DevKit8600

Integrated with LCD, USB Host/Device, Audio in/out, CAN, RS485, ADC, JTAG, GPMC, 10/100M Ethernet, Serial, TF card interface based on TI 32-bit microprocessor AM3359

User Manual

mbest

COPYRIGHT

Important Notice

Version updates records:

Rev	Date	Description
1.0	2012-6-1	Initial version
1.1	2012-11-7	Android source code is supported now.

Contact:

If you want to order products from Embest, please contact Marketing Department:

Tel: +86-755-25500944 / 25631357 / 25635656

Fax: +86-755-25616057

E-mail: market@embedinfo.com

If you want to get technical assistance from Embest, please contact Technical Assistance Department:

Tel: +86-755-25503401

E-mail: support@embedinfo.com

URL: http://www.armkits.com

Address: Room 509, Luohu Science&Technology Building, #85 Taining Road, Shenzhen, Guangdong, China (518020)

Content

Chapter 1 Overview

1.1 Product Introduction

DevKit8600 evaluation board is a compact, low-cost with high-performance evaluation board based on Texas instruments (TI) AM3359 processor. Instruments AM3359 microprocessor is an integration of 720MHz ARM Cortex-A8 low-power application processor with 176K-Byte On-chip boot ROM, and provided lots of peripheral interface. DevKit8600 board expands the hardware capabilities including LAN port, audio input/output interface, USB OTG, USB HOST, CAN interface, RS485 interface, SPI interface, IIC interface, ADC interface, GPMC interface, JTAG interface, TF slot, serial port, TFT LCD interface, touch screen interface and keyboard interface.

DevKit8600 board can be used for the following applications:

- Gaming Peripherals
- Home and Industrial Automation
- Consumer Medical Appliances
- Printers
- Smart Toll Systems
- Connected Vending Machines
- Weighing Scales
- Educational Consoles
- Advanced Toys

1.2 Features

DevKit8600 evaluation board is based on AM3359 processor which integrates all functions and features of TI's AM3359 ARM Cortex-A8 processor. Some of the board features are mentioned below:

Mechanical Parameters

- Working temperature: 0°C ~ 70°C
- Humidity Range: 20% ~ 90%
- Dimensions: 130mm x 86mm
- Power Consumption：12V@0.19A (Boot from Linux Without peripherals)

Processor

- 720-MHz ARM Cortex™-A8 32-Bit RISC Microprocessor
  - NEON™ SIMD Coprocessor
  - 32KB/32KB of L1 Instruction/Data Cache with Single-Error Detection (parity)
  - 256KB of L2 Cache with Error Correcting Code (ECC)
- SGX530 Graphics Engine
- Programmable Real-Time Unit Subsystem

Memory and Storage

- 512MByte NAND Flash
- 512MByte DDR3 SDRAM

Audio/Video Interfaces

- LCD/Touch Screen interface (Up to 16-Bits Data Output; 8-Bits per Pixel (RGB), 50-pin FPC connector J2)
- One audio input interface (3.5mm audio jack)
- One 2-channel audio output interface (3.5mm audio jack)

Data Transfer Interfaces

- 10/100M Ethernet interface (RJ45 connector)
- One CAN 2.0 interface and RS485 interface(8 Pin Phoenix Connector)
- One USB 2.0 High-Speed OTG Ports with Integrated PHY (Mini USB type interface)
- One USB 2.0 High-Speed HOST Ports with Integrated PHY (USB A type interface)
- Wi-Fi/Bluetooth interface(only Linux supports)

- One 5 line Debug serial port, RS232 (DB9 connector)
- Expansion interface
  - UART1, UART3 interface (J5 connector, UART1 Compatible with RS485 and UART3 compatible with wifi/Bluetooth)
  - I2C0, UART2 interface (J6 connector)
  - SPI0 interface (J8 connector)
  - Four ADC interfaces (J10 connector)
- One GPMC bus interface(J14 connector)

Input Interfaces

- Three user buttons (HOME, MENU, BACK)
- One reset button

LED

- One power indicator
- Two user LEDs

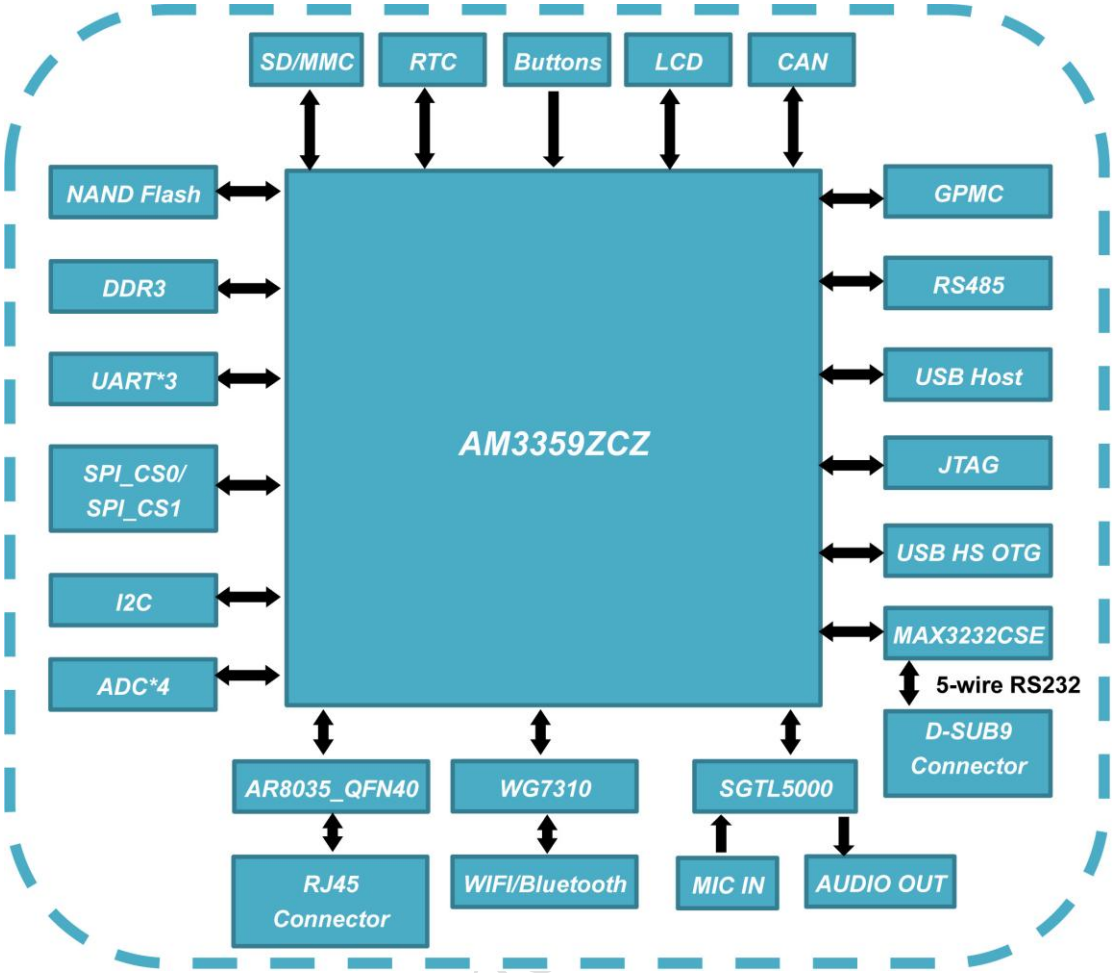


Figure 1-1

1.3 DevKit8600 Optional Modules List

Modules	Linux	Android	WinCE	Relevant Materials
VGA8000	YES*	YES*	YES*	Available in the CD-ROM of Devkit8600
WF8000-U	YES*	NO	NO	Available in a dedicated CD-ROM
CAM8100-U	YES*	NO	NO	Available in a dedicated CD-ROM
CDMA8000-U	YES*	NO	NO	Download here
WCDMA8000-U	YES*	NO	NO	Download here
LVDS8000	YES*	YES*	YES*	Available in the CD-ROM of Devkit8600 and on our Website

Table 1-1

1.4 How to Quick Start

This section will tell the user how to understand and use DevKit8600 better and faster through this DevKit8600 Quick Operation Manual. For more information please refer to the listed document and location.

For hardware development:

Hardware system	Introduce CPU, expanded chip and hardware interface	User Manual->2 Hardware System
CPU Datasheet	Know principle and configuration of AM335x	CD->\HW design\datasheet\CPU\
Schematic diagram of DevKit8600	Know hardware principle of DevKit8600	CD->\HW design\schematic
Dimensional drawing of	Refer to the actual length and height of DevKit8600 to bring	User Manual->Appendix-> Appendix I

DevKit8600	convenience for opening die	

Table 1-2

For software development:

Establish testing environment	To connect with external hardware devices, set serial port terminals and boots the system	User Manual ->1.4.1 Establishment of hardware environment 1.4.2 Preparation of Windows XP system environment
Test functionality of interface	Test the interface of the board carrier through the operating system	User Manual ->3.8.1 Various Tests Scenario
DEMO demonstration	Establish a DEMO demonstration system (Android, TISDK)	User Manual->3.8.2.1 Demonstration of Android System 3.8.2.2 Demonstration of TISDK System
Establish developing and compilation environment	Linux developing and compilation environment	User Manual -> 3.4.1 Establishing operating system development environment
	Windows Embedded Compact 7 developing and compilation environment	User Manual->4.4.1 Installation of Compilation Tool 4.4.2 Extract BSP and project files to IDE
Recompile system image	Recompile Linux system image	User Manual->3.4.2 System Compilation
	Recompile Windows Embedded Compact 7 system image	User Manual->4.4.3 Sysgen & Build BSP
Software development	Refer to introduction of Linux driver and related driver development process	User Manual->3.5 Introduction of driver 3.6 Driver Development
	Refer to introduction of Windows Embedded Compact 7 driver and related driver development process	User Manual -> 4.4.4 Driver Introduction
Starterware	Introduction of how to develop and run bare-metal programs	CD-ROM -> \Starterware\doc\DevKit8600 Starterware User Manualt.pdf

Table 1-3

For marketing:

Hardware system	CPU feature, board carrier interface data	User Manual->2 Hardware System
About Linux /	Know basic Linux software	User Manual->3.2 Software

Windows Embedded Compact 7 software	components and features, and purpose of compilation tool	Resources 3.3 Software Features
	Know basic Windows Embedded Compact 7 software components and features, and purpose of compilation tool	User Manual->4.2 Software Resources 4.3 Software Features
Dimensional drawing of DevKit8600	Refer to the actual length and height of DevKit8600 to bring convenience for opening die	User Manual->Appendix->Appendix I
DEMO demonstration	Establish a DEMO demonstration system (Android, TISDK)	User Manual->3.8.2.1 Android System Demonstration 3.8.2.2 TISDK System Demonstration

Table 1-4

For learning personnel:

It is suggested to browse each section in each chapter of this Manual in order.

1.4.1 Establishment of hardware environment

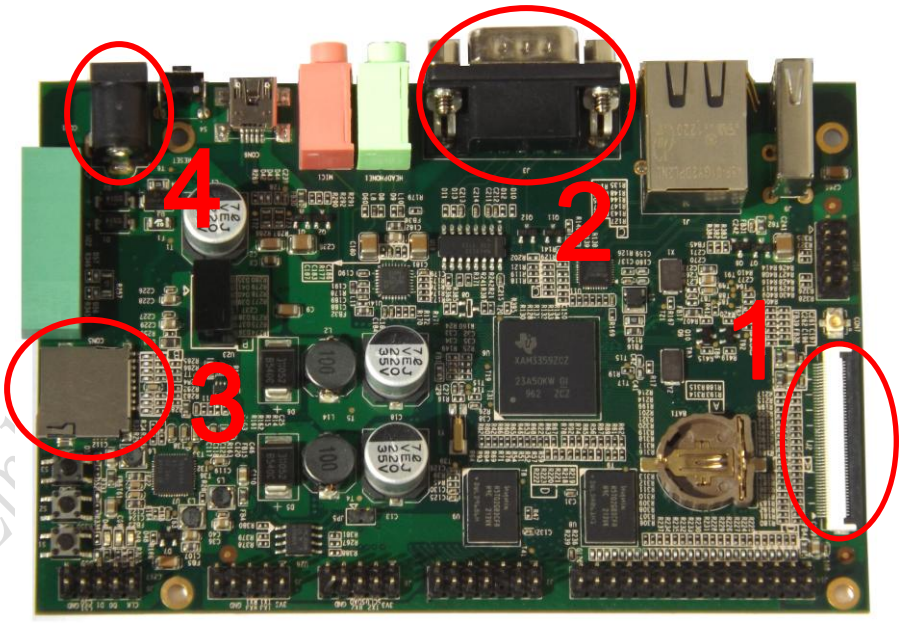


Figure 1-2

Please establish the hardware environment according to the following steps:

1) Connect TFT-LCD

Connect your 4.3-inch/7-inch TFT-LCD to the TFT-LCD interface.

2) Connect serial port for communication

Use serial cable to connect the DevKit8600 debugger serial port and PC serial port

3) Insert TF card

If you want to boot and operate Windows Embedded Compact 7 operating system, please insert the TF card into the TF card slot of DevKit8600, otherwise the system will automatically load the Linux operating system from the NAND Flash

4) Connect the 12V power adapter to the evaluation board

1.4.2 Preparation of Windows XP system environment

Before DevKit8600 boot-up, you need to establish a HyperTerminal on PC; follow the below process in order to setup Hyper Terminal connection:

1) Windows XP -> Start -> All Programs -> Accessories -> Communication -> Hyper Terminal. Find the HyperTerminal, as shown below:

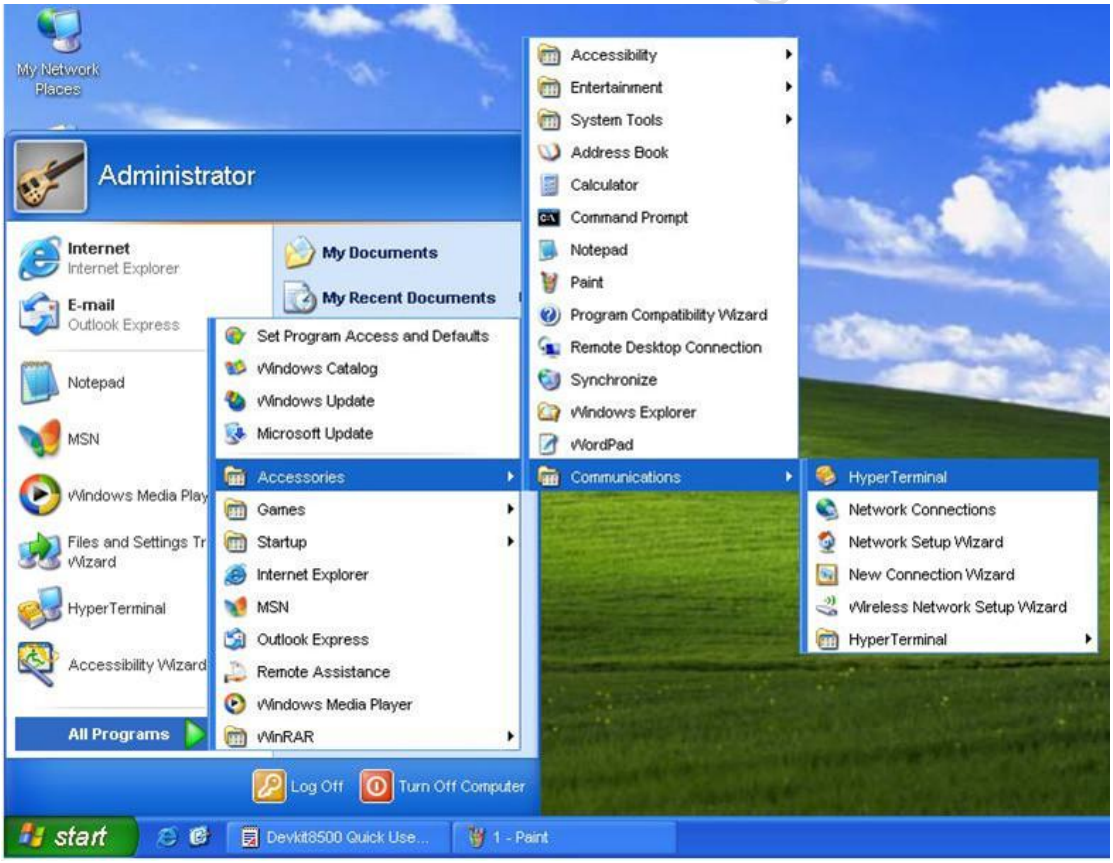


Figure 1-3

2) Establish HyperTerminal connection, Enter a name and select the icon for the connection:

Figure 1-4

3) Select the specific serial port from the list as per your computer COM port configuration:

Figure 1-5

4) Set parameters for serial port connection as follows:

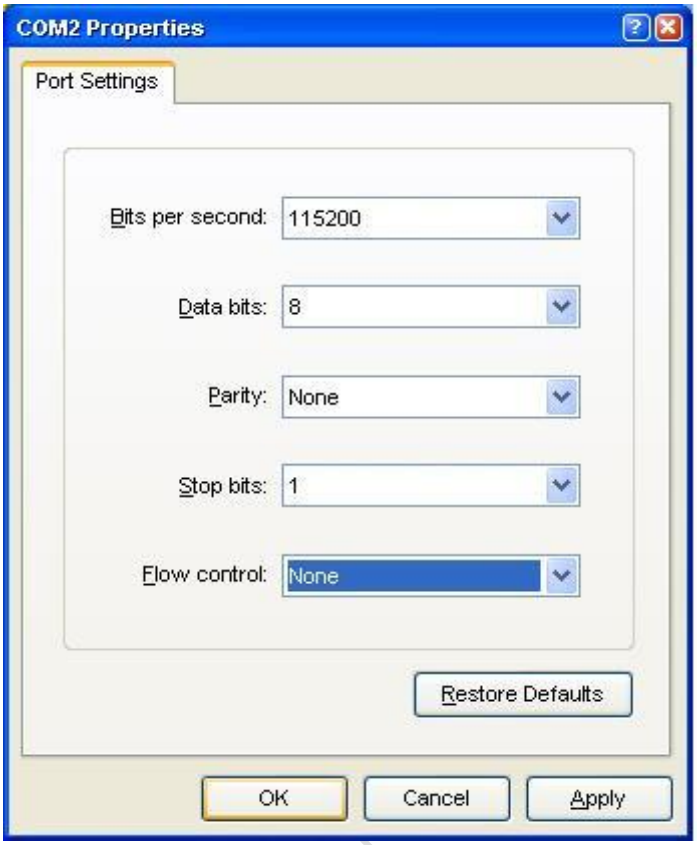


Figure 1-6

5) So we have successfully established a Hyper Terminal connection with PC serial port:

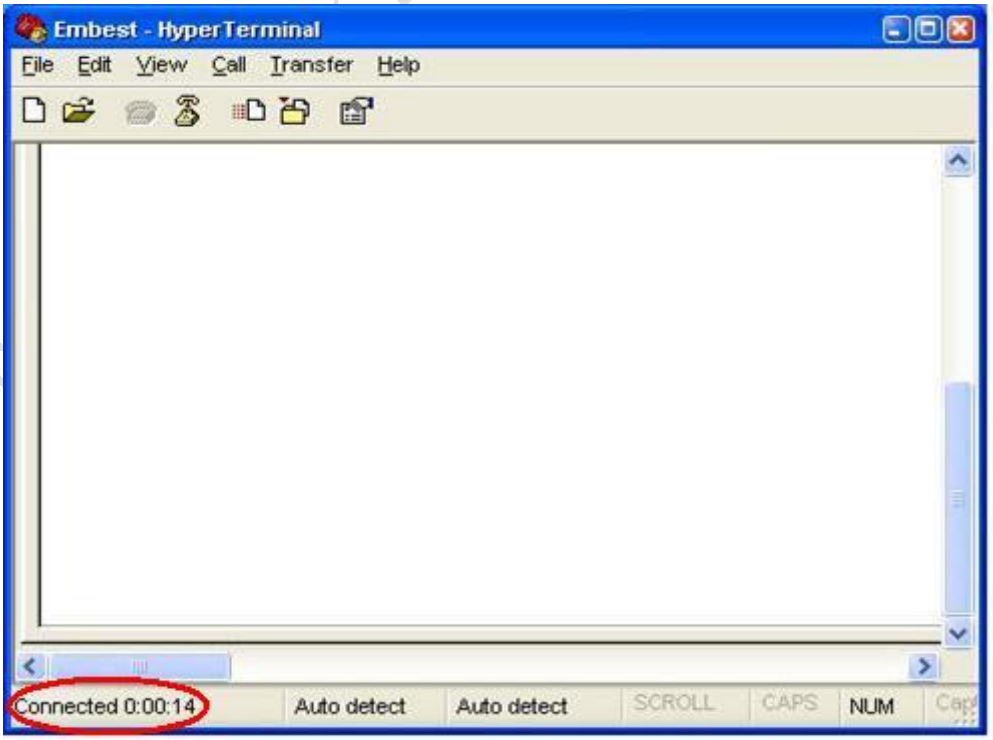


Figure 1-7

Chapter 2 Hardware System

2.1 CPU

2.1.1 CPU Introduction

The AM3359 microprocessors based on the ARM Cortex-A8 are enhanced with image, graphics processing, peripherals and industrial interface options such as etherCAT and Profibus. The device supports the following high-level operating systems (OSs), that are available free of charge from TI: Linux, Windows CE and Android.

The AM3359 microprocessor contains these subsystems: Microprocessor unit (MPU) subsystem based on the ARM Cortex-A8 microprocessor. POWERVR SGX Graphics Accelerator subsystem for 3D graphics acceleration to support display and gaming effects. Programmable Real-Time Unit Subsystem (PRUSS) enables the user to create a variety of digital resources beyond native peripherals of the device. In addition, the PRUSS is separate from the ARM core. This allows independent operation and clocking to give the device greater flexibility in complex system solutions.

Note: The subsystem available on this device is the next-generation PRUSS (PRUSSv2).

2.1.2 CPU Features

Clock

The AM3359 device has two clock inputs: OSC1 and OCC0, two clock output signals: CLKOUT1 and CLKOUT2.

The OSC1 oscillator provides a 32.768-kHz reference clock to the real-time clock (RTC) and is connected to the RTC_XTALIN and RTC_XTALOUT terminals.

The OSC0 oscillator provides a 19.2-MHz, 24-MHz, 25-MHz, or 26-MHz reference clock which is used to clock all non-RTC functions and is connected to the XTALIN and XTALOUT terminals.

Reset

The function of reset is decided by the PWRONRSTn signal on the CPU, Reset is enabled when LOW level signal (high to low) is given.

General-Purpose Interface

The general-purpose interface combines four general-purpose input/output (GPIO) banks. Each GPIO bank provides 32 dedicated general-purpose pins with input and output capabilities; thus, it supports up to 128 (4 x 32) general-purpose interface pins.

Programmable Real-Time Unit Subsystem

AM3359 Programmable Real-Time Unit Subsystem (PRUSS) includes two Programmable Real-Time Units (PRUs), 12 KB of shared RAM with Single-Error Detection (parity), three 120-byte Register Banks Accessible by Each PRU, Interrupt Controller Module (INTC) for handing system input events and the following Peripherals inside the PRUSS:

- One UART Port with Flow Control Pins, Supports Up to 12 Mbps
- Two MII Ethernet Ports that Support Industrial Ethernet, such as EtherCAT™
- One MDIO Port
- One Enhanced Capture (eCAP) Module

3D Graphics Engine

POWERVR® SGX Graphics Accelerator subsystem for 3D graphics acceleration to support display and gaming effects. The key feathers supported by the subsystem are:

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

2.2 Description of different IC blocks

2.2.1 TPS65910

The TPS65910 is an integrated power-management IC which provides three step-down converters, one step-up converter, and eight LDOs. TPS65910 is communicated with CPU through I2C protocol, its main role is to onboard chip as the CPU, NAND Flash, DDR voltage of 1.1V, 1.2V, 1.5V, 1.8V or 3.3V, to make it work properly.

About the more information please refer to the DISK-Devkit8600\HW design\datasheet\Power\tps65910.pdf

2.2.2 NAND Flash H27U4G8F2DTR-BC

NAND Flash is using H27U4G8F2DTR-BC, size for 512MB, 1 pcs has included in Devkit8600.

About the more information please refer to the DISK-Devkit8600\HW design\datasheet\NAND Flash\H27(U_S)4G8_6F2D_rev1.5.pdf

2.2.3 DDR H5TQ2G83CFR-H9C

DDR3 SDRAM is using H5TQ2G83CFR-H9C, size for 256MB, 2 pcs has included in Devkit8600.

About the more information please refer to the DISK-Devkit8600\HW design\datasheet\DDR\H5TQ2G4(8)3CFR(Rev0[1].2).pdf

2.2.4 Ethernet AR8035

The AR8035 is a low power, low BOM(Bill of Materials) cost Ethernet chip for Devkit8600. It integrated 10/100/1000 Gigabit Transceiver. It is Single port 10/100/1000 Mbps Tri-speed Ethernet PHY, and supports RGMII interface to the MAC.™.

The AR8035 supports IEEE 802.3az Energy Efficient Ethernet (EEE) standard and Atheros proprietary SmartEEE, which allows legacy MAC/SoC devices without 802.3az support to function as the complete 802.3az system.

DevKit8600 can be connected to network hub through a direct cable, also can be directly

connected with a computer through a crossover cable.

About the more information please refer to the DISK-Devkit8600\HW design\datasheet\LAN\AR8035.pdf

2.2.5 MAX3232

The function of MAX3232 is mainly to translate TTL logic level signal into RS232 logic level, which helps in communicating the board with PC.

DevKit8600 uses UART0 as debugging serial port; as the default voltage of UART0 is 1.8V, it is necessary to convert this voltage to 3.3V in order to connect to eternal world.

