智龙开发板手册

——基于龙芯 1C 的嵌入式开发板

开源龙芯社区是一群龙芯爱好者聚集的虚拟社区。	文档状态：	测试 Bate
	文档版本：	V2.0
	手册作者：	开源龙芯社区
	发布日期：	2015-10-21

版本历史

版本	日期	备注
V1.0	2015-05-30	基于智龙 V1.0，创建手册
V2.0	2015-10-21	结合智龙开发者的经验和龙芯官方 1C300B 开发板手册编写

目录

1 硬件篇

主要介绍龙芯 1C 芯片以及智龙开发板相关硬件设计

图 龙芯 1C300A

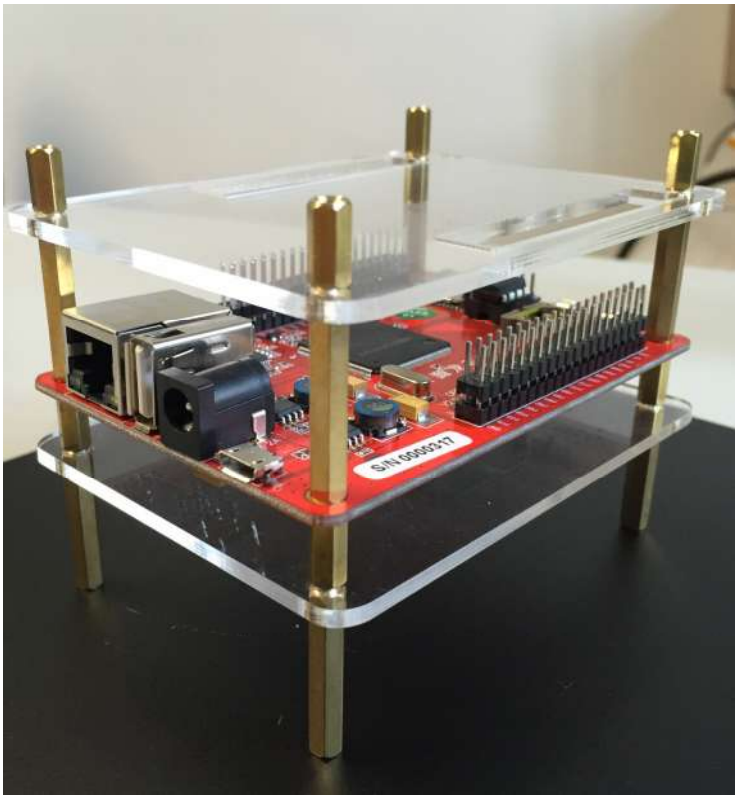


图 智龙 V2.0 开发板

1.1 龙芯 1C 芯片介绍

龙芯 1C300(以下简称 1C)芯片是基于 LS232 处理器核的高性价比单芯片系统，可应用于指纹生物识别、物联传感等领域。 龙芯 1C 是一款实现 MIPS32 兼容且支持 EJTAG 调试的双发射处理器，通过采用转移预测、寄存器重命名、乱序发射、路预测的指令 CACHE、非阻塞的数据 CACHE、写合并收集等技术来提高流水线的效率。1C300 有两个型号：1C0300A 和 1C0300B。关于这两款芯片的区别，前者为老款，后者为新款，老款不支持引脚的第五复用，新款支持。除了第五复用以外，1C0300B 串口增加到 12 个双线串口，可复用管教配置成 2 个全功能串口（串口 0，串口 8），或者 2 个四线串口（串口 0、串口 1），并且提供了对红外和智能卡的支持。修复 sdio 控制器 bug，去掉 ac97 控制器，新增一个高精度定时器，变动一些管教复用的映射关系。1C 包含浮点处理单元，可以有效增强系统浮点数据处理能力。

1C 的内存接口，支持多种类型的内存，允许灵活的系统设计。支持 8-bit SLC NAND 或 MLC NAND FLASH，提供高容量的存储扩展接口。 1C 为开发者提供了丰富的外设接口及片上模块，包括 Camera 控制器， USB OTG 2.0 及 USB HOST 2.0 接口， AC97/I2S 控制器， LCD 控制器， ADC 控制器， 高速 SPI 接口，全功能 UART 接口等，提供足够的计算能力和多应用的连接能力。

龙芯 1C 内部采用多级总线结构。处理器核、内存控制器、图形显示控制器、CAMERA 接口模块和 AXI_MUX 使用交叉开关互连。 OTG、 MAC、 USB、 DMA 控制器、 SPI 通过 AXI_MUX 连接至交叉开关。 低速外设（ I2C、 I2S、 PWM、UART 等） 通过 AXI2APB 连接至交叉开关。

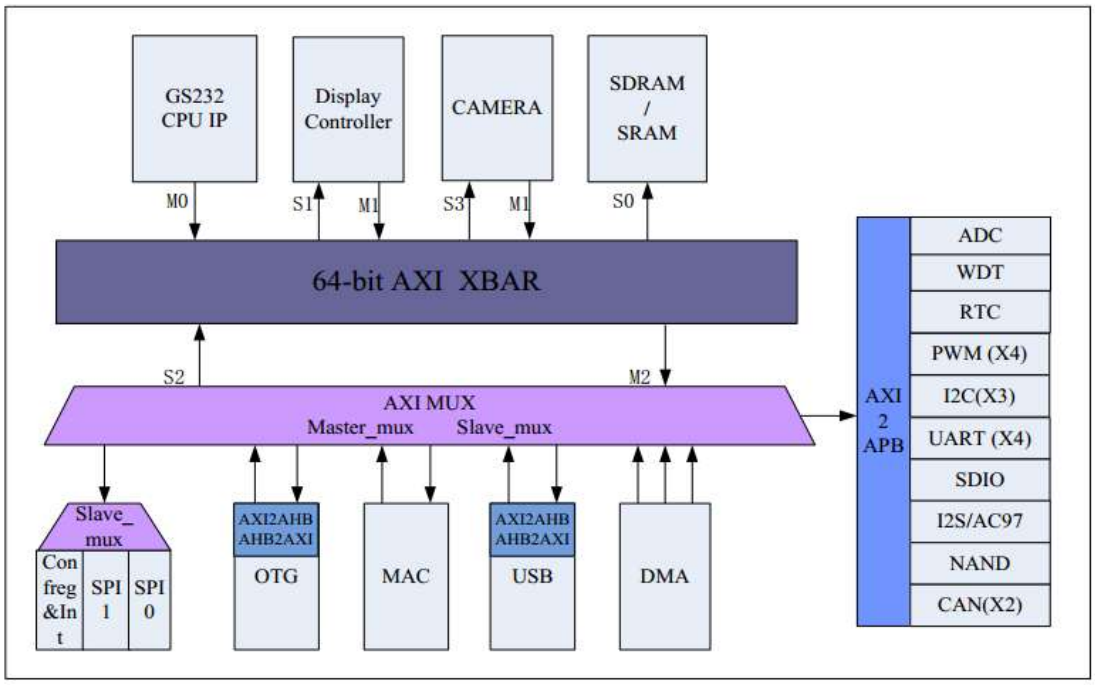


图 1.1 龙芯 1C 内部结构图

处理器内核：

- 单核心 LS232， MIPS32 指令集兼容， 主频 300MHZ

- 支持高效双发射（一个时钟节拍执行两条指令）技术
- 支持寄存器重命名、动态调度、转移预测等乱序发射、乱序执行技 术
- 五级流水线（取指、译码、发射、执行并写回、提交）微体系结构 16KB 数据 cache 和 16KB 指令 cache
- 集成 64 位浮点处理部件，支持全流水的 64 位浮点加法和浮点乘法 运算，硬件实现浮点除法运算

1.2 智龙开发板介绍

“开源龙芯主板”（即智龙开发板）是一款以开源方式推广的龙芯嵌入式主板，在和“树莓派”一样尺寸的小电路板上集成了龙芯 1C0300A ，网口、USB 口、电源，SD 卡插槽和 RTC 时钟等主要部件，可以运行嵌入式 Linux 系统和 RT-Thread 实时操作系统，实现各种应用。PCB 电路板尺寸做到了和“树莓派”一样大，如一张明信片大小。接口丰富： 最大可复用 12 路串口，在物联网应用中可满足较多的串口数据和串口传感器通讯需求，同时集成网口可实现多路串口设备接入互联网。而且龙芯 1C 芯片价格便宜，性价比高，适合规模化推广。

主板可完全手工焊接，方便 DIY： 龙芯 1C 芯片采用 QFP 封装，芯片管脚外露适合电子工程师自主焊接，而“开源龙芯主板”只需要一个熟练的焊工，一个晚上就可以焊好所有的元器件。本土厂家支持大，芯片可定制性强： 从长远来看，龙芯芯片是自主的 CPU 内核，其定制芯片设计能力强，授权费低，从设计应用到产品成本来看都是非常不错的选择。

表 1.1 智龙开发板参数

名称	参数
CPU	LS232，1C0300A，252MHz
内存	32M
Flash	128M，Nand flash，V2.0 增加了一块 DIP8 的 4M flash
接口	USB，网口，OTG，TTL
外部存储	TF 卡槽
系统时钟	RTC
输入电压	5V
扩展	V2.0：2*40 双排针，V1.0:2*38 双排母
主板尺寸	V2.0：10*7cm，V1.0：8.5*5.5cm

智龙开发板从首发到现在一共有两个版本：智龙 V1.0 和 V2.0。V1.0 主板上有网卡芯片、龙芯 1C、SDRAM、Nand Flash 四块芯片。V1.0 的接口 RJ45、USB、电源接口、OTG、双排母、六针 eJTAG、三针调试串口、RTC 电池插座。

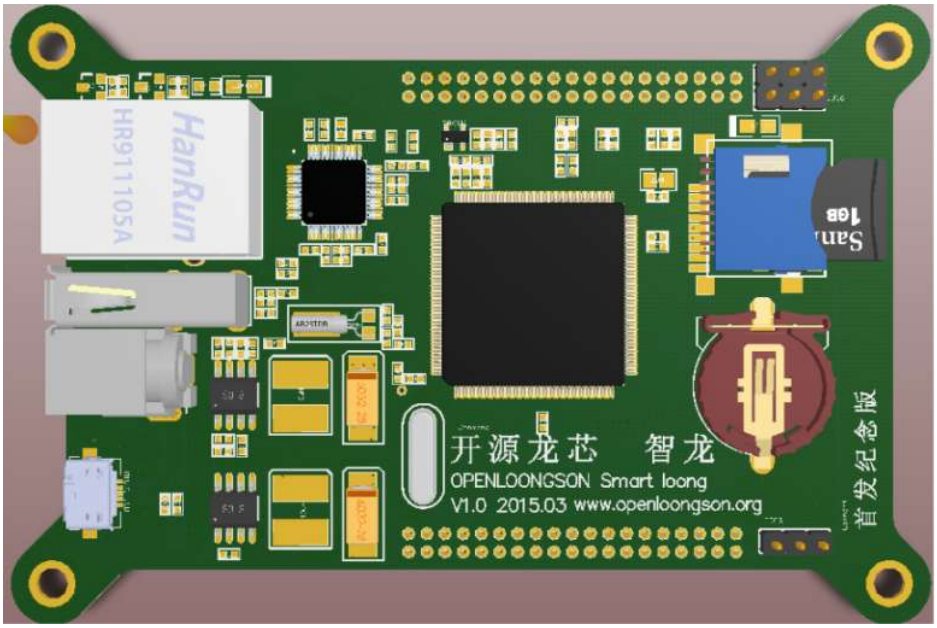


图 左边是智龙 V2.0，中间是智龙 V1.0，右边是树莓派 1.0。

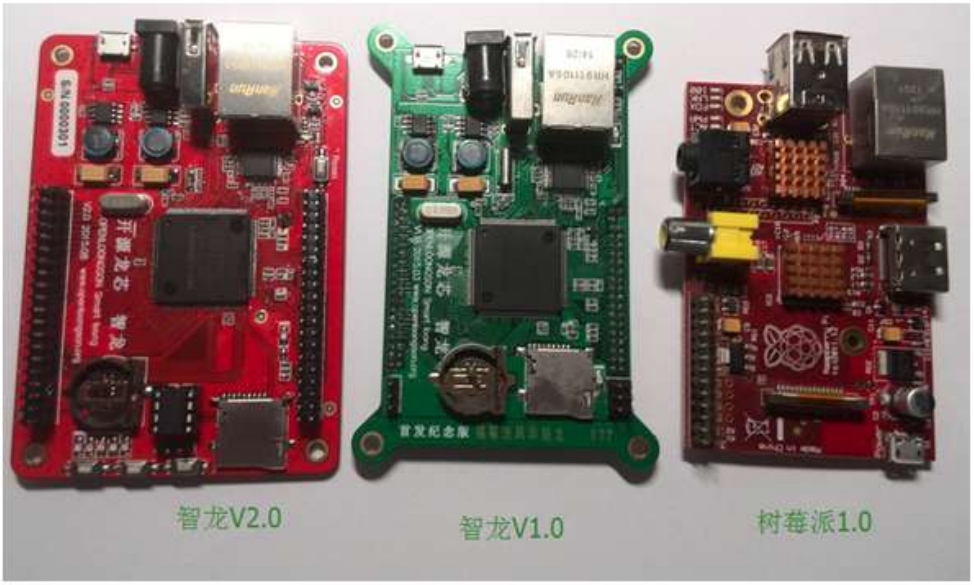


图 智龙开发板和树莓派

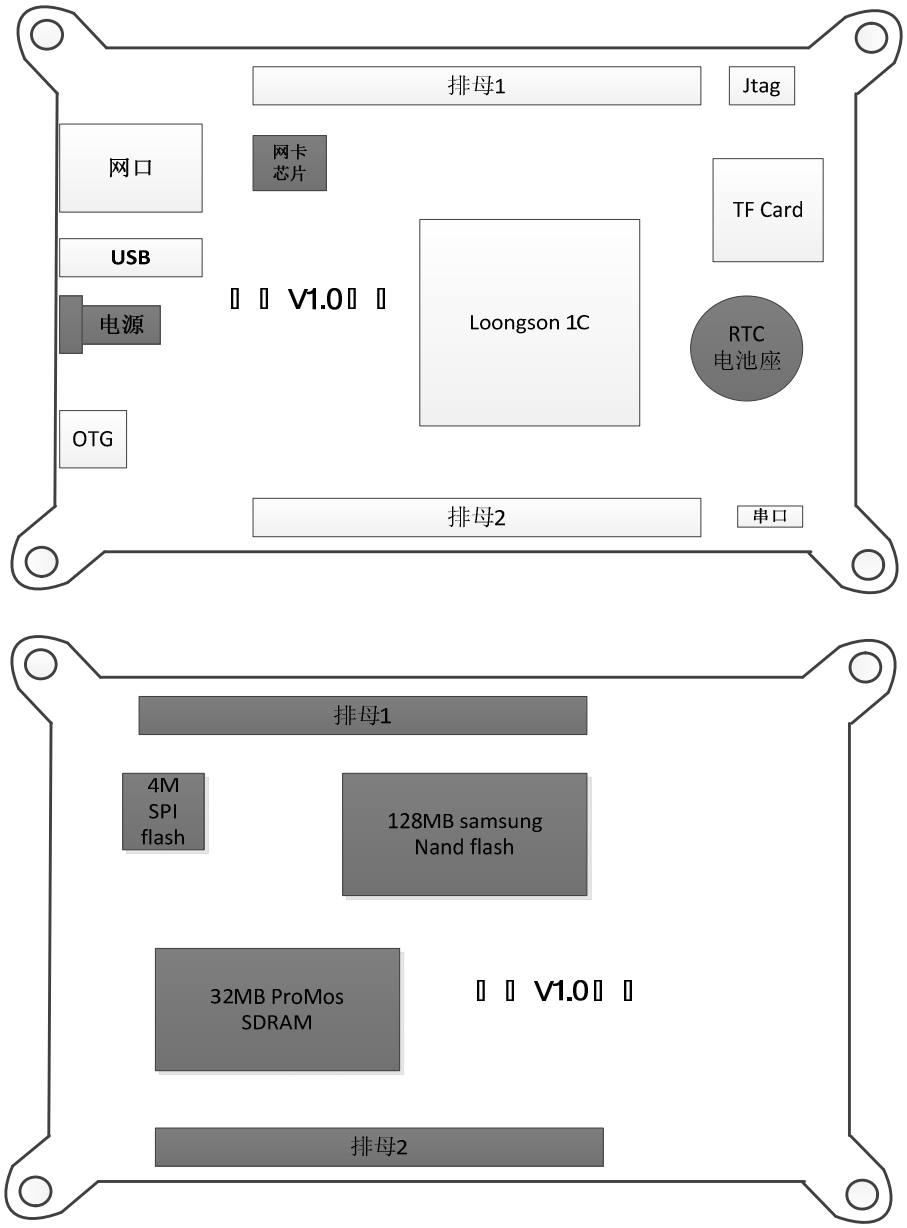
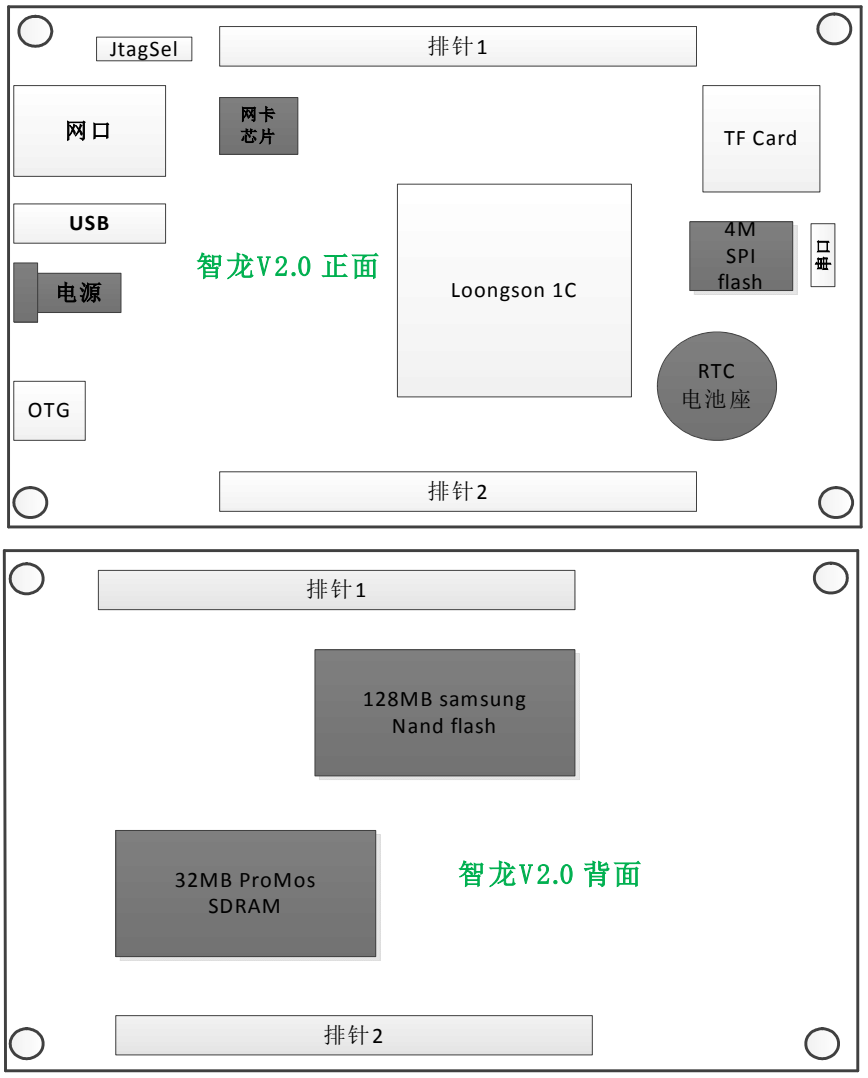

排母1
Jtag
网口
网卡
芯片
TF Card
USB
V1.0
电源
Loongson 1C
RTC
电池座
OTG
排母2
串口
排母1
4M
SPI
flash
128MB samsung
Nand flash
32MB ProMos
SDRAM
V1.0
排母2

图 智龙 V1.0 的框图



d，

图 智龙 V2.0 框图

智龙 V2.0 和智龙 V1.0 区别：

- 尺寸大小，V2.0 是 10*7cm，V1.0 是 8.5*5.5cm；
- 扩展接口，V2.0 是 2*40 双排针，V1.0 是 2*38 双排母；
- SPI flash，V2.0 将 4M flash 由贴片的换成了 DIP8 的插针芯片，且位置由背面换到了正面；
- 按键，V2.0 增加了四个按键，其中一个复位键；
- LED，V2.0 增加了五个 LED；
- JTAG，V2.0 增加了跳线，在不用 JTAG 的情况下去掉跳线帽，这样方便复用串口和 GPIO 口；

如图 1.3 所示，V2.0 针对 V1.0 的设计做了更改。

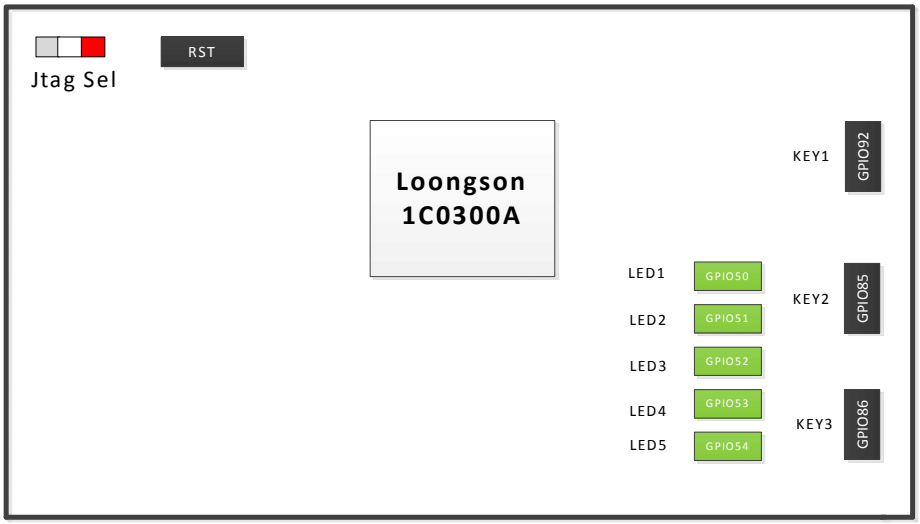


图 智龙 V2.0 增加了按键和 LED

1.3 智龙开发板硬件接口

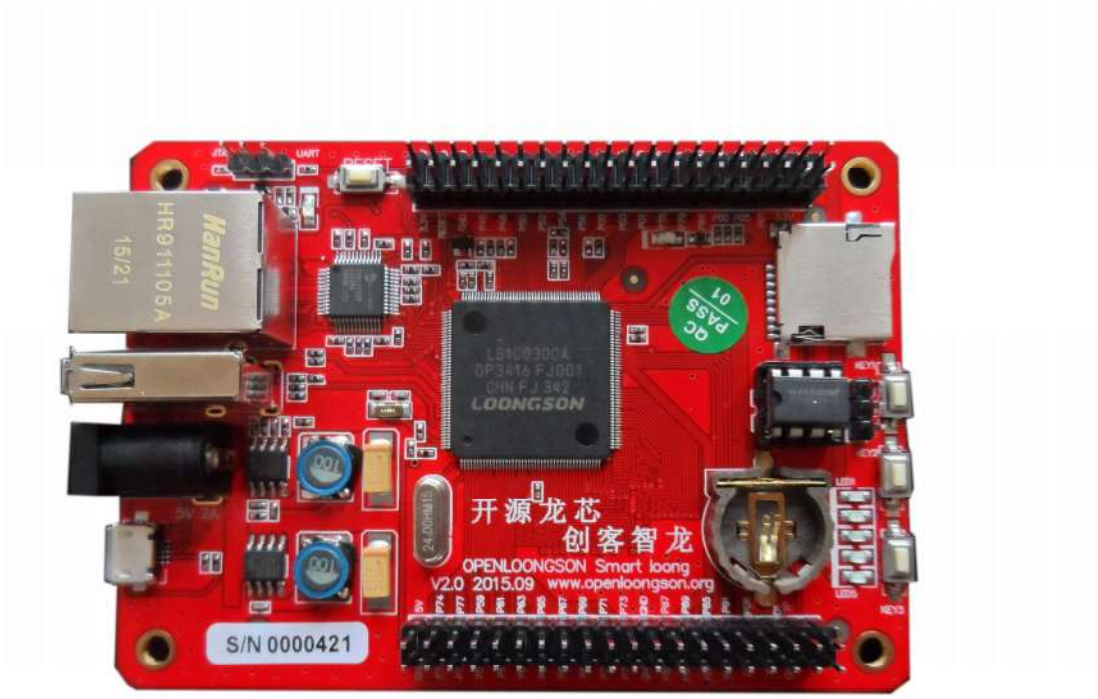


图 智龙 V2.0 正面

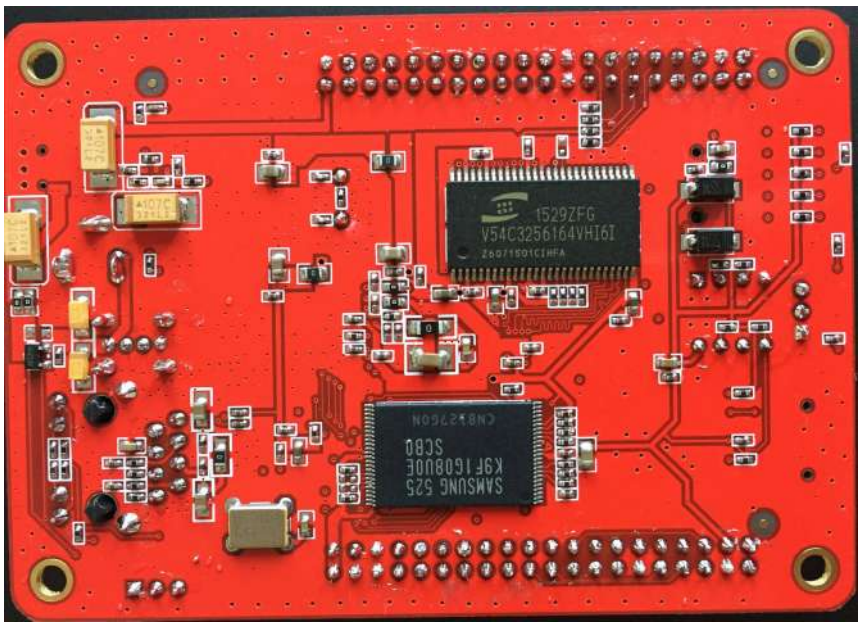


图 智龙 V2.0 背面

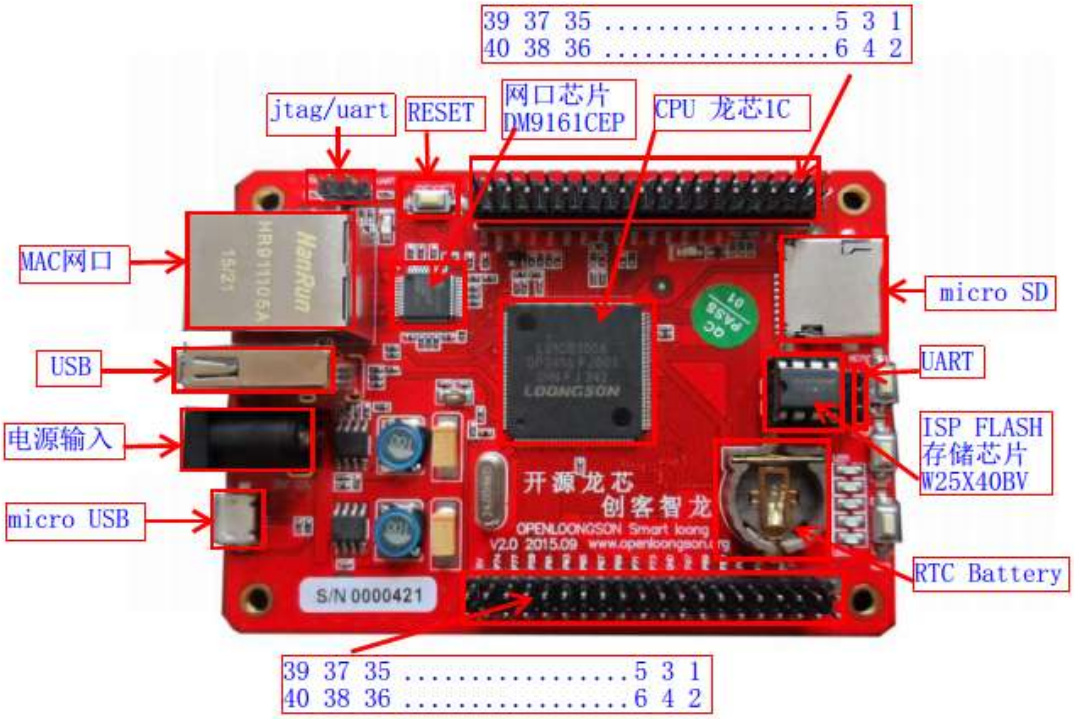


图 智龙 V2.0 标识（正面）

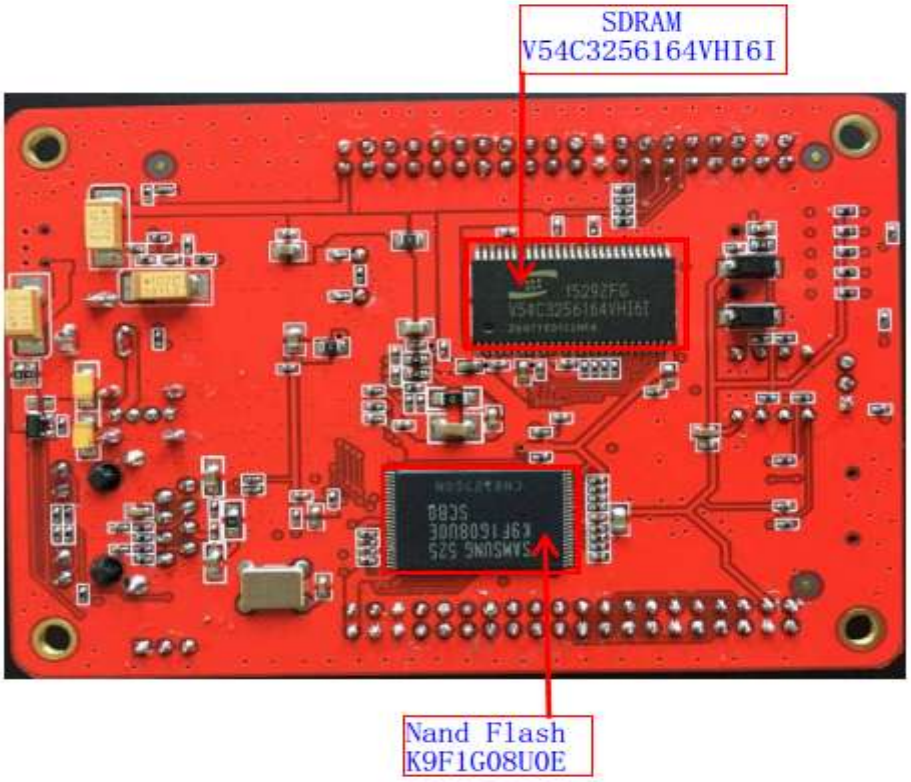


图 智龙 V2.0 标识（背面）

SDRAM 控制器

- SDRAM 接口，工作频率 45~133MHz
- 支持 8/16 位并行数据总线宽度
- 支持自动刷新和自刷新功能，支持页面模式

在智龙主板中运行程序的内存是 32MB，型号是 ProMos 的 V54C3256164VH。

SRAM/NOR FLASH 控制器

- SRAM 以及 NOR Flash 直连接口，工作频率 66~133MHz
- 支持静态存储器片选引脚，可以单独配置
- 支持 8bit/16bit 并行数据总线宽度

NAND 控制器

- 最大支持单颗容量为 4GB NAND FLASH
- 支持 512 字节、 2K 字节页、 4K 字节页和 8K 字节页类型 FLASH
- 硬件 ECC 生成、检测和指示（软件纠错）
- 支持 FLASH 数据读取速度 8~10MB/S，写入速度 5MB/s
- 支持从 NAND Flash 启动
- 支持小尾端模式

在智龙开发板中 V1.0 的 nand flash 为 128MB。V2.0 将 4M flash 由贴片的换成了 DIP8 的插针芯片，且位置由背面换到了正面。PMON 系统是通过 SPI 口从 4M flash 中加载。如果 PMON 损坏了，只需要拔下 SPI flash，用烧写器进行烧写或者直接跟换 SPI flash。SPI flash 有点类似电脑中 BIOS 芯片。SPI flash 是 Nor flash。

Nor flash 和 Nand flash 的区别：

Nor flash 比 Nand flash 的读取的速度稍快一点，Nand flash 的写速度比 Nor flash 快很多。

Nor flash 拥有足够的地址空间。CPU 可以直接将其数据读到内存上。就 Nand flash 而言，只有器控制器拥有独立的地址空间，内部数据无法直接读到内存，但是它可以支持大容量的数据芯片。比如龙芯 1C 可以在 SPI0 和 SPI1 上分别挂 8MB 和 4MB 的 Nor flash，但是可以在 nand 接口上挂 4 片单片容量 8GB 的 nand flash。Nor flash 的控制器很简单，地址空间充足，CPU 可直接访问芯片中的任何数据。Nand flash 的控制器相对复杂，需要其控制软件加载完成后才能正确读写数据。

表 1.2 智龙数据存储模块

名称	品牌	型号
Nand Flash	Samsung	128MB,K9F1G08U0E-SCB0
SPI Flash	Winbond	4MB,W25X40BVAIG
SDRAM	ProMos	32MB,V54C3256164VH

龙芯 1C 的启动方式有多种，可通过配置某些管脚来进行选择启动方式。如表 1.3 所示有 SPI Flash、Nand Flash 和 SDIO 三种启动方式。智龙是从 SPI Flash 中加载 PMON 的。

表 1.3 不同的系统启动方式：

系统启动方式	NAND_D4（1）	NAND_D5（2）
保留	0	0
SPI FLASH 启动	1	0
NAND FLASH 启动	0	1
SDIO 启动	1	1

由下图可知，龙芯 1C 的地址空间分为三级，包括一级 AXI 、AXI_MUX 、APB 。根据 CPU 的地址空间分配情况，SDRAM（即内存）最大可支持 256MB。智龙开发板上实际内存是 32MB。SPI0 分配到的数据地址空间有 8MB，即最大可支持 8MB 的 SPI flash（Nor flash），智龙开发板上实际是 4MB，CPU 无须软件的情况，直接把这 4MB 的数据搬到内存。在 APB 下，Nand 设备的地址空间只有 16KB，

这不是数据空间，因为 1C 最大可以支持 32GB 的数据存储。CPU 不可能为 Nand flash 每一个数据都分配一个内存地址，否则这需要 32GB 内存，这是不可实现的。Nand flash 需要通过软件慢慢的把数据读到内存中，比如 linux 内核数据。

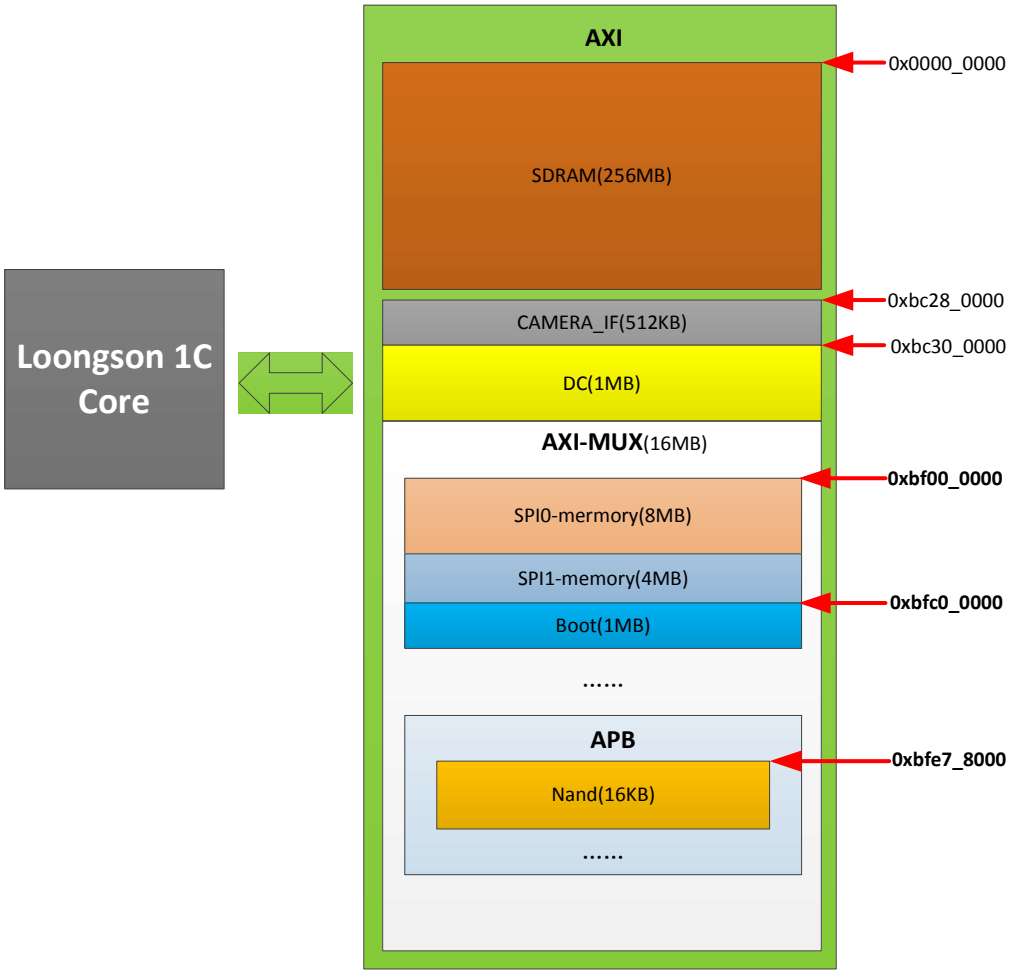


图 龙芯 1C 的 CPU 地址分配

表 AXI 各模块地址分配

地址空间	设备	说明
0x0000_0000 – 0x0fff_ffff	SDRAM	256MB
0xbc28_0000 – 0xbc2f_ffff	CAMERA_IF	512K
0xbc30_0000 – 0xbc3f_ffff	DC	1MB
0xbf00_0000 – 0xbfff_ffff	AXI MUX	16MB

表 AXI-MUX 各模块地址分配

地址空间	设备	说明
0xbd00_0000 – 0xbd7f_ffff	SPI0-memory	8MB

0xbe00_0000 – 0xbe3f_ffff	SPI1 -memory	4MB
0xbfc0_0000 – 0xbfcf_ffff	Boot	1MB，根据系统启动方式映射到 SPI 或 NAND
0xbf00_0000 – 0xbfff_ffff	AXI MUX	16MB
0xbfe4_0000 – 0xbfe7_ffff	APB-devices	256KB

表 APB 模块地址分配

地址空间	设备	说明
0xbfe7_8000-0xbfe7_bfff	NAND	16KB

表 1.不同的 NAND FLASH 颗粒容量：

NAND FLASH 颗粒容量	NAND_D6（3）	NAND_D7（4）
容量<=256MB（512 Bytes 页）	0	0
容量=512MB （512 Bytes 页）	1	0
容量= 1GB （2K 页）	0	1
容量>= 2GB （2K 页）	1	1

智龙开发板采用三星的 Nand Flash，型号为 K9F1G08U0E-SCB0，容量为 128MB。在智龙中，SPI Flash 是默认的启动的方式，即 PMON 的文件是烧写在 4MB SPI Flash 中的。

表 1.Nand Flash 的分区情况

分区	地址范围	大小	用途
kernel	0x00000000 - 0x00e00000	14MB	内核
os	0x00e00000 - 0x07200000	100MB	根文件系统
data	0x04000000 - 0x0f200000	14MB	数据

时钟发生器

- 1 个标准 PLL 输入接口，支持外部无源晶体作为芯片时钟输入
- 支持片内输出可配置时钟一路，供片外外设使用
- PLL 频率软件可配置

I2S 控制器

- 支持 master 模式下 I2S 输入
- 支持 master 模式下 I2S 输出
- 支持 8、 16、 18、 20、 24、 32 位宽
- 支持单声道和立体声道音频数据
- 支持(16、 22.05、 32、 44.1、 48)kHz 采样频率
- 支持 DMA 传输模式

AC97 控制器

- 可变采样率 AC97 编解码器接口（ 48KHz 及以下）
- 支持立体声 PCM 和单声道 MIC 输入
- 支持 2 通道立体声 PCM 输出
- 支持 DMA 和中断操作
- 支持 16、 18 和 20 位采样精度， 支持可变速率
- 支持 16 位、 16 个入口 FIFO 每通道

根据龙芯官方的说明，龙芯的 1C0300B 去掉了 AC97 控制器。而智龙开发板都是采用的 1C0300A，所以还是存在 AC97 的。

LCD 控制器

- 支持 16/24 位像素模式
- 支持 RGB444/555/565/888 显示输出
- 支持 1024x768、 800x600、 640× 480、 320× 240 分辨率
- 支持 DMA 传输模式

Camera 接口

- 支持 ITU-R BT.601/656 8 位输入
- 支持 RAW RGB、 RGB565 及 YUV4:2:2 数据输入
- 支持 YUV、 RGB888、 RGB0888、 RGB565 输出
- 支持 320x240 和 640x480 分辨率缩放
- 支持最大 2Kx2K 分辨率输入，分辨率可配置
- 支持 DMA 传输模式

MAC 控制器

- 支持 10/100Mbps PHY 器件，包括 10 Base-T、 100 Base-TX、 100 Base-FX 和 100 Base-T4;

- 完全兼容 IEEE 标准 802.3
- 完全兼容 802.3x 全双工流控和半双工背压流控
- 支持 VLAN 帧
- 支持 DMA 传输模式
- 支持标准的媒体独立接口（ MII）
- 支持标准的简化 MII 接口（ RMII）可连接外部 PHY 芯片

USB2.0 控制器

- 1 个 USB OTG 2.0 控制器

- 1 个 USB HOST 2.0 控制器
- 支持高速和全速模式
- 支持 DMA 传输模式
- 兼容 USB Rev 1.1 、 USB Rev 2.0 协议

SPI 控制器

- 支持两路独立 SPI 接口，每路 SPI 接口均支持 4 个片选
- 遵循串行外设接口 (SPI)规范
- 支持同步、串行、全双工通信
- 支持 SPI 主模式
- 每次传输 8 位
- 支持查询、中断传输模式
- 支持 SPI nor flash 启动
- 支持 SPI 接口双向输入输出，最高数据传输速度 24~96 Mbps
- 支持最低速率通讯要求，速率达 25KB 以下，方便匹配特殊设备。

龙芯 1C300A 有两个 SPI 口，其中 SPI0 是智龙主板 SPI Flash 和 TF 卡的复用。

I2C 控制器

- 三路标准 I_2C 总线接口
- 支持主、从、或主/从模式配置
- 总线的时钟频率可编程

UART 控制器

- 支持两个全功能串口，其中全功能串口 0 可复用为 4 个两线串口或 2 个 4 线串口，支持智能卡协议
- 基于中断操作的 RxD0， TxD0， RxD1， TxD1， RxD2 和 TxD2；

- 基于中断操作的 RxD0， TxD0， RxD1， TxD1， RxD2 和 TxD2；
- UART 通道 0， 1 和 2 带 IrDA 1.0
- UART 通道 0 和 1 带 RTS0， CTS0， RTS1 和 CTS1

龙芯 1C0300A 是支持复用 12 个串口，在智龙开发板中，暂时可以复用 uart0~uart3 这样 4 个串口。

GPIO

- 最多支持 105 个 GPIO
- 所有 GPIO（启动和系统配置除外）在复位后默认为输入
- 所有 GPIO 支持中断功能
- 每个 GPIO 管脚均支持电平触发、边沿触发模式，可独立配置
- GPIO 管脚速率可达 4MHz

PWM 控制器

- 4 路 32 位可配置 PWM 定时器
- 支持定时器功能
- 支持计数器功能

龙芯 1C0300A 有四个 PWM，其中两个可以直接使用，另外两个需要复用其它脚。由于有四个 PWM，智龙开发板很适合做 3D 打印机、飞控、CNC、四驱电动车、机械手臂等的运动控制主板。开源龙芯社区中有“龙印”的项目就是基于智龙开发 3D 打印机。欢迎大家关注此项目。

RTC

- 计时精确到 0.1 秒
- 支持外部无源晶体作为 RTC 时钟输入
- 支持外部电池供电运行，断电后由电池供电
- 专门的电源管脚，可以与电池或者 3.3V 主电源相连
- 提供秒、分、时、日、月、年

智龙主板设计了 RTC 的电池座，电池型号是 CR1220 纽扣电池，这个电池不太常见，大家购买的是要看对型号。

CAN 控制器

- 2 路独立 CAN 控制器

- 兼容 CAN2.0A 和 CAN2.0B 协议（ PCA82C200 兼容模式中的无源 扩展帧）
- 支持 CAN 协议扩展
- 位速率可达 1Mbits/s

CAN 总线技术是物联网、汽车系统等领域红得发紫的总线技术。通过引脚复用的方式，我们可以将智龙开发板配置 CAN 总线控制器。

SDIO 控制器

- 1 路独立 SDIO 控制器

- 兼容 SD Memory 2.0/MMC/SDIO 2.0 协议
- 支持 SDIO 启动

ADC 控制器

- 采样率最高 1MHz
- 4 路 ADC 输入

- 支持 4 线和 5 线触摸屏
- 支持连续采样和单次采样
- 支持模拟看门狗

1.4 串口调试连接

智龙开发板团购套件提供一个智龙开发板和一个 USB 线。

图 1. 智龙 V1.0 和 USB 线

另外还需自行准备“USB 转 TTL 的转接小板”、“杜邦线”、网线、“对应转接板的 USB 延长线”。注意：转接板的 USB 延长线和智龙开发板的 USB 线是不一样的，两端的头子是有区别的，仔细看图 1.。

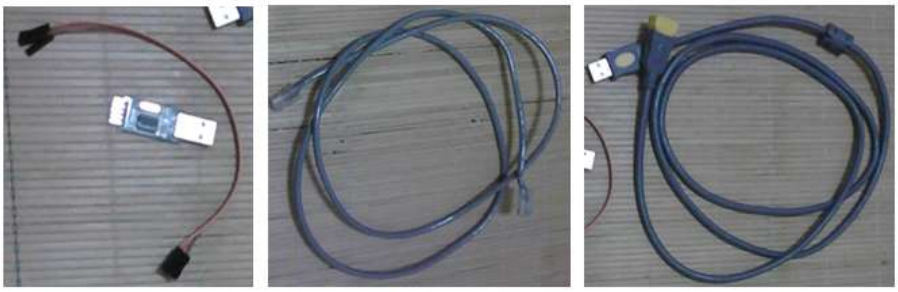


图 1. USB 转 TTL 的转接小板、杜邦线、网线、USB 延长线

串口这里要特别注意，智龙开发板的串口是 TTL 电平的，也就是直接把龙芯 1C 处理器的串口引脚引出来了，没有接 232 芯片，所以必须要外接一个 232 转接小板，由于只引出了三根线 Tx、Rx 和 GND，没有引出 3.3v 或者 5v 的电源，也就是转接小板必须自己提供电源，所以最好选择 usb 转 TTL 的 232 转接小板。因为是 USB 接口的所以供电不是问题。前面的图中已经看到了我的转接小板。除此之外还需要注意串口三个引脚的顺序

	1	2	3
智龙	GND	RX	TX
转接板	GND	TX	RX

图 1. 调试串口的接法

调试串口连接步骤如下：

Step1 同时 USB 转 TTL 的转接小板上也有这三个引脚。他们的连接顺序是：

智龙开发板的 GND ---------- USB 转 TTL 的 GND
智龙开发板的 RX ---------- USB 转 TTL 的 TX
智龙开发板的 TX ---------- USB 转 TTL 的 RX

注意智龙开发板的发送接收引脚与 USB 转 TTL 的转接小板要交叉连接。正确连接如图 1.

图 智龙 V1.0 的调试串口连接图

如果缺少 DIY 能力的，可以使用下图所示的串口调试线。杜邦线插在开发板上，其中红线为电源线不需要用，只需要其它三根线。USB 端接电脑，一般需要安装串口驱动。

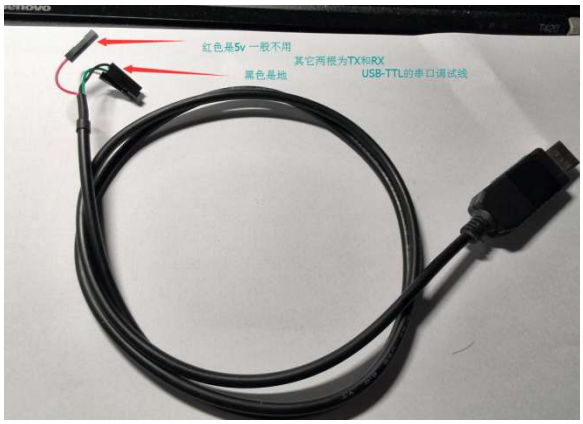


图 智龙 V2.0 调试串口连接图

Step2 在设备管理器里面可以找到具体的串口号，我的是 com7。

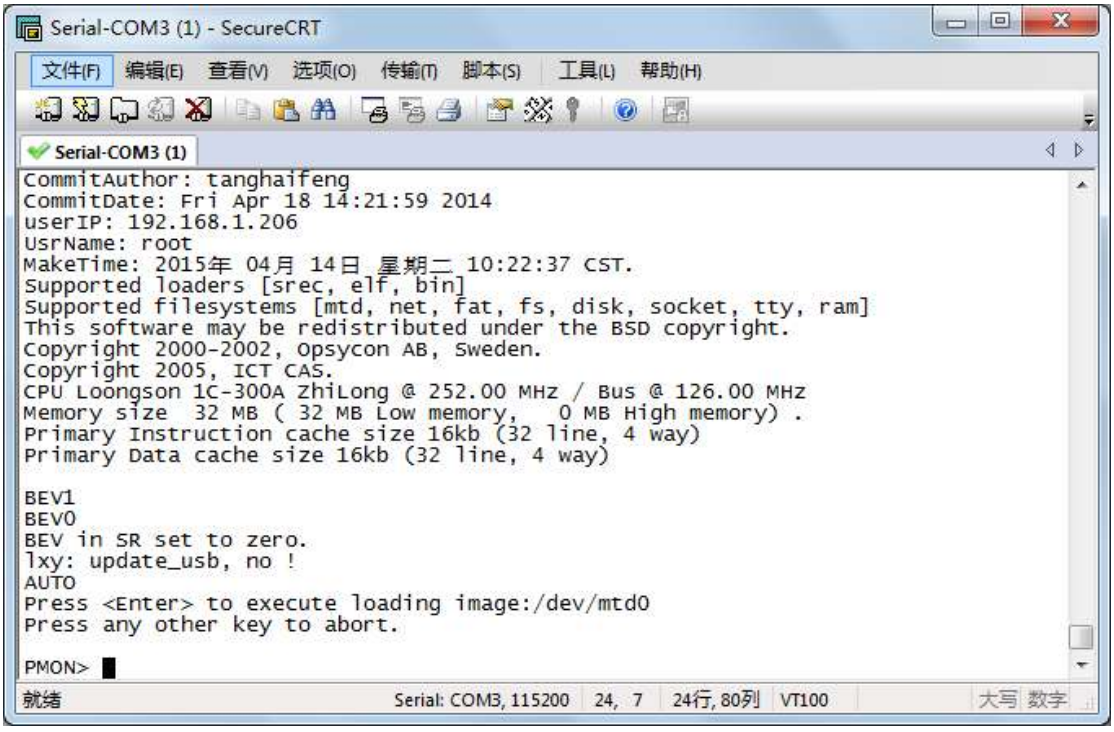
Step3 串口软件 secureCRT（安装略），波特率是 115200bps，注意流控制 RTS/CTS 等 3 个选项不选，参数设置如图 1.。

Step4 如图 1. 开发板上电，进入系统的串口打印，若上电后立即按任意键则进入 PMON(不进行任何操作，则初始化完成后系统开始运行)，如下图：

```
TCP cubic registered
TCP westwood registered
TCP htcp registered
NET: Registered protocol family 17
ls1x-rtc ls1x-rtc.0: setting system clock to 2103-03-02 12:27:1
yaffs: dev is 32505857 name is "mtdblock1" rw
yaffs: passed flags ""
VFS: Mounted root (yaffs2 filesystem) on device 31:1.
devtmpfs: mounted
Freeing unused kernel memory: 200k freed
#mount all.....
#Starting mdev.....
Processing /etc/profile.....
Done!
[root@Loongson-gz:/]#PHY: 0:13 - Link is Up - 100/Full

[root@Loongson-gz:/]#ping 192.168.1.7
PING 192.168.1.7 (192.168.1.7): 56 data bytes
64 bytes from 192.168.1.7: seq=0 ttl=64 time=5.248 ms
64 bytes from 192.168.1.7: seq=1 ttl=64 time=0.651 ms
64 bytes from 192.168.1.7: seq=2 ttl=64 time=0.550 ms
64 bytes from 192.168.1.7: seq=3 ttl=64 time=0.541 ms
^C
--- 192.168.1.7 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.541/1.747/5.248 ms
[root@Loongson-gz:/]#
```

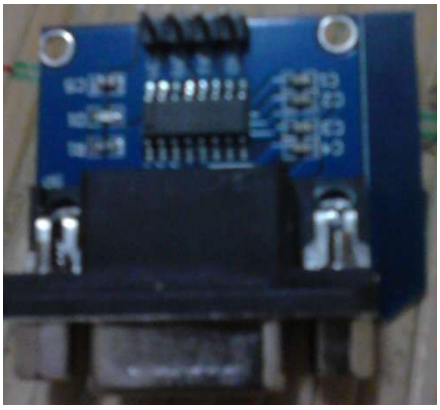
就绪



另外，举两个错误的串口接法

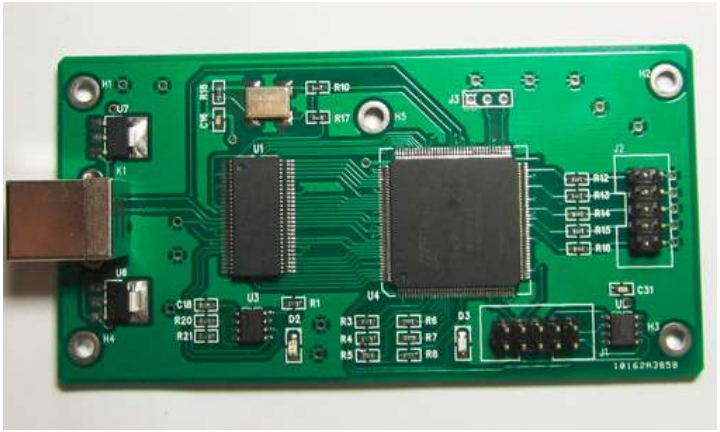
使用如下的串口线直接连接电脑串口是错的。这种串口线只能用于开发板已经有 232 芯片了，串口为 RS232 电平（不是 TTL 电平哦）。

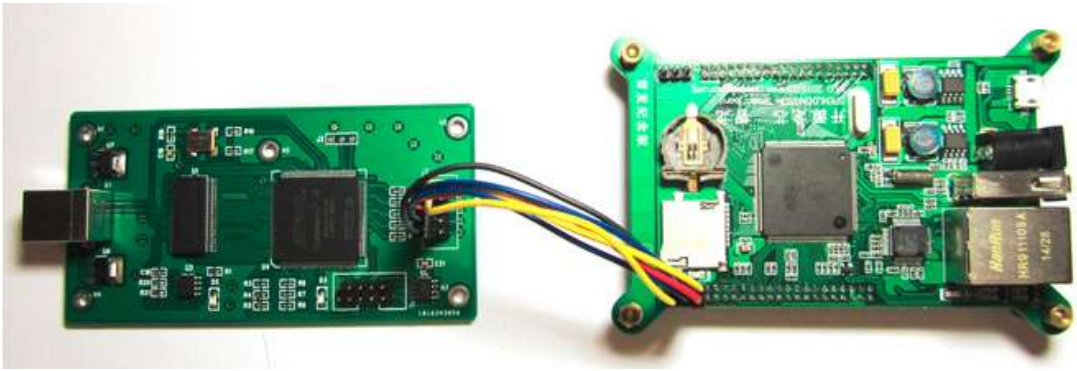
使用图 1.转接小板也不正确。这个转接小板上的 232 芯片需要外部提供电源，所以有 4 根引脚。注意 USB 转接板也有电源引脚，那个是 USB 转接小板给外面开发板提供的电源，因为智龙开发板自己通过 USB 供电了，不需要 USB 转接小板提供电源，所以 USB 转接小板上的 3.3V 和 5V 电源引脚没接，这个要注意区别。

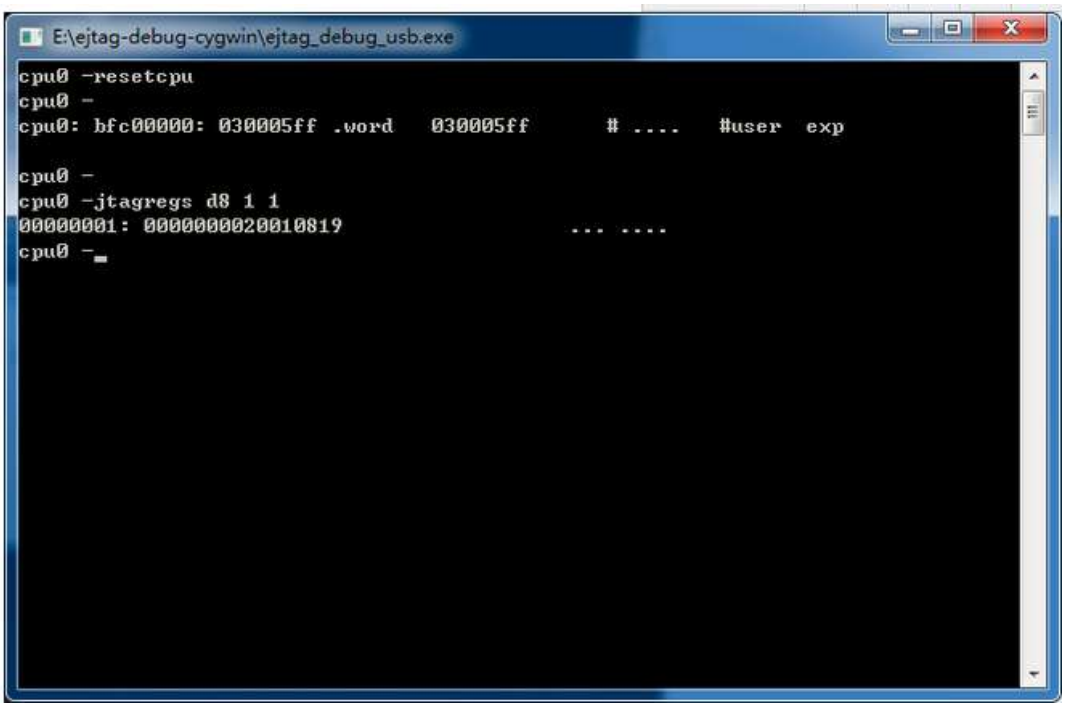


1.5 eJtag 调试系统

下图为网友设计的智龙开发板的 eJTAG 调试器。

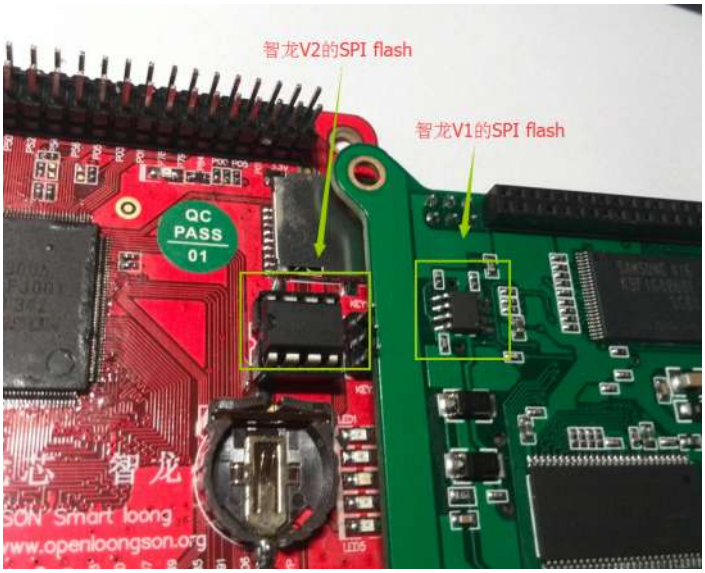






1.6 Flash 烧写 PMON 引导系统

PMON 是智龙开发板的引导系统。一旦该系统数据损坏，智龙开发板将无法启动。PMON 系统存放在智龙开发板的 SPI flash 中。如下图所示，智龙 V2.0 的 SPI flash 在 RTC 电池的旁边，V1.0 的在板子背面的贴片式芯片。



图

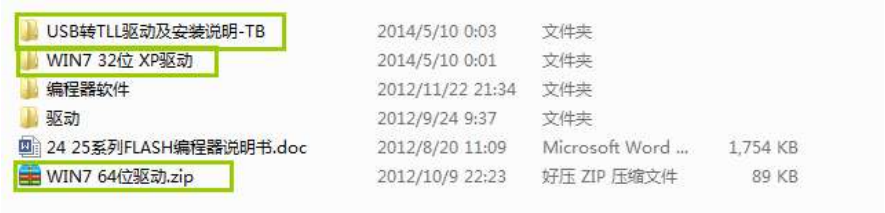
在智龙 V1.0 中，PMON 的数据损坏了，需要将芯片焊下来，在烧完 PMON 系统后再焊回去。整个过程相当复杂，适合动手能力较强的。我的建议如果出现这种情况直接返厂。如果智龙 V2.0 出现这种情况，只需要将芯片拔下来，在烧完 PMON 后再将芯片插回去即可。

Step1 准备烧写工具。

烧写工具主要分为硬件和软件。硬件为基于 CH341A 芯片的 USB 烧写器。CH341A 芯片如下图所示，此芯片经常用于烧写程序、串口数据转换等环境中，如 Arduino 的烧写是该芯片。软件是 CH341A 编程器，将软件信息贴出来的目的是希望大家尽量用正版，支持软件作者。硬件烧写器，网上 30 元以内，送驱动和烧写软件。

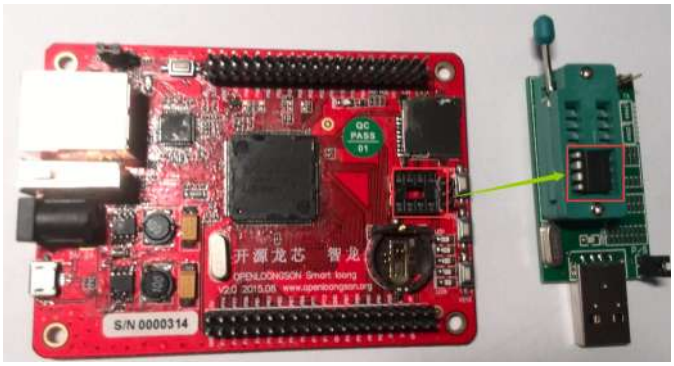
Step2 安装驱动。

驱动主要是 USB 转 TTL 的驱动和编程器芯片的驱动。安装驱动驱动的时候记得将烧写器查到电脑上。

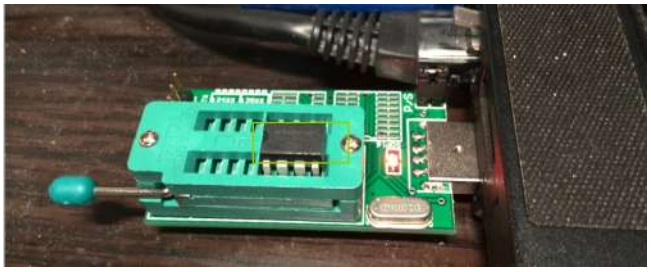


点击安装即可安装 CH341A 的驱动。

Step3 将 SPI flash 芯片拔下来查到烧写器上。



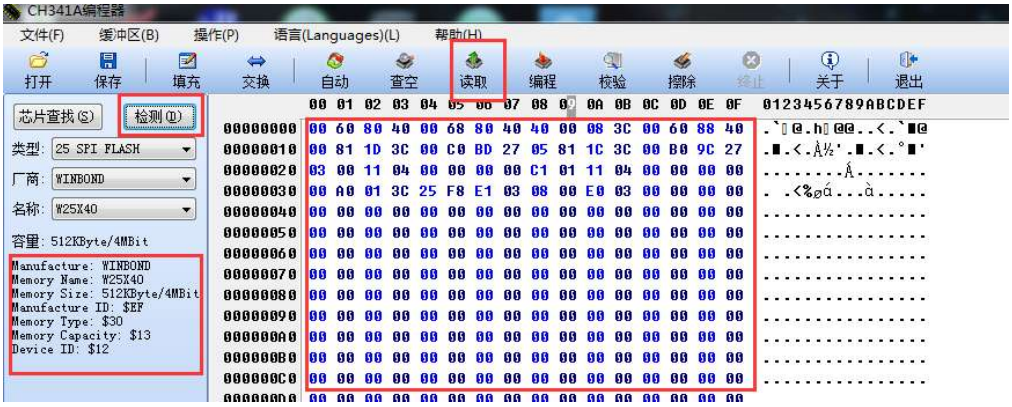
Step4 将烧写器查到电脑上。



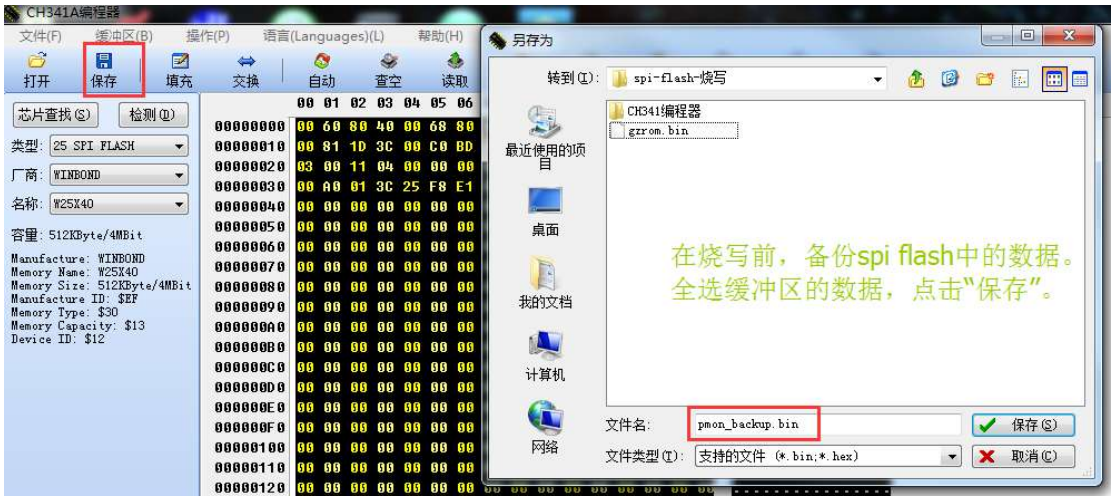
Step5 读取 SPI flash 中的数据进行备份。

影响板子启动原因有可能不是 PMON 数据损坏，而是其它硬件或系统配置原因。为了防止这种情况，我们在重写 SPI flash 之前要先备份数据。

检测芯片并读取数据。智龙开发板采用的是 winbod 的 SPI flash，型号是 v25x40，容量是 4MB。

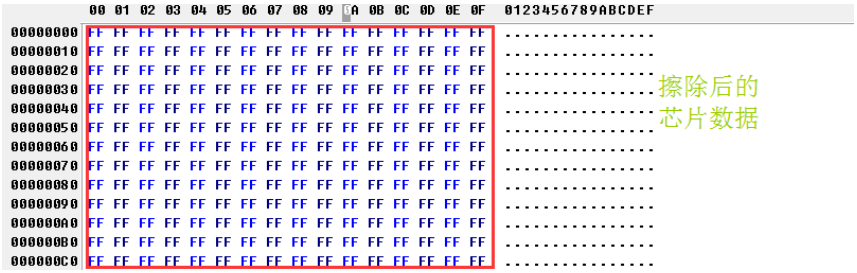


备份芯片数据。先全选缓冲区的数据，点击“保存”，保存文件名为“pmon_backup.bin”。



Step6 擦除 SPI flash 芯片内的数据。

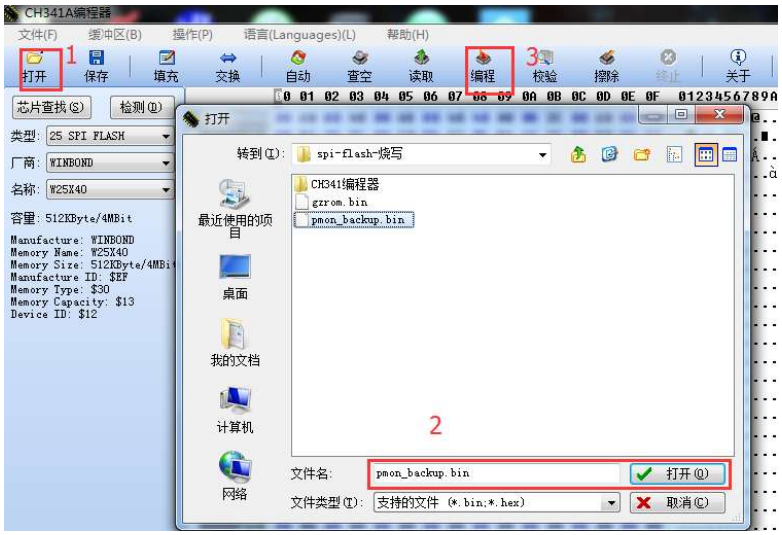
点击“擦除”。芯片完成数据清除后，点击“读取”，你会发现芯片内的数据全是“FF”。



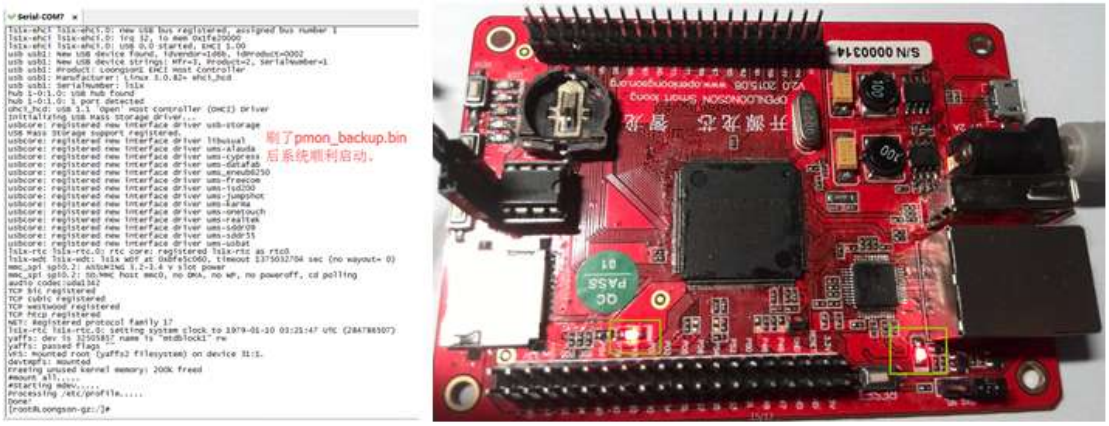
将 SPI flash 芯片插回到智龙开发板。加电后，主板无法启动，只有指示电源的 LED 亮了，在网卡附近的 LED 没有亮。

