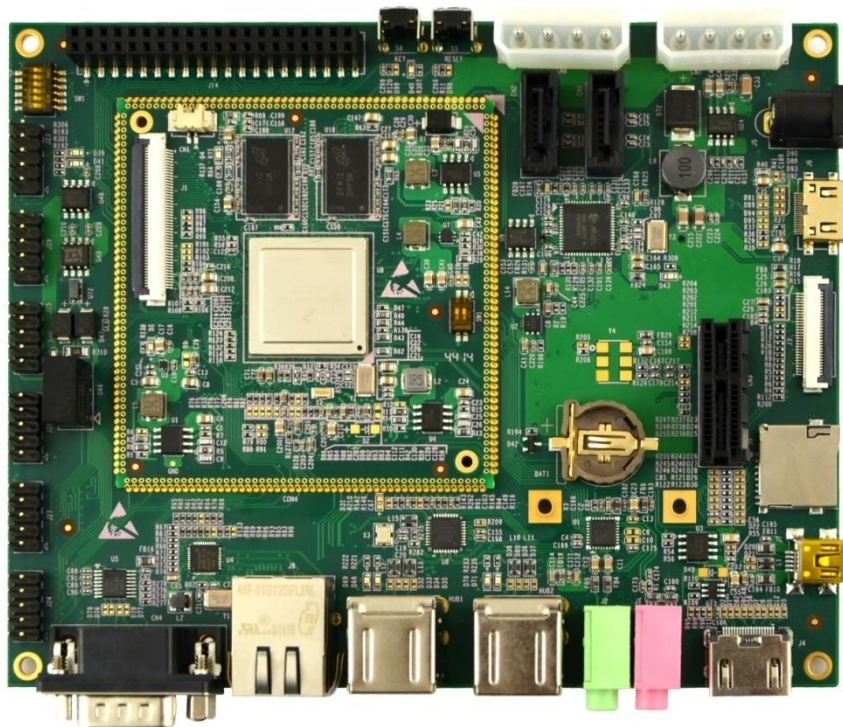


SBC9000 单板机



用户手册

版本 1.0 – 2015 年 1 月 20

版权声明：

- SBC9000 单板机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。

版本更新记录：

版本	更新日期	描述
1.0	2015-1-20	初始版本

目录

第 1 章	产品概述	1
1.1	产品简介	1
1.2	包装内容	1
1.3	Mini9000 核心板	1
1.3.1	产品特性	1
1.3.2	系统框图	2
1.4	扩展板	3
1.4.1	产品特性	3
1.4.2	系统框图	4
1.5	Mini9000 尺寸图	5
1.6	扩展板尺寸图	6
1.7	配套模块	6
第 2 章	硬件系统简介	7
2.1	CPU 简介	7
2.1.1	时钟	7
2.1.2	复位信号	7
2.1.3	通用接口	7
2.1.4	显示接口	7
2.1.5	3D 图形加速系统	8
2.2	CPU 周边器件	8
2.2.1	eMMC Flash 存储器 MTFC4GMDEA-4M IT	8
2.2.2	DDR 存储器 MT41K128M16JT-125 IT	8
2.2.3	AR8035 以太网 PHY	8
2.2.4	USB2514 USB 集线器	9
2.2.5	JMB321 SATA 扩展芯片	9
2.3	Mini9000 的接口/LED/开关	9
2.3.1	CON1 接口	9
2.3.2	启动配置开关 (SW1)	14
2.3.3	LED 指示灯	15
2.3.4	LCD 接口 (J1)	15

2.4	扩展板的接口/开关/按钮.....	17
2.4.1	电源接口（J5）	17
2.4.2	音频输入接口（J1）	17
2.4.3	音频输出接口（J2）	18
2.4.4	Camera 接口（J3）	18
2.4.5	EIM 接口（J14）	19
2.4.6	GPIO 接口（J29）	20
2.4.7	I2C 接口（J23）	20
2.4.8	CAN & SPI 接口（J25）	21
2.4.9	HDMI 接口（J4）	21
2.4.10	LVDS 接口（J6）	22
2.4.11	Mini PCIe 接口（CN5）	22
2.4.12	OTG 接口（J7）	24
2.4.13	PCIe 接口（CN3）	24
2.4.14	RGMII 接口（J8）	25
2.4.15	SATA 接口（CN1/CN7/CN2/CN8）	25
2.4.16	UART 接口（CN4/J28）	26
2.4.17	USB HUB 接口（HUB1/HUB2）	27
2.4.18	SDIO 接口（J9/J27/J24）	28
2.4.19	启动配置开关（SW1）	29
2.4.20	按钮.....	30
2.4.21	LED.....	30
第 3 章	准备工作.....	31
3.1	软件简介.....	31
3.2	关于 Linux 系统.....	31
3.3	关于 Android 系统	32
3.4	设置超级终端	33
第 4 章	下载和运行系统.....	35
4.1	Linux 和 Android 系统下载和运行	35
4.1.1	使用 Mfgtools 下载和运行 Linux 和 Android	35
4.1.2	使用 Linux Host 将 Linux 系统下载到 TF 卡并运行	38
4.2	显示模式设置	39
第 5 章	制作映像文件	42

5.1	制作 Linux 系统映像.....	42
5.1.1	直接编译	42
5.1.2	使用 Yocto 编译	44
5.2	制作 Android 系统映像.....	45
5.3	使用 Yocto 编译 Linux 上层应用程序	47
第 6 章	测试.....	49
6.1	LED 测试.....	49
6.2	按钮测试.....	49
6.3	触摸屏测试.....	50
6.4	RTC 测试	50
6.5	TF 卡测试.....	51
6.6	USB HOST 测试	52
6.7	USB Device 测试	53
6.8	音频测试.....	55
6.9	HDMI 音频测试	57
6.10	以太网测试.....	57
6.11	CAN 测试	58
6.12	串口测试.....	60
6.13	Mini-PCle 测试.....	61
6.14	PCI-E 测试	63
6.14.1	测试一	63
6.14.2	测试二.....	63
6.15	背光测试.....	65
6.15.1	LCD 背光测试	65
6.15.2	电容屏背光测试	65
6.16	SATA 测试	65
附录 1	安装 Ubuntu Linux 系统	67
附录 2	安装 Linux USB Ethernet/RNDIS Gadget 驱动.....	78
	技术支持和保修服务	80

第1章 产品概述

1.1 产品简介

SBC9000 是英蓓特推出的一款基于 i.MX 6Quad 的嵌入式单板机。它采用核心板 Mini9000 加扩展板的分离式结构进行设计。该单板机提供了 4 路串口（1 路支持 RS232 调试）、2 路带隔离 CAN2.0 接口、1 个千兆以太网口、1 路 spi, 3 路 I2C、1 路 SDIO、2 路 SATA、1 路 PCIe、1 路 mini PCIe、4 路 USB Host 和 1 路 USB OTG、LCD 和 LVDS 显示接口、HDMI 接口、音频输入和输出、TF 卡插槽等接口。SBC9000 支持 Linux3.10.17 和 Android4.4.2, 能够帮助开发者针对工业控制、上网本、桌面一体机、高端移动互联网设备、高端掌上电脑、高端便携式媒体播放器、游戏机和便携式导航设备等各种不同领域进行开发。

1.2 包装内容

- SBC9000 单板机（Mini9000 核心板+扩展板）*1
- 交叉串口线
- 12V 电源适配器

1.3 Mini9000 核心板

1.3.1 产品特性

- 一般特性
 - 产品尺寸：70 mm×68 mm
 - 工作温度：0~70℃
 - 工作湿度：20% ~ 90%（无凝结）
 - 输入电压：5V
- 处理器
 - ARM Cortex™-A9 内核的 i.MX 6Quad 处理器

- 集成 32 KByte 一级指令缓存
- 集成 32 KByte 一级数据缓存
- 集成私有计数器和看门狗
- 集成 Cortex-A9 NEON MPE（媒体处理引擎）协处理器
- 集成 2D 图形处理器
- 板载存储器：
 - 4GByte eMMC 存储器
 - 4*256MB DDR3 SDRAM 存储器
- 板载接口：
 - 四路 TTL232 信号（两路 4 位和两路 2 位）
 - 两路 CAN2.0 信号
 - 一路 RGMII 信号
 - 一路 SPI 信号
 - 三路 I2C 信号
 - 两路 8 位 SDIO 信号
 - 一路 SATA 信号
 - 一路 PCIe 信号
 - 一路 USB Host 信号
 - 一路 USB OTG 信号
 - 一路 LVDS 信号
 - 一路 HDMI 信号
 - 一路 LCD 接口
 - 一路 Audio 信号
 - 一路 Camera 信号
 - GPMC 信号
 - Boot 配置信号
 - GPIO 信号

1.3.2 系统框图

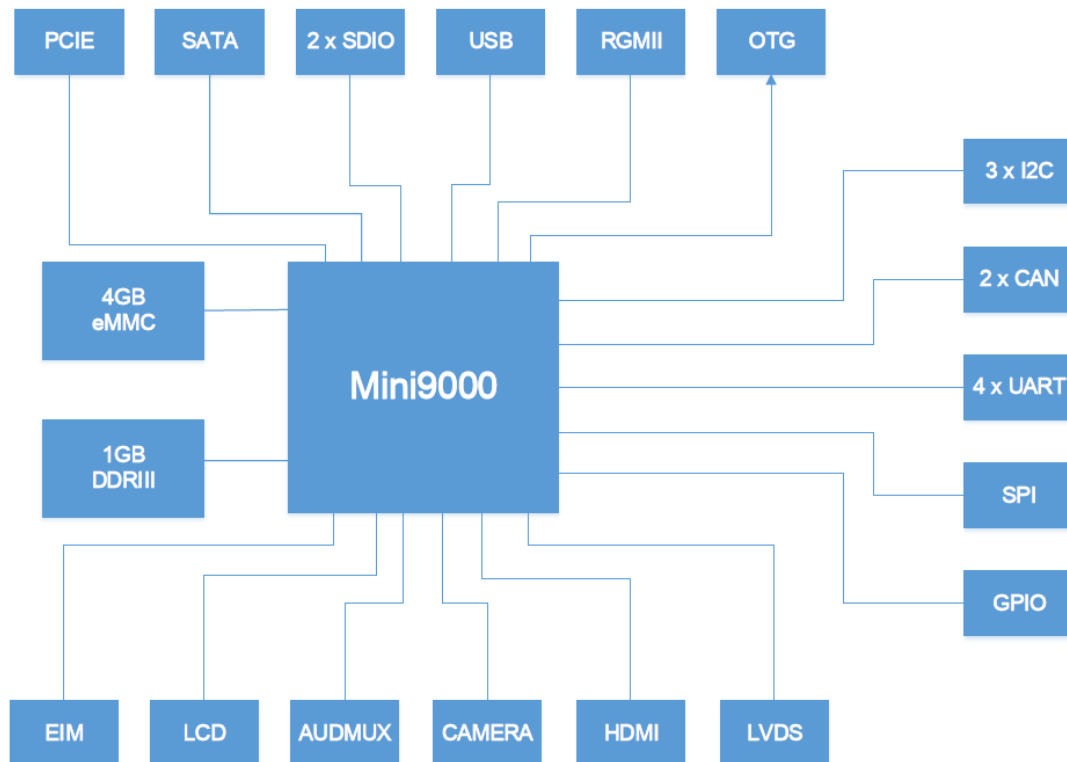


图 1-1 Mini9000 系统框图

1.4 扩展板

1.4.1 产品特性

- 一般特性
 - 产品尺寸：150 mm×120 mm
 - 工作温度：0~70℃
 - 工作湿度：20% ~ 90%（无凝结）
 - 输入电压：12V
- 音频/视频接口
 - 一路 HDMI 接口
 - 一路 LVDS 接口
 - 一路音频输入接口(3.5mm 音频接口)
 - 一路双声道音频输出接口(3.5mm 音频接口)
- 数据传输接口

- 一路 10/100/1000Mbps 以太网接口
- 两路带隔离 CAN 2.0 接口
- 一路 SPI 接口
- 三路 I2C 接口
- 一路 SDIO 接口
- 两路 SATA 接口
- 一路 TF 卡接口
- 一路 PCIe 接口
- 一路 mini PCIe 接口
- 四路 USB host 接口
- 一路 OTG 接口
- 四路串口（一路支持 DB9 调试）
- **LED 和按钮**
 - 一个自定义按键
 - 一个复位按键
 - 一个电源指示灯
 - 两个用户自定义灯

1.4.2 系统框图

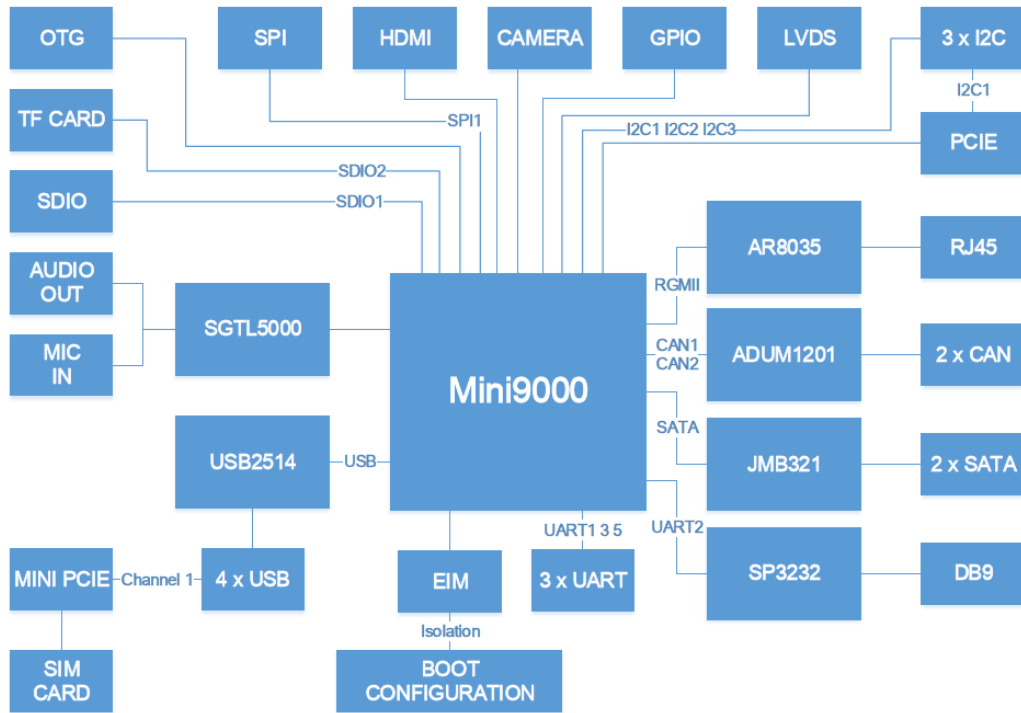


图 1-2 扩展板框图

1.5 Mini9000 尺寸图

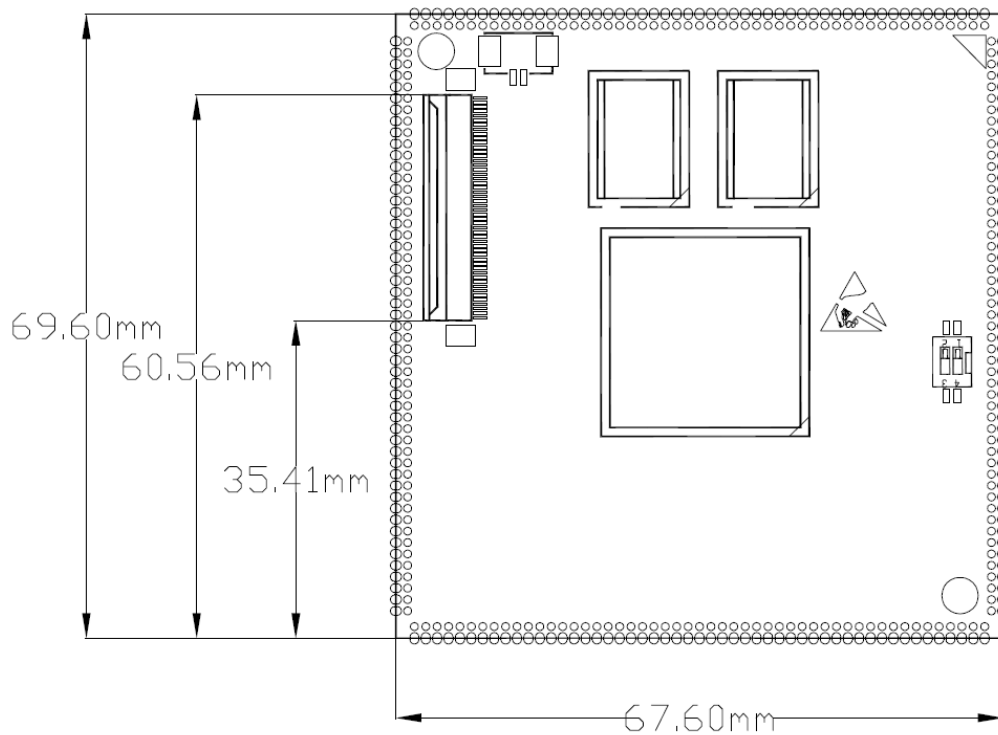


图 1-3 Mini9000 尺寸

1.6 扩展板尺寸图

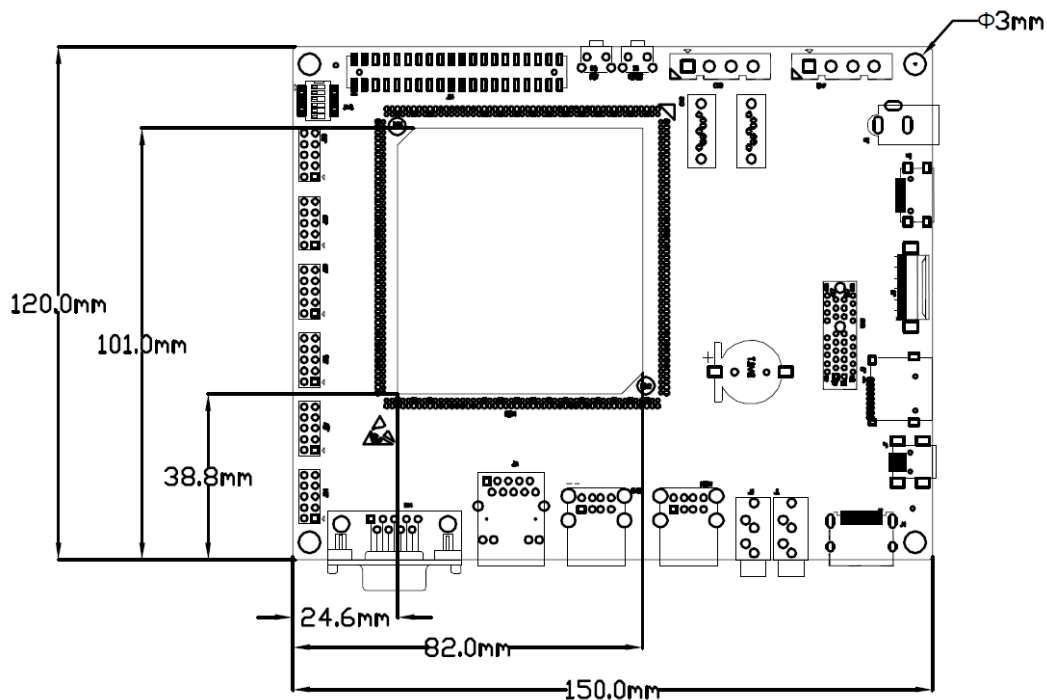


图 1-4 扩展板尺寸

1.7 配套模块

SBC9000 配套模块支持情况如下表所示

表 1-1 配套模块

模块名称	模块描述	接口类型	Linux	Android
WI-PI	WIFI 模块	USB	Yes	Yes
CAM8100-U	200 万像素的 USB 数字摄像头	USB	Yes	No
CAM8200-U	500 万像素的 USB 数字摄像头	USB	Yes	Yes
CAM8000-D	200 万像素的数字摄像头	30Pin FPC	Yes	Yes
CAM8100-D	500 万像素的数字摄像头	30Pin FPC	Yes	Yes
VGA8000	LCD 转 VGA 模块	50Pin FPC	Yes	Yes
LCD8000-97C	9.7 寸 LVDS 屏, 电容屏, 支持多点触摸	Mini HDMI	Yes	Yes
LCD8000-43T	4.3 寸 LCD, 带电阻屏	50Pin FPC	Yes	Yes
LCD8000-70T	7 寸 LCD, 带电阻屏	50Pin FPC	Yes	Yes

