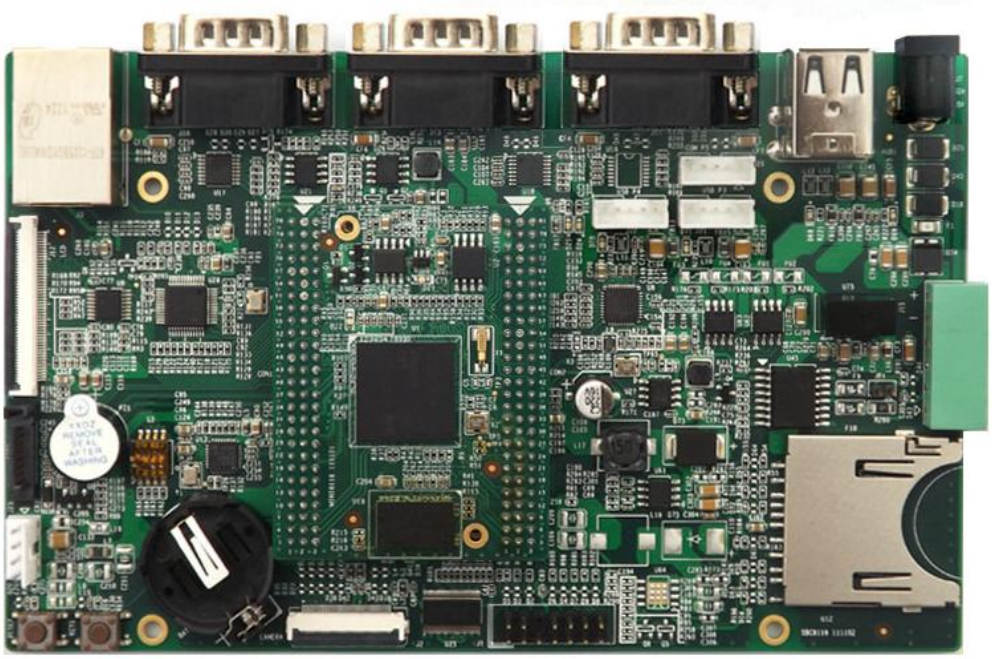
SBC8118

Integrated with SATA, SD, USB, RS485, Ethernet, LCD, CCD/COMS, PRU SUART, JTAG interface based on 32-bit microcontroller AM1808 industrial-grade Single board computer



User Manual

COPYRIGHT

- SBC8118, CAM8000-A, CAM8000-D, GPRS8000-S, GPS8000-S, WCDMA8000-U, CDMA8000-U, WF8000-U, CAM8100-U, VGA8000 are trademarks of Embest Technology Co., Ltd.
- AM1808 are trademarks of TI Corporation.
- Sourcery G++ Lite for ARM GNU/Linux is a trademark of Codesourcery.
- Microsoft, MS-DOS, Windows, Windows95, Windows98, Windows2000, Windows embedded CE 6.0 are trademarks of Microsoft Corporation.

Important Notice

Embest has ownership and rights to the use of this Document.

Information in the Document is within the protection of copyright. Unless specifically allowed, any part of this Document should not be modified, issued or copied in any manner or form without prior written approval of Embest.

Version updates records:

Rev	Date	Description
1.0	2012.9.1	Initial version

Contact:

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

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

Fax: +86-755-25635626-666

E-mail: market@embedinfo.com

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

Tel: +86-755-25635626-872/875/897

E-mail: support@embedinfo.com

URL: http://www.armkits.com or http://www.embest-tech.com

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

Contents

Chapter 1 Overview

1.1 Product introduction

SBC8118 Single board computer is Embest Technology Co., Ltd. launched based on Texas instruments (TI) AM1808 industrial-grade processor Single board computer. AM1808 microprocessor integrated 375/450 MHz ARM9 of kernel and 128 K-Bybe On-the chip memory, and provided a lot of peripheral interfaces. SBC8108 outside enlarge LAN interface, USB HOST, RS485, SATA interface, SD/MMC interface, Debug serial port, PRU serial port, SPI interface, IIC interface, JTAG interface, CAMERA interface, TFT LCD interface, touch screen interface, keyboard interface and bus interface.

Application range of SBC8118:

- Intelligent instrument
- Service point
- Educational plant
- Portable data terminal
- Placeholder intelligent sensor
- A formal data terminal
- Industrial Control
- Home Automation

1.2 Features

SBC8118 single board computer is based on AM1808 processor and is one integrated with all functions and features of this chip. The features of this board are as follows:

1.2.1 Mini8118

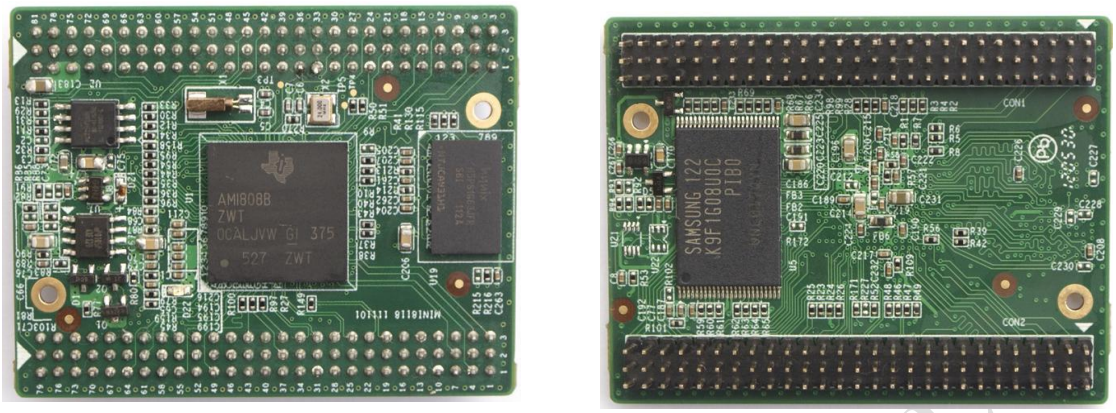


Figure 1-1 Mini8118

Mechanical Parameters

- Dimensions: 58.0 mm*45.0 mm (6 layer PCB design)
- Working temperature: -45°C ~ 85°C
- Humidity Range: 20% ~ 90%
- Input power: 3.3V

Processor

- TI AM1808 ARM9 Microprocessor
  - 375-MHz ARM926EJ-S RISC MPU, also supports 456MHz operation
  - ARM926EJ-S Core
  - ARM9 Memory Architecture
  - Enhanced Direct-Memory-Access Controller 3 (EDMA3)
  - 128K-Byte On-chip Memory

Memory and Storage

- 128MByte Mobile DDR2
- 128MByte NAND Flash (on the rear of the board)
- 8Mb SPI Flash

Board To Board connector and Signals Routed to Pins

- Two 2.0mm space 3*27-pindip connectors
- TFT LCD signal (supports 16-bpp parallel RGB Interface LCD)
- Two channel 8bit camera signals

- One JTAG Debugger signal
- Two channel USB signals（one USB2.0, one USB1.1）
- Five UART signals
- One channel SATA signal, support SATA I(1.5Gbps) and SATA II (3.0Gbps)
- One channel 10/100Mb/s Ethernet MAC(EMAC), with Management Data Input/Output（MDIO）
- One channel 1/2/4/8 bit McASP DMA bus signal(Part of the signals is multiplex with network)
- One channel McBSP signal (with FIFO buffers, multiplex with net)
- Four 64-bit general purpose timers configuration(One configured for watchdog)
- Two SPI signals(SPI0 is multiplex with MII)
- Two channel IIC bus signals(IIC1 is multiplex with UART2）
- Two channel 4-wire SDMMC signals
- GPIO (120 multi-functional input/ output ports at most)

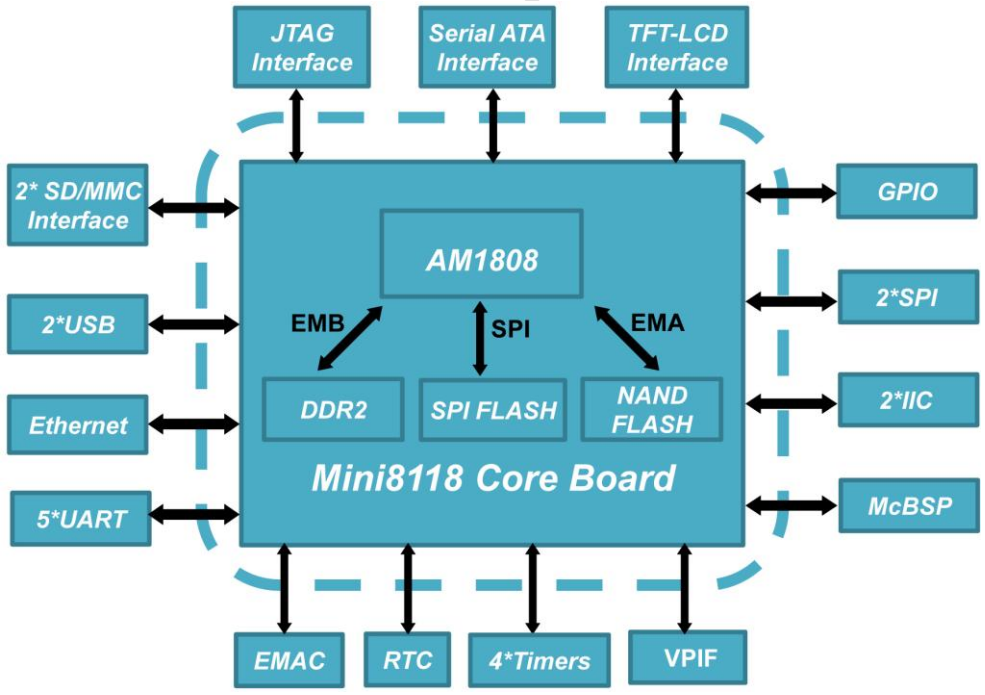


Figure 1-2 Mini8118 hardware structure diagram

1.2.2 SBC8118

Mechanical Parameters

- Dimensions: 160 mm * 100 mm
- Working temperature: -45°C ~ 85°C
- Humidity Range: 20% ~ 90%
- Input power: 12V@1.25A

LCD/Touch screen

- One TFT LCD interface (50pin FPC connector, RGB565)
- Four-line resistive touch screen interface

Data Transfer Interfaces

- Serial Ports
  - One 5-wire serial port, RS232 based voltage (Debugger, DB9 connector）
  - Two 3-wire serial ports, RS232 based voltage, PRU SUART(DB9 connector)
  - One 3-wire serial port, TTL based voltage, PRU SUART(2.0mm space 5pin connector）
- USB2.0 Host interface
  - 2×USB2.0, USB-A type
  - 2× USB2.0, 2.0mm space 5pin connector
- One SD card slot
- One RS485 interface
- One SATA interface, supports SATA I(1.5 Gbps) and SATA II (3.0 Gbps)
- One 10/100Mbps Ethernet Interface (RJ45 jack)
- One 14PIN JTAG interface

Input Interfaces and Other Facilities

- One channel Camera interface (30 PIN FPC connector; Supports CCD or CMOS camera)
- One 4-pin SATA power connector, support 2.5-inch hard disk
- One 4-bit DIP switch, support, support startup mode selection
- One buzzer

- Two buttons(One user button, one reset button)

LED

- One power LED
- One user LED

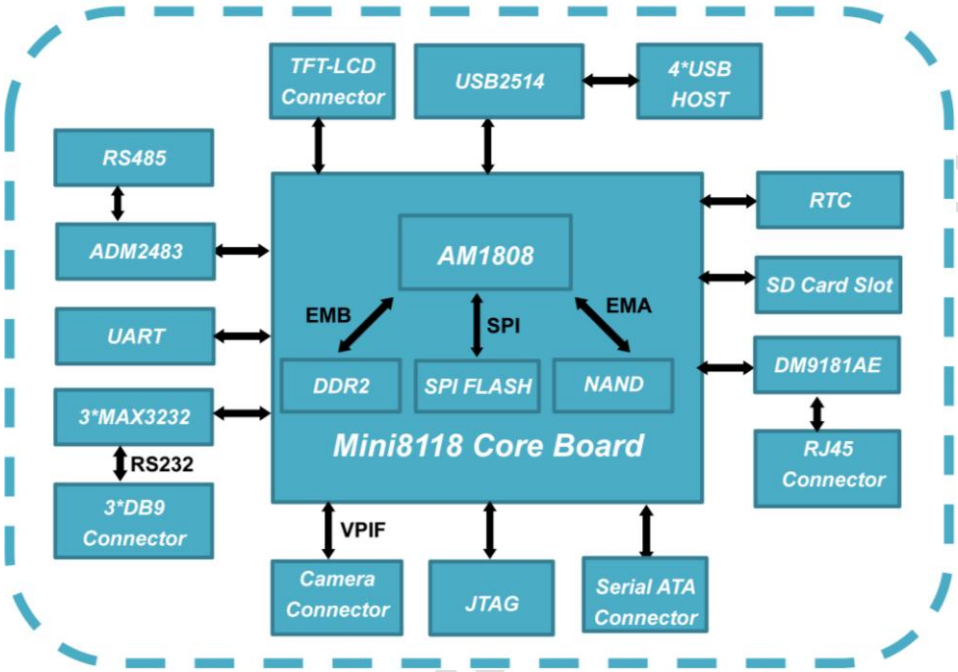


Figure 1-3 SBC8118 hardware structure diagram

Chapter 2 Hardware System

2.1 CPU

2.1.1 CPU Introduction

The AM1808 ARM microprocessor contains an ARM RISC CPU for general-purpose processing and systems control. The AM1808 ARM microprocessor consists of the following primary components:

- ARM926 RISC CPU core and associated memories
- A set of I/O peripherals(The Mini8118 fixed IO level is 3.3V)
- A powerful DMA subsystem and SDRAM EMIF interface

2.1.2 CPU Features

Clock

The OSCIN/ OSCOUT system input clock (24 MHz) is used to generate the main source clock of the device. It supplies the DPLLs as well as several other modules.

Reset

The signal of reset is decided by the RESETN on the CPU, Reset is enable with the LOW level.

ARM Subsystem

The ARM926EJ-S processor (Figure 2-1) is a member of the ARM9 family of general-purpose microprocessors. The ARM926EJ-S processor targets multi-tasking applications where full memory management, high performance, low die size, and low power are all important. The ARM926EJ-S processor supports the 32-bit ARM and the 16-bit THUMB instruction sets, enabling you to trade off between high performance and high code density.

The ARM926EJ-S processor supports the ARM debug architecture and includes logic to assist in both hardware and software debugging. The ARM926EJ-S processor has Harvard architecture and provides a complete high performance subsystem, including the

following:

- ARM926EJ-S 32-bit RISC processor
- A Memory Management Unit (MMU)
- 16-KB Instruction cache
- 16-KB Data cache

ARM Internal Memory

- 8 KB RAM
- 64 KB built-in ROM
- Embedded Trace Module and Embedded Trace Buffer (ETM/ETB)

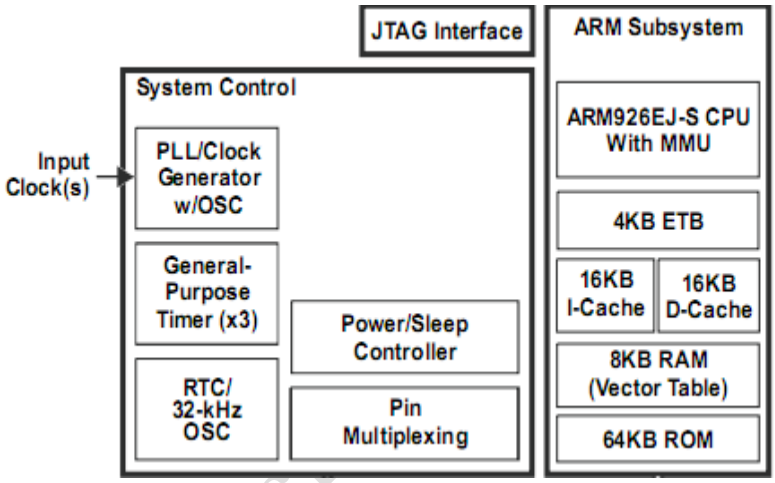


Figure 2-1 ARM Subsystem

2.2 Introduction of expanded chip

2.2.1 K9F1G08U0D

The K9F1G08U0D is SBC8118 NAND Flash chips, size for 128 MB. NAND Flash chips are directly through the CPU NAND Flash interface and the CPU to realize communication.

2.2.2 H5PS1G63JFR

The H5PS1G63JFR is the SBC8118 memory chip, size for 128MB. DDR chip is directly through the CPU DDR/SDRAM interface and the CPU to realize communication.

2.2.3 DM9161

The DM9161 is a fully integrated and cost-effective low pin count single chip Fast Ethernet controller with a general processor interface, it includes a 10/100M PHY, and support MII and RMII transmission mode. It is designed with low power and high performance process that support 3.3V with 5V I/O tolerance.

SBC8118 uses 10/100M adaptive network interface of DM9161, in which, the 10/100M Ethernet module that is built-in DM9161 is compatible to IEEE 802.3 standard protocol. The cable interface is a standard RJ45, with a connection indicator and a transmission indicator.

2.2.4 MAX3232

The function of MAX3232 is mainly to translate TTL level into RS232 level, to adapt to communication with the RS232 serial port of PC.

SBC8118 uses UART2 for debugging; UART2 default voltage is 3.3V that through the max3232 chip.

2.2.5 USB2514

The USB2514 hub is a low-power, high performance, small footprint hub controller IC with

4 downstream ports. The hub supports low-speed, full-speed, and hi-speed(if operating as a hi-speed hub) downstream devices on all of the enabled downstream ports.

2.2.6 SP2526A

The SP2526A contains two separate USB power switch, low power consumption, comes with 600mA internal control switch current limit, supports 3.3V and 5V power input, widely used in the USB output current control.

2.2.7 ADM2483

ADM2483BRWZ is the RS485 transceiver with electrical data isolation; it complies with ANSI TIA/EIA RS-485-A-1998 and ISO 8482:1987(E). The UART network can connect to RS485 network for communication through ADM2483BRWZ.

2.2.8 B0505S

B0505S is the efficiency of up to 86% DC-DC isolated converter. It is high precision, high isolation (1KV DC) and high reliability. With overload protection function, simple circuit connection, a wide range of industrial applications

SBC8118 uses B0505S as the isolation between onboard power and external RS-485 communication power; it can effectively reduce external interference, protect the onboard device and ensure the communications of normal

2.2.9 CDCM61001RHBT

The CDCM61001 is a highly versatile and low-jitter frequency synthesizer that can generate low-jitter clock outputs, selectable between low-voltage positive emitter coupled logic (LVPECL), low-voltage differential signaling (LVDS), or low-voltage(LVCMOS) output. SBC8118 uses the CDCM61001 to outputs 100MHz differential frequency, as SATA PLL input frequency.

2.2.10 TSC2046IPWR

TSC2046, 4 touch screen controller, support from 1.5V to 5.25V low voltage I/O

interface.TSC2046 also has an on-chip 2.5V reference, can be used for auxiliary input, battery monitoring and temperature measurement mode. Internal voltage reference 2.7V supply voltage, at the same time monitoring battery voltage from 0V to 6V

2.3 Hardware interface

2.3.1 Mini8118

2.3.1.1 CON1

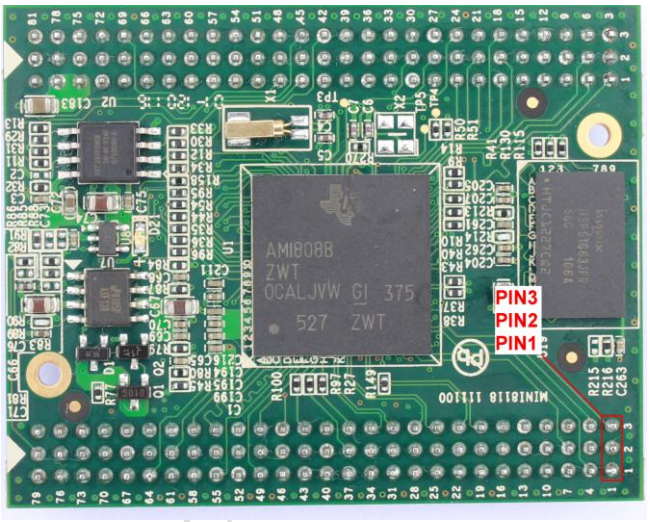


Table 2-1 CON1 Interface

CON1		
Pin	**Signal**	**Function**
1	LCD_D0	VP_DOUT[0] / LCD_D[0] / UPP_XD[8] / GP7[8] / PRU1_R31[8]
2	LCD_D1	VP_DOUT[1] / LCD_D[1] / UPP_XD[9] / GP7[9] / PRU1_R31[9]
3	LCD_D2	VP_DOUT[2] / LCD_D[2] / UPP_XD[10] / GP7[10] / PRU1_R31[10]
4	LCD_D3	VP_DOUT[3] / LCD_D[3] / UPP_XD[11] / GP7[11] / PRU1_R31[11]
5	LCD_D4	VP_DOUT[4] / LCD_D[4] / UPP_XD[12] / GP7[12] /

		PRU1_R31[12]
6	LCD_D5	VP_DOUT[5] / LCD_D[5] / UPP_XD[13] / GP7[13] / PRU1_R31[13]
7	LCD_D6	VP_DOUT[6] / LCD_D[6] / UPP_XD[14] / GP7[14] / PRU1_R31[14]
8	LCD_D7	VP_DOUT[7] / LCD_D[7] / UPP_XD[15] / GP7[15] / PRU1_R31[15]
9	LCD_D8	VP_DOUT[8] / LCD_D[8] / UPP_XD[0] /GP7[0] / BOOT[0]
10	LCD_D9	VP_DOUT[9] / LCD_D[9] / UPP_XD[1] /GP7[1] / BOOT[1]
11	LCD_D10	VP_DOUT[10] / LCD_D[10] / UPP_XD[2] /GP7[2] / BOOT[2]
12	LCD_D11	VP_DOUT[11] / LCD_D[11] / UPP_XD[3] /GP7[3] / BOOT[3]
13	LCD_D12	VP_DOUT[12] / LCD_D[12] / UPP_XD[4] / GP7[4] / BOOT[4]
14	LCD_D13	VP_DOUT[13] / LCD_D[13] / UPP_XD[5] /GP7[5] / BOOT[5]
15	LCD_D14	VP_DOUT[14] / LCD_D[14] / UPP_XD[6] /GP7[6] / BOOT[6]
16	LCD_D15	VP_DOUT[15]/ LCD_D[15]/ UPP_XD[7] /GP7[7] / BOOT[7]
17	DGND	Power ground
18	DGND	Power ground
19	DGND	Power ground
20	DGND	Power ground
21	SATA_100M_CLKN	SATA_REFCLKN
22	DGND	Power ground

23	DGND	Power ground
24	SATA_100M_CLKP	SATA_REFCLKP
25	UART4_TX	AXR10 / DR1 / GP0[2]
26	DGND	Power ground
27	uP_SATA_RXP	SATA_RXP
28	UART4_RX	AXR9 / DX1 / GP0[1]
29	DGND	Power ground
30	uP_SATA_RXN	SATA_RXN
31	UART5_TX	AXR14 / CLKR1 / GP0[6]
32	DGND	Power ground
33	uP_SATA_TXN	SATA_TXN
34	UART5_RX	AXR13 / CLKX1 / GP0[5]
35	DGND	Power ground
36	uP_SATA_TXP	SATA_TXP
37	MMC_WP	EMA_A[7] / PRU1_R30[15] / GP5[7]
38	MMC_SD1_D0	PRU0_R30[25] /MMCSD1_DAT[0] / UPP_CHB_CLOCK /GP8[15]/PRU1_R31[27]
39	MMC_SD1_CLK	PRU0_R30[24]/ MMCSD1_CLK / UPP_CHB_START / GP8[14] / PRU1_R31[26]
40	LCD_AC_ENB_CS N	LCD_AC_ENB_CS / GP6[0]/ / PRU1_R31[28]
41	MMC_SD1_CMD	PRU0_R30[23] /MMCSD1_CMD / UPP_CHB_ENABLE / GP8[13]/PRU1_R31[25]
42	MMC_SD1_D2	VP_CLKOUT2 / MMCSD1_DAT[2] / PRU1_R30[2] / GP6[3] / PRU1_R31[3]
43	LVDS_EN	AMUTE / PRU0_R30[16] / UART2_RTS / GP0[9] / PRU0_R31[16]
44	MMC_SD1_D1	VP_CLKIN3 / MMCSD1_DAT[1] / PRU1_R30[1] / GP6[2] /PRU1_R31[2]

45	MMC_SD1_D3	VP_CLKIN2 / MMCSD1_DAT[3] / PRU1_R30[3] / GP6[4] /PRU1_R31[4]
46	+3P3V_DEV_EN	PRU0_R30[30] / UHPI_HINT / PRU1_R30[11] / GP6[12]
47	LCD_EN	PRU0_R30[29] / UHPI_HCNTL0 / UPP_CHA_CLOCK / GP6[11]
48	UART3_RX	EMA_A[7] / PRU1_R30[15] / GP5[7]
49	LCD_VSYNC	MMCSD1_DAT[4] / LCD_VSYNC / PRU1_R30[4] / GP8[8] /PRU1_R31[5]
50	LCD_HSYNC	MMCSD1_DAT[5] / LCD_HSYNC / PRU1_R30[5] / GP8[9] /PRU1_R31[6]
51	LCD_PCLK	MMCSD1_DAT[7] / LCD_PCLK / PRU1_R30[7] / GP8[11]
52	MII_TXD1	AXR1 / DX0 / GP1[9] / MII_TXD[1]
53	MII_TXD2	AXR2 / DR0 / GP1[10] / MII_TXD[2]
54	MII_TXD3	AXR3 / FSX0 / GP1[11] / MII_TXD[3]
55	MII_COL	AXR4 / FSR0 / GP1[12] / MII_COL
56	MII_TXD0	AXR0 / ECAP0_APWM0 / GP8[7] / MII_TXD[0] / CLKS0
57	MII_TXEN	AXR6 / CLKR0 / GP1[14] / MII_TXEN /PRU0_R31[6]
58	DGND	Power ground
59	MII_TXCLK	AXR5 / CLKX0 / GP1[13] / MII_TXCLK
60	AIC_WCLK	AFSX / GP0[12] / PRU0_R31[19]
61	UART3_TX	AXR12 / FSR1 / GP0[4]
62	AIC_BCLK	ACLKX / PRU0_R30[19]/ GP0[14]/ PRU0_R31[21]
63	+3P3V	Power 3.3V
64	AIC_MCLK/UART1_CTSN	AHCLKX / USB_REFCLKIN / UART1_CTS / GP0[10] /PRU0_R31[17]
65	MII_MDC	SPI0_SCS[1] /TM64P0_OUT12 / GP1[7] / MDIO_CLK/ TM64P0_IN12
66	DEEPSLEEP	RTC_ALARM / UART2_CTS / GP0[8] / DEEPSLEEP

67	USER_MENU	EMA_A[10] / PRU1_R30[18] / GP5[10]
68	MII_RXER	SPI0_SOMI / EPWMSYNCI / GP8[6] / MII_RXER
69	MII_RXD0	SPI0_SCS[2] / UART0_RTS / GP8[1] / MII_RXD[0] /SATA_CP_DET
70	User_MENU	EMA_A[10] / PRU1_R30[18] / GP5[10]
71	MII_RXDV	SPI0_ENA / EPWM0B / PRU0_R30[6] / MII_RXDV
72	MII_MDIO	SPI0_SCS[0] / TM64P1_OUT12 / GP1[6] / MDIO_D/TM64P1_IN12
73	XEINT16	EMA_A[4] / GP5[4]
74	MII_RXD3	SPI0_SCS[5] / UART0_RXD / GP8[4] / MII_RXD[3]
75	MII_CRS	SPI0_SIMO / EPWMSYNCO / GP8[5] / MII_CRS
76	DGND	Power ground
77	MII_RXCLK	SPI0_CLK / EPWM0A / GP1[8] / MII_RXCLK
78	MII_RXD2	SPI0_SCS[4] / UART0_TXD / GP8[3] / MII_RXD[2]
79	PG_400MS	CPU reset input
80	VCORE_EN	+1P2V_CORE_EN
81	MII_RXD1	SPI0_SCS[3] / UART0_CTS / GP8[2] / MII_RXD[1] /SATA_MP_SWITCH

