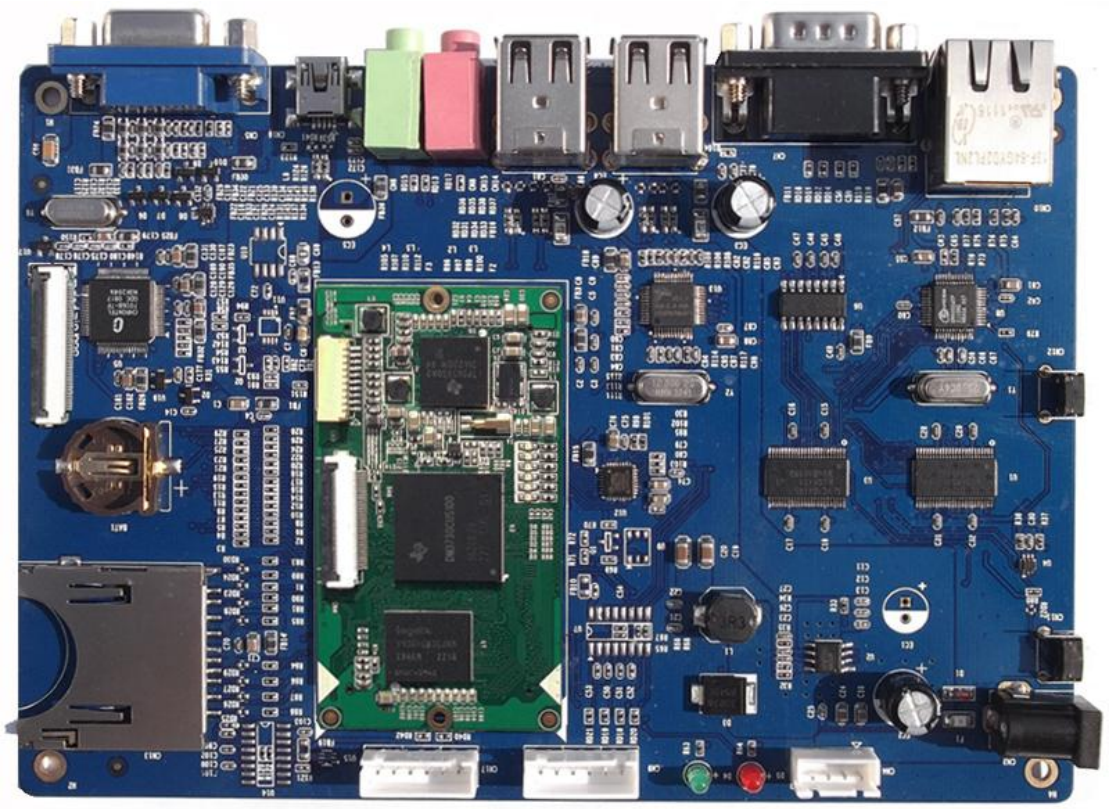
SBC8140 单板机



用户手册

版本 1.0 – 2013 年 3 月 20 日

版权声明：

- SBC8140 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。

免责声明：

- 产品附带光盘所提供的程序源代码、软件、资料文档等，深圳市英蓓特有限公司不提供任何类型的担保；不论是明确的，还是隐含的，包括但不限于合适特定用途的保证，全部的风险，由使用者来承担；如果程序出现缺陷，使用者承担所有必要的服务、修改和改正的费用。

版本更新记录：

版本	更新日期	描述
1.0	2013-3-20	初始版本

目录

第1章 产品概述

1.1 产品介绍

SBC8140 是深圳市英蓓特科技有限公司推出的一款基于 DM3730 处理器的单板机。该单板机采用一块核心板 Mini8510 和一块扩展板的分离式结构。Mini8510 主要包括处理器、MCP（包括 DDR 存储器和 NAND 存储器）和电源管理芯片，组成了一个 TI DM3730 的最小系统。扩展板通过两个 1.27mm 间距的 90-pin DIP 连接器与核心板进行连接，提供了网络接口、音频输入输出接口、USB 接口、SD 卡接口、串行接口、VGA 接口、JTAG 接口、摄像头接口、LCD 接口、触摸屏接口等扩展接口。

SBC8140 单板机的应用场景非常广泛，能够满足包括安防、监控、门禁通信、POS 终端、网络终端和广告机等在内的各个领域的不同需求。

1.2 包装内容

- SBC8140 单板机 x 1
- 交叉串行线缆（DB9 到 DB9）x 1
- 10-pin JTAG 线缆 x 1
- JTAG8000 模块 x 1
- 5V/2A 电源适配器 x 1
- DVD-ROM 光盘 x 1
- LCD 屏 x 1（可选配件，480*272 分辨率的 4.3 寸屏或者 800*480 分辨率的 7 寸屏）

1.3 产品特性

1.3.1 Mini8510 核心板

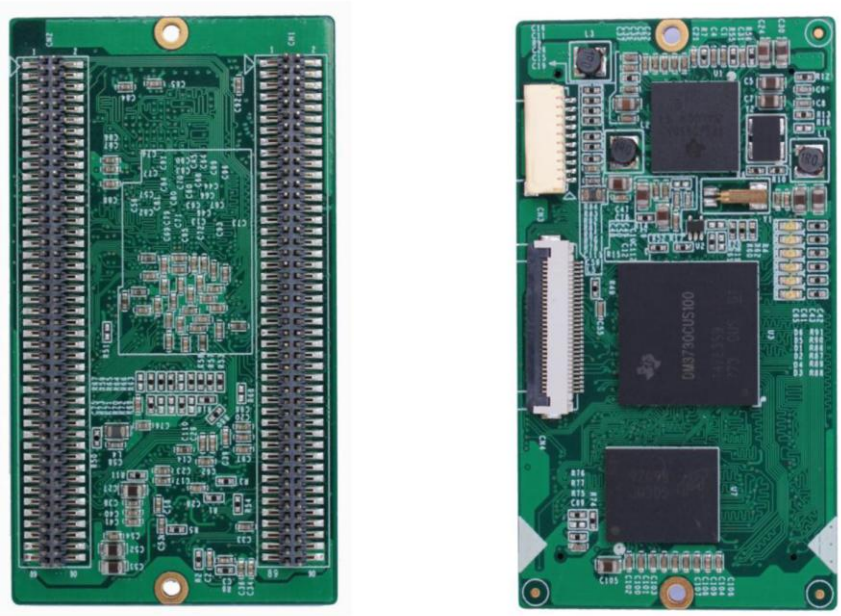


图 1-1 Mini8510 核心板背面（左）和正面（右）

- **产品参数：**
  - 产品尺寸：67mm x 37mm
  - 工作温度：0 ~ 70℃
  - 环境湿度：20% ~ 90%（无凝结）
  - 输入电源：3.3V/0.17A
- **处理器：**
  - 集成 1GHz ARM Cortex™-A8 内核的 TI DM3730
  - 集成 800-MHz TMS320C64x+™ DSP
  - 集成 NEON™ SIMD 协处理器
  - 集成 POWERVR SGX™图形加速器
  - 集成 32KB 指令缓存、32KB 数据缓存、256KB 二级缓存、64KB RAM 和 32KB ROM
- **板载存储器：**

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

- **板载接口和信号：**
  - 一路摄像头接口（可外接 CCD 和 CMOS 的摄像头）
  - 一路 JTAG 接口
  - 2 个 1.27mm 间距 90-pin 的 DIP 连接器
  - 6 个 LED 灯（2 个电源指示灯和 4 个用户自定义 LED 灯）
  - 两路 SPI：SPI1 和 SPI2
  - GPMC 总线（16 位数据、10 位地址、4 个片选、控制信号若干）
  - 三路 UART（5 线，支持硬件流控）
  - 一路 ULPI（USB1 HS）
  - 音频输入/输出
  - 一路 IIC（IIC3）
  - 2 路 McBSP：McBSP1 和 McBSP3（McBSP3 与 UART2 复用）
  - 2 路 MMC/SD：MMC1（8 线）和 MMC2（4 线）
  - 24 位 DSS 接口

1.3.2 扩展板

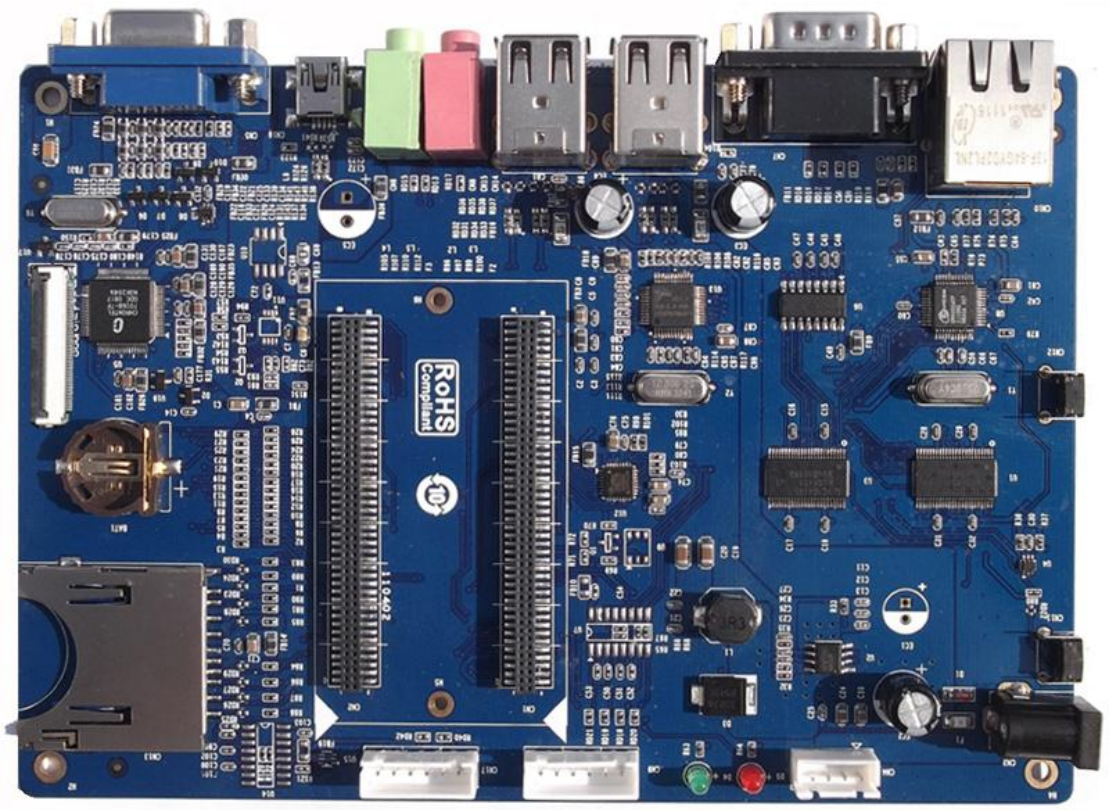


图 1-2 SBC8140 的扩展板

- **产品参数：**
  - 机械尺寸：165mm x 115mm
  - 工作温度：0 ~ 70° C
  - 环境温度：20% ~ 90%（无凝结）
  - 输入电压：5V/2A
- **音频/视频接口：**
  - LCD/触摸屏接口（24 位数据 RGB 全彩色输出；50-pin FPC 连接器）
  - 一个标准 VGA 接口，默认支持 1024*768 分辨率
  - 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：**
  - 一个 10/100Mbps 以太网接口（RJ45 连接器）

- 一个高速 USB 2.0 OTG 接口，集成 PHY（480Mbps Mini USB 接口）
- 四个高速 USB 2.0 HOST 接口，集成 PHY（480Mbps USB-A 接口）
- 一个 SD 卡接口（兼容 SD/MMC 通信）
- 串行接口

表 1-1 串行接口

串口名称	描述
UART1	5 线 RS232 电平，DB9 调试串口
UART2	3 线 TTL 电平，6PIN 插线端子
UART3	5 线 RS232 电平，6PIN 插线端子

- **输入接口：**
  - 1 个 BOOT 按键
  - 1 个复位按键
- **LED 指示灯**
  - 1 个电源指示灯
  - 2 个用户自定义灯

1.4 接口分布

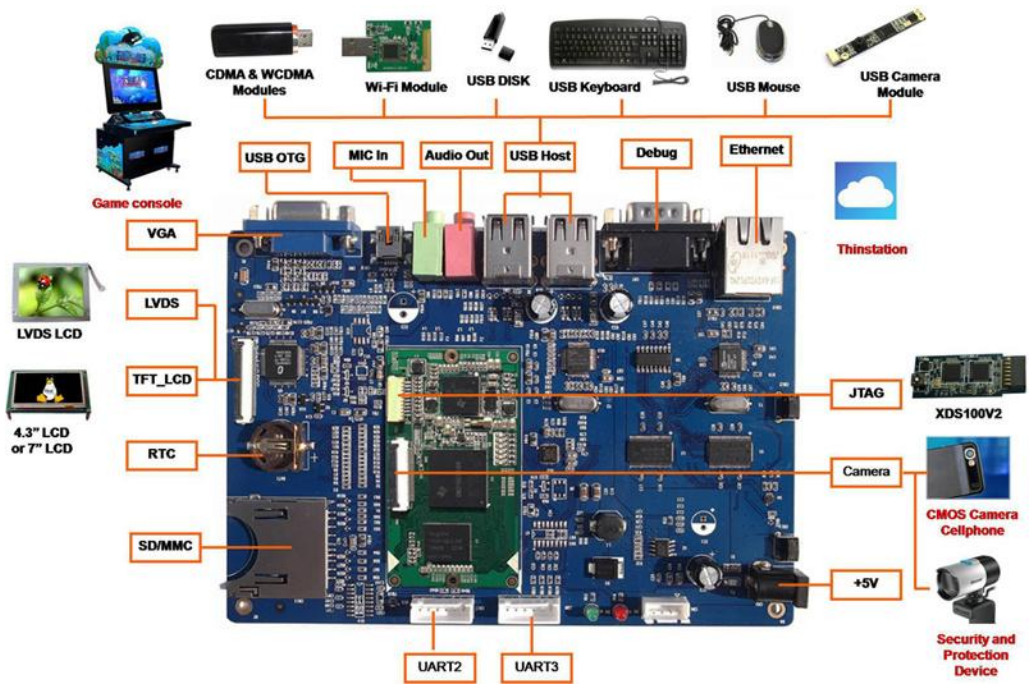


图 1-3 SBC8140 接口分布

1.5 系统框图

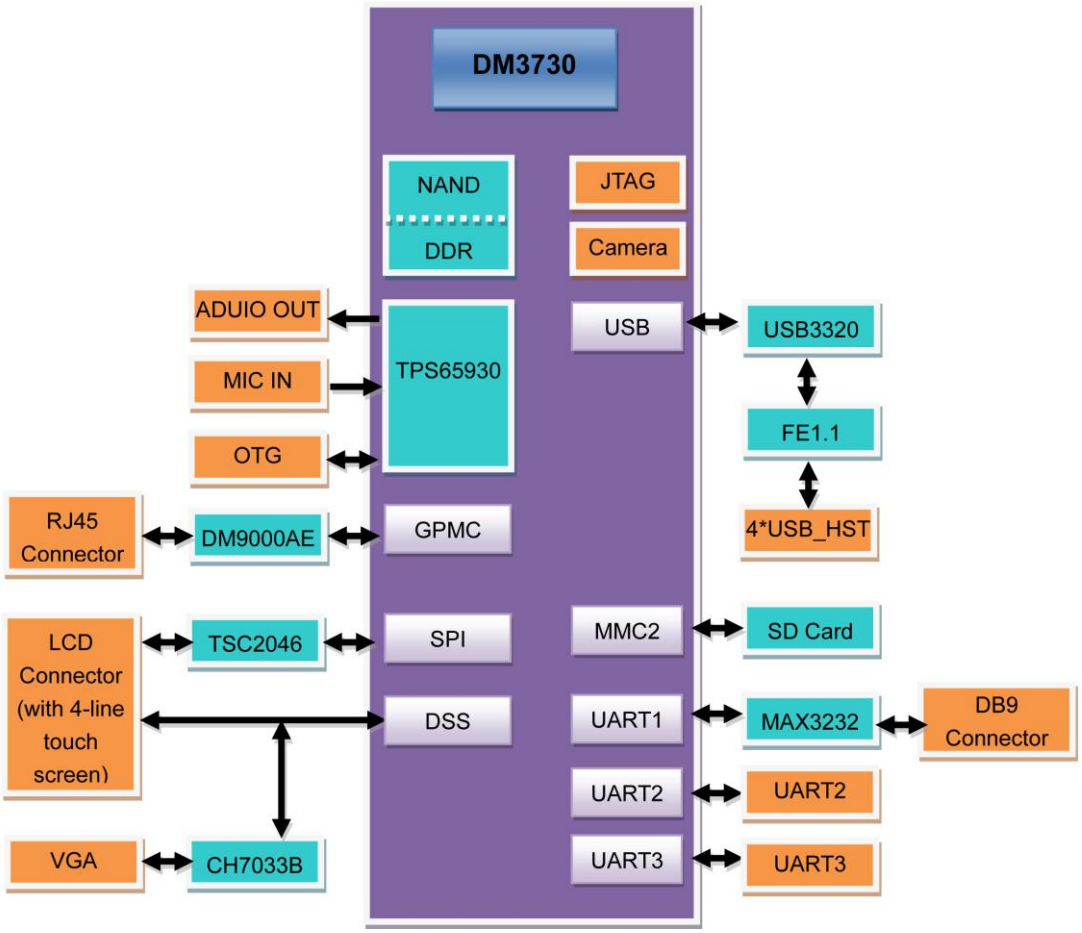


图 1-4 SBC8140 系统框图

1.6 硬件尺寸（毫米）

1.6.1 Mini8510 核心板

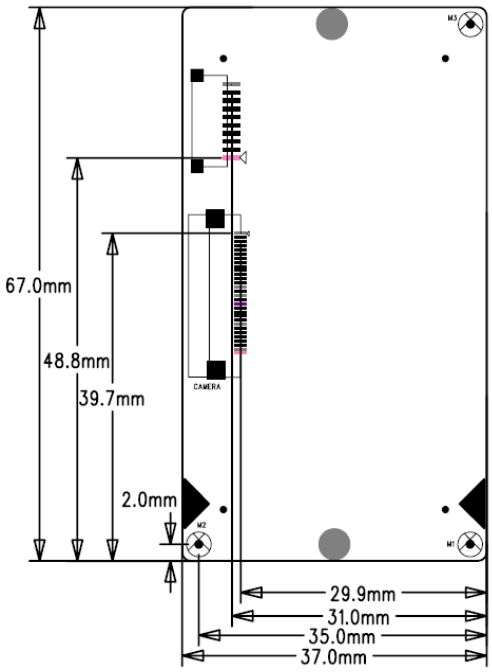


图 1-5 Mini8510 硬件尺寸（正面）

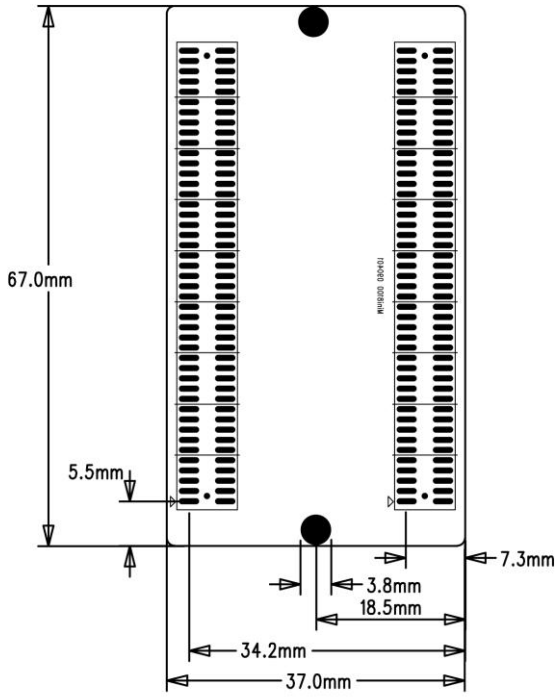


图 1-6 Mini8510 硬件尺寸（背面）

1.6.2 SBC8140 的扩展板

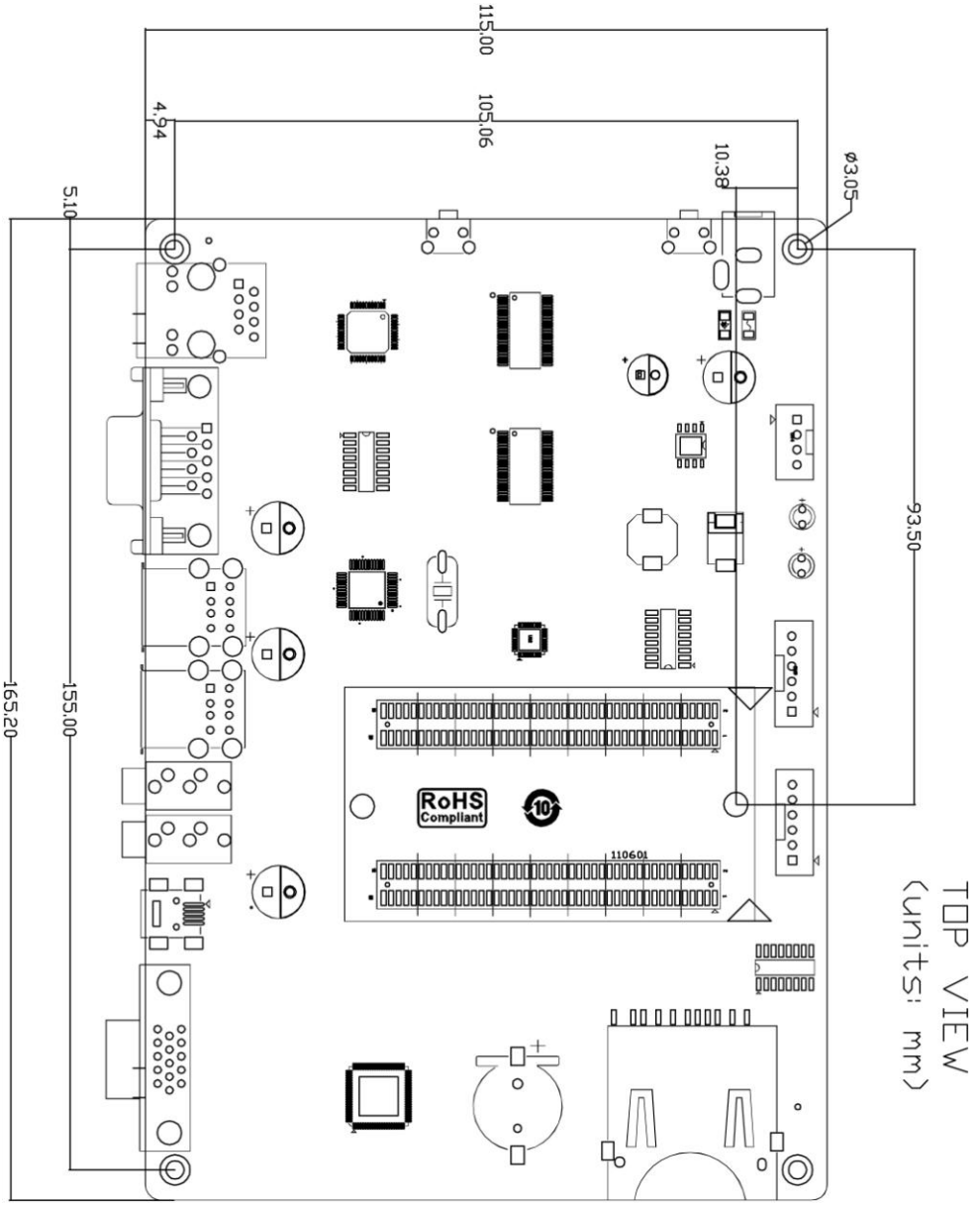


图 1-7 SBC8140 的扩展板尺寸

1.7 SBC8140 支持的模块

表 1-2 支持的模块

名称	Linux	Android	WinCE	相关资料
WF8000-U	Yes*	NO	Yes#	单独光盘提供
CAM8000-A	Yes*	NO	Yes*	开发板配套光盘提供
CAM8000-D	Yes#	Yes#	Yes*	单击此处下载

