

# DevKit8000 Linux 用户手册

版本1.0

---

发布: 2009-03-20



## 版本更新记录

版本	发布日期	描述
1.0	2009-01-16	初始发布

## 接洽信息

访问如下地址，可获取更多信息: <http://www.timll.com>

## 目录

DEVKIT8000 LINUX用户手册 .....	1
第一部分 硬件系统介绍 .....	5
第一章 系统概述 .....	5
1.1 产品简介.....	5
1.2 产品规范.....	6
1.3 产品配件.....	7
第二章 接口规范 .....	8
3.1 接口位置概述.....	8
3.2 接口描述.....	9
第二部分 软件系统介绍 .....	22
第三章 软件系统概述 .....	22
3.1 预装软件.....	23
3.2 BSP features.....	24
3.3 Introduction to CD .....	错误! 未定义书签。
第四章 从这里开始 .....	25
4.1 准备.....	25
4.2 LED测试.....	25
4.3 KEYPAD测试.....	26
4.4 触摸屏测试.....	26
4.5 RTC测试.....	26
4.6 MMC/SD卡测试.....	27
4.7 USB OTG测试.....	27
4.8 AUDIO/VIDEO测试.....	29
4.9 网络测试.....	29
第五章 LINUX系统开发平台搭建 .....	31
5.1 交叉编译环境的搭建.....	31
5.2 生成系统映像文件.....	32
5.3 系统定制.....	34
第六章 映像更新与恢复 .....	38
6.1 设备连接及设置.....	38
6.2 映像更新.....	38
6.3 系统恢复.....	44
第七章 DevKit8000 应用开发示例 .....	46
7.1 开发环境的准备.....	46
7.2 LED Project.....	46
第八章 DevKit8000 DEMO演示 .....	48
8.1 angstrom(GPE)桌面发布版本演示.....	48
8.2 google android系统演示.....	48
第九章 FAQ .....	50
附录一 .....	51
附录二.....	53

附录三..... 58

## 第一部分 概述

### 第一章 系统概述

#### 1.1 产品简介

DevKit8000 是深圳市天漠科技有限公司推出的基于德州仪器 (TI) OMAP35x 处理器的评估套件。OMAP35x 处理器集成了 600MHz 的 ARM Cortex™-A8 内核及 430MHz 的具有高级数字信号处理算法的 DSP 核, 并提供了丰富的外设接口。DevKit8000 外扩了 CPU 外设接口中的网口、S-VIDEO 接口、音频输入输出接口、USB OTG、USB HOST、SD/MMC 接口、串口、SPI 接口、IIC 接口、JTAG 接口、CAMERA 接口、TFT 屏接口、触摸屏接口、键盘接口和总线接口, 并扩展出了 HDMI 接口。

DevKit8000 为开发者使用 OMAP35x 处理器提供了完善的软件开发平台, 支持 Linux-2.6.28 操作系统, 并包含完善的底层驱动程序, 方便用户快速评估 OMAP35x 处理器、设计 Linux 系统驱动及其定制应用软件。并提供有成熟的操作系统 google Android 及 Angstrom(GPE)的发布版本, DVI 输出可达到 720P 的显示标准, 方便用户体验 OMAP35x 处理器的强大的数据运算处理能力。

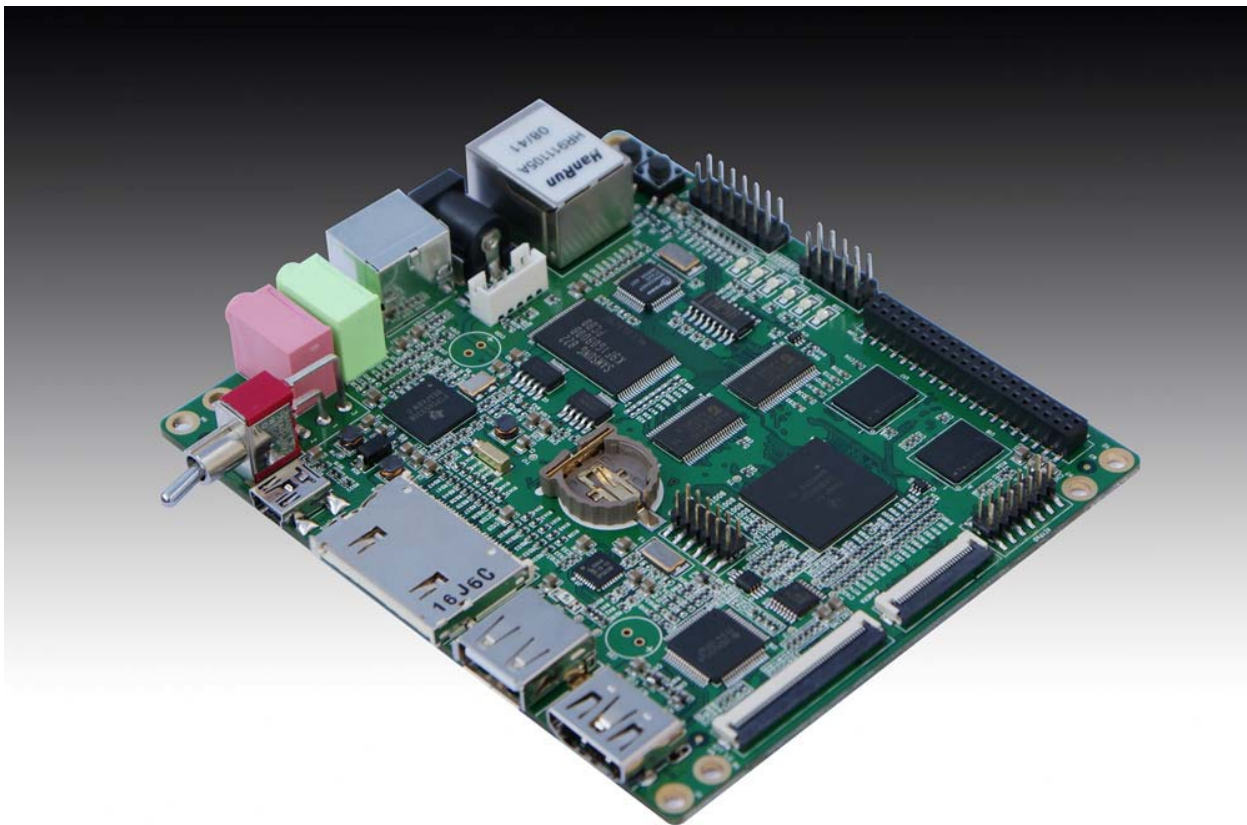


图1.1 产品图片

## 1.2 产品特性

### 1.2.1 硬件资源

#### Processor

- OMAP3530 处理器（完全兼容 OMAP3503 处理器接口）
- 600-MHz ARM Cortex™-A8 Core
- 430-MHz TMS320C64x+™ DSP Core
- 集成存储器用于 ARM CPU (16kB I-Cache, 16kB D-Cache, 256kB L2) 和片上存储 (64kB SRAM, 112kB ROM)

#### Memory

- 128MByte DDR SDRAM, 166MHz
- 128MByte NAND Flash, 16bit

#### 音频/视频接口

- 一个 4 线 S-VIDEO 接口
- 1 个 HDMI 接口（数字信号视频传输的高清晰度多媒体接口）
- 一个 2 声道音频输入接口
- 一个 2 声道音频输出接口

#### 液晶触摸屏

- 分辨率: 480 (W) x 272 (H) dots
- RGB, 391680 colors
- 亮度: 典型值 350 cd/m<sup>2</sup> (最小 300 cd/m<sup>2</sup>)
- 4 线触摸屏

#### 传输接口

- 串口:
  - 1 x 3 线串行接口, RS232 电平
  - 1 x 5 线串行接口, TTL 电平
- USB 接口:
  - 1 x USB2.0 OTG, High-speed, 480Mbps
  - 1 x USB2.0 HOST, High-speed, 480Mbps
- SD/MMC 接口:
  - 1 路 SD/MMC 接口, 支持 3.3V 及 1.8V 逻辑电压
  - 1 路 SD/MMC 接口, 支持 1.8V 逻辑电压
- GPMC 总线接口: 1 x 16bit, 40Mhz
- 网络接口: 10/100Mbps, RJ45 connector
- 1 路 McSPI 接口
- 1 路 McBSP 接口
- 1 路 I2C 接口
- 1 路 HDQ 接口

#### 输入接口

- 1 个 CAMERA 接口（可外接 CCD 和 CMOS 的摄像头）
- 6 X 6 键盘接口
- 1 个 14 针标准 JTAG 接口
- 1 个启动引导按键

- 1 个 Reset 按键。

### 1.2.2 电气特性

主板尺寸 : 110 mm x 95 mm

输入电压 : +5V

功 耗 : 0.5A @ 5V

操作温度 : 0 °C ~ 70 °C

操作湿度 : 20% ~ 90%

### 1.3 产品配件

DevKit8000 评估套件分两种配置：标准配置和完全配置。

- 标准配置：着重评估板的基本功能，主要针对具有一定开发条件的板级开发者；
- 完全配置：包含完善的接口配件的支持，具备了 LCD 屏等相关配件，该配置主要针对特定应用的专业产品开发人员。

配件	标准配置	完全配置
DevKit8000 评估板	√	√
SD 卡（容量：512M）	√	√
交叉串口线	√	√
电源适配器（5V）	√	√
4.3" LCD 屏（带触摸屏）		√
触摸笔		√
USB 转接线（Mini-B to A）		√
USB 转接线（Mini-A to A）		√
USB HUB		√
网线		√
HDMI 转 DVI-D 转接线		√
S-Video 线		√