2.3.1.2 CON2

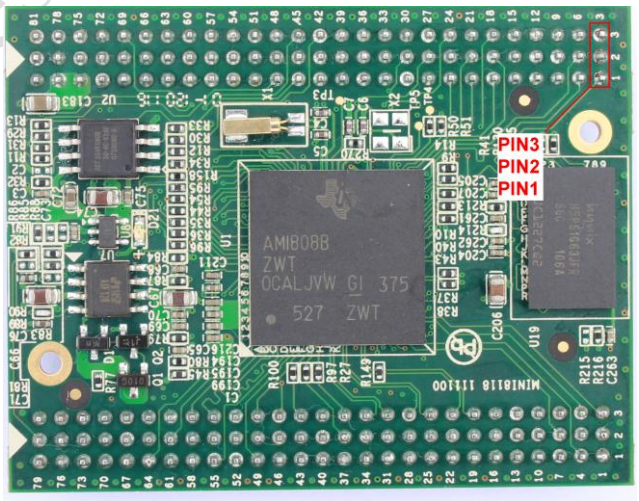


Table 2-2 CON2 Interface

CON2		
Pin	**Signal**	**Function**
1	CAM_PCLK_A0	VP_CLKIN0/UHPI_HCS/PRU1_R30[10]/GP6[7]/UPP_2x TXCLK
2	CAM_D0	VP_DIN[0] / UHPI_HD[8] / UPP_D[8] / RMII_CRS_DV /PRU1_R31[29]
3	CAM_D1	VP_DIN[1] / UHPI_HD[9] / UPP_D[9] / RMII_MHZ_50_CLK /PRU0_R31[23]
4	CAM_D2	VP_DIN[2] / UHPI_HD[10] / UPP_D[10] / RMII_RXER /PRU0_R31[24]
5	CAM_D3	VP_DIN[3] / UHPI_HD[11] / UPP_D[11] / RMII_RXD[0] /PRU0_R31[25]
6	CAM_D4	VP_DIN[4] / UHPI_HD[12] / UPP_D[12] / RMII_RXD[1] /PRU0_R31[26]
7	CAM_D5	VP_DIN[5] / UHPI_HD[13] / UPP_D[13] / RMII_TXEN /PRU0_R31[27]
8	CAM_D6	VP_DIN[6] / UHPI_HD[14] / UPP_D[14] / RMII_TXD[0] /PRU0_R31[28]
9	CAM_D7	VP_DIN[7] / UHPI_HD[15] / UPP_D[15] / RMII_TXD[1] /PRU0_R31[29]
10	CAM_FIELD	VP_DIN[13]_FIELD / UHPI_HD[5] / UPP_D[5] / PRU0_R30[13] /PRU0_R31[13]
11	CAM_HSYNC	VP_DIN[14]_HSYNC / UHPI_HD[6] / UPP_D[6]/ PRU0_R30[14] /PRU0_R31[14]
12	CAM_VSYNC	VP_DIN[15]_VSYNC / UHPI_HD[7] / UPP_D[7] / PRU0_R30[15] /PRU0_R31[15]
13	uP_CAM_STR	PRU0_R30[26] / UHPI_HRW / UPP_CHA_WAIT / GP6[8] /PRU1_R31[17]
14	CAM_XCLKB_A0	PRU0_R30[28] / UHPI_HCNTL1 / UPP_CHA_START / GP6[10]
15	CAM_D12	VP_DIN[12] / UHPI_HD[4] / UPP_D[4]/ PRU0_R30[12] /PRU0_R31[12]
16	CAM_D11	VP_DIN[11] / UHPI_HD[3] / UPP_D[3]/ PRU0_R30[11] /PRU0_R31[11]
17	CAM_D10	VP_DIN[10] / UHPI_HD[2] / UPP_D[2] / PRU0_R30[10] /PRU0_R31[10]

18	CAM_D9	VP_DIN[9] / UHPI_HD[1] / UPP_D[1] / PRU0_R30[9] /PRU0_R31[9]
19	CAM_D8	VP_DIN[8] / UHPI_HD[0] / UPP_D[0] / GP6[5] / PRU1_R31[0]
20	CAM_PCLK_A1	VP_CLKIN1 / UHPI_HDS1 / PRU1_R30[9] / GP6[6] /PRU1_R31[16]
21	DGND	Power ground
22	uP_RESETOUTn	RESETOUT / UHPI_HAS / PRU1_R30[14] / GP6[15]
23	uP_TDI	TDI
24	uP_TDO	TDO
25	uP_TMS	TMS
26	uP_TCK	TCK
27	uP_TRSTn	TRST
28	uP_EMU1	uP_EMU1
29	uP_EMU0	uP_EMU0
30	uP_RTCK	RTCK
31	uP_USB1_DP	USB1_DP
32	uP_USB0_ID	USB0_ID
33	DGND	Power ground
34	uP_USB1_DM	USB1_DM
35	DGND	Power ground
36	DGND	Power ground
37	uP_USB0_DP	USB0_DP
38	DGND	Power ground
39	DGND	Power ground
40	uP_USB0_DM	USB0_DM
41	TP_SPI1_SCSn1	SPI1_SCS[1] / EPWM1A / PRU0_R30[7] / GP2[15] / TM64P2_IN12
42	uP_SPI1_SOMI	SPI1_SOMI / GP2[11]
43	uP_SPI1_SIMO	SPI1_SIMO / GP2[10]
44	uP_SPI1_ENAN	SPI1_ENA / GP2[12]

45	TP_BUSY	EMA_A[5] / GP5[5]
46	uP_SPI1_CLK	SPI1_CLK / GP2[13]
47	uP_NMIn	RSVDN
48	DGND	Power ground
49	DGND	Power ground
50	I2C0_SCL	SPI1_SCS[7] / I2C0_SCL / TM64P2_OUT12 / GP1[5]
51	I2C0_SDA	SPI1_SCS[6] / I2C0_SDA / TM64P3_OUT12 / GP1[4]
52	UART2_RXD	SPI1_SCS[5] / UART2_RXD / I2C1_SCL /GP1[3]
53	MMC_CD	EMA_A[17] / MMCSD0_DAT[4] / PRU1_R30[25] / GP4[1] B
54	DGND	Power ground
55	UART2_TXD	SPI1_SCS[4] / UART2_TXD / I2C1_SDA /GP1[2]
56	SD0_DATA1	EMA_A[20] / MMCSD0_DAT[1] / PRU1_R30[28] / GP4[4] /PRU1_R31[20]
57	SD0_DATA0	EMA_A[21] / MMCSD0_DAT[0] / PRU1_R30[29] / GP4[5] /PRU1_R31[21]
58	UART1_TXD	SPI1_SCS[2] / UART1_TXD /SATA_CP_POD / GP1[0]
59	SD0_CLK	MMCSD0_CLK / PRU1_R30[31] /GP4[7] / PRU1_R31[23]
60	SD0_CMD	EMA_A[22] / MMCSD0_CMD / PRU1_R30[30] / GP4[6] /PRU1_R31[22]
61	UART1_RXD	SPI1_SCS[3] / UART1_RXD / SATA_LED /GP1[1]
62	SD0_DATA3	EMA_A[18] / MMCSD0_DAT[3] / PRU1_R30[26] / GP4[2] /PRU1_R31[18]
63	SD0_DATA2	EMA_A[19] / MMCSD0_DAT[2] / PRU1_R30[27] / GP4[3] /PRU1_R31[19]
64	WLAN_WAKE	EMA_CS[2] / GP3[15
65	WLAN_RESET	EMA_A[6] / GP5[6]
66	WLAN_HOST_WAKE	EMA_CS[0] / GP2[0]
67	BT_HOST_WAKE	EMA_RAS / PRU0_R30[3] / GP2[5] /PRU0_R31[3]
68	BT_WAKE	EMA_A[13] /PRU0_R30[21] / PRU1_R30[21]/ GP5[13]
69	BT_RST	EMA_CS[5] / GP3[12]

70	BUZZER	AXR15 / EPWM0TZ[0] / ECAP2_APWM2 / GP0[7]
71	LCD_PWM	AXR7/EPWM1TZ[0] / PRU0_R30[17] / GP1[15] /PRU0_R31[7]
72	VCC_RTC	1.2V RTC Power
73	+3P3V_STB	Power 3.3V
74	USB_RTSTn	EMA_A[8] / PRU1_R30[16] / GP5[8]
75	uP_RESETn_KEY	CPU reset input
76	+3P3V_STB	Power 3.3V
77	DGND	Power ground
78	USB0_VBUS	USB0_VBUS
79	UART2_RTS	AMUTE / PRU0_R30[16] / UART2_RTS / GP0[9] / PRU0_R31[16]
80	UART1_RTSn	AHCLKR / PRU0_R30[18] / UART1_RTS / GP0[11] /PRU0_R31[18]
81	WL_BT_REG_ON	EMA_CS[4] / GP3[13]

2.3.2 SBC8118

2.3.2.1 Power Input Jack

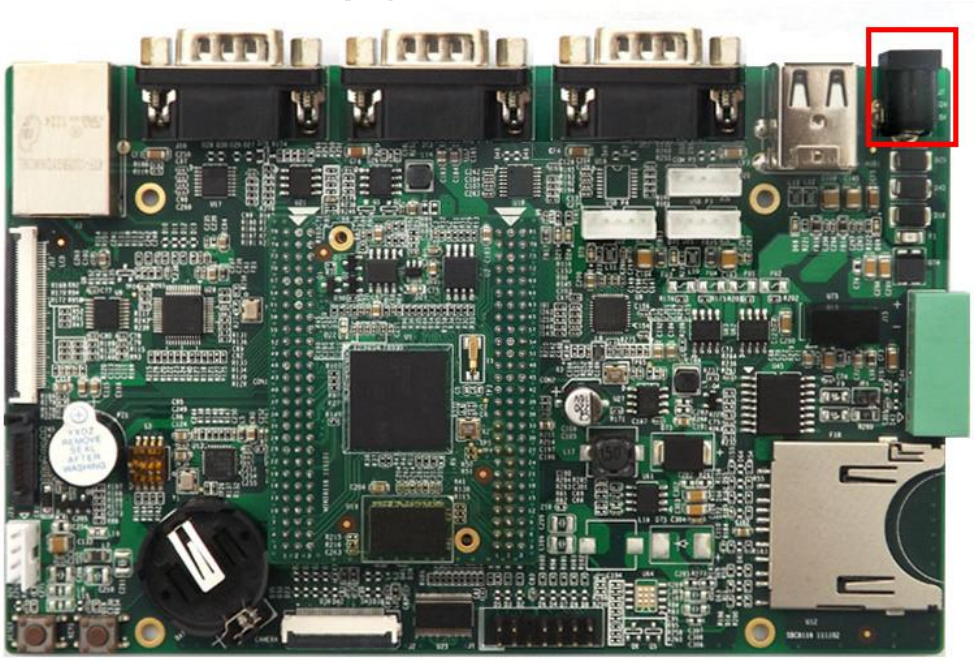


Table 2-3 Power Input Interface

J7		
Pin	**Signal**	**Function**
1	GND_IN	GND_IN
2	NC	NC
3	+12V	Power supply（+12V) 1.25A（Type）

2.3.2.2 SATA Power Supply Interface

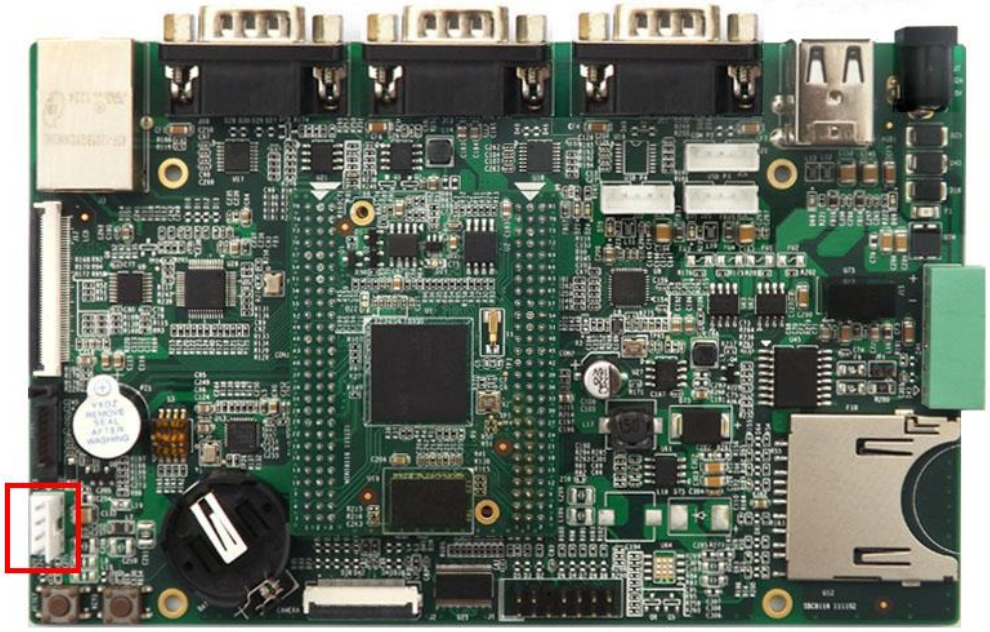


Table 2-4 SATA Power Supply Interface

J23		
Pin	**Signal**	**Function**
1	+3P3V	NC
2	GND	DGND
3	+5V	+5VDC_IN
4	+12V	NC

2.3.2.3TFT_LCD Interface

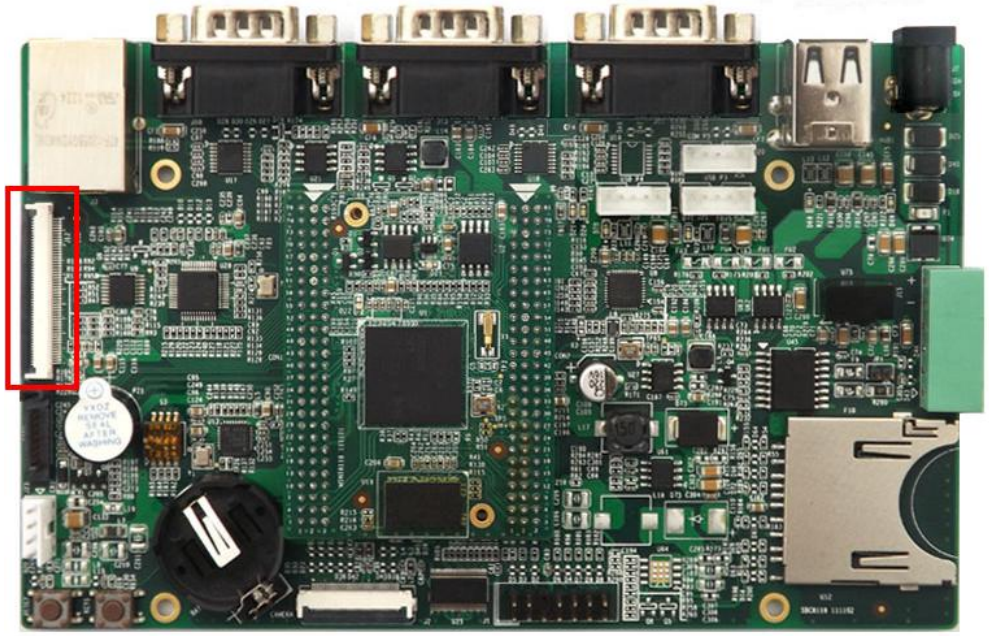


Table 2-5 TFT_LCD Interface

J12		
Pin	**Signal**	**Function**
1	B0	NC
2	B1	NC
3	B2	NC
4	B3	LCD data bit0
5	B4	LCD data bit1
6	B5	LCD data bit2
7	B6	LCD data bit3
8	B7	LCD data bit4
9	GND	DGND
10	G0	NC
11	G1	NC
12	G2	LCD data bit5
13	G3	LCD data bit6
14	G4	LCD data bit7

15	G5	LCD data bit8
16	G6	LCD data bit9
17	G7	LCD data bit10
18	GND	DGND
19	R0	NC
20	R1	NC
21	R2	NC
22	R3	LCD data bit11
23	R4	LCD data bit12
24	R5	LCD data bit13
25	R6	LCD data bit14
26	R7	LCD data bit15
27	GND	DGND
28	DEN	LCD AC bias enable chip select
29	HSYNC	LCD vertical sync
30	VSYNC	LCD horizontal sync
31	GND	DGND
32	CLK	LCD Pixel Clock
33	GND	DGND
34	X+	Input 1 to the x-plate for the touch screen
35	X-	Input 2 to the x-plate for the touch screen
36	Y+	Input 1 to the y-plate for the touch screen
37	Y-	Input 2 to the y-plate for the touch screen
38	SPI_CLK	SPI clock
39	SPI_SIMO	Slave data in, master data out
40	SPI_SOMI	Slave data out, master data in
41	SPI_ENAN	SPI enable
42	IIC_CLK	IIC master serial clock,
43	IIC_DAT	IIC serial bidirectional data,
44	GND	DGND

45	VDD1	+3.3V
46	VDD2	+3.3V
47	VDD3	+5VDC_IN
48	VDD3	+5VDC_IN
49	RESET	NC
50	PWREN	Power on enable

2.3.2.4 Camera Interface

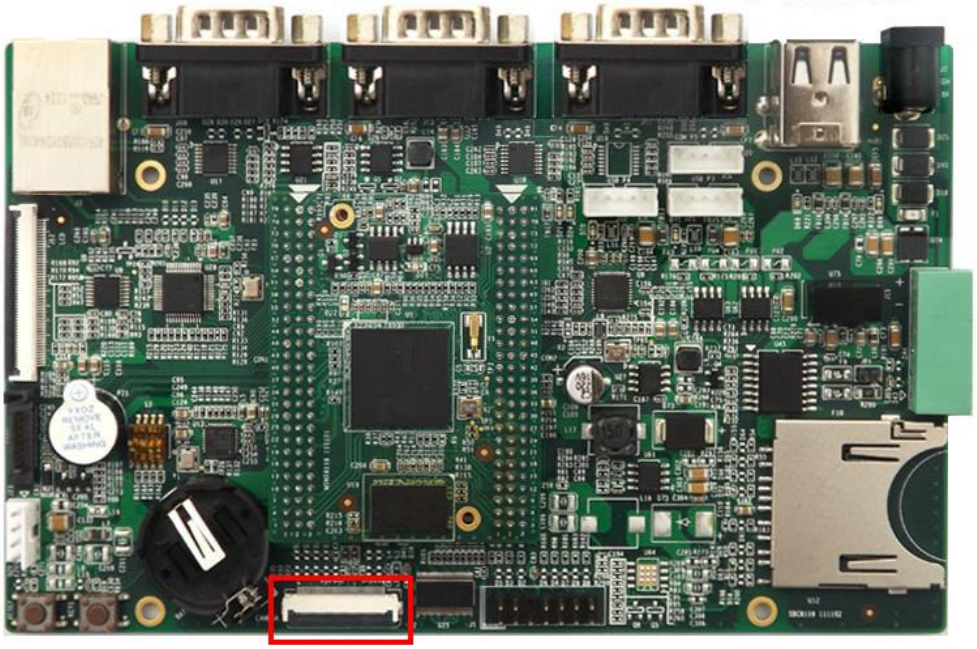


Table 2-6 Camera Interface

J2		
Pin	**Signal**	**Function**
1	GND	DGND
2	D0	NC
3	D1	NC
4	D2	VPIF capture data bit 0
5	D3	VPIF capture data bit 1
6	D4	VPIF capture data bit 2
7	D5	VPIF capture data bit 3
8	D6	VPIF capture data bit 4

9	D7	VPIF capture data bit 5
10	D8	VPIF capture data bit 6
11	D9	VPIF capture data bit 7
12	D10	NC
13	D11	NC
14	GND	DGND
15	PCLK	Pixel clock
16	GND	DGND
17	HSYNC	NC
18	VDD50	+5V
19	VSYNC	NC
20	VDD33	+3.3V
21	XCLKA	NC
22	XCLKB	Test Point
23	GND	DGND
24	FLD	NC
25	WEN	NC
26	STROBE	Test Point
27	SDA	I2C0 serial data
28	SCL	I2C0 serial clock
29	GND	DGND
30	VDD18	+1.8V

2.3.2.5 RS485 Interface

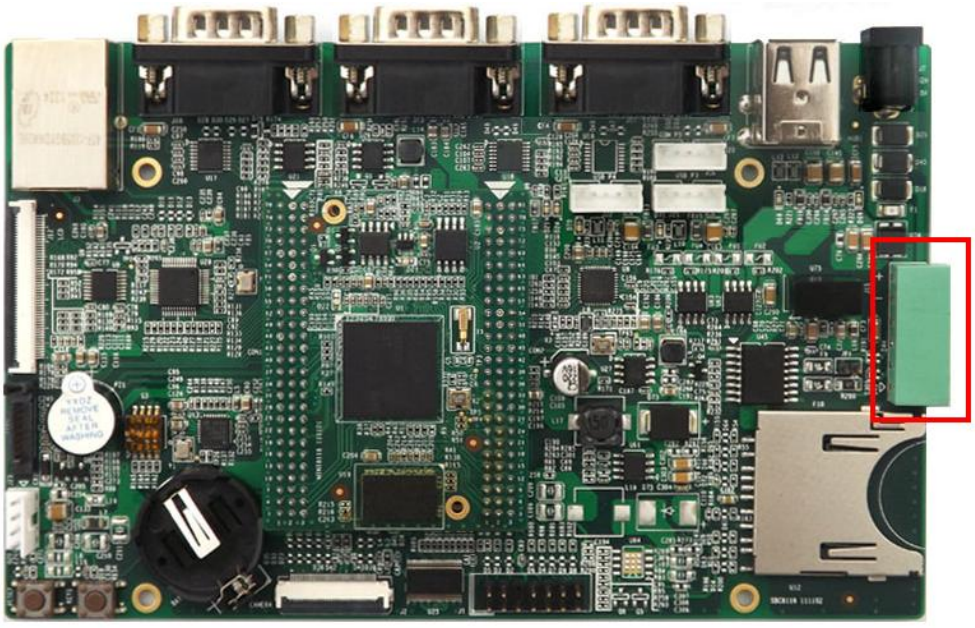


Table 2-7 RS485 Interface

J15		
Pin	**Signal**	**Function**
1	GND_ISO	GND_ISO
2	2	485A
3	3	485B
4	GND_ISO	GND_ISO
5	5	GND_IN
6	6	PWR_EXT

2.3.2.6 Serial Ports

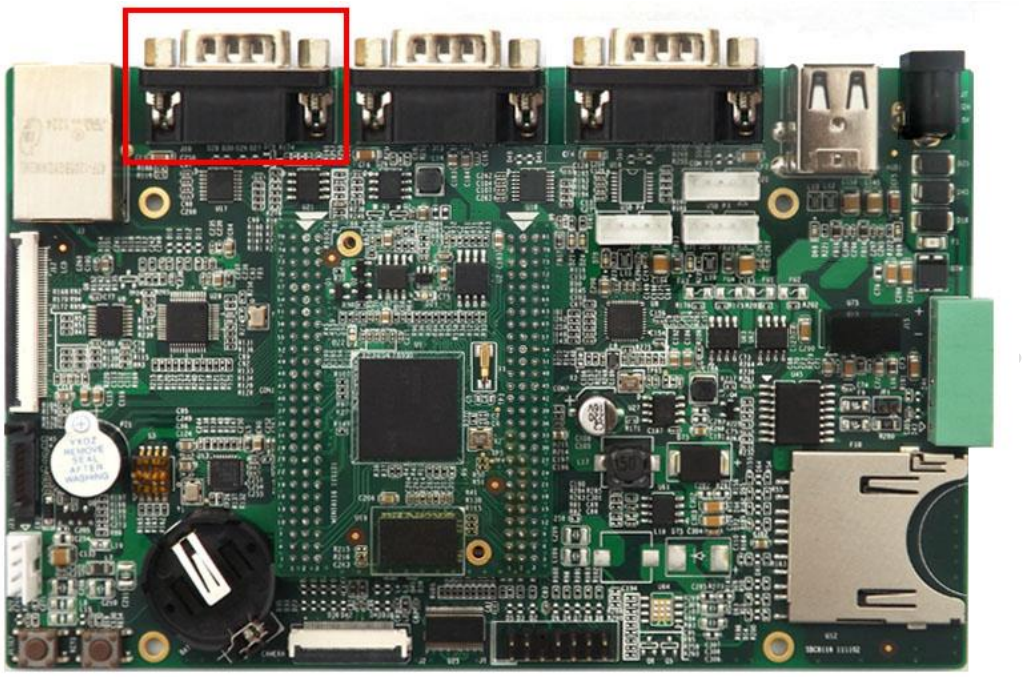


Table 2-8 Debug Serial Port

J10		
Pin	**Signal**	**Function**
1	NC	NC
2	RSA_RXD	Receive data
3	RSA_TXD	Transit data
4	NC	NC
5	DGND	DGND
6	NC	NC
7	RSA_RTS	Request To Send
8	RSA_CTS	Clear To Send
9	NC	NC
10	FGND	FGND
11	FGND	FGND

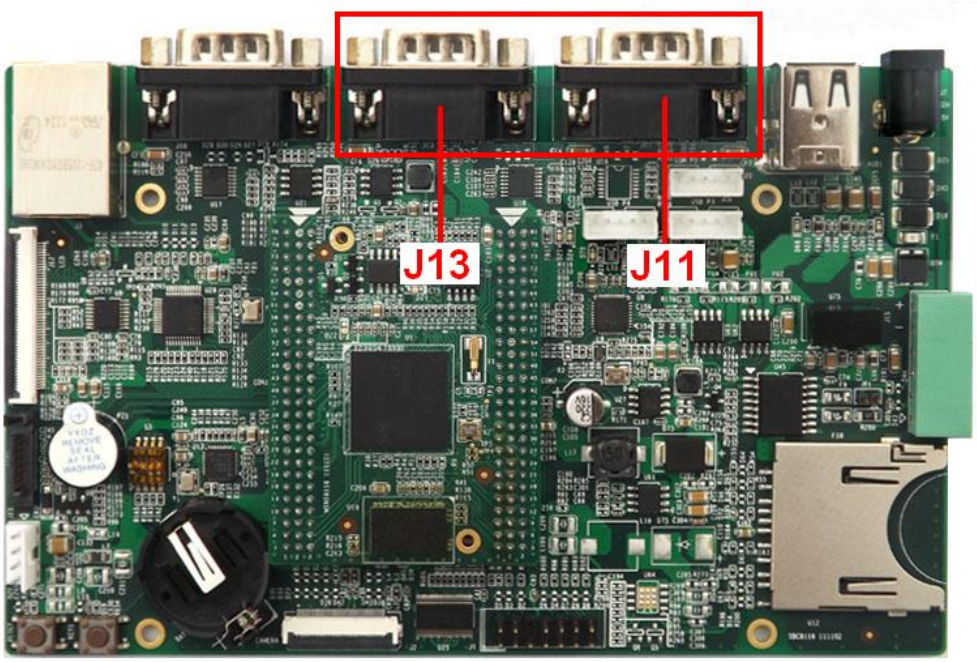


Table 2-9 PRU Serial Ports(RS232 Level)

J11, J13		
Pin	**Signal**	**Function**
1	NC	NC
2	RSA_RX3	Receive data
3	RSA_TX3	Transit data
4	NC	NC
5	DGND	DGND
6	NC	NC
7	NC	NC
8	NC	NC
9	NC	NC
10	FGND	FGND
11	FGND	FGND