CAM8100-U	Yes*	Yes*	Yes#	单独光盘提供
CDMA8000-U	Yes*	No	Yes#	单击此处下载
WCDMA8000-U	Yes*	No	Yes#	单击此处下载
LVDS8000	Yes*	Yes*	Yes*	开发板配套光盘和网站提供

备注：“*”表示提供源码，“#”表示不提供源码

第2章 硬件系统介绍

本章节将通过简要介绍核心板 Mini8510 上所采用的 CPU、CPU 周边芯片、以及整个产品（Mini8510+扩展板）所提供的各种接口的引脚定义来让您对该硬件系统有一个大致的了解。

2.1 CPU 简介

核心板 Mini8510 采用了来自 TI 的 45 纳米高性能、低功耗、增强型数字媒体处理器 DM3730。该处理器由 1GHz 的 Cortex-A8 内核和 800MHz 的 TMS320C64+ DSP 内核组成，并集成了 3D 图形处理器、视频加速器（IVA）和 USB 2.0 等，提供了对 720p 视频编解码支持。

2.1.1 时钟

DM3730 的时钟信号包括 sys_32k、sys_altclk、sys_clkout1、sys_clkout2、sys_xtalout、sys_xtalin 和 sys_clkreq，其中：

- **sys_32k**：该时钟频率为 32KHz，由电源管理芯片 TPS65930 产生，用于低频率运算；它通过管脚 sys_32k 来启用低耗电模式（off mode）。
- **sys_xtalou** 和 **sys_xtalin**：系统输入时钟，频率为 26MHz，用于为 DPLLs 和其他模组提供主时钟信号。

2.1.2 复位信号

复位信号是由 CPU 的 SYS_NRESPWRON 决定；低电平代表复位有效。

2.1.3 通用接口

通用接口设备包括 6 组通用输入输出接口（GPIO）。每一个 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 可以拥有最多 192 个（6x32）管脚。这些

管脚可以针对数据输入输出（驱动）、键盘接口、终端控制等不同的应用进行配置。

2.1.4 显示子系统

显示子系统主要为 LCD 或者 TV 接口提供用于存储帧缓存（SDRAM 或者 SRAM）的逻辑视频图像；显示子系统包括以下部分：

- 显示控制（DISPC）模组
- 远程帧缓冲接口（RFBI）模组
- 显示串行接口（DSI）的 I/ O 模块和 DSI 协议引擎
- DSI PLL 控制器驱动（DSI PLL 和高速 HS 分频器）
- NTSC/PAL 视频编码

显示控制器和 DSI 协议引擎连接到 L3 和 L4 的内部总线上，而 RFBI 与 TV 输出编码模组连接到 L4 内部总线。

2.1.5 3D 图形加速系统

2D/3D 图形加速（SGX）系统加快了 2D 和 3D 的图形应用速度。SGX 系统基于 Imagination Technologies 公司的 POWERVR® SGX 核心，它是新一代可编程的 POWERVR 图形核心。POWERVR SGX530 v1.2.5 的架构具有可伸展性，可用于从主流移动设备到高端桌面图形处理等各种特定的应用。其目标应用包括功能手机、PDA 和一些掌上游戏机等。

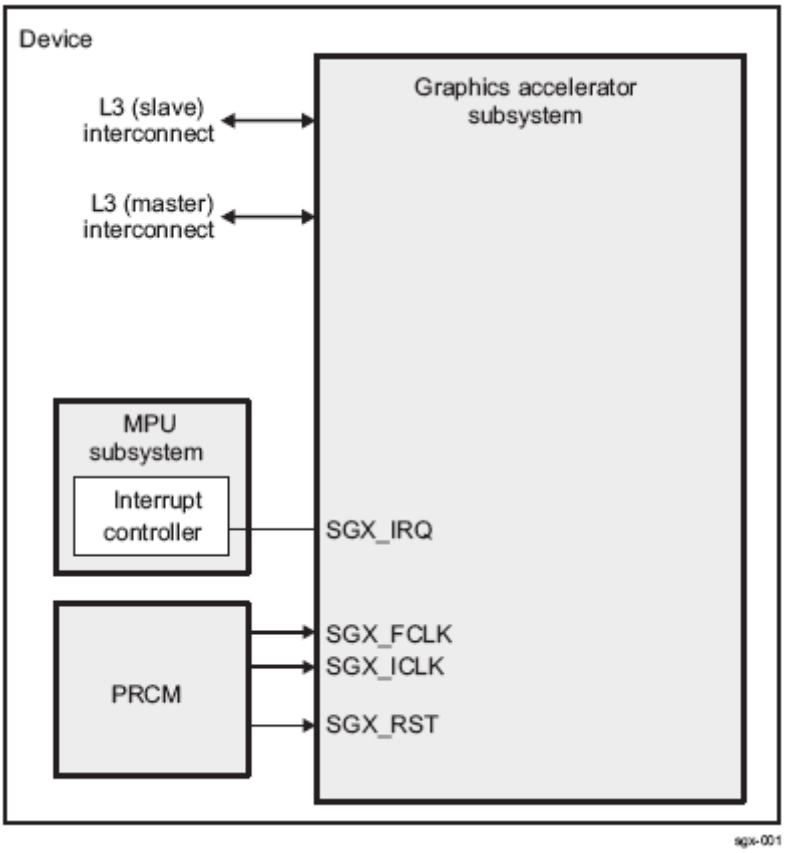


图 2-1 SGX 图形加速系统

SGX 图形加速系统的架构采用两级调度和数据分区来实现多线程任务切换，可以同时处理像素、顶点、视频和通用数据。

2.2 CPU 周边芯片

2.2.1 TPS65930 电源管理芯片

TPS65930 是用于 OMAP™系列的电源管理芯片，它包括电源管理控制器、USB 高速传输控制、LED 驱动控制、模数转换（ADC）、实时时钟（RTC）和嵌入式时钟管理（EPC）等。TPS65930 还包括完整的两路音频数模转换通道、两路 ADC 双语音通道、一路标准音频采样率（I2S™ ）/时分复用（TDM）接口，可以在立体声下行通道上播放标准的音频。

TPS65930 与 CPU 之间使用 I2C 协议进行通信，TPS65930 主要的作用是将 1.2V、1.8V 提供给 CPU，让 CPU 正常运作。此外 TPS65930 还有 Audio in、Audio out、OTG PHY、Keyboard、ADC、GPIO 功能。

2.2.2 H9DA4GH2GJAMCR 存储器

H9DA4GH2GJAMCR 是 SBC8140 上的 NAND Flash 与 SDRAM DDR 二合一存储芯片，其中 NAND Flash 的容量是 512MB，SDRAM DDR 的容量是 256MB。NAND Flash 是通过 GPMC 总线实现数据访问，而 DDR 则是使用 SDRAM Controller（SDRC）实现数据访问。

2.2.3 DM9000 以太网控制器

M9000 是高度集成了快速以太网控制器与通用处理器的低功耗芯片，配备一路 10/100M PHY 和 4K DWORD SRAM。该芯片支持 3.3V 与 5V 的 IO 接口。

SBC8140 使用的是 DM9000 的 10/100M 自适应网络接口。DM9000 内置的 10/100M Ethernet 模块，它兼容 IEEE 802.3 标准协议。网线接口为标准的 RJ45，并且带有连接指示灯和传输指示灯。

SBC8140 可通过直通网线连接到集线器上，也可用交叉网线与 PC 直接相连。

2.2.4 FE1.1 USB 集线器

FE1.1 是 USB 2.0 高速 4 端口集线器的 USB 解决方案。它通过 USB3320 扩展出 4 路 USB 接口，支持高速（480MHz）、全速（12MHz）和低速（1.5MHz）模式。

2.2.5 TFP410 平板显示芯片

TFP410 是德州仪器兼容 PanelBus 平板显示产品的终端至终端的 DVI1.0 解决方案，面向 PC 行业与消费类电子行业。

TFP410 是用于图形控制的无胶合连接通用接口，其接口的特性优势是可选择总线宽度，可适应不同的电平信号和信号沿时序，适应 1.1V – 1.8V 范围内的数字电平 low-EMI。高速总线宽度提供 12 位或者 24 位宽度选择。在 24 位真彩色格式的范围内，DVI 的显示分辨率频率高达 165MHz。

2.2.6 MAX3232 收发器

MAX3232 采用专有低压差发送器输出级，利用双电荷泵在 3.0V 至 5.5V 电源供电时

能够实现真正的 RS-232 性能。器件仅需要四个外部小尺寸 0.1uF 电荷泵电容。MAX3232 能够保证 120kbps 的数据速率，同时保持 RS-232 输出电平。

MAX3232 具有二路接收器和二路驱动器，提供 1uA 关断模式，能有效降低功耗并延长便携式产品的电池使用寿命。关断模式下，接收器保持有效状态，对外部设备进行监测，仅消耗 1uA 电源电流。MAX3232 的引脚、封装和功能与符合工业标准的 MAX232 兼容。即使工作在高数据速率下，MAX3232 仍然能保持 RS-232 标准要求的正负 5.0V 最小发送器输出电压。

MAX3232 在最差工作条件下能够保证 120kbps 的数据速率。通常情况下，能够工作于 235kbps 数据速率，发送器可并联驱动多个接收器和鼠标。

2.3 Mini8510 硬件接口和 LED 指示灯

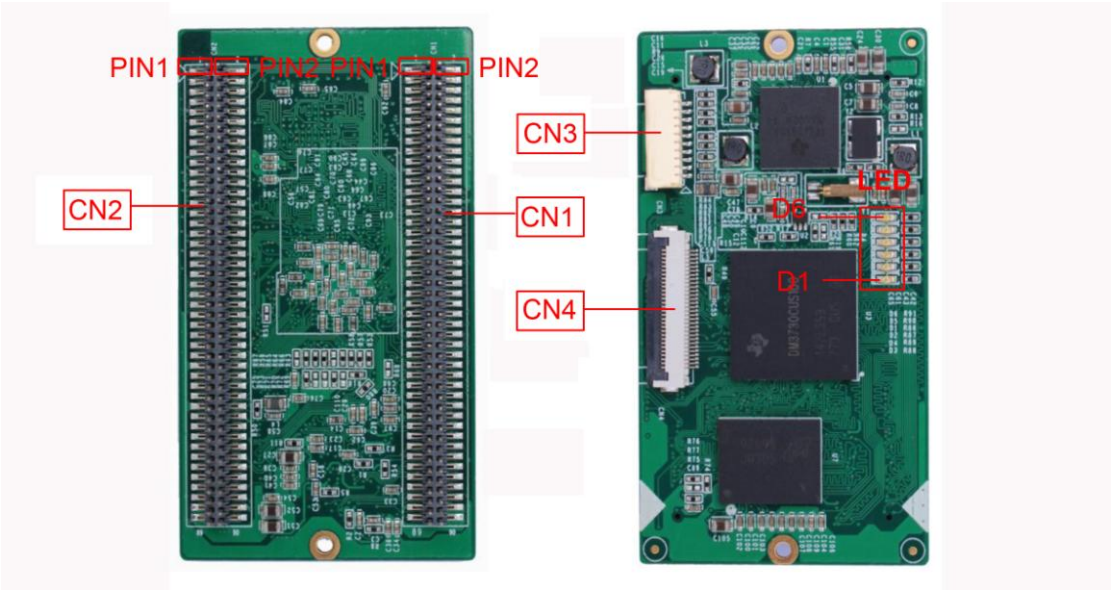


图 2-2 Mini8510 硬件接口

2.3.1 CN1 90pin DIP 连接器（右列管脚）

表 2-1 CN1 90pin DIP 连接器（右列管脚）

引脚	信号定义	描述
1	GND1	GND
2	G_D14	GPMC data bit 14
3	G_D13	GPMC data bit 13

引脚	信号定义	描述
4	G_D10	GPMC data bit 10
5	G_D8	GPMC data bit 8
6	G_D9	GPMC data bit 9
7	G_D5	GPMC data bit 5
8	G_D7	GPMC data bit 7
9	G_D3	GPMC data bit 3
10	G_D6	GPMC data bit 6
11	G_D12	GPMC data bit 12
12	G_D2	GPMC data bit 2
13	G_D11	GPMC data bit 11
14	G_D1	GPMC data bit 1
15	G_D4	GPMC data bit 4
16	G_D0	GPMC data bit 0
17	G_A2	GPMC address bit 2
18	G_A3	GPMC address bit 3
19	G_A1	GPMC address bit 1
20	G_A6	GPMC address bit 6
21	G_A4	GPMC address bit 4
22	G_A7	GPMC address bit 7
23	G_A5	GPMC address bit 5
24	G_A8	GPMC address bit 8
25	G_A9	GPMC address bit 9
26	G_D15	GPMC data bit 15
27	G_A10	GPMC address bit 10
28	GND2	GND
29	SPI2_CS1/GPT8	SPI Enable 1PWM or event for GP timer 8
30	SPI2_CS10/GPT11	SPI Enable 0PWM or event for GP timer 11
31	SPI2_SIMO/GPT9	Slave data in, master data out PWM or event for GP timer 9
32	SPI2_CLK	SPI Clock
33	SPI2_SOMI GPT10	Slave data out, master data inPWM or event for GP timer 10
34	SPI1_CS3	SPI Enable 3
35	SPI1_CS0	SPI Enable 0
36	SPI1_SIMO	Slave data in, master data out
37	SPI1_SOMI	Slave data out, master data in
38	SPI1_CLK	SPI Clock
39	GND3	GND
40	GPIO0	GPIO0 /card detection 1

引脚	信号定义	描述
41	MMC2_D2/SPI3_CS1	MMC/SD Card Data bit 2SPI Enable 1
42	MMC2_D3/SPI3_CS0	MMC/SD Card Data bit 3SPI Enable 0
43	MMC2_D0/SPI3_SOMI	MMC/SD Card Data bit 0Slave data out, master data in
44	MMC2_D1	MMC/SD Card Data bit 1
45	MMC2_CMD/SPI3_SIMO	MMC/SD command signalSlave data in, master data out
46	MMC2_CLK/SPI3_CLK	MMC/SD Output ClockSPI Clock
47	BSP3_DR/UART2_RTS	Received serial dataUART2 Request To Send
48	BSP3_CLK/UART2_TX	Combined serial clockUART2 Transmit data
49	BSP3_FSX/UART2_RX	Combined frame synchronizationUART2 Receive data
50	BSP3_DX/UART2_CTS	Transmitted serial dataUART2 Clear To Send
51	GND4	GND
52	UART1_CTS	UART1 Clear To Send
53	UART1_TX	UART1 Transmit data
54	UART1_RX	UART1 Receive data
55	UART1_RTS	UART1 Request To Send
56	USB1HS_STP	Dedicated for external transceiver Stop signal
57	USB1HS_D3	Dedicated for external transceiver Bidirectional data bus
58	USB1HS_D5	Dedicated for external transceiver Bidirectional data bus
59	USB1HS_6	Dedicated for external transceiver Bidirectional data bus
60	USB1HS_D7	Dedicated for external transceiver Bidirectional data bus
61	USB1HS_D1	Dedicated for external transceiver Bidirectional data bus
62	USB1HS_D2	Dedicated for external transceiver Bidirectional data bus
63	USB1HS_D4	Dedicated for external transceiver Bidirectional data bus
64	USB1HS_D0	Dedicated for external transceiver Bidirectional data bus
65	USB1HS_NXT	Dedicated for external transceiver Next signal from PHY

引脚	信号定义	描述
66	USB1HS_CLK	Dedicated for external transceiver 60-MHz clock
67	GND6	GND
68	USB1HS_DIR	Dedicated for external transceiver data form PHY
69	SYS_CLKOUT1	Configurable output clock1
70	LEDA	LED leg A
71	LEDB	LED leg B
72	ADCIN0	ADC input0 (Battery type)
73	NRESPWRON	Power On Reset
74	NRESWARM	Warm Boot Reset (open drain output)
75	SYSEN	System enable output
76	GND6	GND
77	REGEN	Enable signal for external LDO
78	ADCIN1	ADC input1 (General-purpose ADC input)
79	KC0	Keypad column 0
80	KC1	Keypad column 0
81	KC2	Keypad column 0
82	KC3	Keypad column 0
83	AUDIO_IN	Analog microphone bias 1
84	AUDIO_OR	Predriver output right P for external class-D amplifier
85	AUXR	Auxiliary audio input right
86	AUDIO_OL	Predriver output left P for external class-D amplifier
87	GND7	GND
88	VBAT1	Power supply (3V - 4.2V 1.5A)
89	ON/OFF	Input; detect a control command to start or stop the system
90	VBAT2	Power supply (3V - 4.2V 1.5A)

2.3.2 CN2 90pin DIP 连接器（左列管脚）

表 2-2 CN2 90pin DIP 连接器（左列管脚）

引脚	信号定义	描述
1	G_NWE	GPMC Write Enable
2	G_NOE	GPMC Read Enable
3	G_NCS7/GPT8/G_DIR	GPMC Chip Select bit 7PWM / event for

引脚	信号定义	描述
		GP timer 8GPMC / IO direction control for use with external transceivers
4	G_NCS4/DMAREQ1	GPMC Chip Select bit 7PWM /DMA request 1
5	G_NCS6/DMAREQ3	GPMC Chip Select bit 7PWM / DMA request 3
6	G_NCS3DMAREQ0	GPMC Chip Select bit 7External DMA request 0
7	GND1	GND
8	G_WAIT0	External indication of wait
9	G_NBE0 / G_CLE	Lower Byte Enable. Also used for Command Latch Enable
10	G_ALE	Address Latch Enable
11	G_NBE1	Upper Byte Enable
12	HDQ_SIO	Bidirectional HDQ 1-Wire control and data
13	MMC1_D0	MMC/SD Card Data bit 0
14	MMC1_D1	MMC/SD Card Data bit 1
15	MMC1_D2	MMC/SD Card Data bit 2
16	MMC1_D6/IO128	MMC/SD Card Data bit 6
17	MMC1_D5/IO127	MMC/SD Card Data bit 5
18	MMC1_D4/IO126	MMC/SD Card Data bit 4
19	MMC1_D7/IO129	MMC/SD Card Data bit 7
20	MMC1_D3	MMC/SD Card Data bit 3
21	GND2	GND
22	MMC1_CLK	MMC/SD Output Clock
23	MMC1_CMD	MMC/SD command signal
24	VMMC1	Power supply for SD/MMC1 (3.0 / 1.8V)
25	UART3_RX	UART3 Receive data
26	UART3_CTS	UART3 Clear To Send
27	UART3_TX	UART3 Transmit data
28	UART3_RTS	UART3 Request To Send
29	DSS_ACBIAS	AC bias control (STN) or pixel data enable (TFT) output
30	DSS_VSYNC	LCD Vertical Synchronization
31	GND3	GND
32	DSS_HSYNC	LCD Horizontal Synchronization
33	DSS_CLK	LCD Pixel Clock
34	DSS_D6	LCD Pixel Data bit 6
35	DSS_D8	LCD Pixel Data bit 8
36	DSS_D7	LCD Pixel Data bit 7

引脚	信号定义	描述
37	DSS_D9	LCD Pixel Data bit 9
38	DSS_D20	LCD Pixel Data bit 20
39	DSS_D17	LCD Pixel Data bit 17
40	DSS_D16	LCD Pixel Data bit 16
41	DSS_D18	LCD Pixel Data bit 18
42	DSS_D10	LCD Pixel Data bit 10
43	DSS_D5	LCD Pixel Data bit 5
44	DSS_D4	LCD Pixel Data bit 4
45	GND4	GND
46	DSS_D2	LCD Pixel Data bit 2
47	DSS_D3	LCD Pixel Data bit 3
48	DSS_D0	LCD Pixel Data bit 0
49	DSS_D15	LCD Pixel Data bit 15
50	DSS_D11	LCD Pixel Data bit 11
51	DSS_D23	LCD Pixel Data bit 23
52	DSS_D22	LCD Pixel Data bit 22
53	DSS_D14	LCD Pixel Data bit 14
54	DSS_D19	LCD Pixel Data bit 19
55	DSS_D13	LCD Pixel Data bit 13
56	DSS_D21	LCD Pixel Data bit 21
57	DSS_D1	LCD Pixel Data bit 1
58	DSS_D12	LCD Pixel Data bit 12
59	GND5	GND
60	MCBSP1_FSR/IO157	Receive frame synchronization
61	MCBSP1_CLKR/IO156	Receive Clock
62	MCBSP1_FSX/IO161	Transmit frame synchronization
63	MCBSP1_CLKS/IO160	External clock input
64	MCBSP1_CLKX/IO162	Transmit clock
65	MCBSP1_DR/IO159	Received serial data
66	MCBSP1_DX/IO158	Transmitted serial data
67	GND6	GND
68	TV_OUTC	TV analog output S-VIDEO: TV_OUT2
69	TV_OUTY	TV analog output Composite: TV_OUT1
70	VDD33_1	Power supply for camera (3.3V 500mA)
71	IIC3_SCL	I2C Master Serial clock. Output is open drain
72	IIC3_SDA	I2C Serial Bidirectional Data. Output is open drain
73	IO25	General-purpose IO 183
74	IO27	General-purpose IO 183
75	BOOTJUMP	Boot configuration mode bit 5.