## 第二部分 硬件系统

### 第二章 接口规范

#### 2.1 接口位置

##### 2.1.1 架构图

图 2.1 为 DevKit8000 评估套件的结构框图，主要介绍该板的外设设备。

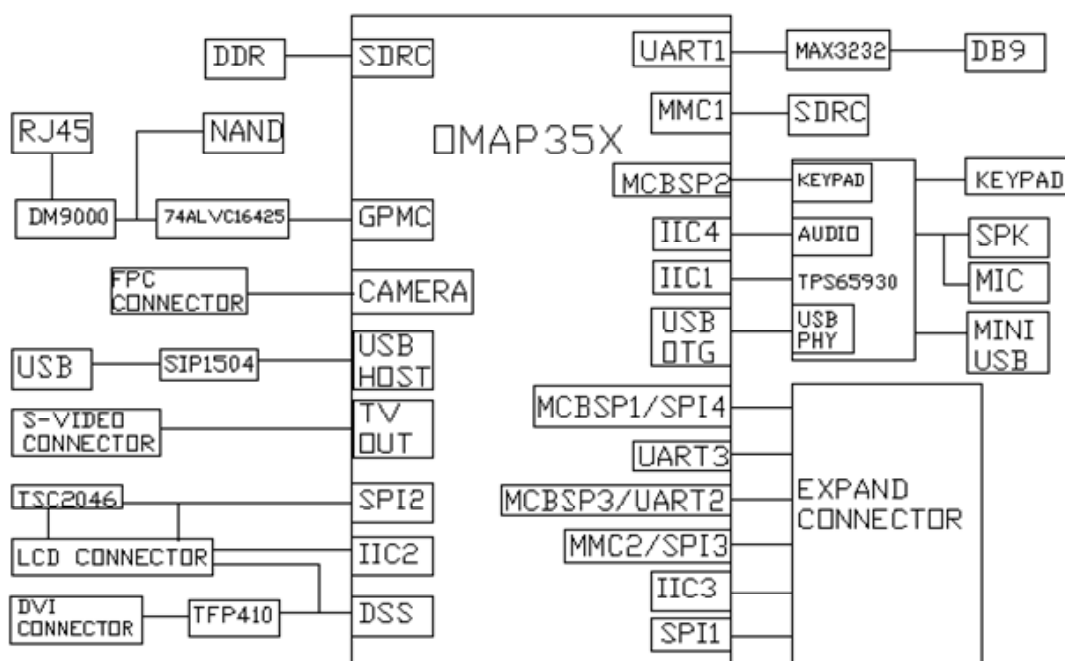


图2.1 结构框图

##### 2.1.2 硬件接口图

DevKit8000 评估套件的硬件接口图如图 2.2 及图 2.3 所示。



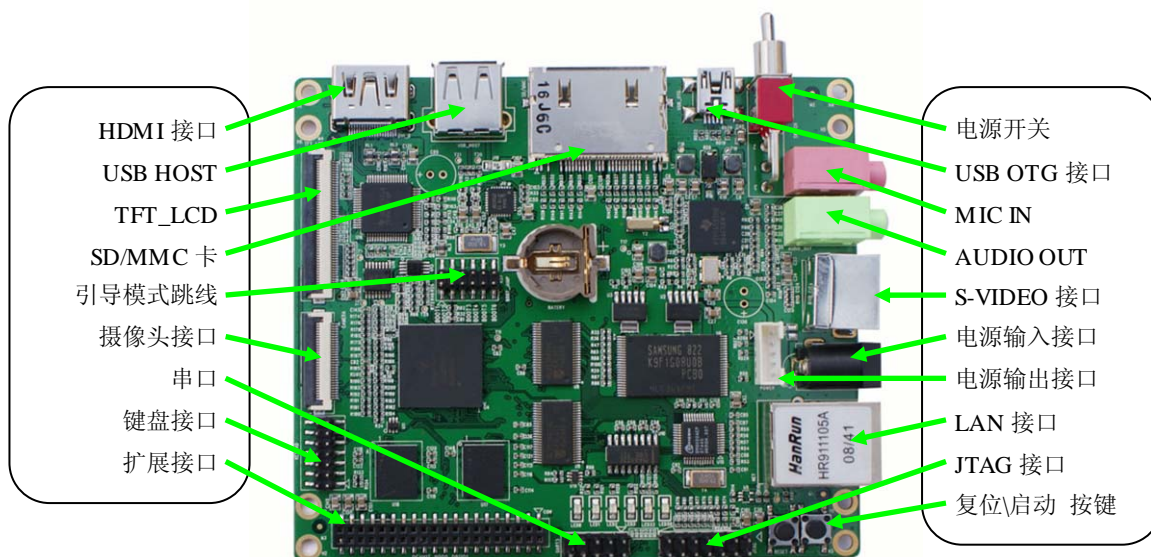


图2.2 正面接口图

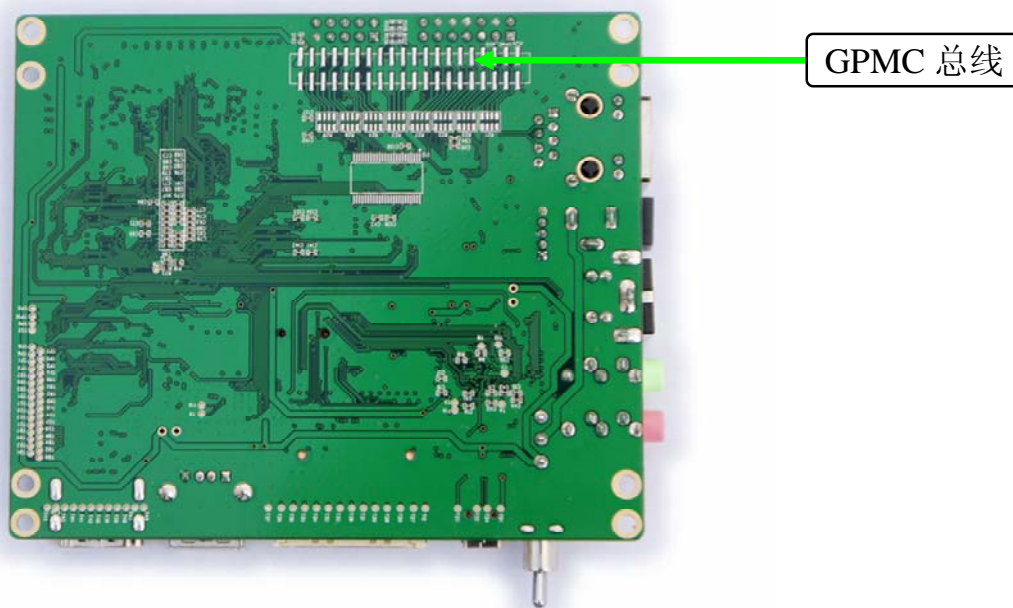


图2.3 背面接口图


## 2.2 接口描述

### 2.2.1 电源输入接口

功能描述：为 DevKit8000 提供 5V 电压输入。

接口描述：参考表 2-1。

表 2-1 电源输入接口

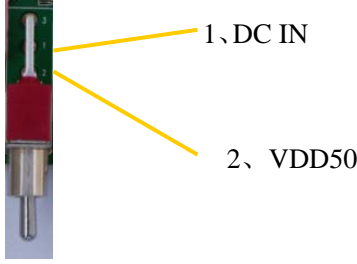
引脚	信号定义	功能描述	引脚描述
1	GND	Power input (+5V)	
2	NC	NC	
3	+5V	Power supply (+5V) 2A (Type)	

### 2.2.2 电源开关

功能描述：+5V 电源开关。

接口描述：参考表 2-2。

表 2-2 电源开关


引脚	信号定义	功能描述	引脚描述
1	DC IN	VDD Input	
2	VDD50	+5V	
3	NC	NC	

### 2.2.3 LAN 接口

功能描述：网络接口。

接口描述：参考表 2-3。

表 2-3 LAN 接口

引脚	信号定义	功能描述	引脚描述
1	TX+	TX+ output	
2	TX-	TX- output	
3	RX+	RX+ input	
4	VDD25	2.5V Power for TX/RX	
5	VDD25	2.5V Power for TX/RX	
6	RX-	RX- input	
7	NC	NC	
8	NC	NC	
9	VDD	3.3V Power for LED	
10	LED1	Speed LED	
11	LED2	Link LED	
12	VDD	3.3V Power for LED	

### 2.2.4 S-VIDEO 接口

功能描述：标准的 4 线 S-VIDEO 接口。

接口描述：参考表 2-4。

表 2-4 S-VIDEO 接口

引脚	信号定义	功能描述	引脚描述
1	GND	GND	 <p>PIN 4      PIN 3 PIN 2      PIN 1</p>
2	GND	GND	
3	OUTPUT1	VIDEO Y	
4	OUTPUT2	VIDEO C	

### 2.2.5 MIC IN 接口

功能描述：标准的 MIC IN 接口。

接口描述：参考表 2-5。

表 2-5 MIC IN 接口


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

### 2.2.6 AUDIO OUT 接口

功能描述：标准的 Audio out 接口。

接口描述：参考表 2-6。

表 2-6 AUDIO OUT 接口


引脚	信号定义	功能描述	引脚描述
1	GND	GND	
2	NC	NC	
3	Right	Right output	
4	NC	NC	
5	Left	Left output	

### 2.2.7 USB OTG 接口

功能描述：Mini 的 USB A 接口。

接口描述：参考表 2-7。

表 2-7 USB OTG 接口

引脚	信号定义	功能描述	引脚描述
1	VBUS	+5V	 <p>PIN 1</p>
2	DN	USB Data-	
3	DP	USB Data+	
4	ID	USB ID	
5	GND	GND	

## 2.2.8 USB HOST 接口

功能描述：标准的 USB 接口。

接口描述：参考表 2-8。

表 2-8 USB HOST 接口


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

## 2.2.9 SD/MMC 卡接口

功能描述：标准的 SD/MMC 卡接口，采用插入、保护自动检测设计。

接口描述：参考表 2-9。

表 2-9 SD/MMC 卡接口

引脚	信号定义	功能描述	引脚描述
1	MINISD_CD1	Mini SD Card detect 1	
2	MINISD_CD2	Mini SD Card detect 2	
3	DAT2	MMC card data 2	
4	DAT3	MMC card data 3	
5	DAT4	MMC card data 4	
6	MINISD_DAT2	Mini SD card data 2	
7	GND	GND	
8	MINISD_DAT3	Mini SD card data 3	
9	DAT5	MMC card data 5	
10	MINISD_CMD	Mini SD card command	
11	VSS	GND	
12	MINISD_VSS	GND	
13	NC	NC	
14	VDD	VDD	
15	NC	NC	
16	MINISD_VDD	VDD	
17	CLK	MMC card clock	
18	MINISD_CLK	Mini SD card clock	
19	DAT6	MMC card data 6	
20	MINISD_VSS	GND	
21	VSS	GND	
22	MINISD_DAT0	Mini SD card data 0	
23	DAT7	MMC card data 7	
24	MINISD_DAT1	Mini SD card data 1	


25	DAT0	MMC card data 0	
26	DAT1	MMC card data 1	
27	SD_CD	SD Card detect	
28	SD_WP	SD write protect	
29	GND	GND	
30	GND	GND	

## 2.2.10 HDMI 接口

功能描述：标准的 HDMI 接口。

接口描述：参考表 2-10。

表 2-10 HDMI 接口

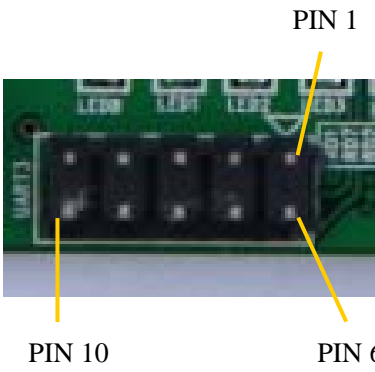
引脚	信号定义	功能描述	引脚描述
1	DAT2+	TMDS data 2+	 <p>PIN 1</p>
2	DAT2_S	TMDS data 2 shield	
3	DAT2-	TMDS data 2-	
4	DAT1+	TMDS data 1+	
5	DAT1_S	TMDS data 1 shield	
6	DAT1-	TMDS data 1-	
7	DAT0+	TMDS data 0+	
8	DAT0_S	TMDS data 0 shield	
9	DAT0-	TMDS data 0-	
10	CLK+	TMDS data clock+	
11	CLK_S	TMDS data clock shield	
12	CLK-	TMDS data clock-	
13	CEC	Consumer Electronics Control	
14	NC	NC	
15	SCL	IIC master serial clock	
16	SDA	IIC serial bidirectional data	
17	GND	GND	
18	5V	5V	
19	HPLG	Hot plug and play detect	