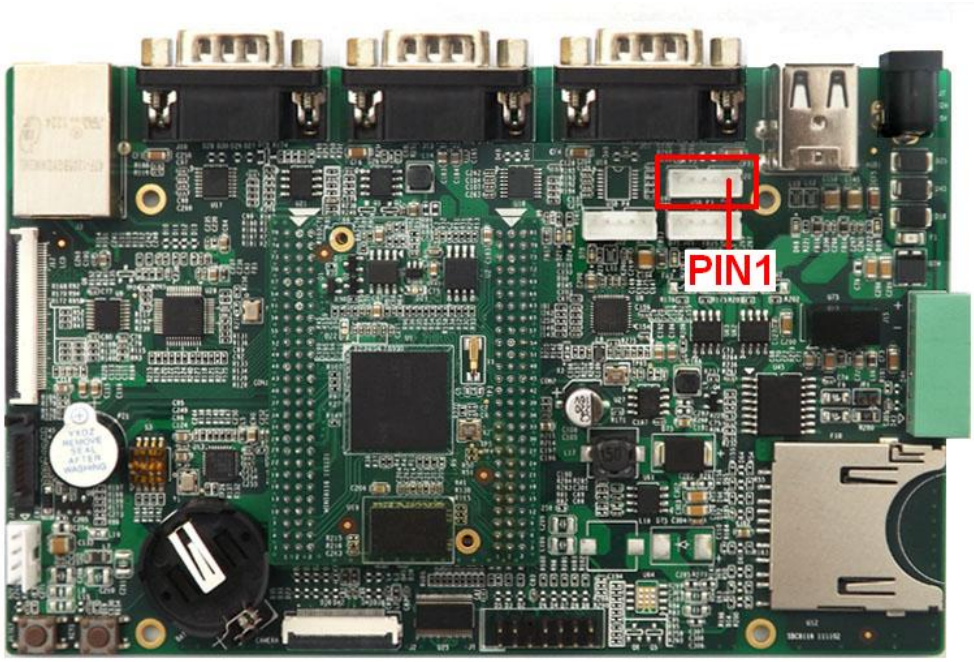


Table 2-10 PRU Serial Port (TTL Level)

J20		
Pin	**Signal**	**Function**
1	1	+3P3V
2	2	RSA_TX5
3	3	RSA_RX5
4	4	DGND
5	5	DGND

2.3.2.7 Ethernet Interface

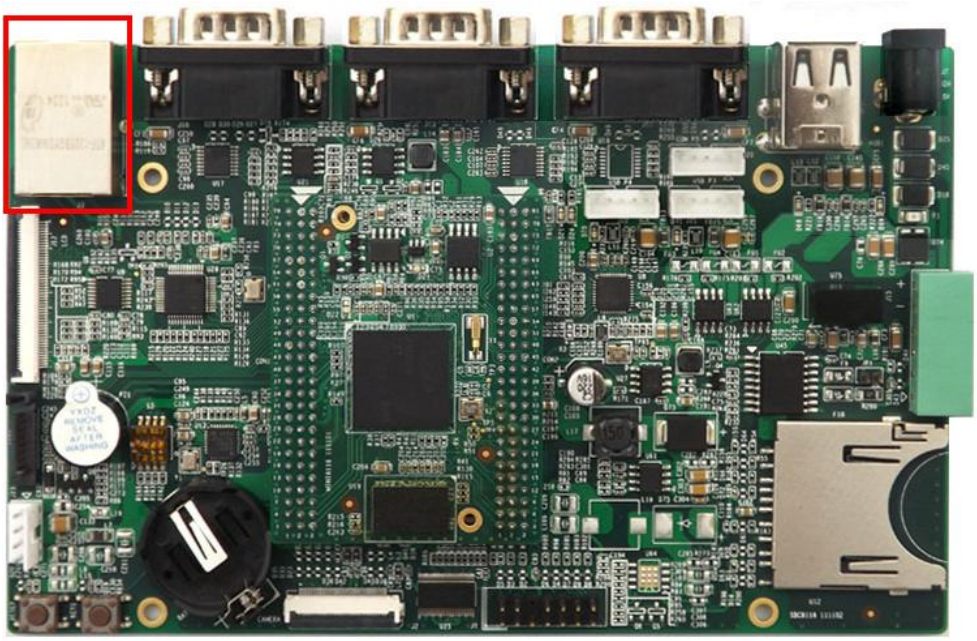


Table 2-11 Ethernet Interface

J3		
Pin	**Signal**	**Function**
1	TX+	TX+ output
2	TX-	TX- output
3	CT	Link to analog transmit power input with 0R resistance
4	4&5	Transformer
5	7&8	Transformer
6	CT	Link to analog transmit power input with 0R resistance
7	RX+	RX+ input
8	RX-	RX- input
9	YEL	Link LED
10	VDD	3.3V Power for LED
11	GRN	Speed LED
12	VDD	3.3V Power for LED
13	CHGND	DGND
14	CHGND	DGND

15	NC	NC
16	NC	NC

2.3.2.8 SATA Interface

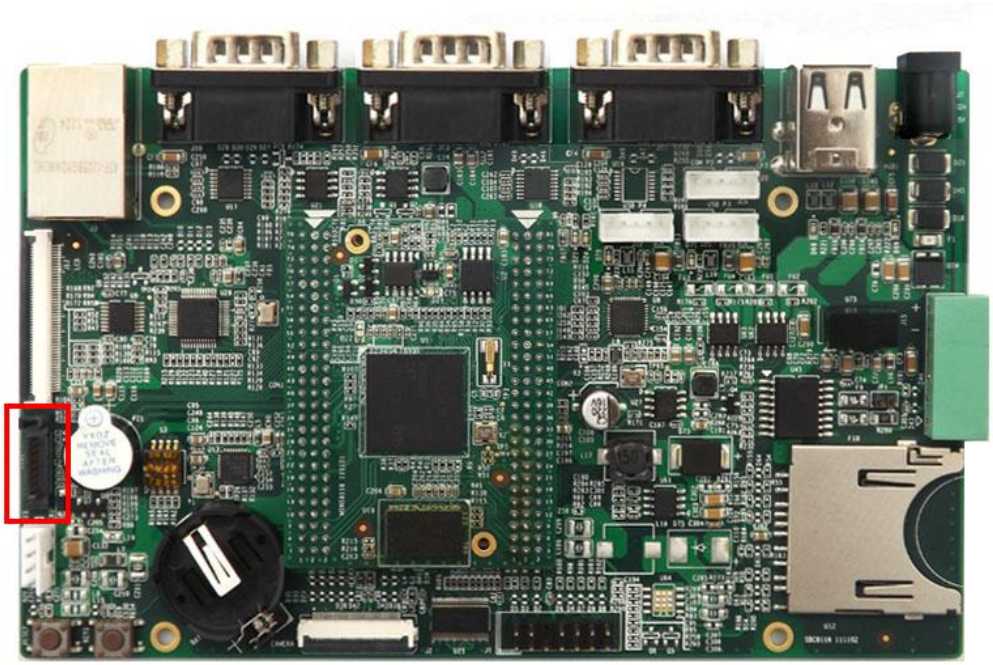


Table 2-12 SATA Interface

J6		
Pin	**Signal**	**Function**
1	GND1	DGND
2	A+	SATA receive data（positive）
3	A-	SATA receive data（negative)
4	GND2	DGND
5	B-	SATA transmit data（negative）
6	B+	SATA transmit data（positive)
7	GND3	DGND
8	HOLD1	NC
9	HOLD2	NC

2.3.2.9 USB HOST Interface

The SBC8118 total of four USB HOST interface, which has two USB HOST interface for the 2.0mm space 5pin connector, another two USB HOST interface for USB -A type .

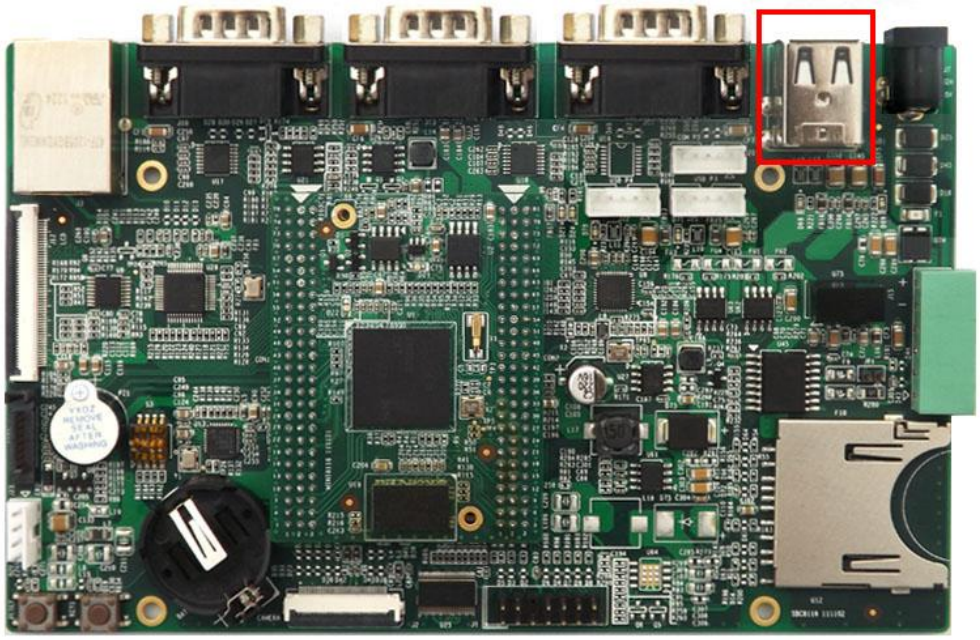


Table 2-13 USB HOST Interface

HUB1		
Pin	**Signal**	**Function**
1	APV	+5V
2	AD-	USB Data1-
3	AD+	USB Data1+
4	GNDA	DGND
5	BPV	+5V
6	BD-	USB Data2-
7	BD+	USB Data2+
8	GNDB	DGND
9	GND	FGND
10	GND	FGND
11	GND	FGND

12	GND	FGND

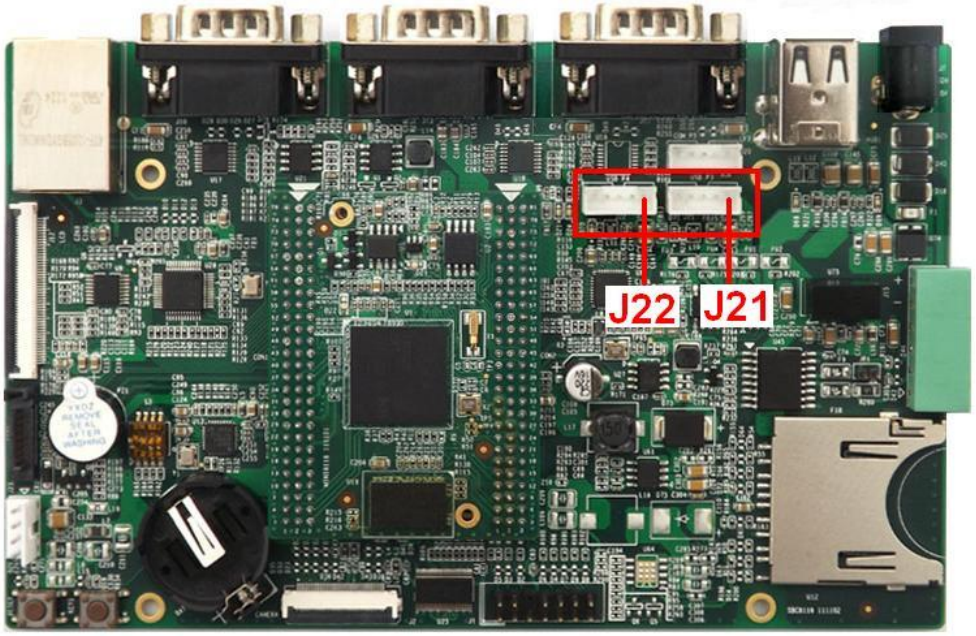


Table 2-14 USB HOST Interface

J21, J22		
Pin	**Signal**	**Function**
1	1	+5V
2	2	DM3
3	3	DP3
4	4	DGND
5	5	DGND

2.3.2.10 MMC/SD Card slot

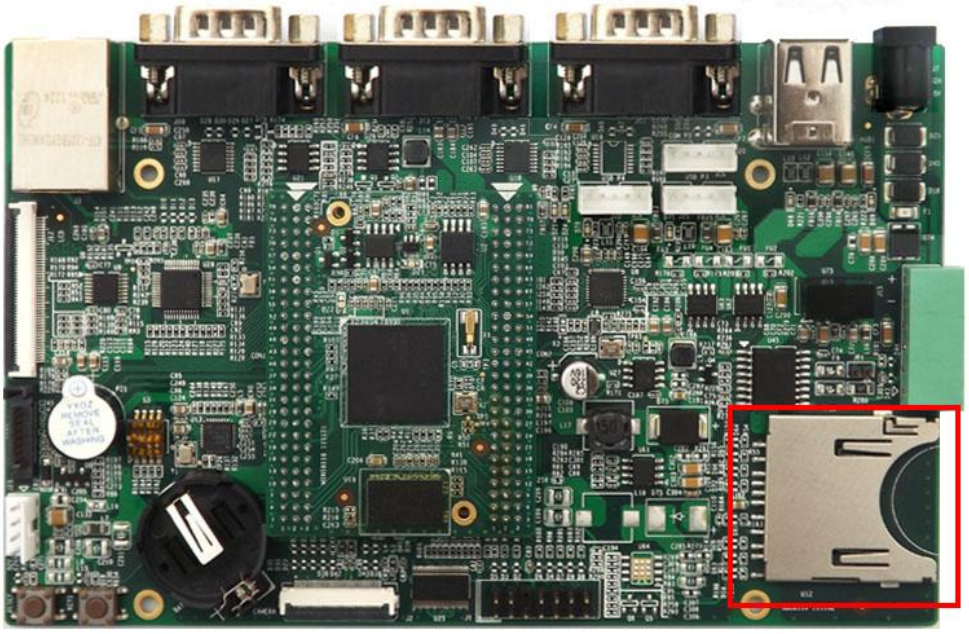


Table 2-15 MMC/SD Card slot

U12		
Pin	**Signal**	**Function**
1	CD/DAT3	Card data 3
2	DCMD	Command Signal
3	VSS	DGND
4	VDD	+3P3V_IO
5	CLK	Clock
6	VSS	DGND
7	TF_DAT0	Card data 0
8	TF_DAT1	Card data 1
9	TF_DAT2	Card data 2
10	SW_1	Card write protected
11	SW_2	Card Detect
12	GND	DGND

2.3.2.11 JTAG Interface

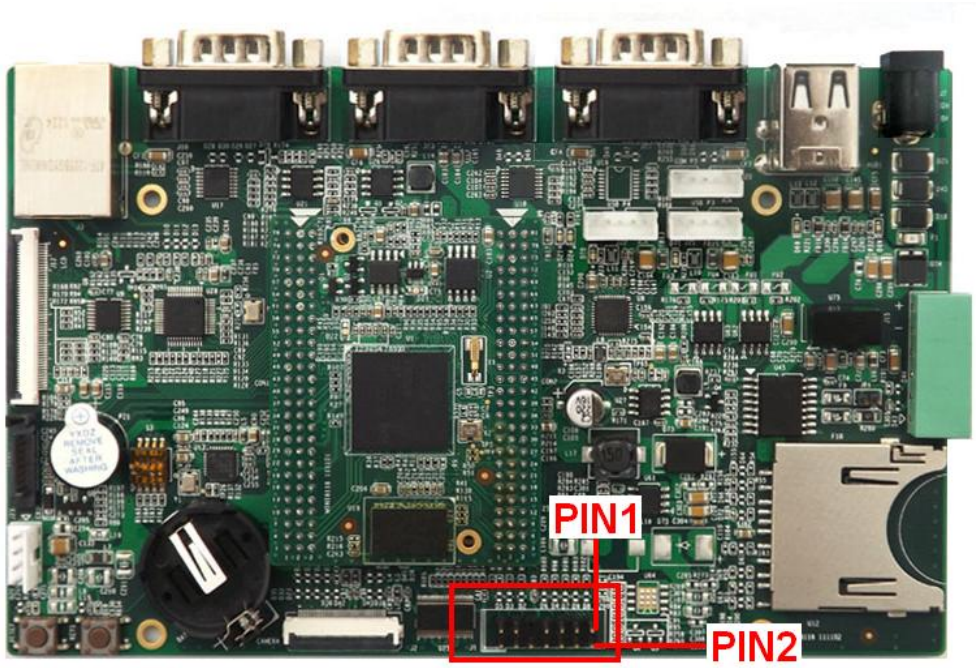


Table 2-16 JTAG Interface

J1		
Pin	**Signal**	**Function**
1	TMS	Test mode select
2	NTRST	Test system reset
3	TDI	Test data input
4	GND	DGND
5	VIO	+3P3V
6	NC	NC
7	TDO	Test data output
8	GND	DGND
9	RTCK	Receive test clock
10	GND	DGND
11	TCK	Test clock
12	DGND	DGND
13	EMU0	Test emulation 0
14	EMU1	Test emulation 1

2.3.2.12 Dip Switch

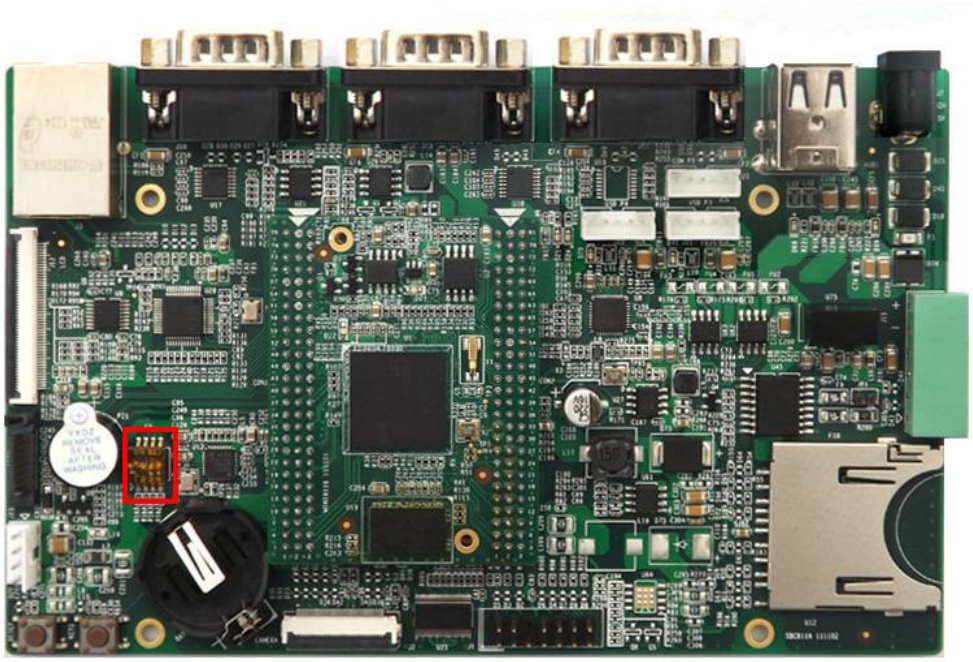


Table 2-17 Dip Switch

S3		
Pin		**Function**
1	PWR_BOOT	Link to PWR_BOOT with 1K resistor
2	DGND	Link to DGND with 1K resistor
3	DGND	Link to DGND with 1K resistor
4	PWR_BOOT	Link to PWR_BOOT with 1K resistor
5	LCD_D12	LCD data bit 12
6	LCD_D11	LCD data bit 11
7	LCD_D10	LCD data bit 10
8	LCD_D9	LCD data bit 9

2.3.2.13 Keys

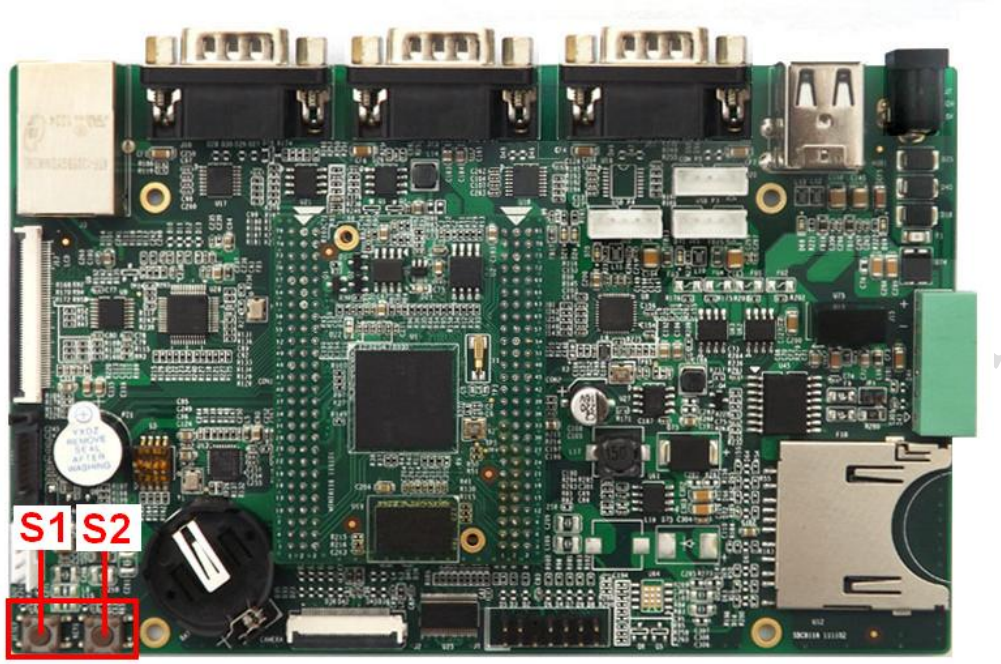


Table 2-18 Keys

Key		
Pin		**Function**
S1	RESET	Device reset input
S2	KEY1	User button

2.3.2.14 LED

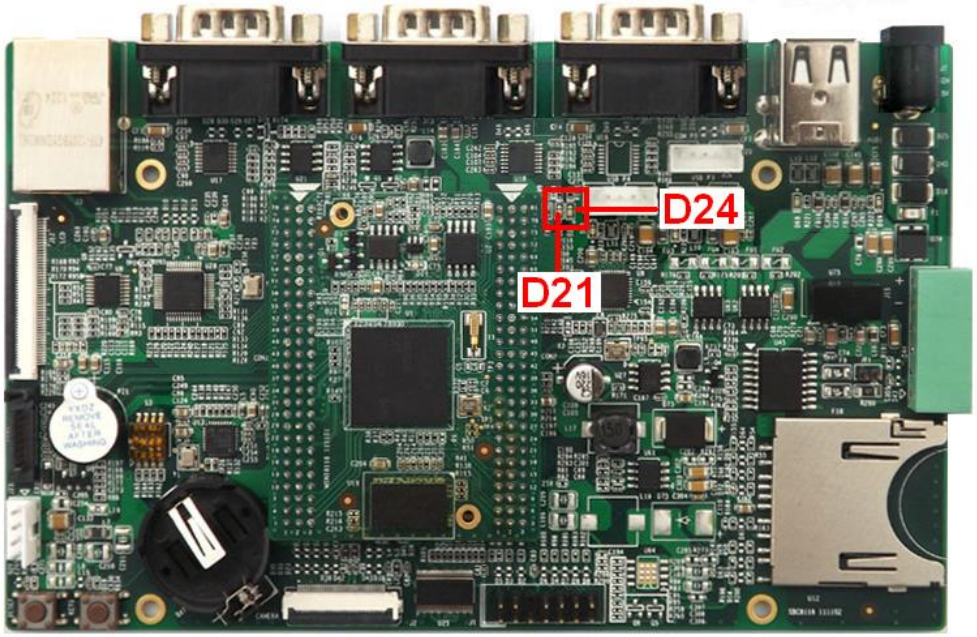


Table 2-19 LED

LED		
Pin	**Signal**	**Function**
D21	User_LED_1	User Definited
D24	+3P3V	3.3V power indicator

2.3.2.15 Buzzer

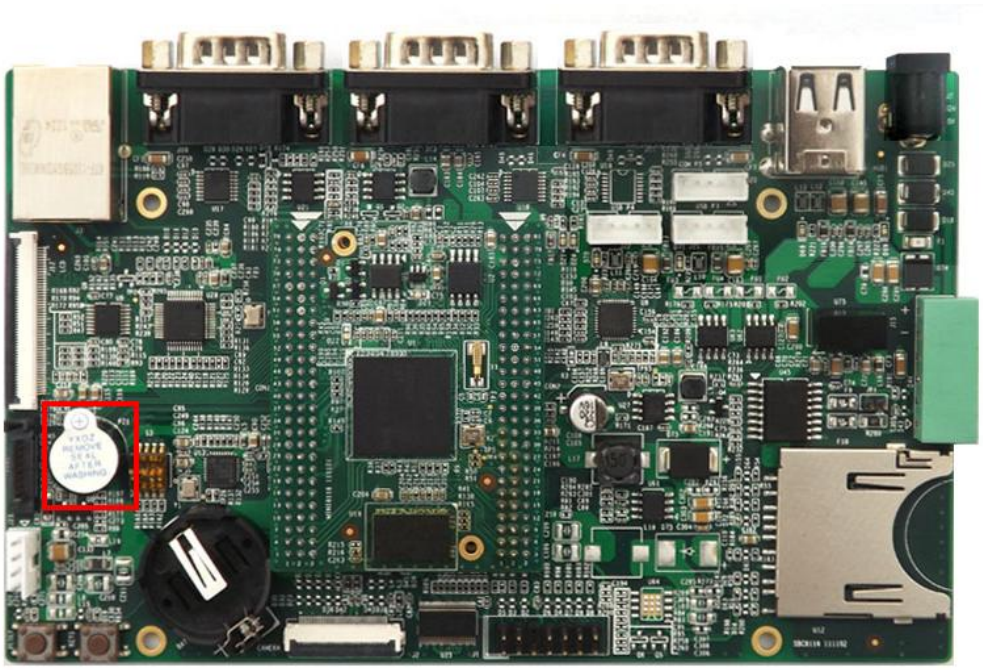


Table 2-20 Buzzer

BUZZER		
Pin	**Signal**	**Function**
PZ1	GPIO	User Definited

Chapter 3 Linux Operating System

3.1 Introduction

This section is intended to provide detailed instruction on Operating System Software development of SBC8118 board.

1) Describes the Software Resources provided by SBC8118.
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 SBC8118
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 SBC8118. A basic software system consists of three parts: u-boot, kernel and rootfs. The Figure 3-1 shows the structure of the system:

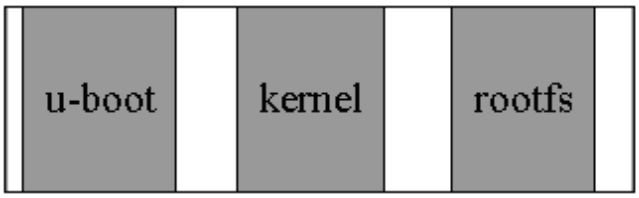


Figure 3-1 System compose map

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

1) U-boot is a bootstrap program. It is used for interacting with users and updating images and leading the kernel;
2) The latest 2.6.x kernel is employed here and it can be customized based on SBC8118
3) Rootfs employs Open-source system “ubifs”.

3.3 BSP Features

SBC8118 BSP is used for customizing and generating the Linux operating system applicable to SBC8118 hardware platform. Users can conduct a secondary development on the basis of this BSP. The BSP in the CD attached in SBC8118 contains the following showed in table 3-1.

Table 3-1 BSP Features

Item		**Note**
BIOS	u-boot	NAND
		MMC/SD
		FAT
		NET
Kernel	Linux-2.6.x	Supports ROM/CRAM/EXT2/EXT3/FAT/NFS/ UBIFS and various file systems
Device Driver Debug	serial	Series driver
	rtc	Hardware clock driver
	net	10/100M Ethernet card driver
	flash	nand flash driver (supports nand boot)
	lcd	TFT LCD driver
	touch screen	Touch screen controller TSC2046 driver
	tf	mmc/sd controller driver
	usb otg	usb otg 2.0 driver (only can be configured as USB Host currently)

	suart	PRU serial driver
	sata	1.5-3.0G SATA driver
	camera	Camera driver (support CAM8000-A camera module)
	button	gpio button driver
	led	user led lamp driver
	buzzer	buzzer driver

3.4 System Development

This section will introduce how to establish a Linux system development platform run on SBC8118 hardware platform with the use of SBC8118 BSP. Details to be provided contain the formation of cross compilation environment, the generation of system image and demonstrate how to customize the system.

The Linux said thereof is ubuntu 10.04 which will be referred as ubuntu

3.4.1 Establishing operating system development environment

Before executing software development on SBC8118, 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.

```
mkdir $HOME/tools
```

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

3.4.1.2 Installation of other 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:

```
cd /media/cdrom/linux/tools
cp mkimage $HOME/tools
chmod a+xr $HOME/tools/mkimage
cp mkfs.ubifs $HOME/tools
chmod a+xr $HOME/tools/mkfs.ubifs
cp ubinize $HOME/tools
chmod a+xr $HOME/tools/ubinize
cp pasm $HOME/tools
chmod a+xr $HOME/tools/pasm
cp ubinize.cfg $HOME/tools
```

3.4.1.3 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:$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.

