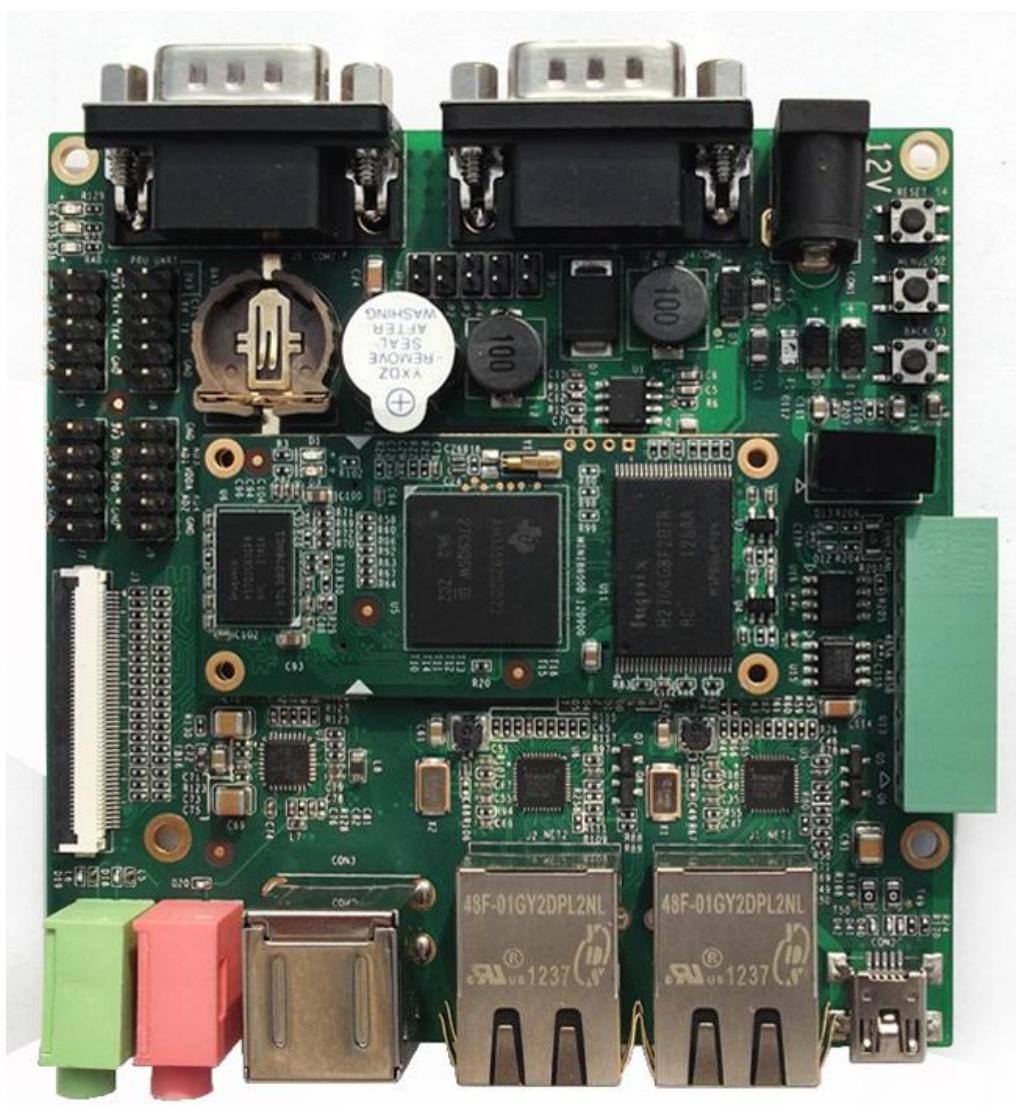


# SBC8600B 单板机



## 用户手册

---

版本 1.0 – 2012 年 12 月 21 日

## 版权声明：

- SBC8600B 开发套件及其相关知识产权由深圳市英蓓特科技有限公司所有。
- 本文档由深圳市英蓓特科技有限公司版权所有，并保留一切权利。在未经英蓓特公司书面许可的情况下，不得以任何方式或形式来修改、分发或复制本文档的任何部分。
- Microsoft, MS-DOS, Windows, Windows95, Windows98, Windows2000, Windows xp, Windows Embedded Compact 7 由微软公司授权使用。

## 版本更新记录：

版本	更新日期	描述
1.0	2012-12-21	初始版本

# 目录

<b>第1章 概述.....</b>	<b>1</b>
1.1 产品介绍 .....	1
1.2 硬件特性 .....	1
1.2.1 Mini8600B .....	1
1.2.2 扩展板.....	3
1.3 配套模块支持情况.....	5
<b>第2章 硬件系统 .....</b>	<b>6</b>
2.1 CPU.....	6
2.1.1 CPU 介绍.....	6
2.1.2 CPU 特性.....	6
2.2 外围芯片介绍.....	7
2.2.1 NAND Flash H27U4G8F2DTR-BC .....	7
2.2.2 DDR H5TQ2G83CFR-H9C .....	8
2.2.3 Ethernet AR8035.....	8
2.2.4 MAX3232.....	8
2.3 硬件接口 .....	9
2.3.1 Mini8600B .....	9
2.3.2 扩展板 .....	15
<b>第3章 LINUX 操作系统 .....</b>	<b>23</b>
3.1 介绍.....	23
3.2 软件资源 .....	23
3.3 软件特性 .....	24
3.4 系统开发 .....	25
3.4.1 开发环境搭建 .....	25
3.4.2 系统编译 .....	26

3.4.3 系统定制 .....	28
3.5 驱动介绍 .....	30
3.5.1 NAND .....	30
3.5.2 SD/MMC .....	31
3.5.3 LCDC .....	32
3.5.4 Audio in/out .....	33
3.6 驱动开发 .....	34
3.6.1 GPIO_keys 驱动 .....	34
3.6.2 GPIO_leds 驱动 .....	39
3.7 系统更新 .....	42
3.7.1 TF 卡系统映像更新 .....	42
3.7.2 NAND Flash 更新/恢复 .....	46
3.8 使用说明 .....	48
3.8.1 显示方式选择 .....	48
3.8.2 测试 .....	49
3.8.3 Demo .....	63
3.9 上层开发 .....	66
<b>第 4 章 WINDOWS EMBEDDED COMPACT 7 操作系统 .....</b>	<b>68</b>
4.1 介绍 .....	68
4.2 软件资源 .....	68
4.3 特性 .....	69
4.4 系统开发 .....	70
4.4.1 集成开发环境安装 .....	70
4.4.2 提取 BSP 及样例工程文件到集成开发环境 .....	70
4.4.3 Sysgen & BSP 编译 .....	71
4.4.4 驱动介绍 .....	71
4.5 系统映像更新 .....	72

4.5.1 TF 卡映像更新 .....	73
4.5.2 NAND Flash 映像更新 .....	78
4.6 使用说明 .....	78
4.7 应用开发 .....	78
4.6.1 如何使用 openGL ES demo .....	79
4.7.1 应用程序接口与示例 .....	79
4.7.2 GPIO 应用程序接口与示例 .....	79
附录 .....	82
附录一 硬件尺寸图 .....	82
附录二 UBUNTU 安装 .....	84
附录三 LINUX USB ETHERNET/RNDIS GADGET 驱动安装 .....	96
附录四 LINUX BOOT DISK FORMAT .....	98
附录五 TFTP 服务器搭建 .....	103
附录六 FAQ 总结 .....	105
技术支持和保修服务 .....	106

# 第 1 章 概述

## 1.1 产品介绍

SBC8600B 是英蓓特推出的一款基于 AM3359 的嵌入式单板机，它采用核心板 Mini8600B 加底板的分离式结构进行设计。主板板载 6 路串口（其中 1 路带隔离 RS485 接口），1 路带隔离 CAN2.0 接口，2 个千兆以太网口，2 路 USB Host 和 1 路 USB OTG，LCD 触摸屏，TF 卡等接口。支持 Linux3.2.0，WinCE 7 及 Android 2.3 三种操作系统。资料提供包括用户手册、PDF 原理图、外扩接口驱动、BSP 源码包、开发工具等，为开发者提供了完善的软件开发环境，降低产品开发周期，实现面向包括便携式导航系统、数字视频机顶盒、便携式教育/游戏设备、工业自动化、楼宇自动化、人机界面、教学/医疗设备等行业应用产品快速上市。

## 1.2 硬件特性

SBC8600B 评估板是基于 AM3359 处理器，同时也是集成了此芯片主要功能与特性的评估板，以下是板子的特性：

### 1.2.1 Mini8600B

#### 电气参数

- 工作温度：0 °C~ 70°C
- 环境温度：20% ~ 90%，非冷凝
- 机械尺寸：60mm x 27mm
- 输入电压：3.3V

#### 处理器

- 720-MHz ARM Cortex™-A8 32-Bit RISC Microprocessor
  - NEON™ SIMD Coprocessor

- 32KB/32KB of L1 Instruction/Data Cache with Single-Error Detection (parity)
- 256KB of L2 Cache with Error Correcting Code (ECC)
- SGX530 Graphics Engine
- Programmable Real-Time Unit Subsystem

#### 存储器

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

#### 板对板连接器和引出接口信号

- 两个0.4mm间距2\*40-pin排针
- TFT LCD 信号(支持24-bpp 并行RGB接口LCD)
- 2路USB2.0 High-Speed OTG信号
- 6路UART信号
- 1路SPI信号
- 2路 10/100 /1000Mb/s 以太网MAC(EMAC), 带管理数据输入/输出模块 (MDIO)
- 1路McASP信号
- 8路12bit ADC接口
- 3路IIC总线信号
- 2路4线SDMMC信号
- GPMC信号

#### 注意:

UART、IIC、SPI、CAN 存在部分引脚复用，详细情况请参考芯片手册和附带原理图

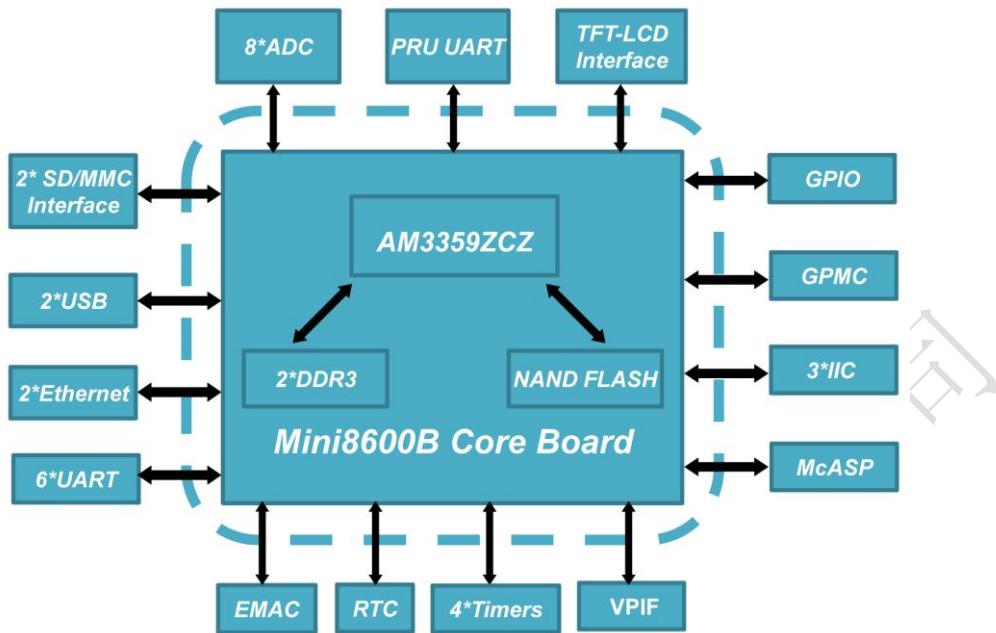


图1-1 Mini8600B结构图

## 1.2.2 扩展板

### 电气参数

- 工作温度: 0 °C~ 70°C
- 环境湿度: 20% ~ 90%, 非冷凝
- 机械尺寸: 95m x 95m
- 输入电压: 12V/1.25A

### 音频/视频接口

- LCD/4线电阻触摸屏接口 (24位数据RGB全彩色输出, 50-pin FPC连接器 )
- 一个音频输入接口(3.5mm音频接口)
- 一个双声道音频输出接口(3.5mm音频接口)

### 数据传输接口

- 两个10/100/1000Mbps以太网接口(WinCE 7仅支持一个以太网口)
- 一个CAN 2.0 接口和一个RS485接口(8 Pin 凤凰端子连接器)
- 一个USB 2.0 High-Speed OTG Ports with Integrated PHY (480Mbps , Mini USB接口)

- 两个USB 2.0 High-Speed HOST Ports with Integrated PHY (480Mbps, USB-A接口)
- 一个TF卡接口（兼容SD/MMC通信，3.3V逻辑）
- 串口
  - UART0, 3线RS232电平, DB9调试串口
  - UART2, 3线RS232电平, DB9普通串口
  - UART3, 3线TTL电平, 排针引出
  - UART4, 3线TTL电平, 排针引出
  - UART5, 3线TTL电平, 排针引出
- GPIO接口

#### 输入接口及其他

- 二个自定义按键（MENU、BACK）
- 一个复位按键
- 一个蜂鸣器
- 一个电源指示灯
- 两个用户自定义灯

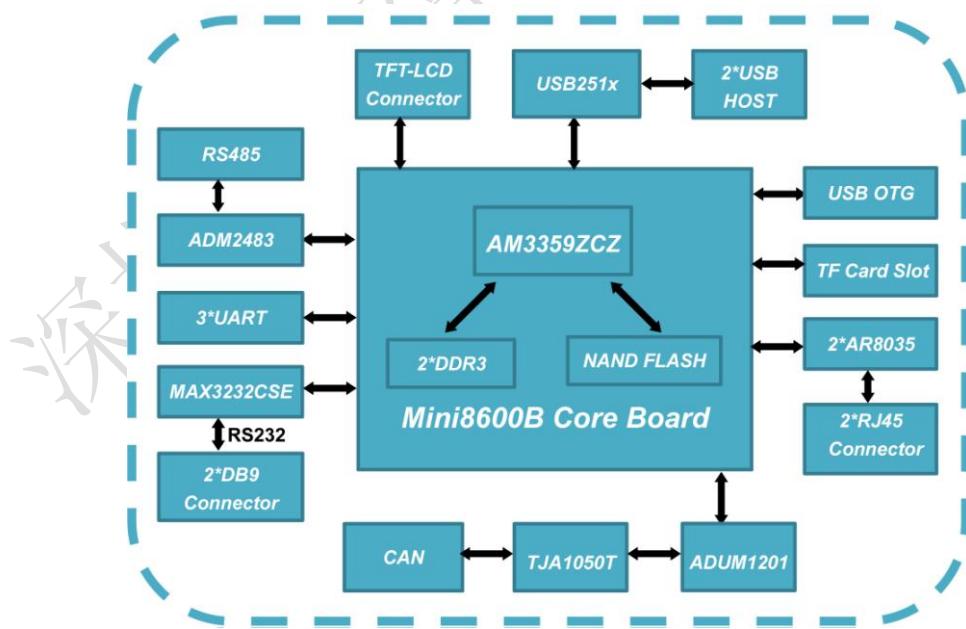


图1-2 SBC8600B结构图

### 1.3 配套模块支持情况

表1

名称	Linux	Android	WinCE	相关资料
VGA8000	YES*	YES*	YES*	开发板配套光盘提供
WF8000-U	YES*	NO	NO	单独光盘提供
CAM8100-U	YES*	NO	NO	单独光盘提供
CDMA8000-U	YES*	NO	NO	<a href="#">点击下载</a>
WCDMA8000-U	YES*	NO	NO	<a href="#">点击下载</a>
LVDS8000	YES*	YES*	YES*	开发板配套光盘和网站提供

图1-3

## 第 2 章 硬件系统

### 2.1 CPU

#### 2.1.1 CPU 介绍

AM3359 是基于 ARM Cortex-A8 的微处理器，在图像、图形处理、外设和诸如 etherCAT 和 PROFIBUS 的工业接口选项方面进行了增强，并支持 Linux, WinCE, Android 等高级操作系统。

微处理器包含下列子系统：MPU 子系统基于 ARM Cortex-A8 微处理器；POWERVR SGX 图形加速子系统，主要用于 3D 图形加速以支持显示和游戏效果；可编程实时单元子系统(PRUSS)，使用户可以创建各种超越本地外设的数字资源。此外，PRUSS 独立于 ARM 核，这就允许设备有独立的操作和时钟，在复杂系统解决方案中有更大的灵活性。

#### 2.1.2 CPU 特性

##### Clock

AM3359 微处理器有两个输入时钟：OSC1 和 OCC0，两个输出时钟信号：LCKOUT1 和 LCKOUT2.