Step7 烧写 bin 文件。

点击“打开”，选择 bin 文件，可以选社区提供的“gzrom.bin”或者刚才备份的“pmon_backup.bin”。最后点击“编程”，PMON 系统就会被烧写到 SPI flash 中了。



烧写完成后，顺利进入系统，并且网卡附近的 LED 也亮了。

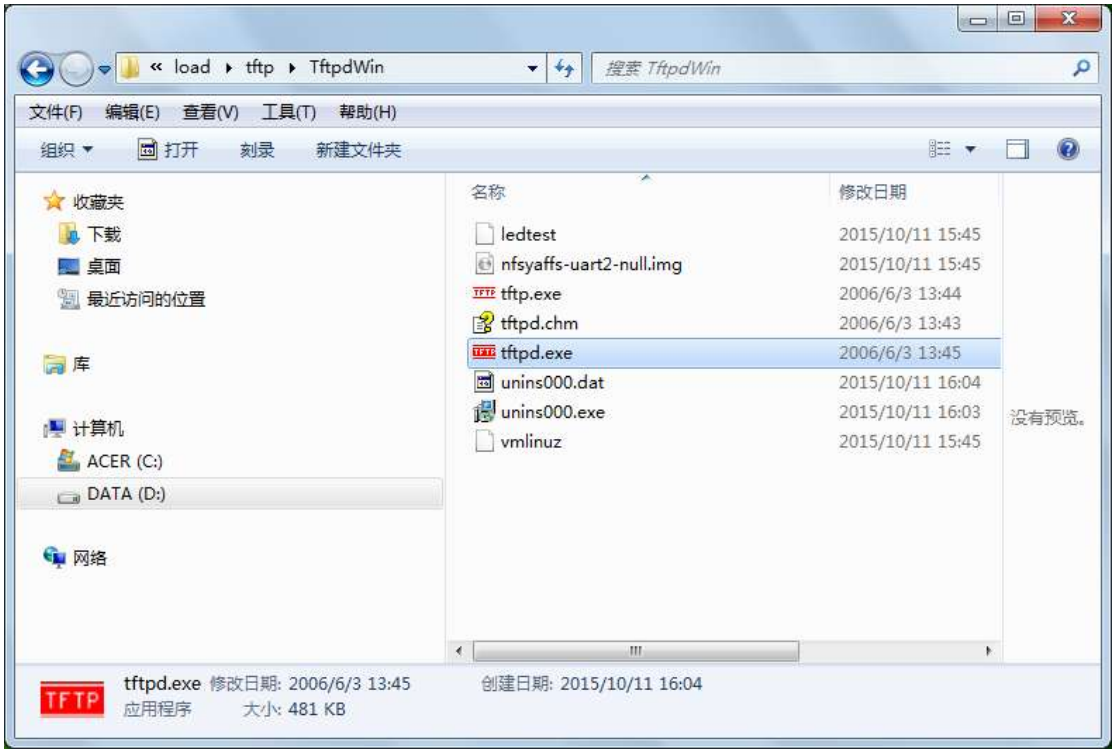


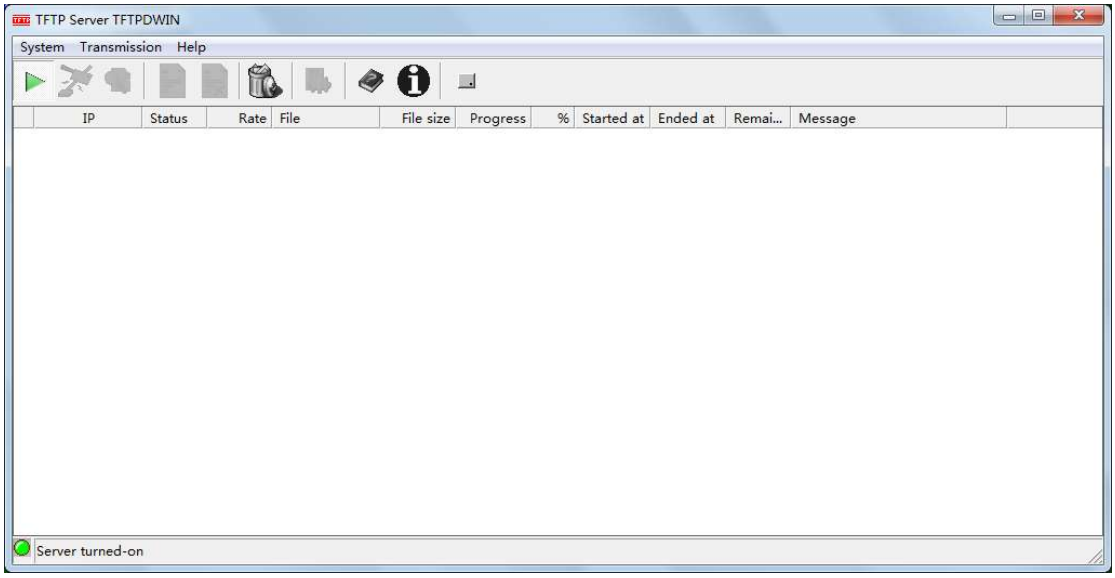
总之，有了烧写工具，对我们开发 PMON 和移植 u-boot 是很有帮助的。

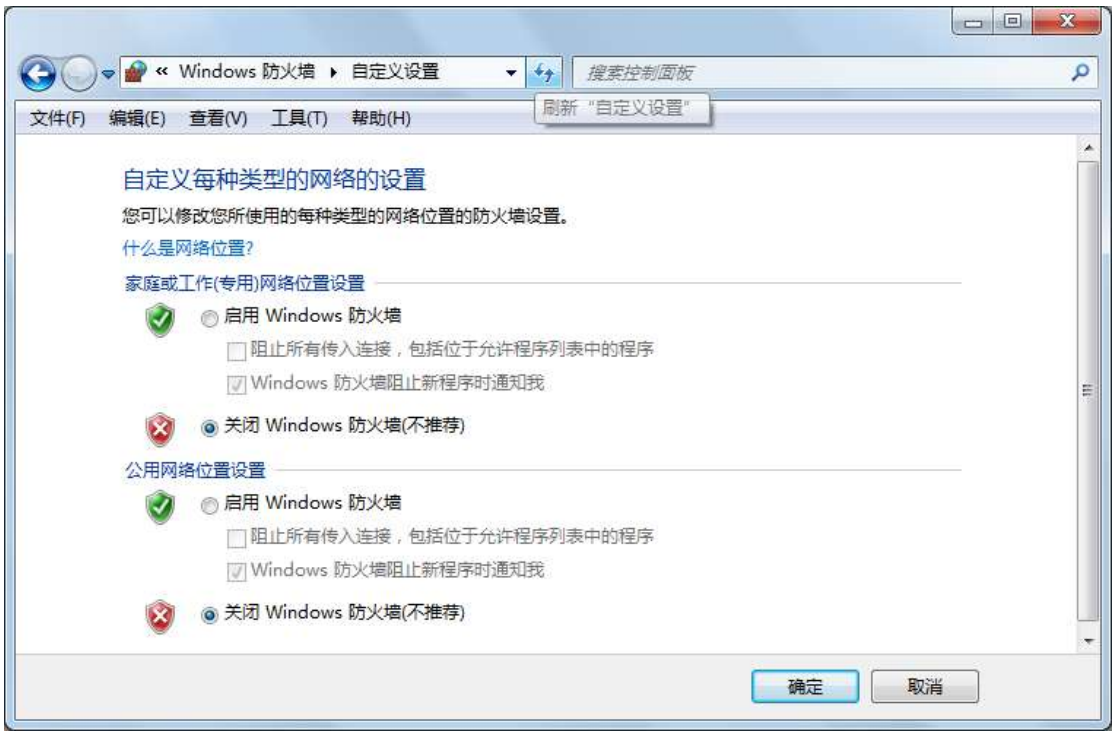
1.7 Flash 烧写 linux 系统（附带跑马灯实验）

1. 开发板连接：串口与 PC 连接；采用 USB 给开发板供电；网线 1 连接开发板与交换机，网线 2 连接 PC 与交换机（手册采用直通网线，若用户使用交叉网线，则直接用交叉网线连接开发板与 PC）如下图：

2. 打开 TFTP（安装参考 2.6.7 基于 windows），并把编译好的内核、文件系统放置于 TFTP 的安装目录下，关闭 PC 防火墙，如下图：



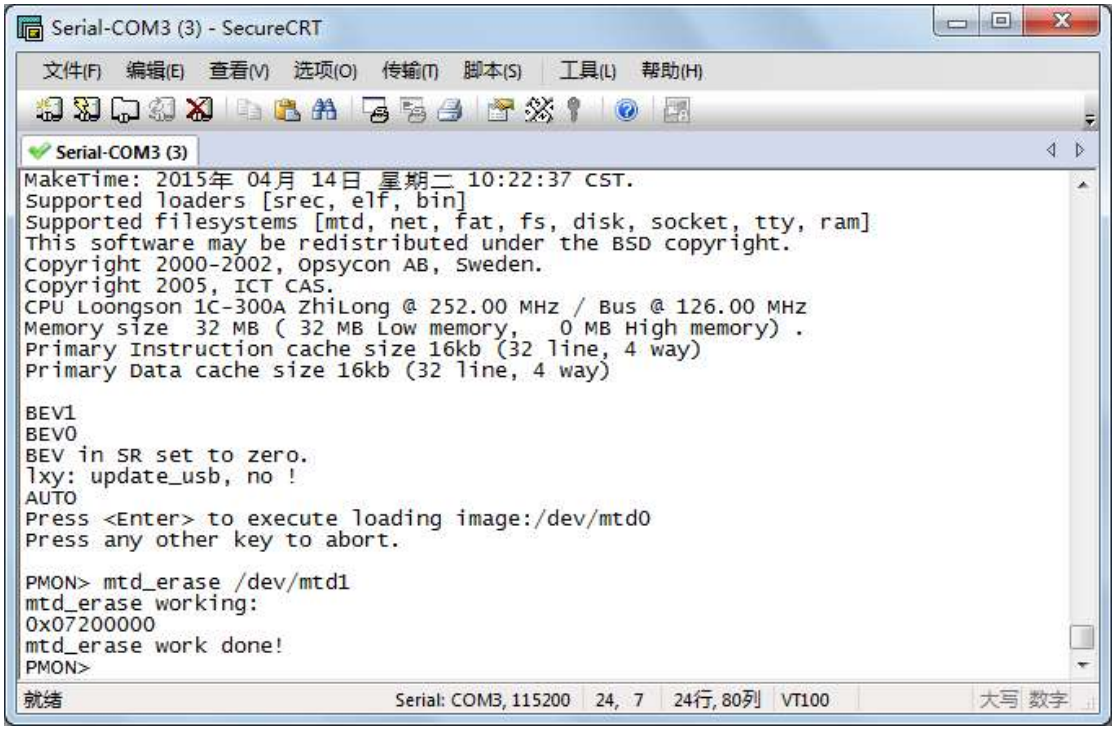




3. 进入 PMON（参考 2.6.7 基于 windows）
4. 在 PMON 中输入命令：

   mtd_erase /dev/mtd1　//擦除数据

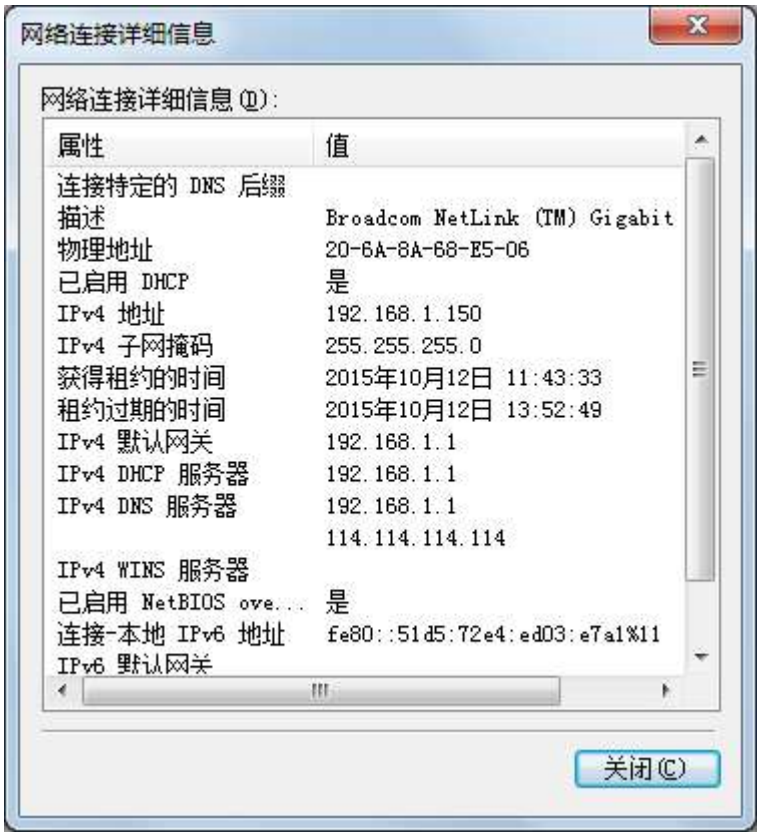
   然后回车，如下图

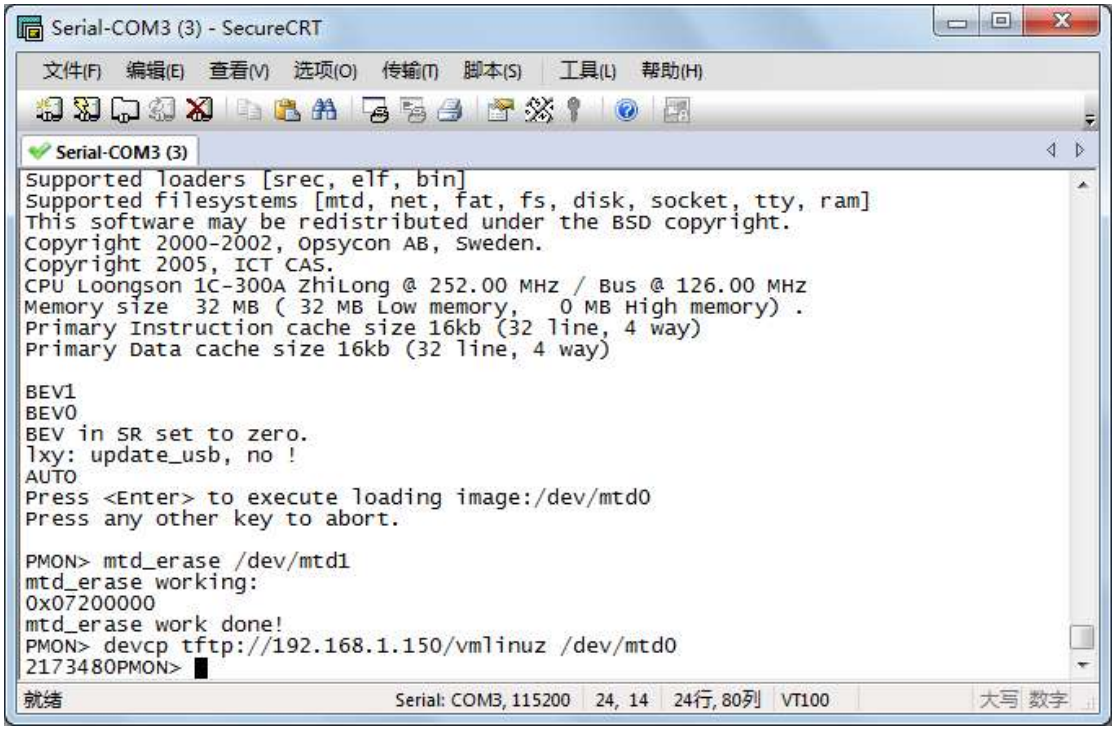


5. 在 PMON 中输入命令：

   devcp tftp://192.168.1.150/vmlinuz /dev/mtd0　　//下载内核

   然后回车（注意命令中应写 PC 的 IP），如下图：

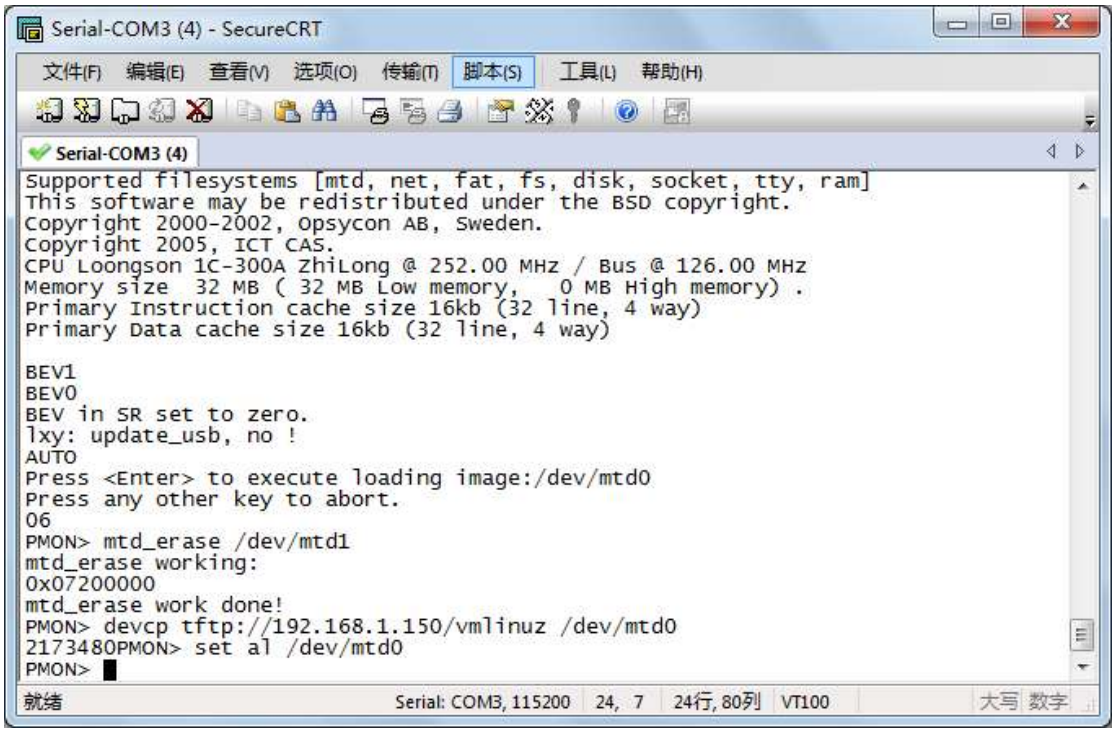




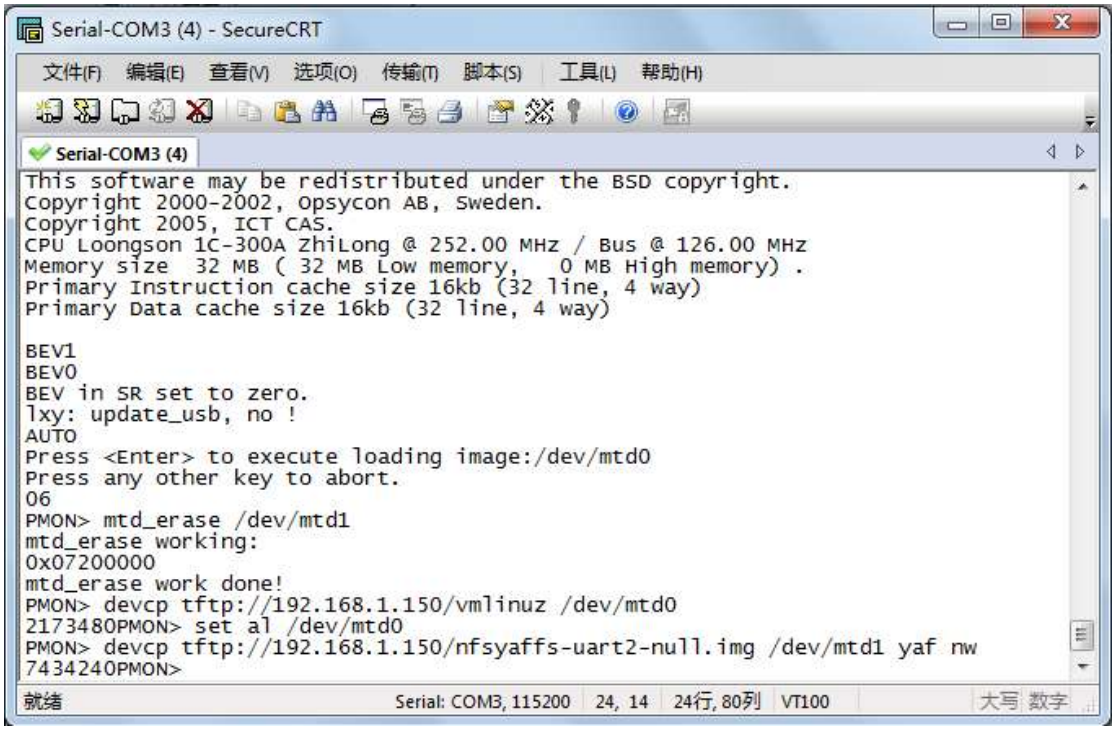
6. 在 PMON 中输入命令：

   set al /dev/mtd0　　//设置启动参数

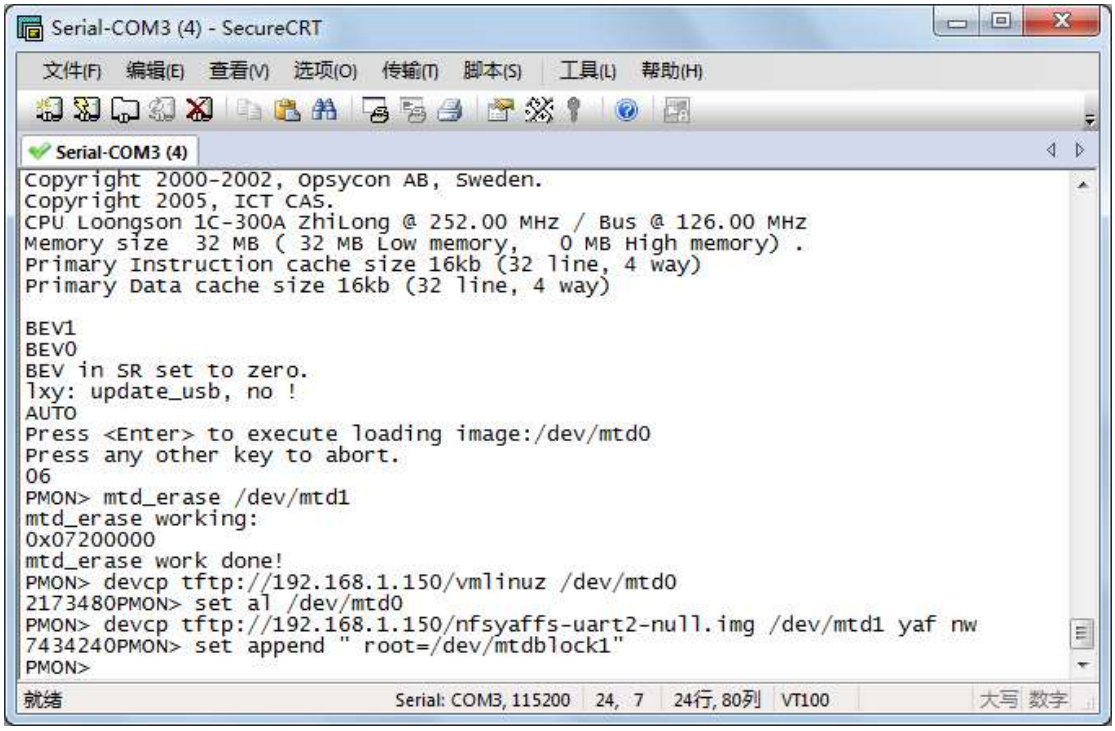
   然后回车，如下图：



7. 在 PMON 中输入命令：
devcp tftp://192.168.1.150/nfsyaffs-uart2-null.img /dev/mtd1 yaf nw
//烧写文件系统
然后回车，如下图：



8. 在 PMON 中输入命令：s
et append " root=/dev/mtdblock1" //根目录位置，块设备
然后回车，如下图：

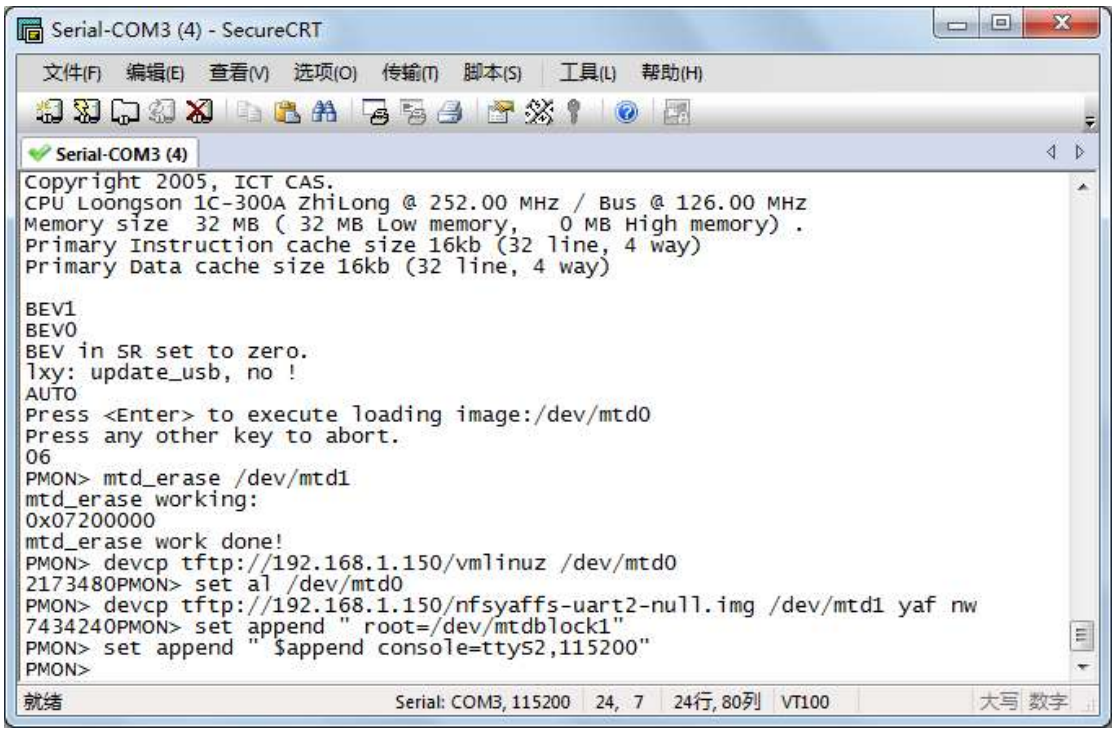


9. 在 PMON 中输入命令：

set append " $append console=ttyS2,115200"

//设置串口 3，115200 波特率

然后回车，如下图：

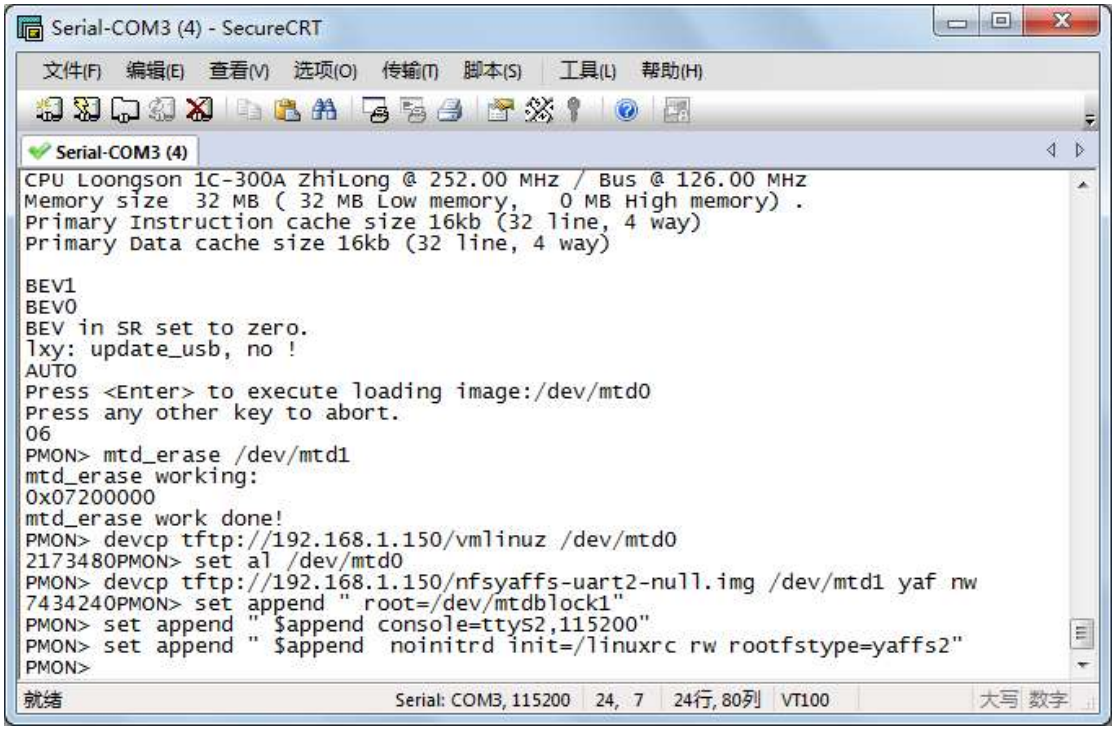


10. 在 PMON 中输入命令：

set append " $append noinitrd init=/linuxrc rw rootfstype=yaffs2"

//noinitrd 代表没有使用 ramdisk；init=/linuxrc 是指内核启动起来后进入系统中运行的第一个脚本，挂载之后文件系统是只读的，所以就加了个 rw；rootfstype=yaffs2 指明文件系统类型为 yaffs2 不然没法挂载根分区

然后回车，如下图：

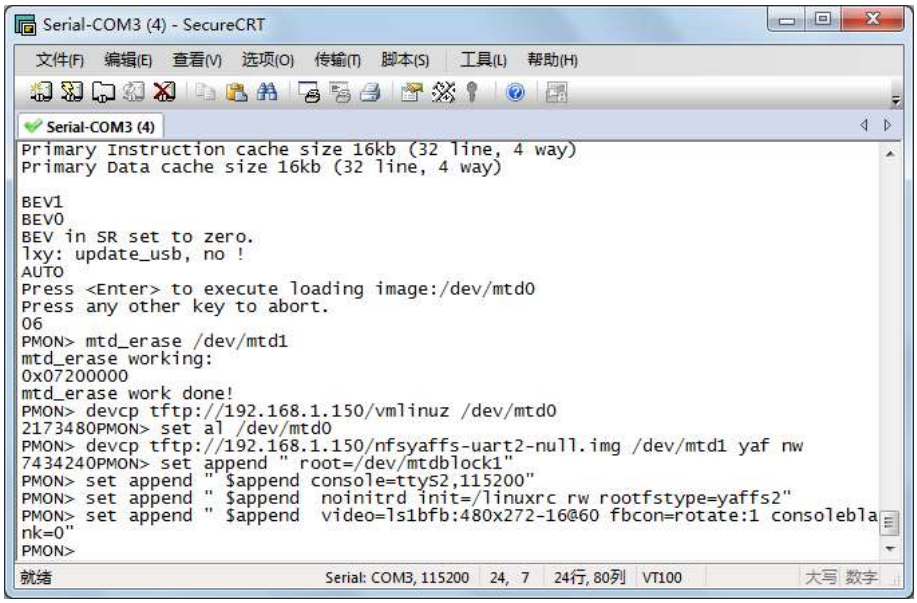


11. 在 PMON 中输入命令：

set append " $append video=ls1bfb:480x272-16@60 fbcon=rotate:1 consoleblank=0"

//fbcon=rotate:1 标示屏幕可旋转；consoleblank=0 禁用屏幕白色待机，

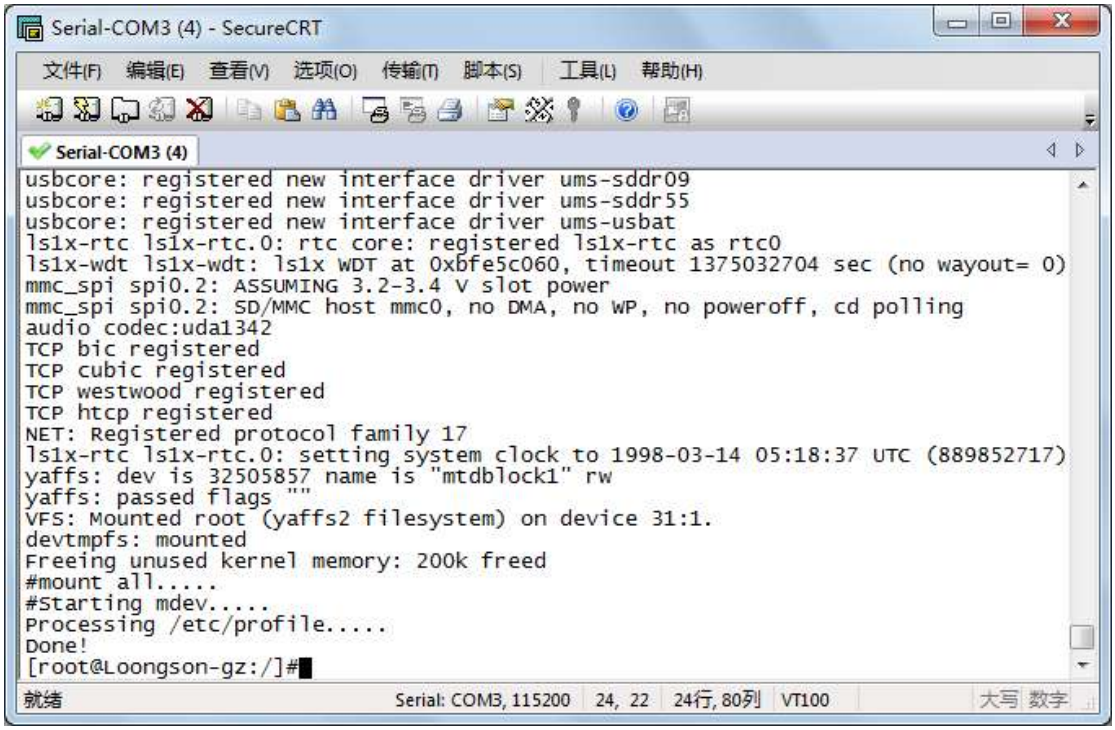
然后回车，如下图：



12. 在 PMON 中输入命令：

reboot //重启

然后回车，如下图：



至此，烧写 linux 系统结束。

【基于 linux 的 LED 跑马灯实验】

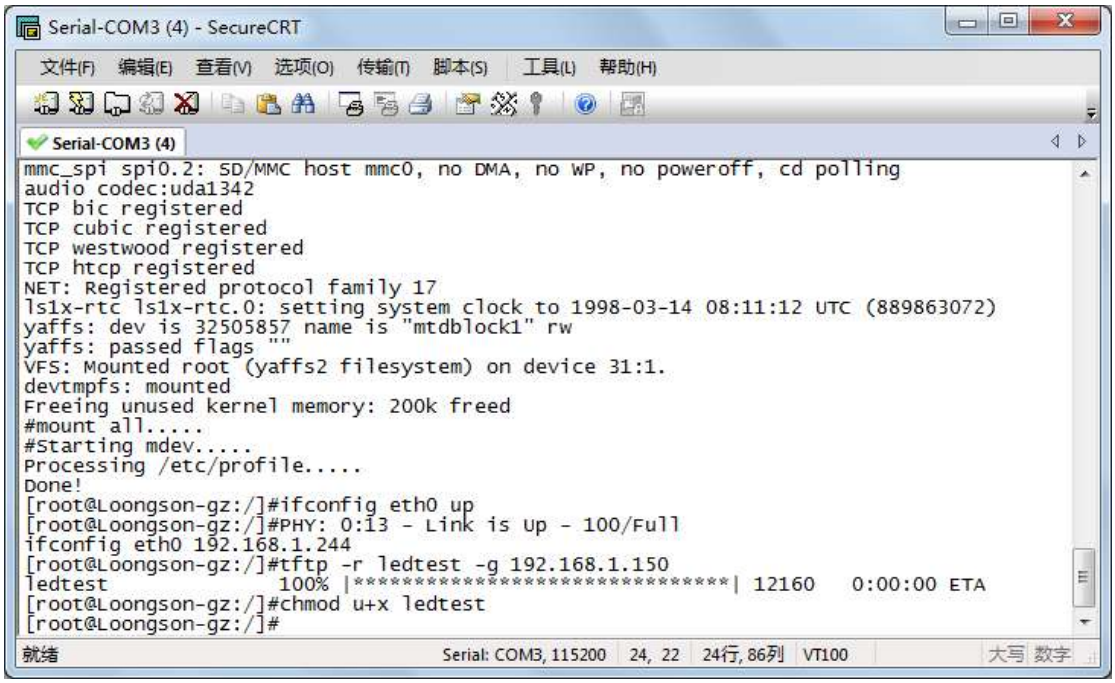
1. 把编译好的 LED 文件放置于 TFTP 安装目录下
2. 连接 PC 与开发板
3. 打开 SecureCRT 并与开发板连接，打开 TFTP，关闭 PC 防火墙
4. 系统正常运行后在 CRT 中输入一下命令：

ifconfig eth0 up　　//在文件系统里启动网口　，然后回车。

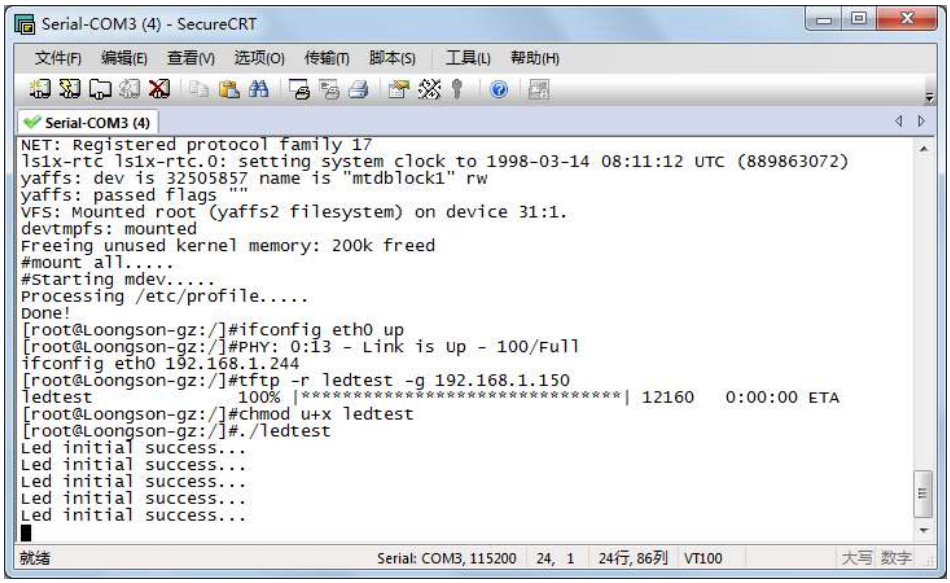
ifconfig eth0 192.168.1.244　//文件系统里配置网络 ip(ip 不要和主机一样)　，然后回车。

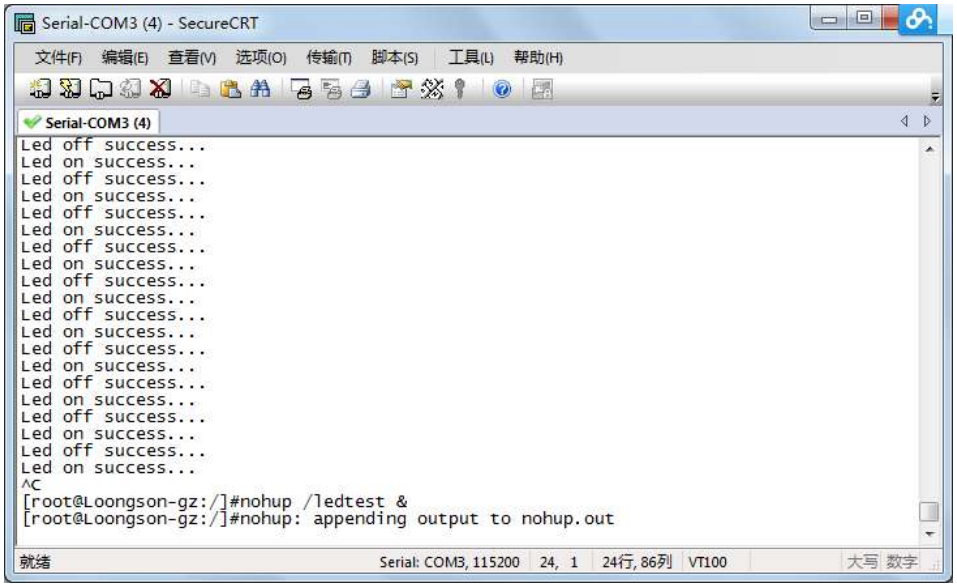
tftp -r ledtest -g 192.168.1.150　//下载编译好的 LED 文件，回车。

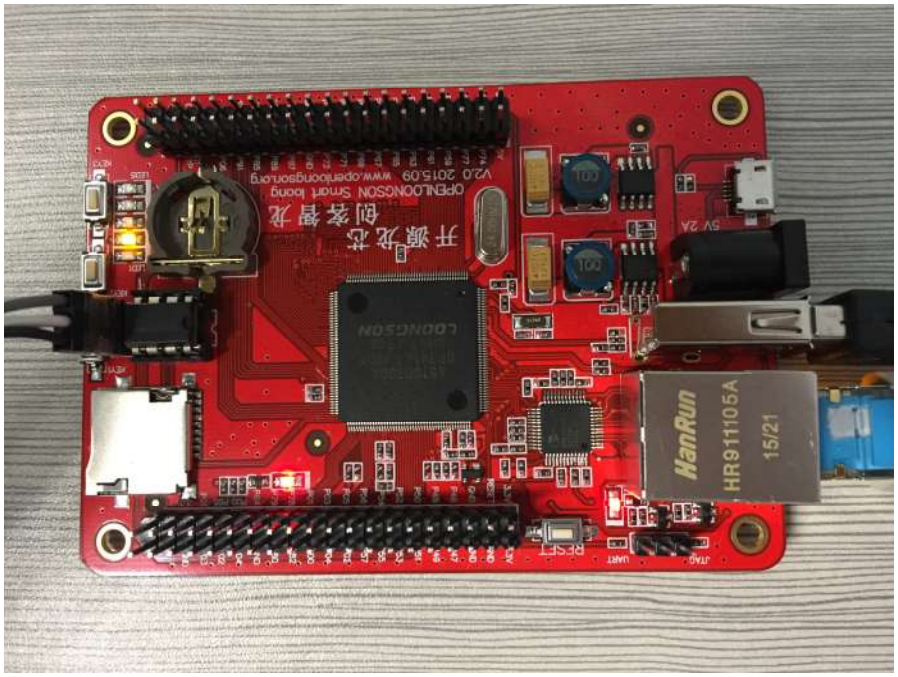
chmod u+x ledtest　　//给予权限，回车。



5. 在 SecureCRT 中输入命令：
./ledtest //运行 LED 文件 ，回车
或者输入：
nohup /ledtest & //在后台运行 LED 文件，回车
（在 CRT 中使用 ctrl+c 可以暂停运行）







2 软件篇

2.1PMON 引导系统

龙芯使用 PMON(Prom Monitor)作为基本的输入输出系统（BIOS）。PMON 是一个兼有 BIOS 和 boot loader 部分功能的开放源码软件，多用于 MIPS 系统。与 BIOS 相比功能不足，与常见的 bootloader 相比，功能要丰富的多。基于龙芯的系统采用 PMON 作为类 BIOS 兼 bootloader，并做了很多完善工作。

功能：

- 硬件初始化、操作系统引导和硬件测试、程序调试等功能。
- 提供多种加载操作系统的方式，可以从优盘、光盘、tftp 服务器和硬盘等媒介加载。
- 提供对内存、串口、显示、网络、硬盘等的基础测试工具。
- 支持软件升级。

智龙开发板内置的 PMON 命令:

PMON 中内置了很多命令，下面举例部分命令：

类型	命令	说明	例子	例子含义
帮助	h	查看帮助信息	h	列出所有可以使用的命令
			h ping	查看 ping 命令的用法
调试	d1	读某个地址的值(读一个 byte)	d1 0x80300000	查看地址 0x80300000 处的值
	d2	读某个地址的值(读一个 halt word)	d2 0x80300000	查看地址 0x80300000 处的值
	d4	读某个地址的值(读一个 word)	d4 0x80300000	查看地址 0x80300000 处的值
	m1	在某个地址处写入一个值(写入一个 byte 大小)	m1 0x80300000 0x12	在地址 0x80300000 处写入 0x12
	m2	在某个地址处写入一个值(写入一个 halt word 大小的值)	m2 0x80300000 0x1234	在地址 0x80300000 处写入 0x1234
	m4	在某个地址处写入一个值(写入一个 word 大小的值)	m4 0x80300000 0x12345678	在地址 0x80300000 处写入 0x12345678
内存	mt	内存测试命令	mt	测试开发板的内存是否正常
	load	下载 Linux 内核到内存	load tftp://192.168.1.12/vmlinux	通过网络从 IP 为 192.168.1.12 的主机上下载内核 vmlinx 到内
网络	ifaddr	设置 PMON 的 ip 地址(当次有效，断电后丢失)	ifaddr syn0 192.168.1.12	设置 PMON 的 IP 地址为 192.168.1.12
	ping	测试网络	ping 192.168.1.12	测试与 192.168.1.12 网口是否连通
环境管理	set	设置环境变量：设置的参数会保存到 nor	set	列出所有已经设置好的环境变量

		flash 高位地址，在 pmon 一开始运行时就会自动去调用。	set ifconfig syn0:192.168.1.12	设置 PMON 的 IP 地址，重启系统后 IP 地址固定存在
			set al /dev/mtd0	设置系统启动后 PMON 自动从 Nand Flash 分区 0 加载内核到内存
			set append "console=ttyS2"	设置系统的运行启动参数
	env	查看板上已经设好的环境变量	env	列出所有环境变量
FLASH	mtd_erase	擦除 Nand Flash 某分区的数据	mtd_erase /dev/mtd0	擦除 Nand Flash 分区 0 的数据
	devcp	拷贝文件到 Nand Flash 某分区中	devcp tftp://192.168.1.117/vmlinux /dev/mtd0	通过网络拷贝内核文件到 Nand Flash 分区 0
系统管理	reboot	重启系统	reboot	重启系统
其他	devls	查看设备列表	devls -n	查看网络设备

2.2Linux 内核裁剪和配置

Linux 内核很庞大，Linux 初学者以及致力于 Linux 应用软件开发的技术人员，熟悉内核的好的开始就是对内核进行配置，得到符合自己需求的经过裁剪的内核，并将编译后的内核下载到开发板中运行使用。

本篇内容是为想要对内核进行个性配置的人员准备的，不涉及到代码编写，学习 Linux 不必一切从“零”开始，一切可从学会配置、编译、下载运行开始。

Linux 内核的编译分为两个步骤，一、内核配置；二、内核编译。开发包默认提供一个配置文件，用户可以根据此配置文件对内核进行裁剪或者增加新的功能。

下面描述整个 Linux 内核的配置和编译过程。

获得龙芯 1C 的 linux 源码：

通过共享目录将该源码包拷贝到 Ubuntu 虚拟机普通用户目录下。

2.2.1 安装图形化配置工具 Ncurses

Ncurses 提供字符终端处理库，包括面板和菜单。

安装 Ncurses 有两种方式：一种连网在线安装；另一种是使用源码包安装（略）。

在终端下使用命令在线安装 Ncurses

```
$ sudo apt-get install libncurses5-dev
```

2.2.2 运行图形化配置界面

（1）建立交叉编译环境

```
$ source ~/Workstation/tools/toolchain/setenv-toolchain.sh
```

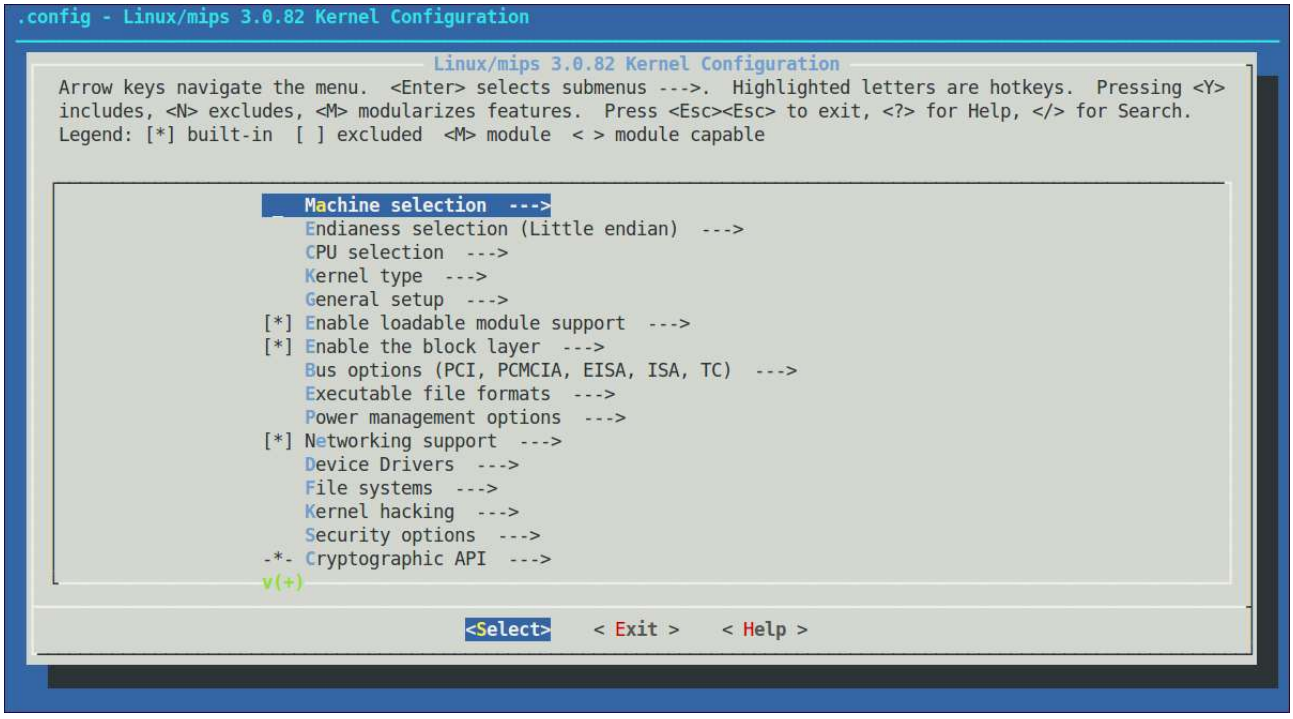
（2）解压 Linux 内核源码包，并拷贝 1C300B 开发板默认配置文件到内核源码包根目录下。

```
$ tar zxvf 1c300b-linux-3.0.tar.gz
$ cd 1c300b-linux-3.0
$ cp arch/mips/configs/ls1cv2_defconfig .config
```

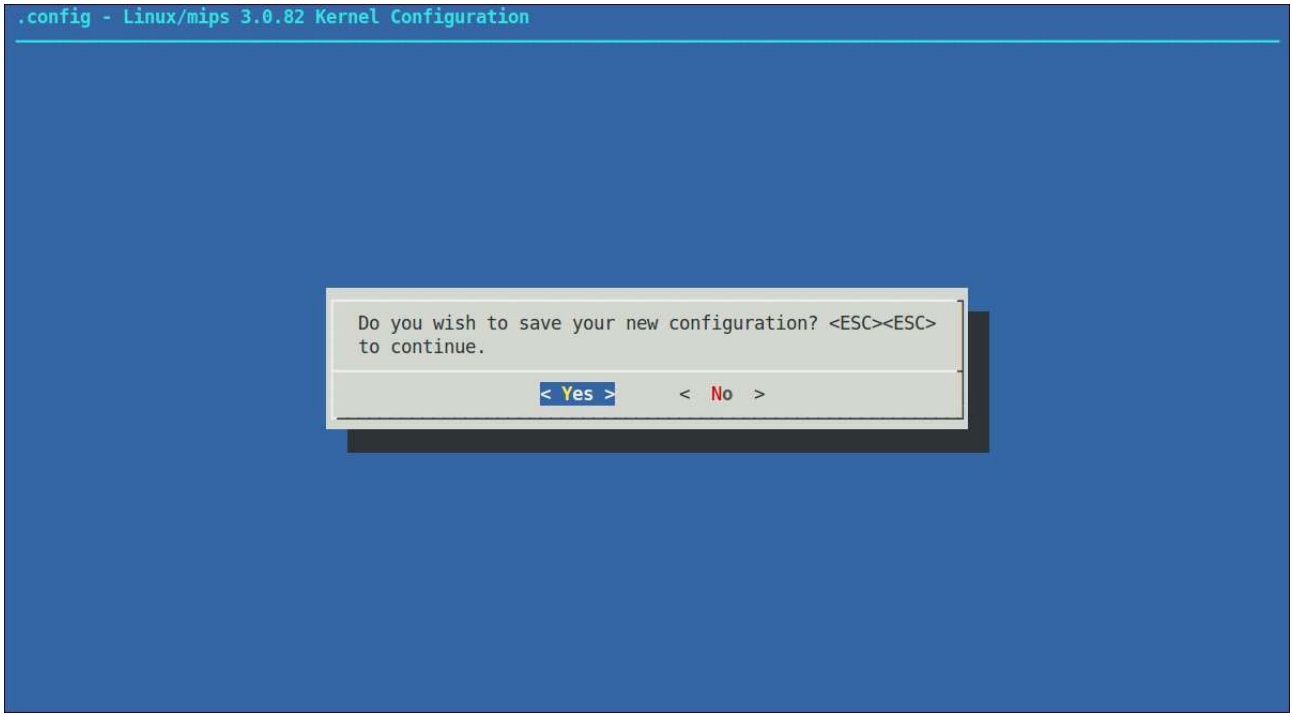
（3）运行图形化配置命令

```
$ make ARCH=mips CROSS_COMPILE=mipsel-linux- menuconfig
```

执行后进入内核配置主菜单界面。



在该界面中，配置操作通过键盘来完成。配置完成后，使用方向键选中“< Exit >”并按回车键退出，这时弹出一个提示界面，选择“< Yes >”保存配置。



2.2.3 编译 Linux 内核

配置完内核后，执行编译命令（要使用交叉编译工具链）。

```
$ make ARCH=mips CROSS_COMPILE=mipsel-linux-
```

编译完成后，在当前目录下生成内核镜像文件 vmlinux，还有其经压缩过的 vmlinuz，烧写其中一个均可。

2.2.4 开发板各模块驱动源码

- **1C300B 开发板平台文件**
  arch/mips/loongson/ls1x/ls1c/ls1cv2_platform.c

- **网卡驱动**
  drivers/net/stmmac/

- **音频驱动**
  OSS： sound/oss/ls1x_iis.c
  ALSA：sound/soc/loongson1/
- **串口驱动**
  drivers/tty/serial/8250.c

- **实时时钟驱动**

drivers/rtc/rtc-ls1cv2-toy.c

- 触摸屏驱动
  drivers/input/touchscreen/ads7846.c（没有焊上对应的硬件模块）

- **LCD** 驱动
  drivers/video/ls1xfb.c

- 按键驱动
  drivers/input/keyboard/gpio_keys.c

- **Nand Flash** 控制器驱动
  drivers/mtd/nand/ls1x-nand.c

- **USB** 控制器驱动
  drivers/usb/host/ehci-ls1x.c 与 ohci-ls1x.c

- **SPI** 控制器驱动
  drivers/spi/spi_ls1x.c

- **I2C** 控制器驱动
  drivers/i2c/busses/i2c-ls1x.c

- **SD** 卡驱动
  drivers/mmc/host/ls1x_mci.c

- **PWM** 驱动
  arch/mips/loongson/ls1x/pwm.c

- **CAN** 总线驱动
  drivers/net/can/sja1000/

- 红外驱动
  drivers/char/ls1b_ir.c

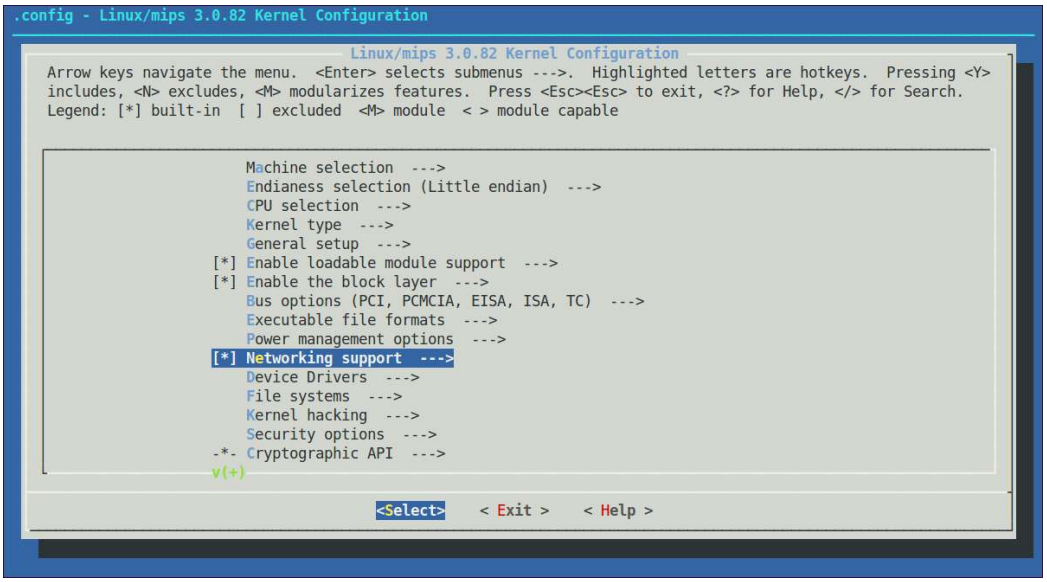
- **ADC** 驱动
  drivers/hwmon/ls1x-hwmon.c

- 看门狗驱动
  drivers/watchdog/ls1x_wdt.c

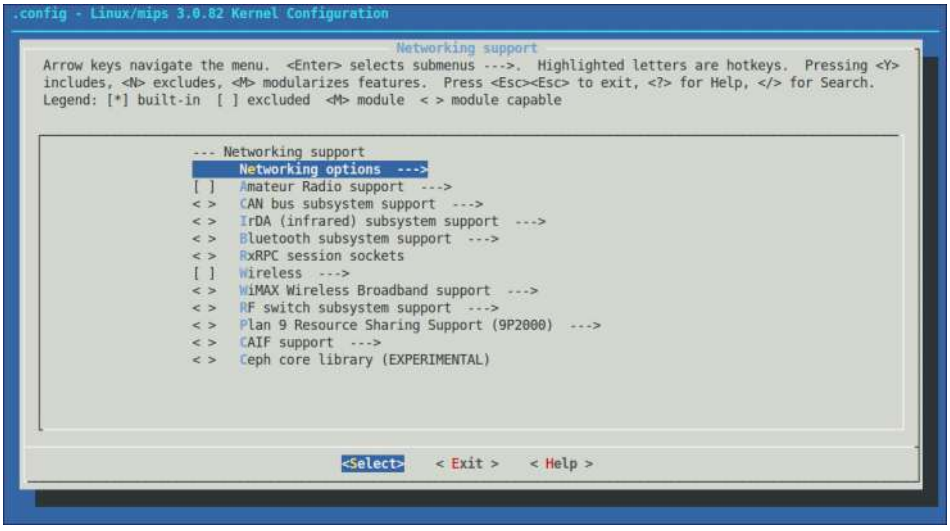
2.3 配置内核各模块驱动

2.3.1 配置网卡驱动

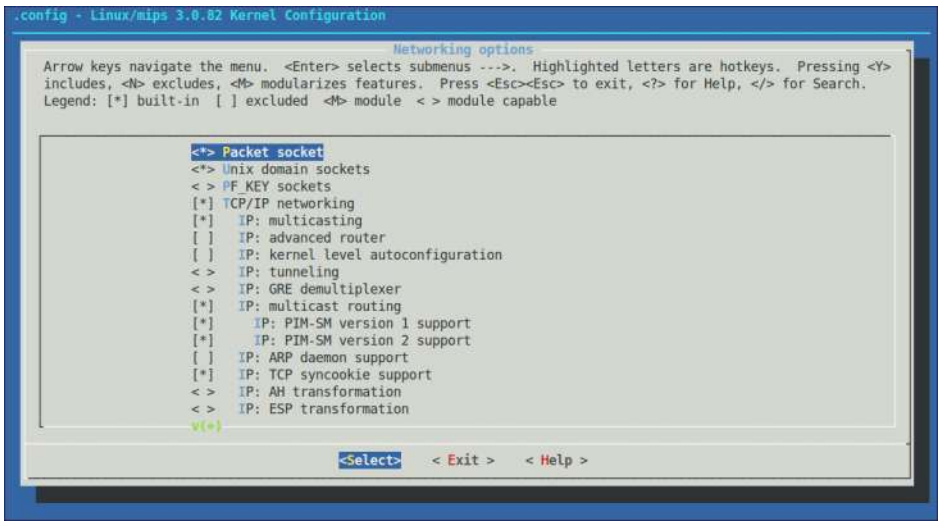
进入 Linux 内核配置主界面。选择“[*] Networking support　--->”并进入。

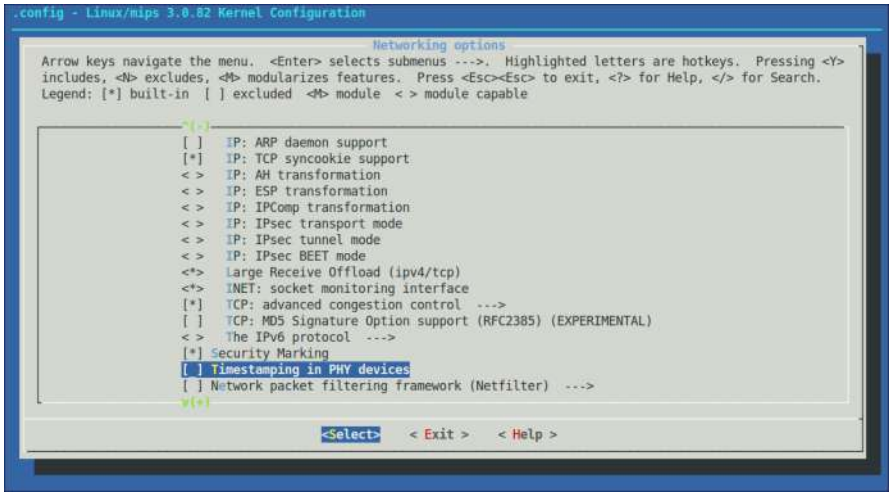


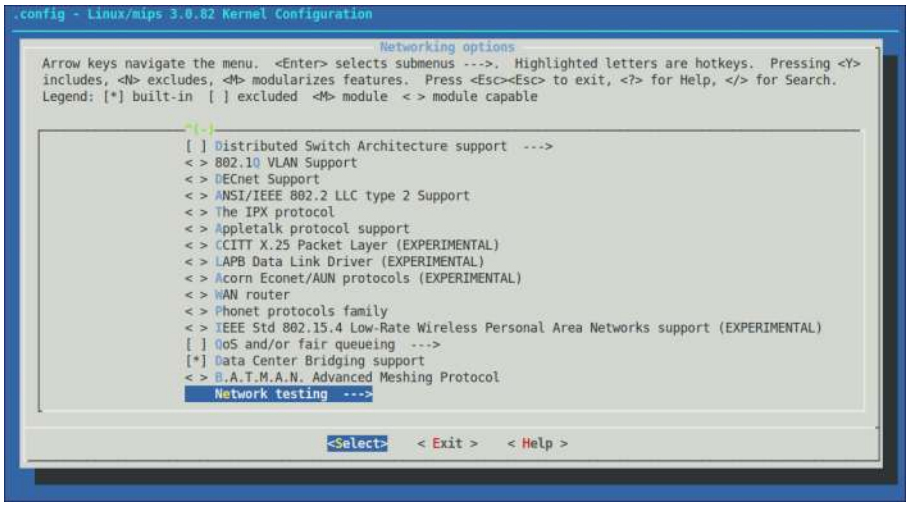
选择“Networking options　--->”并进入。



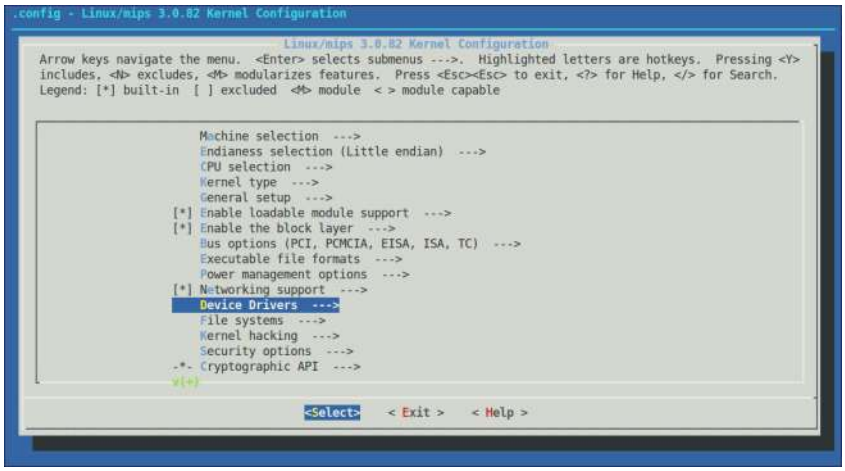
在网络选项配置界面中一般采用默认设置即可。


.config - Linux/mips 3.0.82 Kernel Configuration
Networking options
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search.
Legend: [*] built-in [ ] excluded <M> module < > module capable
<*> Packet socket
<*> Unix domain sockets
< > PF_KEY sockets
[*] TCP/IP networking
[*] IP: multicasting
[ ] IP: advanced router
[ ] IP: kernel level autoconfiguration
< > IP: tunneling
< > IP: GRE demultiplexer
[*] IP: multicast routing
[*] IP: PIM-SM version 1 support
[*] IP: PIM-SM version 2 support
[ ] IP: ARP daemon support
[*] IP: TCP syncookie support
< > IP: AH transformation
< > IP: ESP transformation
<Select> < Exit > < Help >

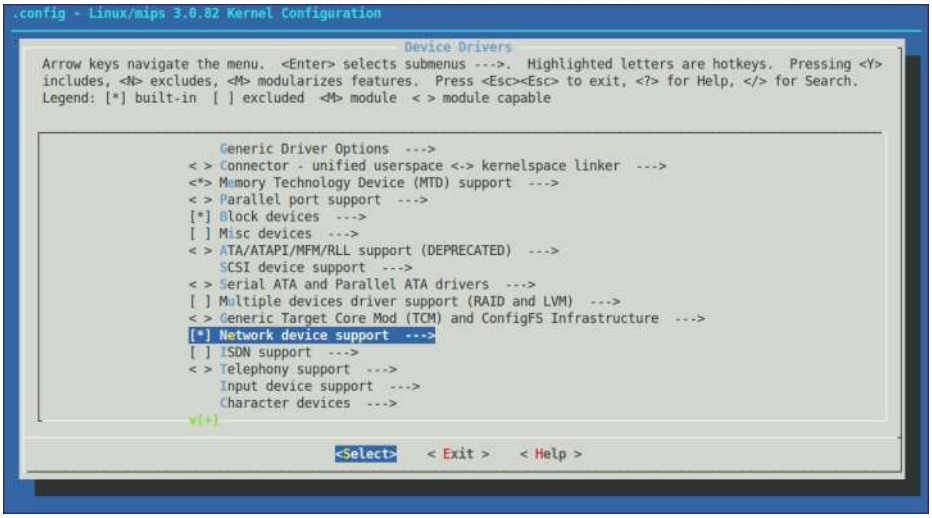

.config - Linux/mips 3.0.82 Kernel Configuration
Networking options
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search.
Legend: [*] built-in [ ] excluded <M> module < > module capable
[ ] IP: ARP daemon support
[*] IP: TCP syncookie support
< > IP: AH transformation
< > IP: ESP transformation
< > IP: IPComp transformation
< > IP: IPsec transport mode
< > IP: IPsec tunnel mode
< > IP: IPsec BEET mode
<*> Large Receive Offload (ipv4/tcp)
<*> INET: socket monitoring interface
[*] TCP: advanced congestion control --->
[ ] TCP: MD5 Signature Option support (RFC2385) (EXPERIMENTAL)
< > The IPv6 protocol --->
[*] Security Marking
[ ] Timestamping in PHY devices
[ ] Network packet filtering framework (Netfilter) --->
<Select> < Exit > < Help >


.config - Linux/mips 3.0.82 Kernel Configuration
Networking options
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search.
Legend: [*] built-in [ ] excluded <M> module < > module capable
[ ] Distributed Switch Architecture support --->
< > 802.1Q VLAN Support
< > DECnet Support
< > ANSI/IEEE 802.2 LLC type 2 Support
< > The IPX protocol
< > Appletalk protocol support
< > CCITT X.25 Packet Layer (EXPERIMENTAL)
< > LAPB Data Link Driver (EXPERIMENTAL)
< > Acorn Econet/AUN protocols (EXPERIMENTAL)
< > WAN router
< > Phonet protocols family
< > IEEE Std 802.15.4 Low-Rate Wireless Personal Area Networks support (EXPERIMENTAL)
[ ] QoS and/or fair queueing --->
[*] Data Center Bridging support
< > B.A.T.M.A.N. Advanced Meshing Protocol
Network testing --->
<Select> < Exit > < Help >