About the more information please refer to the DISK-Devkit8600\HW design\datasheet\Serial\MAX3232.pdf

2.3 Hardware interface

2.3.1 Power Input Interface

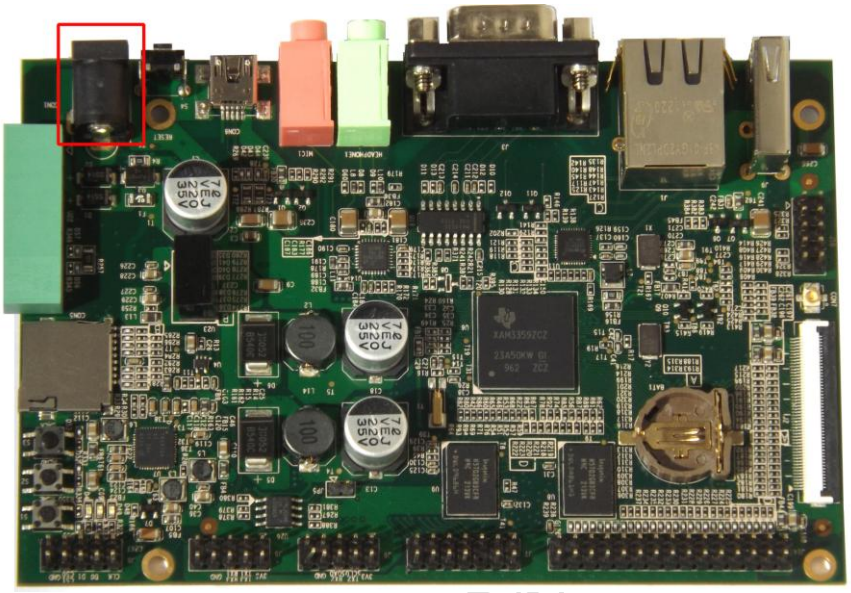


Table 2-1 Power input interface

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

2.3.2 TFT_LCD Interface

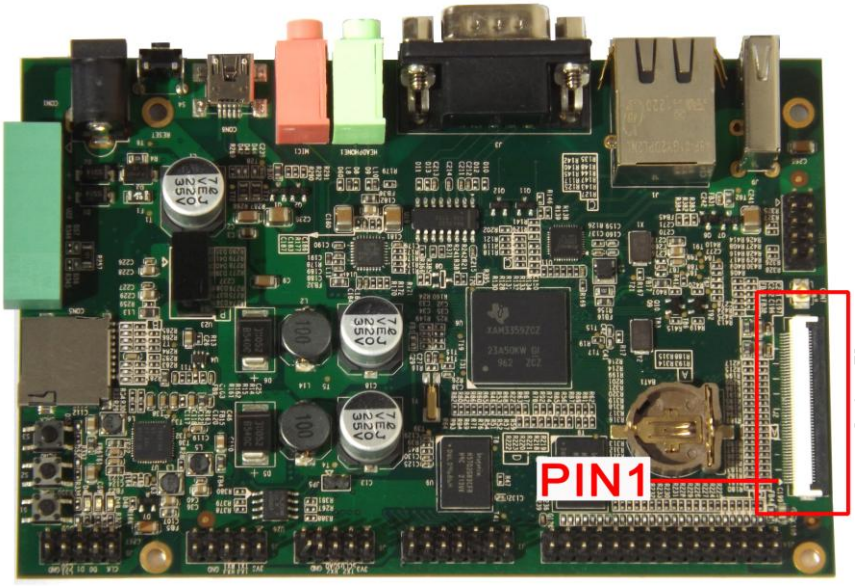


Table 2-2 TFT_LCD interface

J2		
Pin	**Signal**	**Function**
1	B0	GND
2	B1	GND
3	B2	GND
4	B3	LCD Pixel data bit 0
5	B4	LCD Pixel data bit 1
6	B5	LCD Pixel data bit 2
7	B6	LCD Pixel data bit 3
8	B7	LCD Pixel data bit 4
9	GND	GND
10	G0	GND
11	G1	GND
12	G2	LCD Pixel data bit 5
13	G3	LCD Pixel data bit 6
14	G4	LCD Pixel data bit 7
15	G5	LCD Pixel data bit 8

16	G6	LCD Pixel data bit 9
17	G7	LCD Pixel data bit 10
18	GND1	GND
19	R0	GND
20	R1	GND
21	R2	GND
22	R3	LCD Pixel data bit 11
23	R4	LCD Pixel data bit 12
24	R5	LCD Pixel data bit 13
25	R6	LCD Pixel data bit 14
26	R7	LCD Pixel data bit 15
27	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	GND4	GND
34	X+	X+ Position Input
35	X-	X- Position Input
36	Y+	Y+ Position Input
37	Y-	Y- Position Input
38	NC	NC
39	NC	NC
40	NC	NC
41	NC	NC
42	IIC_CLK	IIC master serial clock
43	IIC_DAT	IIC serial bidirectional data
44	GND5	GND

45	VDD1	3.3V
46	VDD2	3.3V
47	VDD3	5V
48	VDD4	5V
49	NC	NC
50	PWREN	Backlight enable

Do not charged pluggable LCD cable

2.3.3 Audio Out Interface

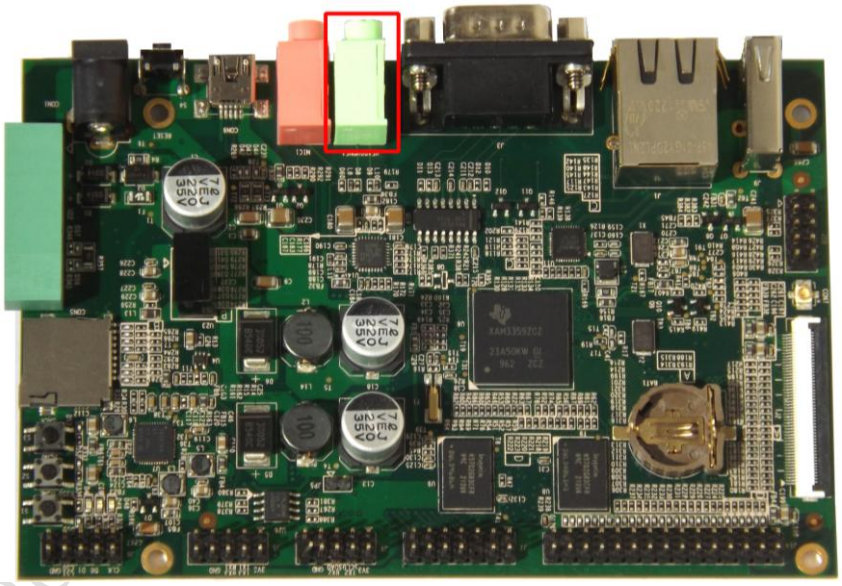


Table 2-3 Audio out interface

Headphone		
Pin	**Signal**	**Function**
1	GND	GND
2	NC	NC
3	Right	Right output
4	NC	NC
5	Left	Left output

2.3.4 MIC In Interface

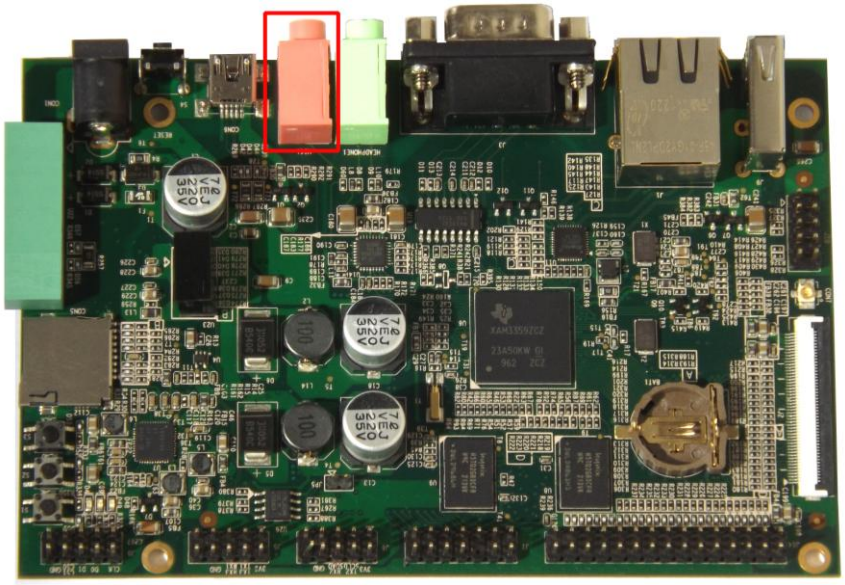


Table 2-4 MIC IN interface

MIC1		
Pin	**Signal**	**Function**
1	GND	GND
2	NC	NC
3	MIC MAIN P	Right input
4	NC	NC
5	MIC MAIN N	Left input

2.3.5 USB HOST Interface

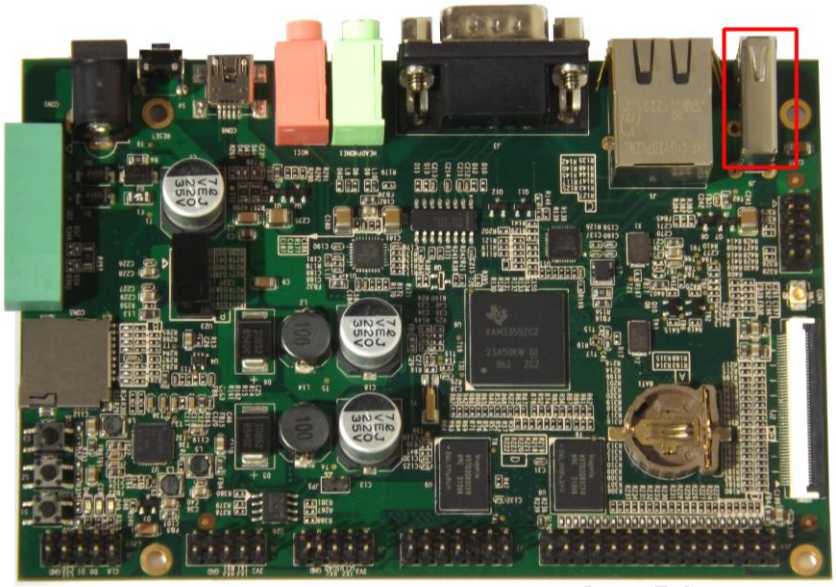


Table 2-5 USB HOST interface

J9		
Pin	**Signal**	**Function**
1	VB	+5V
2	D-	USB Data-
3	D+	USB Data+
4	GND	GND

2.3.6 USB OTG Interface

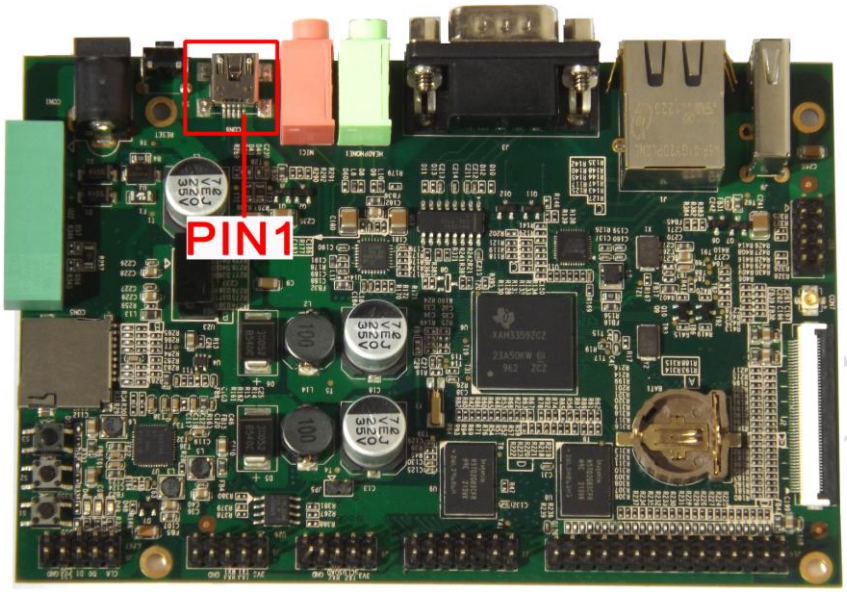


Table 2-6 USB OTG interface

CON6		
Pin	**Signal**	**Function**
1	VB	+5V
2	D-	USB Data-
3	D+	USB Data+
4	ID	USB ID
5	G1	GND

2.3.7 TF Card slot

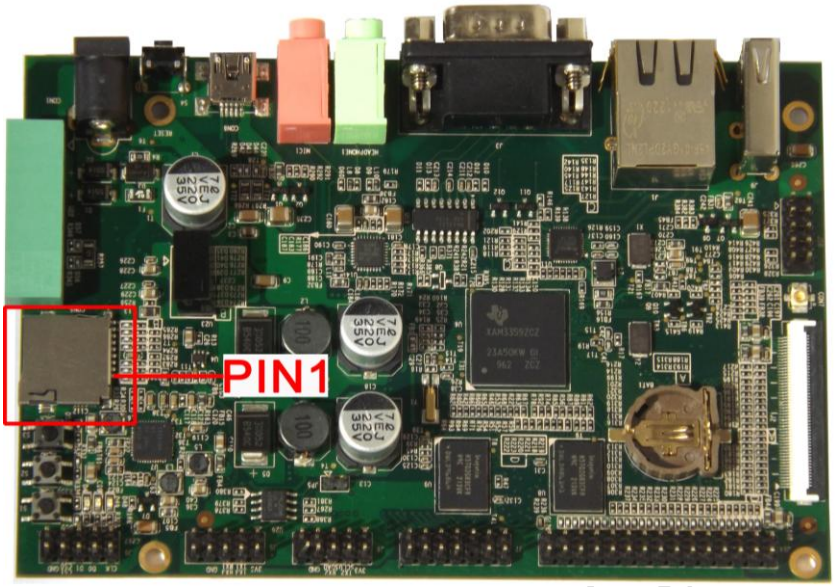


Table 2-7 TF card slot

CON5		
Pin	**Signal**	**Function**
1	DAT2	Card data 2
2	CD/DAT3	Card data 3
3	CMD	Command Signal
4	VDD	VDD
5	CLOCK	Clock
6	VSS	VSS
7	DAT0	Card data 0
8	DAT1	Card data 1
9	CD	Card detect

2.3.8 LAN Interface

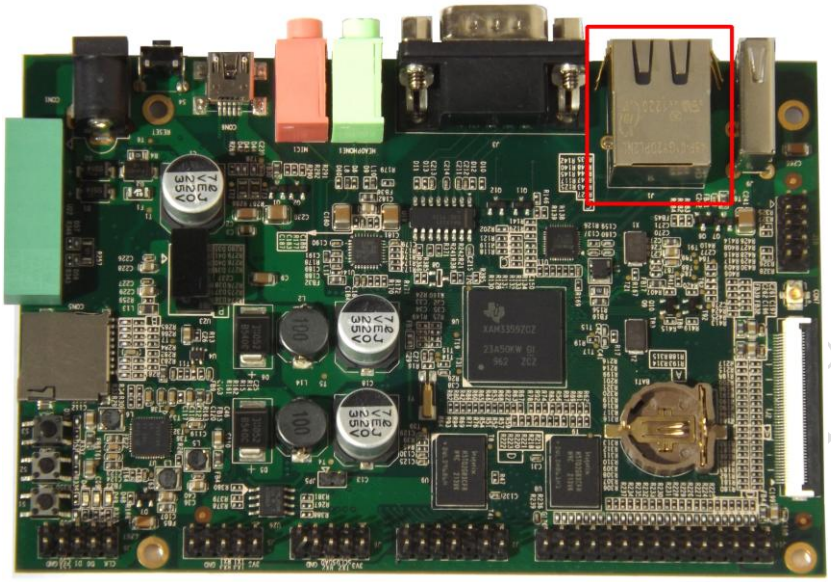


Table 2-8 LAN interface

J1		
Pin	**Signal**	**Function**
1	TD1+	Transmit Data1+
2	TD1-	Transmit Data1-
3	TDT2+	Media-dependent interface 1, 100 transmission line
4	TDT2-	Media-dependent interface 1, 100 transmission line
5	TCT	Transmit common terminal
6	RCT	Isolating transformer
7	RD1+	Media-dependent interface 2, 100 transmission line
8	RD1-	Media-dependent interface 2, 100 transmission line
9	RD2+	Media-dependent interface 3, 100 transmission line
10	RD2-	Media-dependent interface 3, 100 transmission line
11	GRLA	+2.5V
12	GRLC	LINK active LED

13	YELC	Linked LED
14	YELA	+2.5V

2.3.9 Serial Port

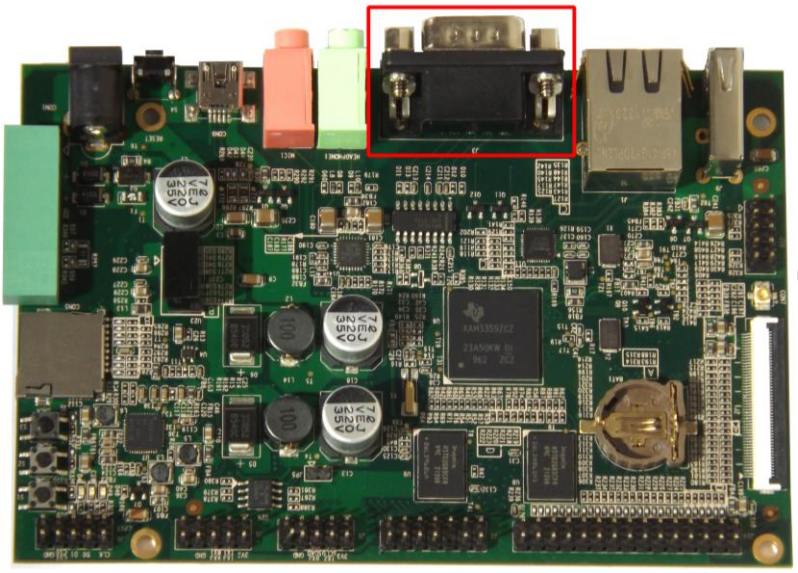


Table 2-9 Serial port

J3		
Pin	**Signal**	**Function**
1	NC	NC
2	RXD	Receive data
3	TXD	Transmit data
4	NC	NC
5	GND	GND
6	NC	NC
7	RTS	Request To Send
8	CTS	Clear To Send
9	NC	NC

2.3.10 CAN&RS485 Interface

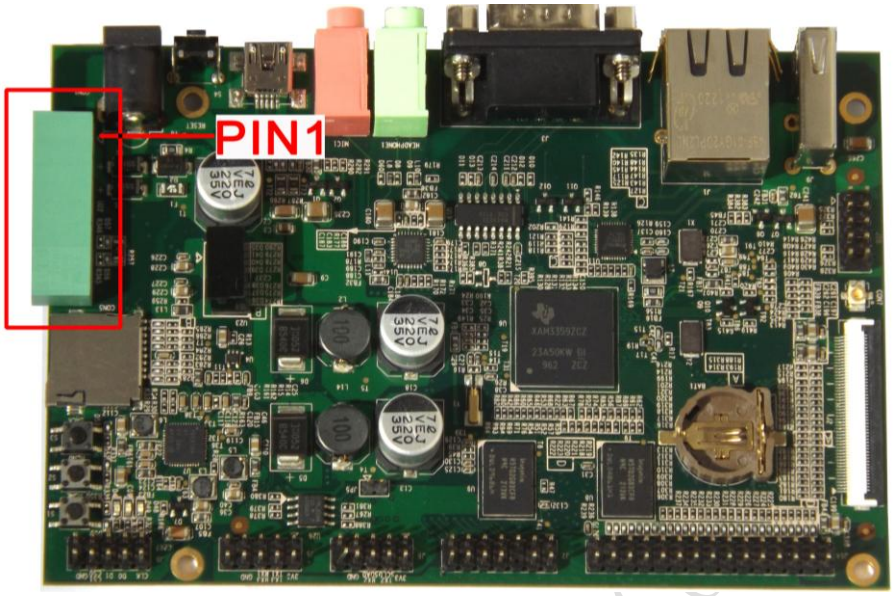


Table 2-10 CAN&RS485 interface

U22		
Pin	**Signal**	**Function**
1	+12V	+12V
2	GND	GND
3	GND2	Isolated GND
4	485B	485B
5	485A	485A
6	GND1	Isolated GND
7	CANL	CANL
8	CANH	CANH

2.3.11 JTAG Interface

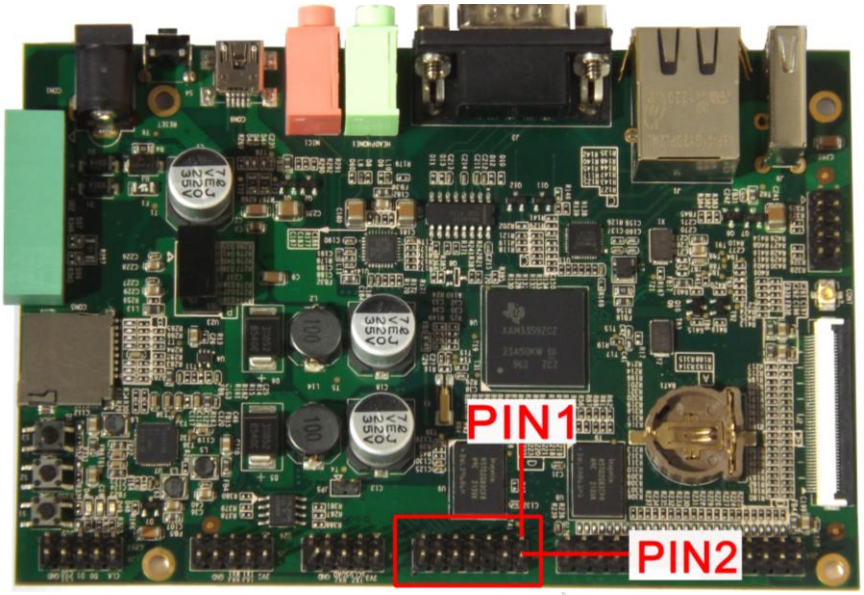


Table 2-11 JTAG interface

J7		
Pin	**Signal**	**Function**
1	TMS	Test mode select
2	NTRST	Test system reset
3	TDI	Test data input
4	GND	GND
5	VIO	3.3V
6	NC	NC
7	TDO	Test data output
8	GND	GND
9	RTCK	Receive test clock
10	GND	GND
11	TCK	Test clock
12	GND	GND
13	EMU0	Test emulation 0
14	EMU1	Test emulation 1

2.3.12 ADC

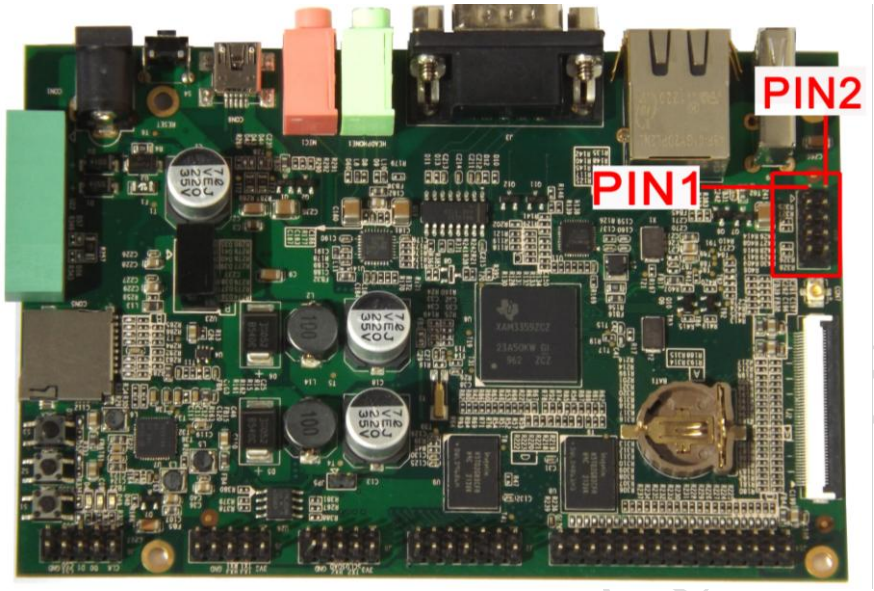


Table 2-12 ADC

J10		
Pin	**Signal**	**Function**
1	GND	GND
2	GND	GND
3	ADC_CH1	ADC1
4	ADC_CH3	ADC3
5	VDDA_ADC	Power
6	VDDA_ADC	Power
7	ADC_CH2	ADC2
8	ADC_CH4	ADC4
9	GND	GND
10	GND	GND

2.3.13 SPI interface

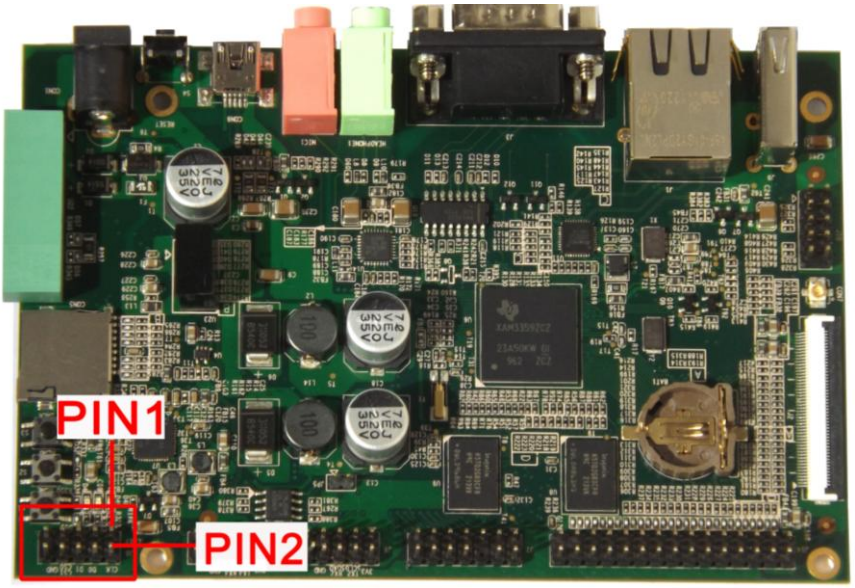


Table 2-13 SPI

J8		
Pin	**Signal**	**Function**
1	SPI_CLK	SPI clock
2	SPI_CLK	SPI clock
3	SPI_D0	SPI DATA0
4	SPI_D0	SPI DATA0
5	SPI_D1	SPI data1
6	SPI_D1	SPI data1
7	SPI_CS0	SPI chip select 0
8	SPI_CS1	SPI chip select 1
9	GND	GND
10	VIO_3V3	+3.3V

