b>	X:\WINCE600\app\GPIOAppDemo.rar
----------------------------	---------------------------------

**Table 4-4 Pre-Compiled Image Files**

<b>Pre-Compiled Image files</b>	X:\WINCE600\image\	
	MLO	First bootloader for SD card boot
	XLDRNAND.nb0	First bootloader for nand boot
	EBOOTSD.nb0	Second bootloader for SD card boot
	EBOOTNAND.nb0	Second bootloader for nand boot
	NK.bin	WinCE runtime image

## 4.2 BSP Package

**Table 4-5   BSP Package**

Categories	Codes	Code Types
X-Loader (First-Level Boot Code)	NAND	Source Code
	NOR	Source Code
	SD	Source Code
EBOOT (Second-Level Boot Code)	NAND	Source Code
	NOR	Source Code
	SD	Source Code
OAL	KILT(USB RNDIS)	Source Code
	REBOOT	Source Code
	Watchdog	Source Code
	RTC	Source Code
	System timer	Source Code
	Interrupt controller	Source Code
	Low power suspend	Source Code
驱动程序	NLED driver	Source Code
	GPIO/I2C/SPI/MCBSP driver	Source Code
	Series port driver	Source Code
	6X6 keyboard driver	Source Code
	Audio driver	Source Code
	NAND driver	Source Code
	Display driver (LCD/DVI/VGA/S-Video/Composite Video)/ TOUCH driver	Source Code
	SD/MMC/SDIO driver	Source Code
	DM9000 network card driver	Source Code
	USB OTG driver	Source Code
	USB EHCI driver	Source Code
	VRFB driver	Source Code
	DSPLINKK/CMEMK driver	Binary Code
	AAC/MPEG2/MPEG4/H264 DSP Hardware decode fitler	Binary Code
	GPIO keyboard driver	Source Code
	PWM(TPS65930) driver	Source Code
	ADC(TPS65930) driver	Source Code
	ONENAND driver	Source Code
	Analog Camera driver	Source Code

Categories	Codes	Code Types
	Digital Camera driver	Source Code
	DMA driver	Source Code
	RTC driver	Source Code
	Backlight driver	Source Code
	Battery driver	Source Code
	Sleep / wakeup button driver	Source Code
	DVFS/Smart Reflex	Source Code
<b>SDK</b>	powerVR DDK & SDK	Binary Code & Source Code

## 4.3 Process of System Development

This section will walk you through the system development process by introducing installation of IDE (integrated development environment), uncompressing/copying BSP and example project, and compilation of first-level sysgen and BSP.

### 4.3.1 Installing IDE

To build a WinCE IDE, a series of software as listed in the following table need to be installed under Windows XP or Vista; please make sure they are installed in the order specified in the table below so as to avoid unexpected errors.

**Table 4-6 Software to be Installed**

No.	Software Names
1	Visual Studio 2005
2	Visual Studio 2005 SP1
3	Visual Studio 2005 SP1 Update for Vista (vista system require)
4	Windows Embedded CE 6.0 Platform Builder
5	Windows Embedded CE 6.0 SP1
6	Windows Embedded CE 6.0 R2
7	Windows Embedded CE 6.0 Product Update Rollup 12/31/2008
8	Windows Embedded CE 6.0 R3
9	Windows Embedded CE 6.0 Product Update Rollup 12/31/2009
10	ActiveSync 4.5
11	Windows Mobile 6 Professional SDK

### 4.3.2 Uncompressing/Copying BSP and Example Project

Please follow the information in the table below to uncompress/copy BSP and example project from DVD-ROM to specified location on PC.

**Table 4-7 Uncompress/Copy Files**

Operations	Source Address	Destination Address
<b>Uncompress</b>	X:\WNCE600\bsp\mini8510.rar	C:\WINCE600\PLATFORM
<b>Uncompress</b>	X:\WINCE600\bsp\COMMON_TI_V1.rar	C:\WINCE600\PLATFORM\COMMON\SR C\SOC
<b>Uncompress</b>	X:\WINCE600\bsp\dvssdk_wince_01_11_00_02.rar	C:\WINCE600\3rdParty\
<b>Install</b>	X:\WINCE600\SGX\wince_gfx_sgx_01_01_00_patch_01_setup.exe	C:\TI\wince_gfx_sgx_01_01_00_patch_01
<b>Copy</b>	C:\TI\wince_gfx_sgx_01_01_00_patch_01\poweVR	C:\WINCE600\public
<b>Copy</b>	X:\WINCE600\prj\mini8510	C:\WINCE600\OSDesigns

**Note:**

- ❑ The default installation path of Windows Embedded CE 6.0 is C:\WINCE600 in this document.

### 4.3.3 Compiling Sysgen and BSP

Please follow the steps listed below to complete compilation of sysgen and BSP.

- 1) Open project file mini8510.sln under C:\WINCE600\OSDesigns\mini8510;
- 2) Select **Build > Build Solution** on the menu bar of Visual Studio 2005 window to start compiling sysgen and BSP;
- 3) After compilation is done, copy images MLO, EBOOTSD.nb0 and NK from **C:\WINCE600\OSDesigns\mini8510\mini8510\RelDir\mini8510\_ARMV4I\_Release** to SD card and insert it on SBC8140, and the power on the kit;
- 4) Press **Space** key on your PC's keyboard to enter eboot menu and type **a** to select a proper graphic output, and then type number **0** to boot the system;

## 4.4 Introduction to Drivers

The figure shown below illustrates the architecture of BSP for SBC8140;

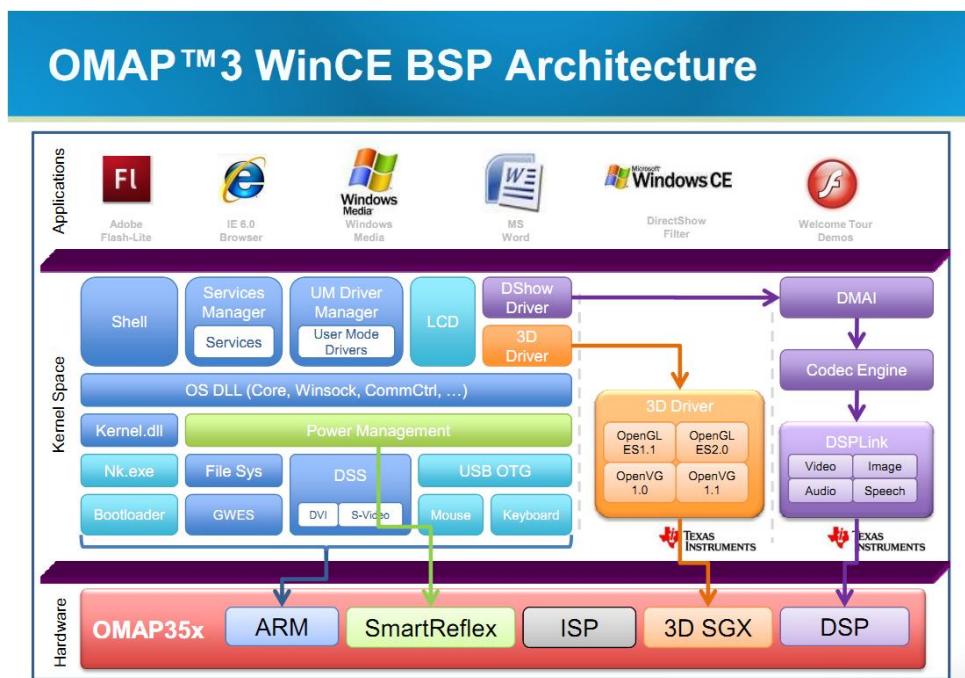


Figure 4-1 BSP Architechture

The following table lists the paths of all the driver source code in BSP;

Table 4-8 Paths of Driver Souce Code in BSP

Driver Names	Paths
<b>NLED driver</b>	bsp\mini8510\SRC\DRIVERS\NLED
<b>GPIO</b>	bsp\mini8510\SRC\DRIVERS\GPIO
	bsp\COMMON_TI_V1\COMMON_TI\GPIO
<b>I2C</b>	bsp\COMMON_TI_V1\COMMON_TI\OAL\OMAP_OALI2C
	bsp\COMMON_TI_V1\COMMON_TI\CEDDKI2C
<b>SPI</b>	bsp\COMMON_TI_V1\COMMON_TI\SPI
<b>MCBSP driver</b>	bsp\COMMON_TI_V1\COMMON_TI\MCBSP
	bsp\COMMON_TI_V1\OMAP3530\MCBSP
<b>Series port driver</b>	bsp\COMMON_TI_V1\COMMON_TI\SERIAL
<b>6X6 keyboard driver</b>	bsp\COMMON_TI_V1\COMMON_TI\KEYPAD
	bsp\mini8510\SRC\DRIVERS\TPS659XX_KEYPAD
<b>Audio driver</b>	bsp\mini8510\SRC\DRIVERS\TPS659XX_WAVE
	bsp\COMMON_TI_V1\TPS659XX\WAVE
<b>NAND driver</b>	bsp\OMAP35XX_TPS659XX_TI_V1\omap35xx\BLOCK
	bsp\COMMON_TI_V1\COMMON_TI\BLOCK