3.4.2 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/uboot-03.20.00.14.tar.bz2
tar xvf /media/cdrom/linux/source/linux-03.20.00.14.tar.bz2
sudo tar xvf /media/cdrom/linux/source/rootfs.tar.bz2
```

When the above steps are finished, the current directory will generate uboot-03.20.00.14, linux-03.20.00.14 and rootfs directories.

3.4.2.2 X-loader image generation

SBC8118 supports UART boot, NAND Flash and SPI Flash boot. The image files are different with the different boot modes, and the corresponding methods for mapping are different too.

1) To generate image file u-boot

```
cd uboot-03.20.00.14
make distclean
make mini8118_config
make
```

When the above steps are finished, the current directory will generate the file u-boot which we need.

Copy current u-boot file to Windows system, the AISgen tool will launch on Windows.

2) To generate the u-boot-uart-ais.bin for UART start-up

Put the new u-boot to the folder d:\SBC8118\.

Make sure you had installed AISgen_d800k006_Install_v1.7.exe from CD under the folder

CD\linux\tools\.

Click Windows XP -> Start -> All Programs -> Texas Instruments -> AISgen for D800K006 -> AISgen for D800K006 to open AISgen tool.

a) Launches the ASIgen, click “File”-> “Load Configuration” to open AM1808-UART.cfg from the CD under the folder CD\linux\tools\.

b) Added the u-boot under the folder d:\SBC8118\ to the [ARM Application File:]

c) Sets the output file path [AIS Output File] as d:\SBC8118\u-boot-uart-ais.bin.

d) Click [Generate AIS], u-boot-uart-ais.bin had generated under the folder d:\SBC8118.

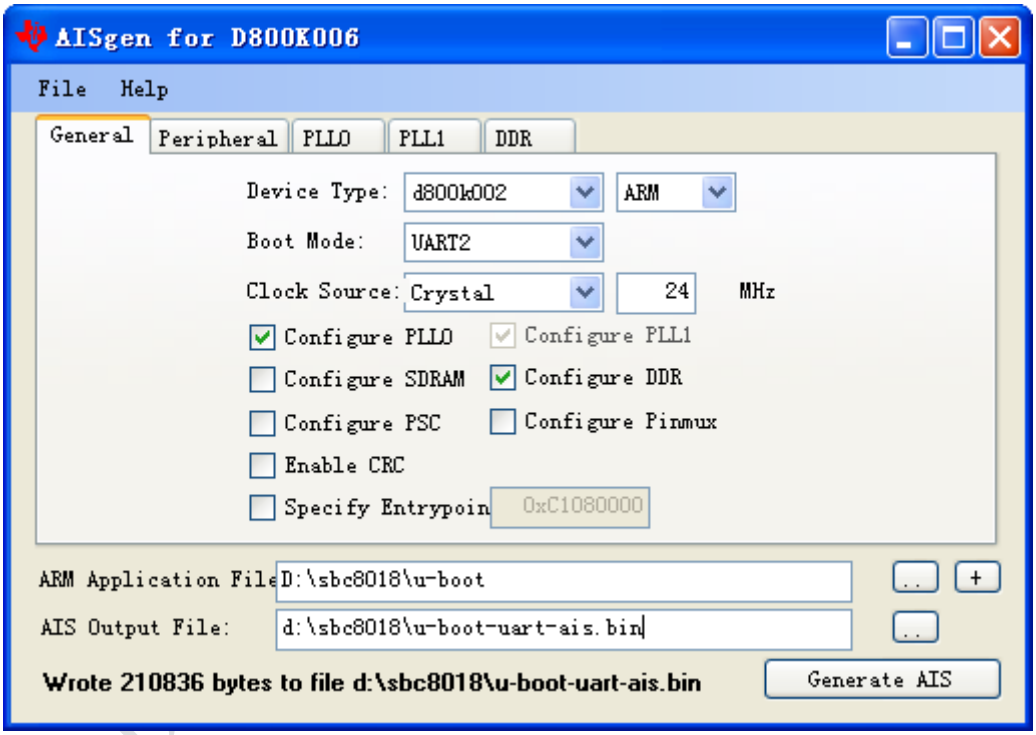


Figure 3-2 ASIgen for D800K006

3) To generate u-boot-nand-ais.bin for NAND Flash boot up

a) Launches the ASIgen, click “File”-> “Load Configuration” to open “AM1808-NAND.cfg” from the CD under the folder CD\linux\tools\.

b) Added the u-boot under the folder d:\SBC8118\ to the [ARM Application File:]

c) Sets the output file path [AIS Output File] as d:\SBC8118\u-boot-nand-ais.bin.

d) Click [Generate AIS], u-boot-nand-ais.bin had generated under the folder d:\SBC8118.

4) To generate u-boot-nand-ais.bin for SPI Flash boot up

a) Launches the ASIgen, click “File”-> “Load Configuration” to open “AM1808-SPI.cfg” from the CD under the folder CD\linux\tools\.

b) Added the u-boot under the folder d:\SBC8118\ to the [ARM Application File:]

c) Sets the output file path [AIS Output File] as d:\SBC8118\u-boot-spi-ais.bin.

d) Click [Generate AIS], u-boot-spi-ais.bin had generated under the folder d:\SBC8118.

3.4.2.3 Kernel compilation

```
cd linux-03.20.00.14
make distclean
make mini8118_defconfig
make uImage
```

When the above steps are finished, the required uImage file will be generated under the directory arch/arm/boot.

3.4.2.4 Generation of file system

```
cd $HOME/work
sudo -S $HOME/tools/mkfs.ubifs -r rootfs -m 2048 -e 129024 -c 812 -o ubifs.img
sudo -S $HOME/tools/ubinize -o rootfs.img -m 2048 -p 128KiB -s 512 $HOME/tools/ubinize.cfg
```

After above operations are executed, the current directory will generate the file rootfs.img which we need.

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-03.20.00.14/arch/arm/configs/mini8118_defconfig

User can carry out system customization on this basis:

```
cd linux-03.20.00.14
cp arch/arm/configs/mini8118_defconfig .config
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
```

The system customization will be described below by taking WCDMA8000-U module supports as an example:

1) Select Device driver

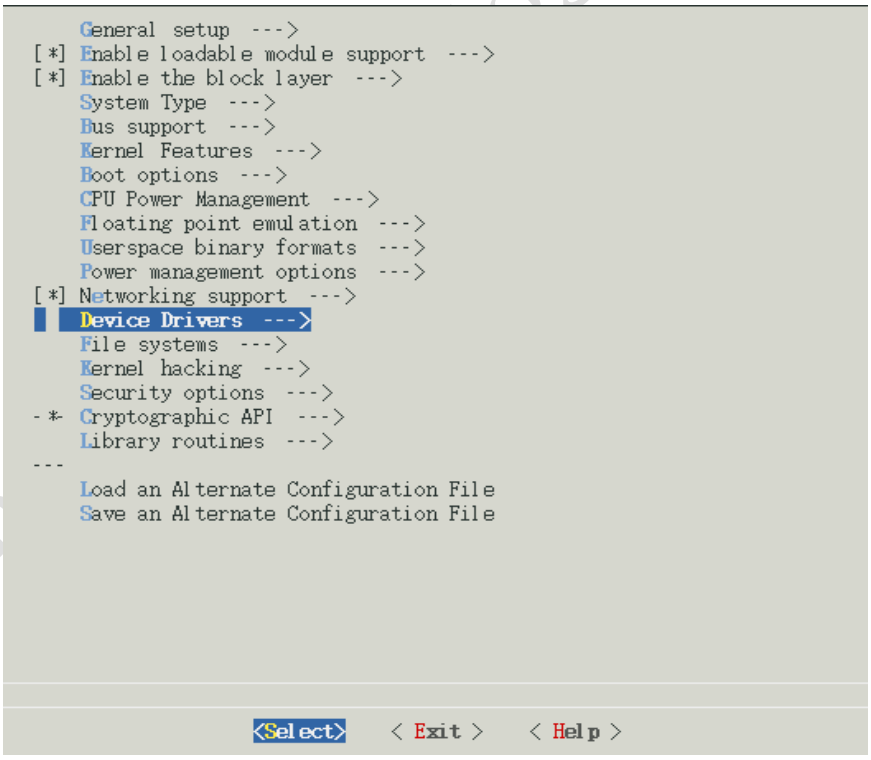


Figure 3-3

2) Select USB support

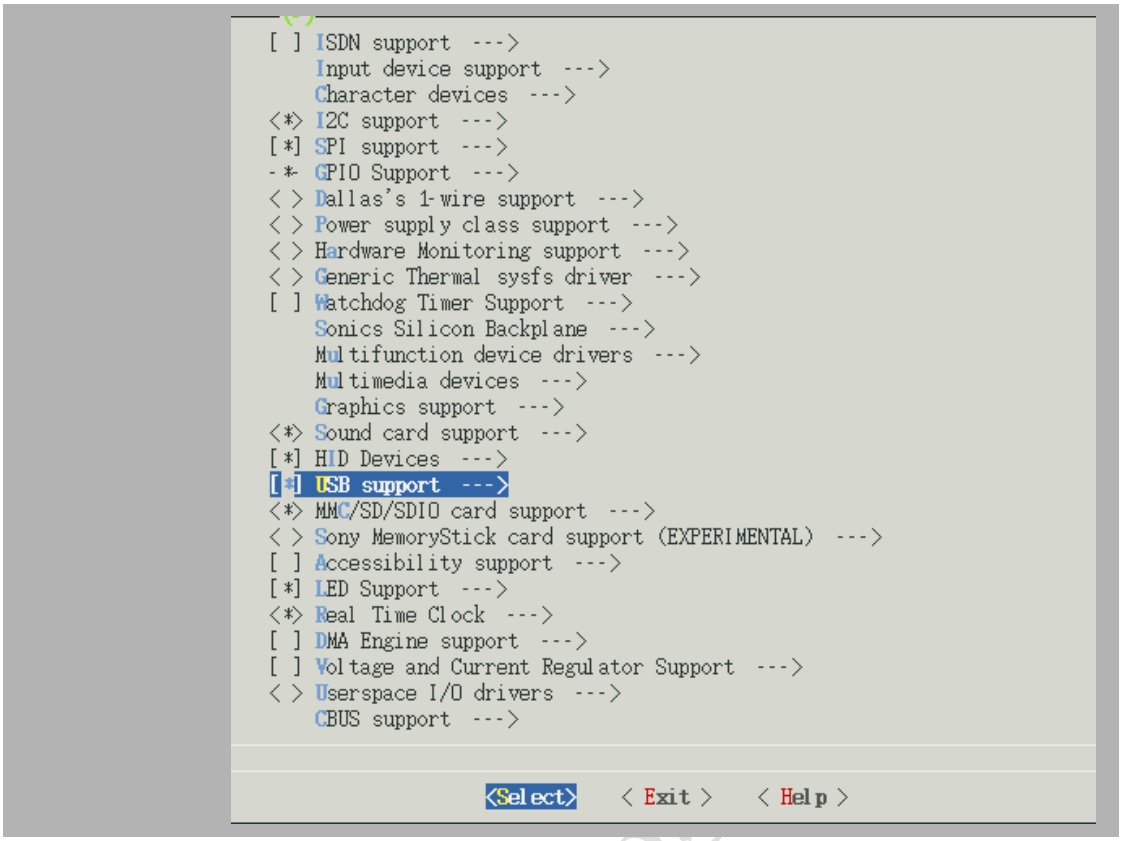


Figure 3-4

3) Select USB Serial Converter Support type as M

```
<M>    USB Serial Converter support   --->

    [*]    USB Generic Serial Driver
```

4) Return to the Device Drivers levels , and then configure the following options

```
[*] Network device support   --->

    <*>    PPP (point-to-point protocol) support

    [*]     PPP multilink support (EXPERIMENTAL)

    [*]     PPP filtering

    <*>       PPP support for async serial ports

    <*>       PPP support for sync tty ports

    <*>       PPP Deflate compression

    <*>       PPP BSD-Compress compression

    <*>       PPP MPPE compression (encryption) (EXPERIMENTAL)

    <*>       PPP over Ethernet (EXPERIMENTAL)

    <*>       PPP over L2TP (EXPERIMENTAL)
```

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 usbserial.ko will be generated under the directory drivers/usb/serial.

3.4.3.3 Test

Refer to chapter 3.7 Updated of system updated kernel image. copy file usbserial.ko to the TF card.

Reboot the system, execute the following commands to load usbserial.ko driver:

```
root@SBC8118:~# cd /media/mmcblk0p1/
root@SBC8118:/media/mmcblk0p1# insmod /usr/lib/usbserial.ko vendor=0x05c6 product=0x0015
usbcore: registered new interface driver usbserial
USB Serial support registered for generic
usbcore: registered new interface driver usbserial_generic
usbserial: USB Serial Driver core
```

At this time, insert the WCDMA8000-U module into the USB HOST interface, it can be recognized by the system.

Please make sure that the kernel image has been updated, otherwise, module usbserial.ko will fail to load and the similar tips will show:

```
insmod: cannot insert '/media/mmcblk0p1/usbserial.ko': Device or resource busy
```

3.5 Introduction of driver

3.5.1 LED

Kernel Driver reference path:

```
linux-03.20.00.14/drivers/leds/led-class.c
linux-03.20.00.14/drivers/leds/led-core.c
linux-03.20.00.14/drivers/leds/led-gpio.c
```

Platform devices

Located in the source file[linux-03.20.00.14/arch/arm/mach-davinci/board-mini8118.c]

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

Device Registration:

Located in [linux-03.20.00.14/arch/arm/mach-davinci/board-mini8118.c]

```
 platform_device_register (&leds_gpio);
```

menuconfig Configuration

```
Device Drivers ---> LED Support --->LED Support for GPIO connected LEDs
```

3.5.2 Button

Kernel Driver reference path:

```
linux-03.20.00.14/drivers/input/keyboard/gpio_keys.c
```

Platform devices

Located in the source file[linux-03.20.00.14/arch/arm/mach-davinci/board-mini8118.c]

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

Device Registration:

Located in the source file [linux-03.20.00.14/arch/arm/mach-davinci/board-mini8118.c]

```
platform_device_register (&button_gpio) ;
```

menuconfig Configuration:

```
Device Drivers ---> Input device support --->Keyboards --->GPIO_Buttons
```

3.5.3 Touch Screen

Kernel Driver reference path:

```
linux-03.20.00.14/drivers/input/touchscreen/ads7846.c
```

Platform devices

Located in the source file [linux-03.20.00.14/arch/arm/mach-davinci/board-mini8118.c]

```
static struct spi_board_info da850_spi_board_info[] = {
        [0] = {
                .modalias               = "ads7846",
                .bus_num                = 1,
                .chip_select            = 1,
                .max_speed_hz           = 600000,
                .platform_data          = &ads7846_config,
                .mode                   = SPI_MODE_0,
                /* max sample rate at 3V */
        },
};
```

Device Registration:

Located in the source file [linux-03.20.00.14/arch/arm/mach-davinci/board-mini8118.c]

```
        da850_init_spi1(0, da850_spi_board_info,
                ARRAY_SIZE(da850_spi_board_info));
```

menuconfigConfiguration:

```
Device Drivers ---> Input device support --->Touchscreens
--->ADS7846/TSC2046 and ADS7843 based touchscreens
```

3.5.4 RTC

Kernel Driver reference path:

```
linux-03.20.00.14/drivers/rtc/rtc-omap.c
```

Platform devices

Located in the source file [linux-03.20.00.14/arch/arm/mach-davinci/devices-da8xx.c]

```
struct platform_device da8xx_rtc_device = {
        .name     = "omap_rtc",
        .id       = -1,
        .num_resources = ARRAY_SIZE(da8xx_rtc_resources),
        .resource   =   da8xx_rtc_resources,
        },
};
```

Device Registration:

Located in the source file [linux-03.20.00.14/arch/arm/mach-davinci/board-mini8118.c]

```
da8xx_register_rtc(void);
```

menuconfig Configuration:

```
Device Drivers   ---> Real Time Clock   ---> TI OMAP1
```

3.5.5 TF Card

Kernel Driver reference path:

```
linux-03.20.00.14/drivers/mmc/host/davinci_mmc.c
```

Platform devices

Located in the source file [linux-03.20.00.14/arch/arm/mach-davinci/devices-da8xx.c]

```
struct platform_device da8xx_mmcsd0_device = {
        .name     = "davinci_mmc",
        .id       = 0,
        .num_resources = ARRAY_SIZE(da8xx_mmcsd0_resources),
        .resource     =   da8xx_mmcsd0_resources,
};
```

Device Registration:

Located in the source file [linux-03.20.00.14/arch/arm/mach-davinci/board-mini8118.c]

```
da8xx_register_mmcsd0(&da850_mmc_config);
```

menuconfigConfiguration:

```
Device Drivers ---> MMC/SD/SDIO card support ---> TI DAVINCI Multimedia Card Interface support
```

3.5.6 USB HOST

The USB Host function use USB OTG interface.

Kernel Driver reference path:

```
linux-03.20.00.14/drivers/usb/musb/musb_core.c
linux-03.20.00.14/drivers/usb/musb/da8xx.c
```

Platform devices

Located in the source file [linux-03.20.00.14/arch/arm/mach-davinci/usb.c]

```
pdev = platform_device_alloc(name, -1);
```

Device Registration:

Located in the source file [linux-03.20.00.14/arch/arm/mach-davinci/usb.c]

```
platform_device_add(pdev);
```

menuconfig Configuration:

```
Device Drivers ---> USB support ---> Inventra Highspeed Dual Role Controller (TI, ADI, ...) ---> Driver Mode (USB Host)
```

3.5.7 SUART

SUART function is implemented with AM1808 PRUSS

Kernel Driver reference path:

```
linux-03.20.00.14/drivers/serial/omapl_pru/suart/ti_omapl_pru_suart.c
```

Platform devices

Located in the source file [linux-03.20.00.14/arch/arm/mach-davinci/devices-da8xx.c]

```
struct platform_device omapl_pru_suart_device = {
    .name     = "ti_omapl_pru_suart",
```

```
        .id                 = 1,
        .num_resources    = ARRAY_SIZE(omapl138_pru_suart_resources),
        .resource           = omapl138_pru_suart_resources,
    };
```

Device Registration:

Located in the source file [linux-03.20.00.14/arch/arm/mach-davinci/devices-da8xx.c]

```
            return platform_device_register(&omapl_pru_suart_device);
```

menuconfig Configuration:

```
    Device Drivers ---> Character devices ---> Serial drivers --->PRU based SUART emulation for OMAPL
```

3.5.8 UART /RS485

Kernel Driver reference path:

```
    linux-03.20.00.14/drivers/serial/8250.c
```

Platform devices

Located in the source file [linux-03.20.00.14/arch/arm/mach-davinci/devices-da8xx.c]

```
    struct platform_device da8xx_serial_device = {
            .name     = "serial8250",
            .id       = PLAT8250_DEV_PLATFORM,
            .dev      = {
                        .platform_data    = da8xx_serial_pdata,
            },
    };
```

Device Registration:

Located in the source file[linux-03.20.00.14/arch/arm/mach-davinci/board-mini8118.c]

```
            davinci_serial_init(&da850_evm_uart_config);
```

menuconfig Configuration:

```
    Device Drivers ---> Character devices --->Serial drivers --->8250/16550 and compatible serial support
```

3.5.9 SATA

Kernel Driver reference path:

```
linux-03.20.00.14/drivers/ata/ahci.c
```

Platform devices

Located in the source file[linux-03.20.00.14/arch/arm/mach-davinci/devices-da8xx.c]

```
struct platform_device da8xx_serial_device = {
        .name     = "ahci",
        .id       = -1,
        .dev      = {
                .platform_data    = &da850_ahci_pdata,
                .resource         = da850_ahci_resources},
        .num_resourtces   =   ARRAY_SIZE(da850_ahci_resources),
        .resource         =   da850_ahci_resources,
};
```

Device Registration:

Located in the source file [linux-03.20.00.14/arch/arm/mach-davinci/board-mini8118.c]

```
        platform_device_register (&da850_ahci_device);
```

menuconfig Configuration:

```
Device Drivers   ---> Serial ATA and Parallel ATA drivers   --->AHCI SATA support
```

3.5.10 CAMERA

Kernel Driver reference path:

```
linux-03.20.00.14/drivers/media/video/tvp514x.c
```

Platform devices

Located in the source file[linux-03.20.00.14/arch/arm/mach-davinci/devices-da8xx.c]

```
static struct vpif_capture_config da850_vpif_capture_config = {
        .setup_input_channel_mode = da850_setup_vpif_input_channel_mode,
        .intr_status = da850_vpif_intr_status,
        .subdev_info = da850_vpif_capture_sdev_info,
```

```
        .subdev_count = ARRAY_SIZE(da850_vpif_capture_sdev_info),
        .chan_config[0] = {
                .inputs = da850_ch0_inputs,
                .input_count = ARRAY_SIZE(da850_ch0_inputs),
        },
        .chan_config[1] = {
                .inputs = da850_ch1_inputs,
                .input_count = ARRAY_SIZE(da850_ch1_inputs),
        },
        .card_name          = "DA850/OMAP-L138 Video Capture",
};
```

Device Registration

Located in the source file[linux-03.20.00.14/arch/arm/mach-davinci/board-daxx.c]

```
ret = da850_register_vpif_capture(&da850_vpif_capture_config);
```

menuconfig Configuration:

Device Drivers ---> Multimedia support --->Video capture adapters --> DaVinci Video VPIF Capture

3.5.11 Ethernet

Kernel Driver reference path:

linux-03.20.00.14/drivers/net/davinci_emac.c

Platform devices

Located in the source file[linux-03.20.00.14/arch/arm/mach-davinci/devices-da8xx.c]

```
static struct platform_device da8xx_emac_device = {
        .name                = "davinci_emac",
        .id                  = 1,
        .dev = {
                .platform_data    = &da8xx_emac_pdata,
        },
        .num_resources    = ARRAY_SIZE(da8xx_emac_resources),
```

```
        .resource       = da8xx_emac_resources,
};
```

Device Registration

Located in the source file[linux-03.20.00.14/arch/arm/mach-davinci/board-mini8118.c]

```
ret = da8xx_register_emac();
```

menuconfig Configuration:

```
Device Drivers --->Network device support --->Ethernet (10 or 100Mbit) --->TI DaVinci EMAC Support
```

3.6 Driver Development

3.6.1 Driver For The LED

1）Device Definition

linux-03.20.00.14/arch/arm/mach-davinci/board-mini8118.c

The driver will introduce how to create the driver on the kernel and enable the LED1, LED2, the kernel configuration respectively are: led1(GPIO5.9), led2(GPIO5.11), , low level is enable:

```
static struct gpio_led gpio_leds[] = {
        {
                .name                   = "led1",
                .default_trigger        = "heartbeat",
                .gpio                   = GPIO_TO_PIN(5,9),
                .active_low             = true,
        }, {
                .name                   = "led2",
                .default_trigger        = "none",
                .gpio                   = GPIO_TO_PIN(5,11),
                .active_low             = true,
        },
};
```

```
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）Pin Definitions

linux-03.20.00.14/arch/arm/mach-davinci/da850.c

```
const short mini8118_led_pins[] __initdata = {
                DA850_GPIO5_9, DA850_GPIO5_11,
                -1
};
```

3）Pin Setup

linux-03.20.00.14/arch/arm/mach-davinci/board-mini8118.c

```
                ret = da8xx_pinmux_setup(mini8118_led_pins);
```

The “Da8xx_pinmux_setup” function prototype is in linux-03.20.00.14/arch/arm/mach-davinci/mux.c

3）Driver design

linux-03.20.00.14/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);
}
```

3.6.2 Driver For The Key

1）Device Definition

linux-03.20.00.14/arch/arm/mach-davinci/board-mini8118.c

Setup GPIO5.10 as “menu” key, return value as “KEY_F1”, triggered on low level;

```
static struct gpio_keys_button gpio_buttons[] = {
    {
        .code    = BTN_0,
        .gpio    = GPIO_TO_PIN(5,10),
        .desc    = "menu",
        .wakeup = 0,
    },
};

static struct gpio_keys_platform_data gpio_key_info = {
    .buttons  = gpio_buttons,
    .nbuttons    = ARRAY_SIZE(gpio_buttons),
};
static struct platform_device buttons_gpio = {
    .name        = "gpio-keys",
    .id        = -1,
    .dev       = {
                .platform_data = &gpio_key_info,
    },
};
```

2）Pin Definitions

linux-03.20.00.14/arch/arm/mach-davinci/da850.c

```
const short mini8118_button_pins[] __initdata = {
        DA850_GPIO5_10,
        -1
};
```

3）Pin Setup

linux-03.20.00.14/arch/arm/mach-davinci/board-mini8118.c

```
ret = da8xx_pinmux_setup(mini8118_button_pins);
```

The “da8xx_pinmux_setup” function prototype is in

linux-03.20.00.14/arch/arm/mach-davinci/mux.c

4）Driver Design

linux-03.20.00.14/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);
}
```

3.7 Updated of system

SBC8118 single board computer default NAND Flash is installed with Linux + 4.3-inch screen display. It can be booted without connecting TF card once it's powered on or reset. And input the "root" to enter Linux system.

Please make sure the toggle switch [S3] as below:

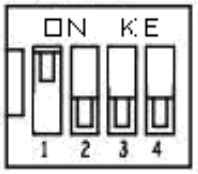


Figure 3-5 Boot-up from NAND Flash

3.7.1 Boot-up from Serial port

1) Establish the hardware environment, make sure the toggle switch [S3] as below:

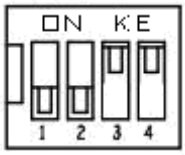


Figure 3-6 Boot-up from Serial port

2) Open the AISgen_d800k006_Install_v1.7.exe:

- Windows XP -> Start -> All Programs -> Texas Instruments -> AISgen for D800K006 -> UART Boot Host
- Add u-boot-uart-ais.bin [Directory: CD\linux\image\] to the "AIS-File" as below:

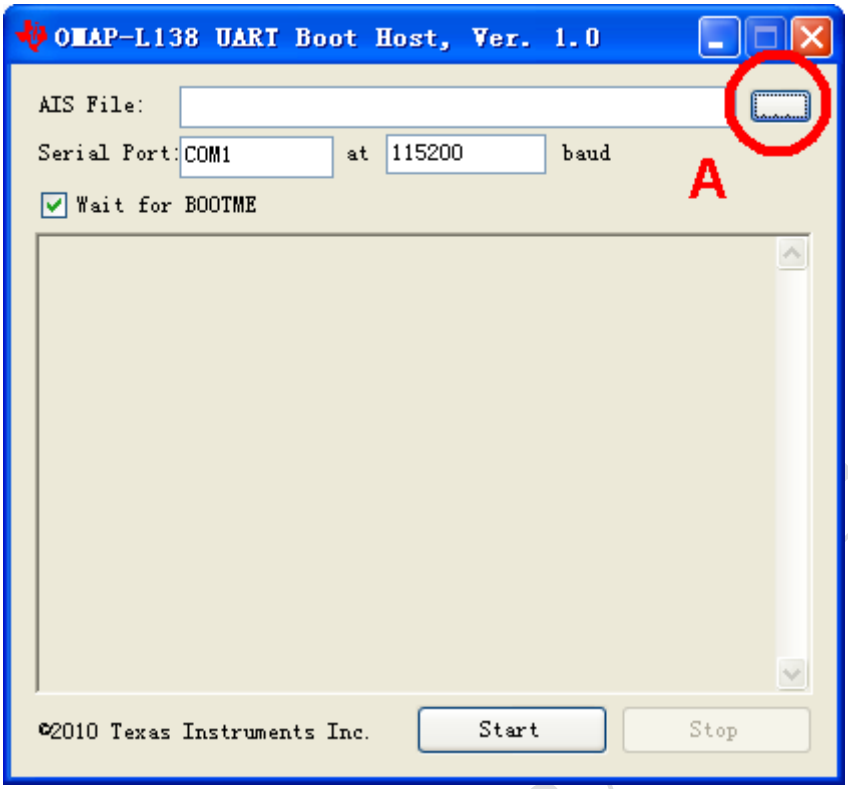


Figure 3-7

3) If the using computer serial port is not COM1, Please change the serial port.

4) Click the “Start” and power on the single board computer to boot-up from serial port.

5) Wait for moment, the target window will display “(Serial Port): Closing COM1.”, close the tool and open the Hyperterminal to catch the serial port information.

User should open Hyperterminal and Input any key to enter U-BOOT prompts in three seconds, or else U-BOOT will load default parameter.