第2章 硬件系统简介

本章节将通过简要介绍 SBC9000 单板机上所采用的 CPU、周边芯片、各种接口的管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

Freyscale 公司的 i.MX 6Quad 是基于 ARM™ Cortex-A9 内核的四核处理器，运行频率高达 1.0 GHz，集成了 2D 和 3D 图形处理器、3D 1080p 视频处理器和电源管理模块，并且提供 64 位 DDR3/LVDDR3/LVDDR2-1066 储存器接口以及包括高清显示和摄像头在内的许多其他接口。

2.1.1 时钟

i.MX 6Quad 的时钟信号包括一个 32.768KHz 的 RTC 时钟和一个 24MHz 的外频时钟；

- **RTC 时钟：**由外部无源晶振产生，用于低频率运算；
- **外频时钟：**用于产生设备的主时钟，以便提供给 PLL 和 CMM 等其他模组；

2.1.2 复位信号

复位信号由 CPU 的 POR_B 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 7 组通用输入输出接口（GPIO），每一个 GPIO 模组提供 32 个（其中 GPIO7 提供 14 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 206 个管脚。

2.1.4 显示接口

- 一路并行的 24bit RGB 接口，支持 60 Hz WUXGA 输出
- 两路 LVDS 接口，最大支持 165 Mpixels/sec 输出
- 一个 HDMI 1.4 接口

- 一个 MIPI/DSI 接口，输出速率 1 Gbps

2.1.5 3D 图形加速系统

i.MX 6Quad 集成了 GPU3Dv4 3D 图形处理单元，为 3D 图像算法提供了硬件加速，可以使桌面交互式图像应用最高达到 HD1080P 的分辨率，并且支持 OpenGL ES 2.0 标准及其扩展、OpenGL ES 1.1 和 OpenVG 1.1 标准。

另外 i.MX 6Quad 还通过其 GPUVGv2 向量图形处理单元为 2D 图像算法提供了硬件加速功能。

2.2 CPU 周边器件

2.2.1 eMMC Flash 存储器 MTFC4GMDEA-4M IT

MTFC4GMDEA-4M IT 是 Mini9000 上的 eMMC Flash 存储器，大小为 4GB。该 flash 存储器支持时钟速率高达 52MHz 的高速 DDR 传输和三种数据线宽模式：1-bit(默认)、4-bit 和 8-bit；其同步电源管理技术使得芯片具有高速启动、自动终止和睡眠等特点；同时 MTFC4GMDEA-4M IT 还支持高速双数据传输引导模式。

2.2.2 DDR 存储器 MT41K128M16JT-125 IT

MT41K128M16JT-125 IT 是 Mini9000 上的 DDR3 SDRAM 存储器，大小为 256MB。MT41K128M16JT-125 IT 适用于大容量和高带宽需求的场合，支持差分时钟输入、差分数据选通、自动刷新以及异步管脚复位等功能。Mini9000 集成了 4 片 MT41K128M16JT-125 IT，共计 1GB 容量。

2.2.3 AR8035 以太网 PHY

AR8035 是 SBC9000 上的低功耗、低成本、单端口 10/100/1000 Mbps 三速以太网 PHY。AR8035 支持 MAC.TM RGMII 接口，采用 Atheros 专有的 SmartEEE 技术来支持 IEEE 802.3az 高效节能以太网（EEE）标准，让不支持 802.3az 标准的传统 MAC/SoC 设备具有更高能效。SBC9000 可通过直通网线连接到网络集线器上，也可用交叉网线与电脑直接相连。

2.2.4 USB2514 USB 集线器

USB2514 是一个四端口输出的 USB HUB 2.0 控制器，该芯片为 SBC9000 引出了四路 USB HUB 信号并连接到 USB 插座，其中一路 USB 信号同时连接到 mini PCIe 接口来实现复用。

2.2.5 JMB321 SATA 扩展芯片

JMB321 是一个 SATA 扩展芯片，该芯片为 SBC9000 引出两路 SATA 接口。

2.3 Mini9000 的接口/LED/开关

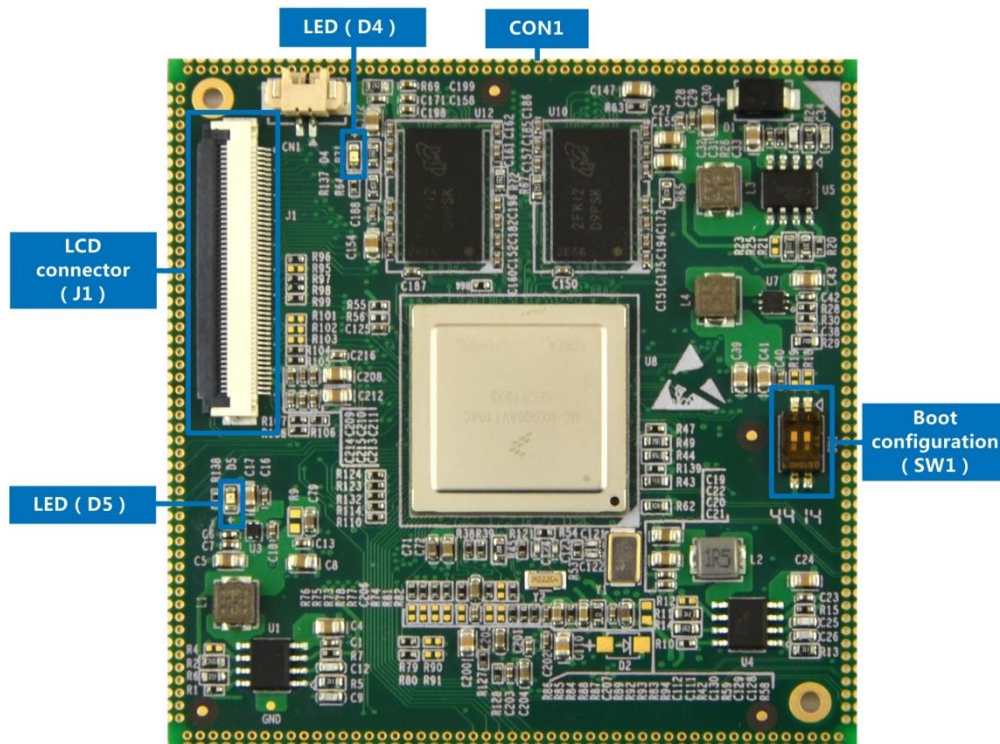


图 2-1 Mini9000 的接口/LED/开关

2.3.1 CON1 接口

表 2-1 CON1 接口引脚定义

引脚	信号定义	描述
1	GPIO4_IO07	GPIO signal
2	GND	GND
3	OTG_VBUS	OTG bus, +5V

引脚	信号定义	描述
4	5VIN	Mini9000 power supply, +5V
5	5VIN	Mini9000 power supply, +5V
6	5VIN	Mini9000 power supply, +5V
7	GND	GND
8	VDD_RTC	Real-time clock power
9	2P5V	System power, +2.5V
10	2P5V	System power, +2.5V
11	GND	GND
12	3P3V	System power, +3.3V
13	3P3V	System power, +3.3V
14	3P3V	System power, +3.3V
15	GND	GND
16	ON_OFF	System power on/off control signal
17	RESET_N_B	System reset control signal
18	GND	GND
19	CAN2_RXD	CAN2 receive data
20	CAN2_TXD	CAN2 transmit data
21	I2C2_SDA	I2C2 master serial data
22	I2C2_SCL	I2C2 master serial clock
23	GPIO4_IO11	GPIO signal
24	GPIO4_IO06	GPIO signal
25	EIM_WAIT	EIM ready/busy/wait signal
26	EIM_EB1	EIM byte enable
27	EIM_LBA	EIM address valid
28	EIM_CS0	EIM chip select
29	EIM_OE	EIM output enable
30	EIM_CS1	EIM chip select
31	EIM_EB0	EIM byte enable
32	EIM_RW	EIM memory write enable
33	EIM_D29	EIM MSB data bus signal
34	EIM_D28	EIM MSB data bus signal
35	EIM_CRE	Used as CRE/PS for Cellular Rammemory.
36	EIM_EB3	EIM byte enable
37	EIM_EB2	EIM byte enable
38	EIM_BCLK	EIM burst clock
39	EIM_A26	EIM MSB address bus signal
40	EIM_A23	EIM MSB address bus signal
41	EIM_A22	EIM MSB address bus signal
42	EIM_A18	EIM MSB address bus signal
43	EIM_A24	EIM MSB address bus signal
44	EIM_A21	EIM MSB address bus signal
45	EIM_A25	EIM MSB address bus signal

引脚	信号定义	描述
46	EIM_A16	EIM MSB address bus signal
47	EIM_A20	EIM MSB address bus signal
48	EIM_A19	EIM MSB address bus signal
49	EIM_A17	EIM MSB address bus signal
50	GND	GND
51	GPIO1_IO27	GPIO signal
52	EIM_DA11	EIM LSB multiplexed address/data bus signal
53	EIM_DA13	EIM LSB multiplexed address/data bus signal
54	EIM_DA14	EIM LSB multiplexed address/data bus signal
55	EIM_DA9	EIM LSB multiplexed address/data bus signal
56	EIM_DA12	EIM LSB multiplexed address/data bus signal
57	EIM_DA10	EIM LSB multiplexed address/data bus signal
58	EIM_DA0	EIM LSB multiplexed address/data bus signal
59	EIM_DA8	EIM LSB multiplexed address/data bus signal
60	EIM_DA15	EIM LSB multiplexed address/data bus signal
61	EIM_DA7	EIM LSB multiplexed address/data bus signal
62	EIM_DA4	EIM LSB multiplexed address/data bus signal
63	EIM_DA3	EIM LSB multiplexed address/data bus signal
64	EIM_DA5	EIM LSB multiplexed address/data bus signal
65	EIM_DA2	EIM LSB multiplexed address/data bus signal
66	EIM_DA6	EIM LSB multiplexed address/data bus signal
67	EIM_DA1	EIM LSB multiplexed address/data bus signal
68	CSPI1_SS1	SPI1 chip select
69	CSPI1_CLK	SPI1 clock
70	CSPI1_MOSI	SPI1 master output slave input
71	CSPI1_MISO	SPI1 master input slave output
72	UART3_RXD	UART3 receive data
73	UART3_TXD	UART3 transmit data
74	UART2_RXD	UART2 receive data
75	UART2_TXD	UART2 transmit data
76	GND	GND
77	UART3_CTS	UART3 clear to send
78	UART3_RTS	UART3 request to send
79	UART2_RTS	UART2 request to send
80	UART2_CTS	UART2 clear to send
81	USB_H1_OC	Host 1 external input for VBUS
82	USB_OTG_OC	overcurrent detection
83	USB_HOST_DP	OTG External input for VBUS
84	USB_HOST_DN	overcurrent detection
85	USB_RSTn	USB host data+
86	SD1_WP	USB host data-
87	SD1_DATA3	USB host reset control signal

引脚	信号定义	描述
88	SD1_CLK	SD1 card write protect detect signal
89	SD1_DATA2	SD1 data 3
90	SD1_DATA1	SD1 clock
91	SD1_DATA6	SD1 data 2
92	SD1_DATA0	SD1 data 1
93	SD1_DATA7	SD1 data 6
94	GND	SD1 data 0
95	SD1_DATA5	SD1 data 7
96	SD1_DATA4	GND
97	SD1_CMD	SD1 data 5
98	SD1_CD	SD1 data 4
99	SD2_WP	SD1 command signal
100	SD2_DATA2	SD1 card detect signal
101	SD2_DATA3	SD2 card write protect detect signal
102	SD2_DATA0	SD2 data 2
103	SD2_DATA5	SD2 data 3
104	GND	SD2 data 0
105	SD2_CD	SD2 data 5
106	SD2_DATA4	GND
107	SD2_CLK	SD2 card detect signal
108	SD2_DATA1	SD2 data 4
109	SD2_CMD	SD2 clock
110	SD2_DATA6	SD2 data 1
111	SD2_DATA7	SD2 command signal
112	GND	SD2 data 6
113	RGMII_REF_CLK	SD2 data 7
114	RGMII_MDIO	GND
115	RGMII_INT	RGMII reference clock
116	RGMII_MDC	RGMII information transferring control
117	RGMII_RXD3	RGMII interrupting signal
118	RGMII_TXD2	RGMII output clock
119	RGMII_RXDV	RGMII receive data
120	GND	RGMII transmit data
121	RGMII_TXD0	RGMII receive data valid
122	RGMII_TXCLK	GND
123	RGMII_nRST	RGMII transmit data
124	RGMII_TXD1	RGMII transmit clock
125	RGMII_RXD0	RGMII reset signal
126	RGMII_RXCLK	RGMII transmit data
127	RGMII_RXD2	RGMII receive data
128	RGMII_TXEN	RGMII receive clock
129	RGMII_TXD3	RGMII receive data

引脚	信号定义	描述
130	RGMII_RXD1	RGMII transmit enable
131	GPIO7_IO06	RGMII transmit data
132	GPIO7_IO04	RGMII receive data
133	GPIO7_IO05	GPIO signal
134	GPIO7_IO01	GPIO signal
135	UART1_RXD	GPIO signal
136	UART1_TXD	GPIO signal
137	USB_OTG_DN	UART1 receive data
138	USB_OTG_DP	UART1 transmit data
139	USB_OTG_ID	OTG data-
140	USB_OTG_PWR_EN	OTG data+
141	GND	OTG ID signal
142	CSI0_DAT16	OTG power enable control signal
143	CSI0_DAT17	GND
144	CSI0_DAT15	CSI0 capture data bit 16
145	CSI0_DAT13	CSI0 capture data bit 17
146	CSI0_DAT14	CSI0 capture data bit 15
147	CSI0_DAT12	CSI0 capture data bit 13
148	CSI0_DAT19	CSI0 capture data bit 14
149	CSI0_DAT18	CSI0 capture data bit 12
150	GND	CSI0 capture data bit 19
151	CSI0_HSYNC	CSI0 capture data bit 18
152	CSI0_PIXCLK	GND
153	CSI0_VSYNC	CSI0 horizontal synchronization
154	CAM_MCLK	CSI0 pixel clock
155	CAM_RST	CSI0 vertical synchronization
156	CAM_EN	Camera clock
157	SATA_RXN	Camera reset control signal
158	SATA_RXP	Camera data enable control signal
159	SATA_TXP	SATA receive data-
160	SATA_TXN	SATA receive data+
161	PCIE_WAKEn	SATA transmit data+
162	PCIE_REFCLK_DP	SATA transmit data-
163	PCIE_REFCLK_DN	PCIe wake enable control signal
164	GND	PCIe reference clock+
165	PCIE_TXP	PCIe reference clock-
166	PCIE_TXM	GND
167	PCIE_RXP	PCIe transmit data+
168	PCIE_RXM	PCIe transmit data-
169	PRSNT2_N_X1	PCIe receive data+
170	I2C1_SDA	PCIe receive data-
171	I2C1_SCL	Add-in card presence detect signal

引脚	信号定义	描述
172	AUD3_RXD	I2C1 master serial clock
173	AUD3_TXFS	I2C1 master serial data
174	AUD3_TXD	Audio data receive signal
175	AUD3_TXC	Audio receive frame sync signal
176	GND	Audio data transmit signal
177	HDMI_HPD	Audio transmit clock signal
178	HDMI_CLKM	GND
179	HDMI_CLKP	HDMI hot plug and play detect
180	GND	HDMI data clock-
181	HDMI_D0M	HDMI data clock+
182	HDMI_D0P	GND
183	HDMI_D1M	HDMI data 0-
184	HDMI_D1P	HDMI data 0+
185	HDMI_D2M	HDMI data 1-
186	HDMI_D2P	HDMI data 1+
187	I2C3_SDA	HDMI data 2-
188	I2C3_SCL	HDMI data 2+
189	UART5_TXD	I2C3 master serial data
190	UART5_RXD	I2C3 master serial clock
191	CAN1_RXD	UART5 transmit data
192	CAN1_TXD	UART5 receive data
193	PWM4	CAN1 receive data
194	LED_PWR_EN	CAN1 transmit data
195	Touch_Int	Pulse width modulation
196	LCD_PWR_EN	LED backlight enable
197	LVDS0_TX1_N	Touch interrupt signal
198	LVDS0_TX1_P	Touch reset signal
199	LVDS0_CLK_N	LVDS0 data1-
200	LVDS0_CLK_P	LVDS0 data1+
201	LVDS0_TX0_N	LVDS0 clock-
202	LVDS0_TX0_P	LVDS0 clock+
203	LVDS0_TX2_N	LVDS0 data0-
204	LVDS0_TX2_P	LVDS0 data0+

2.3.2 启动配置开关（SW1）

SBC9000 上带有两组拨码开关用于选择启动方式，一组位于 Mini9000 上，另一组位于扩展板上。

表 2-2 Mini9000 上的启动配置开关

位号	1	2	描述
状态	OFF	ON	Serial Downloader
	ON	OFF	Internal boot
	OFF	OFF	Boot from Fuses
	ON	ON	Reserved

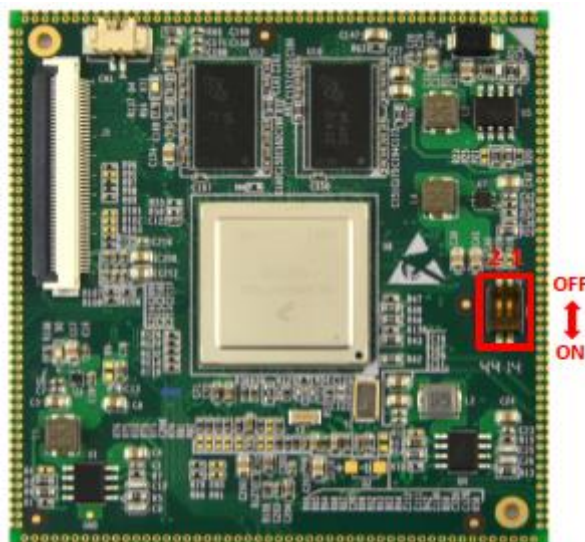


图 2-2 核心板启动配置开关状态说明

2.3.3 LED 指示灯

表 2-3 LED 指示灯

引脚	描述
D5	电源指示灯
D4	用户定义 LED

2.3.4 LCD 接口（J1）

表 2-4 LCD 接口

引脚	信号定义	描述
1	B0	LCD Pixel data bit 0
2	B1	LCD Pixel data bit 1
3	B2	LCD Pixel data bit 2
4	B3	LCD Pixel data bit 3
5	B4	LCD Pixel data bit 4
6	B5	LCD Pixel data bit 5
7	B6	LCD Pixel data bit 6
8	B7	LCD Pixel data bit 7