<b>Driver Names</b>	<b>Paths</b>
<b>Display driver (LCD/DVI. S -Video/Composite Video)</b>	bsp\COMMON_TI_V1\COMMON_TI\DSS
	bsp\mini8510\SRC\BSP_COMMON\DISPLAY
	bsp\mini8510\SRC\DRIVERS\DISPLAY
<b>TOUCH driver</b>	bsp\mini8510\SRC\DRIVERS\TOUCH
<b>SD/MMC/SDIO driver</b>	bsp\mini8510\SRC\DRIVERS\SDBUS
	bsp\mini8510\SRC\DRIVERS\SDHC
	bsp\mini8510\SRC\DRIVERS\SDMEMORY
	bsp\COMMON_TI_V1\COMMON_TI\SDHC
<b>SMSC9514 network card driver</b>	bsp\mini8510\SRC\DRIVERS\SMSC9514
<b>USB OTG driver</b>	bsp\mini8510\SRC\DRIVERS\MUSB
	bsp\COMMON_TI_V1\OMAP3530\MUSB
	bsp\COMMON_TI_V1\TPS659XX\USBOTG
<b>USB EHCI driver</b>	bsp\COMMON_TI_V1\COMMON_TI\USB
	bsp\COMMON_TI_V1\OMAP3530\USB
	mini8510\SRC\DRIVERS\USBHS
<b>VRFB driver</b>	bsp\COMMON_TI_V1\COMMON_TI\VRFB
<b>DSPLINKK/CMEMK</b>	bsp\3rdParty\dvstk_wince_01_11_00_02
<b>AAC/MPEG2/MPEG4/H264 DSP hardware decode filter</b>	bsp\3rdParty\dvstk_wince_01_11_00_02
<b>GPIO keyboard driver</b>	bsp\COMMON_TI_V1\COMMON_TI\KEYPAD
	bsp\mini8510\SRC\DRIVERS\TPS659XX_KEYPAD
<b>PWM(TPS65930)driver</b>	bsp\COMMON_TI_V1\TPS659XXTLED
<b>ADC(TPS65930)driver</b>	bsp\COMMON_TI_V1\TPS659XX\MADC
<b>Camera driver</b>	bsp\mini8510\SRC\DRIVERS\CAMERA
	bsp\mini8510\SRC\DRIVERS\CAMERA_Digital
<b>Backlight driver</b>	bsp\mini8510\SRC\DRIVERS\BACKLIGHT
<b>Battery driver</b>	bsp\mini8510\SRC\DRIVERS\BATTERY
<b>Sleep/wake-up button driver</b>	bsp\mini8510\SRC\DRIVERS\TPS659XX_PWRKEY (known issue: system can not be waked up from suspending mode when tps65930 otg is involved)
<b>DVFS/Smart Reflex</b>	bsp\COMMON_TI_V1\COMMON_TI\PM
	bsp\mini8510\SRC\DRIVERS\PM
<b>DMA driver</b>	bsp\COMMON_TI_V1\COMMON_TI\SDMA
<b>RTC driver</b>	bsp\COMMON_TI_V1\TPS659XX\OALRTC
	bsp\mini8510\SRC\DRIVERS\TPS659XX_RTC

If you need more example of developing WinCE drivers, please select **Start > All Programs > Microsoft Vistual Studio 2005 > MicroSoft Visual Studio Document >**

Content(C) > Windows Embedded CE 6.0 > Develop a Device Driver on your PC's desktop.

## 4.5 System Update

This section will show you how to update WinCE system in SD card and NAND Flash.

### 4.5.1 Updating System in SD Card

- 1) You can download HP USB Disk Storage Format Tool 2.0.6 from <http://www.embedinfo.com/english/download/SP27213.exe>, and use it to format SD card; the figure shown below is the tool's interface;

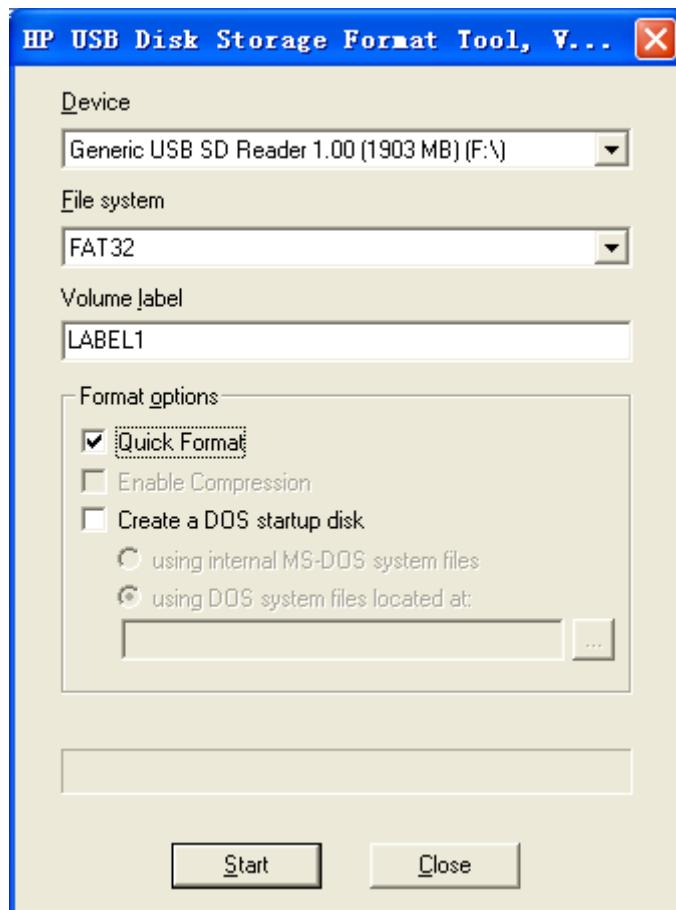


Figure 4-2 Format SD card

Select **FAT32** in the **File system** drop-down menu, and then click **Start** to format SD card.

Copy the files MLO, EBOOTSD.nb0 and NK.bin from X:\WINCE600\image\  
(X is label of your DVD drive) to SD card;

**Notice:**

 HP USB Disk Storage Format Tool will erase the partitions of TF card. If partitions need to be retained, please use the format function of Windows system.

- 2) insert the SD card on SBC8140 and power it on while pressing and holding BOOT button (Button CN11); the information in HyperTerminal window is shown below;

**Table 4-9 Booting Information**

```
60
Texas Instruments Windows CE SD X-Loader for EVM 3730
Built May 29 2012 at 14:43:06
Version BSP_WINCE_ARM_A8 1.01.00.03
open ebootsd.nb0 file
Init HW: controller RST
SDCARD: requested speed 1000000, actual speed 1000000
SDCARD: requested speed 25000000, actual speed 19200000
jumping to ebootsd image

Microsoft Windows CE Bootloader Common Library Version 1.4 Built May 29 2012 14:39:28

Texas Instruments Windows CE EBOOT for OMAP35xx/37xx, Built May 29 2012 at
15:19:04
EBOOT Version 0.0, BSP BSP_WINCE_ARM_A8 1.01.00.03

TI OMAP3730 Version 0x00000012 (ES1.2)
TPS659XX Version 0x30 (ES1.3)
System ready!
Preparing for download...
INFO: Predownload....
Checking bootloader blocks are marked as reserved (Num = 14)
Skip bad block 4
Skip bad block 5
Skip bad block 6
Skip bad block 8
Skip bad block 11
```