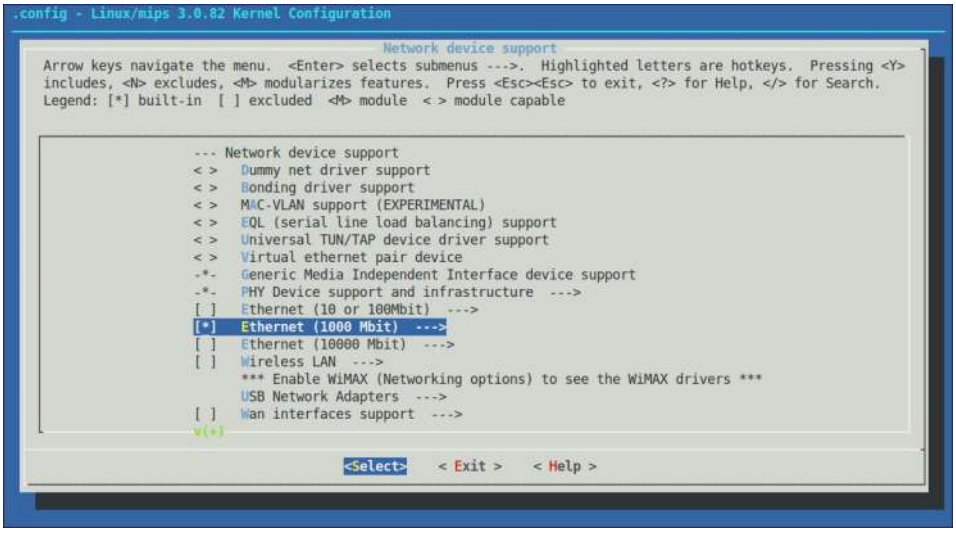
回到内核配置主界面中，选择“Device Drivers　--->”并进入。



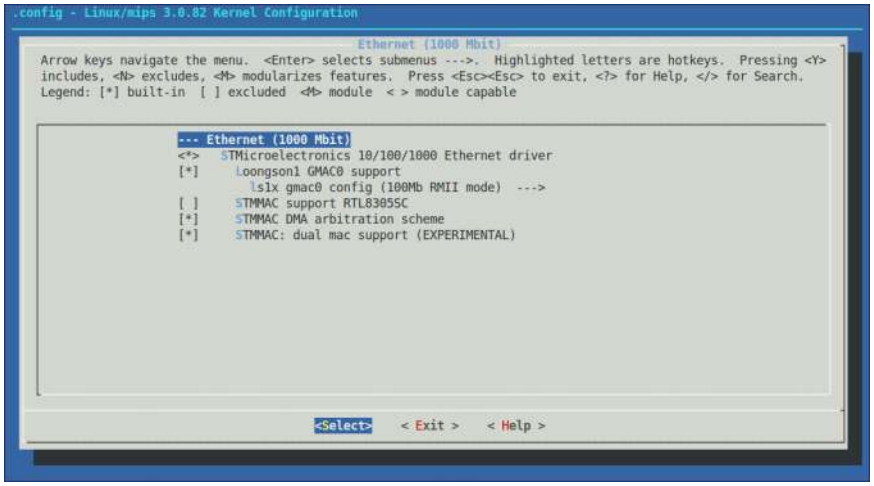
然后选择“[*] Network device support　--->”选项并进入。



选择相应的网络设备选项，这里配置“[*]　Ethernet (1000 Mbit)　--->”。

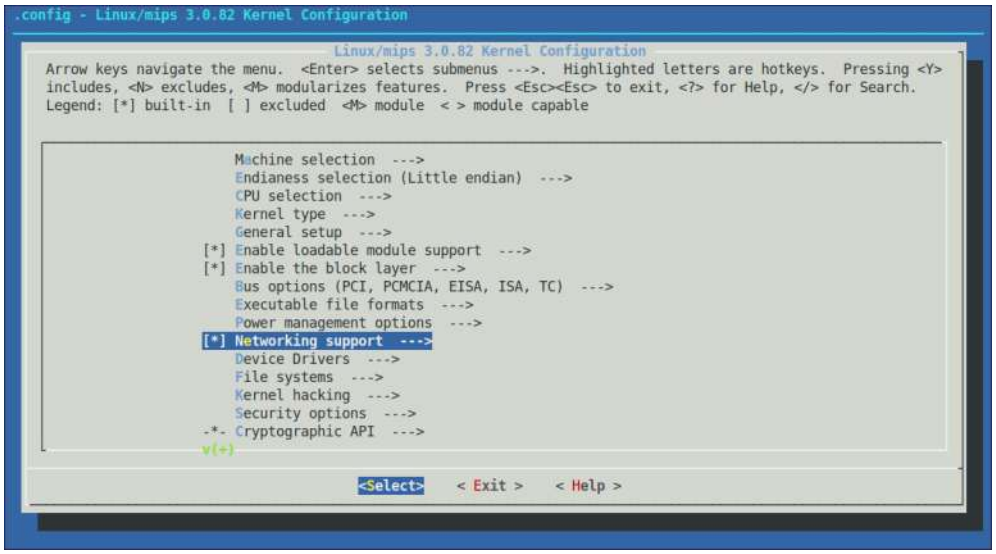


网络设置具体配置如下。

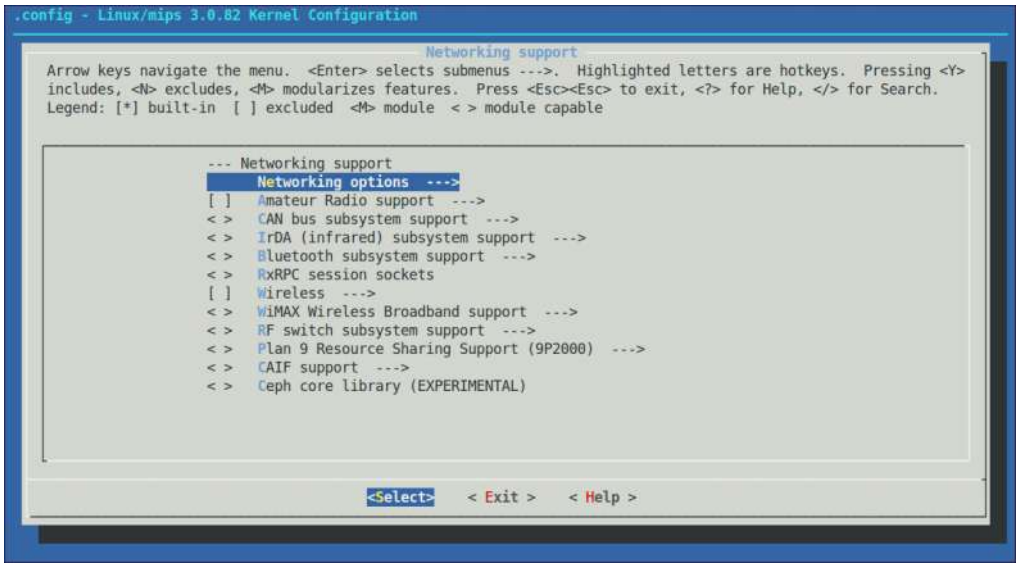


2.3.2 配置 NFS 支持

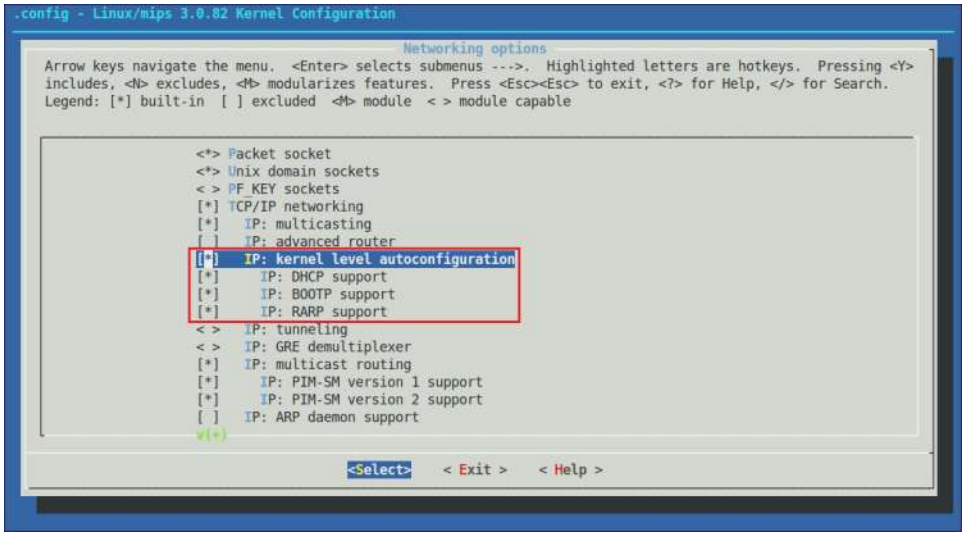
先配置网络协议的支持，在主菜单界面中，选择“Networking support”选项，按回车进入。



选择“Networking options”选项并进入。

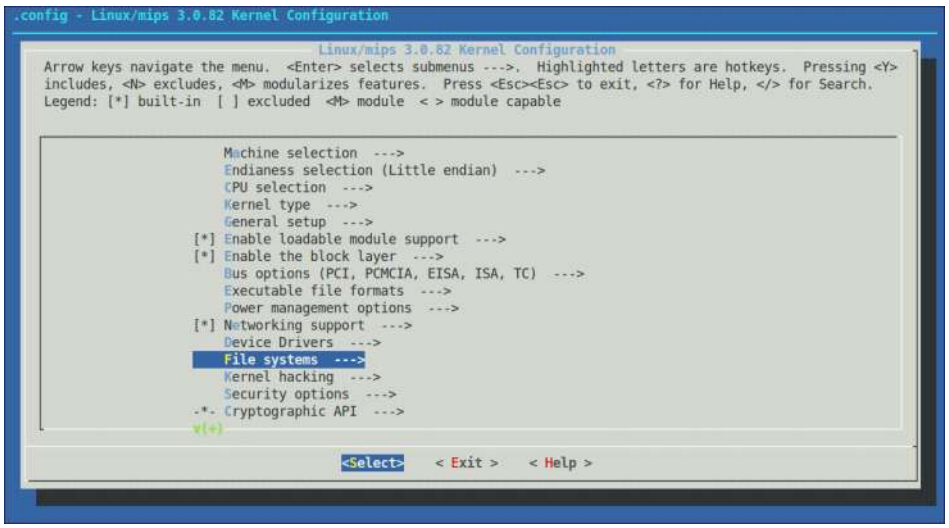


确保网络协议支持的配置已选择“IP：kernel level autoconfiguration”，及其三个子项。

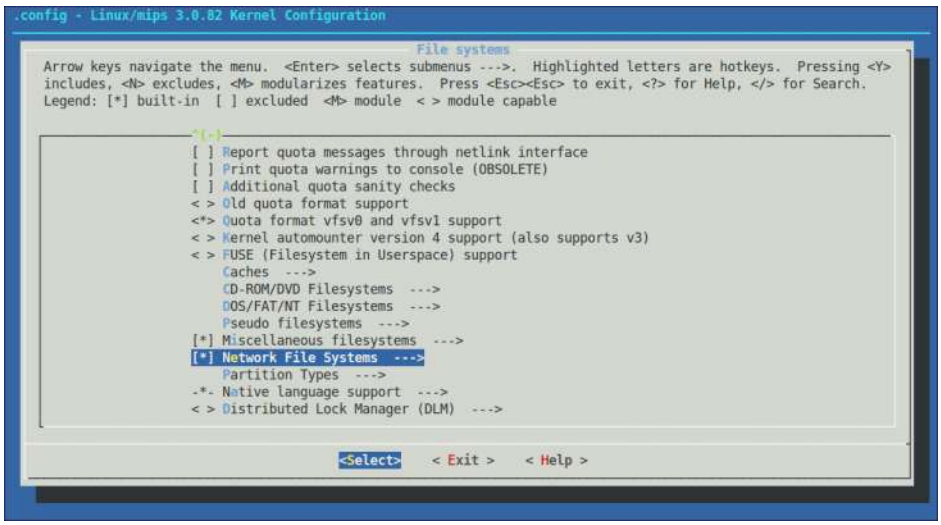


到这里就完成 NFS 的网络协议支持配置，然后一直退回主界面。

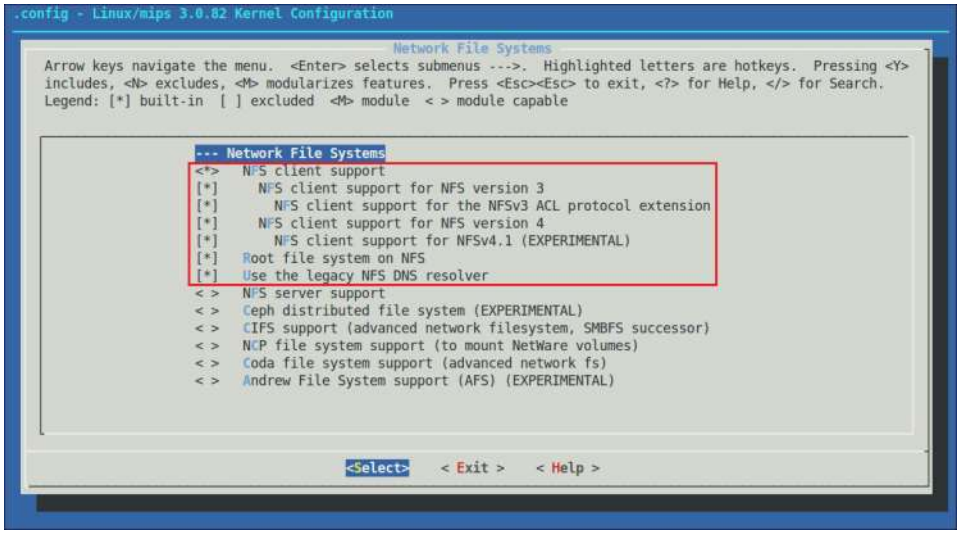
接着配置文件系统支持，在主菜单界面中，选择设备驱动“File systems”选项，按回车进入。



选择网络文件系统“Network File Systems”选项并进入。

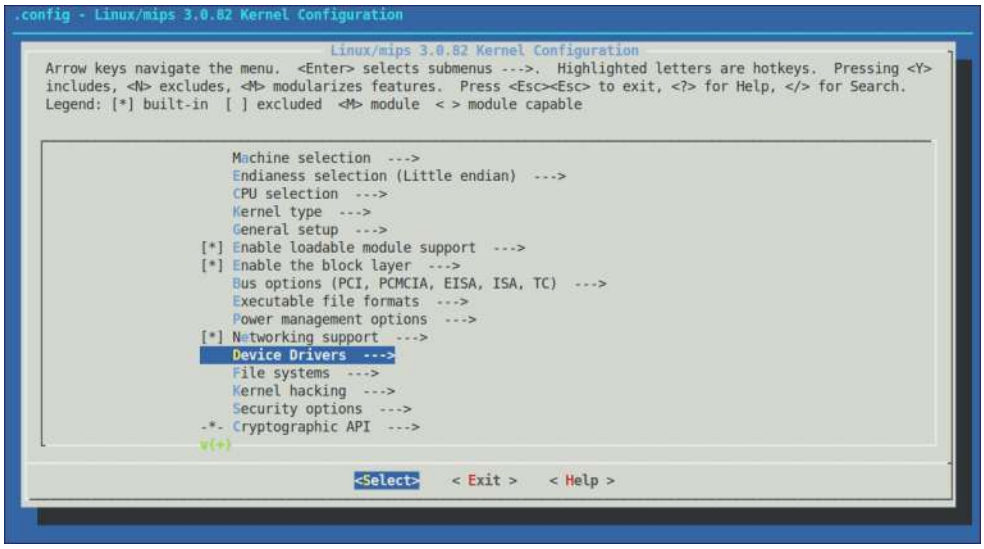


网络文件系统的配置，这里配置客户端支持“NFS client support”选项。

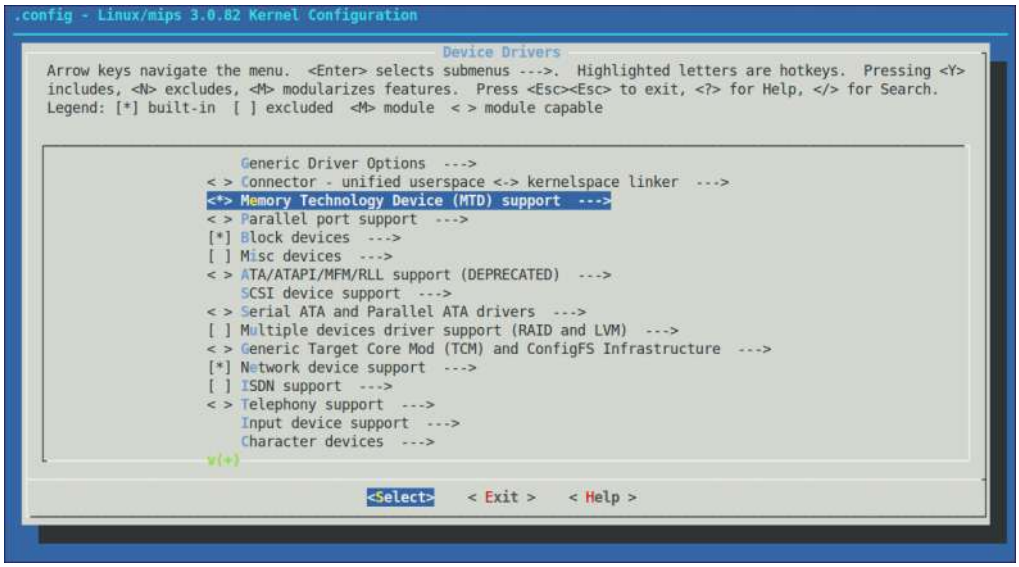


2.3.3 配置 UBIFS 支持

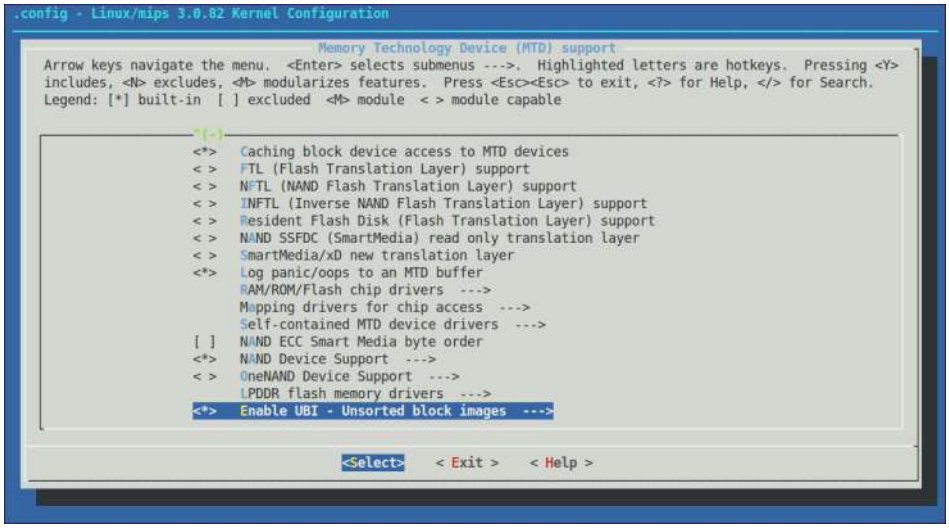
在主菜单界面中，选择“Device Drivers”选项，按回车进入。



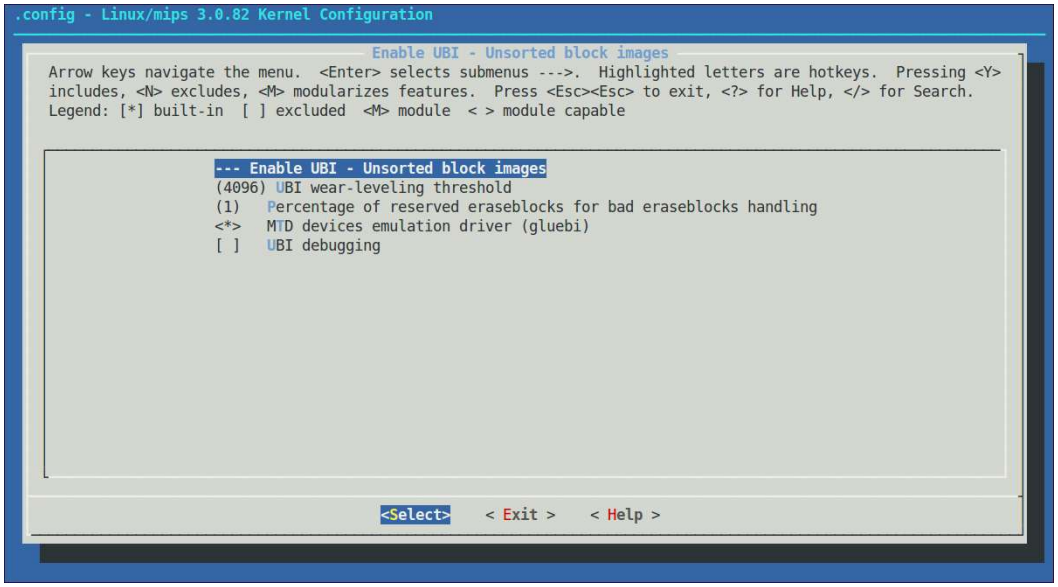
选择“Memory Technology Device (MTD) support”选项并进入。



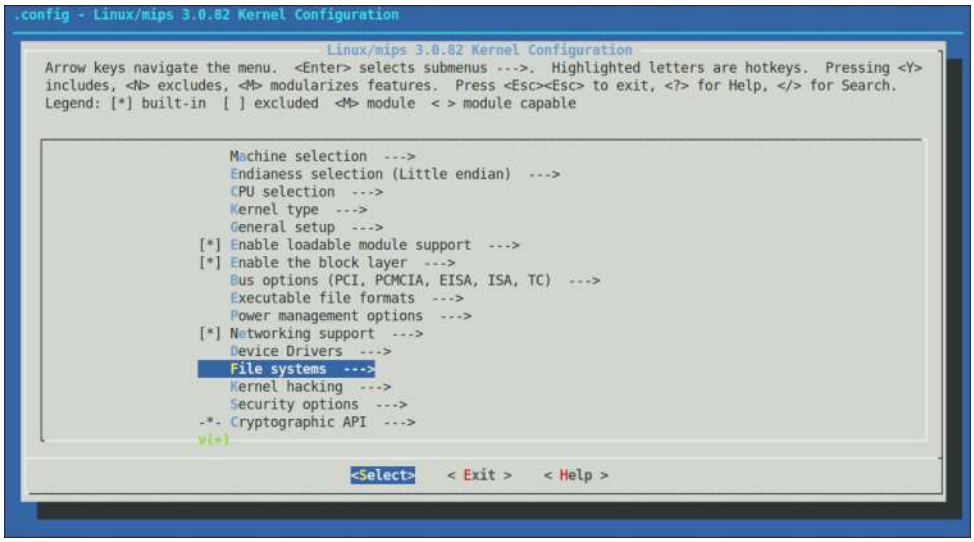
找到“Enable UBI - Unsorted block images”选项，选择并进入。



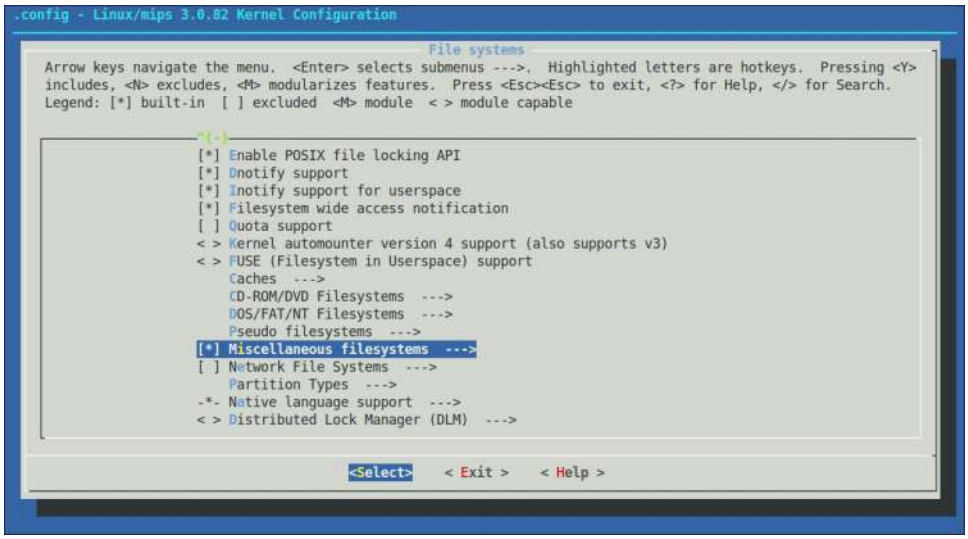
采用如下的配置。



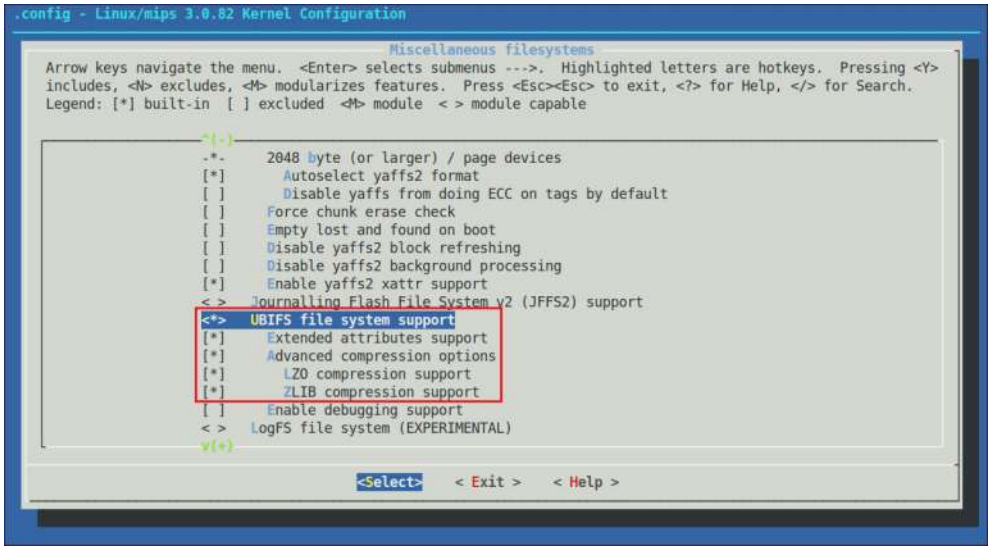
返回到配置主界面，进入“File systems”选项。



选择并进入“Miscellaneous filesystems”选项。

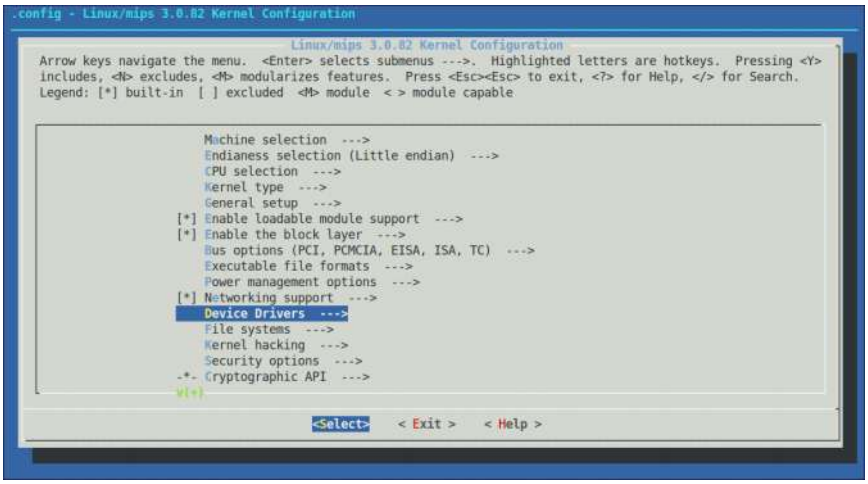


配置 UBIFS 文件系统的支持。

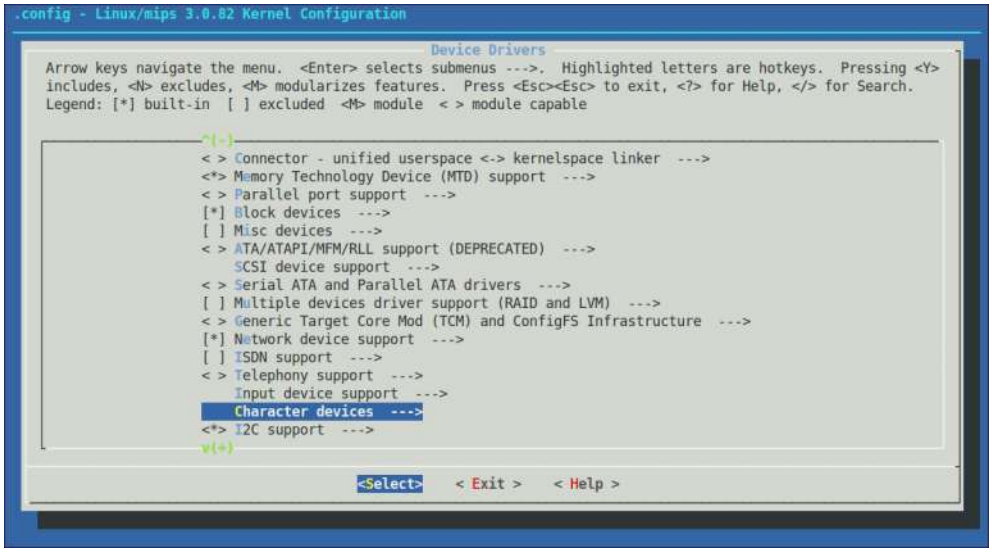


2.3.4 配置串口驱动

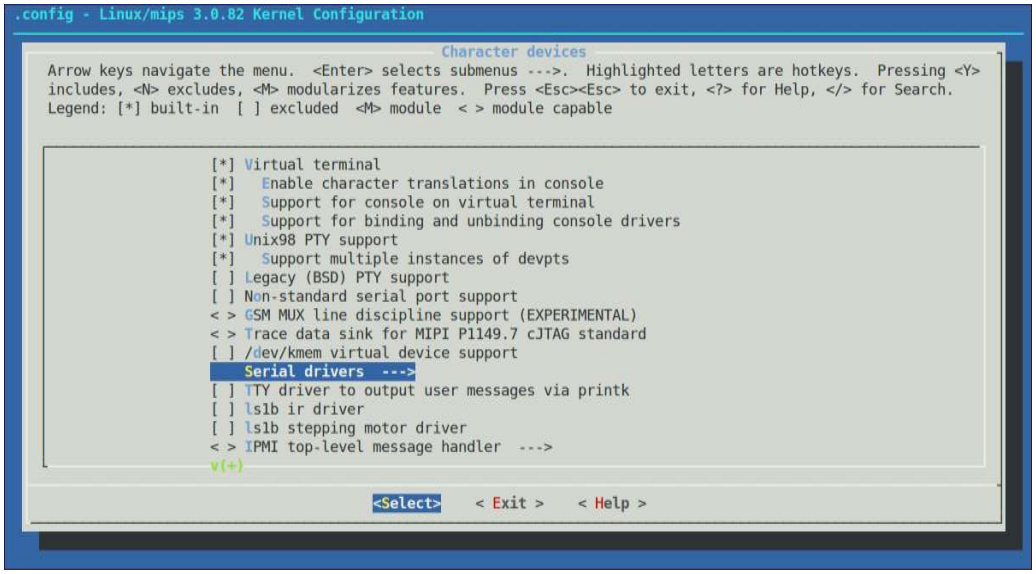
在主菜单界面中，选择“Device Drivers”选项，按回车进入。



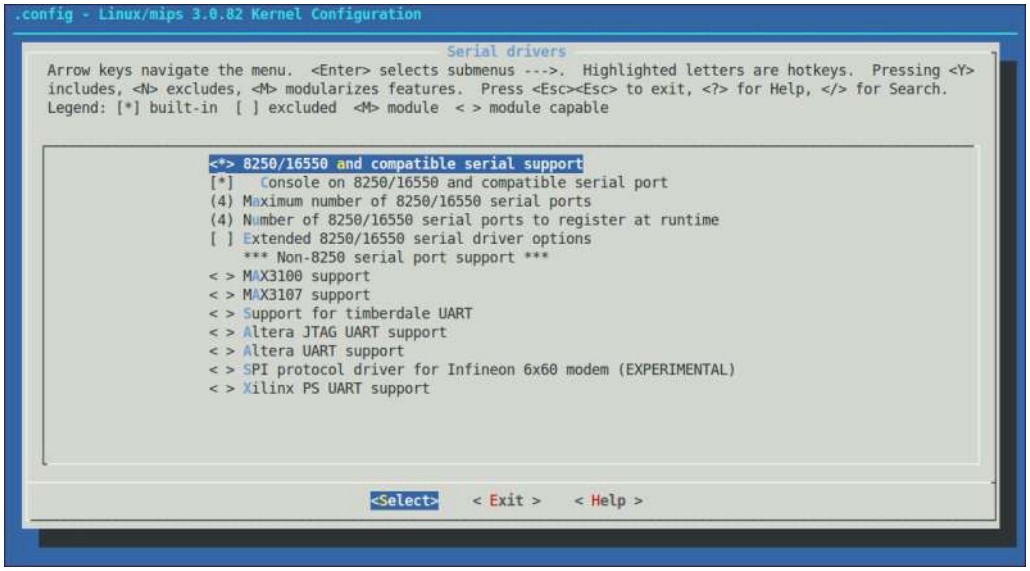
找到字符设备“Character devices”选项并进入。



选择串口驱动“serial drivers”选项并进入。

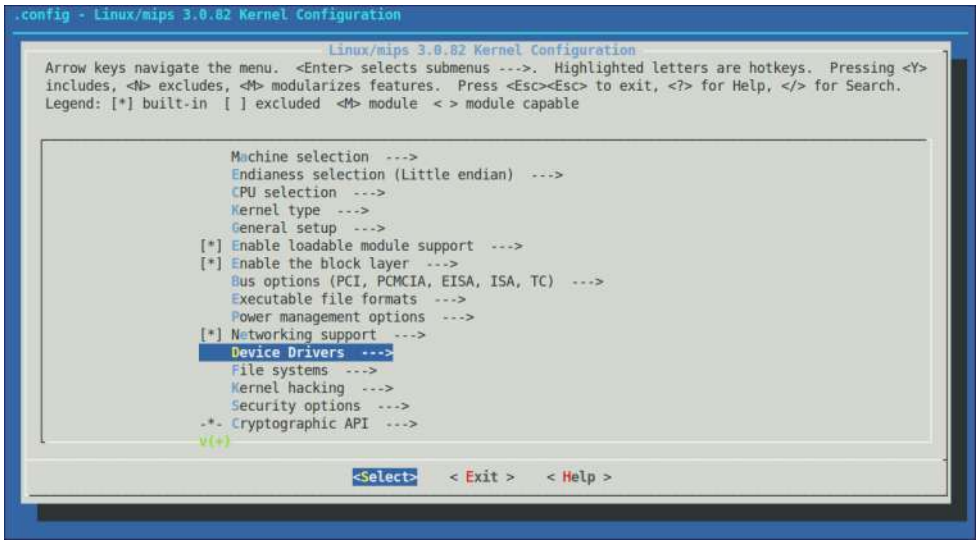


按照下图进行配置。

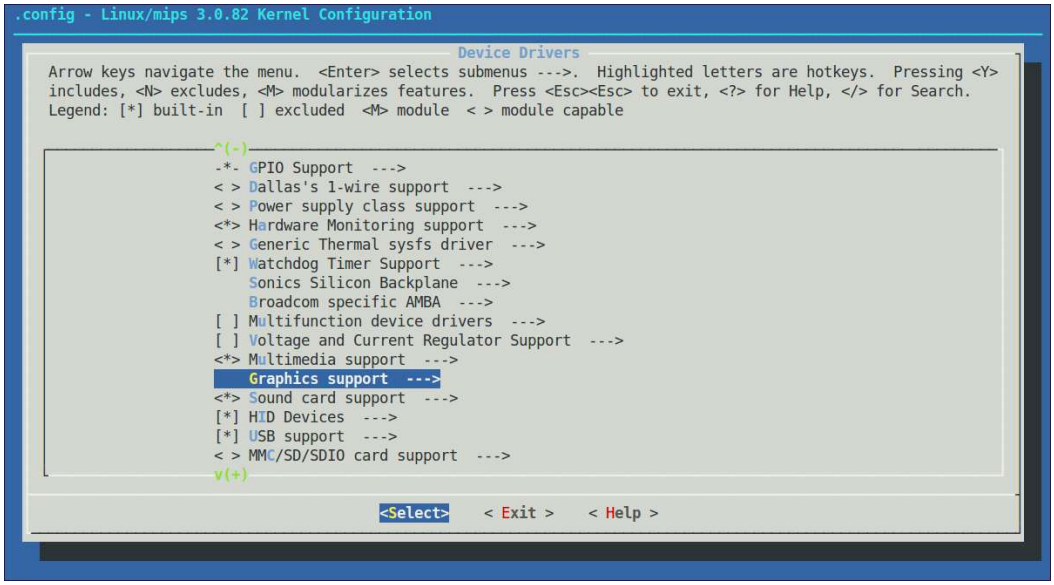


2.3.5 配置 LCD 驱动

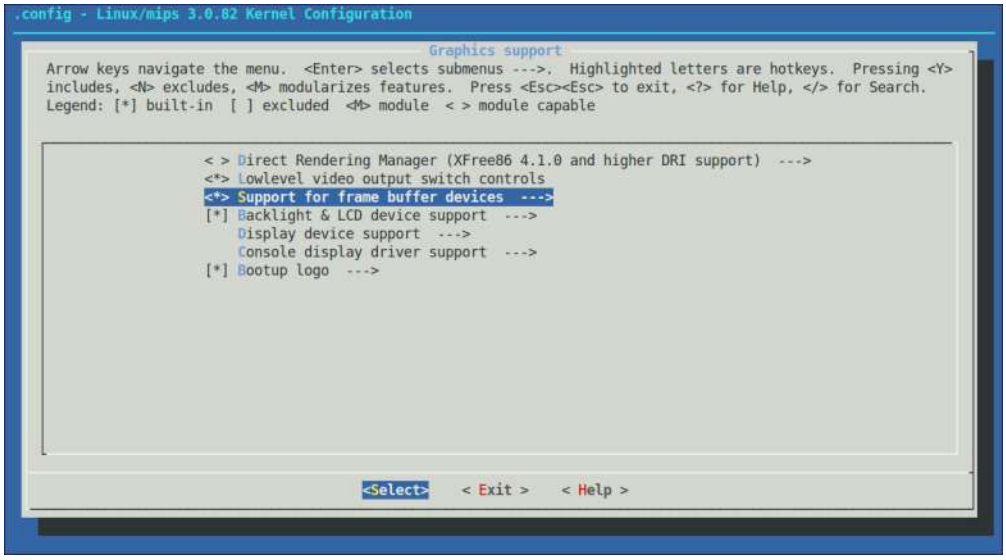
在主菜单界面中，选择“Device Drivers”选项，按回车进入。



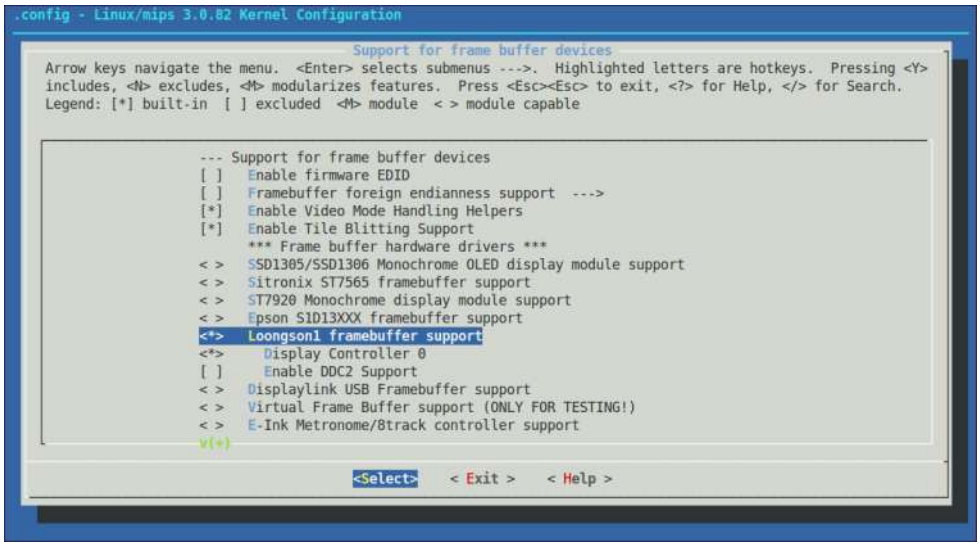
找到图形支持“Graphics support”选项并进入。



选择“Support for frame buffer devices”选项并进入。

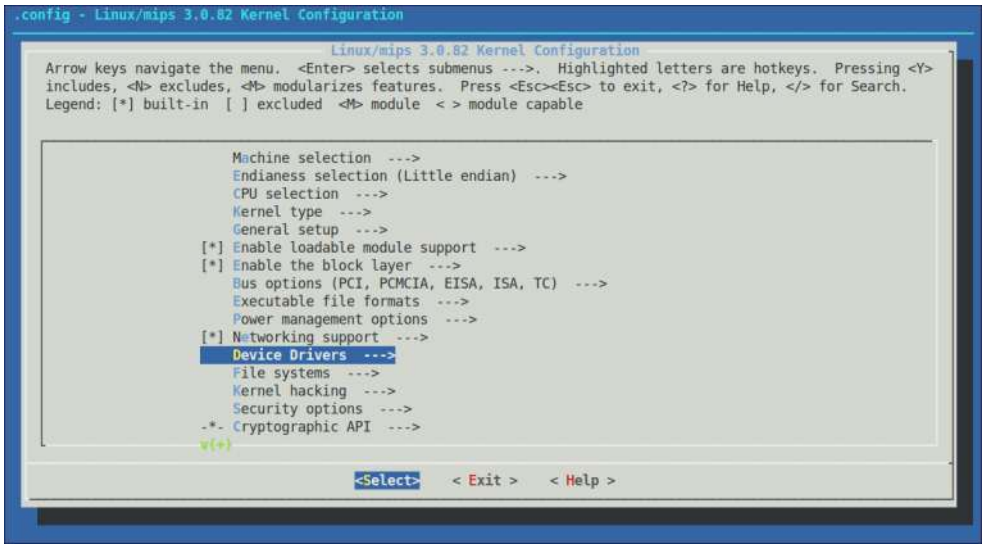


选择“Loongson1 framebuffer support”选项及其子项。

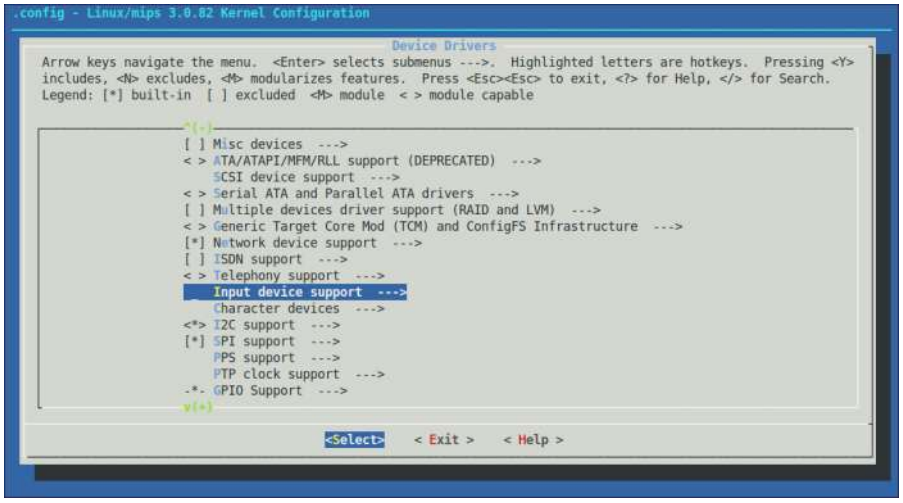


2.3.6 配置按键驱动

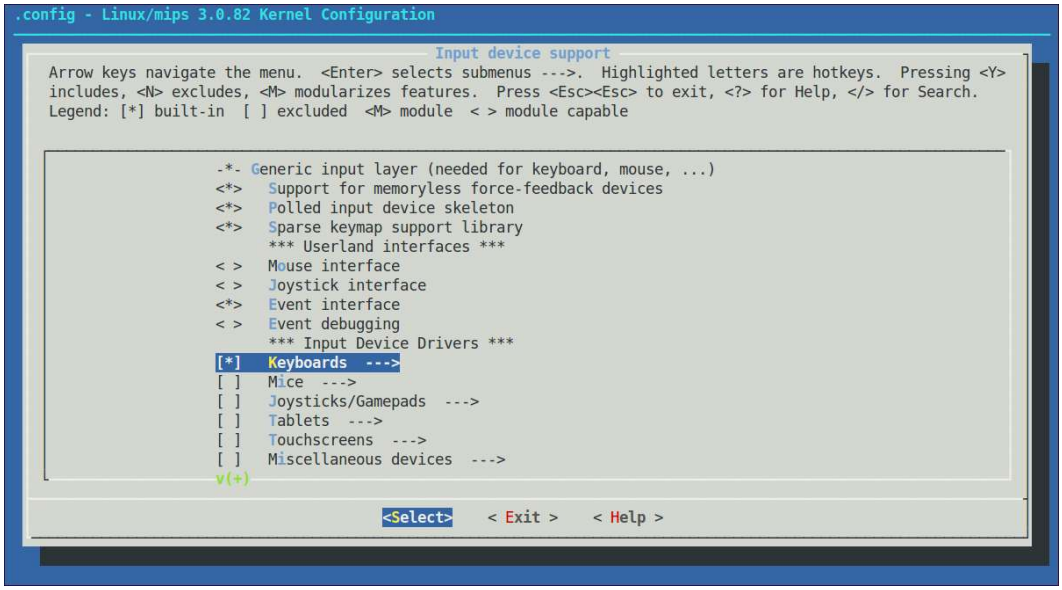
在 Linux 内核配置主界面中选择“Device Drivers --->”并进入。



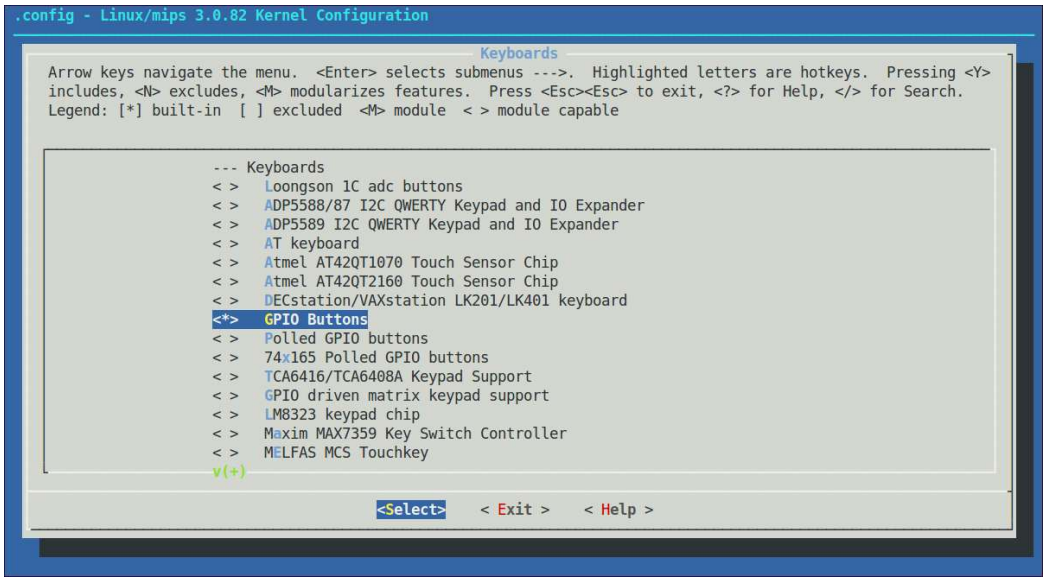
选择“Input device support --->”。



选择“[*] Keyboards --->”。



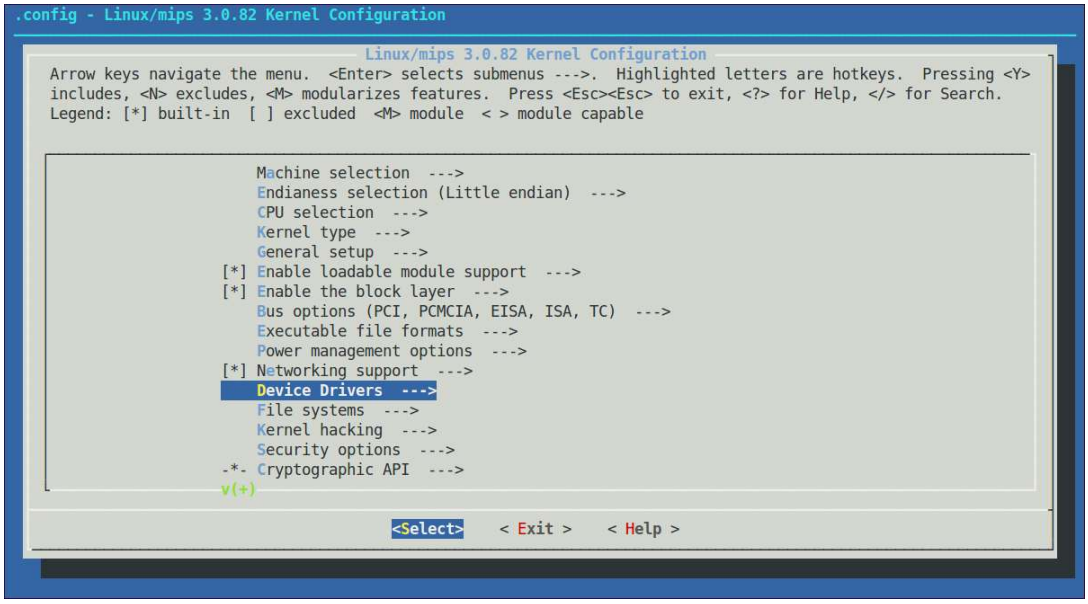
按键驱动选择“<*>　GPIO Buttons”。



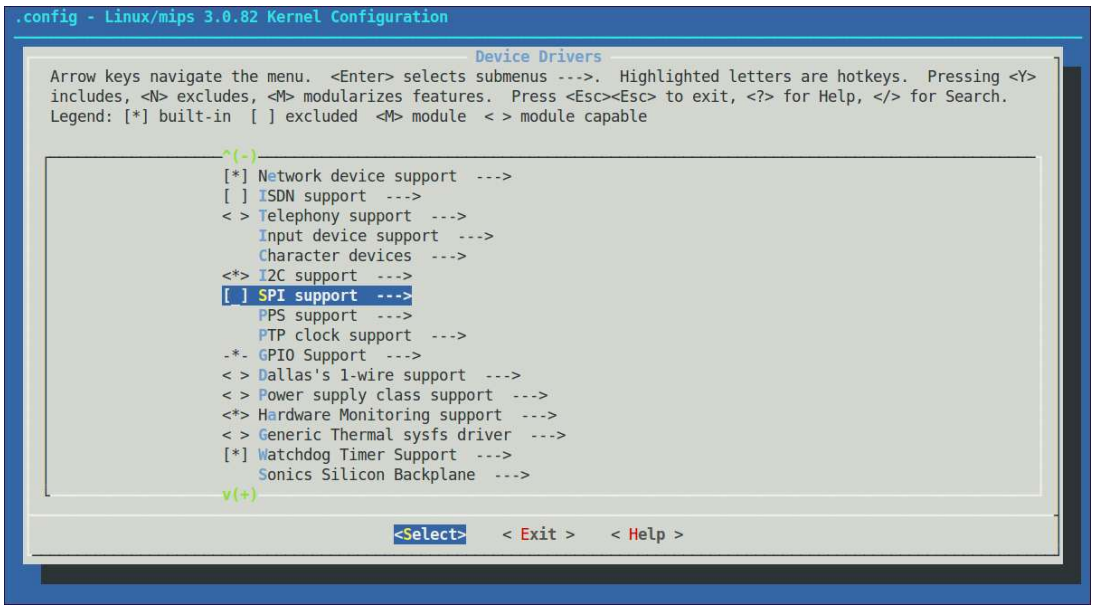
设备节点：**/dev/input/event0**

2.3.7 配置 SD 卡驱动

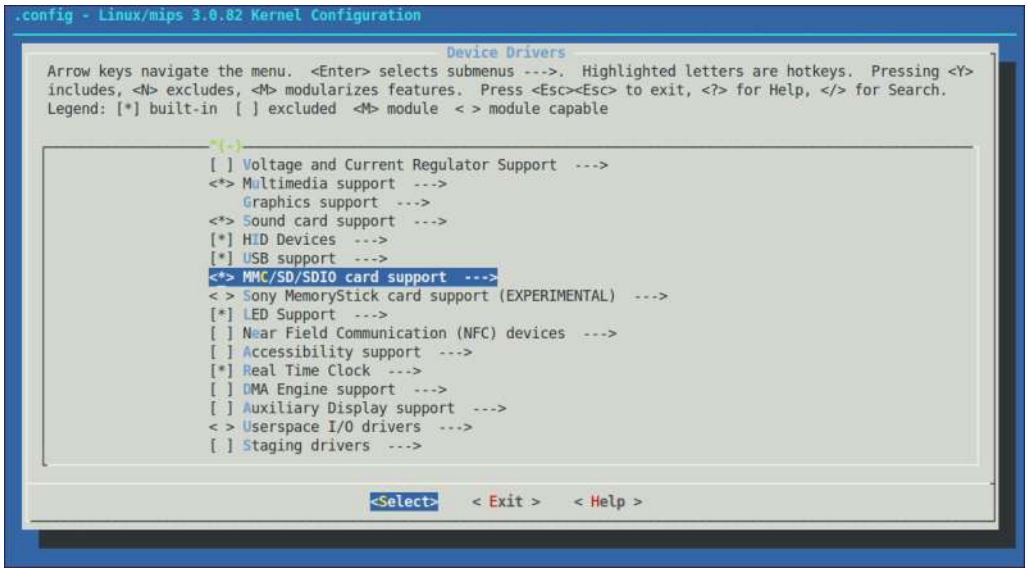
在 Linux 内核配置主界面中选择“Device Drivers　--->”并进入。



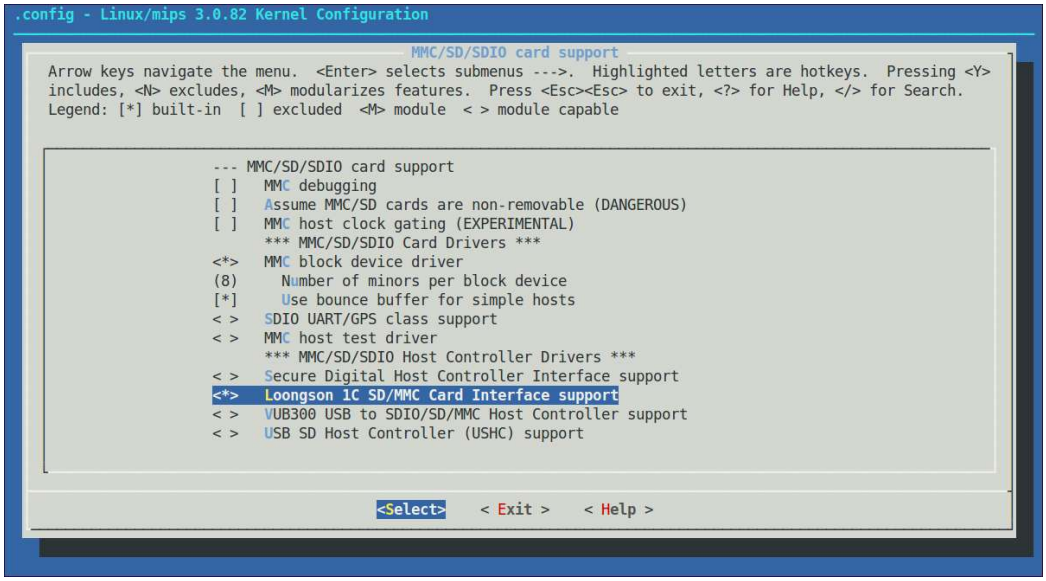
选空“[ ] SPI support　--->”选项。



然后往下找到并选择“<*> MMC/SD/SDIO card support ---> ”，然后进入该选项。



选择“<*> Loongson 1C SD/MMC Card Interface support”。



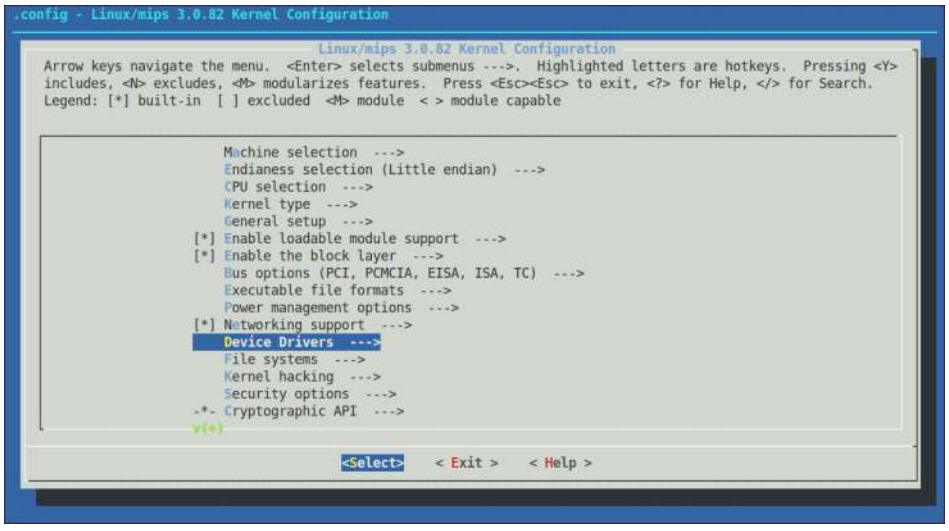
除了在内核上配置相应的驱动之外，PMON 要改为从 NAND FLASH 启动的 PMON（参考 5.2 配置 PMON）。

然后硬件上也要做相应的设置，核心板的拨码开关要设置为从 NAND FLASH 启动并使用 SDIO 和 GPIO2&3&4&5&SD_CS1。（参考 2.2.3.2 拨码开关）

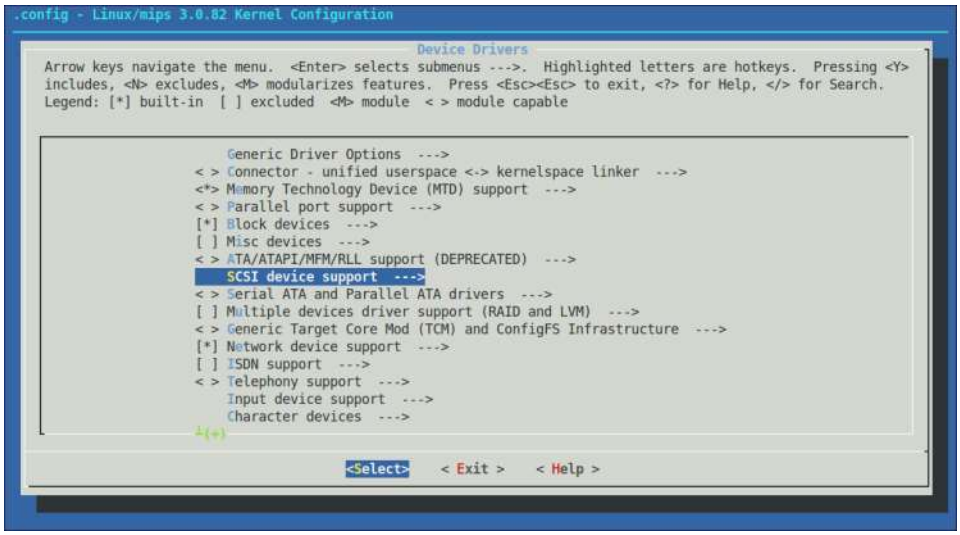
编译并更新开发板内核后，插入 SD 卡，生成设备节点：/dev/mmcblk0p1。

2.3.8 配置 U 盘驱动

在 Linux 内核配置主界面中选择“Device Drivers ---->”并进入。



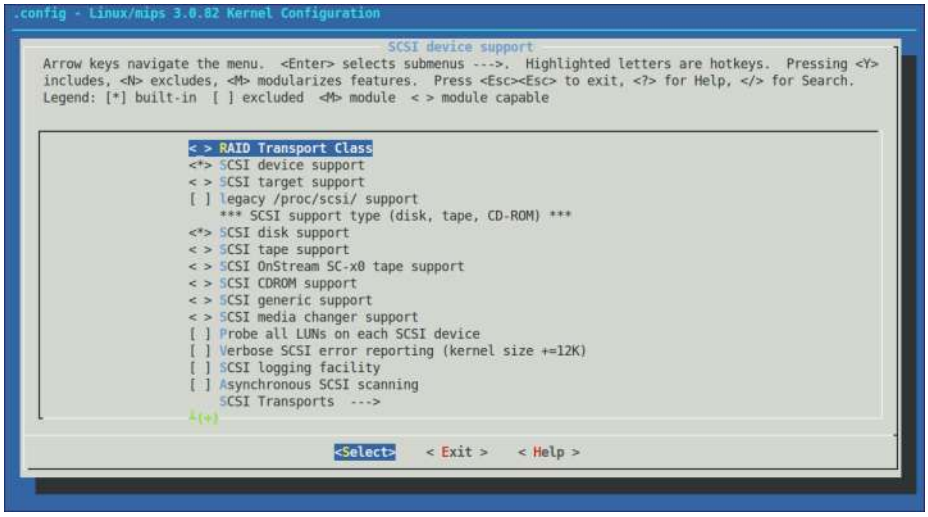
选择“SCSI device support ---->”选项并进入。



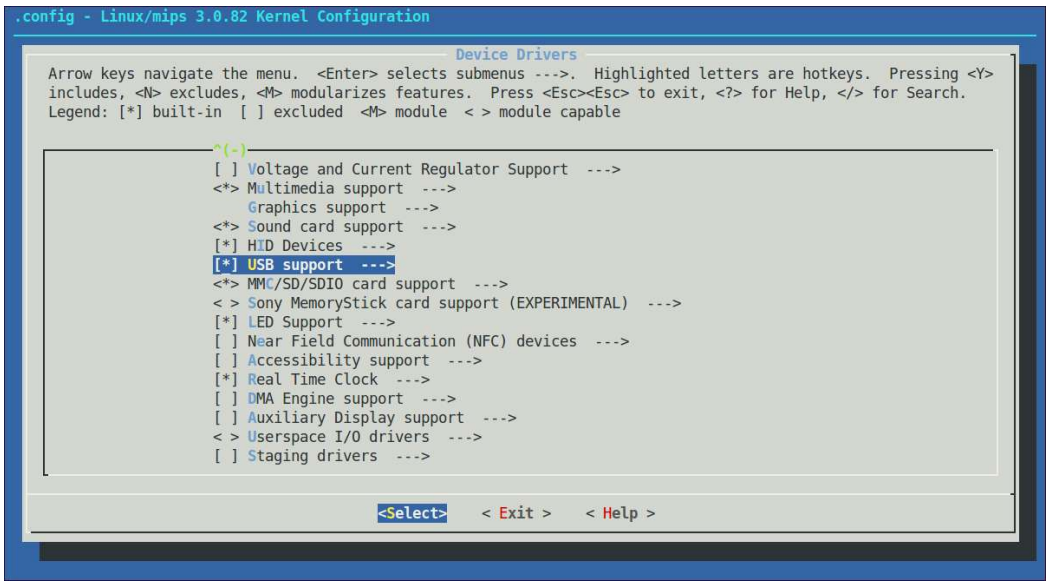
选中如下选项。

<*> SCSI device support

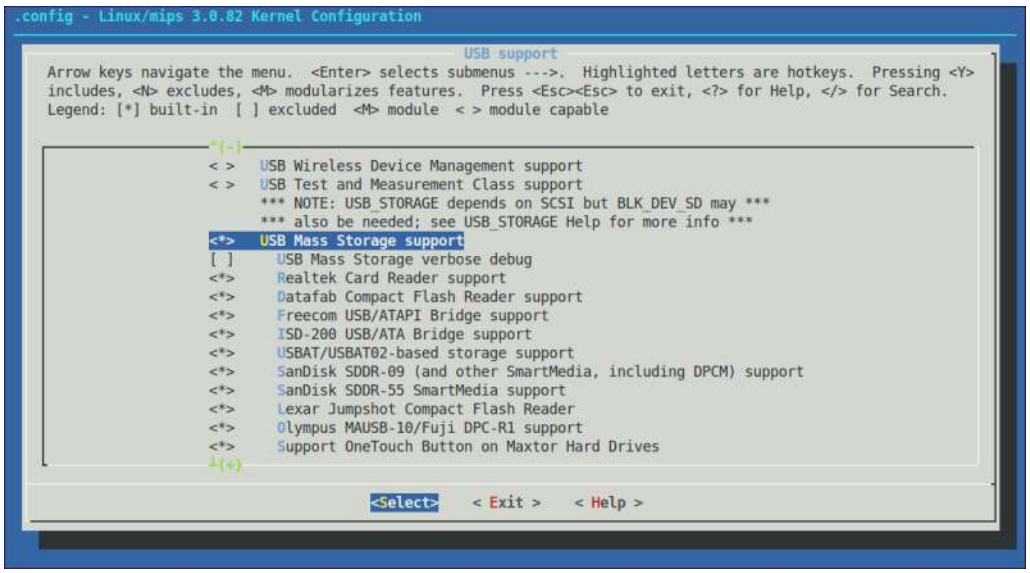
<*> SCSI disk support



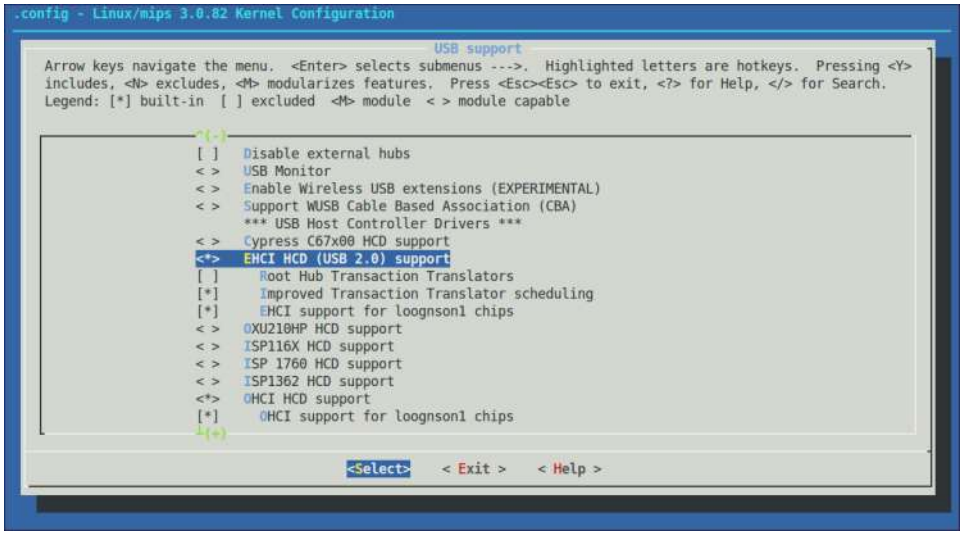
返回上一界面，找到并进入“[*] USB support ---->”选项。



选中“<*> USB Mass Storage support”选项。



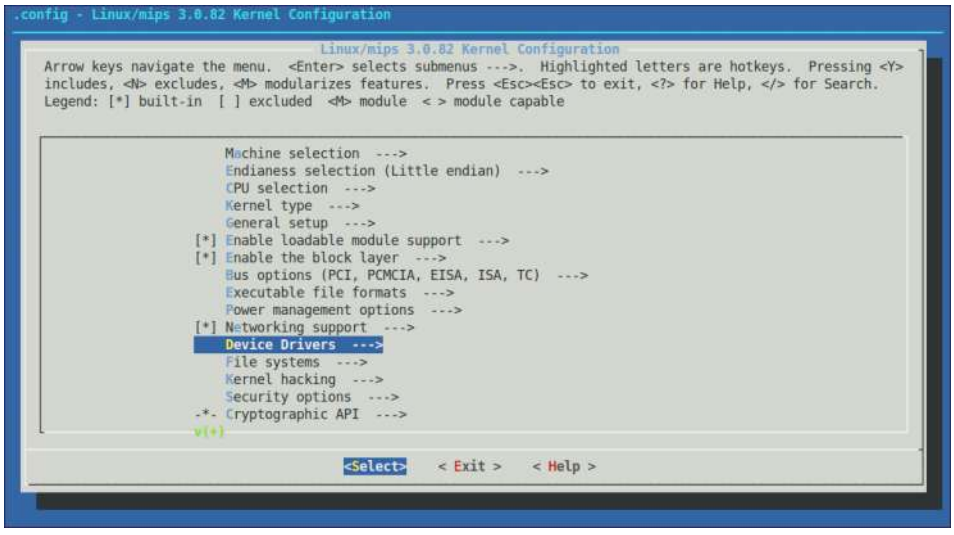
目前大多使用中的 U 盘为 USB 2.0 设备，故需要选中 EHCI 控制器驱动支持，在 USB support 选项卡中选择如下选项：



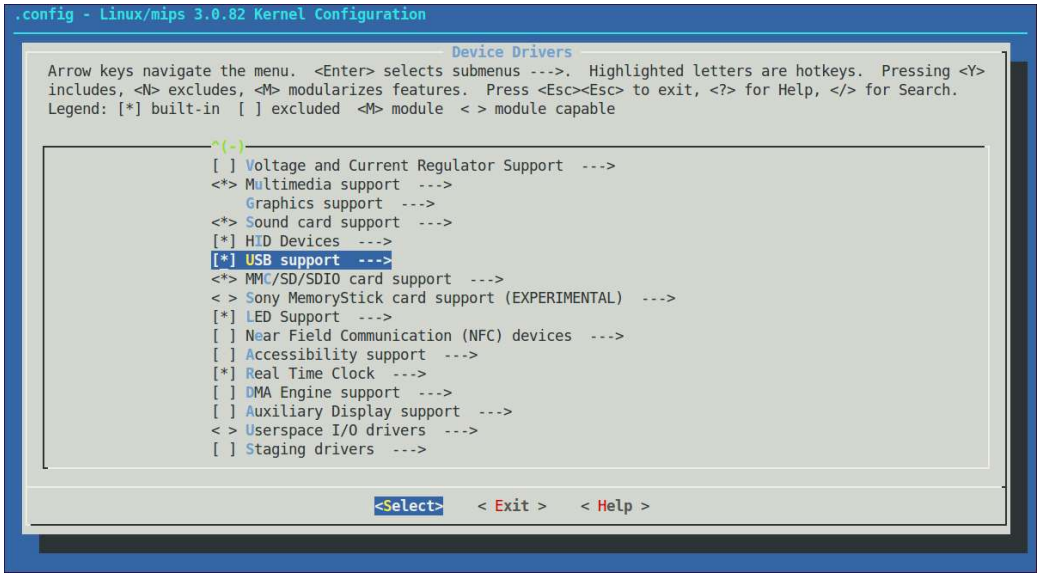
编译并更新开发板内核后，插入 U 盘，生成设备节点：/dev/sda1。

2.3.9 配置 USB 鼠标和键盘驱动

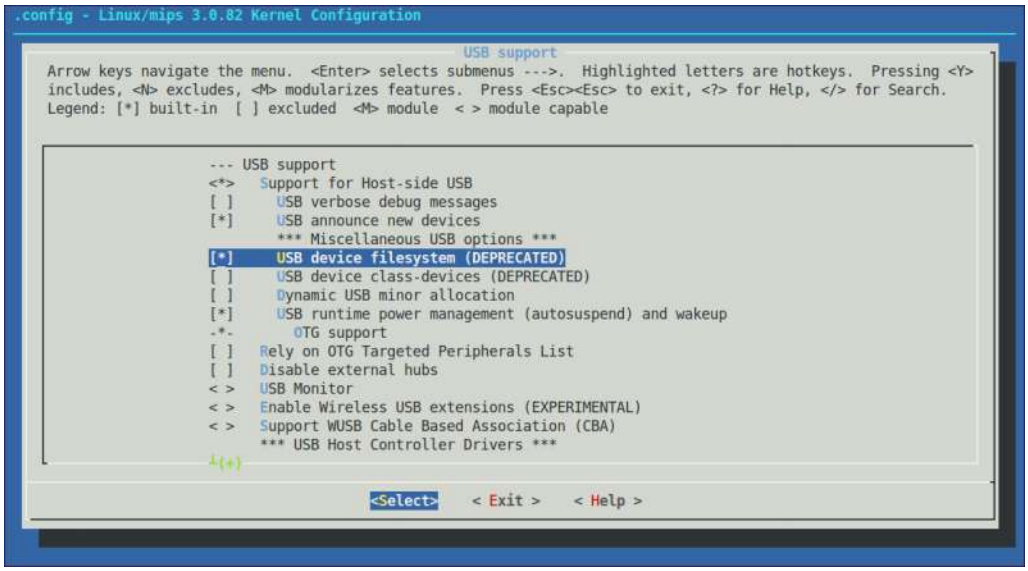
在 Linux 内核配置主界面中选择“Device Drivers ---->”并进入。