引脚	信号定义	描述
9	GND1	GND
10	G0	LCD Pixel data bit 8
11	G1	LCD Pixel data bit 9
12	G2	LCD Pixel data bit 10
13	G3	LCD Pixel data bit 11
14	G4	LCD Pixel data bit 12
15	G5	LCD Pixel data bit 13
16	G6	LCD Pixel data bit 14
17	G7	LCD Pixel data bit 15
18	GND2	GND
19	R0	LCD Pixel data bit 16
20	R1	LCD Pixel data bit 17
21	R2	LCD Pixel data bit 18
22	R3	LCD Pixel data bit 19
23	R4	LCD Pixel data bit 20
24	R5	LCD Pixel data bit 21
25	R6	LCD Pixel data bit 22
26	R7	LCD Pixel data bit 23
27	GND3	GND
28	DEN	AC bias control (STN) or pixel data enable (TFT)
29	HSYNC	LCD Horizontal Synchronization
30	VSYNC	LCD Vertical Synchronization
31	GND	GND
32	CLK	LCD Pixel Clock
33	GND4	GND
34	X+	X+ Position Input
35	X-	X- Position Input
36	Y+	Y+ Position Input
37	Y-	Y- Position Input
38	SPI_CLK	SPI serial clock
39	SPI_MOSI	SPI Master Output, Slave Input
40	SPI_MISO	SPI Master Input, Slave Output
41	SPI_CS	SPI Chip Select
42	IIC_CLK	IIC master serial clock
43	IIC_DAT	IIC serial bidirectional data
44	GND5	GND
45	VDD1	3.3V
46	VDD2	3.3V
47	VDD3	5V
48	VDD4	5V
49	RESET	Reset

引脚	信号定义	描述
50	PWREN	Backlight enable

2.4 扩展板的接口/开关/按钮

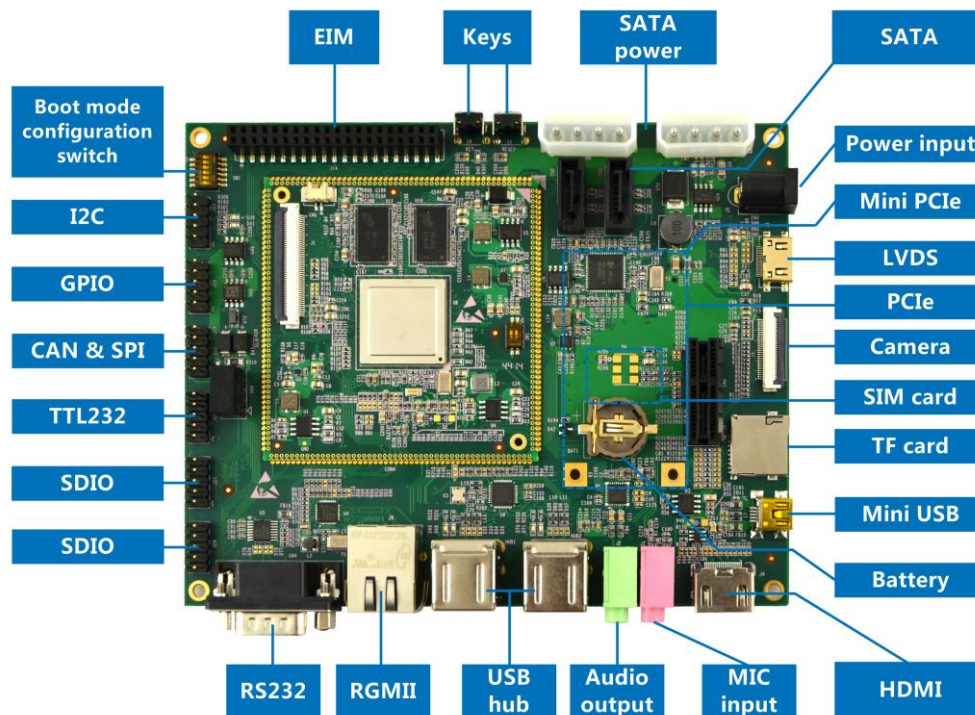


图 2-3 扩展板的接口/开关/按钮

2.4.1 电源接口（J5）

表 2-5 电源接口

引脚	信号定义	描述
1	GND	GND
2	+12V	Power supply (+12V)
3	NC	NC

2.4.2 音频输入接口（J1）

表 2-6 音频输入接口

引脚	信号定义	描述
1	GND	GND

引脚	信号定义	描述
2	MIC In	input
3	MIC In	input
4	MIC In	input
5	MIC In	input

2.4.3 音频输出接口（J2）

表 2-7 音频输出接口

引脚	信号定义	描述
1	GND	GND
2	Left	Left output
3	Right	Right output
4	Right	Right output
5	Left	Left output

2.4.4 Camera 接口（J3）

表 2-8 Camera 接口

引脚	信号定义	描述
1	GND	GND
2	NC	NC
3	NC	NC
4	CSI0_DAT12	CSI0 capture data bit 12
5	CSI0_DAT13	CSI0 capture data bit 13
6	CSI0_DAT14	CSI0 capture data bit 14
7	CSI0_DAT15	CSI0 capture data bit 15
8	CSI0_DAT16	CSI0 capture data bit 16
9	CSI0_DAT17	CSI0 capture data bit 17
10	CSI0_DAT18	CSI0 capture data bit 18
11	CSI0_DAT19	CSI0 capture data bit 19
12	NC	NC
13	NC	NC
14	GND	GND
15	CSI0_PIXCLK	CSI0 pixel clock
16	GND	GND
17	CSI0_HSYNC	CSI0 HSYNC
18	NC	NC
19	CSI0_VSYNC	CSI0 VSYNC
20	VDD_NVCC	3.3V

引脚	信号定义	描述
21	CAM_MCLK	Camera clock
22	NC	NC
23	GND	GND
24	NC	NC
25	CAM_RST	CSI0 reset
26	CAM_EN	CSI0 data enable
27	I2C1_SDA	I2C2 serial data
28	I2C1_SCL	I2C2 serial clock
29	GND	GND
30	3P3V	3.3V

2.4.5 EIM 接口 (J14)

表 2-9 EIM 接口

引脚	信号定义	描述
1	5VIN	5V
2	3P3V	3.3V
3	GND	GND
4	GND	GND
5	EIM_DA0	EIM LSB multiplexed address/data bus signal
6	EIM_DA1	EIM LSB multiplexed address/data bus signal
7	EIM_DA2	EIM LSB multiplexed address/data bus signal
8	EIM_DA3	EIM LSB multiplexed address/data bus signal
9	EIM_DA4	EIM LSB multiplexed address/data bus signal
10	EIM_DA5	EIM LSB multiplexed address/data bus signal
11	EIM_DA6	EIM LSB multiplexed address/data bus signal
12	EIM_DA7	EIM LSB multiplexed address/data bus signal
13	EIM_DA8	EIM LSB multiplexed address/data bus signal
14	EIM_DA9	EIM LSB multiplexed address/data bus signal
15	EIM_DA10	EIM LSB multiplexed address/data bus signal
16	EIM_DA11	EIM LSB multiplexed address/data bus signal
17	EIM_DA12	EIM LSB multiplexed address/data bus signal
18	EIM_DA13	EIM LSB multiplexed address/data bus signal
19	EIM_DA14	EIM LSB multiplexed address/data bus signal
20	EIM_DA15	EIM LSB multiplexed address/data bus signal
21	EIM_A16	EIM MSB address bus signal
22	EIM_A17	EIM MSB address bus signal
23	EIM_A18	EIM MSB address bus signal
24	EIM_A19	EIM MSB address bus signal
25	EIM_A20	EIM MSB address bus signal

引脚	信号定义	描述
26	EIM_A21	EIM MSB address bus signal
27	EIM_A22	EIM MSB address bus signal
28	EIM_A23	EIM MSB address bus signal
29	EIM_A24	EIM MSB address bus signal
30	EIM_A25	EIM MSB address bus signal
31	EIM_WAIT	EIM ready/busy/wait signal
32	EIM_LBA	EIM address valid
33	EIM_EB0	EIM byte enable
34	EIM_EB1	EIM byte enable
35	EIM_RW	EIM memory write enable
36	EIM_CRE	Used as CRE/PS for Cellular Rammemory
37	EIM_BCLK	EIM burst clock
38	EIM_CS0	EIM chip select
39	EIM_CS1	EIM chip select
40	EIM_OE	EIM output enable

2.4.6 GPIO 接口（J29）

表 2-10 GPIO 接口

引脚	信号定义	描述
1	5VIN	5V
2	3P3V	3.3V
3	GND	GND
4	GND	GND
5	EIM_A26	EIM MSB address bus signal
6	EIM_EB2	EIM byte enable
7	EIM_D28	EIM MSB data bus signal
8	EIM_EB3	EIM byte enable
9	EIM_D29	EIM MSB data bus signal
10	ON_OFF	System power on/off control signal

2.4.7 I2C 接口（J23）

表 2-11 I2C 接口

引脚	信号定义	描述
1	3P3V	+5V
2	GND	GND
3	I2C3_SCL	I2C3 master serial clock
4	GPIO7_IO05	GPIO signal

引脚	信号定义	描述
5	I2C3_SDA	I2C3 master serial data
6	GPIO7_IO	GPIO signal
7	I2C1_SCL	I2C1 master serial clock
8	I2C2_SCL	I2C2 master serial clock
9	I2C1_SDA	I2C1 master serial data
10	I2C2_SDA	I2C2 master serial data

2.4.8 CAN & SPI 接口（J25）

表 2-12 CAN & SPI 接口

引脚	信号定义	描述
1	3P3V	3.3V
2	GND	GND
3	CAN2_L	CAN2_L
4	CSPI1_SS1	SPI1 chip select
5	CAN2_H	CAN2_H
6	CSPI1_CLK	SPI1 clock
7	CAN1_L	CAN1_L
8	CSPI1_MISO	SPI1 master input slave output
9	CAN1_H	CAN1_H
10	CSPI1_MOSI	SPI1 master output slave input

2.4.9 HDMI 接口（J4）

表 2-13 HDMI 接口

引脚	信号定义	描述
1	HDMI_D2P	TMDS data 2+
2	GND	GND
3	HDMI_D2M	TMDS data 2 shield
4	HDMI_D1P	TMDS data 1+
5	GND	GND
6	HDMI_D1M	TMDS data 1-
7	HDMI_D0P	TMDS data 0+
8	GND	TMDS data 0 shield
9	HDMI_D0M	TMDS data 0-
10	HDMI_CLKP	TMDS data clock+
11	GND	GND
12	HDMI_CLKM	TMDS data clock-
13	NC	NC

引脚	信号定义	描述
14	NC	NC
15	BI2C2_SCL	IIC master serial clock
16	BI2C2_SDA	IIC serial bidirectional data
17	GND	GND
18	5V	5V
19	HDMI_HPD_R	Hot plug and play detect

2.4.10 LVDS 接口（J6）

表 2-14 LVDS 接口

引脚	信号定义	描述
1	LVDS_3P3V	+3.3V
2	LVDS0_TX2_P	LVDS0 Data2+
3	LVDS0_TX2_NN	LVDS0 Data2-
4	GND	GND
5	LVDS0_TX1_P	LVDS0 Data1+
6	LVDS0_TX1_N	LVDS0 Data1-
7	GND	GND
8	LVDS0_TX0_P	LVDS0 Data0+
9	LVDS0_TX0_N	LVDS0 Data-
10	GND	GND
11	LVDS0_CLK_PP	LVDS0_CLK+
12	LVDS0_CLK_N	LVDS0_CLK-
13	LCD_PWR_ENN	Touch Reset Signal
14	Touch_Int	Touch Interrupt Signal
15	I2C3_SCL	I2C3 Master Serial Clock
16	I2C3_SDA	I2C3 Master Serial Data
17	LED_PWR_EN	Backlight Enable
18	5VIN	+5V
19	PWM4	Pulse Width Modulation

2.4.11 Mini PCIe 接口（CN5）

表 2-15 Mini PCIe 接口

引脚	信号定义	描述
1	GPIO1_IO27	GPIO Signal
2	MPCIE_3P3V	3.3V
3	NC	Not connected
4	GND	GND

引脚	信号定义	描述
5	NC	Not connected
6	NC	Not connected
7	NC	Not connected
8	UIM_PWR	UIM power supply
9	GND	GND
10	UIM_DATA	UIM data
11	NC	Not connected
12	UIM_CLK	UIM clock
13	NC	Not connected
14	UIM_RESET	UIM reset control signal
15	GND	GND
16	NC	Not connected
17	NC	Not connected
18	GND	GND
19	NC	Not connected
20	W_DISABLE	Close wireless communications
21	GND	GND
22	PERST	Reset signal
23	NC	Not connected
24	MPCIE_3P3V	3.3V
25	NC	Not connected
26	GND	GND
27	GND	GND
28	NC	Not connected
29	GND	GND
30	NC	Not connected
31	NC	Not connected
32	NC	Not connected
33	NC	Not connected
34	GND	GND
35	GND	GND
36	DM1	USB data-
37	GND	GND
38	DP1	USB data+
39	MPCIE_3P3V	3.3V
40	GND	GND
41	MPCIE_3P3V	3.3V
42	LED_WWAN	Status indicated signal
43	GND	GND
44	NC	Not connected
45	NC	Not connected
46	NC	Not connected

引脚	信号定义	描述
47	NC	Not connected
48	NC	Not connected
49	NC	Not connected
50	GND	GND
51	NC	Not connected
52	MPCIE_3P3V	3.3V

2.4.12 OTG 接口 (J7)

表 2-16 OTG 接口

引脚	信号定义	描述
1	OTG_VBUS	OTG bus, +5V
2	USB_OTG_DN	OTG data-
3	USB_OTG_DP	OTG data+
4	USB_OTG_ID	OTG ID signal
5	GND	GND

2.4.13 PCIe 接口 (CN3)

表 2-17 PCIe 接口

引脚	信号定义	描述
A1	GND	GND
A2	12VIN	12V
A3	12VIN	12V
A4	GND	GND
A5	NC	Not connected
A6	NC	Not connected
A7	NC	Not connected
A8	NC	Not connected
A9	3P3V	3.3V
A10	3P3V	3.3V
A11	RESET_N_B	System reset control signal
A12	GND	GND
A13	REFCLK+	PCIe reference clock+
A14	REFCLK+	PCIe reference clock-
A15	GND	GND
A16	PCIE_RXP	PCIe receive data+
A17	PCIE_RXN	PCIe receive data-
A18	GND	GND

引脚	信号定义	描述
B1	12VIN	12V
B2	12VIN	12V
B3	12VIN	12V
B4	GND	GND
B5	I2C1_SCL	I2C1 master serial clock
B6	I2C1_SDA	I2C1 master serial data
B7	GND	GND
B8	3P3V	3.3V
B9	NC	Not connected
B10	3P3V	3.3V
B11	PCIE_WAKEn	PCIe wake enable control signal
B12	NC	Not connected
B13	GND	GND
B14	PCIE_TXP	PCIe transmit data+
B15	PCIE_TXN	PCIe transmit data-
B16	GND	GND
B17	PRSNT2_N_X1	Add-in card presence detect signal
B18	GND	GND

2.4.14 RGMII 接口（J8）

表 2-18 RGMII 接口

引脚	信号定义	描述
1	TD1+	TD1+ output
2	TD1-	TD1- output
3	TD2+	TD2+ output
4	TD2-	TD2- output
5	TCT	2.5V Power for TD
6	RCT	2.5V Power for RD
7	RD1+	RD1+ input
8	RD1-	RD1- input
9	RD2+	RD2+ input
10	RD2-	RD2- input
11	GRLA	Green LED link signal
12	GRLC	Power supply for Green LED
13	YELC	Yellow LED action signal
14	YELA	Power supply for Yellow LED

2.4.15 SATA 接口（CN1/CN7/CN2/CN8）

表 2-19 SATA 信号接口 1 (CN1)

引脚	信号定义	描述
1	GND	GND
2	SATA_TXP0	SATA transmit data 0+
3	SATA_TXN0	SATA transmit data 0-
4	GND	GND
5	SATA_RXN0	SATA receive data 0-
6	SATA_RXP0	SATA receive data 0+
7	GND	GND

表 2-20 SATA 供电接口 1 (CN7)

引脚	信号定义	描述
1	5VIN	5V
2	GND	GND
3	GND	GND
4	12VIN	12V

表 2-21 SATA 信号接口 2 (CN2)

引脚	信号定义	描述
1	GND	GND
2	SATA_TXP1	SATA transmit data 1+
3	SATA_TXN1	SATA transmit data 1-
4	GND	GND
5	SATA_RXN1	SATA receive data 1-
6	SATA_RXP1	SATA receive data 1+
7	GND	GND

表 2-22 SATA 供电接口 2 (CN8)

引脚	信号定义	描述
1	5VIN	5V
2	GND	GND
3	GND	GND
4	12VIN	12V

2.4.16 UART 接口 (CN4/J28)

表 2-23 DB9 接口 (CN4)

引脚	信号定义	描述
----	------	----

引脚	信号定义	描述
1	NC	Not connected
2	COM2_RXD	DB9 receive data
3	COM2_TXD	DB9 transmit data
4	NC	Not connected
5	GND	GND
6	NC	Not connected
7	COM2_RTS	DB9 request to send
8	COM2_CTS	DB9 clear to send
9	NC	Not connected

表 2-24 TTL232 接口 (J28)

引脚	信号定义	描述
1	3P3V	3.3V
2	GND	GND
3	UART1_RXD	UART1 receive data
4	UART3_RTS	UART3 request to send
5	UART1_TXD	UART1 transmit data
6	UART3_CTS	UART3 clear to send
7	UART5_TXD	UART5 transmit data
8	UART3_TXD	UART3 transmit data
9	UART5_RXD	UART5 receive data
10	UART3_RXD	UART3 receive data

2.4.17 USB HUB 接口 (HUB1/HUB2)

表 2-25 USB HUB 接口 1 (HUB1)

引脚	信号定义	描述
1	USB_PWR1	USB power supply1
2	DM1	USB data 1-
3	DP1	USB data 1+
4	GND	GND
5	USB_PWR2	USB power supply2
6	DM2	USB data 2-
7	DP2	USB data 2+
8	GND	GND

表 2-26 USB HUB 接口 2 (HUB2)

引脚	信号定义	描述
1	USB_PWR3	USB power supply 3

引脚	信号定义	描述
2	DM3	USB data 3-
3	DP3	USB data 3+
4	GND	GND
5	USB_PWR4	USB power supply 4
6	DM4	USB data 4-
7	DP4	USB data 4+
8	GND	GND

2.4.18 SDIO 接口（J9/J27/J24）

表 2-27 TF 卡接（J9）

引脚	信号定义	描述
1	SD2_DATA2	SD2 data 2
2	SD2_DATA3	SD2 data 3
3	SD2_CMD	SD2 command signal
4	3P3V	3.3V
5	SD2_CLK	SD2 clock
6	GND	GND
7	SD2_DATA0	SD2 data 0
8	SD2_DATA1	SD2 data 1
9	SD2_CD	SD2 card detect signal
10	GND	GND