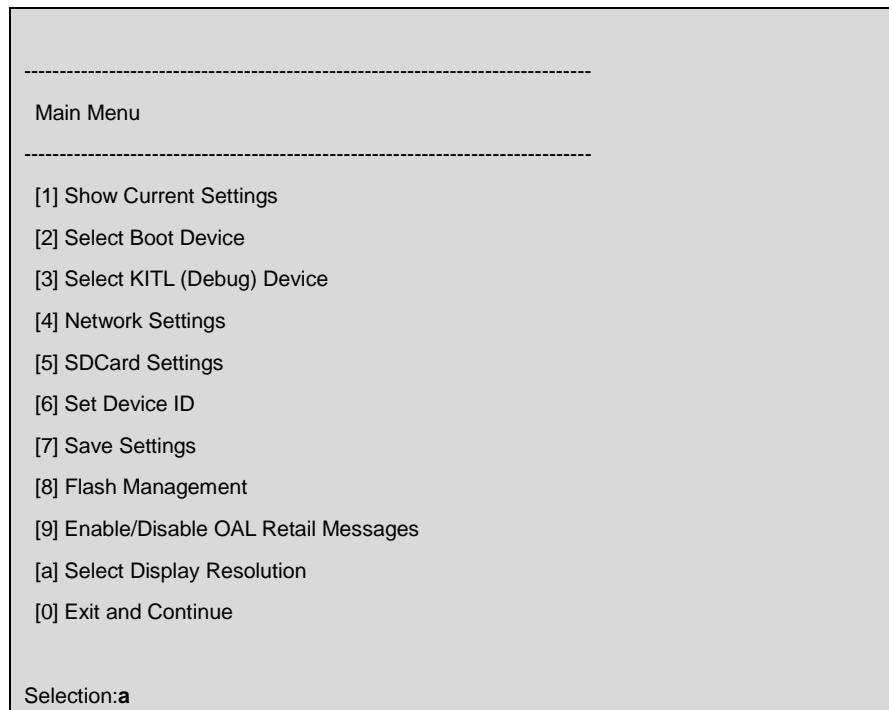
```
WARN: Boot config wasn't found, using defaults
INFO: SW4 boot setting: 0x2f

>>> Forcing cold boot (non-persistent registry and other data will be wiped) <<<
Hit space to enter configuration menu 5... (press Space key to enter eboot menu)
```

When you information counting down in seconds, please press Space key on your PC's keyboard to enter eboot menu.

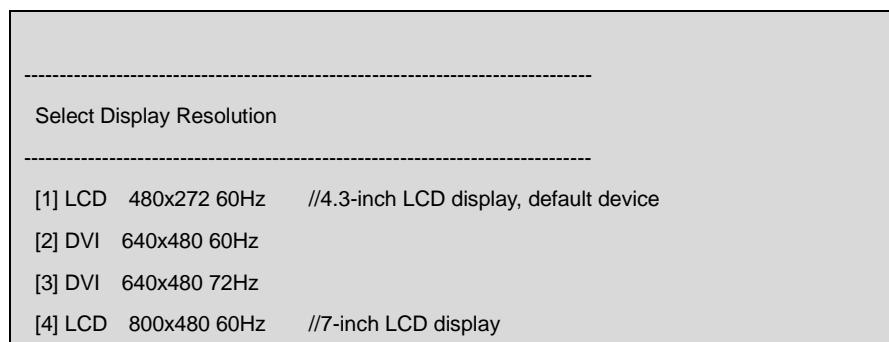
- 3) Type **a** in the following eboot menu;

**Table 4-10 EBOOT Menu**



- 4) Select a proper display mode in the following menu according to your display device;

**Table 4-11 Select Display Mode**



[5] DVI 800x600 60Hz	// LVDS display
[6] DVI 800x600 56Hz	
[7] VGA 1024x768 60Hz	//VGA display
[8] DVI 1280x720 60Hz	
[0] Exit and Continue	

Selection (actual LCD 480x272 60Hz):4

- 5) Type **7** and **y** in the following menu to save settings, and then type number **0** to continue booting system;

**Table 4-12 Continue Booting**

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
[0] Exit and Continue

Selection:**0**

The WinCE system is updated and booted up successfully after booting process is completed.

**Note:**

- By default SBC8140 boots from NAND Flash; pressing and holding BOOT button before connecting power can make it boot from SD card.

#### 4.5.2 Updating System in NAND Flash

- 1) Formatting SD Card;

Please refer to 3.6.1 Updating System in SD Card to find how to format SD card. After SD card formatting is done, copy the files MLO, EBOOTSD.nb0, EBOOTNAND.nb0, NK.bin and XLDRNAND.nb0 from X:WINCE600\image\ (X is label of your DVD drive) to SD card, and then rename EBOOTNAND.nb0 to EBOOTND.nb0.

- 2) Insert SD card on SBC8140, and then power it on while pressing and holding BOOT button; when you see information counting down in seconds, press **Space** key to enter eboot menu;
- 3) Type **8** in eboot meu to enter flash management menu;
- 4) Type **a, b** and **c** to program XLDR, EBOOT and NK image file into flash;
- 5) Type number **0** to go back to main menu, and then type **2** and **4** to select booting from NAND Flash;
- 6) Type **a** in main menu to select display mode, and then type **7** and **y** to save changes;
- 7) Remove SD card from SBC8140 and reboot it; the system will boot up from NAND Flash;

## 4.6 Other Operations

This section will briefly introduce how to run demo programs and use modules on SBC8140.

### 4.6.1 OpenGL ES demo

- 1) Check all the check-box in PowerVR branch in the **Catalog Items View** treeview on the left side of Visual Studio 2005 window as shown below;

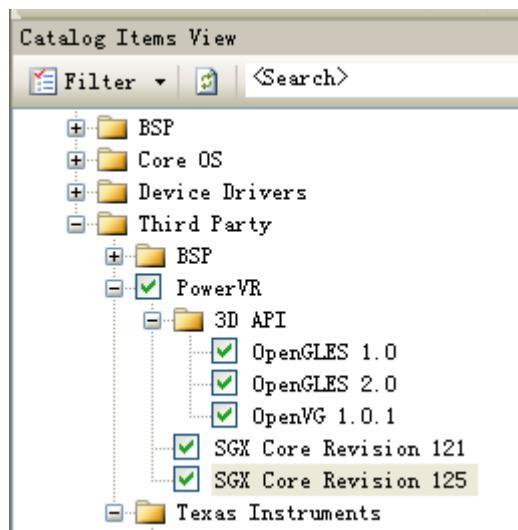


Figure 4-3 Select PowerVR branch

- 2) Select **Build > Build Solution** on the menu bar of Visual Studio 2005 window to generate nk.bin file, and then use it to overwrite the file with the same name in SD card;
- 3) Copy the files under  
**C:\T\wince\_gfx\_sgx\_01\_01\_00\_patch\_01\PowerVR-SDK\OGLES1.1\Binaries\ Demos** or the \*.exe file under  
**C:\WINCE600\PUBLIC\PowerVR\oak\target\Rev125\ARMV4I\retail\** to WinCE system of SBC8140, and then double-click the demo to run it;

#### 4.6.2 CAM8000-A Module

- 1) Modify the line **set BSP\_NODIGITAL\_CAMERA** in file mini8510.bat as shown below;  
**set BSP\_NODIGITAL\_CAMERA=1**
- 2) Check the options under DirectShow branch in **Catalog Items View** treeview on the left side of Visual Studio 2005 window as shown below;