引脚	信号定义	描述
76	GND7	GND
77	VBUS	VBUS power rail (5V 10mA)
78	USB_DN	USB Data N
79	USB_ID	USB ID
80	USB_DP	USB Data P
81	PWM0	Pulse width driver 0
82	KR0	Keypad row 0
83	KR1	Keypad row 1
84	KR2	Keypad row 2
85	KR3	Keypad row 3
86	KR4	Keypad row 4
87	VDD18_1	Power supply from TPS65930 (VIO 1.8V)
88	GND8	GND
89	VDD18_2	Power supply from TPS65930 (VIO 1.8V)
90	BKBAT	Backup battery

2.3.3 CN3 JTAG 接口

表 2-3 CN3 JTAG 接口

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

2.3.4 CN4 摄像头接口

表 2-4 CN4 摄像头接口

引脚	信号定义	描述

引脚	信号定义	描述
1	GND0	GND
2	D0	Digital image data bit 0
3	D1	Digital image data bit 1
4	D2	Digital image data bit 2
5	D3	Digital image data bit 3
6	D4	Digital image data bit 4
7	D5	Digital image data bit 5
8	D6	Digital image data bit 6
9	D7	Digital image data bit 7
10	D8	Digital image data bit 8
11	D9	Digital image data bit 9
12	D10	Digital image data bit 10
13	D11	Digital image data bit 11
14	GND1	GND
15	PCLK	Pixel clock
16	GND2	GND
17	HS	Horizontal synchronization
18	VDD50	5V
19	VS	Vertical synchronization
20	VDD33	3.3V
21	XCLKA	Clock output a
22	XCLKB	Clock output b
23	GND3	GND
24	FLD	Field identification
25	WEN	Write Enable
26	STROBE	Flash strobe control signal
27	SDA	IIC master serial clock
28	SCL	IIC serial bidirectional data
29	GND4	GND
30	VDD18	1.8V

2.3.5 LED 指示灯

表 2-5 LED 指示灯

LEDs	信号定义	描述
D1	LED1	用户自定义 LED 灯
D2	LED2	用户自定义 LED 灯
D3	LED3	用户自定义 LED 灯
D4	LED4	用户自定义 LED 灯

LEDs	信号定义	描述
D5	VDD18	电源指示灯
D6	VBAT	电源指示灯

2.4 扩展板接口

2.4.1 电源接口

表 2-6 电源接口

引脚	信号定义	描述
1	GND	GND
2	+5V	Power supply (+5V) 2A (Type)

2.4.2 TFT_LCD 接口

表 2-7 TFT_LCD 接口

引脚	信号定义	描述
1	DSS_D0	LCD Pixel data bit 0
2	DSS_D1	LCD Pixel data bit 1
3	DSS_D2	LCD Pixel data bit 2
4	DSS_D3	LCD Pixel data bit 3
5	DSS_D4	LCD Pixel data bit 4
6	DSS_D5	LCD Pixel data bit 5
7	DSS_D6	LCD Pixel data bit 6
8	DSS_D7	LCD Pixel data bit 7
9	GND	GND
10	DSS_D8	LCD Pixel data bit 8
11	DSS_D9	LCD Pixel data bit 9
12	DSS_D10	LCD Pixel data bit 10
13	DSS_D11	LCD Pixel data bit 11
14	DSS_D12	LCD Pixel data bit 12
15	DSS_D13	LCD Pixel data bit 13
16	DSS_D14	LCD Pixel data bit 14
17	DSS_D15	LCD Pixel data bit 15
18	GND	GND
19	DSS_D16	LCD Pixel data bit 16
20	DSS_D17	LCD Pixel data bit 17

引脚	信号定义	描述
21	DSS_D18	LCD Pixel data bit 18
22	DSS_D19	LCD Pixel data bit 19
23	DSS_D20	LCD Pixel data bit 20
24	DSS_D21	LCD Pixel data bit 21
25	DSS_D22	LCD Pixel data bit 22
26	DSS_D23	LCD Pixel data bit 23
27	GND	GND
28	DEN	AC bias control (STN) or pixel data enable (TFT)
29	HSYNC	LCD Horizontal Synchronization
30	VSYNC	LCD Vertical Synchronization
31	GND	GND
32	CLK	LCD Pixel Clock
33	GND	GND
34	X+	X+ Position Input
35	X-	X- Position Input
36	Y+	Y+ Position Input
37	Y-	Y- Position Input
38	SPI_CLK	SPI clock
39	SPI_MOSI	Slave data in, master data out
40	SPI_MISO	Slave data out, master data in
41	SPI_CS	SPI enable
42	IIC_CLK	IIC master serial clock
43	IIC_SDA	IIC serial bidirectional data
44	GND	GND
45	VDD18	1.8V
46	VDD33	3.3V
47	VDD50	5V
48	VDD50	5V
49	RESET	Reset
50	PWREN	Power on enable

2.4.3 音频输出接口

表 2-8 音频输出接口

引脚	信号定义	描述
1	GND	GND
2	NC	NC
3	Right	Right output

引脚	信号定义	描述
4	NC	NC
5	Left	Left output

2.4.4 音频输入接口

表 2-9 音频输入接口

引脚	信号定义	描述
1	GND	GND
2	NC	NC
3	MIC MAIN P	Right input
4	NC	NC
5	MIC MAIN N	Left input

2.4.5 串行接口

表 2-10 串行接口

引脚	信号定义	描述
1	NC	NC
2	RXD	Receive data
3	TXD	Transit data
4	NC	NC
5	GND	GND
6	NC	NC
7	RTS	Request To Send
8	CTS	Clear To Send
9	NC	NC

2.4.6 以太网接口

表 2-11 以太网接口

引脚	信号定义	描述
1	TX+	TX+ output
2	TX-	TX- output
3	RX+	RX+ input
4	VDD25	2.5V Power for TX/RX

引脚	信号定义	描述
5	VDD25	2.5V Power for TX/RX
6	RX-	RX- input
7	NC	NC
8	NC	NC
9	VDD	3.3V Power for LED
10	LED1	Speed LED
11	LED2	Link LED
12	VDD	3.3V Power for LED

2.4.7 USB OTG 接口

表 2-12 USB OTG 接口

引脚	信号定义	描述
1	VBUS	+5V
2	DN	USB Data-
3	DP	USB Data+
4	ID	USB ID
5	GND	GND

2.4.8 USB HOST 接口

表 2-13 USB HOST 接口

引脚	信号定义	描述
1	VBUS	+5V
2	DN	USB Data-
3	DP	USB Data+
4	ID	USB ID

2.4.9 SD 卡接口

表 2-14 SD 卡接口

引脚	信号定义	描述
1	CD/DAT3	Card detect/Card data 2
2	DCMD	Command Signal
3	VSS	GND

引脚	信号定义	描述
4	VDD	VDD
5	CLK	Clock
6	VSS	GND
7	TF_DAT0	Card data 0
8	TF_DAT1	Card data 1
9	TF_DAT2	Card data2
10	SW_2	SD write protect
11	SW_1	Card detect
12	GND	GND

2.4.10 LED 指示灯

表 2-15 LED 指示灯

LEDs	信号定义	描述
D4	LED_POWER	3.3V 电源指示灯
D5	User LED	用户自定义 LED

2.4.11 按钮

按钮	信号定义	描述
CN12	BOOTJUMP	引导系统从 TF 卡启动
CN11	Reset	系统复位

第3章 Linux 操作系统

SBC8140 的板载 NAND Flash 中出厂默认安装了一套完整的 Linux 系统（仅支持 4.3 寸 LCD）。本章节将通过多个小节来对 SBC8140 的 Linux 软件系统进行详细的介绍，包括嵌入式 Linux 系统结构、软件特性、系统开发流程、驱动介绍和开发、系统更新等等。

注意：

- 部分指令前加上了符号“•”，以便防止由于指令较长占用多行而造成误解。
- 本文档使用 Ubuntu Linux 系统作为操作系统。如果您的 PC 尚未安装 Linux 系统，请参考附录 1 安装 Ubuntu Linux 系统章节的内容。

3.1 嵌入式 Linux 系统结构

下图是嵌入式 Linux 系统的结构示意图：

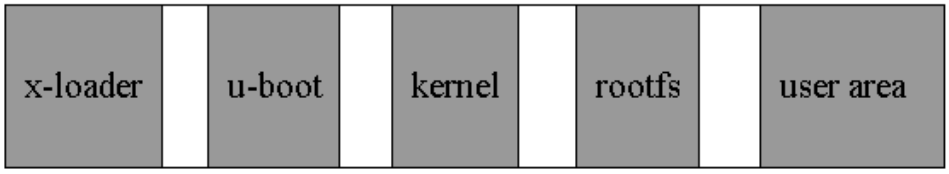


图 3-1 嵌入式 Linux 系统结构

- **x-loader**：一级引导程序；系统通电后，该程序由 CPU 内部的 ROM 存储器被复制到 RAM 存储器中执行，对 CPU 进行初始化并将 u-boot 复制到 RAM 中，然后将控制权交个 u-boot。
- **u-boot**：二级引导程序；该程序用于和用户进行交互，提供映像更新和引导内核等功能。
- **kernel**：2.6.32 版内核；针对 SBC8140 定制。
- **roofs**：开源的 ubifs 文件系统；适用于嵌入式系统。

3.2 软件特性

表 3-1 软件特性

名称		描述	代码类型
BIOS	x-loader	NAND / ONENAND	提供源代码
		MMC/SD	提供源代码
		FAT	提供源代码
	u-boot	NAND / ONENAND	提供源代码
		MMC/SD	提供源代码
		FAT	提供源代码
		NET	提供源代码
Kernel	Linux-2.6.x	支持 ROM/CRAM/EXT2/EXT3/FAT/NFS/ JFFS2/UBIFS 等多种文件系统	提供源代码
Device Driver	serial	串口驱动	提供源代码
	rtc	硬件时钟驱动	提供源代码
	net	10/100M 以太网卡 DM9000 驱动	提供源代码
	flash	nand flash 驱动(支持 nand boot)	提供源代码
	lcd	TFT LCD 驱动	提供源代码
	touch screen	触摸屏控制器 ads7846 驱动	提供源代码
	mmc/sd	mmc/sd 控制器驱动	提供源代码
	usb otg	usb otg 2.0 驱动（目前只支持 usb device 模式）	提供源代码
	usb ehci	usb ehci 驱动	提供源代码
	VGA	支持 VGA 信号输出	提供源代码
	audio	声卡驱动(支持录/放音)	提供源代码
	camera	摄像头驱动	提供源代码
	Key	按键驱动	提供源代码
	led	led 灯驱动	提供源代码
Demo	Android	android 2.2 系统	提供源代码
	DVSDK	DVSDK 4_00_00_22 系统	提供源代码

3.3 系统开发流程

本小节将从搭建开发环境开始到系统定制来大致介绍系统开发的整个流程。

3.3.1 搭建开发环境

1） 安装交叉编译工具；

将产品附带的光盘放入 PC 的光盘驱动器，Ubuntu 会自动将其挂载到 /media/cdrom 目录下，然后在 Ubuntu 的终端窗口中执行以下命令来将 /media/cdrom/linux/tools 目录下的交叉编译工具解压到$HOME 目录下；

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

2） 复制更多工具；

继续执行以下命令来将源代码编译过程中需要使用的工具从 linux/tools 目录下复制到$HOME/tools 目录下；

- **mkdir $HOME/tools**
- **cp /media/cdrom/linux/tools/mkimage $HOME/tools**
- **cp /media/cdrom/linux/tools/signGP $HOME/tools**
- **cp /media/cdrom/linux/tools/mkfs.ubifs $HOME/tools**
- **cp /media/cdrom/linux/tools/ubinize $HOME/tools**
- **cp /media/cdrom/linux/tools/ubinize.cfg $HOME/tools**

3） 添加环境变量；

执行以下命令来将之前安装的工具添加到临时环境变量中；

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

注意：

- 您可以将添加环境变量的命令复制到用户目录下的.bashrc 文件中，以便让系统启动时自动添加环境变量。
- 通过 echo $PATH 命令可以查看路径。

3.3.2 系统编译

1） 解压源代码；

执行以下命令来将光盘内 linux/source 目录下的源代码解压到 Ubuntu 系统中；

- **mkdir $HOME/work**
- **cd $HOME/work**

- **tar xvf /media/cdrom/linux/source/x-loader-03.00.02.07.tar.bz2**
- **tar xvf /media/cdrom/linux/source/u-boot-03.00.02.07.tar.bz2**
- **tar xvf /media/cdrom/linux/source/linux-2.6.32-sbc8140.tar.bz2**
- **sudo tar xvf /media/cdrom/linux/source/rootfs.tar.bz2**
- **tar xvf /media/cdrom/linux/demo/Android/source/rowboat-android-froyo-sbc8140.tar.bz2**

执行完成后在当前目录下会生成 x-loader-03.00.02.07、u-boot-03.00.02.07、linux-2.6.32-sbc8140、rootfs 和 rowboat-android-froyo-sbc8140 目录。

2） 编译一级启动代码；

A. 执行以下命令来为 SD 卡启动方式编译一级启动代码；

- **cd x-loader-03.00.02.07**
- **make distclean**
- **make omap3sbc8140_config**
- **make**
- **signGP x-load.bin**
- **mv x-load.bin.ift MLO**

执行完成后在当前目录下会生成一个 MLO 文件。

B. 执行以下命令来为 NAND Flash 启动方式编译一级启动代码；

- **cd x-loader-03.00.02.07**
- **vi include/configs/omap3sbc8140.h** → 在 omap3sbc8100_plus.h 文件中注释以下行**// #define CONFIG_MMC 1**
- **make distclean**
- **make omap3sbc8140_config**
- **make**
- **signGP x-load.bin**
- **mv x-load.bin.ift x-load.bin.ift_for_NAND**

执行完成后，当前目录下会生成一个 x-load.bin.ift_for_NAND 文件。

3） 编译二级启动代码；

执行以下命令编译二级启动代码；

- **cd u-boot-03.00.02.07**
- **make distclean**
- **make omap3_sbc8140_config**

- **make**

执行完成后，当前目录下会生成一个 u-boot.bin 文件。

4） 编译内核；

A. 执行以下命令编译 Linux 内核：

- **cd linux-2.6.32-sbc8140**
- **make distclean**
- **make omap3_sbc8140_defconfig**
- **make uImage**

执行完后在 arch/arm/boot 目录下生成 uImage 系统映像文件

B. 执行以下命令编译 Android 内核：

- **cd linux-2.6.32-sbc8140**
- **make distclean**
- **make omap3_sbc8140_android_defconfig**
- **make uImage**

执行完后在 arch/arm/boot 目录下生成 uImage 系统映像文件

5） 生成文件系统；

A. 制作 Ramdisk 文件系统；

请参考 http://www.elinux.org/DevKit8600_FAQ 页面的内容。

B. 执行以下命令来制作 UBI 文件；

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

执行完成后，在当前目录下会生成 ubi.img 文件。

6） Android 系统编译

A. 执行以下命令来编译 Android 系统；

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

B. 编译完后，输入下述指令，开始制作 ubi 文件系统：

- **sudo ./build_ubi.sh**

在temp/下即可找到ubi.img。

在编译 android 前，必须先编译 **linux-2.6.32-sbc8140** 内核。

3.3.3 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。

1） 进入配置菜单；

默认配置文件保存在 linux-2.6.32-SBC8140/arch/arm/configs/omap3_sbc8140_defconfig 目录下；请执行以下命令来进入系统配置菜单；

- **cd linux-2.6.32-sbc8140**
- **cp arch/arm/configs/omap3_sbc8140_defconfig .config**
- **make menuconfig**

注意：

如果在执行 make menuconfig 命令时出错，可能是因为 Ubuntu 系统缺少 ncurses 字符图形库。请执行命令 sudo apt-get install ncurses-dev 来安装该库文件。

2） 配置定制选项；

进入配置菜单后根据定制要求进行修改，例如进入 Device Drivers > USB support > USB Gadget Support > USB Gadget Drivers 子菜单，如下图所示；

```
--- USB Gadget Support
[ ]   Debugging messages (DEVELOPMENT)
[ ]   Debugging information files (DEVELOPMENT)
[ ]   Debugging information files in debugfs (DEVELOPMENT)
(2)   Maximum VBUS Power usage (2-500 mA)
      USB Peripheral Controller (Inventra HDRC USB Peripheral (TI, ADI, ...))  --->
<M>   USB Gadget Drivers
< >     Gadget Zero (DEVELOPMENT)
< >     Audio Gadget (EXPERIMENTAL)
<M>     Ethernet Gadget (with CDC Ethernet support)
[*]       RNDIS support
[ ]       Ethernet Emulation Model (EEM) support
< >     Gadget Filesystem (EXPERIMENTAL)
<M>     File-backed Storage Gadget
[*]       File-backed Storage Gadget testing version
< >     Mass Storage Gadget
< >     Serial Gadget (with CDC ACM and CDC OBEX support)
< >     MIDI Gadget (EXPERIMENTAL)
< >     Printer Gadget
< >     CDC Composite Device (Ethernet and ACM)
< >     Multifunction Composite Gadget (EXPERIMENTAL)
```

图 3-2 USB Gadget Drivers 子菜单

将 **File-backed Storage Gadget** 设置为 **M**，然后退出并保存配置。

3） 执行以下命令来编译内核；

- **make uImage**
- **make modules**

执行完成后，在 arch/arm/boot 目录和 drivers/usb/gadget 目录下会分别生成内核映像 uImage 和模块文件 g_file_storage.ko。

3.4 驱动介绍

本章节将对 Linux 系统中所需要的各种驱动进行介绍，包括 NAND Flash 驱动、SD/MMC 驱动、显示子系统驱动、视频采集驱动和音频输入输出驱动等。

下表列出了所有驱动的源代码路径；

表 3-2 驱动源代码

名称		描述	路径
BIOS	x-loader	ONENAND	x-loader-03.00.02.07/drivers/onenand.c
		NAND	x-loader-03.00.02.07/drivers/k9f1g08r0a.c
		MMC/SD	x-loader-03.00.02.07/cpu/omap3/mmc.c
		FAT	x-loader-03.00.02.07/fs/fat/
	u-boot	NAND	u-boot-03.00.02.07/drivers/mtd/nand/
		ONENAND	u-boot-03.00.02.07/drivers/mtd/onenand/
		MMC/SD	u-boot-03.00.02.07/drivers/mmc
		FAT	u-boot-03.00.02.07/fs/fat/
		NET	u-boot-03.00.02.07/drivers/net/dm9000x.c
Kernel	Linux-2.6.x	支持 ROM/CRAM/EXT2/EXT3/ FAT/NFS/JFFS2/UBIFS 等多种文件系统	linux-2.6.32-sbc8140/fs/
Device Driver	serial	串口驱动	linux-2.6.32-sbc8140/drivers/serial/8250.c
	rtc	硬件时钟驱动	linux-2.6.32-sbc8140/drivers/rtc/rtc-twl.c
	net	10/100M 以太网卡 DM9000 驱动	linux-2.6.32-sbc8140/drivers/net/dm9000.c
	flash	nand flash 驱动(支持 nand boot)	linux-2.6.32-sbc8140/drivers/mtd/nand/omap2.c
	lcd	TFT LCD 驱动	linux-2.6.32-sbc8140/drivers/video/omap2/omapfb/omapfb-main.c

名称		描述	路径
			linux-2.6.32-sbc8140/drivers/video/omap2/displays/panel-omap3-sbc8140.c
	touch screen	触摸屏控制器 ads7846 驱动	linux-2.6.32-sbc8140/drivers/input/touchscreen/ads7846.c
	mmc/sd	mmc/sd 控制器驱动	linux-2.6.32-sbc8140/drivers/mmc/host/omap_hsmmc.c
	usb otg	usb otg 2.0 驱动（目前只支持 usb device 模式）	linux-2.6.32-sbc8140/drivers/usb/otg/twl4030-usb.c
	usb ehci	usb ehci 驱动	linux-2.6.32-sbc8140/drivers/usb/host/ehci-hcd.c
	VGA	支持 VGA 信号输出	linux-2.6.32-sbc8140/drivers/i2c/chips/ch7033.c
	audio	声卡驱动(支持录/放音)	linux-2.6.32-sbc8140/sound/soc/omap/omap3sbc8140.c
			linux-2.6.32-sbc8140/sound/soc/codecs/twl4030.c
	camera	摄像头驱动	Digital: linux-2.6.32-sbc8140/drivers/media/video/omap34xxcam.c
			Catolog: linux-2.6.32-sbc8140/drivers/media/video/tvp514x-int.c
	Keypad	按键驱动	linux-2.6.32-sbc8140/drivers/input/keyboard/gpio_keys.c
	LED	led 灯驱动	linux-2.6.32-sbc8140/drivers/leds/leds-gpio.c

3.4.1 NAND Flash 驱动

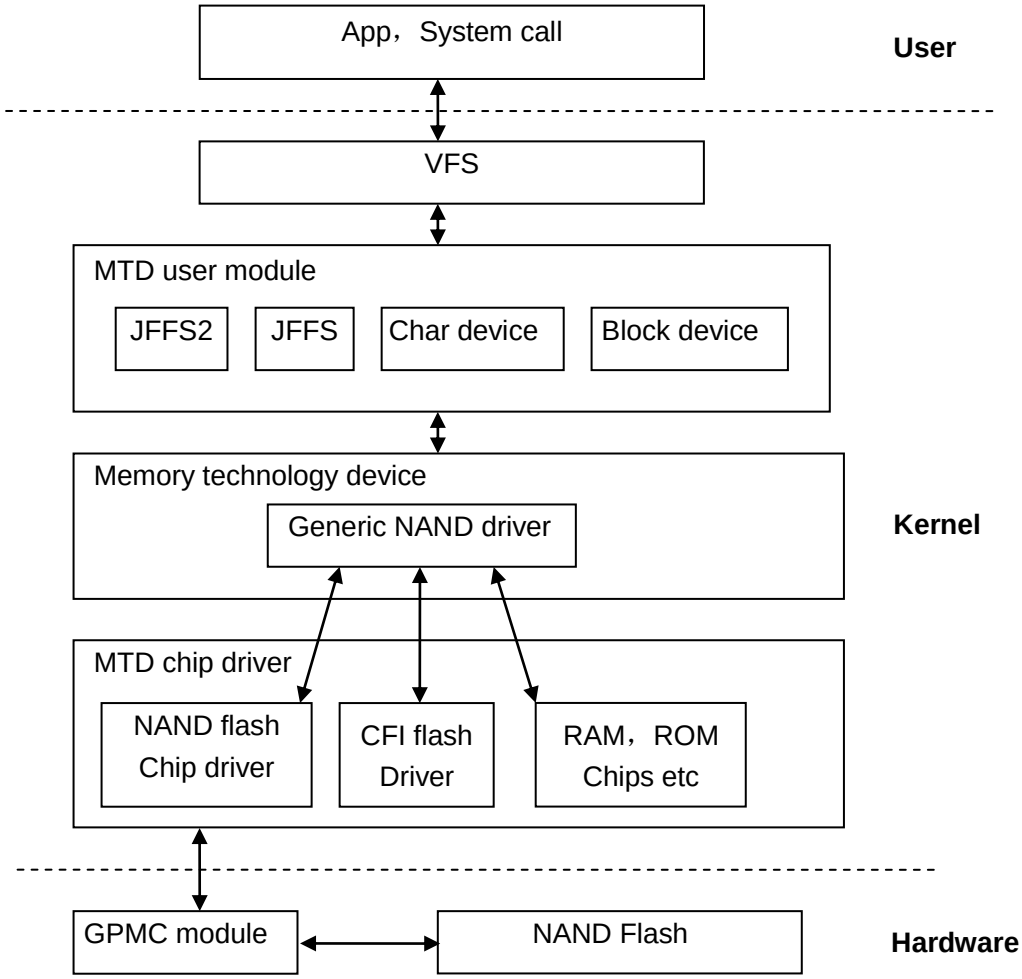


图 3-3 NAND Flash 工作原理

NAND flash 作为块设备使用，并在其中建立了文件系统。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，用户需要经过以下流程访问 NAND Flash；

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

表 3-3 驱动参考文件

参考文件	linux-2.6.32-sbc8140/drivers/mtd/nand/
	linux-2.6.32-sbc8140/drivers/mtd/nand/omap2.c