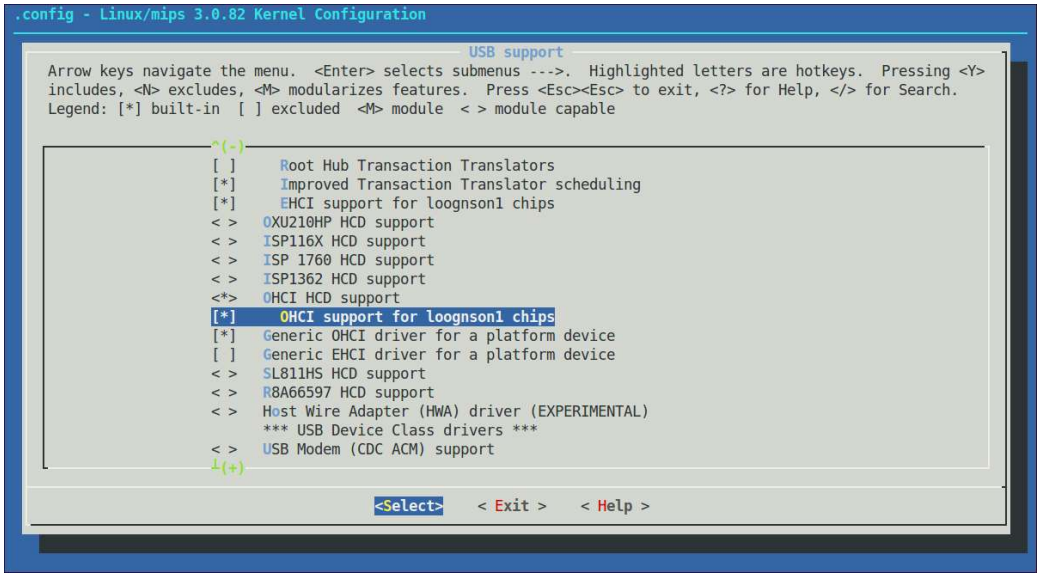
找到“[*] USB support ---->”选项，选择并进入。



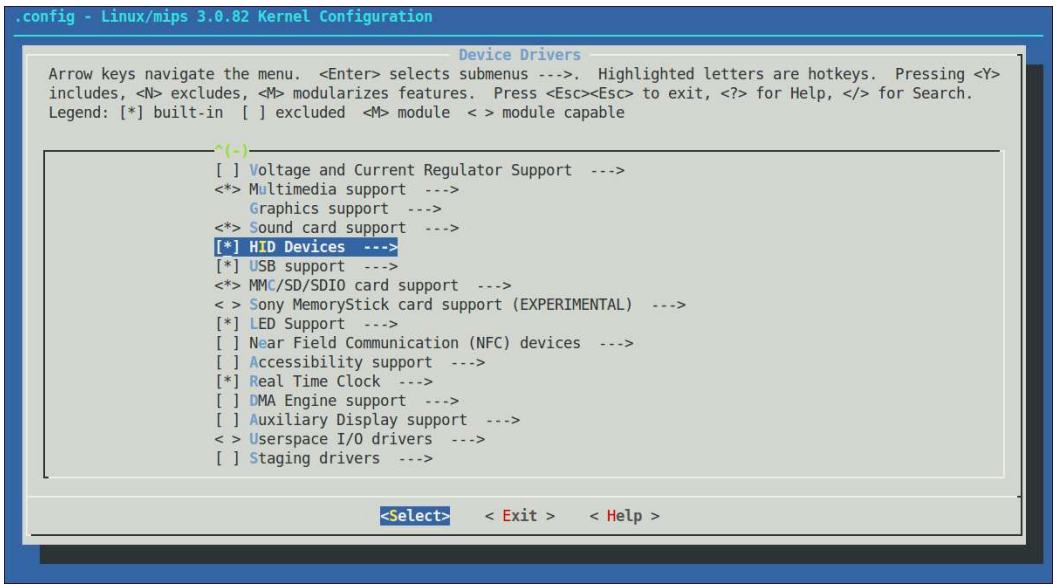
配置 USB support，选择图中所有标有“*”的选项。



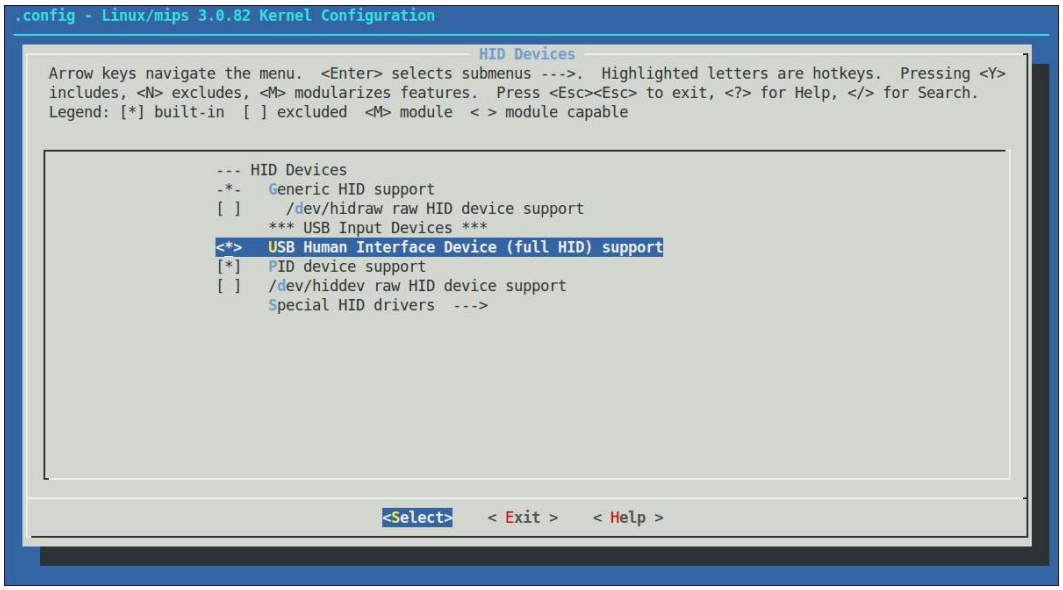
因为键盘和鼠标属于 USB 低速设备，则需要配置 ohci 选项。在该配置界面中往下找到 OHCI 选项并配置，如图示。



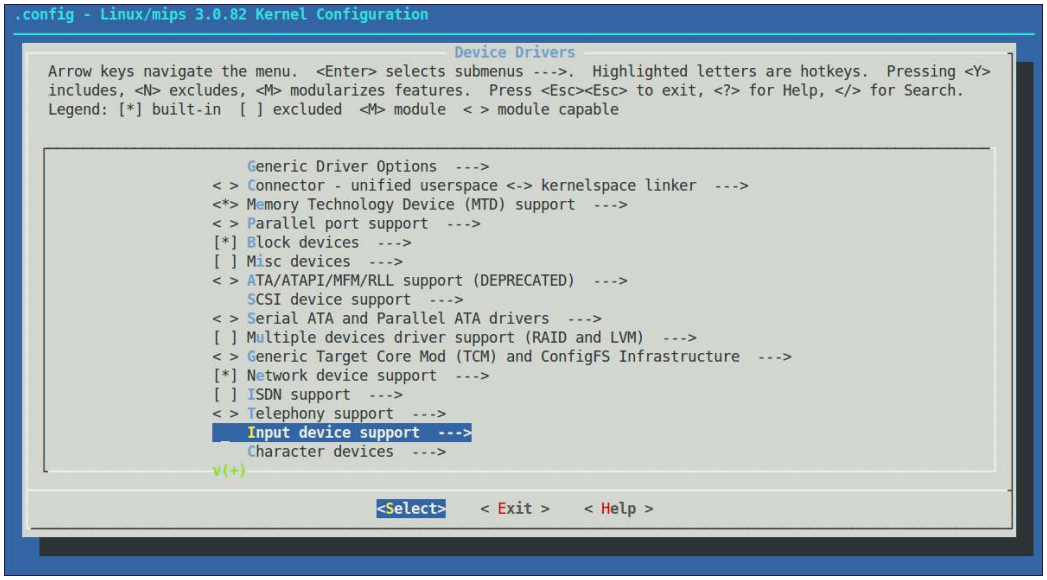
然后退出该界面到“Device Drivers”配置界面，选择并进入“[*] HID Devices --->”选项。



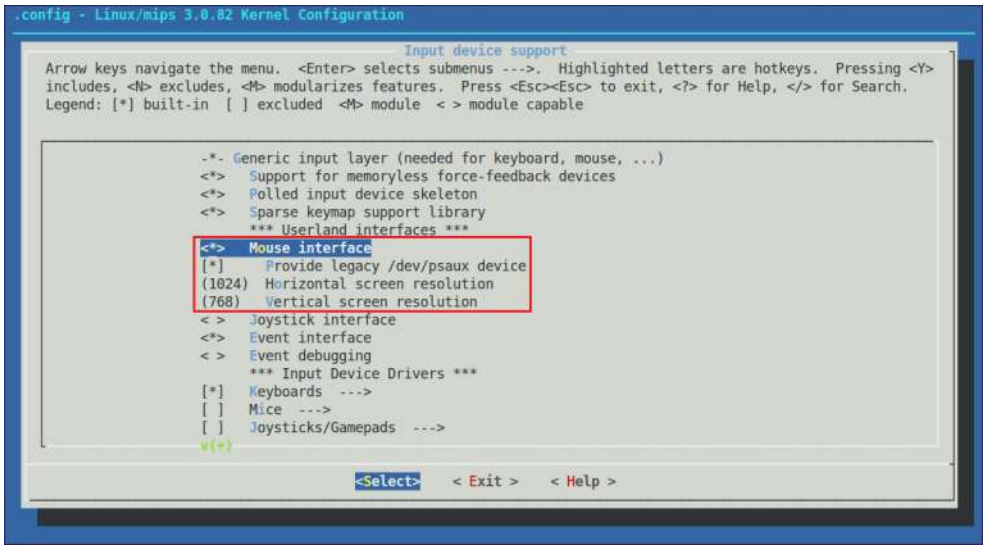
按照下图进行配置。



回到“Device Drivers”配置界面，找到“Input device support”选项并进入。

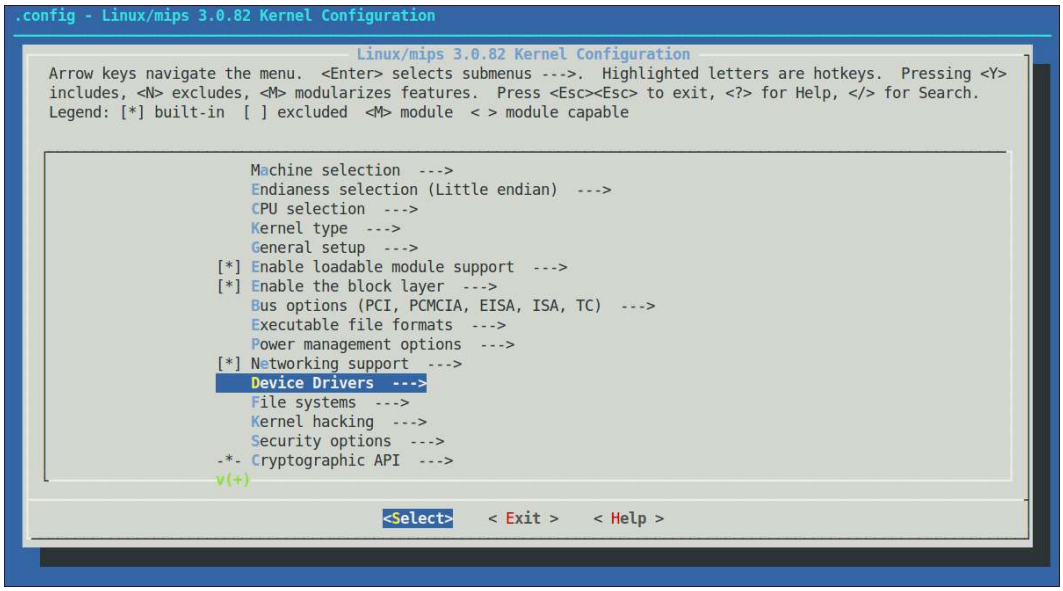


找到“Mouse interface”选项并选上，该选项用于配置鼠标驱动。

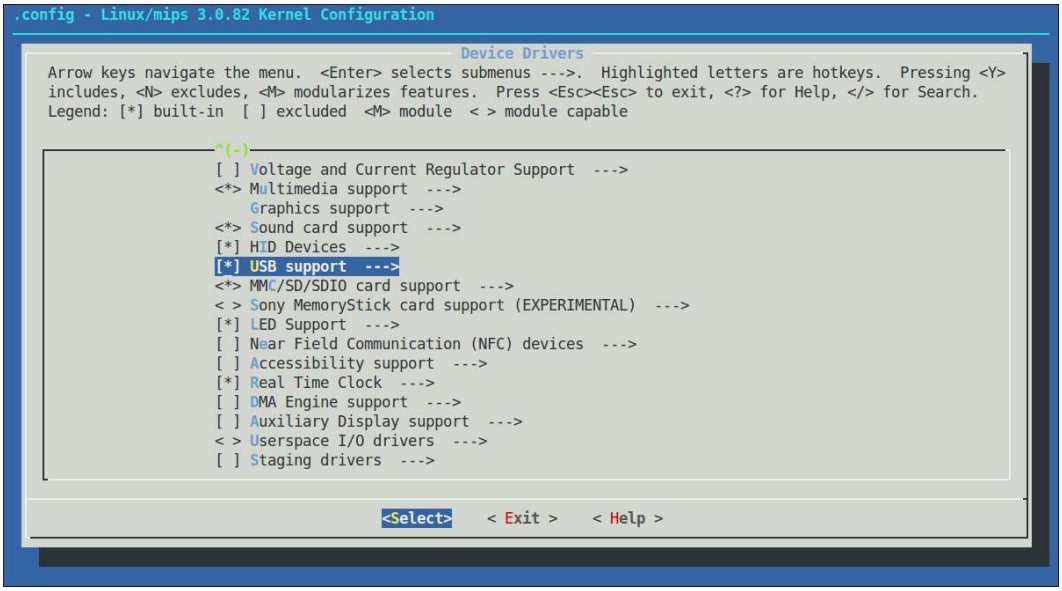


2.3.10 配置 USB OTG 驱动

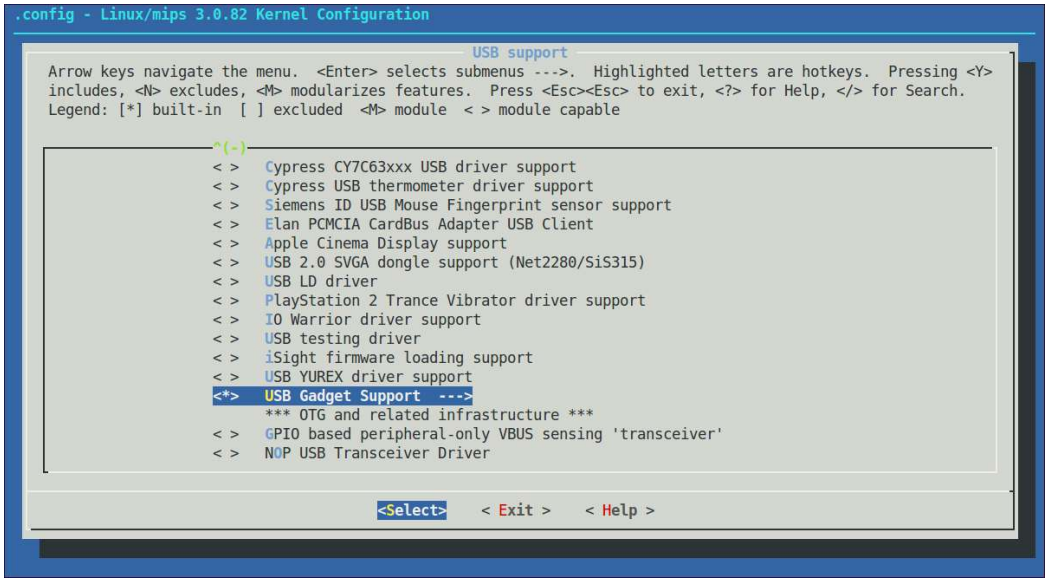
在 Linux 内核配置主界面中选择“Device Drivers ---->”并进入。



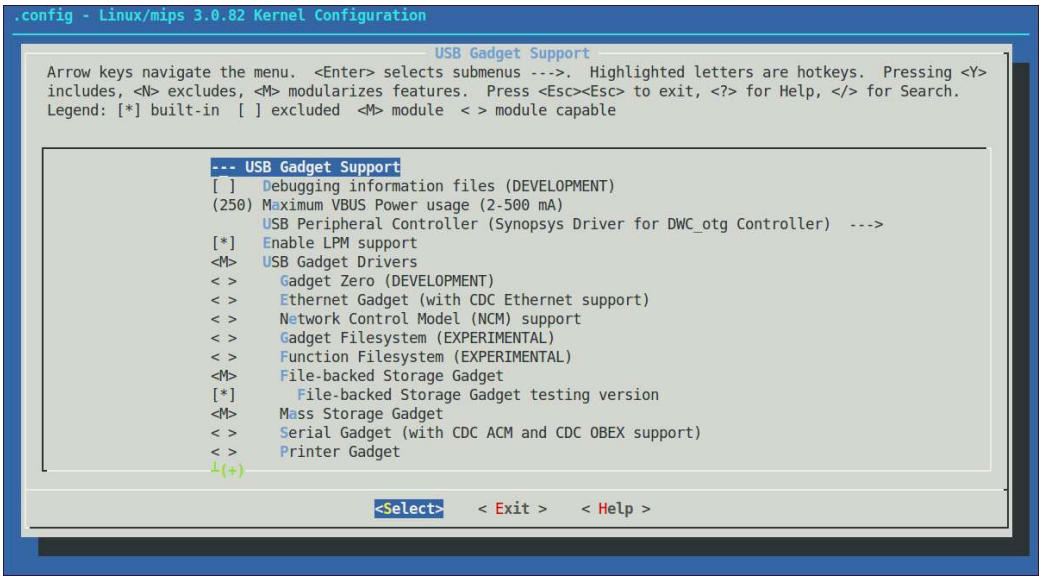
选择“[*] USB support ---->”选项并进入。



找到“<*> USB Gadget Support --->”选项并进入。

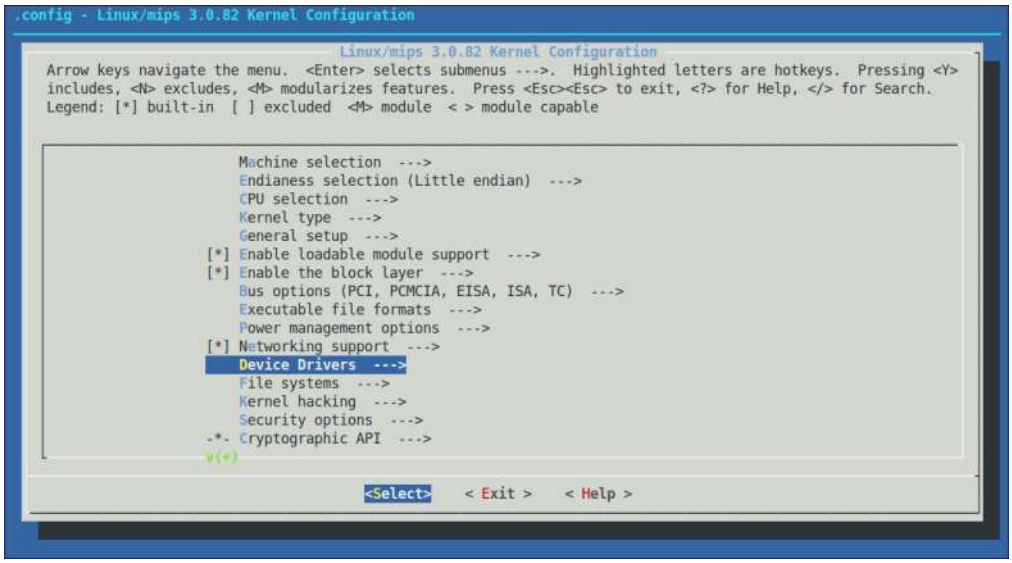


按照下图进入配置。

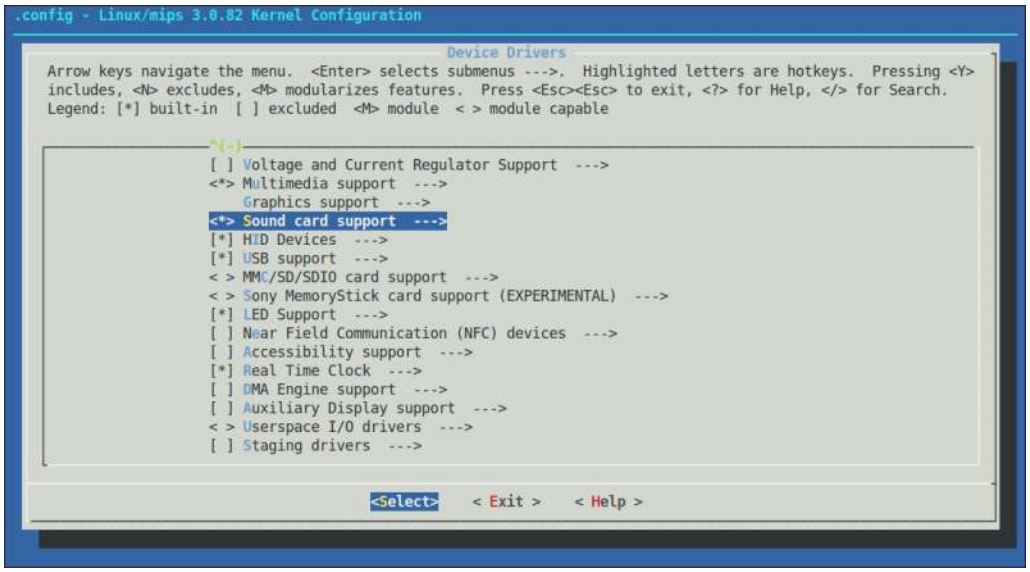


2.3.11 配置音频驱动

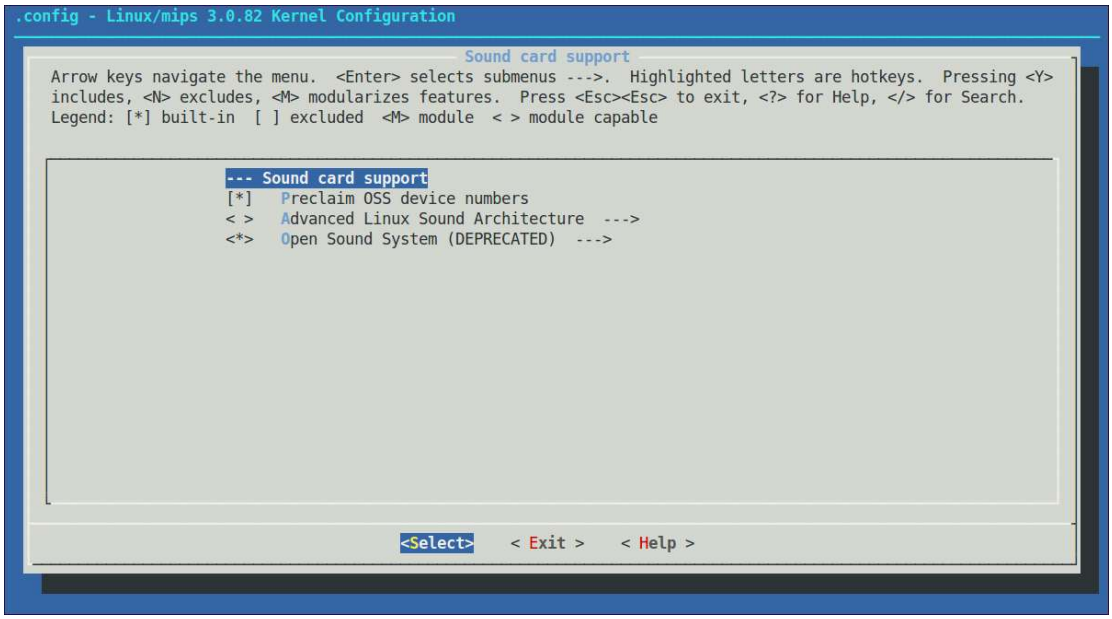
在 Linux 内核配置主界面中选择“Device Drivers ---->”并进入。



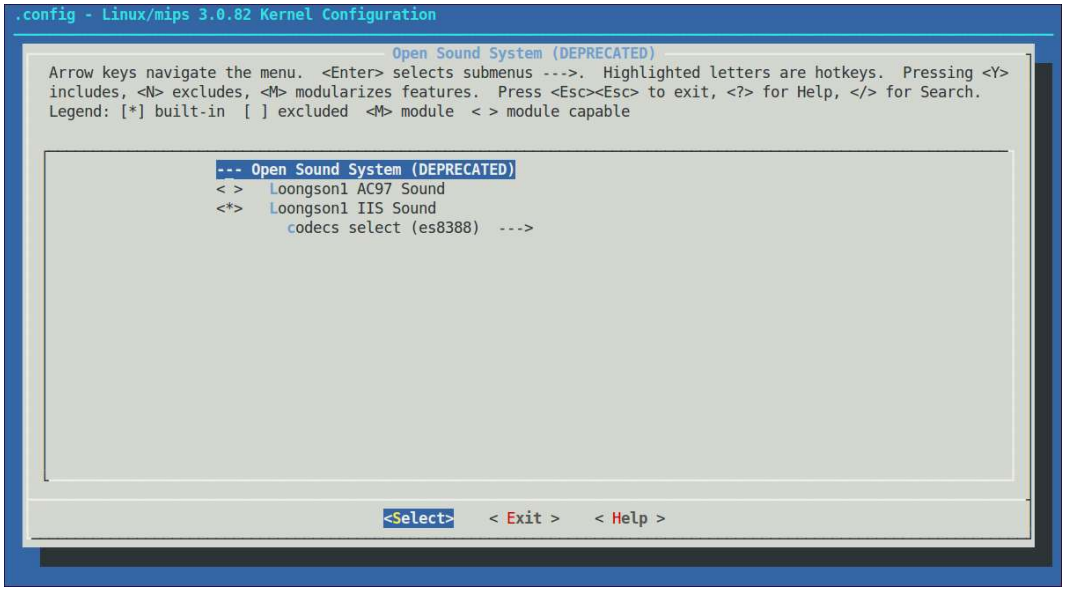
选择“<*> Sound card support ---->”。



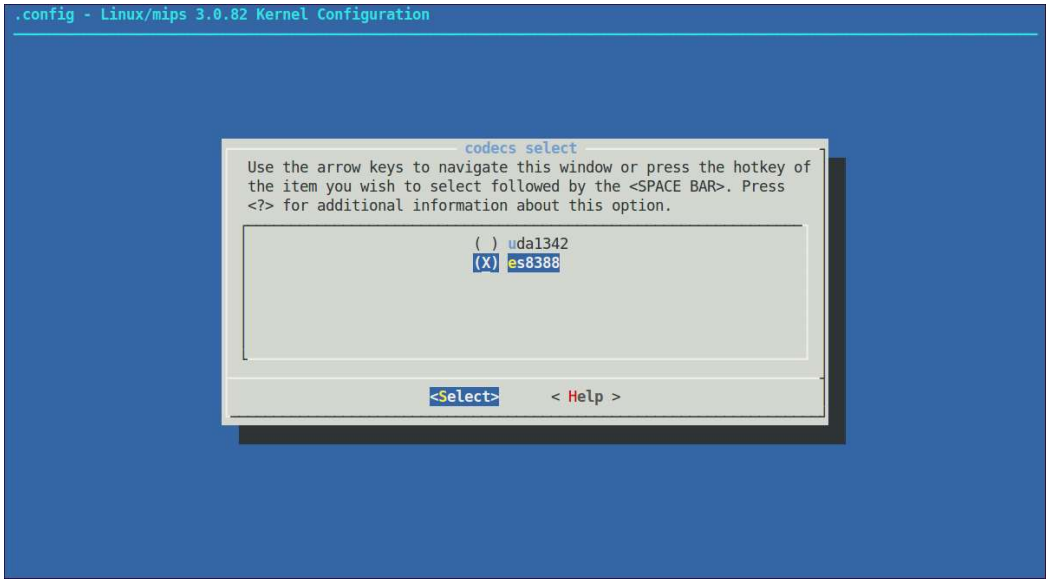
音频驱动默认配置 OSS 驱动。



其中 Open Sound System 具体配置如下。

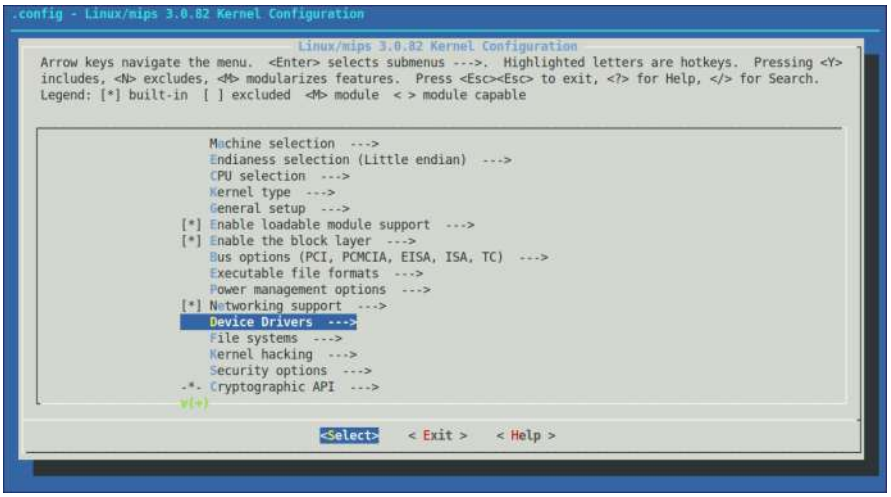


音频解码芯片采用 es8388。

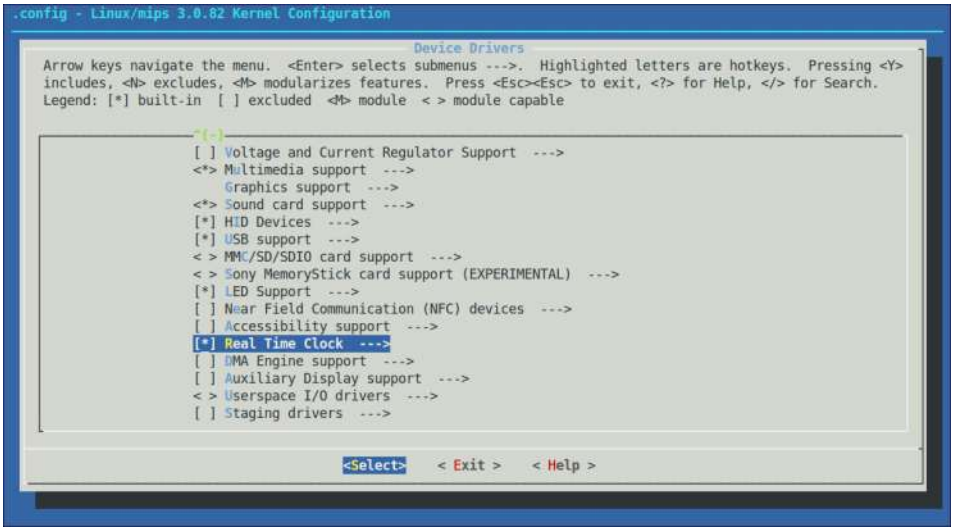


2.3.12 配置 RTC 驱动

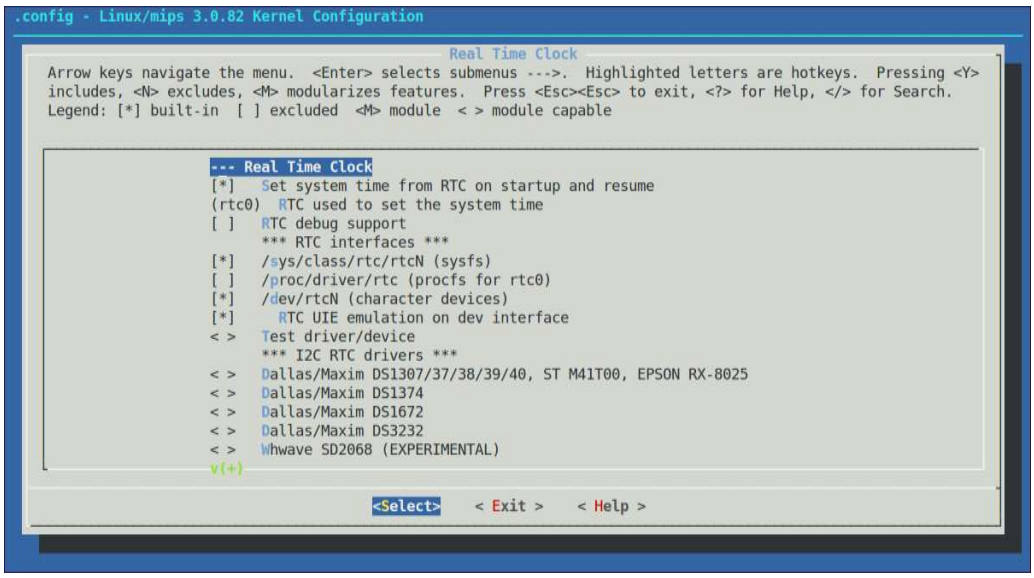
在 Linux 内核配置主界面中选择“Device Drivers ---->”并进入。

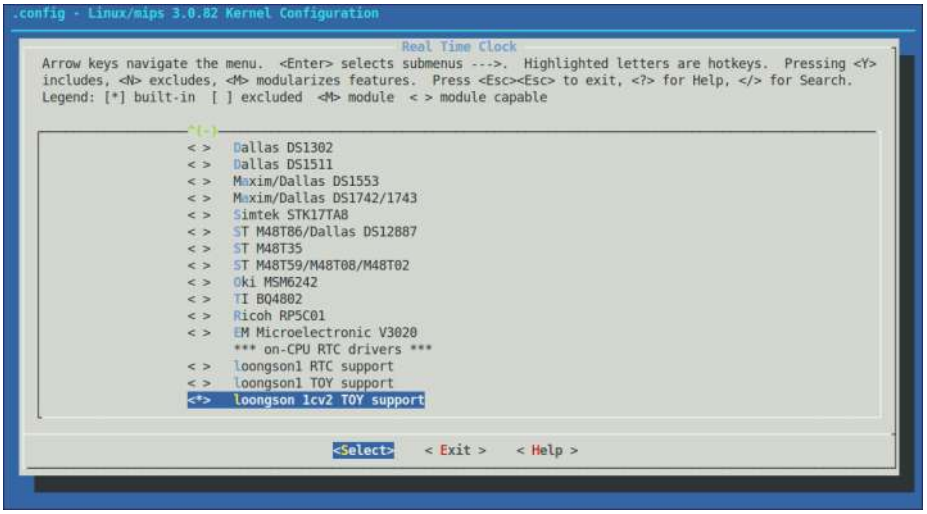


选择“[*] Real Time Clock ---->”。



默认采用如下配置。

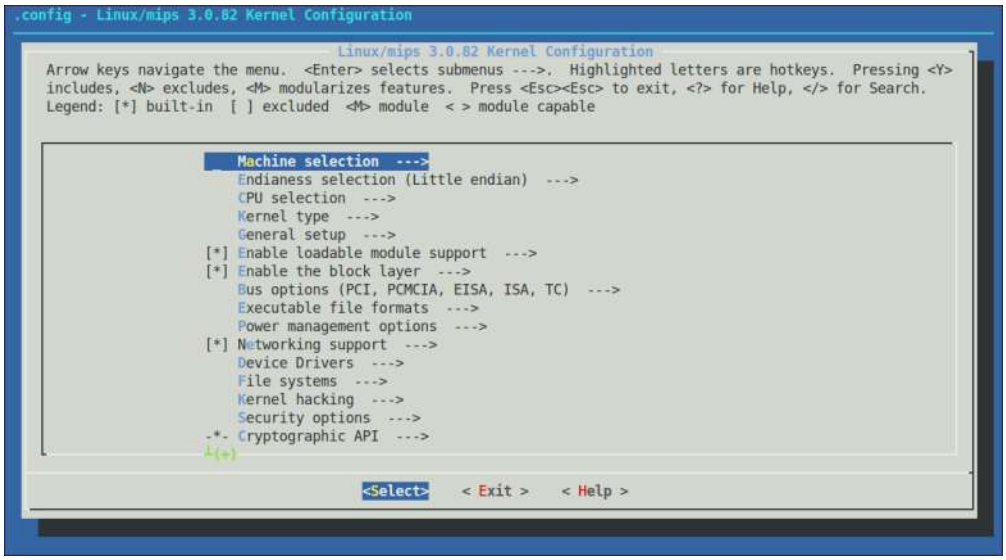




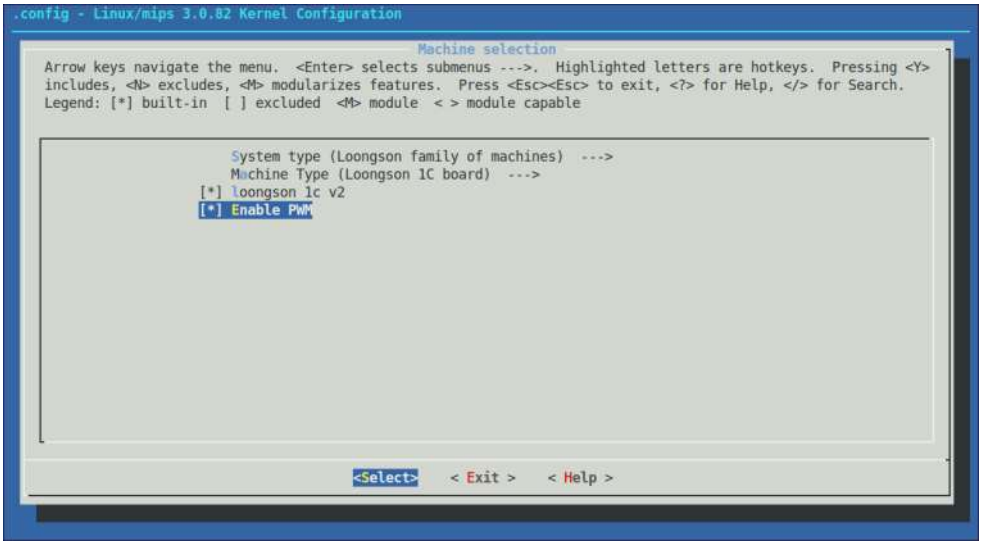
设备节点：/dev/rtc0

2.3.13 配置 PWM 驱动

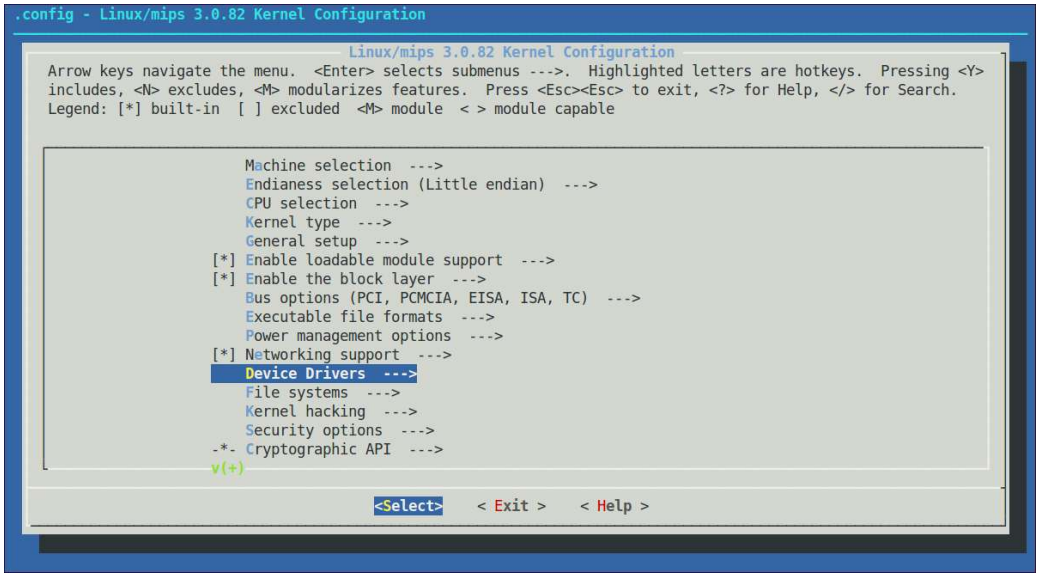
在内核配置的主界面中选择“Machine selection ---->”选项并进入。



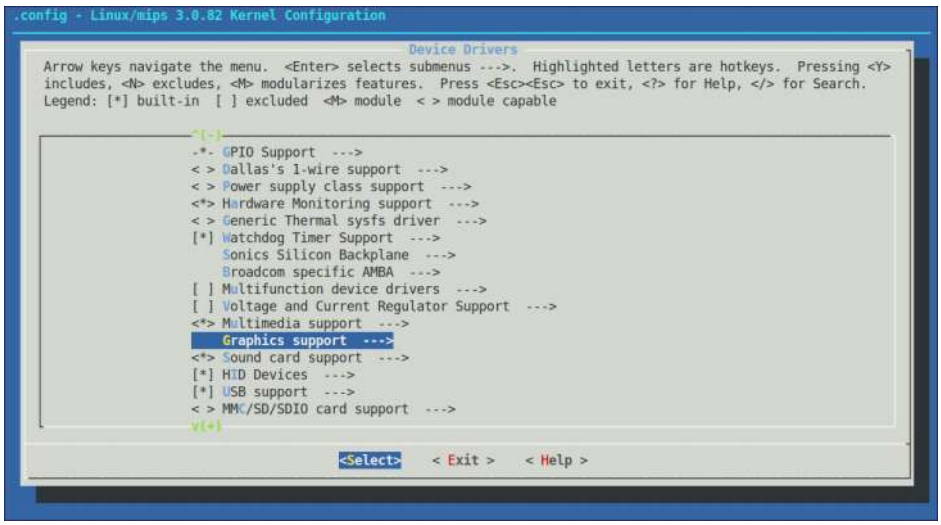
按照下图进行配置。



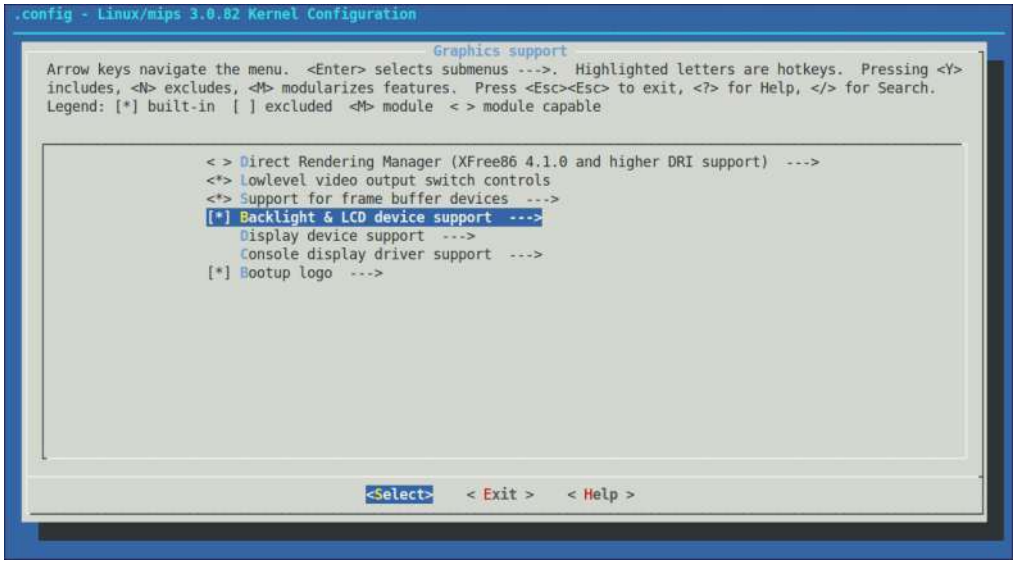
回到内核配置的主界面，选择“Device Drivers”选项并进入。



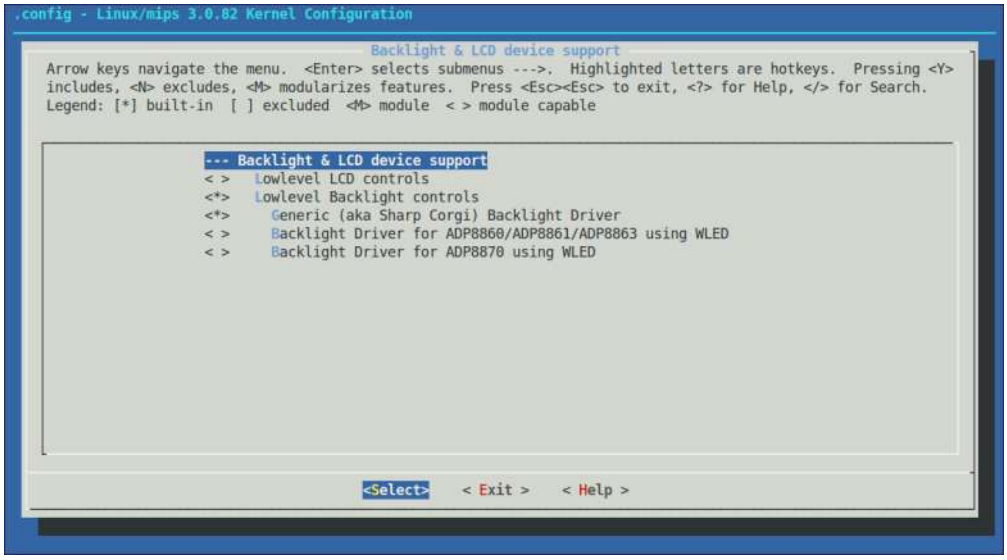
找到图形支持“Graphics support”选项并进入。



选择“Backlight & LCD device support”选项并进入。

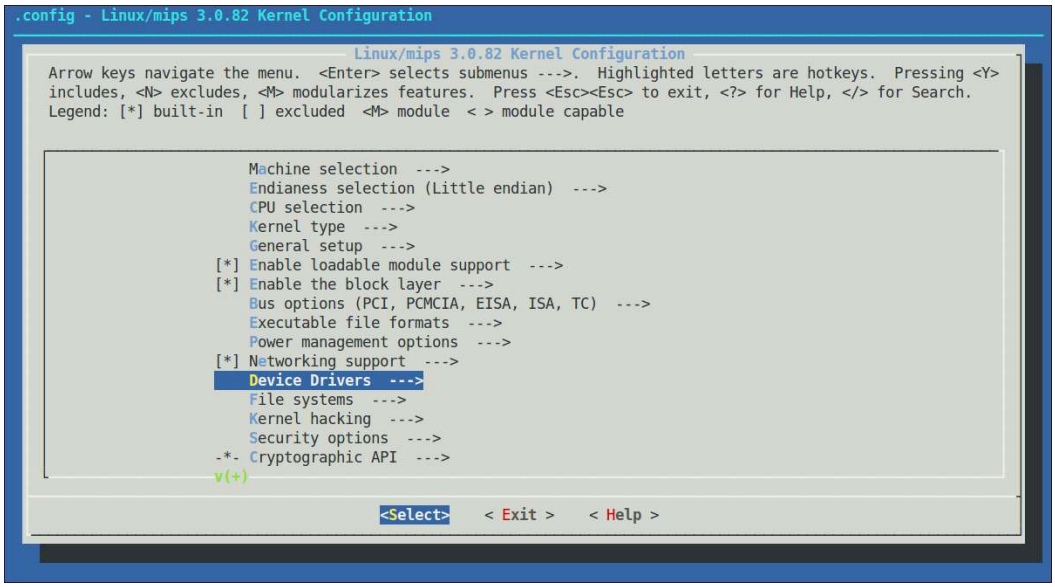


采用如下配置。

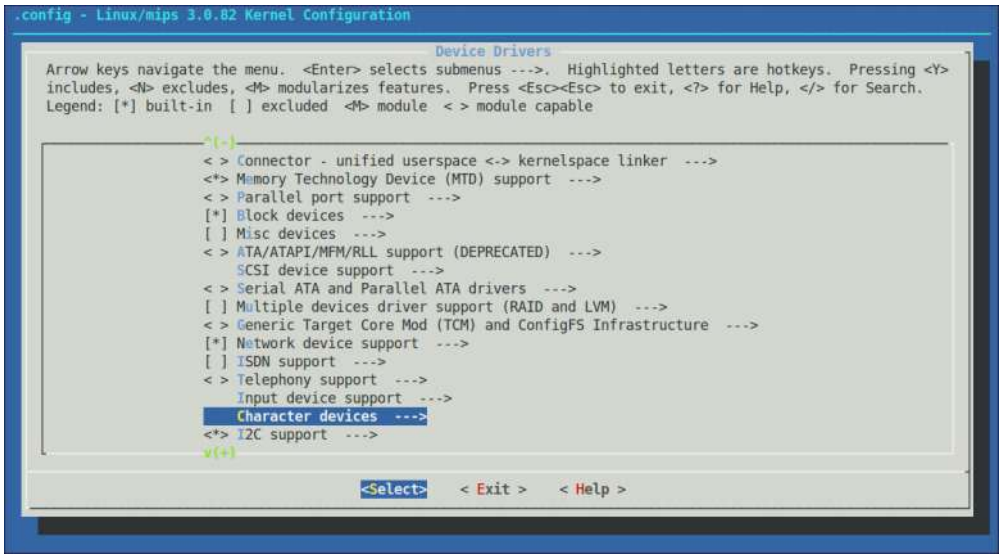


2.3.14 配置红外驱动

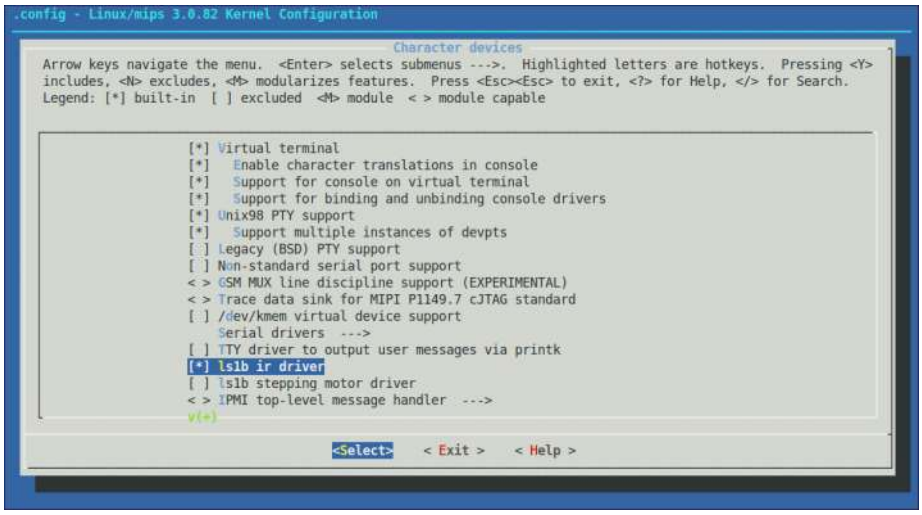
在主菜单界面中，选择“Device Drivers”选项，按回车进入。



找到字符设备“Character devices”选项并进入。



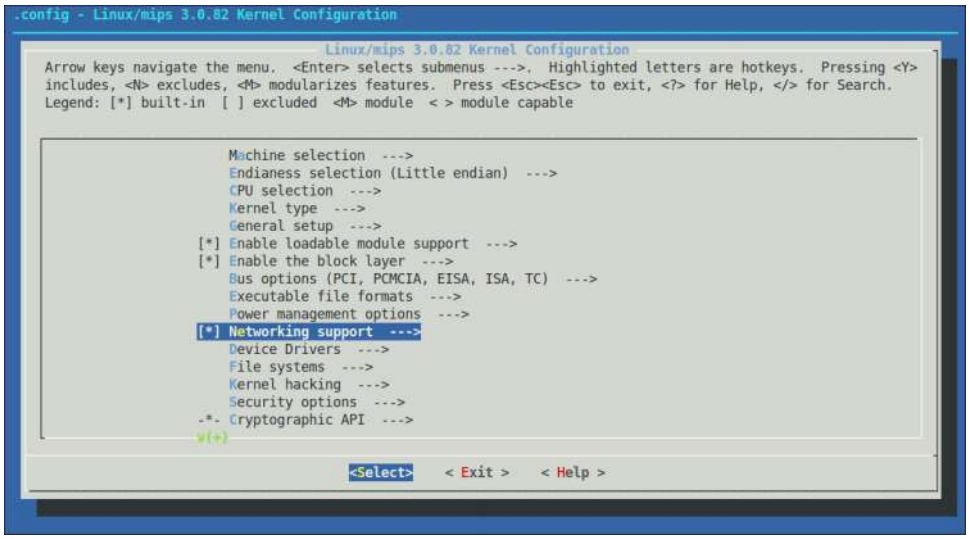
选择红外驱动“ls1b ir driver”选项。



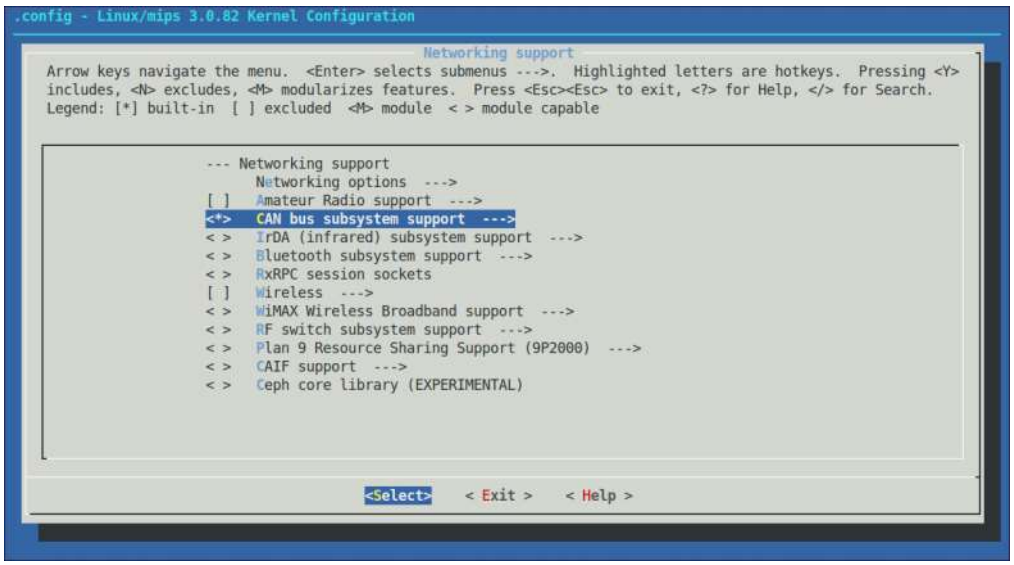
设备节点：/dev/ls1b_ir

2.3.15 配置 CAN 总线驱动

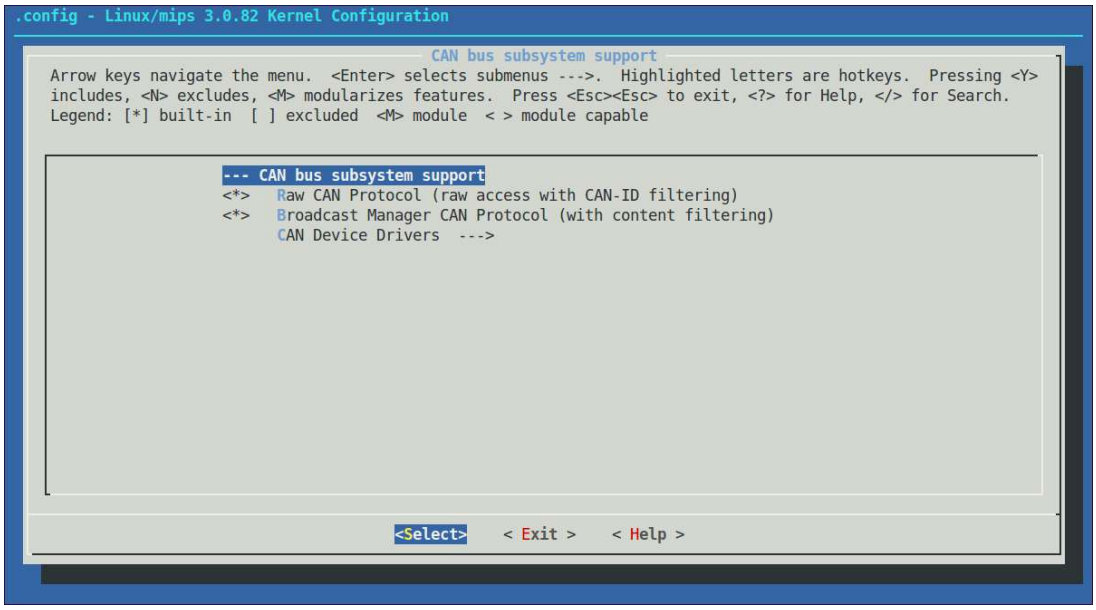
在内核配置的主界面中选择“[*] Networking support --->”选项并进入。



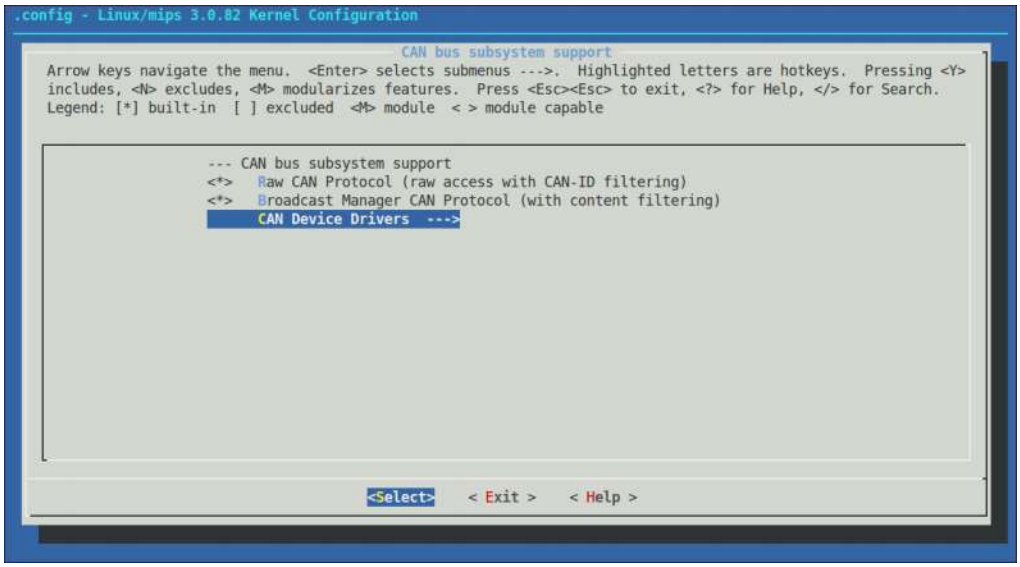
在“Networking support”配置界面配置“<*> CAN bus subsystem support --->”选项并进入。



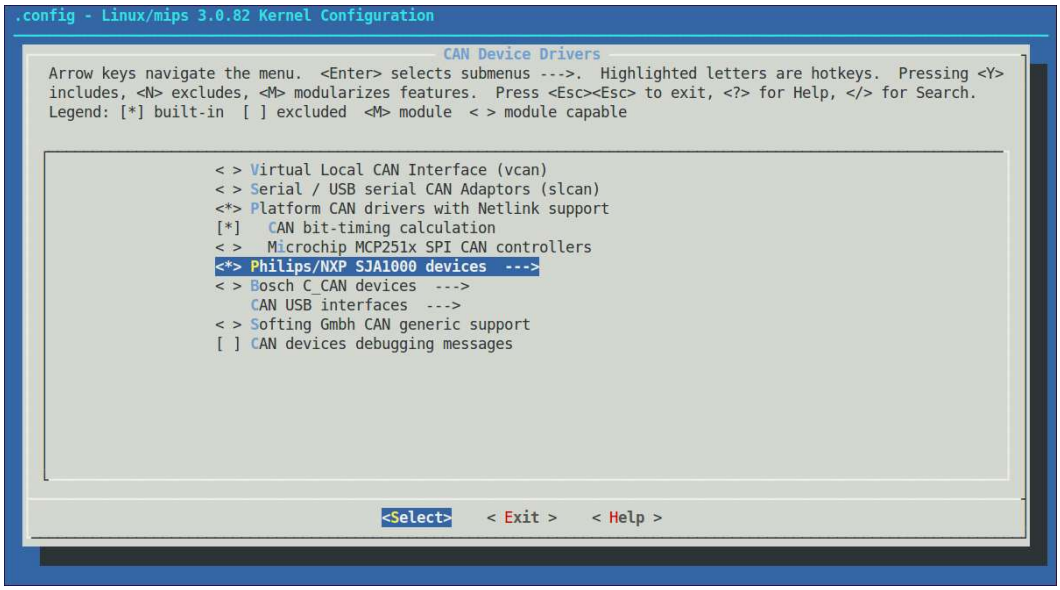
在“CAN bus subsystem support”界面中配置以下两个选项。



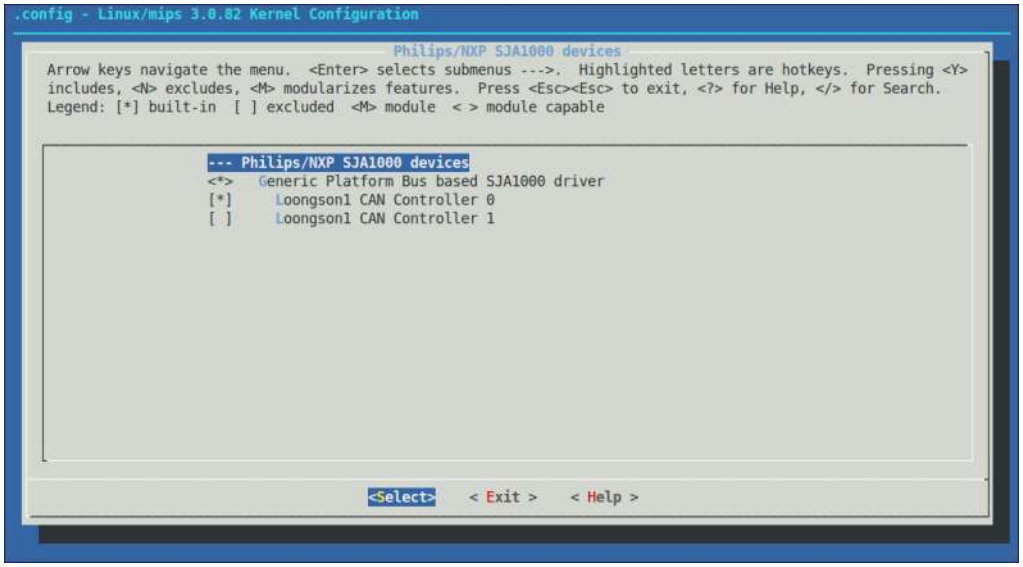
然后选择“CAN Device Drivers ---->”选项并进入。



在“CAN Device Drivers”配置界面中配置好以下三个选项，然后进入“<*> Philips/NXP SJA1000 devices ---->”选项。



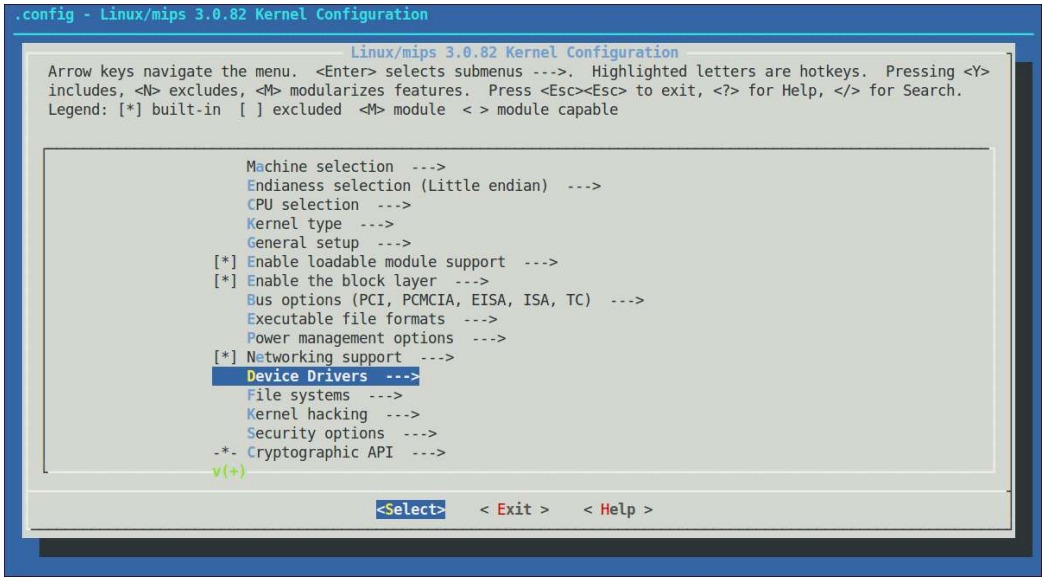
在“Philips/NXP SJA1000 devices”配置界面中配置好以下选项。



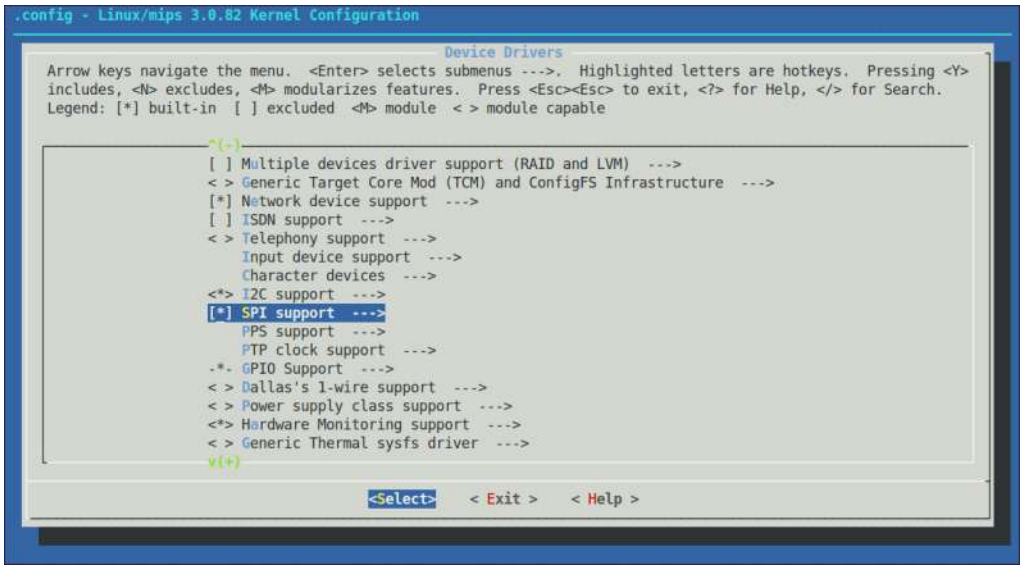
配置完 CAN 总线驱动后还要配置 SPI 控制器驱动（参考 “6.5.16 配置 SPI 控制器驱动”）。

2.3.16 配置 SPI 控制器驱动

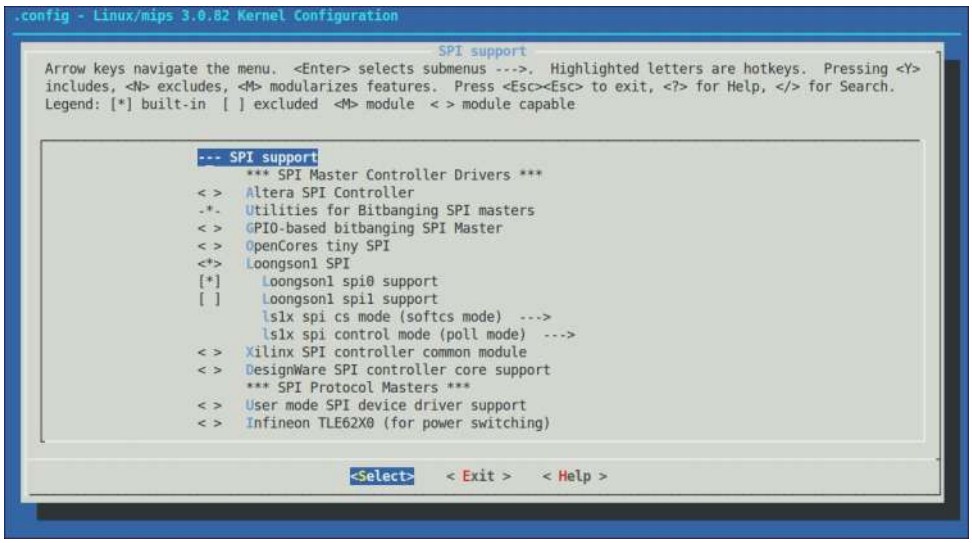
在 Linux 内核配置主界面中选择“Device Drivers ---->”并进入。