## 2.2.11 串口

功能描述：3 线串口排针。

接口描述：参考表 2-11。

表 2-11 串口

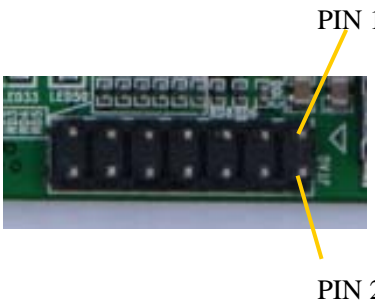
引脚	信号定义	功能描述	引脚描述
1	NC	NC	
2	TXD	Transit data	
3	RXD	Receive data	
4	NC	NC	
5	GND	GND	
6	NC	NC	
7	NC	NC	
8	NC	NC	
9	NC	NC	

### 2.2.12 JTAG 接口

功能描述：JTAG 接口排针。

接口描述：参考表 2-12。

表 2-12 JTAG 接口

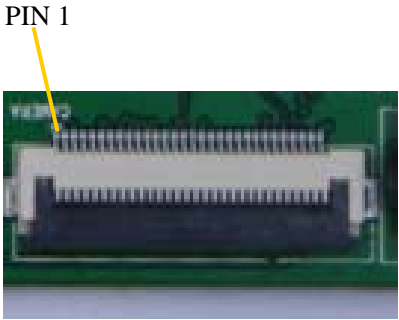
引脚	信号定义	功能描述	引脚描述
1	TMS	Test mode select	
2	NTRST	Test system reset	
3	TDI	Test data input	
4	GND	GND	
5	VIO	1.8V	
6	NC	NC	
7	TDO	Test data output	
8	GND	GND	
9	RTCK	Receive test clock	
10	GND	GND	
11	TCK	Test clock	
12	GND	GND	
13	EMU0	Test emulation 0	
14	EMU1	Test emulation 1	

### 2.2.13 摄像头接口

功能描述：Camera image sensor 接口。

接口描述：参考表 2-13。

表 2-13 摄像头接口

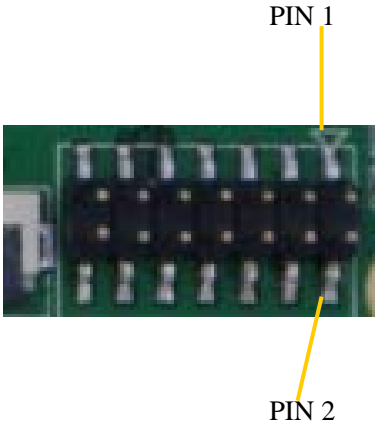
引脚	信号定义	功能描述	引脚描述
1	GND	GND	
2	D0	Digital image data bit 0	
3	D1	Digital image data bit 1	
4	D2	Digital image data bit 2	
5	D3	Digital image data bit 3	
6	D4	Digital image data bit 4	
7	D5	Digital image data bit 5	
8	D6	Digital image data bit 6	
9	D7	Digital image data bit 7	
10	D8	Digital image data bit 8	
11	D9	Digital image data bit 9	
12	D10	Digital image data bit 10	
13	D11	Digital image data bit 11	
14	GND	GND	
15	PCLK	Pixel clock	
16	GND	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	GND	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	GND	GND	
30	VDD18	1.8V	

### 2.2.14 键盘接口

功能描述：6X6 键盘接口排针。

接口描述：参考表 2-14。

表 2-14 键盘接口

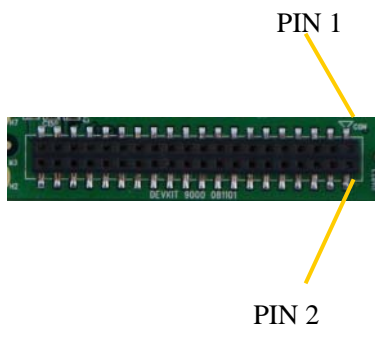
引脚	信号定义	功能描述	引脚描述
1	KC0	Keypad matrix column 0 output	
2	KR0	Keypad matrix row 0 input	
3	KC1	Keypad matrix column 1 output	
4	KR1	Keypad matrix row 1 input	
5	KC2	Keypad matrix column 2 output	
6	KR2	Keypad matrix row 2 input	
7	KC3	Keypad matrix column 3 output	
8	KR3	Keypad matrix row 3 input	
9	KC4	Keypad matrix column 4 output	
10	KR4	Keypad matrix row 4 input	
11	KC5	Keypad matrix column 5 output	
12	KR5	Keypad matrix row 5 input	
13	VDD18	1.8V	
14	GND	GND	

### 2.2.15 扩展接口

功能描述：自定义的各种扩展接口。

接口描述：参考表 2-15。

表 2-15 扩展接口

引脚	信号定义	功能描述	引脚描述
1	GND	GND	
2	BSP1_DX	Transmitted serial data 1	
3	BSP1_DR	Received serial data 1	
4	BSP1_CLKR	Received clock 1	
5	BSP1_FSX	Transmit frame synchronization 1	
6	BSP1_CLKX	Transmit clock 1	
7	BSP1_CLKS	External clock input 1	
8	BSP1_FSR	Receive frame synchronization 1	
9	UART1_CTS	UART1 clear to send	
10	UART1_RTS	UART1 request to send	
11	UART1_RX	UART1 receive data	
12	UART1_TX	UART1 transmit data	
13	GND	GND	
14	MMC2_CLK	MMC2 card clock	
15	MMC2_CMD	GND	
16	MMC2_D0	MMC2 card data 0	
17	MMC2_D1	MMC2 card data 1	
18	MMC2_D2	MMC2 card data 2	



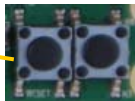
19	MMC2_D3	MMC2 card data 3
20	MMC2_D4	MMC2 card data 4
21	MMC2_D5	MMC2 card data 5
22	MMC2_D6	MMC2 card data 6
23	MMC2_D7	MMC2 card data 7
24	BSP3_DX	Transmitted serial data 3
25	BSP3_DR	Received serial data 3
26	BSP3_CLKX	Transmit clock 3
27	BSP3_FSX	Transmit frame synchronization 3
28	GND	GND
29	IIC3_SCL	IIC3 master serial clock
30	IIC3_SDA	IIC3 serial bidirectional data
31	SPI1_SIMO	Slave data in, master data out
32	SPI1_SOMI	Slave data out, master data in
33	SPI1_CLK	SPI1 clock
34	SPI1_CS0	SPI enable 0
35	SPI1_CS3	SPI enable 3
36	HDQ_SIO	Bidirectional HDQ
37	VDD33	3.3V
38	VDD18	1.8V
39	VDD50	5V
40	VDD50	5V

### 2.2.16 BOOT 按键

功能描述：启动引导按键。

接口描述：参考表 2-16。

表 2-16 BOOT 按键


引脚	信号定义	功能描述	引脚描述
1	Boot	System boot configuration	BOOT KEY 

### 2.2.17 RESET 按键

功能描述：Reset 按键。

接口描述：参考表 2-17。

表 2-17 RESET 按键

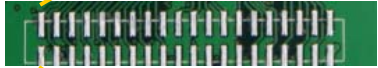
引脚	信号定义	功能描述	引脚描述
1	Reset	Power on reset	 RESET KEY

## 2.2.18 OMAP 总线

功能描述：OMAP 3530 总线。

接口描述：参考表 2-18。

表 2-18 OMAP 总线

引脚	信号定义	功能描述	引脚描述
1	GND	GND	
2	BDO0	GPMC data bit 0	
3	BDO1	GPMC data bit 1	
4	BDO2	GPMC data bit 2	
5	BDO3	GPMC data bit 3	
6	BDO4	GPMC data bit 4	
7	BDO5	GPMC data bit 5	
8	BDO6	GPMC data bit 6	
9	BDO7	GPMC data bit 7	
10	BDO8	GPMC data bit 8	
11	BDO9	GPMC data bit 9	
12	BDO10	GPMC data bit 10	
13	BDO11	GPMC data bit 11	
14	BDO12	GPMC data bit 12	
15	BDO13	GPMC data bit 13	
16	BDO14	GPMC data bit 14	
17	BDO15	GPMC data bit 15	
18	GND	GND	
19	NCS7	GMPM chip select bit 7	
20	NOE	Output enable	
21	NCS5	GMPM chip select bit 5	
22	NWE	Write enable	
23	NCS4	GMPM chip select bit 4	
24	CLE	Lower bytes enable. Also used for command latch enable	
25	NCS3	GMPM chip select bit 3	
26	NBE1	Upper byte enable	
27	ALE	Address valid or address latch enable	
28	GPMC_WAI T0	External indication of wait	
29	BAO2	GPMC memory address bit 2	
30	BAO1	GPMC memory address bit 1	
31	BAO4	GPMC memory address bit 4	
32	BAO3	GPMC memory address bit 3	
33	BAO6	GPMC memory address bit 6	
34	BAO5	GPMC memory address bit 5	


35	BAO8	GPMC memory address bit 8
36	BAO7	GPMC memory address bit 7
37	BAO10	GPMC memory address bit 10
38	BAO9	GPMC memory address bit 9
39	VDD33	3.3V
40	GND	GND

### 2.2.19 电源输出接口

功能描述：为外设提供电源输出。

接口描述：参考表 2-19。

表 2-19 电源输出接口

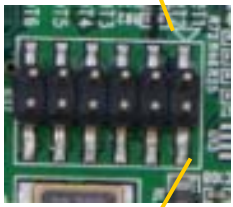
引脚	信号定义	功能描述	引脚描述
1	VDD50	5V output	 PIN 1
2	VDD42	4.2V output	
3	VDD33	3.3V output	
4	ADCIN	ADC input	
5	GND	GND	

### 2.2.20 引导模式跳线

功能描述：引导系统顺序的启动。

接口描述：参考表 2-20。

表 2-20 引导模式跳线


引脚	信号定义	功能描述	引脚描述
1	BOOT0	Boot configuration mode bit 0	 PIN 1 PIN 2
2	VDD18	1.8V	
3	BOOT1	Boot configuration mode bit 1	
4	VDD18	1.8V	
5	BOOT2	Boot configuration mode bit 2	
6	VDD18		
7	BOOT3	Boot configuration mode bit 3	
8	VDD18	1.8V	
9	BOOT4	Boot configuration mode bit 4	
10	VDD18	1.8V	
11	BOOT5	Boot configuration mode bit 5	
12	VDD18	1.8V	