3.4.2 SD/MMC 驱动

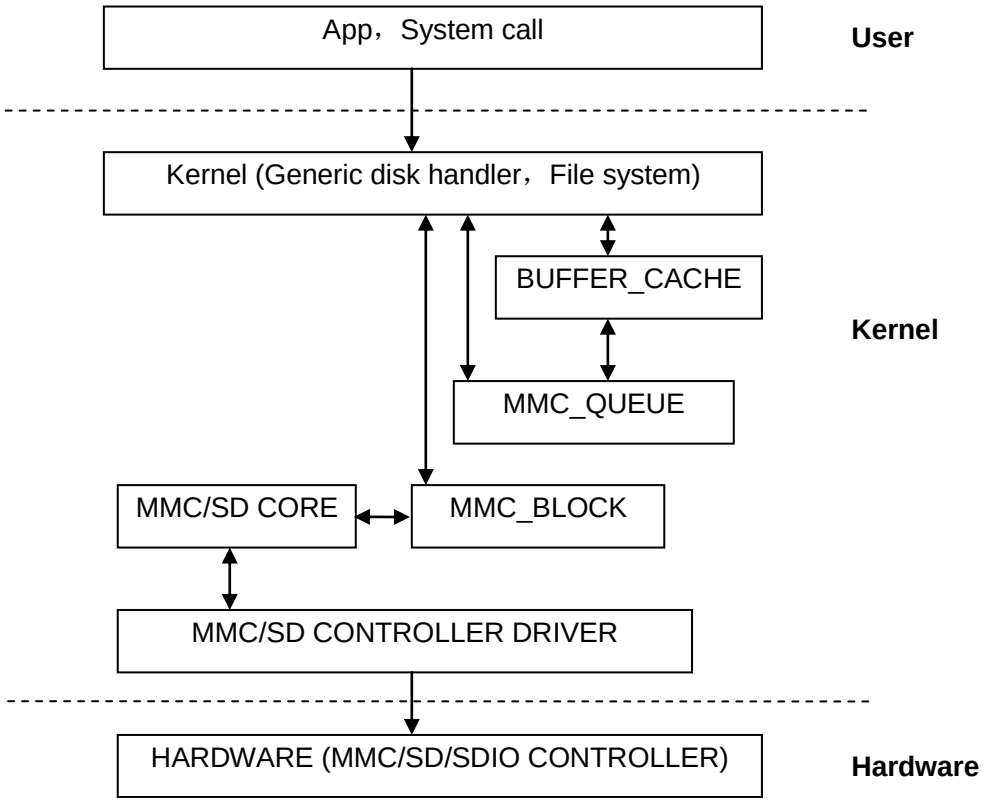


图 3-4 SD/MMC 工作原理

Linux 系统下 SD/MMC 卡驱动主要分为 SD/MMC core、mmc_block、mmc_queue 和 SD/MMC driver 四大部分：

- SD/MMC core 实现 SD/MMC 卡操作中与结构无关的核心代码；
- mmc_block 实现 SD/MMC 卡作为块设备使用时的驱动结构；
- mmc_queue 实现请求队列的管理；
- SD/MMC driver 实现具体的控制器驱动；

表 3-4 驱动参考文件

参考文件	linux-2.6.32-sbc8140/drivers/mmc/
	linux-2.6.32-sbc8140/drivers/mmc/host/omap_hsmmc.c

3.4.3 显示子系统驱动

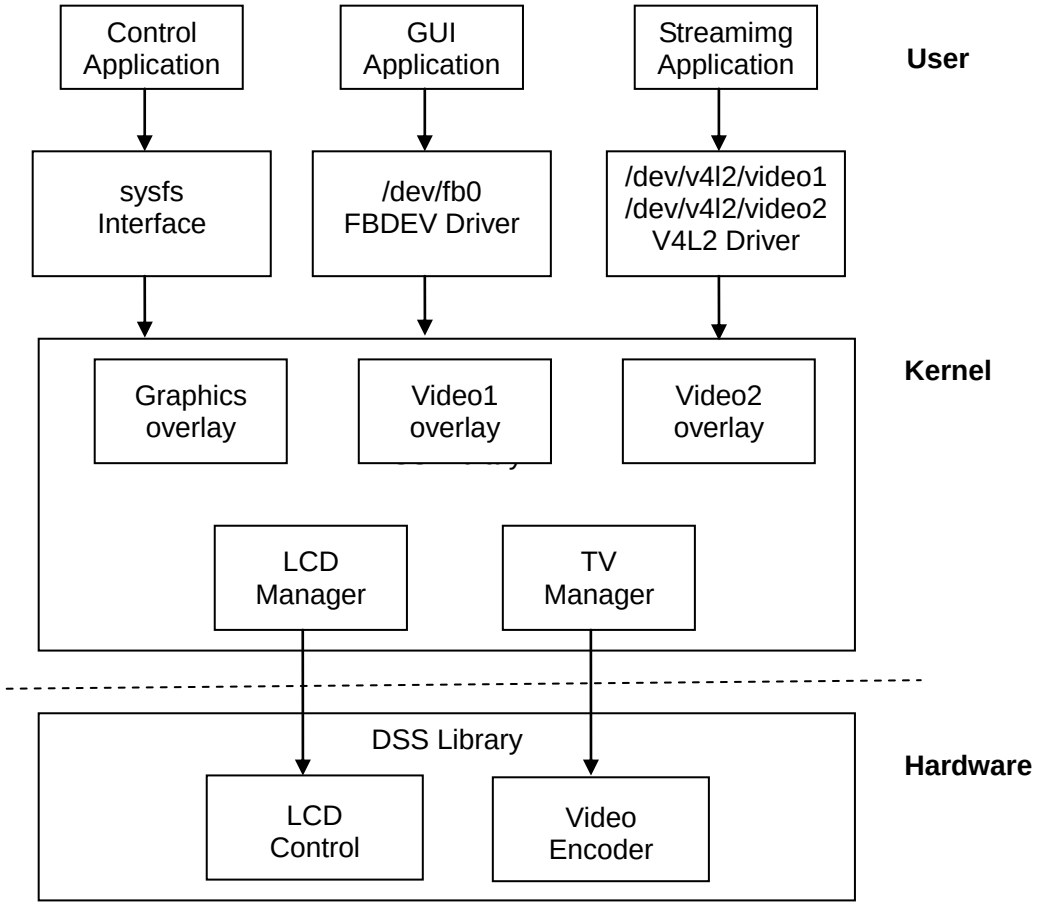


图 3-5 显示子系统工作原理

显示子系统硬件部分由一个 graphics 通道，两个 video 通道和两个 overlay 管理单元所组成，其中一个 overlay 管理单元负责数字接口，另一个 overlay 管理单元负责模拟接口。数字接口负责管理 LCD 的输出，模拟接口负责管理 TV 输出。

显示驱动的主要功能是提供上层应用层调用的接口，并管理显示子系统各个硬件部分。

表 3-5 驱动参考文件

参考文件	linux-2.6.32-sbc8140/drivers/video/omap2/
	linux-2.6.32-sbc8140/drivers/video/omap2/omapfb/omapfb-main.c
	linux-2.6.32-sbc8140/drivers/video/omap2/displays/panel-omap3-sbc8140.c

3.4.4 视频采集驱动

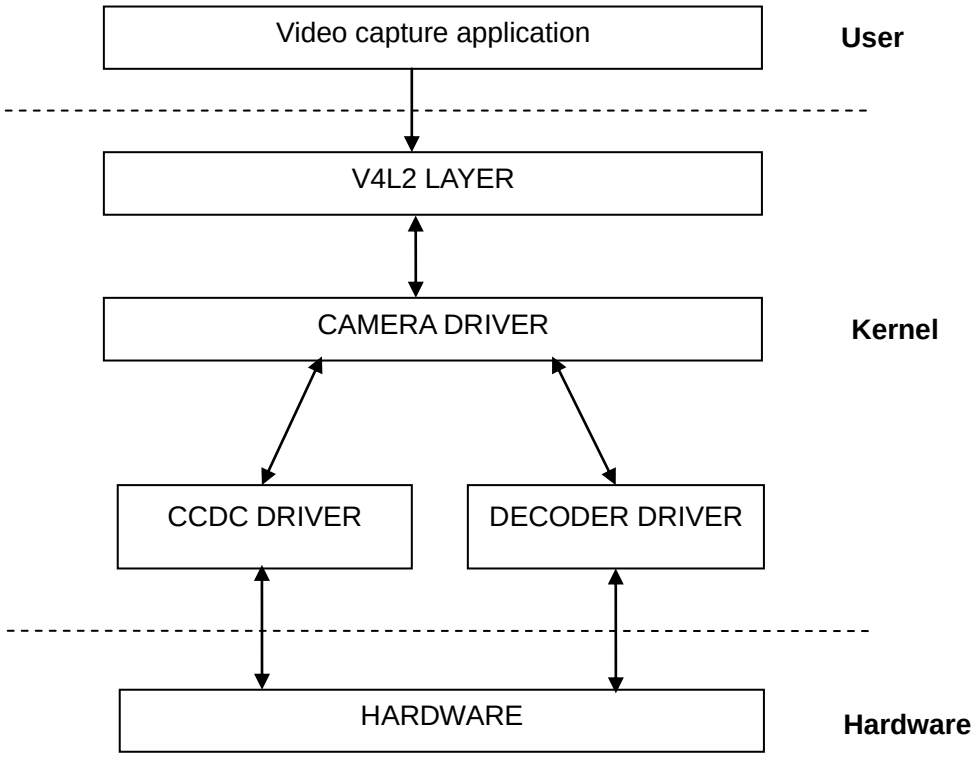


图 3-6 视频采集工作原理

- V4L2 子系统

  Linux 的 V4L2 子系统主要作为访问摄像头驱动的中间层支持。基于摄像头的上层应用可以通过 V4L2 的 API 去访问摄像头驱动。Linux 2.6 内核的 V4L2 子系统，就是基于 V4L2 规范的标准进行设计的。

- 视频缓冲库

  Video Buffer Library 是 V4L2 自带的一部分，它提供辅助的功能，通过队列的方式有效地管理视频缓冲。

- 摄像头驱动

  摄像头驱动允许通过外部解码器捕获视频。摄像头驱动以 master device 的方式，注册到 V4L2 层。任何以 slave device 方式添加到 V4L2 层上的解码器驱动都通过新的 V4L2 master-slave 接口与摄像头控制器驱动关联。目前驱动只实现一个摄像头控制器关联一个解码器设备。

- 解码器驱动

  解码驱动必须遵循 V4L2 master-slave 接口标准，以 slave device 方式注册到 V4L2 层。更换解码器只需要重新编写解码器驱动，而不需要修改摄像头驱动，每一个解码驱动都会提供一套 IOCTLs 接口供摄像头驱动调用。

- CCDC 库

  CCDC 作为数据输入的硬件模块通过并行接口从 sensor/decoder 获取数据。CCDC 库提供配置 CCDC 模组的 API，供摄像头驱动调用。

表 3-6 驱动参考文件

参考文件	linux-2.6.32-sbc8140/drivers/media/video/
	linux-2.6.32-sbc8140/drivers/media/video/omap34xxcam.c
	linux-2.6.32-sbc8140/drivers/media/video/tvp514x-int.c

3.4.5 音频输入输出驱动

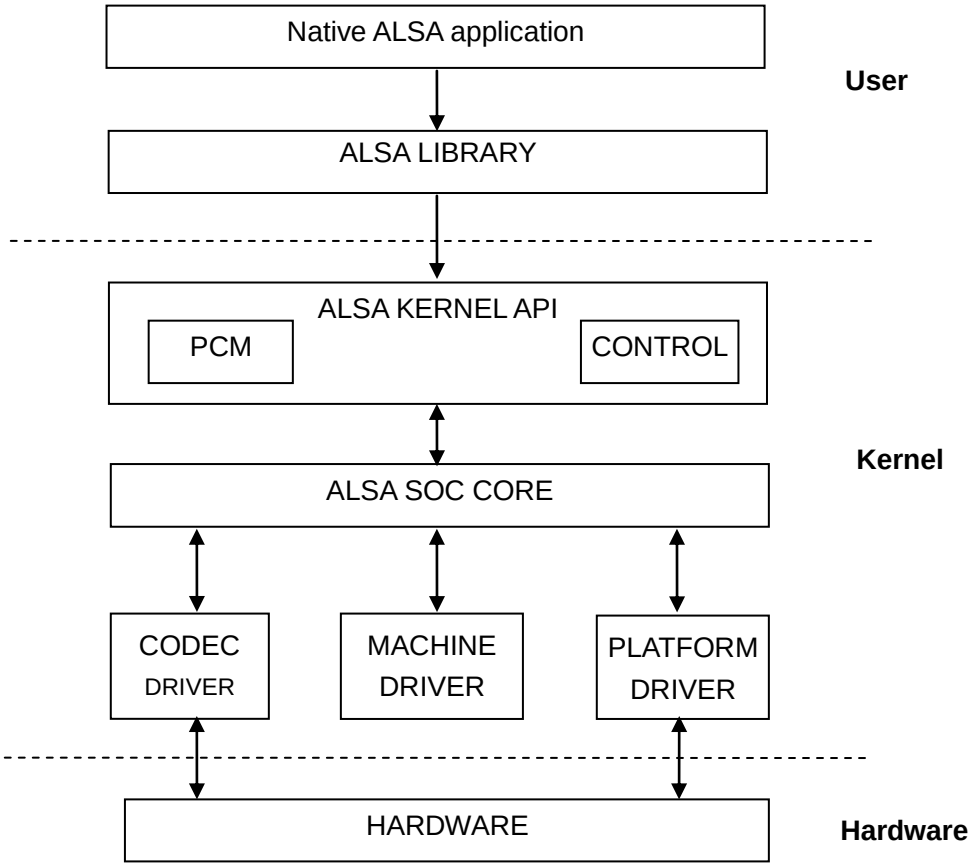


图 3-7 音频输入输出工作原理

ASoC 嵌入式音频系统基本分为以下三部分；

- 编解码器驱动；

  编解码器驱动与平台无关，包括音频控制、音频接口能力、编解码动态音频电源管理和编解码器 IO 函数；

- 平台驱动；

  平台驱动包括平台相关的音频 DMA 引擎和音频接口驱动（例如 I2S、AC97 和 PCM）；

- Machine 驱动；

  Machine 驱动管理任何与 machine 相关的控制和音频事件，即在回放开始启动放大器；

表 3-7 驱动参考文件

参考文件	linux-2.6.32-sbc8140/sound/soc/
	linux-2.6.32-sbc8140/sound/soc/omap/omap3sbc8140.c
	linux-2.6.32-sbc8140/sound/soc/codecs/twl4030.c

3.5 驱动开发

本章节将通过 GPIO_Keys 驱动开发和 GPIO_LEDs 驱动开发实例来介绍如何进行驱动程序的开发。

3.5.1 GPIO_Keys 驱动

1） 设备定义；

源代码文件 board-omap3sbc8140.c 保存在 linux-2.6.32-sbc8140/arch/arm/mach-omap2/目录下；

表 3-8 设备定义源代码

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

```
        },
        {
                .code                   = KEY_ESC,
                .gpio                   = 29,
                .desc                   = "back",
                .active_low             = true,
        },
};

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

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

配置 gpio 26 为“menu”键，返回键值“KEY_F1”，低电平触发；gpio 29 为“back”键，返回键值“KEY_ESC”，低电平触发。

2） GPIO pinmux 配置；

文件 sbc8140.h 保存在 u-boot-03.00.02.07/board/timll/sbc8140/目录下；

表 3-9 GPIO pinmux 配置

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

配置 GPIO 26 和 GPIO 29 为 M4（gpio 模式）和 IEN（允许输入）

3） 驱动设计；

源代码文件 gpio_keys.c 保存在 linux-2.6.32-sbc8140/drivers/input/keyboard/ 目录下；

A. 调用 platform_driver_register 来注册 gpio_keys 驱动；

表 3-10 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,
#ifdef CONFIG_PM
                .pm     = &gpio_keys_pm_ops,
#endif
        }
};

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

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

module_init(gpio_keys_init);
module_exit(gpio_keys_exit);

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

B. 调用 input_register_device 注册 input 驱动；

表 3-11 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(dev, bdata, button);
                if (error)
                        goto fail2;

                if (button->wakeup)
                        wakeup = 1;

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

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

C. 申请 gpio，配置 gpio 为输入，注册 gpio 中断；

表 3-12 注册 gpio 中断

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

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

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

```
        }

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

        irq = gpio_to_irq(button->gpio);
        if (irq < 0) {
                error = irq;
                dev_err(dev, "Unable to get irq number for GPIO %d, error %d\n",
                        button->gpio, error);
                goto fail3;
        }

        error = request_irq(irq, gpio_keys_isr,
                 IRQF_SHARED |
                        IRQF_TRIGGER_RISING | IRQF_TRIGGER_FALLING,
                        desc, bdata);
        if (error) {
                dev_err(dev, "Unable to claim irq %d; error %d\n",
                        irq, error);
                goto fail3;
        }

        return 0;

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

D. 中断处理；按键按下时产生中断，汇报键值：

表 3-13 中断处理

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

```
…
}

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

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

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

3.5.2 GPIO_LEDs 驱动

1） 设备定义；

源代码文件 board-omap3sbc8140.c 保存在 linux-2.6.32-sbc8140/arch/arm/mach-omap2/目录下；

表 3-14 设备定义源代码

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

```
                .name                   = "led2",
                .gpio                   = 138,          /* gets replaced */
                .active_low             = true,
        },
        {
                .name                   = "led3",
                .gpio                   = 139,          /* gets replaced */
                .active_low             = true,
        },

};
```

配置 GPIO 136 对应“led0”（系统心跳灯）、GPIO137 对应“led1”、GPIO138 对应“led2”、GPIO139 对应“led3”，均为低电平有效。

2） GPIO pinmux 配置；

文件 sbc8140.h 保存在 u-boot-03.00.02.07/board/timll/sbc8140/目录下；

表 3-15 GPIO pinmux 配置

```
/*
 * IEN  - Input Enable
 * IDIS - Input Disable
 * PTD  - Pull type Down
 * PTU  - Pull type Up
 * DIS  - Pull type selection is inactive
 * EN   - Pull type selection is active
 * M0   - Mode 0
 * The commented string gives the final mux configuration for that pin
 */
        MUX_VAL(CP(MMC2_DAT4),          (IDIS | PTD | DIS | M4)) /*GPIO_136*/\
        MUX_VAL(CP(MMC2_DAT5),          (IDIS | PTD | DIS | M4)) /*GPIO_137*/\
        MUX_VAL(CP(MMC2_DAT6),          (IDIS | PTD | DIS | M4)) /*GPIO_138*/\
        MUX_VAL(CP(MMC2_DAT7),          (IDIS | PTU | DIS | M4)) /*GPIO_139*/\
```

配置 GPIO136、GPIO137、GPIO138、GPIO139 为 M4(gpio 模式)和 IDIS（不允许输入）

3） 驱动设计；

源代码文件 leds-gpio.c 保存在 linux-2.6.32-sbc8140/drivers/leds/目录下；

A. 调用 platform_driver_register 注册 gpio_leds 驱动；

表 3-16 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,
        },
};

static int __init gpio_led_init(void)
{
        int ret;

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

        return ret;
}

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

module_init(gpio_led_init);
module_exit(gpio_led_exit);

MODULE_AUTHOR("Raphael Assenat <raph@8d.com>, Trent Piepho
```

```
<tpiepho@freescale.com>");
MODULE_DESCRIPTION("GPIO LED driver");
MODULE_LICENSE("GPL");
```

B. 申请 gpio，调用 led_classdev_register 注册 led_classdev 驱动；

表 3-17 注册 led_classdev 驱动

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

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

C. 通过访问/sys/class/leds/xxx/ brightness 文件，调用 gpio_led_set 函数来控制 LED 灯的状态；

表 3-18 控制 LED 状态

```
static void gpio_led_set(struct led_classdev *led_cdev,
        enum led_brightness value)
{
```

```
...
                gpio_set_value(led_dat->gpio, level);
}
```

3.6 系统更新

本章节将对 SD 卡和 NAND Flash 上的系统更新过程进行简要介绍；

3.6.1 SD 卡系统更新

1） 格式化 SD 卡；

请使用 HP USB Disk Storage Format Tool 2.0.6 格式化 SD 卡；（请访问 http://www.embedinfo.com/english/download/SP27213.exe 来下载该格式化软件）；软件界面如下；

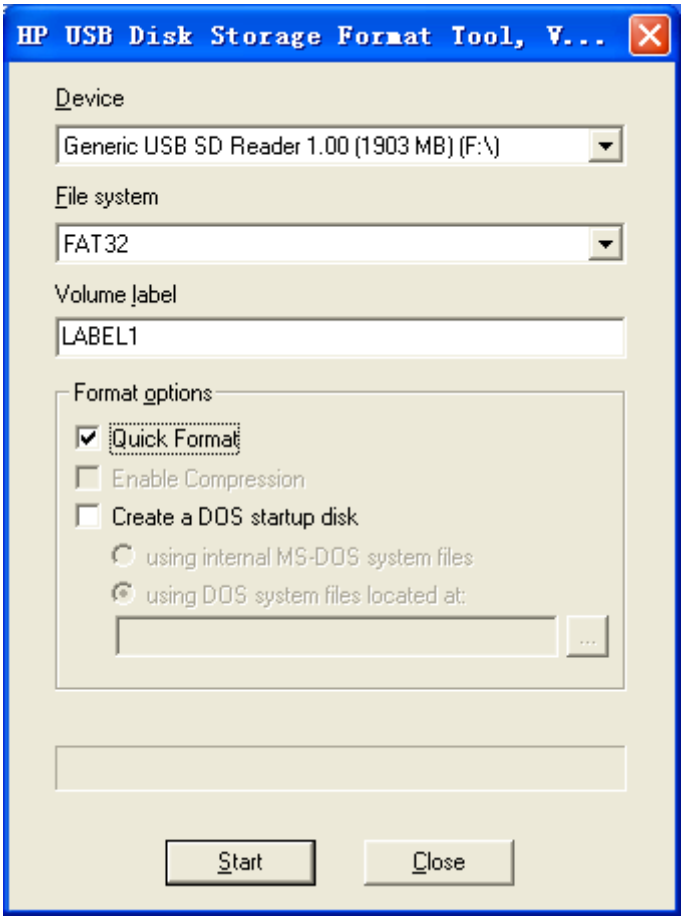


图 3-8 格式化 SD 卡

在 **File system** 下拉菜单中选择 FAT32 并单击 **Start** 按钮来进行格式化。

注意：

HP USB Disk Storage Format Tool 会清除 TF 存储卡的分区。如果需要保留分区，请使用 Windows 系统的格式化功能。

2） 更新映像文件；

将光盘 linux\image 目录下的所有文件复制到 SD 卡上，然后将 SD 卡插到 SBC8140 并启动系统，串口信息显示如下：

表 3-19 更新映像文件

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

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



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

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

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

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

```
2548700 bytes read
reading ramdisk.gz

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

Starting kernel ...

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