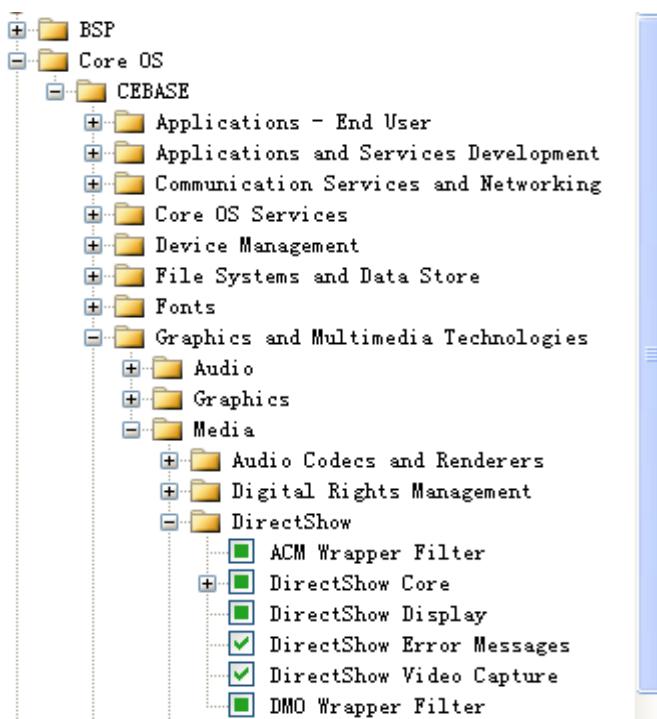


Figure 4-4 DirectShow branch

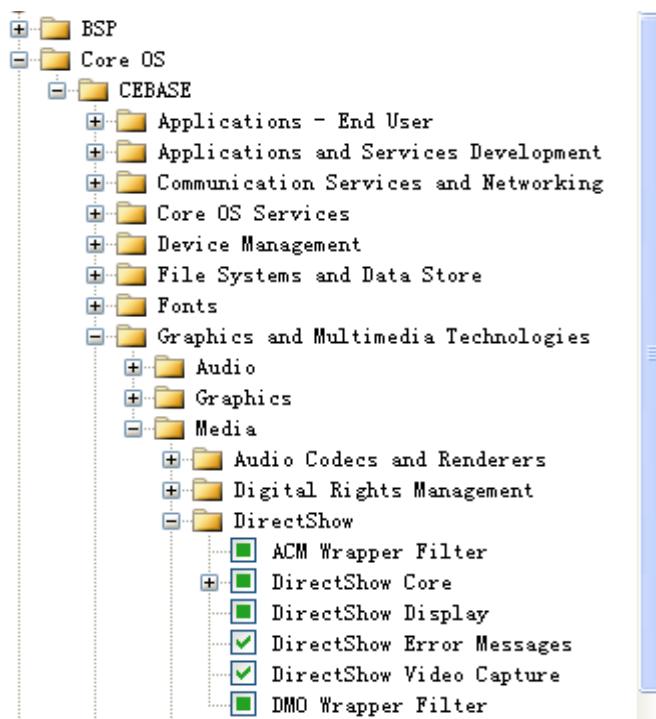
- 3) Select **Third Party > BSP > mini8510:ARMV4I > drivers > camera** in the **Catalog Items View** treeview on the left side of Visual Studio 2005 window, and then select **Build > Rebuild Solution** on the menu bar to start compiling;
- 4) Copy the generated file CameraDshowApp\_analog.exe from **C:\WINCE600\platform\mini8510\files\** to SD card, and then insert it on SBC8140;
- 5) Connect camera module (needs to be purchased separately) to SBC8140 and power it on, and then execute CameraDshowApp\_analog.exe saved in SD card under WinCE system to test CAM8000-A module;

#### 4.6.3 CAM8000-D Module

- 1) Modify the line **set BSP\_NODIGITAL\_CAMERA** in file **mini8510.bat** as shown below;

```
set BSP_NODIGITAL_CAMERA=
```

- 2) Check the options under DirectShow branch in **Catalog Items View** treeview on the left side of Visual Studio 2005 window as shown below;



**Figure 4-5** DirectShow branch

- 3) UNSELECT Third Party > BSP > mini8510:ARMV4I > drivers > camera in the **Catalog Items View** treeview on the left side of Visual Studio 2005 window, and then select **Build > Rebuild Solution** on the menu bar to start compiling;
- 4) Copy the generated file CameraDshowApp\_digital.exe from C:\WINCE600\platform\mini8510\files\ to SD card, and then insert it on SBC8140;
- 5) Connect camera module (needs to be purchased separately) to SBC8140 and power it on, and then execute CameraDshowApp\_digital.exe saved in SD card under WinCE system to test CAM8000-D module;

**Notice:**

- ❑ If **still sink** is selected in Capture Parameters window when running CameraDshowApp\_digital.exe, application memory space should be set as 170000KB or higher by selecting **Control Panel > System** in WinCE system to ensure DirectSHow Graph can run properly.

## 4.7 GPIO API and Example Applications

This section will show you how to develop applications under WinCE environment by giving an introduction to GPIO API and example applications.

The APIs involved when developing applications based on SBC8140 are using definition of Windows Embedded CE 6.0 API and have been expanded only in GPIO interface definition. The applications that control pin status can be found under X:\WINCE600\app\GPIOAppDemo (X is label of your PC's DVD drive).

Please view the relavant documents of MSDN Windows Embedded CE 6.0 API to learn about Windows Embedded CE 6.0 standard API definition.

**Note:**

- ❑ The interfaces derived from some drivers can only be used by drivers, while applications do not have access to the interfaces.

GPIO device name is L"GIO1:" with expanded DeviceIoControl interface definition; the following table lists the corresponding IOCTL code;

Table 4-13 IOCTL Code

IOCTL Code	Descriptions
IOCTL_GPIO_SETBIT	GPIO pin set as 1
IOCTL_GPIO_CLRBIT	GPIO pin cleared
IOCTL_GPIO_GETBIT	Read GPIO pin status
IOCTL_GPIO_SETMODE	Set working mode of GPIO pin
IOCTL_GPIO_GETMODE	Read working mode of GPIO pin
IOCTL_GPIO_GETIRQ	Read corresponding IRQ number of GPIO pin

The tables below contain examples of GPIO applications;

- 1) Enable GPIO device;

**Table 4-14 Open Device**

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

- 2) Set the working mode as reading GPIO;

**Table 4-15 Set as Reading Mode**

```
DWORD id = 0, mode = 0;
```

- 3) Set the working mode of GPIO;

**Table 4-16 Set Working Mode**

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

- 4) Read the working mode of GPIO;

**Table 4-17 Read the Working Mode of GPIO**

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

**Id** is GPIO pin number; **mode** is GPIO mode definition; the following table lists all the mode definitions;

**Table 4-18 GPIO Mode Definitions**

Mode Definitions	Descriptions
GPIO_DIR_OUTPUT	Output mode
GPIO_DIR_INPUT	Output mode
GPIO_INT_LOW_HIGH	Triggered on rising edge
GPIO_INT_HIGH_LOW	Triggered on falling edge
GPIO_INT_LOW	Triggered by low level
GPIO_INT_HIGH	Triggered by high level
GPIO_DEBOUNCE_ENABLE	Debounce enabled

- 5) Operations on GPIO pins

**Table 4-19 Operations on Pins**

```
DWORD id = 0, pin = 0;
```

**Table 4-20 High Level Output**

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

**Table 4-21 Low Level Output**

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

**Table 4-22 Read Status of Pins**

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

**Id** is GPIO pin number; **pin** returns pin status.

- 6) Read corresponding IRQ number of GPIO pins;

**Table 4-23 Read IRQ Number**

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

**Id** is GPIO pin number; **irq** returns IRQ number;

- 7) Disable GPIO device;

**Table 4-24 Disable Device**

```
CloseHandle(hFile);
```

**Note:**

- ❑ GPIO pin definitions: 0~191 MPU Bank1~6 GPIO pin, 192~209 TPS65930 GPIO 0~17.
- ❑ GPIO pins 0~191 must be defined as GPIO in both xldr/platform.c and oalib/oem\_pinmux.c.
- ❑ GPIO interrupt mode is used by drivers, but not applications.

# Appendix 1 – Installing Ubuntu Linux System

---

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

## Installing VirtualBox

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

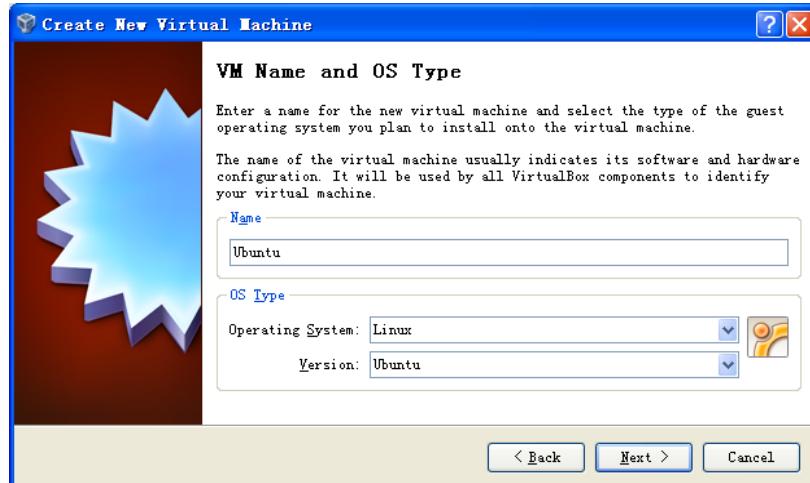
- 1) The installation process is simple and will not be introduced. Please start VirtualBox from the **Start** menu of Windows, and then click **New** in VirtualBox window. A pop-up window **Create New Virtual Machine** will be shown as below;