2.3.14 GPMC interface

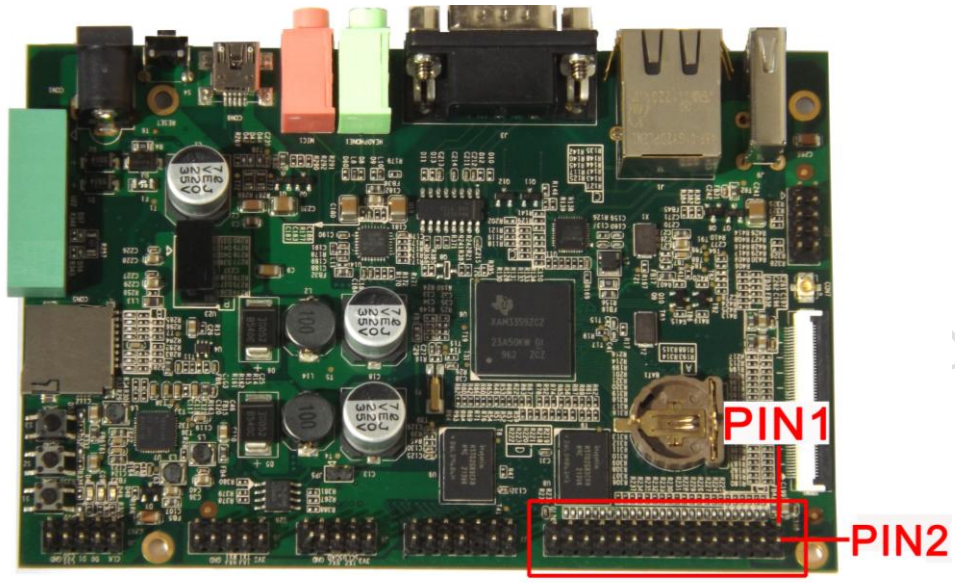


Table 2-14

J14		
Pin	**Signal**	**Function**
1	GND	GND
2	VDD3V3_GPMC	3.3V power
3	GPIO0_31	GPIO
4	GPIO0_30	GPIO
5	GPIO1_28_R	GPIO
6	GPIO2_5	GPIO
7	GPIO2_2	GPIO
8	GPIO2_3	GPIO
9	GPIO1_29	GPIO
10	GPIO2_4	GPIO
11	GPMC_A11	GPMC Address
12	GPMC_A10	GPMC Address
13	GPMC_A9	GPMC Address
14	GPMC_A8	GPMC Address
15	GPMC_A7	GPMC Address
16	GPMC_A6	GPMC Address
17	GPMC_A5	GPMC Address
18	GPMC_A4	GPMC Address

19	GPMC_A3	GPMC Address
20	GPMC_A2	GPMC Address
21	GPMC_A1	GPMC Address
22	GPMC_A0	GPMC Address
23	GPMC_AD7	GPMC Address & Data
24	GPMC_AD6	GPMC Address & Data
25	GPMC_AD5	GPMC Address & Data
26	GPMC_AD4	GPMC Address & Data
27	GPMC_AD3	GPMC Address & Data
28	GPMC_AD2	GPMC Address & Data
29	GPMC_AD1	GPMC Address & Data
30	GPMC_AD0	GPMC Address & Data

2.3.15 Expansion Interface

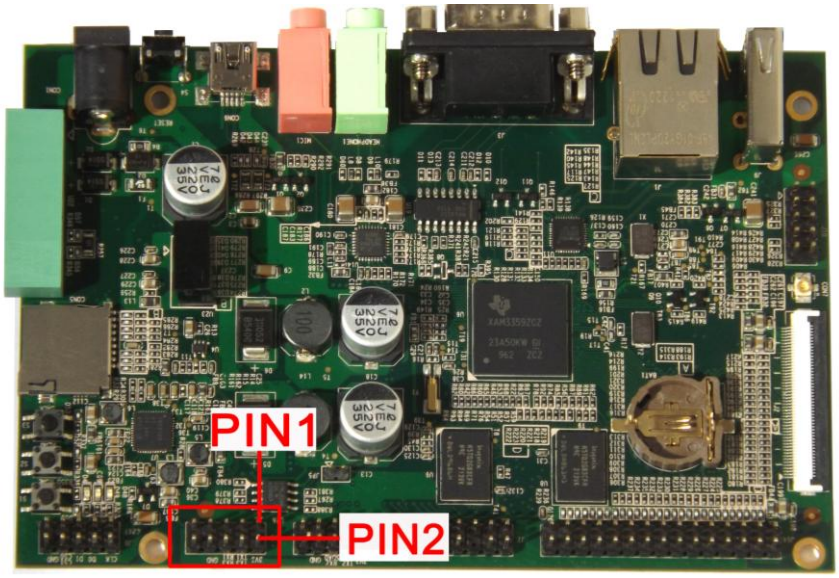


Table 2-15 Expansion interface

J5		
Pin	**Signal**	**Function**
1	VIO_3V3	+3.3V
2	VIO_3V3	+3.3V
3	UART1_TX_3V3	UART1 Transit data 3.3V level
4	UART3_TX_3V3	UART3 Transit data 3.3V level
5	UART1_RX_3V3	UART1 receive data 3.3V level
6	UART3_RX_3V3	UART3 receive data 3.3V level

7	GND	GND
8	GND	GND
9	GND	GND
10	GND	GND

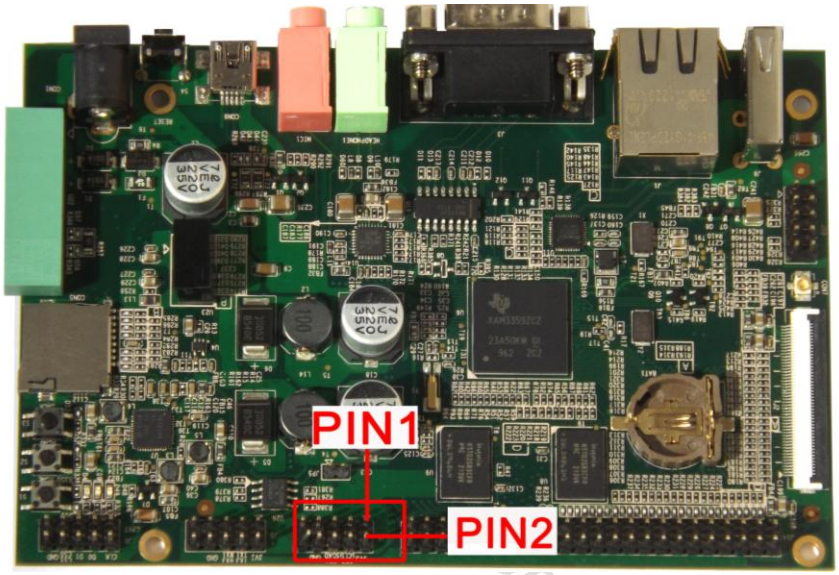


Table 2-16

J6		
Pin	**Signal**	**Function**
1	VIO_3V3	+3.3V
2	VIO_3V3	+3.3V
3	I2C0_SCL_3V3	IIC0 master serial clock 3.3V level
4	UART2_TX_3V3	UART2 transit data 3.3V level
5	I2C0_SDA_3V3	I2C0 master serial data 3.3V level
6	UART2_RX_3V3	UART2 receive data 3.3V level
7	GND	GND
8	GND	GND
9	GND	GND
10	GND	GND

2.3.16 KEY

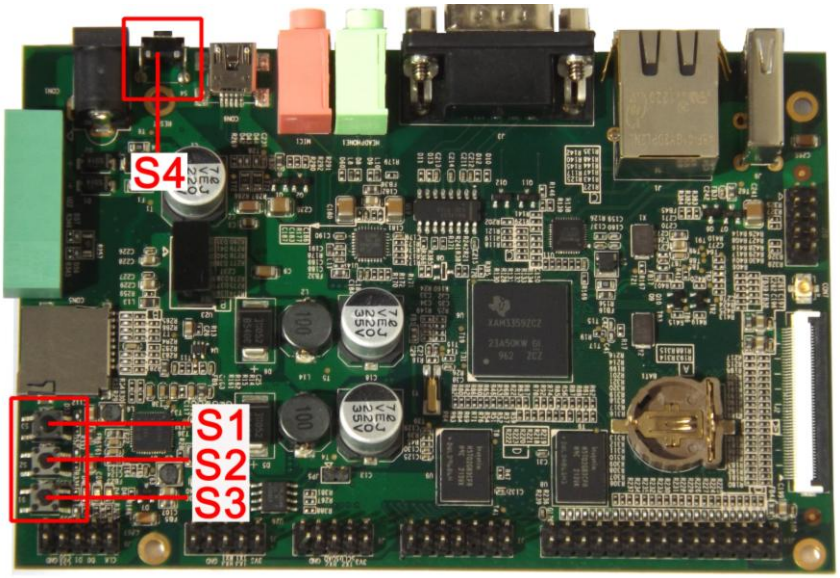


Table 2-17 KEY

S1-3		
Pin	**Signal**	**Function**
S1	HOME	User-defined key
S2	MENU	System menu key
S3	BACK	System back key
S4	SW PUSHBUTTOn	Power Switch button

2.3.17 LED

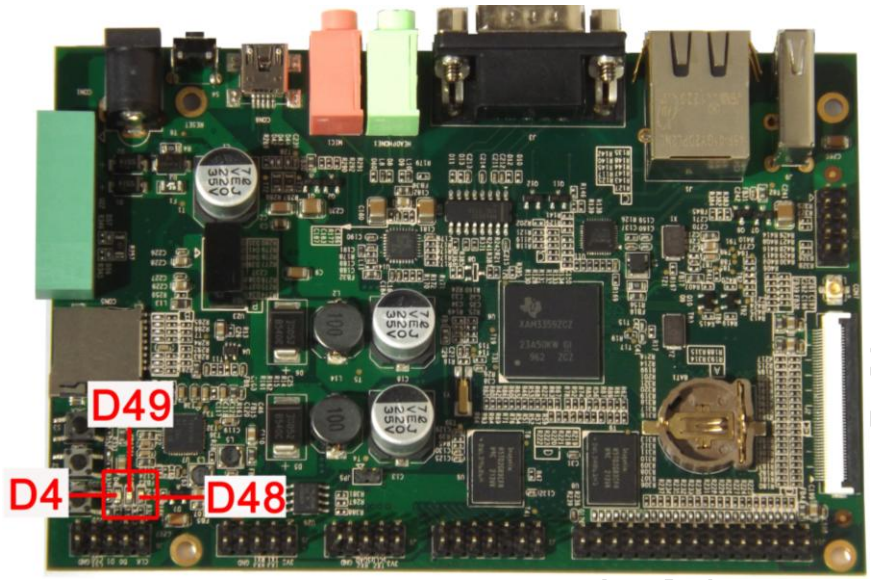


Table 2-18 LED

D4,D48,D49		
Number	**Title**	**Function**
1	D4	System Indicator
2	D48	User-defined LED
3	D49	User-defined LED

3 Chapter 3 Linux Operating System

3.1 Introduction

This section is intended to provide detailed instruction on Operating System Software development of DevKit8600 board.

1) Describes the Software Resources provided by DevKit8600.
2) Describes the software feature.
3) Explains the software Development including how to set up the development environment, the building guidance of the boot loader, kernel and file system, and the development of device driver.
4) Provides flashing methods using boot loader commands.
5) Shows the usage of DevKit8600
6) Shows the application development.

In this part, it is suggested to:

1) Install Ubuntu Linux in advance, please refer to Appendix II for details;
2） Master relative embedded Linux development technology.

3.2 Software Resources

This chapter provides an overview of software system components of DevKit8600. A basic software system consists of four parts: spl, u-boot, kernel and rootfs. The Figure 3-1 shows the structure of the system:

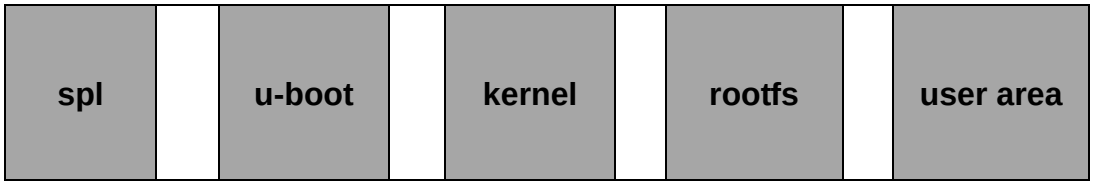


Figure 3-1

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

1) spl is a first level bootstrap program. After the system start-up, the ROM inside the CPU will copy the spl to internal RAM and perform its routine work. Its main function is to initialize the CPU, copy u-boot into the memory and give the control to u-boot;
2) u-boot is a second level bootstrap program. It is used for interacting with users, updating images and leading the kernel;
3) The latest Linux3.1.0 kernel is employed here and it can be customized based on DevKit8600;
4) rootfs employs Open-source system. It is small in capacity and powerful, very suitable for embedded systems;

3.3 Software Features

Items		Notes
BIOS	spl	NAND
		MMC/SD
		FAT
	u-boot	NAND
		MMC/SD
		FAT
		NET
Kernel	Linux-3.1.0	Supports ROM/CRAM/EXT2/EXT3/FAT/NFS/ JFFS2/UBIFS and various file systems
Device Driver	serial	Series driver
	rtc	Hardware clock driver
	net	10/100M Ethernet driver
	can	can bus driver
	flash	nand flash driver (supports nand boot)
	LCD	TFT LCD driver
	Touch screen	Touch screen controller driver

	mmc/sd	mmc/sd controller driver
	usb otg	usb otg 2.0 driver
	Audio	Audio driver
	keypad	gpio keyboard driver
	Led	User led driver
Demo	Android	android 2.3.4 system
	TISDK	TISDk system

Table 3-1

3.4 System Development

3.4.1 Establishing operating system development environment

Before executing software development on DevKit8600, the user has to establish a Linux cross development environment and install it in computer. How to establish a cross development environment will be introduced below by taking Ubuntu operating system as an example.

3.4.1.1 Installation of cross compilation tools

Installation of cross compilation tools is done by using the software CD provided along with this kit, to start the process insert the CD and allow it for auto run, Ubuntu will mount the disc under the directory /media/cdrom, the cross compilation tools are saved under the directory /media/cdrom/linux/tools.

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

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

Some of the other development tools used for source code compilation are present in the

directory linux/tools of the disc; the user can execute the following commands to copy them to local folder:

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

3.4.1.2 Addition of environment variables

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

```
export PATH=$HOME/tools/arm-2009q1/bin:$HOME/tools/arm-eabi-4.4.0/bin:
$HOME/tools:$PATH
```

The user can write it in the .bashrc file under the user directory, such that the addition of environment variables will be finished automatically when the system is booted; command echo $PATH can be used to check the path.

3.4.1.3 Building Android development environment

In addition to the installation of cross compiling tools and environment variables, there are some software packages and configurations need to be handled before you can implement compilation of Android source codes. For detailed information, please refer to "Setting up a Linux build environment" on the Android web site http://source.android.com/source/initializing.html.

3.4.2 System compilation

3.4.2.1 Preparation

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

```
mkdir $HOME/work
cd $HOME/work
tar xvf /media/cdrom/linux/source/u-boot-2011.09-psp04.06.00.03.tar.bz2
tar xvf /media/cdrom/linux/source/linux-3.1.0-psp04.06.00.03.sdk.tar.bz2
tar xvf /media/cdrom/linux/demo/android/source/linux-3.1.0-android.tar.bz2
sudo tar xvf /media/cdrom/linux/source/rootfs.tar.bz2
tar xvf
/media/cdrom/linux/demo/android/source/rowboat-android-gingerbread-am335xevm.tar.bz2
```

After the above commands are executed, the directories u-boot-2011.09-psp04.06.00.03, linux-3.1.0-psp04.06.00.03.sdk, linux-3.1.0-android, rowboat-android-gingerbread-am335xevm and rootfs will be created under current directory**.**

Please do not uncompress the source file to other directory, or errors might occur during compilation.

3.4.2.2 Bootstrap program generation

DevKit8600 supports TF Card boot or NAND boot. System preferences for MMC / SD start, if MMC / SD failed to start, turning to NAND Flash start.

We will introduce the generation of bootstrap program.

```
cd u-boot-2011.09-psp04.06.00.03
make distclean
make devkit8600_config
make
```

When the above steps are finished, the current directory will generate the files MLO and u-boot.img which we need.

3.4.2.3 Kernel compilation

Before kernel compilation, the user has to select correct display mode from the custom

menu of kernel according to your display device.

As for Linux system, please enter following commands in the terminal window of Ubuntu:

```
cd linux-3.1.0-psp04.06.00.03.sdk
make distclean
make devkit8600_defconfig
make menuconfig
```

As for Android system, please enter following commands in the terminal window of Ubuntu:

```
cd linux-3.1.0-android
make distclean
make devkit8600_android_defconfig
make menuconfig
```

If an error occurs in the system when make menuconfig is input, it is necessary to install ncurse in the Ubuntu system; ncurse library is a character graphic library, used for make menuconfig of kernel; the specific installation instruction is:

```
sudo apt-get install ncurses-dev
```

Enter the kernel customize menu now, enter “PANEL_TYPE” according to the following pointing paths:

```
Location:
  -> Device Drivers
    -> Graphics support
      -> Support for frame buffer devices (FB [=y])
        -> DA8xx/OMAP-L1xx Framebuffer support (FB_DA8XX [=y])
Selected by: HAS_IOMEM [=y] && FB_DA8XX [=y] && m
```

Figure 3-2

Select under “PANEL_TYPE” according to actually displayed screen size:

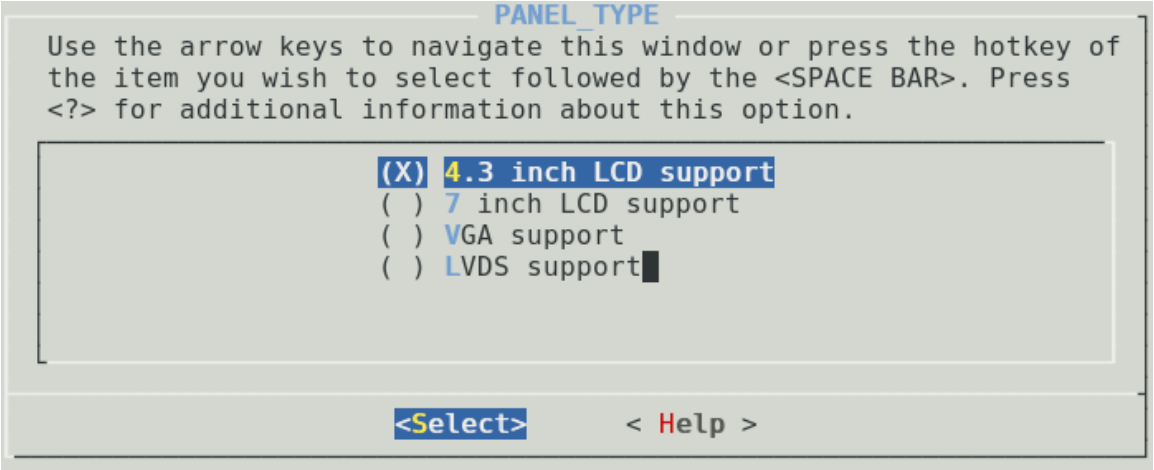


Figure 3-3

After determining “PANEL_TYPE”, jump to parent directory, select “Exit” to exit, until the following picture appears, then select “Yes”:

Figure 3-4

```
make uImage
```

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

3.4.2.4 Generation of file system

1) Ramdisk file making

For Ramdisk making, please refer to

http://www.elinux.org/DevKit8600_FAQ

2) UBI file making

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

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

3.4.2.5 Building Android Filesystem

1) Please enter the following commands to compile the source file of Android system:

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

2) Enter the following command to modify the file Rules.make under hardware/ti/sgx/:

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

Modify KERNEL_INSTALL_DIR=$(HOME)/work/linux-3.1.0-android as KERNEL_INSTALL_DIR=/home/**user_name**/work/linux-3.1.0-android

The /home/user_name is the value of $(HOME). You can enter whoami in the terminal window of Linux to view the value.

3) Enter the following command to create an ubi file system:

```
source ./build_ubi.sh
```

The ubi.img can be found under temp/.

Before you start the compilation of Android file system, you need to first compile **linux-3.1.0-android**, the kernel source code of Android, or errors might occur.

3.4.3 System Customization

As Linux kernel has many kernel configuration options, the user can increase or reduce the driver or some kernel features based on the default configuration to meet the demands in better ways. The general process of system customization will be described with examples below.

3.4.3.1 Modification of kernel configuration

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

linux-3.1.0-psp04.06.00.03.sdk/arch/arm/configs/devkit8600_defconfig

User can carry out system customization on this basis:

```
cd linux-3.1.0-psp04.06.00.03.sdk
```

```
cp arch/arm/configs/devkit8600_defconfig .config
make menuconfig
```

The system customization will be described below by taking usb gadget and usb mass storage device as an example:

Select the configuration below:

-> Device Drivers

-> USB support

-> USB Gadget Support

-> USB Gadget Drivers

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

Figure 3-5

Select “File-backed Storage Gadget” as <M>, exit, and finally select Save to recompile kernel.

3.4.3.2 Compilation

Save configuration, execute the following commands to recompile kernel:

```
make uImage
make modules
```

After above operations are executed, a new kernel image uImage will be generated under the directory arch/arm/boot, and a module file g_file_storage.ko will be generated under the directory drivers/usb/gadget.

3.5 Introduction of driver

3.5.1 NAND

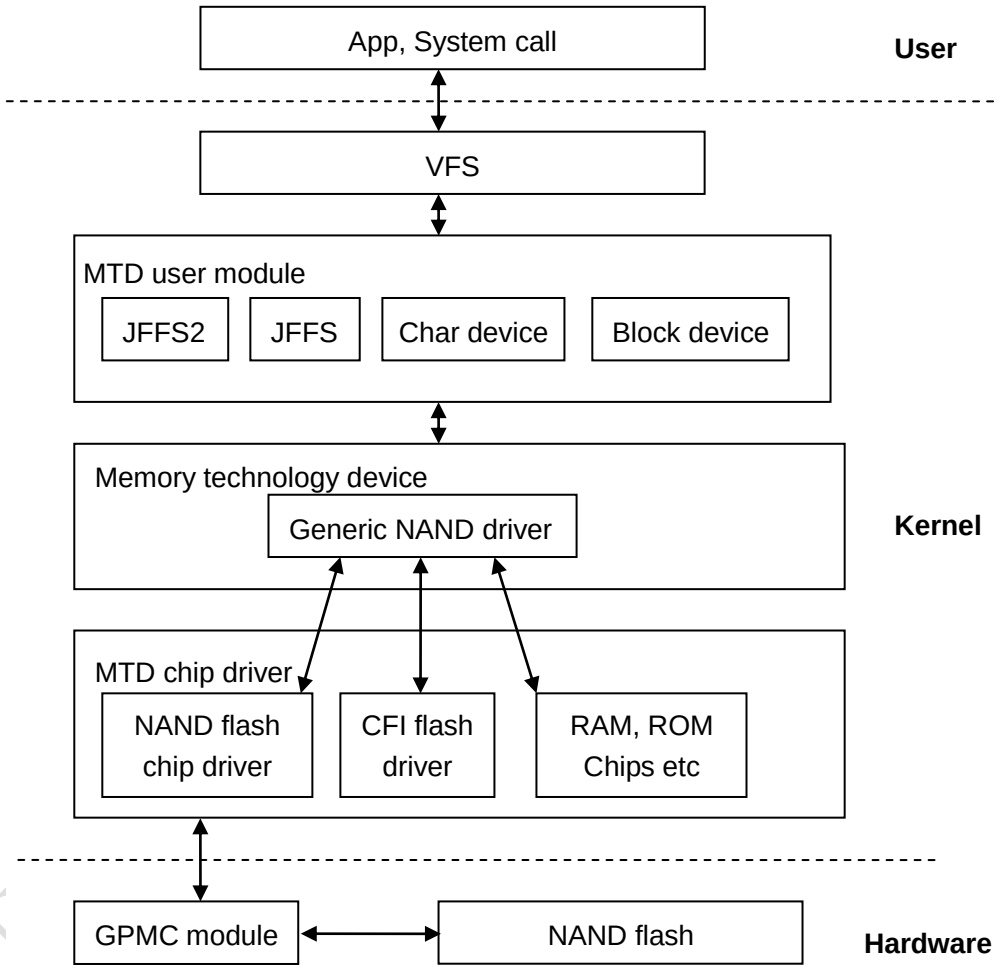


Figure 3-6 Modular structure for NAND

Solid-state memory used in embedded systems is mainly flash; it is NAND flash in this system.

NAND flash is used as a block device, on which the file system is arranged; interaction

between user and NAND flash is mainly realized by a specific file system. In order to shield difference in different flash memories, kernel inserts an MTD subsystem between the file system and the specific flash driver for management.