OSC1 振荡器为 RTC 提供 32.768KHZ 参考时钟并用于连接 RTC\_XTALIN 和 RTC\_XTALOUT 终端。

OCC0 振荡器为所有无 RT 功能的时钟提供 19.2-MHz, 24-MHz, 25-MHz 或 26-MHz 参考时钟，并用于连接 XTALIN 和 XTALOUT 终端

##### Reset

复位功能取决于 CPU 的 PWRONRSTn 信号，低电平有效。

#### 通用接口（General-Purpose Interface）

通用接口包括 4 组通用输入输出接口 (GPIO)，每一组 GPIO 模组提供 32 个专用的通用接口输入输出管脚，因此通用的 GPIO 可以高达 128 个 (4x32) 管脚。

### 可编程实时单元子系统 (Programmable Real-Time Unit Subsystem)

AM3359 下的可编程实时单元子系统(PRUSS)包括两个可编程实时单元、12KB 带 Single-Error 检测 (奇偶校验) 的共享 RAM、3 个可被每个 PRU 的 120 字节寄存器 BANK、用于处理系统输入事件的中断控制器模块和以下内部外设：

- 1 路 UART，具有流控制，最高 12Mbps
- 两路 MII 以太网端口，支持工业以太网，例如 EtherCAT™
- 1 路 MDIO 端口
- 一路增强型捕获模块 (eCAP)

### 3D 图形引擎

POWERVR® SGX 图形加速器子系统用于 3D 图形加速以支持显示和游戏效果，该子系统的主要特性如下：

- Tile-Based 架构，处理能力高达 20Mploy/秒
- 通用可扩展渲染引擎是一个具有像素和顶点渲染功能的多线程引擎
- 超过 Microsoft VS3.0、PS3.0 和 OGL2.0 的高级渲染功能指令集
- 工业标准 API，支持 Direct3D Mobile、OGL-ES 1.1 和 2.0、OpenVG 1.0 和 OpenMaxI

## 2.2 外围芯片介绍

### 2.2.1 NAND Flash H27U4G8F2DTR-BC

H27U4G8F2DTR-BC 是 SBC8600B 的 NAND Flash 芯片，大小为 512M。

若想了解更多关于此芯片的信息，请打开 Disk-SBC8600B\HW design\datasheet\NAND Flash\ H27U4G8F2DTR-BC.pdf 文档。

## 2.2.2 DDR H5TQ2G83CFR-H9C

H5TQ2G83DFR-H9C 是 SBC8600B 的 DDR3 SDRAM 芯片，大小为 256MB，SBC8600B 由 2 片 H5TQ2G83DFR-H9C 芯片构成。

若想了解更多关于此芯片的信息，请打开 Disk-SBC8600B\HW design\datasheet\DDR\H5TQ2G83DFR.pdf 文档

## 2.2.3 Ethernet AR8035

AR8035 是 SBC8600B 低功耗、低 BOM 成本的以太网芯片，它集成了 10/100/1000 千兆位收发器。它是单端口 10/100/1000 Mbps 三速以太网 PHY，并支持 MAC.TM RGMII 接口。

AR8035 支持 IEEE 802.3az 高效节能以太网（EEE）标准和 Atheros 专有的 SmartEEE，它允许无需 802.3az 功能支持的传统 MAC/SoC 设备作为完整的 802.3az 系统。

SBC8600B 可通过直通网线连接到网络 hub 上，也可用交叉网线与电脑直接相连。

若想了解更多关于此芯片的信息，请打开 Disk-SBC8600B\HW design\datasheet\LAN\AR8035.pdf 文档。

## 2.2.4 MAX3232

MAX3232 的功能主要是将 TTL 电平转换为 RS232 电平，以适应与 PC 的 RS232 串口 相互通信。

SBC8600B 使用的是 UART0 作调试串口，因 CPU 的 UART0 默认电压是 1.8V，需要 将电压转换为 3.3V，方可满足外部使用。

若想了解更多关于此芯片的信息，请打开 Disk-SBC8600B\HW design\datasheet\Serial\MAX3232CSE.pdf 文档。

## 2.3 硬件接口

### 2.3.1 Mini8600B

#### 2.3.1.1 CN1 接口



图2-1 Mini8600 CN1 接口示意图

表2 CN1 接口

CN1		
管脚	信号	描述
1	GND	GND
2	VDDS_RTC	Supply voltage for RTC
3	CLK_OUT1	Clock out1
4	CLK_OUT2	Clock out2
5	MMC0_DAT0	MMC0 data bus
6	MMC0_DAT1	MMC0 data bus
7	MMC0_DAT2	MMC0 data bus
8	GLOBLE_RESETN	SYS_RESET IN/ OUTPUT
9	MMC0_DAT3	MMC0 data bus
10	AM335X_PWRON_RESETN	CPU PWRON Reset
11	GND	GND
12	GND	GND
13	AM355X_PRU_UART0_CTS	PRU UART0 Clear To Send
14	AM355X_PRU_UART0_RX	PRU UART0 receive data
15	AM355X_PRU_UART0_RTS	PRU UART0 request to send
16	AM355X_PRU_UART0_TX	PRU UART0 transmit data
17	AM355X_UART0_RX	UART0 receive data

CN1		
管脚	信号	描述
18	AM355X_UART3_RX	UART3 receive data
19	AM355X_UART0_TX	UART0 transmit data
20	AM355X_UART3_TX	UART3 transmit data
21	AM355X_CAN0_RX	CAN0 receive data
22	AM355X_I2C0_SDA	I2C0 master serial data
23	AM355X_CAN0_TX	CAN0 transmit data
24	AM355X_I2C0_SCL	I2C0 master serial clock
25	AM355X_UART4_RX	UART4 receive data
26	AM355X_UART1_RX	UART1 receive data
27	AM355X_UART4_TX	UART4 transmit data
28	AM355X_UART1_TX	UART1 transmit data
29	GND	GND
30	GND	GND
31	MII1_COL	MII1 collision detect
32	AM355X_USB0_DRVVBUS	USB0 controller VBUS control output
33	MII1_TX_CLK	MII1 transmit clock
34	AM355X_USB1_DRVVBUS	USB1 controller VBUS control output
35	MII1_TX_EN	MII1 transmit enable
36	MII1_REF_CLK	MII1 reference clock
37	MII1_TXD3	MII1 transmit data
38	MII1_CRS	MII1 carrier sense
39	MII1_TXD2	MII1 transmit data
40	MII1_RX_ER	MII1 receive data error
41	MII1_TXD1	MII1 transmit data
42	MII1_RX_DV	MII1 receive data valid
43	MII1_TXD0	MII1 transmit data
44	MII1_RX_CLK	MII1 receive clock
45	MII_MDIO	MII MDIO DATA
46	MII1_RXD3	MII1 receive data
47	MII_MDC	MII MDIO CLK
48	MII1_RXD2	MII1 receive data
49	GND	GND
50	MII1_RXD1	MII1 receive data

CN1		
管脚	信号	描述
51	AM355X_USB0_DM	USB0 DM-
52	MII1_RXD0	MII1 receive data
53	AM355X_USB0_DP	USB0 DP
54	MMC0_CMD	MMC0 Command Signal
55	GND	GND
56	USB0_VBUS	USB0 bus voltage
57	AM355X_USB1_DM	USB1 data-
58	AM355X_USB1_ID	USB1 ID
59	AM355X_USB1_DP	USB1 data+
60	AM355X_USB0_ID	USB0 ID
61	GND	GND
62	USB1_VBUS	USB1 bus voltage
63	GPMC_A0	GPMC address
64	GPMC_A7	GPMC address
65	GPMC_A5	GPMC address
66	GPMC_A11	GPMC address
67	GPMC_A4	GPMC address
68	GPMC_A10	GPMC address
69	GPMC_A3	GPMC address
70	GPMC_A9	GPMC address
71	GPMC_A2	GPMC address
72	GPMC_A8	GPMC address
73	GPMC_A6	GPMC address
74	GPMC_A1	GPMC address
75	GND	GND
76	GND	GND
77	VDD_3V3	Power
78	VDD_3V3	Power
79	VDD_3V3	Power
80	VDD_3V3	Power

### 2.3.1.2 CN2 接口



图2-2 Mini8600B CN2 接口示意图

表3 CN1 接口

CN2		
管脚	信号	描述
1	GND	GND
2	GND	GND
3	MCASP0_AHCLKX	MCASP0 transmit master clock
4	MCASP0_ACLKX	MCASP0 transmit bit clock
5	MCASP0_FSX	MCASP0 transmit frame sync
6	MCASP0_AXR0	MCASP0 serial data(I/O)
7	MCASP0_AHCLKR	MCASP0 receiver master clock
8	MMC0_CLK	MMC0 clock
9	MCASP0_FSR	MCASP0 receive frame sync
10	MCASP0_AXR1	MCASP0 serial data(I/O)
11	GND	GND
12	GND	GND
13	VDDA_ADC	Supply voltage range for ADC
14	AM355X_ADC0	ADC0
15	AM355X_ADC1	ADC1
16	AM355X_ADC2	ADC2
17	AM355X_ADC3	ADC3
18	AM355X_ADC4	ADC4
19	AM355X_ADC5	ADC5
20	AM355X_ADC6	ADC6
21	AM355X_ADC7	ADC7
22	GND_ADC	GND ADC

CN2		
管脚	信号	描述
23	GND	GND
24	GND	GND
25	LCD_DATA1	LCD data bus
26	LCD_DATA12	LCD data bus
27	LCD_DATA0	LCD data bus
28	LCD_DATA10	LCD data bus
29	LCD_DATA5	LCD data bus
30	LCD_DATA13	LCD data bus
31	LCD_DATA4	LCD data bus
32	LCD_DATA11	LCD data bus
33	LCD_DATA6	LCD data bus
34	LCD_DATA14	LCD data bus
35	LCD_DATA8	LCD data bus
36	LCD_VSYNC	LCD vertical sync
37	GND	GND
38	GND	GND
39	LCD_DATA9	LCD data bus
40	LCD_PCLK	LCD pixel clock
41	LCD_DATA15	LCD data bus
42	GPMC_AD11	GPMC address & data
43	LCD_DATA3	LCD data bus
44	GPMC_AD15	GPMC address & data
45	LCD_DATA2	LCD data bus
46	GPMC_AD14	GPMC address & data
47	LCD_DATA7	LCD data bus
48	GPMC_WAIT0	GPMC wait0
49	LCD_HSYNC	LCD horizontal sync
50	GPMC_BEN1	GPMC byte enable 1
51	GND	GND
52	GND	GND
53	LCD_EN	LCD AC bias enable chip select
54	GPMC_WPN	GPMC write protect
55	GPMC_AD13	GPMC address & data
56	GPMC_CSN3	GPMC chip select
57	GPMC_AD9	GPMC address & data

CN2		
管脚	信号	描述
58	GPMC_CSN2	GPMC chip select
59	GPMC_AD10	GPMC address & data
60	GPMC_CLK	GPMC clock
61	GPMC_AD8	GPMC address & data
62	GPMC_AD6	GPMC address & data
63	GPMC_AD12	GPMC address & data
64	GND	GND
65	GND	GND
66	GPMC_CSN1	GPMC chip select1
67	GPMC_ADVN_ALE	GPMC address valid/address latch enable
68	GPMC_AD5	GPMC address & data
69	GPMC_BEN0_CLE	GPMC byte enable 0/Command latch enable
70	GPMC_AD4	GPMC address & data
71	GPMC_OEN_REN	GPMC output /read enable
72	GPMC_AD1	GPMC address & data
73	GPMC_AD2	GPMC address & data
74	GPMC_AD0	GPMC address & data
75	GPMC_AD3	GPMC address & data
76	GPMC_CSN0	GPMC chip select0
77	GPMC_AD7	GPMC address & data
78	GPMC_WEN	GPMC write enable
79	GND	GND
80	GND	GND

### 2.3.2 扩展板

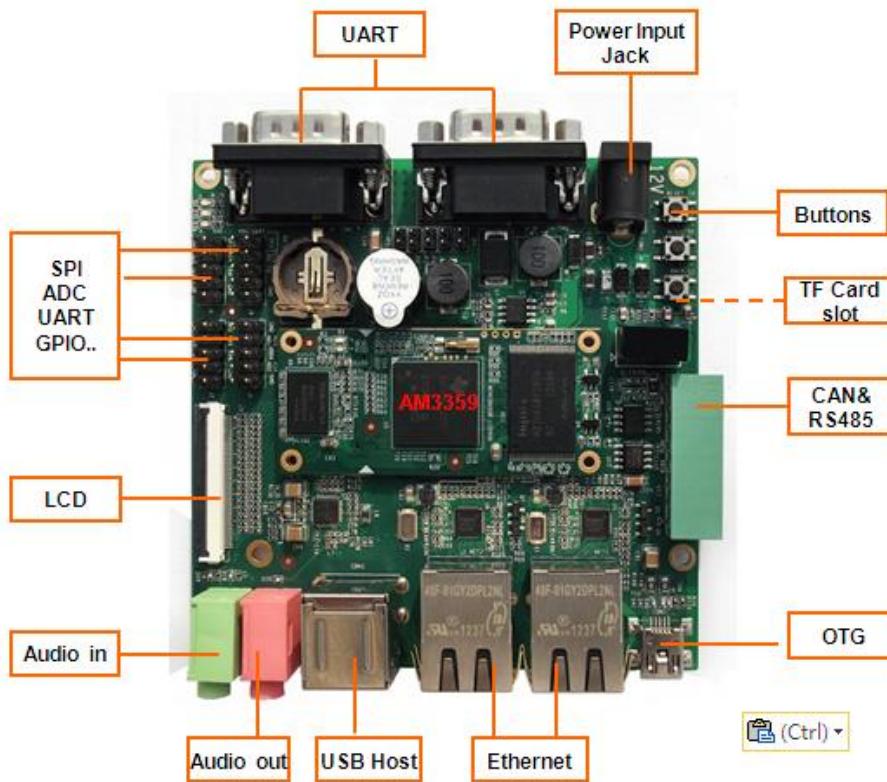


图2-3 SBC8600B 扩展板接口示意图

--- 代表接口在板子背面

— 代表接口在板子正面

#### 2.3.2.1 电源接口

表4 电源接口

CON1		
管脚	信号	描述
1	GND	GND
2	+12V	Power supply (+12V)
3	NC	NC

### 2.3.2.2 TFT\_LCD 接口

表5 TFT\_LCD 接口

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

J3		
管脚	信号	描述
32	CLK	LCD Pixel Clock
33	GND4	GND
34	X+	X+ Position Input
35	X-	X- Position Input
36	Y+	Y+ Position Input
37	Y-	Y- Position Input
38	NC	NC
39	NC	NC
40	NC	NC
41	NC	NC
42	IIC_CLK	IIC master serial clock
43	IIC_DAT	IIC serial bidirectional data
44	GND5	GND
45	VDD1	3.3V
46	VDD2	3.3V
47	VDD3	5V
48	VDD4	5V
49	RESET	Reset
50	PWREN	Backlight enable

**注意：**

请不要带电拔插 LCD 排线

### 2.3.2.3 音频输出接口

表6 音频输出接口

HEADPHONE1		
管脚	信号	描述
1	GND	GND
2	NC	NC
3	Right	Right output
4	NC	NC
5	Left	Left output

### 2.3.2.4 音频输入接口

表7 音频输入接口

MIC1		
管脚	信号	描述
1	GND	GND
2	NC	NC
3	MIC In	input
4	NC	NC
5	MIC In	input

### 2.3.2.5 USB HOST 接口

表8 USB HOST 接口

CON3		
管脚	信号	描述
1	VBUA	+5V
2	DA-	USB Data-
3	DA+	USB Data+
4	GNDA	GND

### 2.3.2.6 USB OTG 接口

表9 USB OTG 接口

CON2		
管脚	信号	描述
1	VB	+5V
2	D-	USB Data-
3	D+	USB Data+
4	ID	USB ID
5	G1	GND

### 2.3.2.7 TF 卡接口

表10 TF 卡接口

TF1		
管脚	信号	描述
1	DAT2	Card data 2
2	CD/DAT3	Card data 3
3	CMD	Command Signal

TF1		
管脚	信号	描述
4	VDD	VDD
5	CLOCK	Clock
6	VSS	VSS
7	DAT0	Card data 0
8	DAT1	Card data 1
9	CD	Card detect

### 2.3.2.8 LAN 接口

表11 LAN 接口

J1,J2		
管脚	信号	描述
1	TD1+	Transmit Data1+
2	TD1-	Transmit Data1-
3	TD2+	Transmit Data2+
4	TD2-	Transmit Data2-
5	TCT	Transmit common terminal
6	RCT	Receive common terminal
7	RD1+	Receive Data1+
8	RD1-	Receive Data1-
9	RD2+	Receive Data2+
10	RD2-	Receive Data2-
11	GRLA	+2.5V
12	GRLC	LINK active LED
13	YELC	100M linked LED
14	YELA	+2.5V

### 2.3.2.9 串口

表12 串口

J4(UART0), J5(UART2)		
管脚	信号	描述
1	NC	NC
2	RXD	Receive data
3	TXD	Transmit data
4	NC	NC
5	GND	GND

<b>J4(UART0), J5(UART2)</b>		
管脚	信号	描述
6	NC	NC
7	RTS	Request To Send
8	CTS	Clear To Send
9	NC	NC

### 2.3.2.10 CAN&RS485 接口

表13 CAN&RS485 接口

<b>U22</b>		
管脚	信号	描述
1	+12V	+12V
2	GND	GND
3	GND2	Isolated GND
4	485B1	485B
5	485A1	485A
6	GND1	Isolated GND
7	CANL1	CANL
8	CANH	CANH

### 2.3.2.11 ADC

表14 ADC

<b>J9</b>		
管脚	信号	描述
1	GND	GND
2	GND	GND
3	ADC_CH1	ADC1
4	ADC_CH3	ADC3
5	VDDA_ADC	Power
6	VDDA_ADC	Power
7	ADC_CH2	ADC2
8	ADC_CH4	ADC4
9	GND	GND
10	GND	GND

### 2.3.2.12 SPI 接口

表15 SPI 接口

J8		
管脚	信号	描述
1	+3.3V	3.3V
2	+3.3V	3.3V
3	SPI0_D1	SPI0 data1
4	SPI0_CLK	SPI0 clock
5	SPI0_CS0	SPI enable0
6	SPI0_D0	SPI data0
7	GND	GND
8	GND	GND
9	GND	GND
10	GND	GND

### 2.3.2.13 扩展接口

表16 扩展接口 1

J6		
管脚	信号	描述
1	VIO_3V3	+3.3V
2	VIO_3V3	+3.3V
3	UART3_TX_3V3	UART3 Transit data 3.3V level
4	UART4_TX_3V3	UART4 Transit data 3.3V level
5	UART3_RX_3V3	UART3 receive data 3.3V level
6	UART4_RX_3V3	UART4 receive data 3.3V level
7	GND	GND
8	GND	GND
9	GND	GND
10	GND	GND

表17 扩展接口 2

J7		
管脚	信号	描述
1	VIO_3V3	+3.3V
2	VIO_3V3	+3.3V
3	UART5_TX_3V3	UART5 Transit data 3.3V level
4	GPIO0_9	GPIO

J7		
管脚	信号	描述
5	UART5_RX_3V3	UART5 receive data 3.3V level
6	GPIO2_0	GPIO
7	GND	GND
8	GND	GND
9	GND	GND
10	GND	GND

### 2.3.2.14 按钮

表18 按钮

S1-3		
管脚	信号	描述
S2	MENU	System menu key
S3	BACK	System back key
S4	Reset	System Reset key

### 2.3.2.15 LED

表19 LED 灯

LEDs		
LED 灯	灯定义	描述
1	D4	电源指示灯
2	D35	用户自定义灯
3	D36	用户自定义灯

# 第3章 Linux 操作系统

## 3.1 介绍

此部分主要介绍 SBC8600B Linux 软件系统，内容如下：

- SBC8600B 提供的软件资源介绍；
- 软件的特性介绍；
- 开发环境搭建，系统开发，驱动原理以及驱动开发介绍；
- 系统更新介绍；
- 系统使用说明；
- 上层开发介绍。

**注意：**

此部分建议用户：

- 预先安装好 Ubuntu Linux 系统，具体可以参考附录二部分；
- 掌握相关的嵌入式 Linux 开发技术。

## 3.2 软件资源

SBC8600B出厂时，NAND Flash已经烧写好Linux操作系统。嵌入式Linux系统由spl、u-boot、kernel和rootfs四部分组成，系统结构如图所示：

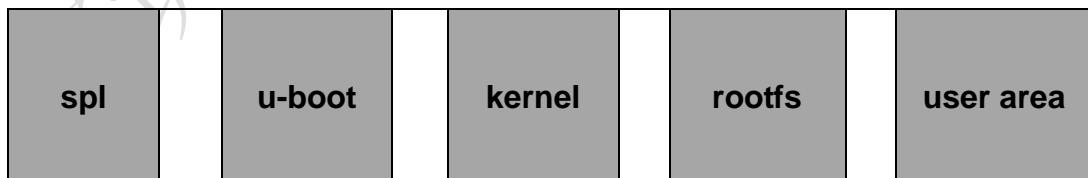


图 3-1

系统各组成部分特性及作用介绍如下：

1) spl是一级引导程序，系统上电后由CPU内部ROM自动拷贝到内部RAM并执行。

主要作用为初始化CPU，拷贝u-boot到内存中，然后把控制权交给u-boot；

- 2) u-boot是二级引导程序，主要用于和用户进行交互，提供映像更新、引导内核等功能；
- 3) kernel使用Linux3.2.0 内核，根据SBC8600B的硬件进行定制；
- 4) rootfs采用开源文件系统ubifs，特别适用于嵌入式系统。