选择“[*] SPI support ---->”。

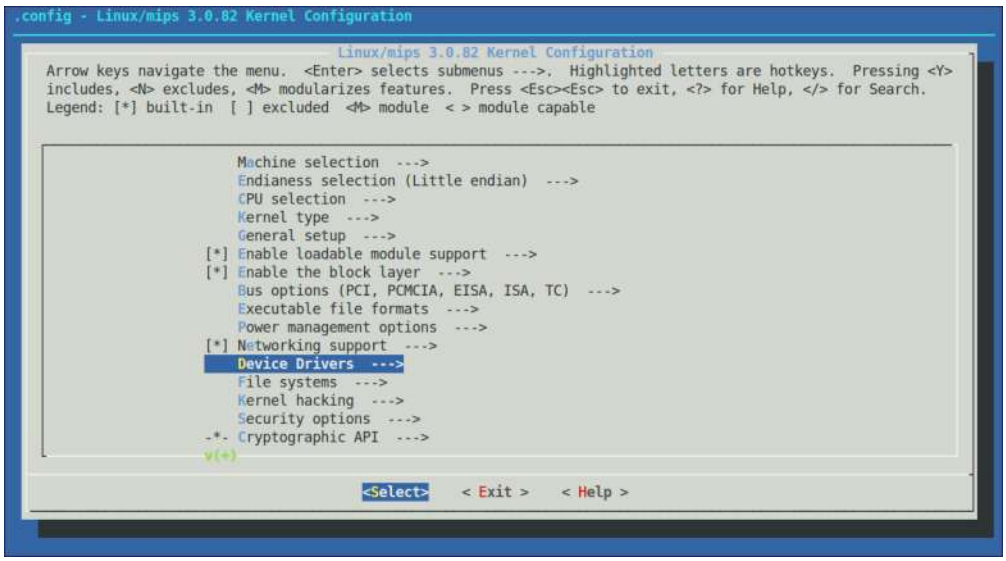


默认采用如下配置。

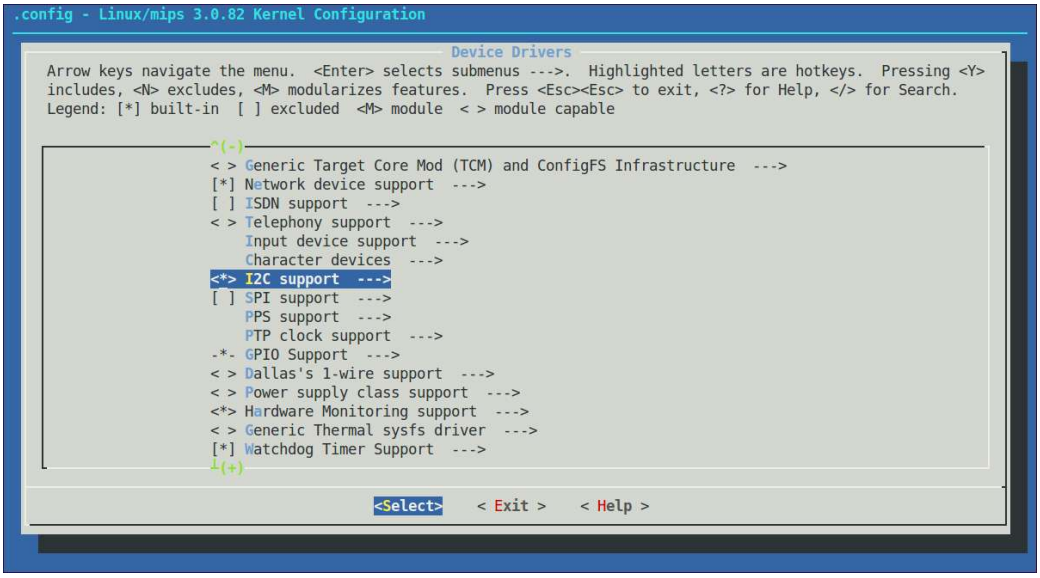


2.3.17 配置 I2C 控制器驱动

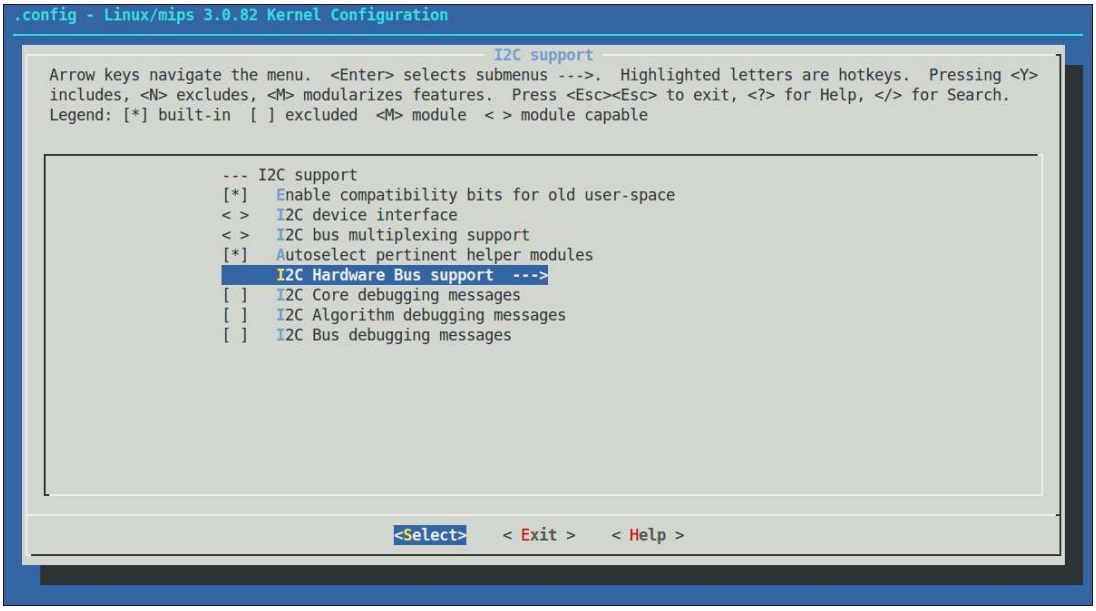
在 Linux 内核配置主界面中选择“Device Drivers ---->”并进入。



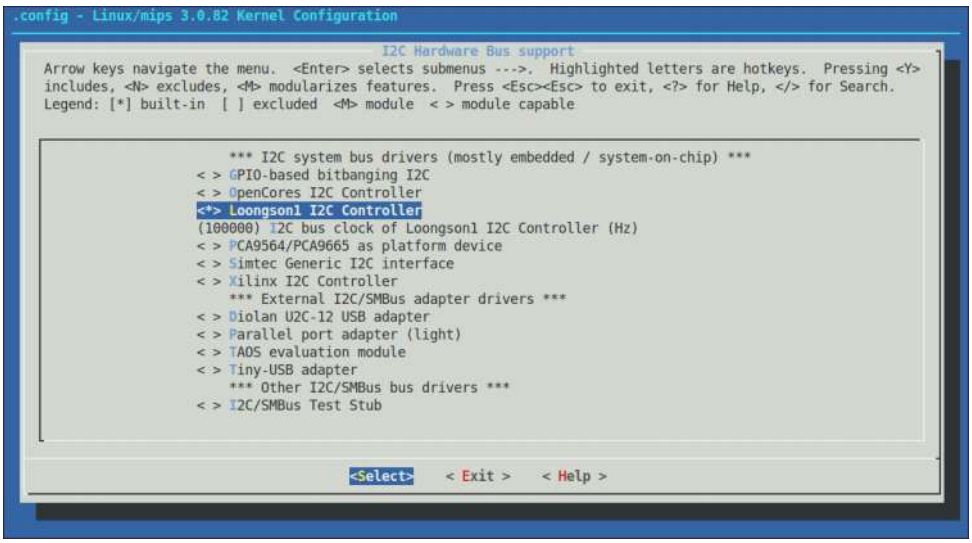
选择“<*> I2C support ---->”选项并进入。



在“I2C support”配置界面下，配置下图中带“*”的选项。然后选择“I2C Hardware Bus support --->”并进入下一个配置界面。

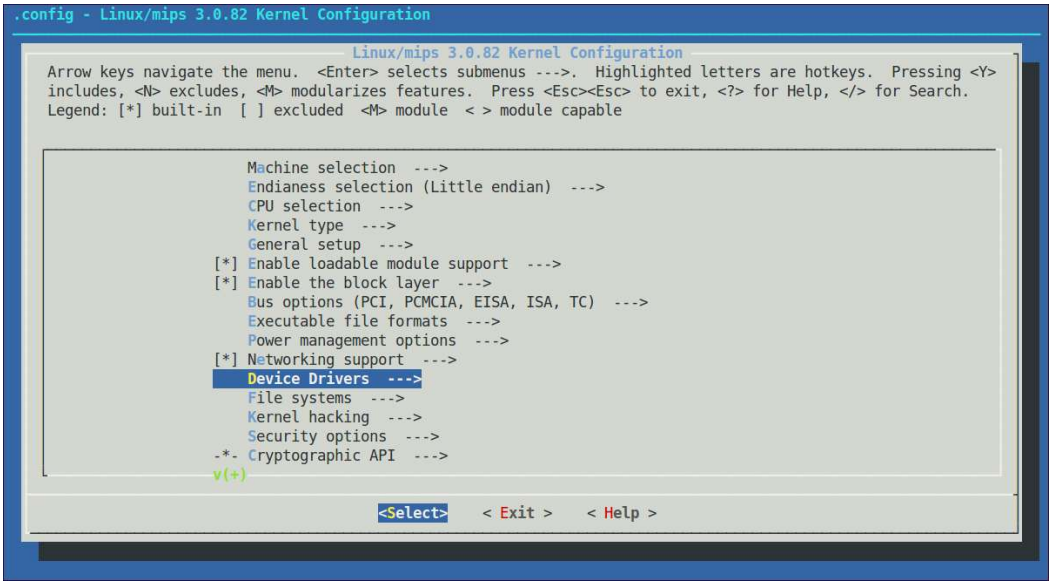


按照下图进行配置。

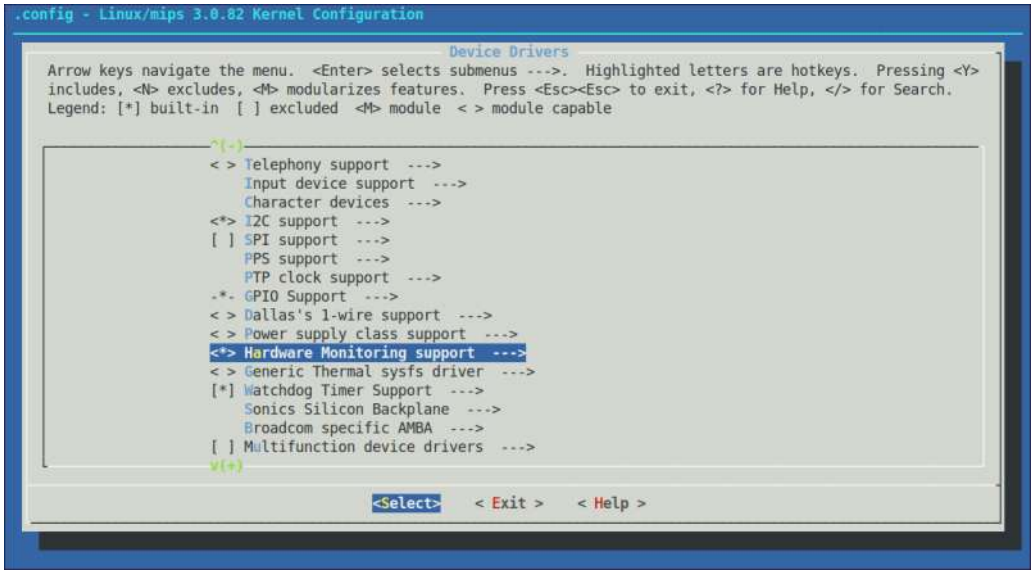


2.3.18 配置 ADC 驱动

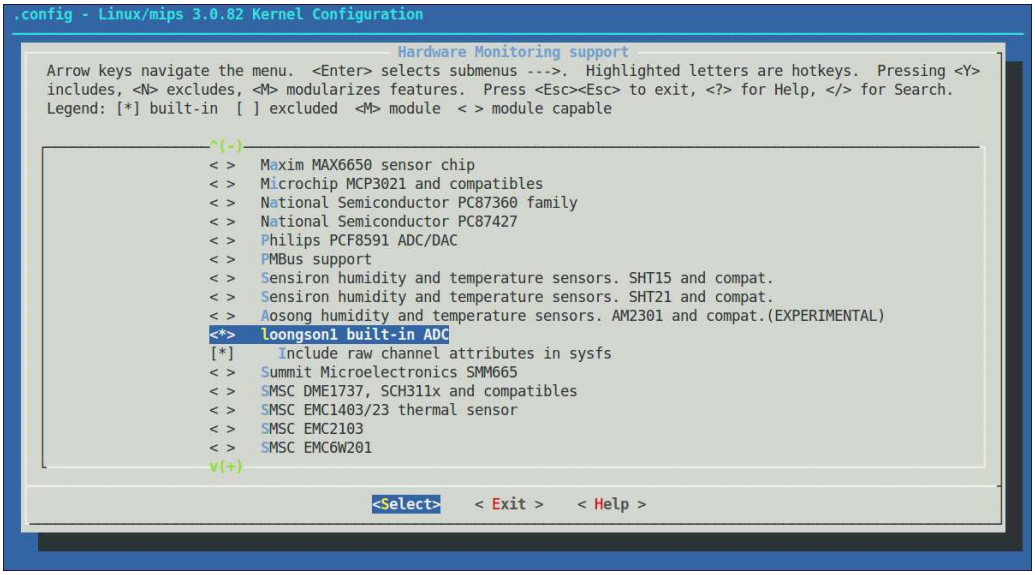
在主菜单界面中，选择“Device Drivers”选项，按回车进入。



选择“Hardware Monitoring support”选项并进入。

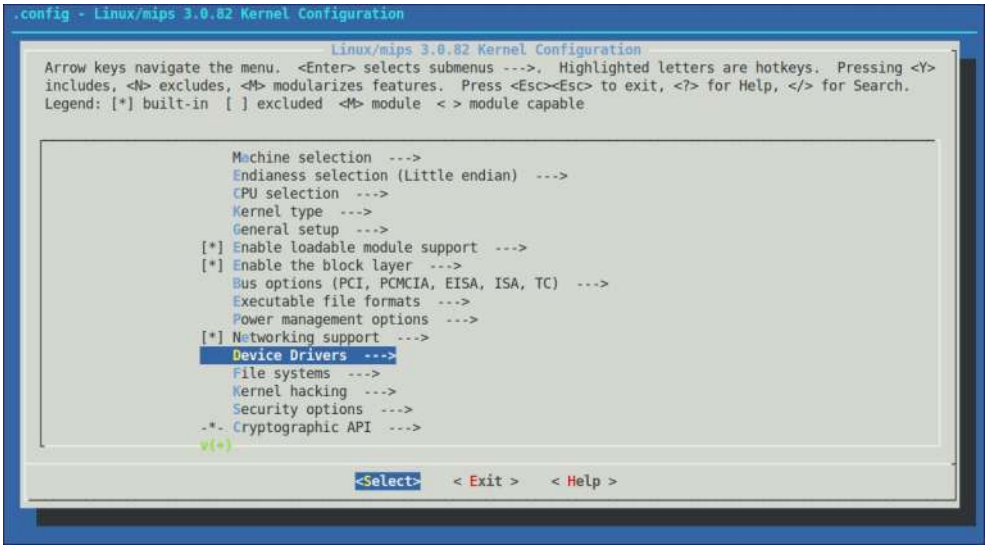


配置如下选项。

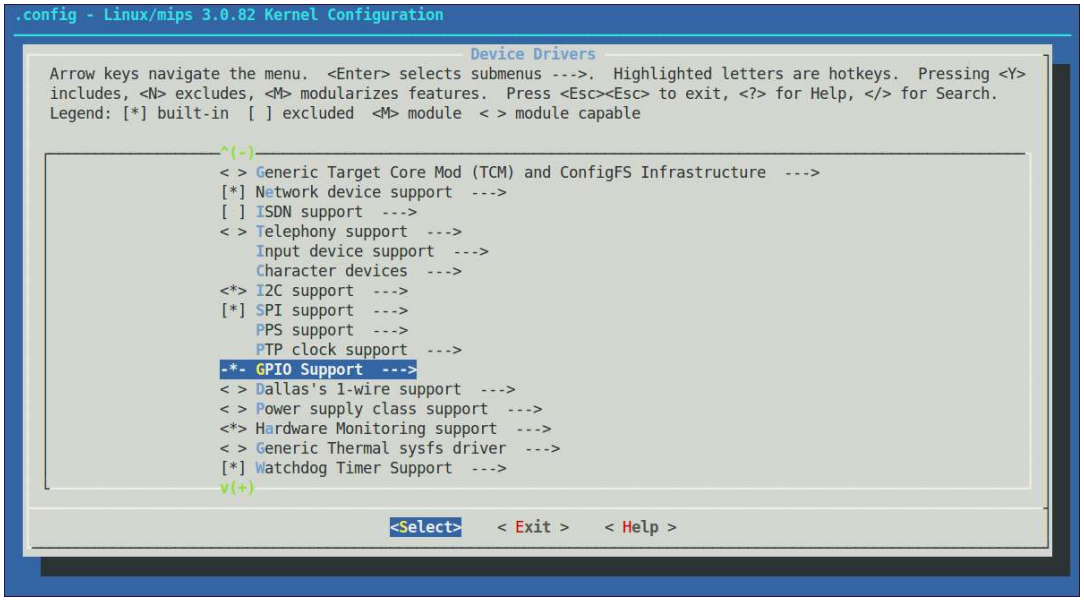


2.3.19 配置 GPIO 驱动

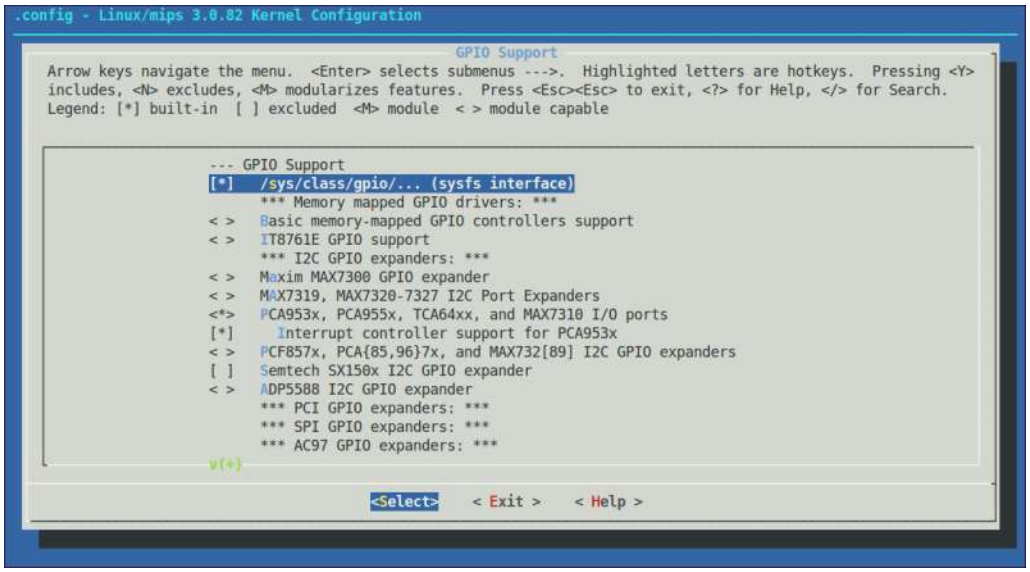
在主菜单界面中，选择“Device Drivers”选项，按回车进入。



选择 GPIO 支持“GPIO Support”选项并进入。

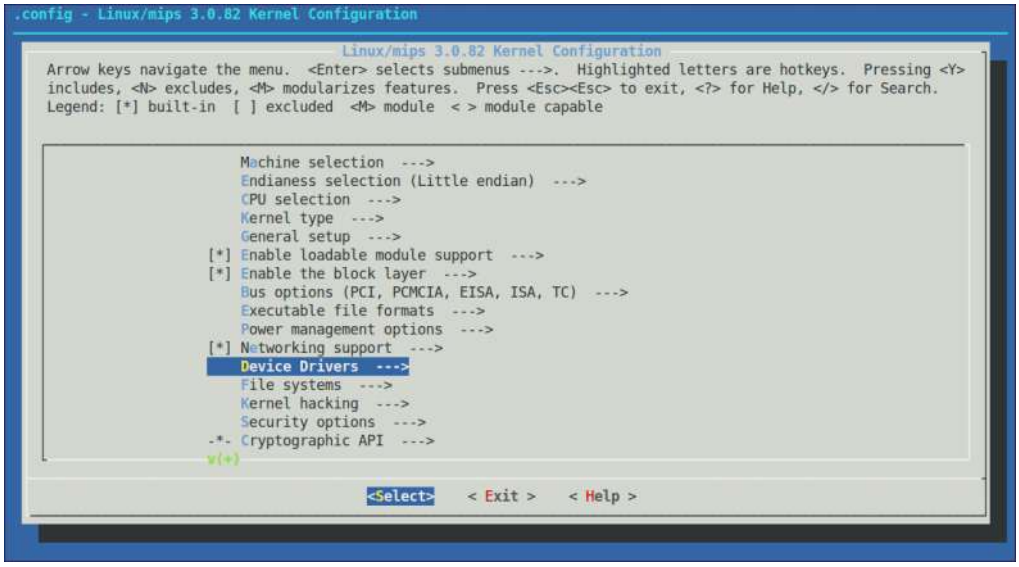


配置 GPIO 的 sysfs 接口。

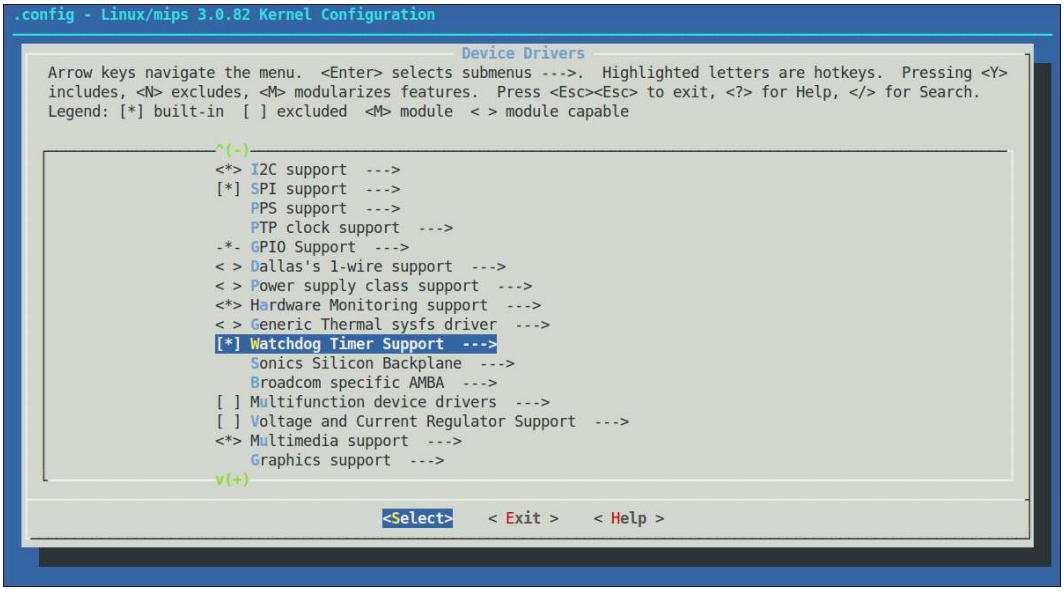


2.3.20 配置看门狗驱动

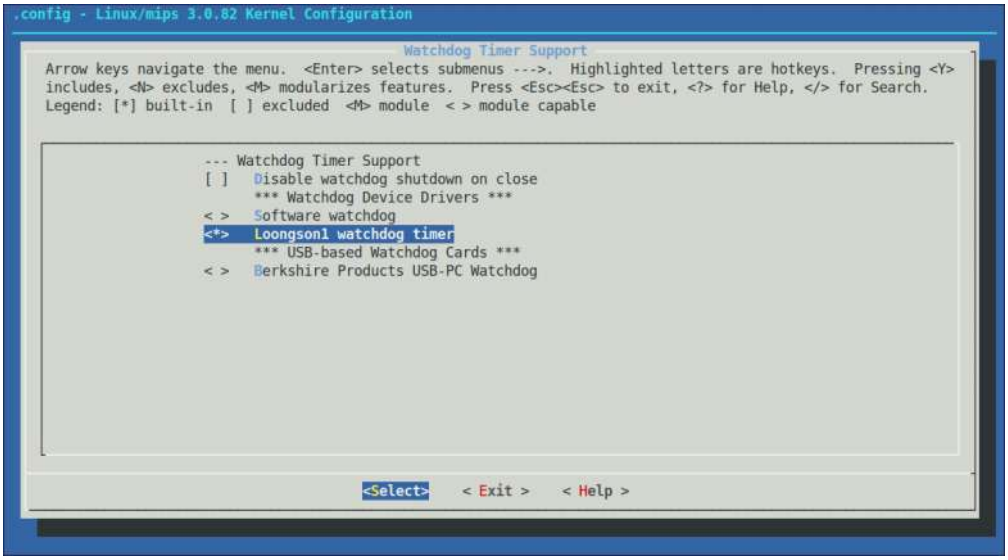
在主菜单界面中，选择“Device Drivers”选项，按回车进入。



选择看门狗时钟支持“Watchdog Timer Support”选项并进入。

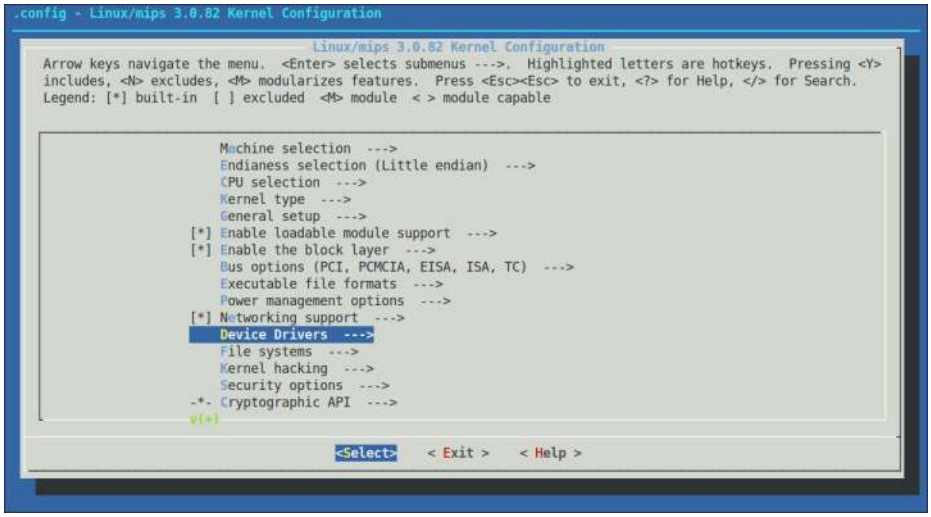


选择“Loongson1 watchdog timer”选项。

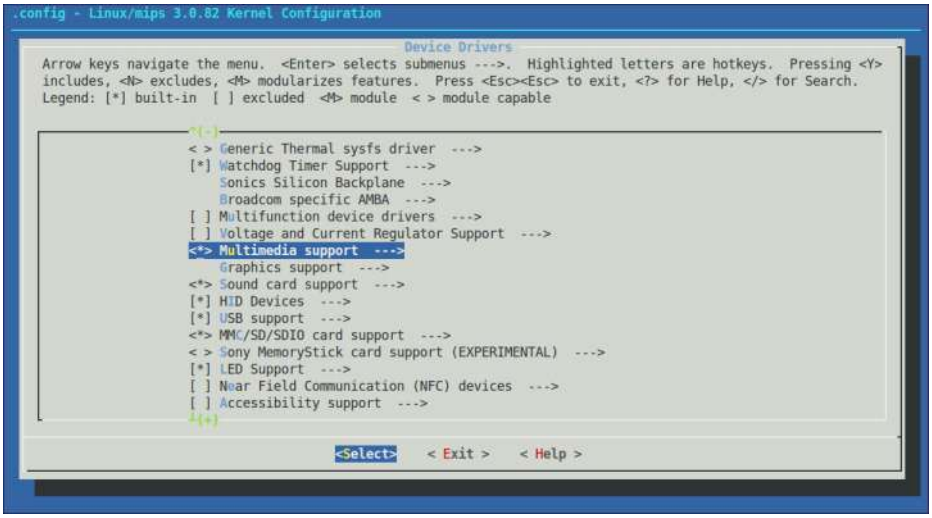


2.3.21 配置中星微 zc301 USB 摄像头驱动

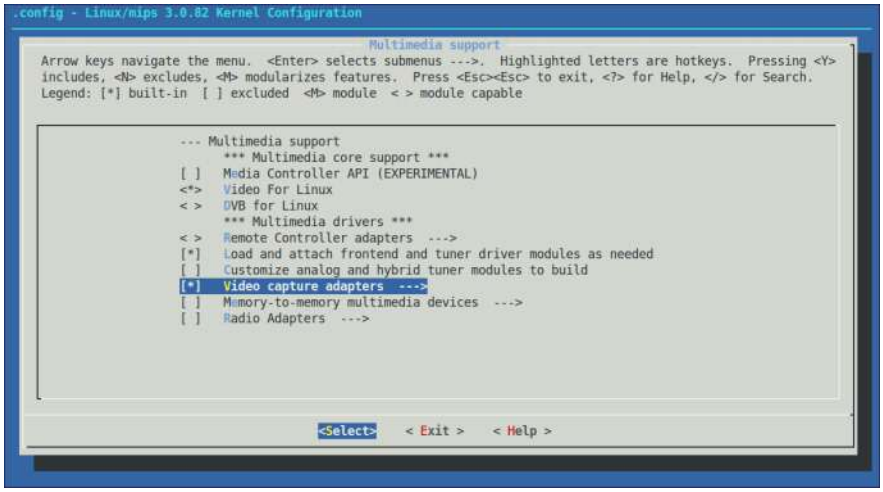
在 Linux 内核配置主界面中选择“Device Drivers ---->”并进入。



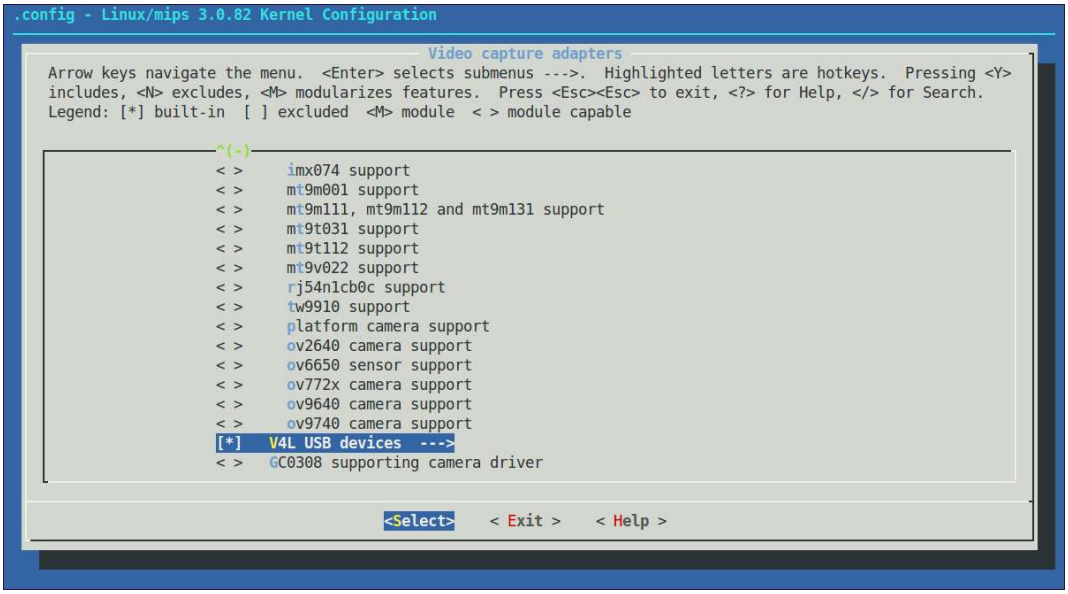
找到“<*> Multimedia support ---->”选项，选择并进入。



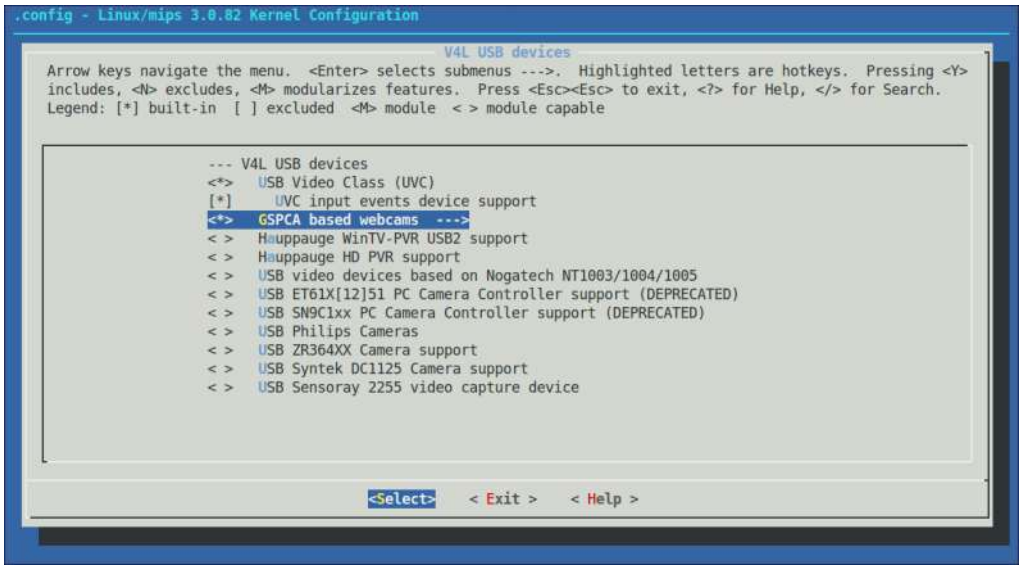
找到“[*] Video capture adapters ---->”选项，选择并进入。



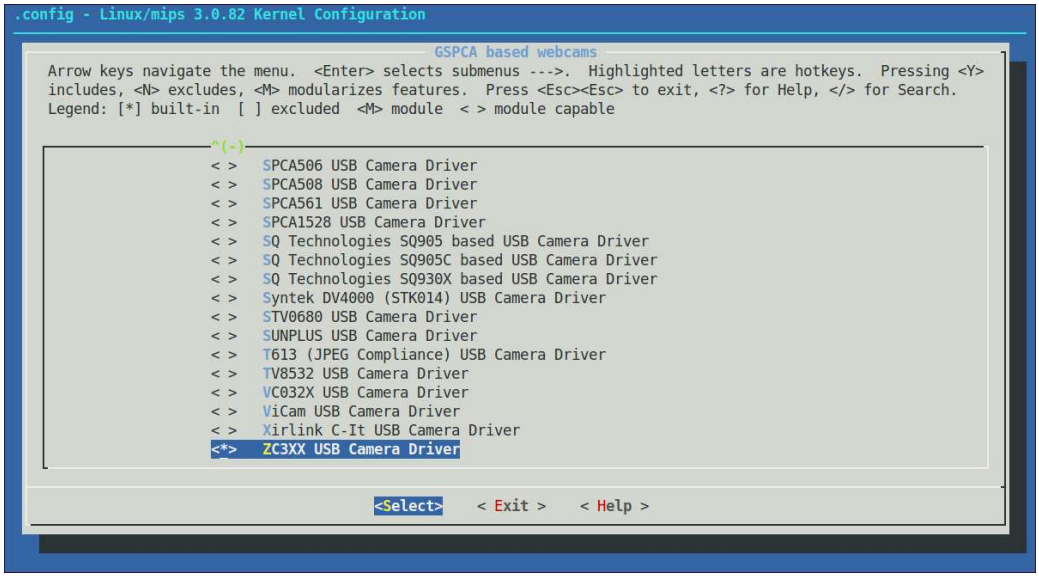
找到“[*]　V4L USB devices　--->”选项，选择并进入。



找到“<*>　GSPCA based webcams　--->”选项，选择并进入。

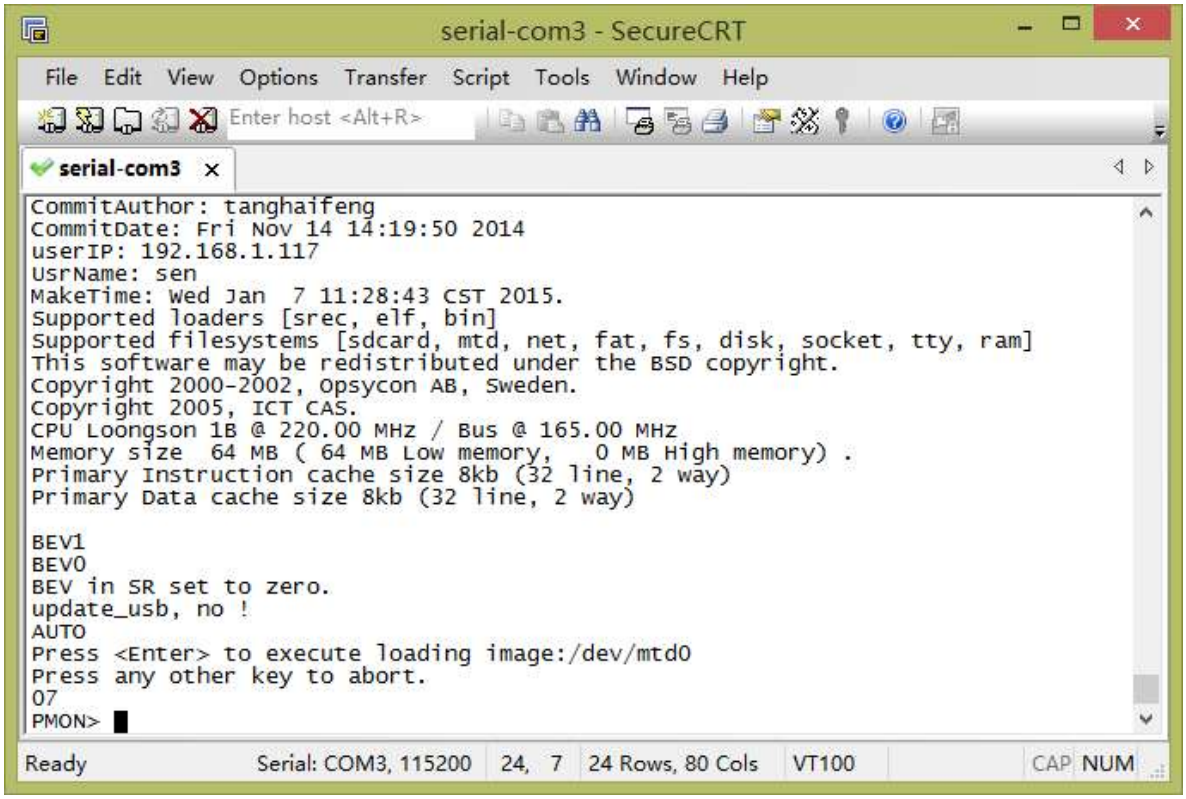


选择 zc301 USB 摄像头驱动的选项“<*>　ZC3XX USB Camera Driver”。



2.4 Linux、PMON、Rootfs 镜像制作

使用串口调试线和网线连接开发板和主机，打开串口终端软件 SecureCRT 并打开相应的串口，开发板上电后在 SecureCRT 窗口中按空格键，即可进入开发板 PMON 命令行模式。

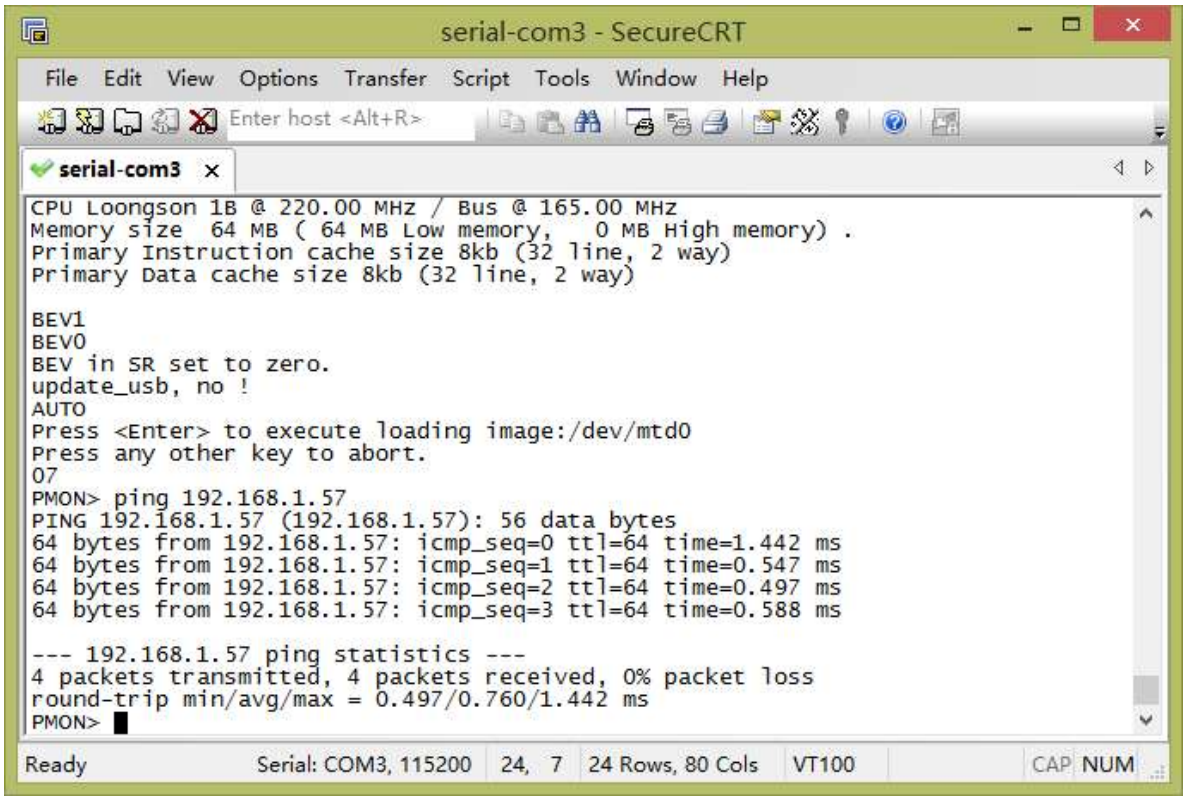


设置 PMON 的 IP 地址（IP 的前三个网段要与主机的一样），并 reboot 重启系统。

PMON> set ifconfig syn0:192.168.1.107

```
PMON> reboot
```

同样再按空格键进入 PMON 命令行模式，在恢复与更新前最好使用 ping 命令测试 Windows 与开发板之间的网络是否连通，在 SecureCRT 中 ping 主机的 IP，若收到回应帧则表示网络通信正常。按“Ctrl+C”即可结束测试。



（2）恢复与更新 PMON、Linux 内核和根文件系统

恢复与更新 PMON 到 SPI Flash：（更新到 Nand Flash 请参考“6.1 烧写 PMON”）

```
PMON> load -r -f bfc00000 tftp://192.168.1.57/gzrom-spi.bin
```

擦除内核分区并恢复与更新 Linux 内核：

```
PMON> mtd_erase /dev/mtd1
PMON> devcp tftp://192.168.1.57/vmlinux /dev/mtd1
```

擦除文件系统分区并恢复与更新根文件系统：

```
PMON> mtd_erase /dev/mtd2
PMON> devcp tftp://192.168.1.57/rootfs-yaffs2.img /dev/mtd2 yaf nw
```

设置 PMON 的启动参数：

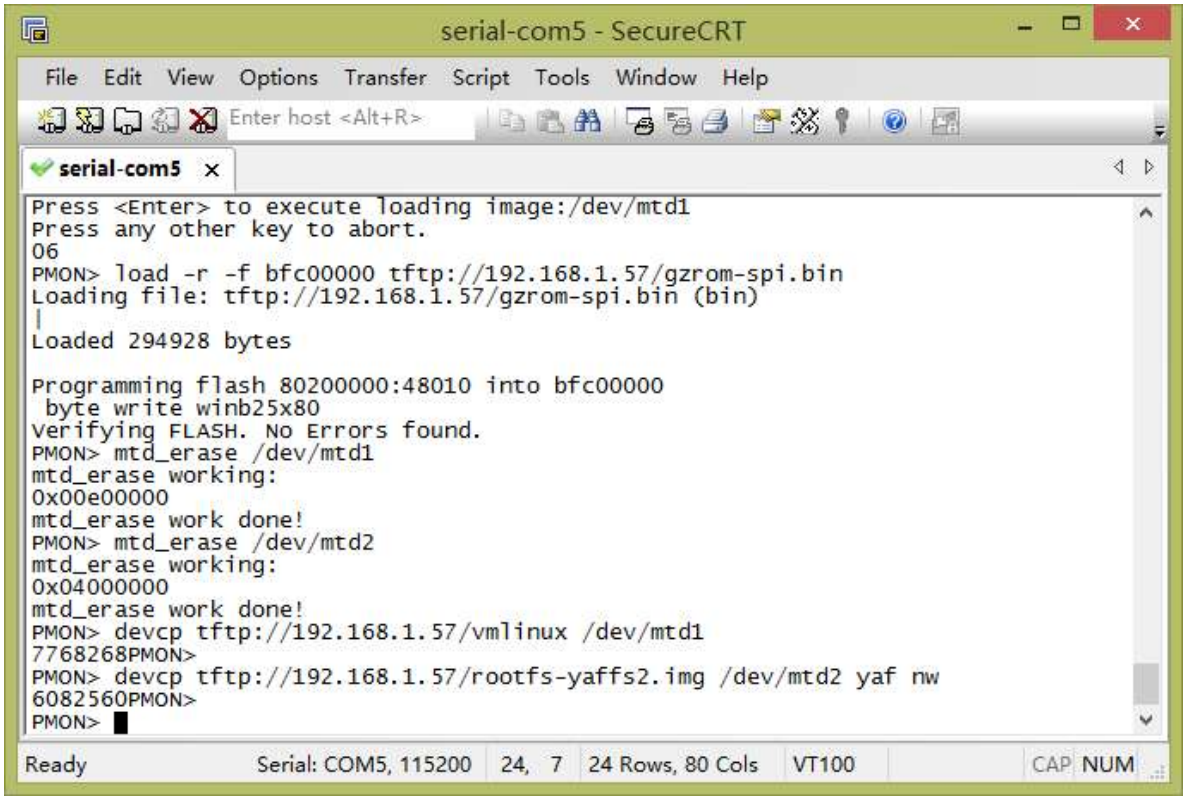
```
PMON> set al /dev/mtd1
PMON> set append "root=/dev/mtdblock2 console=ttyS3,115200 noinitrd init=/linuxrc"
PMON> set append "$append rootfstype=yaffs2 rw video=ls1xfb:800x480-16@60 consoleblank=0"
```

输入“env”命令可查看 PMON 下的各个参数（回车键查看剩余参数，“q”

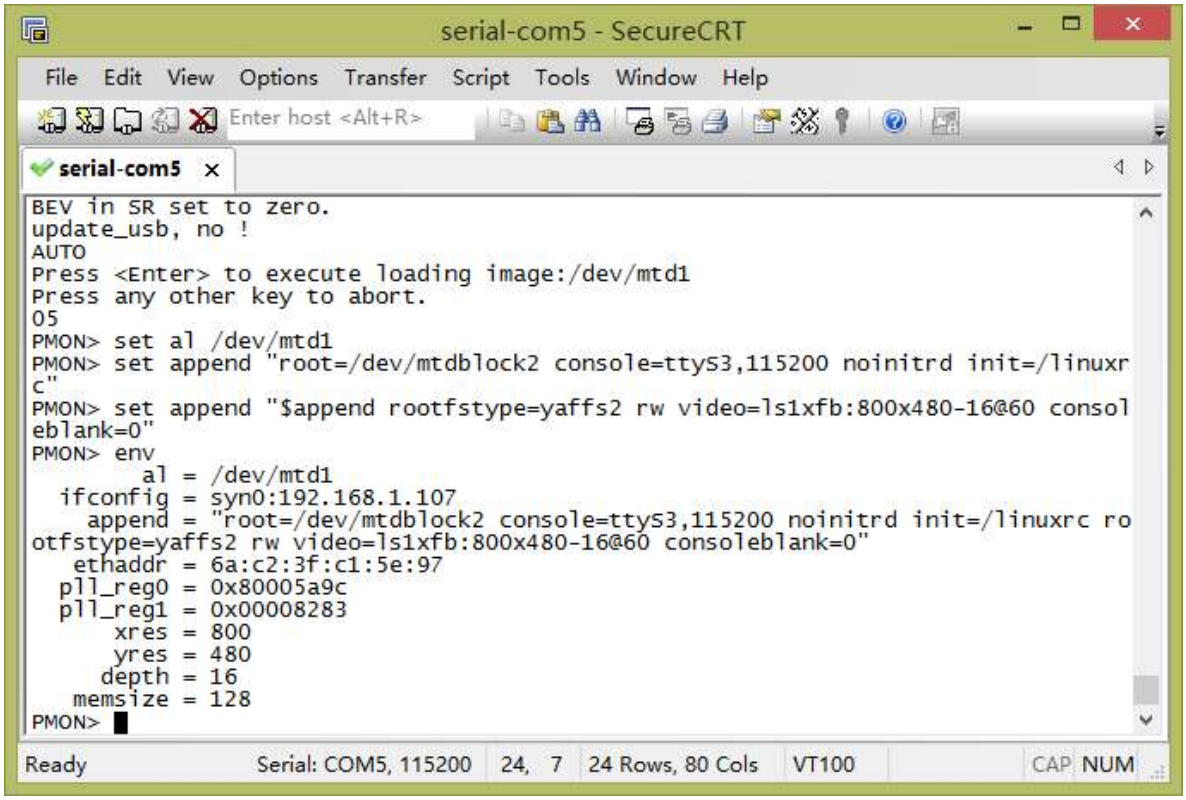
退出查看）。

```
PMON> env
```

恢复与更新 PMON、Linux 内核和根文件系统如下图示：



设置和查看启动参数如下图示：



2.5 Linux 系统的交叉编译环境的搭建

在嵌入式开发过程中，通常由于目标板（开发板）没有足够的资源来运行开发和调试工具，所以需要借助建立了交叉编译环境的宿主机（本文采用主机上的虚拟机+Linux 操作系统方式）通过使用串口、以太网或其他方式来完成开发和调试。

2.5.1 安装 Ubuntu12.04

开发板的嵌入式操作系统为 Linux，主机上也应为 Linux 操作系统。Ubuntu 是开源、免费的 Linux 操作系统，其中 12.04 版本是较新和稳定的长期支持版，操作方便、界面友好。

VMware 虚拟机软件可从官网（http://www.vmware.com/cn/）下载，这里使用 10.0 版本。

VMware Workstation 虚拟机是一个在 Windows 操作系统上运行的应用程序，它可以模拟一个基于 x86 的标准 PC 环境。

安装步骤：

Step1 双击打开安装程序，在“安装类型”界面中选择“典型”安装，如图：

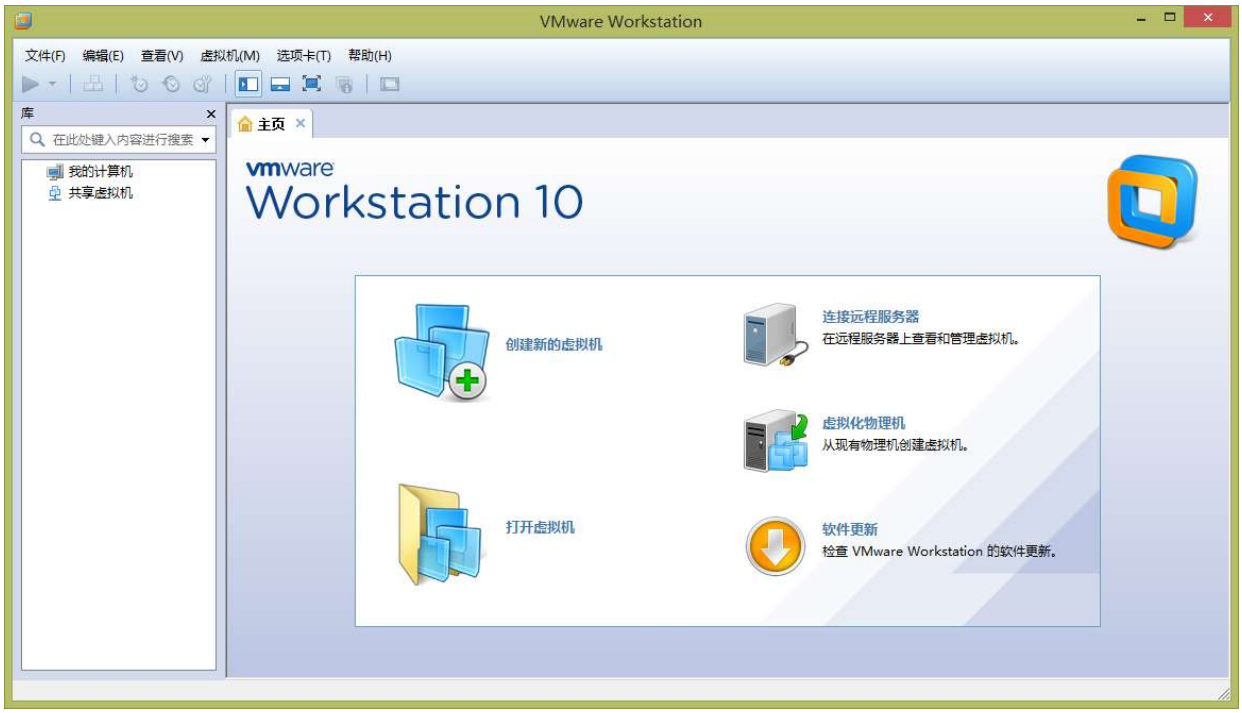
Step2 修改安装路径，选择自己的安装路径，如“D:\Softwares\VMware Workstation”，如图：

Step3 往下的几个步骤选择“下一步”。其中有两步是“启动时检查产品更新”和“帮助改善 VMware Workstation”，可勾选也可不勾选。

Step4 最后进入注册界面，输入注册码后完成安装。

2.5.2 新建 Ubuntu 虚拟机

Step1 运行 VMware Workstation，新建一个虚拟机。在主界面“主页”上选择“创建新的虚拟机”。



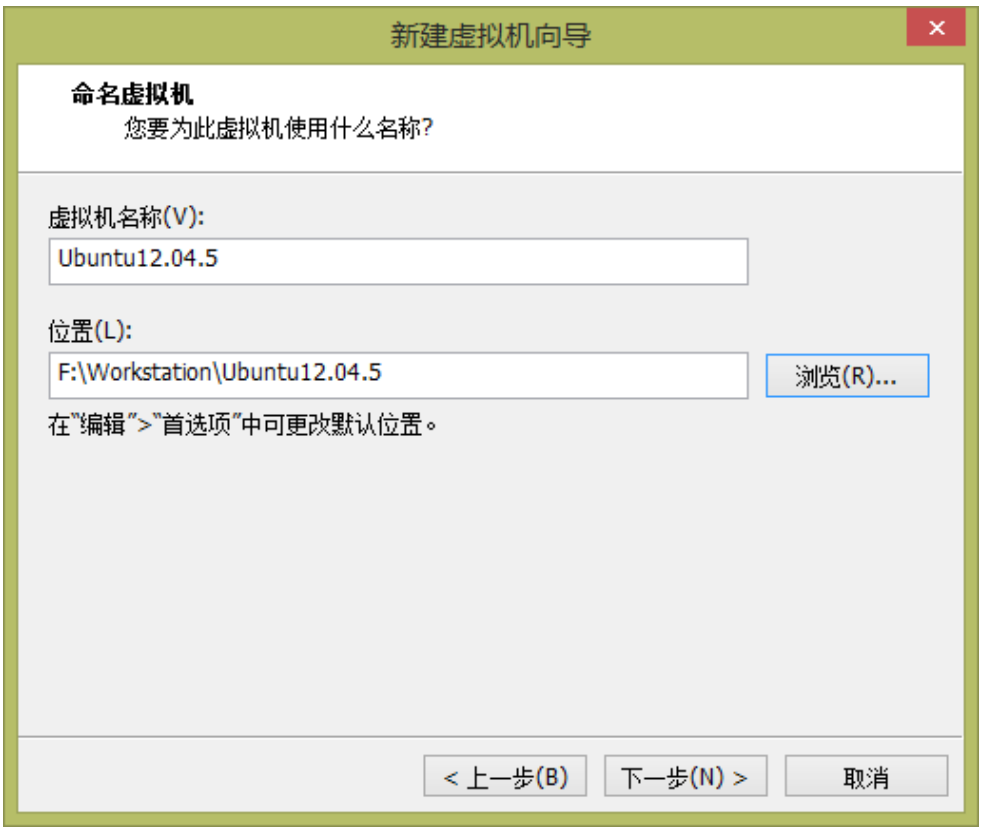
Step2 在弹出框中，选择“自定义（高级）”。

Step3 选择虚拟机的硬件兼容性为 VMware Workstation 10.0。

Step4 选择稍后安装操作系统。

Step5 选择要安装的客户机操作系统 Linux - Ubuntu。

Step6 命名虚拟机名称并选择存放的目录。



Step7 根据主机 CPU 型号来配置虚拟机处理器数量和每个处理器的核心数量。

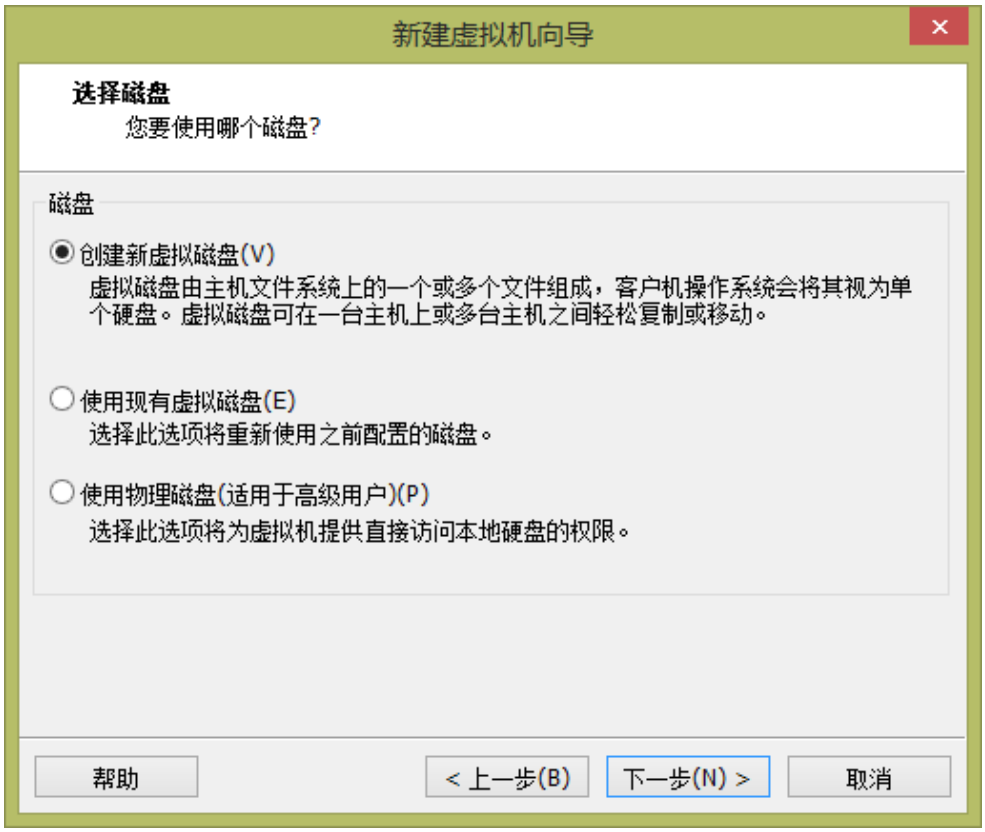
Step8 选择虚拟机的内存值，可根据主机实际内存来选定，这里选择 1GB 内存。

Step9 选择网络类型，默认即可。

Step10 I/O 控制器类型选择默认值。

Step11 磁盘类型选择默认值。

Step12 选择创建新虚拟磁盘。



Step13 指定磁盘容量，用户可根据实际硬盘分区的大小选择虚拟机磁盘的大小，这里分配 30GB 磁盘空间。磁盘空间是否立即分配和磁盘是否拆分多个文件可根据界面中的提示来选择，这里选择立即分配所有磁盘空间和将磁盘存储为单个文件。

Step14 选择一个目录来存放磁盘文件。

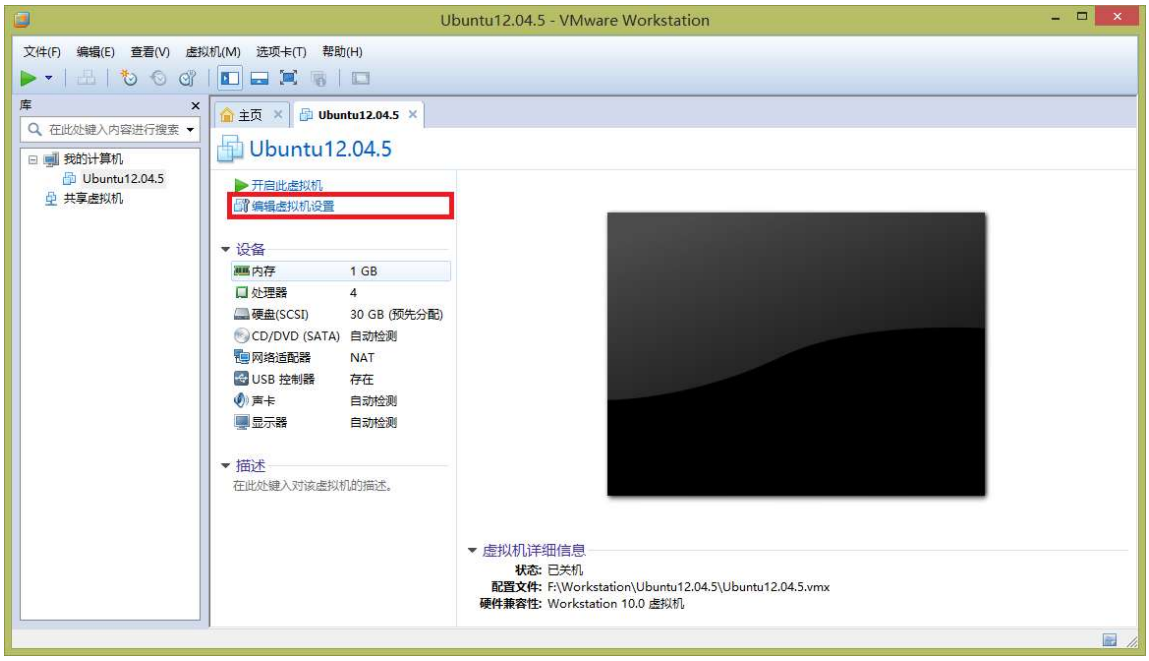
Step15 最后完成虚拟机的创建，点击完成（若在之前的指定磁盘大小选择了立即分配磁盘空间，点击完成后即弹出开始分配磁盘空间的进度条）。

2.5.3 安装 Ubuntu 系统

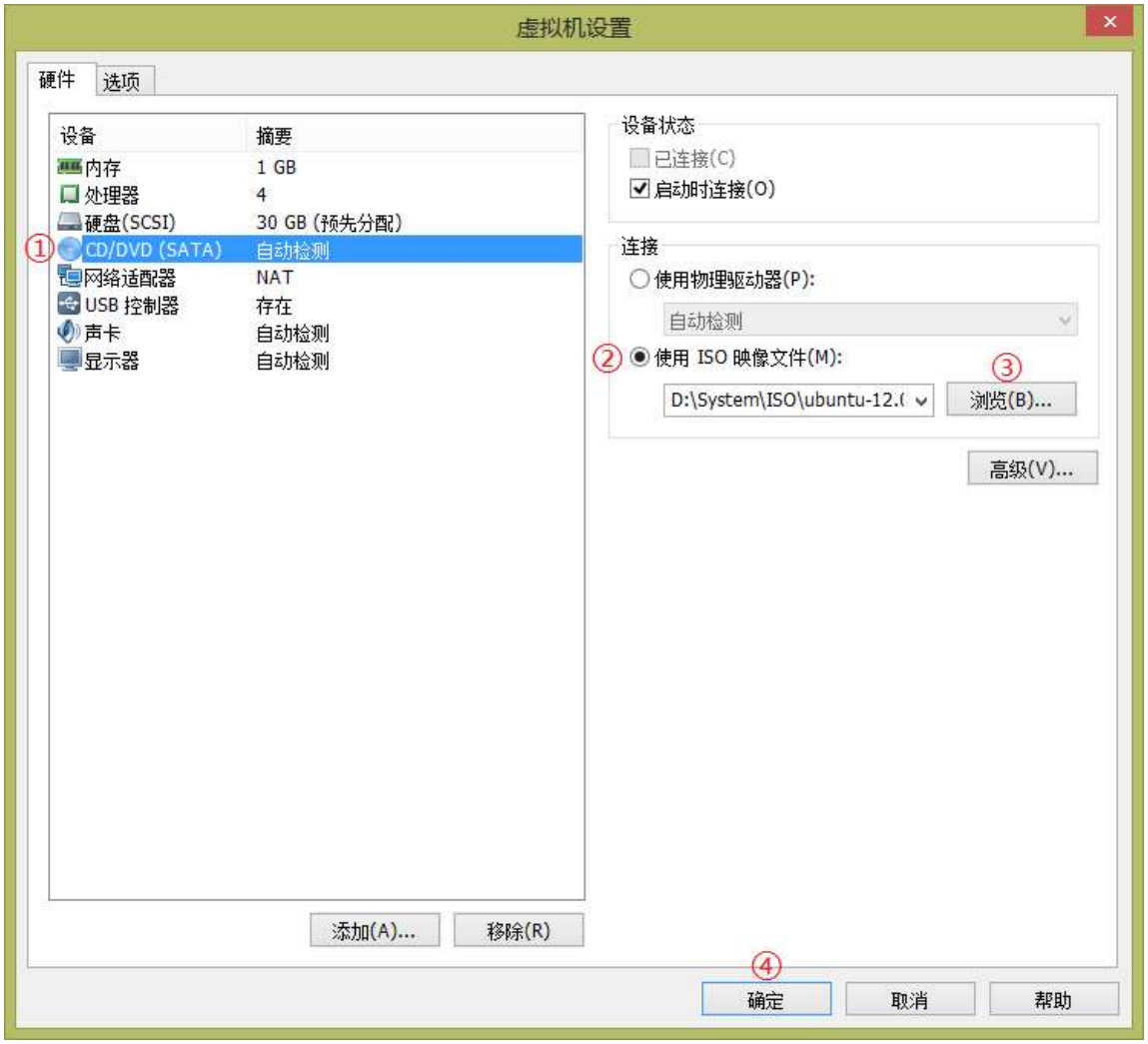
Ubuntu 的 ISO 镜像文件可从官网（ http://cdimage.ubuntu.com/releases/12.04/release/ ）下载，这里下载的是 x86 的版本，文件名为 ubuntu-12.04.5-dvd-i386.iso。

提示：默认情况下，安装 Ubuntu 时必须能上外网，以便在安装过程中下载工具包和语言包。Ubuntu 虚拟机需上外网时网络连接模式为“NAT 模式”，在“4.1.2 新建 Ubuntu 虚拟机”时已选用默认的“NAT 模式”。

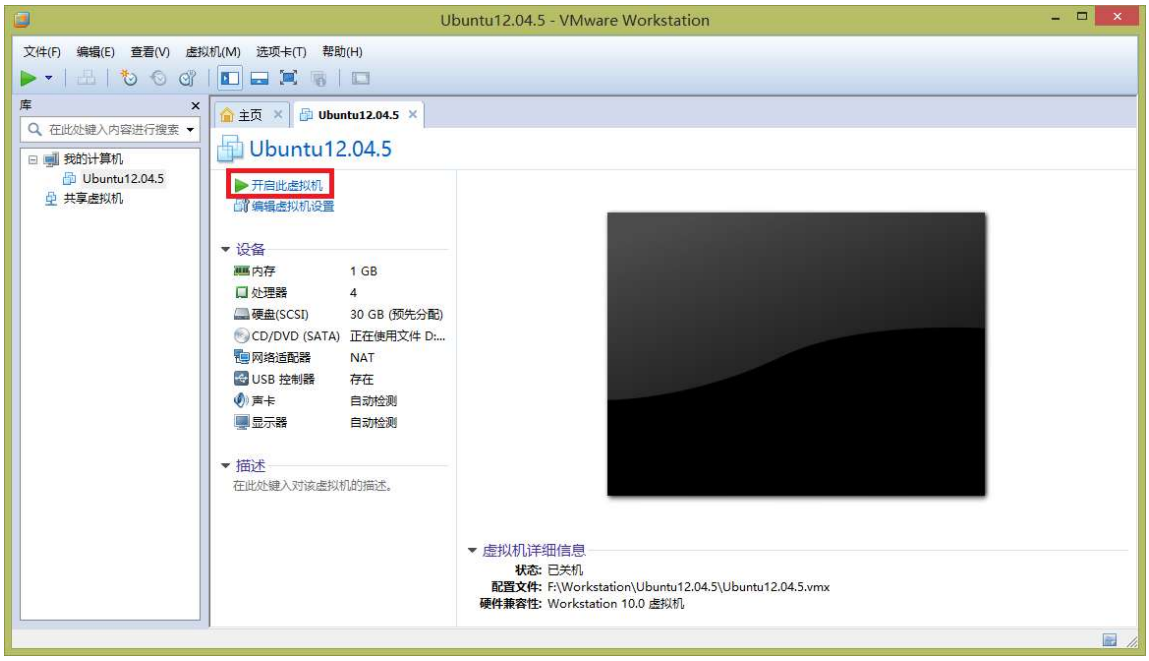
（1）在安装 Ubuntu 系统前先编辑虚拟机设置，选择 Ubuntu 系统映像文件。在 Ubuntu 虚拟机界面中点击“编辑虚拟机设置”。