表 2-28 SDIO 接口 1（J27）

引脚	信号定义	描述
1	SD1_CD	SD1 card detect signal
2	SD1_WP	SD1 card write protect detect signal
3	SD1_CMD	SD1 command signal
4	SD1_CLK	SD1 clock
5	3P3V	3.3V
6	GND	GND
7	SD1_DATA2	SD1 data 2
8	SD1_DATA3	SD1 data 3
9	SD1_DATA0	SD1 data 0
10	SD1_DATA1	SD1 data 1

表 2-29 SDIO 接口 2（J24）

引脚	信号定义	描述
1	3P3V	3.3V

引脚	信号定义	描述
2	GND	GND
3	SD2_DATA7	SD2 data 7
4	SD1_DATA7	SD1 data 7
5	SD2_DATA6	SD2 data 6
6	SD1_DATA6	SD1 data 6
7	SD2_DATA5	SD2 data 5
8	SD1_DATA5	SD1 data 5
9	SD2_DATA4	SD2 data 4
10	SD1_DATA4	SD1 data 4

2.4.19 启动配置开关（SW1）

表 2-30 启动配置开关

引脚	信号定义	描述
1	BT_CFG1_6	BT_CFG1_6
2	BT_CFG1_5	BT_CFG1_5
3	BT_CFG2_6	BT_CFG2_6
4	BT_CFG2_5	BT_CFG2_5
5	BT_CFG2_4	BT_CFG2_4
6	BT_CFG2_3	BT_CFG2_3

表 2-31 通过 DIP1 至 DIP6 配置 SBC9000 启动方式

位号	1	2	3	4	5	6	启动方式
状态	ON	ON	ON	OFF	ON	ON	8-bit eMMC
	ON	OFF	X	ON	OFF	ON	4-bit TF Card
	ON	OFF	X	ON	OFF	OFF	4-bit SDIO1

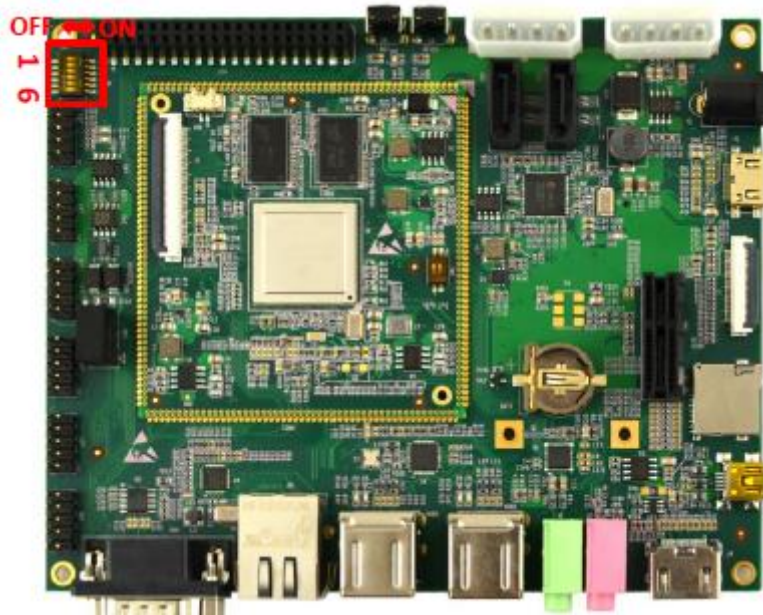


图 2-4 扩展板启动配置开关状态说明

2.4.20 按钮

表 2-32 按钮

引脚	描述
S5	Reset button
S8	User defined button

2.4.21 LED

表 2-33 LED

引脚	描述
D40	Power LED
D39	User defined LED
D41	User defined LED

第3章 准备工作

在开始使用 SBC9000 之前，请阅读以下内容，以便熟悉开发过程中涉及的系统映像、驱动代码和工具等。

3.1 软件简介

以下列表列出了后面将会用到的 Linux 和 Android 两个版本的操作系统，以及各种驱动介绍。

表 3-1 操作系统和驱动

分类		注释
操作系统	Linux	版本 3.10.17
	Android	版本 4.4.2
设备驱动	Serial	串行接口驱动
	RTC	硬件时钟驱动
	Net	10/100/1Gb IEEE1588 以太网
	Display	三个显示接口（RGB、LVDS 和 HDMI 1.4a）
	mmc/sd	一个 SD 3.0/SDXC 插槽和 eMMC
	USB	5 个高速 USB 接口（4xHost, 1xOTG）
	Audio	模拟音频（Audio out 和 Mic In）数字音频（HDMI）
	Camera	一个摄像头接口(1xParallel)
	LED	用户 LED 驱动

3.2 关于 Linux 系统

以下列表列出了建立 Linux 系统需要的映像文件和 eMMC 存储器分区。

表 3-2 Linux 映像

映像	路径
u-boot 映像	u-boot.imx
内核映像	UImage/imx6q-sbc9000.dtb
文件系统	fsl-image-fb-sbc9000.tar.bz2

表 3-3 eMMC 存储分区

分区类型/索引	名称	起始偏移量	容量	文件系统	内容
N/A	BOOT Loader	0	4MB	N/A	u-boot.imx
N/A	Kernel	4M	20MB	FAT	ulmage, imx6q-sbc900 0.dtb
Primary 1	Rootfs	20M	Total - Other	EXT3	fsl-image-fb-sb c9000.tar.bz2

- **分区类型/索引**: 定义保存在 MBR 中。
- **名称**: 只针对 Android 系统才具有实际意义; 您可以在建立 Linux 分区时忽略该项内容。
- **起始偏移量**: 分区起始位置, 单位 MB。

3.3 关于 Android 系统

以下列表列出了建立 Android 系统需要的映像文件和 eMMC 存储器分区。

表 3-4 Android 映像

映像	路径
u-boot 映像	u-boot.bin
启动映像	boot.img
Android 系统根映像	system.img
Recovery 根映像	recovery.img

表 3-5 Android 存储分区

分区类型/索引	名称	起始偏移量	容量	文件系统	内容
N/A	BOOT Loader	0	1MB	N/A	bootloader
主分区 1	Boot	8M	8MB	boot.img format, a kernel + ramdisk	boot.img
主分区 2	Recovery	Boot 分区后	8MB	boot.img format, a kernel + ramdisk	recovery.img
逻辑分区 4 (扩展分区 3)	DATA	Recovery 分区后	>1024MB	EXT4. Mount as /data	系统应用程序 数据
逻辑分区 5 (扩	SYSTEM	Recovery 分区后	512MB	EXT4. Mount	Android 系统文

分区类型/索引	名称	起始偏移量	容量	文件系统	内容
展分区 3)				as /system	件, 挂载到 /system 目录
逻辑分区 6 (扩展分区 3)	CACHE	SYSTEM 分区后	512MB	EXT4. Mount as /cache	Android 缓存, 用于 OTA 映像保存
逻辑分区 7 (扩展分区 3)	device	CACHE 分区后	8MB	EXT4 Mount at /device	保存 MAC 地址
逻辑分区 8 (扩展分区 3)	Misc	Vendor 分区后	8M	N/A	恢复和保存 bootloader 信息
主分区 4	MEDIA	Misc 分区后	Total - Other images	VFAT	内部媒体分区, 挂载到 /mnt/sdcard 目录

- **SYSTEM:** 系统分区, 用于保存 Android 文件。
- **DATA:** 数据分区, 用于保存应用程序的解压数据和系统配置数据库等。

在正常模式下, 根文件系统会从 ramdisk 进行挂载。而 recovery 模式下, 根文件系统会从 RECOVERY 分区进行挂载。

3.4 设置超级终端

使用 DB9 交叉串口线连接 SBC9000 的 Debug 接口 CN4 和 PC 的串口, 然后在 PC 桌面选择开始 > 程序 > 附件 > 通讯 > 超级终端, 根据下图中的参数进行设置;

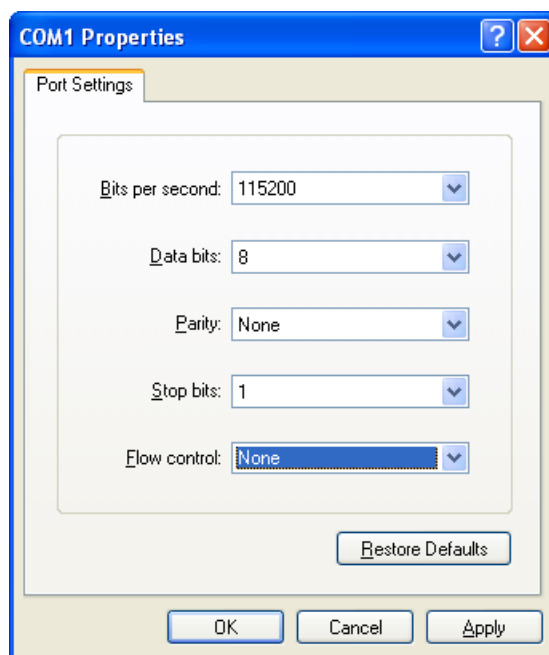




图 3-1 设置超级终端

第4章 下载和运行系统

现在您可以开始已有的系统映像文件下载到 SBC9000 上并运行系统了。

注意：

-  以下章节中的每一条指令前都加上了符号“•”，以便防止由于指令较长占用多行而造成误解。
-  请注意命令行中的空格：漏掉任何空格都会造成程序运行失败。

4.1 Linux 和 Android 系统下载和运行

4.1.1 使用 Mfgtools 下载和运行 Linux 和 Android

- 1) 从 [Embest 官方网站](#) 下载 SBC9000 Linux 和 Android 映像及它们的烧写工具至 PC 硬盘的根目录下（假设为 C:\ 目录）；
- 2) 用一根 Mini USB 线连接 SBC9000 的 USB OTG 接口(J7)和 PC 的 USB Host 口；
- 3) 将 Mini9000 上的启动配置核心板开关 SW1 按照下表中的参数设置为 MFG 工具模式；

表 4-1 启动开关设置

Switch	D1	D2
SW1	OFF	ON

- 4) 拷贝映像文件

Linux: 将下载的 Linux 映像文件解压，然后将除 fsl-image-fb-sbc9000.sdcard 以外的所有文件复制至 SBC9000 映像烧写工具“Mfgtools-Rel-12.04.01_ER_MX6Q_UPDATER \Profiles\MX6Q Linux Update\OS Firmware\files\”目录下，拷贝完成后，files 目录下应存在以下文件：

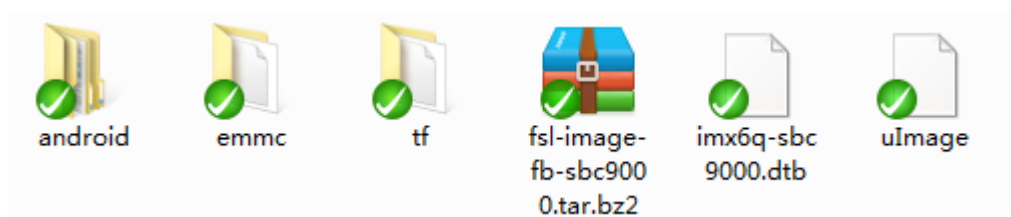


图 4-1 Files 目录下文件

Android: 将下载的 Android 映像文件解压，然后将所有文件拷贝至 SBC9000 映像烧写工具 Mfgtools-Rel-12.04.01_ER_MX6Q_UPDATER \Profiles\MX6Q Linux Update\OS Firmware\files\android\，拷贝完成后 android 目录下应存在以下文件



图 4-2 Android 目录下文件

- 5) 运行“tools\Mfgtools-Rel-12.04.01_ER_MX6Q_UPDATER”目录下的“MfgTool.exe”程序，然后启动 SBC9000，可以看到“MfgTool.exe”程序显示检测到设备，如下图所示：



图 4-3 Mfgtool 软件界面

- 6) 从以上界面菜单栏的 Options 进入配置界面，然后选择将系统映像（SBC9000 支持 Yocto 和 Android）烧写到 eMMC 或 TF 卡（默认烧写到 eMMC），如下图所示：

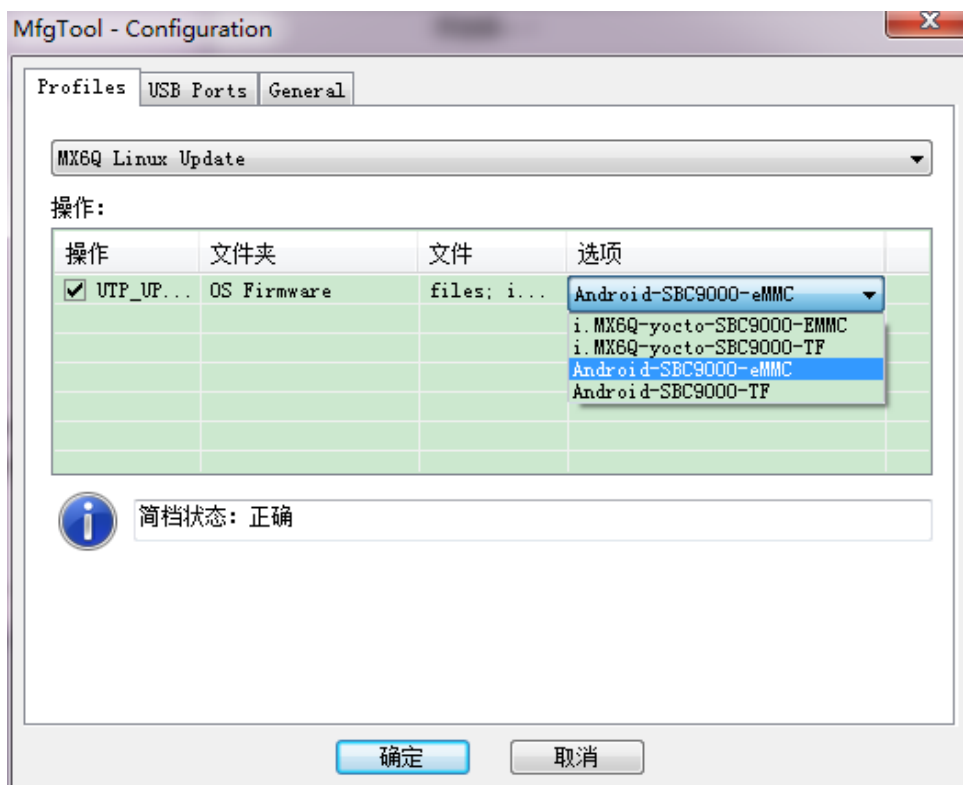


图 4-4 Mfgtool 配置

- 7) 单击“开始”按钮进行烧写；当烧写过程中弹出“格式化磁盘”对话框时，请单击“取消”；

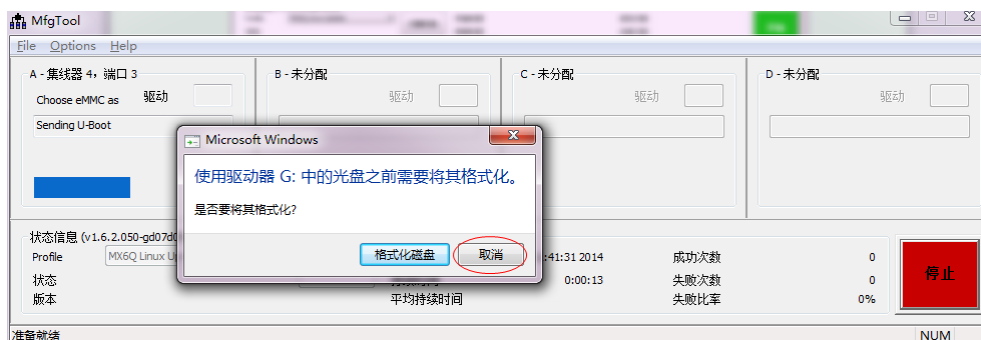


图 4-5 开始烧写

- 8) 当看到如下图所示的绿色进度条时，请单击“停止”按钮完成烧写；



图 4-6 烧写完成

- 9) 断开 SBC9000 的电源，并按照下表中的参数将 SW1 配置开关设置为 eMMC 或 TF 卡启动模式；

表 4-2 eMMC 启动模式配置

核心板	Switch		D1			D2	
	SW1		ON			OFF	
扩展板	Switch	D1	D2	D3	D4	D5	D6
	SW1	ON	ON	OFF	ON	ON	ON

表 4-3 TF 卡启动配置

核心板	Switch		D1			D2	
	SW1		ON			OFF	
扩展板	Switch	D1	D2	D3	D4	D5	D6
	SW1	ON	OFF	OFF	ON	OFF	ON

- 10) 设置完成后接通 5V 电源即可启动系统。

4.1.2 使用 Linux Host 将 Linux 系统下载到 TF 卡并运行

- 1) 从 [英蓓特官方网站](#) 下载 Linux 映像文件，解压后获得映像文件 fsl-image-fb-sbc9000.sdcard，或者通过编译 Yocto 工程来自动生成（详细信息请参考 5.1.2 章节）；
- 2) 在 Linux 系统下执行以下命令来将系统映像烧写到 TF 卡中；
 - `$ sudo dd if= fsl-image-fb-sbc9000.sdcard of=/dev/sd<partition> bs=1M`
 - `$ sync`
- 3) 断开 SBC9000 的电源，并根据表 4-3 中的方法将 SW1 开关设置为 TF 卡启动模式；

4) 将 TF 卡插入 SBC9000 的 TF 卡插槽，然后接通电源即可从 TF 启动 Linux 系统。

4.2 显示模式设置

SBC9000 的 Linux 和 Android 系统都支持多种显示模式输出。用户可通过设置 uboot 参数来选择不同的显示输出模式。当终端窗口中系统启动信息显示 “Hit any key to stop autoboot” 时按下键盘任意键即可进入 uboot，如下表所示：

表 4-4 进入 uboot

```
U-Boot 2013.04-04992-g002bd44 (Sep 23 2014 - 14:48:51)

CPU:   Freescale i.MX6Q rev1.2 at 792 MHz
CPU:   Temperature 31 C, calibration data: 0x5654b67d
Reset cause: POR
Board: MX6Q-SBC9000
DRAM:  1 GiB
force_idle_bus: sda=0 scl=1 sda.gp=0xcb scl.gp=0x5
force_idle_bus: failed to clear bus, sda=0 scl=1
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
*** Warning - bad CRC, using default environment

wait_for_sr_state: Arbitration lost sr=93 cr=80 state=2020
i2c_init_transfer: failed for chip 0x4 retry=0
force_idle_bus: sda=0 scl=1 sda.gp=0xcb scl.gp=0x5
force_idle_bus: failed to clear bus, sda=0 scl=1
i2c_init_transfer: give up i2c_regs=021a8000
wait_for_sr_state: failed sr=a1 cr=80 state=2000
wait_for_sr_state: failed sr=a1 cr=80 state=2000
i2c_imx_stop: trigger stop failed
i2c_init_transfer: failed for chip 0x38 retry=0
force_idle_bus: sda=0 scl=1 sda.gp=0xcb scl.gp=0x5
force_idle_bus: failed to clear bus, sda=0 scl=1
i2c_init_transfer: give up i2c_regs=021a8000
wait_for_sr_state: Arbitration lost sr=93 cr=80 state=2020
i2c_init_transfer: failed for chip 0x48 retry=0
force_idle_bus: sda=0 scl=1 sda.gp=0xcb scl.gp=0x5
force_idle_bus: failed to clear bus, sda=0 scl=1
i2c_init_transfer: give up i2c_regs=021a8000
No panel detected: default to HDMI
unsupported panel HDMI
In:    serial
```

```

Out:  serial
Err:  serial
Net:  ----phy_id= 0x4dd072
FEC [PRIME]
Warning: failed to set MAC address

Normal Boot
Hit any key to stop autoboot:  0  (按 PC 键盘的任意键进入 uboot)
MX6QSBC9000 U-Boot >

