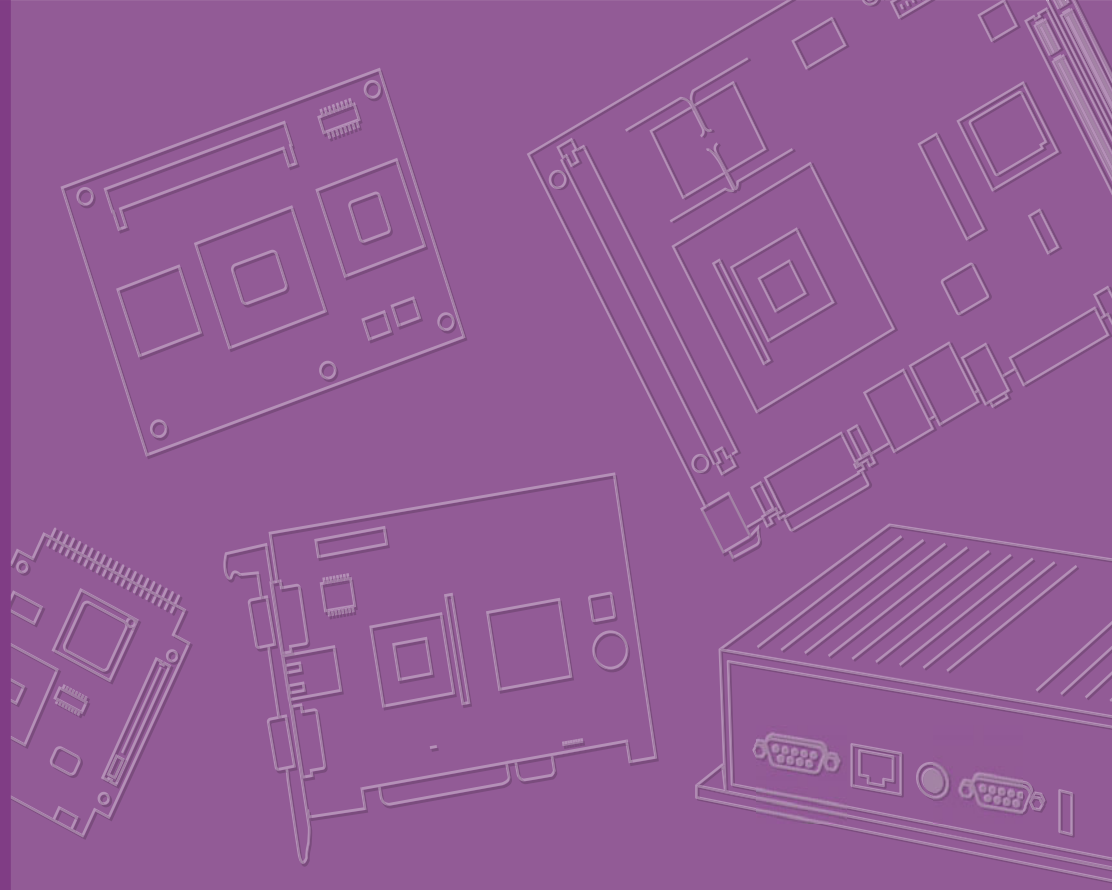


用户手册



UBC-330

微型计算机

ADVANTECH

Enabling an Intelligent Planet

版权声明

随附本产品发行的文件为研华公司 2015 年版权所有，并保留相关权利。针对本手册中相关产品的说明，研华公司保留随时变更的权利，恕不另行通知。未经研华公司书面许可，本手册所有内容不得通过任何途径以任何形式复制、翻印、翻译或者传输。本手册以提供正确、可靠的信息为出发点。但是研华公司对于本手册的使用结果，或者因使用本手册而导致其它第三方的权益受损，概不负责。

认可声明

ARM 为 ARM Corporation 的商标。

TI 为 TI Corporation 的商标。

所有其它产品名或商标均为各自所属方的财产。

有关该产品以及其它研华的产品更多信息，请访问研华公司网站：

<http://www.Advantech.com/>

产品质量保证（两年）

从购买之日起，研华为原购买商提供两年的产品质量保证。但对那些未经授权的维修人员维修过的产品不予提供质量保证。研华对于不正确的使用、灾难、错误安装产生的问题有免责权利。

如果研华产品出现故障，在质保期内我们提供免费维修或更换服务。对于出保产品，我们将会酌情收取材料费、人工服务费用。请联系相关销售人员了解详细情况。

如果您认为您购买的产品出现了故障，请遵循以下步骤：

1. 收集您所遇到的问题信息（例如，CPU 主频、使用的研华产品及其它软件、硬件等）。请注意屏幕上出现的任何不正常信息显示。
2. 打电话给您的供货商，描述故障问题。请借助手册，产品和任何有帮助的信息。
3. 如果您的产品被诊断发生故障，请从您的供货商那里获得 RMA (Return Material Authorization) 序列号。这可以让我们尽快地进行故障产品的回收。
4. 请仔细地包装故障产品，并在包装中附上完整的售后服务卡片和购买日期证明（如销售发票）。我们对无法提供购买日期证明的产品不提供质量保证服务。
5. 把相关的 RMA 序列号写在外包装上，并将其运送给销售人员。

料号：200K033010

中国印刷

第一版

2015 年 4 月

符合性声明

FCC B 级

根据 FCC 规则第 15 条，本设备已经过检测并被判定符合 B 级数字设备标准。这些限制旨在为居住环境下的系统操作提供合理保护，使其免受有害干扰。本设备会产生、使用 and 发射无线电频率能量。如果没有按照手册说明正确安装和使用，可能对无线电通讯造成有害干扰。但即使按照手册说明进行安装和使用，也并不能保证不会产生干扰。若本设备会对无线电或电视信号接收产生有害干扰，用户可通过开、关设备进行确认。当本设备产生有害干扰时，用户可采取下面的措施来解决干扰问题：

- 调整接收天线的方向或位置
- 增大本设备与接收器之间的距离
- 将本设备的电源接头插在与接收器使用不同电路的电源插座
- 若需技术支持，请咨询经销商或经验丰富的无线电 / 电视技术人员

技术支持与服务

1. 有关该产品的最新信息，请访问研华公司的网站：
<http://support.advantech.com.cn>
2. 用户若需技术支持，请与当地分销商、销售代表或研华客服中心联系。进行技术咨询前，用户须将下面各项产品信息收集完整：
 - 产品名称及序列号
 - 外围附加设备的描述
 - 用户软件的描述（操作系统、版本、应用软件等）
 - 产品所出现问题的完整描述
 - 每条错误信息的完整内容

警告与注意

警告！ 在操作过程中，用户须特别注意该手册中的警告信息，以免造成人身伤害。



注意！ 该手册中的注意信息可帮助用户避免损坏硬件或丢失数据，例如：
如果电池放置不正确，将有爆炸的危险。因此，只可以使用制造商推荐的同一种或者同等型号的电池进行替换。请按照制造商的指示处理旧电池。



注！ 此项提供其它额外信息。



包装清单

安装系统之前，用户需确认包装中含有本设备以及下面所列各项，并确认设备完好。若有任何不符，请立即与经销商联系。

- 1 x UBC-330 微型计算机
- 1 x (2 x 20)PIN 连接头
- 螺丝配件包

订购信息

型号名称	说明
UBC-330NS-JLA1E	UBC-330 TI Sitara AM3352 Cortex A8 1GHz
1652006830-01	20 x 2P 2.54mm 180D 0156-1A40
193A200001	POST F=#4-40x4L M=#4-40x7.1L B=4.7 H=4.9 Cu Ni
1931041020	S/S M4*10L D=7.2 H=2.0 + ST BLK

可选附件

配件料号	配件描述
1757003553	Adapter 100~240V 36W 12V 3A w/o PFC 9NA0361603
SQF-ISDS1-2G-86E	SQF SD C6 SLC 2G, 1CH (-40~85° C)
170203183C	Power Code 3P Europe (WS-010+WS-083) 183cm
1700023307-01	A cable DC JACK/Plug-in 1*2P-5.0 10cm RSB-4220

安全指示

1. 请仔细阅读此安全操作说明。
2. 请妥善保存此用户手册供日后参考。
3. 用湿抹布清洗设备前，请从插座拔下电源线。请不要使用液体或去污喷雾剂清洗设备。
4. 对于使用电源线的设备，设备周围必须有容易接触到的电源插座。
5. 请不要在潮湿环境中使用设备。
6. 请在安装前确保设备放置在可靠的平面上，意外跌落可能会导致设备损坏。
7. 设备外壳的开口是用于空气对流，从而防止设备过热。**请不要覆盖这些开口。**
8. 当您连接设备到电源插座上前，请确认电源插座的电压是否符合要求。
9. 请将电源线布置在人们不易绊到的位置，并不要在电源线上覆盖任何杂物。
10. 请注意设备上的所有警告标识。
11. 如果长时间不使用设备，请将其同电源插座断开，避免设备被超标的电压波动损坏。
12. 请不要让任何液体流入通风口，以免引起火灾或者短路。
13. 请不要自行打开设备。为了确保您的安全，请由经过认证的工程师来打开设备。
14. 如遇下列情况，请由专业人员来维修：
 - 电源线或者插头损坏；
 - 设备内部有液体流入；
 - 设备曾暴露在过于潮湿的环境中使用；
 - 设备无法正常工作，或您无法通过用户手册来使其正常工作；
 - 设备跌落或者损坏；
 - 设备有明显的外观破损。
15. 请不要把设备放置在超出我们建议的温度范围的环境，即 0 ~ 55° C，否则可能会损坏设备。
16. **注意：**计算机配置了由电池供电的实时时钟电路，如果电池更换不正确，将有爆炸的危险。因此，只可以使用制造商推荐的同一种或者同等型号的电池进行替换。请按照制造商的指示处理旧电池。