3.7.2 Updated images from net

SBC8118 can through net, TF card to update images with u-boot prompts, this manual mainly teach you how to update image with net.

IP will be giving an example as:

PC: 192.192.192.154

Single board computer: 192.192.192.215

1) PC TFTP service

a) As shown in the following figure, launch the tftpd32.exe from the cd under the

folder CD\linux\tools, and click “Browse” to set the sharing space, the directory will be giving an example as “D:\SBC8118”.

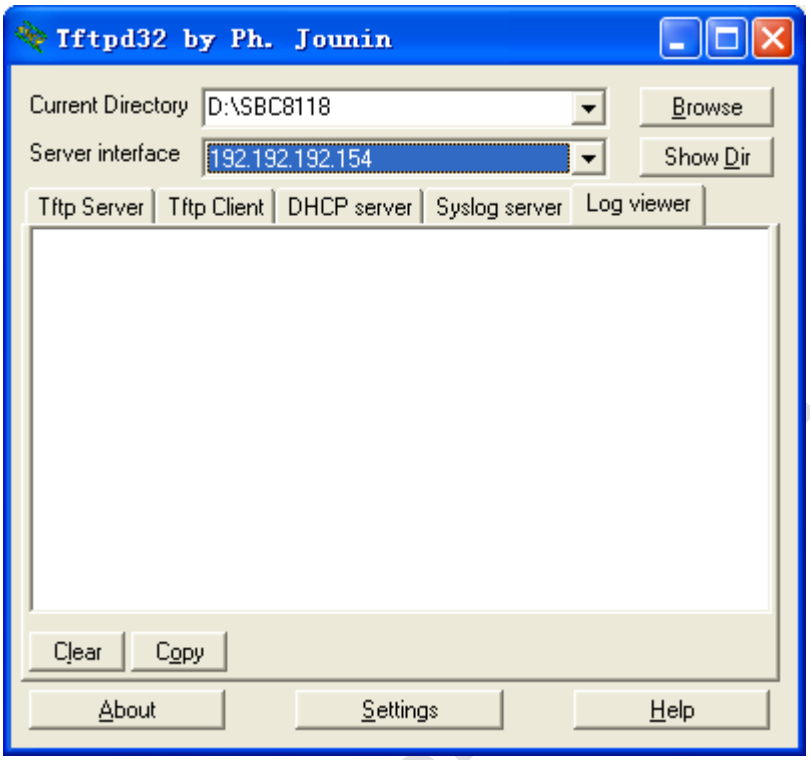


Figure 3-8 tftpd32 tool

b) Copy u-boot-nand-ais.bin, u-boot-spi-ais.bin, uImage, rootfs.img from the CD under the folder CD\linux\image\ to the folder d:\SBC8118 .

2) U-BOOT prompts

Input the commands with U-BOOT prompts as below:

a) Set the environment with “ipaddr” and “serverip”:

```
U-Boot > setenv ipaddr 192.192.192.215
U-Boot > setenv serverip 192.192.192.154
```

b) Erase the NAND Flash

```
U-Boot > nand erase
NAND erase: device 0 whole chip
Skipping bad block at  0x0ff80000
Skipping bad block at  0x0ffa0000
Skipping bad block at  0x0ffc0000
Skipping bad block at  0x0ffe0000
```

```
OK
```

c) Write U-BOOT to NAND Flash

```
U-Boot > nand erase 0x20000 0xa0000

NAND erase: device 0 offset 0x20000, size 0xc0000
Erasing at 0xc0000 -- 100% complete.
OK
U-Boot > nandecc hw
HW ECC selected
U-Boot > tftp 0xc0700000 u-boot-nand-ais.bin;nand write.i 0xc0700000 0x20000
${filesize}
Using    device
TFTP from server 192.192.192.154; our IP address is 192.192.192.215
Filename 'u-boot-nand-ais.bin'.
Load address: 0xc0700000
Loading: #################
done
Bytes transferred = 210860 (337ac hex)
NAND write: device 0 offset 0x20000, size 0x337ac
 210944 bytes written: OK
```

d) Write U-BOOT to SPI Flash

```
U-Boot > sf probe 0
1024 KiB SST25VF080B at 0:0 is now current device
U-Boot > sf erase 0 100000
U-Boot > tftp 0xc0700000 u-boot-spi-ais.bin; sf write c0700000 0 40000
Using    device
TFTP from server 192.192.192.154; our IP address is 192.192.192.215
Filename 'u-boot-spi-ais.bin'.
Load address: 0xc0700000
```

```
Loading: ################
done
Bytes transferred = 210860 (337ac hex)
U-Boot >
```

e) Write kernel

```
U-Boot > nand erase 0x200000 0x280000
NAND erase: device 0 offset 0x200000, size 0x280000
Erasing at 0x460000 -- 100% complete.
OK
U-Boot > nandecc sw
SW ECC selected
U-Boot > tftp 0xc0700000 uImage;nand write.i 0xc0700000 0x200000 ${filesize}
Using    device
TFTP from server 192.192.192.154; our IP address is 192.192.192.215
Filename 'uImage'.
Load address: 0xc0700000
Loading:
#################################################################


#################################################################
                ############################
done
Bytes transferred = 2299460 (231644 hex)
NAND write: device 0 offset 0x200000, size 0x231644
 2299904 bytes written: OK
```

f) Write file system

```
U-Boot > nand erase 0x600000 0x79c0000
U-Boot > nandecc sw
SW ECC selected
U-Boot > tftp 0xc2000000 rootfs.img;nand write.i 0xc2000000 0x600000
```

```
${filesize}
    Using    device
    TFTP from server 192.192.192.154; our IP address is 192.192.192.215
    Filename 'rootfs.img'.
    Load address: 0xc2000000
    Loading:
#################################################################
             ######T
#################################################################

#################################################################
             ############################################################T  #T
##########
             #####
    done
    Bytes transferred = 3889116 (3b57dc hex)


    NAND write: device 0 offset 0x600000, size 0x3b57dc
     3889152 bytes written: OK
```

You can use one command to complete all operation which contains TFTP download and image flashing

U-Boot > **run updatesys**

At this time, flickering of LED lamp on the board indicates that update has been finished; you just need to reboot it.

3) Boot-up

a) NAND Flash boot-up

Make sure the toggle switch [S3] as below:

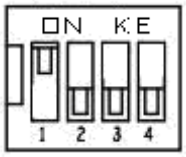


Figure 3-9 Boot-up from NAND Flash

b) SPI Flash boot-up

Make sure the toggle switch [S3] as below:

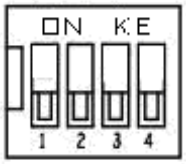


Figure 3-10 Boot-up from SPI Flash

3.8 Instructions

3.8.1 Choose the Display Device

System supports a wide range of display mode; the default display mode was the 4.3”LCD, and the user can change the display mode by change the U-Boot configure parameters.

For 4.3-inch LCD

U-Boot > **setenv bootargs 'console=ttyS2,115200n8 ubi.mtd=4 root=ubi0:rootfs rootfstype=ubifs vpif_capture.ch0_bufsize=831488 vpif_display.ch2_bufsize=831488 sbc8018lcd=43'**

U-Boot > **saveenv**

Saving Environment to NAND...

Erasing Nand...

Erasing at 0x0 -- 100% complete.

Writing to Nand... done

For 7-inch LCD

U-Boot > **setenv bootargs 'console=ttyS2,115200n8 ubi.mtd=4 root=ubi0:rootfs rootfstype=ubifs vpif_capture.ch0_bufsize=831488 vpif_display.ch2_bufsize=831488 sbc8018lcd=70'**

U-Boot > **saveenv**

```
Saving Environment to NAND...
Erasing Nand...
Erasing at 0x0 -- 100% complete.
Writing to Nand... done
```

3.8.2 Various Tests scenario

3.8.2.1 Buzzer/LED Testing

On the board, D24 is power LED lamp, D21 is heartbeat led lamp; D21 and D22 is user' led lamp.

The following operation carried out in HyperTerminal:

1) Control D21

```
root@SBC8118:~# echo 0 > /sys/class/leds/led1/brightness
root@SBC8118:~# echo 1 > /sys/class/leds/led1/brightness
```

2) Control D22

```
root@SBC8118:~# echo 0 > /sys/class/leds/led2/brightness
root@SBC8118:~# echo 1 > /sys/class/leds/led2/brightness
```

The user pushes a LED with operation are to kill bright.

3) Control Buzzer

```
root@SBC8118:~# echo 0 > /sys/class/leds/buzzer/brightness
root@SBC8118:~# echo 1 > /sys/class/leds/buzzer/brightness
```

The user pushes a Buzzer with operation are to kill tweet.

3.8.1.2 Button Testing

The SBC8118 single board computer is having one user key switch KEY1; users can perform the following testing:

First, enter the following command, and then press KEY1 key.

```
root@SBC8118:~# hexdump /dev/input/event0
0000000 06f5 49e5 206f 0001 0001 0100 0000 0000
0000010 06f5 49e5 208d 0001 0000 0000 0000 0000
0000020 06f5 49e5 1c89 0003 0001 0100 0001 0000
```

```
0000030 06f5 49e5 1ca9 0003 0000 0000 0000 0000
```

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

3.8.2.3 Touch Screen Testing

1) Run the command to test the touch screen.

```
root@SBC8118:~# 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@SBC8118:~# ts_test
```

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

3.8.2.4 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@SBC8118: # date 032220002012
Thu Mar 22 20:00:00 UTC 2012
```

2) Write the system clock into RTC

```
root@SBC8118:~# hwclock -w
```

3) Read the RTC

```
root@SBC8118: # 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@SBC8118: # hwclock -s
root@SBC8118: # date
Thu Mar 22 20:01:30 2012  0.000000 seconds
```

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

The SBC8118 Development board RTC battery can use model ML2032, user needs to prepare it themselves.

3.8.2.5 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@SBC8118:~#  cd /media/
root@SBC8118:/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@SBC80root@SBC811818:/media#  ls mmcblk0p1/
flash-uboot.bin          u-boot.bin              x-load.bin.ift_for_NAND
mlo                      uImage
ramdisk.gz               ubi.img
```

3.8.2.6 Network Testing

1) The board has a 10/100M self-adapting network card; users can connect the board to the LAN and enter the following commands for a test:

```
root@SBC8118:~# ifconfig eth0 192.192.192.203
root@SBC8118:~# # ifconfig
eth0      Link encap:Ethernet  HWaddr 00:11:22:33:44:55
          inet addr:192.192.192.203  Bcast:192.192.192.255
Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:185 Base address:0x2000
```

```
root@SBC8118:~# ping 192.192.192.154
PING 192.192.192.154 (192.192.192.154): 56 data bytes
64 bytes from 192.192.192.154: seq=0 ttl=128 time=4.486 ms
64 bytes from 192.192.192.154: seq=1 ttl=128 time=0.336 ms
64 bytes from 192.192.192.154: seq=2 ttl=128 time=0.336 ms
```

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

The IP address in the network card of development board and PC should be in the same network segment, for example: 192.192.192.x. Press CONTROL+C to quit the test.

3.8.2.7 SUART Testing

The SBC8118 single board computer is having three SUART ports, the relations between port and device name as below:

Port	Device name
PRU_COM1（RS232 connector）	/dev/ttySU0
PRU_COM2（RS232 connector）	/dev/ttySU1
PRU_COM3 (2.0mm space 5pin connector)	/dev/ttySU2

After ping success of their own PC in the Network testing, do not pull out the cable, and start the SUART functional testing, the following test method of PRU_COM1 interface as an example

1) Open the SBC8118 telnet

```
root@SBC8118:~# telnetd
```

2) Use a serial cable to connect SBC8118 PRU_COM1 and PC COM1, open PC Windows Hyper terminal software and set the following:

- Baud rate: 57600
- Flow control: no

And then, press “ok” to connect.

3) Run cmd.exe, execute the “telnet 192.192.192.203 command (change IP if required) to logon SBC8118.

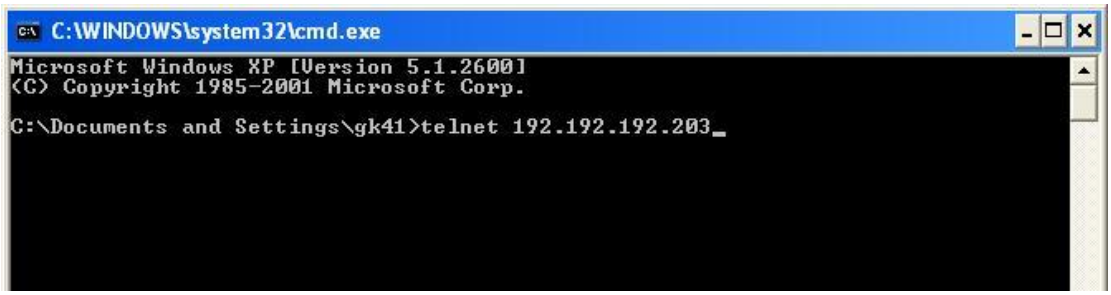


Figure 3-11

4) After a successful login, execute the following command to send the message “I am ttySU0” to the Hyper Terminal

```
root@sbc8018:~# stty -F /dev/ttySU0 57600
root@sbc8018:~# echo I am ttySU0 > /dev/ttySU0
```

5) Execute the following command to receive the message from Hyper Terminal

```
root@sbc8018:~# cat /dev/ttySU0
```

6) The final results as shown in below (the test “OK, I know” is inputted in Hyper terminal)

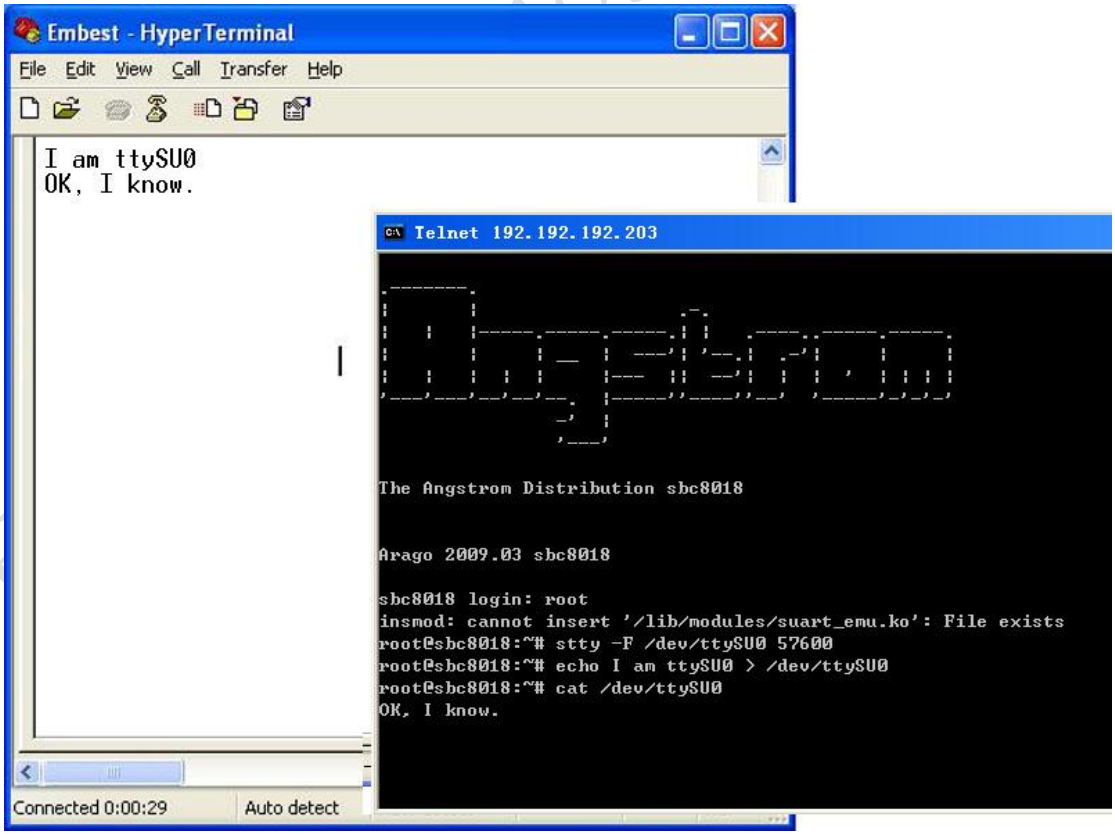


Figure 3-12

.3.8.2.8 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@SBC8118:~#  cd /media/
root@SBC8118:/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@SBC8118:/media#  ls sda1/
flash-uboot.bin              u-boot.bin                  x-load.bin.ift_for_NAND
mlo                          uImage
ramdisk.gz                   ubi.img
```

3.8.2.9 RS485 Testing

SBC8118 can be used as a RS485 device. Principle of connection as shown below, and refer to the schematic to find the corresponding pin, use the cable to connect SBC8118 RS485 interface and another RS485 device.

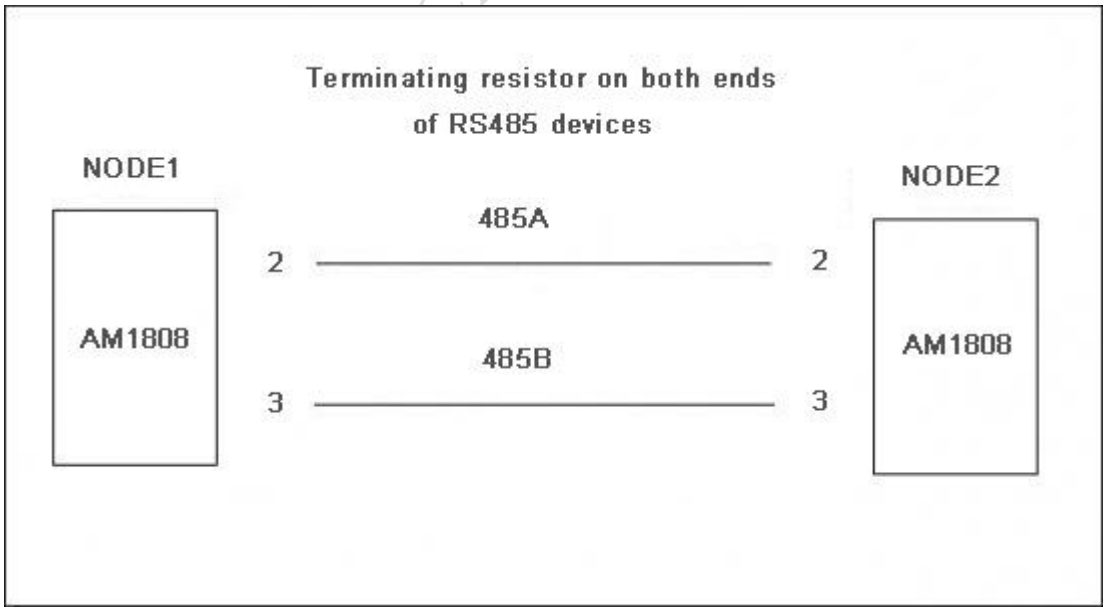


Figure 3-13

RS485 only supports half-duplex communication; the communication at one end at the same time can only send or only receive information. Put one of the SBC8118 or another

RS485 device as the sender and the other as receiving end.

The sender executes the following command:

```
root@sbc8018:~# 485_test -b 115200 -d /dev/ttyS1
*********************************************
            485 TEST
*********************************************



Select 1 : Send a message
Select 2 : Receive messages
>1
        Please enter the information to be sent off!
I am 485!
message = I am 485!
len = 11
Information is sent.....
Select 3 : Stop Send
>
```

The receiver executes the following command:

```
root@sbc8018:~# 485_test -b 115200 -d /dev/ttyS1
*********************************************
            485 TEST
*********************************************



Select 1 : Send a message
Select 2 : Receive messages
>2
Select 3 : Stop Receive
>
```

```
I am 485!
I am 485!
I am 485!
```

If the receiver receives the sender's message, indicating that the communication is normal. Press **Ctrl+C** to stop communicating.

Exchange SBC8118 with another RS485 device sending and receiving roles, and then test again.

3.8.2.10 Camera Testing

If you have bought the specific camera module（CAM8000-A） of SBC8118, after connecting CAMERA module and CCD camera, connect LCD screen, power on the board. carry out the testing by executing the following commands:

```
root@SBC8118:~# camera_test -i 0 -n 3 -f nv16 -S -s 720x576 /dev/video0
The framebuffer device was opened successfully.
fscreen line_length : 960
vscreen 480x272, 16bpp, xoffset=0, yoffset=0
The framebuffer device mapped successfully.
Device /dev/video0 opened: DA850/OMAP-L138 Video Capture.
```

At this time, LCD display screen will display images collected by the CCD camera. The camera driver in [drivers/media/video/tvp514x.c]

3.8.2.11 SATA Testing

1) Connect SATA device to the SATA interface on the single board computer, and the SATA device power cable connect to interface J23 on the single board computer.
2) Power on the single board computer, after entering the system, contents in the SATA equipment will be seen after the following instruction is input:

```
root@SBC8118:~# ls -l /media/
drwxr-xr-x    2 root     root          1024 Apr 15  2009 mmc1
drwxr-xr-x    2 root     root          4096 Jan  1  1970 mmcblk0p1
drwxr-xr-x    2 root     root          4096 Jan  1  1970 sda5
root@SBC8118:~# ls -l /media/sda5/
```

```
-rwxr-xr-x    1 root     root             6 Apr 14 20:25 hello.txt
-rwxr-xr-x    1 root     root      30781440 Jul 28  2011 taobeini.WAV
```

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.

1) To Edit code

led_app.c source code: control LED lamps D21(on the mother board) and D22(on the core board) to flicker in a way of accumulator.

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#define LED_1 "/sys/class/leds/led1/brightness"
#define LED_2 "/sys/class/leds/led2/brightness"

int main(int argc, char *argv[])
{
    int i;
    int f_led1, f_led2;
    unsigned char dat1, dat2;

    if((f_led1 = open(LED_1, O_RDWR)) < 0){
        printf("error in open %s", LED_1);
        return -1;
    }
    if((f_led2 = open(LED_2, O_RDWR)) < 0){
        printf("error in open %s",LED_2);
        return -1;
    }
```

```
        for(i = 0; i < 100; i++) {
            dat1 = (i & 0x1)? '1': '0';
            dat2 = (i & 0x2)? '1': '0';
            write(f_led1, &dat1, sizeof(dat1));
            write(f_led2, &dat2, sizeof(dat2));
            usleep(300000);
        }
        return 0;
    }
```

2) **To Cross-compile**

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

3) Download and run

Upload to the development board system through SD card, USB flash disk or network, enter the directory with the led_app file, input the following commands and press Enter to run led_app.

```
chmod a+x ./led_app
./led_app
```

Chapter 4 WinCE Operating System

4.1 Introduction

This section mainly introduces SBC8118 system and application development of Windows Embedded CE 6.0 R3, as well as software resources in disc, software features, installation 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\WINCE600\BSP\SBC8118.rar

CD\WINCE600\BSP\OMAPL13X_TI_V1.rar

Windows Embedded CE 6.0 R3 sample project

CD\WINCE600\sample project\ SBC8118

Pre-compile image

CD\WINCE600\Image\

EBOOTNANDFLASH.nb0	Eboot for NAND boot
EBOOTSPIFLASH.nb0	Eboot for SPI boot
NK.bin	WinCE runtime image (packed)
NK.nb0	WinCE runtime image

4.3 Software Features

Resources of BSP:

Catalog	Item	Source code / binary
EBOOT	NAND	Source
	NOR	source
	SPI	Source
OAL	KILT(EMAC)	Source
	Boot parameter	Source
	Watchdog	Source
	RTC	Source
	System timer	Source
	Interrupt controller	Source
	MMU	Source
	Serial Debug Port	Source
	Kernel Profiler-use timer0, high 32 bits	Source
	Library Abstrations (PSC, PLL, GPIO, abstrations)	Source
	Power management: CPU idle support (ARM wait for interrupt in OEMIdle()) This BSP release implements support for some basic power management (PM). The drivers support D0 and D4 states and system suspend / resume support is implemented. There is no support for voltage or frequency scaling.	Source
Driver	EDMA driver	Source

	I2C driver	Source
	SPI driver	Source
	MCASP driver	Source
	AIC3106 Audio driver	Source
	USB 1.1 OHCI HOST driver	Source
	USB OTG 2.0 HOST driver	Source
	USB OTG 2.0 FUNCTION driver	Source
	USB OTG driver	Source
	USB CDMA driver	Source
	Raster LCD Display driver	Source
	Character LCD Display driver	Source
	NDIS Ethernet driver	Source
	NAND FLASH driver	Source
	Serial driver	Source
	UPP driver	Source
	SD/MMC HOST controller driver	Source
	PWM driver	Source
	Notification LED driver	Source
	Touch Screen driver	Source
	McBSP driver	Source
	VPIF driver	Source
	PRU driver	Source
	SUART dirver	Source

4.4 System Development

4.4.1 Installation of IDE(Integrated Development Environment)

Please install items below to windows XP/Vista by step:

1) Visual Studio 2005
2) Visual Studio 2005 SP1
3) Visual Studio 2005 SP1 Update for Vista (vista system require)
4) Windows Embedded CE 6.0 Platform Builder
5) Windows Embedded CE 6.0 SP1
6) Windows Embedded CE 6.0 R2
7) Windows Embedded CE 6.0 Product Update Rollup 12/31/2008
8) Windows Embedded CE 6.0 R3
9) Windows Embedded CE 6.0 Product Update Rollup 12/31/2009
10) ActiveSync 4.5
11) Windows Mobile 6 Professional SDK

CD does not provide WinCE 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 [CDROM\WINCE600\bsp\SBC8118.rar] to [C:\WINCE600\PLATFORM] directory.
2) Extract [CDROM\WINCE600\bsp\OMAPL13X_TI_V1.rar] to [C:\WINCE600\PLATFORM\COMMON\SRC\SOC].
3) Copy CD directory [CDROM\WINCE600\sample project\SBC8118] to [C:\WINCE600\OSDesigns] directory.
4) Please modify the LCD module type before sysgen and build BSP:

For 4.3" LCD

Modify the line below in platform/SBC8118/SBC8118.bat

```
set BSP_LCD43INCH=1
set BSP_LCD7INCH=
```

For 7" LCD

Modify the line below in platform/SBC8118/SBC8118.bat

```
set BSP_LCD43INCH=
set BSP_LCD7INCH=1
```

The default installation path of the Windows Embedded CE 6.0 in this context is [C:\WINCE600].

4.4.3 Sysgen & Build BSP

1) Open the existing project file SBC8118.sln [C:\WINCE600\OSDesigns\SBC8118].
2) Click [Build-> Build Solution] in vs2005 to sysgen and build BSP.
3) After sysgen and build progress finish, EBOOTSPIFLASH.nb0, EBOOTNANDFLASH.nb0, NK.bin, NK.nb0 would be created in the directory of [C:\WINCE600\OSDesigns\SBC8118\SBC8118\RelDir\SBC8118_ARMV4I_Release], copy EBOOTSPIFLASH.nb0, EBOOTNANDFLASH.nb0 to directory [D:\SBC8118\bin], copy NK.bin, NK.nb0 to a TF card (FAT/FAT32 format).
4) Follow the section of update system to update kit.