### 2.2.21 TFT\_LCD 接口

功能描述：TFT\_LCD 接口。

接口描述：参考表 2-21。

表 2-21 TFT\_LCD 接口

引脚	信号定义	功能描述	引脚描述
1	DSS_D0	LCD Pixel data bit 0	 <p>PIN 1</p>
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_D 14	LCD Pixel data bit 14	
17	DSS_D15	LCD Pixel data bit 15	
18	GND	GND	
19	DSS_D16	LCD Pixel data bit 16	
20	DSS_D17	LCD Pixel data bit 17	
21	DSS_D18	LCD Pixel data bit 18	
22	DSS_D19	LCD Pixel data bit 19	
23	DSS_D20	LCD Pixel data bit 20	
24	DSS_D21	LCD Pixel data bit 21	
25	DSS_D22	LCD Pixel data bit 22	
26	DSS_D23	LCD Pixel data bit 23	
27	GND	GND	
28	DEN	AC bias control (STN) or pixel data enable (TFT)	
29	HSYNC	LCD Horizontal Synchronization	
30	VSYNC	LCD Vertical Synchronization	
31	GND	GND	
32	CLK	LCD Pixel Clock	
33	GND	GND	
34	X+	X+ Position Input	
35	X-	X- Position Input	
36	Y+	Y+ Position Input	
37	Y-	Y- Position Input	

38	SPI_CLK	SPI clock
39	SPI_MOSI	Slave data in, master data out
40	SPI_MISO	Slave data out, master data in
41	SPI_CS	SPI enable
42	IIC_CLK	IIC master serial clock
43	IIC_SDA	IIC serial bidirectional data
44	GND	GND
45	VDD18	1.8V
46	VDD33	3.3V
47	VDD50	5V
48	VDD50	5V
49	RESET	Reset
50	PWREN	Power on enable

## 第三部分 软件系统

### 第三章 软件系统概述

本章主要概述DevKit8000的软件系统，包括预装软件介绍，DevKit8000 BSP包规格说明以及DevKit8000出货光盘中提供各种资源的规格说明。

DevKit8000软件系统包括：预编译的映像及系统各部分源码，交叉编译工具，开发辅助工具等，均可从DevKit8000评估套件发行光盘中找到。

DevKit8000评估套件预装软件包括：

- x-loader----- (x-load.bin.ift\_for\_NAND)
- u-boot----- (flash-uboot.bin)
- 2.6 kernel----- (ulmage)
- rootfs----- (ubi.img)

另外，发行光盘中，还提供了以下程序和软件：

- 烧写用映像文件
- 交叉编译工具
- 系统各部分源码
- 用户测试程序及开发示例
- 用户在使用DevKit8000时可能会用到的一些工具

### 3.1 预装软件

DevKit8000出厂的时候，软件映像已经固化在FLASH上。完整的系统由x-loader、u-boot、kernel和rootfs四部分组成，系统结构如图3.1所示：

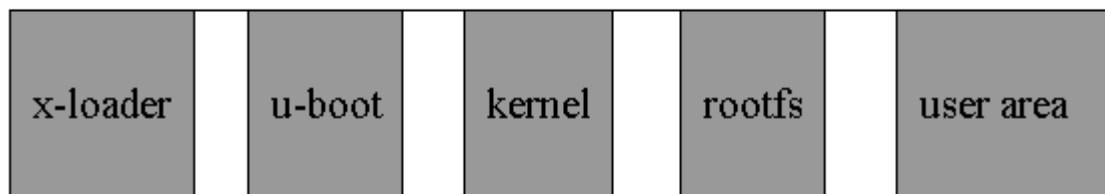


图3.1 System compose map

系统各组成部分特性及作用介绍如下：

- 1、x-loader是一级引导程序，系统上电后由CPU内部ROM自动拷贝到内部RAM并执行。主要作用为初始化CPU，拷贝u-boot到内存中，然后把控制权交给u-boot；
- 2、u-boot是二级引导程序，主要用于和用户进行交互，提供映像更新、引导内核等功能；
- 3、kernel使用最新2.6.x内核，根据DevKit8000进行定制；
- 4、rootfs采用开源文件系统，体积小，功能强大。

## 3.2 BSP features

DevKit8000 BSP包主要用于定制、生成运行于DevKit8000硬件平台上的Linux操作系统，用户基于该开发包进行二次开发。DevKit8000出厂光盘中所提供的BSP包中包括表3-1所示的内容。

表3-1 BSP规格

名称		备注	
BIOS	x-loader	NAND / ONENAND	
		MMC/SD	
		FAT	
	u-boot	NAND / ONENAND	
		MMC/SD	
		FAT	
Kernel	Linux-2.6.x	支持ROM/CRAM/EXT2/EXT3/FAT/NFS/ JFFS2/UBIFS等多种文件系统	
		serial	串口驱动
		rtc	硬件时钟驱动
		net	10/100M以太网卡DM9000驱动
Device Driver	flash	nand flash驱动(支持nand boot)	
	lcd	TFT LCD 驱动	
	touch screen	触摸屏控制器ads7846驱动	
	mmc/sd	mmc/sd控制器驱动	
	usb otg	Usb otg 2.0驱动(可配置为主/从设备)	
	dvi	支持dvi-d信号输出	
	s-video	支持s-video信号输出	
	keypad	6x6矩阵键盘驱动	
	led	用户led灯驱动	
	GUI	Angstrom	针对于嵌入式设备的桌面环境发行版
		Android	google android系统



## 第四章 从这里开始

### 4.1 准备

如图 4.1 所示连接好开发板与 PC，打开串行终端软件，例如：window 下的超级终端，linux 下的 minicom，设置属性如下：

```
波特率：115200
数据位：8
奇偶校验：无
停止位：1
流控制：无
```

朱工图片准备

图 4.1 开发板与 PC 连接图

接通电源，终端上会出现系统启动信息，稍等片刻，进入系统后即可开始体验Linux了。

### 4.2 LED测试

如图 4.2 所示，led0，led1，led2 为用户 led 灯。其中 led0 已用作系统心跳的指示，led1 已用作 SD 卡数据传输指示。

图 4.2 开发板的 LED 灯

以下示范如何操作用户 led 灯 led2:

1、终端中输入如下命令点亮 led2

```
root@DevKit8000:~# echo -n 1 >/sys/class/leds/led2/brightness
```

2、终端中输入如下命令熄灭 led2

```
root@DevKit8000:~# echo -n 0 >/sys/class/leds/led2/brightness
```

led2 会随着用户的操作进行亮灭。

## 4.3 KEYPAD测试

开发板扩展了 6x6 矩阵键盘接口，使用 `evtest` 工具可测试矩阵键盘工作是否正常：

```
root@DevKit8000:~# evtest /dev/input/event0
```

按下矩阵键盘的任意键，例如 ‘1’，终端上会出现类似提示：

```
Event: time 946684837.310027, type 1 (Key), code 2 (1), value 1  
Event: time 946684837.402160, type 1 (Key), code 2 (1), value 0
```

其中，“type 1 (Key), code 2 (1), value 1”，表明发生按键事件，键值为 2(对应全键盘的 ‘1’ 键，状态为按下( ‘0’ 表示按键提起))，

---

*注意：按 CONTROL+C 退出测试。*

---

## 4.4 触摸屏测试

1、输入以下指令执行触摸屏校准程序：

```
root@DevKit8000:~# ts_calibrate
```

按照屏幕上提示，点击 “+” 图标 5 次完成校准。

2、校准完成后，输入以下指令进行触摸屏测试：

```
root@DevKit8000:~# ts_test
```

按照屏幕提示，可选择画点、画线测试。

---

*注意：按 CONTROL+C 退出测试。*

---

## 4.5 RTC测试

开发板带硬件时钟，用于保存并恢复系统时间，可参考如下方法进行测试：

1、设置系统时间为 2008 年 8 月 8 日晚上 8 时正

```
root@DevKit8000:~# date 080820002008  
Fri Aug 8 20:00:00 UTC 2008
```

2、把系统时钟写入 RTC

```
root@DevKit8000:~# hwclock -w
```

3、读取 RTC

```
root@DevKit8000:~# hwclock  
Fri Aug 8 20:00:21 2008 0.000000 seconds
```

可以看到，硬件时钟 RTC 被设置成 2008 年 8 月 8 日，系统时钟被保存到硬件时钟里。

4、重启系统，输入以下命令恢复系统时钟

```
root@DevKit8000:~# hwclock -s  
root@DevKit8000:~# date  
Fri Aug 8 20:01:45 UTC 2008
```

可以看到，系统时间被恢复为硬件时间。

## 4.6 MMC/SD卡测试

插入 MMC/SD 卡，系统会自动检测并挂载 MMC/SD 卡到/media 目录下

```
root@DevKit8000:~# cd /media/
root@DevKit8000:/media# ls
card      hdd      mmcblk0p1  ram      union
cf        mmc1     net        realroot
root@DevKit8000:/media# cd mmcblk0p1/
```

## 4.7 USB OTG测试

### 1、USB OTG 作为 DEVICE 使用：

1) 系统起来后，使用 USB mini B to USB A 转接线连接开发板与 pc 机，其中 USB mini B 接口连接开发板，USB A 接口连接 pc 机。