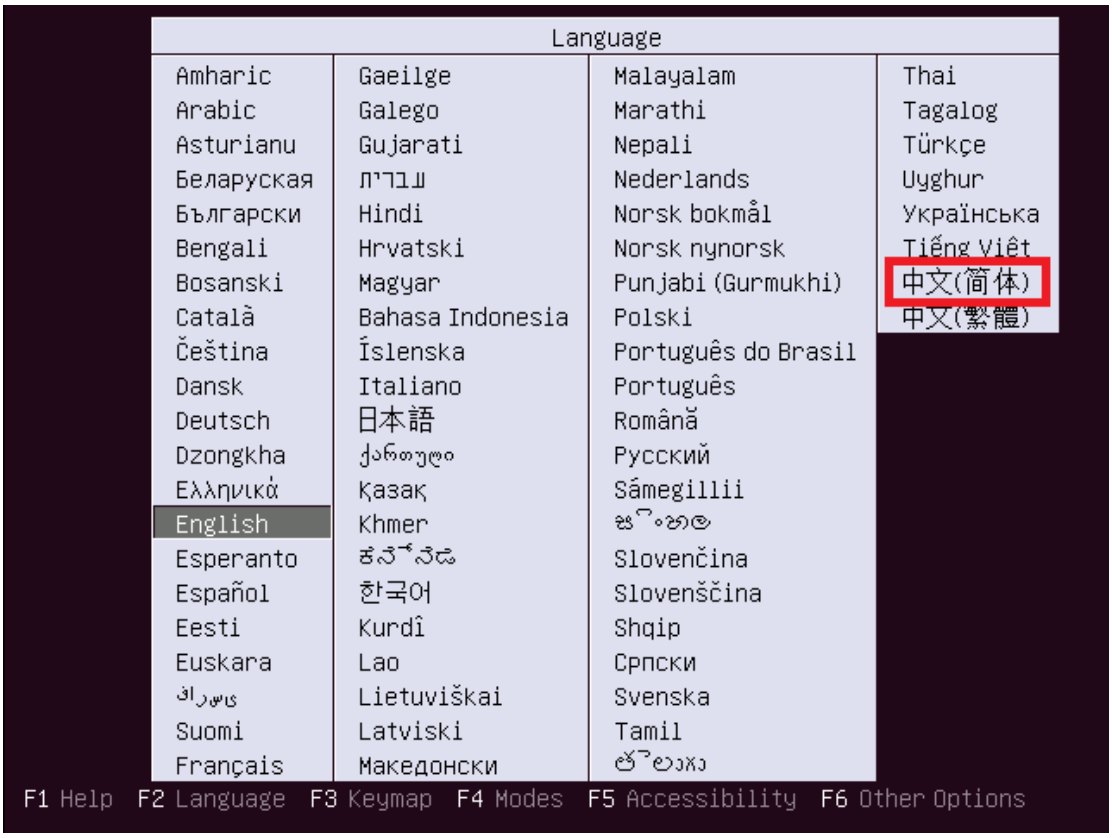
在弹出虚拟机设置对话框中，点击“CD/DVD(SATA)”，选择“使用 ISO 映像文件(M)”，通过浏览选择 Ubuntu 映像文件。



然后点击 Ubuntu 虚拟机界面中的“开启此虚拟机”，开始安装 Ubuntu 系统。

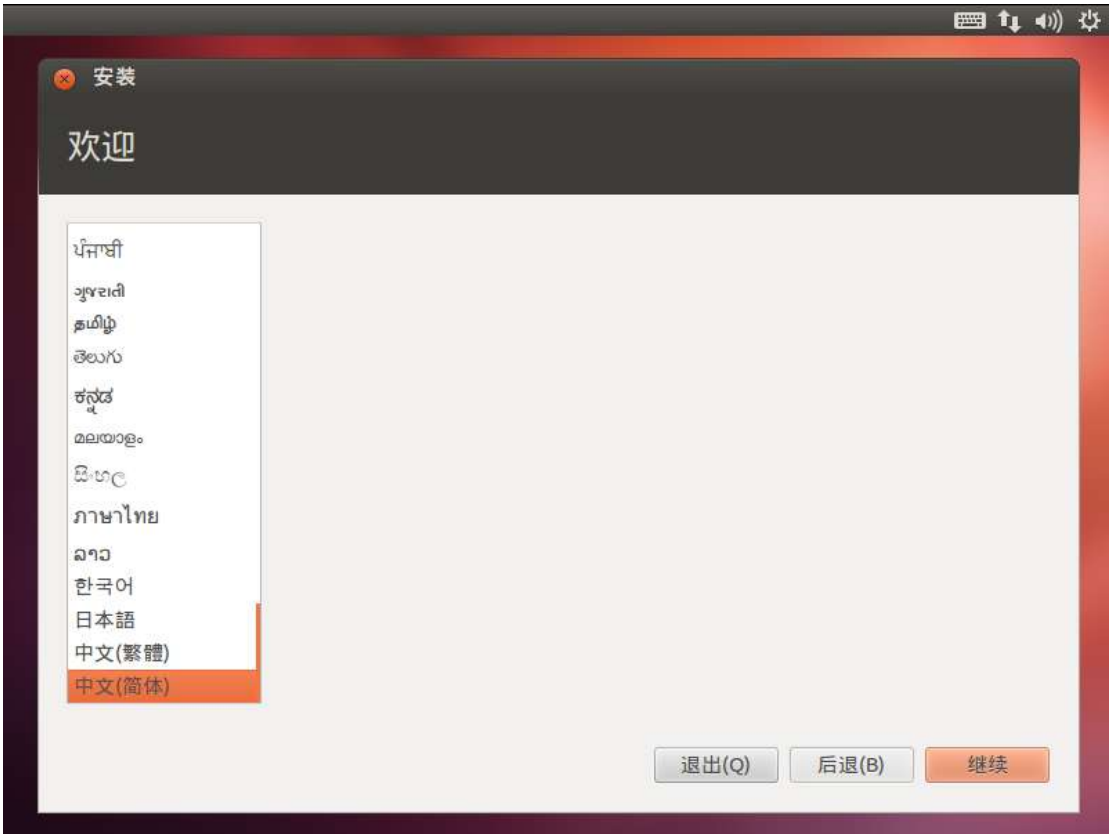


（2）启动虚拟机后选择语言 - 中文（简体）。



（3）选择“安装 Ubuntu(I)”选项。

（4）选择“中文(简体)”，然后点击“继续”。



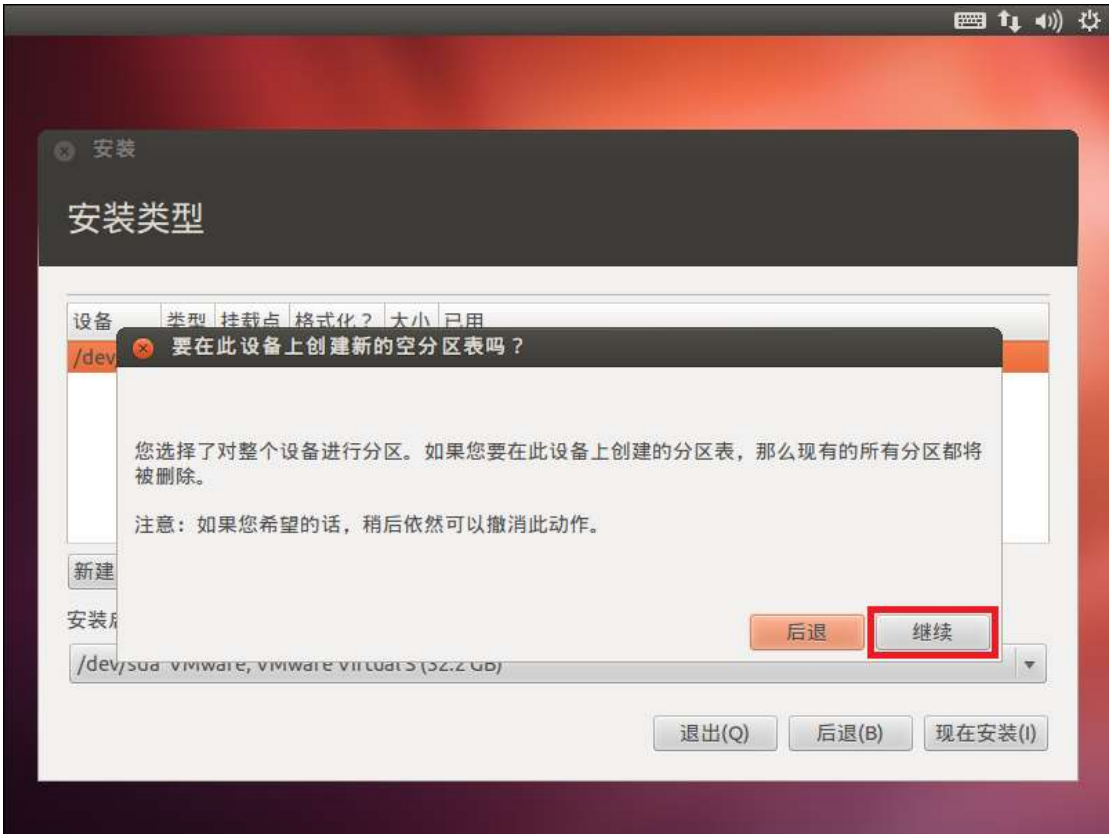
（5）确认网络已连接，系统安装过程中会下载安装语言包及其他安装包。可根

据需要是否要选择安装中下载更新和第三方软件，然后点击“继续”进入下一步。

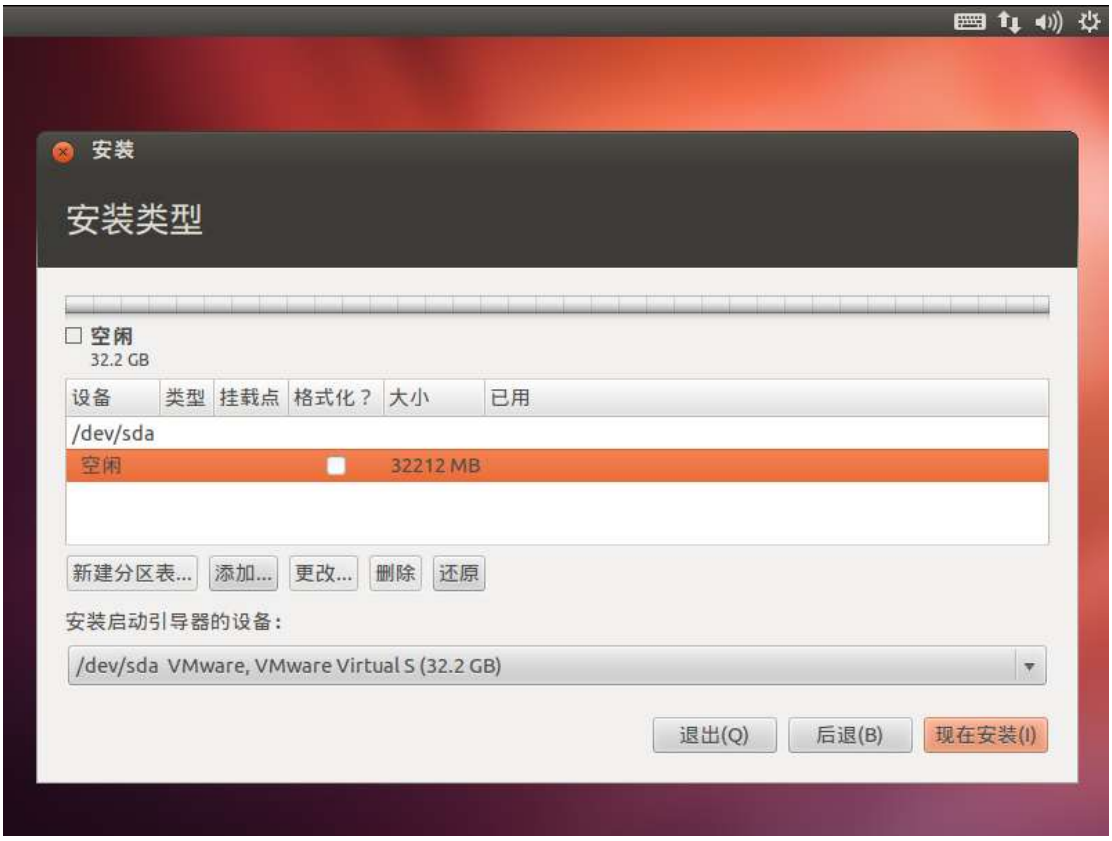
（6）在安装类型中选择“其他选项”，手动分区虚拟机磁盘。

（7）点击“新建分区表”。

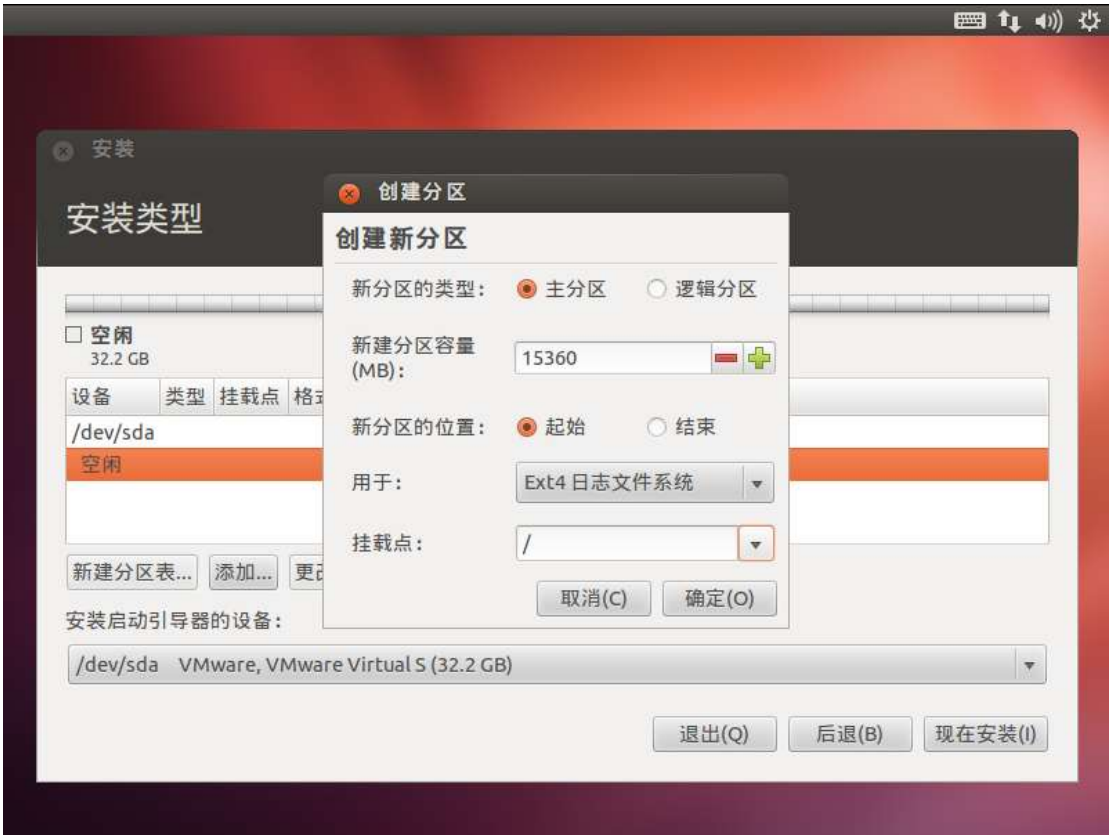
在弹出框中选择“继续”。



（8）选中新出现的“空闲”，点击“添加…”。

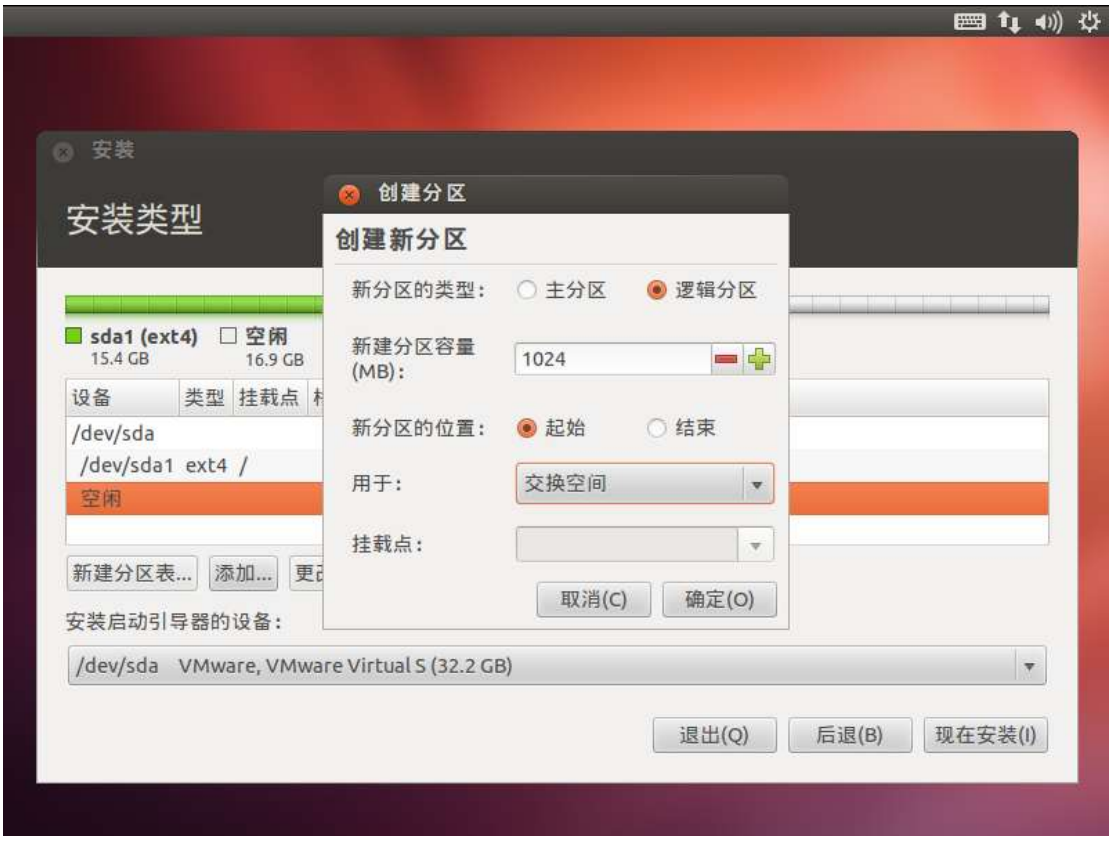


（9）创建根目录“/”分区。分配分区容量，这里分配 15G 空间，挂载点选择“/”（“/”的分区建议大小至少 8G），其它选项均为默认。点击“确定”完成创建。



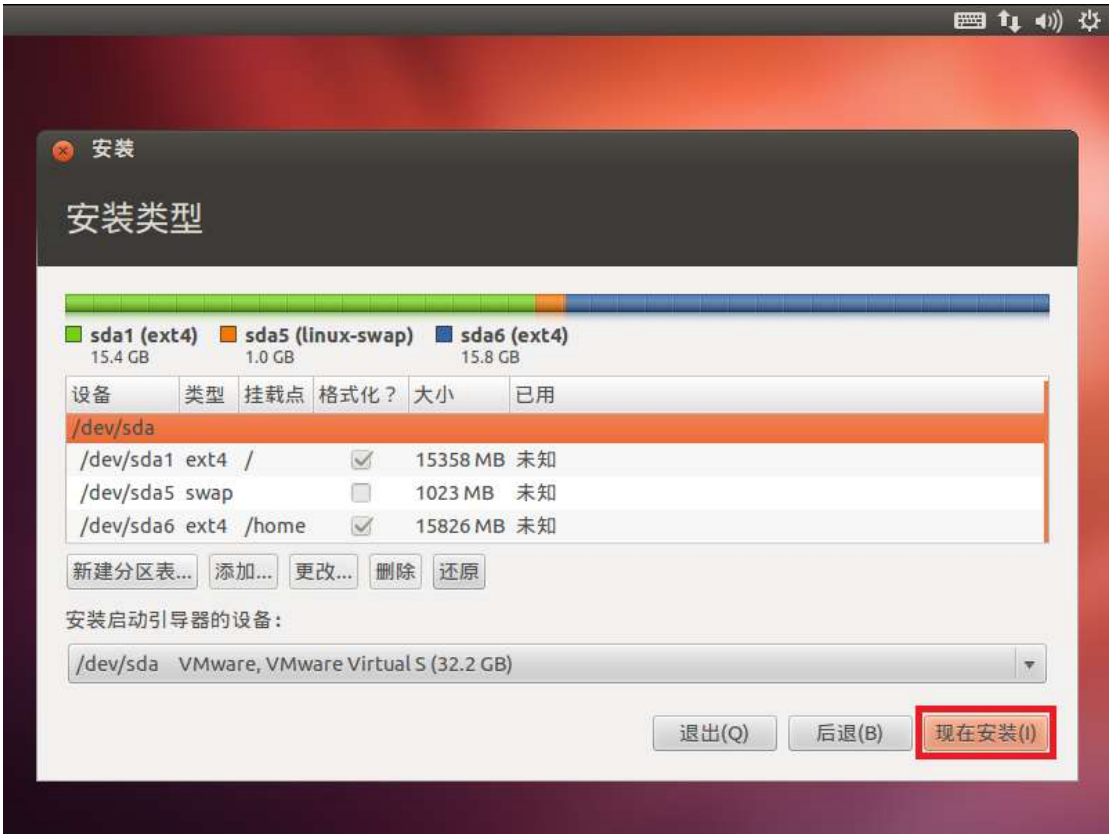
（10）创建交换分区。选中“空闲”并点击“添加...”，分区容量设为 1G，在“用

于：”下拉列表中选择“交换空间”，其它选项均为默认，点击“确定”完成创建。



（11）创建 home 分区。选中“空闲”并点击“添加...”，分区容量为剩余的磁盘容量，挂载点选择“/home”，其它选项均为默认，点击“确定”完成创建。

（12）分区创建完成后，点击“现在安装”开始安装系统。



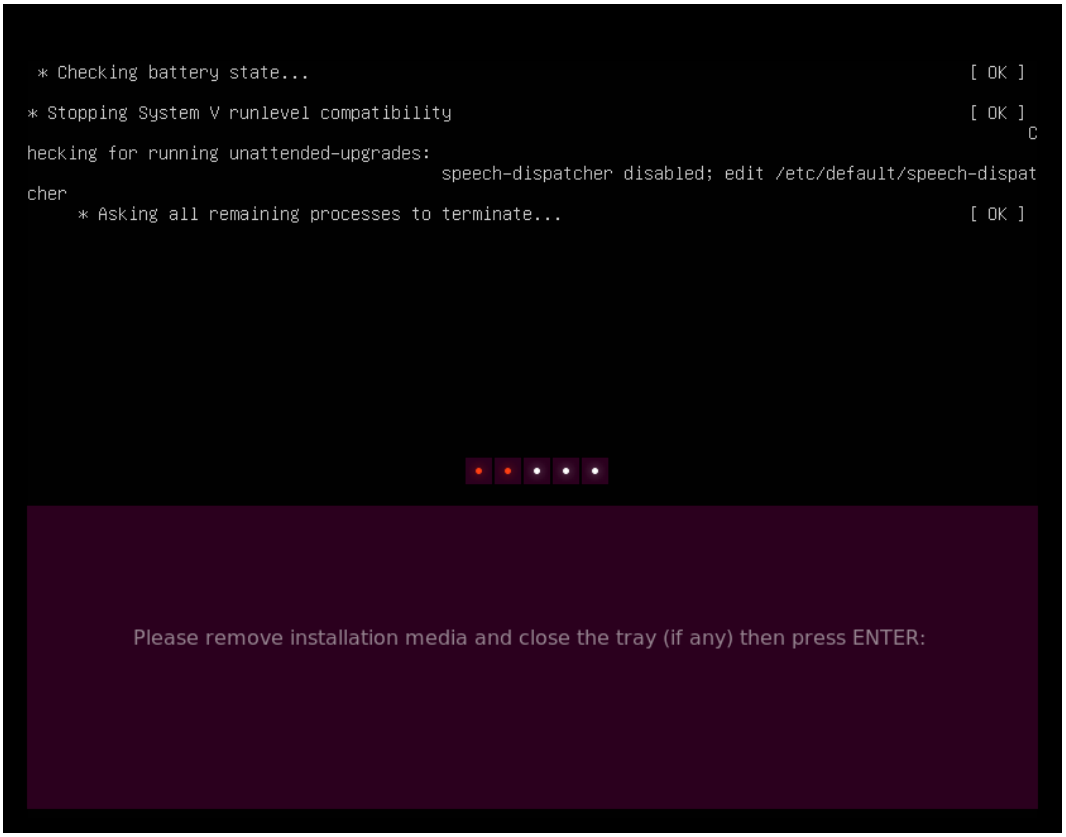
（13）选择所在的时区。

（14）选择键盘布局，布局为英语(美国)。

（15）新建用户，填写用户名、计算机名和密码等，之后系统开始安装。

（16）等待系统安装完成之后，在提示框下点击“现在重启”，

（17）当出现以下界面时，按“回车”即可重启系统并进入 Ubuntu 系统登录界面。至此，Ubuntu 系统安装完成。



2.5.4 备份恢复 Ubuntu 虚拟机

在使用 Ubuntu 系统过程中，可能会因为操作失误或其它原因导致 Ubuntu 系统出现问题，此时如果有其备份文件，将会很方便地恢复到备份时的状态，安全地保护了重要的文件资源。备份 Ubuntu 虚拟机有两种方式：第一种是简单的压缩并拷贝备份虚拟机目录；第二种是使用 VMware Workstation 软件的快照备份功能。

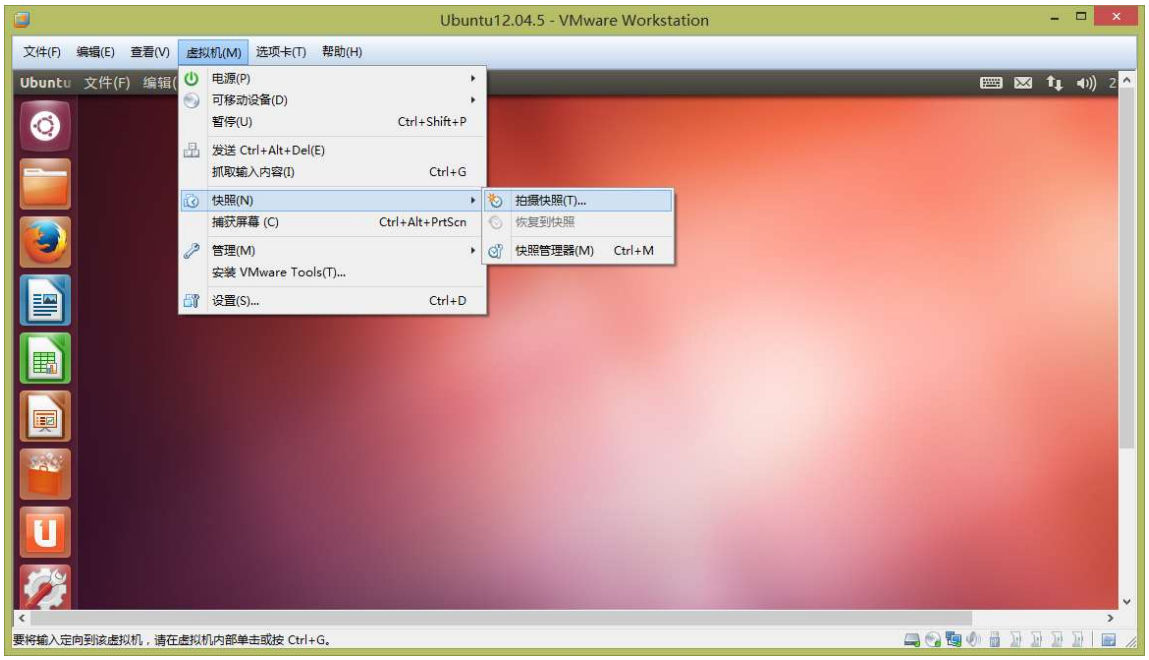
（1）拷贝式备份恢复

若 Ubuntu 虚拟机所在文件夹为“F:\Workstation\Ubuntu12.04.5”，使用压缩软件打包该文件夹，将生成的压缩包拷贝到移动硬盘或其它磁盘备份，在需要

恢复时删除原有虚拟机再解压该压缩包即可，也可拷贝到其它的 PC 机，避免繁琐的重复地在虚拟机上安装 Ubuntu 系统。

（2）快照式备份

拍摄 Ubuntu 虚拟机快照。点击 VMware Workstation 软件中菜单栏的“虚拟机(M)”，选择下拉菜单的“快照(N)”，再点击其右侧下拉菜单的“拍摄快照(T)…”。



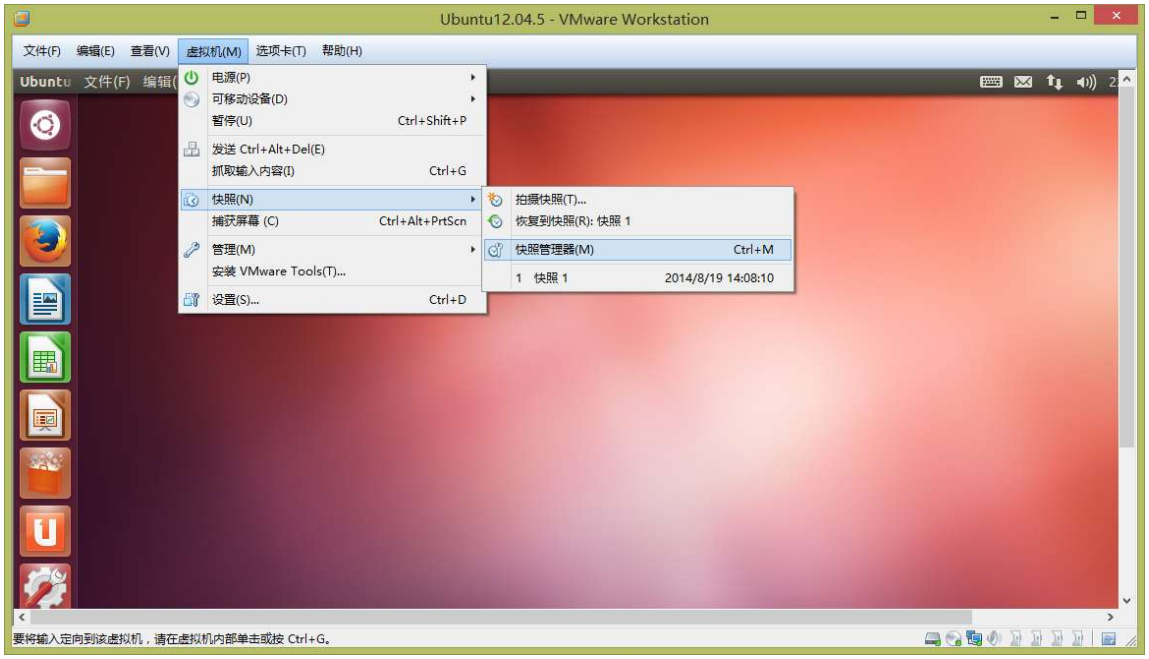
在弹出的拍摄快照对话框中，在“名称”一栏中可自定义快照的名称，在“描述”一栏中可输入该快照的描述信息点击“拍摄快照”完成快照的创建，这样就备份好当前状态下的 Ubuntu 虚拟机。

快照式备份可以备份多个不同时间点的虚拟机快照，恢复时可以选择某个时间点的快照来恢复。

（3）快照式恢复

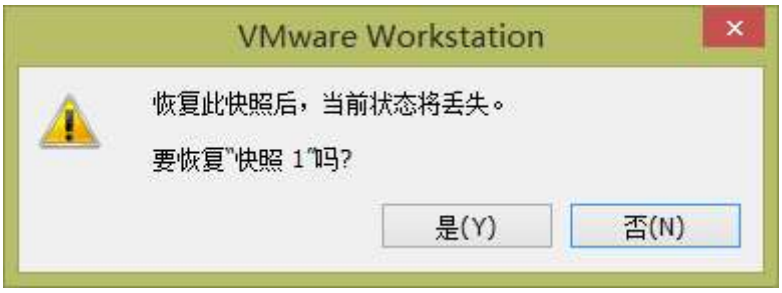
点击 VMware Workstation 软件中菜单栏的“虚拟机(M)”，选择下拉菜单的

“快照(N)”，再点击其右侧下拉菜单的“快照管理器(M)…”。



在弹出的快照管理器对话框中选择需要恢复的快照（如快照 1），点击“转到(G)”。

在弹出框中点击“是(Y)”。



最后 Ubuntu 虚拟机将会恢复到该快照备份时的虚拟机状态。

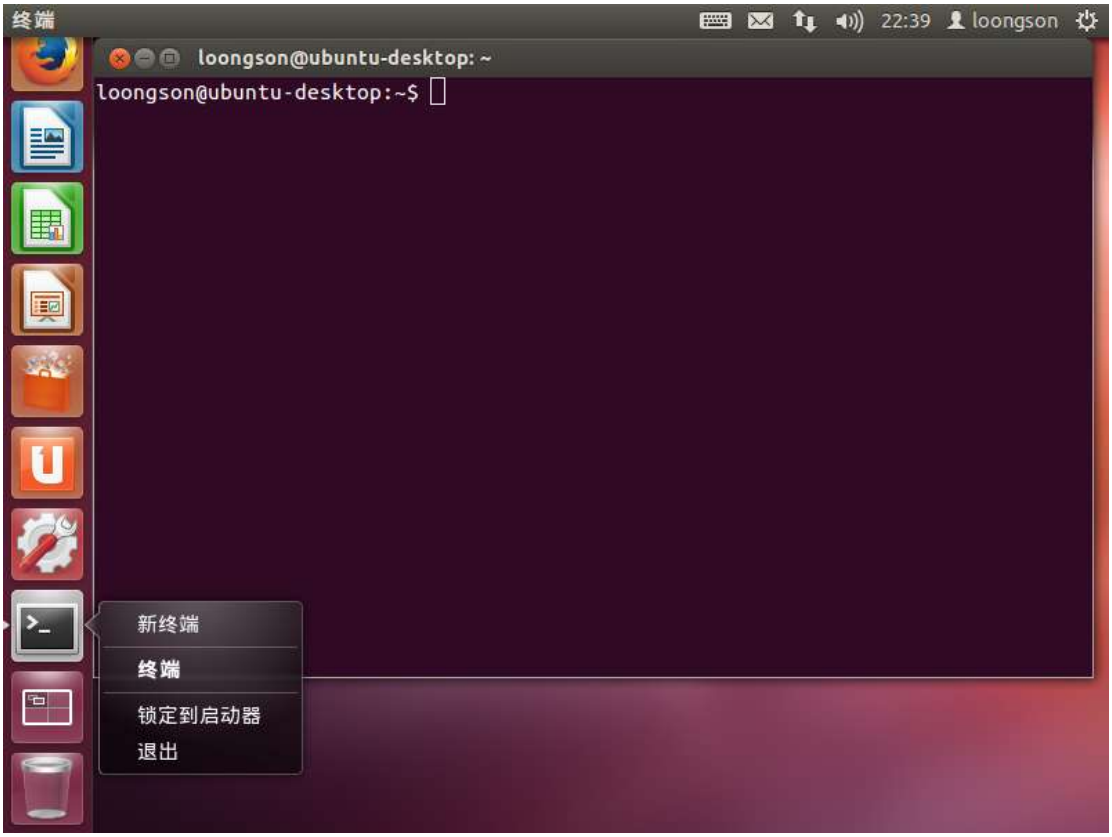
提示：将 Ubuntu 虚拟机恢复到某个时间点的快照之前，要先备份好那个时间点之后在 Ubuntu 系统中改动过的数据资料。否则，Ubuntu 系统下所有目录都会恢复成当初的状态。

2.6 使用 Ubuntu12.04

2.6.1 Ubuntu 终端

Ubuntu 中的所有管理任务都可以在终端中完成。Ubuntu 终端使用命令行模式，许多情况下，使用终端比使用图形化的程序更快捷，而且还可能实现额外的功能。

启动终端。进入系统按 Ctrl+Alt+T 快捷键启动终端，然后右键点击终端图标将其锁定到启动器，方便快速打开终端。



在 Ubuntu 下的绝大多数操作都是通过命令来完成的，因此需要熟悉常用到的命令。

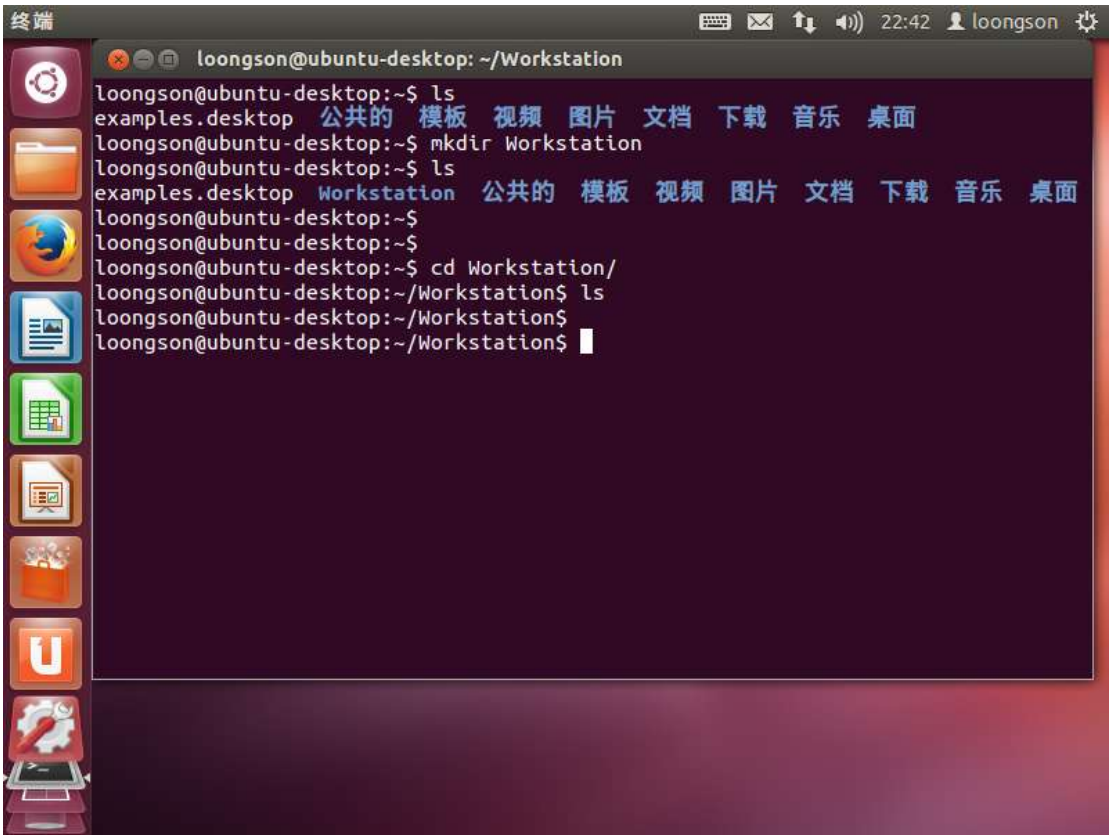
终端中的字符串“loongson@ubuntu-desktop:~$”是命令行提示符。其中：

1）loongson 是当前登录的用户名。

2）ubuntu-desktop 是计算机名。

3）~表示当前用户的主目录（即/home/loongson）。

4）$表示当前用户为普通用户（若是#，表示当前用户为超级用户，用户名为 root）。



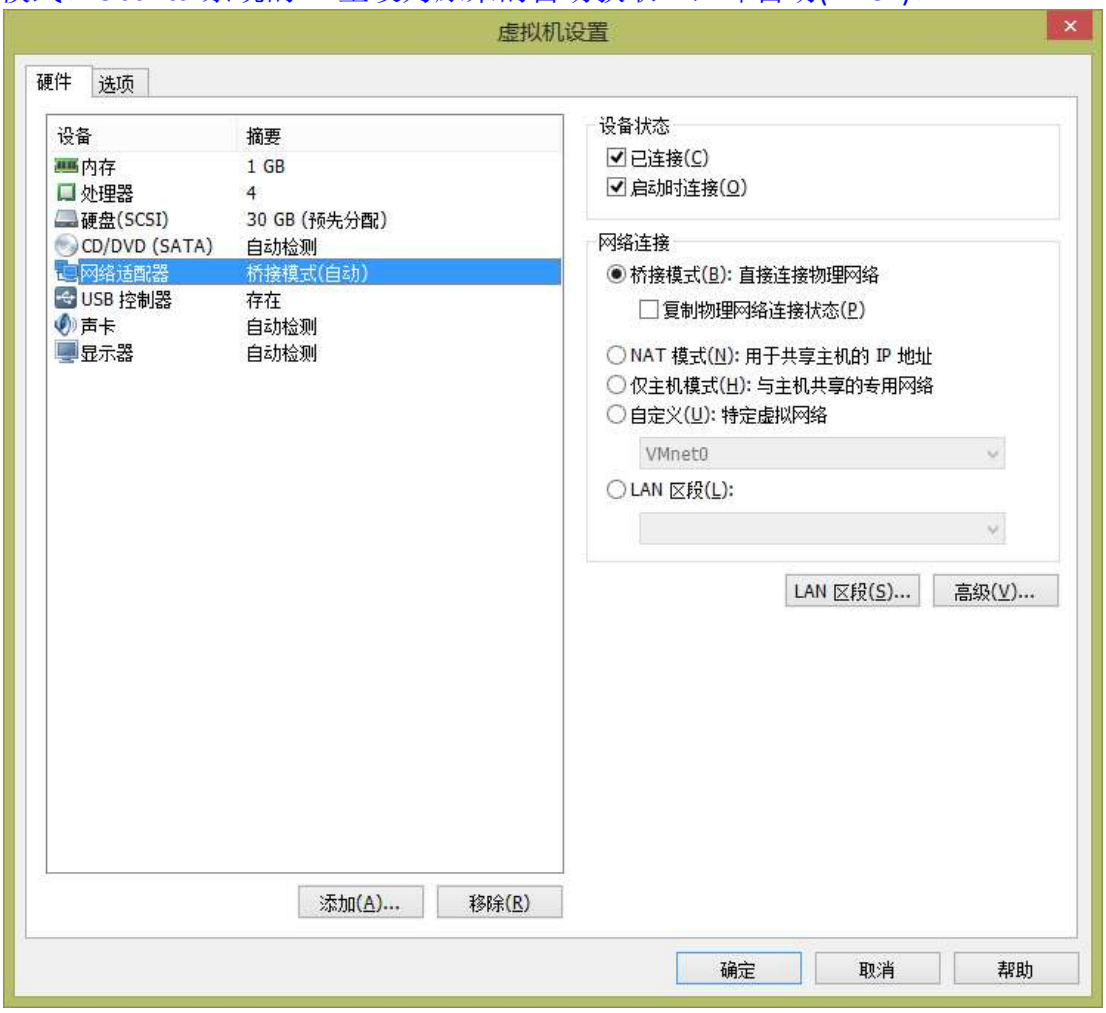
2.6.2 设置 Ubuntu 虚拟机网络

Ubuntu 虚拟机要与开发板进行网络通信，或与 Windows 主机共享文件时，需要将网络连接模式设置为“桥接模式”，同时要在 Ubuntu 系统中手动设置个固定 IP。

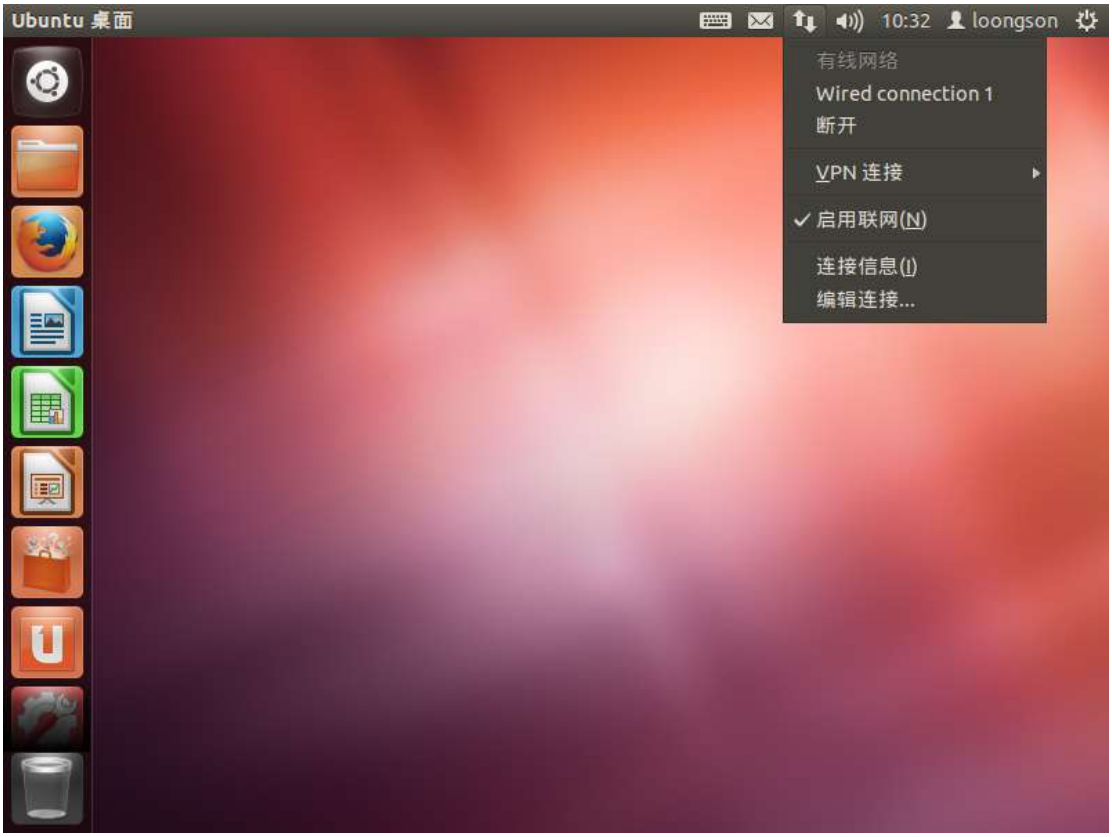
在 VMware Workstation 的菜单栏中，选择“虚拟机”→“设置”，“网络适

配器”→“网络连接”，确认网络连接模式为“桥接模式”。

提示：若在桥接模式下无法在线安装软件包，可将网络连接模式重设为 NAT 模式。Ubuntu 系统的 IP 重设为原来的自动获取 IP，即自动(DHCP)。



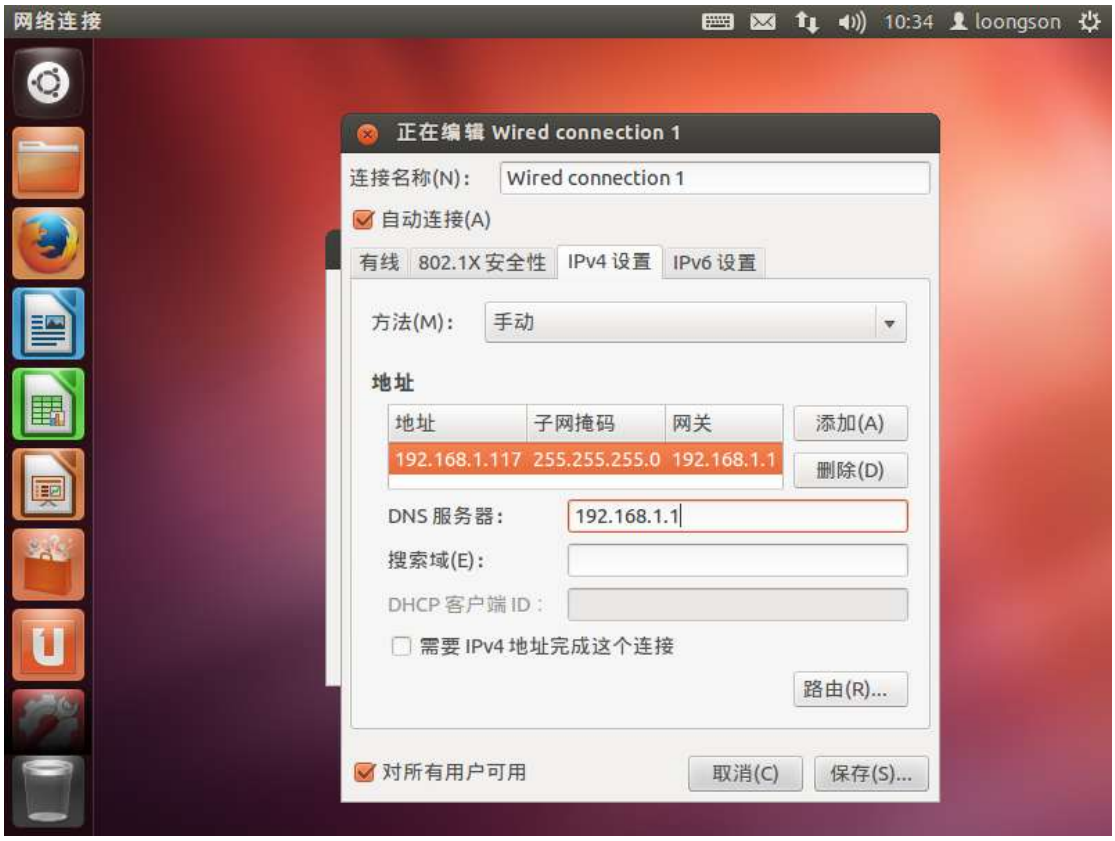
设置 Ubuntu 系统的 IP 地址，点击右上角的网络图标，断开现有网络，再点击编辑连接。



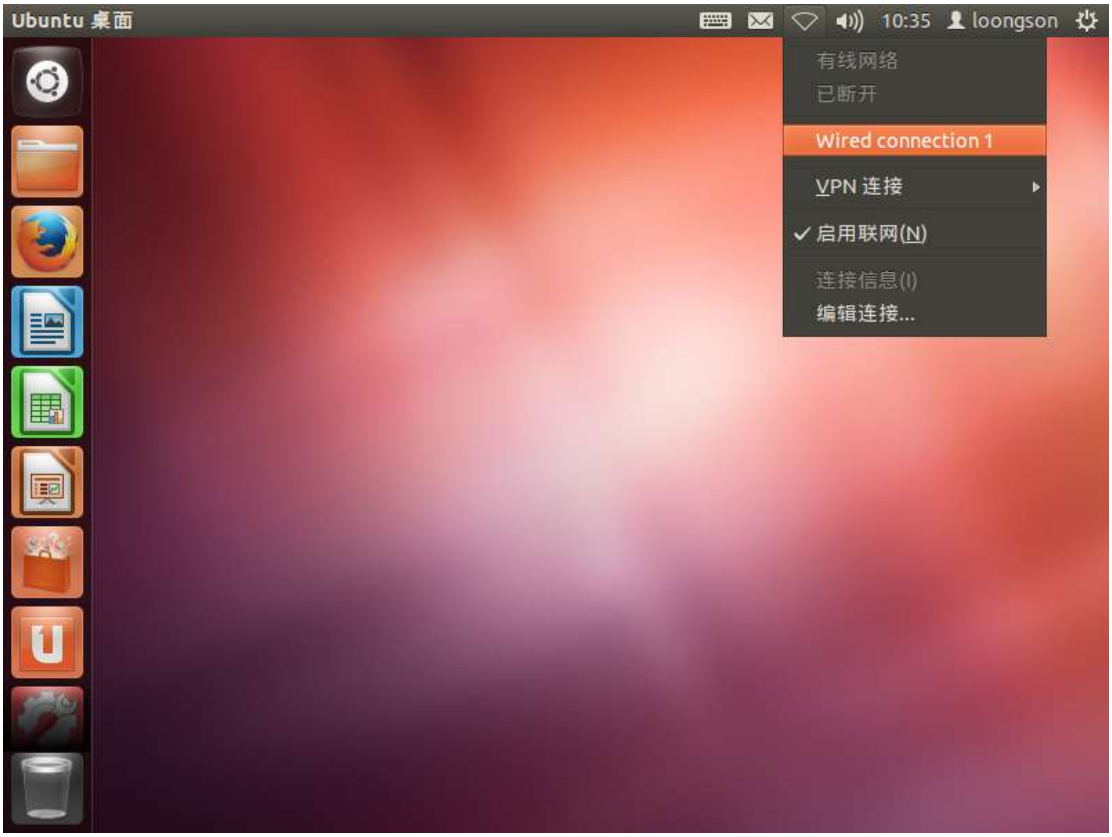
弹出网络连接对话框后，点击选中“Wired connection 1”，再点击右边的编辑按钮。

弹出编辑对话框后，选择“IPv4 设置”选项卡，将“方法(M)”改为手动。

然后点击“添加”按钮，依次添加地址、子网掩码和网关，再添加 DNS 服务器。保存后即可关闭对话框。（提示：在 IPv4 设置中 IP 的前三个网段要与 Windows 主机的一样）



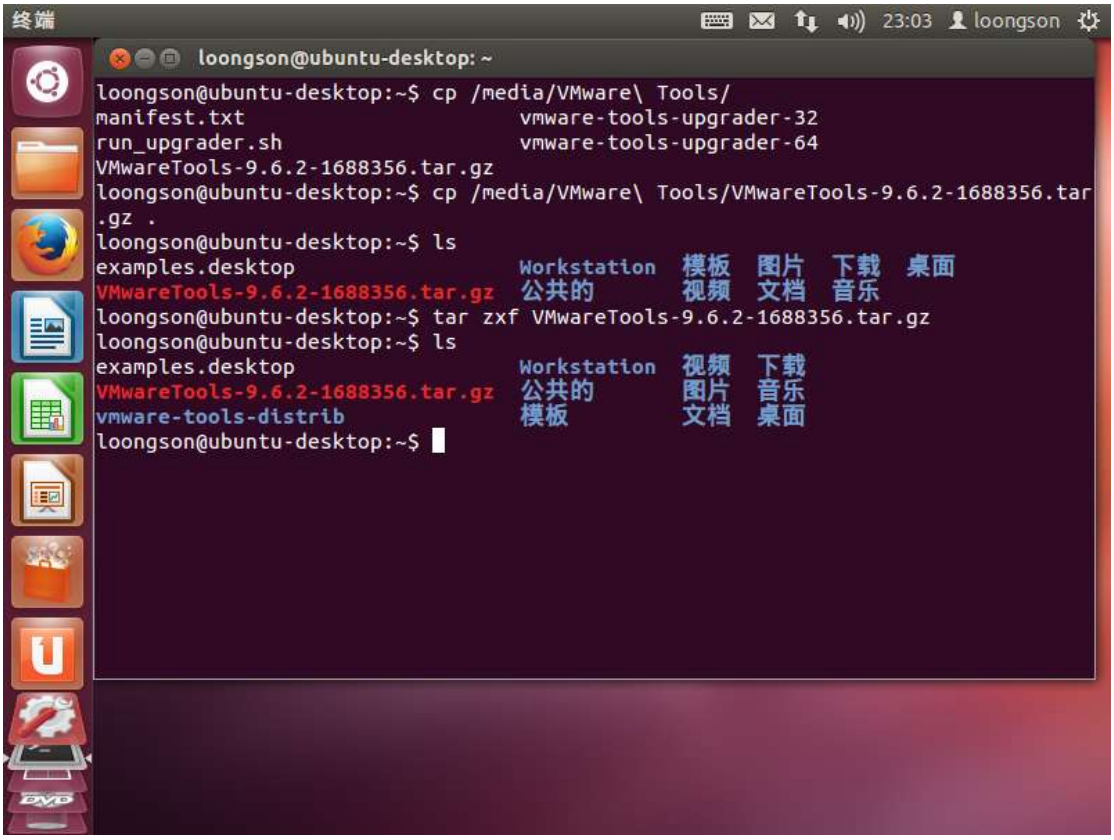
然后点击右上角的网络图标，点击“Wired connection 1”连接网络。



2.6.3 安装 VMware Tools

在 VMware Workstation 菜单栏中选择“虚拟机”→“安装 VMware Tools”。

打开终端，拷贝 VMware Tools 安装包到当前目录并解压。



拷贝工具源码包并解压。

```
$ cp /media/VMware\ Tools/VMwareTools-9.6.2-1688356.tar.gz .
$ tar zxvf VMwareTools-9.6.2-1688356.tar.gz
```

进入解压后的目录，并运行安装程序。

```
$ cd vmware-tools-distrib
$ sudo ./vmware-install.pl
```

在安装过程中不需输入任何信息，中间会出现路径选项以及 Yes 和 No 选项的提示，这里只需按回车键（使用默认值），直到安装结束。

```
Creating a new VMware Tools installer database using the tar4 format.

Installing VMware Tools.

In which directory do you want to install the binary files?
[/usr/bin]

What is the directory that contains the init directories (rc0.d/ to rc6.d/)?
[/etc]

What is the directory that contains the init scripts?
[/etc/init.d]

In which directory do you want to install the daemon files?
[/usr/sbin]

In which directory do you want to install the library files?
[/usr/lib/vmware-tools]

The path "/usr/lib/vmware-tools" does not exist currently. This program is
going to create it, including needed parent directories. Is this what you want?
[yes]

In which directory do you want to install the documentation files?
[/usr/share/doc/vmware-tools]

The path "/usr/share/doc/vmware-tools" does not exist currently. This program
is going to create it, including needed parent directories. Is this what you
want? [yes]
```

安装完成之后，会出现“Enjoy”的提示。然后重启 Ubuntu 系统。

```
To enable advanced X features (e.g., guest resolution fit, drag and drop, and
file and text copy/paste), you will need to do one (or more) of the following:
1. Manually start /usr/bin/vmware-user
2. Log out and log back into your desktop session; and,
3. Restart your X session.

Enjoy,

--the VMware team
```

2.6.4 更新 Ubuntu 软件包列表

刚安装 Ubuntu 系统后，在没有更新软件包列表的情况下，使用 apt-get install 命令在线安装软件包时，会安装失败并提示没有发现该软件包。

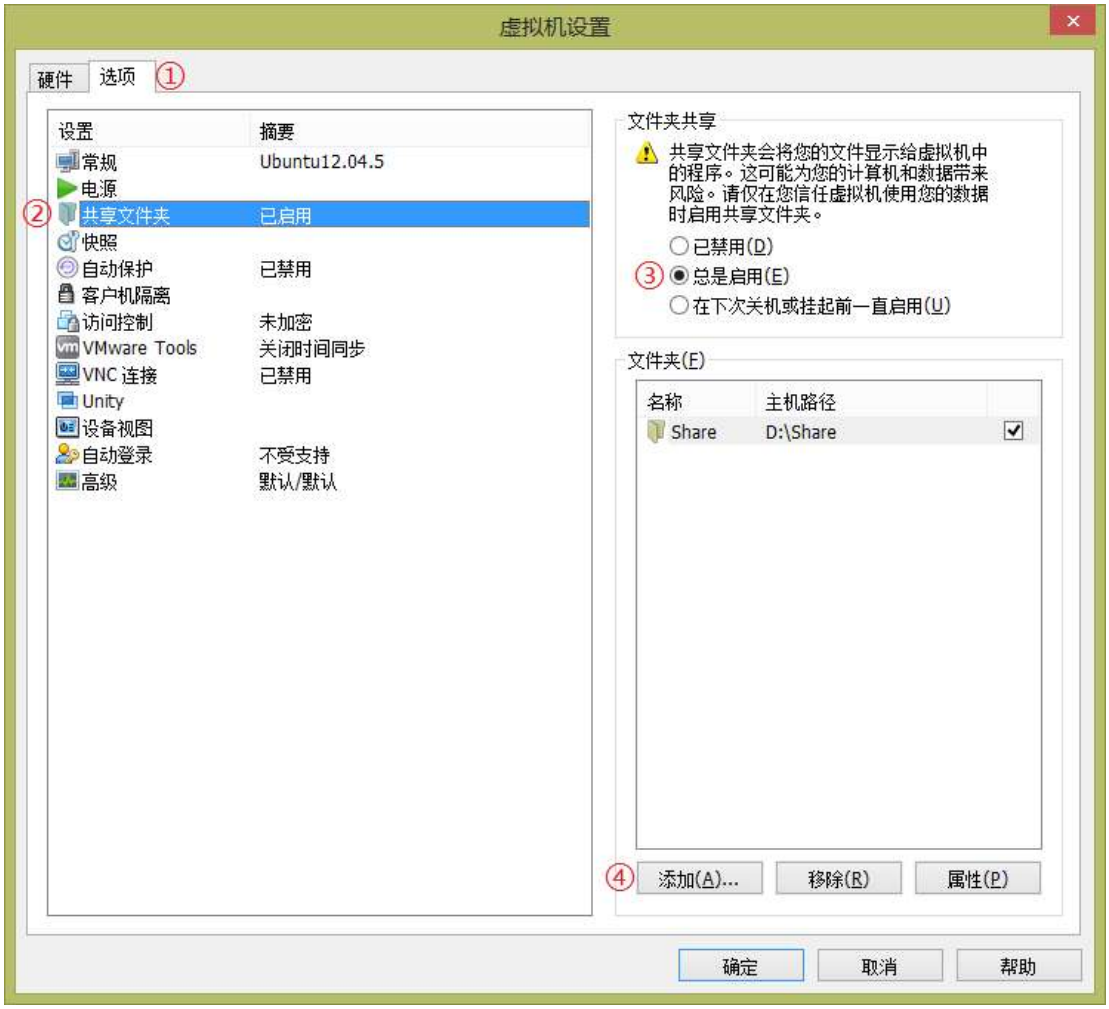
apt-get 命令主要用于自动从互联网的软件仓库中搜索、安装、升级、卸载软件或操作系统。

更新软件包命令：

```
$ sudo apt-get update
```

2.6.5 设置 Windows 和 Ubuntu 的共享文件夹

在 VMware Workstation 菜单栏中选择“虚拟机”→“设置”，在弹出框中选择“选项”选项卡，选中“共享文件夹”，选择“总是启用”，然后点击“添加…”选择主机下共享目录。



弹出的添加共享文件夹向导对话框中，在主机路径一栏中选择共享目录的路径，名称一栏为共享目录的名称，点击下一步到完成即可。

最后点击“确定”完成共享目录的添加。

完成后，Windows 上的文件放于“D:\Share”，Ubuntu 可以通过访问“/mnt/hgfs/Share/”来共享 Windows 上的文件；反之，Ubuntu 上的文件放于“/mnt/hgfs/Share/”，Windows 可以通过访问“D:\Share”来共享 Ubuntu 上的文件。这样就实现了 Windows 与 Ubuntu 间文件的共享与传输。提示：可以设置多个共享目录。

另外，还有一种较简单的传输方式——直接拖拽：在安装了 VMware Tools 之后，可以将 Windows 下的文件直接拖拽到 Ubuntu 的桌面或普通用户的目录下。反过来也一样。

2.6.6 安装配置 minicom 串口工具

minicom 是 Linux 上最常用的串口终端软件，它类似于 Windows 下的“超级终端”的软件，用来与串口设备通信，下面介绍它的使用方法。

1. 安装 minicom

minicom 的安装方法有两种：一种是连网在线安装（默认方式），另一种是使用源码包安装。

（1）连网在线安装

```
$ sudo apt-get install minicom
```

若提示没有发现 minicom 软件包，须更新系统软件包列表。（参考“4.2.4 更新 Ubuntu 软件包列表”）

（2）使用源码包安装

源码包位置：**Loongson1C300B\Tools\Linux_Tools\Minicom\install-minicom.tar.gz**

通过共享目录将该源码包拷贝到 Ubuntu 虚拟机普通用户目录下，然后解压运行安装脚本。

```
$ mkdir Workstation/tools/minicom
$ cp /mnt/hgfs/Share/install-minicom.tar.gz Workstation/tools/minicom/
$ cd Workstation/tools/minicom/
$ tar zxvf install-minicom.tar.gz
$ sudo ./install_minicom.sh
```

安装脚本内容如下：

```
#!/bin/bash

echo "=======tar minicom======="
tar zxf minicom_2.7.orig.tar.gz

echo "=======install minicom======="
cd minicom-2.7
./configure && make && make install
cd ..
rm -rf minicom-2.7

echo "=======install success======="
```

2. 配置 minicom

使用 minicom 串口工具之前，需要配置好相应串口的参数。

（1）Linux 串口设备

Linux 的设备都是以设备文件的方式出现在/dev 目录下，串口设备在该目录下是以“tty”开头的设备文件名。若串口是主机配备的，则设备文件名为 ttyS0、ttyS1 等。若是使用 USB 转串口线，则设备文件名为 ttyUSB0、ttyUSB1 等。这里使用的 USB 转串口线。

插入 USB 转串口线，在终端使用命令查看串口设备的设备文件名。输入“ls /dev/ttyU”，再按两下“Tab”键可查看当前接入 PC 机的串口设备，再根据需要选择其中一个。

```
loongson@ubuntu-desktop:~$ ls /dev/ttyU
ttyUSB0
```