### 3.3 软件特性

表20

名称		备注
BIOS	spl	NAND
		MMC/SD
		FAT
	u-boot	NAND
		MMC/SD
		FAT
		NET
Kernel	Linux-3.2.0	支持 ROM/CRAM/EXT2/EXT3/FAT/NFS/JFFS2/UBIFS等多种文件系统
Device Driver	serial	串口驱动
	rtc	硬件时钟驱动
	net	10/100M/1000M以太网驱动
	can	can总线驱动
	flash	nand flash驱动(支持nand boot)
	lcd	TFT LCD 驱动
	touch screen	4 线触摸屏控制器驱动
	mmc/sd	mmc/sd控制器驱动
	usb otg	usb otg 2.0 驱动
	audio	声卡驱动(支持录/放音)
	keypad	gpio键盘驱动
Demo	led	用户led灯驱动
	Android	android 2.3.4 系统
	TISDK	TISDK系统

## 3.4 系统开发

### 3.4.1 开发环境搭建

用户使用SBC8600B进行软件开发之前，必须先搭建Linux交叉开发环境，并安装到电脑的Linux系统。下面以Ubuntu操作系统为例，介绍如何搭建交叉开发环境。

#### 3.4.1.1 交叉编译工具安装

插入光盘，Ubuntu默认把光盘挂载到/media/cdrom目录下，交叉编译工具就存放在/media/cdrom/linux/tools目录下。

在Ubuntu终端执行下面指令，将交叉编译工具解压到\$HOME目录下面：

**注意：**

每一条指令前都加上了符号“•”，以便防止由于指令较长占用多行而造成误解

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

源码编译中用到其它的一些工具，同样存放在光盘的 linux/tools 目录下，用户可执行以下命令拷贝：

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

#### 3.4.1.2 添加环境变量

以上工具安装完成后，还需要使用如下命令把它们添加到临时环境变量：

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

**注意:**

- 用户可把它写入用户目录的.bashrc 文件中，那么系统启动的时候自动完成环境变量的添加，查看路径可以使用 echo \$PATH 命令。

### 3.4.1.3 android 开发环境搭建

除了安装交叉编译工具和添加环境变量以外，在Ubuntu中还必须安装一些安装包和进行一些配置才能编译android文件系统源码包，具体请参考：

<http://source.android.com/source/initializing.html>。

## 3.4.2 系统编译

### 3.4.2.1 准备

系统所有组成部分的源码位于光盘的 linux/source 目录下，用户在进行开发前需要把它们解压至 Ubuntu 系统：

```
• mkdir $HOME/work  
• cd $HOME/work  
• tar xvf /media/cdrom/linux/source/u-boot-2011.09-psp04.06.00.03.tar.bz2  
• tar xvf /media/cdrom/linux/source/linux-3.2.0-psp04.06.00.08.sdk.tar.bz2  
• tar xvf /media/cdrom/linux/demo/android/source/linux-3.1.0-android.tar.bz2  
• sudo tar xvf /media/cdrom/linux/source/rootfs.tar.bz2  
• tar xvf  
  /media/cdrom/linux/demo/android/source/rowboat-android-gingerbread-am335xevm.tar.bz2
```

执行完以上操作后，当前目录下会生成 u-boot-2011.09-psp04.06.00.03、Linux-3.2.0-psp04.06.00.08.sdk、Linux-3.1.0-android、rootfs 和 rowboat-android-gingerbread-am335xevm 目录。

**注意:**

- 源码文件请不要解压到其他位置，以免编译时出错。

### 3.4.2.2 启动代码编译

SBC8600B 支持 MMC/SD 启动与 NAND Flash 启动，系统优先选择 MMC/SD 启动，如果 MMC/SD 启动失败，转向 NAND Flash 启动。

下面介绍启动代码映像文件的生成方法：

- **cd u-boot-2011.09-psp04.06.00.03**
- **make distclean**
- **make sbc8600\_config**
- **make**

执行完以上操作后，当前目录下会生成我们需要的启动代码映像 MLO 和 u-boot.img 文件。

### 3.4.2.3 内核编译

对于 Linux 系统，在 Ubuntu 终端输入如下命令：

- **cd Linux-3.2.0-psp04.06.00.08.sdk**
- **make distclean**
- **make sbc8600\_defconfig**
- **make menuconfig**
- **make uImage**

对于 Andorid 系统，在 Ubuntu 终端输入如下命令：：

- **cd Linux-3.1.0-android**
- **make distclean**
- **make sbc8600\_android\_defconfig**
- **make menuconfig**
- **make uImage**

执行完以上操作后，arch/arm/boot 目录下会生成我们需要的 uImage 文件。

### 3.4.2.4 文件系统生成

#### 1) Ramdisk 文件制作：

关于 Ramdisk 的制作，请参考 [http://www.elinux.org/DevKit8600\\_FAQ](http://www.elinux.org/DevKit8600_FAQ)，本文档不作介绍。

#### 2) UBI 文件制作：

- **cd \$HOME/work**
- **sudo \$HOME/tools/mkfs.ubifs -r rootfs -m 2048 -e 126976 -c 812 -o ubifs.img**

```
• sudo $HOME/tools/ubinize -o ubi.img -m 2048 -p 128KiB -s 512 -O 2048  
$HOME/tools/ubinize.cfg
```

执行完以上操作后，当前目录下会生成我们需要的ubi.img文件。

### 3.4.2.5 Android 文件系统编译

1) 执行下面指令开始编译 Android:

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

2) 修改 hardware/ti/sgx/Rules.make 文件

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

将 KERNEL\_INSTALL\_DIR=\$(HOME)/work/Linux-3.1.0-android 修改为

KERNEL\_INSTALL\_DIR=/home/**user\_name**/work/Linux-3.1.0-android

/home/user\_name 为用户名所在目录即为\$(HOME)值，在 linux 终端输入 whoami，  
即可获知。

3) 输入下述指令，开始制作 ubi 文件系统:

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

在temp/下即可找到ubi.img。

#### 注意：

在编译 android 文件系统前，必须先对 Android 内核源码 Linux-3.1.0-android 进行编译，否则会报错。

### 3.4.3 系统定制

Linux内核有很多内核配置选项，用户可以在默认配置的基础上，增加或裁减驱动和一些内核特性，以更适合用户的需要。下面举例说明系统定制的一般流程。

#### 3.4.3.1 修改内核配置

出厂内核源码中提供有默认配置文件：

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

用户可在其基础上进行系统定制：

- `cd Linux-3.2.0-psp04.06.00.08.sdk`
- `cp arch/arm/configs/sbc8600_defconfig .config`
- `make menuconfig`

**注意：**

若输入 `make menuconfig` 系统出错，Ubuntu 系统是需要安装 `ncurse`, `ncurses` 库是字符图形库，用于 kernel 的 `make menuconfig`，具体的安装指令：  
`sudo apt-get install ncurses-dev`。

下面以 usb gadget 模拟 usb mass storage device 为例子，举例介绍系统的定制：

Select the configuration below:

-> Device Drivers

-> USB support

-> USB Gadget Support

-> USB Gadget Drivers

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

图 3-2

选择 “File-backed Storage Gadget” 为<M>，退出后选择保存重新编译内核即可。

### 3.4.3.2 编译

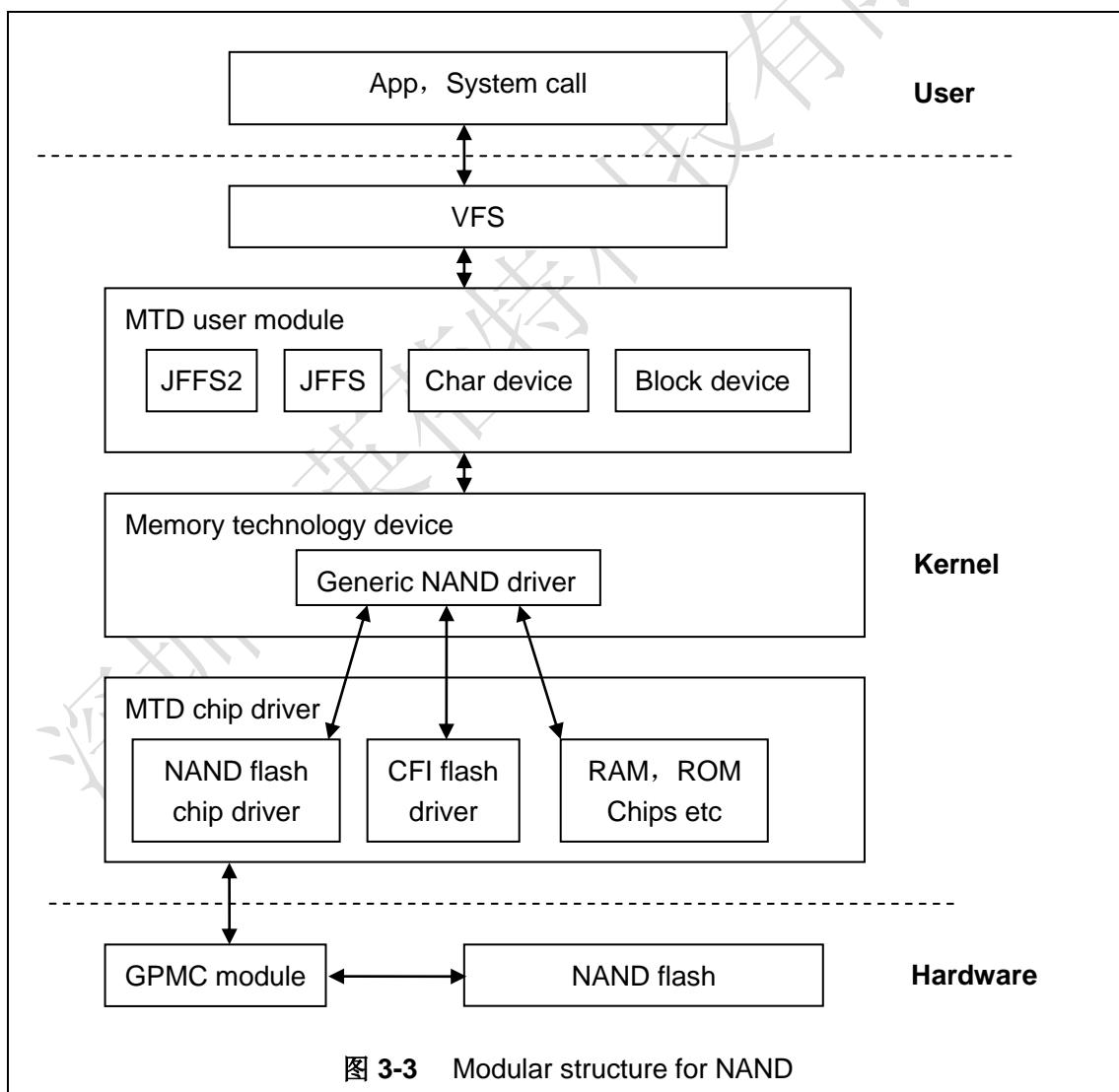
保存配置，执行以下命令重新编译内核：

- **make uimage**
- **make modules**

执行完以上操作后，arch/arm/boot 目录下生成新的内核映像 uimage，drivers/usb/gadget 目录下生成模块文件 g\_file\_storage.ko。

## 3.5 驱动介绍

### 3.5.1 NAND



嵌入式系统中使用的固态存储器主要为 flash，在本系统中为 NAND Flash。

NAND Flash 作为块设备使用，其上建立有文件系统，用户与 NAND flash 的交互主要通过具体的文件系统来完成。为了屏蔽不同 flash 存储器之间的差异，内核在文件系统与具体的 flash 驱动之间插入了 MTD 子系统进行管理。

所以，用户访问 NAND Flash 经过以下流程：

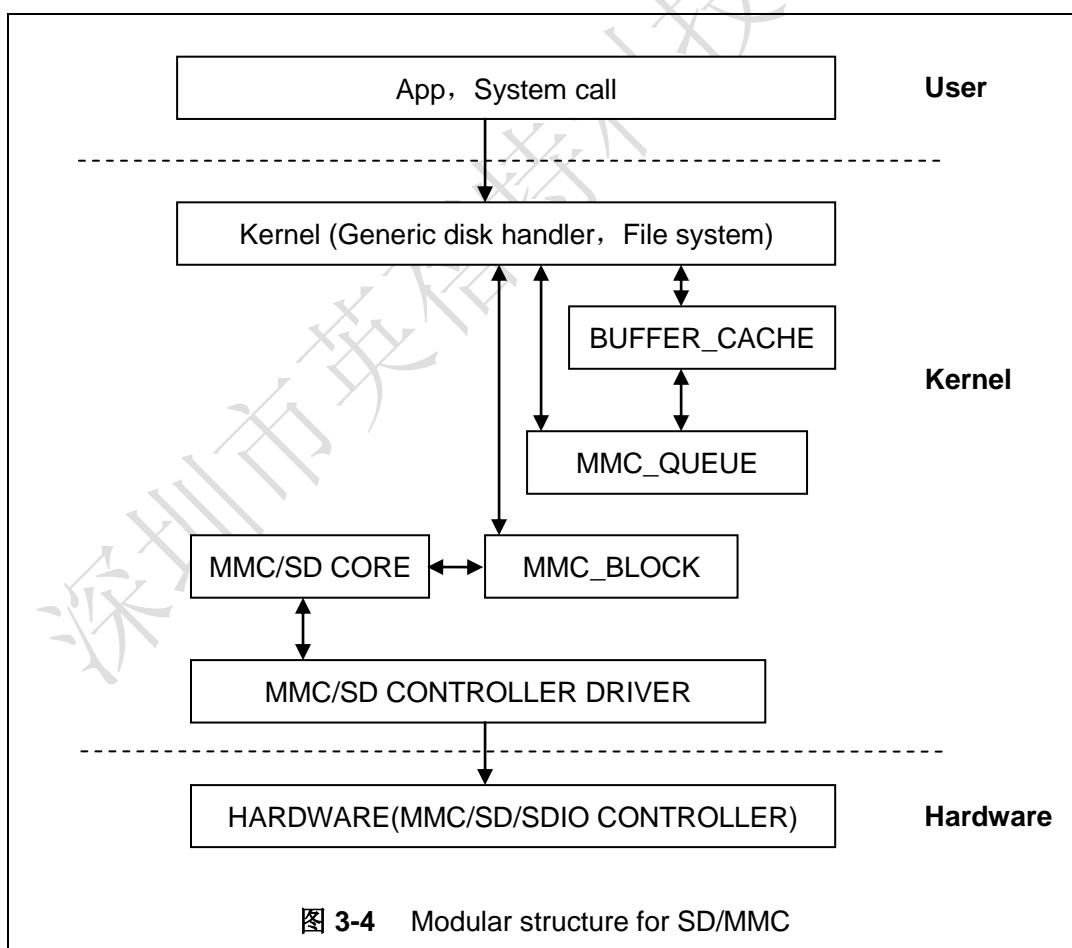
User->System Call->VFS->Block Device Driver->MTD->NAND Flash Driver->NAND Flash。

驱动参考文件：

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

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

### 3.5.2 SD/MMC



Linux 下 SD/MMC 卡驱动主要分为 SD/MMC core、mmc\_block、mmc\_queue、SD/MMC

driver 四大部分：

- 1) SD/MMC core 实现 SD/MMC 卡操作中与结构无关的核心代码。
- 2) mmc\_block 实现 SD/MMC 卡作为块设备使用时的驱动结构。
- 3) mmc\_queue 实现请求队列的管理。
- 4) SD/MMC driver 实现具体的控制器驱动。

驱动参考文件：

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

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

### 3.5.3 LCDC

AM335x 下的 LCD 控制器（LCDC）是 OMAP-L138 SoC 中 LCDC 的更新版本，与 OMAP-L138 比较具有如下特点：

- 1) 中断配置和状态寄存器是不同的
- 2) 分辨率提升至 2048\*2048
- 3) 每像素 24 位有源 TFT 光栅配置

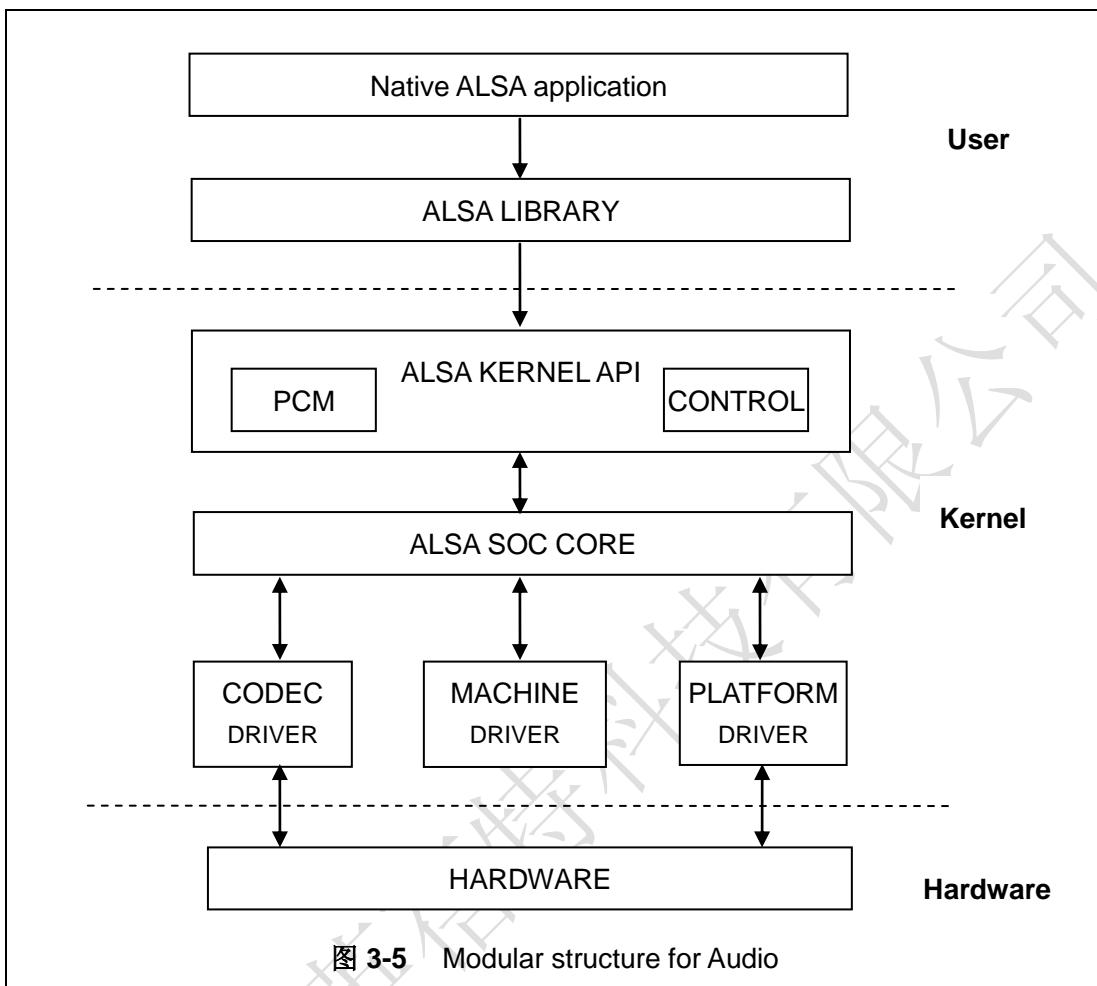
因此 da8xx-fb LCD 驱动可用于 LCD\_VERSION2 代码下的改进。通过读 PID 寄存器可以检测到 LCDC 版本的更新。