```
dpll3_m2_clk rate change failed: -22
GPMC revision 5.0
IRQ: Found an INTC at 0xfa200000 (revision 4.0) with 96 interrupts
Total of 96 interrupts on 1 active controller
OMAP GPIO hardware version 2.5
OMAP clockevent source: GPTIMER12 at 32768 Hz
Console: colour dummy device 80x30
Calibrating delay loop... 480.01 BogoMIPS (lpj=1875968)
Mount-cache hash table entries: 512
CPU: Testing write buffer coherency: ok
regulator: core version 0.5
NET: Registered protocol family 16
Found NAND on CS0
Registering NAND on CS0
Target VDD1 OPP = 4, VDD2 OPP = 2
OMAP DMA hardware revision 5.0
bio: create slab <bio-0> at 0
SCSI subsystem initialized
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
i2c_omap i2c_omap.1: bus 1 rev4.0 at 2600 kHz
twl4030: PIH (irq 7) chaining IRQs 368..375
twl4030: power (irq 373) chaining IRQs 376..383
twl4030: gpio (irq 368) chaining IRQs 384..401
regulator: VUSB1V5: 1500 mV normal standby
regulator: VUSB1V8: 1800 mV normal standby
regulator: VUSB3V1: 3100 mV normal standby
twl4030_usb twl4030_usb: Initialized TWL4030 USB module
regulator: VMMC1: 1850 <--> 3150 mV normal standby
regulator: VDAC: 1800 mV normal standby
regulator: VPLL2: 1800 mV normal standby
regulator: VMMC2: 1850 <--> 3150 mV normal standby
regulator: VSIM: 1800 <--> 3000 mV normal standby
i2c_omap i2c_omap.2: bus 2 rev4.0 at 400 kHz
i2c_omap i2c_omap.3: bus 3 rev4.0 at 400 kHz
Switching to clocksource 32k_counter
musb_hdrc: version 6.0, musb-dma, otg (peripheral+host), debug=0
musb_hdrc: USB OTG mode controller at fa0ab000 using DMA, IRQ 92
NET: Registered protocol family 2
IP route cache hash table entries: 2048 (order: 1, 8192 bytes)
TCP established hash table entries: 8192 (order: 4, 65536 bytes)
TCP bind hash table entries: 8192 (order: 3, 32768 bytes)
```

```
TCP: Hash tables configured (established 8192 bind 8192)
TCP reno registered
UDP hash table entries: 256 (order: 0, 4096 bytes)
UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
NET: Registered protocol family 1
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
RPC: Registered tcp NFSv4.1 backchannel transport module.
Trying to unpack rootfs image as initramfs...
rootfs image is not initramfs (no cpio magic); looks like an initrd
Freeing initrd memory: 65536K
omap-iommu omap-iommu.0: isp registered
NetWinder Floating Point Emulator V0.97 (double precision)
ashmem: initialized
VFS: Disk quotas dquot_6.5.2
Dquot-cache hash table entries: 1024 (order 0, 4096 bytes)
msgmni has been set to 473
alg: No test for stdrng (krng)
io scheduler noop registered
io scheduler deadline registered
io scheduler cfq registered (default)
OMAP DSS rev 2.0
OMAP DISPC rev 3.0
OMAP VENC rev 2
Serial: 8250/16550 driver, 4 ports, IRQ sharing enabled
serial8250.0: ttyS0 at MMIO 0x4806a000 (irq = 72) is a ST16654
console [ttyS0] enabled
serial8250.1: ttyS1 at MMIO 0x4806c000 (irq = 73) is a ST16654
serial8250.2: ttyS2 at MMIO 0x49020000 (irq = 74) is a ST16654
serial8250.3: ttyS3 at MMIO 0x49042000 (irq = 80) is a ST16654
brd: module loaded
loop: module loaded
omap2-nand driver initializing
NAND device: Manufacturer ID: 0xad, Chip ID: 0xbc (Hynix NAND 512MiB 1,8V 16-bit)
cmdlinepart partition parsing not available
Creating 5 MTD partitions on "omap2-nand":
0x000000000000-0x000000080000 : "X-Loader"
0x000000080000-0x000000260000 : "U-Boot"
0x000000260000-0x000000280000 : "U-Boot Env"
0x000000280000-0x000000680000 : "Kernel"
0x000000680000-0x000020000000 : "File System"
PPP generic driver version 2.4.2
PPP Deflate Compression module registered
```

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

```
input: ADS7846 Touchscreen as /devices/platform/omap2_mcspi.1/spi1.0/input/input2
using rtc device, twl_rtc, for alarms
twl_rtc twl_rtc: rtc core: registered twl_rtc as rtc0
twl_rtc twl_rtc: Power up reset detected.
twl_rtc twl_rtc: Enabling TWL-RTC.
i2c /dev entries driver
ch7033 id:5e
Linux video capture interface: v2.00
tvp514x 2-005d: Registered to v4l2 master omap34xxcam!!
omap-iommu omap-iommu.0: isp: version 1.1
usbcore: registered new interface driver uvcvideo
USB Video Class driver (v0.1.0)
OMAP Watchdog Timer Rev 0x31: initial timeout 60 sec
Registered led device: led0
Registered led device: led1
Registered led device: led2
Registered led device: led3
Registered led device: led4
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
Advanced Linux Sound Architecture Driver Version 1.0.21.
usb 1-1: new high speed USB device using ehci-omap and address 2
No device for DAI omap-mcbsp-dai-0
No device for DAI omap-mcbsp-dai-1
No device for DAI omap-mcbsp-dai-2
No device for DAI omap-mcbsp-dai-3
No device for DAI omap-mcbsp-dai-4
OMAP3 SBC8140 SoC init
asoc: twl4030 <-> omap-mcbsp-dai-0 mapping ok
ALSA device list:
  #0: omap3sbc8140 (twl4030)
TCP cubic registered
NET: Registered protocol family 17
NET: Registered protocol family 15
Power Management for TI OMAP3.
Unable to set L3 frequency (400000000)
Switched to new clocking rate (Crystal/Core/MPU): 26.0/266/1000 MHz
IVA2 clocking rate: 800 MHz
SmartReflex driver initialized
VFP support v0.3: implementor 41 architecture 3 part 30 variant c rev 3
Console: switching to colour frame buffer device 60x34
regulator_init_complete: incomplete constraints, leaving VDVI on
regulator_init_complete: incomplete constraints, leaving VDAC on
```

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

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

```
udhcpc: sendto: Network is down
No lease, failing
done.
Tue Jan 27 08:47:00 UTC 2009

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

.-------.
|       |                        .-.
|   |   |-----.-----.-----.| |   .----..-----.-----.
|       |     | __  |   ---'| '--.|  .-'|     |     |
|   |   |  |  |     |---  ||  --'|  |  |  '  | | | |
'---'---'--'--'--.  |-----"----"--'   '-----'-'-'-'
                 -'  |
                 '---'

The Angstrom Distribution SBC8140 ttyS0

Angstrom 2008.1-test-20090127 SBC8140 ttyS0

SBC8140 login: root （输入 root 登录系统）
```

以上信息表示 Linux 系统已经成功地从 SD 卡启动。

注意：

- SBC8140 默认从 NAND Flash 启动；接通电源时按住 BOOT 键（CN11 键）可以从 SD 卡启动；
- 系统默认支持 4.3 寸屏幕，如果需要修改显示模式，请参考 3.7 显示模式配置章节的内容，然后在 u-boot 模式下输入 boot 继续启动；

3.6.2 NAND Flash 系统更新

Nand Flash 的映像更新需要借助 u-boot 来完成。不管 NAND Flash 中是否存在数据，都可以利用 SD 卡启动 u-boot 的方式对 NAND Flash 中的映像文件进行更新操作。

1） 用 HP USB Disk Storage Format Tool 2.0.6 将 SD 卡格式化为 FAT 或 FAT32 文件系统；

2） 将光盘目录 linux\image 下的 MLO、x-load.bin.ift_for_NAND、u-boot.bin、

flash-uboot.bin、uImage 和 ubi.img 文件复制到 SD 卡中；

3） 将 SD 卡插入 SBC8140 并接通电源，当串行接口信息进入读秒倒计时，在 PC 键盘上按下任意键进入 u-boot 模式，如下表所示；

表 3-20 进入 u-boot 模式

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

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



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

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

In:     serial
Out:    serial
Err:    serial
Die ID #3d1400029e3800000168682f07003018
Net:    dm9000
Hit any key to stop autoboot: 0（在这里按任意键进入 u-boot 命令行）
```

4） 输入 **run updatesys** 并按下 **Enter** 键开始更新系统，串行接口信息显示如下；

表 3-21 更新系统

```
OMAP3 SBC8140 # run updatesys

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

11648 bytes read
```

```
HW ECC selected

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

231872 bytes read
SW ECC selected

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

2548700 bytes read
SW ECC selected

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

12320768 bytes read
SW ECC selected

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

当底板上的 LED 指示灯开始流水灯闪烁时，表示更新完成；请拔出 SD 卡并重新启动系统即可。

注意：

📖 系统默认支持 4.3 寸屏幕，如果需要修改显示模式，请参考 3.7 显示模式配置章节的内容，然后在 u-boot 模式下输入 boot 继续启动；

3.7 显示模式配置

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。下面的内容将介绍如何针对 4.3 寸 LCD、7 寸 LCD、VGA 和 LVDS 显示模式进行配置。

在开始配置前，需要首先进入 u-boot 模式。请重新启动开发套件，然后在系统提示倒数读秒时按下 PC 键盘上的任意键进入 u-boot 模式，如下表所示；

表 3-22 启动信息

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

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



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

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

In:      serial
Out:     serial
Err:     serial
Die ID #3d1400029e3800000168682f07003018
Net:     dm9000
Hit any key to stop autoboot:    0 （在这里按任意键进入 u-boot 命令行）
```

1） 使用 4.3 寸 LCD 显示；

在 u-boot 模式下执行以下命令来配置 4.3 寸 LCD 显示模式；

OMAP3 SBC8140 # **setenv defaultdisplay lcd**

OMAP3 SBC8140 **# setenv dispmode 4.3inch_LCD**

OMAP3 SBC8140 # **saveenv**

2） 使用 7 寸 LCD 显示；

在 u-boot 模式下执行以下命令来配置 7 寸 LCD 显示模式；

OMAP3 SBC8140 # **setenv defaultdisplay lcd**

OMAP3 SBC8140 # **setenv dispmode 7inch_LCD**

OMAP3 SBC8140 # **saveenv**

3） 使用 VGA 显示；

在 u-boot 模式下执行以下命令来配置 VGA 显示模式；

OMAP3 SBC8140 # **setenv defaultdisplay lcd**

OMAP3 SBC8140 # **setenv dispmode VGA**

OMAP3 SBC8140 # **saveenv**

4） 使用 LVDS 显示；

在 u-boot 模式下执行以下命令来配置 LVDS 显示模式；

OMAP3 SBC8140 # **setenv defaultdisplay lcd**

OMAP3 SBC8140 # **setenv dispmode LVDS**

OMAP3 SBC8140 # **saveenv**

3.8 测试和演示

本章节将对 SBC8140 上各个设备进行测试，并且会针对 Android 系统和 DVSDK 系统进行演示。

注意：

🕮 以下测试过程均通过超级终端窗口输入命令。

3.8.1 LED 测试

1） 执行以下命令来测试 LED0；

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

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

2） 执行以下命令来测试 LED1；

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

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

3） 执行以下命令来测试 LED2；

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

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

4） 执行以下命令来测试 LED3；

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

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

执行每条命令后，相应的 LED 会点亮或者熄灭；

3.8.2 触摸屏测试

从 NAND Flash 启动 SBC8140，然后开始测试；

1） 执行以下命令来校准触摸屏；

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

按照提示触摸所有“+”符号来完成校准；

2） 执行以下命令来测试触摸屏；

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

按照提示在触摸屏上绘制点和线段来进行测试。

3.8.3 RTC 测试

SBC8140 开发套件带有硬件时钟，可保存并恢复系统时间；请通过以下步骤方法进行测试；

1） 执行以下命令将系统时间设置为 2011 年 8 月 8 日晚上 8 点正；

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

超级终端窗口显示信息如下；

表 3-23 设置系统时间

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

2） 执行以下命令将系统时钟写入 RTC；

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

3） 执行以下命令来读取 RTC；

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

超级终端窗口显示信息如下；

表 3-24 RTC 时间

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

以上信息显示系统时钟已经保存到硬件时钟里；

4） 重新启动系统，然后执行以下命令来恢复系统时钟；

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

超级终端窗口显示信息如下；

表 3-25 系统时钟

Mon Aug 8 20:01:01 2011 0.000000 seconds

以上信息显示系统时钟已经恢复为硬件时钟；

注意：

🕮 SBC8140 默认不提供 CR2032 电池，请自行购买。

3.8.4 SD 卡测试

1） 将 SD 插入 SBC8140 的 SD 卡接口，系统会自动将 SD 卡挂载到/media 目录下；请执行以下命令来查看 SD 卡设备名称；

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

超级终端窗口显示信息如下；

表 3-26 SD 卡设备名称

card	hdd	mmcblk0p1	ram	union
cf	mmc1	net	realroot	

2） 执行以下命令来查看 SD 卡的内容；

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

超级终端窗口显示信息如下；

表 3-27 SD 卡的内容

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

3.8.5 USB Device 测试

USB DEVICE 测试的目的是通过开发板的 miniUSB 接口与 PC 端的 USB 接口实现网络通信。

1） 系统启动完成后，使用 USB mini B to USB A 线缆连接 SBC8140 与 PC。此时 PC 需要安装 Linux USB Ethernet 驱动，详细的安装方法请参考附录 2 安装 Linux USB Ethernet/RNDIS Gadget 驱动的内容；

2） 执行以下命令来设置 SBC8140 的 IP 地址（以下 IP 地址仅为范例，您也可以设置为不同的 IP 地址）；

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

超级终端窗口显示信息如下；

表 3-28 配置 IP 地址

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

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

3） 在 PC 桌面上右键单击**网上邻居**，并选择**属性**进入网络连接窗口；窗口中会出现一个新增的本地连接图标，如下图所示；

图 3-9 新增本地连接

4） 右键单击新增的本地连接并选择**属性**，然后在弹出窗口中双击 **Internet 协议（TCP/IP）**选项，进入以下界面；

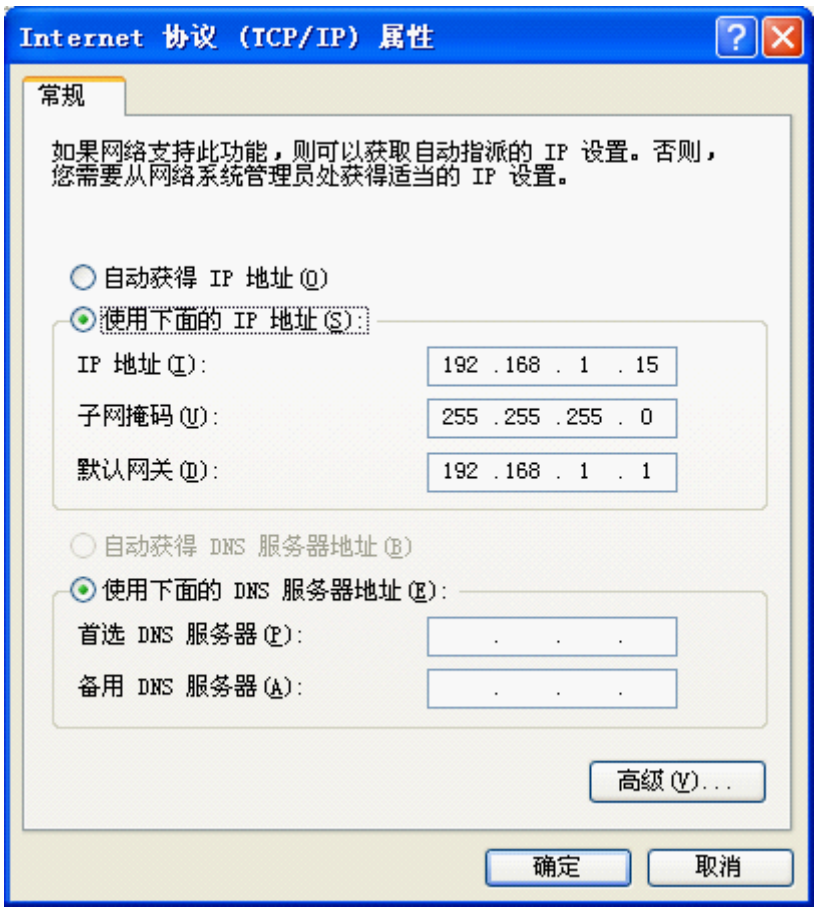


图 3-10 配置 IP 地址

将 USB 虚拟网卡的 IP 地址设置为与 SBC8140 相同的网段，然后单击**确定**；

5） 在超级终端中使用 ping 命令测试开发板是否设置成功；

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

表 3-29 测试信息

```
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
```

以上信息表示测试成功。

3.8.6 USB HOST 测试

1） 将一个 U 盘插入 SBC8140 的 USB 接口，系统会显示以下串口信息；

表 3-30 串口信息

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

2） 系统会自动将 U 盘挂载到/media 目录下；请执行以下命令查看 U 盘的设备名称；

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

超级终端窗口显示信息如下；

表 3-31 U 盘的内容

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

3.8.7 音频测试

SBC8140 上带音频输入/输出接口，支持录音和放音。文件系统内带 alsa-utils 音频播放和录制测试工具，用户通过以下步骤进行测试：

1） 将一只麦克风插入 SBC8140 上的 3.5mm 音频输入接口（红色接口），然后执行以下命令开始录音；

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

超级终端窗口显示信息如下；

表 3-32 开始录音

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

```
  format          : S16_LE
  subformat       : STD
  channels        : 2
  rate            : 44100
  exact rate      : 44100 (44100/1)
  msbits          : 16
  buffer_size     : 22052
  period_size     : 5513
  period_time     : 125011
  tstamp_mode     : NONE
  period_step     : 1
  avail_min       : 5513
  period_event : 0
  start_threshold     : 1
  stop_threshold      : 22052
  silence_threshold: 0
  silence_size : 0
  boundary        : 1445199872
  appl_ptr        : 0
  hw_ptr          : 0
```

2） 将一副耳机插入 SBC8140 上的 3.5mm 音频输出接口（绿色接口），然后执行以下命令来回放记录的音频；

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

超级终端窗口显示信息如下；

表 3-33 开始回放

```
Playing WAVE 'k' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Plug PCM: Hardware PCM card 0 'omap3evm' device 0 subdevice 0
Its setup is:
  stream          : PLAYBACK
  access          : RW_INTERLEAVED
  format          : S16_LE
  subformat       : STD
  channels        : 2
  rate            : 44100
  exact rate      : 44100 (44100/1)
  msbits          : 16
  buffer_size     : 22052
  period_size     : 5513
  period_time     : 125011
```

```
   tstamp_mode   : NONE
   period_step   : 1
   avail_min     : 5513
   period_event : 0
   start_threshold   : 22052
stop_threshold     : 22052
   silence_threshold: 0
   silence_size : 0
   boundary        : 1445199872
   appl_ptr        : 0
   hw_ptr          : 0
```

3.8.8 网络测试

1） 将 SBC8140 的 IP 地址设置为与您的 PC 处于同一网段，例如执行以下命令；

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

超级终端窗口显示信息如下；

表 3-34 配置 IP 地址

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

2） 执行以下命令来测试 SBC8140 和 PC 的网络通信；

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

超级终端窗口显示信息如下；

表 3-35 网络测试

```
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
```

以上信息表示网络通信正常。

3.8.9 摄像头测试

将摄像头模块 CAM8000-A（需要购买）、CCD 摄像头和 LCD 屏幕连接到 SBC8140 上，然后执行以下命令；

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

超级终端窗口显示信息如下；

表 3-36 摄像头测试

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

Capture: Opened Channel
Capture: Current Input: COMPOSITE
Capture: Current standard: PAL
Capture: Capable of streaming
Capture: Number of requested buffers = 3
Capture: Init done successfully



Display: Opened Channel
Display: Capable of streaming
Display: Number of requested buffers = 3
Display: Init done successfully

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

这时 LCD 屏幕上会显示 CCD 摄像头所采集到的图像。

3.8.10 CDMA8000-U 模块测试

请访问 http://www.timll.com/chinese/uploadFile/cdma8000.rar 下载该模块的用户手册，并按照手册进行测试。

注意：

CDMA8000-U 模块属于可选配件，用户可以根据实际的需求选择是否购买。

3.8.11 WCDMA8000-U 模块测试

请访问 http://www.timll.com/chinese/uploadFile/WCDMA8000.zip 下载该模块的用户手册，并按照手册进行测试。

注意：

📖 WCDMA8000-U 模块属于可选配件，用户可以根据实际的需求选择是否购买。

3.8.12 Android 系统演示

将 X:\linux\demo\Android\image 目录下（X 为光盘盘符）所有文件复制到 SD 卡并插入 SBC8140 的 SD 卡插槽中，然后按住 BOOT 键（CN11 键）并接通电源，超级终端窗口显示信息如下；

表 3-37 烧写信息

```
60


Texas Instruments X-Loader 1.47 (Apr 23 2012 - 09:09:16)
Starting X-loader on MMC
Reading boot sector

1154092 Bytes Read from MMC
Starting OS Bootloader from MMC...
Starting OS Bootloader...



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

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

In:     serial
Out:    serial
Err:    serial
```

```
SBC8140 xM Rev A
Die ID #259000029e38000001683b060d023028

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

11668 bytes read
HW ECC selected

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

1152640 bytes read
SW ECC selected

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

2573772 bytes read
SW ECC selected

NAND write: device 0 offset 0x280000, size 0x2745cc
 2574336 bytes written: OK
reading ubi.img
79036416 bytes read
SW ECC selected

NAND write: device 0 offset 0x680000, size 0x4b60000
 79036416 bytes written: OK
```

当底板上的 LED 指示灯闪烁时，表示烧写完成，请拔出 SD 卡，然后重新启动系统即可进入 Android 操作系统。

注意：

📖 系统默认支持 4.3 寸屏幕，如果需要修改显示模式，请参考 3.7 显示模式配置章节的内容。

3.8.13 DVSDK 系统演示

DVSDK(Digital Video Software Development Kit)是 TI 公司推出的一款软件，作用是建立 ARM 与 DSP 之间的联系。

应用程序运行在 ARM 端，由 ARM 处理 IO 接口和应用程序。ARM 使用由 Codec Engine 提供的 VISA APIs 接口来处理视频、图像、语音信号。然后 Codec Engine 通过 DSP/BIOS Link 以及 xDIAS 与 xDM 协议来与 DSP 端建立的 Codec Engine 服务器进行通信。DSP 将处理这些信号，处理的结果存放在与 ARM 共享的存储空间，从而 ARM 端也可以访问这些结果。

1） 将 SD 卡格式化为两个分区（请参考附录 3 制作 Linux 启动盘），并通过读卡器连接到 PC 上，然后在 Ubuntu Linux 系统中执行以下命令；

- **cp /media/cdrom/linux/demo/dvsdk/image/MLO /media/LABEL1**
- **cp /media/cdrom/linux/demo/dvsdk/image/u-boot.bin /media/LABEL1**
- **cp /media/cdrom/linux/demo/dvsdk/image/uImage /media/LABEL1/uImage**
- **rm -rf /media/LABEL2/***
- **sudo tar xvf /media/cdrom/linux/demo/dvsdk/image/dvsdk-dm37x-evm-rootfs.tar.bz2 -C /media/LABEL2**
- **sync**
- **umount /media/LABEL1**
- **umount /media/LABEL2**

2） 将 SD 卡插入 SBC8140 的 SD 卡插槽并接通电源，超级终端窗口显示信息如下；

表 3-38 登录系统

```
2548012 bytes read
## Booting kernel from Legacy Image at 80300000 ...
   Image Name:   Linux-2.6.32
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2547948 Bytes = 2.4 MiB
```

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

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

Arago 2010.07 dm37x-evm ttyS2

dm37x-evm login:root        （输入用户名 root 进入系统）
```

在 dm37x-evm login 提示信息出现后，输入用户名 **root** 登录系统；

3） DVSDK 文件系统中带有一些预装的应用程序；下面以运行 GStreamer pipelines 为例来演示 H.264 的解码；请在超级终端窗口中执行以下命令；

- root@dm37x-evm:**cd /usr/share/ti/gst/omap3530**
- root@dm37x-evm:/usr/share/ti/gst/omap3530# **./loadmodules.sh**
- root@dm37x-evm:/usr/share/ti/gst/omap3530# **gst-launch filesrc location=/usr/share/ti/data/videos/davincieffect_480p30.264 \! typefind ! TIViddec2 ! TIDmaiVideoSink -v**

超级终端窗口显示信息如下；

表 3-39 运行应用程序