4.4.4 Source code path of all drivers in BSP:

EDMA driver	bsp\SBC8118\SRC\DRIVERS\EDMA
I2C driver	bsp\SBC8118\SRC\DRIVERS\I2C bsp\OMAPL13X_TI_V1\I2C
SPI driver	bsp\OMAPL13X_TI_V1\SPI bsp\SBC8118\SRC\DRIVERS\SPI
McASP driver	bsp\OMAPL13X_TI_V1\MCASP
AIC3106 Audio driver	bsp\SBC8118\SRC\DRIVERS\WAVEDEV2
USB 1.1 OHCI Host driver	Bsp\SBC8118\SRC\DRIVERS\USB\OHCI
USB 2.0 OTG Host driver	bsp\OMAPL13X_TI_V1\USB\USBH bsp\SBC8118\SRC\DRIVERS\USB\USBH
USB 2.0 OTG Function Driver	bsp\OMAPL13X_TI_V1\USB\USBFN bsp\SBC8118\SRC\DRIVERS\USB\USBFN
USB 2.0 OTG driver	bsp\OMAPL13X_TI_V1\USB\USBOTG bsp\SBC8118\SRC\DRIVERS\USB\USBOTG
USB CDMA driver	bsp\OMAPL13X_TI_V1\USB\USBCDMA bsp\SBC8118\SRC\DRIVERS\USB\USBCDMA
Raster LCD Display driver	bsp\OMAPL13X_TI_V1\DISPLAY bsp\SBC8118\SRC\DRIVERS\DISPLAY

Character LCD Display driver	bsp\SBC8118\SRC\DRIVERS\CHARLCD bsp\OMAPL13X_TI_V1\LIDD
NDIS Ethernet driver	bsp\OMAPL13X_TI_V1\EMAC bsp\SBC8118\SRC\DRIVERS\EMAC
NAND Flash driver	bsp\SBC8118\SRC\COMMON\NAND bsp\SBC8118\SRC\DRIVERS\NAND
Serial driver	bsp\OMAPL13X_TI_V1\SERIAL bsp\SBC8118\SRC\DRIVERS\SERIAL
UPP driver	Bsp\OMAPL13X_TI_V1\UPP bsp\SBC8118\SRC\DRIVERS\UPP
SD/MMC Host Controller driver	Bsp\OMAPL13X_TI_V1\SDHC bsp\SBC8118\SRC\DRIVERS\SDHC bsp\OMAPL13X_TI_V1\SDBUS
PMW driver	bsp\OMAPL13X_TI_V1\PWM bsp\SBC8118\SRC\DRIVERS\PWM
Notification LED driver	bsp\SBC8118\SRC\DRIVERS\NLED
Touch Screen driver	bsp\SBC8118\SRC\DRIVERS\TOUCH
McBSP driver	bsp\OMAPL13X_TI_V1\MCBSP bsp\SBC8118\SRC\DRIVERS\MCBSP
VPIF drive	bsp\SBC8118\SRC\DRIVERS\CAMERA\LAYERED

PRU driver	bsp\OMAPL13X_TI_V1\PRU bsp\SBC8118\SRC\DRIVERS\PRU
SUART driver	bsp\OMAPL13X_TI_V1\SUART bsp\SBC8118\SRC\DRIVERS\SUART

If the user wants to refer to more WinCE driver development, please refer to the specific reference document of the Windows Embedded CE 6.0 PB 6.0:

Start->

All programs->

Microsoft visual Studio 2005->

MicroSoft Visual Studio Document->

Content(C)->

Windows Embedded CE 6.0->

Develop a Device Driver.

4.5 Update system image

SBC8118 system update is divided into two steps, one step is flash EBOOT to SPI Flash or NAND Flash, another step is flash NK (NK.bin or NK.nb0) to TF card or NAND Flash. this chapter for different system updated way of introduction .

4.5.1 Flashing EBOOT to SPI Flash

EBOOT can update to SPI Flash with sfh_OMAP-L138.exe (.net framework require, install dotNetFx40_Full_x86_x64.exe from Microsoft).

1) Make sure serial cable had established single board computer and PC.

2) Switch S3-3 and S3-4 to ON position, the other switch to OFF position.

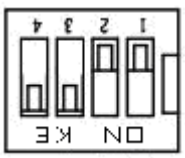


Figure 4-1

3) Copy the folder “bin” from the CD folder CD\wince6\tools to the folder D:\SBC8118.

4) According to your LCD size, copy the EBOOTSPIFLASH.nb0 from the CD folder WINCE600\Image\4_3INCH or WINCE600\image\7INCH to the folder D:\SBC8118\bin.

5) Click Start -> All Programs -> run, and input “CMD” on the pop-up dialog to enter Windows Command Prompt(cmd.exe), input the following commands as below:

```
d:
cd \SBC8118\bin
```

6) Run the flash tool to erase the SPI Flash: (change COM port if required)

```
sfh_OMAP-L138.exe -erase -targetType AM1808 -flashType SPI_MEM -p COM1
```

7) Power on the Kit. You should see progress being reported, wait until it completes, and then power off the kit.

Note: If the erase sequence does not complete after 30 seconds press a key to terminate the sfh_OMAP-L138.exe program and continue with the flashing procedure.

8) Run the flash tool to write an appropriate UBL(First Bootloader) and EBOOT to flash (change COM port if required).

```
sfh_OMAP-L138.exe -flash -targetType AM1808 -flashType SPI_MEM -v -p COM1 -appStartAddr 0xc7f60000 -appLoadAddr 0xc7f60000 ubl-spi-ais.bin EBOOTSPIFLASH.nb0
```

9) Power on the Kit. You should see progress being reported, wait until it completes.

```
Flashing UBL arm-nand-ais-456mhz.bin (13776 bytes) at 0x00000000

        Target: SENDIMG
        Target:   BEGIN
 100% [
                  Image data transmitted over UART.

        Target:    DONE
 100% [
                      UBL programming complete

        Target: SENDING
        Target:    DONE

Flashing application EBOOTNANDFLASH.nb0 (262144 bytes) at 0x00010000

        Target: SENDIMG
        Target:   BEGIN
 100% [
                  Image data transmitted over UART.

        Target:    DONE
 100% [
                   Application programming complete

        Target: Number of blocks needed for header and data: 0x0x00000003
        Target: Attempting to start in block number 0x0x00000006.
        Target: Magicnum: 0x0x55424CBB
        Target: Entrypoint: 0x0xC7F60000
        Target: Numpage: 0x0x00000080
        Target: Writing header and image data to Block 0x00000006, Page 0x000000
00
        Target:    DONE
        Target:    DONE

Operation completed successfully.
```

Figure 4-2

10) Power off the Kit and set DIP switches S3-1~S3-4 to OFF to select boot from SPI.

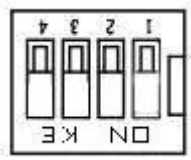


Figure 4-3

11) Start your serial terminal application (**115200 baud, 8N1**)

12) Power on the Kit and check that the new EBOOT image boot.

4.5.2 Flashing EBOOT to NAND Flash

EBOOT can update to NAND Flash with sfh_OMAP-L138.exe (.net framework require, install dotNetFx40_Full_x86_x64.exe from Microsoft).

1) Make sure serial cable had established single board computer and PC.

2) Switch S3-3 and S3-4 to ON position, the other switch to OFF position.

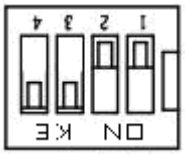


Figure 4-4

3) Copy the folder “bin” from the CD folder CD\WINCE600\tools to the folder D:\SBC8118.

4) According to your LCD size, copy the EBOOTNANDFLASH.nb0 from the CD folder WINCE600\Image\4_3INCH or WINCE600\image\7INCH to the folder D:\SBC8118\bin.

5) Click Start -> All Programs -> run, and input “CMD” on the pop-up dialog to enter Windows Command Prompt(cmd.exe), input the following commands as below:

```
d:
cd \SBC8118\bin
```

6) Run the flash tool to erase the NAND Flash: (change COM port if required)

```
sfh_OMAP-L138.exe -erase -targetType AM1808 -flashType NAND -p COM1
```

7) Power on the Kit. You should see progress being reported, wait until it completes, and then power off the kit.

Note: If the erase sequence does not complete after 30 seconds press a key to terminate the sfh_OMAP-L138.exe program and continue with the flashing procedure.

8) Run the flash tool to write an appropriate UBL(First Bootloader) and EBOOT to flash (change COM port if required).

```
sfh_OMAP-L138.exe -flash -targetType AM1808 -flashType NAND -v -p COM1 -appStartAddr 0xc7f60000 -appLoadAddr 0xc7f60000 ubl-nand-ais.bin EBOOTNANDFLASH.nb0
```

9) Power on the Kit. You should see progress being reported, wait until it completes.

```
Flashing UBL arm-nand-ais-456mhz.bin (13776 bytes) at 0x00000000

        Target: SENDIMG
        Target:   BEGIN
 100% [
                  Image data transmitted over UART.

        Target:    DONE
 100% [
                      UBL programming complete

        Target: SENDING
        Target:    DONE

Flashing application EBOOTNANDFLASH.nb0 (262144 bytes) at 0x00010000

        Target: SENDIMG
        Target:   BEGIN
 100% [
                  Image data transmitted over UART.

        Target:    DONE
 100% [
                   Application programming complete

        Target: Number of blocks needed for header and data: 0x0x00000003
        Target: Attempting to start in block number 0x0x00000006.
        Target: Magicnum: 0x0x55424CBB
        Target: Entrypoint: 0x0xC7F60000
        Target: Numpage: 0x0x00000080
        Target: Writing header and image data to Block 0x00000006, Page 0x000000
00
        Target:    DONE
        Target:    DONE

Operation completed successfully.
```

Figure 4-5

10) Power off the Kit and set DIP switches S3-1 to ON, all others to OFF.

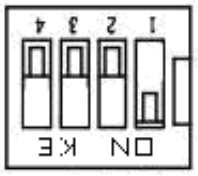


Figure 4-6

11) Start your serial terminal application (115200 baud, 8N1)

12) Power on the Kit and check that the new EBOOT image boot.

4.5.3 Update TF Card NK runtime images

1) Format TF card

Format the TF Card in FAT/FAT32 file system.

2) Copy NK runtime image

a) Navigate to the directory WINCE600/image/lcd7inch or WINCE600/image/lcd4.3inch, according to the LCD size.

b) Copy NK.nb0/NK.bin to TF card.

3) Change the EBOOT settings to boot NK from TF Card

Insert the TF card to kit, power on the kit and press space button to enter the EBOOT menu.

a) Press the key [2] -> [2] -> [2] by step to select boot NK from TF card

```
Booting with TI UBL
Device OPP (456MHz, 1.3V)01
Microsoft Windows CE Bootloader Common Library Version 1.4 Built Sep 23 2011 15:29:43
INFO:OALLogSetZones: dpCurSettings.ulZoneMask: 0xb
Microsoft Windows CE EBOOT 1.0 for AM1808 OMAPL138/AM18X EVM. Built Sep 23 2011 at 15:30:38
BSP version 1.3.0, SOC version 1.3.0
    CODE : 0xC7F60000 -> 0xC7FA0000
    DATA : 0xC7FA0000 -> 0xC7FE0000
   STACK : 0xC7FE0000 -> 0xC8000000
Enabled OAL Log Zones :  ERROR, WARN, INFO,
Platform Init done
System ready!
Preparing for download...
Predownload...
FMD: ReadID (Mfg=0x2c, Dev=0xda)
WARN: Invalid boot configuration found (using defaults)
Lan MAC: 00:08:ee:00:00:00
INFO: MAC address: 00:08:ee:00:00:00
WARN: Invalid BSP_ARGS data found (using defaults)
WARN: Unable to get hardware entropy
```

Hit space to enter configuration menu 2

Main Menu

---

[1] Show Current Settings

[2] Boot Settings

[3] Network Settings

[5] Video Settings

[6] Save Settings

[7] Peripheral Tests

[R] Reset Settings To Default Values

[0] Exit and Continue

Selection: **2**

Boot Settings

---

[1] Show Current Settings

[2] Select Boot Device

[3] Select Boot Delay

[4] Select Debug Device

[5] Force Clean Boot

[6] Write Download RAM NK to Flash

[7] Set Device ID String

[8] Allow DSP to Boot

[0] Exit and Continue

Selection: **2**

```
---------------------------------------------------------------------------
 Select Boot Device
---------------------------------------------------------------------------
 [1] EMAC
 [2] NK from SD
 [3] NK from NAND flash
 [0] Exit and Continue

 Selection (actual NK from SD): 2
 Boot device set to NK from SD
```

b) Press the key [0] -> [0] by step to start system from SD card, and you would see the following message in serial terminal:

```
---------------------------------------------------------------------------
 Boot Settings
---------------------------------------------------------------------------
 [1] Show Current Settings
 [2] Select Boot Device
 [3] Select Boot Delay
 [4] Select Debug Device
 [5] Force Clean Boot
 [6] Write Download RAM NK to Flash
 [7] Set Device ID String
 [8] Allow DSP to Boot
 [0] Exit and Continue

 Selection: 0

---------------------------------------------------------------------------
 Main Menu
```

```
--------------------------------------------------------------------------------

 [1] Show Current Settings

 [2] Boot Settings

 [3] Network Settings

 [5] Video Settings

 [6] Save Settings

 [7] Peripheral Tests

 [R] Reset Settings To Default Values

 [0] Exit and Continue


 Selection: 0
Device ID set to AM1808-0
BLFlashDownload: LogicalLoc   - 0x01C40000
Loading from SD card
+ReadNKFromSDMMC
ReadFileFromSDMMC: reading file 'nk.bin'
SDBootPDD: PDD_SDInitializeHardware: MMCSD
SDBootMDD: SDInitializeHardware: SD card detected
SDBootMDD: SDInitializeHardware: V2.0 card detected
SDBootMDD: SDInitializeHardware: timeOut = 0
SDBootMDD: SDInitializeHardware: timeOut = 1
SDBootMDD: SDInitializeHardware: timeOut = 2
SDBootMDD: SDInitializeHardware: timeOut = 3
SDBootMDD: Card address is 1234
SDBootMDD: 4-bit data bus selected
InitMasterBootRecord: Partition 0, type 12
InitMasterBootRecord: Partition 0, FAT32, start 0x7e00, length 0x753f8200
InitPartition: Offset 0x7e00, length 0x753f8200
ReadFileFromSDMMC: file size = 16138467 bytes
UnpackBINImage: unpacking binary from 0xc2000000
```

```
UnpackBINImage: Image start = 0x80000000
UnpackBINImage: Image length = 0x102fd2c
UnpackBINImage: record 0, start=0x80000000, length=0x4, checksum=0x1eb
…….
UnpackBINImage: record 296, start=0x0, length=0x80001000, checksum=0x0
CheckCEImage: checking image at 0xc0000000

ROMHDR (pTOC = 0xc102de3c) ----------
   DLL First                : 0x4001c001
   DLL Last                 : 0x40b5c097
   Physical First           : 0x80000000
   Physical Last            : 0x8102fd2c
   Num Modules              :          181
   RAM Start                : 0x81030000
   RAM Free                 : 0x8103f000
   RAM End                  : 0x8373f800
   Num Copy Entries         :            2
   Copy Entries Offset : 0x804f4fd4
   Prof Symbol Length    : 0x00000000
   Prof Symbol Offset    : 0x00000000
   Num Files                :           73
   Kernel Flags             : 0x00000000
   FileSys RAM Percent : 0x30303030
   Driver Glob Start     : 0x00000000
   Driver Glob Length    : 0x00000000
   CPU                      :       0x01c2
   MiscFlags                :       0x0002
   Extensions               : 0x80001070
   Tracking Mem Start    : 0x00000000
   Tracking Mem Length : 0x00000000
```

```
-----------------------------------------------


Image Start .......: 0x00000000
Image Size ........: 0x00000000
Image Launch Addr .: 0x00000000
Image ROMHDR ......: 0x00000000
Boot Device/Type ..: 3 / 6
ADEhellounch Windows Embedded CE by jumping to 0xc0000000...
Windows CE Kernel for ARM (Thumb Enabled) Built on Oct 20 2009 at 18:39:19
OEMInit: init.c built on Sep 28 2011 at 15:51:27.
BSP version 1.3.0, SOC version 1.3.0
INFO:OALLogSetZones: dpCurSettings.ulZoneMask: 0xf
WARN: Updating local copy of BSP_ARGS
Intr Init done...
Timer Init done...
+OALDumpClocks
Clock Configuration :
  Reference Clock 0 ..  24000000 Hz
    PLL0 ..... 456000000 Hz
    PLL0:SYSCLK1 ..... 456000000 Hz (DSP Subsystem)
    PLL0:SYSCLK2 ..... 228000000 Hz (UART,EDMA,SPI,MMC/SD,VPIF,LCDC,SATA,uPP,USB2.0,HPI,PRU)
    PLL0:SYSCLK3 .....  91200000 Hz (EMIFA)
    PLL0:SYSCLK4 ..... 114000000 Hz (INTC, SYSCFG, GPIO, PSC, I2C1, USB1.1, EMAC/MDIO, GPIO)
    PLL0:SYSCLK5 ..... 152000000 Hz (reserved)
    PLL0:SYSCLK6 ..... 456000000 Hz (ARM Subsystem)
    PLL0:SYSCLK7 .....  76000000 Hz (EMAC)
    PLL0:AUXCLK ......  24000000 Hz (I2C0, Timers, McASP0 serial clock, RTC, USB2.0 PHY)
```

```
        PLL1 ..... 264000000 Hz
        PLL1:SYSCLK1 ..... 264000000 Hz (DDR2/mDDR PHY)
        PLL1:SYSCLK2 ..... 132000000 Hz (Optional for:
McASP0,McBSP,ePWM,eCAP,SPI1)
        PLL1:SYSCLK3 ..... 88000000 Hz (PLL0 input)
    -OALDumpClocks
    -OEMInit
      PINMUX14=0x00000000
      PINMUX15=0x00000000
      PINMUX16=0x22222200
      PINMUX17=0x22222222
      PINMUX18=0x22000022
      PINMUX19=0x02000022
    OEMGetExtensionDRAM: Added 0x84400000 -> 0x88000000
    OEM: Cleaning system hive
    OEM: Cleaning user profiles
    WARN: Updating local copy of BSP_ARGS
    OEM: Not cleaning system hive
    FMD: ReadID (Mfg=0x2c, Dev=0xda)
    MICBIASHardwareContext::Init 555
    Adapter's MAC address is 00:08:EE:00:00:00
    SDHC +Init
            SDHC Active RegPath: Drivers\Active\21
                                    +SDHCPDD_Init: Ctrl 0,
Entry
    SDHC -Init
            SDHC +Open
                SDHC +Open
                    SDHC_CARD_DETECT = 1
    SDHC CommandCompleteHandler: Command response timeout
```

```
SDHC CommandCompleteHandler: Command response timeout
SDHC CommandCompleteHandler: Command response timeout
SDHC CommandCompleteHandler: Command response timeout
SDHC CommandCompleteHandler: Command response timeout
SDHC CommandCompleteHandler: Command response timeout
SDHC CommandCompleteHandler: Command response timeout
SDHC CommandCompleteHandler: Command response timeout
```

4.5.4 Flashing NK.bin to NAND flash

This section introduce how to flashing NK.bin to nand flash via ethernet and VS2005:

1) Confirm you have performed the release build in C:\WINCE600\OSDesigns\SBC8118\ SBC8118.sln.
2) Connect PC and SBC8118 with RJ45 Cable.
3) Switch the S3-1 to ON position, the other bits switch to OFF position:

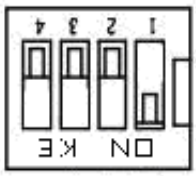


Figure 4-7 Boot up from NAND Flash

4) Select EMAC as boot media in eboot menu, press key [2]->[2]->[1] , steps as below:

```
------------------------------------------------------------------------
Select Boot Device
------------------------------------------------------------------------
 [1] EMAC
 [2] NK from SD
 [3] NK from NAND flash
 [0] Exit and Continue

 Selection (actual NK from SD): 1
```

```
Boot device set to EMAC
```

5) Press [6] -> [y] enable flashing NK to NAND Flash in EBOOT menu:

```
---------------------------------------------------------------------------
Boot Settings
---------------------------------------------------------------------------
[1] Show Current Settings
[2] Select Boot Device
[3] Select Boot Delay
[4] Select Debug Device
[5] Force Clean Boot
[6] Write Download RAM NK to Flash
[7] Set Device ID String
[8] Allow DSP to Boot
[0] Exit and Continue
Selection: 6
Enable Write Download RAM NK to Flash (actually disabled) [y/-]: y
Write Download RAM NK to Flash enabled
```

6) Press key [0] return to eboot main menu, press key [3] to set the network property. Setting the DHCP, IP, NETMASK according to your network environment, confirm that the IP, NETMASK of kit is in the same sub network of your PC.

```
---------------------------------------------------------------------------
Network Settings
---------------------------------------------------------------------------
[1] Show Current Settings
[2] Enable/disable KITL
[3] KITL interrupt/poll mode
[4] Enable/disable DHCP
[5] Set IP address
[6] Set IP mask
```

```
 [7] Set default router
 [8] Enable/disable VMINI
 [0] Exit and Continue
Selection: 1
Network:
  KITL state ..... disabled
  KITL mode ..... interrupt
  DHCP ..... disabled
  MAC address ..... 00:08:ee:00:00:00
  IP address ..... 192.192.192.200
  IP mask ..... 255.255.255.0
  IP router ..... 192.192.192.101
  VMINI ..... disabled
```

7) Press key [0] return to EBOOT main menu, then press key [0] prepare to download NK.bin from PC, and the serial terminal would display the message as below:

```
INFO: Boot device uses MAC 00:08:ee:ff:ff:ff
+EbootSendBootmeAndWaitForTftp
Sent BOOTME to 255.255.255.255
Sent BOOTME to 255.255.255.255
Sent BOOTME to 255.255.255.255
Sent BOOTME to 255.255.255.255
Sent BOOTME to 255.255.255.255
Sent BOOTME to 255.255.255.255
Sent BOOTME to 255.255.255.255
```

8) Click [Target->Connectivity Options] in VS2005 Menu, and then pop up the dialog below, Select Ethernet in Download Item select list.

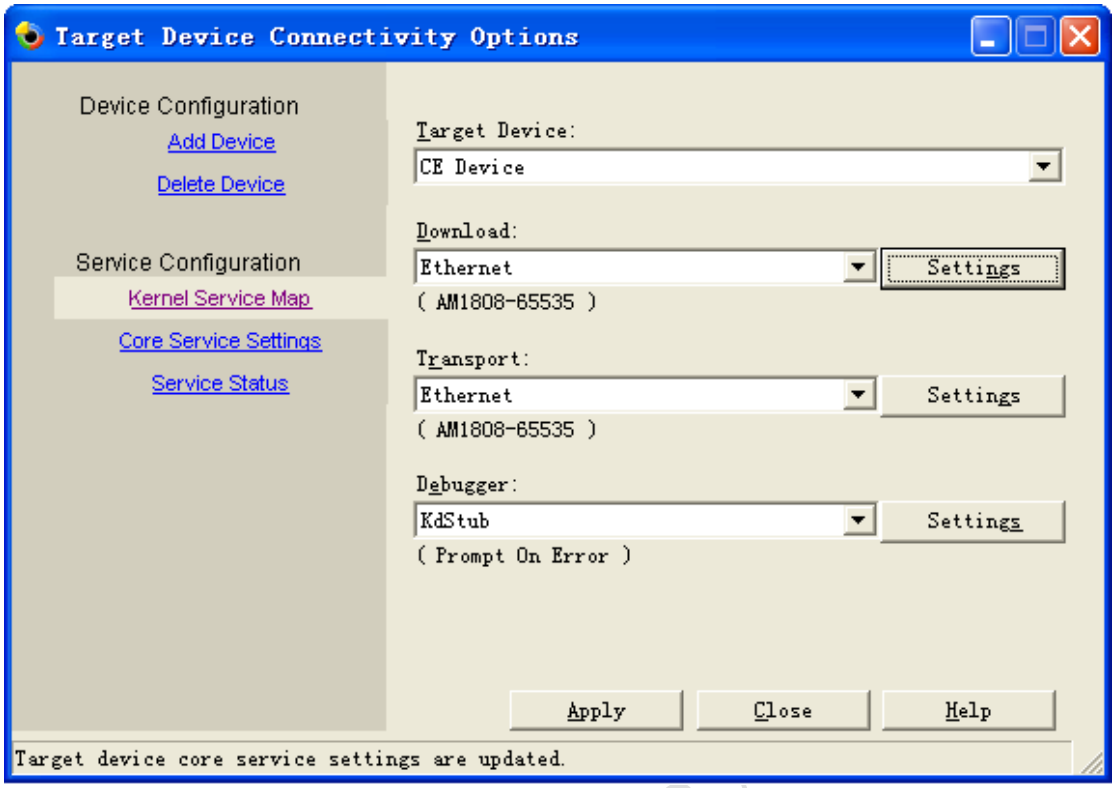


Figure 4-8

9) Click Settings button locate in the right of Download Item select list, then it would pop up a dialog as picture below, if the network properly setting and network cable is OK, you would see device AM1808-65535 in Active target devices edit box, select device AM1808-65535 and click [OK] button back to Connectivity Options setting dialog, click [Apply->Close->finish] Connectivity Options setting.

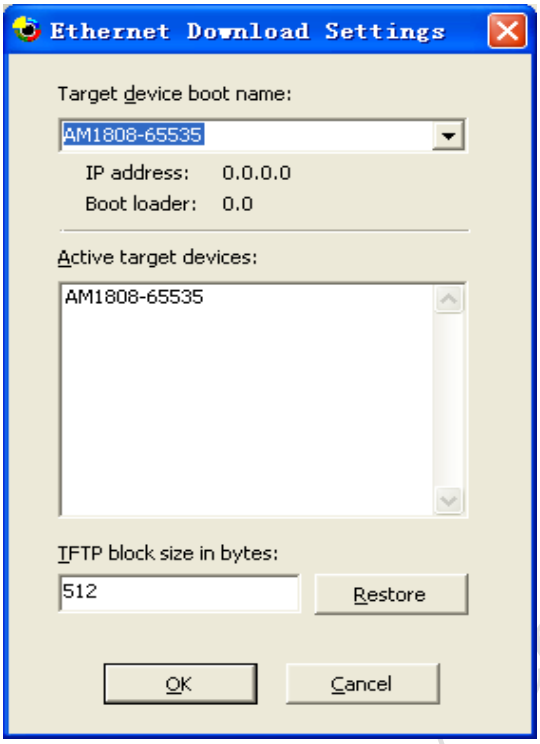


Figure 4-9

10) Click [Target] -> [Attach Device] in VS2005 menu Start download NK.bin, then VS2005 would pop up a dialog to indicate download progress, after download and NK.bin flashing finish the serial terminal would display the message below:

```
OEMWriteFlash: NK written
ROMHDR at Address 80000044h
Image Start .......: 0x80000000
Image Size ........: 0x00ff9a74
Image Launch Addr .: 0x80001000
Image ROMHDR ......: 0xc0ff7be0
Boot Device/Type ..: 2 / 6
Got EDBG_CMD_JUMPIMG
Got EDBG_CMD_CONFIG, flags:0x00000000
BLFlashDownload: LogicalLoc   - 0x62000000
Load NK image from flash memory (NAND)
FMD: ReadID (Mfg=0x2c, Dev=0xda)
BLFlashDownload: cp1
```

11) power down and power on the kit, press space button to eboot menu, press key

[2]->[2]->[3] by step to select boot NK from NAND flash, press [0] -> [0] to start system from NAND flash.

4.6 Instructions for use

4.6.1 How to use Power Management

1) This BSP release implements support for some basic power manages states and system suspend / resume support is implemented. There is no support for voltage or frequency scaling.
2) Power Management Configuration

   The sample PM configuration is enabled by default in platform.reg. The configuration can be disabled by setting the BSP_POWERMAN variable in:…\<BSP Folder>\SBC8118.bat.

Notes:

1) When the sample PM configuration is disabled the default CE 6.0 PM configuration is used.
2) In the suspend state, the CE Power Manager powers down all the peripherals (sets all drivers to D4 state).
3) In the suspend state, the SDRAM is put into self-refresh mode and powered down, the PLL controllers are powered down and the SOC is put into Deep Sleep mode.
4) On L138/AM18x there are two wake sources supported:
   • Internal source – RTC alarm
   • External source via the DEEPSLEEP pin – connected to UART2 CTS
5) The pmsuspend tool can be used to enter the system suspend state, and the wake source can be specified on the command line.
6) On L138 / AM18x when resuming the SDRAM and PLL controllers are powered up.
7) On resume the CE Power Manager powers up all the peripherals (sets all drivers to D0 state).