驱动参考文件：

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

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

### 3.5.4 Audio in/out



ASoC 嵌入式音频系统基本分割以下三部分：

- 1) 编解码器驱动：编解码器驱动是一个平台无关，包括 audio controls, audio interface capabilities, codec dapm definition and codec IO functions;
- 2) 平台驱动：平台驱动包括平台相关的 audio dma engine and audio interface drivers (e.g. I2S, AC97, PCM);
- 3) Machine 驱动：Machine 驱动管理任何 machine 相关的 controls and audio events i.e. turning on an amp at start of playback;

驱动参考文件：

[Linux-3.2.0-psp04.06.00.08.sdk/sound/soc/](#)

[Linux-3.2.0-psp04.06.00.08.sdk/sound/soc/davinci/davinci-evm.c](#)

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

## 3.6 驱动开发

### 3.6.1 GPIO\_keys 驱动

#### 1) 设备定义

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

配置 gpio0.20 为"menu"键，返回键值"KEY\_F1"，低电平触发； gpio2.1 为"back"键，返回键值"KEY\_ESC"，低电平触发。

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

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

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

```
    },  
};
```

## 2) GPIO pinmux 配置

配置 GPIO0.20 和 GPIO2.1 为 MODE7(gpio 模式)、AM33XX\_PIN\_INPUT(配置输入)

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

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

## 3) 驱动设计

Linux-3.2.0-psp04.06.00.08.sdk/drivers/input/keyboard/gpio\_keys.c

### a) 调用 platform\_driver\_register 注册 gpio\_keys 驱动

```
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) 调用 `input_register_device` 注册 input 驱动

```
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) 申请 gpio，配置 gpio 为输入，注册 gpio 中断

```
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) 中断处理

按键被按下，产生中断，汇报键值：

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

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

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

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

### 3.6.2 GPIO\_leds 驱动

#### 1) 设备定义

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

配置 GPIO1.30 为“sys\_led”（系统心跳灯）、GPIO1.31 为“user\_led”，均为高电平有效。

```
static struct gpio_led gpio_leds[] = {
{
    .name          = "sys_led",
    .default_trigger = "heartbeat",
    .gpio          = GPIO_TO_PIN(1, 30),
},
{
    .name          = "user_led",
    .gpio          = GPIO_TO_PIN(1, 31),
},
};

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

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

```
    },  
};
```

## 2) GPIO pinmux 配置

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

配置 GPIO1.30 和 GPIO1.31 为 MODE7(gpio 模式)、AM33XX\_PIN\_OUTPUT(配置输出)

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

## 3) 驱动设计

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

### a) 调用 platform\_driver\_register 注册 gpio\_leds 驱动

```
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) 申请 gpio，调用 led\_classdev\_register 注册 led\_classdev 驱动

```
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) 用户通过访问/sys/class/leds/xxx/brightness 文件，调用 gpio\_led\_set 函数，控制 led 灯的状态

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

## 3.7 系统更新

SBC8600B 支持从 TF 卡和 NAND Flash 启动系统，下面将详细介绍两种不同的系统映像更新方式。

### 3.7.1 TF 卡系统映像更新

#### 1) TF 卡格式化

请使用 HP USB Disk Storage Format Tool 2.0.6 格式 TF 卡。

软件下载链接：<http://www.embedinfo.com/english/download/SP27213.exe>

- a) 把 MMC/SD 卡插入 PC 下读卡器中
- b) 打开 HP USB Disk Storage Format Tool，出现类似提示如下：

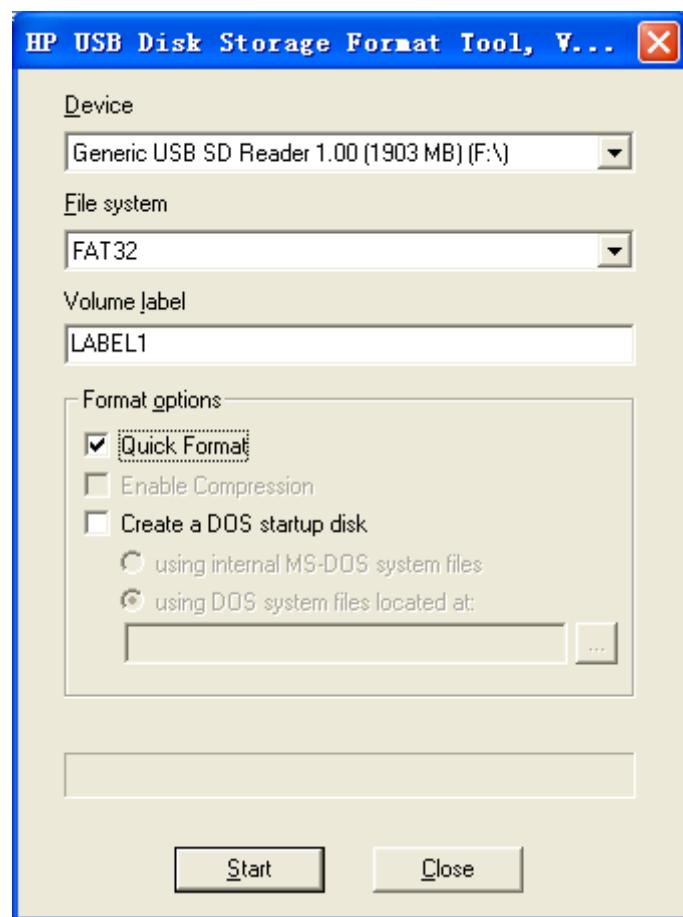


图 3-6

- c) 选择“FAT32”系统格式
- d) 点击“Start”
- e) 等待格式化完成，点击“OK”

**注意：**

- HP USB Disk Storage Format Tool 会将清除 TF 卡的分区。
- 当系统不能从 TF 卡启动时，建议再格式化一次试一下

**2) 映像更新**

将 linux/image 目录下的所有文件拷贝到 TF 卡上，将 TF 卡接入板子，上电启动，串口信息显示如下：

**注意：**

- 默认 4.3 寸 LCD 显示。如想使用其他的显示设备，在启动时进入 **u-boot** 设置显示方式，再输入 **boot** 继续启动即可。显示方式的设置方法请参考 **【3.8.1 显示方式选择】**。
- SBC8600B 默认优先从 **nand flash** 启动，如果 **nand flash** 里面已经有映像，则需要用跳线帽短接板上的 **JP5** 引脚，使 SBC8600B 从 **TF 卡** 启动。

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

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

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

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

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

3224184 bytes read
reading ramdisk.gz

12514633 bytes read
## Booting kernel from Legacy Image at 80007fc0 ...
Image Name:  Linux-3.2.0
Image Type:  ARM Linux Kernel Image (uncompressed)
Data Size:   3224120 Bytes = 3.1 MiB
```

```
Load Address: 80008000
Entry Point: 80008000
Verifying Checksum ... OK
XIP Kernel Image ... OK
OK

Starting kernel ...

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

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

Tue Jan 27 08:47:00 UTC 2009
INIT: Entering runlevel: 5
Starting syslogd/klogd: done

.....
| | .-
```

```
| | |-----|-----| | -----|-----| | | | | | | | | | |
| | | | _ | --|'-| .| -| | | |
| | | | | | |---|| -| | | | ' |||||
'-----'-----|----"---"-|-----'----'
               |
               '|'

The Angstrom Distribution SBC8600 ttyO0

Angstrom 2008.1-test-20090127 SBC8600 ttyO0

SBC8600 login: (输入“root”即可)
```

超级终端显示上述信息，则代表已经成功从 TF 卡启动 Linux 系统。

### 3.7.2 NAND Flash 更新/恢复

Nand 启动映像的更新需要借助于 u-boot 来完成。不管 NAND Flash 是否有数据，都可以利用 u-boot 对 NAND Flash 更新映像。

1) 准备

- a) 用 HP USB Disk Storage Format Tool 2.0.6 将 TF 卡格式化为 FAT 或 FAT32 文件系统
- b) 将光盘里的 **MLO**, **u-boot.img**, **ulimage**, **ubi.img** 映像文件拷贝到 TF 卡中。

2) 更新

- a) 将带有系统映象的 TF 卡插入开发板，用跳线帽短接 **JP5** 引脚，上电启动，按照下面提示读秒处，按电脑端键盘任意键进入 u-boot。

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

```
U-Boot SPL 2011.09-svn55 (Nov 20 2012 - 10:37:42)
I2C: ready
DRAM: 512 MiB
```

```
WARNING: Caches not enabled  
Did not find a recognized configuration, assuming General purpose EVM in Profile 0 with  
Daughter board  
NAND: HW ECC Hamming Code selected  
512 MiB  
MMC: OMAP SD/MMC: 0  
*** Warning - bad CRC, using default environment  
  
Net:   cpsw  
Hit any key to stop autoboot:  0 (在这里按任意键进入 u-boot 命令行)
```

b) 进入 u-boot 命令行后, PC 键盘输入 “**run updatesys**”, 开始自动更新系统

```
SBC8600# run updatesys  
  
NAND erase.chip: device 0 whole chip  
Erasing at 0x7fe0000 -- 100% complete.  
OK  
reading MLO  
  
36079 bytes read  
HW ECC BCH8 Selected  
  
NAND write: device 0 offset 0x0, size 0x8cef  
36079 bytes written: OK  
reading u-boot.img  
  
234896 bytes read  
HW ECC BCH8 Selected  
  
NAND write: device 0 offset 0x80000, size 0x39590  
234896 bytes written: OK  
reading ulimage  
  
3224184 bytes read  
HW ECC BCH8 Selected  
  
NAND write: device 0 offset 0x280000, size 0x313278  
3224184 bytes written: OK  
reading ubi.img  
  
14811136 bytes read  
SW ECC selected
```

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

此时板上的 LED 灯会闪烁，代表已经更新完成，拔出 TF 卡和 **JP5** 上的跳线帽，

重启开发板，即可从 Nand Flash 启动 Linux 系统。

### 3) Uboot 参数设置

映像默认为 4.3 寸屏显示，如想使用其他显示设备，用户必须根据所使用的显示设备修改 UBOOT 参数，具体方法可参考【3.8.1 显示方式选择】。

## 3.8 使用说明

### 3.8.1 显示方式选择

系统支持多种显示输出模式，用户可通过配置启动参数的方法选择不同的显示输出模式，操作方法如下：

上电启动，按照下面提示读秒处，按电脑端键盘任意键进入 u-boot

```
U-Boot SPL 2011.09-svn55 (Nov 20 2012 - 10:37:42)  
Texas Instruments Revision detection unimplemented  
Booting from MMC...  
OMAP SD/MMC: 0  
reading u-boot.img  
reading u-boot.img  
  
U-Boot SPL 2011.09-svn55 (Nov 20 2012 - 10:37:42)  
I2C: ready  
DRAM: 512 MiB  
WARNING: Caches not enabled  
Did not find a recognized configuration, assuming General purpose EVM in Profile 0 with  
Daughter board  
NAND: HW ECC Hamming Code selected  
512 MiB  
MMC: OMAP SD/MMC: 0  
*** Warning - bad CRC, using default environment
```

```
Net: cpsw  
Hit any key to stop autoboot: 0 (在这里按任意键进入 u-boot 命令行)
```

### 3.8.1.1 使用 4.3”LCD 显示

u-boot 下修改启动参数如下：

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

### 3.8.1.2 使用 7”LCD 显示

u-boot 下修改启动参数如下：

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

### 3.8.1.3 使用 VGA 显示

u-boot 下修改启动参数如下：

```
SBC8600# setenv dispmode VGA  
SBC8600# saveenv
```

### 3.8.1.4 使用 LVDS 显示

u-boot 下修改启动参数如下：

```
SBC8600# setenv dispmode LVDS  
SBC8600# saveenv
```

## 3.8.2 测试

### 3.8.2.1 LED 测试

主板上的 D35 为系统心跳灯、D36 为用户 LED 灯。

以下操作在超级终端中进行：

1) 控制系统心跳灯：

```
root@SBC8600:~# echo 1 > /sys/class/leds/sys_led/brightness  
root@SBC8600:~# echo 0 > /sys/class/leds/sys_led/brightness
```

## 2) 控制用户 LED 灯:

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

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

LED 灯会随着用户的操作进行亮灭。

### 3.8.2.2 KEYPAD 测试

板子有两个用户键盘 BACK 和 MENU，用户可执行以下命令进行测试：

```
root@SBC8600:~# evtest /dev/input/event1
Input driver verevdev: (EVIOCGBIT): Suspicious buffer size 511
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys"
Supported events:
    Event type 0 (Sync)
    Event type 1 (Key)
        Event code 1 (Esc)
        Event code 59 (F1)
Testing ... (interrupt to exit)
Event: time 1233046135.256046, type 1 (Key), code 1 (Esc), value 1
Event: time 1233046135.256053, ----- Report Sync -----
Event: time 1233046135.426967, type 1 (Key), code 1 (Esc), value 0
Event: time 1233046135.426970, ----- Report Sync -----
Event: time 1233046136.373255, type 1 (Key), code 59 (F1), value 1
Event: time 1233046136.373260, ----- Report Sync -----
Event: time 1233046136.548841, type 1 (Key), code 59 (F1), value 0
Event: time 1233046136.548844, ----- Report Sync -----
```

#### 注意：

按 CONTROL+C 退出测试，后续测试同理。

### 3.8.2.3 触摸屏测试

此测试要求 Linux 从 NAND Flash 启动：

#### 1) 输入以下指令执行触摸屏校准程序：

```
root@SBC8600: # ts_calibrate
```

按照屏幕上提示，点击“+”图标 5 次完成校准。

- 2) 校准完成后，输入以下指令进行触摸屏测试：

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

按照屏幕提示，可选择画点、画线测试。

### 3.8.2.4 背光测试

背光的亮度设置范围为（0—100），100 表示亮度最高。0 表示关闭背光亮度，进入系统后在终端下输入如下命令进行背光测试。

- 1) 查看背光的亮度默认值。

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

- 2) 设置背光亮度为 0

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

此时背光被关闭，屏幕是黑的。

- 3) 把屏幕亮度设置为 100

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

此时屏幕变亮。

### 3.8.2.5 RTC 测试

开发板带硬件时钟，用于保存并恢复系统时间，可参考如下方法进行测试：

- 1) 设置系统时间为 2012 年 3 月 22 日晚上 8 时正

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

- 2) 把系统时钟写入 RTC

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

- 3) 读取 RTC

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

可以看到，硬件时钟 RTC 被设置成 2012 年 3 月 22 日，系统时钟被保存到硬件时钟里。

#### 4) 重启系统，输入以下命令恢复系统时钟

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

可以看到，系统时间被恢复为硬件时间。

#### 注意：

- 书 RTC 存在断电后时间停止的 BUG，此为 CPU 本身存在的问题，详情请看 TI 官网
- 书 开发板自身未带电池（型号 CR1220），用户需自行购买。

### 3.8.2.6 TF 卡测试

#### 1) 接入 TF 卡后，系统会自动将 TF 卡的文件系统挂载到/media 目录下：

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

#### 2) 输入下述指令后，即可看到 TF 卡里面的内容：

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

#### 3) 手动卸载 TF 卡。

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

#### 4) 手动挂载 TF 卡。

```
root@SBC8600:~# mount -t vfat /dev/mmcblk0p1 /mnt/cf  
root@SBC8600:~# df -h  
Filesystem      Size   Used Available Use% Mounted on
```

```
rootfs          31.0M   19.7M   11.3M  64% /
...
tmpfs          250.6M    3.0M   247.6M  1% /media/ram
/dev/mmcblk0p1     1.8G   101.8M   1.8G  5% /media/cf
root@SBC8600:~# ls /media/cf
u-boot.img        mlo           ulimage
ramdisk.gz        ubi.img
```

**注意：**

书 SBC8600B 目前存在自动挂载 TF 卡，写速度慢的问题，此问题可以通过手动卸载，然后再手动挂载来解决。

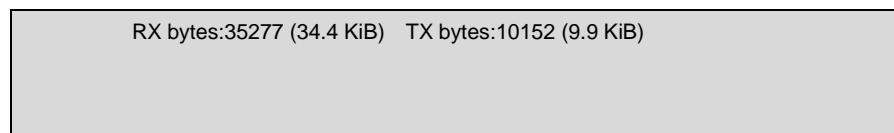
### 3.8.2.7 USB DEVICE 测试

USB DEVICE 测试主要是使用连接线连接开发板的 miniUSB 接口与电脑端的 USB 接口，对于电脑端，开发板被识别成一个网络设备，实现两端 ping 通讯。

- 1) 系统起来后，使用 USB mini B to USB A 转接线连接开发板（CON2 接口）与电脑端，其中 USB mini B 接口连接开发板，USB A 接口连接电脑端。此时电脑需要安装 Linux USB Ethernet 驱动，详细的安装方法请参考[附录三](#)。
- 2) 配置 usb 虚拟网卡的 IP 地址

```
root@SBC8600:~# ifconfig usb0 192.168.1.115
root@SBC8600:~# ifconfig
lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
              UP LOOPBACK RUNNING MTU:16436 Metric:1
              RX packets:26 errors:0 dropped:0 overruns:0 frame:0
              TX packets:26 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:0
              RX bytes:2316 (2.2 KiB) TX bytes:2316 (2.2 KiB)

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



- 3) 配置好开发板，点击我的电脑-网上邻居-查看网络连接，PC 端会增加一个虚拟网卡的网络连接。
- 4) 在虚拟网卡的网络连接图标上单击电脑端鼠标右键，选择“属性”，在弹出的属性窗口，双击“Internet 协议 (TCP/IP)”进入“Internet 协议 (TCP/IP) 属性”窗口，配置虚拟网卡的 IP 地址：

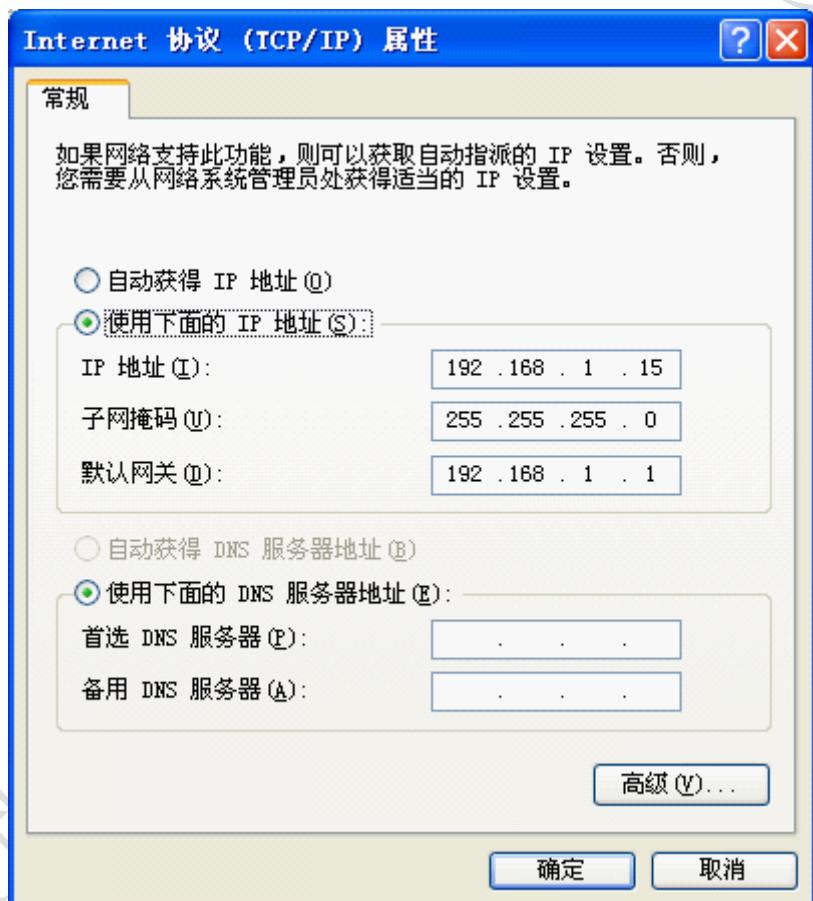


图 3-7

- 5) 在超级终端中使用 ping 命令测试开发板是否设置成功：

```
root@SBC8600:~# ping 192.168.1.15
PING 192.168.1.15 (192.168.1.15): 56 data bytes
64 bytes from 192.168.1.15: seq=0 ttl=128 time=0.885 ms
64 bytes from 192.168.1.15: seq=1 ttl=128 time=0.550 ms
```