---

*注意: Linux USB Ethernet/RNDIS Gadget 驱动的安装请参考附录一的说明。*

---

2) 成功连接后，PC 端会生成一虚拟网卡，如图 4.3 所示：



图 4.3 虚拟网卡

3) 配置虚拟网卡的 IP 地址，例如：

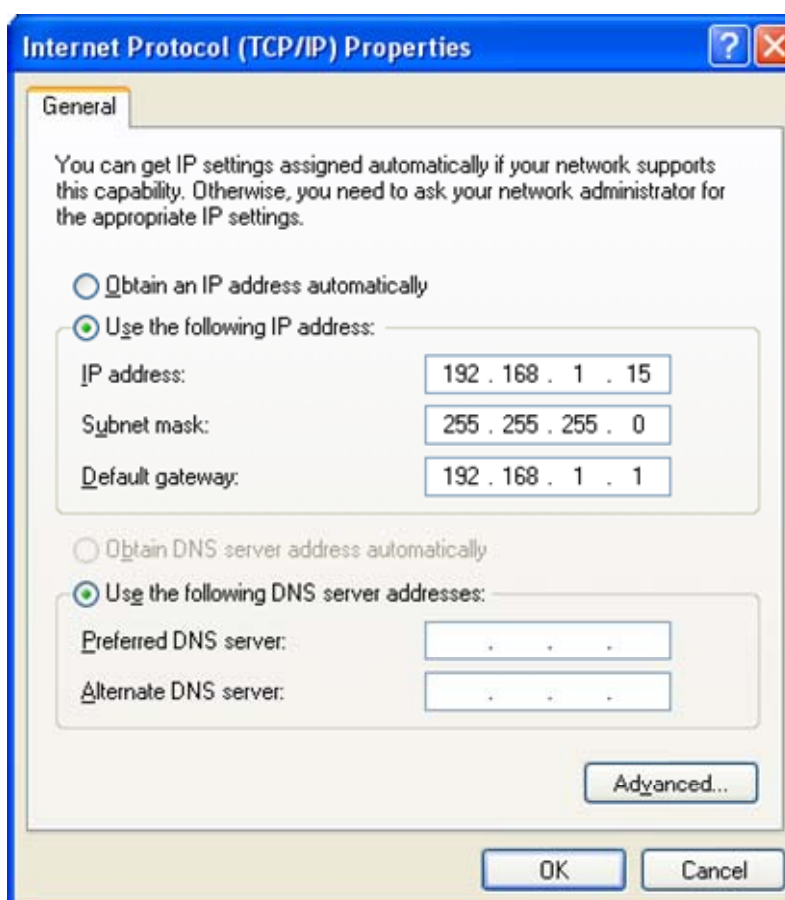


图 4.4 IP 设置

4) 配置开发板 usb 虚拟网卡的 IP 地址为同一网段，例如：

```
root@DevKit8000:~# ifconfig usb0 192.168.1.115
root@DevKit8000:~# ifconfig
lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            UP LOOPBACK RUNNING  MTU:16436  Metric:1
            RX packets:26 errors:0 dropped:0 overruns:0 frame:0
            TX packets:26 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:2316 (2.2 KiB)  TX bytes:2316 (2.2 KiB)

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

5) 测试：

```
root@DevKit8000:~# ping 192.168.1.15
PING 192.168.1.15 (192.168.1.15): 56 data bytes
64 bytes from 192.168.1.15: seq=0 ttl=128 time=0.885 ms
64 bytes from 192.168.1.15: seq=1 ttl=128 time=0.550 ms
```

## 2、USB OTG 作为 HOST 使用:

使用 USB mini A to USB A 转接线连接开发板与 USB 设备(例如 U 盘), 其中 USB mini A 接口连接开发板, USB A 接口连接 USB 设备。

系统起来的时候会检测并自动挂载 USB 设备到/media 目录下:

```
root@DevKit8000:~# cd /media/
root@DevKit8000:/media# ls
card      hdd      mmcblk0p1  ram      sda1
cf        mmc1     net        realroot union
root@DevKit8000:/media# cd sda1
```

---

注意: 某些 U 盘可能会被识别为 sda

---

## 4.8 AUDIO/VIDEO测试

板上带音频输入、输出接口, 目前实现的功能为音频输出。文件系统内带 mplayer 音/视频播放工具, 支持 wav、mp3、avi 等格式, 用户可使用如下命令进行测试:

```
root@DevKit8000:~# mplayer /sample_video.avi
```

插上耳机, 即可听到动听的立体声音乐, 接上屏可以看到演示视频的播放。

## 4.9 网络测试

板上带10/100M自适应网卡DM9000, 用户可把开发板接入局域网, 使用如下命令进行测试:

```
root@DevKit8000:~# ifconfig eth0 192.192.192.200
eth0: link down
root@DevKit8000:~# eth0: link up, 100Mbps, full-duplex, lpa 0x41E1

root@DevKit8000:~# ping 192.192.192.90
PING 192.192.192.90 (192.192.192.90): 56 data bytes
64 bytes from 192.192.192.90: seq=0 ttl=128 time=1.007 ms
64 bytes from 192.192.192.90: seq=1 ttl=128 time=0.306 ms
64 bytes from 192.192.192.90: seq=2 ttl=128 time=0.397 ms
64 bytes from 192.192.192.90: seq=3 ttl=128 time=0.367 ms

--- 192.192.192.90 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.306/0.519/1.007 ms
```

---

注意: 开发板的网卡 ip 地址应与 PC 机在同一网段, 例如 192.192.192.x

按 CONTROL+C 退出测试。

---

## 第五章 Linux 系统开发平台搭建

本章介绍如何利用DevKit8000 BSP包搭建运行于DevKit8000硬件平台上的Linux系统开发环境。具体内容包括交叉编译环境的搭建，系统映像的生成，并通过示例介绍系统的定制。

---

注意: 本文中使用的Linux发行版为ubuntu 7.10, 下文中简称为ubuntu。

---

### 5.1 交叉编译环境的搭建

用户在使用DevKit8000进行开发前，必须先搭建好ARM Linux交叉开发环境。下面以ubuntu操作系统为例，介绍交叉开发环境的搭建，其它Linux系统的操作与ubuntu系统类似。

#### 5.1.1 交叉编译工具的安装

插入光盘，ubuntu默认把光盘挂载到/media/cdrom目录下，交叉编译工具就存放在/media/cdrom/linux/tools目录下，名字为arm-2007q3-51-arm-none-linux-gnueabi.bin。

用户可执行如下命令，启动交叉编译工具的图形化安装：

```
cd /media/cdrom/linux/tools
./arm-2007q3-51-arm-none-linux-gnueabi.bin
```

根据出现的图形化界面提示进行安装即可，其中，以下界面选择工具的安装路径：

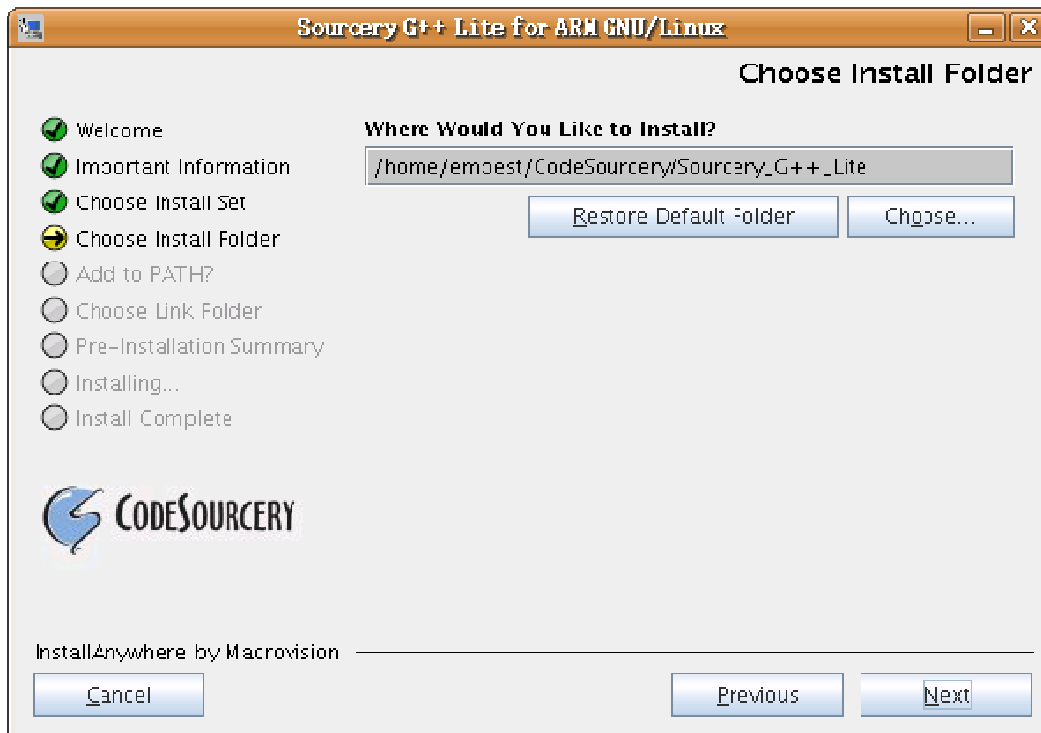


图 5.1 交叉编译工具的安装

---

注意: 默认安装到用户目录下, 本文以/home/embest 为准, 用户可适当换成自身目录即可。

---

### 5.1.2 其它工具的安装

源码编译中用到其它的一些工具, 同样存放在光盘的 linux/tools 目录下, 用户可执行以下命令一并安装:

```
mkdir /home/embest/tools
cp /media/cdrom/linux/tools/mkimage /home/embest/tools
cp /media/cdrom/linux/tools/signGP /home/embest/tools
cp /media/cdrom/linux/tools/mkfs.ubifs /home/embest/tools
cp /media/cdrom/linux/tools/ubinize /home/embest/tools
cp /media/cdrom/linux/tools/ubinize.cfg /home/embest/tools
```

### 5.1.3 添加环境变量

以上工具安装完成后, 还需要使用如下命令把它们添加到环境变量中:

```
export
PATH=/home/embest/CodeSourcery/Sourcery_G++_Lite/bin:/home/embest/tools:$PATH
```

---

注意: 用户可把它写入用户目录的.bashrc 文件中, 那么系统启动的时候自动完成环境变量的添加。

---

## 5.2 生成系统映像文件

### 5.2.1 准备

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

```
mkdir /home/embest/work
cd /home/embest/work
tar xvf /media/cdrom/linux/source/x-load-1.41.tar.bz2
tar xvf /media/cdrom/linux/source/u-boot-1.3.3.tar.bz2
tar xvf /media/cdrom/linux/source/linux-2.6.28-omap.tar.bz2
sudo tar xvf /media/cdrom/linux/source/rootfs.tar.bz2
```

执行完以上操作后, 当前目录下会生成 linux-2.6.22-omap、u-boot-1.3.3、x-load-1.41、rootfs 四个目录。

### 5.2.1 x-loader 映像生成

DevKit8000 支持 MMC/SD 启动或 NAND 启动, 不同的启动方式烧写的 x-loader 的映像文件是不一



样的，对应的映射生成方法也不同。

下面分别介绍用于不同启动方式下的 x-loader 映像文件的生成。

#### 1、生成用于 SD 卡启动的 x-loader 映像文件 MLO

```
cd x-load-1.41
make distclean
make omap3devkit8000_config
make
signGP x-load.bin
mv x-load.bin.ift MLO
```

执行完以上操作后，当前目录下会生成我们需要的 MLO 文件。

#### 2、生成用于 NAND 启动的 x-load.bin.ift\_for\_NAND

1) 修改 x-loader-1.4.1/include/configs/omap3devkit8000.h 文件，注释以下行：

```
///#define CFG_CMD_MMC 1
```

2) 交叉编译

```
cd x-load-1.41
make distclean
make omap3devkit8000_config
make
signGP x-load.bin
mv x-load.bin.ift x-load.bin.ift_for_NAND
```

执行完以上操作后，当前目录下会生成我们需要的 x-load.bin.ift\_for\_NAND 文件。

### 5.2.2 u-boot 映像生成

```
cd u-boot-1.3.3
make distclean
make omap3devkit8000_config
make
```

执行完以上操作后，当前目录下会生成我们需要的 u-boot.bin 文件。

### 5.2.3 kernel 映像生成

```
cd linux-2.6.28-omap
make distclean
make omap3_devkit8000_defconfig
make uImage
```

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

### 5.2.4 ubifs 映像生成