```
Setting pipeline to PAUSED ...
/GstPipeline:pipeline0/GstTypeFindElement:typefindelement0.GstPad:src:       caps       =
video/x-h264
Pipeline is PREROLLING ...
/GstPipeline:pipeline0/GstTIViddec2:tividdec20.GstPad:sink: caps = video/x-h264
/GstPipeline:pipeline0/GstTIViddec2:tividdec20.GstPad:src:    caps    =    video/x-raw-yuv,
format=(fourcc)UYVY, framerate=(fraction)30000/1001, width=(int)720, height=(int)576
/GstPipeline:pipeline0/GstTIViddec2:tividdec20.GstPad:src:    caps    =    video/x-raw-yuv,
format=(fourcc)UYVY, framerate=(fraction)30000/1001, width=(int)720, height=(int)480
/GstPipeline:pipeline0/GstTIDmaiVideoSink:tidmaivideosink0.GstPad:sink:       caps       =
video/x-raw-yuv,    format=(fourcc)UYVY,    framerate=(fraction)30000/1001,    width=(int)720,
height=(int)480
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
```

这时 LCD 屏幕上会播放一段视频。

注意：

- 关于 DVSDK 的详细信息，请参考 TI 公司网站或者查看光盘目录 X:\linux\demo\dvsdk\source 下的 TMS320DM3730_Software_Developers_Guide.pdf 文件（X 为光盘盘符）；
- 系统默认支持 4.3 寸屏幕，如果需要修改显示模式，请参考 3.7 显示模式配置章节的内容。

3.9 应用程序开发

本章节将通过一个 LED 应用程序实例来介绍应用程序开发的一般流程。

1） 编写 led_acc.c 源码，用于控制开发板上的三个 LED 灯按累加器的方式闪烁；

表 3-40 LED 应用程序源代码

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

#define LED1 "/sys/class/leds/led1/brightness"
#define LED2 "/sys/class/leds/led2/brightness"
#define LED3 "/sys/class/leds/led3/brightness"

int main(int argc, char *argv[])
{
        int f_led1, f_led2, f_led3;
        unsigned char i = 0;
        unsigned char dat1, dat2, dat3;
        if((f_led1 = open(LED1, O_RDWR)) < 0){
                printf("error in open %s",LED1);
                return -1;
        }
        if((f_led2 = open(LED2, O_RDWR)) < 0){
                printf("error in open %s",LED2);
                return -1;
        }
        if((f_led3 = open(LED3, O_RDWR)) < 0){
```

```
                printf("error in open %s",LED3);
                return -1;
        }
        for(;;){
                i++;
                dat1 = i&0x1 ? '1':'0';
                dat2 = (i&0x2)>>1 ? '1':'0';
                dat3 = (i&0x4)>>2 ? '1':'0';
                write(f_led1, &dat1, sizeof(dat1));
                write(f_led2, &dat2, sizeof(dat2));
                write(f_led3, &dat3, sizeof(dat3));
                usleep(300000);
        }
}
```

2） 在 Ubuntu Linux 系统中执行以下命令来进行交叉编译；

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

3） 将编译好的文件下载到 SBC8140，然后进入 led_acc 文件所在目录并执行以下命令来运行 LED 应用程序；

- **./led_acc &**

第4章 WinCE 操作系统

本章主要介绍 Windows Embedded CE 6.0 R3 下 SBC8140 的系统与应用开发，以及光盘中提供的软件资源和软件特性、开发环境的搭建、以及如何编译工程与 BSP（板级支持包）等。

4.1 软件资源

SBC8140 附带的光盘中包含了丰富的软件资源，以下列表详细的列出了所有软件的位置（X 为光盘盘符）。

表 4-1 BSP 板级支持包

BSP	X:\WINCE600\bsp\mini8510.rar
	X:\WINCE600\bsp\COMMON_TI_V1.rar
	X:\WINCE600\bsp\dvsdk_wince_01_11_00_02.rar
	X:\WINCE600\SGX\wince_gfx_sgx_01_01_00_patch_01_setup.exe

表 4-2 Windows Embedded CE 6.0 R3 例子工程

例子工程	X:\WINCE600\prj\mini8510.rar

表 4-3 应用范例

应用范例	X:\WINCE600\app\GPIOAppDemo.rar

表 4-4 预编译映像文件

预编译映像	X:\WINCE600\image\	
	MLO	First bootloader for SD card boot
	XLDRNAND.nb0	First bootloader for nand boot
	EBOOTSD.nb0	Second bootloader for SD card boot
	EBOOTNAND.nb0	Second bootloader for nand boot
	NK.bin	WinCE runtime image

4.2 BSP 软件包信息

表 4-5 BSP 信息

类别	项目	代码类型
X-Loader（一级启动代码）	NAND	源代码
	NOR	源代码
	SD	源代码
EBOOT（二级启动代码）	NAND	源代码
	NOR	源代码
	SD	源代码
OAL	KILT(USB RNDIS)	源代码
	REBOOT	源代码
	Watchdog	源代码
	RTC	源代码
	System timer	源代码
	Interrupt controller	源代码
	Low power suspend	源代码
驱动程序	NLED driver	源代码
	GPIO/I2C/SPI/MCBSP driver	源代码
	Series port driver	源代码
	6X6 keyboard driver	源代码
	Audio driver	源代码
	NAND driver	源代码
	Display driver (LCD/DVI/VGA/S-Video/Composite Video)/ TOUCH driver	源代码
	SD/MMC/SDIO driver	源代码
	DM9000 network card driver	源代码
	USB OTG driver	源代码
	USB EHCI driver	源代码
	VRFB driver	源代码
	DSPLINKK/CMEMK driver	二进制代码
	AAC/MPEG2/MPEG4/H264 DSP Hardware decode fitler	二进制代码
	GPIO keyboard driver	源代码
	PWM(TPS65930) driver	源代码
	ADC(TPS65930) driver	源代码
	ONENAND driver	源代码
	Analog Camera driver	源代码
	Digital Camera driver	源代码
	DMA driver	源代码
	RTC driver	源代码
	Backlight driver	源代码

类别	项目	代码类型
	Battery driver	源代码
	Sleep / wakeup button driver	源代码
	DVFS/Smart Reflex	源代码
SDK	powerVR DDK & SDK	二进制代码 &源代码

4.3 系统开发流程

本章节将通过安装集成开发环境、解压/复制 BSP 和例子工程、一级 Sysgen 和 BSP 的编译来介绍 WinCE 环境下系统的开发流程。

4.3.1 安装集成开发环境

WinCE 集成开发环境的搭建需要在 Windows XP 或 Vista 系统中安装多个软件，如下表所示；为了避免安装错误，请按照列表中的顺序安装。

表 4-6 安装软件

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

4.3.2 解压/复制 BSP 和例子工程

请按照下表中的内容来将光盘中的 BSP 和例子工程解压/复制到 PC 上指定的位置。

表 4-7 解压/复制文件

操作	源地址	目的地址

操作	源地址	目的地址
解压	X:\WNCE600\bsp\mini8510.rar	C:\WINCE600\PLATFORM
解压	X:\WINCE600\bsp\COMMON_TI_V1.rar	C:\WINCE600\PLATFORM\COMMON\SRC\SOC
解压	X:\WINCE600\bsp\dvsdk_wince_01_11_00_02.rar	C:\WINCE600\3rdParty\
安装	X:\WINCE600\SGX\wince_gfx_sgx_01_01_00_patch_01_setup.exe	C:\TI\wince_gfx_sgx_01_01_00_patch_01
复制	C:\TI\wince_gfx_sgx_01_01_00_patch_01\poweVR	C:\WINCE600\public
复制	X:\WINCE600\prj\mini8510	C:\WINCE600\OSDesigns

注意：

📖 本手册中 Windows Embedded CE 6.0 的安装路径默认为 C:\WINCE600。

4.3.3 Sysgen 和 BSP 编译

请按照以下步骤完成对 sysgen 和 BSP 的编译。

1） 打开 C:\WINCE600\OSDesigns\mini8510 目录下的工程文件 mini8510.sln；

2） 在 Visual Studio 2005 程序窗口的菜单栏中选择 **Build** > **Build Solution**，开始 sysgen 和 BSP 编译；

3） 编译完成后，将 C:\WINCE600\OSDesigns\mini8510\mini8510\RelDir\mini8510_ARMV4I_Release 目录下的 MLO，EBOOTSD.nb0 和 NK 等映像文件复制到 SD 卡，然后将 SD 卡插入 SBC8140 的 SD 卡插槽并接通电源；

4） 按下空格键进入 eboot 菜单，然后输入字符 **a** 来选择相应的显示输出，最后输入 **0** 启动系统；

4.4 驱动介绍

下图是 SBC8140 的 BSP 架构；

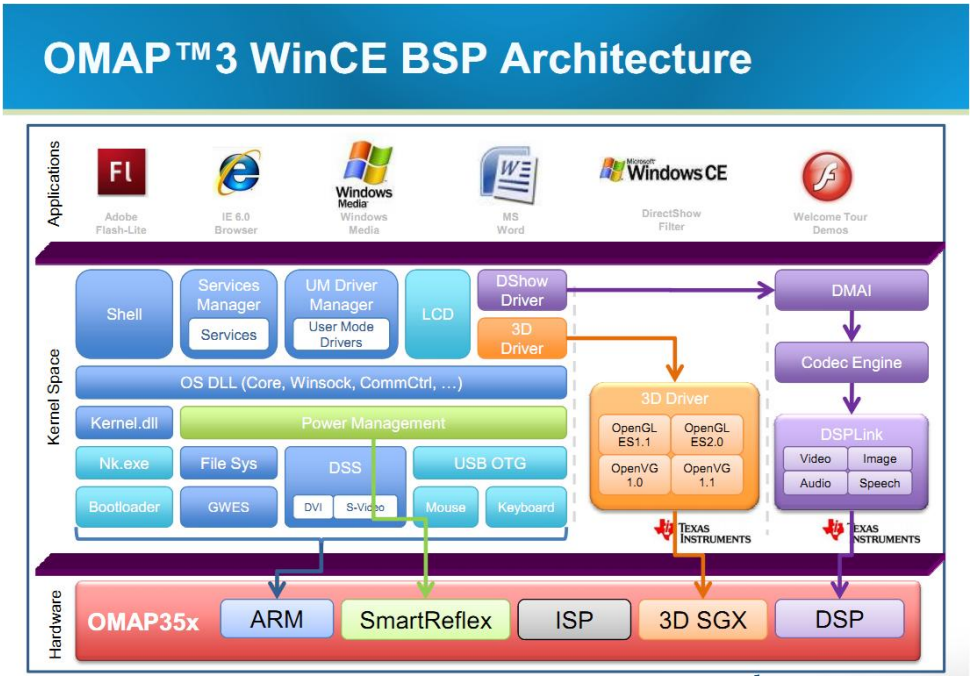


图 4-1 BSP 架构

下表列出了 BSP 的所有驱动源代码的路径；

表 4-8 BSP 中的驱动路径

驱动名称	路径
NLED driver	bsp\mini8510\SRC\DRIVERS\NLED
GPIO	bsp\mini8510\SRC\DRIVERS\GPIO
	bsp\COMMON_TI_V1\COMMON_TI\GPIO
I2C	bsp\COMMON_TI_V1\COMMON_TI\OAL\OMAP_OALI2C
	bsp\COMMON_TI_V1\COMMON_TI\CEDDK\I2C
SPI	bsp\COMMON_TI_V1\COMMON_TI\SPI
MCBSP driver	bsp\COMMON_TI_V1\COMMON_TI\MCBSP
	bsp\COMMON_TI_V1\OMAP3530\MCBSP
Series port driver	bsp\COMMON_TI_V1\COMMON_TI\SERIAL
6X6 keyboard driver	bsp\COMMON_TI_V1\COMMON_TI\KEYPAD
	bsp\mini8510\SRC\DRIVERS\TPS659XX_KEYPAD
Audio driver	bsp\mini8510\SRC\DRIVERS\TPS659XX_WAVE
	bsp\COMMON_TI_V1\TPS659XX\WAVE
NAND driver	bsp\OMAP35XX_TPS659XX_TI_V1\omap35xx\BLOCK
	bsp\COMMON_TI_V1\COMMON_TI\BLOCK
Display driver（LCD/DVI. S -Video/Composite Video）	bsp\COMMON_TI_V1\COMMON_TI\DSS
	bsp\mini8510\SRC\BSP_COMMON\DISPLAY
	bsp\mini8510\SRC\DRIVERS\DISPLAY
TOUCH driver	bsp\mini8510\SRC\DRIVERS\TOUCH

驱动名称	路径
SD/MMC/SDIO driver	bsp\mini8510\SRC\DRIVERS\SDBUS
	bsp\mini8510\SRC\DRIVERS\SDHC
	bsp\mini8510\SRC\DRIVERS\SDMEMORY
	bsp\COMMON_TI_V1\COMMON_TI\SDHC
SMSC9514 network card driver	bsp\mini8510\SRC\DRIVERS\SMSC9514
USB OTG driver	bsp\mini8510\SRC\DRIVERS\MUSB
	bsp\COMMON_TI_V1\OMAP3530\MUSB
	bsp\COMMON_TI_V1\TPS659XX\USBOTG
USB EHCI driver	bsp\ COMMON_TI_V1\COMMON_TI\USB
	bsp\ COMMON_TI_V1\OMAP3530\USB
	mini8510\SRC\DRIVERS\USBHS
VRFB driver	bsp\ COMMON_TI_V1\COMMON_TI\VRFB
DSPLINKK/CMEMK	bsp\3rdParty\dvsdk_wince_01_11_00_02
AAC/MPEG2/MPEG4/H264 DSP hardware decode fitler	bsp\3rdParty\dvsdk_wince_01_11_00_02
GPIO keyboard driver	bsp\COMMON_TI_V1\COMMON_TI\KEYPAD
	bsp\mini8510\SRC\DRIVERS\TPS659XX_KEYPAD
PWM(TPS65930)driver	bsp\ COMMON_TI_V1\TPS659XX\TLED
ADC(TPS65930)driver	bsp\ COMMON_TI_V1\TPS659XX\MADC
Camera driver	bsp\mini8510\SRC\DRIVERS\CAMERA
	bsp\mini8510\SRC\DRIVERS\CAMERA_Digital
Backlight driver	bsp\mini8510\SRC\DRIVERS\BACKLIGHT
Battery driver	bsp\mini8510\SRC\DRIVERS\BATTERY
Sleep/wake-up button driver	bsp\mini8510\SRC\DRIVERS\TPS659XX_PWRKEY（已知问题：当包含 tps65930 otg 驱动时无法将系统从挂起状态唤醒）
DVFS/Smart Reflex	bsp\ COMMON_TI_V1\COMMON_TI\PM
	bsp\ mini8510\SRC\DRIVERS\PM
DMA driver	bsp\ COMMON_TI_V1\COMMON_TI\SDMA
RTC driver	bsp\ COMMON_TI_V1\TPS659XX\OALRTC
	bsp\mini8510\SRC\DRIVERS\TPS659XX_RTC

如果您需要参考更多 WinCE 驱动开发，请在 PC 桌面上单击**开始** > **所有程序** > **MicroSoft Vistual Studio 2005** > **MicroSoft Visual Studio Document** > **Content(C)** > **Windows Embedded CE 6.0** > **Develop a Device Driver**。

4.5 系统更新

本章节将介绍如何对 SD 卡和 NAND Flash 中 WinCE 系统进行更新操作。

4.5.1 SD 卡系统更新

1） 请使用 HP USB Disk Storage Format Tool 2.0.6 格式化 SD 卡；（请访问 http://www.embedinfo.com/english/download/SP27213.exe 来下载该格式化软件）；软件界面如下；；

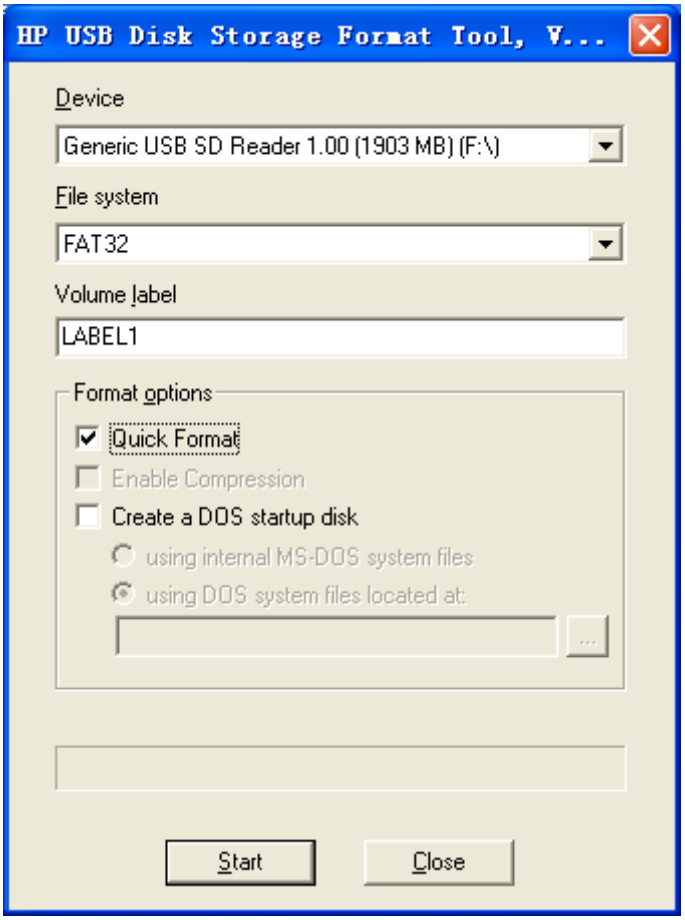


图 4-2 格式化 SD 卡

在 **File system** 下拉菜单中选择 FAT32 并单击 **Start** 按钮来进行格式化。

完成后将光盘目录 X:\WINCE600\image 下的 MLO、EBOOTSD.nb0 和 NK.bin 映像文件复制到 SD 卡中（X 为光盘盘符）；

注意：

🕮 HP USB Disk Storage Format Tool 会清除 TF 存储卡的分区。如果需要保留分区，请使用 Windows 系统的格式化功能。

2） 将 SD 卡插入 SBC8140 的 SD 卡插槽，然后按住 BOOT 键（CN11 键）并接通电源，超级终端窗口显示以下信息；

表 4-9 启动信息

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

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

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

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

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

>>> Forcing cold boot (non-persistent registry and other data will be wiped) <<<
Hit space to enter configuration menu 5... (按空格键进入 EBOOT 菜单)
```

当出现倒数读秒提示信息时，请按下空格键进入 EBOOT 菜单；

3） 在下面的 EBOOT 菜单界面中输入字符 **a**；

表 4-10 EBOOT 菜单

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

Selection:a
```

4） 在下面的菜单界面中根据您的显示设备选择适当的显示模式；

表 4-11 选择显示模式

```
---------------------------------------------------------------------------
 Select Display Resolution
---------------------------------------------------------------------------
 [1] LCD   480x272 60Hz      //4.3 寸 LCD 显示，默认显示设备
 [2] DVI   640x480 60Hz
 [3] DVI   640x480 72Hz
 [4] LCD   800x480 60Hz      //7 寸 LCD 显示
 [5] DVI   800x600 60Hz      //LVDS 显示
 [6] DVI   800x600 56Hz
 [7] VGA 1024x768 60Hz       //VGA 显示
 [8] DVI 1280x720 60Hz
 [0] Exit and Continue

 Selection (actual LCD   480x272 60Hz):4
```

5） 在下方的菜单界面中输入 **7** 和 **y** 保存设置，然后输入数字 **0** 继续启动进程；

表 4-12 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
 [0] Exit and Continue

 Selection:0
```

启动完成后即成功更新了 SD 卡中系统文件并启动了 WinCE 系统。

注意：

🕮 SBC8140 默认从 NAND Flash 启动；接通电源时按住 BOOT 键（CN11 键）可以从 SD 卡启动；

4.5.2 NAND Flash 系统更新

1） 格式化 SD 卡；

请参考第 82 页中的请使用 HP USB Disk Storage Format Tool 2.0.6 格式化 SD 卡；（请访问 http://www.embedinfo.com/english/download/SP27213.exe 来下载该格式化软件）；软件界面如下；；步骤的内容。格式化完成后，将光盘目录 X:\WINCE600\image 下的 MLO、EBOOTSD.nb0、EBOOTNAND.nb0、NK.bin 和 XLDRNAND.nb0 文件复制到 SD 卡中，并将 EBOOTNAND.nb0 重命名为 EBOOTND.nb0（X 为光盘盘符）；

2） 将 SD 卡插入 SBC8140 的 SD 卡插槽中，然后按住 BOOT 键并接通电源；当超级终端窗口中出现倒数读秒提示信息时，按下空格键进入 EBOOT 菜单；

3） 在 EBOOT 菜单中输入数字 **8** 进入 flash 管理菜单；

4） 输入字符 **a**、**b** 和 **c** 分别写入 XLDR、EBOOT 和 NK 映像文件；

5） 输入数字 **0** 回到主菜单，然后输入数字 **2** 和 **4** 选择从 NAND Flash 启动；

6） 在主菜单中输入字符 **a** 选择显示模式，然后同样在主菜单中输入数字 **7** 和字符 **y** 保存设置；

7） 从 SBC8140 上拔出 SD 卡并重新启动系统；这时系统将从 NAND Flash 启动；

4.6 其他操作说明

本章节将简要介绍 SBC8140 的一些模块功能和演示程序的使用方法，包括 OpenGL ES demo、CAM8000-A 模块和 CAM8000-D 模块。

4.6.1 使用 OpenGL ES demo

1） 在 Visual Studio 2005 窗口左方的 **Catalog Items View** 树形查看框中选中 PowerVR 分支内的所有选项，如下图所示；

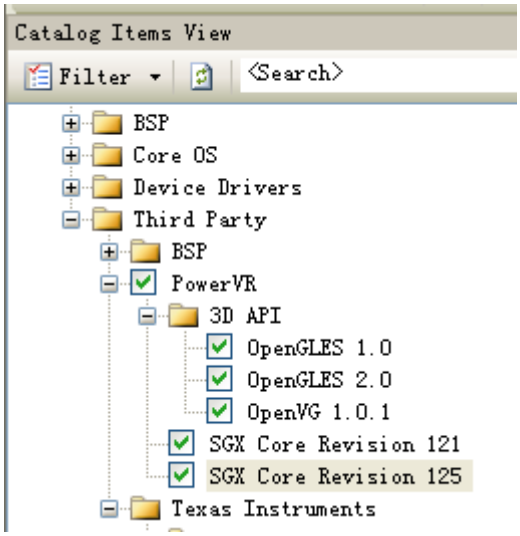


图 4-3 选中 PowerVR 分支

2） 在 Visual Studio 2005 窗口的菜单栏中选择 **Build** > **Build Solution** 来生成 nk.bin 文件，然后用该文件覆盖 SD 卡中的同名文件；

3） 将 C:\TI\wince_gfx_sgx_01_01_00_patch_01\PowerVR-SDK\OGLES1.1\Binaries\ Demos 目录下的文件或者 C:\WINCE600\PUBLIC\PowerVR\oak\targe

t\Rev125\ARMV4I\retail\目录下的*.exe 文件复制到 SBC8140 的 WinCE 系统中，然后双击运行 Demo 程序；

4.6.2 使用 CAM8000-A 模块

1） 对 mini8510.bat 文件中的 set BSP_NODIGITALCAMERA 语句进行修改；