- 6) 出现上述串口信息，代表测试成功。

**注意：**

书 OTG 虚拟网卡的 IP 地址不能与开发板以太网口的 IP 地址在同一个网段内。

### 3.8.2.8 USB HOST 测试

- 1) 进入 Linux 系统，将 U 盘连接到板上的 USB-HUB (CON3) 接口，系统会自动将 U 盘的文件系统挂载到 /media/sda1 目录下：

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

- 2) 手动卸载 U 盘：

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

- 3) 查看 U 盘是否已经卸载，当输入 df 命令后，发现没有 /media/sda1/ 目录。

```
root@SBC8600:~# df
Filesystem      1k-blocks   Used Available Use% Mounted on
rootfs          31729    20185    11544  64% /
/dev/root       31729    20185    11544  64% /
none            256624     684    255940  0% /dev
/dev/mmcblk0p1  1939712  104316  1835396  5% /media/mmcblk0p1
tmpfs           256624      20    256604  0% /var/volatile
tmpfs           256624      0    256624  0% /dev/shm
tmpfs           256624    3104    253520  1% /media/ram
```

- 4) 手动挂载 U 盘。

```
root@SBC8600:~# mount -t vfat /dev/sda1 /mnt/card/
root@SBC8600:~# df -h
```

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

**注意：**

书 SBC8600B 目前存在自动挂载 U 盘，写速度慢的问题，此问题可以通过手动卸载，然后再手动挂载来解决。

### 3.8.2.9 AUDIO 测试

板上带音频输入、输出接口，支持录放音。文件系统内带 alsound 工具，用户可使用如下命令进行测试：

1) 录音测试：

插上麦克风，在超级终端输入以下命令即可进行录音

```
root@SBC8600:~# arecord -t wav -c 1 -r 44100 -f S16_LE -v k
Recording WAVE 'k' : Signed 16 bit Little Endian, Rate 44100 Hz, Mono
Plug PCM: Route conversion PCM (sformat=S16_LE)
Transformation table:
 0 <- 0*0.5 + 1*0.5
Its setup is:
  stream      : CAPTURE
  access       : RW_INTERLEAVED
  format       : S16_LE
  subformat    : STD
  channels     : 1
  rate         : 44100
  exact rate   : 44100 (44100/1)
  msbits       : 16
  buffer_size  : 32768
  period_size  : 2048
  period_time  : 46439
  tstamp_mode  : NONE
```

```
period_step      : 1
avail_min        : 2048
period_event    : 0
start_threshold  : 1
stop_threshold   : 32768
silence_threshold: 0
silence_size     : 0
boundary         : 1073741824
.....
```

## 2) 放音测试:

插上耳机，执行以下操作，即可听刚才的录音内容。

```
root@SBC8600:~# aplay -t wav -c 2 -r 44100 -f S16_LE -v k
Playing WAVE 'k' : Signed 16 bit Little Endian, Rate 44100 Hz, Mono
Plug PCM: Route conversion PCM (sformat=S16_LE)
Transformation table:
  0 <- 0
  1 <- 0
Its setup is:
  stream       : PLAYBACK
  access        : RW_INTERLEAVED
  format        : S16_LE
  subformat     : STD
  channels      : 1
  rate          : 44100
  exact rate    : 44100 (44100/1)
  msbits        : 16
  buffer_size   : 32768
  period_size   : 2048
  period_time   : 46439
  tstamp_mode   : NONE
  period_step   : 1
  avail_min     : 2048
  period_event  : 0
  start_threshold : 32768
  stop_threshold  : 32768
  silence_threshold: 0
  silence_size   : 0
  boundary       : 1073741824
.....
```

### 3.8.2.10 网络测试

SBC8600B 板载两个以太网口，它们分别是 NET1 (J1) 和 NET2 (J2)，它们对应的设备节点分别为 eth0 和 eth1。用网线连接以太网口和路由器，使用以下命令进行测试：

**注意：**

注意：SBC8600B 两个网口的 IP 地址必须设置为不同网段，否则不能正常使用，两个网卡设置不同网段的 IP 地址后能同时使用。

```
[root@SBC8600/]# ifconfig eth0 192.192.192.200
[root@SBC8600/]# ifconfig
eth0      Link encap:Ethernet HWaddr D4:94:A1:8D:EB:25
          inet addr:192.192.192.200 Bcast:192.192.192.255 Mask:255.255.255.0
                  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                  RX packets:137 errors:0 dropped:4 overruns:0 frame:0
                  TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                  RX bytes:13792 (13.4 KiB) TX bytes:0 (0.0 B)
                  Interrupt:40

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

[root@SBC8600/]# ping 192.192.192.170
PING 192.192.192.170 (192.192.192.170): 56 data bytes
64 bytes from 192.192.192.170: seq=0 ttl=128 time=4.486 ms
64 bytes from 192.192.192.170: seq=1 ttl=128 time=0.336 ms
[root@SBC8600/]# ifconfig eth1 192.168.168.116
[root@SBC8600/]# ifconfig
eth1      Link encap:Ethernet HWaddr 00:17:EA:96:34:D5
          inet addr:192.168.168.116 Bcast:192.168.168.255 Mask:255.255.255.0
                  UP BROADCAST MULTICAST MTU:1500 Metric:1
                  RX packets:0 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
```

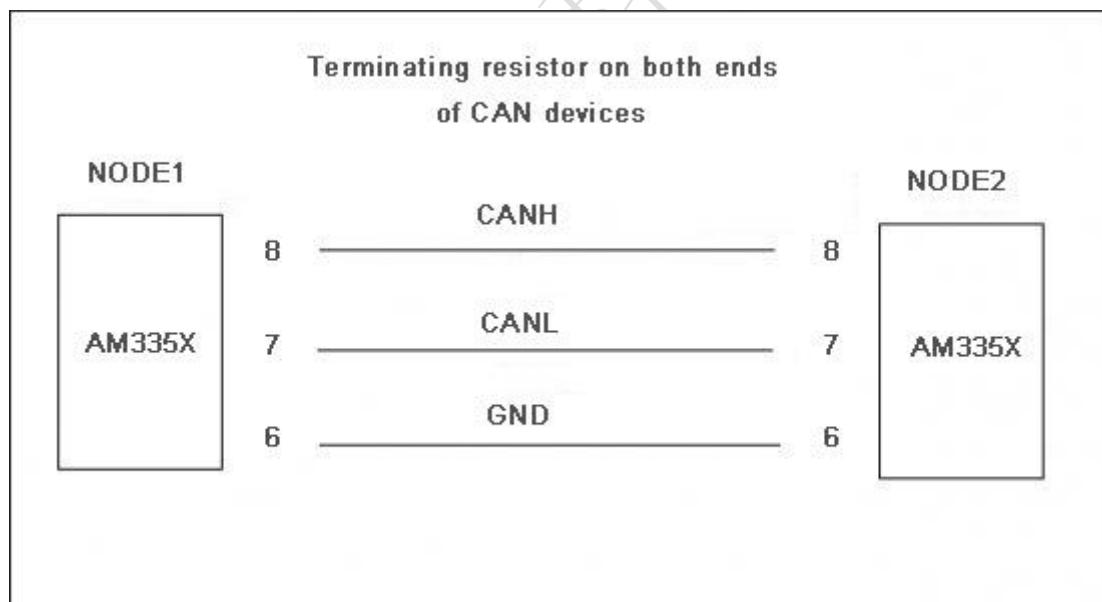
```
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
Lo      Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
      UP LOOPBACK RUNNING MTU:16436 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

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

出现上述串口信息，代表测试成功。

### 3.8.2.11 CAN 测试

SBC8600B 可以作为一个 CAN 设备使用。按照下图所示连接原理，并参考原理图找到对应的引脚，用连接线连接 SBC8600B 的 CAN 接口和另一个 CAN 设备。



测试方法如下：

- 1) 将 SBC8600B 和另一个 CAN 设备的通信波特率都设置为 125KBPS，并使能 CAN 设备。

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

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

- 2) 在 SBC8600B 和另一个 CAN 设备两端分别执行发送和接收数据的命令，执行以下命令发送数据包。

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

**注意：**

- 书 该命令每次只发送一次数据，若再次发送数据，需重新输入该命令。
- 书 要确保另一端处于接收状态。这样接收端才会打印发送的信息。

- 3) 接收数据包。

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

执行命令后，当收到数据时会在终端下打印接收的数据。

- 4) 关闭设备。

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

用户可以根据以上命令进行相互收发测试，还可以设置不同的波特率进行通信，在设置不同波特率之前必须先关闭设备，可设置的波特率有：

25KBPS (250000)

50KBPS (50000)

125KBPS (125000)

500KBPS (500000)

650KBPS (650000)

1MKBPS (1000000)

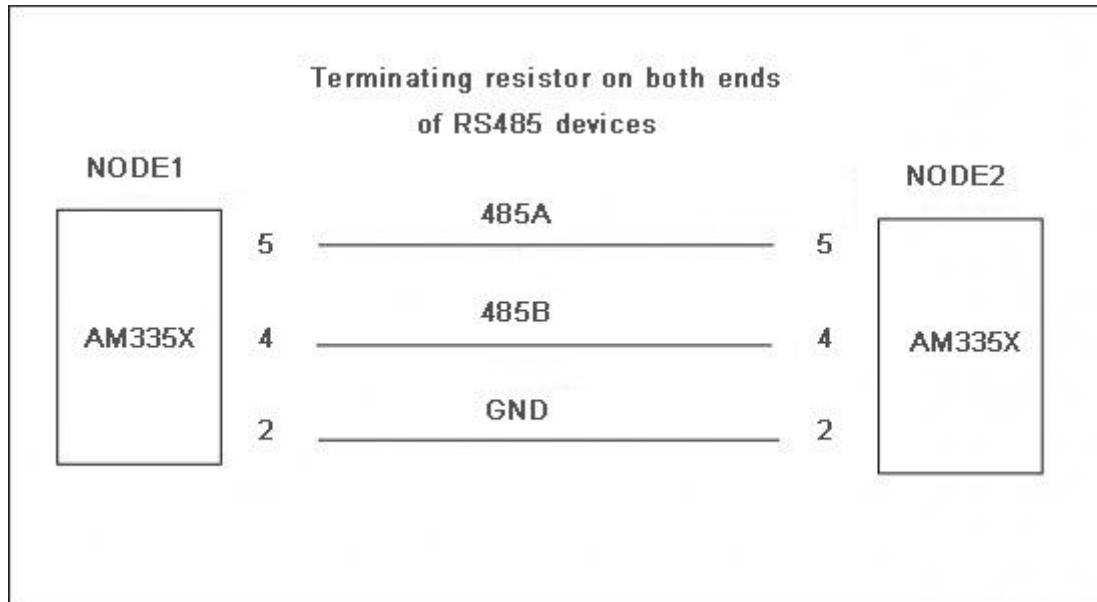
以上的波特率均能正常通信，还有其它波特率可以设置，用户可以自己尝试，看能否通信。

**注意：**

- 书 两个开发板通过 CAN 通信必须设置成相同的波特率。

### 3.8.2.12 RS485 测试

SBC8600B 可以作为一个 RS485 使用。按照下图所示连接原理，并参考原理图找到对应的引脚，用连接线连接 SBC8600B 的 RS485 接口和另一个 RS485 设备。



485 通信只支持半双工通信，即通信一端同一时间只能发送或者只能接收信息。拷贝 linux\example\uart\_test 下的 uart\_test 到 TF 卡中，将 TF 卡插入 SBC8600B 的 TF 卡座子，在终端下运行如下命令：

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

### 3.8.2.13 串口测试

短接板上 J5 接口的 RX3V3 和 TX3V3 管脚，拷贝 linux\example\uart\_test 下的 uart\_test 文件到 TF 卡中，将 TF 卡插入 SBC8600B 的 TF 卡座子，在 linux 终端输入如下命令：

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

打印如下信息表示测试成功。

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

SBC8600B 上扩展串口 3、扩展串口 4 和扩展串口 5（J6 和 J7）的测试方法同上。

### 3.8.2.14 蜂鸣器测试

打开蜂鸣器。

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

关闭蜂鸣器。

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

### 3.8.2.15 CDMA8000-U 模块

CDMA8000-U 模块属于可选配件，用户可以根据实际的需求选择购买。

模块资料下载：

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

### 3.8.2.16 WCDMA8000-U 模块

WCDMA8000-U 模块属于可选配件，用户可以根据实际的需求选择购买。

模块资料下载：

<http://www.timli.com/chinese/uploadFile/WCDMA8000.zip>

## 3.8.3 Demo

### 3.8.3.1 Android 系统演示

SBC8600B 提供 Android 系统演示，使用方法如下。

- 1) 拷贝 CD\linux\demo\android\image 目录下所有文件到 TF 卡。
- 2) 将 TF 卡放入开发板，用跳线帽帽短接板上的 **JP5** 引脚，上电启动，超级终端将会显示下述信息：

**注意：**

如客户需要从 TF 卡启动，而不想短接 JP5，可以在 uboot 下输入命令擦除 nand flash 中的映像，重启后将会从 TF 卡启动。

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

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

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

```
NAND: HW ECC Hamming Code selected
512 MiB
MMC: OMAP SD/MMC: 0
*** Warning - bad CRC, using default environment

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

36079 bytes read
HW ECC BCH8 Selected

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

234620 bytes read
HW ECC BCH8 Selected

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

2719416 bytes read
HW ECC BCH8 Selected

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

72744960 bytes read
SW ECC selected

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

- 3) 烧写完成后，板上 led 灯会闪烁提示，请拔掉 TF 卡和跳线帽。
- 4) 重启开发板，即可进入 android 操作系统。

- 5) 映像默认为 4.3 寸屏显示，如想使用其他显示设备，用户必须根据所使用的显示设备修改 UBOOT 参数，具体方法可参考【3.8.1 显示方式选择】。

### 3.8.3.2 TISDK 系统演示

- 1) 将 TF 卡分为两个分区，具体的分区操作步骤参考[附录四](#)。
- 2) 插入光盘，将 TF 卡连接到 PC，然后在 Ubuntu 终端执行如下命令：

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

- 3) 在进行上述操作后，用跳线帽短接开发板上的 **JP5** 引脚。将 TF 卡插入 SBC8600B 卡槽，上电启动，系统启动串口信息如下：(黑体字为需要输入的字符内容)

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

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

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

```
Net: cpsw
Hit any key to stop autoboot: 0
Booting from dv-sdk ...
reading ulimage

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

Arago 2011.09 am335x-evm ttyO0

am335x-evm login: root //输入用户名 root 进入系统
```

- 4) TISDK 文件系统中带有一些预装的应用程序，基于 QT 来实现，完全图形化操作，用户可以轻松的执行里面的演示程序。
- 5) 映像默认为 4.3 寸屏显示，如想使用其他显示设备，用户必须根据所使用的显示设备修改 UBOOT 参数，具体方法可参考【3.8.1 显示方式选择】。

## 3.9 上层开发

本节主要介绍应用程序的开发，并通过 LED 应用程序开发实例来说明应用程序开发的一般流程。

### 1) 编写代码

led\_acc.c 源码，控制开发板上的 led 灯按累加器的方式闪烁。

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

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

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

## 2) 交叉编译

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

## 3) 下载运行

通过 TF 卡或 U 盘或网络下载到开发板系统，进入 led\_acc 文件所在的目录，输入下面命令回车 led\_acc 即在后台运行。

```
./led_acc &
```

# 第 4 章 Windows Embedded Compact 7 操作系统

## 4.1 介绍

此部分主要介绍 Windows Embedded Compact 7 下 SBC8600B 的系统与应用开发，以及光盘中软件资源的提供情况，软件特性，开发环境搭建，还有包括如何编译工程与 BSP（板级支持包）等

## 4.2 软件资源

### BSP (板级支持包)

CD\WINCE700\BSP\SBC8600.rar

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

CD\WINCE700\BSP\3rdParty.rar

CD\WINCE700\BSP\PowerVR.rar

### Windows Embedded Compact 7 sample project

CD\WINCE700\project\SBC8600

### 应用范例

CD\WINCE700\app\

### 预编译映像

CD\WINCE700\Image\

MLO

First bootloader for TF card boot

xldrnnand.nbo

First bootloader for NAND flash boot

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

## 4.3 特性

BSP 的资源提供情况:

表21

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

	Backlight driver	Source
	Battery driver	Source
	PRU driver	Source
SDK	powerVR DDK & SDK	Binary & Source

## 4.4 系统开发

### 4.4.1 集成开发环境安装

请按照下面步骤将集成开发环境安装到 windows XP:

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

#### 注意:

默认光盘没有提供 Windows Embedded Compact 7 集成开发环境, 请自行参考下载:  
<http://www.microsoft.com/download/en/default.aspx>。

### 4.4.2 提取 BSP 及样例工程文件到集成开发环境

请按照下面步骤进行:

- 1) 解压[CD\WINCE700\BSP\SBC8600.rar] 到 [C:\WINCE700\PLATFORM] 目录下。
- 2) 解压[CD\WINCE700\BSP\COMMON\_TI\_V1.rar]到[C:\WINCE700\PLATFORM\COMMON\SRC\SOC].
- 3) 解压[CD\WINCE700\BSP\3rdParty.rar] 到[C:\WINCE700].
- 4) 解压[CD\WINCE700\BSP\powerVR.rar] 到[C:\WINCE700\public].
- 5) 将[CD\WINCE700\project\SBC8600] 拷贝到 [C:\WINCE700\OSDesigns] 目录

下。

**注意：**

本文的 Windows Embedded Compact 7 默认安装路径是[C:\WINCE700]。.