```

以下内容包含了各种显示模式的配置命令；

- 使用 4.3" LCD 屏幕

- MX6QSBC9000 U-Boot > **setenv dispmode**
video=mxcfb0:dev=lcd,4.3inch_LCD,if=RGB24
video=mxcfb1:dev=ldb,LDB-XGA,if=RGB666 fbmem=10M vmalloc=400M
androidboot.console=ttymx1 androidboot.hardware=freescale calibration
- MX6QSBC9000 U-Boot > **saveenv**

- 使用 7" LCD 屏幕

- MX6QSBC9000 U-Boot > **setenv dispmode**
video=mxcfb0:dev=lcd,7inch_LCD,if=RGB24
video=mxcfb1:dev=ldb,LDB-XGA,if=RGB666 fbmem=10M vmalloc=400M
androidboot.console=ttymx1 calibration androidboot.hardware=freescale
calibration
- MX6QSBC9000 U-Boot > **saveenv**

- 使用 9.7" LVDS 屏幕

- MX6QSBC9000 U-Boot > **setenv dispmode**
video=mxcfb0:dev=ldb,LDB-XGA,if=RGB666 video=mxcfb1:off fbmem=10M
vmalloc=400M androidboot.console=ttymx1 androidboot.hardware=freescale
- MX6QSBC9000 U-Boot > **saveenv**

- 使用 HDMI 显示器（默认）

- MX6QSBC9000 U-Boot > **setenv dispmode**
video=mxcfb0:dev=hdmi,1280x720M@60,if=RGB24
video=mxcfb1:dev=ldb,LDB-XGA,if=RGB666 fbmem=40M vmalloc=400M

```
androidboot.console=ttyMXC1 androidboot.hardware=freescale
```

- MX6QSBC9000 U-Boot > **saveenv**

注意:

📖 HDMI 显示模式默认分辨率为 1280x720。用户可以修改分辨率,例如 1920x1080 或 640x480,修改方法为:将上述参数中的“dev=hdmi,1280x720M@60”更改为“dev=hdmi,1920x1080M@60”,其他参数不变。

- 使用 VGA8000 进行显示

- MX6QSBC9000 U-Boot > **setenv dispmode**

```
video=mxcfb0:dev=lcd,1024x768M@60,if=RGB24
```

```
video=mxcfb1:dev=ldb,LDB-XGA,if=RGB666 fbmem=10M vmalloc=400M
```

```
androidboot.console=ttyMXC1 androidboot.hardware=freescale
```

- MX6QSBC9000 U-Boot > **saveenv**


注意:

📖 VGA 显示模式默认分辨率为 1024x768。用户可以将分辨率修改为 800x600、1440x900 或 1280x1024,例如将上述参数中的“video=mxcfb0:dev=lcd,1024x768M@60”更改为“video=mxcfb0:dev=lcd,800x600M@60”,其它参数不变。

第5章 制作映像文件

本章节将介绍如何使用 BSP 数据包来制作映像文件。针对 SBC9000 提供的 BSP 数据包包含了可用于生成 u-boot 启动文件、Linux 内核映像和 Android 文件系统的二进制代码、源代码和支持文件。

注意：

 以下章节中的命令在 Ubuntu 系统中执行。

5.1 制作 Linux 系统映像

下面将介绍两种 Linux 系统映像的制作方法，即直接编译和使用 Yocto 进行编译。

5.1.1 直接编译

1) 执行以下命令来获取交叉编译工具：

- `$ cd ~`
- `$ git clone git://github.com/embest-tech/fsl-linaro-toolchain.git`

2) 从[英蓓特官方网站 SBC9000 产品页面](#)的“资源下载”标签下载最新的 Linux 系统源代码并保存到用户目录；或者通过 git 工具从 Internet 获取 u-boot-imx 和 linux-imx 源代码，命令如下：

- `$ git clone https://github.com/embest-tech/u-boot-imx.git`
- `$ git checkout -b embest_imx_v2013.04_3.10.17_1.0.0_ga`
`origin/embest_imx_v2013.04_3.10.17_1.0.0_ga`
- `$ git clone https://github.com/embest-tech/linux-imx.git`
- `$ git checkout -b embest_imx_3.10.17_1.0.0_ga`
`remotes/origin/embest_imx_3.10.17_1.0.0_ga`

3) 执行以下命令来解压缩下载的源代码：

- `$ cd ~`

- `$ tar xvf u-boot-imx.tar.bz2`
- `$ tar xvf linux-imx.tar.bz2`

4) 执行以下命令来编译启动映像:

- `$ cd ~/u-boot-imx`
- `$ export ARCH=arm`
- `$ export CROSS_COMPILE=~/fsl-linaro-toolchain/bin/arm-fsl-linux-gnueabi-`
- `$ make distclean`
- `$ make mx6q_sbc9000_emmc_config`
- `$ make`

执行完成后, 在当前目录下会生成 `u-boot.imx` 文件;

注意:

📖 以上的 `make mx6q_sbc9000_emmc_config` 命令用于编译 eMMC 启动映像。如果需要从 TF 卡启动, 请执行 `make mx6q_sbc9000_tf_config`。

5) 执行以下命令来编译内核映像:

- `$ export PATH=~/u-boot-imx/tools:$PATH`
- `$ cd ~/linux-imx`
- `$ export ARCH=arm`
- `$ export CROSS_COMPILE=~/fsl-linaro-toolchain/bin/arm-fsl-linux-gnueabi-`
- `$ make imx_v7_sbc9000_defconfig`
- `$ make ulmage LOADADDR=0x10008000`
- `$ make imx6q-sbc9000.dtb`

执行完成后, 在 `arch/arm/boot/` 目录和 `arch/arm/boot/dts/` 目录下会分别生成 `ulmage` 文件和 `imx6q-sbc9000.dtb` 文件。

注意:

- 📖 用于生成内核和 ramfs 等映像文件的 mkimage 工具是在编译 u-boot.bin 文件后自动生成的并保存于 tools/目录下，因此在编译内核映像前需要首先完成 uboot 的编译。
- 📖 根据编译时设置的启动方式，用新编译生成的 u-boot.imx 覆盖 SBC9000 映像烧写工具 Mfgtools-Rel-12.04.01_ER_MX6Q_UPDATER \Profiles\MX6Q Linux Update\OS Firmware\files\目录下 emmc 或 tf 文件夹下的同名文件，用 ulmage 和 imx6q-sbc9000.dtb 文件覆盖映像烧写工具 Mfgtools-Rel-12.04.01_ER_MX6Q_UPDATER \Profiles\MX6Q Linux Update\OS Firmware\files\目录下的同名文件，然后从 4.1.1 节第 2) 步开始操作即可验证新的 Linux 系统。

5.1.2 使用 Yocto 编译

1) 执行以下命令来获取 repo 工具:

- `$ mkdir ~/bin`
- `$ curl https://raw.githubusercontent.com/android/tools_repo/master/repo > ~/bin/repo`
- `$ chmod a+x ~/bin/repo`
- `$ export PATH=~/bin:$PATH`

2) 执行以下命令来获取 Yocto 编译环境:

- `$ mkdir ~/fsl-arm-yocto-bsp`
- `$ cd ~/fsl-arm-yocto-bsp`
- `$ repo init --no-repo-verify --repo-url=git://github.com/android/tools_repo.git -u git://github.com/embest-tech/fsl-arm-yocto-bsp.git -b embest_imx-3.10.17-1.0.0_ga`
- `$ repo sync`

3) 执行以下命令，使用 yocto 构建 linux 系统:

- `$ cd ~/fsl-arm-yocto-bsp`
- `$ MACHINE=sbc9000 source fsl-setup-release.sh -b build -e fb`
- `$ bitbake fsl-image-fb`

执行完成后，在 build/tmp/deploy/images/sbc9000 目录下将生成 u-boot.imx、ulmage、ulmage-imx6q-sbc9000.dtb 和 fsl-image-fb-sbc9000.tar.bz2，请将 uIMAGE-imx6q-sbc9000.dtb 重命名 imx6q-sbc9000.dtb。

注意:

- 📖 用编译生成的 u-boot.imx 映像烧写工具 Mfgtools-Rel-12.04.01_ER_MX6Q_UPDATER \Profiles\MX6Q Linux Update\OS Firmware\files\ 目录下的 emmc 或 tf 文件夹下的同名文件, 用 ulmage 、 imx6q-sbc9000.dtb 和 fsl-image-fb-sbc9000.tar.bz2 文件覆盖 Mfgtools-Rel-12.04.01_ER_MX6Q_UPDATER \Profiles\MX6Q Linux Update\OS Firmware\files\ 目录下的的同名文件, 然后从 4.1.1 节第 2) 步开始操作即可验证新的 Linux 系统。
- 📖 Yocto 提供了多种可用的 image 配置, Freescale 也针对自身的 SoC 提供了一些新的特性。详细请参考 “[Freescale Yocto Project User's Guide.pdf](#)”。

4) 重新创建编译环境;

如果已经使用 fsl-setup-release.sh 脚本创建了 build 目录, 则在 PC 重新启动后或打开一个新的 Linux 终端时, 只需要使用 setup-environment 脚本重新配置编译所需要的环境变量即可, 而不需要再次执行 fsl-setup-release.sh 脚本。

- `$ cd ~/fsl-arm-yocto-bsp`
- `$ source setup-environment build`

5) 单独编译 u-boot 或内核映像;

- `$ bitbake u-boot-sbc9000` //编译 u-boot
- `$ bitbake linux-sbc9000` //编译 kernel

注意:

- 📖 当 u-boot 或 kernel 被成功编译后, 如源码文件被修改, 需使用 “-c compile -f” 选项强制 Yocto Project 重新编译:

- `$ bitbake -c compile -f u-boot-sbc9000` //强制编译 u-boot
- `$ bitbake -c compile -f linux-sbc9000` //强制编译内核

5.2 制作 Android 系统映像

1) 执行以下命令来获取 repo 工具;


- `$ mkdir ~/bin`
- `$ curl https://raw.githubusercontent.com/android/tools_repo/master/repo > ~/bin/repo`
- `$ chmod a+x ~/bin/repo`

- `$ export PATH=~/.bin:$PATH`

2) 执行以下命令来获取 Android 源代码；

- `$ mkdir ~/android-imx6-kk4.4.2-1.0.0`
- `$ cd ~/android-imx6-kk4.4.2-1.0.0`
- `$ repo init --no-repo-verify --repo-url=git://github.com/android/tools_repo.git -u git://github.com/embest-tech/imx-manifest.git -m embest_android_kk4.4.2_1.0.0`
- `$ repo sync`

注意：

 Android 系统源代码也可以从[英蓓特官方网站 SBC9000 产品页面](#)的“资源下载”标签下载，然后执行以下命令进行解压缩；

`$ cd ~`

`$ tar xvf android-imx6-kk4.4.2-1.0.0-xxx.tar.bz2`（请根据下载的文件名来替换 xxx）

3) 使用记事本打开“android-imx6-kk4.4.2-1.0.0/device/fsl/sbc9000_6solo/”目录下的“BoardConfig.mk”文件，然后按照以下内容修改“BUILD_TARGET_LOCATION”参数来选择启动方式；

- 从 eMMC 启动：BUILD_TARGET_LOCATION ? =emmc
- 从 TF 卡启动：BUILD_TARGET_LOCATION ? =sdmmc

4) 执行以下命令来编译 Android 映像；

- `$ cd ~/android-imx6-kk4.4.2-1.0.0`
- `$ source build/envsetup.sh`
- `$ lunch sbc9000_6q-user`
- `$ make clean`
- `$ make`

执行上述命令后，Android 映像文件全部生成，编译生成的映像保存在“android-imx6-kk4.4.2-1.0.0/out/target/product/sbc9000_6q/”目录下；下表列出了编译完成后生成的映像以及目录；

表 5-1 生成的映像和目录

映像和目录	注释
-------	----

映像和目录	注释
root/	根文件系统目录，挂载于根目录
system/	Android 系统目录，挂载于/system 目录
data/	Android 数据区，挂载于/data 目录
recovery/	Recovery 模式下的根文件系统目录，不直接使用
boot.img	复合映像，包含内核 zImage，ramdisk 和启动参数
ramdisk.img	Root/生成的 ramdisk 映像，不直接使用
system.img	System/生成的 EXT4 映像，可通过 dd 命令写入 SD/eMMC 存储卡的 SYSTEM 分区
recovery.img	Recovery/生成的 EXT4 映像，可通过 dd 命令写入 SD/eMMC 存储卡的 RECOVERY 分区
u-boot.bin	uboot 映像

注意:

- Android 映像需要在用户模式下编译;
- 关于搭建开发环境的更多信息，请访问 <http://source.android.com/source/building.html>。

5) 执行以下命令来单独编译生成 boot.img 映像;

- `$ source build/envsetup.sh`
- `$ lunch sbc9000_6q-user`
- `$ make bootimage`

执行完成后在“android-imx6-kk4.4.2-1.0.0/out/target/product/sbc9000_6q/”目录下会生成 boot.img 映像文件。

注意:

- 根据编译时配置的启动方式，用编译生成的 boot.img、recovery.img、system.img 和 u-boot.bin 文件覆盖 Android 映像烧写工具“Mfgtools-Rel-4.1.0_130816_MX6DL_UPDATER\ Profiles\MX6DL Linux Update\OS Firmware\files\android”目录下 emmc 或 tf 文件夹下的同名文件，然后从第 4.1.1 节中第 2) 步开始操作即可验证新的 Android 系统。

5.3 使用 Yocto 编译 Linux 上层应用程序

本章节将以应用程序 hello_world.c 为实例来介绍如何使用 Yocto 编译 Linux 上层应用程序。

1) 请执行以下命令来生成交叉编译工具;

- `$ cd ~/fsl-arm-yocto-bsp`
- `$MACHINE=sbc9000 source fsl-setup-release.sh -b build -e fb`
- `$ bitbake meta-toolchain //生成交叉编译器`
- `$ cd ~/fsl-arm-yocto-bsp/build/tmp/deploy/sdk/`

执行完成后在当前目录下会生成交叉编译工具安装脚本文件
poky-eglibc-x86_64-meta-toolchain-cortexa9hf-vfp-neon-toolchain-1.5.2.sh;


2) 执行以下命令来安装交叉编译工具; 默认安装路径是 “/opt/poky/1.5.2”;

- `$sudo ./poky-eglibc-x86_64-meta-toolchain-cortexa9hf-vfp-neon-toolchain-1.5.2.sh`
- `$source/opt/poky/1.5.2/environment-setup-cortexa9hf-vfp-neon-poky-linux-gnueabi`

3) 执行以下命令来编译 hello_world.c;

- `$ arm-poky-linux-gnueabi-gcc -o hello_world hello_world.c`

注意:

 由于 Yocto 工程是采用硬浮点编译器, 因此上层应用程序也应该使用硬浮点交叉编译器, 并且编译器版本必须保持一致。

第6章 测试

本章节将介绍如何测试 SBC9000 上的各个接口。以下测试均在 Yocto 下进行，并且在没有特别说明的情况下，所有命令均通过超级终端执行。

注意：

在 PC 键盘上按下 Ctrl+C 组合键可以退出测试程序。

6.1 LED 测试

SBC9000 板载三个用户 LED 灯，分别为核心板上的 D4(LED)，扩展板上的 D39(LED2) 和 D41(LED1)。

1) 执行以下命令来熄灭用户灯；

- `root@sb9000:~# echo 1 > /sys/class/leds/user_led1/brightness`
- `root@sb9000:~# echo 1 > /sys/class/leds/user_led2/brightness`
- `root@sb9000:~# echo 1 > /sys/class/leds/sys_led/brightness`

2) 执行以下命令来点亮用户灯；

- `root@sb9000:~# echo 0 > /sys/class/leds/user_led1/brightness`
- `root@sb9000:~# echo 0 > /sys/class/leds/user_led2/brightness`
- `root@sb9000:~# echo 0 > /sys/class/leds/sys_led/brightness`

6.2 按钮测试

在 SBC9000 的扩展板上有一个用户按钮 S8。请执行以下命令执行 `evtest` 工具可测试用户按键工作是否正常：

- `root@sb9000:~# evtest /dev/input/event1`

然后按下按钮 S8 即可看到以下串口信息：

表 6-1 按钮测试信息

```
Input driver version is 1.0.1evdev: (EVIOCGBIT): Suspicious buffer size
511, limiting output to 64 bytes. See
http://userweb.kernel.org/~dtor/eviocgbit-bug.html

Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys"
Supported events:
  Event type 0 (Sync)
  Event type 1 (Key)
    Event code 102 (Home)
Testing ... (interrupt to exit)
Event: time 278.544212, type 1 (Key), code 102 (Home), value 1
Event: time 278.544220, ----- Report Sync -----
Event: time 278.714484, type 1 (Key), code 102 (Home), value 0
Event: time 278.714487, ----- Report Sync -----
Event: time 279.080212, type 1 (Key), code 102 (Home), value 1
Event: time 279.080217, ----- Report Sync -----
Event: time 279.293936, type 1 (Key), code 102 (Home), value 0
Event: time 279.293940, ----- Report Sync -----
```

其中“value 1”表示按键按下，“value 0”表示按键松开。

6.3 触摸屏测试

- 1) 启动 SBC9000 后，执行以下命令来启动触摸屏校准程序，然后按照屏幕上的提示触摸

“+”图标来完成校准；

- `root@sbc9000:~# ts_calibrate`

- 2) 校准完成后，执行以下指令来运行触摸屏测试程序，然后按照屏幕上的提示进行画点、

画线测试；

- `root@sbc9000:~# ts_test`

6.4 RTC 测试

- 1) 执行以下命令来将 SBC9000 的系统时间设置为 2014 年 3 月 11 日晚上 5 时正；

- `root@sbc9000:~# date 031117002014`

终端窗口显示信息如下；

表 6-2 设置系统时间

```
Tue Mar 11 17:00:00 UTC 2014
```

2) 执行以下命令来把系统时钟写入 RTC;

- `root@sbc9000:~# hwclock -w`

3) 执行以下命令来读取 RTC;

- `root@sbc9000:~# hwclock`

终端窗口显示信息如下;

表 6-3 读取 RTC

```
Tue Mar 11 17:01:01 2014 0.000000 seconds
```

以上信息显示系统时钟已成功保存到硬件时钟里。

4) 重新启动 SBC9000, 然后执行以下命令来恢复系统时钟;

- `root@sbc9000:~# hwclock -s`

- `root@sbc9000:~# date`


终端窗口显示信息如下;

表 6-4 系统时钟

```
Tue Mar 11 17:03:33 UTC 2014
```

以上信息显示系统时钟已成功恢复为硬件时钟。

注意:

 SBC9000 的扩展板上需要装上纽扣电池才能在重新启动系统时保持正确的硬件时间。纽扣电池型号为 CR1220, 并未随产品附送, 需要单独购买。

6.5 TF 卡测试

1) 将一张 TF 卡插入 SBC9000, 系统会自动检测出 TF 卡并显示以下信息;