根据 IEC 704-1: 1982 的规定，操作员所在位置的声压级不可高于 70dB(A)。

免责声明：该安全指示符合 IEC 704-1 的要求。研华公司对其内容的准确性不承担任何法律责任。

目录

第 1 章	概述	1
1.1	简介.....	2
1.2	产品特性.....	2
1.2.1	主要特性.....	2
1.2.2	一般特性.....	2
1.2.3	功耗.....	2
1.3	机械规格.....	3
1.3.1	重量.....	3
1.3.2	尺寸.....	3
1.4	电源要求.....	4
1.4.1	系统电源.....	4
1.4.2	RTC 电池	4
1.5	环境规格.....	4
1.5.1	工作温度.....	4
1.5.2	相对湿度.....	4
1.5.3	存储温度.....	4
1.5.4	EMC.....	4
第 2 章	硬件信息	5
2.1	简介.....	6
2.2	UBC-330 I/O 接口位置	6
	图 2.1: UBC-330 前视图	6
	图 2.2: UBC-330 LED 指示灯).....	6
	图 2.3: UBC-330 左视图	6
2.3	UBC-330 I/O 接口	7
2.3.1	电源输入接口.....	7
	图 2.4: 电源输入接口.....	7
2.3.2	重启按钮.....	7
	图 2.5: 重启按钮.....	7
2.3.3	以太网端口	7
	图 2.6: 以太网端口.....	7
	表 2.1: 以太网端口针脚定义.....	7
2.3.4	USB 接口	8
	图 2.7: USB 接口	8
	表 2.2: USB 接口针脚定义	8
2.3.5	SD 卡接口	9
	图 2.8: SD 卡接口	9
	表 2.3: SD 卡针脚定义	9
2.3.6	COM/GPIO/CAN etc... 端口	10
	图 2.9: COM/GPIO/CAN etc... 端口	10
	表 2.4: 其它 I/O 接口针脚定.....	10
	图 2.10: 2X20 pin Connector.....	11
2.3.7	Watch Dog timer.....	12
2.4	UBC-330 快速入门指南	13
2.4.1	调试口连接.....	13
2.4.2	调试口设置.....	13
2.5	测试工具.....	14
2.5.1	eMMC 测试.....	14
2.5.2	USB 测试	15
2.5.3	SD 卡测试	15
2.5.4	SPI 测试	16
2.5.5	I2C 测试.....	17
2.5.6	CAN 测试.....	17

2.5.7	GPIO 测试.....	18
2.5.8	网口测试.....	18
2.5.9	UART 测试.....	20

第 3 章 软件信息 21

3.1	介绍.....	22
3.2	软件包.....	22
3.2.1	Image 映像文件包.....	22
3.2.2	BSP 源码包.....	22
	图 3.1: 源码包目录结构.....	22
	图 3.2: image\rootfs.....	24
3.3	建立开发环境.....	25
3.4	编译源码.....	26
3.4.1	编译 u-boot.....	26
3.4.2	编译 Linux 内核.....	26
3.4.3	编译日志.....	26
3.5	内核源码修改.....	27
	图 3.3: Linux 内核配置.....	27
	图 3.4: 选择 Seiko Instruments S-35390A.....	28
3.6	创建 Linux 系统.....	29
3.6.1	存储介质 (eMMC/SD 卡).....	29
3.6.2	创建 SD 卡 Linux 系统.....	29
3.6.3	从 eMMC flash 启动 Linux.....	30
3.7	调试信息.....	30
	图 3.5: 超级终端设定.....	30
3.8	Linux 系统配置和使用.....	31
3.8.1	服务配置.....	31
3.8.2	网络配置.....	33
3.8.3	日期 / 时间配置.....	33
3.8.4	系统版本.....	34
3.8.5	添加 / 移除开机启动项.....	34
3.8.6	在线安装软件.....	34
3.9	开发向导.....	35
3.9.1	C/C++ 应用程序开发.....	35
3.9.2	示例程序源码.....	36

第 1 章

概述

本章介绍 UBC-330 的基本信息。

1.1 简介

UBC-330 是基于 TI Sitara AM3352 Cortex-A8 1GHz 处理器的微型计算机。它提供了两个千兆以太网，6 个串口，8 个 GPIO 等等外扩接口以应用于工业控制设备。其中 UART & GPIO 还分别被赋予了 ESD 和隔离保护功能，它可以保护系统免于不稳定因素的伤害。另外，UBC-330 还可以支持多类型电压输入。总体来讲，在自动化控制领域，UBC-330 是一个理想的解决方案。

1.2 产品特性

1.2.1 主要特性

- **核心处理器：TI Sitara AM3352**
 - TI Sitara AM3352 Cortex A8 Single core 1GHz
- **系统内存**
 - DDR3 800 MHz
 - 容量：on board 512MB
- **千兆网络**
 - 网络芯片：Realtek 8211
 - 2 x10/100/1000 Mbps

1.2.2 一般特性

- 内核：V 3.2.0
- 闪存：板载 4GB EMMC
- 1 个 USB host 接口
- 2 个千兆网口
- 1 个 SD 卡插槽
- 1 个重启键
- 1 个 40PIN 端子台，用于扩展 8 x GPIO / 1 x CAN / 6 x COM 等功能
- 支持 HW WDT (默认 WDT 时间为 60s，关机后间隔 1s 开机)

1.2.3 功耗

- 闲置：2.8W

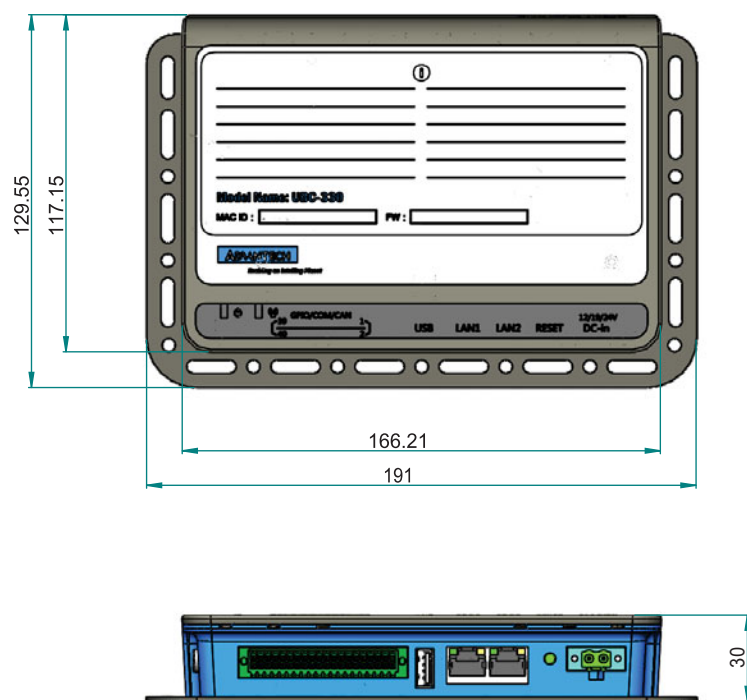
1.3 机械规格

1.3.1 重量

640g（毛重）

1.3.2 尺寸

- 129.55 x 191 mm（带金属板）
- 117.15 x 166.21 mm（不带金属板）



1.4 电源要求

1.4.1 系统电源

电源输入：DC 12 / 19 / 24 V

1.4.2 RTC 电池

电源输入：3V / 210mAh

1.5 环境规格

1.5.1 工作温度

0~55° C (32~131° F)

1.5.2 相对湿度

95% @ 40° C (非凝结)

1.5.3 存储温度

-20~60° C (-4~104° F)

1.5.4 EMC

CE/FCC/CCC/B 级

第 2 章

硬件信息

本章介绍硬件特性和部分 I/O 接口信息。

2.1 简介

以下内容介绍了 UBC-330 的外部接口和针脚定义

2.2 UBC-330 I/O 接口位置

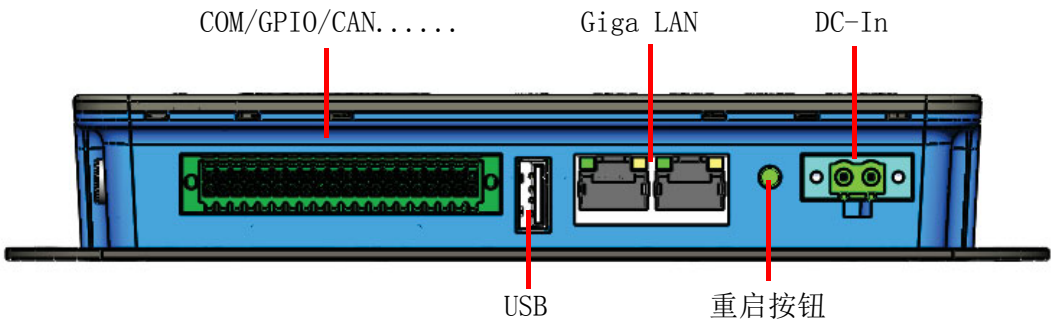


图 2.1：UBC-330 前视图

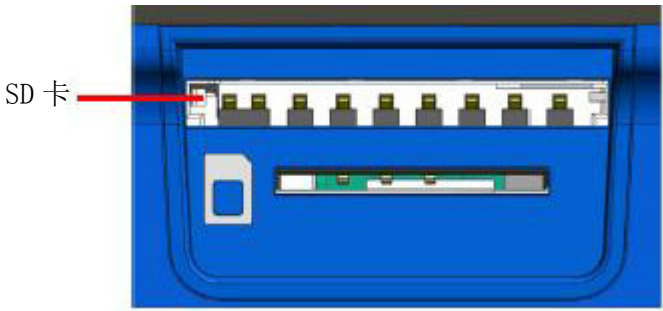


图 2.2：UBC-330 LED 指示灯)



图 2.3：UBC-330 左视图

2.3 UBC-330 I/O 接口

2.3.1 电源输入接口

UBC-330 带有一个DC 插孔，搭配“凤凰端子”，支持 12/19/24 V DC 的外部电源输入。

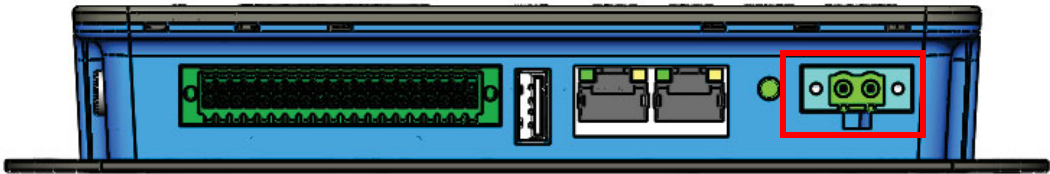


图 2.4：电源输入接口

2.3.2 重启按钮

UBC-330 前部有一个“内嵌防误操作式”重启按钮，按下该按钮将激活硬件重启功能。

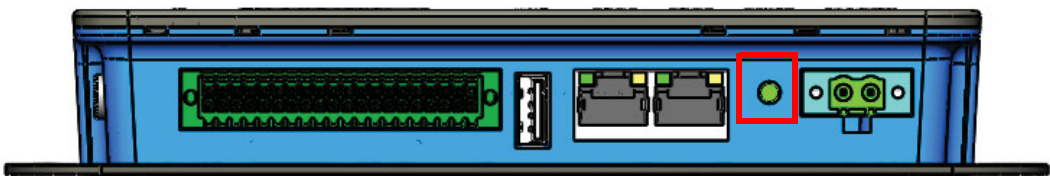


图 2.5：重启按钮

2.3.3 以太网端口

UBC-330 提供了两个 RJ45 LAN 接口。该接口完全符合 IEEE 802.3u 10/100/1000 Base-T CSMA/CD 标准。以太网端口提供了标准的 RJ-45 接口，其前部带 LED 指示灯，用于显示传输 / 连接状态。