```
set BSP_NODIGITALCAMERA=1
```

2） 在 Visual Studio 2005 窗口左方的 **Catalog Items View** 树形查看框中选中 DirectShow 分支下的项目，如下图所示；

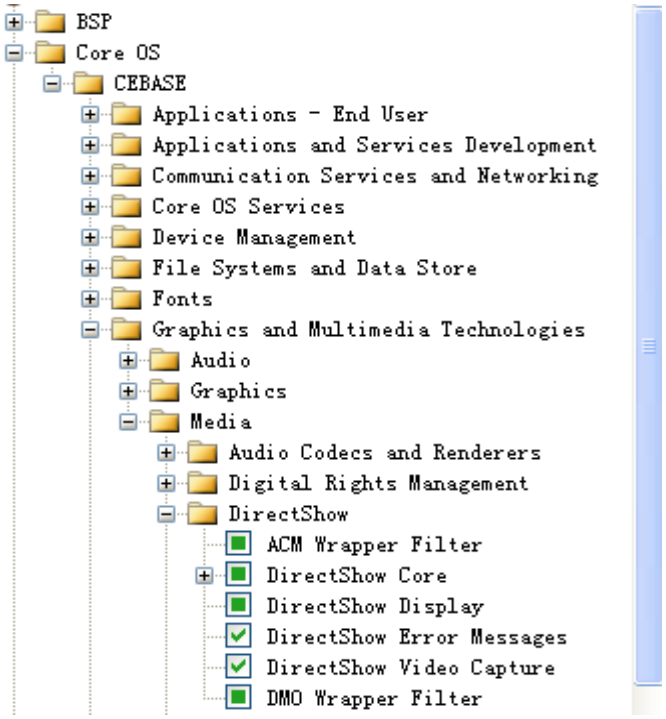


图 4-4 选中 DirectShow 分支

3） 在 Visual Studio 2005 窗口左方的 **Catalog Items View** 树形查看框中选中 **Third Party** > **BSP** > **mini8510:ARMV4I** > **drivers** > **camera**；然后在菜单栏中选择 **Build** > **Rebuild Solution** 开始编译；

4） 将 C:\WINCE600\platform\mini8510\files\目录下编译生成的 CameraDshowApp_analog.exe 文件复制到 SD 卡中，然后将 SD 卡插入 SBC8140；

5） 将摄像头模块（需自行购买）连接到 SBC8140 并接通电源，然后在 SBC8140 上的 WinCE 系统中执行 SD 卡上的 CameraDshowApp_analog.exe 文件来

测试 CAM8000-A 模块；

4.6.3 使用 CAM8000-D 模块

1） 对 mini8510.bat 文件中的 set BSP_NODIGITALCAMERA 语句进行修改；

```
set BSP_NODIGITALCAMERA=
```

2） 在 Visual Studio 2005 窗口左方的 **Catalog Items View** 树形查看框中选中 DirectShow 分支下的项目，如下图所示；

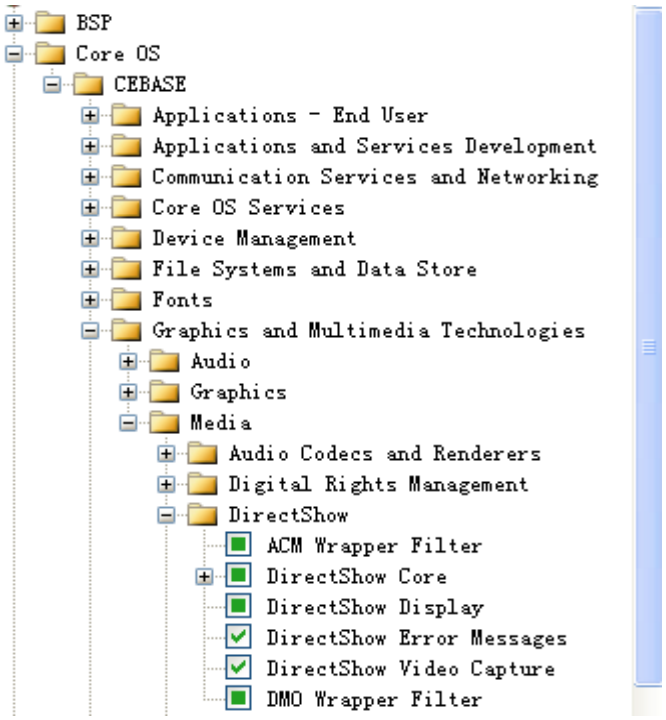


图 4-5 选中 DirectShow 分支

3） 在 Visual Studio 2005 窗口左方的 **Catalog Items View** 树形查看框中取消选中 **Third Party** > **BSP** > **mini8510:ARMV4I** > **drivers** > **camera**；然后在菜单栏中选择 **Build** > **Rebuild Solution** 开始编译；

4） 将 C:\WINCE600\platform\mini8510\files\ 目录下编译生成的 CameraDshowApp_digital.exe 文件复制到 SD 卡中，然后将 SD 卡插入 SBC8140；

5） 将摄像头模块（需自行购买）连接到 SBC8140 并接通电源，然后在 SBC8140 上的 WinCE 系统中执行 SD 卡上的 CameraDshowApp_digital.exe 文件来测

试 CAM8000-D 模块；

注意：

🕮 如果在 CameraDshowApp_digital.exe 运行时，在 Capture Parameters 窗口选择了 still sink，那么在 WinCE 系统的 Control Pannel > System 里需要将应用程序内存设置为 170000KB 或更大的空间才能成功运行 DirectSHow Graph。

4.7 GPIO 应用程序接口与开发范例

本章节将通过介绍 GPIO 应用程序接口和示例来展示如何在 WinCE 环境下进行应用程序的开发工作。

SBC8140 应用程序开发所用到的 API 均采用微软 Windows Embedded CE 6.0 标准应用程序接口定义；SBC8140 仅在标准 API 基础上扩展了 GPIO 的接口定义，控制 GPIO 管脚状态的应用程序请参考 X:\WINCE600\app\GPIOAppDemo（X 为光盘盘符）。

Windows Embedded CE 6.0 标准应用程序接口定义可以查看 MSDN Windows Embedded CE 6.0 API 相关帮助文档。

注意：

🕮 部分驱动导出的接口仅供驱动使用，应用程序无权限调用。

GPIO 设备名为 L"GIO1:"，扩展了 DeviceIoControl 接口定义；下表列出了对应 IOCTL 码；

表 4-13 IOCTL 码

IOCTL 码	描述
IOCTL_GPIO_SETBIT	GPIO 引脚置 1
IOCTL_GPIO_CLRBIT	GPIO 引脚清 0
IOCTL_GPIO_GETBIT	读 GPIO 引脚状态
IOCTL_GPIO_SETMODE	设置 GPIO 引脚工作模式
IOCTL_GPIO_GETMODE	读 GPIO 引脚工作模式
IOCTL_GPIO_GETIRQ	读 GPIO 引脚对应的 IRQ 号

以下表格列出了 GPIO 应用程序的范例；

1） 打开 GPIO 设备；

表 4-14 打开设备

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

2） 设置读 GPIO 工作模式；

表 4-15 设置为读模式

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

3） 设置 GPIO 工作模式；

表 4-16 设置工作模式

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

4） 读 GPIO 工作模式；

表 4-17 读取 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；下表列出了所有的模式定义；

表 4-18 GPIO 模式定义

模式定义	描述
GPIO_DIR_OUTPUT	输出模式
GPIO_DIR_INPUT	输入模式
GPIO_INT_LOW_HIGH	上升沿触发模式
GPIO_INT_HIGH_LOW	下降沿触发模式
GPIO_INT_LOW	低电平触发模式
GPIO_INT_HIGH	高电平触发模式
GPIO_DEBOUNCE_ENABLE	跳变触发使能

5） GPIO 引脚操作

表 4-19 引脚操作

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

表 4-20 输出高电平

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

表 4-21 输出低电平

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

表 4-22 读取引脚状态

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

id 为 GPIO 引脚号；**pin** 返回引脚状态；

6） 读取 GPIO 引脚对应的 IRQ 号码；

表 4-23 读取 IRQ 号码

```
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 号码；

7） 关闭 GPIO 设备；

表 4-24 关闭设备

```
CloseHandle(hFile);
```

注意：

- GPIO 引脚定义为：0~191 MPU Bank1~6 GPIO 引脚，192~209 TPS65930 GPIO 0~17。
- GPIO 引脚 0~191 必须在 xldr/platform.c 与 oalib/oem_pinmux.c 两个文件中被定义为 GPIO。
- GPIO 中断模式仅供驱动使用，对应用程序式无效。

附录 1 安装 Ubuntu Linux 系统

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 1 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；

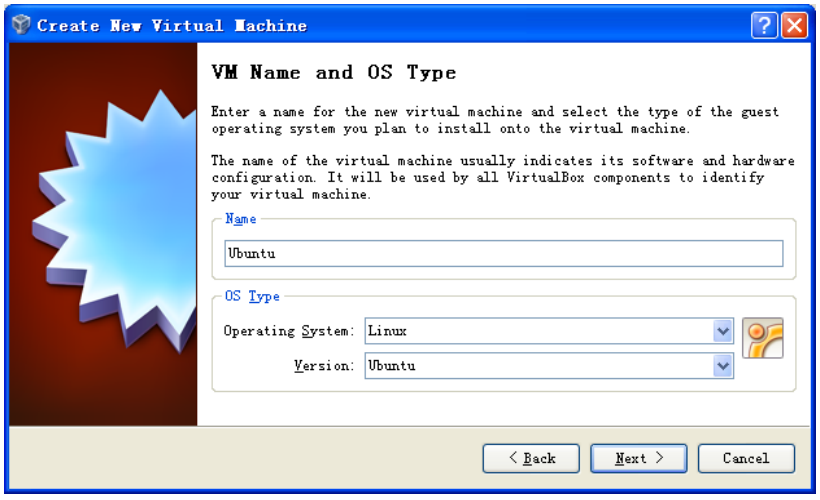


图 2 虚拟机名称和操作系统类型

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；

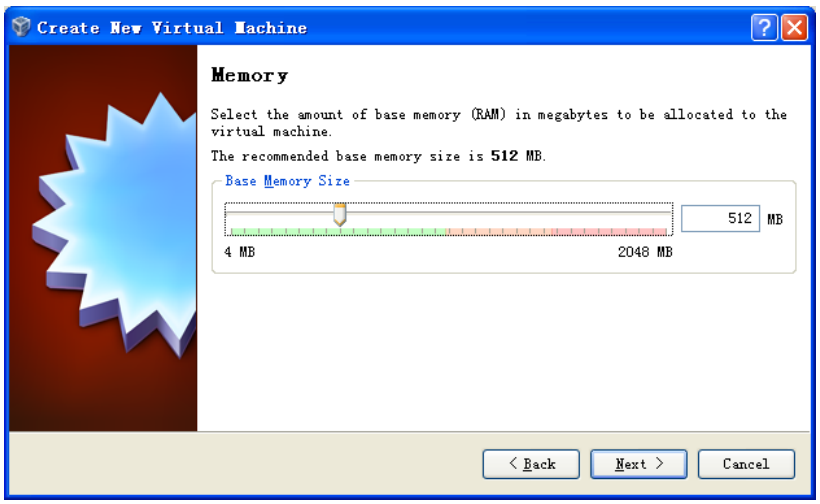


图 3 内存分配窗口

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 或者更少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；