Therefore, the user accesses NAND flash through the following process:

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

Kernel Driver reference path:

linux-3.1.0-psp04.06.00.03.sdk/drivers/mtd/nand/

linux-3.1.0-psp04.06.00.03.sdk/drivers/mtd/nand/omap2.c

3.5.2 SD/MMC

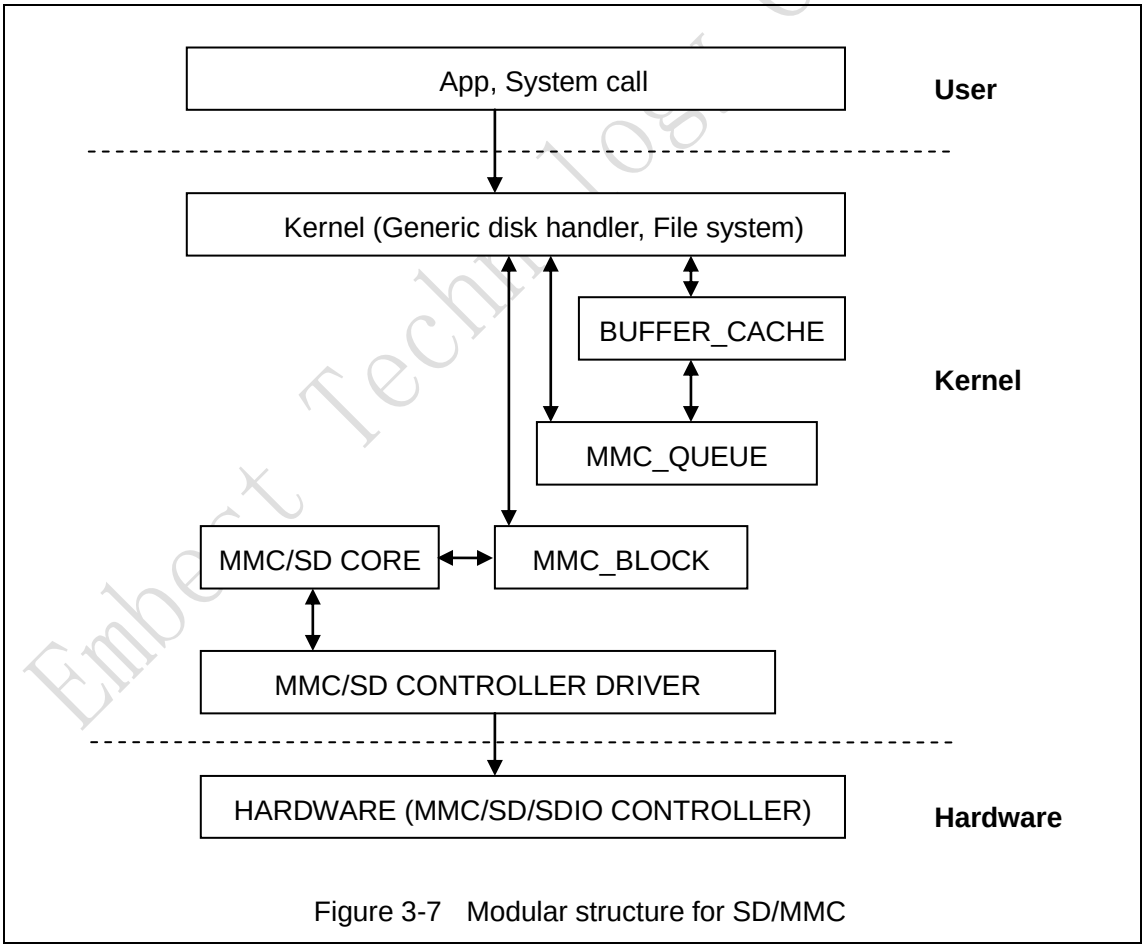


Figure 3-7 Modular structure for SD/MMC

SD/MMC card drivers under Linux mainly include SD/MMC core, mmc_block, mmc_queue and SD/MMC driver four parts:

1) SD/MMC core realizes core codes unlated to structure in the SD/MMC card

operation.

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

3) mmc_queue realizes management of request queue.

4) SD/MMC driver realizes specific controller driver.

Kernel Driver reference path:

linux-3.1.0-psp04.06.00.03.sdk/drivers/mmc/

linux-3.1.0-psp04.06.00.03.sdk/drivers/mmc/host/omap_hsmmc.c

3.5.3 LCDC

LCD controller (LCDC) on AM335x is an updated version of LCDC that is found on OMAP-L138 SoC. It has following updates in comparison with OMAP-L138

- Interrupt configuration and status registers are different.
- Increased resolution of 2048*2048.
- 24 bits per pixel active TFT raster configuration.

So da8xx-fb LCD driver can be used by having enhancements under LCD_VERSION2 code. This update in LCDC version can be detected by reading PID register.

Kernel Driver reference path:

linux-3.1.0-psp04.06.00.03.sdk/drivers/video/

linux-3.1.0-psp04.06.00.03.sdk/drivers/video/da8xx-fb.c

3.5.4 Audio in/out

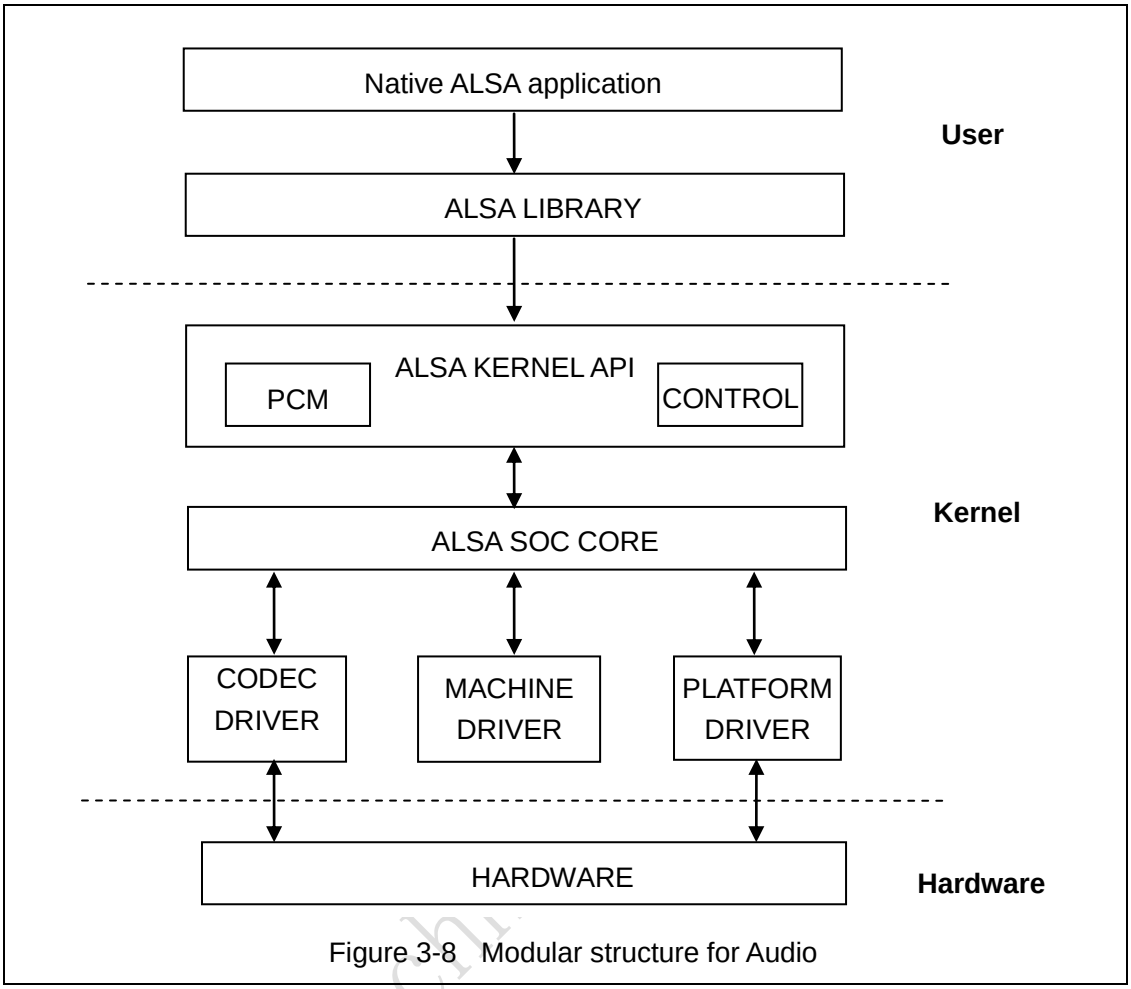


Figure 3-8 Modular structure for Audio

ASoC basically splits an embedded audio system into three components:

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

Kernel Driver reference path:

linux-3.1.0-psp04.06.00.03.sdk/sound/soc/

linux-3.1.0-psp04.06.00.03.sdk/sound/soc/davinci/davinci-evm.c

linux-3.1.0-psp04.06.00.03.sdk/sound/soc/codecs/sgtl5000.c

3.6 Driver Development

3.6.1 Driver For The gpio_keys

1) Device Definition

linux-3.1.0-psp04.06.00.03.sdk/arch/arm/mach-omap2/board-am335xevm.c

Setup **GPIO 1.30** as “menu” key, return value as “KEY_F1”, triggered on low level; **GPIO 1.31** as ”back” key, return value as ”KEY_ESC”, triggered on low level. The structure template is shown below. **GPIO 0.22** as “home” key, return value as “KEY_HOME”, triggered on low level.

```
static struct gpio_keys_button gpio_key_buttons[] = {
        {
                .code                   = KEY_F1,
                .gpio                   = GPIO_TO_PIN(1, 30),
                .active_low             = true,
                .desc                   = "menu",
                .type                   = EV_KEY,
//              .wakeup                 = 1,
        },
        {
                .code                   = KEY_ESC,
                .gpio                   = GPIO_TO_PIN(1, 31),
                .active_low             = true,
                .desc                   = "back",
                .type                   = EV_KEY,
//              .wakeup                 = 1,
        },
        {
                .code                   = KEY_HOME,
                .gpio                   = GPIO_TO_PIN(0, 22),
```

```
                .active_low             = true,
                .desc                   = "home",
                .type                   = EV_KEY,
//              .wakeup                 = 1,
        },

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

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

2) GPIO pinmux Configuration

Setup the GPIO 1.30, GPIO1.31 and GPIO0.22 as M7(GPIO mode), IEM (Input enable).

linux-3.1.0-psp04.06.00.03.sdk/arch/arm/mach-omap2/board-am335xevm.c

```
static struct pinmux_config gpio_keys_pin_mux[] = {
        {"gpmc_csn1.gpio1_30",  OMAP_MUX_MODE7 | AM33XX_PIN_INPUT},
        {"gpmc_csn2.gpio1_31",  OMAP_MUX_MODE7 | AM33XX_PIN_INPUT},
        {"gpmc_ad8.gpio0_22",   OMAP_MUX_MODE7 | AM33XX_PIN_INPUT},
        {NULL, 0},
};
```

3) Driver Design

linux-3.1.0-psp04.06.00.03.sdk/drivers/input/keyboard/gpio_keys.c

a) Structure for platform_driver_register to register gpio_keys driver.

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

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

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

late_initcall(gpio_keys_init);
module_exit(gpio_keys_exit);

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

b) Structure for input_register_device to register input driver.

```
static int __devinit gpio_keys_probe(struct platform_device *pdev)
```

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

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

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

                if (button->wakeup)
                        wakeup = 1;

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

        error = input_register_device(input);
        if (error) {
```

```
                dev_err(dev, "Unable to register input device, error: %d\n",
                        error);
                goto fail3;
        }
…
```

c) Apply GPIO and setup the GPIO as the input, registration GPIO interrupt.

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

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

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

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

```
                goto fail3;
        }

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

        irq = gpio_to_irq(button->gpio);
        if (irq < 0) {
                error = irq;
                dev_err(dev, "Unable to get irq number for GPIO %d, error %d\n",
                        button->gpio, error);
                goto fail3;
        }
        irqflags = IRQF_TRIGGER_RISING | IRQF_TRIGGER_FALLING;
        /*
         * If platform has specified that the button can be disabled,
         * we don't want it to share the interrupt line.
         */
        if (!button->can_disable)
                irqflags |= IRQF_SHARED;

        error = request_threaded_irq(irq, NULL, gpio_keys_isr, irqflags, desc,
bdata);
        if (error < 0) {
                dev_err(dev, "Unable to claim irq %d; error %d\n",
```

```
                                irq, error);
                        goto fail3;
                }

                return 0;

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

d) Interrupt handling,

Button is pressed, an interrupt is generated, reporting key

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

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

static void gpio_keys_report_event(struct gpio_button_data *bdata)
{
        struct gpio_keys_button *button = bdata->button;
```

```
        struct input_dev *input = bdata->input;
        unsigned int type = button->type ?: EV_KEY;
        int state = (gpio_get_value(button->gpio) ? 1 : 0) ^ button->active_low;

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

3.6.2 Driver For The gpio_leds

1) Device Definition

linux-3.1.0-psp04.06.00.03.sdk/arch/arm/mach-omap2/board-am335xevm.c

The kernel configuration respectively are: sys_led (GPIO1.26), usr_led (GPIO1.27), low level is enable:

```
static struct gpio_led gpio_leds[] = {
        {
                .name                   = "sys_led",
                .default_trigger        = "heartbeat",
                .gpio                   = GPIO_TO_PIN(1, 26),
        },
        {
                .name                   = "user_led",
                .gpio                   = GPIO_TO_PIN(1, 27),
        },
};

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

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

2) GPIO pinmux Setup:

linux-3.1.0-psp04.06.00.03.sdk/arch/arm/mach-omap2/board-am335xevm.c

Configure GPIO 1.26 and GPIO 1.27 as M7(MODE 7 = GPIO), IDIS(Input not allowed)

```
static struct pinmux_config gpio_led_pin_mux[] = {
        {"gpmc_a10.gpio1_26",   OMAP_MUX_MODE7 | AM33XX_PIN_OUTPUT},
        {"gpmc_a11.gpio1_27",                   OMAP_MUX_MODE7       |
AM33XX_PIN_OUTPUT},
        {NULL, 0},
};
```

3) Driver design:

linux-3.1.0-psp04.06.00.03.sdk/drivers/leds/leds-gpio.c

a) Structure for platform_driver_register to register gpio_leds.

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


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

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

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

module_init(gpio_led_init);
module_exit(gpio_led_exit);

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

b) Apply GPIO and called led_classdev_regisiter to register led_classdev.

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

                priv->num_leds = pdata->num_leds;
                for (i = 0; i < priv->num_leds; i++) {
```

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

        static int __devinit create_gpio_led(const struct gpio_led *template,
                struct gpio_led_data *led_dat, struct device *parent,
                int (*blink_set)(unsigned, unsigned long *, unsigned long *))
        {
        …
                ret = gpio_request(template->gpio, template->name);
        …
                ret = gpio_direction_output(led_dat->gpio, led_dat->active_low ^ state);
        …
                ret = led_classdev_register(parent, &led_dat->cdev);
        …
        }
```

c) User can access brightness file on the directory of /sys/class/leds/xxx/brightness, called function gpio_led_set to configure led states.

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

3.7 Updated of system

3.7.1 Update of TF card system image

1) **The formatting of MMC/SD card**

HP USB Disk Storage Format Tool 2.0.6 is recommended:

The software is downloading from

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

a) Insert TF card into the card reader in PC.

b) Open the HP USB Disk Storage Format Tool, the following steps will show in detail:

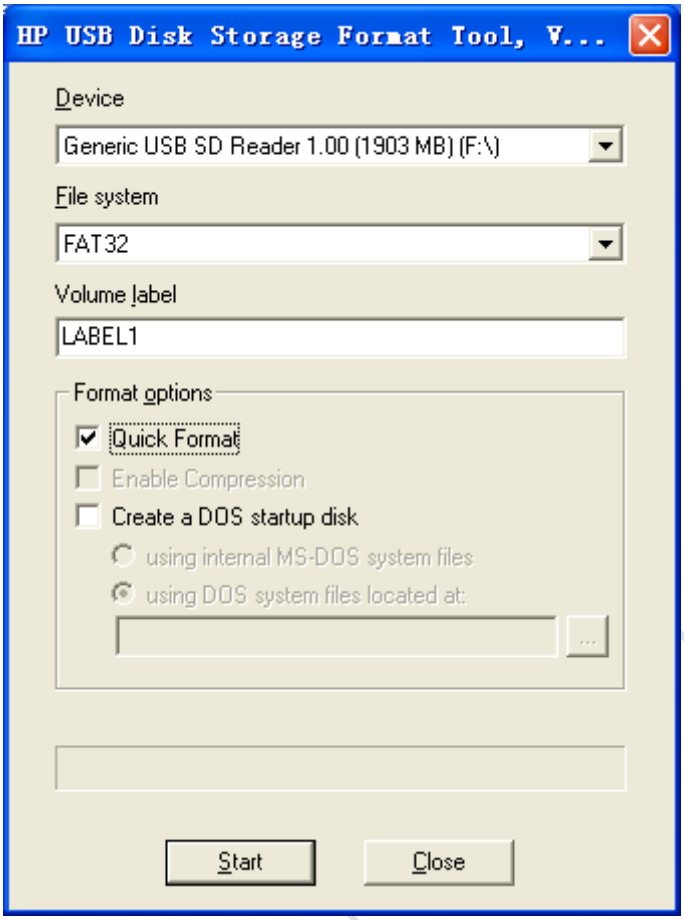


Figure 3-9

c) Select “FAT32”.

d) Click “Start”.

e) When formatting is completed, click “OK”.

HP USB Disk Storage Format Tool will clear partitions of the TF card. Please use the formatting software provided in the computer system

2) **Update of images**

Copy all files under the directory linux/image to the TF card, and **rename uImage_xx as uImage** according to the used display device LCD (4.3", 7"), LVDS or VGA. Connect the TF card, power on and boot it, the serial port information will be displayed as follows:

```
U-Boot SPL 2011.09-svn (Mar 02 2012 - 17:15:32)
Texas Instruments Revision detection unimplemented
Booting from MMC...
OMAP SD/MMC: 0
```

```
reading u-boot.img
reading u-boot.img


U-Boot 2011.09-svn (Mar 02 2012 - 17:15:32)

I2C:   ready
DRAM:  512 MiB
WARNING: Caches not enabled
Did not find a recognized configuration, assuming General purpose EVM in Profile 0
with Daughter board
NAND:  HW ECC Hamming Code selected
nand_get_flash_type: unknown NAND device: Manufacturer ID: 0xad, Chip ID: 0xd7
No NAND device found!!!
0 MiB
MMC:   OMAP SD/MMC: 0
*** Warning - readenv() failed, using default environment

Net:   cpsw
Hit any key to stop autoboot:  0
SD/MMC found on device 0
reading uEnv.txt

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

2993120 bytes read
reading ramdisk.gz

12132646 bytes read
```

```
## Booting kernel from Legacy Image at 80007fc0 ...
   Image Name:   Linux-3.1.0
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2993056 Bytes = 2.9 MiB
   Load Address: 80008000
   Entry Point:  80008000
   Verifying Checksum ... OK
   XIP Kernel Image ... OK
OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
Linux version 3.1.0 (luofc@TIOP) (gcc version 4.3.3 (Sourcery G++ Lite 2009q1-203) ) #28 Mon Mar 5 11:04:25 CST 2012
.........
.........
RAMDISK: gzip image found at block 0
VFS: Mounted root (ext2 filesystem) on device 1:0.
Freeing init memory: 240K
INIT: version 2.86 booting
Starting udevudevd (623): /proc/623/oom_adj is deprecated, please use /proc/623/oom_score_adj instead.
tar: removing leading '/' from member names

Remounting root file system...
mount: mounting /dev/root on / failed: Invalid argument
mount: mounting /dev/root on / failed: Invalid argument
root: mount: mounting rootfs on / failed: No such file or directory
Setting up IP spoofing protection: rp_filter.
```

```
Configuring network interfaces... udhcpc (v1.11.3) started
Sending discover...
udhcpc: sendto: Network is down
Sending discover...
udhcpc: sendto: Network is down
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 DevKit8600 ttyO0


Angstrom 2008.1-test-20090127 DevKit8600 ttyO0


DevKit8600 login: (Just input “root”)
```

HyperTerminal displays above information to indicate that it is successful to boot Linux system from TF card.

3.7.2 Update of NAND Flash

Update of NAND boot image is finished in aid with u-boot. No matter whether NAND Flash has data or not, u-boot of the TF card can be used to update NAND Flash images.

1) Preparation

a) Format the TF card to FAT or FAT32 file system through HP USB Disk Storage Format Tool 2.0.6

b) Copy **MLO, u-boot.img, uImage_xx and ubi.img** image files in the disc to the TF card, and rename uImage_xx as uImage according to the display device LCD (4.3", 7"), LVDS or VGA you used.

2) **Update**

a) Insert the TF card with the system images into the development board, power on and boot it, and press any key on the PC keyboard to enter the u-boot according to the following clock prompts:

```
U-Boot SPL 2011.09-svn (Mar 02 2012 - 17:15:32)
Texas Instruments Revision detection unimplemented
Booting from MMC...
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img



U-Boot 2011.09-svn (Mar 02 2012 - 17:15:32)


I2C:   ready
DRAM:  512 MiB
WARNING: Caches not enabled
Did not find a recognized configuration, assuming General purpose EVM in Profile 0
with Daughter board
NAND:  HW ECC Hamming Code selected
```

```
nand_get_flash_type: unknown NAND device: Manufacturer ID: 0xad, Chip ID: 0xd7
No NAND device found!!!
0 MiB
MMC:    OMAP SD/MMC: 0
*** Warning - readenv() failed, using default environment

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

b) After entering the u-boot command line, input "**run updatesys**" from the PC keyboard, to start to update the system automatically:

```
Devkit8600# run updatesys

NAND erase.chip: device 0 whole chip
Erasing at 0x7fe0000 -- 100% complete.
OK
reading MLO

38151 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x0, size 0x9507
 38151 bytes written: OK
reading u-boot.img

232456 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x80000, size 0x38c08
 232456 bytes written: OK
reading uImage
```

```
2984304 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x280000, size 0x2d8970
 2984304 bytes written: OK
reading ubi.img

20447232 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x780000, size 0x1380000
 20447232 bytes written: OK
```

c) At this time, flickering of LED lamp on the board indicates that update has been finished; you just need to pull out the TF card and reboot the board.

3.8 Instructions

3.8.1 Various Tests Scenario

3.8.1.1 LED Testing

In the board, D48 is System heartbeat lamp, D49 is user' led lamp.

The following operation carried out in HyperTerminal:

1) Control System heartbeat lamp:

```
root@DevKit8600:~# echo 1 > /sys/class/leds/sys_led/brightness
root@DevKit8600:~# echo 0 > /sys/class/leds/sys_led/brightness
```

2) Control user' led lamp:

```
root@DevKit8600:~# echo 1 > /sys/class/leds/user_led/brightness
root@DevKit8600:~# echo 0 > /sys/class/leds/user_led/brightness
```

3.8.1.2 KEYPAD Testing

Board has three users' keyboard BACK, MENU and HOME; users can and perform the following command testing:

```
root@DevKit8600:~# evtest /dev/input/event1
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys"
Supported events:
  Event type 0 (Sync)
  Event type 1 (Key)
    Event code 1 (Esc)
    Event code 59 (F1)
    Event code 102 (Home)
Testing ... (interrupt to exit)
Event: time 1233046035.953970, type 1 (Key), code 102 (Home), value 1
Event: time 1233046035.953975, ---------- Report Sync ------------
Event: time 1233046036.095752, type 1 (Key), code 102 (Home), value 0
Event: time 1233046036.095753, ---------- Report Sync ------------
Event: time 1233046037.867785, type 1 (Key), code 59 (F1), value 1
Event: time 1233046037.867788, ---------- Report Sync ------------
Event: time 1233046038.000793, type 1 (Key), code 59 (F1), value 0
Event: time 1233046038.000795, ---------- Report Sync ------------
Event: time 1233046038.854748, type 1 (Key), code 1 (Esc), value 1
Event: time 1233046038.854751, ---------- Report Sync ------------
Event: time 1233046039.022872, type 1 (Key), code 1 (Esc), value 0
```

Press CONTROL+C to quit the test. The back of the test is the same.

3.8.1.3 Touch Screen Testing

This testing requires Linux boot from NAND Flash

1) Run the command to test the touch screen.

```
root@DevKit8600: # ts_calibrate
```

Then follow the LCD prompt, click the "+" icon 5 times to complete the calibration

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

```
root@DevKit8600: # ts_test
```

Follow the LCD prompts to choose draw point, draw line test.

3.8.1.4 Backlight Testing

After entering the system, execute the following command to test the backlight.

Backlight brightness setting ranges (0-100), 100 means highest brightness. 0 means turning off the backlight brightness.

a) View the backlight brightness of the default value.