表 6-5 TF 卡信息


```
mmc1: host does not support reading read-only switch. assuming
write-enable.
mmc1: new high speed SD card at address b7f5
mmcblk1: mmc0:b7f5 SD02G 1.83 GiB
mmcblk1: p1
```

2) 执行以下命令来将 TF 卡手动挂载到/mnt 目录下并查看其中的内容;


- `root@sbc9000:~# mount -t vfat /dev/mmcblk1p1 /mnt`
- `root@sbc9000:~# ls /mnt`

终端窗口显示信息如下;

表 6-6 TF 卡内容

Mlo	nand	ramdisk.gz	u-boot.bin ulmage
-----	------	------------	-------------------

注意:

 插入 TF 卡时, 请快速插入, 否则系统可能无法识别 TF 卡。如果第一次无法成功识别, 请再次尝试。

6.6 USB HOST 测试

1) 将一个 U 盘插入 SBC9000 的 USB HUB 接口, 系统会自动检测出设备并显示以下信息;

表 6-7 USB 设备信息

```
usb 2-1.4: new high speed USB device number 3 using fsl-ehci
scsi1 : usb-storage 2-1.4:1.0
scsi 1:0:0:0: Direct-Access      Generic- Multi-Card          1.00 PQ: 0
ANSI: 0 CCS
sd 1:0:0:0: [sda] 15394816 512-byte logical blocks: (7.88 GB/7.34 GiB)
sd 1:0:0:0: [sda] Write Protect is off
sd 1:0:0:0: [sda] No Caching mode page present
sd 1:0:0:0: [sda] Assuming drive cache: write through
sd 1:0:0:0: [sda] No Caching mode page present
sd 1:0:0:0: [sda] Assuming drive cache: write through
sda: sda1 sda2
sd 1:0:0:0: [sda] No Caching mode page present
sd 1:0:0:0: [sda] Assuming drive cache: write through
sd 1:0:0:0: [sda] Attached SCSI removable disk
EXT2-fs (sda2): warning: mounting ext3 filesystem as ext2
```

```
EXT2-fs (sda2): warning: mounting unchecked fs, running e2fsck is recommended
```

2) 执行以下命令来手动将 U 盘挂载到/mnt 目录下;

- `root@sbc9000:~# mount -t vfat /dev/sda1 /mnt/`

3) 执行以下命令来察看 U 盘中的文件;

- `root@sbc9000:~# ls /mnt/`

终端窗口显示信息如下;

表 6-8 U 盘内容

```
speed.avi  madplay  i believe.mp3  i believe.wav
```

6.7 USB Device 测试

在 Linux 系统下, SBC9000 的 OTG 接口(J7)默认作为 USB Device。通过一根 MiniUSB 线连接 SBC9000 的 OTG 接口与 PC 上的 USB Host 可以实现网络通信。请根据以下步骤进行测试 (以 Window 7 系统为例):

1) 使用一根 USB MINI B 转 USB A 的线缆将 SBC9000 连接到 PC 并安装 Linux USB Ethernet 驱动; (详细安装方法请参考附录 2)

2) 执行以下命令来设置和查看 SBC9000 的 USB OTG 端口的 IP 地址 (以下 IP 地址仅为范例, 您可以选择其他 IP 地址, 但需要确保该 IP 地址与 PC 的以太网端口 IP 处于不同网段);

- `root@sbc9000:~# ifconfig usb0 192.168.1.115`

- `root@sbc9000:~# ifconfig`

终端窗口显示信息如下;

表 6-9 USB 端口 IP 配置

```
lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
```

```

TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

usb0    Link encap:Ethernet  HWaddr 1E:B1:B5:11:E5:46
        inet addr:192.168.1.115  Bcast:192.168.1.255
        Mask:255.255.255.0
        inet6 addr: fe80::1cb1:b5ff:fe11:e546/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500
        Metric:1
        RX packets:14 errors:0 dropped:0 overruns:0 frame:0
        TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:1338 (1.3 KiB) TX bytes:4994 (4.8 KiB)
    
```

- 3) 在 PC 桌面上右键单击“网络”，并选择“属性”进入网络连接窗口；再单击“更改适配器设置”按钮，窗口中会出现一个新增的本地连接图标，如下图所示：



图 6-1 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(TCI/IP)”打开以下窗口；

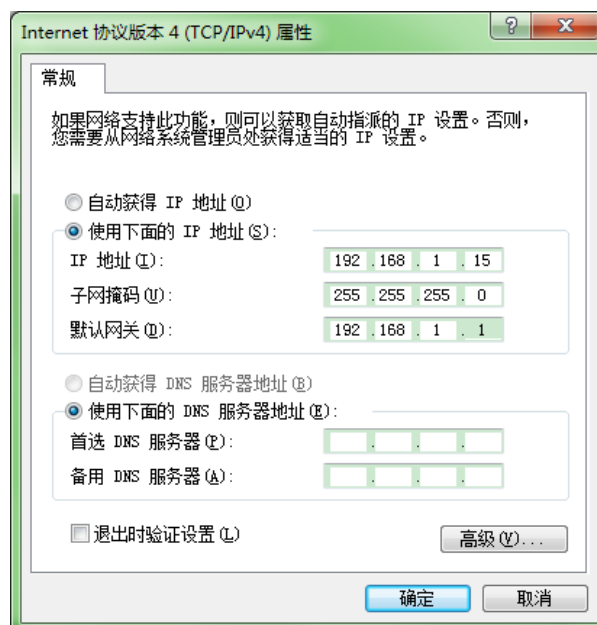


图 6-2 配置 IP 地址

将 IP 地址设置为与 SBC9000 的 USB OTG 端口相同的网段，然后单击“确定”；

5) 执行以下命令来测试网络连接：

- `root@sbc9000:~# ping 192.168.1.15`

终端窗口显示信息如下：

表 6-10 Ping 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
```

以上信息表示网络连接正常。

6.8 音频测试

SBC9000 板载音频输入、输出接口，系统自带 `alsa-utils` 音频播放和录制工具。请将耳麦连接到 SBC9000，然后按照以下步骤进行测试。

1) 执行以下命令来启动录音功能：

- `root@sbc9000:~# cd /tmp`
- `root@sbc9000:/tmp# arecord -t wav -c 1 -r 44100 -f S16_LE -v k`

终端窗口显示信息如下：

表 6-11 启动录音功能

```
Recording WAVE 'k' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Plug PCM: Hardware PCM card 0 'k14' 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
  timestamp_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

```

这时您可以通过耳麦来录制音频。

2) 执行以下命令来回放刚才录制的音频;

- `root@sbc9000:/tmp# aplay -t wav -c 2 -r 44100 -f S16_LE -v k`

终端窗口显示信息如下;

表 6-12 录音回放

```

Playing WAVE 'k': Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Plug PCM: Hardware PCM card 0 'k14' 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

```

这时可以通过耳麦听到录音回放。

6.9 HDMI 音频测试

- 1) 使用一根 HDMI 线缆连接 SBC9000 和 HDMI 显示器;
- 2) 接通电源并根据 4.2 章节中的内容将显示模式设置为 HDMI;
- 3) 执行以下命令来测试 HDMI 音频输出;

- `root@sbc9000:~# aplay Windows.wav -D plughw:1,0`

6.10 以太网测试

- 1) 使用一根网线连接 SBC9000 和 PC, 然后执行以下命令来设置 SBC9000 的 IP 地址
(SBC900 的 IP 地址必须与 PC 的 IP 处于同一网段)

- `root@sbc9000:~# ifconfig eth0 192.168.8.52`

终端窗口显示信息如下;

表 6-13 设置 IP 地址

```
eth0: Freescale FEC PHY driver [Generic PHY] (mii_bus:phy_addr=1:04,
irq=-1)
root@sbc9000:~# PHY: 1:04 - Link is Up - 100/Full
```

- 2) 执行以下命令来察看网络配置;

- `root@sbc9000:~# ifconfig`

表 6-14 查看 IP 地址

```
eth0      Link encap:Ethernet  HWaddr 1E:ED:19:27:1A:B3
          inet addr:192.168.8.52  Bcast:192.168.8.255
          Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500
          Metric:1
          RX packets:2 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:366 (366.0 B)  TX bytes:0 (0.0 B)
```

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

3) 执行以下命令来测试网络的连通性;

- `root@sbc9000:~# ping 192.168.8.1`

终端窗口显示信息如下;

表 6-15 网络测试信息

```
PING 192.168.8.1 (192.168.8.1): 56 data bytes
64 bytes from 192.168.8.1: seq=0 ttl=64 time=0.378 ms
64 bytes from 192.168.8.1: seq=1 ttl=64 time=0.285 ms
64 bytes from 192.168.8.1: seq=2 ttl=64 time=0.222 ms
64 bytes from 192.168.8.1: seq=3 ttl=64 time=9.928 ms
64 bytes from 192.168.8.1: seq=4 ttl=64 time=0.217 ms
64 bytes from 192.168.8.1: seq=5 ttl=64 time=0.289 ms

--- 192.168.8.1 ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 0.217/1.886/9.928 ms
```

6.11 CAN 测试

SBC9000 在 J25 上引出了两个 CAN 接口, 分别为 CAN1 和 CAN2, 在 Linux 系统下生成的设备节点分别为 can0 和 can1。本测试将 can1 作为发送端, can0 作为接收端。请参考下图来连接 SBC9000 的两个 CAN 接口。

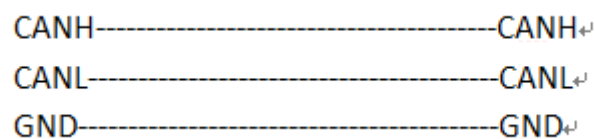


图 6-3 CAN 接口连接

J25 的引脚定义如下图所示：

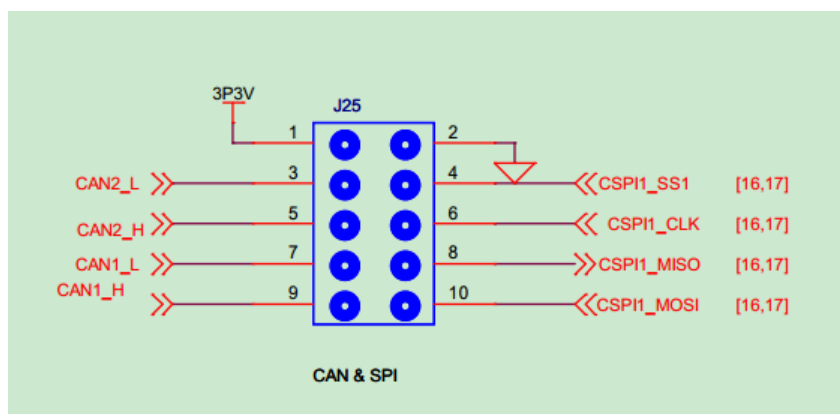


图 6-4 J25 引脚定义

1) 请执行以下命令来将通信波特率设置为 125KBPS，并使能 CAN 设备；

- `root@sbc9000:~# ip link set can0 type can bitrate 125000 triple-sampling on`
- `root@sbc9000:~# ip link set can1 type can bitrate 125000 triple-sampling on`
- `root@sbc9000:~# ip link set can0 up`
- `root@sbc9000:~# ip link set can1 up`

2) 执行以下命令来将 can0 设置为接收端；

- `root@sbc9000:~# candump can0 &`

3) 执行以下命令来让 can1 发送数据；

- `root@sbc9000:~# cansend can1 123#1122334455667788`

终端窗口显示信息如下；

表 6-16 can0 接收到的数据

can0	123	[8]	11 22 33 44 55 66 77 88
------	-----	-----	-------------------------

4) 执行以下命令来关闭 CAN 总线连接；

- `root@sbc9000:~# ip link set can0 down`
- `root@sbc9000:~# ip link set can1 down`

注意:

-  **cansend** 命令每执行一次仅发送一次数据。
-  **CAN** 设备的通信速率不一定设置为 125000，但发送和接受端必须保持一致，并且需要先关闭设备才能进行修改波特率。

6.12 串口测试

除了调试串口外，SBC9000 还带有 UART1 (ttymxc0)、UART3 (ttymxc2) 和 UART5 (ttymxc4) 串口。以下测试将以串口 UART3 为例，其它串口的测试方法相同。

1) 执行以下命令来运行串口测试程序；

- `root@sb9000:~# uart_test -d /dev/ttymxc2 -b 115200`

2) 将串口 UART3 的发送 (TX) 与接收引脚 (RX) 短接起来，进行自发自收测试。终端窗口显示信息如下。

表 6-17 串口测试

```
/dev/ttymxc2 SEND: 1234567890
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 1
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 2
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 3
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 4
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 5
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 6
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 7
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 8
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 9
/dev/ttymxc2 RECV 1 total
/dev/ttymxc2 RECV: 0
```

6.13 Mini-PCle 测试

将一块带有中国电信 SIM 卡的 MC2716(CDMA2000 模块,需另行购买)插入 SBC9000 的 Mini-PCle 插槽 (Mini-PCle 接口支持带电拔插), 然后按照以下步骤进行测试。

1) 执行以下命令来加载 MC2716 驱动;

- `root@sb9000:~# modprobe usbserial vendor=0x19d2 product=0xffed`

终端窗口显示信息如下;

表 6-18 加载驱动

```
usbcore: registered new interface driver usbserial
USB Serial support registered for generic
usbserial_generic 2-1.1:1.0: generic converter detected
usb 2-1.1: generic converter now attached to ttyUSB0
usbserial_generic 2-1.1:1.1: generic converter detected
usb 2-1.1: generic converter now attached to ttyUSB1
usbserial_generic 2-1.1:1.2: generic converter detected
usb 2-1.1: generic converter now attached to ttyUSB2
usbserial_generic 2-1.1:1.3: generic converter detected
usb 2-1.1: generic converter now attached to ttyUSB3
usbcore: registered new interface driver usbserial_generic
usbserial: USB Serial Driver core
```

2) 执行以下命令开始拨号;

- `root@sb9000:~# pppd connect 'chat -v "" "AT" "" "AT" "" "AT&C1" "" "AT" ""
"ATDT#777 CONNECT" user CARD password CARD /dev/ttyUSB0 115200 updetach
nocrtscts nocdtrcts multilink usepeerdns defaultroute debug`

终端窗口显示信息如下;

表 6-19 开始拨号

```
Serial connection established.
using channel 2
Starting negotiation on /dev/ttyUSB0
rcvd [LCP ConfReq id=0x1 <asynmap 0x0> <auth chap MD5> <magic  
0xd6709950> <pcomp> <accomp>]
sent [LCP ConfReq id=0x1 <asynmap 0x0> <magic 0x6734606d>  
<pcomp> <accomp> <mrru 1500> <endpoint [MAC:1e:ed:19:27:1a:b3]>]
sent [LCP ConfAck id=0x1 <asynmap 0x0> <auth chap MD5> <magic
```

```

0xd6709950> <pcomp> <accomp>]
rcvd [LCP ConfRej id=0x1 <mrru 1500> <endpoint
[MAC:1e:ed:19:27:1a:b3]>]
sent [LCP ConfReq id=0x2 <asyncmap 0x0> <magic 0x6734606d>
<pcomp> <accomp>]
rcvd [LCP ConfAck id=0x2 <asyncmap 0x0> <magic 0x6734606d>
<pcomp> <accomp>]
rcvd [CHAP Challenge id=0x1
<911cc74daa1650438af632f437aefc73ad064933>, name = ""]
sent [CHAP Response id=0x1 <b3ccdba9f56ad3a146d0fa3478bc5e41>,
name = "CARD"]
rcvd [CHAP Success id=0x1 ""]
CHAP authentication succeeded
CHAP authentication succeeded
Using interface ppp0
sent [CCP ConfReq id=0x1 <deflate 15> <deflate(old#) 15> <bsd v1 15>]
sent [IPCP ConfReq id=0x1 <compress VJ 0f 01> <addr 0.0.0.0> <ms-dns1
0.0.0.0> <ms-dns3 0.0.0.0>]
rcvd [IPCP ConfReq id=0x1 <compress VJ 0f 00> <addr 115.168.82.165>]
sent [IPCP ConfAck id=0x1 <compress VJ 0f 00> <addr 115.168.82.165>]
rcvd [LCP ProtRej id=0x1 80 fd 01 01 00 0f 1a 04 78 00 18 04 78 00 15 03
2f]
Protocol-Reject for 'Compression Control Protocol' (0x80fd) received
rcvd [IPCP ConfNak id=0x1 <addr 14.27.146.31> <ms-dns1
202.96.128.86> <ms-dns3 202.96.134.133>]
sent [IPCP ConfReq id=0x2 <compress VJ 0f 01> <addr 14.27.146.31>
<ms-dns1 202.96.128.86> <ms-dns3 202.96.134.133>]
rcvd [IPCP ConfAck id=0x2 <compress VJ 0f 01> <addr 14.27.146.31>
<ms-dns1 202.96.128.86> <ms-dns3 202.96.134.133>]
local IP address 14.27.146.31
remote IP address 115.168.82.165
primary DNS address 202.96.128.86 //获取的 DNS 服务器地址
secondary DNS address 202.96.134.133

```

3) 执行以下命令以便在 `resolv.conf` 文件中添加 DNS 服务器地址；请根据模块成功拨号后的串口信息来确定 DNS 服务器地址；

- `root@sbc9000:~# echo nameserver 202.96.128.86 > /etc/resolv.conf`

4) 执行以下命令测试 MC2716 模块是否正常连接至网络；

- `root@sbc9000:~# ping www.baidu.com`


终端窗口显示信息如下；

表 6-20 ping 信息

```
PING www.baidu.com (220.181.6.19): 56 data bytes
64 bytes from 220.181.6.19: seq=0 ttl=56 time=91.491 ms
64 bytes from 220.181.6.19: seq=1 ttl=56 time=101.196 ms
64 bytes from 220.181.6.19: seq=2 ttl=56 time=95.459 ms
64 bytes from 220.181.6.19: seq=3 ttl=56 time=96.893 ms

--- www.baidu.com ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 91.491/96.259/101.196 ms
```

注意:

 当 Mini-pcie 接口正在使用时，usb-hub 的第一个接口将被禁用。

6.14 PCI-E 测试

6.14.1 测试一

将一个 PCI-E 转 USB 模块插入 SBC9000 的 PCI-E 插槽并连接电源，然后将一个 U 盘插入 PCI-E 转 USB 模块的 USB 接口。系统将会自动识别 U 盘。

6.14.2 测试二

将一块 AR5B93 无线网卡（需另行购买）插入 SBC9000 的 PCI-E 插槽并连接电源启动系统。

1) 执行以下命令来加载该模块；

- `root@sb9000:~# wpa_passphrase GK-TIOP-TEST`
- `Embest8877 >/etc/wpa_supplicant.conf`
- `root@sb9000:~# wpa_supplicant -B -P /var/run/wpa_supplicant.wlan0.pid -i wlan0 -c /etc/wpa_supplicant.conf -D wext`

表 6-21 加载模块

```
P /var/run/wpa_supplicant.wlan0.pid -i wlan0
Successfully initialized wpa_supplicant
rfkill: Cannot open RFKILL control device
IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
```