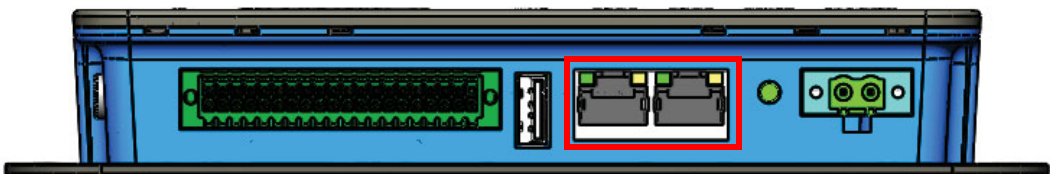


图 2.6：以太网端口

表 2.1：以太网端口针脚定义	
Pin	Description
A1	MDI20+
A2	MDI20-
A3	MDI21+
A4	MDI21-
A5	GND
A6	GND
A7	MDI22+
A8	MDI22-
A9	MDI23+
A10	MDI23-

表 2.1：以太网端口针脚定义	
A11	LAN2_100_LINK
A12	LAN2_1000_LINK
A13	+3.3V
A14	LAN2_ACT
B1	MDI10+
B2	MDI10-
B3	MDI11+
B4	MDI11-
B5	GND
B6	GND
B7	MDI12+
B8	MDI12-
B9	MDI13+
B10	MDI13-
B11	LAN1_100_LINK
B12	LAN1_1000_LINK
B13	+3.3V
B14	LAN1_ACT

2.3.4 USB 接口

UBC-330 提供了一个符合 USB UHCI, Rev 2.0 标准的 USB 接口。USB 接口支持即插即用功能，使用户可以随时连接或断开设备，而无需关闭计算机。

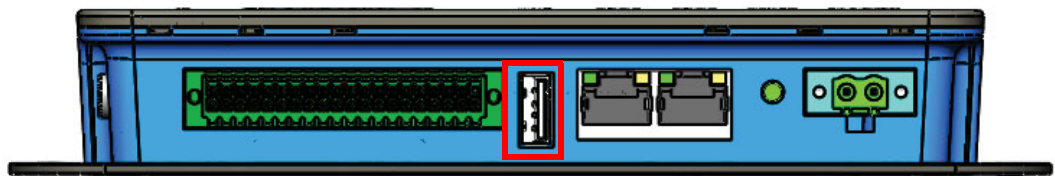


图 2.7：USB 接口

表 2.2：USB 接口针脚定义	
Pin	Description
1	+5V
2	USB Data-
3	USB Data+
4	GND

2.3.5 SD 卡接口

UBC-330 提供了一个 SD 卡插槽。用户可以轻松地插入一张 SD card。

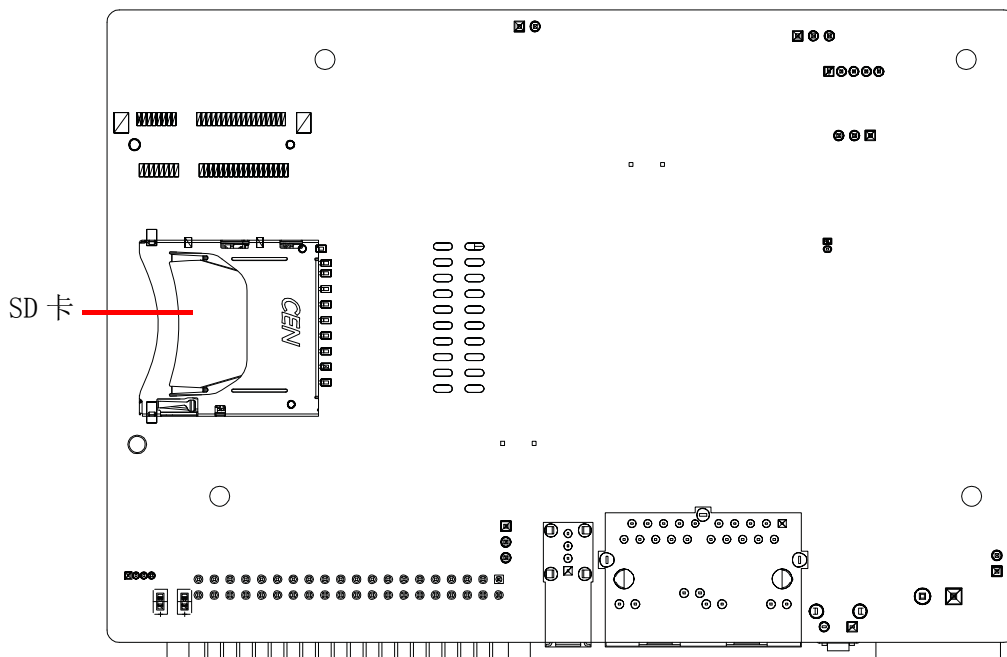


图 2.8: SD 卡接口

表 2.3: SD 卡引脚定义

Pin	Signal Name
1	DAT3
2	CMD
3	GND
4	+3.3V
5	CLK
6	GND
7	DAT0
8	DAT1
9	DAT2



2.3.6 COM/GPIO/CAN etc… 端口

UBC-330 提供一个 2 x 20PIN 的连接头，用户可以通过这个连接头实现 COM（UART） / CAN/GPIO/I2C 等功能。

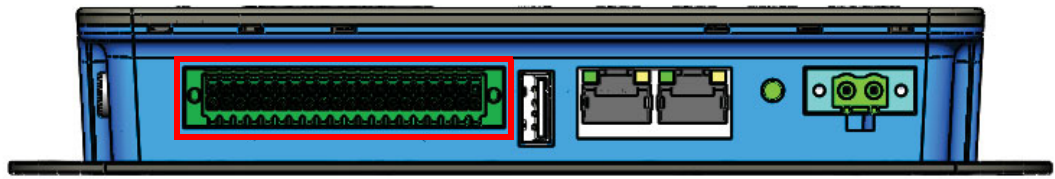


图 2.9: COM/GPIO/CAN etc… 端口

表 2.4: 其它 I/O 接口针脚定

Pin	Description
1	IDI0
2	ID00
3	IDI1
4	ID01
5	IDI2
6	ID02
7	IDI3
8	ID03
9	GND_iso
10	PCOM
11	NC
12	GND_iso
13	NC
14	NC
15	422_RXD-
16	NC
17	422_RXD+
18	I2C0_SCL
19	COM1_CTS
20	I2C0_SDA
21	COM1_TXD
22	GND
23	COM1_RTS
24	COM5_TX
25	COM1_RXD
26	COM5_RX
27	422-485_TXD+
28	CAN1_D+
29	422-485_TXD-
30	CAN1_D-
31	COM2_RX
32	COM0_TX
33	COM2_TX
34	COM0_RX

表 2.4: 其它 I/O 接口引脚定	
35	COM4_TX
36	COM3_TX
37	COM4_RX
38	COM3_RX
39	GND
40	GND

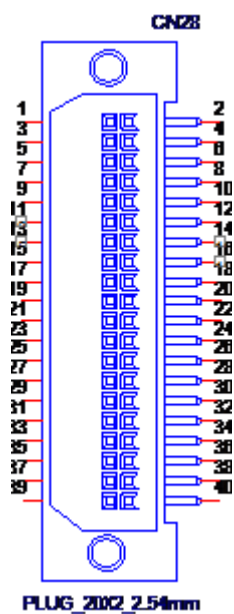
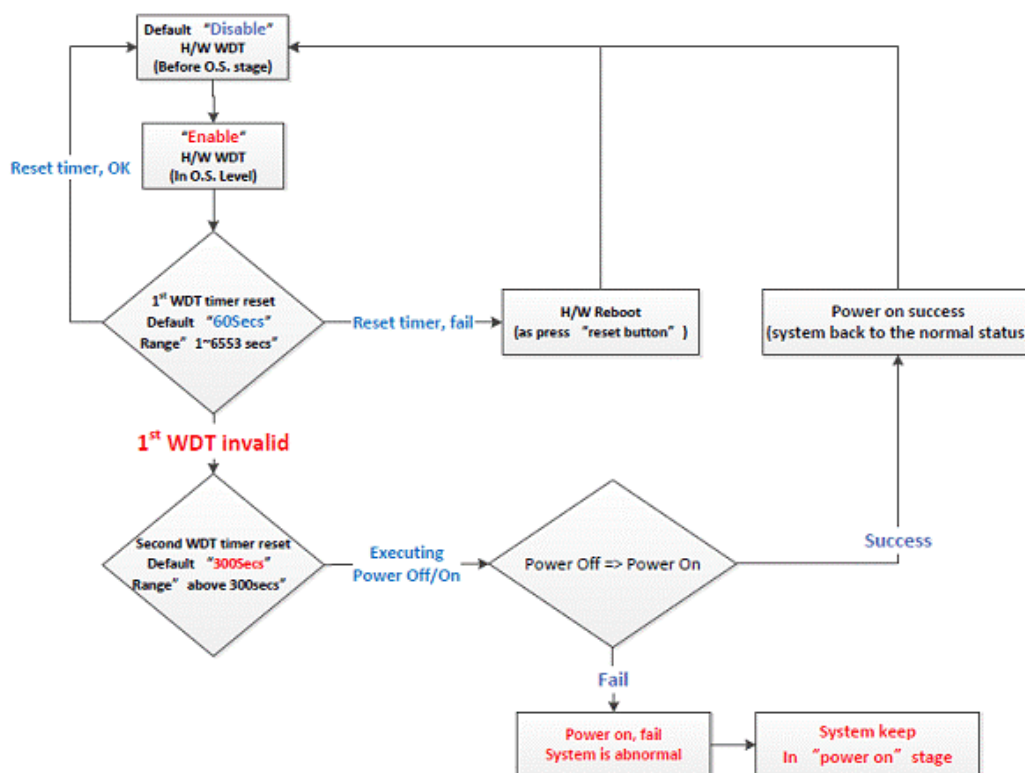


图 2.10: 2X20 pin Connector

2.3.7 Watch Dog timer

UBC-330 可实现一个使用 TI MSP430G2202 MCU 的 WDT 功能。当用户的系统发生意外错误时，可以实现重启功能。



产品出货时默认 time out 时间为 60 秒，当然用户可以根据自己的需求在 “1~6553” 秒这个区间内任意设置。

以下为 WTD 功能测试验证方法：

用户可以执行 AutoRun_WTD 程式来去体验它：

```
#. /AutoRun_WTD n
```

此处 “n” 代表 “喂狗” 时间，其实它就是表示测试程式需要在每隔 “n” 秒时间对系统 “喂狗” 一次

```
. root@am335x-adv:/unit_tests# ./AutoRun_WTD 15
```

Get the timeout value from driver: timeout = 60 seconds

Now, we set the timeout value to 10 seconds

Get the timeout value from driver again: timeout = 10 seconds

Setting succeeded and watchdog is enabled.