```
root@DevKit8600:~# cat /sys/class/backlight/pwm-backlight/brightness
50
```

b) set the backlight brightness to 0

```
root@DevKit8600:~# echo 0 > /sys/class/backlight/pwm-backlight/brightness
root@DevKit8600:~# cat /sys/class/backlight/pwm-backlight/brightness
0
```

At this time the backlight is turned off, the screen goes black.

c) Set the backlight brightness to 100

```
root@DevKit8600:~# echo 100 > /sys/class/backlight/pwm-backlight/brightness
root@DevKit8600:~# cat /sys/class/backlight/pwm-backlight/brightness
100
```

At this time the backlight is set to the maximum

3.8.1.5 RTC Testing

The development board contains hardware clock for save and synchronize the system time. Test can be made with the following steps:

1) Set the system time as March 22, 2012 8:00 pm

```
root@DevKit8600: # date 032220002012
Thu Mar 22 20:00:00 UTC 2012
```

2) Write the system clock into RTC

```
root@ DevKit8600: # hwclock -w
```

3) Read the RTC

```
root@ DevKit8600: # hwclock
Thu Mar 22 20:00:10 2012  0.000000 seconds
```

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

4) Restart the system, enter the following commands to renew the system clock

```
root@DevKit8600: # hwclock -s
root@DevKit8600: # date
Thu Mar 22 20:01:30 2012  0.000000 seconds
```

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

1) RTC will halt up after turn off, this is a bug for the CPU, and TI had release the corrigendum, please refer to the http://www.ti.com/lit/er/sprz360b/sprz360b.pdf
2) The DevKit8600 Development board RTC battery can use model CR1220, user needs to prepare it themselves.

3.8.1.6 TF Card Testing

1) After connecting TF card, the system will mount the file system of the TF card under the directory /media automatically:

```
root@DevKit8600:~#  cd /media/
root@DevKit8600:/media#  ls
card        hdd         mmcblk0p1   ram         union
cf          mmc1        net         realroot
```

2) Enter the following command , you can see the contents inside the TF card:

```
root@DevKit8600:/media#  ls mmcblk0p1/
flash-uboot.bin          u-boot.bin          x-load.bin.ift_for_NAND
mlo                      uImage
ramdisk.gz               ubi.img
```

3.8.1.7 USB Devices Testing

In the USB DEVICE testing, a connection line is used to connect the miniUSB interface of the development board and the USB interface at the computer end; for the computer end, the development board is recognized as a network device to realize ping communication of two ends.

1) After booting the system, a USB mini B to USB A transfer line is used to connect the development board and the computer end, wherein USBmini B interface is connected with the development board, and the USB A interface is connected with the computer end. At this time, the computer needs to be installed with Linux USB Ethernet driver. Please refer to Appendix III for detailed installation method.

2) The following commands are input at the HyperTerminal, for example:

```
root@DevKit8600:~#  ifconfig usb0 192.168.1.115
root@DevKit8600:~#  ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:26 errors:0 dropped:0 overruns:0 frame:0
          TX packets:26 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2316 (2.2 KiB)  TX bytes:2316 (2.2 KiB)

usb0      Link encap:Ethernet  HWaddr 5E:C5:F6:D4:2B:91
          inet addr:192.168.1.115  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:253 errors:0 dropped:0 overruns:0 frame:0
```

```
TX packets:43 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:35277 (34.4 KiB)  TX bytes:10152 (9.9 KiB)
```

3) After the development board is configured, please click My Computer-Network Neighborhood-Check Network Connection； a virtual network adapter will be added at the PC end.

4) Right-click virtual network adapter at the computer end, left-click “Attribute”, double-left-click to enter the “Internet Protocol (TCP/IP)” to configure the IP address of the virtual network adapter:

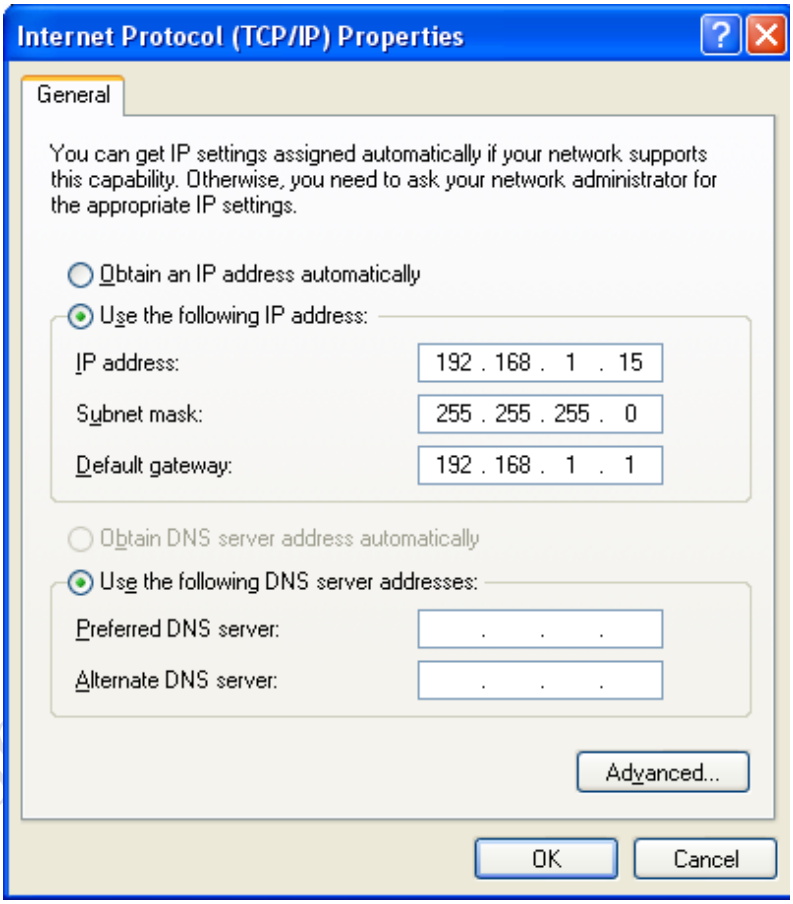


Figure 3-10

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

```
root@DevKit8600:~#  ping 192.168.1.15
PING 192.168.1.15 (192.168.1.15): 56 data bytes
64 bytes from 192.168.1.15: seq=0 ttl=128 time=0.885 ms
```

```
64 bytes from 192.168.1.15: seq=1 ttl=128 time=0.550 ms
```

6) Occurrence of above serial port information indicates that the testing is successful.

IP address of the network adapter configured in OTG cannot be the same as that of Ethernet interface.

3.8.1.8 USB HOST Testing

1) After connecting USB flash disk, the system will mount the file system of the USB flash disk under the directory /media automatically:

```
root@DevKit8600:~#  cd /media/
root@ DevKit8600:/media# ls
card          hdd          mmcblk0p1   ram          sda1
cf            mmc1         net         realroot     union
```

2) Contents in the USB flash disk will be seen after the following instruction is input:

```
root@DevKit8600:/media#  ls sda1/
flash-uboot.bin          u-boot.bin                x-load.bin.ift_for_NAND
mlo                      uImage
ramdisk.gz               ubi.img
```

3.8.1.9 Audio Testing

The board has audio input and output interface, and we have alsa-utils audio test tools in the file system, users can enter the following commands for a test:

1) Recording Test：

Plug in a microphone, you can test recording.

```
root@DevKit8600:~# arecord -t wav -c 1 -r 44100 -f S16_LE -v k
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) Playback Testing：

Plug in the headphones; you can hear what you have just recorded.

```
root@DevKit8600:~# aplay -t wav -c 2 -r 44100 -f S16_LE -v k
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
```

3.8.1.10 Network Testing

1) Users can connect the board to the router or switch and enter the following commands for a test:

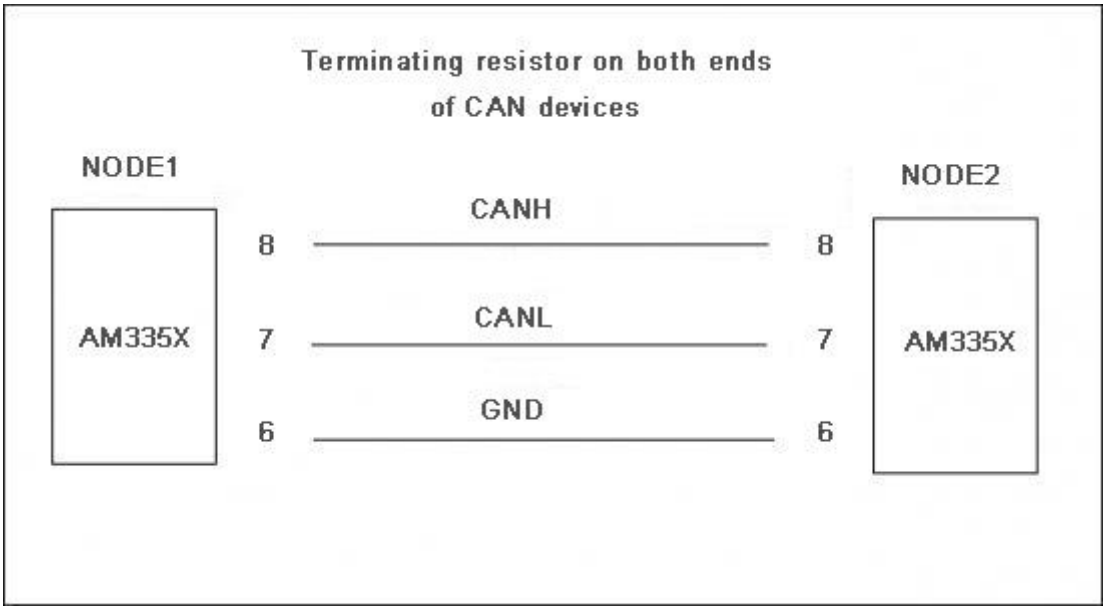
```
root@DevKit8600:~# ifconfig eth0 192.192.192.200
[root@DevKit8600 /]# ifconfig
eth0        Link encap:Ethernet  HWaddr D4:94:A1:8D:EB:25
            inet addr:192.192.192.200  Bcast:192.192.192.255
Mask:255.255.255.0
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:137 errors:0 dropped:4 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
```

```
          collisions:0 txqueuelen:1000
          RX bytes:13792 (13.4 KiB)  TX bytes:0 (0.0 B)
          Interrupt:40

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

[root@DevKit8600 /]# ping 192.192.192.170
PING 192.192.192.170 (192.192.192.170): 56 data bytes
64 bytes from 192.192.192.170: seq=0 ttl=128 time=4.486 ms
64 bytes from 192.192.192.170: seq=1 ttl=128 time=0.336 ms
```

2) Occurrence of above serial port information indicates that the testing is successful.

3.8.1.11 CAN Testing

DevKit8600 can be used as a CAN device. Accordance with the following figure shows the connection principle and with reference to the schematic to find the corresponding pin, use the cable to connect DevKit8600 CAN interface and another CAN device.



The test method as show below:

1) Both DevKit8600 and another CAN device communication baud rate is set to 125KBPS, and enable CAN device.

```
root@DevKit8600:~# canconfig can0 bitrate 125000 ctrlmode triple-sampling on
root@DevKit8600:~# canconfig can0 start
```

2) In DevKit8600 and another CAN device are respectively executed send data and receive data command, enter the following command to send data packets

```
root@DevKit8600:~# cansend can0 -i 0x10 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88
```

1) This command is a time to send data once, if resend data, need to re-enter the command.
2) To ensure that the other side of the receiving state. So the receiver side will print the information sent

3) Receive data packets

```
root@DevKit8600:~# candump can0
```

Execute the command; the terminal will print the received data.

4) Close the device.

```
root@DevKit8600:~# canconfig can0 stop
```

According to the above command, user can send and receive data each other; even more, it is allowed to set different baud rate for communication. You must close the CAN device before modify the baud rate.

The baud rate can be set as below:

25KBPS（250000）

50KBPS（50000）

125KBPS（125000）

500KBPS（500000）

650KBPS（650000）

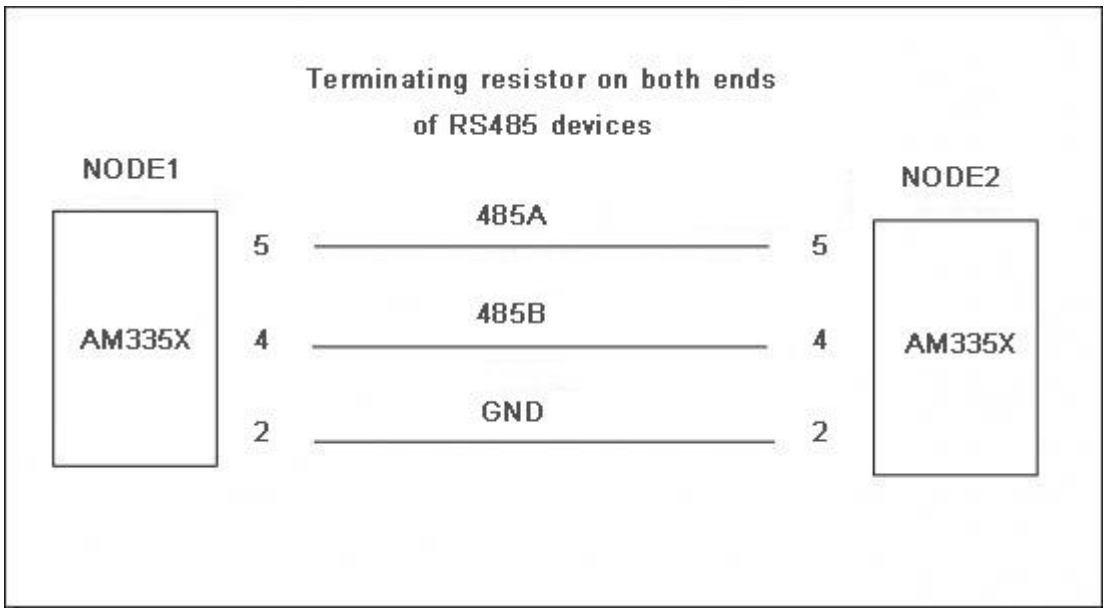
1MKBPS（1000000 ）

Use the above baud rate, the CAN device can communicate normal. For the other baud rate, User can try to set it.

The baud rates of two CAN devices need to be consistent.

3.8.1.12 RS485 Testing

DevKit8600 can be used as a RS485 device. Accordance with the following figure shows the connection principle and with reference to the schematic to find the corresponding pin, use the cable to connect DevKit8600 CAN interface and another CAN device.



RS485 communication only support half-duplex communication, ie communication at one end at the same time can only send or only receive information. Copy 485_test(linux\example\rs485_test) to TF card, Insert the TF card to Devkit8600 TF slot and execute the following command:

```
root@DevKit8600:~# cd /media/mmcblk0p1/
root@DevKit8600:/media/mmcblk0p1# ./485_test -d /dev/ttyO1 -b 115200
*********************************************
            485 TEST
*********************************************



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

One end to send information:

```
Select 1 : Send a message
Select 2 : Receive messages
>1
            Please enter the information to be sent off!
--------gpio0_13 set to 0
```

```
115200
message = 115200
len = 6
Information is sent.....
Select 3 : Stop Send
>
```

Another end to receive:

```
Select 1 : Send a message
Select 2 : Receive messages
>2
--------gpio0_13 set to 1
Select 3 : Stop Receive
>
RECV: 6 : 115200
RECV: 6 : 115200
RECV: 6 : 115200
RECV: 6 : 115200
```

To stop receiving information:

```
Select 3 : Stop Receive
>3
```

3.8.1.13 Serial port testing

Because UART1 (Compatible with RS485) and UART3 (Compatible with wifi/Bluetooth) are occupied, this chapter will show you how to test UART2.

Short circuiting UART2_RX_3V3 pin and UART2_TX_3V3 pin, copy the file uart_test (linux\example\uart_test) to the TF card, insert the TF card to the Devkit8600 TF slot, and input the commands as below:

```
root@DevKit8600:~# cd /media/mmcblk0p1/
root@DevKit8600:/media/mmcblk0p1# ./uart_test -d /dev/ttyO2 -b 115200
```

Print the following information, said the test is successful.

```
/dev/ttyO2 SEND: 1234567890
```

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

3.8.1.14 WIFI Testing

1) Test the connection of non-encrypted wireless router

a) Enable wifi

```
root@DevKit8600:~# ifconfig wlan0 up
wl1271: firmware booted (Rev 6.1.5.50.74)
```

b) Scan wifi router

```
root@DevKit8600:~# iwlist wlan0 scan
wlan0        Scan completed :
             Cell 01 - Address: 94:0C:6D:17:0A:BC
                       Channel:1
                       Frequency:2.412 GHz (Channel 1)
                       Quality=44/70    Signal level=-66 dBm
                       Encryption key:off
                       ESSID:"TIOP"
                       Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s
```

```
                    Mode:Master
                    Extra:tsf=0000000023f51ee7
                    Extra: Last beacon: 560ms ago
                    IE: Unknown: 000454494F50
                    IE: Unknown: 010482848B96
                    IE: Unknown: 030101
……..
```

c) Connect to wifi router

```
root@DevKit8600:~# iwconfig wlan0 essid TIOP
Association completed.
```

d) Testing：

```
root@DevKit8600:~# ifconfig wlan0 192.192.192.215
root@DevKit8600:~# ping 192.192.192.90
PING 192.192.192.90 (192.192.192.90): 56 data bytes
64 bytes from 192.192.192.90: seq=0 ttl=64 time=32.260 ms
64 bytes from 192.192.192.90: seq=1 ttl=64 time=20.662 ms
64 bytes from 192.192.192.90: seq=2 ttl=64 time=20.419 ms
```

2) Test the connection of the WEP encrypted wireless router

a) Enable wifi：

```
root@DevKit8600:~# ifconfig wlan0 up
wl1271: firmware booted (Rev 6.1.5.50.74)
```

b) Scan wifi router

```
root@DevKit8600:~# iwlist wlan0 scan
wlan0        Scan completed :
             Cell 01 - Address: 94:0C:6D:17:0A:BC
                       Channel:1
                       Frequency:2.412 GHz (Channel 1)
                       Quality=19/70    Signal level=-91 dBm
                       Encryption key:on //表示已加密
                       ESSID:"TIOP"
```

```
Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s
Mode:Master
Extra:tsf=000000141698b596
Extra: Last beacon: 33540ms ago
IE: Unknown: 000454494F50
IE: Unknown: 010482848B96
IE: Unknown: 030101
```

c) Connect to wifi router

```
root@DevKit8600:~# iwconfig wlan0 essid TIOP key s:abcde
root@DevKit8600:~# ifconfig wlan0 down
wl1271: down
root@DevKit8600:~# ifconfig wlan0 up
wl1271: firmware booted (Rev 6.1.5.50.74)
Association completed.
```

1) **iwconfig wlan0 essid TIOP key s:abcde** command indicates that the connected wireless router name is TIOP, the KEY format is ASCII characters, the KEY is “abcede”. connection to the wireless router is "TIOP", if the key format of WEP router used is hexadecimal characters , the key is: 0123456789, WEP encrypted wireless router connect command is: **iwconfig wlan0 ESSID TIOP key 0123-4567-89**
2) Before the test, you must use the **ifconfig wlan0 down** to turn off the WIFI device, use the **ifconfig wlan0 up** command to turn on
3) Wireless router with encryption cannot connect, in addition to WEP Wireless router

d) Testing

```
root@DevKit8600:~# ifconfig wlan0 192.192.192.216
root@DevKit8600:~# ping 192.192.192.90
```

```
PING 192.192.192.90 (192.192.192.90): 56 data bytes
64 bytes from 192.192.192.90: seq=0 ttl=64 time=32.260 ms
64 bytes from 192.192.192.90: seq=1 ttl=64 time=20.662 ms
```

3.8.1.15 BT Testing

1) Enable BT:

```
root@DevKit8600:~# insmod /lib/modules/3.1.0/kernel/drivers/bt_enable/gpio_en.ko

Gpio value is :23
WL1271: BT Enable
```

2) Configure BT

```
root@DevKit8600:~# hciattach /dev/ttyO3 texas 115200
Found a Texas Instruments' chip!
Firmware file : /lib/firmware/TIInit_7.2.31.bts
Loaded BTS script version 1
texas: changing baud rate to 921600, flow control to 1
Device setup complete
```

3) Testing：

```
root@DevKit8600:~# hciconfig hci0 up
root@DevKit8600:~# hcitool scan
Scanning ...
        00:12:FE:B7:75:A0       Lenovo-TD80t
```

3.8.1.16 CDMA8000-U module

If the camera modules are from Embest then you can download the module material from below link:

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

3.8.1.17 WCDMA8000-U module

If the camera modules are from Embest then you can download the module material from below link:

http://www.timll.com/chinese/uploadFile/WCDMA8000-110113.zip

3.8.2 Demo

3.8.2.1 Demonstration of Android System

DevKit8600 provides Android system demonstration, please follow below steps:

1) Copy all files under the directory CD\linux\demo\android\image to the TF card, rename the cooresponding file uImage_xx as uImage according to the size of the LCD you have.
2) Insert the TF card in the development card and power it on; the HyperTerminal will display the following information:

```
CCCCCCCC
U-Boot SPL 2011.09-svn (May 22 2012 - 11:19:00)
Texas Instruments Revision detection unimplemented
Booting from MMC...
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img



U-Boot 2011.09-svn (May 24 2012 - 11:17:39)


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

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

38167 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x0, size 0x9517
 38167 bytes written: OK
reading flash-uboot.img

230148 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x80000, size 0x38304
 230148 bytes written: OK
reading uImage

2709040 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x280000, size 0x295630
 2709040 bytes written: OK
reading ubi.img

72744960 bytes read
SW ECC selected
```

```
NAND write: device 0 offset 0x780000, size 0x4560000
Skip bad block 0x03620000
```

3) LED lamp on the board will flicker to prompt after programming is finished, at this time, please pull the TF card out.

4) Power it on again and boot to enter the android operating system.

3.8.2.2 Demonstration of TISDK System

1) Format a TF card into two partitions (please refer to Appendix IV for detailed instructions)

2) Put CD-COM in the drive on PC and Insert the TF card into PC, and then enter the following commands in the terminal window of Ubuntu according to the your LCD sreen size.

For 4.3-inch LCD

```
cp /media/cdrom/linux/demo/tisdk/image/MLO /media/LABEL1
cp /media/cdrom/linux/demo/tisdk/image/u-boot.img /media/LABEL1
cp /media/cdrom/linux/demo/tisdk/image/uImage_4.3 /media/LABEL1/uImage
rm -rf /media/LABEL2/*
sudo tar jxvf linux/demo/dvsdk/image/tisdk-rootfs-am335x-evm.tar.gz -C /media/LABEL2
sync
umount /media/LABEL1
umount /media/LABEL2
```

For 7-inch LCD

```
cp /media/cdrom/linux/demo/tisdk/image/MLO /media/LABEL1
cp /media/cdrom/linux/demo/tisdk/image/u-boot.img /media/LABEL1
cp /media/cdrom/linux/demo/tisdk/image/uImage_7 /media/LABEL1/uImage
rm -rf /media/LABEL2/*
sudo tar xvf
/media/cdrom/linux/demo/tisdk/image/tisdk-rootfs-am335x-evm.tar.gz -C
```

/media/LABEL2

sync

umount /media/LABEL1

umount /media/LABEL2

For LVDS (10.4”-inch)

cp /media/cdrom/linux/demo/tisdk/image/MLO /media/LABEL1

cp /media/cdrom/linux/demo/tisdk/image/u-boot.img /media/LABEL1

cp /media/cdrom/linux/demo/tisdk/image/uImage_LVDS /media/LABEL1/uImage

rm -rf /media/LABEL2/*

sudo tar xvf

/media/cdrom/linux/demo/tisdk/image/tisdk-rootfs-am335x-evm.tar.gz -C

/media/LABEL2

sync

umount /media/LABEL1

umount /media/LABEL2

3) After the above commands are executed, please power on the board and then hit any key on your keyboard when you see the prompt information "Hit any key to stop autoboot:" to enter u-boot mode as shown below:

CCCCCCCC

U-Boot SPL 2011.09-svn (May 03 2012 - 10:49:04)

Texas Instruments Revision detection unimplemented

Booting from MMC...

OMAP SD/MMC: 0

reading u-boot.img

reading u-boot.img

U-Boot 2011.09-svn (May 03 2012 - 10:49:04)

I2C: ready

DRAM: 512 MiB

WARNING: Caches not enabled

Did not find a recognized configuration, assuming General purpose EVM in Profile 0 with Daughter board

NAND: HW ECC Hamming Code selected

512 MiB

MMC: OMAP SD/MMC: 0

*** Warning - bad CRC, using default environment

Net: cpsw

Hit any key to stop autoboot: 0 **(input any key here)**

4) Enter the following commands. The system will continue the booting process.

Devkit8600# **setenv bootargs console=ttyO0,115200n8 earlyprintk root=/dev/mmcblk0p2 rw rootfstype=ext3 rootwait**

Devkit8600# **setenv bootcmd 'mmc rescan;fatload mmc 0 80300000 uImage;bootm 80300000'**

Devkit8600# **saveenv**

Saving Environment to NAND...

Erasing Nand...

Erasing at 0x260000 -- 100% complete.

Writing to Nand... done

Devkit8600# **boot**

reading uImage

2949384 bytes read

Booting kernel from Legacy Image at 80300000 ...

Image Name: Linux-3.1.0

Image Type: ARM Linux Kernel Image (uncompressed)

Data Size: 2949320 Bytes = 2.8 MiB

Load Address: 80008000

```
    Entry Point:    80008000
    Verifying Checksum ... OK
    Loading Kernel Image ... OK
OK

Starting kernel ...
……
Arago Project http://arago-project.org am335x-evm ttyO0

Arago 2011.09 am335x-evm ttyO0

am335x-evm login: root
```

5) TISDK file system has some preinstalled application programs, which are based on QT UI; it can be executed by the user easily.

3.9 The Development of Application

This section mainly introduces to development of application programs, and illustrates the general process of development of application programs with cases.

Development example of LED application program

1) To Edit code

led_acc.c source code: control LED lamps on the development board to flicker in a way of accumulator.

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

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

int main(int argc, char *argv[])
{
        int f_led1, f_led2;
        unsigned char i = 0;
        unsigned char dat1, dat2;
        if((f_led1 = open(LED1, O_RDWR)) < 0){
            printf("error in open %s",LED1);
            return -1;
        }
        if((f_led2 = open(LED2, O_RDWR)) < 0){
            printf("error in open %s",LED2);
            return -1;
        }

        for(;;){
            i++;
            dat1 = i&0x1 ? '1':'0';
            dat2 = (i&0x2)>>1 ? '1':'0';
            write(f_led1, &dat1, sizeof(dat1));
            write(f_led2, &dat2, sizeof(dat2));
            usleep(300000);
        }
}
```

2) **To Cross-compile**

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

3) Download and run

Upload to the development board system through TF card, USB flash disk or network,

enter the directory with the led_acc file, input the following commands and press Enter, to run led_acc in the background.

```
./led_acc &
```

Chapter 4 Windows Embedded Compact 7 Operating System

4.1 Introduction

This section mainly introduces Devkit8600 system and application development of Windows Embedded Compact 7, as well as software resources in disc, software features, establishment of development environment, and how to sysgen and build BSP (board support package) and so on.