```

root@sbc9000:~# wlan0: authenticate with 94:0c:6d:17:0a:bc
wlan0: send auth to 94:0c:6d:17:0a:bc (try 1/3)
wlan0: authenticated
ath9k 0000:01:00.0 wlan0: disabling HT as WMM/QoS is not supported by
the AP
ath9k 0000:01:00.0 wlan0: disabling VHT as WMM/QoS is not supported by
the AP
wlan0: associate with 94:0c:6d:17:0a:bc (try 1/3)
wlan0: RX AssocResp from 94:0c:6d:17:0a:bc (capab=0x431 status=0
aid=4)
IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
wlan0: associated

```

2) 执行以下命令来自动获取动态 IP 地址;

- `root@DevKit8600:~# udhcpc -i wlan0`

终端窗口显示信息如下;

表 6-22 获取 IP 地址

```

udhcpc (v1.21.1) started
Sending discover...
Sending select for 192.168.8.169...
Lease of 192.168.8.169 obtained, lease time 86400
/etc/udhcpc.d/50default: Adding DNS 192.192.192.248
/etc/udhcpc.d/50default: Adding DNS 202.96.134.133
/etc/udhcpc.d/50default: Adding DNS 202.96.128.86

```

3) 执行以下命令来测试网络是否正常;

- `root@sbc9000:~# ping www.baidu.com`

终端窗口显示信息如下;


表 6-23 ping 信息