```
cd /home/embest/work
sudo mkfs.ubifs -r rootfs -m 2048 -e 129024 -c 812 -o ubifs.img
```

```
sudo ubinize -o ubi.img -m 2048 -p 128KiB -s 512 /home/embest/tools/ubinize.cfg
```

执行完以上操作后，当前目录下会生成我们需要的ubi.img文件。

## 5.3 系统定制

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

### 5.3.1 修改内核配置

出厂内核源码中提供有默认配置文件：arch/arm/configs/omap3\_devkit8000\_defconfig  
用户可在其基础上进行系统定制：

```
cd linux-2.6.28-omap
cp arch/arm/configs/omap3_devkit8000_defconfig .config
make menuconfig
```

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

#### 1、选择 Device drivers

```
General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
System Type --->
Bus support --->
Kernel Features --->
Boot options --->
CPU Power Management --->
Floating point emulation --->
Userspace binary formats --->
Power management options --->
[*] Networking support --->
Device Drivers --->
File systems --->
Kernel hacking --->
Security options --->
-* Cryptographic API --->
Library routines --->
---
Load an Alternate Configuration File
Save an Alternate Configuration File

<Select> <Exit> <Help>
```

## 2、选择 USB support

```
[ ] ISDN support --->
  Input device support --->
  Character devices --->
<*) I2C support --->
[*] SPI support --->
-* GPIO Support --->
<> Dallas's 1-wire support --->
<> Power supply class support --->
<> Hardware Monitoring support --->
<> Generic Thermal sysfs driver --->
[ ] Watchdog Timer Support --->
  Sonics Silicon Backplane --->
  Multifunction device drivers --->
  Multimedia devices --->
  Graphics support --->
<*) Sound card support --->
[*] HID Devices --->
[+] USB support --->
<*) MMC/SD/SDIO card support --->
<> Sony MemoryStick card support (EXPERIMENTAL) --->
[ ] Accessibility support --->
[*] LED Support --->
<*) Real Time Clock --->
[ ] DMA Engine support --->
[ ] Voltage and Current Regulator Support --->
<> Userspace I/O drivers --->
  CBUS support --->
```

**<Select>**   < Exit >   < Help >

## 3、选择 USB Gadget Support

```

<> EMI 6|2m USB Audio interface support
<> EMI 2|6 USB Audio interface support
<> ADU devices from Ontrak Control Systems
<> USB 7-Segment LED Display
<> USB Diamond Rio500 support
<> USB Lego Infrared Tower support
<> USB LCD driver support
<> USB BlackBerry recharge support
<> USB LED driver support
<> Cypress CY7C63xxx USB driver support
<> Cypress USB thermometer driver support
<> USB Phidgets drivers
<> Siemens ID USB Mouse Fingerprint sensor support
<> Elan PCMCIA CardBus Adapter USB Client
<> Apple Cinema Display support
<> USB 2.0 SVGA dongle support (Net2280/SiS315)
<> USB LD driver
<> PlayStation 2 Trance Vibrator driver support
<> IO Warrior driver support
<> USB testing driver
<> iSight firmware loading support
<> USB VST driver
<+> USB Gadget Support --->
    *** OTG and related infrastructure ***
<> GPIO based peripheral-only VBUS sensing 'transceiver'
<> Philips ISP1301 with OMAP OTG
<*> TWL4030 USB Transceiver Driver

<Select> < Exit > < Help >

```

#### 4、修改 USB Gadget Support 配置如下

```

--- USB Gadget Support
[ ] Debugging messages (DEVELOPMENT)
[ ] Debugging information files (DEVELOPMENT)
(2) Maximum VBUS Power usage (2-500 mA)
USB Peripheral Controller (Inventra HDRC USB Peripheral (TI, ADI, ...)) --->
<H> USB Gadget Drivers
<> Gadget Zero (DEVELOPMENT)
<> Ethernet Gadget (with CDC Ethernet support)
<> Gadget Filesystem (EXPERIMENTAL)
<+> File-backed Storage Gadget
[ ] File-backed Storage Gadget testing version (NEW)
<> Serial Gadget (with CDC ACM and CDC OBEX support)
<> MIDI Gadget (EXPERIMENTAL)
<> Printer Gadget
<> CDC Composite Device (Ethernet and ACM)

<Select> < Exit > < Help >

```

### 5.3.2 编译

保存配置，执行以下命令重新编译内核：

```
make
make uImage
```

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

### 5.3.2 测试

更新 SD 卡下内核映像文件 zImage，拷贝 g\_file\_storage.ko 文件到 sd 卡中，重新启动系统，执行如下命令把 sd 卡模拟成 usb mass storage device 供 PC 访问：

```
root@DevKit8000:~# cd /media/mmcblk0p1/
root@DevKit8000:/media/mmcblk0p1# insmod g_file_storage.ko file=/dev/mmcblk0p1
stall=0 removable=1
g_file_storage gadget: File-backed Storage Gadget, version: 7 August 2007
g_file_storage gadget: Number of LUNs=1
g_file_storage gadget-lun0: ro=0, file: /dev/mmcblk0p1
musb_hdrc musb_hdrc: MUSB HDRC host driver
musb_hdrc musb_hdrc: new USB bus registered, assigned bus number 2
usb usb2: configuration #1 chosen from 1 choice
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 1 port detected
```

使用 USB mini B to USB A 转接线连接开发板与 pc 机,PC 上会提示发现 usb mass storage device，然后出现新的移动硬盘，用户可对其进行正常操作。

---

*注意：请务必更新内核映像，否则 g\_file\_storage.ko 模块会加载失败，出现类似提示：*

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

---

## 第六章 映像更新与恢复

本章介绍DevKit8000 映像的烧写与更新。

在对映像进行更新之前请先准备好表6-1所示的相关设备及映像文件。

表6-1 映像更新的准备

设备名称	数量	备注
硬件	1台	公用设备
PC 机（最低配置） CPU: Intel PIII 500MHz 内存: 512MB 硬盘: 40GB 系统: Windows XP or higher 串口: 1 个		
DevKit8000	1块	
PC电源	1个	
串口连接线	1条	
SD卡	1个	存放映像文件
系统映像		一级引导程序
MLO、x-load.bin.ift_for_NAND		
u-boot.bin、flash-uboot.bin		二级引导程序
ulmage		内核
ramdisk.gz、ubi.img		文件系统

*注意：除了以上的配件之外，用户还需要自行将 DEVKIT8000 上的需要用到的通信端口扩展出来方可与 PC 机进行相连，具体信号请参考硬件手册的相关接口的定义。*

### 6.1 设备连接及设置

图6.1为DevKit8000映像更新的设备连接示意图，不同的更新方式连接方法有所不同。

朱工准备

图6.1 设备连接示意图

### 6.2 映像更新

DevKit8000 支持从 MMC/SD 或 NAND 启动，不同启动方式下映像的更新方法有所不同。下面分

别介绍不同启动方式下启动映像的更新。

### 6.2.1 SD 卡启动映像更新

直接在 pc 下用编译生产的映像文件替换 sd 卡下对应映像文件即可

---

*注意: 可适当对原有映像文件进行备份。*

---

### 6.2.2 NAND 启动映像更新

Nand 启动映像的更新需要借助于 u-boot 来完成。u-boot 下可通过 MMC/SD 或 TFTP 两种方式对 nand 启动映像进行更新。下面分别介绍这两种更新方法。

#### 1、SD卡更新方式

可把需要更新的映像存放到SD中，u-boot启动后通过SD卡更新nand flash下映像

##### 1) x-loader启动映像的更新

```
OMAP3 DevKit8000 # mmcinit
OMAP3 DevKit8000 # fatload mmc 0:1 80000000 x-load.bin.ift_for_NAND
reading x-load.bin.ift_for_NAND

9664 bytes read
OMAP3 DevKit8000 # nand unlock
device 0 whole chip
nand_unlock: start: 00000000, length: 134217728!
NAND flash successfully unlocked
OMAP3 DevKit8000 # nand ecc hw
OMAP3 DevKit8000 # nand erase 0 80000

NAND erase: device 0 offset 0x0, size 0x80000
Erasing at 0x60000 -- 100% complete.
OK
OMAP3 DevKit8000 # nand write.i 80000000 0 80000

NAND write: device 0 offset 0x0, size 0x80000

Writing data at 0x7f800 -- 100% complete.
524288 bytes written: OK
```

##### 2) u-boot启动映像的更新

```
OMAP3 DevKit8000 # mmcinit
OMAP3 DevKit8000 # fatload mmc 0:1 80000000 flash-uboot.bin
reading flash-uboot.bin

1085536 bytes read
OMAP3 DevKit8000 # nand unlock
```

```
device 0 whole chip
nand_unlock: start: 00000000, length: 134217728!
NAND flash successfully unlocked
OMAP3 DevKit8000 # nand ecc sw
OMAP3 DevKit8000 # nand erase 80000 160000

NAND erase: device 0 offset 0x80000, size 0x160000
Erasing at 0x1c0000 -- 100% complete.
OK
OMAP3 DevKit8000 # nand write.i 80000000 80000 160000

NAND write: device 0 offset 0x80000, size 0x160000

Writing data at 0x1df800 -- 100% complete.
1441792 bytes written: OK
```

### 3) 内核映像更新

```
OMAP3 DevKit8000 # mmcinit
OMAP3 DevKit8000 # fatload mmc 0:1 80000000 ulmage
reading ulmage

1991900 bytes read
OMAP3 DevKit8000 # nand unlock
device 0 whole chip
nand_unlock: start: 00000000, length: 268435456!
NAND flash successfully unlocked
OMAP3 DevKit8000 # nand ecc sw
OMAP3 DevKit8000 # nand erase 280000 200000

NAND erase: device 0 offset 0x280000, size 0x200000
Erasing at 0x460000 -- 100% complete.
OK
OMAP3 DevKit8000 # nand write.i 80000000 280000 200000

NAND write: device 0 offset 0x280000, size 0x200000

Writing data at 0x47f800 -- 100% complete.
2097152 bytes written: OK
```

### 4) 文件系统映像更新

```
OMAP3 DevKit8000 # mmcinit
OMAP3 DevKit8000 # fatload mmc 0:1 80000000 ubi.img
reading ubi.img
```



```
12845056 bytes read
OMAP3 DevKit8000 # nand unlock
device 0 whole chip
nand_unlock: start: 00000000, length: 268435456!
NAND flash successfully unlocked
OMAP3 DevKit8000 # nand ecc sw
OMAP3 DevKit8000 # nand erase 680000 7980000

NAND erase: device 0 offset 0x680000, size 0x7980000
Erasing at 0x7fe0000 -- 100% complete.
OK
OMAP3 DevKit8000 # nand write.i 80000000 680000 $(filesize)