4.2 Software Resources

BSP (Board Support Package)

CD\WINCE700\BSP\AM33x_BSP.rar

CD\WINCE700\BSP\COMMON_TI_V1.rar

CD\WINCE700\BSP\3rdParty.rar

CD\WINCE700\BSP\PowerVR.rar

Windows Embedded Compact 7 sample project

CD\WINCE700\project\AM335X_OS

Sample application

CD\WINCE700\app\

Pre-compile image

CD\WINCE700\Image\

MLO	First bootloader for TF card boot
xldrnand.nb0	First bootloader for NAND flash boot
Ebootsd.nb0	Second bootloader for TF card boot

Ebootnd.nb0	Second bootloader for NAND flash boot
Nk.bin	WinCE runtime image

4.3 Software Features

Resources of BSP:

Catalog	**Item**	**Source code / binary**
X-Loader (First boot loader)	NAND	Source
	SD	Source
EBOOT (Second boot loader)	NAND	Source
	SD	source
OAL	Boot parameter	Source
	KILT(EMAC)	Source
	Serial debug	Source
	REBOOT	Source
	Watchdog	Source
	RTC	Source
	Kernel profiler	Source
	System timer	Source
	Interrupt controller	Source
	MMU	Source
Driver	NLED driver	Source
	GPIO/I2C/SPI/MCASP driver	Source
	Serial port driver	Source
	Audio driver	Source
	NAND driver	Source
	Display driver	Source
	TOUCH driver	Source

	SD/MMC/SDIO driver	Source
	EMAC driver	Source
	USB OTG driver	Source
	GPIO keyboard driver	Source
	DMA driver	Source
	Backlight driver	Source
	Battery driver	Source
	RPU driver	Source
SDK	powerVR DDK & SDK	Binary & Source

Table 4-1

4.4 System Development

4.4.1 Installation of IDE(Integrated Development Environment)

Please install items below to windows XP by step:

1) Visual Studio 2008
2) Visual Studio 2008 SP1
3) Windows Embedded Compact 7
4) Windows Embedded Compact 7 Updates
5) ActiveSync 4.5

CD does not provide Windows Embedded Compact 7 development environment tools, please down from: http://www.microsoft.com/download/en/default.aspx

4.4.2 Extract BSP and project files to IDE

The following preparations should be made:

1) Extract [CD\WINCE700\BSP\AM33x_BSP.rar] to [C:\WINCE700\PLATFORM] directory.
2) Extract [CD\WINCE700\BSP\ COMMON_TI_V1.rar] to [C:\WINCE700\PLATFORM\COMMON\SRC\SOC].
3) Extract [CD\WINCE700\BSP\3rdParty.rar] to [C:\WINCE700].
4) Extract [CD\WINCE700\BSP\powerVR.rar] to [C:\WINCE700\public].
5) Copy directory [CD\WINCE700\project\AM335X_OS] to [C:\WINCE700\OSDesigns] directory.

The default installation path of the Windows Embedded Compact 7 in this context is [C:\WINCE700].

4.4.3 Sysgen & Build BSP

Below are the steps given for Sysgen and BSP build:

1) Open the existing project file AM335X_OS.sln locates in [C:\WINCE700\OSDesigns\AM335X_OS].
2) Click [Build-> Build Solution] in vs2008 to sysgen and build BSP.
3) Images including MLO, EBOOTSD.nb0, NK.bin will be created after sysgen phase and build phase finished successfully, Copy the files MLO, EBOOTSD.nb0, and NK.bin Locate in [C:\WINCE700\OSDesigns\AM335X_OS\AM335X_OS\RelDir\AM33X_BSP_ARMV7_Release] to the TF card.
4) Insert the TF card to the device and boot the device for a test.

4.4.4 Driver Introduction

Source code path of all drivers in BSP:

NLED driver	BSP\AM33X_BSP\SRC\DRIVERS\NLED
GPIO	BSP\AM33X_BSP\SRC\DRIVERS\GPIO BSP\COMMON_TI_V1\COMMON_TI_AMXX\GPIO
I2C	BSP\COMMON_TI_V1\COMMON_TI_AMXX\OAL\OALI2C
SPI	BSP\COMMON_TI_V1\COMMON_TI_AMXX\SPI BSP\AM33X_BSP\SRC\DRIVERS\MCSPI
MCASP driver	BSP\COMMON_TI_V1\COMMON_TI_AMXX\MCASP
Serial port driver	BSP\COMMON_TI_V1\COMMON_TI_AMXX\SERIAL BSP\AM33X_BSP\SRC\DRIVERS\UART
Audio driver	BSP\AM33X_BSP\SRC\DRIVERS\WAVEDEV2
NAND driver	BSP\AM33X_BSP\SRC\DRIVERS\BLOCK BSP\COMMON_TI_V1\COMMON_TI_AMXX\BLOCK
Display driver	BSP\COMMON_TI_V1\COMMON_TI_AMXX\DSS_Netra BSP\AM33X_BSP\SRC\DRIVERS\DISPLAY
TOUCH driver	BSP\AM33X_BSP\SRC\DRIVERS\TOUCH
SD/MMC/SDIO driver	BSP\AM33X_BSP\SRC\DRIVERS\SDHC BSP\COMMON_TI_V1\COMMON_TI_AMXX\SDHC

	BSP\COMMON_TI_V1\COMMON_TI\SDHC
EMAC driver	BSP\COMMON_TI_V1\AM33X\CPSW3Gminiport BSP\AM33X_BSP\SRC\DRIVERS\EMAC
USB OTG driver	BSP\AM33X_BSP\SRC\DRIVERS\USB BSP\COMMON_TI_V1\AM33X\USB
GPIO keyboard driver	BSP\AM33X_BSP\SRC\DRIVERS\KEYPAD
Backlight driver	BSP\AM33X_BSP\SRC\DRIVERS\BACKLIGHT
Battery driver	BSP\ AM33X_BSP \SRC\DRIVERS\BATTERY
PRU driver	BSP\COMMON_TI_V1\AM33X\PRU BSP\ AM33X_BSP\SRC\DRIVERS\PRU
DMA driver	BSP\ AM33X_BSP\SRC\DRIVERS\EDMA BSP\COMMON_TI_V1\COMMON_TI_AMXX\EDMA

Table 4-2

If the user wants to refer to more Windows Embedded Compact 7 driver development, please refer to the specific reference document of the PB7.0:

Start->

All programs->

Microsoft Visual Studio 2008->

Microsoft Visual Studio 2008 Document->

Content(C)->

Windows Embedded Compact 7->Device Driver.

4.5 Update system image

Devkit8600 supports boot-up from TF card and NAND; this section will respectively introduce two different system update ways.

4.5.1 Update TF card image

1) Format TF card

HP USB Disk Storage Format Tool 2.0.6 is recommended:

The software is downloading from

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

a) Insert TF card into the card reader in PC.

b) Open the HP USB Disk Storage Format Tool, the following steps will show in detail.

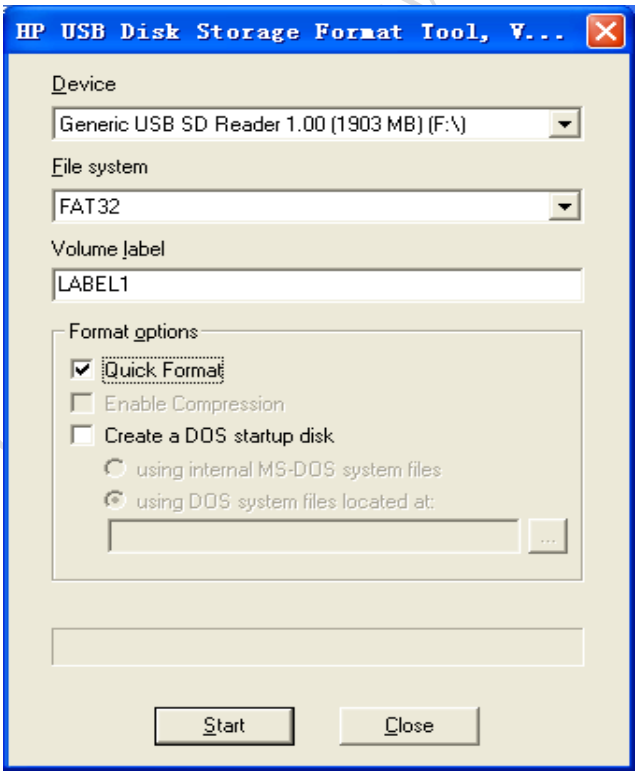


Figure 4-1

c) Select “FAT32”.

d) Click “Start”.

e) When formatting is completed, click “OK”.

HP USB Disk Storage Format Tool will clear partitions of the TF card. Please use the formatting software provided in the computer system

2) Copy runtime image

- Copy **MLO, EBOOTSD.nb0 and NK.bin** image files Locate in CD\WINCE700\image to the TF card.

3) Boot system

Insert TF card and reboot the system. And then, the system boots from TF card. Press the space button enter EBOOT menu to select boot device and LCD module display output, follow these steps:

a) Enter EBOOT menu

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

jumping to ebootsd image

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

Texas Instruments Windows CE EBOOT for AM33x, Built Jul 27 2012 at 11:25:53
```

EBOOT Version 0.0.1, BSP BSP_WINCE_ARM_A8 02.30.00.03

AHCLKX pinmux:0

AHCLKX CTRL:0x8001

pin function:0x0

pin dir:0x8000000

TI AM33X

ecc type:3

System ready!

Preparing for download...

INFO: Predownload....

Checking bootloader blocks are marked as reserved (Num = 18)

BOOT_CFG_SIGNATURE is different, read -1, expect 1111705159

WARN: Boot config wasn't found, using defaults

INFO: SW3 boot setting: 0x04

IsValidMBR: MBR sector = 0x480 (valid MBR)

OpenPartition: Partition Exists=0x1 for part 0x20.

>>> Forcing cold boot (non-persistent registry and other data will be wiped) <<<

e0311800 56e4 -> 0 18 31 e0 e4 56

e0311800 57e4 -> 0 18 31 e0 e4 57

Hit space to enter configuration menu [56] 5...（**Press the space key to enter EBOOT menu**）

b) Type [2]->[2] select boot from TF card

Main Menu

---

[1] Show Current Settings

[2] Select Boot Device

[3] Select KITL (Debug) Device

[4] Network Settings

[5] SDCard Settings

[6] Set Device ID

[7] Save Settings

[8] Flash Management

[9] Enable/Disable OAL Retail Messages

[a] Select Display Resolution

[b] Select OPP Mode

[0] Exit and Continue

Selection: **2**

Select Boot Device

---

[1] Internal EMAC

[2] NK from SDCard FILE

[3] NK from NAND

[0] Exit and Continue

Selection (actual Internal EMAC): **2**

Boot device set to NK from SDCard FILE

c) Type [a] to select the LCD/LVDS module display output

Main Menu

---

[1] Show Current Settings

[2] Select Boot Device

[3] Select KITL (Debug) Device

[4] Network Settings

[5] SDCard Settings

[6] Set Device ID

[7] Save Settings

[8] Flash Management

[9] Enable/Disable OAL Retail Messages

[a] Select Display Resolution

[b] Select OPP Mode

[0] Exit and Continue

Selection: **a**

Select Display Resolution

---

[1] LCD 480x272 60Hz //For 4.3-inch LCD

[2] DVI 640x480 60Hz(N/A)

[3] DVI 640x480 72Hz(N/A)

[4] LCD 800x480 60Hz //For 7-inch LCD

[5] DVI 800x600 60Hz(N/A) //For LVDS

[6] DVI 800x600 56Hz(N/A)

[7] VGA 1024x768 60Hz //For VGA

[8] DVI 1280x720 60Hz(N/A)

[0] Exit and Continue Selection (actual LCD 480x272 60Hz): **4**

d) Type [0] to boot the system

Main Menu

---

[1] Show Current Settings

[2] Select Boot Device

[3] Select KITL (Debug) Device

[4] Network Settings

[5] SDCard Settings

[6] Set Device ID

[7] Save Settings

[8] Flash Management

[9] Enable/Disable OAL Retail Messages

[a] Select Display Resolution

[b] Select OPP Mode

[0] Exit and Continue

Selection: **0**

mode = 3

LcdPdd_LCD_GetMode:3

mode = 3

LcdPdd_LCD_Initialize:3

OEMPreDownload: Filename nk.bin

Init HW: controller RST

SDCARD: requested speed 1000000, actual speed 1000000

SDCARD: requested speed 25000000, actual speed 19200000

BL_IMAGE_TYPE_BIN

+OEMMultiBinNotify(0x8feb24d8 -> 1)

Download file information:

---

[0]: Address=0x80002000 Length=0x03c9e9bc Save=0x80002000

```
Download file type: 1
+OEMIsFlashAddr(0x80002000) g_eboot.type 1
.....
.....
.....rom_offset=0x0.
..ImageStart = 0x80002000, ImageLength = 0x3c9e9bc, LaunchAddr = 0x8000b6a0

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

Windows CE Kernel for ARM (Thumb Enabled)
CPU CP15 Control Register = 0xc5387f
CPU CP15 Auxiliary Control Register = 0x42
I2C EEPROM returned wrong magic value 0xffffffff
+OALTimerInit(1, 24000, 200)
--- High Performance Frequecy is 24 MHz---
```

4.5.2 Update NAND Flash image

1) Format TF card

Please refer to contents of 4.5.1 Update of TF card –> 1) Format TF card.

2) Copy runtime image

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

3) flashing image file

Insert TF card and reboot the system. At this time, the system boots from TF card. The HyperTerminal will display boot message, you can press [SPACE] to enter the EBOOT

menu. Flashing image to NAND flash according to the following steps:

- Press [8] to enter the Flash menu.
- Press[9]->[4]->[A], [9]->[3]->[B] and[9]->[2]->[C] to write XLDR, EBOOT and NK images respectively.
- Then press [0] to return to main menu, and respectively press [2], [3] Select boot from NAND flash, then type [A] select the LCD,VGA or LVDS output mode [7] and [y] to save the Boot setting.

Unplug TF card and then reboot the system, and then the system will boot from NAND Flash.

4.6 Instructions for use

4.6.1 How to use openGL ES demo

1) Select PowerVR items in catalog items view in VS2008 as below:

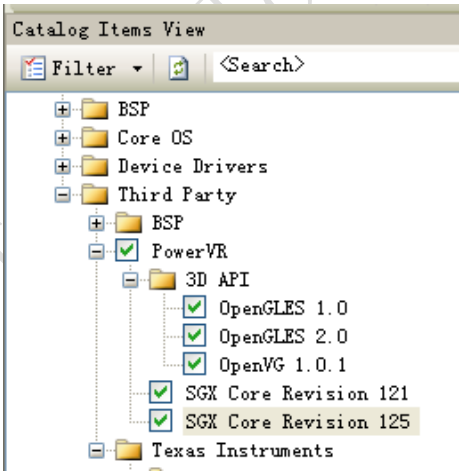


Figure 4-2

2) Click [Build-> Build Solution] in VS2008 menu, after sysgen and build BSP finished, replace the nk.bin locate in TF card with the newly generated nk.bin

3) Copy C:\WINCE700\PUBLIC\PowerVR\oak\target\Rev125\ARMV4I\retail*.exe to Devkit8600 windows embedded compact 7 system. And double click the demos to test.

4.7 Application Development

This chapter introduces how to develop Windows Embedded Compact 7 application program in Devkit8600.

4.7.1 Application program interfaces and examples

API used for development of Devkit8600 application programs employs Microsoft Windows Embedded Compact 7 standard application program interface definition, Devkit8600 only expands interface definition of GPIO based on standard API. Please refer to the CD\WINCE700\app\GPIOAppDemo to see how to control the GPIO pin status.

Please check relative Help documents of MSDN Windows Embedded Compact 7 API for Windows Embedded Compact 7 standard application program interface definition.

4.7.2 GPIO application program interfaces and examples

GPIO device name is L"GIO1:" Extend the DeviceIoControl interface definition corresponding device IOCTL code includes:

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

Table 4-3

Operation example is showed below:

1) Open GPIO 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 of GPIO

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

Set the working mode of GPIO:

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

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, including:

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

Table 4-4

3) The operation of GPIO Pin:

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

Output high level:

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

Output low level:

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

Read the pin state

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

"id" is GPIO pin number, "pinSate" returns to pin state

4) Other optional operation

Read the corresponding IRQ number of GPIO pin

```
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

5) Close GPIO device

```
CloseHandle (hFile);
```

1) Definition of GPIO pin: 0~127 MPU Bank0~3 GPIO pin.
2) GPIO pin 0~127 has to be configured as GPIO under AM33X_BSP/SRC/inc/bsp_padcfg.h when refer as a GPIO pin.

Appendix

Appendix I Hardware Dimensions

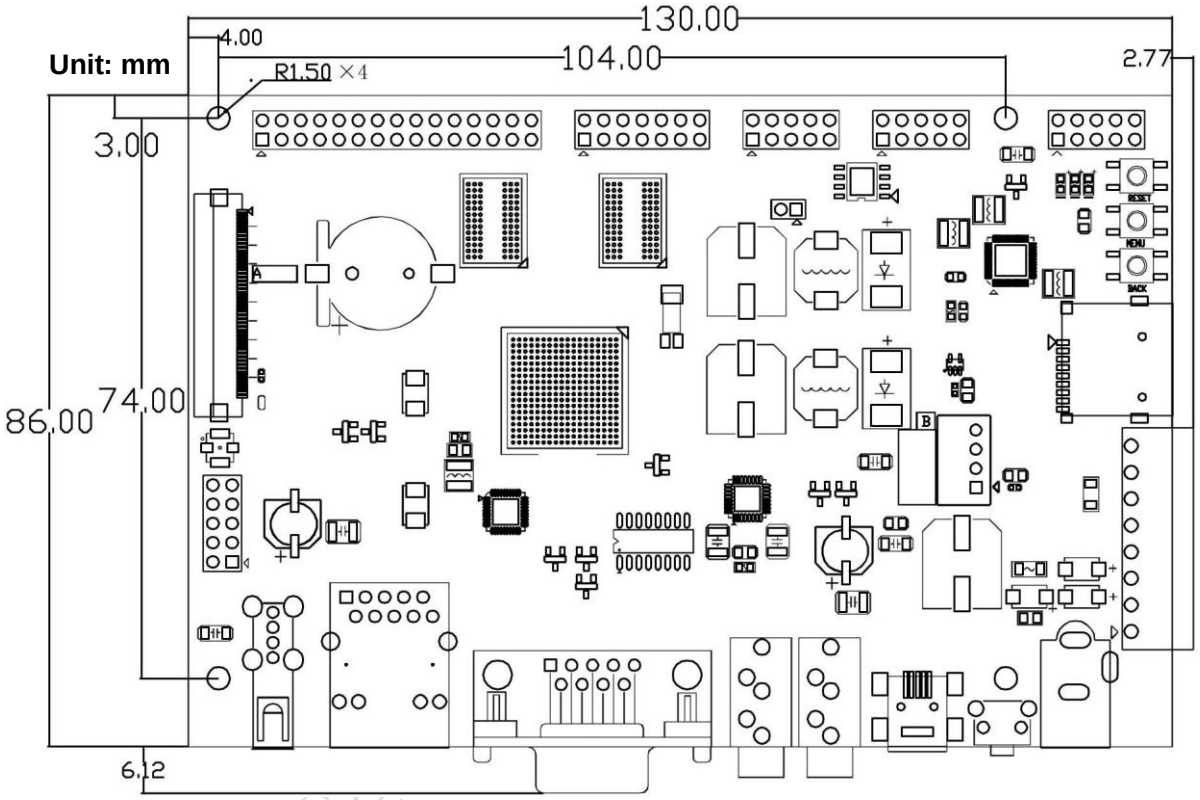


Figure Appendix 1-1 Hardwar Dimensions Diagram

Appendix II The Installation Of Ubuntu

Installing Ubuntu in Windows using VirtualBox

The screenshots in this tutorial use Ubuntu 11.04, but the same principles apply also to Ubuntu 10.10, 11.04, and any future version of Ubuntu. Actually, you can install pretty much any Linux distribution this way.

VirtualBox allows you to run an entire operating system inside another operating system. Please be aware that you should have a minimum of 512 MB of RAM. 1 GB of RAM or more is recommended.

Installation Process

1. Download software

Before installing Ubuntu, you must get VirtualBox software and Ubuntu disk image (ISO file).

Available in the VirtualBox download page VirtualBox program VirtualBox-4.0.10-72479-Win.exe.

In the Ubuntu download page to get Ubuntu disk image ubuntu-11.04-desktop-i386.iso.

2. Create New Virtual machine

Figure Appendix 2-1

After you launch VirtualBox from the Windows Start menu, click on New to create a new virtual machine. When the New Virtual Machine Wizard appears, click Next.

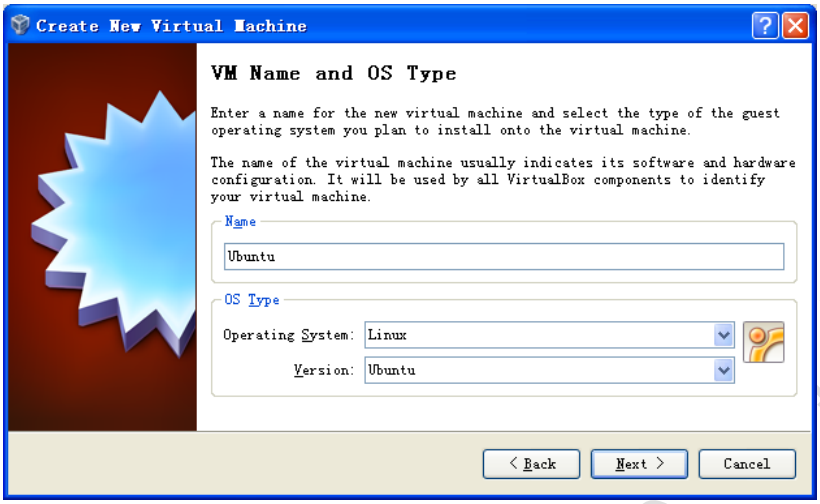


Figure Appendix 2-2

You can call the machine whenever you want. If you're installing Ubuntu, it makes sense to call it Ubuntu, I guess. You should also specify that the operating system is **Linux.**

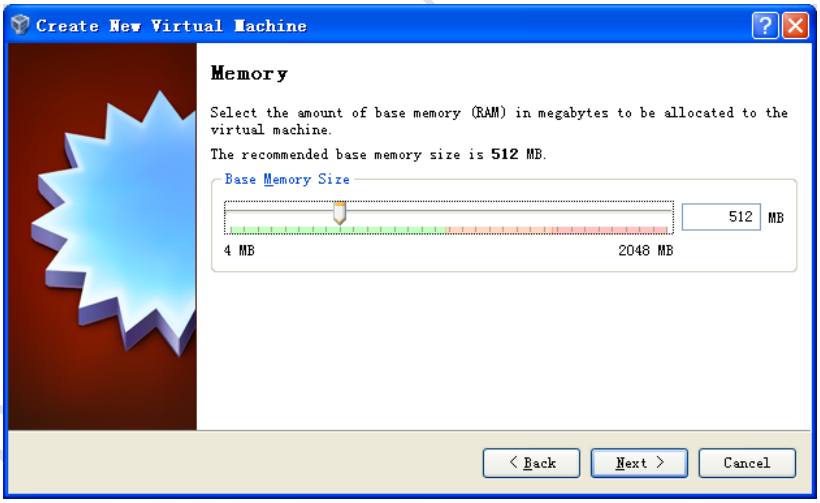


Figure Appendix 2-3

VirtualBox will try to guess how much of your memory (or RAM) to allocate for the virtual machine. If you have 1 GB or less of RAM, I would advise you stick with the recommendation. If, however, you have over 1 GB, about a quarter PC RAM or less should be fine. For example, if you have 2 GB of RAM, 512 MB is fine to allocate. If you have 4 GB of RAM, 1 GB is fine to allocate. If you have no idea what RAM is or how much

of it you have, just go with the default.

Click **Next**.

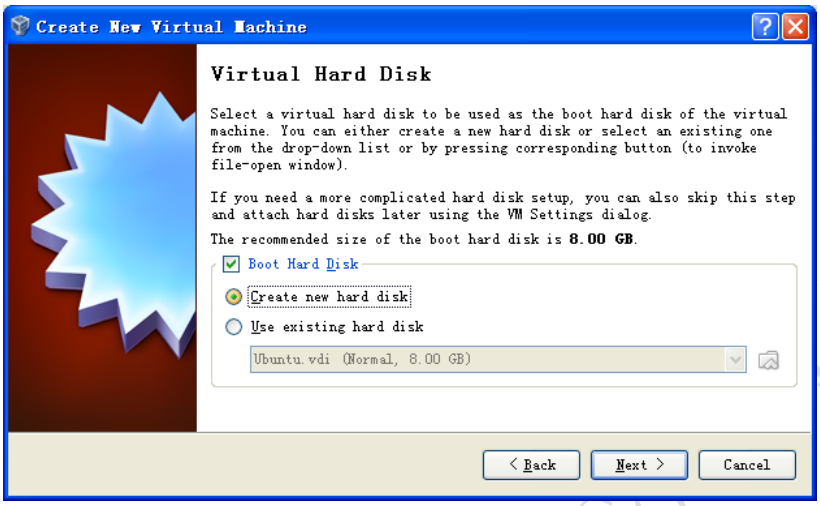


Figure Appendix 2-4

If this is your first time using VirtualBox (which it probably is if you need a tutorial on how to use it), then you do want to create new hard disk and then click **Next**.