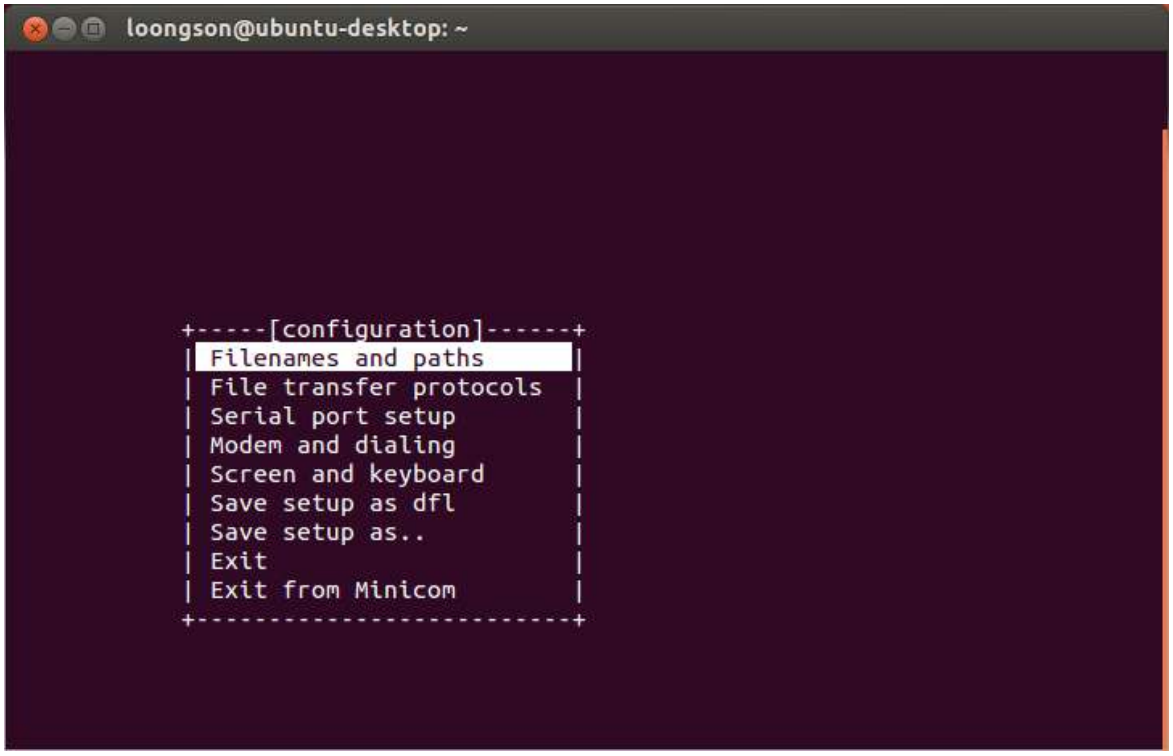
（2）运行 minicom

在终端输入命令，进入 minicom 配置界面。

```
$ sudo minicom -s
```

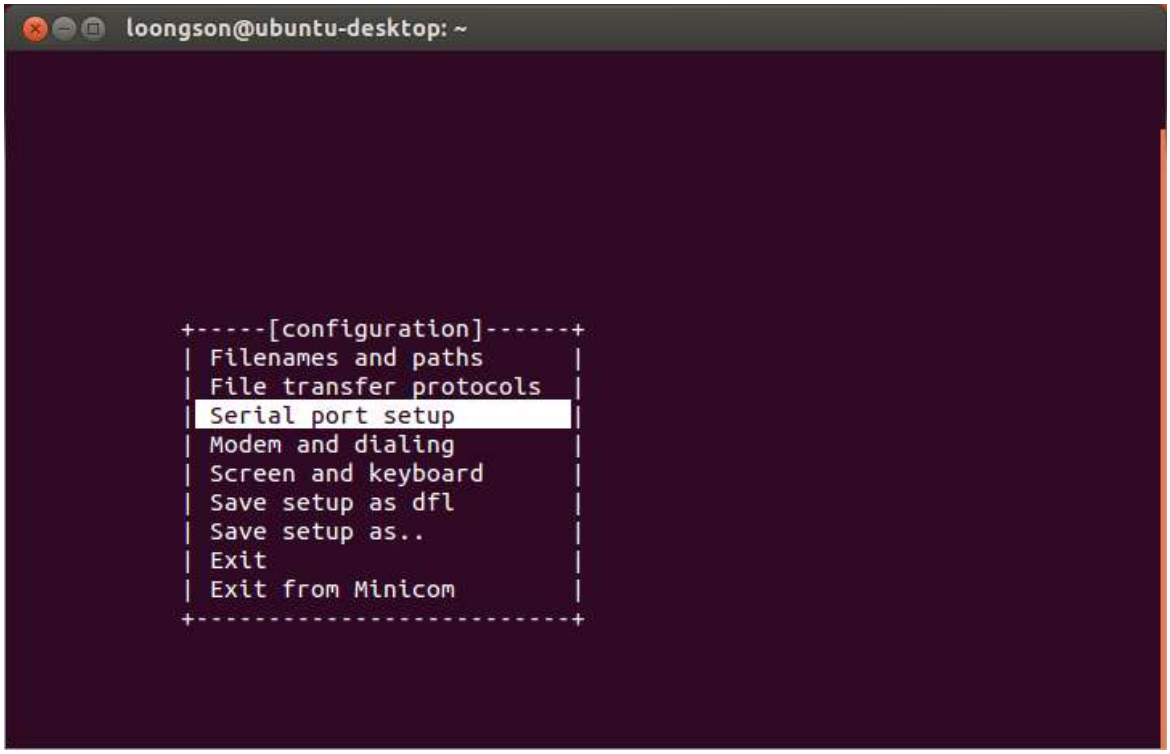
“-s”参数表示进入 minicom 配置界面。

在配置界面中，通过键盘的上下方向键来选中某个菜单，按回车键进入子菜单或执行该菜单的功能。

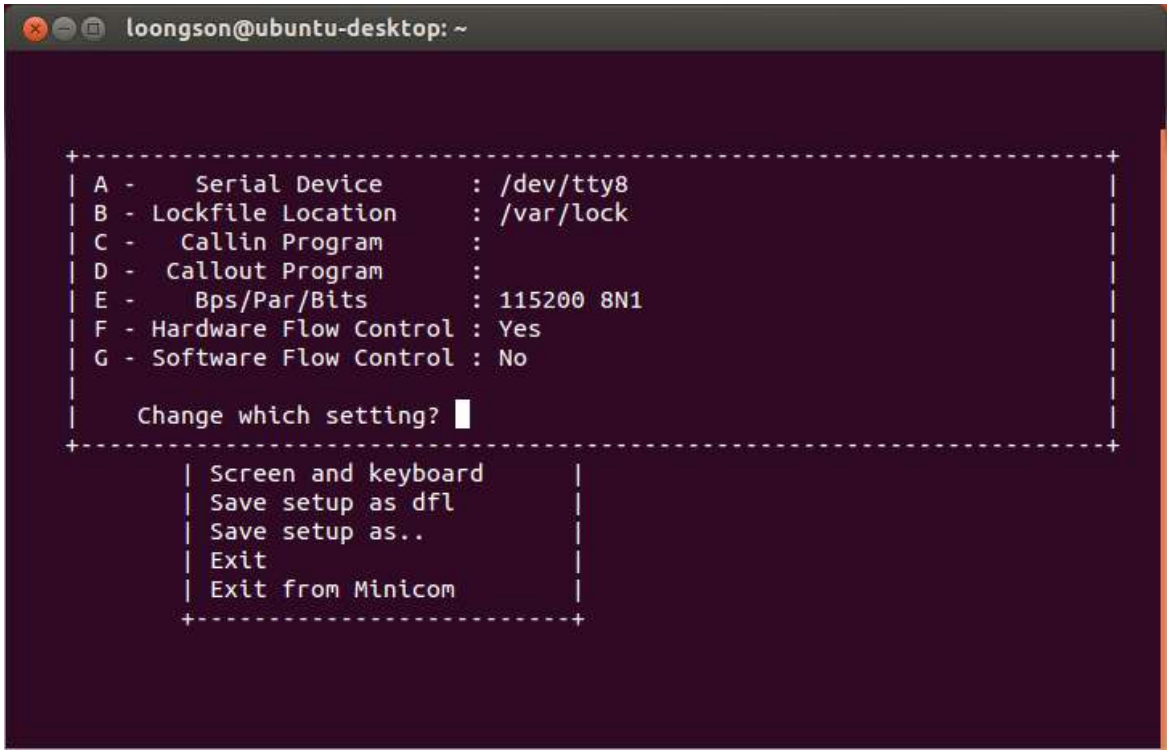


（3）进入串口配置界面

选中串口配置菜单“Serial port setup”，按回车键进入串口配置界面。

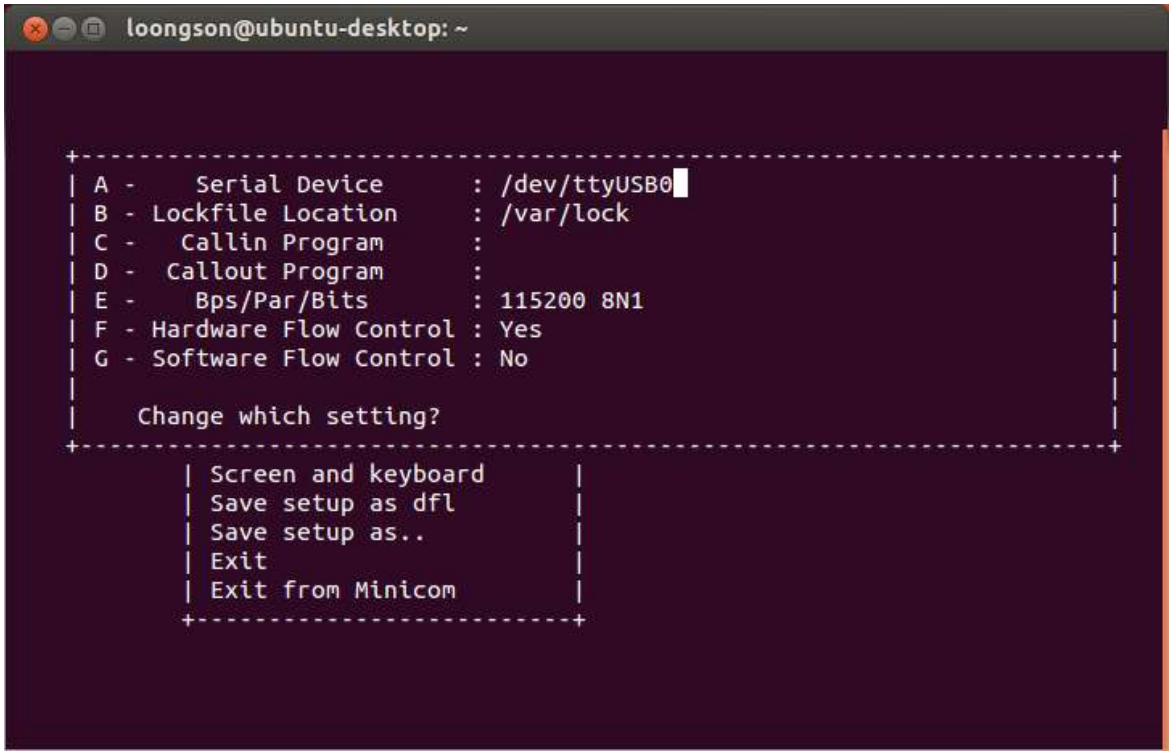


串口配置界面菜单中有 A-G 的选项，按相对应字母的键（不分大小写）即可进入相对应的菜单。



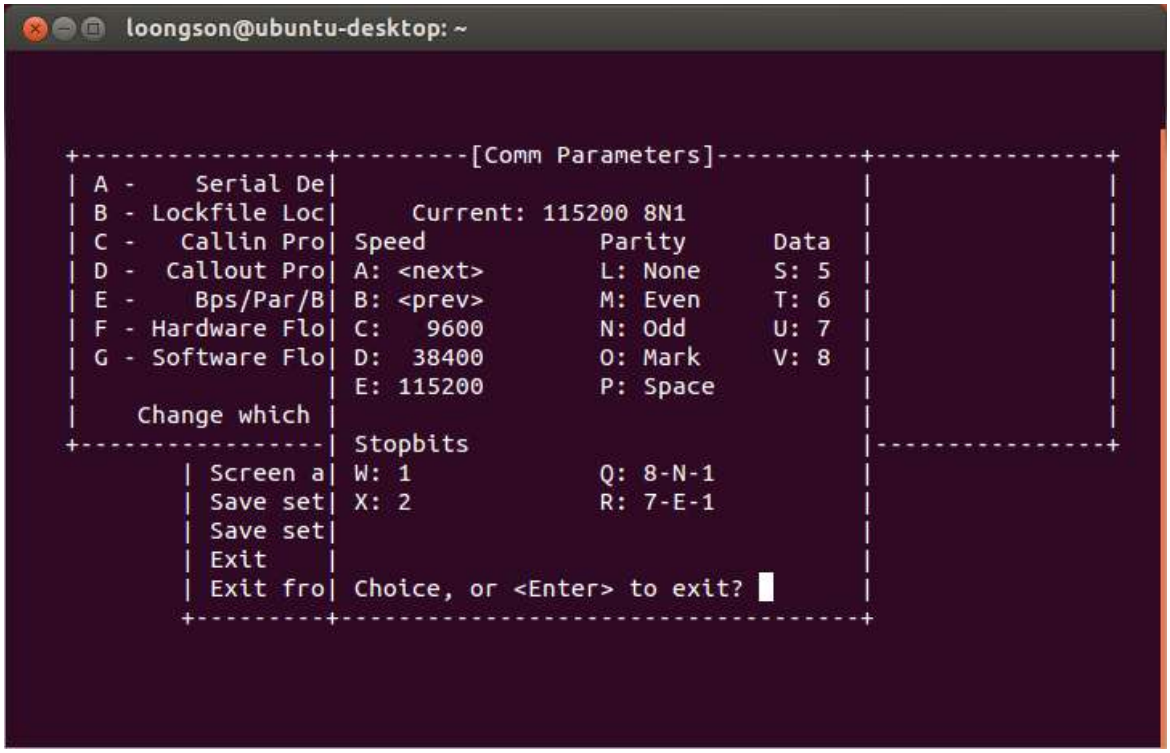
（4）配置串口设备

按“a”（或“A”）键进入“Serial Device”菜单项，将默认的串口设备文件名修改为实际的串口的设备文件名。按回车键退出该菜单项。

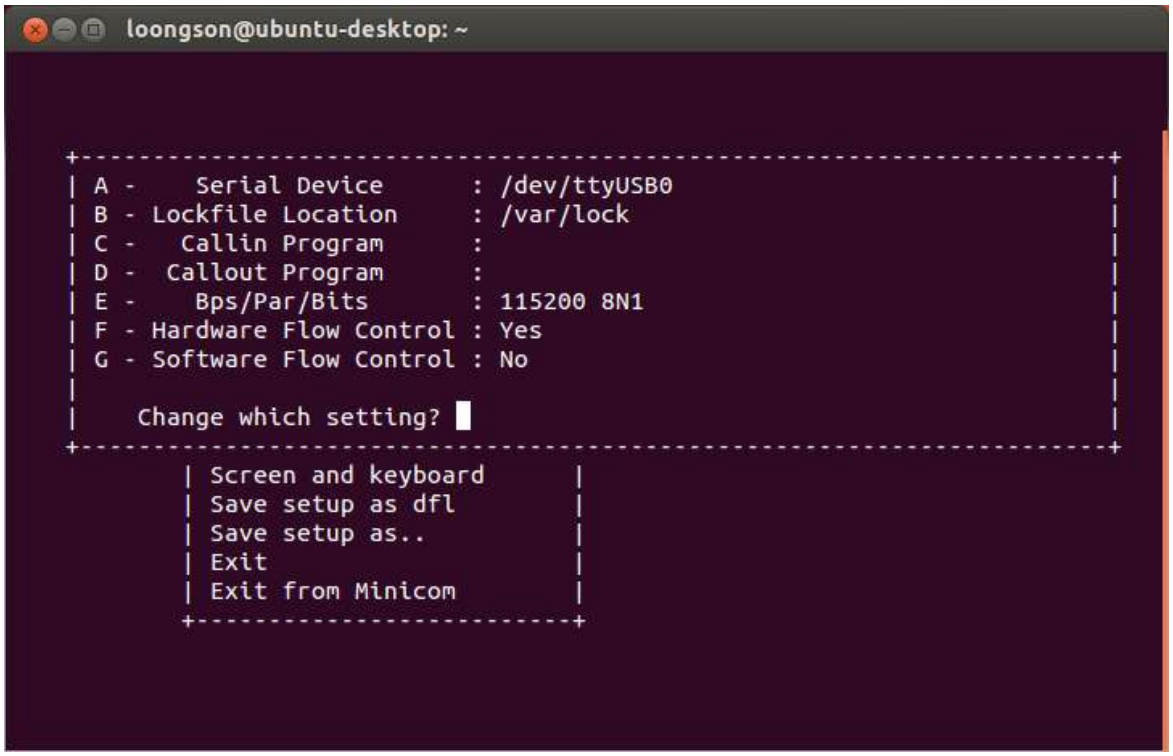


（5）配置串口属性

在实际使用中，串口设备的属性需设置为“115200 8N1”（115200 波特率、8 位数据位、无奇偶校验、1 位停止位）。若默认已配置好，则可跳过该步骤；若默认配置不是“115200 8N1”，按“e”（或“E”）键进入该项的配置界面。

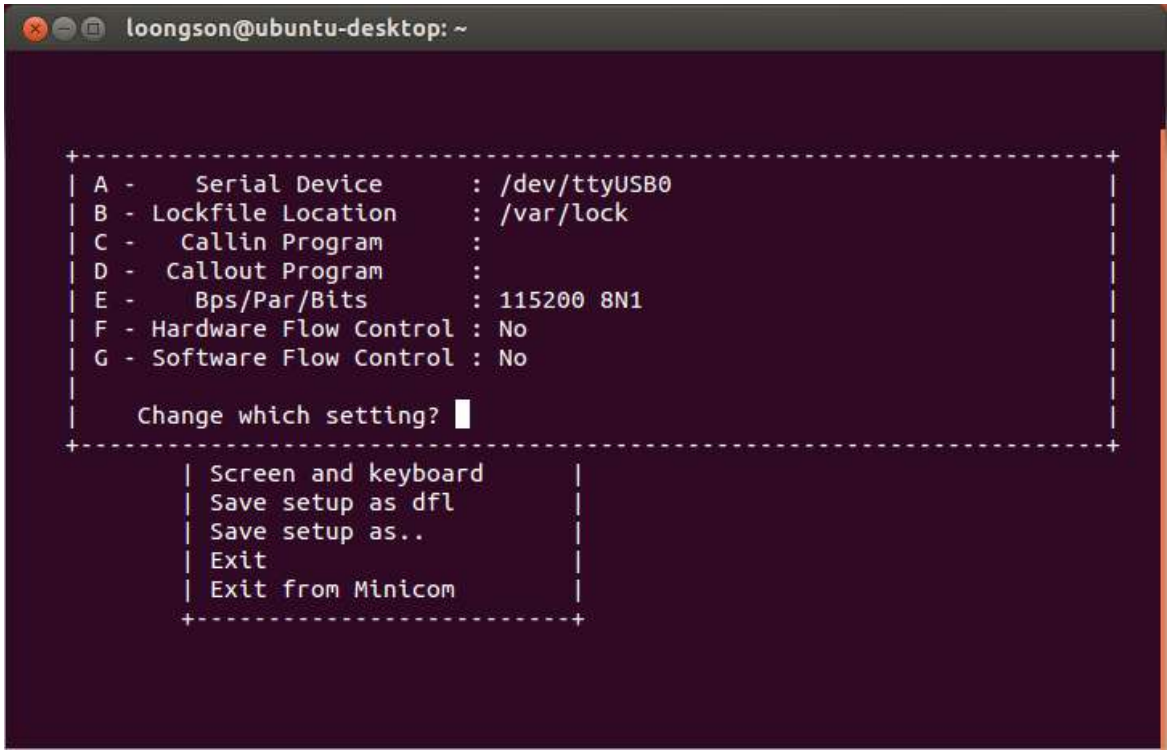


在该配置界面中，按“e”（或“E”）键选择 115200 的波特率，按“q”（或“Q”）键选择“8-N-1”，最后按回车键退出该配置界面。



（6）配置硬/软件流

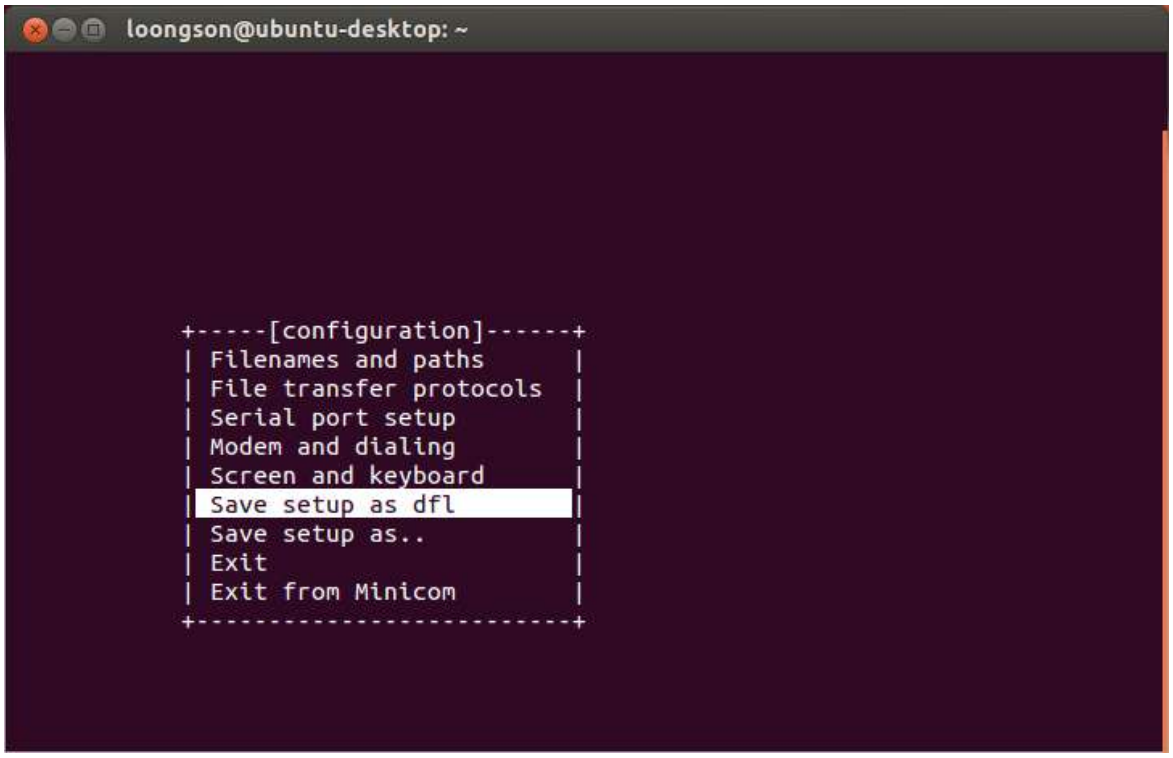
F 项和 G 项分别是硬件流控制和软件流控制，都只有 Yes 和 No 的配置选项。按“f”（或“F”）键可切换“Yes”和“No”的设置，这里设置为“No”。同样设置 G 项为“No”。



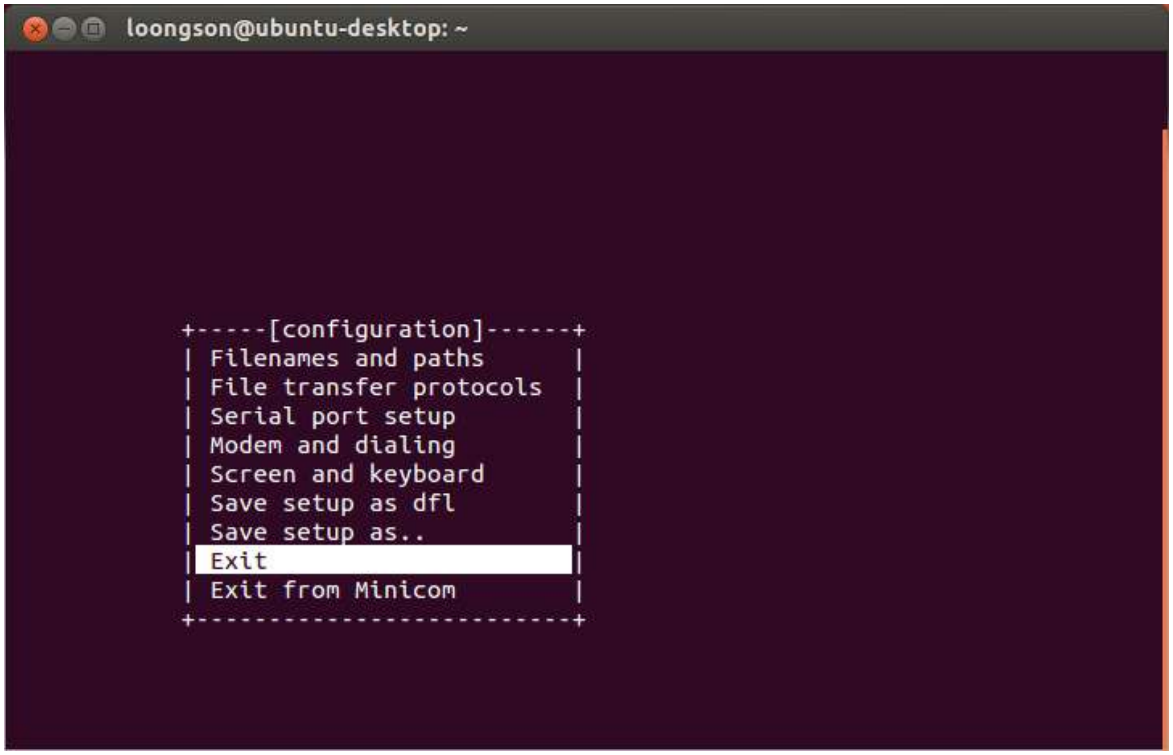
（7）保存配置并退出

配置以上选项后按回车键返回 minicom 配置主菜单。选中“Save setup as dfl”

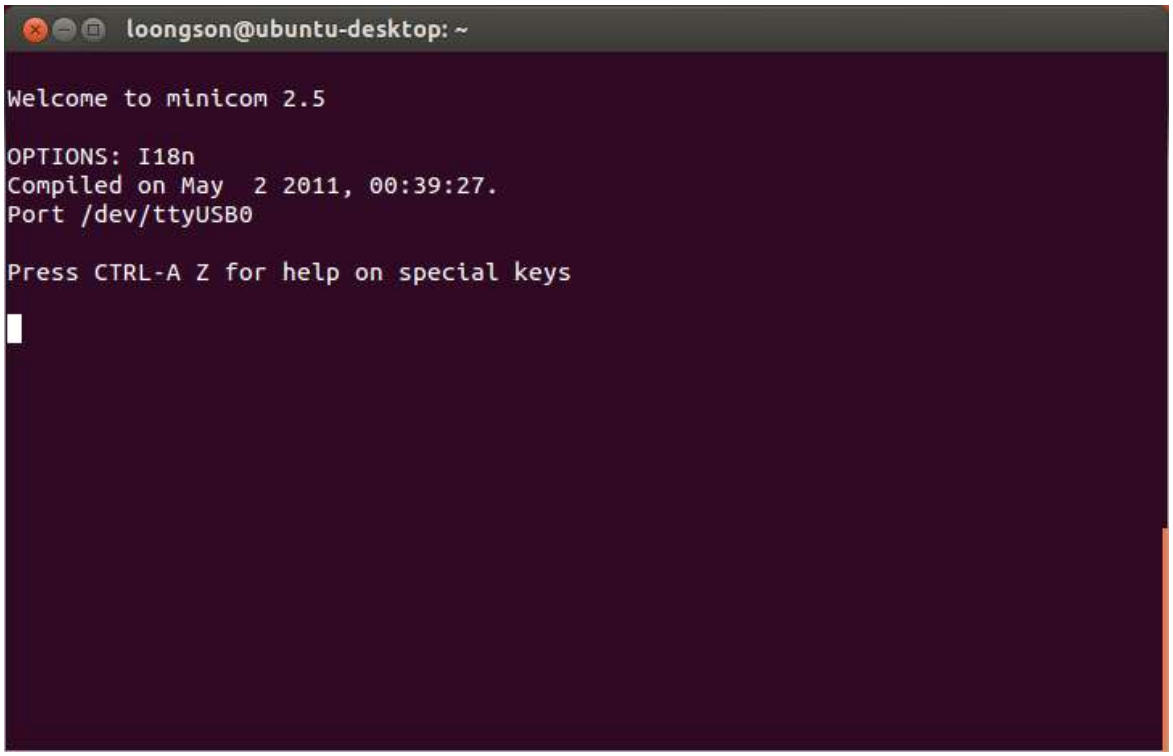
选项，按回车键保存为默认配置，在以后使用 minicom 时，输入“sudo minicom”即使用该默认配置。



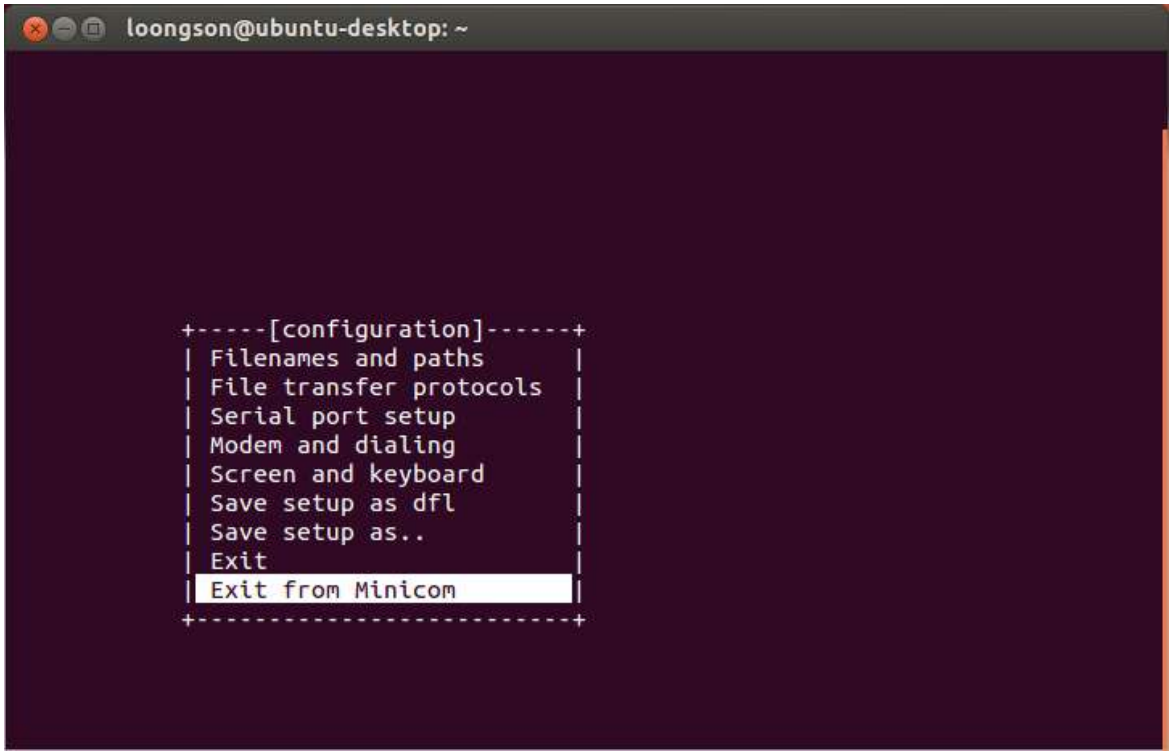
选中“Exit”选项按回车键，退出 minicom 配置主界面。



进入 minicom 终端界面。



选中“Exit from Minicom”选项并按回车键，则退出 minicom 配置主界面并返回 shell 终端界面。

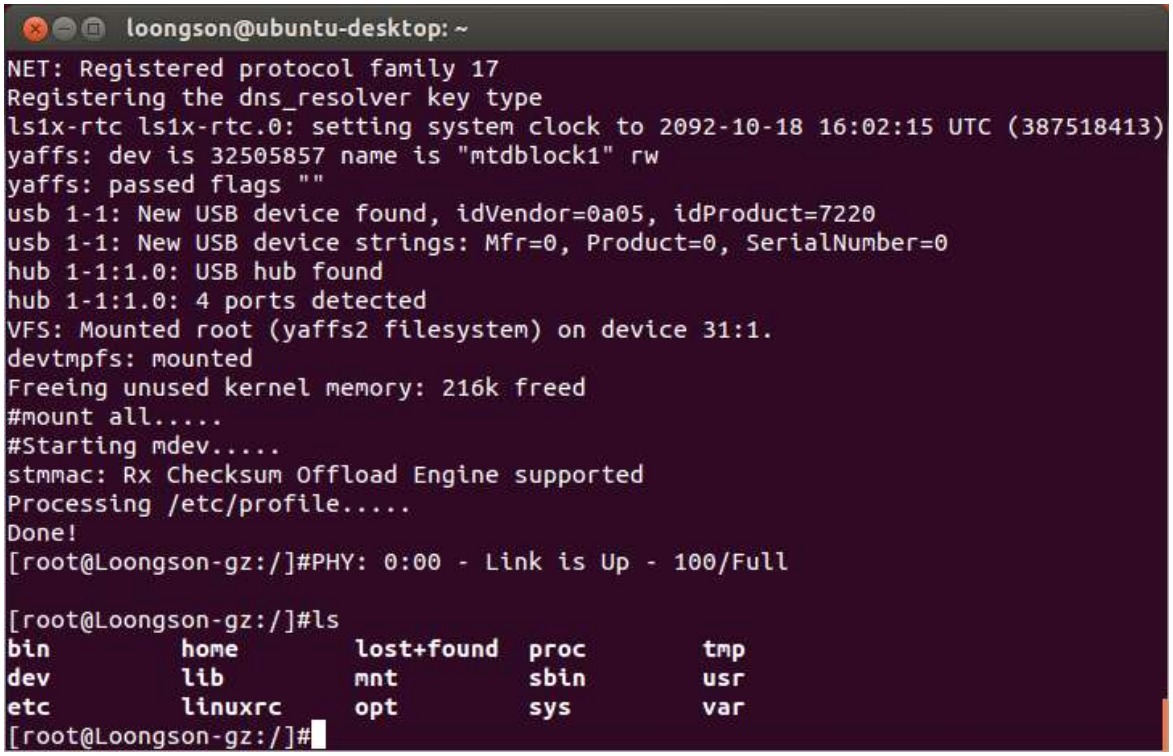


3. 使用 minicom

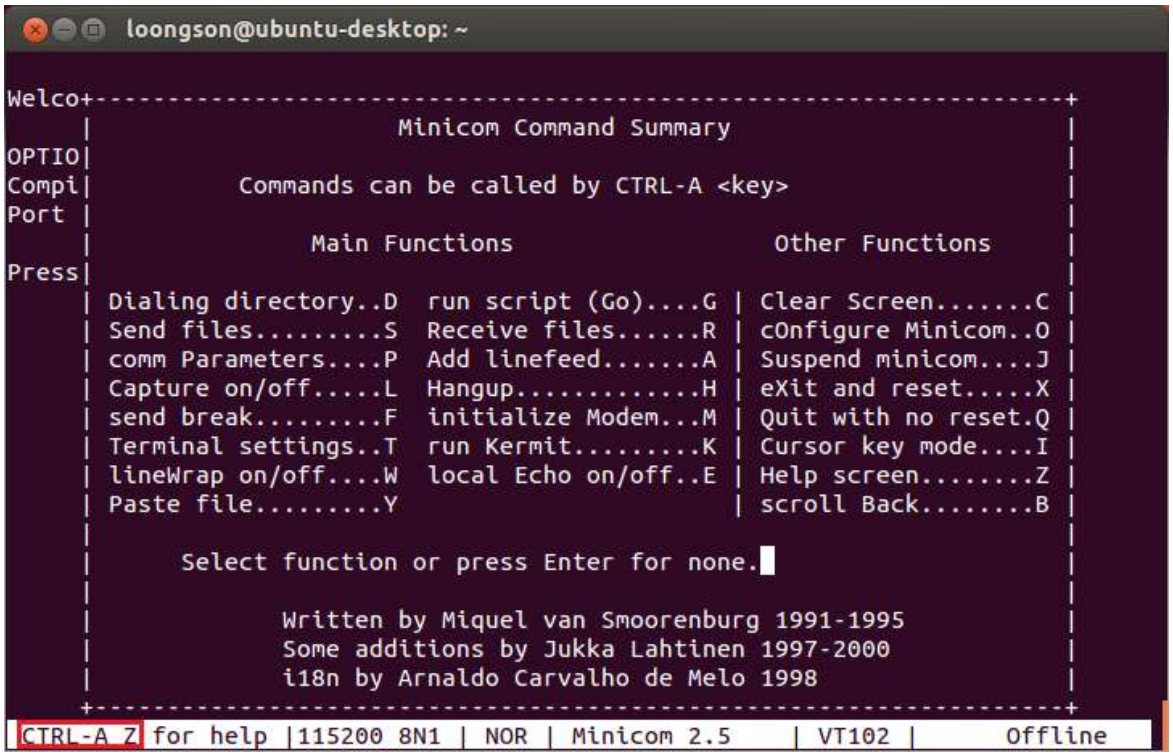
在终端输入命令运行 minicom。

$ sudo minicom

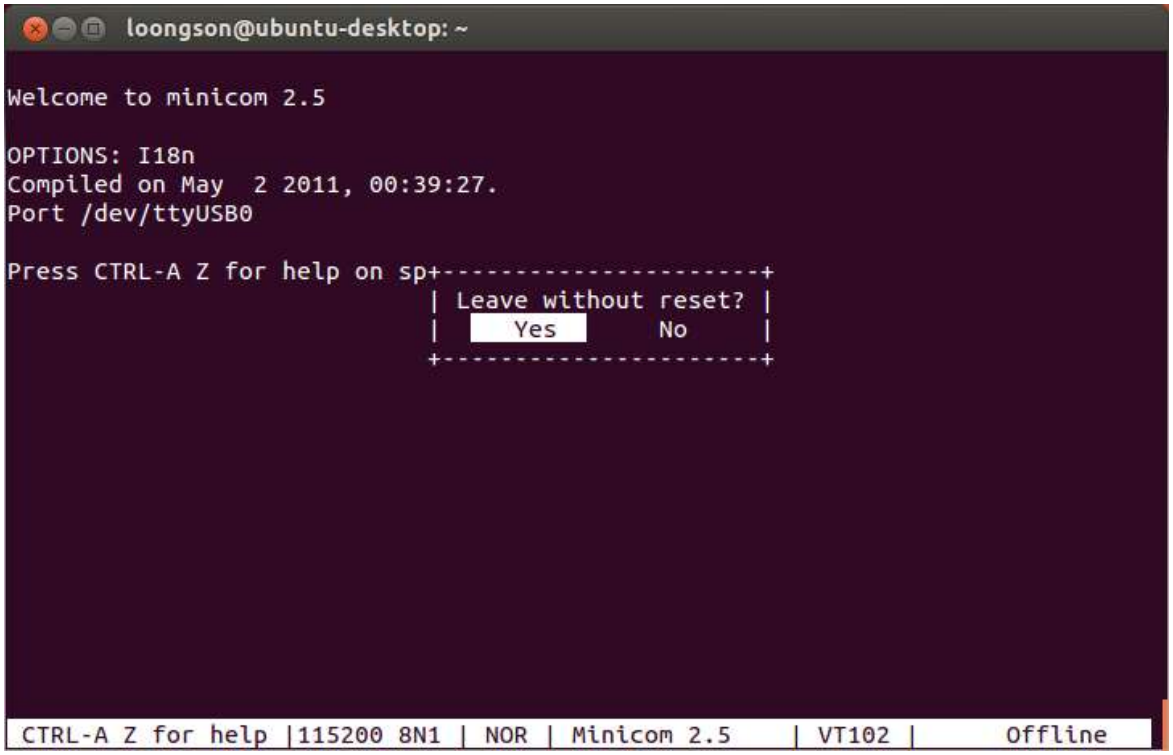
开发板上电后，minicom 会打印出系统启动时的打印信息，之后进入系统命令行模式。



运行 minicom 串口终端后要怎么退出呢？使用组合键“Ctrl+A”+“Z”（不分大小写）即可打开 minicom 的帮助界面。



按“q”（或“Q”）键选择“Quit with no reset”项弹出退出框。选中“Yes”按回车键即可退出 minicom。



2.6.7 安装配置 TFTP 服务器

TFTP（Trivial File Transfer Protocol,简单文件传输协议）是 TCP/IP 协议族中的一个用来在客户机与服务器之间进行简单文件传输的协议，提供不复杂、开销不大的文件传输服务。

由于智龙 V2.0 开发板的系统固件的恢复与更新通常是通过网络 TFTP 协议来传输的，所以需要使用 TFTP 服务器。因为在开发板的 PMON 上已经内置了 TFTP 的客户端，所以只需在主机上安装 TFTP 的服务端。

一、基于 linux 系统

1. 安装 TFTP 软件

使用终端在线安装 TFTP 软件和守护进程 xinetd

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

2. 创建 TFTP 目录

创建 TFTP 目录（上传及下载文件的位置），并更改其权限。

提示：TFTP 目录的路径可根据实际情况来创建。

```
$ mkdir Workstation/server/tftp
$ chmod 777 Workstation/server/tftp
```

3. 配置 TFTP 服务器

（1）修改配置文件/etc/default/tftpd-hpa

```
$ sudo gedit /etc/default/tftpd-hpa
```

将原来的内容改为：

```
# /etc/default/tftpd-hpa

TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/home/loongson/Workstation/server/tftp"
#tftp 目录路径
TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="-l -c -s"
```

（2）创建并修改配置文件/etc/xinetd.d/tftp

```
$ sudo touch /etc/xinetd.d/tftp
$ sudo gedit /etc/xinetd.d/tftp
```

添加如下内容：

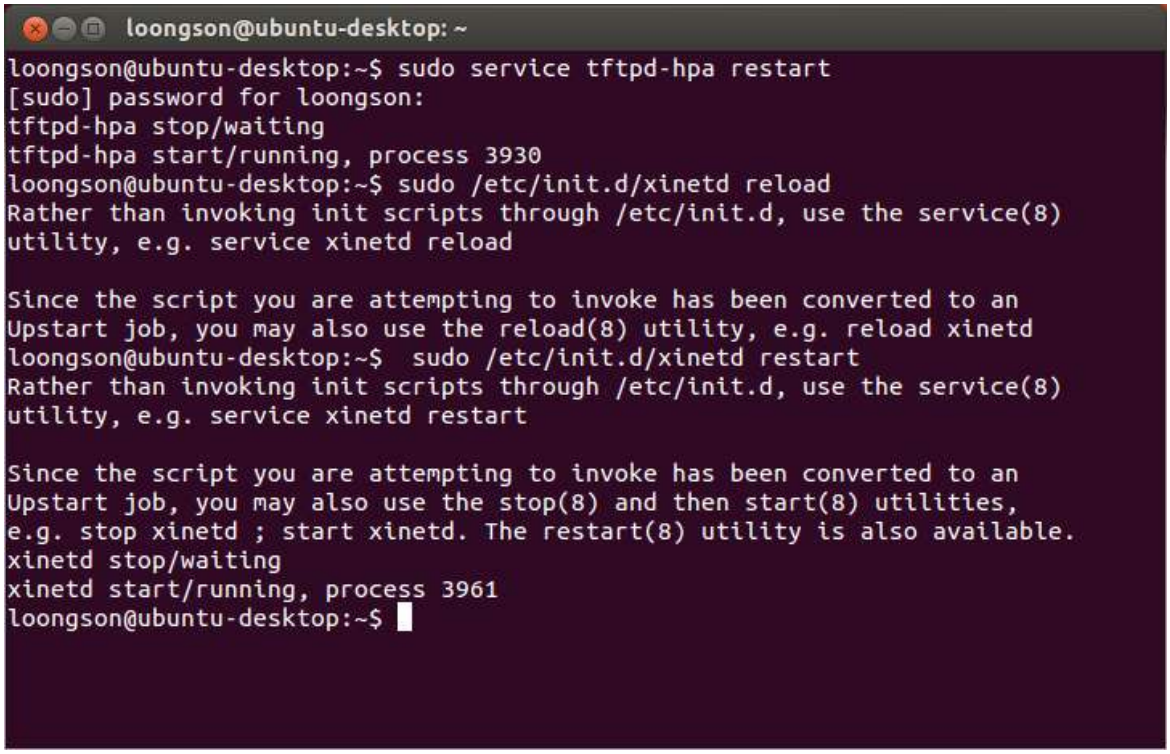
```
service tftp
{
        socket_type = dgram
        protocol        = udp
        wait             = yes
        user             = root
        server          = /usr/sbin/in.tftpd
        server_args     =     -s     -c     /home/loongson/Workstation/server/tftp
#tftp 目录路径
        disable         = no
        per_source    = 11
        cps              = 100 2
        flags           = IPv4
}
```

4. 重启 TFTP 服务

使用以下命令重启 TFTP 服务，或者重启 Ubuntu 系统。在以后的使用中 TFTP 服务都会随 Ubuntu 系统启动而开启。

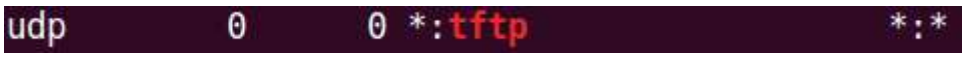
```
$ sudo service tftpd-hpa restart
$ sudo /etc/init.d/xinetd reload
$ sudo /etc/init.d/xinetd restart
```

提示：每次修改完配置文件后，都需要重新启动一下服务。



可通过以下命令查看 TFTP 服务是否开启，开启的查看结果下图示。

```
$ netstat -a | grep tftp
```



5. TFTP 服务测试

```
$ cd Workstation/server/tftp
$ echo “hello tftp service” > a.txt
$ cd ..
$ echo "hello tftp service, put to tftp service" > b.txt
$ tftp localhost
tftp> get a.txt
tftp> put b.txt
tftp> q
```

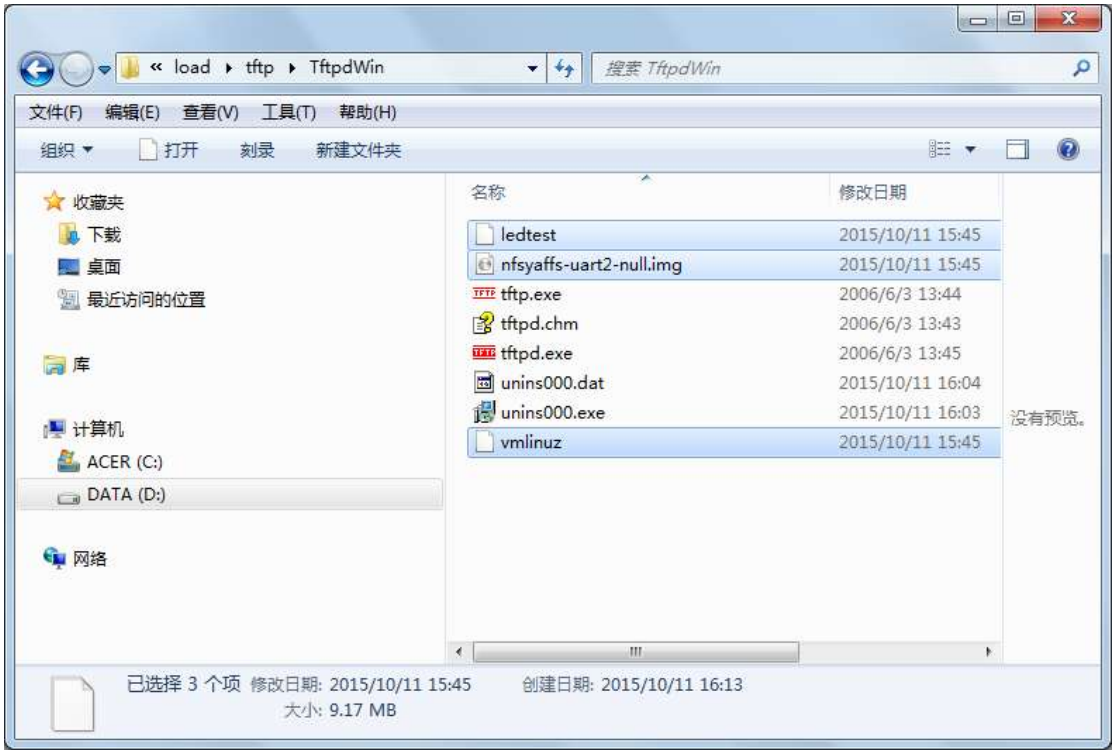
其中，get 是取得文件，put 是将文件上传到 TFTP 服务器上，q 是退出 tftp。提示：tftp> 是 tftp 运行后的命令行模式。

能从 TFTP 服务目录中 get 到文件，能将文件 put 到 TFTP 服务目录中，即完成 TFTP 配置。

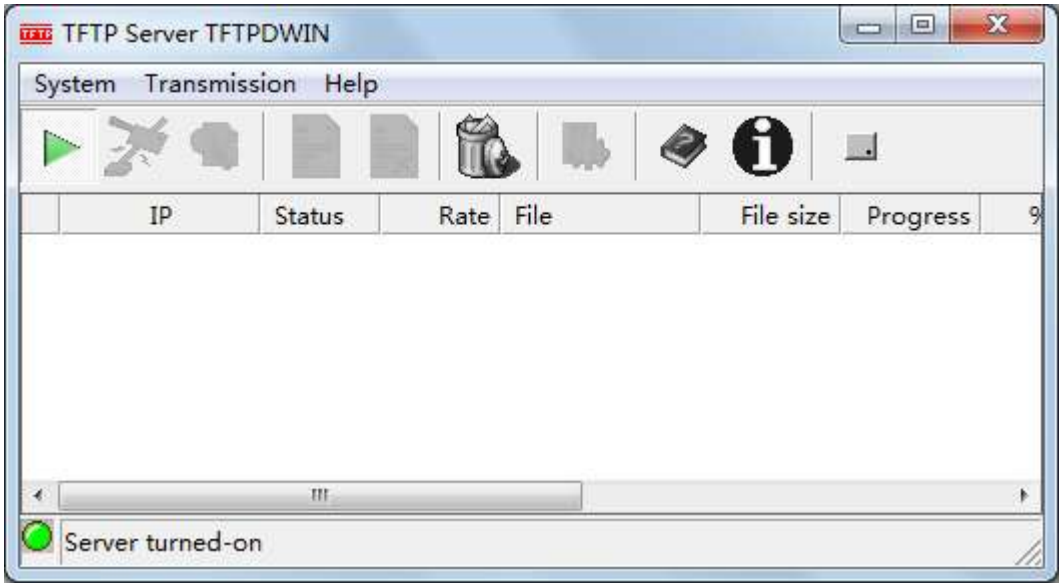
二、基于 windows 系统

1. 安装 TFTP

2. 把要编译好的文件放置于 TFTP 所在目录



3. 打开 TFTP，即可运行。



2.6.8 安装配置 NFS 服务器

NFS 不是传统意义上的文件系统，而是访问远程文件系统的协议。其最主要的功能就是让网络上的 linux 主机可以共享目录及档案。我们可以将远端所分享出来的档案系统，挂载(mount)在本地端的系统上，然後就可以很方便的使用远端的档案，而操作起来就像在本地操作一样，不会感到有什么不同。而使用 NFS 也有相当多的好处，例如档案可以集中管理等等，特别是对开发中的调试程序有极大的方便。下面则是 NFS 服务器搭建的步骤。

1. 安装 NFS 软件

使用终端在线安装 NFS 软件

```
$ sudo apt-get install nfs-kernel-server
```

2. 创建 NFS 目录

创建 NFS 目录，并更改其权限。提示：NFS 目录的路径可根据实际情况来创建。

```
$ mkdir Workstation/server/nfs
$ chmod 777 Workstation/server/nfs
```

3. 配置 NFS 服务器

修改配置文件/etc/exports

```
$ sudo vim /etc/exports
```

在文件的最后新建一行，添加如下内容：

```
/home/loongson/Workstation/server/nfs *(rw,sync)
```

内容含义：

/home/loongson/Workstation/server/nfs ： 要共享的目录，需要先创建后更改权限

* ：表示所有主机，也可指定特定的主机

192.168.1.13 指定 IP 地址的主机

nfsclient.test.com 指定域名的主机

192.168.1.0/24 指定网段中的所有主机

*.test.com 指定域下的所有主机

rw ： 读写访问

sync ： 同步写入

修改完成后按 Esc 键退出 vi 的编辑模式，输入“:wq”，回车保存退出。

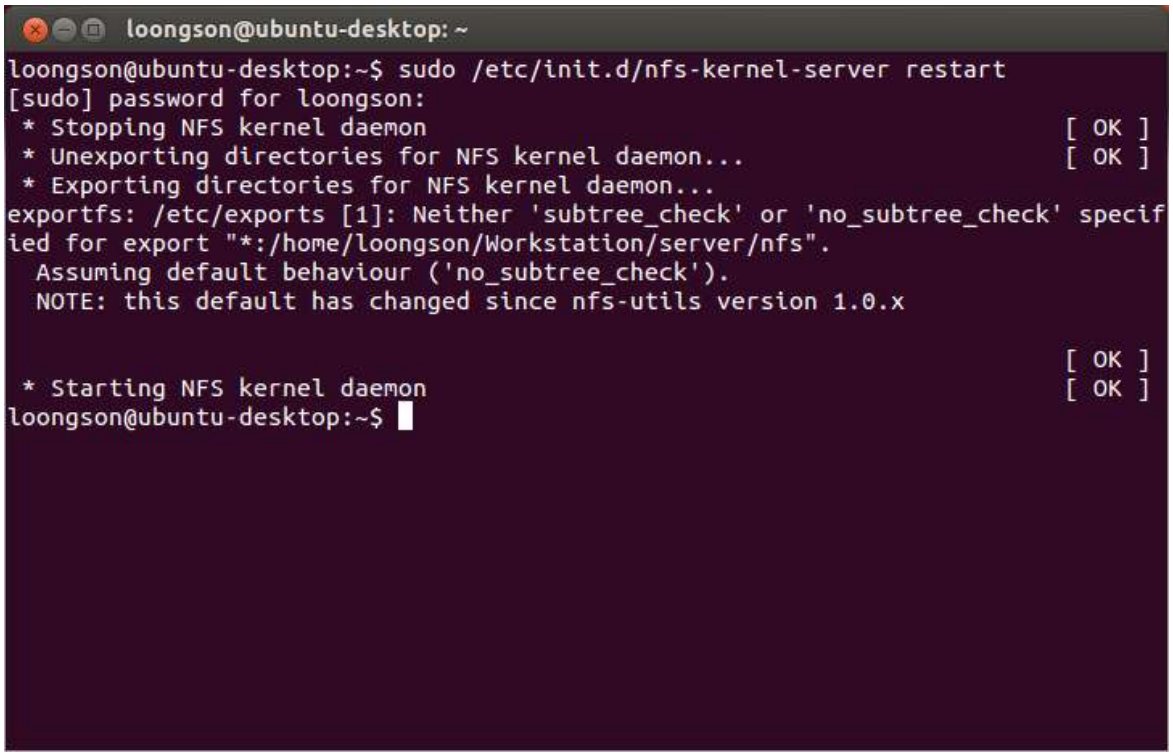
4. 重启 NFS 服务

使用以下命令重启 NFS 服务，或者重启 Ubuntu 系统。在以后的使用中无需再重启，除非修改配置文件。

```
$ sudo /etc/init.d/nfs-kernel-server restart
```

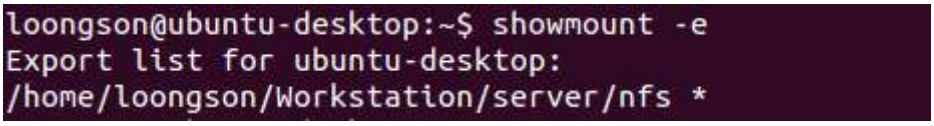
提示：每次修改完配置文件后，都需要重新启动一下服务。

当出现以下信息时，说明 NFS 服务正常重启。



查看 NFS 服务目录

```
$ showmount -e
```



5. NFS 服务测试

（1）在本机下测试 NFS 服务

在 NFS 服务目录中创建一个文件。

```
$ touch Workstation/server/nfs/nfs-test.txt
```

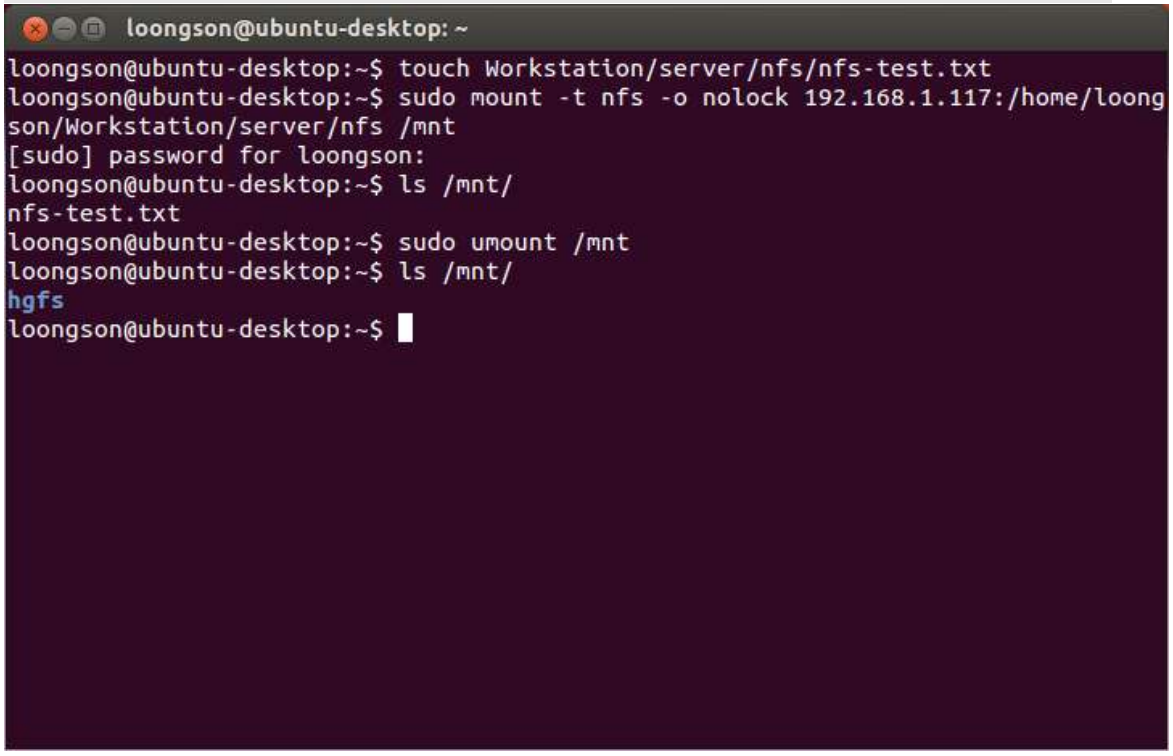
然后挂载 NFS 服务目录到/mnt 目录中。

```
$ sudo mount -t nfs -o nolock 192.168.1.117:/home/loongson/Workstation/server/nfs /mnt
```

可以查看到/mnt 目录下同样有个 nfs-test.txt 的文件。同样，若在/mnt 创建文件，在 NFS 服务目录中也可看到刚创建的文件。说明 NFS 服务已经开启。

取消挂载。

```
$ sudo umount /mnt
```



（2）在 1C300B 开发板下测试 NFS 服务

使用交叉串口线和网线连接开发板和主机，开发板上电并进入 minicom 串口终端。

注意：开发板的内核须配置有 NFS 的驱动（参考“6.5.4 配置 NFS 支持”）

使用 ping 命令确认开发板与主机间的网络是否连通。

```
# ping 192.168.1.117                                    //IP 为主机的 IP
```

网络连通后，在开发板上挂载 NFS 服务目录。

```
# mount -t nfs -o nolock 192.168.1.117:/home/loongson/Workstation/server/nfs /mnt
```

在/mnt 目录下可查看到在本机上创建的文件，说明在开发板挂载 NFS 服务目录成功。

```
root@ubuntu-desktop: /home/loongson
devtmpfs: mounted
Freeing unused kernel memory: 204k freed
#mount all.....
#Starting mdev.....
stmmac: Rx Checksum Offload Engine supported
Processing /etc/profile.....
Done!
[root@Loongson-gz:/]#PHY: 0:00 - Link is Up - 100/Full

[root@Loongson-gz:/]#ping 192.168.1.117
PING 192.168.1.117 (192.168.1.117): 56 data bytes
64 bytes from 192.168.1.117: seq=0 ttl=64 time=5.889 ms
64 bytes from 192.168.1.117: seq=1 ttl=64 time=0.986 ms
64 bytes from 192.168.1.117: seq=2 ttl=64 time=1.007 ms
64 bytes from 192.168.1.117: seq=3 ttl=64 time=0.994 ms
^C Ctrl+C 结束测试
--- 192.168.1.117 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.986/2.219/5.889 ms
[root@Loongson-gz:/]#mount -t nfs -o nolock 192.168.1.117:/home/loongson/Worksta
tion/server/nfs /mnt
[root@Loongson-gz:/]#ls /mnt/
nfs-test.txt
[root@Loongson-gz:/]#
```

2.6.9 建立交叉编译环境

源码包位置：

Loongson_1C300B\Tools\Linux_Tools\Toolchain\gcc-4.7.3-mips32.tar.gz

通过共享目录将该源码包拷贝到 Ubuntu 虚拟机普通用户目录下。

```
$mkdir Workstation/tools/toolchain
$ cp /mnt/hgfs/Share/gcc-4.7.3-mips32.tar.gz Workstation/tools/toolchain/
```

将交叉编译工具链解压到/opt 目录下。

```
$ sudo tar zxvf gcc-4.7.3-mips32.tar.gz -C /opt
```

新建脚本文件 setenv-toolchain.sh，用于将工具链路径添加到系统环境变量中。内容如下：

```
#!/bin/bash
export PATH=/opt/gcc-4.7.3-mips32/usr/bin:$PATH
```

需要用到交叉编译工具链时，source 该脚本文件就可以了，可查看环境变量 PATH 值，看是否添加成功。

```
$ source setenv-toolchain.sh
$ echo $PATH
```

```
loongson@ubuntu-desktop:~/Workstation/tools/toolchain$ source setenv-toolchain.s
h
loongson@ubuntu-desktop:~/Workstation/tools/toolchain$ echo $PATH
/opt/gcc-4.7.3-mips32/usr/bin:/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/loca
l/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
```

2.7 PMON 的配置和编译

PMON 是一个兼有 BIOS 和 bootloader 部分功能的开放源码软件，多用于嵌入式系统。基于龙芯的系统采用 PMON 作为类 BIOS 兼 bootloader，并在其基础上做了很多完善工作，支持 BIOS 启动配置，内核加载，程序调试，内存寄存器显示、设置以及内存反汇编等等。

源码包位置：**Loongson_1C300B\BSP\Pmon\PMON.tar.gz**
通过共享目录将该源码包拷贝到 Ubuntu 虚拟机普通用户目录下。

```
$ mkdir Workstation/tools/pmon
$ cp /mnt/hgfs/Share/PMON.tar.gz Workstation/tools/pmon
```

2.7.1 安装依赖库和编译工具

依赖库的安装有两种方式：一种是连网在线安装（默认方式）；另一种是使用源码包安装。

1 连网在线安装

编译 PMON 需要使用到工具 pmoncfg，该工具位于 PMON/tools/pmoncfg 目录下，而编译该工具又需要依赖下面的库：

```
$ sudo apt-get install bison
$ sudo apt-get install flex
```

解压 PMON 源码包，并进入相应的目录。

```
$ cd Workstation/tools/pmon
$ tar zxvf PMON.tar.gz
$ cd PMON/tools/pmoncfg
```

执行编译命令，将生成的可执行文件拷贝到交叉编译工具链的 bin 目录下。

```
$ make
$ sudo cp pmoncfg /opt/gcc-4.7.3-mips32/usr/bin
```

编译 PMON 还依赖于工具 makedepend：

```
$ sudo apt-get install xutils-dev
```

2 使用源码包安装

源码包位置：

Loongson_1C300B\Tools\Linux_Tools\Pmon\pmon-dependtools.tar.gz

提示：需要把源码包 pmon-dependtools.tar.gz 放在与 PMON.tar.gz 相同的目录下。

通过共享目录将该源码包拷贝到 Ubuntu 虚拟机普通用户目录下。

```
$ cp /mnt/hgfs/Share/pmon-dependtools.tar.gz Workstation/tools/pmon
```

解压源码包并执行安装脚本文件。

```
$ cd Workstation/tools/pmon
$ tar zxvf pmon-dependtools.tar.gz
$ sudo ./pmon-dependtools.sh
```

pmon-dependtools.sh 的内容为：

```
#!/bin/bash

echo "=======pmon depend tools install========"

tar jxf m4_1.4.16.orig.tar.bz2 #configure 时需要
tar jxf bison_2.5.dfsg.orig.tar.bz2
tar zxf flex_2.5.35.orig.tar.gz
tar zxf xutils-dev_7.7~1.tar.gz
tar zxf PMON.tar.gz

cd m4-1.4.16
./configure && make && make install
cd ..
rm -rf m4-1.4.16

cd bison-2.5.dfsg
./configure && make && make install
cd ..
rm -rf bison-2.5.dfsg

cd flex-2.5.35
./configure && make && make install
cd ..
rm -rf flex-2.5.35

cd xutils-dev-7.7~1
./configure && make && make install
cd ..
rm -rf xutils-dev-7.7~1

cd PMON/tools/pmoncfg
make
```

```
cp pmoncfg /opt/gcc-4.7.3-mips32/usr/bin
cd ../../..
```

2.7.2 配置 PMON

1 配置系统启动方式

1 SPI Flash 启动

PMON 默认支持 SPI Flash 启动，只需将源码包解压，然后编译即可。
PMON 的默认配置源码如下：

```
select      nand
#option     NAND_BOOT_EN            #表示使用nandflash启动
#option     NAND_PAGE_SIZE=2048     #nand页大小
#option     NAND_PARAMETER=0x000    #外部颗粒容量大小 NAND_PARAMETER（寄存器地址：0x1fe7_8018）
                                    #注意：根据nand flash大小修改,低8bit保留
```

硬件上设置参考“2.4 1C300B 系统启动方式配置说明”。

2 Nand Flash 启动

修改 PMON 的配置文件，使其支持 NAND FLASH 启动。

```
$ cd PMON
$ vi Targets/LS1X/conf/ls1c
```

修改为如下内容（即取消原有的注释）：

```
select      nand
option      NAND_BOOT_EN            #表示使用nandflash启动
option      NAND_PAGE_SIZE=2048     #nand页大小
option      NAND_PARAMETER=0x000    #外部颗粒容量大小 NAND_PARAMETER（寄存器地址：0x1fe7_8018）
                                    #注意：根据nand flash大小修改,低8bit保留
```

硬件上设置参考“2.4 1C300B 系统启动方式配置说明”。

2 配置串口

开发板支持的串口有 4 个，分别是 UART0、UART3、UART7 和 UART8。（参考“2.3.2 开发底板硬件资源说明”）

打开 PMON 的配置文件，找到串口地址设置选项，默认地址对应的串口为 UART3。

```
$ vi Targets/LS1X/conf/ls1c
option      UART_BASE_ADDR=0xbfe4c000   # 串口控制台基地址 需要不同的串口输出可以修改此值
```

然后对照 Loongson1C300B 处理器用户手册中给出的各个串口寄存器的地址，在 PMON 中设置相对应的串口地址。同时还要修改 PMON 启动参数（参考 8.4.2 设置文件系统启动参数）和文件系统配置文件 inittab（参考 7.4.2 创建系统配置文件）的串口号。

2.7.3 编译 PMON

（1）建立交叉编译环境（参考“4.2.9 建立交叉编译环境”）

```
$ source ~/Workstation/tools/toolchain/setenv-toolchain.sh
```

（2）安装好依赖库和工具后，开始编译 PMON：

```
$ cd PMON/zloader.ls1c
$ make cfg all tgt=rom CROSS_COMPILE=mipsel-linux-
```

执行后在当前目录下生成 PMON 的二进制文件 gzrom.bin。

2.8 基于 linux 的根文件系统

2.8.1 创建文件系统目录

创建文件系统目录

```
$ cd ~/Workstation/tools/makefs
$ mkdir rootfs
$ cd rootfs
$ mkdir dev home proc tmp var etc lib mnt sys opt root etc/rc.d
```

2.8.2 创建系统配置文件

进入刚创建的“rootfs”目录中。

```
$ cd rootfs
```

（1）etc/inittab 文件

说明：inittab 文件是 init 进程的配置文件，系统启动后所访问的第一个脚本文件，后续启动的文件都由它指定。

```
$ vi etc/inittab
```

添加如下内容：

```
::sysinit:/etc/rc.d/rc.sysinit                    #指定系统启动后首先执行的文件

#Example of how to put a getty on a serial line (for a terminal)
ttyS3::respawn:-/bin/sh                           #串口终端，串口号 ttyS3 要与启动参数一致
tty1::respawn:-/bin/sh                            #用于开发板屏幕终端显示

#Stuff to do when restarting the init process
::restart:/sbin/init

#Stuff to do before rebooting
::ctrlaltdel:/sbin/reboot                         #捕捉 ctrl+alt+del 键，重启文件系统
::shutdown:/bin/umount -a -r                      #当关机时卸载所有文件系统
::shutdown:/sbin/swapoff -a
```

然后保存退出。

上面 ttyS3::respawn:-/bin/sh 中的 ttyS3 是指终端的控制台输出口，如想使用 ttyS1 或其他口，则需修改此配置，同时在文件系统启动参数也要相应修改对应的串口。（参考“8.3.4 设置文件系统启动参数”）

（2）etc/rc.d/rc.sysinit 文件

说明：这是一个脚本文件，可以在里面添加想自动执行的命令。以下命令配置环境变量、主机名、dev 目录环境、挂接/etc/fstab 指定的文件系统、建立设备节点与设置 IP。

```
$ vi etc/rc.d/rc.sysinit
```

添加如下内容：

```
#!/bin/sh

#Set binary path
export PATH=/bin:/sbin:/usr/bin:/usr/sbin

#Set hostname
/bin/hostname "Loongson-gz"
```

```
#Config dev enviornment
mount -t tmpfs -o size=64k,mode=0755 tmpfs /dev
mkdir -p /dev/pts
mount -t devpts devpts /dev/pts

#mount all filesystem defined in "/etc/fstab"
echo "#mount all....."
/bin/mount -a

echo "#Starting mdev....."
echo /sbin/mdev > /proc/sys/kernel/hotplug
/sbin/mdev -s

#Set ip
ifconfig eth0 192.168.1.107 up
ifconfig lo 127.0.0.1
```

（3）etc/fstab 文件

说明：执行 mount -a 时挂接/etc/fstab 指定的文件系统。

$ vi etc/fstab

添加如下内容：

```
sysfs    /sys      sysfs    defaults    0    0
proc      /proc    proc      defaults    0    0
tmpfs    /dev      tmpfs    defaults    0    0
tmpfs    /tmp      tmpfs    defaults    0    0
tmpfs    /var      tmpfs    defaults    0    0
```

（4）etc/profile 文件

说明：inittab 中执行了这样一个语句“respawn:-/bin/sh”。启动/bin/sh 程序时会启动 ash 的配置信息，而它就是/etc/profile，sh 会把 profile 的所有配置全部都运行一遍，因此用户可以把自己的启动程序放在这里。

$ vi etc/profile

添加如下内容：

```
#!/bin/sh
#/etc/profile:system-wide .profile file for the Bourne shells
echo "Processing /etc/profile....."

#set search library path
export LD_LIBRARY_PATH=/lib:/usr/lib
```

```
#set user path
export PATH=/bin:/sbin:/usr/bin:/usr/sbin

#Set PS1
USER="root"
#LOGNAME=$USER
HOSTNAME='/bin/hostname'
PS1='[$USER@\h:\w]\$'
echo "Done!"
```

（5）修改系统配置文件权限

```
$ chmod 755 etc/*
$ chmod 755 etc/rc.d/rc.sysinit
```

（6）拷贝 Busybox 文件

将编译安装好的 Busybox 文件拷贝到 rootfs 目录中：

```
$ cp ../busybox-1.22.1/_install/* . -rd
```

2.8.3 拷贝库文件

提示：配置 Busybox 若选择静态编译则省略该步骤。

动态编译 Busybox 制作文件系统，需将工具链目录 /opt/gcc-4.7.3-mips32/usr/mipsel-buildroot-linux-gnu/sysroot/lib/ 中的几个必须库:

```
librt.so.1、ld.so.1、libc.so.6、libcrypt.so.1、libm.so.6、libdl.so.2、libpthread.so.0
```

和目录/opt/gcc-4.7.3-mips32/usr/mipsel-buildroot-linux-gnu/lib/ 中的两个库文件：

```
libgcc_s.so.1、libstdc++.so.6
```

拷贝到根文件系统下的 lib 目录。

这些库文件中除了 libgcc_s.so.1 之外都是链接文件，因此拷贝这些库文件也要拷贝被链接的文件。在相应的目录中执行“ll”命令可查看这些库文件的链接关系。在执行拷贝命令时需要在命令后加“-d”参数，用于保留文件的链接属性。

使用脚本快速拷贝所需的基本动态库：

```
$ vi cplib.sh
```

添加如下内容：

```
#!/bin/bash
# copy mipsel-linux-gcc-4.7.3 lib

libdir=/opt/gcc-4.7.3-mips32/usr/mipsel-buildroot-linux-gnu
fsdir=/home/loongson/Workstation/tools/makefs/rootfs
```

```
cp $libdir/sysroot/lib/ld*                $fsdir -d
cp $libdir/sysroot/lib/libc*              $fsdir -d
cp $libdir/sysroot/lib/libm*              $fsdir -d
cp $libdir/sysroot/lib/librt*            $fsdir -d
cp $libdir/sysroot/lib/libdl*            $fsdir -d
cp $libdir/sysroot/lib/libpthread* $fsdir -d

cp $libdir/lib/libgcc_s.so.1              $fsdir
cp $libdir/lib/libstdc++.so.6*          $fsdir -d

rm $fsdir/libcidn*
rm $fsdir/libmemusage.so
rm $fsdir/*py
```

运行脚本：

```
$ chmod +x cplib.sh
$ ./cplib.sh
```

为了减少根文件系统的库大小，使用交叉编译工具即 gcc-4.7.3-mips32 的 strip 工具来处理库文件，把二进制文件中的包含的符号表和调试信息删除掉,可有效减少库文件大小。

```
$ source ~/Workstation/tools/toolchain/setenv-toolchain.sh
$ mipsel-linux-strip rootfs/lib/*so*
```

至此，根文件系统（目录/home/loongson/Workstation/tools/makefs/rootfs）制作完成。

2.9 制作根文件系统镜像

2.9.1 安装镜像文件制作工具

源码包位置：

Loongson_1C300B\Tools\Linux_Tools\Makefs\install-makefstool.tar.gz

通过共享目录将该源码包拷贝到 Ubuntu 虚拟机普通用户目录下。

```
$ cp /mnt/hgfs/Share/install-makefstool.tar.gz Workstation/tools/makefs
```

源码包中包含的工具有：

（a）zlib-1.2.8.tar.gz	依赖工具
（b）cramfs-1.1.tar.gz	制作 cramfs 文件系统工具
（c）yaffs2-d43e901.tar.gz	制作 yaffs2 文件系统工具
（d）lzo-2.08.tar.gz	依赖工具
（e）e2fsprogs-1.41.14.tar.gz	依赖工具
（f）mtd-utils-1.5.1.tar.bz2	制作 ubifs 文件系统工具

（1）解压源码包并新建安装目录

```
$ cd Workstaion/tools/makefs
$ tar zxvf install-makefstool.tar.gz
$ mkdir install
$ mkdir install/mtd
```

（2）安装依赖工具 zlib

```
$ tar zxvf zlib-1.2.8.tar.gz
$ cd zlib-1.2.8
$ ./configure
$ make
$ sudo make install
$ cd ..
```

（3）安装制作 cramfs 文件系统镜像文件工具 mkcramfs

提示：也可使用 Ubuntu 系统自带的制作 cramfs 文件系统工具 mkfs.cramfs

```
$ tar zxvf cramfs-1.1.tar.gz
$ cd cramfs-1.1
$ make
```

在当前目录下生成 mkcramfs，将其拷贝到/usr/bin 目录下。

```
$ sudo cp mkcramfs /usr/bin
$ cd ..
```

（4）安装制作 yaffs2 文件系统镜像文件工具 mkyaffs2image

```
$ tar zxvf yaffs2-d43e901.tar.gz
$ cd yaffs2-d43e901/utils
$ make
```

在当前目录下生成 mkyaffs2image，将其拷贝到/usr/bin 目录下。

```
$ sudo cp mkyaffs2image /usr/bin
$ cd ../..
```

（5）安装依赖工具 lzo

```
$ tar zxvf lzo-2.08.tar.gz
$ cd lzo-2.08
$ ./configure --build=i686-pc-linux
--prefix=/home/loongson/Workstation/tools/makefs/install
```

```
$ make
$ make install
$ cd ..
```

（6）安装依赖工具 e2fsprogs

```
$ tar zxvf e2fsprogs-1.41.14.tar.gz
$ cd e2fsprogs-1.41.14
$ ./configure --build=i686-pc-linux
--prefix=/home/loongson/Workstation/tools/makefs/install
$ make
$ make install
$ cd lib/uuid
$ make install
$ cd ../../..
```

（7）安装 ubifs 文件系统镜像文件制作工具

```
$ tar jxvf mtd-utils-1.5.1.tar.bz2
$ cd mtd-utils-1.5.1
$ vi Makefile
```

修改 Makefile，在版本说明“VERSION = 1.5.1”一行后面添加如下内容：

```
PREFIX = /home/loongson/Workstation/tools/makefs/install/mtd
DEPEND = /home/loongson/Workstation/tools/makefs/install
ZLIBCPPFLAGS = -I/usr/local/include
ZLIBLDFLAGS = -L/usr/local/lib
LZOCPPFLAGS = -I$(DEPEND)/include
LZOLDFLAGS = -L$(DEPEND)/lib
LDFLAGS += $(ZLIBLDFLAGS) $(LZOLDFLAGS)
CFLAGS ?= -O2 -g $(ZLIBCPPFLAGS) $(LZOCPPFLAGS)
```

```
$ vi common.mk
```