### 4.4.3 Sysgen & BSP 编译

下面是 Sysgen 和 BSP 编译的详细步骤

- 1) 打开[C:\WINCE700\OSDesigns\SBC8600]下的工程文件 SBC8600.sln。
- 2) 在 VS2008 窗口选择[Build-> Build Solution]，开始 sysgen 和 build BSP。
- 3) 在 sysgen 阶段和编译阶段成功完成后，  
[C:\WINCE700\OSDesigns\SBC8600\SBC8600\RelDir\SBC8600\_ARMV7\_Release] 目录下会生成映像文件 MLO、EBOOTSD.nb0 和 NK.bin，将 MLO、  
EBOOTSD.nb0 和 NK.bin 文件拷贝到 TF 卡中。
- 4) 接入 TF 卡，用跳线帽短接 **JP5**，上电启动 SBC8600B。

### 4.4.4 驱动介绍

BSP 的所有驱动源码路径：

表22

驱动名称	驱动位置
NLED driver	BSP\SBC8600\SRC\DRIVERS\NLED
GPIO	BSP\SBC8600\SRC\DRIVERS\GPIO BSP\COMMON_TI_V1\COMMON_TI_AMXX\GPIO
I2C	BSP\COMMON_TI_V1\COMMON_TI_AMXX\OAL\OALI2C
SPI	BSP\COMMON_TI_V1\COMMON_TI_AMXX\SPI BSP\SBC8600\SRC\DRIVERS\MCSPI
MCASP driver	BSP\COMMON_TI_V1\COMMON_TI_AMXX\MCASP
Serial port driver	BSP\COMMON_TI_V1\COMMON_TI_AMXX\SERIAL BSP\SBC8600\SRC\DRIVERS\UART
Audio driver	BSP\SBC8600\SRC\DRIVERS\WAVEDEV2

NAND driver	BSP\SB8600\SRC\DRIVERS\BLOCK BSP\COMMON_TI_V1\COMMON_TI_AMXX\BLOCK
Display driver	BSP\COMMON_TI_V1\COMMON_TI_AMXX\DSS_Netra BSP\SB8600\SRC\DRIVERS\DISPLAY
TOUCH driver	BSP\SB8600\SRC\DRIVERS\TOUCH
SD/MMC/SDIO driver	BSP\SB8600\SRC\DRIVERS\SDHC BSP\COMMON_TI_V1\COMMON_TI_AMXX\SDHC BSP\COMMON_TI_V1\COMMON_TI\SDHC
EMAC driver	BSP\COMMON_TI_V1\AM33X\CPSW3Gminiport BSP\SB8600\SRC\DRIVERS\EMAC
USB OTG driver	BSP\SB8600\SRC\DRIVERS\USB BSP\COMMON_TI_V1\AM33X\USB
GPIO keyboard driver	BSP\SB8600\SRC\DRIVERS\KEYPAD
Backlight driver	BSP\SB8600\SRC\DRIVERS\BACKLIGHT
Battery driver	BSP\SB8600\SRC\DRIVERS\BATTERY
PRU driver	BSP\COMMON_TI_V1\AM33X\PRU BSP\SB8600\SRC\DRIVERS\PRU
DMA driver	BSP\SB8600\SRC\DRIVERS\EDMA BSP\COMMON_TI_V1\COMMON_TI_AMXX\EDMA

假若用户想要参考更多的 Windows Embedded Compact 7 驱动开发，请参考 PB7.0 的参考文档，打开的方法如下：（在电脑端操作）

开始->

所有程序->

Microsoft Visual Studio 2008->

Microsoft Visual Studio 2008 Document->

Content(C) ->

Windows Embedded Compact 7->Device Driver.

## 4.5 系统映像更新

SBC8600B 支持 TF 卡与 NAND 启动，本章会针对两种不同的系统更新方式进行介绍。

#### 4.5.1 TF 卡映像更新

##### 5) TF 卡格式化

请使用 HP USB Disk Storage Format Tool 2.0.6 格式 TF 卡。

软件下载链接: <http://www.embedinfo.com/english/download/SP27213.exe>

- a) 把 MMC/SD 卡插入 PC 下读卡器中
- b) 打开 HP USB Disk Storage Format Tool, 出现类似提示如下:

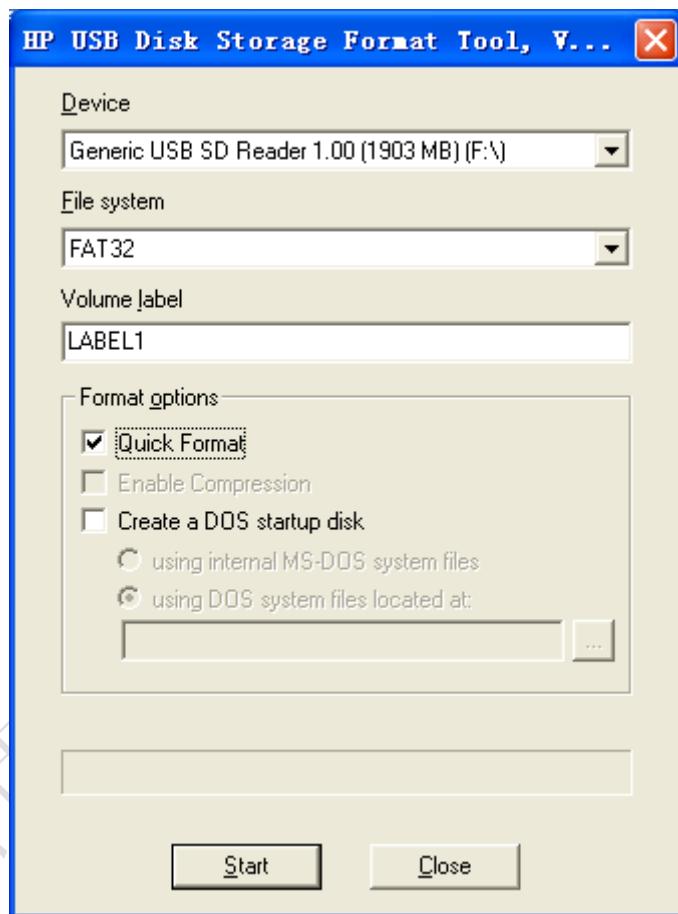


图 4-1

- c) 选择”FAT32“系统格式
- d) 点击”Start“
- e) 等待格式化完成, 点击”OK“

**注意：**

HP USB Disk Storage Format Tool 会将清除 TF 卡的分区。若需要保留分区，请使用电脑系统自带的格式软件。

**6) 拷贝内核映像**

- 将 CD\WINCE700\image 目录下的 **MLO**、**EBOOTSD.nb0** 和 **NK.bin** 映像文件拷贝到 TF 卡中。

**7) 启动系统**

插入 TF 卡，用跳线帽短接 JP5，重启系统，这时系统从 TF 卡启动，在数秒时按空格键进入 eboot 菜单选择启动设备和显示方式，具体步骤如下：

**a) 进入 EBOOT 菜单**

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

jumping to ebootsd image

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

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

TI AM33X
```

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

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

>>> Forcing cold boot (non-persistent registry and other data will be wiped) <<<
e0311800 56e4 -> 0 18 31 e0 e4 56
e0311800 57e4 -> 0 18 31 e0 e4 57
Hit space to enter configuration menu [56] 5... (在此处按空格键进入 EBOOT 菜单)
```

b) 按[2]->[2]选择从 TF 卡启动

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

Selection: 2

```
-----
Select Boot Device
-----

```

```
[1] Internal EMAC
[2] NK from SDCard FILE
[3] NK from NAND
```

[0] Exit and Continue

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

Boot device set to NK from SDCard FILE

c) 按[a]进入“Select Display Resolution”菜单并选择LCD\LVDS输出模式

Main Menu

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

Selection: **a**

Select Display Resolution

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

d) 输入[0]继续启动

Main Menu

- [1] Show Current Settings

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

Selection: 0

mode = 3

LcdPdd\_LCD\_GetMode:3

mode = 3

LcdPdd\_LCD\_Initialize:3

OEMPreDownload: Filename nk.bin

Init HW: controller RST

SDCARD: requested speed 1000000, actual speed 1000000

SDCARD: requested speed 25000000, actual speed 19200000

BL\_IMAGE\_TYPE\_BIN

+OEMMultiBinNotify(0x8feb24d8 -> 1)

Download file information:

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

-----  
Download file type: 1

+OEMIsFlashAddr(0x80002000) g\_eboot.type 1

.....rom\_offset=0x0.

..ImageStart = 0x80002000, ImageLength = 0x3c9e9bc, LaunchAddr = 0x8000b6a0

Completed file(s):

-----  
+OEMIsFlashAddr(0x80002000) g\_eboot.type 1

[0]: Address=0x80002000 Length=0x3c9e9bc Name="" Target=RAM

ROMHDR at Address 80002044h

```
Launch Windows CE image by jumping to 0x8000b6a0...

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

## 4.5.2 NAND Flash 映像更新

### 1) 格式化 TF 卡

请参考 4.5.1 SD 卡系统映像更新中的 SD 卡格式化部分。

### 2) 拷贝映像文件

- 将 CD\WINCE700\image 目录下的 **MLO**、**EBOOTND.nb0**、**NK.bin**、**XLDRNAND.nb0** 和 **EBOOTSD.nb0** 映像文件拷贝至 TF 卡中。

### 3) 更新映像文件

插入 TF 卡，用跳线帽短接 JP5，重新启动系统，这时系统从 TF 卡启动。超级终端输出启动信息，按[SPACE]进入 EBOOT 菜单，按以下步骤更新 NAND Flash 映像：

- 按[8] 进入 Flash 管理菜单。
- 分别按[9]->[4]->[A]、[9]->[3]->[B] 和[9]->[2]->[C] 写 XLDR、EBOOT 和 NK 映像。
- 然后按[0]键回到主菜单，并分别按下[2]、[3]选择从 NAND Flash 启动，按[A] 选择 LCD、LVDS 或 VGA 输出模式，按[7]和[y]保存启动设置。

拔除 TF 卡和 JP5 上的跳线帽，重新启动系统，这时系统将从 NAND Flash 启动。

## 4.6 使用说明

## 4.7 应用开发

本章介绍如何在 SBC8600B 进行 Windows Embedded Compact 7 应用程序开发

### 4.6.1 如何使用 OpenGL ES demo

- 1) 把 vs2008 里的 catalog items view 里把 PowerVR 选上, 如下图所示:

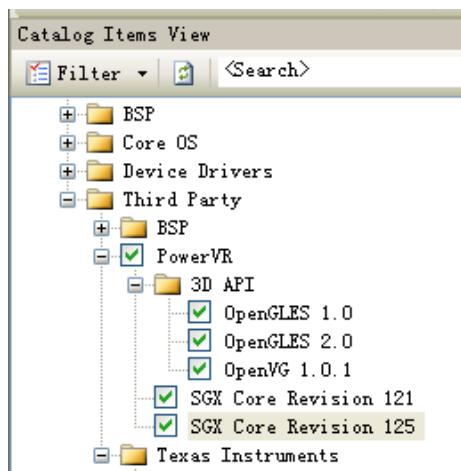


图 4-2

- 2) 在 VS2008 菜单选择[Build-> Build Solution], 在 sysgen 和编译 BSP 完成后, 用新生成的 nk.bin 代替 TF 卡中的 nk.bin。
- 3) 拷贝 C:\WINCE700\PUBLIC\PowerVR\oak\target\Rev125\ARMV4I\retail\\*.exe 到 SBC8600B windows embedded compact 7 系统, 双击 demo 进行测试。

### 4.7.1 应用程序接口与示例

SBC8600B 应用程序开发所用到的 API 均采用微软 Windows Embedded Compact 7 标准应用程序接口定义, SBC8600B 仅在标准 API 基础上扩展了 GPIO 的接口定义, 控制 GPIO PIN 的状态的应用程序请参照 CD\WINCE700\app\GPIOAppDemo。

Windows Embedded Compact 7 标准应用程序接口定义可以查看 MSDN Windows Embedded Compact 7 API 相关帮助文档。

### 4.7.2 GPIO 应用程序接口与示例

GPIO应用程序接口与示例

表23

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

操作示例如下：

### 1) 打开 GPIO 设备

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

### 2) 设置 GPIO 工作模式

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

设置 GPIO 工作模式：

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

读 GPIO 工作模式：

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

"id"为 GPIO 引脚号, "mode"为 GPIO 模式定义, 包括:

表24

模式定义	描述
GPIO_DIR_OUTPUT	Output mode
GPIO_DIR_INPUT	Input mode
GPIO_INT_LOW_HIGH	Rising edge trigger mode
GPIO_INT_HIGH_LOW	Falling edge trigger mode
GPIO_INT_LOW	low level trigger mode
GPIO_INT_HIGH	high level trigger mode
GPIO_DEBOUNCE_ENABLE	Jumping trigger enable

### 3) GPIO 引脚操作

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

输出高电平：

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

输出低电平：

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

读引脚状态

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

"id"为 GPIO 引脚号, "pin"返回引脚状态。

### 4) 其它可选操作

读 GPIO 引脚对应的 IRQ 号：

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

"id"为 GPIO 引脚号, "irq"返回 IRQ 号。

### 5) 关闭 GPIO 设备