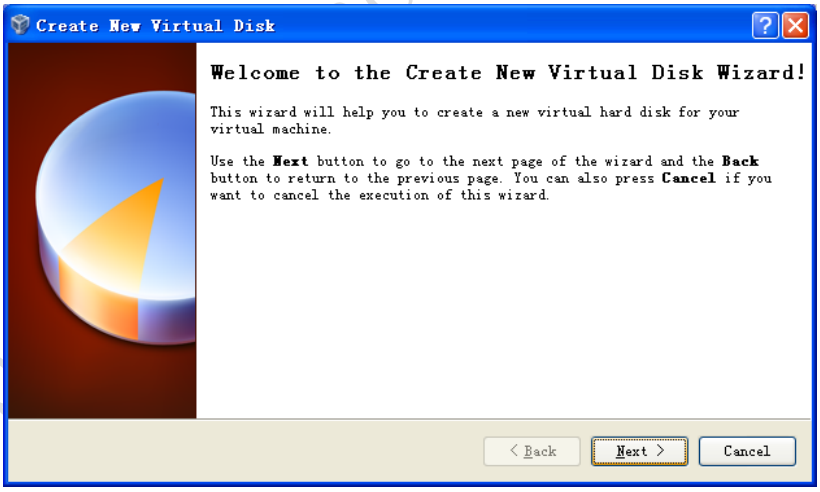


Figure Appendix 2-5

Click **Next** again.

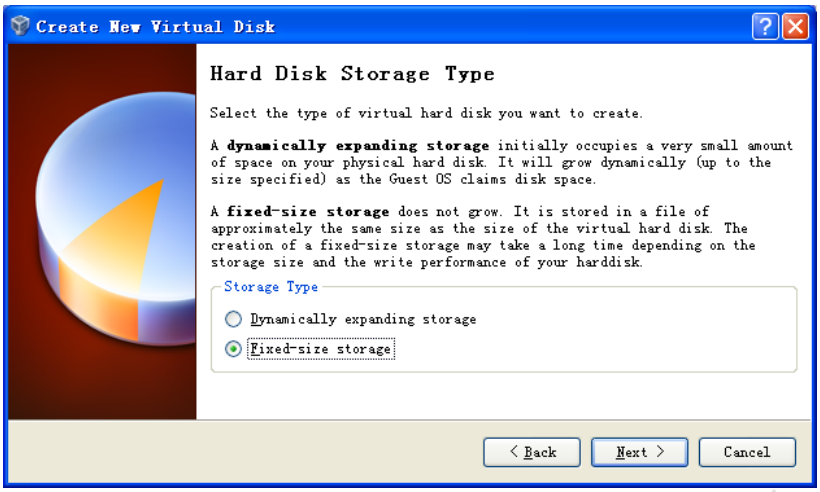


Figure Appendix 2-6

Theoretically, a dynamically expanding virtual hard drive is best, because it'll take up only what you actually use. I have come upon weird situations, though, when installing new software in a virtualized Ubuntu, in which the virtual hard drive just fills up instead of expanding. So I would actually recommend picking **Fixed-size storage**.

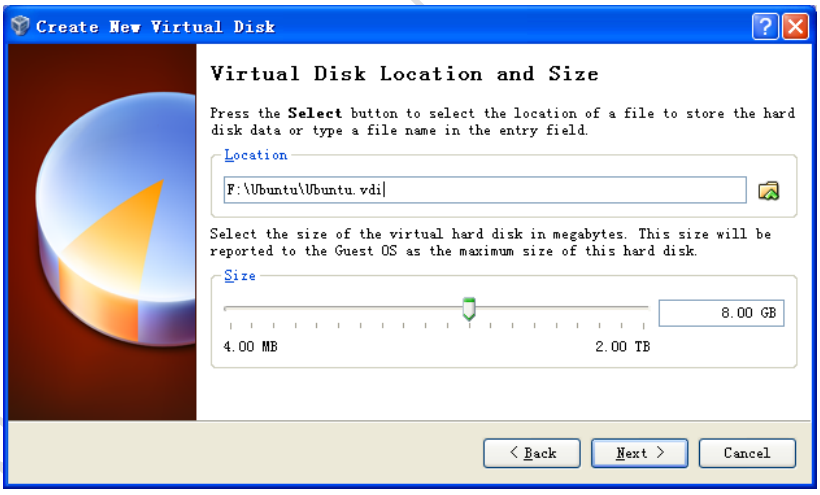


Figure Appendix 2-7

Ubuntu's default installation is less than 8 GB. If you plan on adding software or downloading large files in your virtualized Ubuntu, you should tack on some buffer.

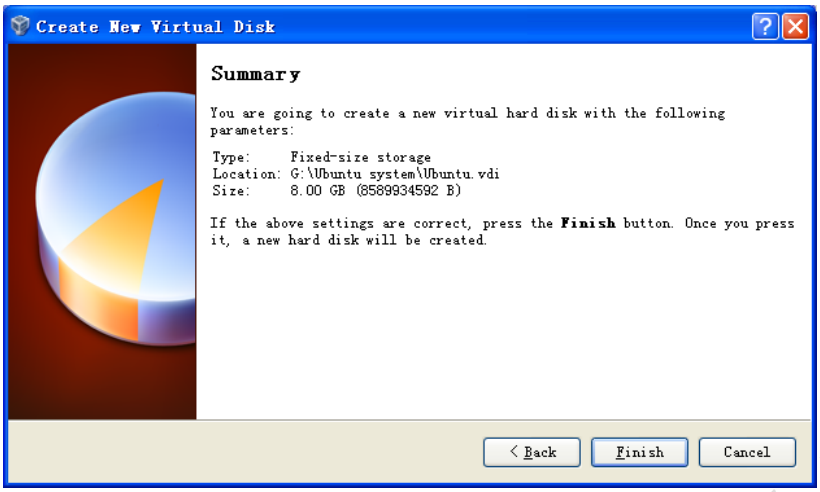


Figure Appendix 2-8

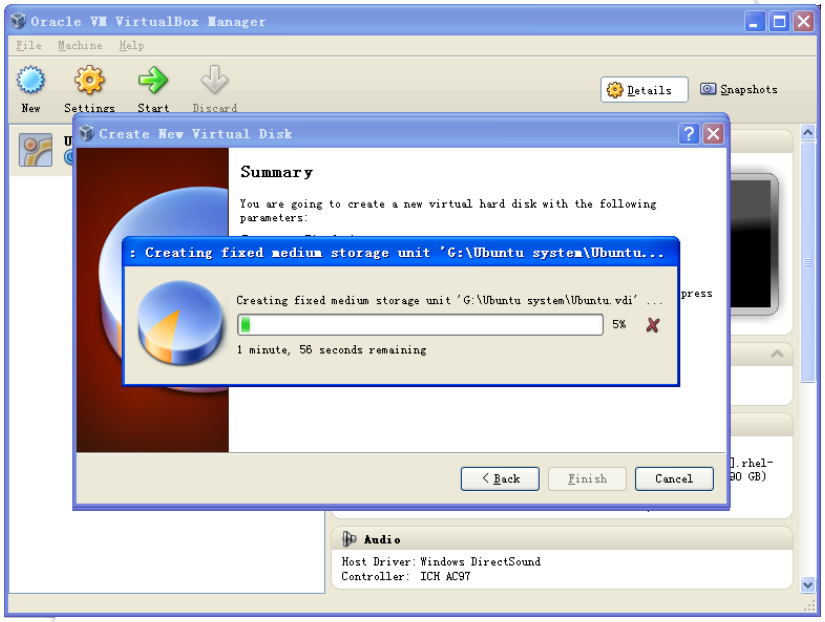


Figure Appendix 2-9

Click **Finish** and wait for the virtual hard drive to be created. This is actually just a very large file that lives inside of your Windows installation.

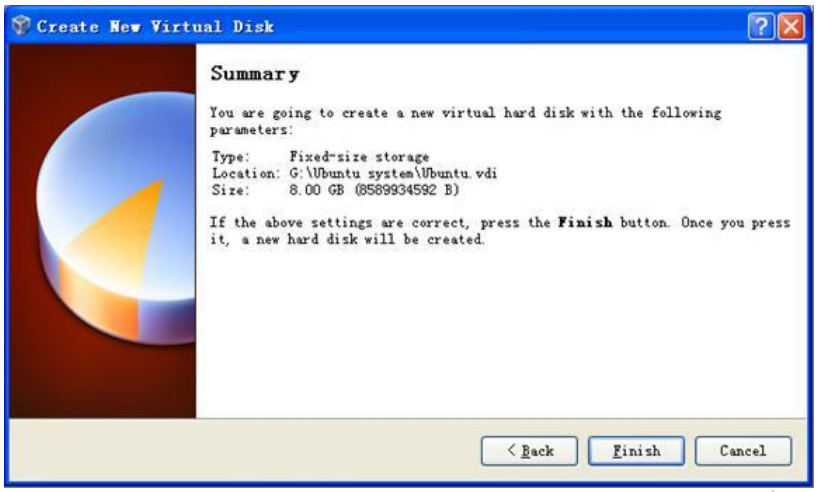


Figure Appendix 2-10

Click **Finish**, the virtual hard drive is successfully created.

3. Installing Ubuntu

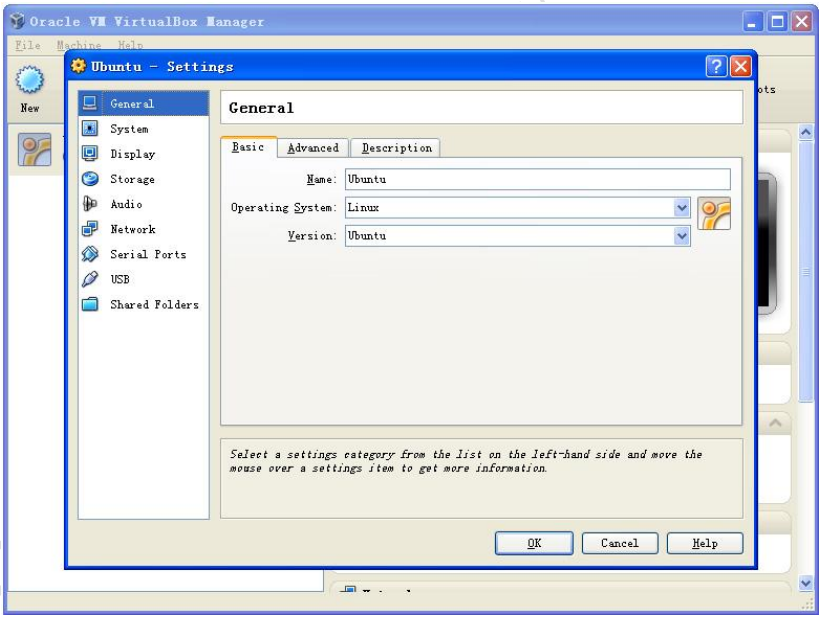


Figure Appendix 2-11

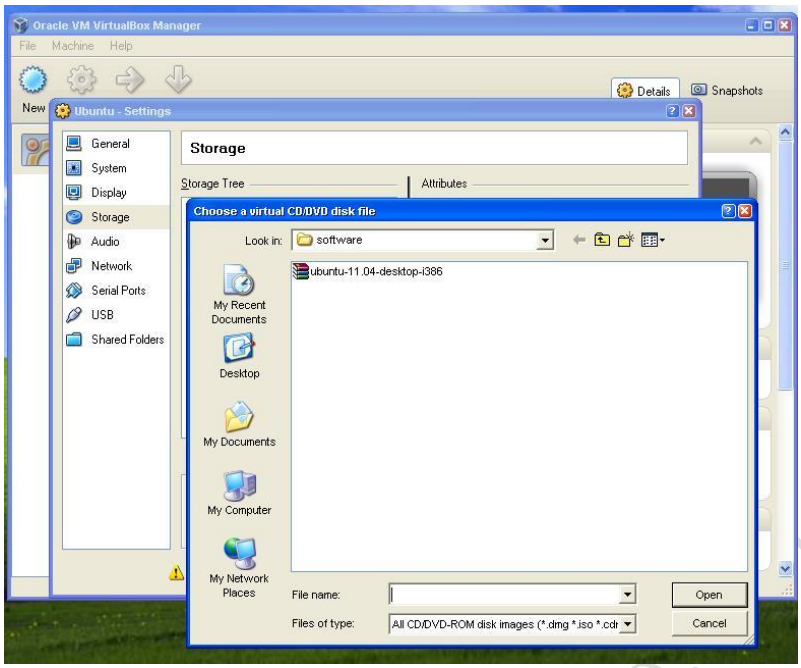


Figure Appendix 2-12

Before Installing Ubuntu in a virtual machine, the first thing to do to make the (currently blank) virtual hard drive useful is to add the downloaded Ubuntu disk image (the .iso) boot on your virtual machine. Click on **Settings** and **Storage**. Then, under CD/DVD Device, next to Empty, you'll see a little folder icon. Click that, and you can select the Ubuntu .iso you downloaded earlier.

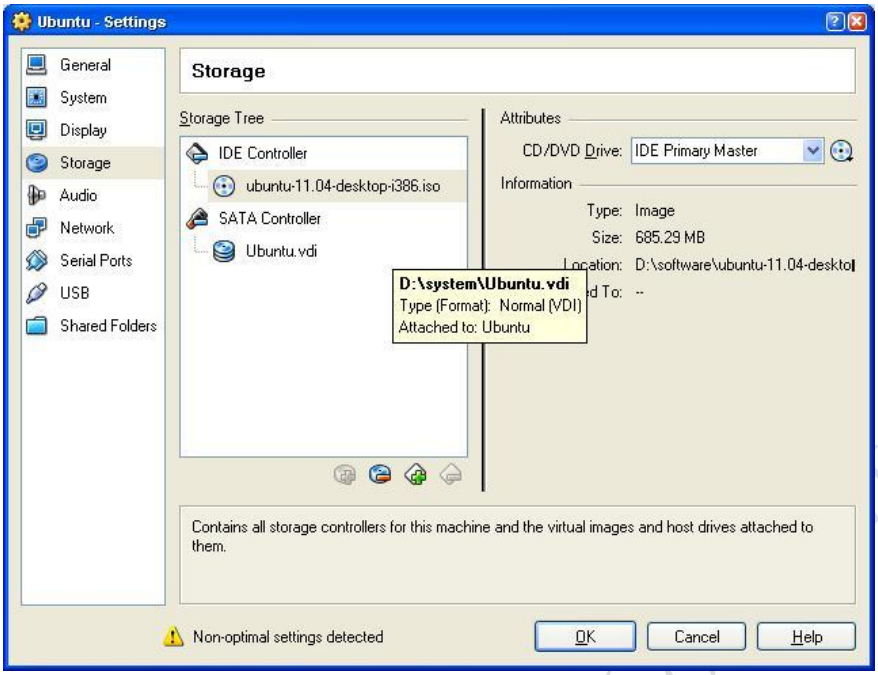


Figure Appendix 2-13

Once you've selected it, click **OK**.

Then double-click your virtual machine to start it up.

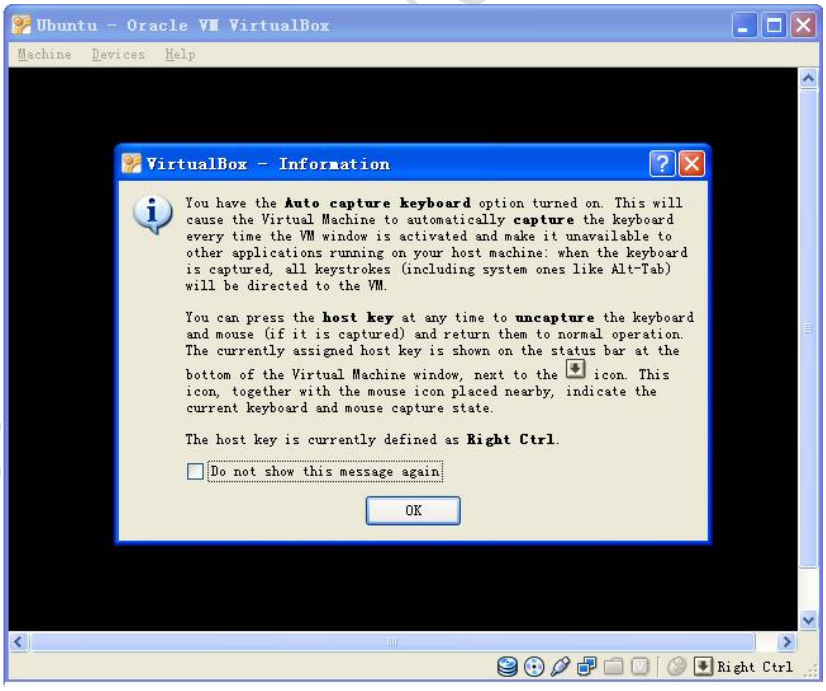


Figure Appendix 2-14

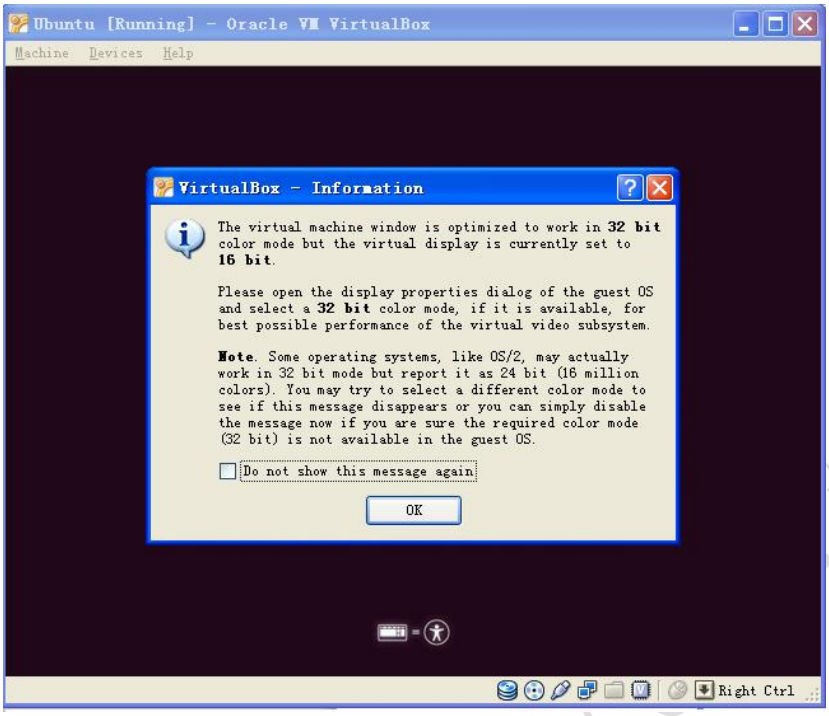


Figure Appendix 2-15

Click **OK**

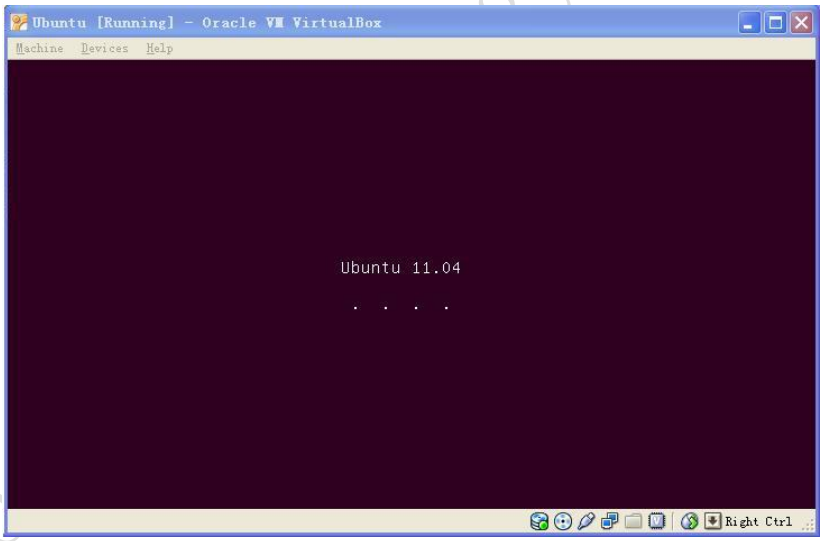


Figure Appendix 2-16

Figure Appendix 2-17

Select language and click **Install Ubuntu**.

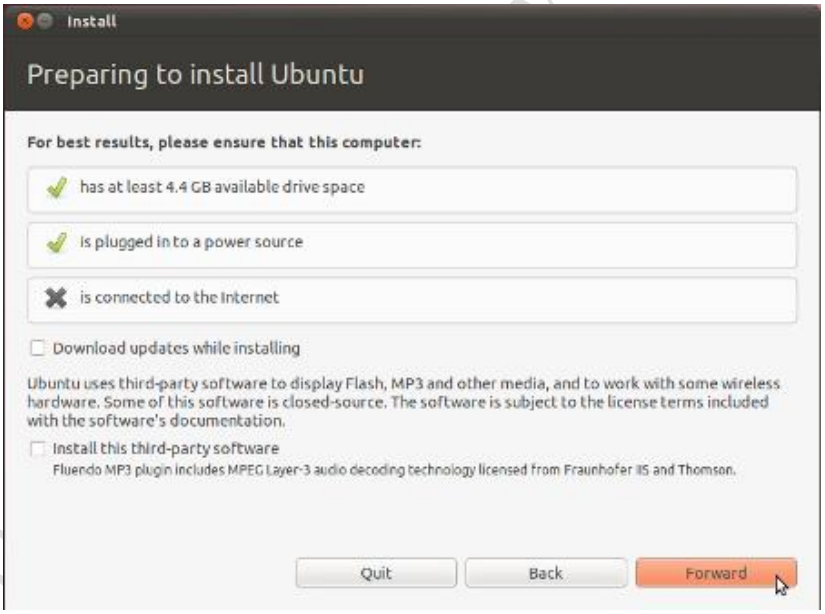


Figure Appendix 2-18

There is a new option in the Ubuntu 11.04 and 10.10 installers that asks if you want to install closed source third-party software for MP3 playback and Flash, for example. I would strongly suggest—unless you know who Richard Stallman is—that you check (or tick) this option.

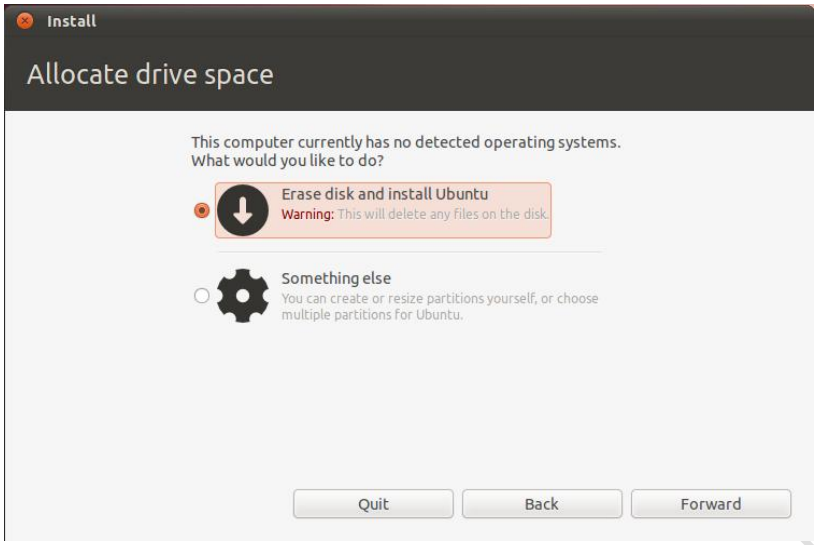


Figure Appendix 2-19

Click **Forward**.

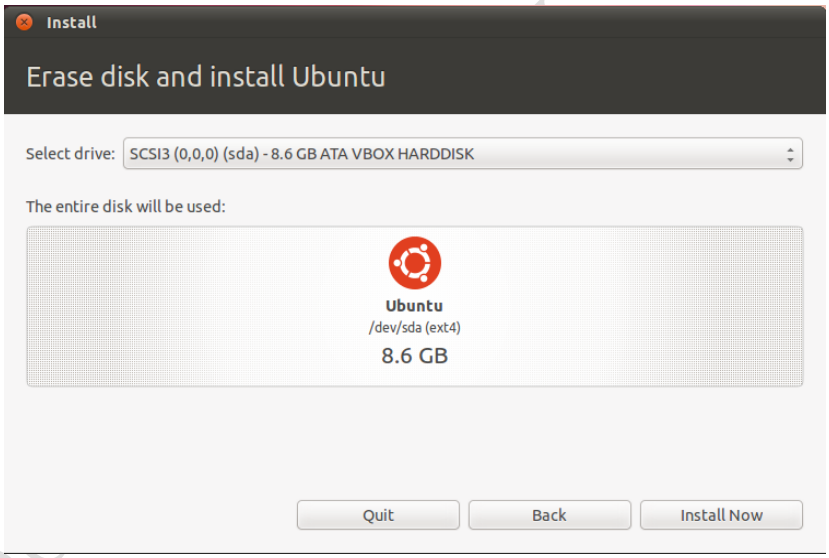


Figure Appendix 2-20

This is the no-turning-back point. If you decide to do this, your hard drive will be repartitioned and part or all of it will be formatted. Before you click this button “Install Now” to continue, make sure you have everything backed up.

Figure Appendix 2-21

While Ubuntu is preparing files to copy over for installation, it'll ask you some questions. They're self-explanatory.