**Figure 1** Create new virtual machine

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

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



**Figure 2** Name and OS type of virtual machine

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

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

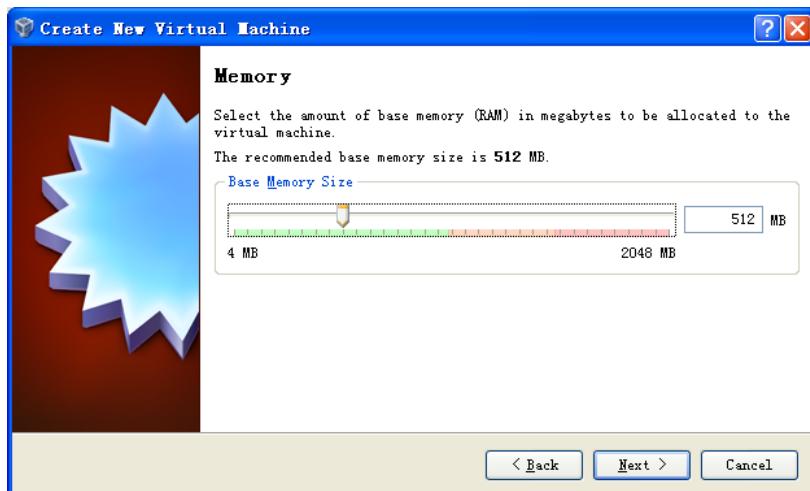


Figure 3 Memory allocation

**Note:**

- ❑ If the memory of your PC is only 1GB or lower, please keep the default setting;
- ❑ If the memory of your PC is higher than 1GB, you can allocate 1/4 or fewer to virtual machine, for example, 512MB out of 2GB memory could be allocated to virtual machine.

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

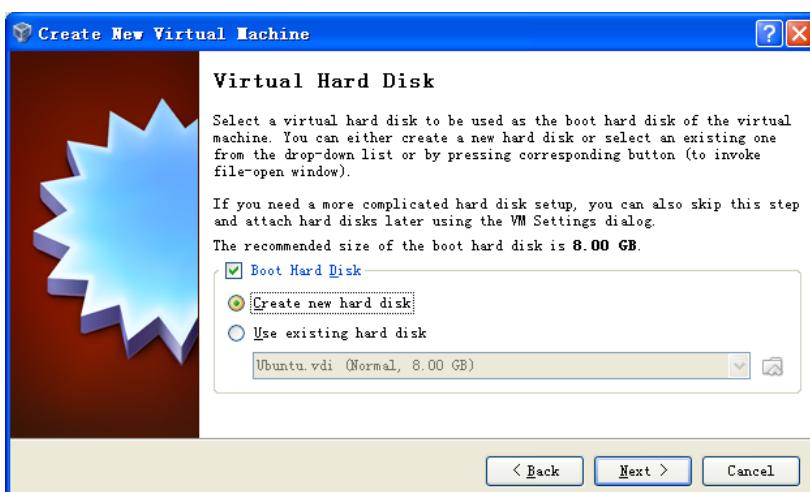


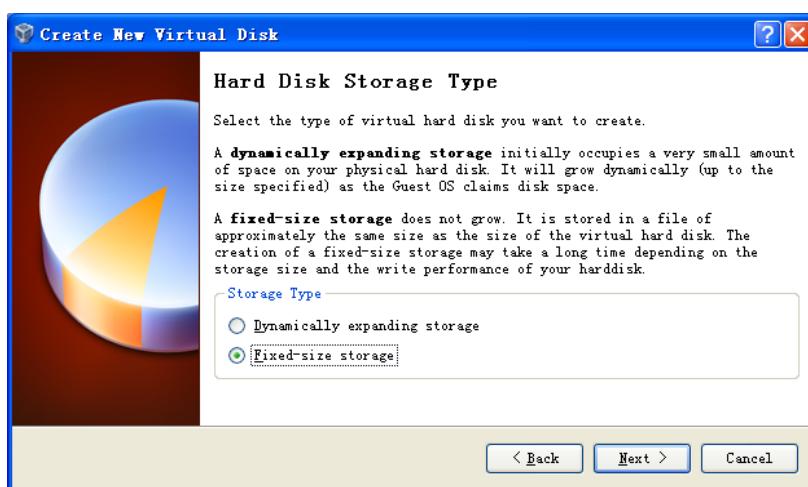
Figure 4 Create new hard disk

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



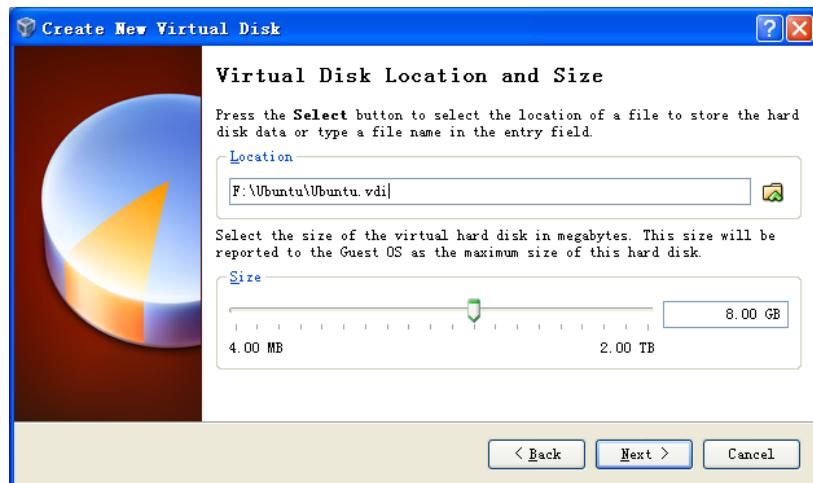
**Figure 5** Wizard of new virtual disk creation

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



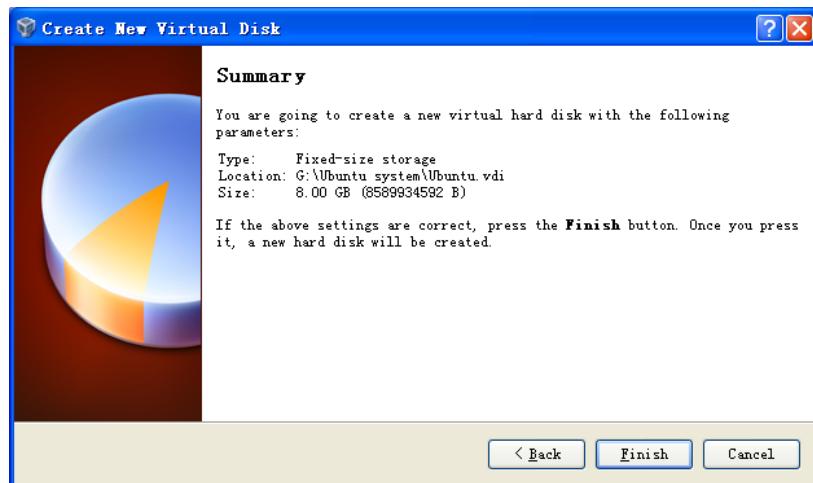
**Figure 6** Select the second option

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



**Figure 7** Virtual disk configuration

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



**Figure 8** Virtual disk summary

- 9) PC is creating a new virtual disk;