```
CloseHandle (hFile);
```

#### 注意：

- 书 GPIO 引脚定义：0~127 MPU Bank0~3 GPIO 引脚。
- 书 GPIO 引脚 0~127 必须在 SBC8600/SRC/inc/bsp\_padcfg.h 文件下被设置为 GPIO。

## 附录

### 附录一 硬件尺寸图

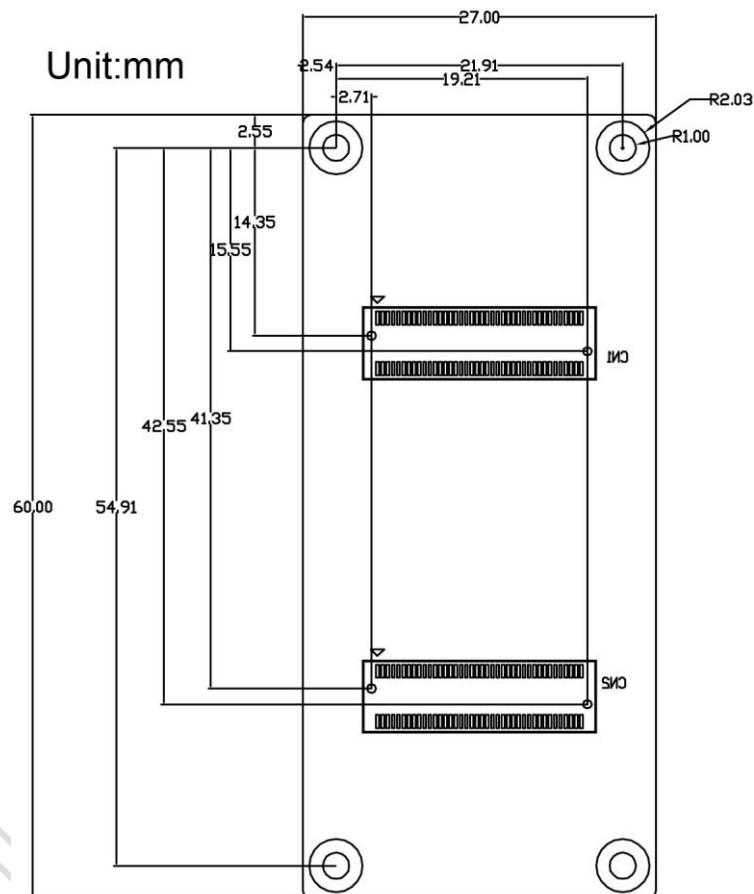


图1 Mini8600B 硬件尺寸图

SBC8600B Bird's-eye View

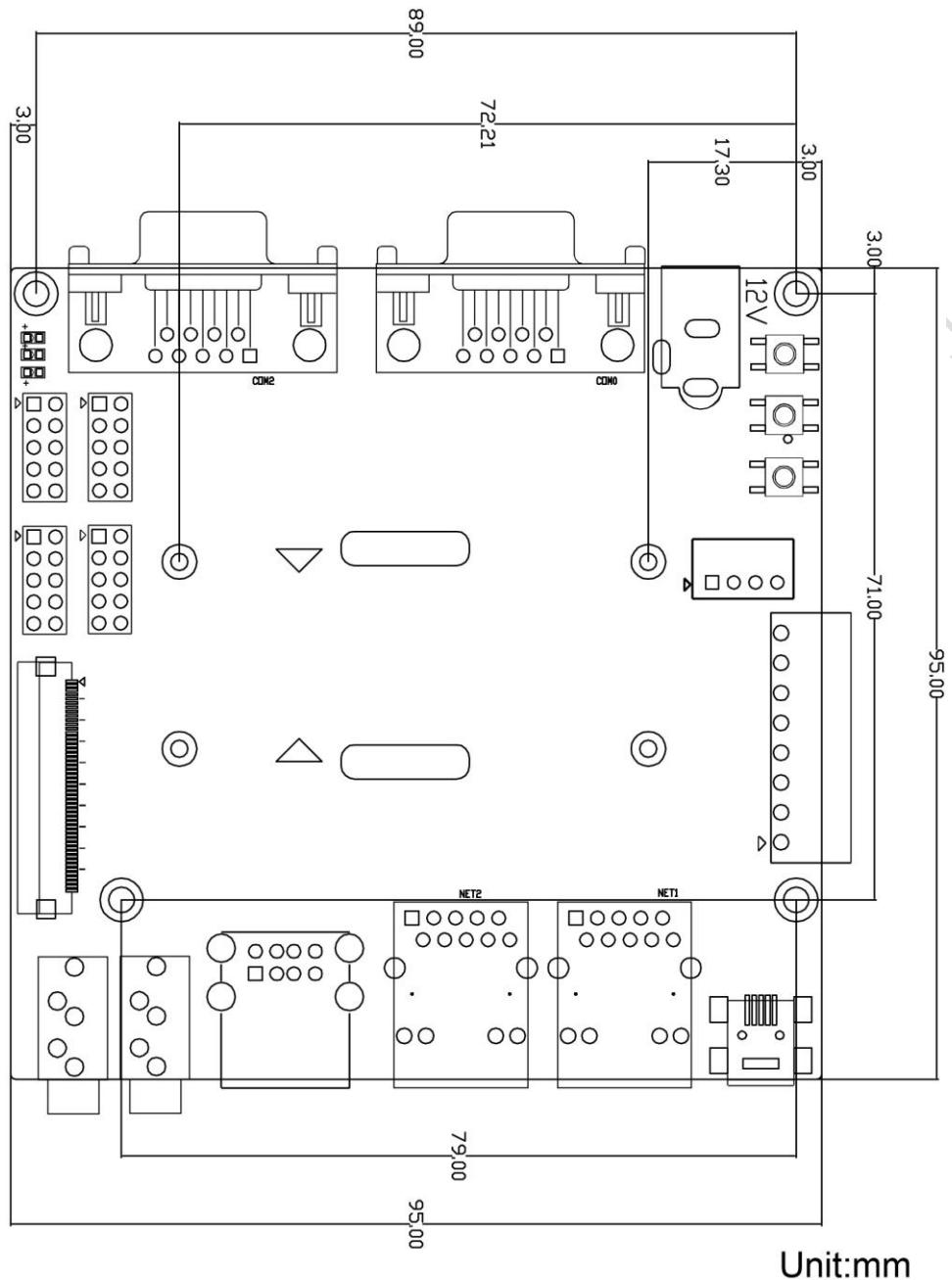


图2 SBC8600B 硬件尺寸图

## 附录二 Ubuntu 安装

我们都知道在开发软件之前需要安装嵌入式开发环境。而产品光盘中提供的开发环境（linux/source 目录下）需要在 Linux 系统下才能运行。如果当前您使用的是 Windows 系统，那么首先需要安装 Linux 操作系统，然后才能在该系统下安装相应的开发环境。我们推荐使用 VirtualBox 虚拟机来在 Windows 中安装 Ubuntu Linux 操作系统。下面我们将依次介绍虚拟机 VirtualBox 和 Ubuntu Linux 系统的安装过程。

### 安装 VirtualBox 虚拟机

您可以访问 <http://www.virtualbox.org/wiki/Downloads> 来下载最新版本的 VirtualBox 虚拟机。在安装 VirtualBox 虚拟机之前，请确保您的 PC 拥有至少 512MB 的内存空间。建议提供 1G 以上的内存空间。

- 1) 安装过程很简单，此处略过。安装后从开始菜单启动 VirtualBox，然后单击程序窗口上方的 **New** 按钮，弹出新建虚拟机窗口；



图3 新建虚拟机窗口

单击 **Next** 按钮开始新建虚拟机。

- 2) 在下方窗口中为新建的虚拟机指定名称和操作系统类型；

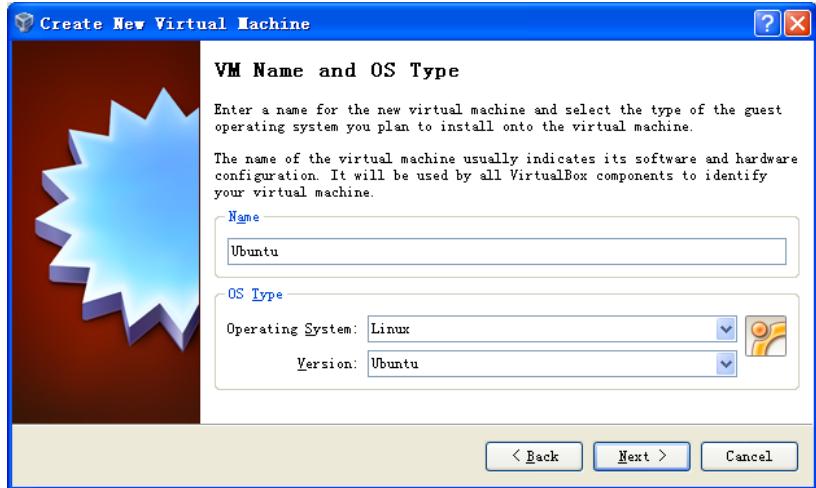


图4 虚拟机名称和操作系统类型

您可以在 **Name** 一栏中输入新建虚拟机的名称，例如 **Ubuntu**。在 **Operating System** 一栏中选择 **Linux**，然后单击 **Next** 按钮。

- 3) 在以下窗口中为虚拟机分配适当大小的内存空间，分配完成后单击 **Next** 按钮；



图5 内存分配窗口

**注意：**

- ❑ 如果您的 PC 内存为 1G 或者更少，请保留默认设置；
- ❑ 如果您的 PC 内存超过 1G，则可以将 1/4 或者更多的内存分配给虚拟机，例如 PC 内存为 2G，则将 512MB 的内存分配给虚拟机。

- 4) 如果这是您第一次使用 VirtualBox, 请在下方窗口中选择 **Create new hard disk** 选项, 然后单击 **Next** 按钮;

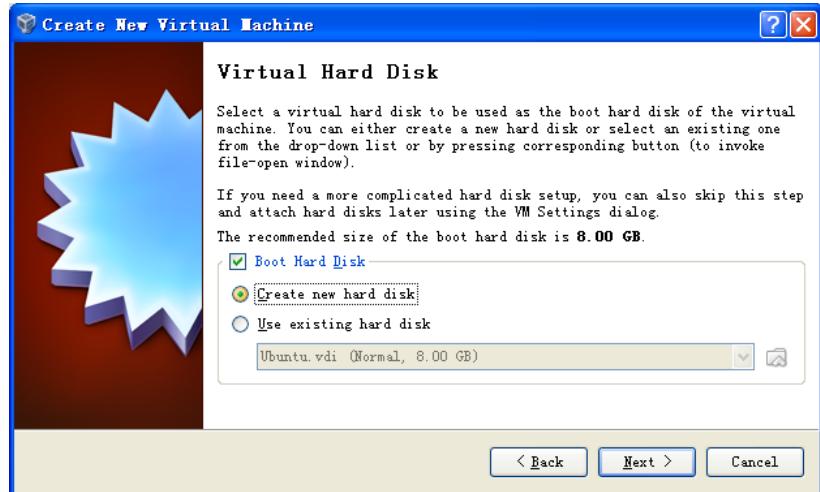


图6 创建新的虚拟硬盘窗口

- 5) 在下方虚拟硬盘创建向导窗口中单击 **Next** 按钮;

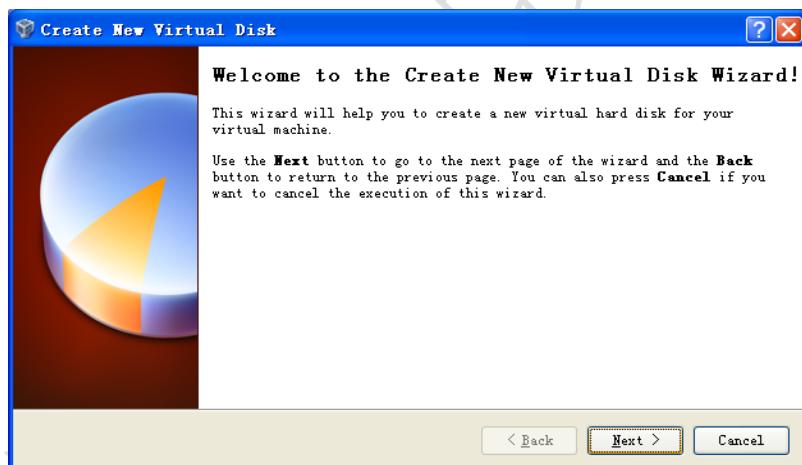


图7 虚拟硬盘创建向导

- 6) 在下方窗口中选择 Fixed-size storage 选项，并单击 Next 按钮；

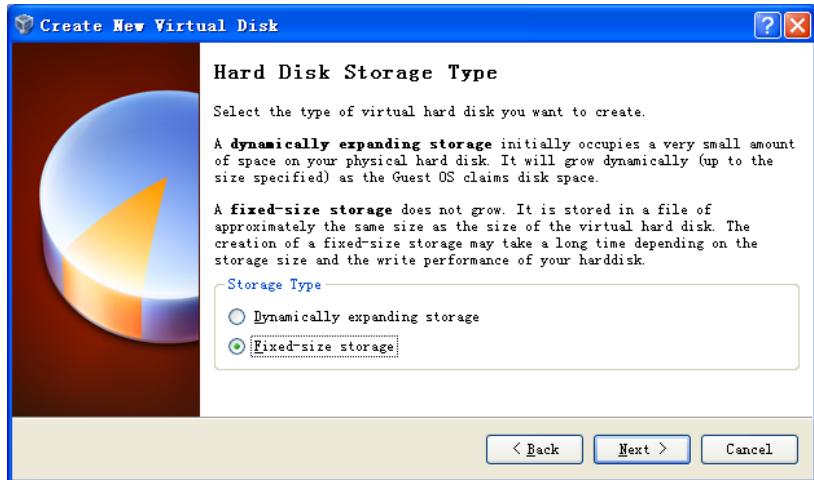


图8 选择第二个选项

- 7) 在下方窗口中指定硬盘数据的存储位置以及默认虚拟硬盘的容量（至少 8G），然后单击 Next 按钮；

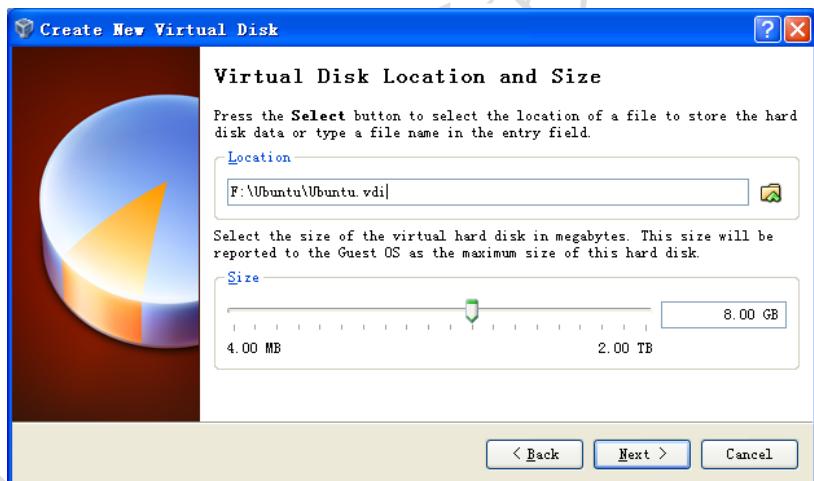


图9 虚拟硬盘设置窗口

- 8) 在下方的虚拟硬盘信息窗口中单击 Finish 按钮;

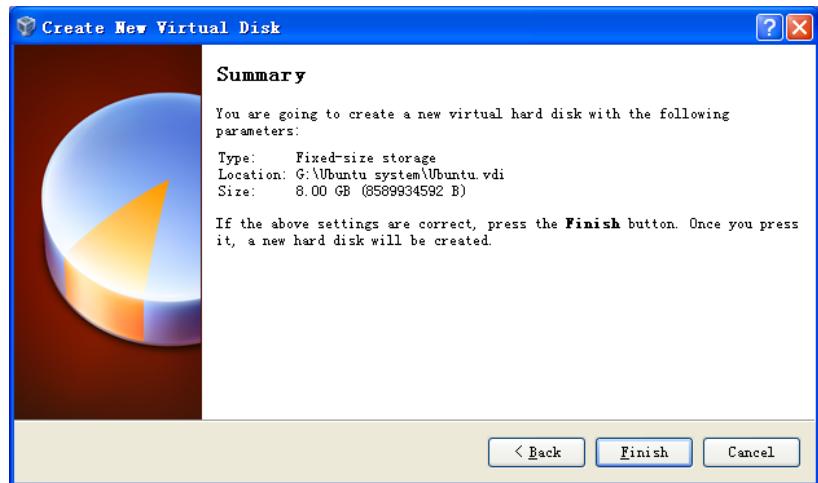


图10 虚拟硬盘信息

- 9) PC 开始创建虚拟硬盘驱动器;

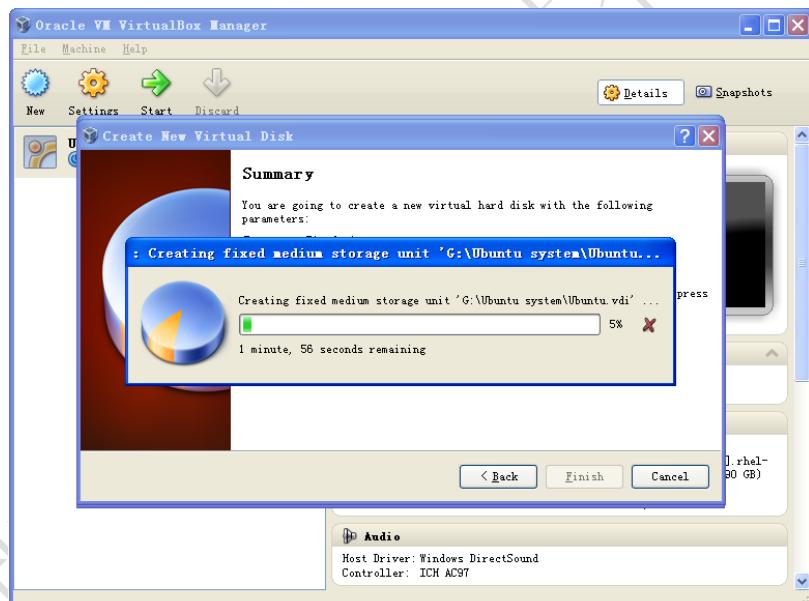


图11 创建虚拟硬盘驱动器

10) 完成驱动器创建后会显示摘要信息。请单击完成按钮即完成虚拟机安装过程；

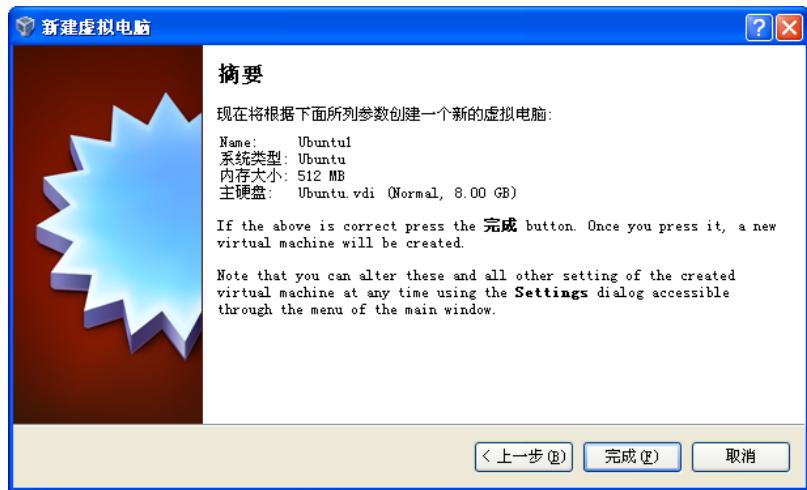


图12 虚拟机配置完成

## 安装 Ubuntu Linux 系统

虚拟机安装完成后，我们就可以开始安装 Ubuntu Linux 系统了。首先请访问 <http://www.Ubuntu.com/download/Ubuntu/download>，下载 Ubuntu 的 ISO 映像文件，然后按照以下步骤进行安装。

- 1) 从开始菜单启动 VirtualBox，然后在程序窗口上方单击 **Setting** 按钮，弹出虚拟机设置窗口，如下图所示；

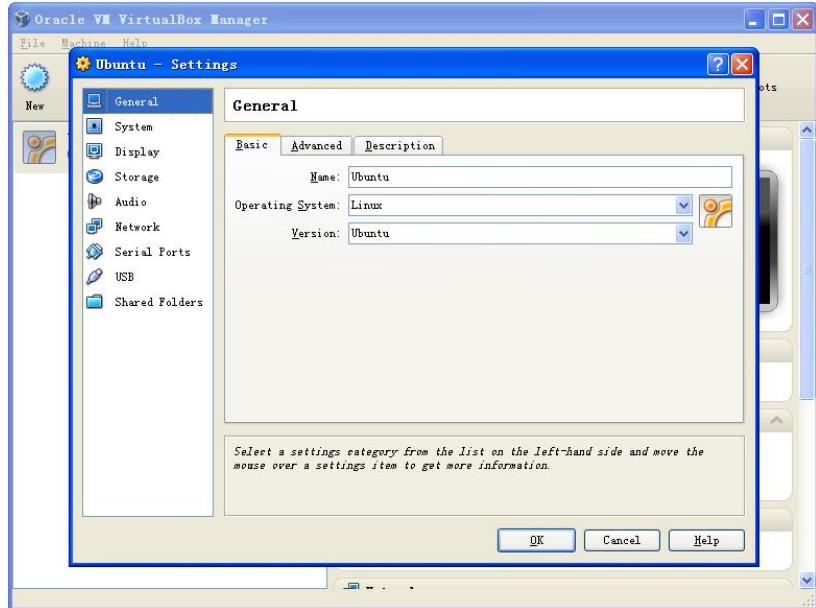


图13 虚拟机设置

- 2) 在窗口左方选择 **Storage** 选项，然后单击 IDC 控制器下 **Empty** 文字右方的光盘图标来指定 Ubuntu 映像文件的位置，如下图所示；

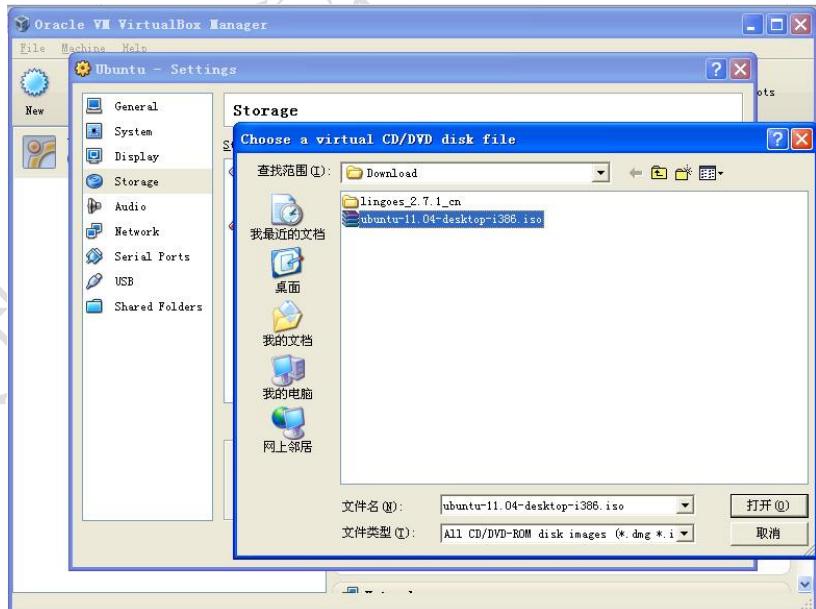


图14 指定 Ubuntu 映像位置

- 3) 选中刚才添加的映像并单击 **OK** 按钮, 如下图所示;

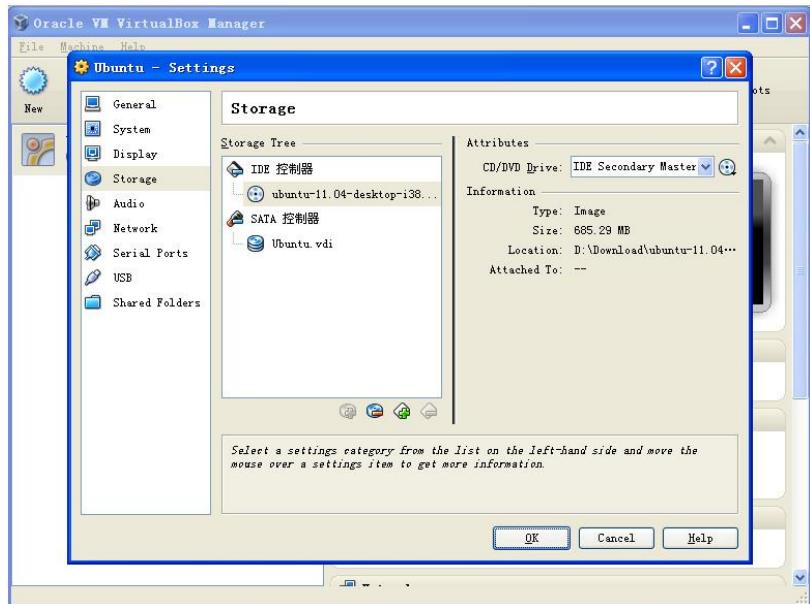


图15 选中 ISO 映像

- 4) 单击 VirtualBox 窗口上方的 **Start** 按钮, 屏幕会弹出 Ubuntu 的安装初始化窗口, 如下图所示;

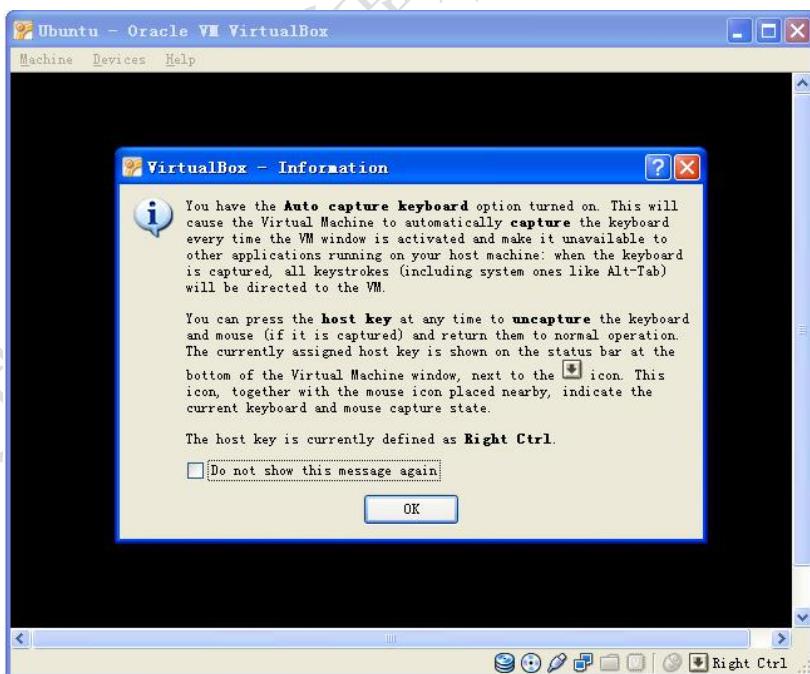


图16 Ubuntu 初始化窗口

在 Ubuntu 安装窗口初始化过程中会出现一些提示信息，只需要单击提示信息下方的 **OK** 按钮即可继续初始化进程。

- 5) 在出现以下窗口时单击 **Install Ubuntu** 进行安装，如下图所示；



图17 Ubuntu 安装窗口

- 6) 单击 **Forward** 按钮继续安装，如下图所示；



图18 安装前的信息

- 7) 选择 Erase disk and install Ubuntu 选项，并单击 Forward 按钮。

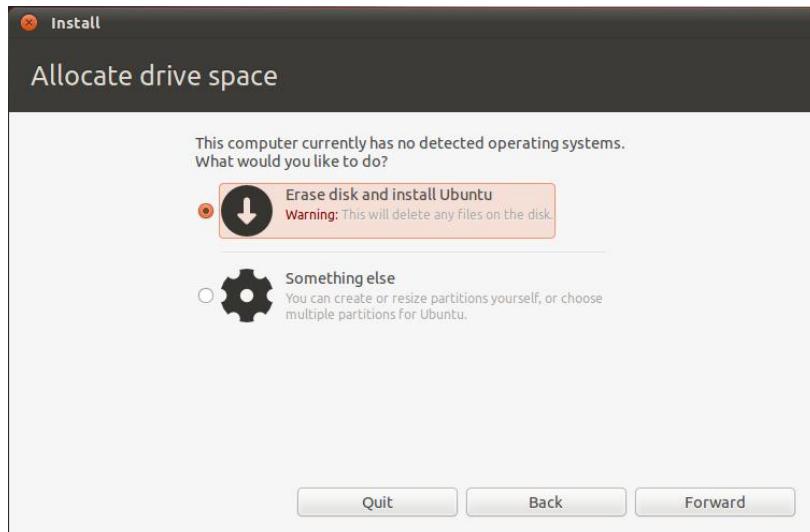


图19 Ubuntu 安装选项

**注意：**

选择该选项并不会删除硬盘上物理分区中的任何内容。

- 8) 单击下方窗口中的 **Install Now** 按钮开始安装 Ubuntu;

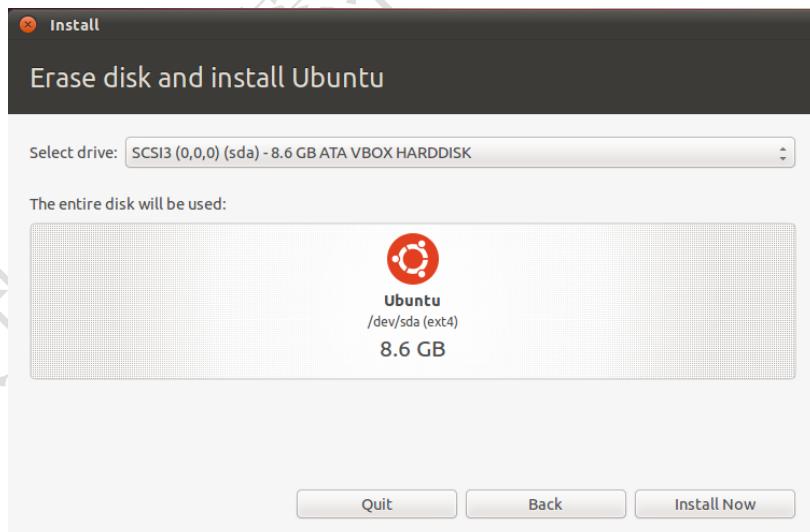


图20 安装确认窗口

- 9) 在安装过程中安装程序会询问一些简单的问题，请输入相应的信息并单击 **Forward** 按钮即可。安装过程中的最后一个问题窗口如下图所示；

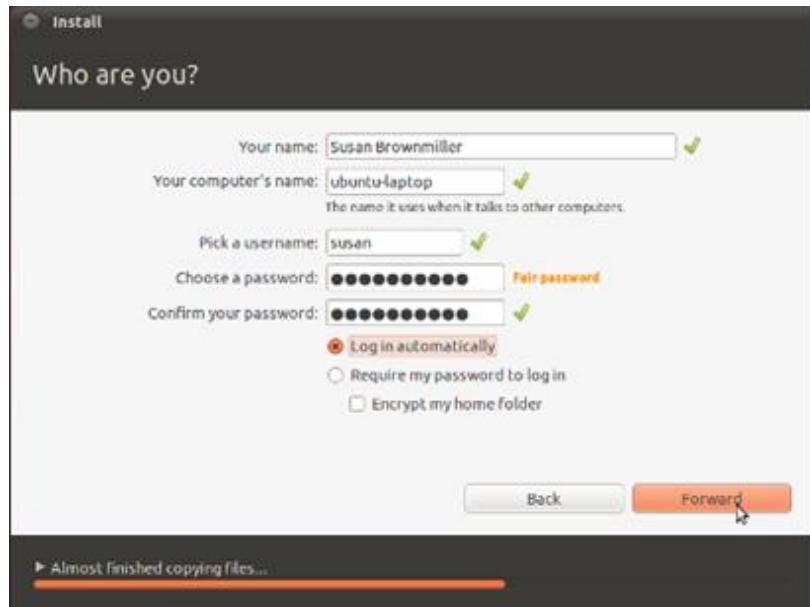


图21 指定用户名和密码

在相应的文本框内输入用户名和密码后，选择 **Log in automatically** 选项并单击 **Forward** 按钮。

- 10) Ubuntu 的安装依据不同的 PC 性能可能需要 15 分钟至 1 小时左右。安装完成后会弹出如下图所示的提示窗口，请选择 **Restart Now** 重新启动 Ubuntu 系统；

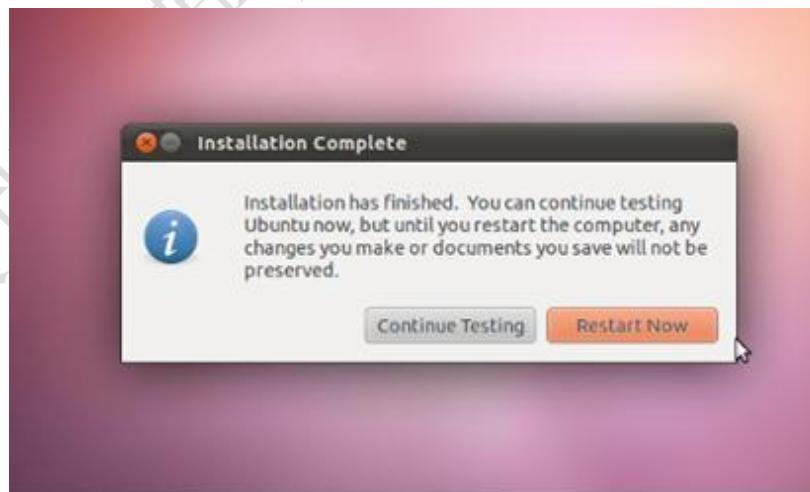


图22 重新启动系统

- 11) 重新启动后就可以使用 Ubuntu 系统了。通常在重新启动 Ubuntu 系统后

VirtualBox 会自动弹出图 15 中载入的映像文件，如果没有自动弹出，您可以在 VirtualBox 的 **Setting** 窗口中手动弹出该映像，使 IDE 控制器下显示为 Empty，如下图所示；

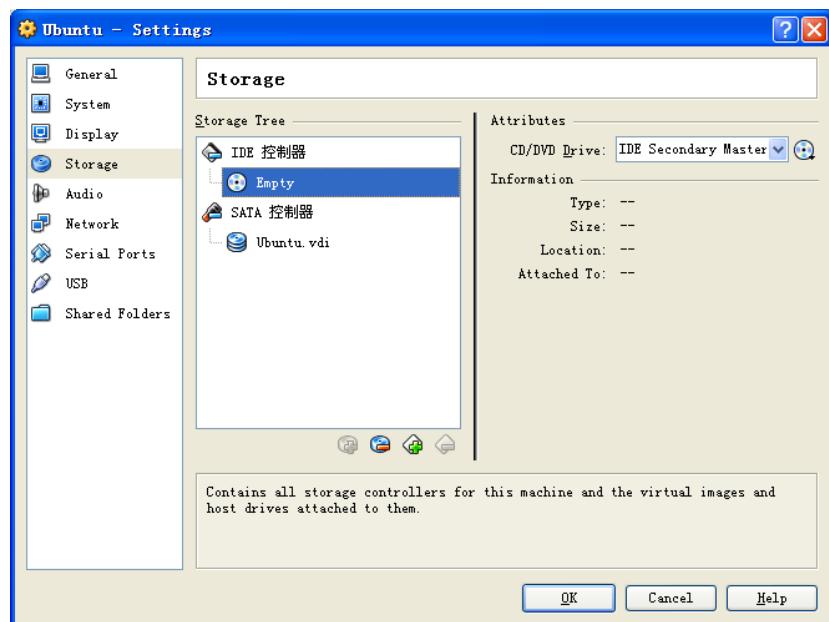


图23 弹出 ISO 映像

### 附录三 Linux USB Ethernet/RNDIS Gadget 驱动安装

- 1) 如果你还没安装 Linux USB Ethernet/RNDIS Gadget 驱动，PC 会提示发现新硬件界面，选中“从列表或指定位置安装”，然后点击“下一步”。



图24 找到新硬件

- 2) 指定 USB 驱动路径[光盘\linux\tools]，然后点击“下一步”。

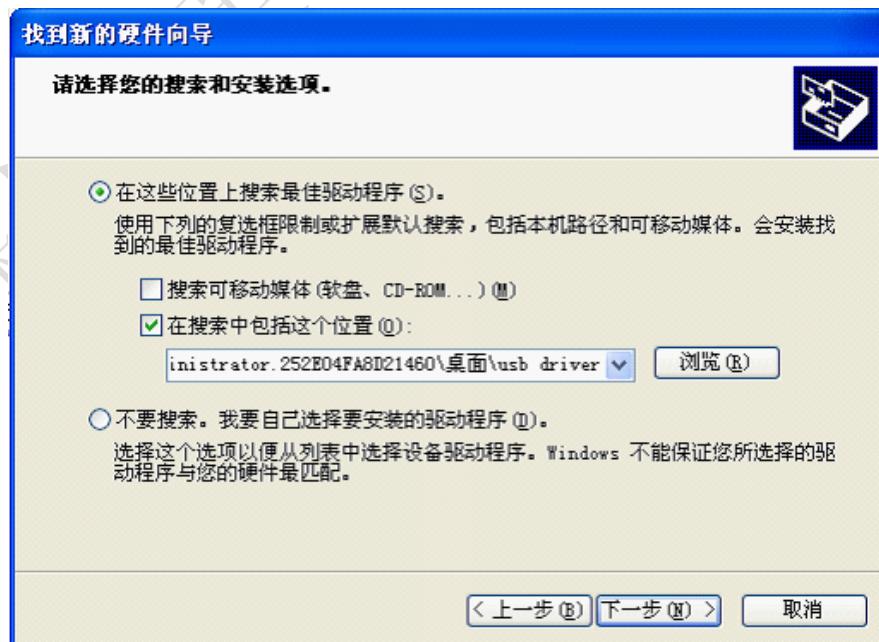


图25 选择驱动位置

- 3) 出现以下提示时，选择继续安装



图26 警告信息

- 4) 等待驱动安装完毕



图27 硬件向导完成窗口

## 附录四 Linux Boot Disk Format

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

### 1、Introduction

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

### 2、Details

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

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

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

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

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

#### 2) Check to see if the automounter has mounted the SD Card

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

\$ [df -h]			
Filesystem	Size	Used	Avail Use% Mounted on