Feed the watchdog every 15 seconds.

Feed watchdog!

After reboot, user will see the follow the boot messages:

U-Boot SPL 2013.01.01-svn132 (Sep 28 2014 - 11:39:20)

```
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, bulk combine, bulk split,  
HB-ISO Rx, HB-ISO Tx, SoftConn)  
musb-hdrc: MHDRC RTL version 2.0  
musb-hdrc: setup fifo_mode 4  
musb-hdrc: 28/31 max ep, 16384/16384 memory  
USB Peripheral mode controller at 47401000 using PIO, IRQ 0  
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, bulk combine, bulk split,  
HB-ISO Rx, HB-ISO Tx, SoftConn)  
musb-hdrc: MHDRC RTL version 2.0  
musb-hdrc: setup fifo_mod
```

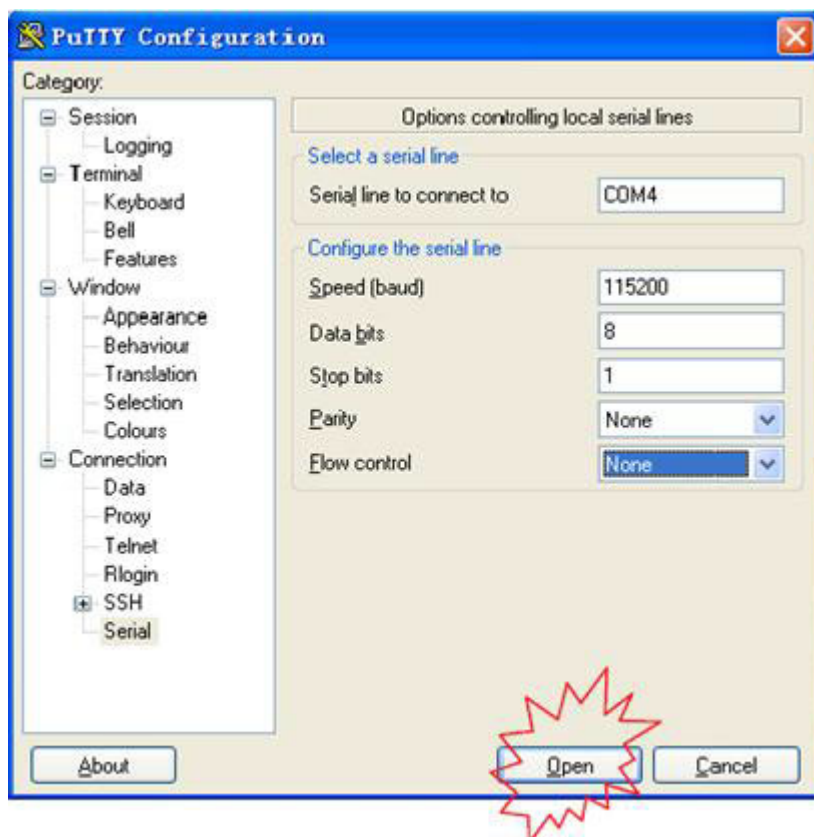
2.4 UBC-330 快速入门指南

2.4.1 调试口连接

1. 连接 “COM0_TX” 和 “COM0_RX” 以及 “GND” 到标准的 DB9 连接头上。
2. 将以上连接好的 DB9 接头连接至主机 COM 端口。

2.4.2 调试口设置

3. 用户通过如下途径下载并安装 “Putty” 应用程序：
<http://putty.cs.utah.edu/download.html>
4. 点击并打开 “Putty” 软体，在如下对话框中输入图示之信息，再点击 “Open”。



5. 将产品通电，用户会在主机显示器上看到通过终端输出出来的引导装置信息，对话框会弹出 “user name” 和 “password”，用户只要在对对话框中的 “user name” 内输入 “root”，就可以以 “超级用户” 的身份点击 “enter” 进入系统。

2.5 测试工具

所有的测试程式在 UBC-330 已通过验证，若需要验证某一端口的功能，必须准备其对应的测试程式及测试工具，如果在此期间遇到什么问题，请联系研华协助解决

2.5.1 eMMC 测试

第一步：开机进入 Linux 系统

第二步：执行以下命令确认 eMMC 的容量

```
root@am335x-evm:~# fdisk -l /dev/mmcblk1
```

```
Disk /dev/mmcblk1: 3909 MB, 3909091328 bytes
4 heads, 16 sectors/track, 119296 cylinders, total 7634944 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xb7e5e6db
```

Device	Boot	Start	End	Blocks	Id	System
/dev/mmcblk1p1		2048	22527	10240	83	Linux
/dev/mmcblk1p2		22528	63487	20480	83	Linux

Step2: Run program to read/write NAND flash.

```
root@am335x-adv:/unit_tests# ./AutoRun_eMMC.sh mmcblk1
```

```
=====
=====Test Read/write and operation of filesystem for eMMC=====
=====

10240+0 records in
10240+0 records out
10240+0 records in
10240+0 records out
10240+0 records in
10240+0 records out
=====MMC Read/Write test pass=====

=====MMC fdisk test pass=====

mke2fs 1.42.1 (17-Feb-2012)
5+0 records in
5+0 records out
=====MMC FS test passes=====

mke2fs 1.42.1 (17-Feb-2012)
=====MMC MKFS test pass=====

=====all MMC function test PASS=====
```

2.5.2 USB 测试

第一步：在 USB 端口插入 USB 设备

第二步：Linux 系统下挂载 USB 设备，保证系统可以识别到 USB 设备

第三步：执行程式进行 USB 设备的读写

```
root@am335x-adv:/unit_tests# ./AutoRun_usb_host.sh
```

```
disk_to_test=sda
---create partition for sda
---format file system
partition_list=sda1
test /dev/sda1
pass
```

2.5.3 SD 卡测试

第一步：将 SD 卡插入 SD 卡槽，上电进入系统

第二步：挂载 SD 卡保证系统可以识别到该 SD 卡，执行以下命令确认

```
root@am335x-adv:/unit_tests# fdisk -l /dev/mmcblk0
```

```
Disk /dev/mmcblk0: 3980 MB, 3980394496 bytes
255 heads, 63 sectors/track, 483 cylinders, total 7774208 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```

Device	Boot	Start	End	Blocks	Id	System
/dev/mmcblk0p1	*	63	144584	72261	c	W95 FAT32 (LBA)
/dev/mmcblk0p2		160650	7759394	3799372+	83	Linux

第三步：执行以下程式进行 SD 测试

```
root@am335x-adv:/unit_tests# ./AutoRun_sd.sh mmcblk0
```

```
10240+0 records in
10240+0 records out
10240+0 records in
10240+0 records out
10240+0 records in
10240+0 records out
=====SD test pass=====
```

2.5.4 SPI 测试

第一步：上电开机进入 Linux 系统

第二步：执行命令进行 SPI Flash 的读写测试

```
root@am335x-adv:/unit_tests# ./AutoRun_spi.sh
```

```
#####-----SPI Test for RSB 4220-----#####
the spi falsh info:

    31          0          4096 mtdblock0
=====
    Test next block:/dev/mtd0  mtdblock0
=====
mtd.type = MTD_NORFLASH
mtd.flags = MTD_CAP_NORFLASH
mtd.size = 4194304 (4M)
mtd.erasesize = 4096 (4K)
mtd.writesize = 1
mtd.oobsize = 0
regions = 0
Erased 4194304 bytes from address 0x00000000 in flash
Copied 4194304 bytes from address 0x00000000 in flash to ./temp-1.img
0000000 ffff ffff ffff ffff ffff ffff ffff ffff
*
0400000

=====
#####                               SPI mtdblock0  Read 1 PASS !!!
=====

4096+0 records in
4096+0 records out
0000000 0000 0000 0000 0000 0000 0000 0000 0000 0000
*
0400000
Copied 4194304 bytes from ./all-0.img to address 0x00000000 in flash
Copied 4194304 bytes from address 0x00000000 in flash to ./temp-0.img
0000000 0000 0000 0000 0000 0000 0000 0000 0000 0000
*
0400000

=====
##### -----SPI mtdblock0 Write 0 PASS !!!
=====

=====
##### -----> SPI Test all mtdblock0 PASS !!!
```

```
##### Finish SPI all blocks  Test PASS !!!
```

2.5.5 I2C 测试

第一步：上电开机进入 Linux 系统

第二步：执行命令侦测 I2C 总线所有的设备

```
root@am335x-adv:/unit_tests# ./AutoRun_i2c.sh
```

```
  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  UU  --  --
30:  UU  UU  UU  UU  UU  UU  UU  UU  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  UU  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  UU  --  --  --  --  --  --  --  --  --  --  --  --
====I2C test Pass!====
```

2.5.6 CAN 测试

第一步：将两台 UBC-330 的 CAN 口对接

第二步：将两台 UBC-330 上电进入 Linux 系统后，执行如下程式：

```
root@am335x-adv:/unit_tests# ./AutoRun_CAN.sh
```

```
Sun Sep 14 01:48:57 UTC 2014
```

```
interface = can0, family = 29, type = 3, proto = 1
```

```
wait for data
```

```
interface = can0, family = 29, type = 3, proto = 1
```

```
====CAN Pass=====
```

```
Sun Sep 14 01:49:01 UTC 2014
```

```
interface = can0, family = 29, type = 3, proto = 1
```

```
wait for data
```

```
interface = can0, family = 29, type = 3, proto = 1
```

```
====CAN Pass=====
```

2.5.7 GPIO 测试

第一步：上电开机进入 Linux 系统

第二步：将 GPIO0 和 GP00 短接，GPIO1 和 GP01 短接，GPIO2 和 GP02 短接，GPIO3 和 GP03 短接。

第三步：执行程式进行 GPIO 读写测试