```

PING www.baidu.com (180.97.33.107): 56 data bytes
64 bytes from 180.97.33.107: seq=0 ttl=54 time=35.663 ms
64 bytes from 180.97.33.107: seq=1 ttl=54 time=35.835 ms
64 bytes from 180.97.33.107: seq=2 ttl=54 time=38.501 ms
64 bytes from 180.97.33.107: seq=3 ttl=54 time=35.776 ms
^C
--- www.baidu.com ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 35.663/36.443/38.501 ms

```

注意:

 PCI-E 接口不支持热插拔。

6.15 背光测试

背光的亮度设置范围为（0—7）。7 表示亮度最高，0 表示关闭背光。

6.15.1 LCD 背光测试

1) 执行以下命令来查看当前背光的亮度值;

- `root@sbc9000:~# cat /sys/class/backlight/backlight-lcd.27/brightness`

2) 执行以下命令来关闭 LCD 屏幕背光;

- `root@sbc9000:~# echo 0 > /sys/class/backlight/backlight-lcd.27/brightness`

3) 执行以下命令将屏幕亮度设置为最大;

- `root@sbc9000:~# echo 7 > /sys/class/backlight/backlight-lcd.27/brightness`

6.15.2 电容屏背光测试

1) 执行以下命令来查看当前背光的亮度值;

- `root@sbc9000:~# cat /sys/class/backlight/backlight-ldb.28/brightness`

2) 执行以下命令来关闭电容屏幕背光;

- `root@sbc9000:~# echo 0 > /sys/class/backlight/backlight-ldb.28/brightness`

3) 执行以下命令来将屏幕亮度设置为最大;

- `root@sbc9000:~# echo 7 > /sys/class/backlight/backlight-ldb.28/brightness`

6.16 SATA 测试

1) 将一个 SATA 接口的硬盘（需另行购买）连接到 SBC9000 的 SATA 接口，并连接硬盘电源，然后在系统启动后执行以下命令来加载 SATA 模块驱动;

- `root@ SBC9000 ~$ modprobe ahci_imx`

- 2) 模块驱动成功加载后，系统会自动挂载 SATA 硬盘；请执行以下命令来查看 SATA 硬盘的挂载目录；

- `root@ SBC9000 ~$ df -h`

表 6-24 目录列表

Filesystem	Size	Used	Available	Use%	Mounted on
/dev/root	3.4G	484.0M	2.8G	14%	/
devtmpfs	340.0M	4.0K	340.0M	0%	/dev
tmpfs	500.1M	192.0K	499.9M	0%	/run
tmpfs	500.1M	52.0K	500.1M	0%	/var/volatile
/dev/mmcblk0p1	19.8M	5.8M	14.0M	29%	
/media/mmcblk0p1					
/dev/sda5	74.5G	480.8M	74G	1%	/media/sda5

以上信息显示 SATA 硬盘挂载目录为/media/sda5。

- 3) 执行以下命令来查看硬盘的内容；

- `root@ SBC9000 ~$ ls /media/sda5`

表 6-25 硬盘内容

can.txt

附录 1 安装 Ubuntu Linux 系统

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 的设置和 Ubuntu Linux 系统的安装过程。

设置 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；

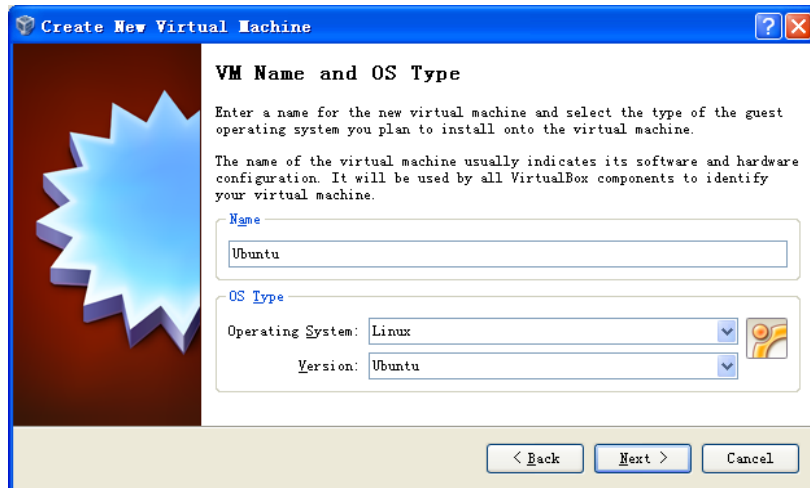


图 2 虚拟机名称和操作系统类型

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；

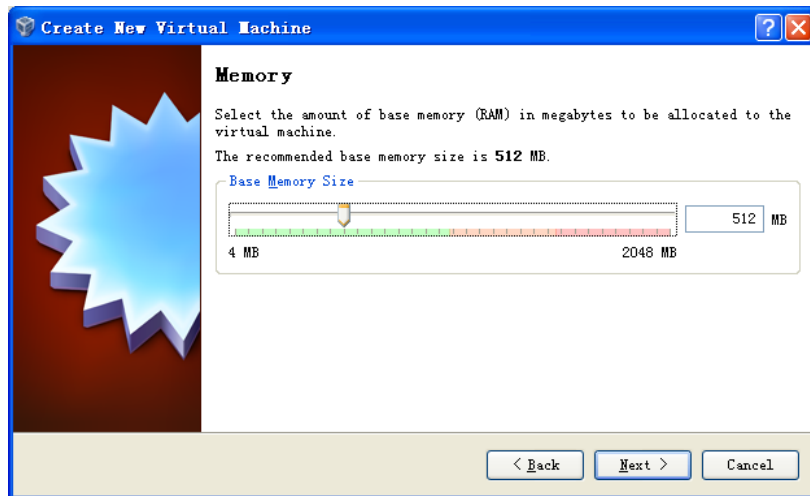


图 3 内存分配窗口

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；

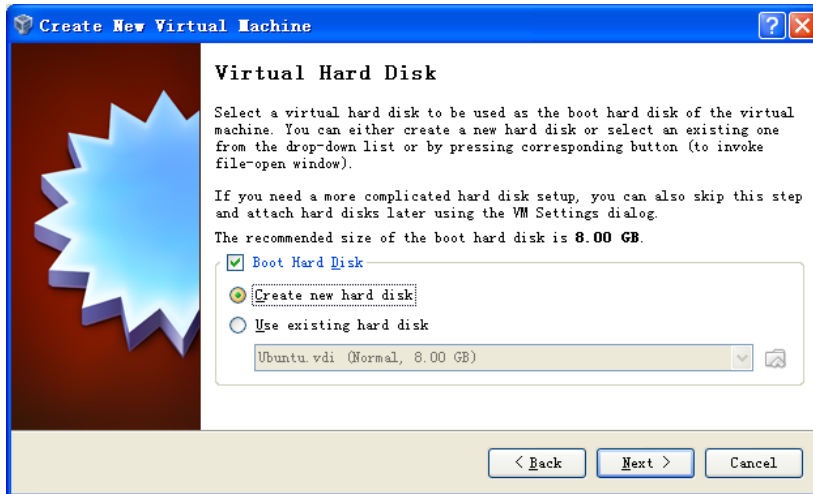


图 4 创建新的虚拟硬盘窗口

5) 在下方虚拟硬盘创建向导窗口中单击 **Next** 按钮；

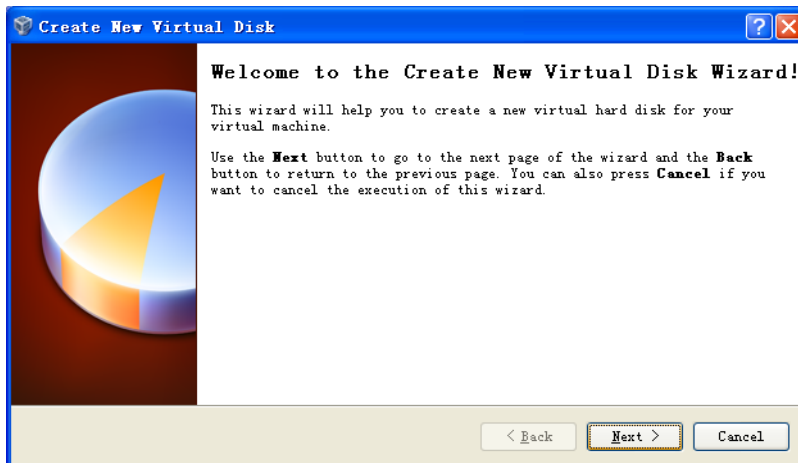


图 5 虚拟硬盘创建向导

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

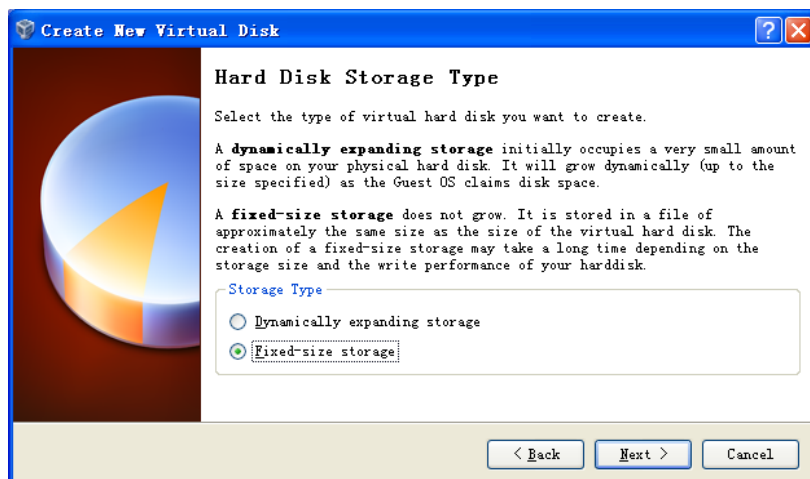


图 6 选择第二个选项

- 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；

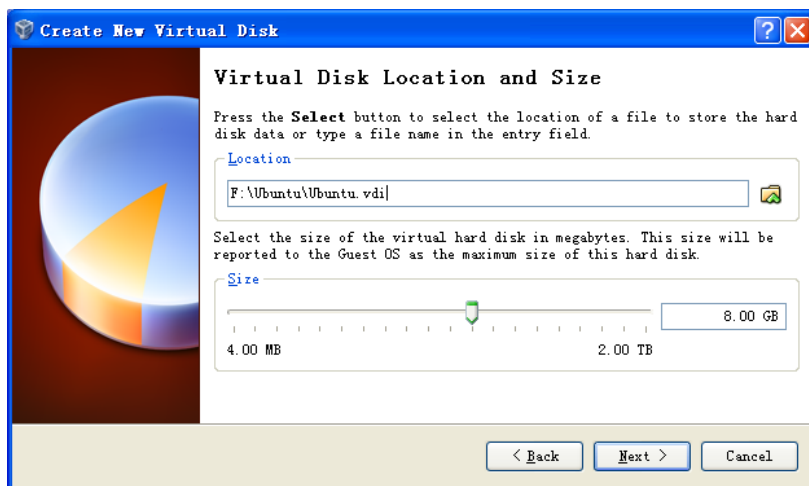


图 7 虚拟硬盘设置窗口

- 8) 在下方的虚拟硬盘信息窗口中单击 **Finish** 按钮；

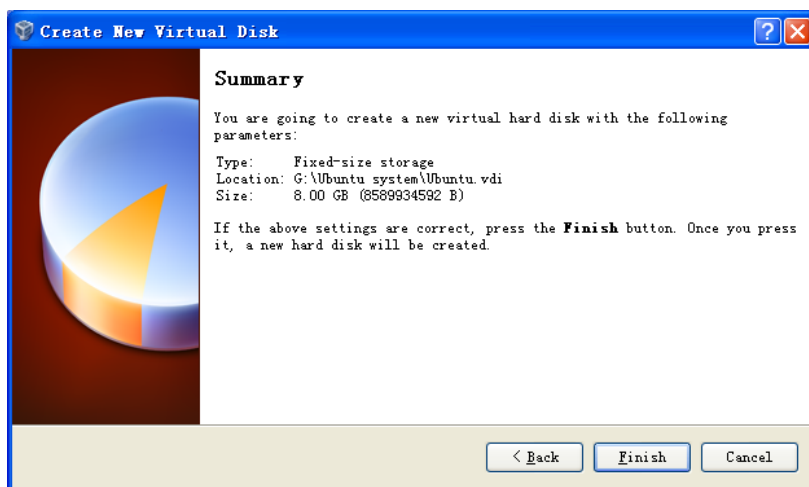


图 8 虚拟硬盘信息

9) PC 开始创建虚拟硬盘驱动器;

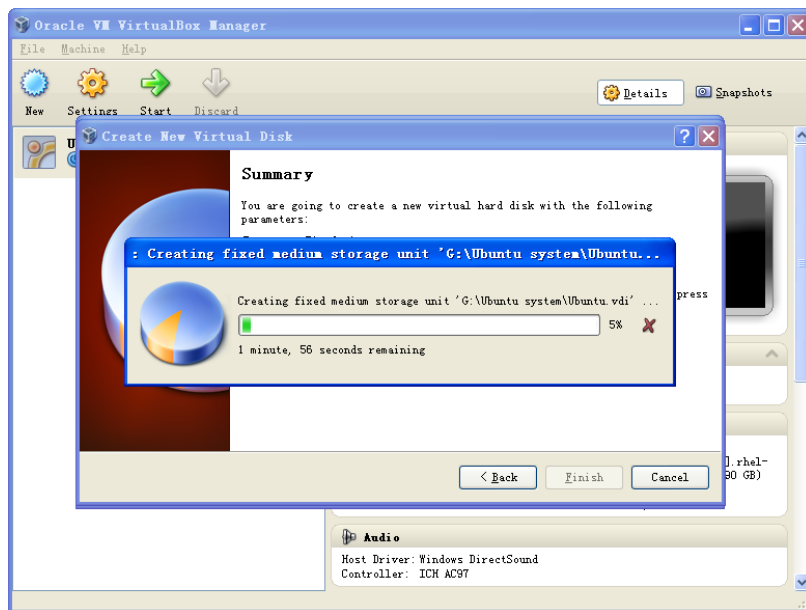


图 9 创建虚拟硬盘驱动器

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;

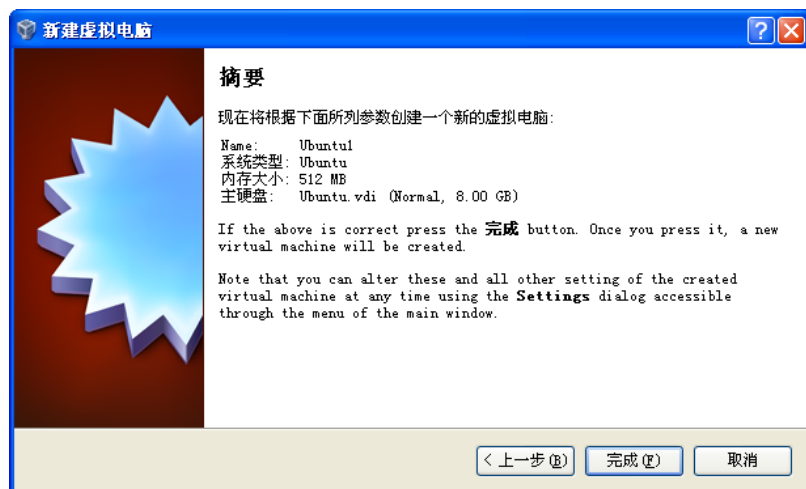


图 10 虚拟机配置完成

安装 Ubuntu Linux 系统

虚拟机设置完成后，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：

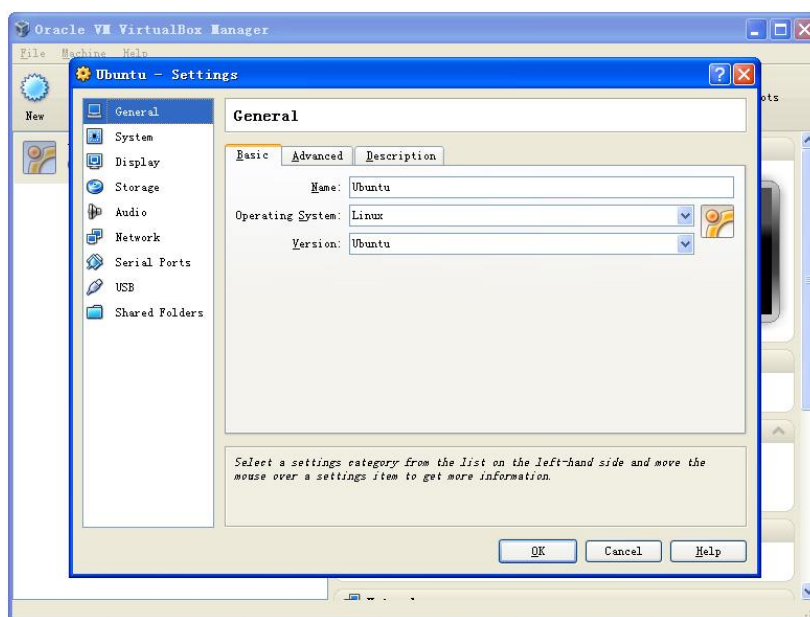


图 11 虚拟机设置

- 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：

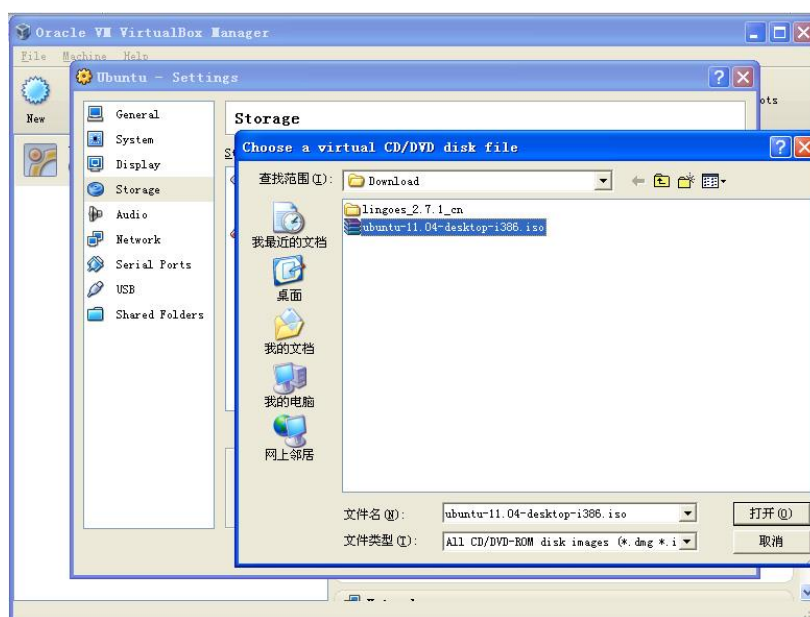


图 12 指定 Ubuntu 映像位置

- 3) 选中刚才添加的映像并单击 **OK** 按钮，如下图所示：

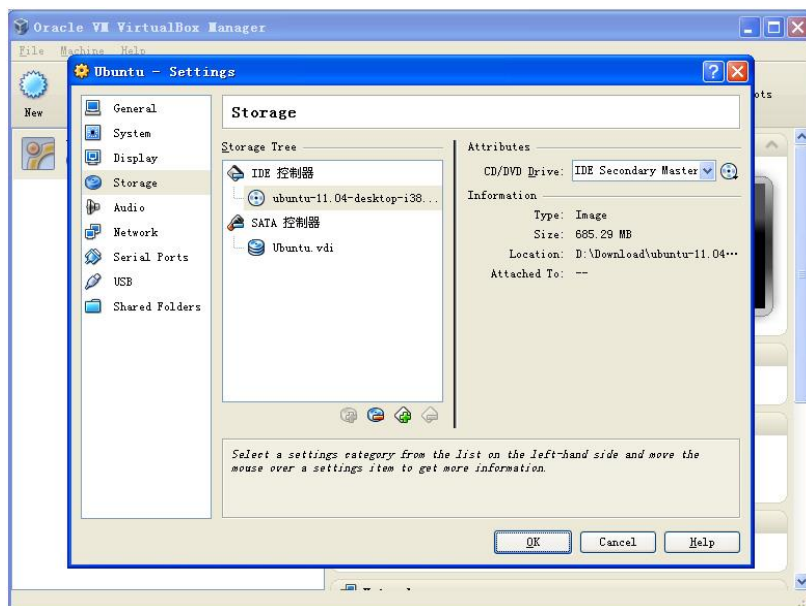


图 13 选中 ISO 映像

- 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：

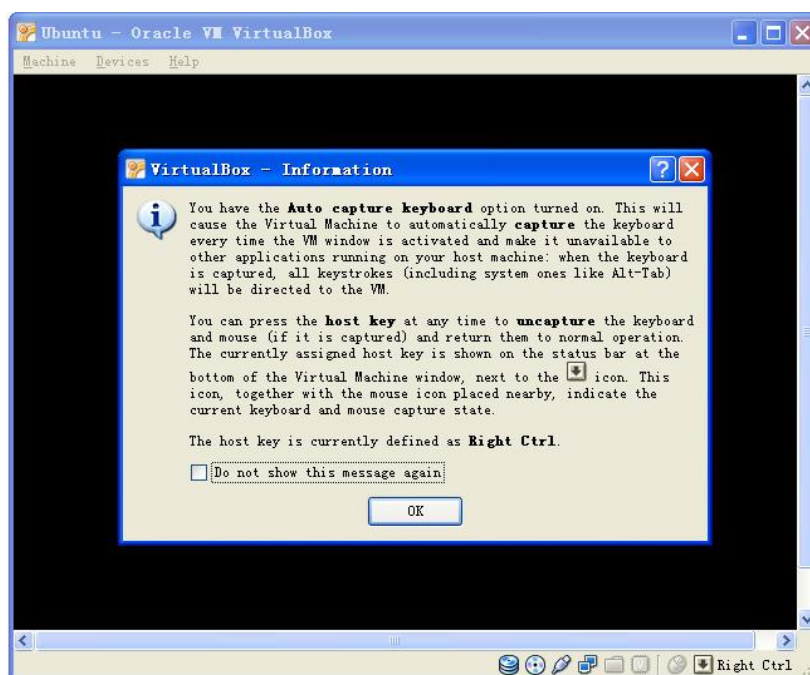


图 14 Ubuntu 初始化窗口

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：

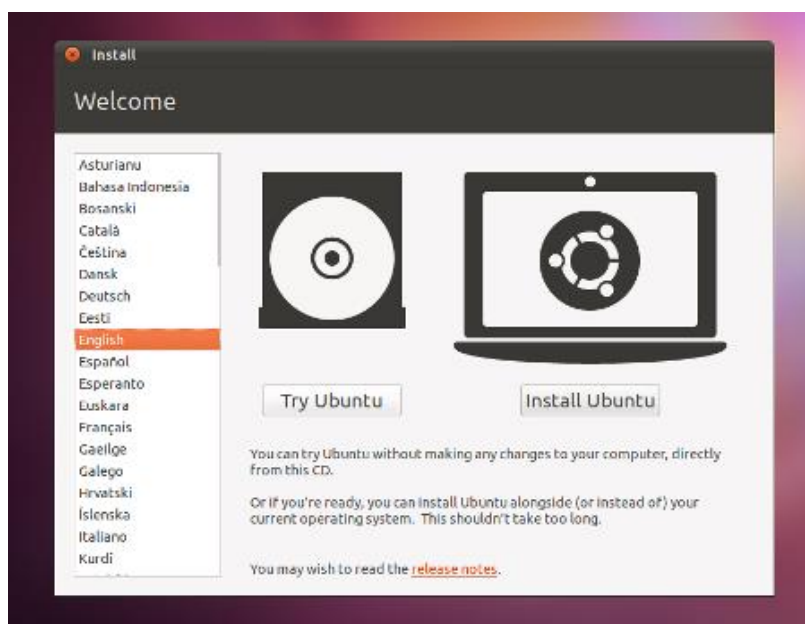


图 15 Ubuntu 安装窗口

6) 单击 **Forward** 按钮继续安装，如下图所示：



图 16 安装前的信息

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

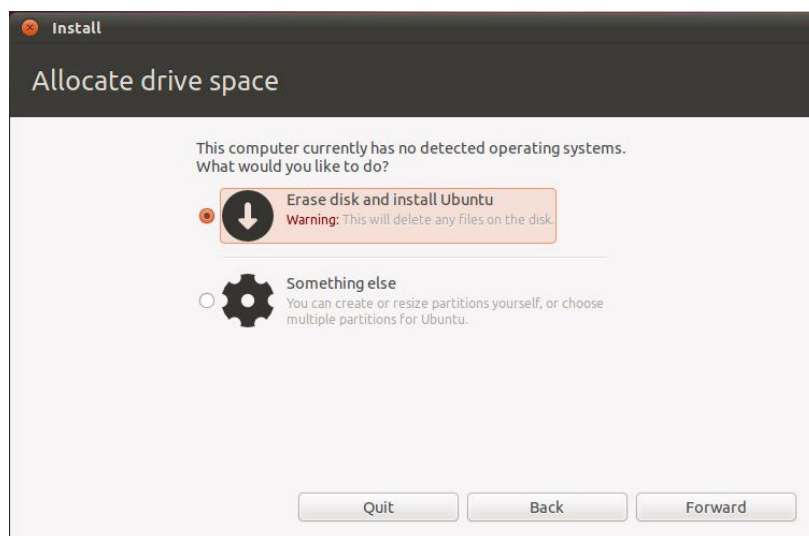


图 17 Ubuntu 安装选项

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;

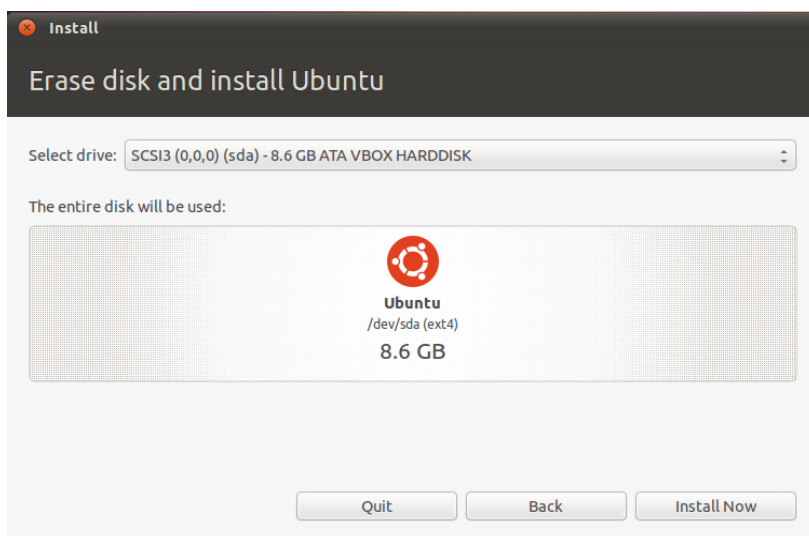


图 18 安装确认窗口

- 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；

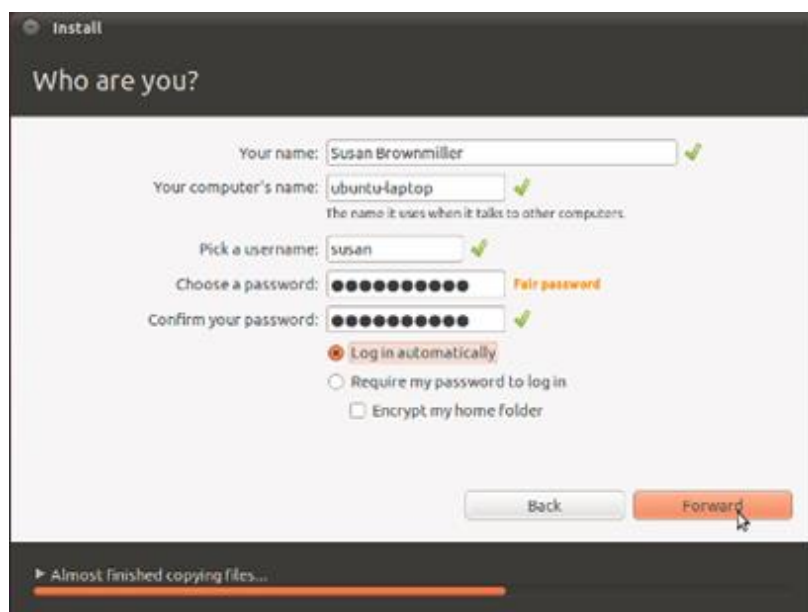


图 19 指定用户名和密码

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；

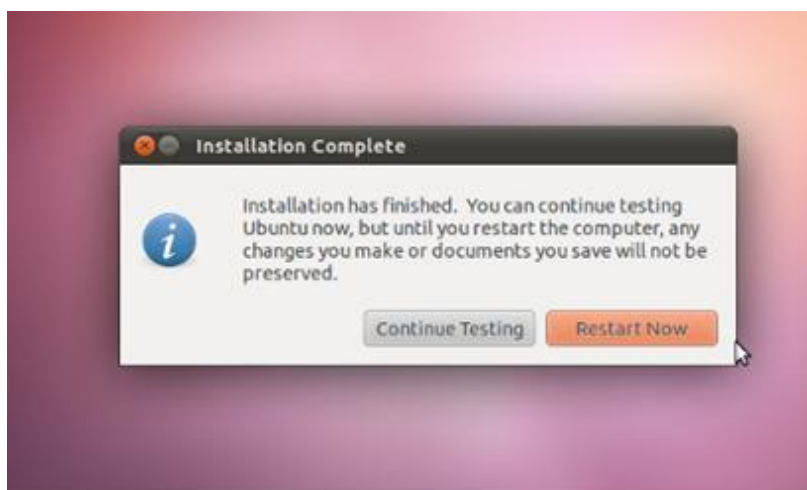


图 20 重新启动系统

- 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 13 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：

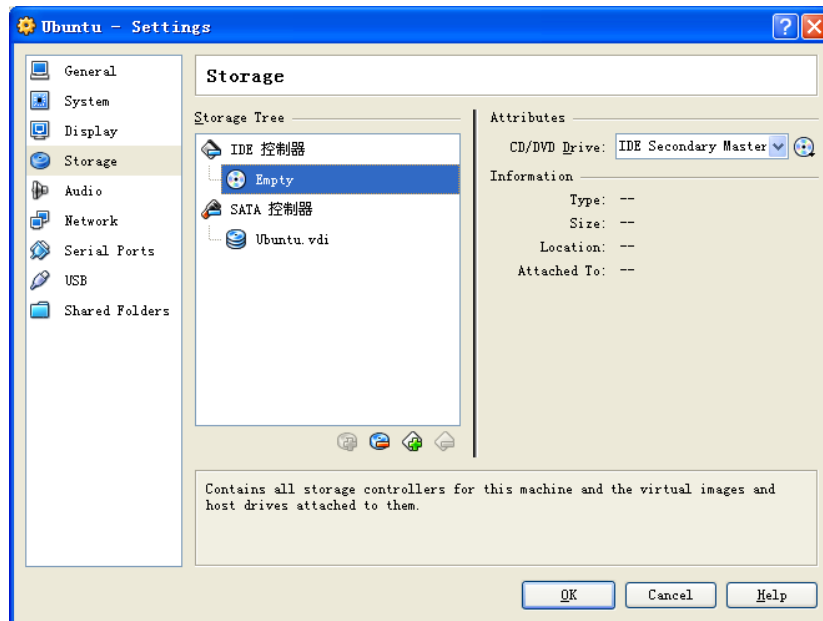


图 21 弹出 ISO 映像

附录 2 安装 Linux USB Ethernet/RNDIS Gadget 驱动

- 1) 从[英蓓特官方网站 SBC9000 产品页面](#)的“资源下载”标签下载“Linux 相关工具”。
- 2) 通过一根 Mini USB 线缆连接 SBC9000 的 USB OTG 接口和 PC 的 USB 接口；如果您未曾安装 Linux USB Ethernet/RNDIS Gadget 驱动，PC 任务栏会出现“正在安装设备驱动程序软件”的气泡提示，如下图所示；

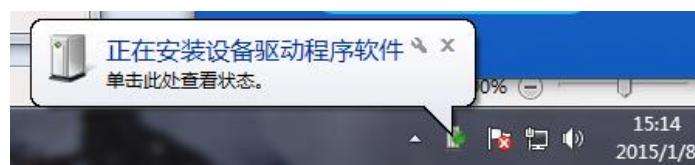


图 22 搜索驱动

- 3) 单击该气泡进入下图所示的驱动安装界面，然后单击“跳过从 Windows Update 获得驱动程序软件”；

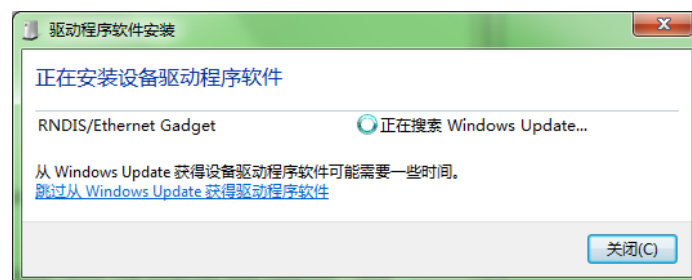


图 23 自动安装驱动

- 4) 在以下窗口单击“是”；

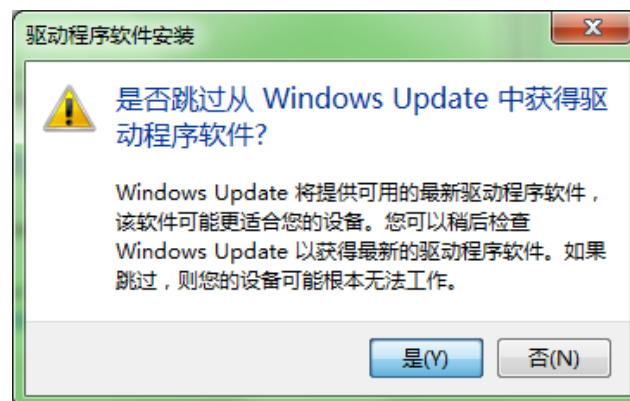


图 24 跳过自动安装

- 5) 在以下窗口中单击“浏览计算机以查找驱动程序软件”；

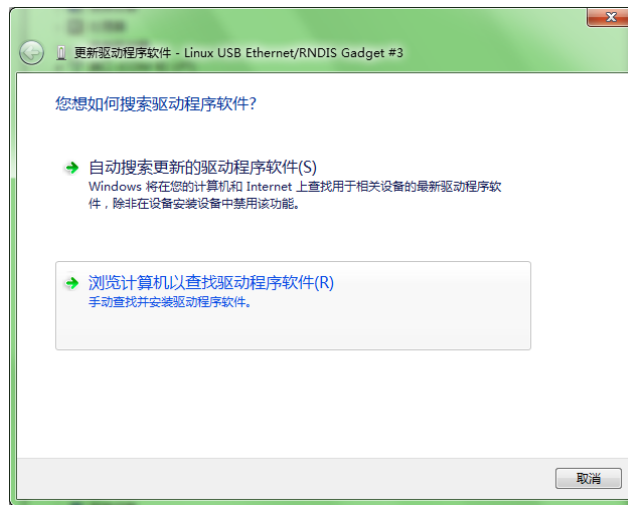


图 25 手动查找驱动

- 6) 在以下窗口中单击“浏览”按钮,指定刚才下载的 Linux 相关工具中的“tools\usb driver”目录,然后单击“下一步”开始安装;

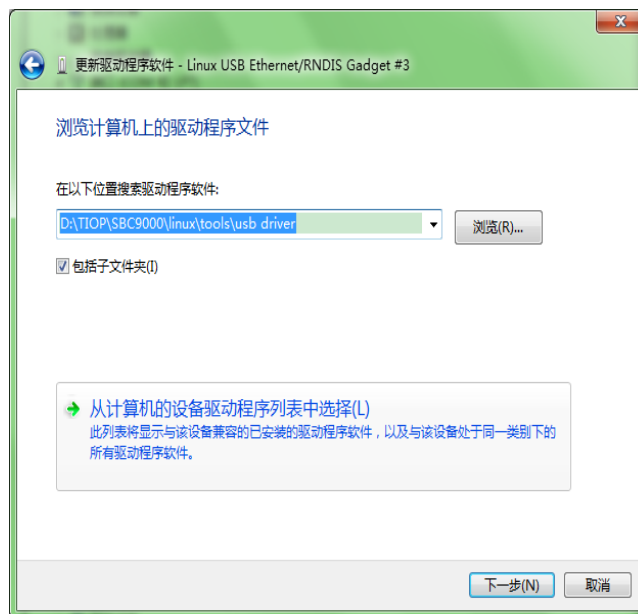


图 26 指定路径

- 7) 安装完成后会出现以下窗口,表示驱动已经成功安装。



图 27 安装完成

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：
 - 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；
 - 未按用户手册操作导致产品损坏的；
 - 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力力导致的产品外观及功能损坏；
 - 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；

- 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；
 - 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；
 - 非经授权渠道购得此产品者。
 - 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：

 英蓓特科技公司对于未经本公司许可私自寄回的产品不承担任何责任。

联系方式

技术支持

电话：+86-755-25635626-872/875/897

Email: support@embest-tech.com

销售信息

电话：+86-755-25635626-860/861/862

传真：+86-755-25616057

Email: chinasales@embest-tech.com

公司信息

网站: <http://www.embest-tech.cn>

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