Power Management Tools

When power management is enabled, a number of tools are available:

- pmtimeout – Sets the idle timeouts used to transition between PM states
- pmsuspend – Sets an RTC alarm and puts the platform into suspend
- pmset – Sets the system power state
- pmget – Gets the current system power state
- pmsetd – Sets the power state for a device
- pmgetd – Gets the power state for a device
- pmreq – Sets a device power requirement
- pmmon – Monitors and reports on power state changes

Most of the tools display help if run with no command line parameters. Some example usage follows.

e.g. Go into suspend with a 1 min RTC alarm:

CE device->Start->CMD prompt->pmsuspend -t 1

4.6.2 How to use CAM8000-A module

1) Select Third Party->BSP->SBC8118: ARMV4I->Device Drivers->Camera->VPIF Capture Driver in catalog items view of VS2005.
2) Confirm that dshow items in the catalog items view have been select like the picture below:

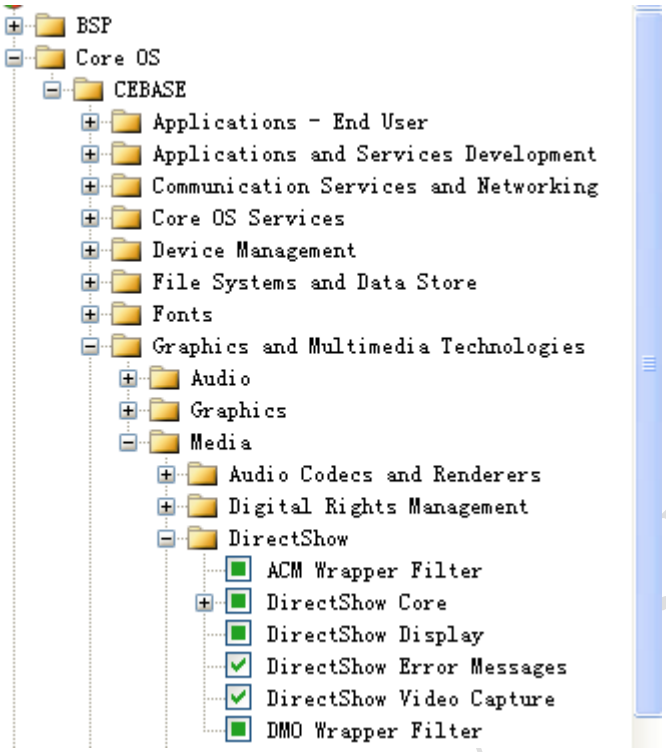


Figure 4-10

3) Click [Build-> Rebuild Solution] in VS2005 to perform sysgen and BSP build.

4) Update NK.bin and NK.nb0 after sysgen and build BSP finished.

5) Confirm CAM8000-A module is connect to SBC8118 correctly, boot system with updated NK, copy C:\WINCE600\PLATFORM\SBC8118\files\CameraDshowApp_analog.exe to wince OS run on the kit, click CameraDshowApp_analog.exe perform camera preview.

The VPIF driver support PAL(720*576) and NTSC (720*480) camera, and the drivers is default set to support the PAL camera, some parameter in VPIF drivers should be change with NTSC camera, if you want to view the whole of picture in LCD, please use 7inch LCD connect to SBC8118.

4.6.3 How to use RS485 Test

1) there is a RS485 Test Applicaton locate in CD\WINCE600\APP\Test485, this test application show how to send and receive data form RS485 bus.

2) copy RS485Test to target wince 6.0 OS, launch and type some text in text edit box and click send the text would transmit to RS485 Bus. The received text in RS485 bus would show in right text box.

NOTE: the RTS pin is use as TX/RX switch in RS485.

4.7 SBC8118 wince 6.0 win32 API application development Demo

SBC8118 wince 6.0 win32 API development demo please refer to bsp\SBC8118\SRC\TEST\APPS.

Appendix

Appendix I Hardware Dimensions

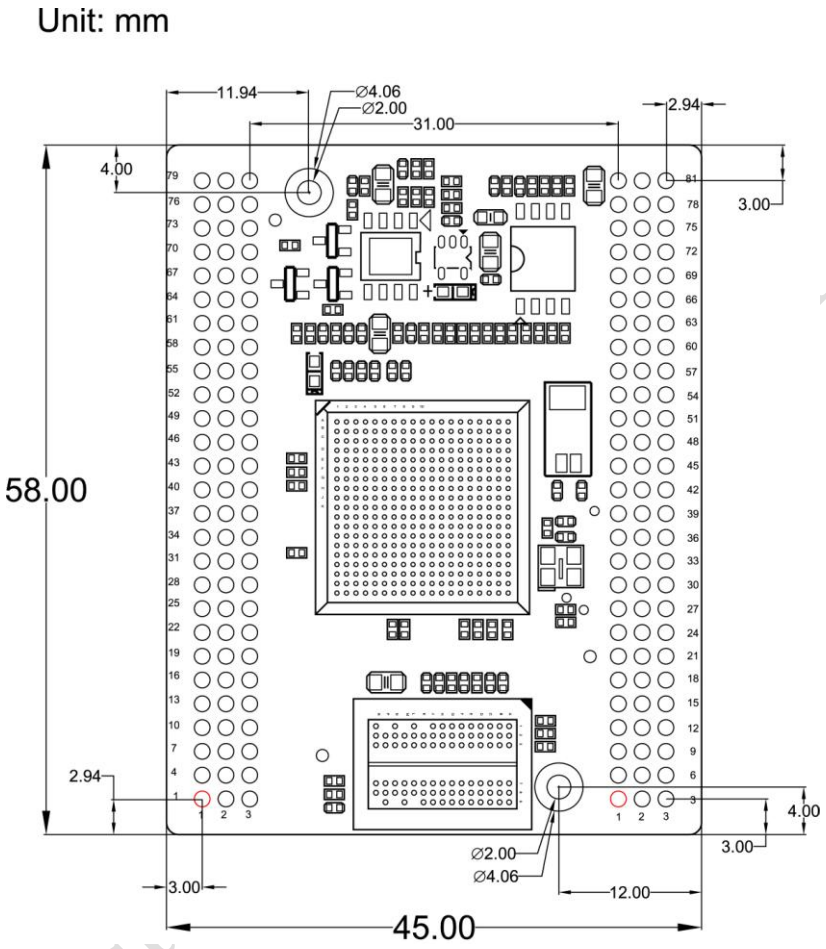


Figure Appedix 1-1 Mini8118 Hardware Dimensions Diagram

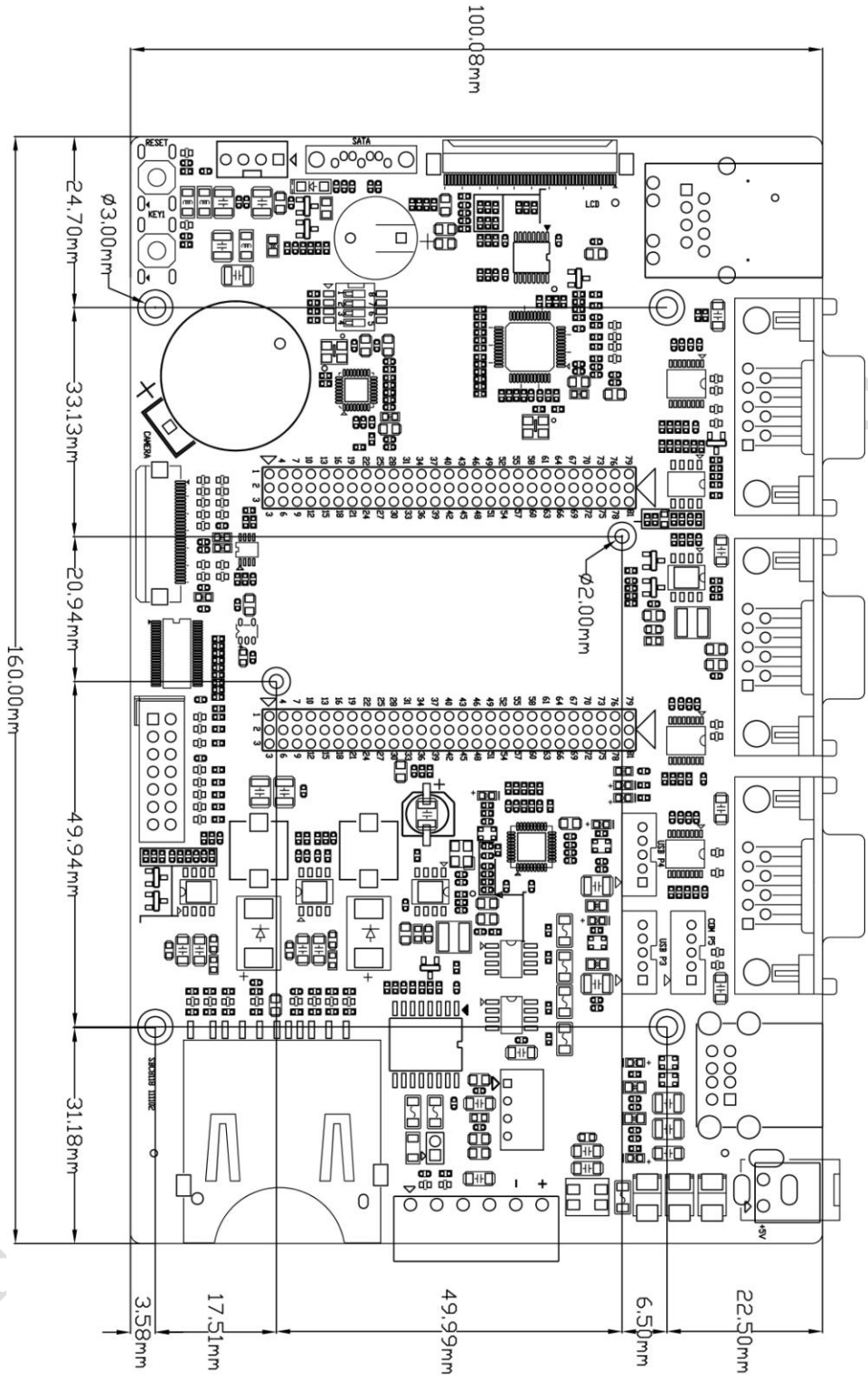


Figure Appedix 1-2 SBC8118 Hardware Dimensions Diagram

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

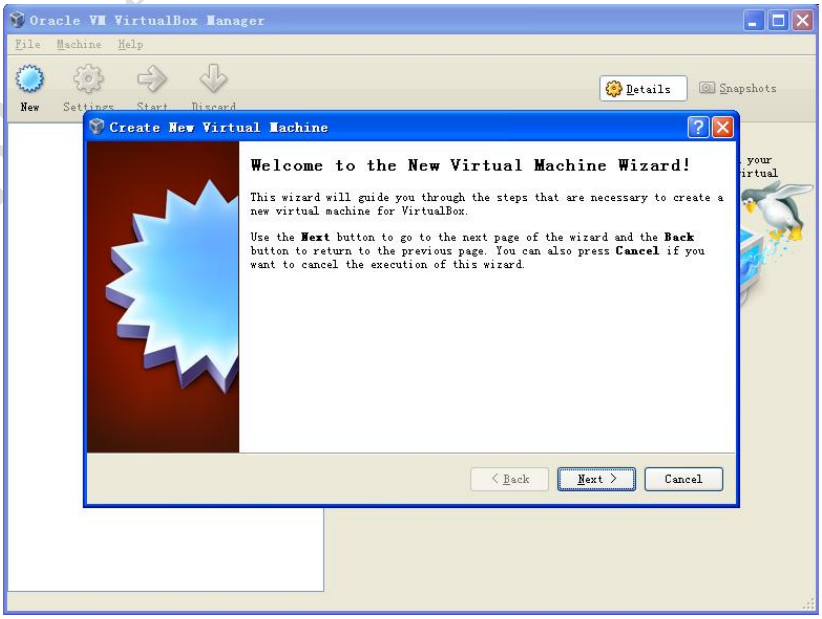


Figure Appedix 2-2-1

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**.

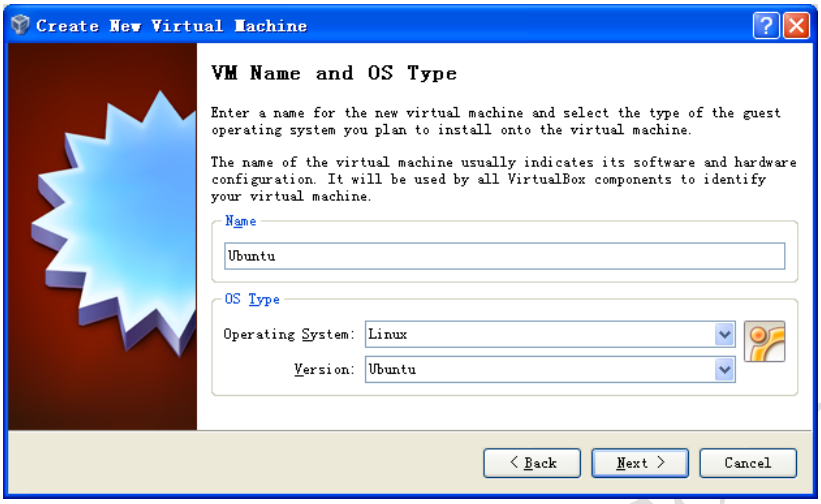


Figure Appedix 2-2-2

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**.

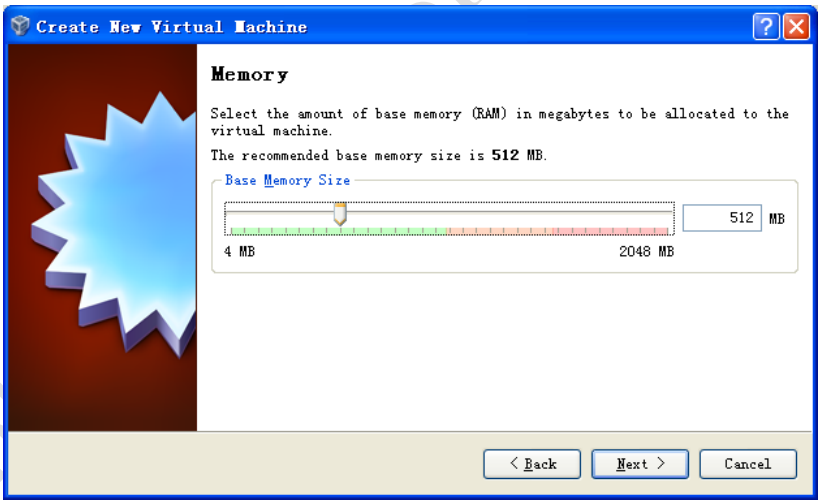


Figure Appedix 2-2-3

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 your 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**.

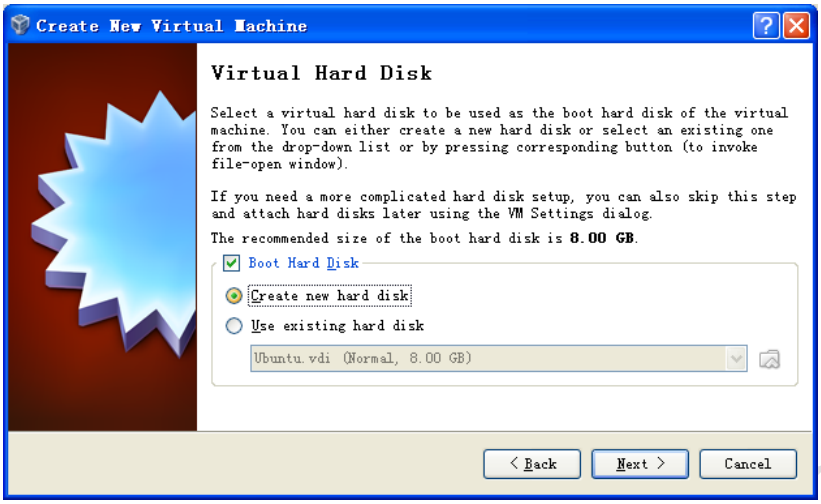


Figure Appedix 2-2-4

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**.

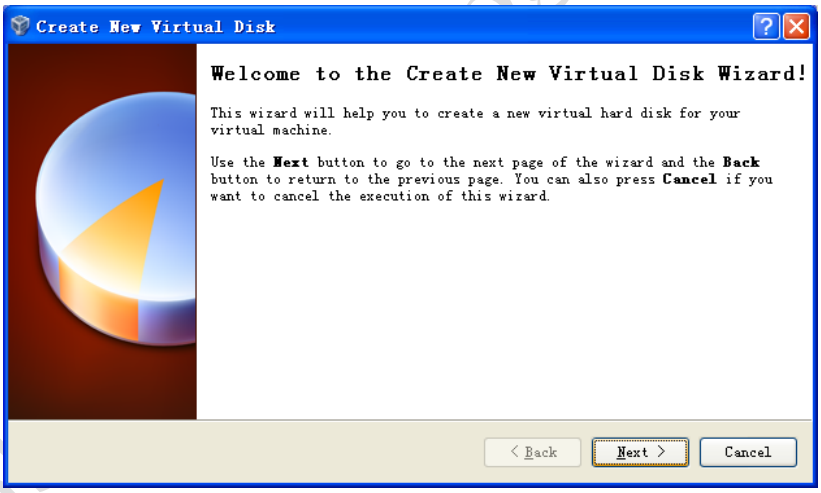


Figure Appedix 2-2-5

Click **Next** again.

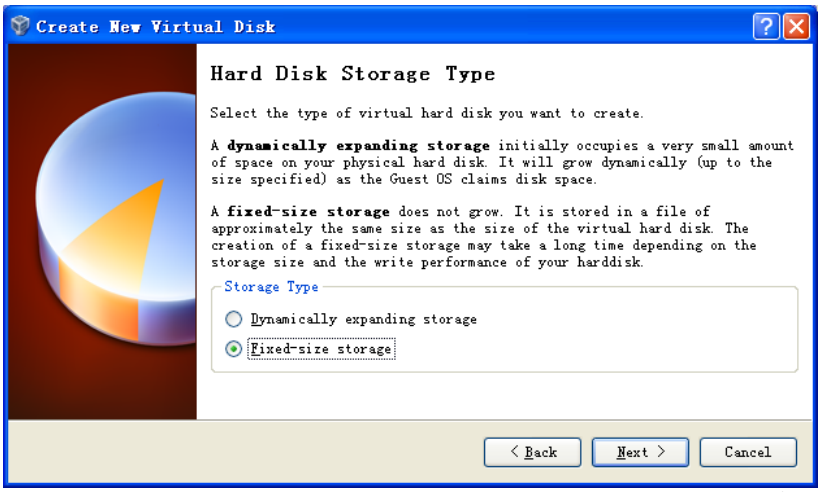


Figure Appedix 2-2-6

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 a **Fixed-size storage**.

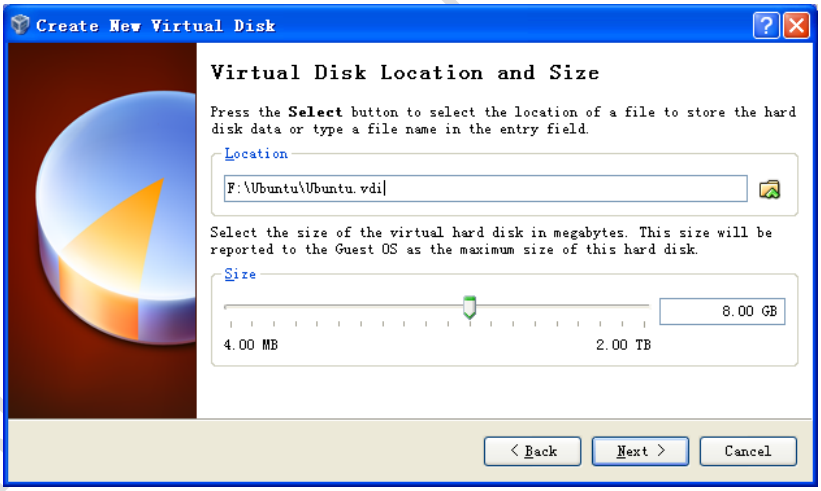


Figure Appdix 2-2-7

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.

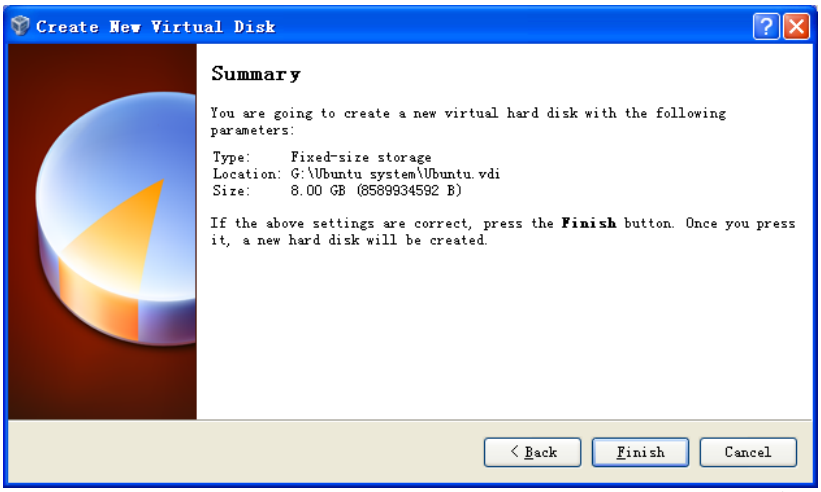


Figure Appedix 2-2-8

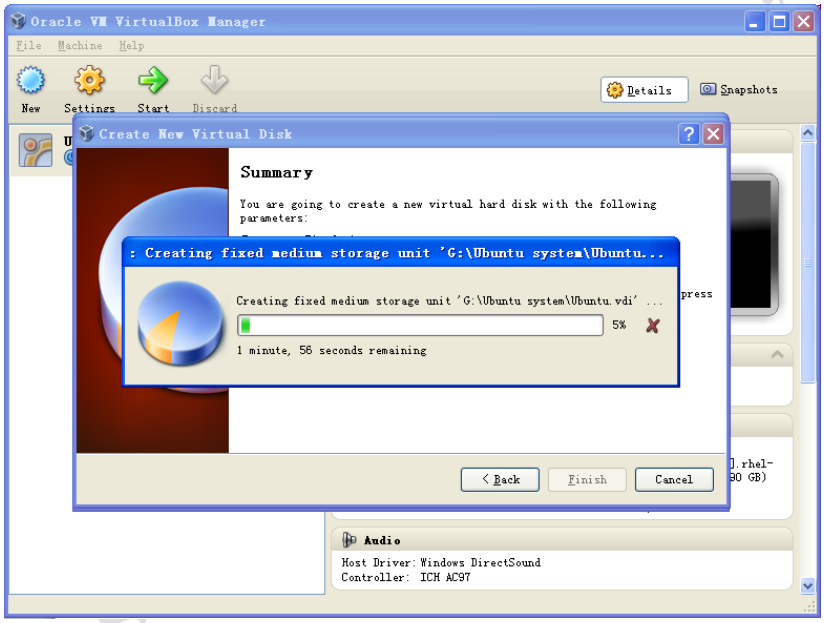


Figure Appedix 2-2-9

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.

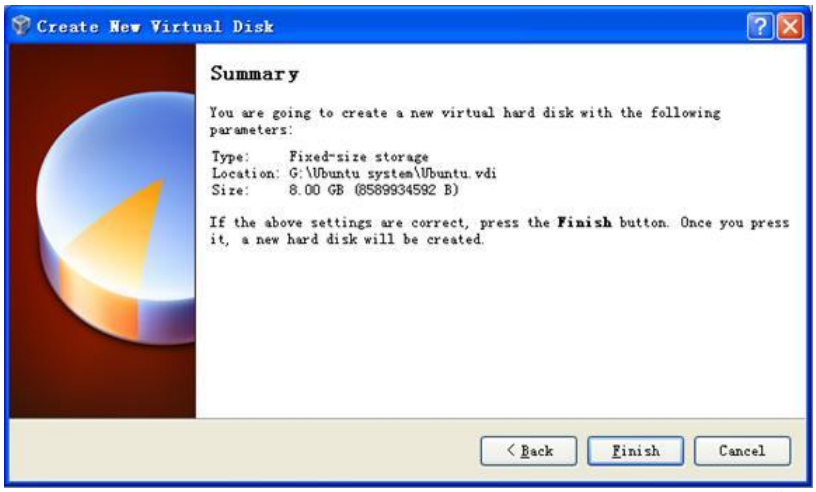


Figure Appedix 2-2-10

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

3. Installing Ubuntu

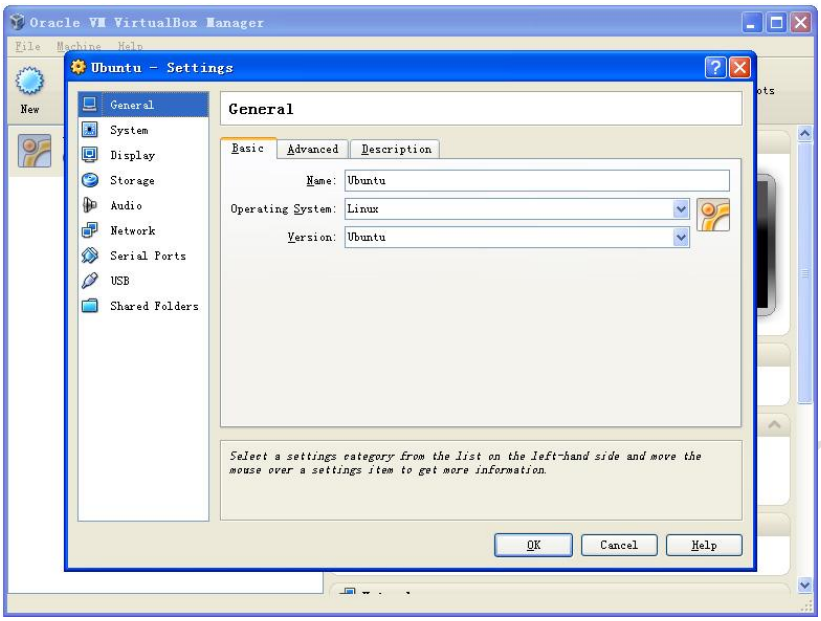


Figure Appedix 2-3-1

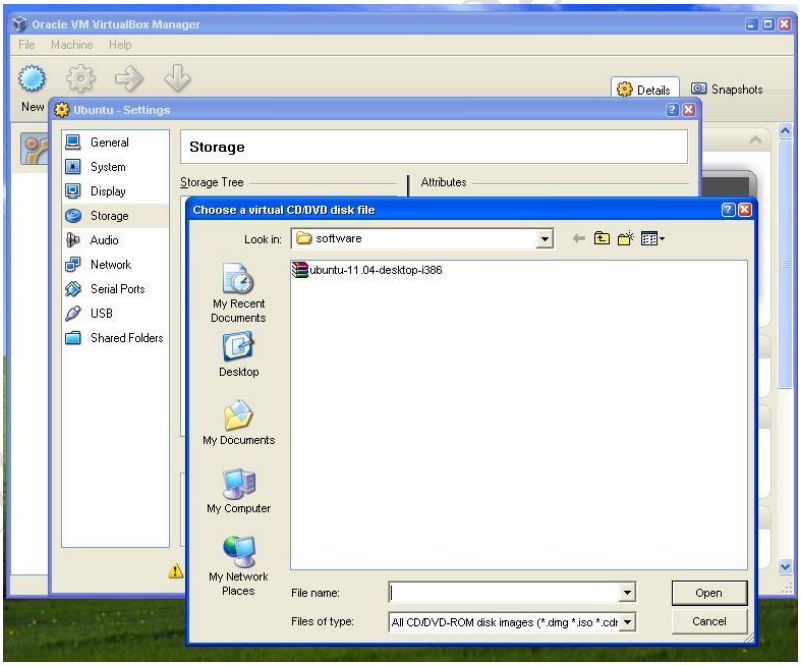


Figure Appedix 2-3-2

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.