```
root@am335x-adv:/unit_tests# ./AutoRun_gpio.sh
```

```
GPIO200 direction is:
```

```
in
```

```
GPIO201 direction is:
```

```
in
```

```
GPIO202 direction is:
```

```
in
```

```
GPIO203 direction is:
```

```
in
```

```
GPIO204 direction is:
```

```
out
```

```
GPIO205 direction is:
```

```
out
```

```
GPIO206 direction is:
```

```
out
```

```
GPIO207 direction is:
```

```
out
```

```
GPIO test PASS!
```

2.5.8 网口测试

UBC-330 默认设置 DHCP 协议。

2.5.8.1 eth0 测试

第一步：将 UBC-330 第 1 个网口通过网线与主机网口连接

第二步：将 UBC-330 的网口 IP 设置为 192.168.1.2。将主机网口的 IP 地址设置为 192.168.1.1

```
root@am335x-adv:~# ifconfig eth0 192.168.1.2
```

```
root@am335x-adv:~# ifconfig eth0
```

```
eth0      Link encap:Ethernet  HWaddr 78:A5:04:DD:E1:0A
```

```
inet addr:192.168.1.2  Bcast:192.168.1.255  Mask:255.255.255.0
```

```
UP BROADCAST RUNNING ALLMULTI MULTICAST MTU:1500 Metric:1
```

```
RX packets:160 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:23334 (22.7 KiB)  TX bytes:0 (0.0 B)
```

第三步：用户可以输入以下命令来确认我们是否有收到来自主机的回执

```
root@am335x-adv:~# ping 192.168.1.1
```

```
PING 192.168.1.1 (192.168.1.1): 56 data bytes
```

```
64 bytes from 192.168.1.1: seq=0 ttl=64 time=0.384 ms
```

```
64 bytes from 192.168.1.1: seq=1 ttl=64 time=0.159 ms
```

```
64 bytes from 192.168.1.1: seq=2 ttl=64 time=0.110 ms
```

```
64 bytes from 192.168.1.1: seq=3 ttl=64 time=0.102 ms
```

```

64 bytes from 192.168.1.1: seq=4 ttl=64 time=0.208 ms
64 bytes from 192.168.1.1: seq=5 ttl=64 time=0.135 ms
64 bytes from 192.168.1.1: seq=6 ttl=64 time=0.186 ms
64 bytes from 192.168.1.1: seq=7 ttl=64 time=0.151 ms
64 bytes from 192.168.1.1: seq=8 ttl=64 time=0.091 ms
64 bytes from 192.168.1.1: seq=9 ttl=64 time=0.203 ms
64 bytes from 192.168.1.1: seq=10 ttl=64 time=0.111 ms
64 bytes from 192.168.1.1: seq=11 ttl=64 time=0.105 ms
64 bytes from 192.168.1.1: seq=12 ttl=64 time=0.098 ms
64 bytes from 192.168.1.1: seq=13 ttl=64 time=0.091 ms
64 bytes from 192.168.1.1: seq=14 ttl=64 time=0.187 ms
64 bytes from 192.168.1.1: seq=15 ttl=64 time=0.123 ms

```

2.5.8.2 eth1 测试

第一步：将 UBC-330 第 2 个网口通过网线与主机网口连接

第二步：将 UBC-330 的网口 IP 设置为 192.168.1.3.

```
root@am335x-adv:~# ifconfig eth1 192.168.1.3
```

```
root@am335x-adv:~# ifconfig eth1
```

```

eth1      Link encap:Ethernet  HWaddr 78:A5:04:DD:E1:0C
          inet addr:192.168.1.3  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:41 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:5035 (4.9 KiB)  TX bytes:0 (0.0 B)

```

第三步：用户可以输入以下命令来确认我们是否有收到来自主机的回执

```
root@am335x-adv:~# ping 192.168.1.1
```

```

PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: seq=0 ttl=64 time=0.373 ms
64 bytes from 192.168.1.1: seq=1 ttl=64 time=0.208 ms
64 bytes from 192.168.1.1: seq=2 ttl=64 time=0.234 ms
64 bytes from 192.168.1.1: seq=3 ttl=64 time=0.115 ms
64 bytes from 192.168.1.1: seq=4 ttl=64 time=0.122 ms
64 bytes from 192.168.1.1: seq=5 ttl=64 time=0.107 ms

```

2.5.9 UART 测试

UBC-330 有 6 个串口，/dev/tty0 作为调试口使用，另外 5 个串口供客户使用

2.5.9.1 UART1 to UART5 RS232 测试

■ UART1 到 UART5 的 RS232 测试

短接 UART1 到 UART5 的 TX 和 RX 再输入以下命令：