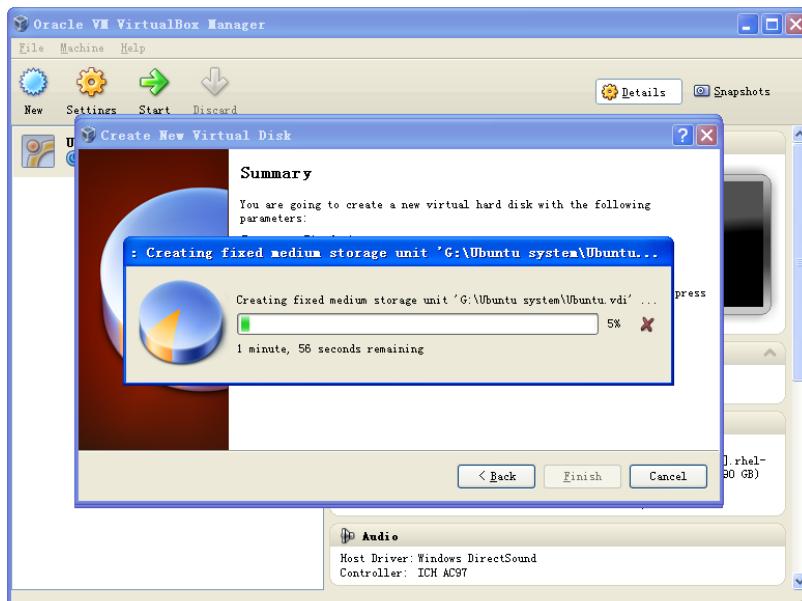


Figure 9 Virtual disk creation in process

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



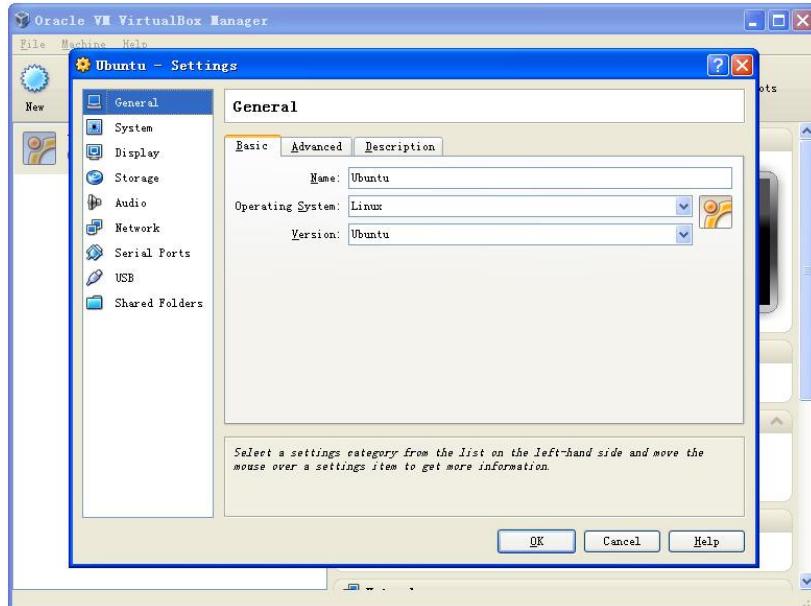
Figure 10 Virtual machine is ready

## 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

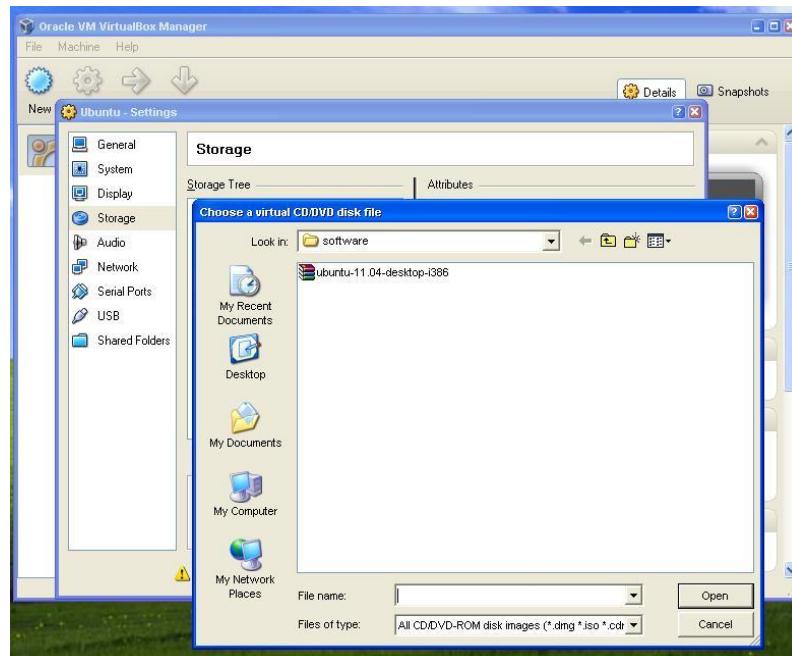
image file of Ubuntu, and then follow the steps.