NAND write: device 0 offset 0x680000, size 0xc40000

Writing data at 0x12bf800 -- 100% complete.
12845056 bytes written: OK
```

## 2、tftp更新方式

板上配置有10/100M自适应网卡dm9000，用户可通过tftp下载的方式更新nand启动映像

---

*注意: tftp 服务器的搭建与配置请参考附录 3*

---

### 1) x-loader启动映像的更新

```
OMAP3 DevKit8000 # tftp 80000000 x-load.bin.ift_for_NAND
dm9000 i/o: 0x2c000000, id: 0x90000a46
MAC: aa:bb:cc:dd:ee:ff
operating at 100M full duplex mode
TFTP from server 192.192.192.90; our IP address is 192.192.192.200
Filename 'x-load.bin.ift_for_NAND'.
Load address: 0x80000000
Loading: T #
done
Bytes transferred = 9664 (25c0 hex)
OMAP3 DevKit8000 # nand unlock
device 0 whole chip
nand_unlock: start: 00000000, length: 134217728!
NAND flash successfully unlocked
OMAP3 DevKit8000 # nand ecc hw
OMAP3 DevKit8000 # nand erase 0 80000

NAND erase: device 0 offset 0x0, size 0x80000
```

```
Erasing at 0x60000 -- 100% complete.
OK
OMAP3 DevKit8000 # nand write.i 8000000 0 80000

NAND write: device 0 offset 0x0, size 0x80000

Writing data at 0x7f800 -- 100% complete.
524288 bytes written: OK
```

### 2) u-boot启动映像的更新

```
OMAP3 DevKit8000 # tftp 80000000 flash-uboot.bin
dm9000 i/o: 0x2c000000, id: 0x90000a46
MAC: aa:bb:cc:dd:ee:ff
operating at 100M full duplex mode
TFTP from server 192.192.192.90; our IP address is 192.192.192.200
Filename 'flash-uboot.bin'.
Load address: 0x80000000
Loading: T #####
          #####
done
Bytes transferred = 1085536 (109060 hex)
OMAP3 DevKit8000 # nand unlock
device 0 whole chip
nand_unlock: start: 00000000, length: 134217728!
NAND flash successfully unlocked
OMAP3 DevKit8000 # nand ecc sw
OMAP3 DevKit8000 # nand erase 80000 160000

NAND erase: device 0 offset 0x80000, size 0x160000
Erasing at 0x1c0000 -- 100% complete.
OK
OMAP3 DevKit8000 # nand write.i 80000000 80000 160000

NAND write: device 0 offset 0x80000, size 0x160000

Writing data at 0x1df800 -- 100% complete.
1441792 bytes written: OK
```

### 3) 内核映像更新

```
OMAP3 DevKit8000 # tftp 80000000 ulmage
dm9000 i/o: 0x2c000000, id: 0x90000a46
MAC: aa:bb:cc:dd:ee:ff
operating at 100M full duplex mode
TFTP from server 192.192.192.90; our IP address is 192.192.192.200
```



```
#####  
#####  
done  
Bytes transferred = 12845056 (c40000 hex)  
OMAP3 DevKit8000 # nand unlock  
device 0 whole chip  
nand_unlock: start: 00000000, length: 268435456!  
NAND flash successfully unlocked  
OMAP3 DevKit8000 # nand ecc sw  
OMAP3 DevKit8000 # nand erase 680000 7980000  
  
NAND erase: device 0 offset 0x680000, size 0x7980000  
Erasing at 0x7fe0000 -- 100% complete.  
OK  
OMAP3 DevKit8000 # nand write.i 8000000 680000 $(filesize)  
  
NAND write: device 0 offset 0x680000, size 0xc40000  
  
Writing data at 0x12bf800 -- 100% complete.  
12845056 bytes written: OK
```

## 6.3 系统恢复

系统镜像遭到破坏或需要更换 MMC/SD 存储卡时，请按照以下步骤进行系统恢复：

### 6.3.1 格式化 MMC/SD 卡

推荐使用 HP USB Disk Storage Format Tool 2.0.6:

<http://selfdestruct.net/misc/usbboot/SP27213.exe>

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

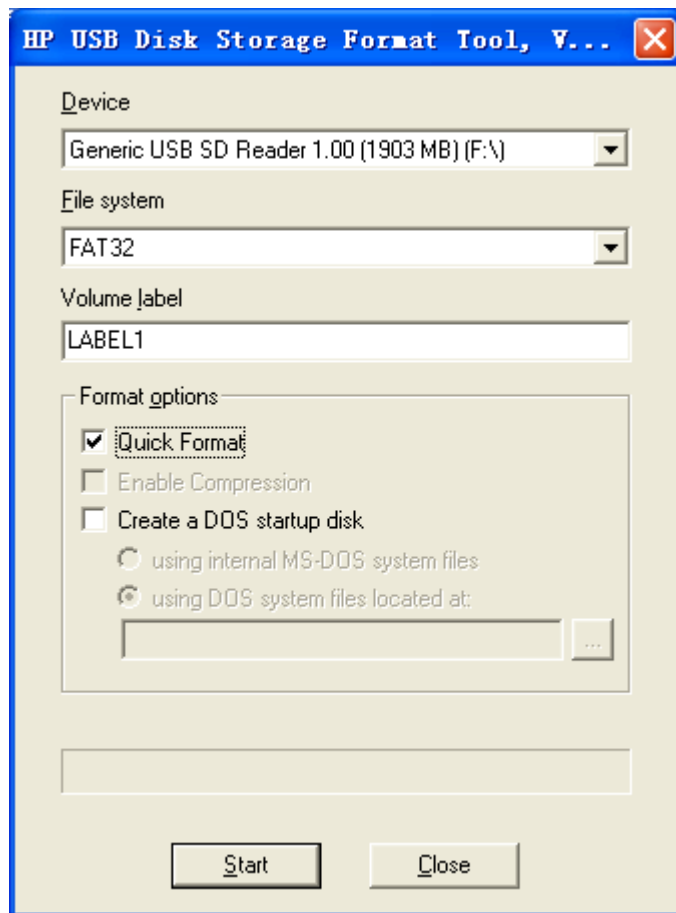


图 6.2 HP USB Disk 格式化工具

- 3、选择“FAT32”系统格式
- 4、点击“Start”
- 5、等待格式化完成，点击“OK”

### 6.3.1 系统映像恢复

- 1、准备好 SD 启动映像

拷贝光盘的 linux/image/sdcard 目录下系统映像文件到 SD 卡中。

- 2、准备好 flash 烧写映像

拷贝光盘的 linux/image/nandflash 目录下系统映像文件到 SD 卡中。

- 3、按住 2.2.16 所示的“BOOT 按键”，重新上电。进入 u-boot 后，参考 6.2.2 “NAND 启动映像更新”一节重新烧写系统映像。

### 6.3.2 u-boot 参数恢复

u-boot 交互模式下，用户可执行以下命令恢复出厂参数：

```
OMAP3 DevKit8000 # setenv bootargs console=ttyS2,115200n8 ubi.mtd=4
root=ubi0:rootfs rootfstype=ubifs
OMAP3 DevKit8000 # setenv bootcmd nand read 80300000 280000 200000\;bootm
80300000
OMAP3 DevKit8000 # saveenv
```

## 第七章 DevKit8000 应用开发示例

本章主要介绍如何在DevKit8000硬件平台上进行应用程序的开发，包括DevKit8000软件环境的搭建等，并通过实例来说明在DevKit8000进行应用程序开发的一般流程。

### 7.1 开发环境的准备

请参考第五章“Linux 系统开发平台搭建”

### 7.2 LED Project

#### 7.2.1 编写代码

led\_acc.c 源码，控制开发板上的三个 led 灯按累加器的方式闪烁。

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

#define LED0 "/sys/class/leds/led0/brightness"
#define LED1 "/sys/class/leds/led1/brightness"
#define LED2 "/sys/class/leds/led2/brightness"

int main(int argc, char *argv[])
{
    int f_led0, f_led1, f_led2;
    unsigned char i = 0;
    unsigned char dat0, dat1, dat2;
    if((f_led0 = open(LED0, O_RDWR)) < 0){
        printf("error in open %s",LED0);
        return -1;
    }
    if((f_led1 = open(LED1, O_RDWR)) < 0){
        printf("error in open %s",LED1);
        return -1;
    }
    if((f_led2 = open(LED2, O_RDWR)) < 0){
        printf("error in open %s",LED2);
        return -1;
    }
    for(;;){
```

```
    i++;  
    dat0 = i&0x1 ? '1':'0';  
    dat1 = (i&0x2)>>1 ? '1':'0';  
    dat2 = (i&0x4)>>2 ? '1':'0';  
    write(f_led0, &dat0, sizeof(dat0));  
    write(f_led1, &dat1, sizeof(dat1));  
    write(f_led2, &dat2, sizeof(dat2));  
    usleep(300000);  
}  
}
```

### 7.2.2 交叉编译

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

### 7.2.3 下载运行

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

```
./led_acc &
```

## 第八章 DevKit8000 Demo 演示

### 8.1 angstrom(GPE)桌面发布版本演示

1、把 SD 卡按附录 2 格式化为两个分区，重新挂载 SD 卡，然后执行如下命令。

```
cp /media/cdrom/linux/demo/angstrom/MLO /media/LABEL1
cp /media/cdrom/linux/demo/angstrom/u-boot.bin /media/LABEL1
cp /media/cdrom/linux/demo/angstrom/uImage /media/LABEL1
rm -rf /media/LABEL2/*
sudo tar jxvf
linux/demo/angstrom/Angstrom-DevKit8000-demo-image-glibc-ipk-2008.1-test-2
0080111-DevKit8000.rootfs.tar.bz2 -C /media/LABEL2
sync
umount /media/LABEL1
umount /media/LABEL2
```

2、取出 SD 卡插到开发板，启动 u-boot 后把内核启动参数设置为好下所示

对于 LCD 屏：

```
OMAP3 DevKit8000 # set bootargs mem=128M console=ttyS2,115200n8
root=/dev/mmcblk0p2 rw noinitrd rootdelay=1
```

对于 DVI 显示器：

```
OMAP3 DevKit8000 # set bootargs mem=128M console=ttyS2,115200n8
root=/dev/mmcblk0p2 rw noinitrd rootdelay=1 video=omapfb:mode:720p60
```

3、执行如下命令即可进入桌面系统：

```
OMAP3 DevKit8000 # mmcinit
OMAP3 DevKit8000 # fatload mmc 0 80300000 uimage
OMAP3 DevKit8000 # bootm 80300000
```

---

*注意：第一次进入桌面系统时，会对系统作大量配置，请耐心等待几分钟。以后启动即可快速进入桌面环境。*

---

### 8.2 google android 系统演示

1、把 SD 卡按附录 2 格式化为两个分区，重新挂载 SD 卡，然后执行如下命令。

```
cp /media/cdrom/linux/demo/android/MLO /media/LABEL1
cp /media/cdrom/linux/demo/android/u-boot.bin /media/LABEL1
cp /media/cdrom/linux/demo/android/uImage /media/LABEL1
rm -rf /media/LABEL2/*
sudo tar jxvf linux/demo/ android/RFS.tar.bz2 -C /media/LABEL2
sync
```



```
umount /media/LABEL1  
umount /media/LABEL2
```