```
root@am335x-adv:/unit_tests# ./AutoRun_uart232
```

```
=====test rs232!=====
```

```
rs232 number: 5
```

```
/dev/tty01 PASS!
```

```
/dev/tty02 PASS!
```

```
/dev/tty03 PASS!
```

```
/dev/tty04 PASS!
```

```
/dev/tty05 PASS!
```

```
+-----+
```

```
| [RS232] Test Pass!
```

第 3 章

软件信息

本章介绍软件特性，如 BSP、镜像等。

3.1 介绍

UBC-330 微型计算机使用嵌入式 Linux 系统，内核版本为 3.2.0。该嵌入式 Linux 系统包括了常用的 shell 命令和 UBC-330 必要的驱动，另外，我们提供交叉编译工具和 UBC-330 BSP 源码包，用户可以使用交叉编译工具，在 BSP 源码包的基础上对 UBC-330 嵌入 Linux 进行二次开发。我们建议使用 Ubuntu 12.04 作为开发环境。

UBC-330 嵌入式 Linux 系统包含三部分：u-boot.img, uImage 和 File System, u-boot.img 是引导程序文件映像，负责引导 Linux 系统，并进行最基本的硬件初始化动作；uImage 是 Linux 内核镜像文件，File System 是 Linux 的根文件系统。

上面三部分对于整个嵌入式 Linux 系统都是必要的，缺少任何一个，系统都将不能正常启动。

本章主要是介绍 UBC-330 的软件使用 and 开发，用户在使用 Linux 系统和软件开发前，请仔细阅读本章，相信对使用和开发人员将会有很大帮助。

UBC-330 的软件开发不能直接在 UBC-330 上开发，也不能在 windows 或 android 环境的主机进行开发，需要使用 Linux 环境的主机进行软件开发。我们推荐开发者使用 Ubuntu 12.04 LTS 操作系统环境进行开发，这是我们标准的开发环境，如果客户使用其他 Linux 版本开发机进行开发，有可能会出现兼容性问题。

3.2 软件包

研华为 UBC-330 提供了两个软件包：一个系统 image 包，用于系统的恢复和还原，另外一个为开发使用的源码包，我们称之为 BSP 源码包，BSP 源码包里面包含有开发所需的整个源码，示例程序和交叉编译工具。

3.2.1 Image 映像文件包

UBC-330 Image 包是文件名为 4220LIVxxxx_yyyy-mm-dd.tar.gz 的压缩包，xxxx 包是版本号，yyyy-mm-dd 表示这个包发布的年月日。解压后，用户可以看到相关的 image 映像文件，通过把这些文件烧录到 UBC-330 中，UBC-330 就可以顺利启动 Linux 系统了。具体方法参考 3.6 创建 Linux 系统。

3.2.2 BSP 源码包

UBC-330 BSP 源码包包括交叉编译工具，Linux 内核源码，uboot 源码，根文件系统，脚本程序，示例程序等。其中有些软件及源码是研华开发的，有一些是直接来自于网络上的开源社区。UBC-330 BSP 源码包主要包括六个目录：cross_compiler, document, image, package, scripts 和 source。

注！ UBC-330 源码包是归属研华的知识产权所有，如果需要，请联系研华。



图 3.1：源码包目录结构

UBC-330 BSP 源码包目录说明:

- “cross_compiler” 目录为交叉编译工具目录
- “document” 录为用户手册目录
- “image” 该目录包含 uImage, u-boot.img
- “image/rootfs” 该目录目录包含了 Linux 根文件系统
- “package” 该目录包含了 TI 为 AM-335x 平台提供的原始源码
- “scripts” 该目录包含了一些配置和编译的脚本
- “source” 该目录包含了 UBC-330 的相关源码

3.2.2.1 cross_compiler 目录

研华提供的交叉编译工具 gcc 版本是 4.7.3 20130226, 用户可以使用该交叉编译工具来编译 Linux 内核及相关软件

3.2.2.2 document 目录

文档目录包含了用户手册

3.2.2.3 image 目录

Image 目录包含了 uImage 和 u-boot.img

3.2.2.4 image/rootfs

rootfs 是嵌入式 Linux 的根文件系统。

主要包含以下目录:

- bin 普通的可执行文件。
- dev 各种设备节点文件。
- etc 重要的系统配置文件和初始化脚本文件。
- home 用户目录。
- lib 库文件, 包含系统和应用程序依赖的各种库文件。
- mnt 这个目录一般是用于存放挂载储存设备的挂载目录的。
- opt 通常包含一些第三方的软件。
- proc 可以在这个目录下获取系统信息。这些信息是在内存中, 由系统自己产生的。
- root root 用户的默认工作目录, 不是系统的根目录。
- sbin 这个目录是用来存放系统管理员的系统管理程序。大多是涉及系统管理的命令的存放, 是超级权限用户 root 的可执行命令存放地。
- sys Linux sys 文件系统。
- tmp 临时文件目录, 用来存放不同程序运行时产生的临时文件。有时用户运行程序的时候, 会产生临时文件。/tmp 就用来存放临时文件的。/var/tmp 目录和这个目录相似。
- usr 包含一些用户应用程序, 软件库, 文档等
- var 通常存放各种用户生成的文件或临时文件, 不如 log 日志文件, 网络下载文件等
- tools 测试程序

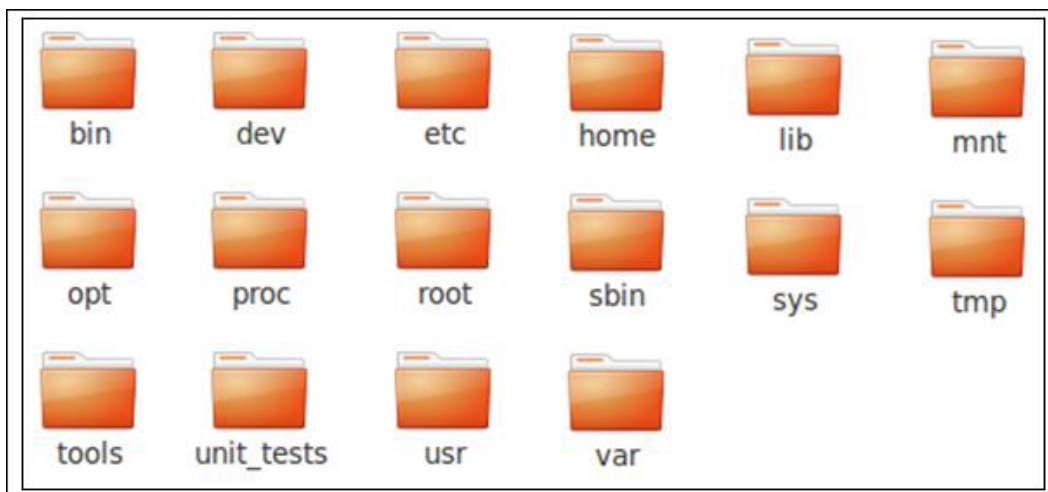


图 3.2: image\rootfs

3.2.2.5 scripts 目录

研华提供的一些脚本文件，帮助用户配置开发环境，生成 Linux image 映像文件，并烧到 SD 和 flash 等。

setenv.sh	设置开发环境，导出开发所需的环境变量。
cfg_uboot.sh	配置 uboot 编译选项。
mk_uboot.sh	编译 u-boot 源码，并拷贝 u-boot 到 ../image 目录。
cfg_kernel.sh	配置内核。
mk_kernel.sh	编译内核，生成 uImage，并将 uImage 拷贝到 ../image 目录。
mk_sd-linux.sh	制作一个 Linux SD 卡，插入设备后可以从 SD 启动。
mk_nand-linux.sh	从 SD 卡启动后，执行该脚本将系统烧录到 flash 上。

3.2.2.6 source 目录

这个目录包含两个子目录：linux-3.2.0-psp04.06.00.11 和 u-boot-2013.01.01-psp06.00.00.00，分别是 Linux 内核源码目录和 u-boot 源码目录。

Linux 系统是一个多任务的 OS，支持虚拟内存，共享库，内存管理，多网路等。

Linux 支持绝大多数 32 位硬件平台，支持 PMMU，甚至可以移植到一个没有 PMMU 单元的硬件上。

“linux-3.2.0-psp04.06.00.11” 内核源码目录主要包含以下子目录：

arch	与体系结构相关的代码。对应于每个支持的体系结构，有一个相应的子目录如 x86、arm 等。
block	部分块设备驱动程序。
crypto	常用加密和散列算法如 AES、SHA 等 还有一些压缩和 CRC 校验算法。
documentation	关于内核各部分的通用解释和注释。
drivers	放置系统所有的设备驱动程序；每种驱动程序又各占用一个子目录。
firmware	一些 firmware 代码。
fs	内核支持文件系统驱动代码。
include	内核头文件。
init	这个目录包含核心的初始化代码。
ipc	进程间通信的代码。

kernel	主要的核心代码，此目录下的文件实现了大多数 linux 系统的内核函数，定义了内核进程，调度，信号等。
lib	库函数代码。
mm	这个目录包括所有独立于 CPU 体系结构的内存管理代码，如页式存储管理内存的分配和释放等。
net	网络支持代码。
security	安全设定代码。
sound	声音驱动相关代码。
virt	虚拟机相关代码。

因特网，书或者杂志上，很容易找到大量这方面文档，用户可以通过这些查找这些信息学习到更多关于 Linux 的知识。

在 `./source/linux-3.2.0-psp04.06.00.11/Documentation` 目录下，也有很多 README 文件，用户可以找到很多内核和驱动相关的说明。

3.3 建立开发环境

这个小节主要介绍在 Ubuntu 12.04 LTS 下搭建和配置开发环境，首先，用户需要在开发机上安装 Ubuntu 12.04 LTS 系统。

然后从研华官方获取 RS-4220 BSP 源码包，安装下面步骤解压源码包。

- 1) 把源码包拷贝到 `/root` 目录下
- 2) 在 Ubuntu 12.04 LTS 下开启一个“Terminal”终端
- 3) **\$sudo su**（获得 root 超级用户权限）
- 4) 根据提示输入超级用户密码
- 5) **#cd /root/**
- 6) **#chmod a+x 335XLBVxxxx_yyyy-mm-dd.bin**
- 7) **#./335XLBVxxxx_yyyy-mm-dd.bin**
- 8) 输入“yes”
- 9) 在 root 目录下会生成“335XLBVxxxx_yyyy-mm-dd”目录

注！ *xxxx 表示版本信息，yyyy 表示年，mm 表示月，dd 天。例如：
335XLBV1010_2014-10-01. 表示版本号是 1010，发布日期是 2014-10-01*



在 BSP 包中提供了一些脚本，通过运行这些脚本，可以快速建立和配置开发环境，请根据下面步骤配置开发环境：

- 1) 在 Ubuntu 12.04 LTS 下开启一个“Terminal”终端
- 2) **\$sudo su**（获得 root 超级用户权限）
- 3) 根据提示输入超级用户密码
- 4) **#cd /root/335XLBVxxxx_yyyy-mm-dd/scripts/**
- 5) **#source setenv.sh**
- 6) 这样，用户就可以开始编译内核和开发应用程序了。

setenv.sh 是用来配置和导出开发环境的环境变量的。

注！ *每次打开一个新的“Terminal”来进行编译，都要首先要执行“#source setenv.sh”*



注! 执行该脚本是，先获得 root 用户权限



3.4 编译源码

本小节主要介绍如何编译 u-boot 和 Linux 内核源码

3.4.1 编译 u-boot

BSP 包中提供了编译 U-boot 的脚本程序，按照下面的方法运行编译脚本，可以快速的编译出 u-boot.

- 1) 在 Ubuntu 12.04 LTS 下开启一个 "Terminal" 终端
- 2) `$sudo su` (获得 root 超级用户权限)
- 3) 根据提示输入超级用户密码
- 4) `#cd /root/335XLBVxxxx_yyyy-mm-dd/scripts/`
- 5) `#source setenv.sh`
- 6) `#./cfg_uboot am335x_rsb4220`
- 7) `#./mk_uboot.sh`
- 8) 完成编译后，可以在 ../image 目录中看到生成了 u-boot.img

3.4.2 编译 Linux 内核

BSP 包中提供了编译 Linux 内核的脚本文件，按照下面的方法运行编译脚本，可以快速的编译出 Linux 内核文件：

- 1) 在 Ubuntu 12.04 LTS 下开启一个 "Terminal" 终端
- 2) `$sudo su` (获得 root 超级用户权限)
- 3) 根据提示输入超级用户密码
- 4) `#cd /root/335XLBVxxxx_yyyy-mm-dd/scripts/`
- 5) `#source setenv.sh`
- 6) `#./cfg_kernel.sh am335x_rsb4220_defconfig`
- 7) `#./mk_kernel.sh (Start to build the uImage)`
- 8) 完成编译后，可以在 ../image 目录中看到生成了 uImage

3.4.3 编译日志

编译后，可以在 `"/335XLBVxxxx_yyyy-mm-dd/"` 目录下找到编译的日志文件。如果在编译过程中出现错误，可以通过查看日志文件定位和分析问题。

3.5 内核源码修改

本小节会说明如何修改 Linux 源码，添加驱动支持。

下面显示如何在内核添加一个 watchdog 看门狗驱动程序 (watchdog 看门狗芯片为 Seiko Instruments S-35390A)。步骤如下：

- 1) 在 Ubuntu 12.04 LTS 下开启一个“Terminal”终端
- 2) `$sudo su` (获得 root 超级用户权限)
- 3) 根据提示输入超级用户密码
- 4) `#cd /root/335XLBVxxxx_yyyy-mm-dd/scripts/`
- 5) `#source setenv.sh`
- 6) `#./cfg_kernel.sh am335x_rsb4220_defconfig`
- 7) `#./cfg_kernel.sh menuconfig`
- 8) 然后，就可以看到下面的内核配置画面

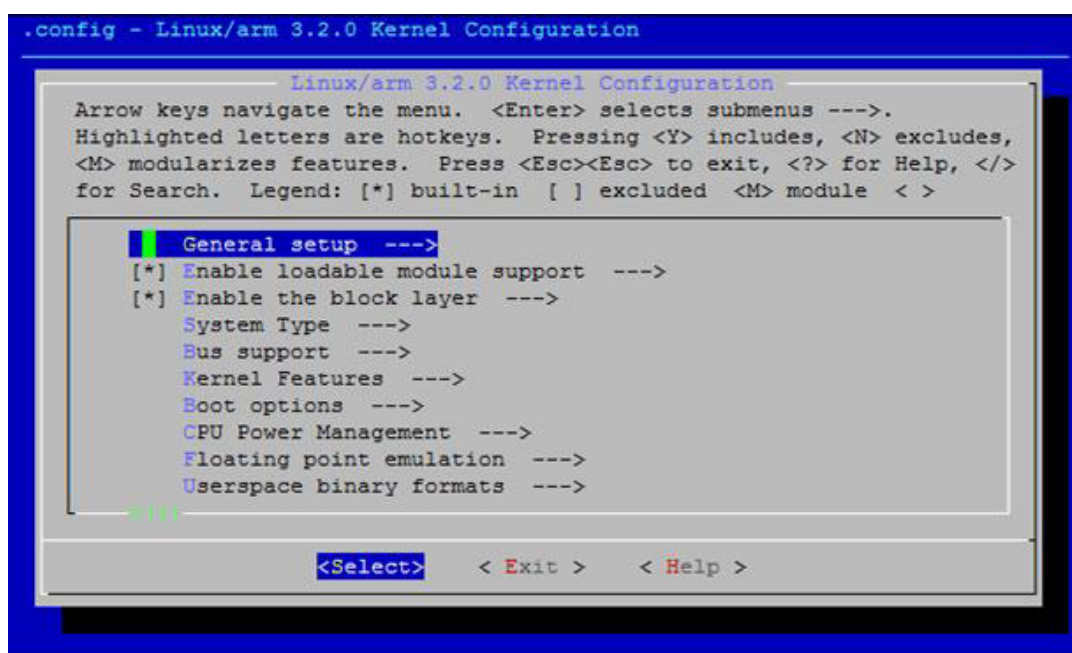


图 3.3: Linux 内核配置

- 9) 选择 “Device Drivers” “Real Time Clock”， 你可以看到 “Seiko Instruments S-35390A” 选项，选中并保存退出。

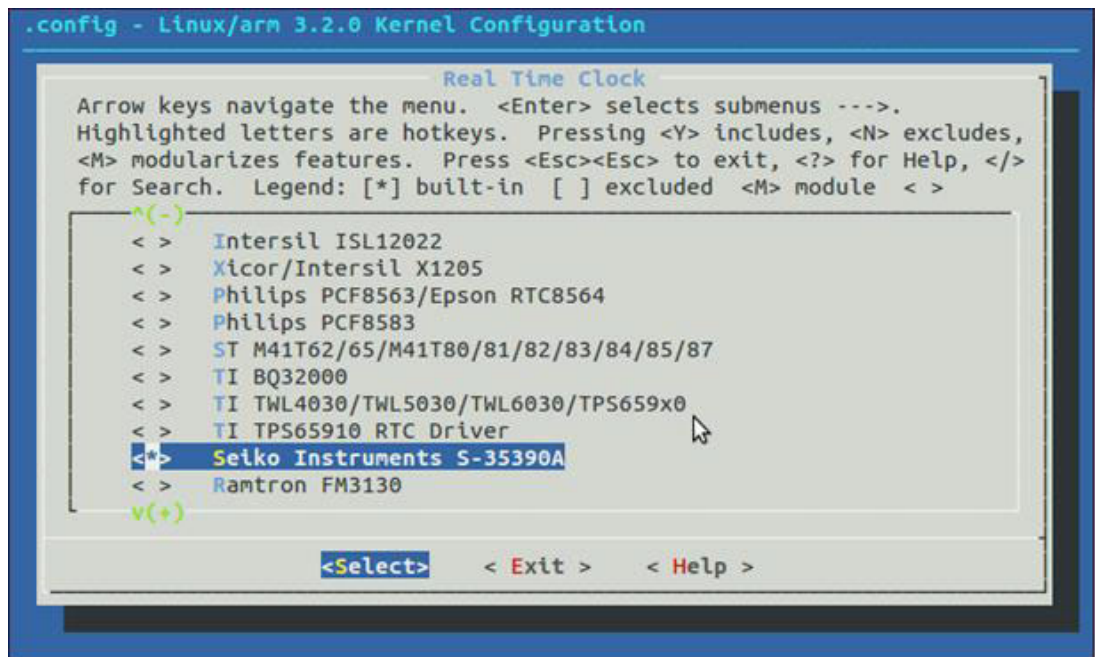


图 3.4: 选择 Seiko Instruments S-35390A

- 10) 切换目录到
“source/ linux-3.2.0-psp04.06.00.11/arch/arm/mach-omap2”，
编辑 “board-rsb4220.h” 和 “board-advantech.c”。
加入如下代码到source/ linux-3.2.0-psp04.06.00.11/arch/arm/mach-mx6/
board-rsb4220.h:

```
/* I2C */
static struct i2c_board_info mxc_i2c0_board_info[] __initdata
= {
    {
        I2C_BOARD_INFO("s35390a", 0x30),
    },
};
```

加入如下代码到

```
source/ linux-3.2.0-psp04.06.00.11/arch/arm/mach-omap2/board-advantech.c
omap_register_i2c_bus(1, 100, am335x_i2c0_boardinfo,
    ARRAY_SIZE(am335x_i2c0_boardinfo));
```

- 11) 然后请参考 3.4.2 章节重编内核，新的内核就集成了
watchdog (Seiko Instruments S-35390A) 驱动。

注!



如果现有内核不支持需要的驱动程序，开发者需要自己开发驱动或联系
芯片厂商获得相应的 Linux 驱动

3.6 创建 Linux 系统

UBC-330 可以从 SD 卡或者板载的 flash 启动，这个章节会介绍如何创建一个可以启动 SD 卡启动的 Linux 系统以及如何从系统从板载 flash 启动 Linux 系统。

3.6.1 存储介质（eMMC/SD 卡）

存储介质设备名称如下：

设备	名称
SD 卡	/dev/mmcblk0
eMMC flash	/dev/mmcblk1

3.6.2 创建 SD 卡 Linux 系统

3.6.2.1 从 image 包来制作 Linux SD 卡

从研华网站下载 UBC-330 image 包，然后按照下面步骤制作可以启动的 SD 卡

- 1) 拷贝 “4220LIVxxxx_yyyy-mm-dd.tar.gz” 到 /root/ 目录。
 - 2) 在 Ubuntu 12.04 LTS 下开启一个 “Terminal” 终端
 - 3) **\$sudo su**（获得 root 超级用户权限）
 - 4) 根据提示输入超级用户密码
 - 5) **#cd /root/**
 - 6) **#tar xzvf 4220LIVxxxx_yyyy-mm-dd.tar.gz** (Unzip files)
 - 7) 插入 SD 卡到开发主机
 - 8) 检查 SD 的设备节点，这里假定是 /dev/sdb
 - 9) **#cd ./4220LIVxxxx_yyyy-mm-dd/scripts**
 - 10) **#./mkfsd-linux.sh /dev/sdb**
 - 11) 输入 “y”（开始拷贝文件，请等待，直到出现 [Done]）
- 然后插入 SD 卡到 UBC-330，Linux 系统就可以顺利启动了。

3.6.2.2 从 BSP 源码包制作 Linux SD 卡

如果用户拿到的是 Linux BSP 源码吧，可以按照下面步骤来创建 Linux SD 卡：

- 1) 在 Ubuntu 12.04 LTS 下开启一个 “Terminal” 终端
- 2) **\$sudo su**（获得 root 超级用户权限）
- 3) 根据提示输入超级用户密码
- 4) 插入 SD 卡到开发主机
- 5) 检查 SD 的设备节点，这里假定是 /dev/sdb
- 6) **#cd /root/335XLBVxxxx_yyyy-mm-dd/scripts**
- 7) **#./mkfsd-linux.sh /dev/sdb**

输入 “y”（开始拷贝文件，请等待，直到出现 [Done]）

然后插入 SD 卡到 UBC-330，Linux 系统就可以顺利启动了。

3.6.3 从 eMMC flash 启动 Linux

首先，按照上面的方法制作一个可以启动 Linux 的 SD 卡，从 SD 启动 Linux 系统，然后，就可以按照下面的步骤，将 SD 卡的系统拷贝到板载的 flash 上，完成后，拔掉 SD，重启系统，系统就可以从板载的 flash 启动了，方法如下：

- 1) 参考 3.6.1 制作可以启动的 SD 卡
- 2) 将可以启动的 SD 卡插入 UBC-330，连结好调试串口，启动系统。
- 3) 在 UBC-330 控制台，输入 `#root` 登录
- 4) 在 UBC-330 控制台，输入 `#cd /mk_inand`
- 5) 在 UBC-330 控制台，输入 `#./mkinand-linux.sh /dev/mmcblk1`
- 6) 关电，拔掉 SD 卡

这时系统就可以从板子 flash 启动了。

3.7 调试信息

UBC-330 可以通过调试串口和主机进行连结，主机是 Linux 和 windows 平台都可以。

为了能和主机顺利进行通信，主机端需要打开串口超级终端，比如 HyperTerminal, Tera Term 或 PuTTY 等软件。下图就是以 windows 为例，说明通过 windows 下的“HyperTerminal”超级终端软件，和 UBC-330 进行通信。“HyperTerminal”界面如下：

- 1) 通过 Debug 串口线连接 UBC-330 调试串口和 windows 主机串口。
- 2) 打开 windows 超级终端 HyperTerminal，设定如下图所示 Figure 3-5.
- 3) 加电启动 UBC-330，UBC-330 的启动信息就会出现在超级终端上。

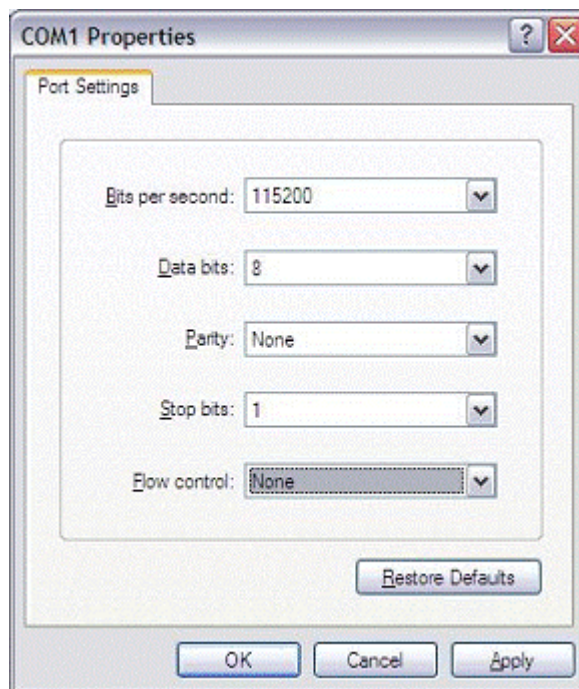


图 3.5：超级终端设定

3.8 Linux 系统配置和使用

3.8.1 服务配置

UBC-330 内置了 5 个通用的网络服务：tftp 服务、ftp 服务、ssh 服务、telnet 服务和 http 服务。

3.8.1.1 tftp 服务

当 UBC-330 启动后，tftp 服务也会默认启动，tftp 服务的工作目录是 /tftpboot。请在 UBC-330 端执行 “chmod 777 /tftpboot”，使 tftp 可以正常工作。然后，用户在其他计算机上可以通过 tftp 命令来上传文件到 UBC-330，或者从 UBC-330，下载文件。

```
hostPC$ tftp TARGET_SYSTEM_IP
tftp>get file1
tftp>put file2
```

注！ 命令 “get file1” 是从 tftp 服务器下载文件。请确保 “file1” 在 UBC_FA30 上的 /tftpboot 下是存在的。



命令 “put file2” 是上传文件到 tftp 服务器。如果上传成功，file2 就会出现在 UBC_FA30 上的 /tftpboot 目录下

启动服务命令如下：

```
root@am335x-adv?/ # /etc/init.d/tftpd start
```

关闭服务命令如下：

```
root@am335x-adv?/ # /etc/init.d/tftpd stop
```

3.8.1.2 ftp 服务

UBC_FA30 使用 vsftpd 作为 FTP 服务器，可以使用下面的命令手动开启该服务：

```
root@am335x-adv?/ # /etc/init.d/vsftpd start
```

停止服务命令如下：

```
root@am335x-adv?/ # /etc/init.d/vsftpd stop
```

注！ 当开启服务后，可以通过下面命令添加 FTP 用户。



```
root@am335x-adv?/ # adduser ftp
root@am335x-adv?/ # chown root:root /home/ftp/
```

这样，就可以使用 UBC-330 的 FTP 服务了。

3.8.1.3 ssh 服务

当启动 UBC-330 后，ssh 服务就会默认启动，其他主机设备可以通过下面的 ssh 命令连接到 UBC-330：

```
hostPC$ sudo ssh -l root TARGET_SYSTEM_IP
```

启动服务命令如下：

```
root@am335x-adv?/ # /etc/init.d/dropbear start
```

停止服务命令如下：

```
root@am335x-adv?/ # /etc/init.d/dropbear stop
```

3.8.1.4 telnet 服务

当启动 UBC-330 后，telnet 服务就会默认启动，其他主机设备可以通过 telnet 命令连接到 UBC-330