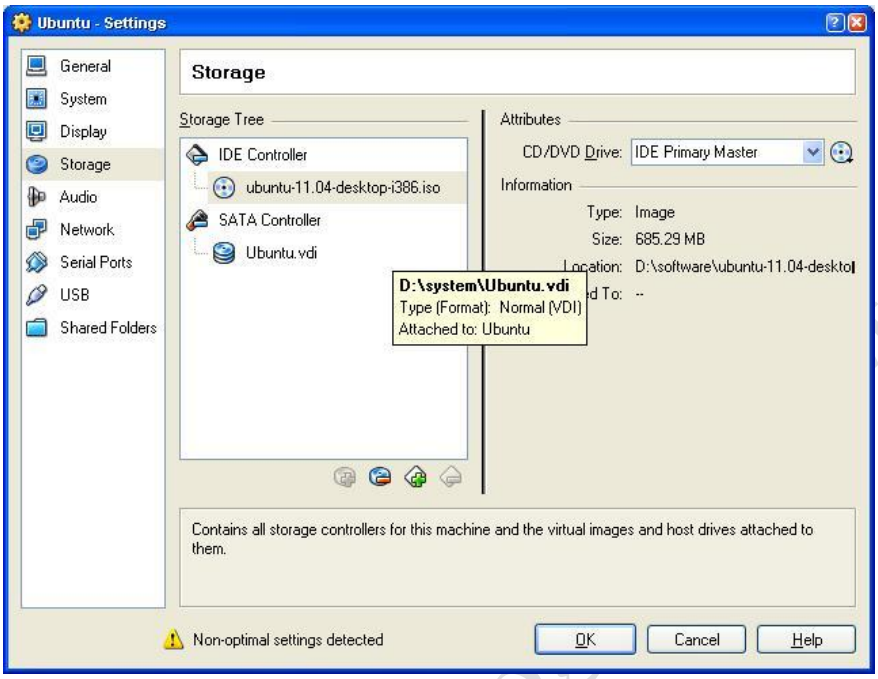


Figure Appedix 2-3-3

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

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

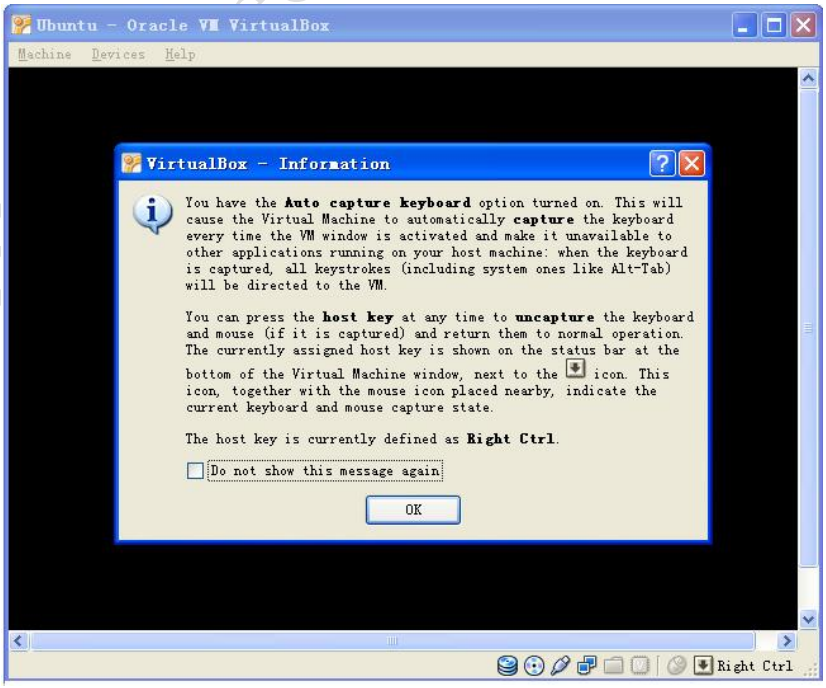


Figure Appedix 2-3-4

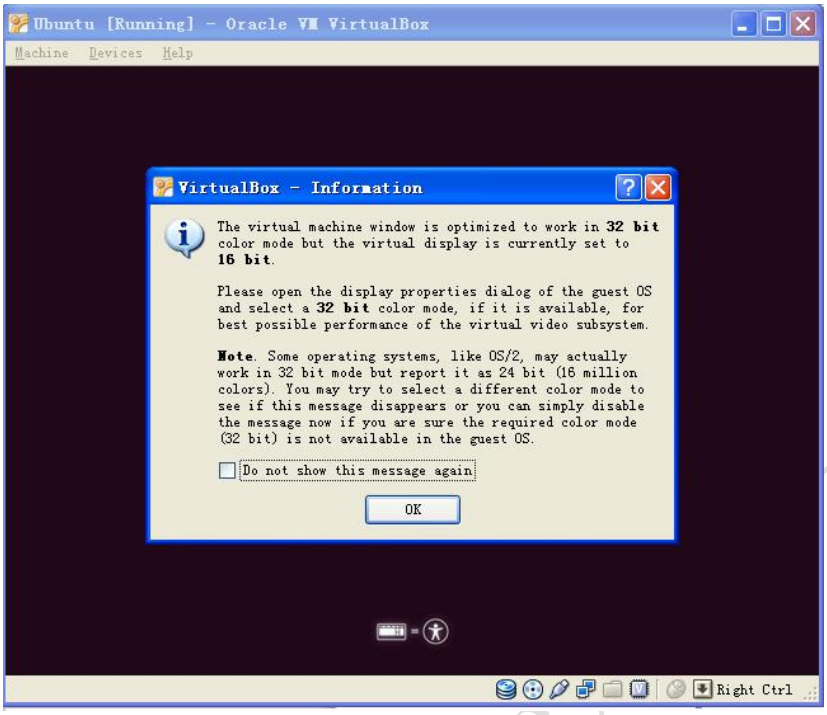


Figure Appedix 2-3-5

Click OK

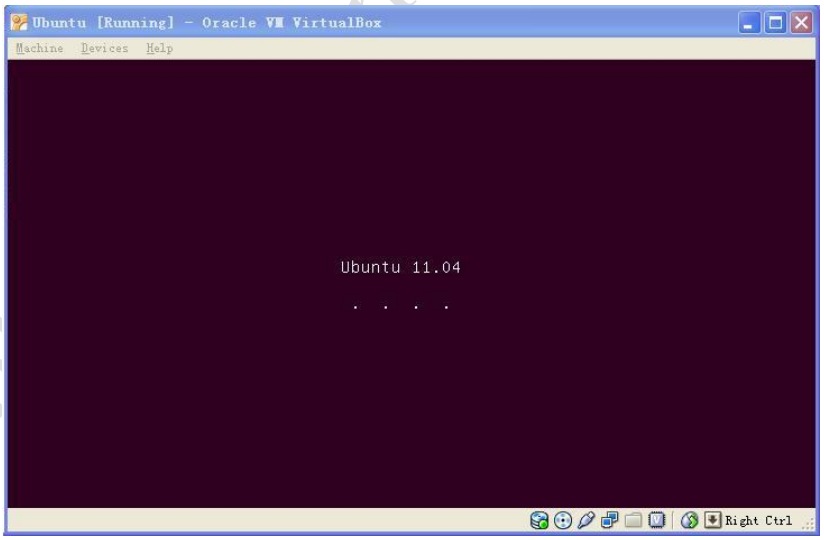


Figure Appedix 2-3-6

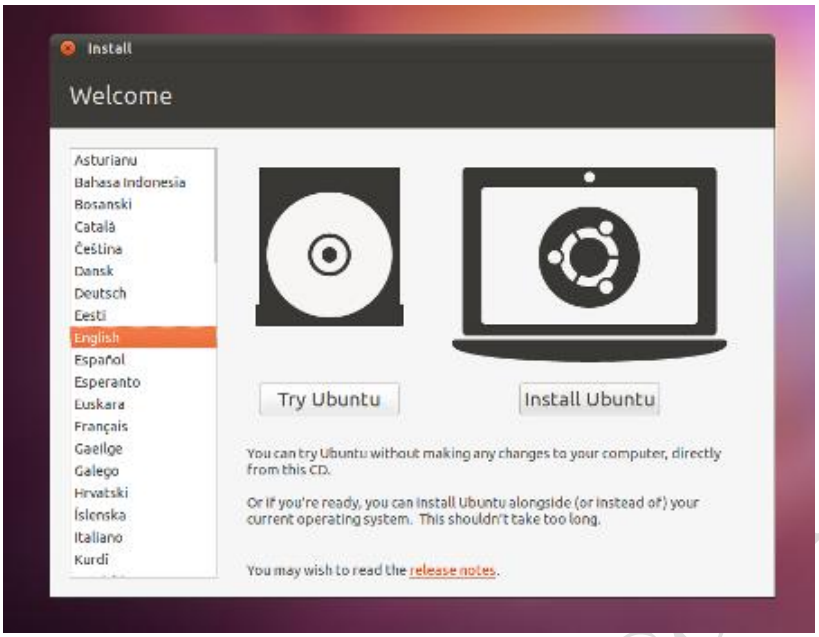


Figure Appedix 2-3-7

Select language and click **Install Ubuntu**.

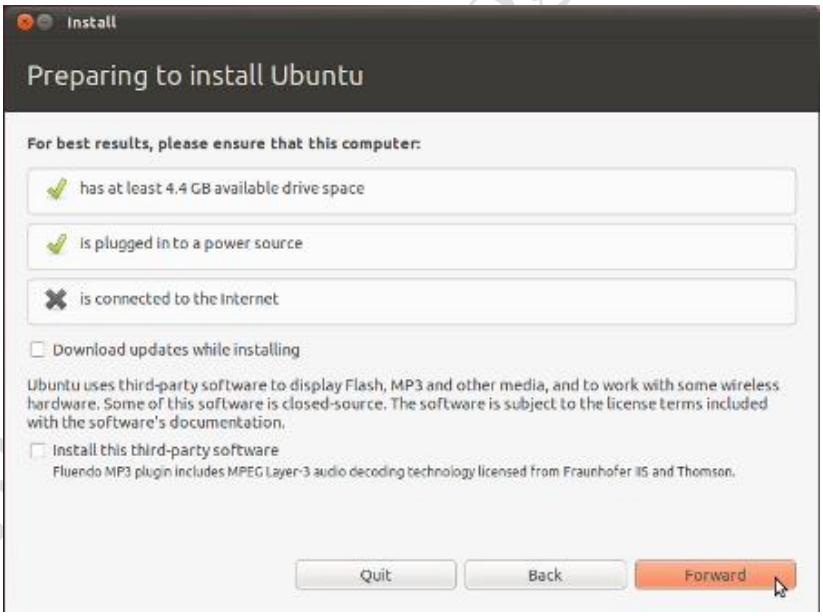


Figure Appedix 2-3-8

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.

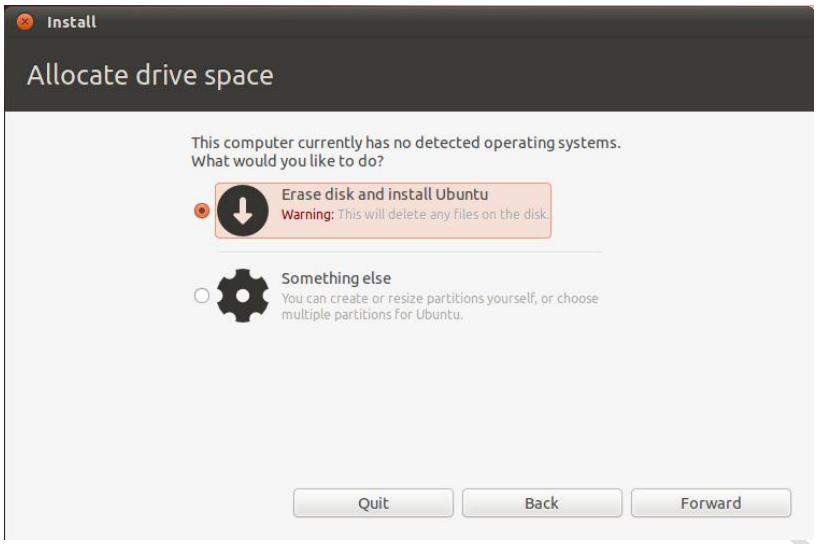


Figure Appedix 2-3-9

Click **Forward**.

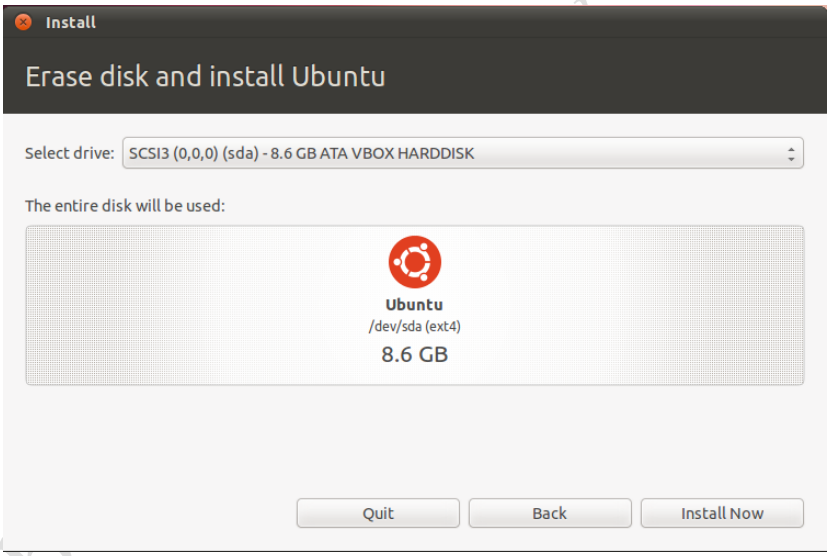


Figure Appedix 2-3-10

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 Appedix 2-3-11

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

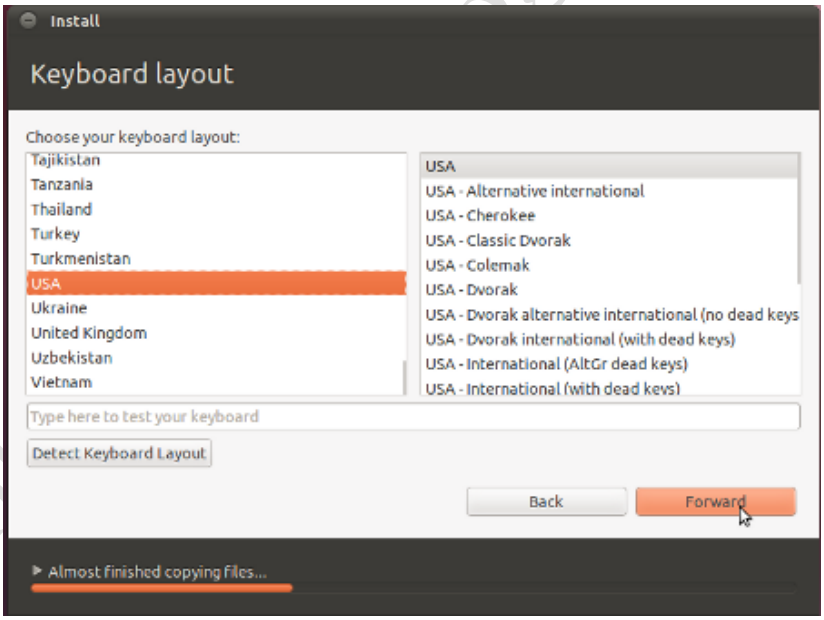


Figure Appedix 2-3-12

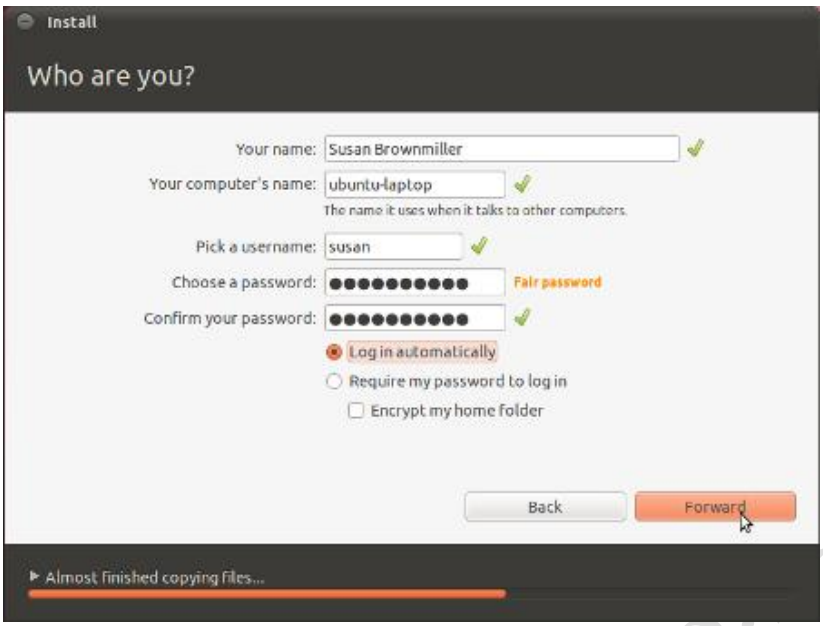


Figure Appedix 2-3-13

Figure Appedix 2-3-14

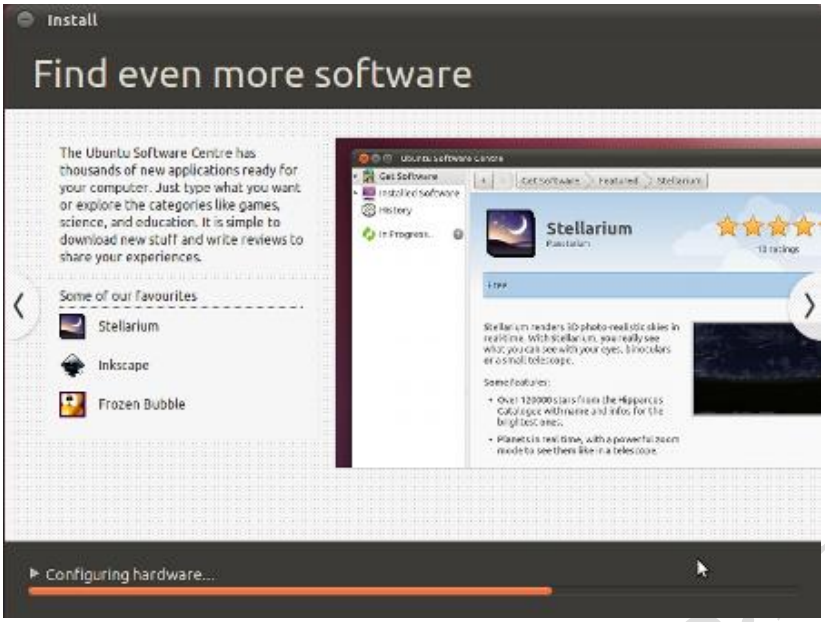


Figure Appedix 2-3-15

Figure Appedix 2-3-16

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

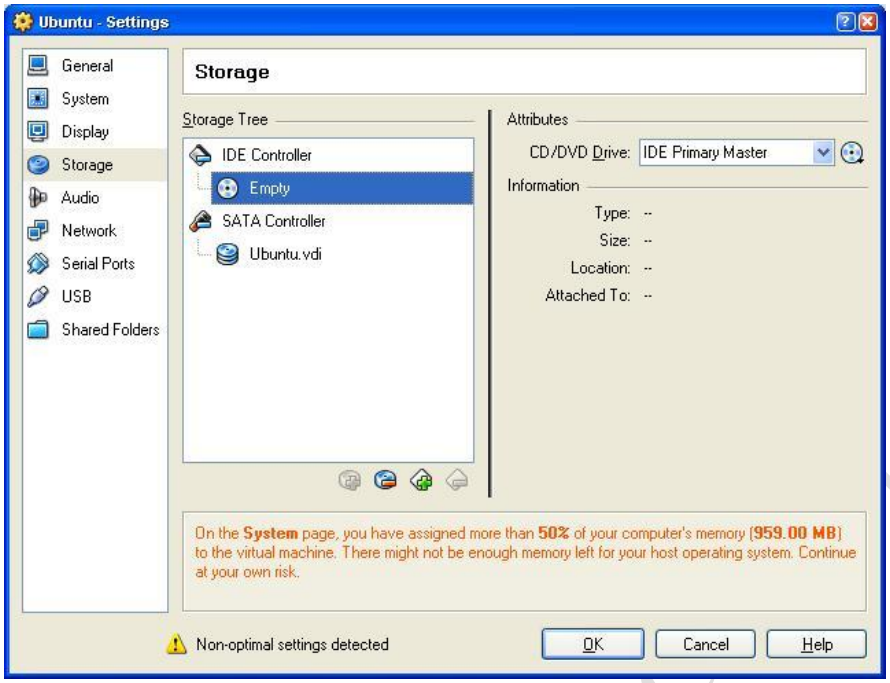


Figure Appedix 2-3-17

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"

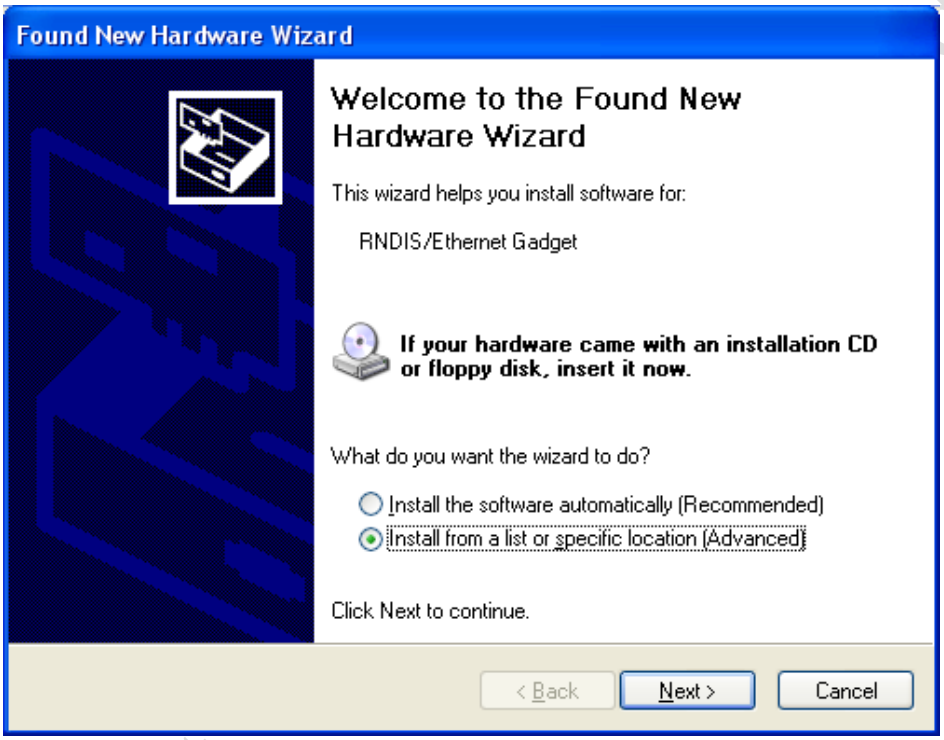


Figure Appedix 3-1

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

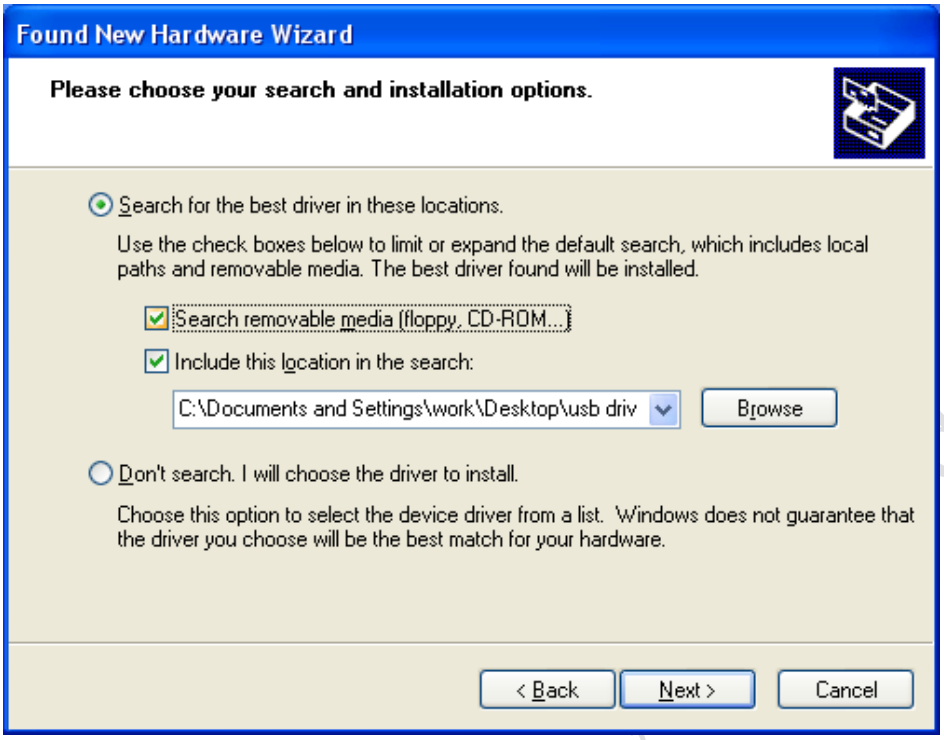


Figure Appedix 3-2

3. When the following appears, select “Continue”

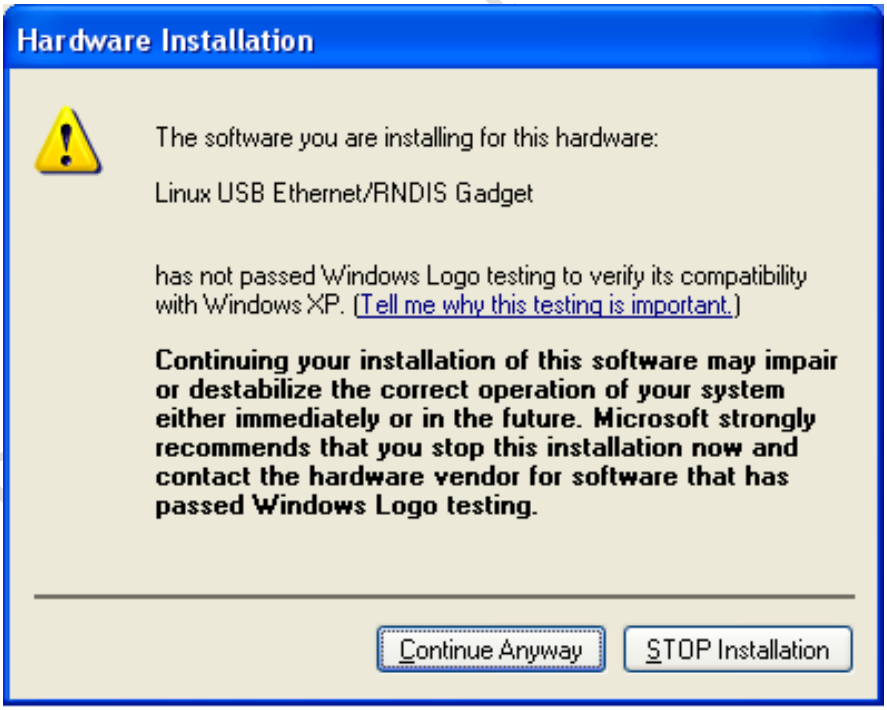


Figure Appedix 3-3

4. Please wait until the installation is completed

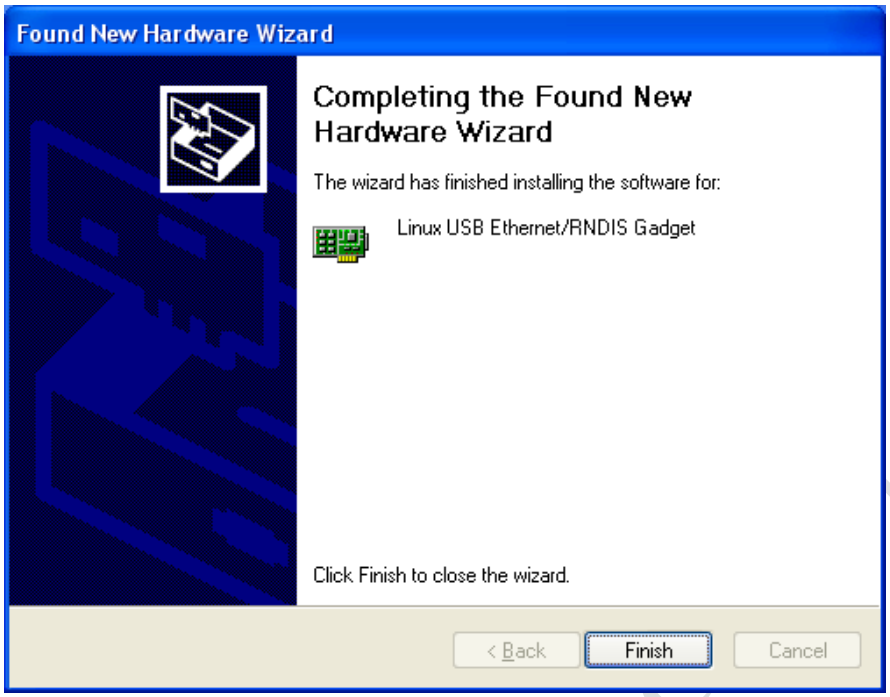


Figure Appedix 3-4

Appendix IV 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.

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 cannot 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 develops.
- Other value added services like power adapter, LCD parts.
- Other OEM/ODM services.
- Technically training.

Please contact Embest to get technical support:

- Support Tel:+86-755-25635626-872/875/897
- Fax:+86-755-25635626-666
- Pre-Sale consultation: market@embedinfo.com
- After-Sale consultation: support@embedinfo.com