```
...
/dev/sdc1          400M  94M  307M  24% /media/disk
...
```

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

**3) If so, unmount it**

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

**4) Start fdisk**

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

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

**5) Print the partition record**

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

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

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

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

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

```
Selected partition 1
```

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

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

**a) Go into expert mode.**

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

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

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

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

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

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

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

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

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

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

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

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

**f) Return to Normal mode.**

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

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

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

Disk /dev/sdc: 2021 MB, 2021654528 bytes

255 heads, 63 sectors/track, 245 cylinders

Units = cylinders of 16065 \* 512 = 8225280 bytes

Device	Boot	Start	End	Blocks	Id	System
--------	------	-------	-----	--------	----	--------

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

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

Command action

**e** extended

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

**[p]**

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

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

Using default value 1

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

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

Selected partition 1

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

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

### 10) Mark it as bootable

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

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

### 11) Create the Linux partition for the root file system

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

Command action

*e* extended

*p* primary partition (1-4)

**[p]**

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

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

Using default value 52

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

Using default value 245

### 12) Print to Check Your Work

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

Disk /dev/sdc: 2021 MB, 2021654528 bytes

255 heads, 63 sectors/track, 245 cylinders

Units = cylinders of 16065 \* 512 = 8225280 bytes

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

### 13) Save the new partition records on the TF Card

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

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

The partition table has been altered!

Calling ioctl() to re-read partition table.

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

The kernel still uses the old table.

The new table will be used at the next reboot.

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

*partitions, please see the fdisk manual page for additional information.*  
*Syncing disks.*

#### 14) Format the partitions

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

```
$ [sudo mkfs.msdos -F32 /dev/sdc1 -n LABEL1]  
mkfs.msdos 2.11 (12 Mar 2005)
```

```
$ [sudo mkfs.ext3 -L LABEL2 /dev/sdc2]  
mke2fs 1.40-WIP (14-Nov-2006)  
Filesystem label=  
OS type: Linux  
Block size=4096 (log=2)  
Fragment size=4096 (log=2)  
195072 inodes, 389576 blocks  
19478 blocks (5.00%) reserved for the super user  
First data block=0  
Maximum filesystem blocks=402653184  
12 block groups  
32768 blocks per group, 32768 fragments per group  
16256 inodes per group  
Superblock backups stored on blocks:  
    32768, 98304, 163840, 229376, 294912  
  
Writing inode tables: done  
Creating journal (8192 blocks): done  
Writing superblocks and filesystem accounting information:
```

#### 注意：

- 在 ubuntu 下格式化好 FAT 和 EXT3 双分区后，FAT 分区需要在 window 下重新格式化一次，否则可能会出现无法从 TF 卡启动的情况。

## 附录五 TFTP 服务器搭建

### 1) 安装客户端

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

### 2) 安装 inet

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

### 3) 服务器配置

首先，在根目录下建一个 tftpboot，并把属性改成任意用户可读写：

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

其次，在/etc/inetd.conf 里添加：

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

然后，重新加载 inetd 进程：

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

最后，进入目录 /etc/xinetd.d/，并在其中新建文件 tftp，把指定的内容加入到 tftp 文件中：

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

```
cps      = 100 2  
}
```

- 4) 重新启动服务:

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

- 5) 测试服务器

在/tftpboot 文件夹下新建立一个文件

```
$>touch abc
```

进入另外一个文件夹

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

如果可以下载说明服务器已经安装成功。

## 附录六 FAQ 总结

请访问: [http://www.elinux.org/SBC8600\\_FAQ](http://www.elinux.org/SBC8600_FAQ)

深圳市英蓓特科技有限公司

# 技术支持和保修服务

## 技术支持



英蓓特科技对所销售的产品提供一年的免费技术支持服务，技术支持服务范围：

- 提供英蓓特科技嵌入式平台产品的软硬件资源；
- 帮助用户正确地编译和运行我们提供的源代码；
- 用户在按照本公司提供的产品文档操作的情况下，如本公司的嵌入式软硬件产品出现异常问题，我们将提供技术支持；
- 帮助用户判定是否存在产品故障。



以下情况不在我们的免费技术支持服务范围内，但我们将根据情况酌情处理：

- 用户自行开发中遇到的软硬件问题；
- 用户自行修改嵌入式操作系统遇到的问题；
- 用户自己的应用程序遇到的问题；
- 用户自行修改本公司提供的软件代码遇到的问题。

## 保修服务

- 1) 产品自出售之日起，在正常使用状况下为印刷电路板提供 12 个月的免费保修服务；
- 2) 以下情况不属于免费服务范围，英蓓特科技将酌情收取服务费用：
  - A. 无法提供产品有效购买凭证、产品识别标签撕毁或无法辨认，涂改标签或标签与实际产品不符；
  - B. 未按用户手册操作导致产品损坏的；
  - C. 因天灾（水灾、火灾、地震、雷击、台风等）或零件之自然耗损或遇不可抗拒力导致的产品外观及功能损坏；

- D. 因供电、磕碰、房屋漏水、动物、潮湿、杂 / 异物进入板内等原因导致的产品外观及功能损坏；
  - E. 用户擅自拆焊零件或修改而导致不良或授权非英蓓特科技认可的人员及机构进行产品的拆装、维修，变更产品出厂规格及配置或扩充非英蓓特科技公司销售或认可的配件及由此引致的产品外观及功能损坏；
  - F. 用户自行安装软件、系统或软件设定不当或由电脑病毒等造成的故障；
  - G. 非经授权渠道购得此产品者。
  - H. 非英蓓特科技对用户做出的超出保修服务范围的承诺（包括口头及书面等）由承诺方负责兑现，英蓓特科技恕不承担任何责任；
- 3) 保修期内由用户发到我们公司的运费由用户承担，由我们公司发给用户的运费由我们承担；保修期外的全部运输费用由用户承担。
- 4) 若板卡需要维修，请联系技术支持服务部。

**注意：**

■ 未经本公司许可私自将产品寄回的，英蓓特科技公司不承担任何责任。

## 联系方式

热线电话: +86-755-25503401

传真号码: +86-755-25616057

售前咨询: [sales@timll.com](mailto:sales@timll.com)

售后支持: [support@timll.com](mailto:support@timll.com)

官方网站: <http://www.timll.com>

通讯地址: 深圳市罗湖区太宁路 85 号罗湖科技大厦 405 室