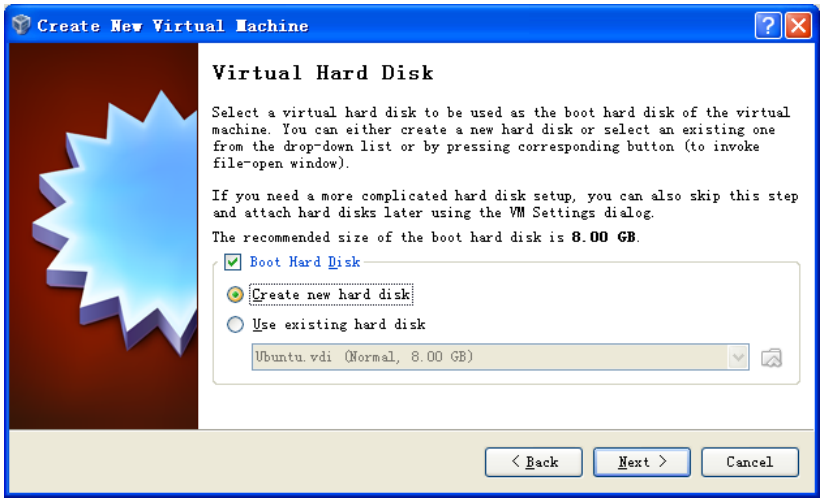


图 4 创建新的虚拟硬盘窗口

5） 在下方虚拟硬盘创建向导窗口中单击 **Next** 按钮；

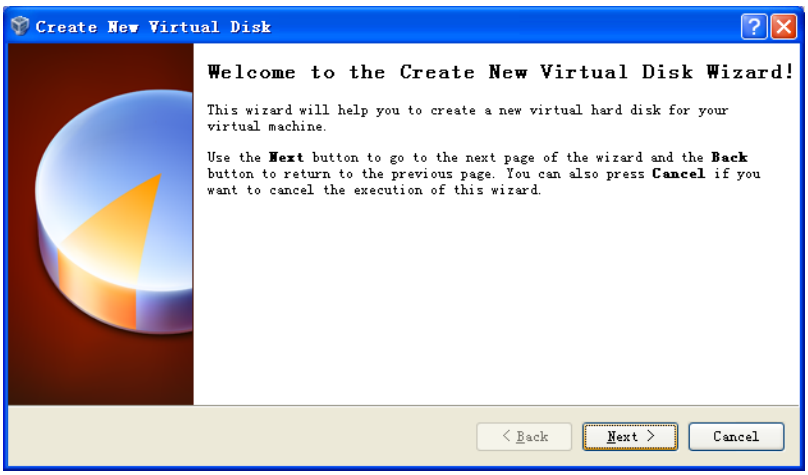


图 5 虚拟硬盘创建向导

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

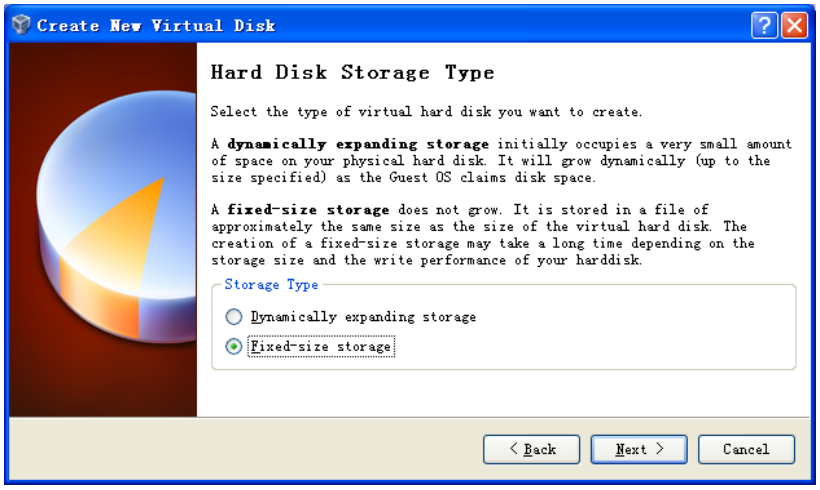


图 6 选择第二个选项

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；

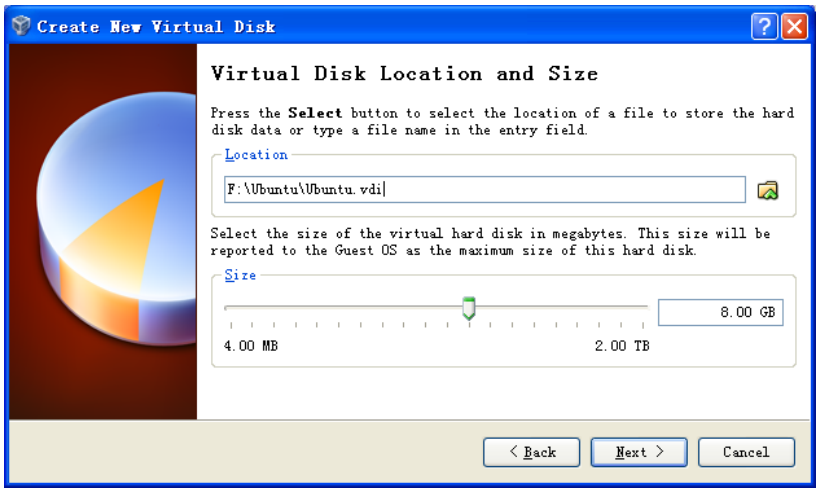


图 7 虚拟硬盘设置窗口

8） 在下方的虚拟硬盘信息窗口中单击 **Finish** 按钮；

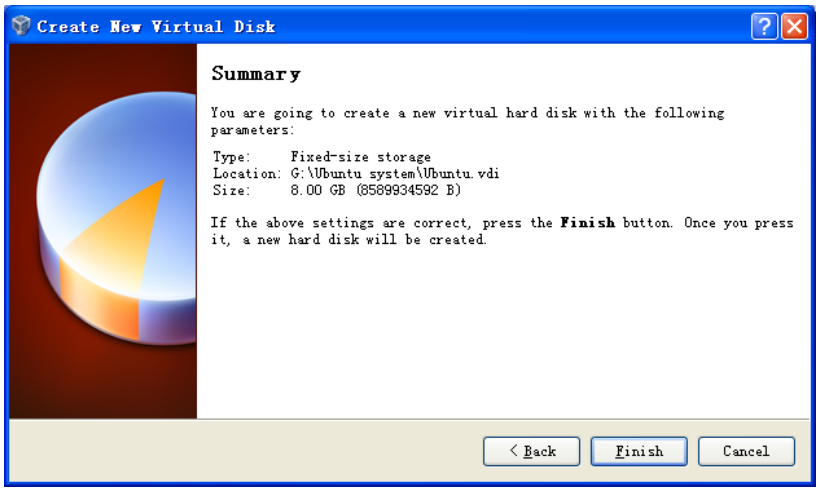


图 8 虚拟硬盘信息

9） PC 开始创建虚拟硬盘驱动器；

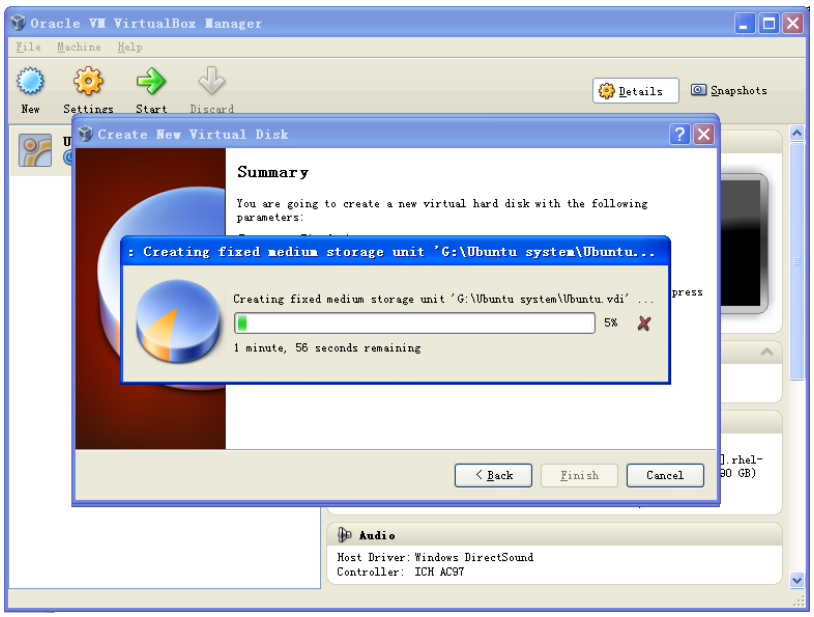


图 9 创建虚拟硬盘驱动器

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；

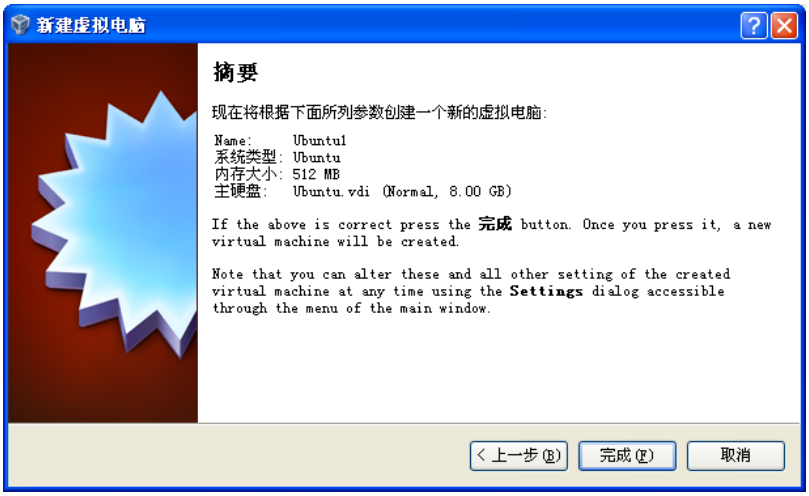


图 10 虚拟机配置完成

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；

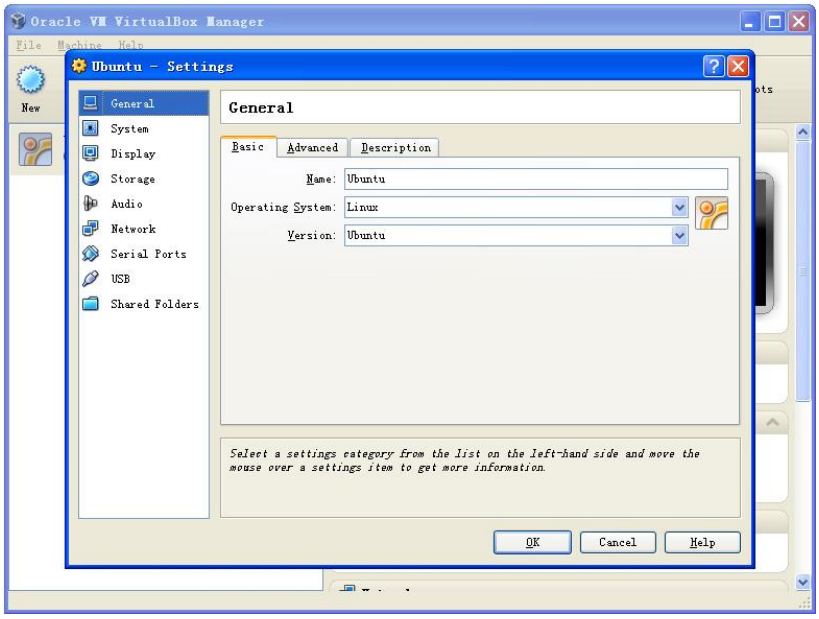


图 1 虚拟机设置

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；

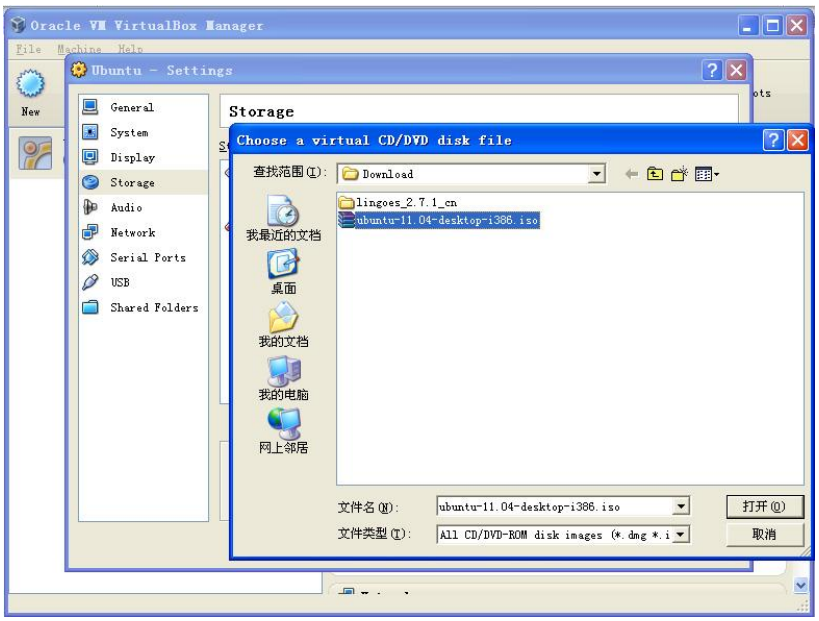


图 2 指定 Ubuntu 映像位置

3） 选中刚才添加的映像并单击 **OK** 按钮，如下图所示；

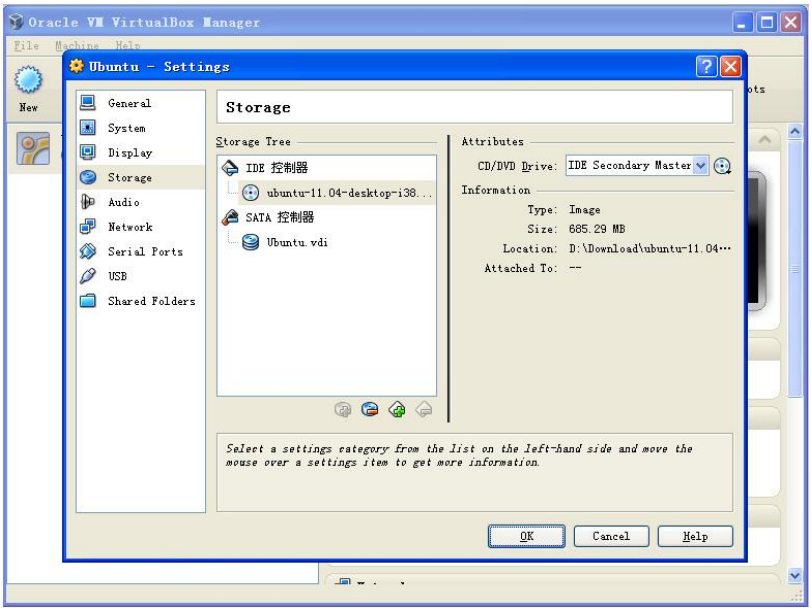


图 3 选中 ISO 映像

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；

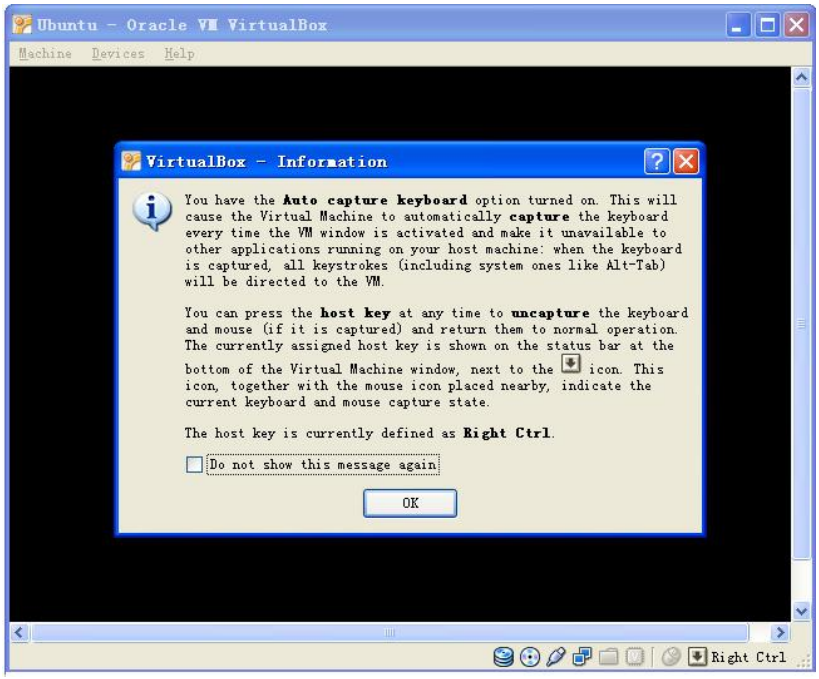


图 4 Ubuntu 初始化窗口

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 5 Ubuntu 安装窗口

6） 单击 **Forward** 按钮继续安装，如下图所示；

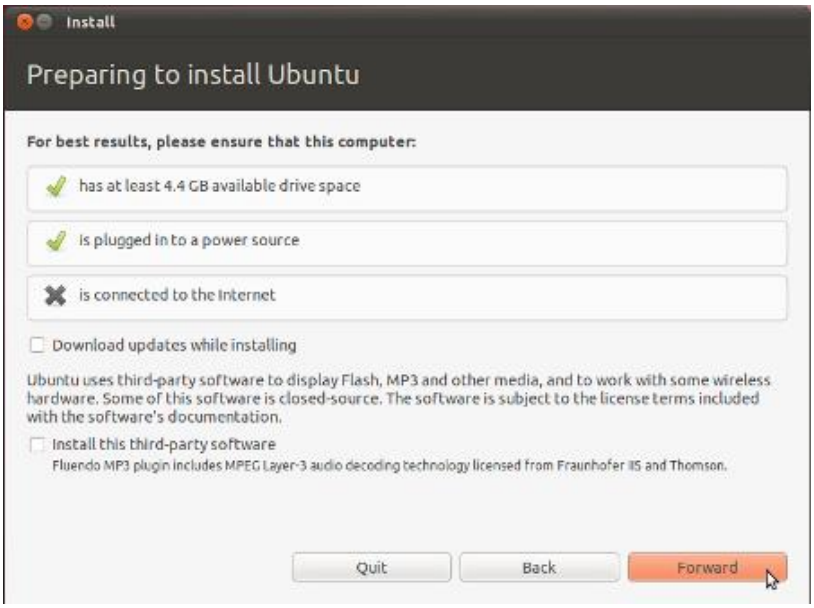


图 6 安装前的信息

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

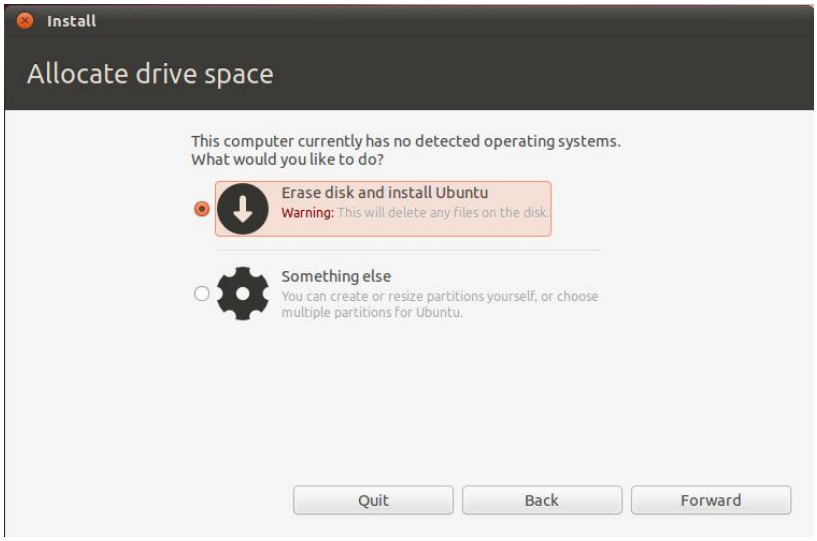


图 7 Ubuntu 安装选项

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；

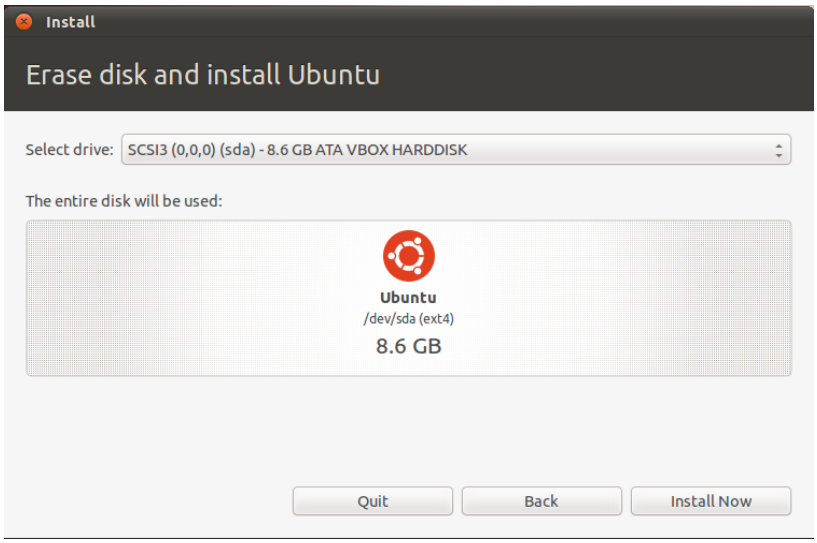


图 8 安装确认窗口

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；

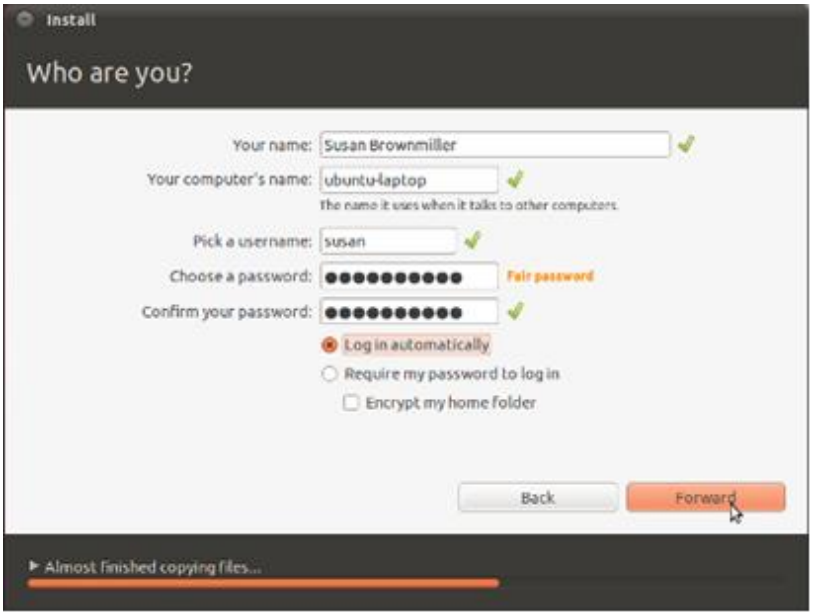


图 9 指定用户名和密码

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；

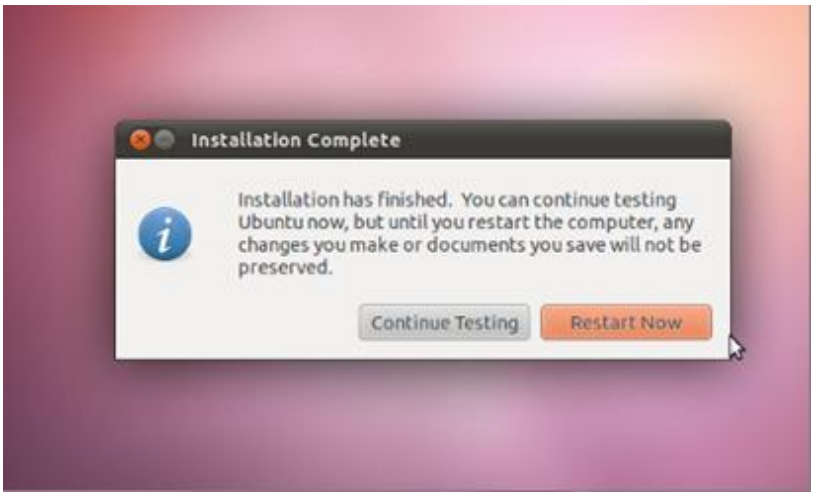


图 10 重新启动系统

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 3 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；

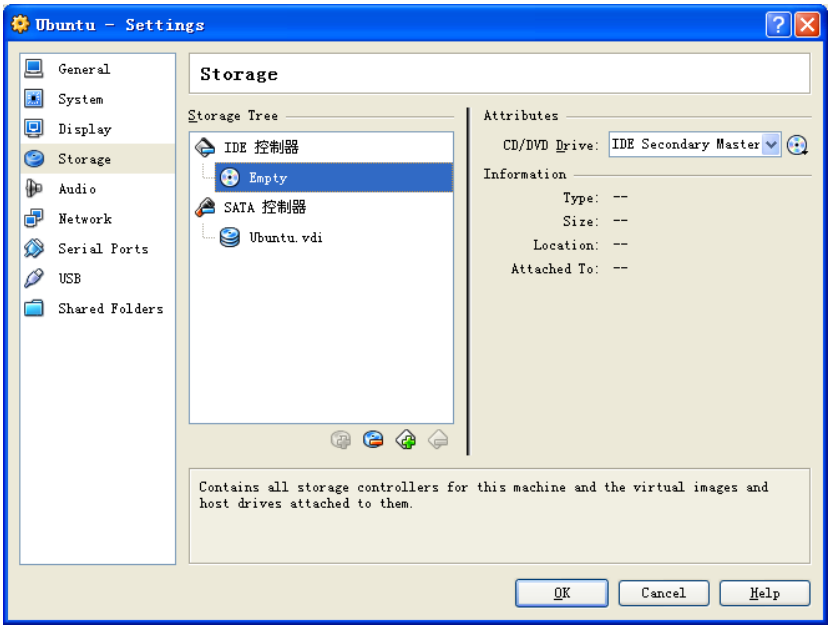


图 11 弹出 ISO 映像

附录 2 安装 Linux USB Ethernet/RNDIS Gadget 驱动

1） 如果你还没安装 Linux USB Ethernet/RNDIS Gadget 驱动，请在下面的**找到新硬件向导**窗口中选择**从列表或指定位置安装**，然后单击**下一步**；

图 1 找到新硬件

2） 指定 USB 驱动路径 X:\linux\tools\（X 为光盘盘符），然后点击“下一步”。

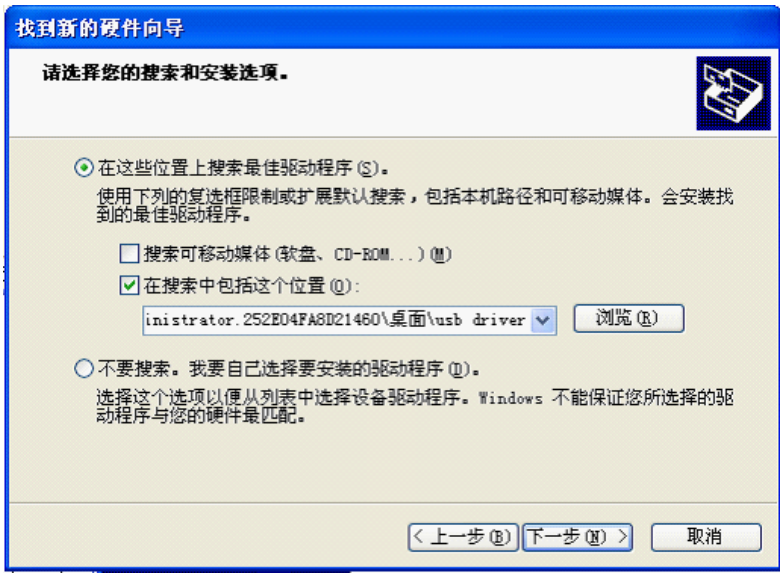


图 2 指定驱动位置

3） 出现以下提示时，单击 **Continue Anyway**；

图 3 选择 Continue Anyway

4） 下面的窗口信息表示驱动已经成功安装，单击完成即可；

图 4 选择完成

附录 3 制作 Linux 启动盘

以下内容将介绍如何制作一个双分区的 Linux 启动盘，以便让 SBC8140 能够从启动盘的第一个分区启动系统，同时在第二个分区中保存根文件系统；

1） 将一张 TF 卡插入读卡器并连接到 PC 上；在 Ubuntu 系统中执行以下命令来确定 TF 卡的设备名；

$ dmesg | tail

表 1 设备信息

```
...
[ 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
...
```

以上信息显示 TF 卡的设备名为**/dev/sdc**；

2） 执行以下命令来查看 Ubuntu 系统自动挂载的设备路径；

$ df -h

表 2 设备路径

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

/dev/sdc1 所在行的尾端显示设备路径为**/media/disk**；

注意：

🕮 如果 TF 卡存在两个或两个以上分区，则以上设备路径信息会以/dev/sdc1、/dev/sdc2、/dev/sdc3 等的形式显示每个分区。

3） 执行以下命令来卸载该设备；

$ umount /media/disk

4） 执行 **fdisk** 命令

$ sudo fdisk /dev/sdc

请确保命令行中输入的是整个设备的路径，而不是分区路径/dev/sdc1 或/dev/sdc2 等；

5） 上面的命令执行完成后，输入字符 **p** 来打印设备的分区记录，如下表所示；

表 3 分区记录

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)

根据以上信息记下设备的字节数，例如上表中的 2021654528 bytes；然后输入字符 **d** 来删除所有分区；

6） 如果表 3 中没有显示 **255 heads** 和 **63 sectors/track**，请按照以下操作来恢复 TF 卡；

A. 按照下表中的提示输入字符来设置 Heads 和 Sectors；

表 4 设置 Heads 和 Sectors

Command (m for help): **[x]**（输入 x 进入专家模式）
Expert Command (m for help): **[h]**（输入 h 来设置 heads）
Number of heads (1-256, default xxx): **[255]**（将 heads 设置为 255）
Expert Command (m for help): **[s]**（输入 s 来设置 sectors）
Number of sectors (1-63, default xxx): **[63]**（将 sector 设置为 63）

B. 使用以下公式计算 Cylinders 的数量；

Cylinders = 之前记下的设备字节数÷255÷63÷512

然后按照下表中的提示输入字符来设置 Cylinders；

表 5 设置 Cylinders

Expert Command (m for help): **[c]**（输入 c 来设置 cylinders）
Number of cylinders (1-256, default xxx): (输入刚才计算的 cylinder 数量)...
Expert Command (m for help): **[r]**（输入 r 回到一般模式）

7） 输入字符 **p** 来检查刚才设置的参数，如下表所示；

表 6 检查参数

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

```
63 sectors/track, 245 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
```

8） 按照下表中的操作来建立 FAT32 启动分区和从 Windows 传输文件；

表 7 建立 FAT32 启动分区

```
Command (m for help): [ n ]（输入 n 开始建立分区）
Command action
   e   extended
   p   primary partition (1-4)
[ p ]（输入 p 建立主分区）
Partition number (1-4): [ 1 ]（将分区号码设置为 1）
First cylinder (1-245, default 1): [ ]（按下 Enter 键）
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-61, default 61): [ +5 ]（输入+5）

Command (m for help): [ t ]（输入 t）
Selected partition 1
Hex code (type L to list codes): [ c ]（输入 c 设置分区系统类型）
Changed system type of partition 1 to c (W95 FAT32 (LBA))
```

9） 输入 **a** 和 **1** 来将 TF 卡设置为 bootable 模式；

表 8 设置 bootable 模式

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

10） 按照下表的提示输入，以便为根文件系统建立分区；

表 9 建立根文件系统分区

```
Command (m for help): [ n ]（输入 n 来建立分区）
Command action
   e   extended
   p   primary partition (1-4)
[ p ]（输入 p 选择主分区）
Partition number (1-4): [ 2 ]（将分区号码设置为 2）
First cylinder (7-61, default 7): [ ]（按下 Enter 键）
Using default value 52
Last cylinder or +size or +sizeM or +sizeK (7-61, default 61): [ ]（按下 Enter 键）
Using default value 245
```

11） 输入字符 **p** 来检查建立的分区，如下表所示；

表 10 检查分区

```
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
```

12） 输入字符 **w** 来保存新分区记录，如下表所示；

表 11 保存分区

```
Command (m for help): [ w ]
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table.
The new table will be used at the next reboot.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
```

13） 执行以下命令来格式化新建的两个分区；

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

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

注意：

- 以上命令中的 LABEL1 和 LABEL2 卷标可以由您自定义为其他名称。
- 在 FAT 和 EXT3 双分区的格式化完成后，FAT 分区需要在 Windows 中重新格式化一次，否则可能会出现无法从 TF 卡启动的情况。

附录 4 建立 TFTP 服务器

请在 Ubuntu Linux 系统中通过以下步骤来建立 TFTP 服务器；

1） 执行以下命令来安装客户端；

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

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

2） 执行以下命令来安装 inet；

```
$>sudo apt-get install xinetd

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

3） 执行以下命令来配置服务器；

A. 在根目录下建立一个 tftpboot，并把属性修改为任意用户可读写；

```
$>cd /

$>sudo mkdir tftpboot

$>sudo chmod 777 tftpboot
```

B. 执行以下命令，然后添加语句 tftpd dgram udp wait root /usr/sbin/in.tftpd /usr/sbin/in.tftpd -s /tftpboot；

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

C. 重新加载 inetd 进程；

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

D. 执行以下命令来新建名为 tftp 的文件，然后将下方表格内的内容添加到该文件中；

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

$>sudo touch tftp

$>sudo vi tftp
```

表 12 添加以下语句

```
service tftp
{
        disable = no
        socket_type     = dgram
        protocol        = udp
```

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

5） 执行以下命令来重新启动服务；

```
$>sudo /etc/init.d/xinetd restart
```

```
$>sudo in.tftpd -l /tftpboot
```

6） 通过以下操作来测试 TFTP 服务器；

A. 在/tftpboot 文件夹下新建一个文件；

```
$>touch abc
```

B. 然后进入另一个文件夹并下载刚才新建的文件（192.168.1.15 为 PC 的 IP 地址）；

```
$>tftp 192.168.1.15
```

```
$>tftp> get abc
```

如果成功下载该文件，说明 TFTP 服务器工作正常。

附录 5 FAQ

请访问 http://www.elinux.org/SBC8600_FAQ 页面的内容。

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：+86-755-25635626-872/875/897

传真号码：+86-755-25635626-666

售前咨询：sales@timll.com

售后支持：support@timll.com

官方网站：http://www.timll.com

通讯地址：深圳市南山区留仙大道 1183 号南山云谷创新产业园山水楼 4 楼 B