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



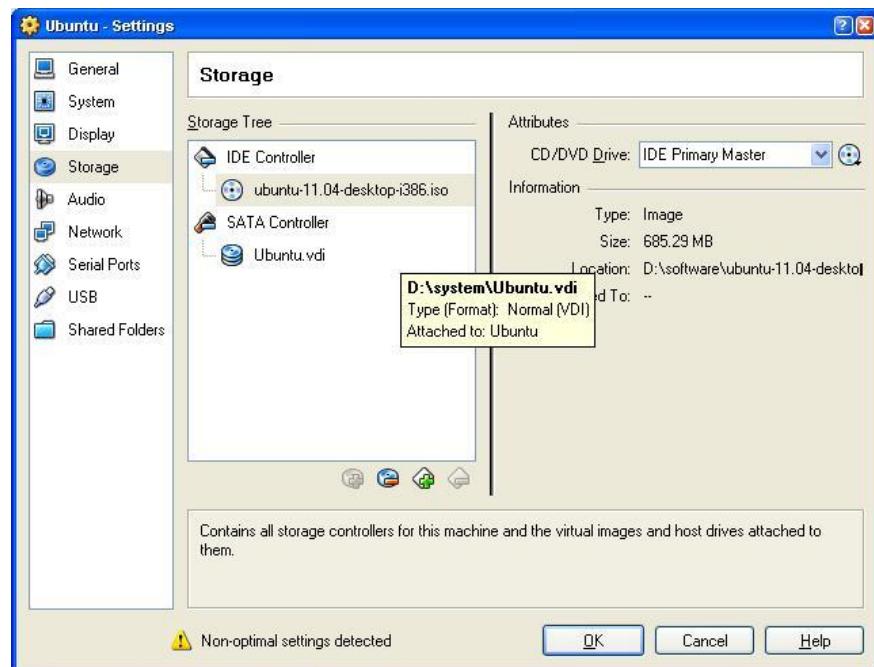
**Figure 11** Setting window

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



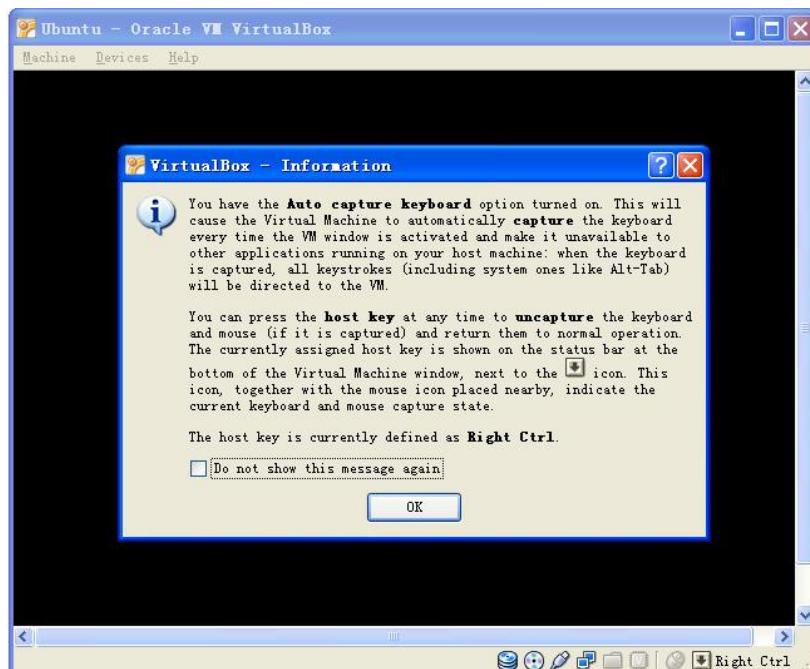
**Figure 12** Find ISO file

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



**Figure 13** Select ISO file

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



**Figure 14** Ubuntu initiating window

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

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

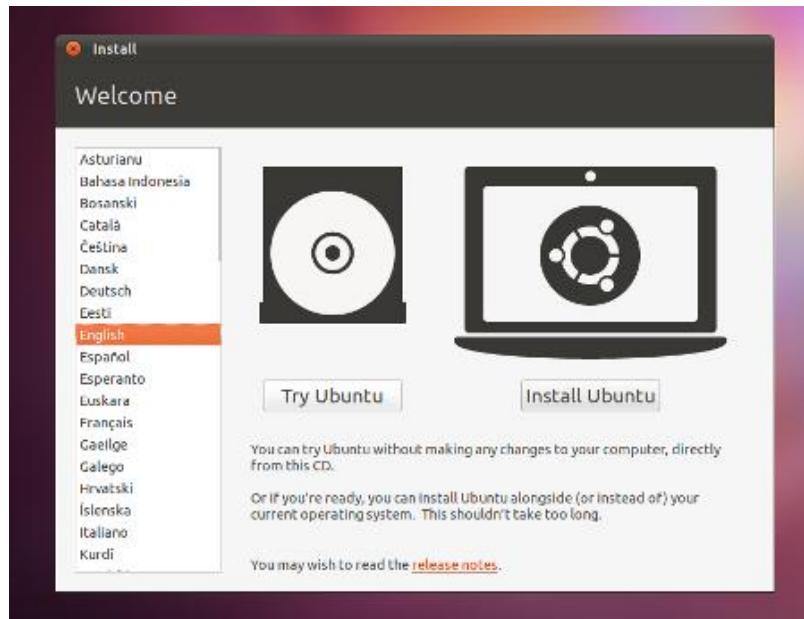


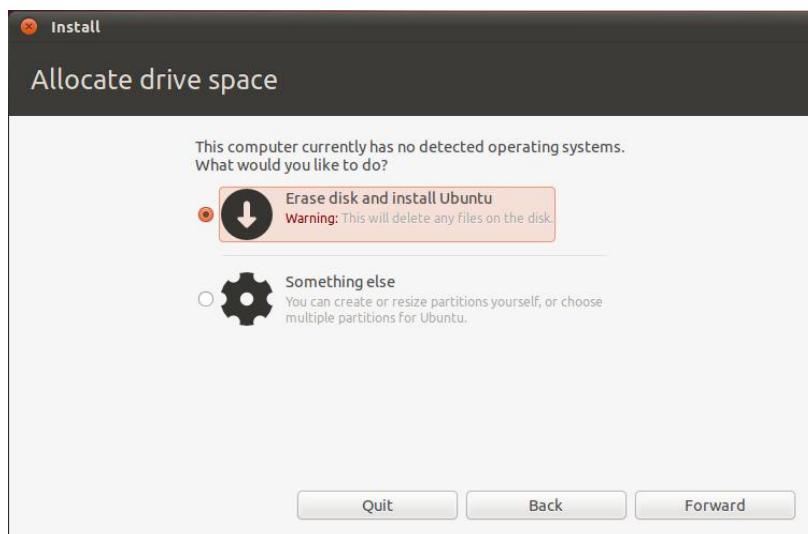
Figure 15 Ubuntu installation window

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



Figure 16 Information before installation

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

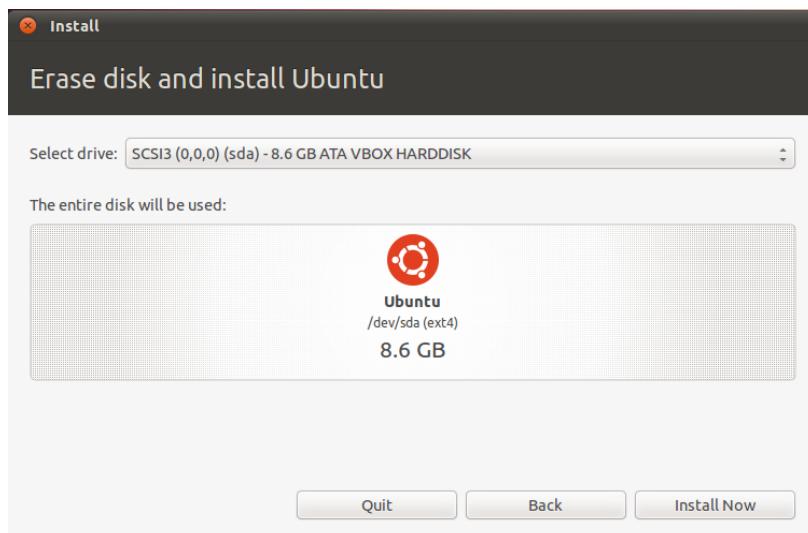


**Figure 17** Options before installation

**Note:**

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

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



**Figure 18** Confirm installation

- 9) Some simple questions need to be answered during the installation process.

Please enter appropriate information and click **Forward**. The following window is the last question that will appear during the process;

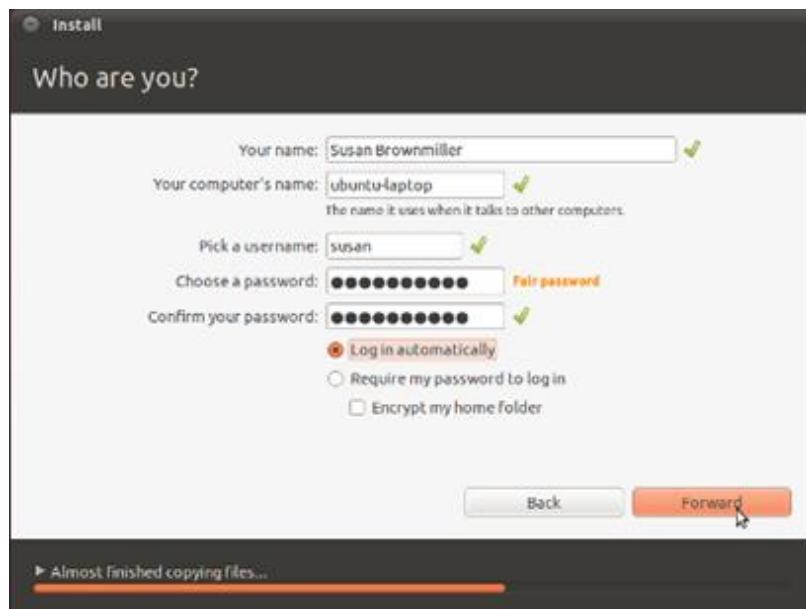


Figure 19 Enter appropriate information

After all the required information is properly entered in to the fields, select

**Log in automatically** and click **Forward**.

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



Figure 20 Restart Ubuntu

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

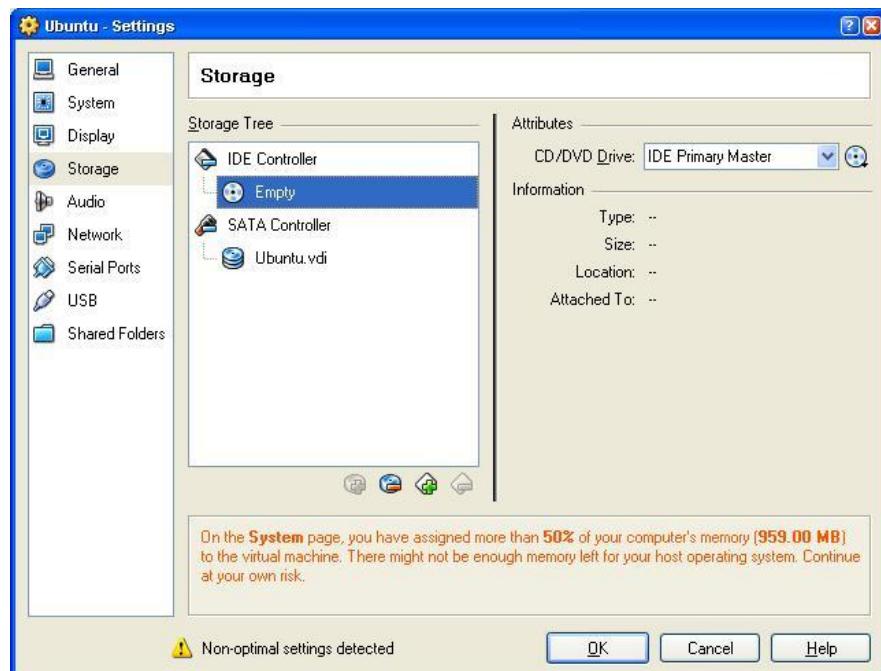


Figure 21 ISO file ejected

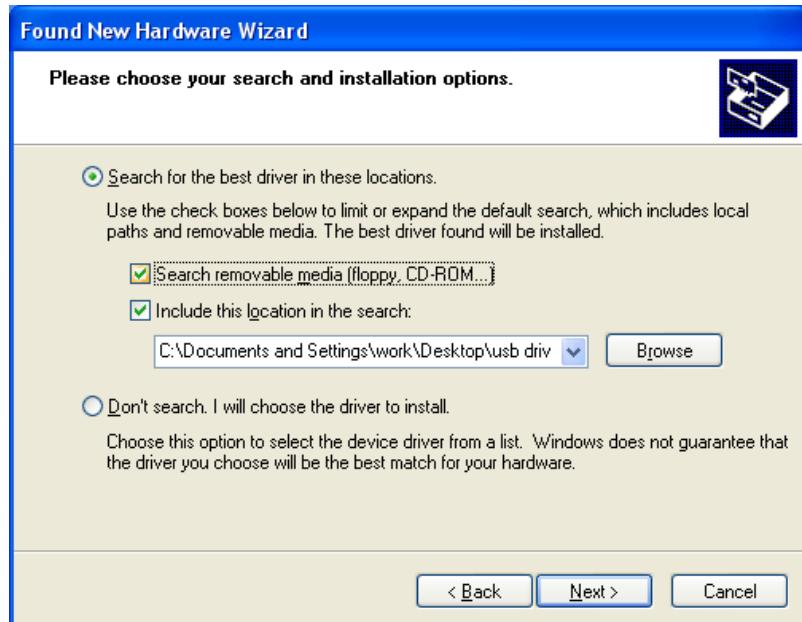
## Appendix 2 – Installing Linux USB Ethernet/RNDIS Gadget Driver