```
hostPC$ sudo telnet TARGET_SYSTEM_IP
```

启动服务命令如下：

```
root@am335x-adv?/ # /etc/init.d/telnetd start
```

停止服务命令如下：

```
root@am335x-adv?/ # /etc/init.d/telnetd stop
```

3.8.1.5 http 服务

UBC-330 使用 lighttpd 作为 web 服务器。

启动服务命令如下：

```
root@am335x-adv?/ # /etc/init.d/lighttpd start
```

停止服务命令如下：

```
root@am335x-adv?/ # /etc/init.d/lighttpd stop
```

3.8.2 网络配置

用户可以通过 debug 串口来配网络的 IP 地址。

自动或者 IP 地址:

```
advantech# /etc/init.d/dhcpc eth0 start
advantech# /etc/init.d/dhcpc eth1 start
```

如果希望系统重启后仍可以自动获得 IP，执行以下命令：

```
advantech# echo "/etc/init.d/dhcpc eth0 start" > /etc/adv.d/
netcfg.eth0
advantech# echo "/etc/init.d/dhcpc eth1 start" > /etc/adv.d/
netcfg.eth1
```

设定固定 IP 地址，首先通过下面命令停止 DHCP

```
advantech# /etc/init.d/dhcpc eth0 stop
advantech# /etc/init.d/dhcpc eth1 stop
```

通过下面命令设定固定 IP 地址

```
advantech# /sbin/ifconfig eth0 172.21.73.191 netmask 255.255.255.0
advantech# /sbin/route add default gw 172.21.73.253 eth0
advantech# echo 'nameserver 172.21.128.251' >> /etc/resolv.conf
advantech# /sbin/ifconfig eth1 192.168.3.102 netmask 255.255.255.0
advantech# /sbin/route add default gw ... eth1
advantech# echo 'nameserver 172.21.128.251' >> /etc/resolv.conf
```

如果希望系统重启后，保留当前的固定 IP 设定，执行下面命令：

```
advantech# echo "/sbin/ifconfig eth0 172.21.73.191 netmask
255.255.255.0; /sbin/route add default gw 172.21.73.253 eth0;echo
'nameserver 172.21.128.251' > /etc/resolv.conf;" > /etc/adv.d/
netcfg.eth0
advantech# echo "/sbin/ifconfig eth1 192.168.3.102 netmask
255.255.255.0; /sbin/route add default gw ... eth1;echo 'nameserver
172.21.128.251' > /etc/resolv.conf;" > /etc/adv.d/netcfg.eth1
```

注！ 上面命令中的 IP 地址需根据实际情况在输入的时候进行替换。



比如：上面命令中的 IP 172. 21. 128. 251 是 DNS 服务器的 IP 地址，用户在实际设定是要替换正确的 DNS 服务器的 IP。

3.8.3 日期 / 时间配置

用户可以使用 date 命令和 hwclock 命令进行日期和时间设定。

3.8.4 系统版本

用户可以执行以下命令获得系统的版本信息：

```
root@am335x-adv: ~# version
Bsp version:      UBC-330 V1.000
Device name?     UBC-330
Release date:     2014-10-09
Kernel version:   3.2.0
```

3.8.5 添加 / 移除开机启动项

1) 移除启动项

```
update-rc.d [-n] [-f] [-r <root>] <basename> remove
```

basename is your service script name

比如： `update-rc.d -f matrix-gui-2.0 remove`

2) 添加启动项

首先确保服务脚本存在，然后通过下面命令加载：

```
update-rc.d [-n] [-r <root>] [-s] <basename> start|stop NN runlvl [runlvl]
[...] .
```

start|stop : 当系统开机或关机是，服务项可以自动开机和关闭

NN: 0~99

runlvl: UBC-330 runlevel 默认为5;

如. `update-rc.d networking start 40 5 .`

可以发现在 /etc/rc5.d 目录下添加了 S40networking 开机脚本。

3.8.6 在线安装软件

3.8.6.1 OPKG 软件包管理器

Opkg 是一个轻量级的包管理软件。使用 C 语言实现。在 OpenEmbedded 和 OpenWrt 等开源的嵌入式 Linux 项目中广泛使用。

研华 UBC-330 设备的嵌入式 Linux 系统也内置有 OPKG 包管理软件，使用这个软件管理，用户可以在线安装大部分所需的软件，并进行卸载或升级等管理。

3.8.6.2 安装一个新的软件包

如果用户需要安装一个当前系统中不存在的软件包，可以执行下面的程序：

Step 1: 更新在线安装源

```
advantech# opkg update
```

Step 2: 搜寻安装源中是否有客户需要的软件包

```
advantech# opkg list | grep package
```

注！



上面命令中的 *Package* 是安装包的关键字，比如，如果用户需要安装一个 *ftp* 服务器，*package* 就替换为 *ftp*。

Step 3: 如果有搜寻到所需的包，通过下面命令安装该包，packagename 是包的全名。

```
advantech# opkg install packagename
```

3.8.6.3 关于 OPKG 更多信息

关于 OPKG 更多信息，请参考以下网址：

<https://code.google.com/p/opkg/>

3.9 开发向导

3.9.1 C/C++ 应用程序开发

这个小节演示如何写的“Hello World”的小程序，并在 UBC-330 上运行，步骤如下：

1) 在 Ubuntu 12.04 LTS 下打开“Terminal”终端程序，并切换到 root 权限：

```
$ sudo su
```

按照提示输入 root 用户密码

2) 设定开发环境变量：

```
#source      /usr/local/cross_compiler/linux-devkit/environment-
setup
# cd /root//4220LBVxxxx_yyyy_mm_dd/source
# mkdir helloworld
# cd helloworld
# gedit hellowrold.c
```

3) 通过文本编辑器（gedit），编写一个字符文件，文件中撰写如下代码：

```
#include <stdio.h>
void main()
{
    printf("Hello World!\n");
}
```

保存为 helloworld.c 并退出文本编辑器。

4) 通过下面命令编译 helloworld.c 源码文件：

```
# $CC -o helloworld helloworld.c
```

编译后，可以看到在当前目录下生成了“helloworld”可执行程序。

5) 在 UBC-330 上运行 helloworld 可执行程序

可以通过 U 盘或 SD 卡将 helloworld 程序拷贝到 UBC-330，比如，在开发机上插入 SD 卡，将 helloworld 可执行程序拷贝到 SD，拔出 SD，插入 UBC-330，打开调试串口，启动系统，运行

```
#./helloworld
```

就可以看到程序在控制台上的打印信息：

```
"Hello World!"
```

3.9.2 示例程序源码

3.9.2.1 串口编程

参考 <BSP_PATH>/source/demo/uart 目录下代码

该目录下包含一个接收程序和一个发送程序

接收数据，执行下面命令：

```
# ./uart_ctrl read /dev/tty01
```

发送数据，执行下面命令：

```
# ./uart_ctrl write /dev/tty02
```

请参照这两个示例程序的源码来开发用户自己的串口通信程序。

3.9.2.2 Watchdog 看门狗编程

UBC-330 支持硬件 watchdog 看门狗功能，看门狗 API 遵循 posix 标准，watchdog 超时时间的有效设定范围是从 1 到 6553，单位为秒，如果设置了非法值，驱动会自动将超时时间设置成默认值，即 60 秒。

以下是一段示例代码

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <signal.h>
#include <linux/watchdog.h>

int fd;

int main(int argc, const char *argv[]) {
    int timeout = 10;

    /*open watchdog device, the watchdog device node is /dev/watchdog */
    fd=open("/dev/watchdog",O_WRONLY);

    /*set timeout to 10 seconds*/
    /*when set timeout value, the watchdog driver will enable the watchdog
    automatically.*/
    ioctl (fd, WDIOC_SETTIMEOUT, &timeout);
    while (1)
    {
        /*feed the watchdog every 5 seconds*/
        /*when call this funtion to feed watchdog, the watchdog will
        reset its internal timer so it doesn't trigger the board reset. If do not
        feed the watchdog more than 10 seconds, the watchdog will timeout and the
        board will reset.*/
        ioctl( fd, WDIOC_KEEPAIVE, NULL );
        sleep (5);
    }
    close (fd);
}
```

Here are some other APIs for watchdog.

Disable the watchdog timer sample code:

```
/*if user want to disable the watchdog before timeout, call the following
ioctl function*/
int i_dis = WDIOS_DISABLECARD;
ioctl( fd, WDIOC_SETOPTIONS, &i_dis );
```

Enable the watchdog timer sample code:

```
/*if user want to enable the watchdog again before timeout when it is
disabled, call the following ioctl function. */
int i_en = WDIOS_ENABLECARD;
ioctl( fd, WDIOC_SETOPTIONS, &i_en);
```

Get the current timeout value:

```
/*get the current timeout value the driver used*/
int timeout = 0;
ioctl( fd, WDIOC_GETTIMEOUT, &timeout);
```

详细请参考 <BSP_PATH>/source/demo/watchdog 目录下的代码。

3.9.2.3 GPIO 编程

RSB-4220 有 8 个 GPIO. 请参考 <BSP_PATH>/source/demo/gpio 目录源码
用法如下:

```
# ./gpio 200 out 1
```

注! “200” 表示 *GPI00*, 即 200~207 相应的表示 *GPI00~GPI07*
 “out” 表示设定成输出
 “1” 表示设置输出值为 1, 即高电平。

3.9.2.4 CAN 编程

请参考 <BSP_PATH>/source/demo/can_test 目录源码

注! 发送代码, 请参考 *can_write.c* C 代码文件。
 接收代码, 请参考 *can_read.c* C 代码文件。

3.9.2.5 背光调节编程

请参考 <BSP_PATH>/source/demo/brightness 目录源码

Brightness 提供 sys 目录接口, 我们可以通过下面的 sys 文件获得背光信息:

```
/sys/class/backlight/pwm-backlight/brightness
```

通过下面命令设置背光

```
# echo "20" > /sys/class/backlight/pwm-backlight/brightness
```

注! 背光有效值为 1~100



通过下面命令获得当前的背光设定值:

```
# cat /sys/class/backlight/pwm-backlight/brightness
```



Enabling an Intelligent Planet

www.advantech.com.cn

使用前请检查核实产品的规格。本手册仅作为参考。

产品规格如有变更，恕不另行通知。

未经研华公司书面许可，本手册中的所有内容不得通过任何途径以任何形式复制、翻印、翻译或者传输。

所有其他产品名或商标均为各自所属方的财产。

© 研华公司 2015