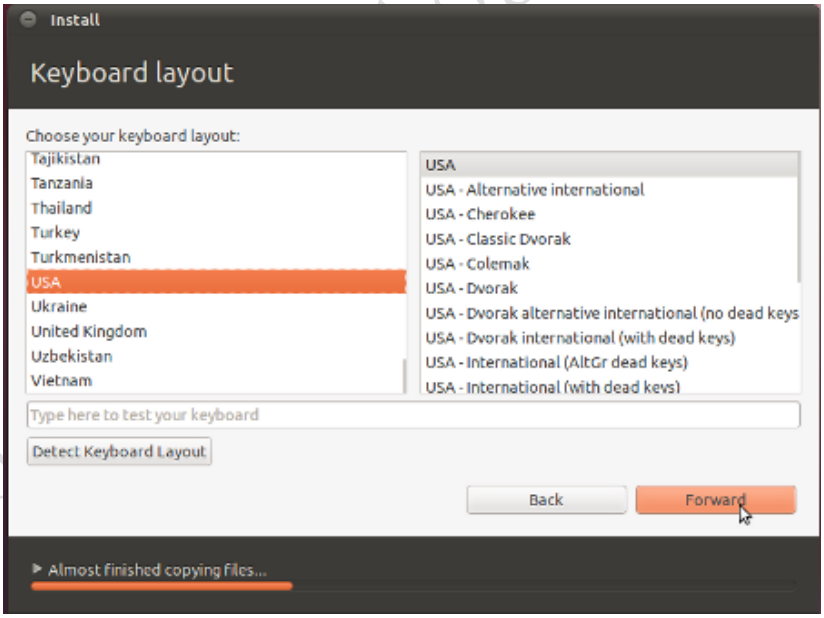


Figure Appendix 2-22

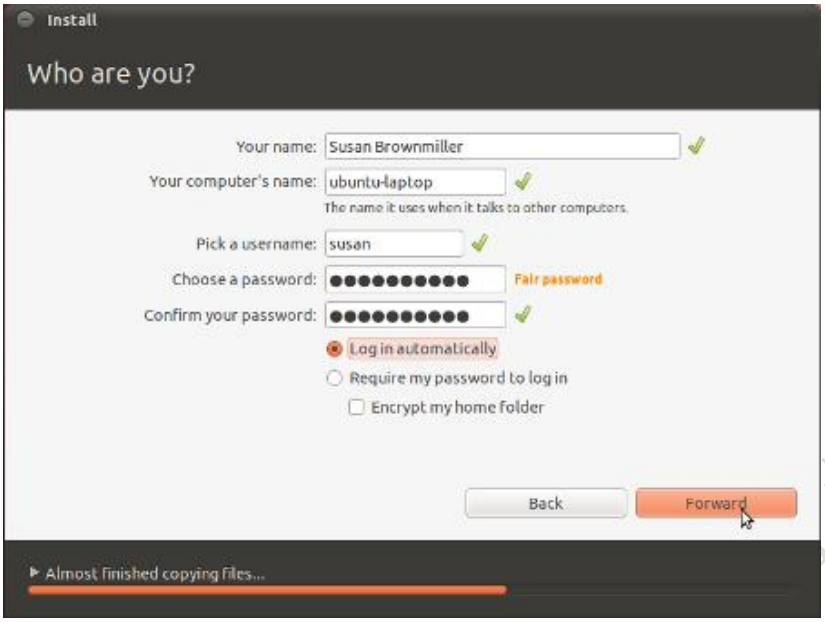


Figure Appendix 2-23

Figure Appendix 2-24

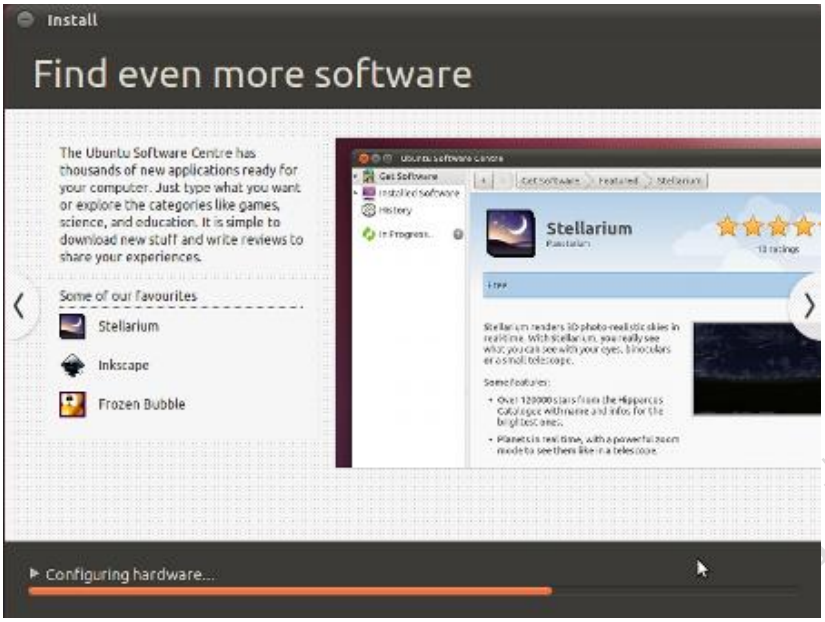


Figure Appendix 2-25

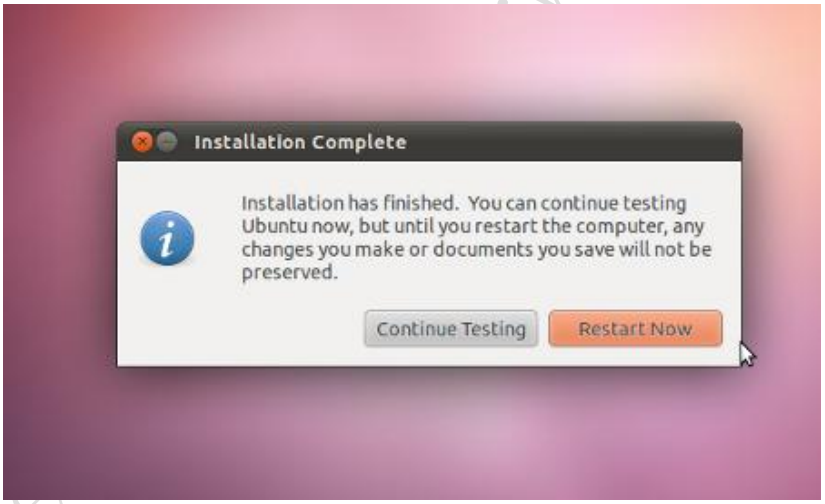


Figure Appendix 2-26

The installation will finish (the whole thing can take anywhere between 15 minutes and an hour, depending on the speed of your computer).

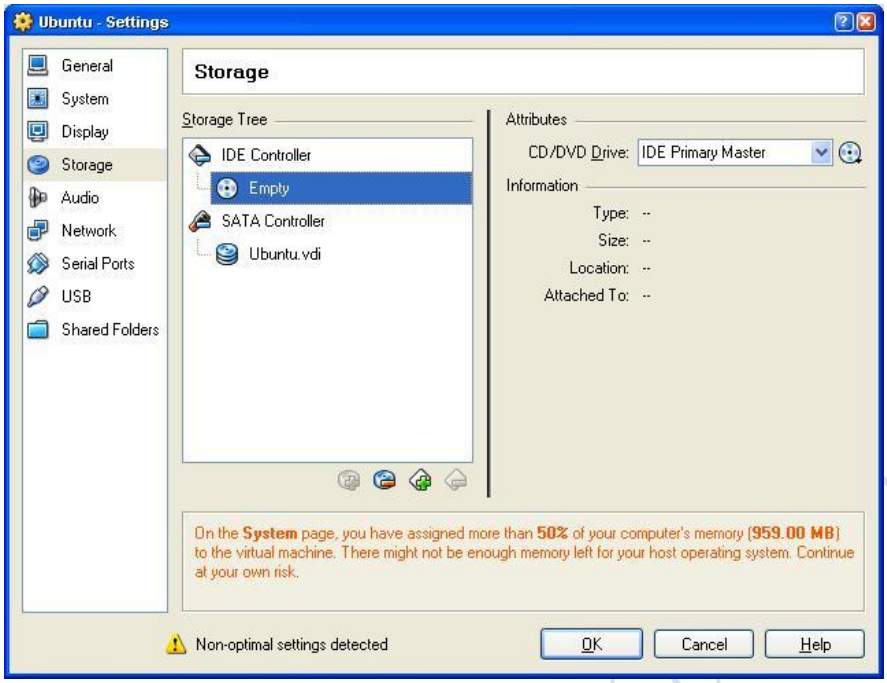


Figure Appendix 2-27

Afterwards, in order to use your virtualized installation (instead of continually booting the live CD); you have to change the CD/DVD Device entry to be Empty again.

Appendix III Driver Installation Of Linux USB Ethernet/RNDIS Gadget

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

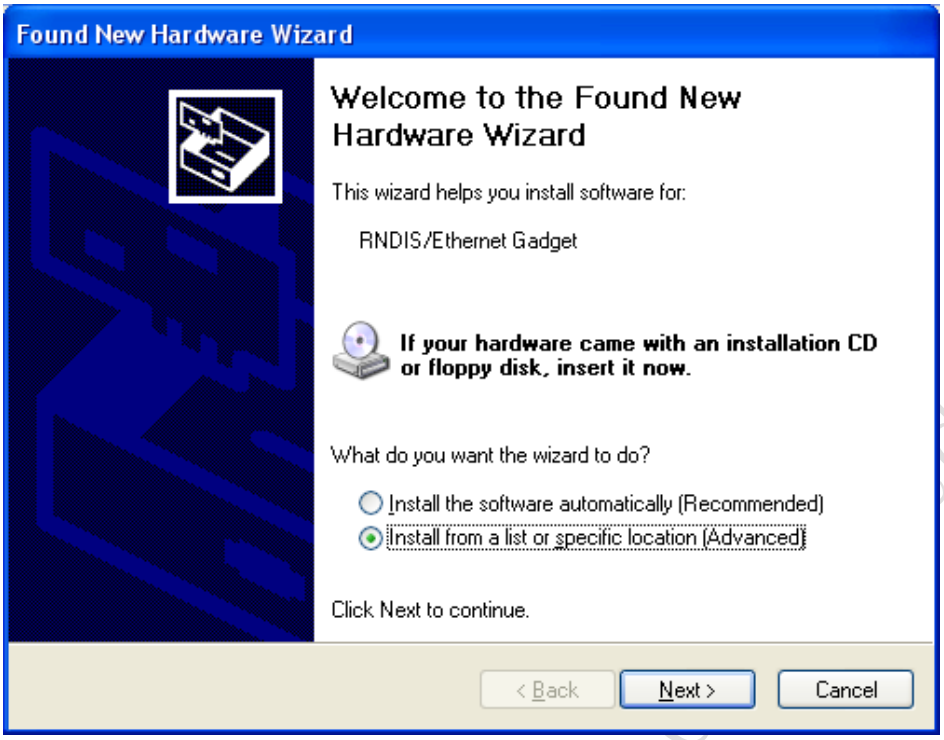


Figure Appedix 3-1

2. Designate a path for the usb driver, and the usb driver directory is [disk\linux\tools], then click “Next”

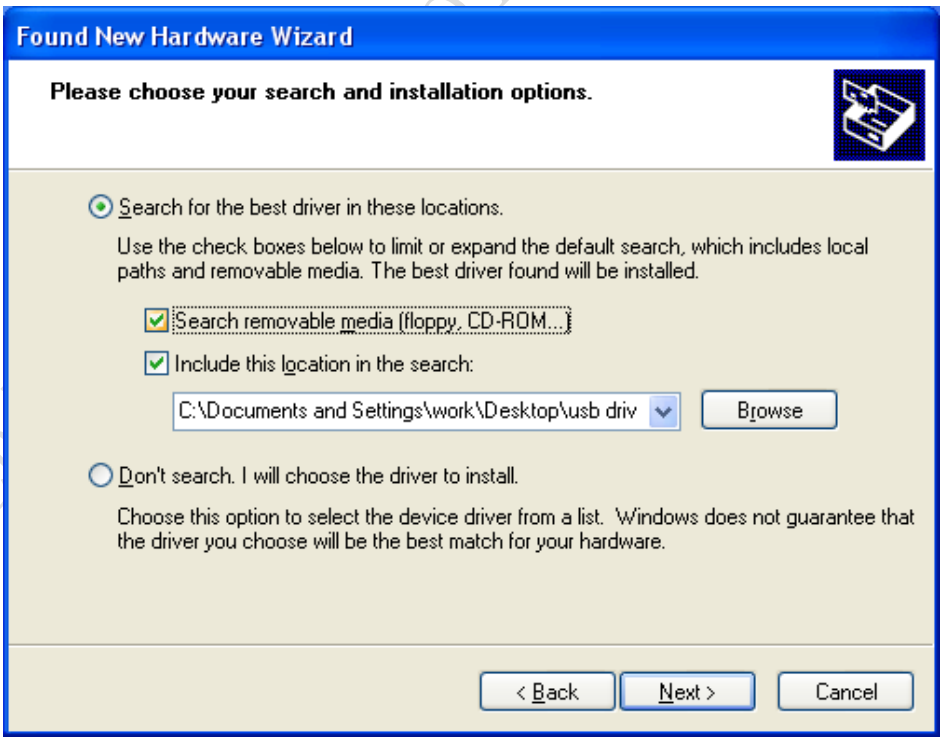


Figure Appedix 3-2

3. When the following appears, select “Continue”

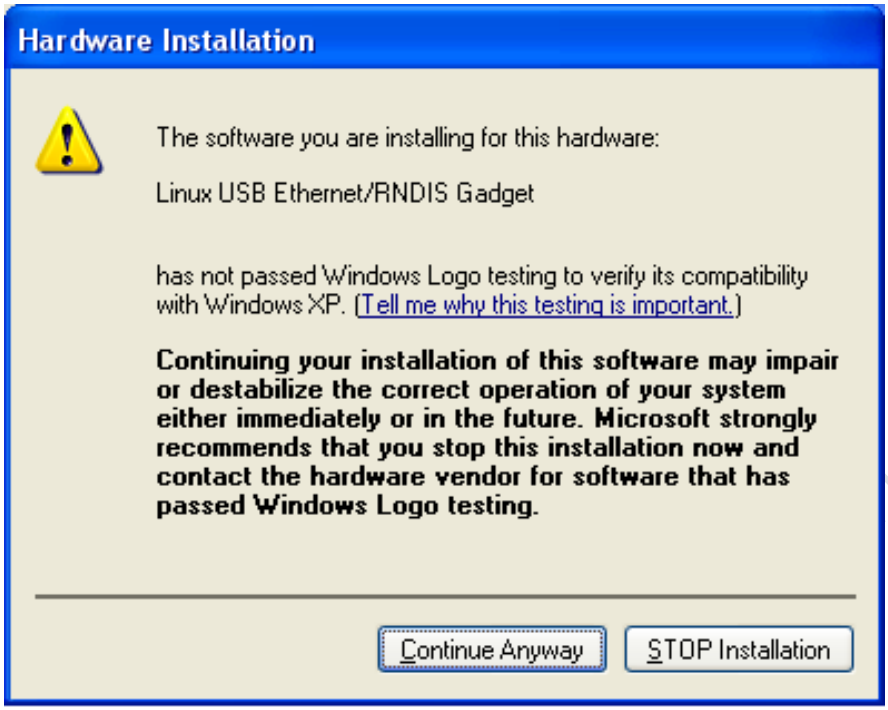


Figure Appedix 3-3

4. Please wait until the installation is completed

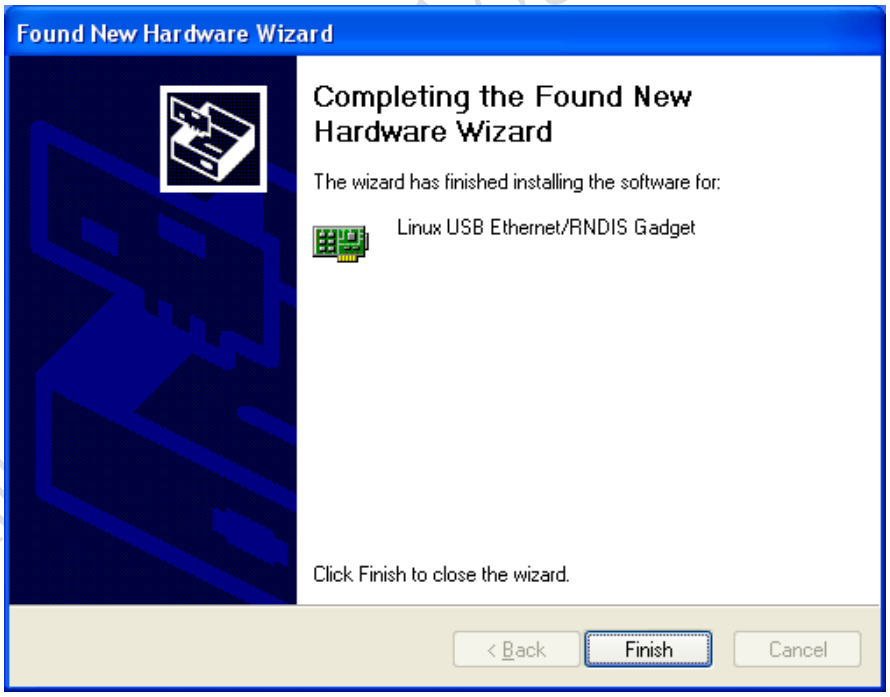


Figure Appedix 3-4

Appendix IV Linux Boot Disk Format

How to create a dual-partition card for DevKit8500D/A to boot Linux from first partition and have root file system at second partition.

一、Introduction

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

二、Details

Text marked with [] shows user input.

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

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

```
$ [dmesg | tail]
...
[ 6854.215650] sd 7:0:0:0: [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
...
```

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

2、Check to see if the automounter has mounted the SD Card

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

```
$ [df -h]
Filesystem            Size  Used Avail Use% Mounted on
...
/dev/sdc1             400M   94M  307M  24% /media/disk
...
```

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

3、If so, unmount it

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

4、Start fdisk

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

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

5、Print the partition record

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

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

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

6、Delete any partitions that are there already

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

Selected *partition* 1

7、Set the Geometry of the TF Card

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

1）Go into expert mode.

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

2）Set the number of heads to 255.

Expert Command *(m* for *help): **[h]***

Number *of heads (*1-256, default *xxx): [*255*]*

3）Set the number of sectors to 63.

Expert Command *(m* for *help): **[s]***

Number *of sectors (*1-63, default *xxx): [*63*]*

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

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

So for this *example:* 2021654528 / 255 / 63 / 512 = 245.79. So *we* use 245 *(i.e. truncate, don*'t round).

5）Set the number of cylinders to the number calculated.

Expert Command *(m* for *help): **[c]***

Number *of cylinders (*1-256, default *xxx): **[enter the number you calculated]***

6）Return to Normal mode.

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

8、Print the partition record to check your work

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

Disk */dev/sdc:* 2021 *MB,* 2021654528 *bytes*

255 *heads,* 63 *sectors/track,* 245 *cylinders*

Units = *cylinders of* 16065 * 512 = 8225280 *bytes*

Device	Boot	Start	End	Blocks	Id	*System*

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

Command *(m* for *help): **[n]***

Command *action*

e *extended*

p *primary partition (*1-4*)*

[p]

Partition *number (*1-4*): **[1]***

First *cylinder (*1-245, default 1*): **[(press* Enter*)]***

Using default *value* 1

Last *cylinder* or *+size* or *+sizeM* or *+sizeK (*1-61, default 61*): **[+5]***

Command *(m* for *help): **[t]***

Selected *partition* 1

Hex *code (type L to list codes): **[c]***

Changed *system type of partition* 1 *to c (W95 FAT32 (LBA))*

10、Mark it as bootable

Command *(m* for *help): **[a]***

Partition *number (*1-4*): **[1]***

11、Create the Linux partition for the root file system

Command *(m* for *help):* ***[n]***

Command *action*

e *extended*

p *primary partition* (1-4)

[p]

Partition *number* (1-4)*:* ***[2]***

First *cylinder* (7-61, default 7)*:* ***[(press Enter)]***

Using default *value* 52

Last *cylinder* or *+size* or *+sizeM* or *+sizeK* (7-61, default 61)*:* ***[(press Enter)]***

Using default *value* 245

12、Print to Check Your Work

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

Disk */dev/sdc:* 2021 *MB,* 2021654528 *bytes*

255 *heads,* 63 *sectors/track,* 245 *cylinders*

Units = *cylinders of* 16065 * 512 = 8225280 *bytes*

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

13、Save the new partition records on the TF Card

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

Command *(m* for *help):* ***[w]***

The *partition table has been altered!*

Calling *ioctl() to re-read partition table.*

WARNING: Re-*reading the partition table failed* with *error* 16*:* Device or *resource busy.*

The *kernel still uses the old table.*

The new *table will be used at the* next *reboot.*

WARNING: If *you have created* or *modified any DOS* 6.x

partitions, please see the fdisk manual page for *additional*

information.

Syncing *disks.*

14、Format the partitions

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

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

mkfs.msdos 2.11 *(*12 Mar 2005*)*

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

mke2fs 1.40-*WIP (*14-Nov-2006*)*

Filesystem *label*=

OS type: Linux

Block *size*=4096 *(log*=2*)*

Fragment *size*=4096 *(log*=2*)*

195072 *inodes,* 389576 *blocks*

19478 *blocks (*5.00*%) reserved* for *the* super *user*

First *data block*=0

Maximum *filesystem blocks*=402653184

12 *block groups*

32768 *blocks per* group, 32768 *fragments per* group

16256 *inodes per* group

Superblock *backups stored on blocks:*

32768, 98304, 163840, 229376, 294912

Writing *inode tables:* done

Creating *journal (*8192 *blocks):* done

Writing *superblocks* and *filesystem accounting information:*

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

Appendix V The Setup Of TFTP Server

1. Install client

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

2. Install inet

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

3. Configure the server

First, create tftpboot under root directory, and set the properties as “a random user can write and read”

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

Secondly, add in /etc/inetd.conf:

$>sudo vi /etc/inetd.conf //copy the follow word to this file

```
tftpd dgram udp wait root /usr/sbin/in.tftpd /usr/sbin/in.tftpd -s /tftpboot
```

Then, reload inetd process:

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

Finally, enter directory /etc/xinetd.d/, and create a new file tftp and put the designated content into file tftp:

```
$>cd /etc/xinetd.d/
$>sudo touch tftp
$>sudo vi tftp          ////copy the follow word to tftp file
service tftp
{
        disable = no
        socket_type     = dgram
        protocol        = udp
        wait            = yes
```

```
        user                    = root
        server                  = /usr/sbin/in.tftpd
        server_args             = -s /tftpboot -c
        per_source              = 11
        cps                     = 100 2
}
```

4. Reboot the server:

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

5. Test the server

Conduct a test; create a file under folder /tftpboot

```
$>touch abc
```

Enter into another folder

```
$>tftp 192.168.1.15    (192.168.1.15was the server IP)
$>tftp> get abc
```

That download can be made means the server has been installed.

Appendix Ⅶ FAQ

Please access http://www.elinux.org/DevKit8600_FAQ.

Technical support & Warranty Service

Embest Technology Co., Ltd. established in March of 2000, is a global provider of embedded hardware and software. Embest aims to help customers reduce time to market with improved quality by providing the most effective total solutions for the embedded industry. In the rapidly growing market of high end embedded systems, Embest provides comprehensive services to specify, develop and produce products and help customers to implement innovative technology and product features. Progressing from prototyping to the final product within a short time frame and thus shorten the time to market, and to achieve the lowest production costs possible. Embest insists on a simple business model: to offer customers high-performance, low-cost products with best quality and service. The content below is the matters need attention for our products technical support and warranty service:

Technical support service

Embest provides one year free technical support service for all products. Technical support service covers:

- Embest embedded platform products software/hardware materials
- Assist customers compile and run the source code we offer.
- Solve the problems accurs on embeded software/hardware platform if users follow the instructions in the documentation we offer.
- Judge whether the product failure exists.

Special explanation, the situations listed below are not included in the range of our free technical support service, and Embest will handle the situation with discretion:

- Software/Hardware issues user meet during the self-develop process

- Issues happen when users compile/run the embedded OS which is tailored by users themselves.
- User's own applications.
- Problems happen during the modification of our software source code

Maintenance service clause

1) The products except LCD, which are not used properly, will take the warranty since the day of the sale:

   PCB: Provide 12 months free maintenance service.

2) The situations listed below are not included in the range of our free maintenance service, Embest will charge the service fees with discretion:

   A. Can't provide valid Proof-of-Purchase, the identification label is torn up or illegible, the identification label is altered or doesn't accord with the actual products;
   B. Don't follow the instruction of the manual in order to damage the product;
   C. Due to the natural disasters (unexpected matters), or natural attrition of the components, or unexpected matters leads to the defects of appearance/function;
   D. Due to the power supply, bump, leaking of the roof, pets, moist, impurities into the boards, all those reasons which lead the defects of appearance/function;
   E. User unauthorized weld or dismantle parts leads the product's bad condition, or let other people or institution which are not authorized by Embest to dismantle, repair, change the product leads the product bad connection or defects of appearance/function;
   F. User unauthorized install the software, system or incorrect configuration or computer virus leads the defects;

G. Purchase the products through unauthorized channel;

H. Those commitments which is committed by other institutions should be responsible by the institutions, Embest has nothing to do with that;

3) During the warranty period, the delivery fee which delivery to Embest should be covered by user, Embest will pay for the return delivery fee to users when the product is repaired. If the warranty period is expired, all the delivery fees will be charged by users.

4) When the board needs repair, please contact technical support department.

Note: Those products are returned without the permission of our technician, we will not take any responsibility for them.

Basic notice to protect and maintenance LCD

1) Do not use finger nails or hard sharp object to touch the surface of the LCD, otherwise user can't enjoy the above service.

2) Embest recommend user to purchase a piece of special wiper to wipe the LCD after long time use , please avoid clean the surface with fingers or hands to leave fingerprint.

3) Do not clean the surface of the screen with chemicals, otherwise user can not enjoy above service.

Note: Embest do not supply maintenance service to LCDs. We suggest the customer first check the LCD after getting the goods. In case the LCD can not run or show no display, customer should inform Embest within 7 business days from the moment of getting the goods.

Value Added Services

We will provide following value added services:

- Provided services of driver develop based on Embest embedded platform, like serial port, USB interface devices, LCD screen.
- Provided the services of control system transplant, BSP drivers develop, API software develop.
- Other value added services like power adapter, LCD parts.
- Other OEM/ODM services.
- Technically training.

Please contact Embest to get technical support:

- Support Tel:+86-755-25503401
- Fax:+86-755-25616057
- Pre-Sale consultation: market@embedinfo.com
- After-Sale consultation: support@embedinfo.com