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



**Figure 1** Found new hardware

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



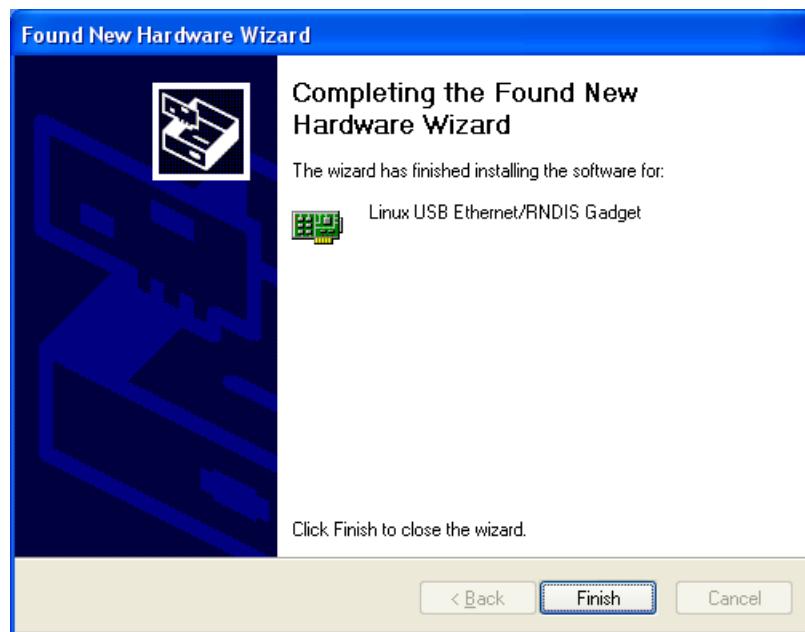
**Figure 2** Path of driver

- 3) Click **Continue Anyway** in the following window;



**Figure 3** Click Continue Anyway

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



**Figure 4** Click Finish

## Appendix 3 – Making Linux Boot Disk

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

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

```
$ dmesg | tail
```

**Table 1 Device Information**

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

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

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

```
$ df -h
```

**Table 2 Device Path**

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

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

**Note:**

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

- 3) Execute the following instruction to unmount the device;

**\$ umount /media/disk**

- 4) Execute **fdisk** instruction;

**\$ sudo fdisk /dev/sdc**

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

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

**Table 3 Partition Records**

Command (m for help): [ <b>p</b> ]														
Disk /dev/sdc: 2021 MB, 2021654528 bytes														
255 heads, 63 sectors/track, 245 cylinders														
Units = cylinders of 16065 * 512 = 8225280 bytes														
<table border="1" style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Device</th> <th style="text-align: left; padding: 2px;">Boot</th> <th style="text-align: left; padding: 2px;">Start</th> <th style="text-align: left; padding: 2px;">End</th> <th style="text-align: left; padding: 2px;">Blocks</th> <th style="text-align: left; padding: 2px;">Id</th> <th style="text-align: left; padding: 2px;">System</th> </tr> </thead> <tbody> <tr> <td style="text-align: left; padding: 2px;">/dev/sdc1</td> <td style="text-align: left; padding: 2px;">*</td> <td style="text-align: left; padding: 2px;">1</td> <td style="text-align: left; padding: 2px;">246</td> <td style="text-align: left; padding: 2px;">1974240+</td> <td style="text-align: left; padding: 2px;">c</td> <td style="text-align: left; padding: 2px;">W95 FAT32 (LBA)</td> </tr> </tbody> </table>	Device	Boot	Start	End	Blocks	Id	System	/dev/sdc1	*	1	246	1974240+	c	W95 FAT32 (LBA)
Device	Boot	Start	End	Blocks	Id	System								
/dev/sdc1	*	1	246	1974240+	c	W95 FAT32 (LBA)								
Partition 1 has different physical/logical endings:														
phys=(244, 254, 63) logical=(245, 200, 19)														

Write down the total bytes shown in the above information, for example

2021654528 bytes, and then type **d** to delete all the partitions;

- 6) If you do not find information **255 heads** and **63 sectors/track** in the above table, please go through the following steps to recover TF card;

- A. Type letters as shown in the following table to set Heads and Sectors;

**Table 4 Set Heads and Sectors**

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

- B. Use the following equation to calculate the number of Cylinders;

Cylinders = the total bytes written down previously  $\div 255 \div 63 \div 512$

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

**Table 5 Set Cylinders**

Expert Command (m for help): [ <b>c</b> ] (type c to set cylinders)
Number of cylinders (1-256, default xxx): (enter the number of cylinders calculated above)...

Expert Command (m for help): [ r ] (type r to go back to normal mode)

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

**Table 6 Check Parameters**

Device	Boot	Start	End	Blocks	Id	System

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

**Table 7 Create FAT32 Boot Partition**

Command (m for help): [ n ] (type n to start creating partition)
Command action
e extended
p primary partition (1-4)
[ p ] (type p to create primary partition)
Partition number (1-4): [ 1 ] (set the partition number to 1)
First cylinder (1-245, default 1): [ ] (press Enter key on your keyboard)
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-61, default 61): [ +5 ] (enter +5)
Command (m for help): [ t ] (type t)
Selected partition 1
Hex code (type L to list codes): [ c ] (type c to set partition type)
Changed system type of partition 1 to c (W95 FAT32 (LBA))

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

**Table 8 Set Bootable Mode**

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

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

**Table 9 Create Root Filesystem Partition**

Command (m for help): [ n ] (type n to create a partition)
Command action
e extended
p primary partition (1-4)
[ p ] (type p to select primary partition)

Partition number (1-4): [ 2 ] (set partition number to 2)  
 First cylinder (7-61, default 7): [ ] (press Enter key on your keyboard)  
 Using default value 52  
 Last cylinder or +size or +sizeM or +sizeK (7-61, default 61): [ ] (press Enter key)  
 Using default value 245

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

**Table 10 Check Partitions**

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

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

**Table 11 Save Partition Records**

Command (m for help): [ w ]  
 The partition table has been altered!  
 Calling ioctl() to re-read partition table.  
 WARNING: Re-reading the partition table failed with error 16: Device or resource busy.  
 The kernel still uses the old table.  
 The new table will be used at the next reboot.  
 WARNING: If you have created or modified any DOS 6.x  
 partitions, please see the fdisk manual page for additional  
 information.  
 Syncing disks.

- 13)** Execute the following instructions to formation new partitions;

**\$ [sudo mkfs.msdos -F 32 /dev/sdc1 -n LABEL1]**

**\$ [sudo mkfs.ext3 -L LABEL2 /dev/sdc2]**

**Note:**

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

## Appendix 4 – Building TFTP Server

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

- 1) Execute the following instructions to install a client;

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

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

- 2) Execute the following instructions to install inetd;

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

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

- 3) Execute the following instructions to configure the server;

- A. Create a tftpboot under root directory and change the properties to readable/writable by users;

```
$>cd /
```

```
$>sudo mkdir tftpboot
```

```
$>sudo chmod 777 tftpboot
```

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

```
$>sudo vi /etc/inetd.conf
```

- C. Reload inetd process;

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

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

```
$>cd /etc/xinetd.d/
```

```
$>sudo touch tftp
```

```
$>sudo vi tftp
```

Table 12 Add Codes

```
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
}
```

- 5) Execute the following instructions to restart the service;

**\$>sudo /etc/init.d/xinetd restart**

**\$>sudo in.tftpd -I /tftpboot**

- 6) Test TFTP server by the following operations;

- A. Create a file under /tftpboot/;

**\$>touch abc**

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

**\$>tftp 192.168.1.15**

**\$>tftp> get abc**

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

## Appendix 5 – FAQ

---

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

# Technical Support and Warranty

## Technical Support



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

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



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

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

## Warranty Conditions

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

- the sales of the product;
- 2) The following conditions are not covered by free services; Embest Technology will charge accordingly:
- 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:**

- 书 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@timll.com](mailto:sales@timll.com)

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

**Website:** <http://www.timll.com>

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