2、取出 SD 卡插到开发板，启动 u-boot 后把内核启动参数设置为如下所示

对于 LCD 屏：

```
OMAP3 DevKit8000 # setenv bootargs console=ttyS2,115200n8 noinitrd  
root=/dev/mmcblk0p2 init=/init rootdelay=1
```

对于 DVI 显示器：

```
OMAP3 DevKit8000 # setenv bootargs console=ttyS2,115200n8 noinitrd  
root=/dev/mmcblk0p2 rootdelay=1 video=omapfb:mode:4.3inch_LCD:720p60
```

3、执行如下命令即可进入 android 系统：

```
OMAP3 DevKit8000 # mmcinit  
OMAP3 DevKit8000 # fatload mmc 0 80300000 uimage  
OMAP3 DevKit8000 # bootm 80300000
```

---

*注意：第一次进入 android 系统时，会对系统作大量配置，请耐心等待两三分钟。以后启动即可快速进入 android 系统。*

---

## 第九章 FAQ

**Q:**开发驱动时如何打开低层调试信息?

**A:**make menuconfig -> kernel hacking -> lowlevel debug

**Q:**如何让显示输出在 lcd 屏与 DVI 输出之间切换?

**A:**只需修改内核启动参数即可.

LCD 屏:set bootargs mem=128M console=ttyS2,115200n8 root=/dev/mmcblk0p2 rw noinitrd rootdelay=1

DVI 显示器:set bootargs mem=128M console=ttyS2,115200n8 root=/dev/mmcblk0p2 rw noinitrd rootdelay=1  
video=omapfb:mode:720p60

**Q:**怎么让 USB OTG 口工作于主从方式?

**A:**在内核启动前,如果让 USB OTG 工作于主方式,在板载 USB OTG 接口接附送的 USB A 型线.如果要让 USB OTG 工作于从方式,则在板载 USB OTG 接口接附送的 USB B 型线.

**Q:**系统默认先从 NAND 启动,再从 MMC/SD 启动.那么,NAND 写入启动映像后,如何切换到 MMC/SD 启动?

**A:**按住 BOOT 按键,上电启动,系统改为先从 MMC/SD 启动,再从 NAND 启动,详细请参考“2.2.16 BOOT 按键”介绍。

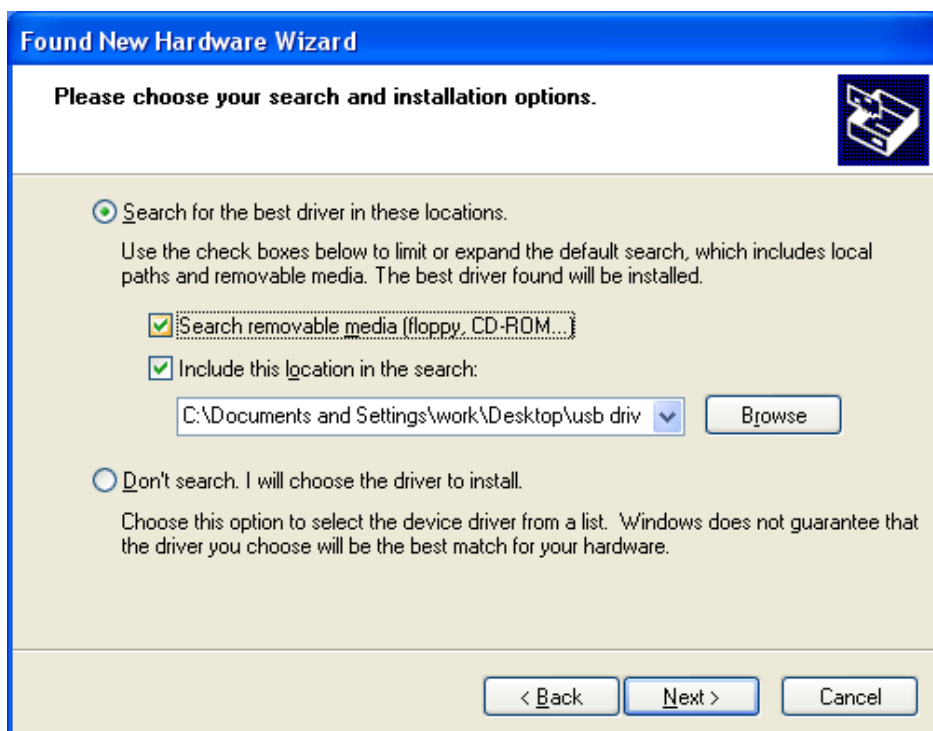
## 附录一

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

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



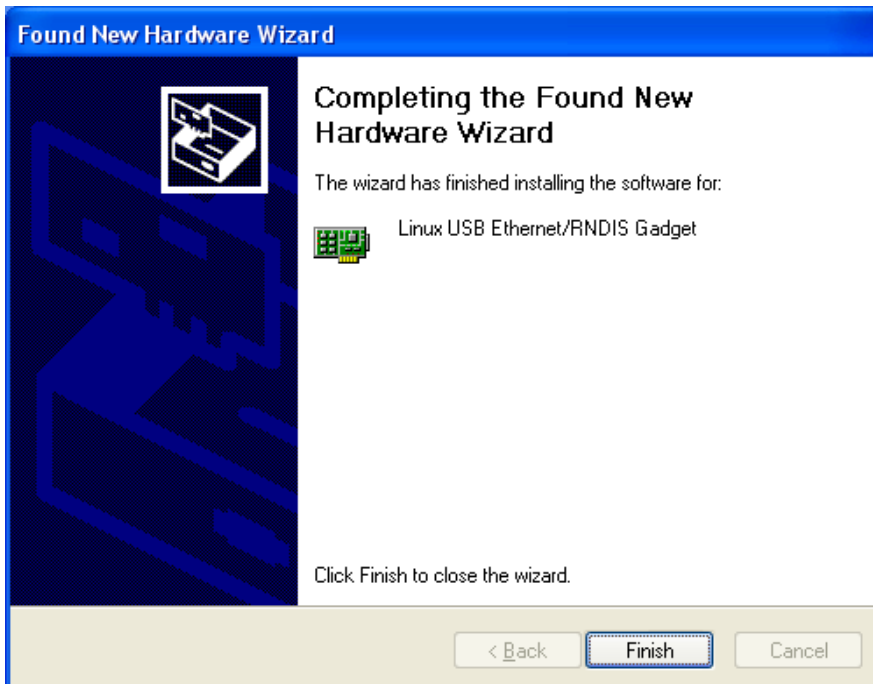
2、指定 usb 驱动路径，然后点击“下一步”。



3、出现以下提示时，选择继续安装



4、等待驱动安装完毕



## 附录二

### Linux Boot Disk Format

How to create a dual-partition card for DevKit8000 to boot Linux from first partition and have root file system at second partition.

#### 一、Introduction

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

#### 二、Details

Text marked with [] shows user input.

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

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

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

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

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

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

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

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

##### 3、If so, unmount it

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

##### 4、Start fdisk

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

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

## 5、Print the partition record

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

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

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

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

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

Device	Boot	Start	End	Blocks	Id	System
/dev/sdc1	*	1	246	1974240+	c	W95 FAT32 (LBA)

```
Partition 1 has different physical/logical endings:
```

```
phys=(244, 254, 63) logical=(245, 200, 19)
```

## 6、Delete any partitions that are there already

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

```
Selected partition 1
```

## 7、Set the Geometry of the SD Card

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

### 1) 、Go into expert mode.

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

### 2) 、Set the number of heads to 255.

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

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

### 3) Set the number of sectors to 63.

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

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

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

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

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

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

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

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

### 6) Return to Normal mode.

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

## 8、Print the partition record to check your work

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

```
Disk /dev/sdc: 2021 MB, 2021654528 bytes  
255 heads, 63 sectors/track, 245 cylinders  
Units = cylinders of 16065 * 512 = 8225280 bytes
```

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

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

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

```
Command action
```

```
  e   extended  
  p   primary partition (1-4)
```

```
[p]
```

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

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

```
Using default value 1
```

```
Last cylinder or +size or +sizeM or +sizeK (1-245, default 245): [+50]
```

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

```
Selected partition 1
```

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

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

## 10、Mark it as bootable

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

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

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

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

```
Command action
```

```
  e   extended  
  p   primary partition (1-4)
```

```
[p]
```

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

```
First cylinder (52-245, default 52): [(press Enter)]
```

```
Using default value 52
```

```
Last cylinder or +size or +sizeM or +sizeK (52-245, default 245): [(press Enter)]
```

```
Using default value 245
```

## 12、Print to Check Your Work

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

```
Disk /dev/sdc: 2021 MB, 2021654528 bytes
255 heads, 63 sectors/track, 245 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sdc1	*	1	51	409626	c	W95 FAT32 (LBA)
/dev/sdc2		52	245	1558305	83	Linux

### 13. Save the new partition records on the SD Card

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

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

```
The partition table has been altered!
```

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

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

```
The kernel still uses the old table.
```

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

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

```
Syncing disks.
```

### 14. Format the partitions

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

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

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

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

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

```
Filesystem label=
```

```
OS type: Linux
```

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

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

```
195072 inodes, 389576 blocks
```

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

```
First data block=0
```

```
Maximum filesystem blocks=402653184
```

```
12 block groups
```

```
32768 blocks per group, 32768 fragments per group
```



16256 inodes per group

Superblock backups stored on blocks:

32768, 98304, 163840, 229376, 294912

Writing inode tables: done

Creating journal (8192 blocks): done

Writing superblocks and filesystem accounting information:

---

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

---

## 附录三

### tftp 服务器搭建

#### 1、 安装客户端

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

#### 2、 安装 inet

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

#### 3、 服务器配置

首先，在根目录下建一个 tftpboot，并把属性改成任意用户可读写：

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

其次，在/etc/inetd.conf 里添加：

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

然后，重新加载 inetd 进程：

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

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

```
$>cd /etc/xinetd.d/ //进入目录 /etc/xinetd.d/  
$>sudo touch tftp //新建文件 tftp  
$>sudo vi tftp //编辑文件 tftp,把下面内容加入 tftp 文件中  
service tftp  
{  
    disable = no  
    socket_type = dgram  
    protocol = udp  
    wait = yes  
    user = root  
    server = /usr/sbin/in.tftpd  
    server_args = -s /tftpboot -c  
    per_source = 11  
    cps = 100 2  
}
```

#### 4、 重新启动服务：

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

#### 5、 测试服务器

测试一下，在/tftpboot 文件夹下新建一个文件

```
$>touch abc
```

进入另外一个文件夹

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

如果可以下载说明服务器已经安装成功。