修改 common.mk，找到并注释掉“PREFIX=/usr”一行，如下：

```
#PREFIX=/usr
```

然后编译安装。

```
$ WITHOUT_XATTR=1 make
$ make install
$ cd ..
```

将在安装目录 install/mtd/sbin 下生成的可执行文件 mkfs.ubifs 和 ubinize 拷贝到/usr/bin 目录下。

```
$ sudo cp install/mtd/sbin/mkfs.ubifs /usr/bin
$ sudo cp install/mtd/sbin/ubinize /usr/bin
```

2.9.2 制作根文件系统镜像文件

使用安装好镜像文件制作工具来制作在“7.4 构建根文件系统”中完成构建的根文件系统目录“rootfs”的镜像文件。

1 cramfs 文件系统

```
$ mkcramfs rootfs/ rootfs-cramfs.img
$ chmod +r rootfs-cramfs.img                    //添加可读权限，避免出现无法烧写的情况
```

或使用系统自带的工具

```
$ mkfs.cramfs rootfs/ rootfs-cramfs.img
$ chmod +r rootfs-cramfs.img                    //添加可读权限，避免出现无法烧写的情况
```

2 yaffs2 文件系统

```
$ mkyaffs2image rootfs/ rootfs-yaffs2.img
$ chmod +r rootfs-yaffs2.img                    //添加可读权限，避免出现无法烧写的情况
```

3 ubifs 文件系统

```
$ vi ubifs-2KB.cfg
```

新建配置文件 ubifs-2KB.cfg，添加如下内容：

```
[ubifs]
mode=ubi
image=ubifs.img
vol_id=0
vol_size=45MiB
vol_type=dynamic
vol_alignment=1
vol_name=rootfs
vol_flags=autoresize
```

制作 ubifs 文件系统镜像文件。

1C300B 开发板使用 2K 页大小的 Nand Flash 芯片，文件系统分区大小为 50M。

```
$ mkfs.ubifs -r rootfs/ -m 2048 -e 126976 -c 370 -o ubifs.img
```

```
$ ubinize -o ubifs-2KB.img -m 2048 -p 131072 -s 2048 ubinize-2KB.cfg
```

最终生成用于烧写的 ubifs 文件系统 ubifs-2KB.img。

提示：配置文件中的 vol_size 的大小不能大于文件系统分区的大小，vol_size 的值约等于-e 参数（逻辑擦除块大小）乘以-c 参数（最大逻辑擦除块数）。

烧写 ubifs 文件系统后启动系统，内核中关于 ubi 的打印信息如下：

提示：内核配置 UBIFS 的支持请参考“6.5.3 配置 UBIFS 支持”。

```
UBI: attaching mtd2 to ubi0
UBI: physical eraseblock size:   131072 bytes (128 KiB)
UBI: logical eraseblock size:    126976 bytes
UBI: smallest flash I/O unit:    2048
UBI: sub-page size:              512
UBI: VID header offset:          2048 (aligned 2048)
UBI: data offset:                4096
UBI: max. sequence number:       0
UBI: volume 0 ("rootfs") re-sized from 372 to 392 LEBs
UBI: attached mtd2 to ubi0
UBI: MTD device name:            "rootfs"
UBI: MTD device size:            50 MiB
UBI: number of good PEBs:        400
UBI: number of bad PEBs:         0
UBI: number of corrupted PEBs:   0
UBI: max. allowed volumes:       128
UBI: wear-leveling threshold:    4096
UBI: number of internal volumes: 1
UBI: number of user volumes:     1
UBI: available PEBs:             0
UBI: total number of reserved PEBs: 400
UBI: number of PEBs reserved for bad PEB handling: 4
UBI: max/mean erase counter: 1/0
UBI: image sequence number:  109206856
UBI: background thread "ubi_bgt0d" started, PID 32
```

2.10 基于 linux 的网络配置

嵌入式 Linux 配置 DHCP 自动获取 IP 地址和域名解析

step1 配置内核：

```
[*] Networking support --->
Networking options --->
<*> Packet socket
<*> Unix domain sockets
[*] TCP/IP networking
[*] IP: kernel level autoconfiguration
```

```
[*] IP: DHCP support
[*] Network packet filtering framework (Netfilter) --->
```

step2. 配置 busybox:

```
Networking Utilities --->
[*] udhcp client (udhcpc)
```

step3.建立配置文件：

从 busybox 的 examples/udhcp/下 copy simple.script 文件到开发板/usr/share/udhcpc/下，并重命名为 default.script

step4. 测试：

在命令台执行 udhcpc，注意：必须确保局域网内存在 DHCP 服务器，否则 udhcp 执行失败。或者在开机脚本“/etc/rc.d/rc.sysinit”中添加“udhcpc -b”，测试结果如下

```
devtmpfs: mounted
Freeing unused kernel memory: 212k freed
#mount all.....
#Starting mdev.....
udhcpc (v1.19.2) started
Setting IP address 0.0.0.0 on eth0
Sending discover...
PHY: 0:13 - Link is Up - 100/Full
Sending discover...
Sending select for 192.168.0.4...
Lease of 192.168.0.4 obtained, lease time 86400
Setting IP address 192.168.0.4 on eth0
Deleting routers
route: SIOCDELRT: No such process
Adding router 192.168.0.1
Recreating /tmp/resolv.conf
 Adding DNS server 192.168.0.1
Processing /etc/profile.....
Done!
[LOONSON@Loongson-gz:/]#ifconfig eth0
eth0          Link encap:Ethernet  HWaddr 00:55:7B:B5:7D:F7
              inet addr:192.168.0.4  Bcast:192.168.0.255
Mask:255.255.255.0
              UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
              RX packets:3 errors:0 dropped:0 overruns:0 frame:0
              TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:1240 (1.2 KiB)  TX bytes:978 (978.0 B)
              Interrupt:35
```

```
[LOONSON@Loongson-gz:/]#ping 192.168.0.3
PING 192.168.0.3 (192.168.0.3): 56 data bytes
64 bytes from 192.168.0.3: seq=0 ttl=64 time=5.406 ms
64 bytes from 192.168.0.3: seq=1 ttl=64 time=0.910 ms
64 bytes from 192.168.0.3: seq=2 ttl=64 time=1.039 ms
64 bytes from 192.168.0.3: seq=3 ttl=64 time=0.937 ms
^C
--- 192.168.0.3 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.910/2.073/5.406 ms
[LOONSON@Loongson-gz:/]#
```

step6. 域名解析

busybox 中要使用 DNS 时，要保证/etc/nsswitch.conf, [/etc/resolv.conf, /etc/hosts, /etc/host.conf,/etc/hostname]文件存在（其中[]内的内容是否需要没有经过验证，反正有了肯定不会错，呵呵）；同时要求 libnss_dns,libnss_files,libresolv 库存在。文件/etc/nsswitch.conf 中 hosts 这一行一定要加上 dns，不然即使有 resolv.conf 文件，域名也解析不了共享库支持：如果你的 busybox 为动态编译，则需要/lib/libc.so.6

不论是动态编译还是静态，如果要做 DNS 解析，就一定需要

```
/lib/libnss_dns.so.2
/lib/libnss_files.so.2
/lib/libresolv.so.2
/etc/resolv.conf
```

resolv.conf 根据具体环境修改。同时，只要使用了共享库，就需要

```
/lib/ld-linux.so.2
/sbin/ldconfig
/etc/ld.so.conf
```

ld.so.conf 根据具体环境修改。

注意：我这里给出的共享库文件名可能只是符号连接，一定要同时拷贝原始文件。

```
[root@localhost LS1xrootfs-demo]# find /opt/gcc-4.3-ls232/ -name libnss_dns*
/opt/gcc-4.3-ls232/sysroot/lib/libnss_dns.so.2
/opt/gcc-4.3-ls232/sysroot/lib/libnss_dns-2.7.so
/opt/gcc-4.3-ls232/sysroot/usr/lib/libnss_dns.so
[root@localhost LS1xrootfs-demo]# cp -a /opt/gcc-4.3-ls232/sysroot/lib/libnss_dns* lib/
```

没有拷贝/opt/gcc-4.3-ls232/sysroot/usr/lib/libnss_dns.so

如果此时还不能 ping 域名，那么自己重新动态编译 busybox，然后拷贝库文件，做个文件系统就可以了。

测试结果如下

```
Freeing unused kernel memory: 212k freed
#mount all.....
#Starting mdev.....
udhcpc (v1.19.2) started
```

```
Setting IP address 0.0.0.0 on eth0
Sending discover...
PHY: 0:13 - Link is Up - 100/Full
Sending discover...
Sending select for 192.168.0.4...
Lease of 192.168.0.4 obtained, lease time 86400
Setting IP address 192.168.0.4 on eth0
Deleting routers
route: SIOCDELRT: No such process
Adding router 192.168.0.1
Recreating /tmp/resolv.conf
 Adding DNS server 192.168.0.1
Processing /etc/profile.....
Done!
[LOONSON@Loongson-gz:/]#ping www.baidu.com
PING www.baidu.com (119.75.218.70): 56 data bytes
64 bytes from 119.75.218.70: seq=0 ttl=54 time=59.815 ms
64 bytes from 119.75.218.70: seq=1 ttl=54 time=61.512 ms
64 bytes from 119.75.218.70: seq=2 ttl=54 time=61.185 ms
^C
--- www.baidu.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 59.815/60.837/61.512 ms
[LOONSON@Loongson-gz:/]#
```

2.11 基于 linux 的交叉编译 Helloworld

交叉编译需要的环境：linux + GCC。

linux：龙芯官方教程使用的是 ubuntu 的长期版本 12.04。

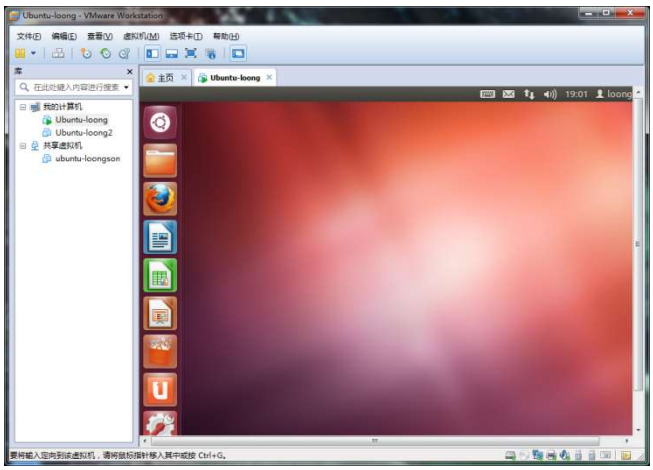
GCC：龙芯优化的 GCC 编译器，文件名：gcc-4.3-ls232-static.tar.gz。

ubuntu 的安装。方案 1:在电脑上装双系统（windows+ubuntu）；方案 2:在 windows 上安装虚拟机，在虚拟机上安装 ubuntu。

我选择方案 2。（具体安装略）

在 windows 7 上安装 VMware 虚拟机，再在虚拟机上安装 ubuntu。龙芯的 gcc 编译器包直接在 ubuntu 上解压，找到 bin 文件夹，下面有 mipsel-linux-gcc 的 bin 文件，就是我们要找的命令。用该命令编译一个 hello.c 的文件。

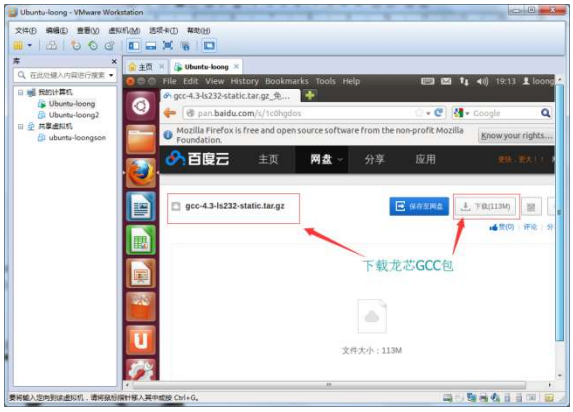
step1.打开虚拟机加载 ubuntu。



step2.在 ubuntu 中下载 GCC 编译包。

http://pan.baidu.com/s/1eQ34rb8

下载的默认路径：~/下载



step3.解压下载的 gcc-4.3-ls232-static.tar.gz。

$ cd ~

$ ls

$ cd 下载

$ ls

```
loong@loong-virtual-machine:~$ cd ~
loong@loong-virtual-machine:~$ ls
examples.desktop  公共的  模板  视频  图片  文档  下载  音乐  桌面
loong@loong-virtual-machine:~$ cd 下载
loong@loong-virtual-machine:~/下载$ ls
gcc-4.3-ls232-static.tar.gz
```

在当前文件夹解压

$ sudo tar -zxvf gcc-4.3-ls232-static.tar.gz

```
opt/gcc-4.3-ls232/libexec/gcc/mipsel-linux/4.3.6/f951
opt/gcc-4.3-ls232/libexec/gcc/mipsel-linux/4.3.6/install-tools/
opt/gcc-4.3-ls232/libexec/gcc/mipsel-linux/4.3.6/install-tools/fixincl
opt/gcc-4.3-ls232/libexec/gcc/mipsel-linux/4.3.6/install-tools/fixinc.sh
opt/gcc-4.3-ls232/libexec/gcc/mipsel-linux/4.3.6/install-tools/mkheaders
opt/gcc-4.3-ls232/libexec/gcc/mipsel-linux/4.3.6/install-tools/mkinstalldirs
loong@loong-virtual-machine:~/下载$ ls
gcc-4.3-ls232-static.tar.gz  opt
loong@loong-virtual-machine:~/下载$
```

解压完成后得到一个 opt 文件夹

step4.写一个 hello.c 的 c 语言文件。

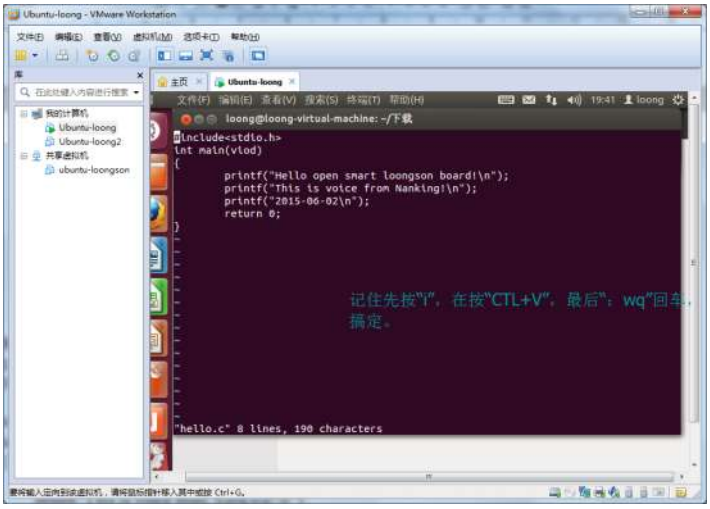
$ sudo vi hello.c

将会打开 vi 编辑器，我们写个 c 文件。

拷贝一下内容，粘贴到 vi 中。

```
#include<stdio.h>
int main(viod)
{
  printf("Hello open smart loongson board!\n");
  printf("This is voice from Nanking!\n");
  printf("2015-06-02\n");
  return 0;
}
```

粘贴步骤：step1:“i” step2:“ CTL+V” step3:“:wq”(冒号与 wq 之间没有空格) step4：“回车”



step5.找到龙芯的编译器命令 mipsel-linux-gcc。这是二进制的文件，可以直接执行。

$ cd opt

$ cd gcc-4.3-ls232/

$ cd bin

$ ls

查看一下编译器的版本

$ ./mipsel-linux-gcc -v

step6.编译 hello.c

$ cd ~/下载/

$ ls

$ sudo ./opt/gcc-4.3-ls232/bin/mipsel-linux-gcc -o hello hello.c

生成绿色的 hello，这就是在开源龙芯板子上直接运行的二进制程序了。

setp7.把绿色 hello 从虚拟机弄出来。这一步超级难。

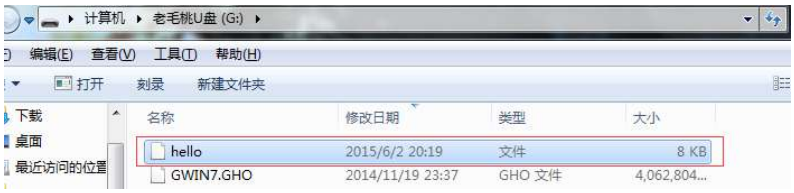
2 楼给的建议：“提点建议，setp7 把文件从虚拟机拿出来可以通过 vmware tools,安装好后支持直接拖放。安自行百度，貌似挺简单”

在龙芯官方教程中，用到了虚拟机的共享文件夹功能。我在实际使用中发现了有兼容的问题。CSDN 上有解决方案，但是有时候不管用。

所以建议大家把 hello 文件传到邮箱或百度网盘里面(大家各显神通吧)，然后在 windows 的环境下下载。

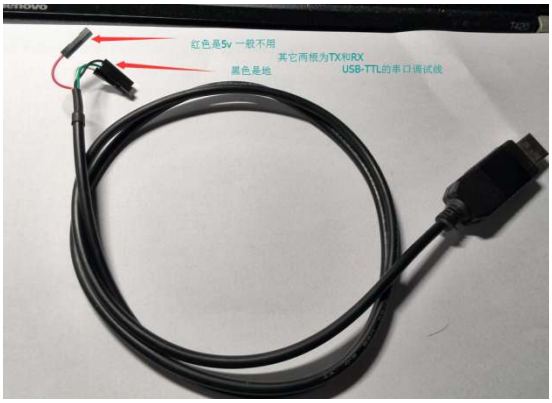
我编译的 hello 文件：http://pan.baidu.com/s/1dDAQocL

step8.在 windows 下，将 hello 文件拷到 U 盘中。

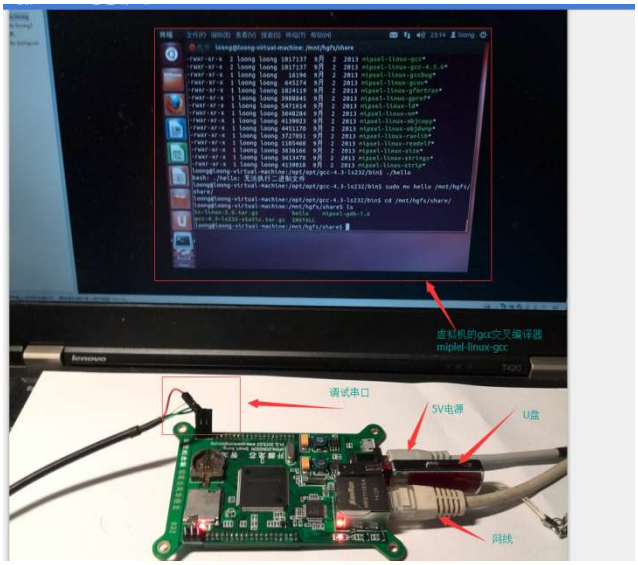


step9.将 U 盘查到龙芯开发板上。串口调试线龙芯开发板。

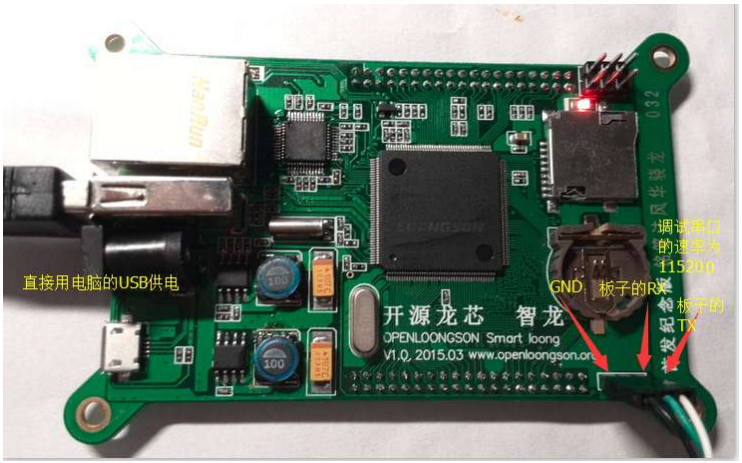
串口调试线：



硬件链接



串口线具体连接如下：



step10.利用串口调试，进入龙芯的 linux 系统（debian）。

在设备管理器里面可以找到具体的串口号，我的是 com7。

串口软件 secureCRT（安装略），波特率是 115200bps，（注意流控制 RTS/CTS 等 3 个选项不选，感谢 Ratking 提醒）参数设置如下图：

在龙芯的 debian 系统中

#ls

#cd mnt

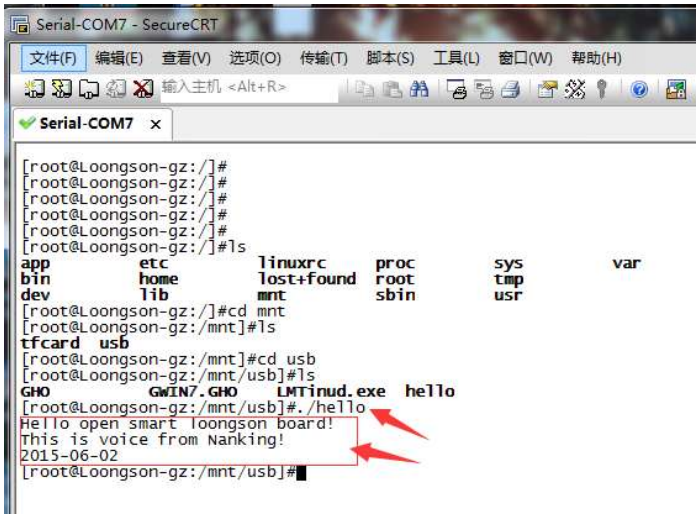
#cd usb

#ls

#./hello

运行结果是打印了三句话：

Hello open smart loongson board!
This is voice from Nanking!
2015-06-02



2.12 基于 linux 的 Python 移植

龙芯 1.0 板子的 2.0 针距母口实在是没法接我现有的各种外设。只能玩玩了大多数树莓派程序都是 python 的，有了 python 应该可以直接移植一大堆东西了。网上大多数 python 交叉编译的教程不太适合最新版本。我改了一下做了一个能用的编译脚本

下载包： make_mips_python.zip (491 Bytes)

适合最新的 2.7.10 版，应该也能用在最新的 3.6.x 上。python2.7.10 下载地址：
https://www.python.org/ftp/python/2.7.10/Python-2.7.10.tar.xz

step1 编译

先检查所需的库，直接跳到 b 开始编译也能成功。但是运行代码的时候可能缺这少那运行时如果提示缺少_collections 或者其他库，需要修改源代码中
Python2.7.10/Modules/Setup.dist 文件
找到对应的库然后取消注释
其中 shared 最好编译，其他的看需求打开一些。

```
array arraymodule.c        # array objects
cmath cmathmodule.c _math.c # -lm # complex math library functions
math mathmodule.c _math.c # -lm # math library functions, e.g. sin()
_struct _struct.c        # binary structure packing/unpacking
time timemodule.c # -lm # time operations and variables
operator operator.c        # operator.add() and similar goodies
#_testcapi _testcapimodule.c    # Python C API test module 要关闭否则编译不过
_random _randommodule.c        # Random number generator
_collections _collectionsmodule.c # Container types
_heapq _heapqmodule.c                # Heapq type
```

```
itertools itertoolsmodule.c        # Functions creating iterators for efficien
t looping
strop stropmodule.c                # String manipulations
_functools _functoolsmodule.c        # Tools for working with functions and ca
llable objects
_elementtree -I$(srcdir)/Modules/expat -DHAVE_EXPAT_CONFIG_H -DUSE_PYEXPAT_CAP
I _elementtree.c # elementtree accelerator
#_pickle _pickle.c        # pickle accelerator
datetime datetimemodule.c        # date/time type
_bisect _bisectmodule.c        # Bisection algorithms
```

解压以后把 make_mips_python.shell 拷贝进去执行，需要跑几分钟最后会有一个 error，不用管生成的 python 在 ./mips_python/install/bin 和 lib 都要拷贝进开发板 mkdir /usr/libmkdir /usr/bin。

step2 安装

```
cp /mnt/usb/mips_python/bin/python2.7 /usr/bin/
cp -r /mnt/usb/mips_python/lib/python2.7 /usr/lib/
export PYTHONHOME=/usr/
再创建文件链接 ln -s /usr/bin/python2.7 /usr/bin/python
```

增加 python 库的数量，还有静态编译，需要继续大家一起努力。如果需要像 pi 一样可以通过 python 控制 GPIO。看一下 Rasberry Pi 的库按着写一个，基础库都抄过来以后，基本上就能无缝直接用树莓派的代码了。

2.13 基于 linux 的 PWM 控制 LED

用户空间驱动访问硬件设备--龙芯 LS1C PWM 控制 LED。例子是一个 PWM 的测试程序，PWM，即脉冲宽度调制，可用于四驱车、机器人舵机控制。

step1. 封装

根据龙芯 1C 的数据手册中,LS1C0300A 封装有三种方式：

QFP100 封装 :100 表示 100 个引脚

QFP176A 封装 :176 表示 176 个引脚,A 表示 ADC

QFP176U 封装 :176 表示 176 个引脚,U 表示 UART

QFP176A 和 QFP176U 的引脚个数一样,区别在于 81~88 引脚不同.

智龙开发板中的 1C 有 176 个脚,所以是 QFP176 封装。图中 81~88 都是 ADC,因此智龙开发板的 LS1C 是 QFP176A 封装.

step2.PWM 引脚复用

根据数据手册， PWM0 和 PWM1 可直接使用；PWM2 和 PWM3 未引出,需和其它复用。

表 PWM 引脚复用

GPIO	默认功能	第四复用
47	CAMCLKOUT	**PWM3**
46	CAMPCLKIN	**PWM2**
92	**PWM1**	
6	**PWM0** CAMCLKOUT	

Step3 openloongson-kernel 源码分析，选 PWM1 为测试口。

```
../arch/mips/loongson/ls1x/pwm.c 中
...
#if defined(CONFIG_LS1A_MACH)
...
#elif defined(CONFIG_LS1C_MACH)
#define LS_GPIO_PWM0    06
#define LS_GPIO_PWM1    92
#define LS_GPIO_PWM2    46
#define LS_GPIO_PWM3    47
#endif
...
        /* 如果该引脚被设置为 gpio 需要释放该引脚 */
...
    /* 设备复用模式为 pwm */
#if defined(CONFIG_LS1A_MACH)
...
#elif defined(CONFIG_LS1B_MACH)
...
#elif defined(CONFIG_LS1C_MACH)

#endif
...
```

从源码可见 LS1A、LS1B 有设复用,而 LS1C 没有。根据智龙主板原理图，可知下方第二行第 16 处有标注 PWM1/GPIO92.所以推断 PWM1 对应 GPIO92。因此挑选 PWM1 作为测试,针脚应插到插槽 GPIO92。

step3. 测试 PWM1

表 PWM1 的配置寄存器

寄存器	地址	说明
CNTR	0xBFE5C010	主计数器
HRC	0xBFE5C014	高脉冲定时参考寄存器
LRC	0xBFE5C018	低脉冲定时参考寄存器
CTRL	0xBFE5C01C	控制寄存器

2)<openloongson-kernel 源码>

```
../arch/mips/include/asm/mach-loongson/ls1x/loongson1.h
...
/* pwm regs */
...
#define LS1X_PWM1_BASE    0x1fe5c010  //此处应是物理地址
...
../arch/mips/include/asm/mach-loongson/ls1x/regs-gpio.h
...
#ifdef CONFIG_LS1C_MACH
/* GPIO 64-95 */
#define LS1X_GPIO_CFG2        ((void __iomem *)0xbfd010c8)//LS1X_GPIO_REG(
0x8)  //此处应是程序空间地址
...
#endif
...


../arch/mips/loongson/ls1x/pwm.c 中
...
#define REG_PWM_CNTR    0x00
#define REG_PWM_HRC    0x04
#define REG_PWM_LRC    0x08
#define REG_PWM_CTRL    0x0c
...
        ls1x_pwm_base = ioremap(LS1X_PWM0_BASE, 0xf);//我不懂 ioremap 的作用,
估计是将物理空间地址转换为程序空间地址?
...
        /* 设置占空比 */
    writel(duty, ls1x_pwm_base + (id << 4) + REG_PWM_HRC);   //内核应该是对
程序空间地址进行读写
    writel(period, ls1x_pwm_base + (id << 4) + REG_PWM_LRC);
...
```

step4 CPU 运 行 态

1)

龙芯的地址空间布局有个 kseg1 段:0xA000 0000 ~ 0xBFFF FFFF(512M),各个硬件设备 I/O 寄存器便落在 kseg1 内,而 kseg1 这些地址通过把最高三位清零的方法来映射到相应的物理地址上.

如 PWM1 主计数器地址是 0xBFE5C010,<LS1C 用户手册>没说明上述地址是程序空间地址还是物理空间地址.
但地址 0xBFE5C010 落在 kseg1 范围内,且 0xBFE5C010 最高三位清零便是 0x1FE5C010(见上面 loongson1.h 的 LS1X_PWM1_BASE).
所以<LS1C 用户手册>里的地址是指程序空间地址

2)
龙芯的 CPU 运行有 3 个态：用户模式、管理模式、核心模式 。kseg1 只能在核心模式下访问，因此<openloongson-kernel 源码>的内核就直接使用程序空间地址。
3)
本例子程序是用户态驱动(用户空间)的例子,本人不懂写内核态的驱动程序,所以本文便是为不会编写内核驱动的读者而写.

用户空间可以直接通过打开 /dev/mem 设备文件，然后 mmap()映射进行访问真实的物理地址.

mmap 函数原型：
void *mmap(void *start, size_t length , int prot , int flags , int fd , off_t offset)

其 offset 在/dev/mem 特指物理空间的地址,且其大小必须满足页大小的整数倍.

获取内存页大小的命令是 getconf PAGE_SIZE,C 函数是 getpagesize(),一般的输出是 4096,即 4K 字节.

openloongson 的页大小是 4096=0x1000
PWM1 第一个寄存器地址是 0xBFE5C010(物理地址 0x1FE5C010),最后一个寄存器地址是 0xBFE5C01C(物理地址 0x1FE5C01C),每个寄存器 32 位即 4 字节.
0x1FE5C010 取模 0x1000=0x10=16
0x1FE5C010-0x10=0x1FE5C000

因此本例中 offset 取值 0x1FE5C000;length 取值 0x1FE5C01C-0x1FE5C000+4=32

step5 LED 驱动程序

```
1  //pwmled.c
2  #include <sys/types.h>
3  #include <stdio.h>
4  #include <sys/ioctl.h>
5  #include <sys/mman.h>
6  #include <fcntl.h>
7  #include <memory.h>
8
```

```
#define pwm_base 0x1FE5C010  //PWM1
// 控制寄存器相对于基址的偏移
#define REG_PWM_CNTR 0x00
#define REG_PWM_HRC  0x04
#define REG_PWM_LRC  0x08
#define REG_PWM_CTRL 0x0c

int main(int argc ,char *argv[])
{ int fd;
  void *map_base;

  fd = open("/dev/mem", O_RDWR|O_SYNC);
  if(fd == -1)
  { printf("Can't open /dev/mem\n");
    return -1;
  }
  printf("open /dev/mem success\n");

  int psize=getpagesize();
  int rem=pwm_base%psize;
  int offset=pwm_base-rem;
  int length=pwm_base+REG_PWM_CTRL-offset+4;
  printf("pagesize:%i ; pwm1 base:%x ; remainder:%i ; offset:%x ; length:%i\n"
,psize,pwm_base,rem,offset,length);

  map_base=(volatile unsigned int*)mmap(NULL,length,PROT_READ|PROT_WRITE,MAP_S
HARED,fd,offset);

  if(map_base==NULL || map_base ==MAP_FAILED)
    printf("Can't mmap\n");
  else
    printf("mmap success\n");

  //--v-- 启动 pwm
  *(volatile unsigned int*)(map_base+rem+REG_PWM_CNTR)= 0;
  *(volatile unsigned int*)(map_base+rem+REG_PWM_CTRL)= 1;
  //--^--

  //--v-- 设置占空比
  unsigned int hrc_val, lrc_val;
  unsigned int data[2] = {0x7ffffff,0x7ffffff};//占空比 0.5

  hrc_val = data[1] - 1;
  lrc_val = data[0] + data[1] -1;
```

```
53
54    printf("hrc:%i ; lrc:%i\n",hrc_val,lrc_val);
55
56    *(volatile unsigned int*)(map_base+rem+REG_PWM_HRC)= hrc_val;
57    *(volatile unsigned int*)(map_base+rem+REG_PWM_LRC)= lrc_val;
58    //--^--
59
60    close(fd);
61    munmap((void*)map_base,length);
62    return 0;
  }
```

step6 编译环境：龙芯 2F 笔记本逸珑 8089,debian 9,o32 编译为静态可执行文件

loongson@debian:~$ gcc -static -o pwmled pwmled.c

step7 运行

准备一发光二极管 LED,负极接地(下方第二行第 1 处标注 DGND) ,正极接 PWM1(下方第二行第 16 处标注 PWM1/GPIO92).有说 LED 需串接一电阻,我问同事电子工程师,说不用串接电阻也没问题，这一点我无法保证会不会弄坏开发板，大家按自己实际情况

程序复制到 TF 卡(FAT32,在我的 TF 卡是第二个分区,请按你自己实际/dev/mmcblk 挂载)

```
1  [root@Loongson-gz:/]#mount /dev/mmcblk0p2 /mnt/tfcard
2  [root@Loongson-gz:/]#cd /mnt/tfcard
3  [root@Loongson-gz:/mnt/tfcard]#ls
4  pwmled
5
6  [root@Loongson-gz:/mnt/tfcard]#./pwmled
7  open /dev/mem success
8  Illegal instruction
9  [root@Loongson-gz:/mnt/tfcard]#./pwmled
10 open /dev/mem success
11 pagesize:4096 ; pwm1 base:1fe5c010 ; remainder:16 ; offset:1fe5c000 ; length:3
12 2
13 mmap success
14 hrc:134217726 ; lrc:268435453
15 [root@Loongson-gz:/mnt/tfcard]#
```

第一次运行。/pwmled ，出现“非法指令”。再次运行“./pwmled” ，LED 已经一闪一闪。

本例只启动 PWM，没停止 PWM。所以“./pwmled”结束，PWM 仍一直输出波形，LED 一闪一闪。

[root@Loongson-gz:/mnt/tfcard]#poweroff

关闭系统后，电源不拔，LED 仍一闪一闪。只要有电，配好参数的 PWM 硬件模块就能独立于 CPU 运行。

2.14 RT-Thread 实时系统移植

教程 for smart loong： http://pan.baidu.com/s/1bnyQa8B

2.14 基于 RTT 编写 PWM 驱动

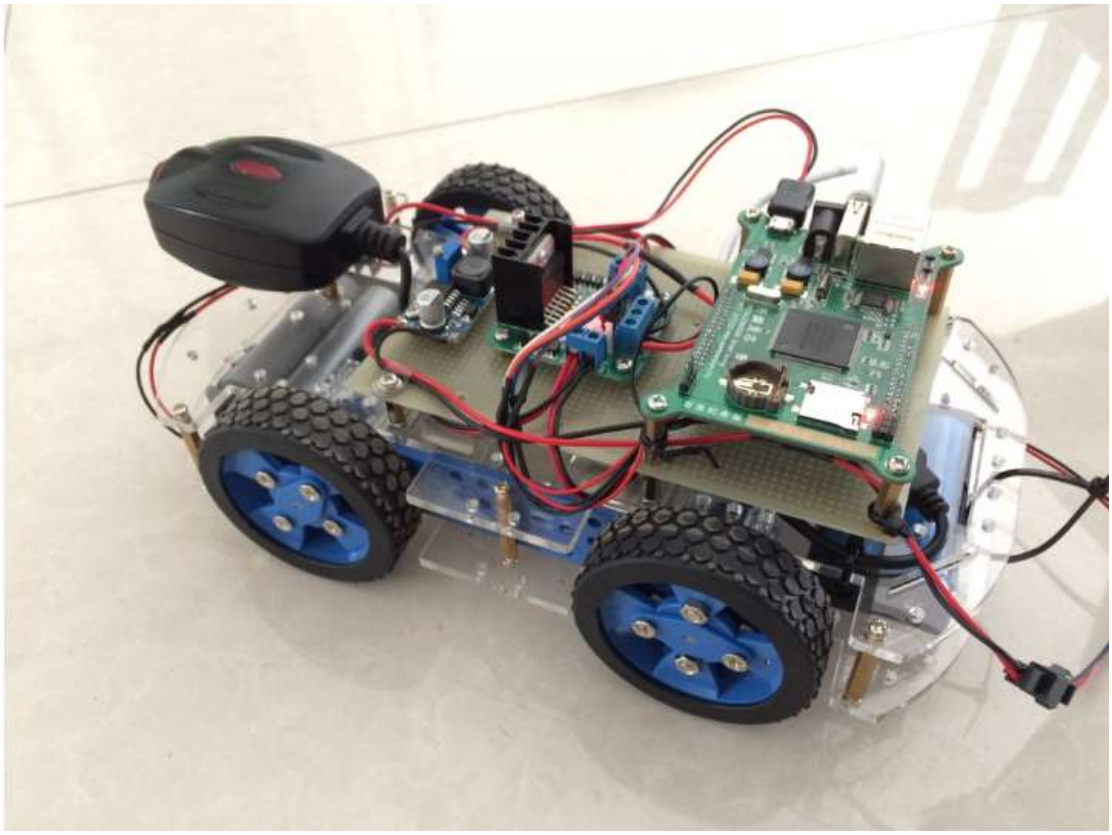
教程 for smart loong： http://pan.baidu.com/s/1bnyQa8B

2.15 基于 RTT 的 LED 和按键的控制

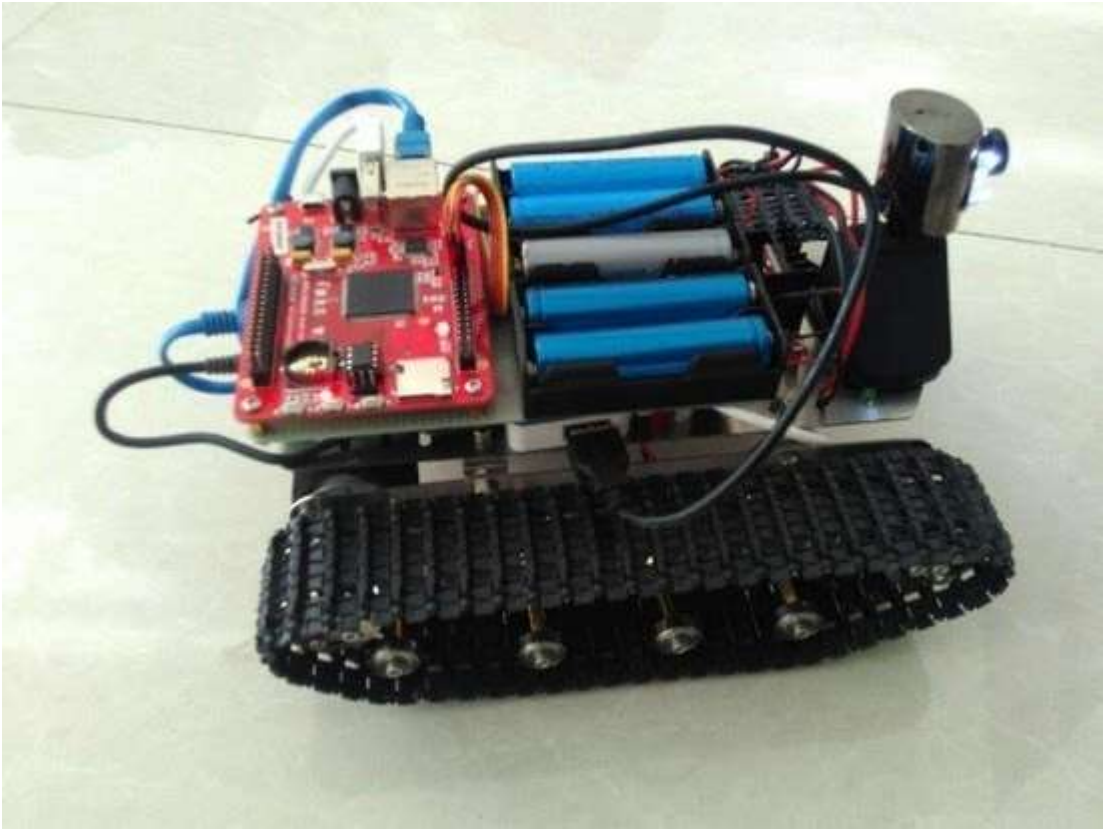
教程 for smart loong： http://pan.baidu.com/s/1bnyQa8B

3 应用篇

3.1 龙芯 wifi 小车



原文网址：
http://www.openloongson.org/forum.php?mod=viewthread&tid=138&extra=page%3D1



奇点极客工作室 智龙 V2 铁甲神探 WIFI 遥控小车

智龙 V2 主板一发布，奇点极客工作室就在淘宝上推出了基于智龙 V2 的履带式 WIFI 小车，可通过智能手机遥控。

3.2 英国智龙摩尔电码播放器

我其实很早就知道我国有龙芯这么一款国产处理器，但对其的印象一直停留在“安全应用”、“试验品”等字眼上，认为一个普通人并不会接触到龙芯系列的处理器；尤其是在得知龙芯所使用的 MIPS 架构无法兼容 Windows 之后，便更加认为龙芯不会在桌面系统上有所成就——直到今年三月份在某新闻客户端上看到了一则新闻，内容大约是“中国的树莓派 开源龙芯主板开始接受团购”，发现其价格为 199 元并不贵，同时由于对国产 CPU 的好奇，便拍下了这款产品。在经过了漫长的等待——五月份发货，以及我八月份才回国过暑假——之后，终

于拿到了这块板子。当然由于这块板子，我也通过社区了解到了龙芯产品的最新动向，比如自主指令集，以及 3A2000 和 3B2000 CPU。这些新产品受到了媒体的广泛关注，龙芯中科也因此被人民日报头版所报道，在此便不再累述。

走出实验室 迈步闯市场

龙芯中科：让“中国芯”奔腾起来

本报北京8月30日电 （记者**赵永新**）从追求高水平论文到注重满足用户需求，从追求单一性能指标到搭建产业生态系统，龙芯中科技术有限公司（以下简称龙芯中科）面向市场搞研发、加快成果产业化，基于龙芯CPU（中央处理器）的信息产业生态系统已初步形成。

CPU是信息产业的基础部件，如果一味依赖进口，信息产业就会受制于人。2001年，32岁的中科院计算所研究员胡伟武带领团队研制自主可控的“中国芯”，先后研制出龙芯1号、2号、3号等系列高性能CPU。令人遗憾的是，龙芯CPU一直未能实现大规模推广应用。2010年，龙芯总设计师胡伟武率领团队骨干脱离事业编制，离开中科院计算所创办龙芯中科，开始从学院派转向市场派，着力推进龙芯的产业化。

“与国外芯片相比，龙芯最大的差距不在于技术指标，而是未能与产业链对接、建立与之相匹配的生态系统。”胡伟武告诉记者，龙芯中科成立后，瞄准用户需求搞研发，与曙光、浪潮、锐捷网络、东软集团等国内中下游软硬件企业紧密合作，自主开发出高性能计算机、高端服务器、网络交换机、千兆防火墙等终端产品。目前，基于龙芯CPU进行下游解决方案开发的合作伙伴已有数百家，基于龙芯CPU的研发人员已经达到上万人规模，围绕龙芯的自主可控的信息产业生态圈已初步形成。“近年来龙芯中科在PC和服务器、智能移动终端、嵌入式应用等三大板块持续发力，2013年我们的芯片销量为1.8万片，去年卖了35万片，今年估计要比去年翻一番。”胡伟武说。

（相关报道见第二版）

行进中国·改革故事

人民日报 8 月 31 日所报道的龙芯中科

事实上，我此前并没有任何嵌入式开发的经验。在最开始拿到板子的时候，甚至不知道要通过串口线连接主板。但是通过开源社区中提供的教程，我很快便掌握了对主板的简单操作。在这里要感谢教程的作者 xieyug2012 以及 shigeng。

小作品——摩斯电码播放器

摩斯码使用不同长短的连续波（嘀和嗒）和间隙来表示 26 个英文字母，10 个数字以及一些标点符号，由于其编码简单、可靠性高，被广泛应用于无线电通讯领域。其简便性使我萌生了利用龙芯主板来将英文文本翻译为摩斯码，并通过扬声器以正弦波播放出来的想法。由于我并没有无线电执照及相应的设备，所以此作品仅限以声音形式播放而不能通过无线电发报，但进行改动，龙芯主板是完全有能力成为全自动发报机的。

摩斯电码表

硬件部分

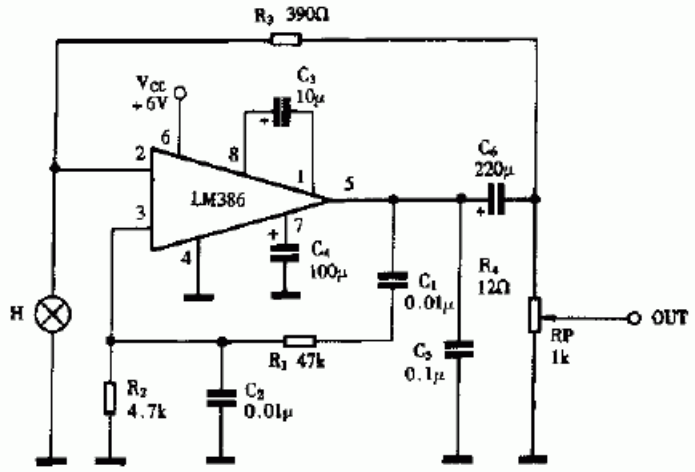
此作品电路板的基本系统结构为：使用运算放大

器芯片构造一个正弦波发生器，其频率可由一个双联电阻器在约 500Hz 至无限大（实际极限数值由芯片决定，但肯定超出人耳听力范围）之间调整，此正弦波将作为扬声器的声源。扬声器与正弦波之间有一个由开源龙芯主板 GPIO 接口所控制的继电器，用来控制线路的断连，从而达到播放摩斯码的目的。

确定了系统结构之后，下一步就要制作电路原理图并确定元件参数。毫无电路设计基础的我在网络上找到了这张原理图作为参考。

巧用 LM386 作正弦波振荡器

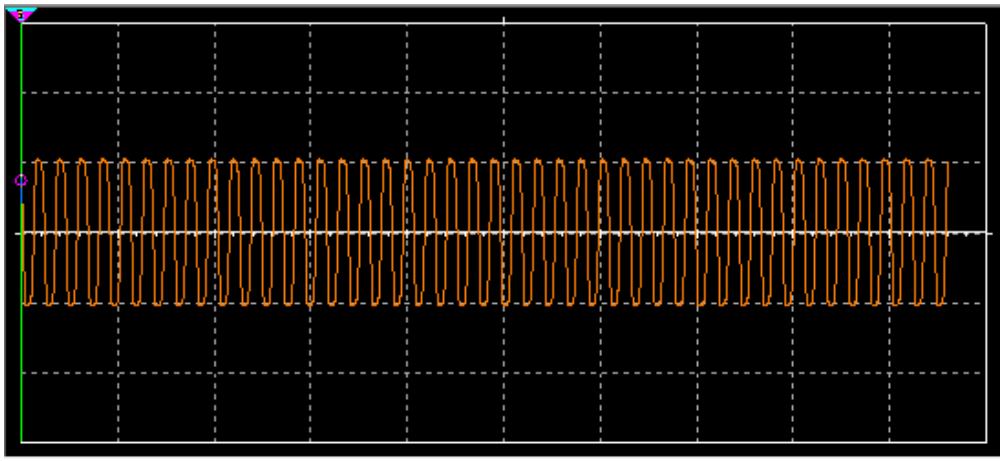
图　　是一个利用 LM386 制作而成的正弦波振荡器，电路采用文氏电桥振荡方式，输出信号的失真系数极低。小电珠 H 与电阻 R_3 组成负反馈电路，它使振荡器输出信号的幅值保



持稳定，而且具有较低的失真。当电容 C_1、C_2 取值相同时，电路振荡频率可由公式 $f = 1/2\pi C_1\sqrt{R_1R_2}$ 求得，采用图示数据时正弦波频率为 1kHz。在实际制作时，H 可选用 3V、15mA 的小电珠。

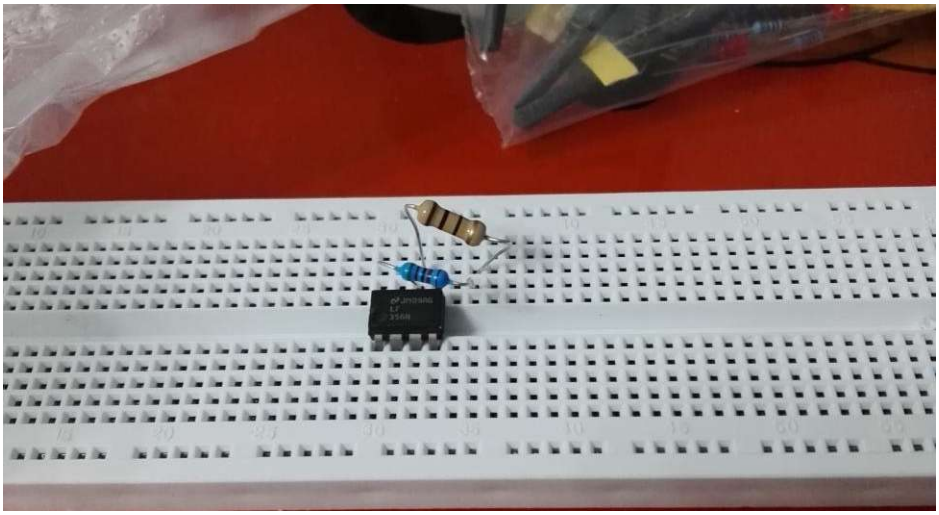
最初所找到的电路图

然而此电路在模拟器中的模拟并不成功。在 QQ 群中 electron 群友的帮助下，将运算放大器改为 LF356N，并对电路做出了大幅度简化及调整，确定了电路构造以及元件参数，并在软件中进行了电路模拟，成功输出了预定频率的正弦波。



电路模拟所得波形：每列表示 10 毫秒，每列有约 5 个波，可见其频率为 500Hz

我住在天津，作为一个大城市购买电子元件并不困难，我很快便购齐了所需的元件及面包板，开始真正的制作。

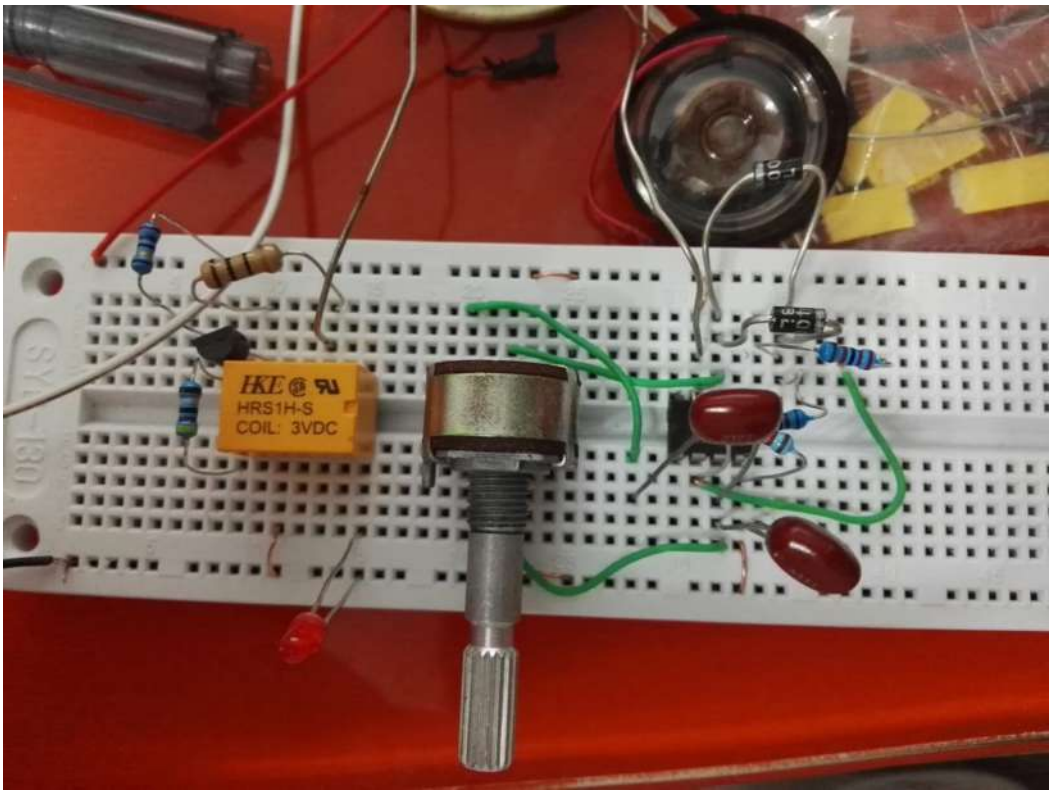


最初的三个元件

杂乱的环境

在实际制作中，我碰到了许多在模拟中没有预料到的问题。首先碰到的问题是 GPIO 无法直接触发继电器。通过查阅所使用的 HRS1H-S 型继电器的数据表，发现其线圈需要约 300mW 的功率来触发，然而龙芯 GPIO 的最大输出功率仅为约 50mW。在 QQ 群中群友的建议下，我使用了一颗 8050 三极管来提升功率，成功触发了继电器。

第二个问题则是吃了不认真和不懂电路原理的亏。当我把一切元件及线路按照电脑自动生成的鼠线连接之后，扬声器没有任何反应。这个问题困扰了我好几天，我一点一点地将面包板上的接线与电脑给出的鼠线比对，希望能发现什么接线错误，然而并没有。而这时我也该回到英国了，于是只能把所有的东西装到盒子里，到了英国再处理。

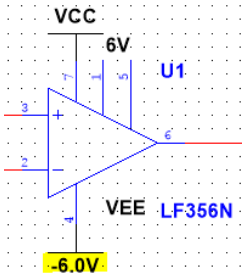


临走之前的情况

到了英国，我就有万用表能用了。通过测试发现，输出端是有直流电压输出的，而且与预期的结果一样，这说明电路并没有断路；但当测试交流电压时，万用表毫无反应，这说明输出的波形是一条直线，而不是正弦波。

既然电路板上没问题，那会不会是电路本身的设计问题？但如果电路设计有问题的话，仿真软件又是如何输出预期的波形的呢？我开始仔细检查起仿真软件

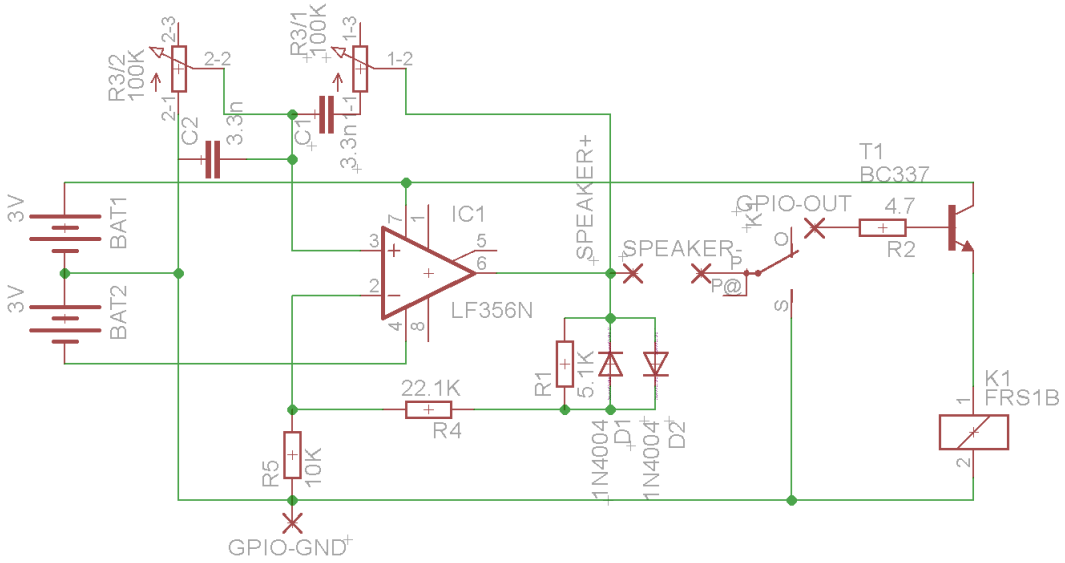
中的电路，最终发现了问题：



对于我这种不懂电路的人来说，这是个很不起眼的细节

问题就出在这个-6.0V 身上。此前我一直是把 4 号针脚接地的，并没有意识到这个负电压的问题。当我在仿真软件中也把 4 号针脚接地后，其输出的波形也变成了一条直线。

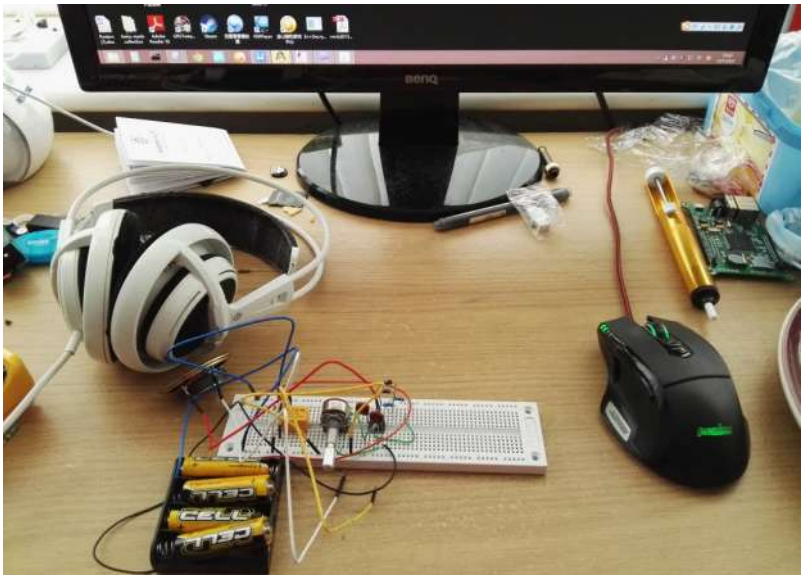
依然是在群友的帮助下，我对线路做了一点小改动，将 4*1.5V AA 电池盒的负极接到 4 号针脚而不是地，并把第二节电池的负极当成了地。由此便创造出了一个±3V 的直流电源，但这也导致了输出电压成了原来的一半，因此不再需要起初串联在扬声器上的限流电阻。



最终电路

以上即为最终的电路图。可见由 2 颗 3.3nF 电容和一个最大电阻为 100kΩ的双联电位器所组成的反馈回路。运放输出端直接连接扬声器正极，扬声器的回路则由继电器所控制。继电器的线圈一边接地，一边与一颗三极管的发射级连接，此三极管的集电极由电池驱动，基极则连接 GPIO 输出口和一个限流电阻。5.1kΩ电阻+2 个 1N4004 二极管并联的设计来自群友 electron，用来限制峰值电压。

最终，扬声器里成功输出了干净的正弦声波，通过调整双联电位器可将其频率调高至人耳听力以外。至于波形实际的噪音水平为多少，最高可达到的频率为多少，由于我并没有示波器，因此无从得知。



面包板上的成品

- **软件部分**

以上内容仅是成功的一半而已。有电路只能让喇叭发出正弦波，要想发送摩斯码，还要进行编程。我按照论坛上的教程在 Ubuntu 上搭建了交叉编译环境，所使用的语言是 C。我的 C 语言水平也是半斤八两，许多地方都靠现学现卖。

程序结构如下：主函数将输入的文本存入字符数组并传递给 morslize 函数，此函数将输入的字符数组遍历，并将每个字符翻译为一个代表此字符摩斯码的整形数组。比如字母 A 的摩斯码是嘀 嗒，那么它的数组就是{0,1}，字母 C 的摩斯码是嗒 嘀 嗒 嘀，那么它的数组就是{1,0,1,0}。总之，用 1 表示嗒，0 表示嘀。此数组再传递给 play 函数，用来控制 GPIO 口的电平高低及电平持续时间。控制 GPIO 口的函数及其头文件来自论坛网友 shigeng。他/她并未写明此头文件的版权协议，在此我将其默认为与龙芯开源主板使用相同的版权协议。如有侵权请联系我。

```
/*************************************************************
    > File Name: gpio.h
    > Author: shigeng
    > Mail: shigeng_bj@sohu.com
    > Created Time: 2015/6/27 18:46:40
 *************************************************************/

#ifndef _MY_GPIO_H_
#define _MY_GPIO_H_
extern int gpio_export(int pin);
extern int gpio_unexport(int pin);
extern int gpio_direction(int pin, char *dir);
extern int gpio_direction_get(int pin);//获取当前方向
extern int gpio_direction_in(int pin);//设置当前方向为输入
extern int gpio_direction_out(int pin);//设置当前方向为输出
extern int gpio_value_get(int pin);//获取当前值
extern int gpio_value_set(int pin, int value);//value 可选值：0, 1
#endif
```

从论坛上获得的控制 GPIO 口的头文件

摩斯码很重要的一点就是嘀、嗒的持续时间以及嘀嗒之间、字母之间以及单词之间间隔的时间。依照标准，一个嘀的长度为一个单位时间，一个嗒的长度为三个单位时间，嘀嗒之间的间隔为一个单位，两个字母之间为三个单位，两个单词之间为七个单位。一个单位时间是由码速所决定的，码速则由 WPM 为单位，意为单词每分钟。这里的单词以”PARIS”一词为标准；因此，码速的完整意思是“在一分钟之内可以使用摩斯码发送多少个’PARIS’”。由此我们可以得到公式：单位时间(秒)=1.2/WPM。但由于 C 语言中 usleep 函数是以微秒计的，因此在源码中应该用 1200000/WPM 来计算单位时间。

```
printf("请输入wpm数\n");
scanf("%d",&wpm);
dit=1200000/wpm;
dah=3600000/wpm;
wrd=8400000/wpm;
```

时间的计算

程序中涉及到了两次遍历数组，第一次是 morslize 函数遍历所输入的字符。此数组长度未知，因为输入的内容未知。然而通过循环遍历数组是需要知道数组长度的。由于在 C 中数组参数的本质是指针，我们不能在 morslize 函数中简单地使用 sizeof(数组)来获得长度，因为这里的数组本质是指针。对此我调用了 string.h 头函数中的 strlen 函数来获得长度。

第一个遍历尚好解决，第二个就比较棘手：对于 play 最终的函数来讲，其参数（也就是上文提到的 0 1 数组）长度未知，因为每个字母摩斯码嘀 嗒的数量都不一样。而其数据类型为 int，并非可以使用 strlen 函数的字符数组。对此问题我的解决方法是把嘀嗒数量当成数组的第一个元素传递出去，而在 play 函数遍历的时候跳过第一个元素。

```
int morslize(char o[]){
    int i;
    for(i=0;i<strlen(o);i++){
        if(o[i]==65|o[i]==97){  //a
            int m[3]={2,0,1};
            play(m);
            usleep(iidit);
        }
        else if(o[i]==66|o[i]==98){  //b
            int m[5]={4,1,0,0,0};
            play(m);
            usleep(iidit);
        }
        else if(o[i]==67|o[i]==99){ //c
            int m[5]={4,1,0,1,0};
            play(m);
            usleep(iidit);
        }
        else if(o[i]==68|o[i]==100){  //d
            int m[4]={3,1,0,0};
```

Morslize 函数的一部分

在 play 函数的具体机理是：如果遍历到的元素是 1，则把 GPIO 口设置为高电平，等待 3 个单位时间（即一个嘀的时间），然后把 GPIO 设置为低电平；如果元素是 0，则同理，不过等待的时间只有一个单位时间。

```
int play(int a[]){
    int i;
    for(i=1;i<a[0]+1;i++){
        if(a[i]==1){
            gpio_value_set(47,1);
            usleep(dah);
            gpio_value_set(47,0);
        }
        else{
            gpio_value_set(47,1);
            usleep(dit);
            gpio_value_set(47,0);
        }
        usleep(dit);
    }
    return 0;
}
```

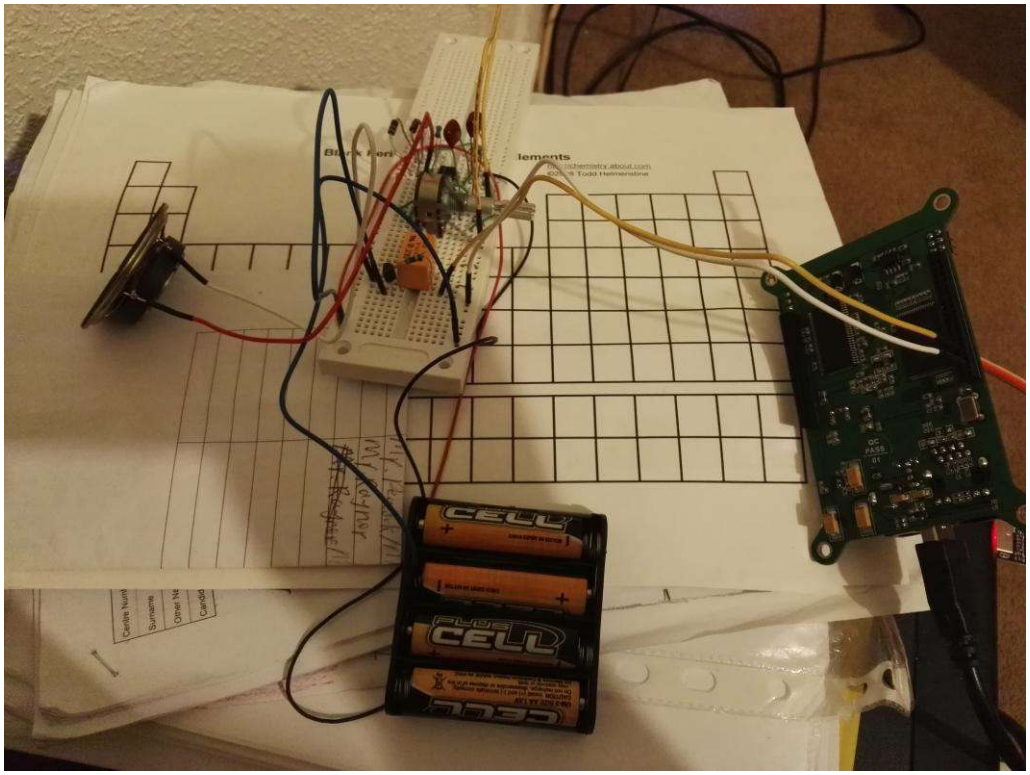
play 函数

程序码好之后,修改一下 shigeng 所提供的 makefile 文件,将源码以及 GPIO 头文件及函数利用 mipsel-linux-gcc 编译为一个独立的、不需要外部依赖的 mipsel 可执行文件，然后通过 ncat 发送到龙芯主板上。

将 GPIO47 口及 GND 与电路连接，执行 morse 程序，输入 WPM 值及要播放的文本，即可听到继电器开闭以及扬声器播放摩斯码的声音。

```
inners@inners-All-Series: ~
ls1x-rtc ls1x-rtc.0: setting system clock to 2069-03-23 12:21:59 UTC (3131266919
)
yaffs: dev is 32505857 name is "mtdblock1" rw
yaffs: passed flags ""
VFS: Mounted root (yaffs2 filesystem) on device 31:1.
devtmpfs: mounted
Freeing unused kernel memory: 200k freed
#mount all.....
#Starting mdev.....
Processing /etc/profile.....
Done!
[root@Loongson-gz:/]#ls
bin         helloLS.o   linuxrc     man         root        sys         var
dev         home        lost+found  mnt         sbin        tmp
etc         lib         lynx.tar    proc        share       usr
[root@Loongson-gz:/]#cd home
[root@Loongson-gz:/home]#ls
2048        listprime   morse
[root@Loongson-gz:/home]#./morse
请输入wpm数
30
请输入要翻译的文本
The quick brown fox jumps over the lazy dog
```

运行时界面



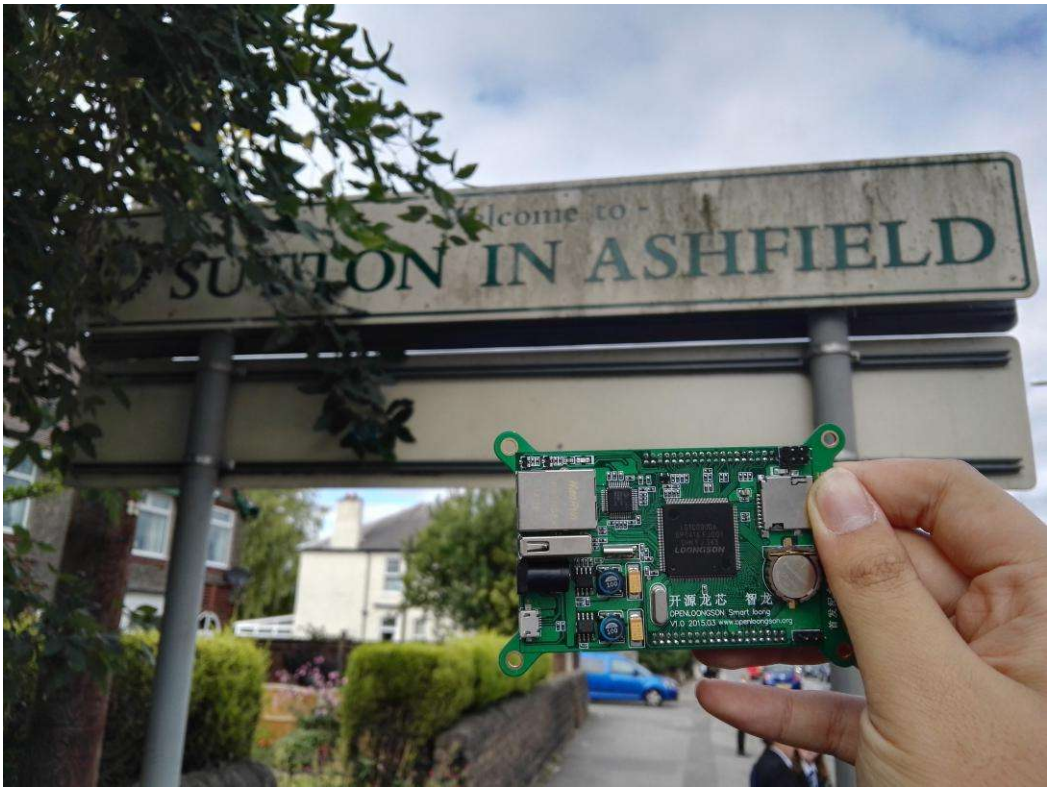
全家福。由于我没有笔记本电脑，这些东西只能放在主机上面，空间略显窘迫

致此，这个在中国开始，在英国完成的项目正式完工了。从构想到完成总共花了约一周多时间。其实这个小项目很简单，对于有电路设计经验的人来说或许三四天就能完成。下一步我可以把这些元件焊到实验板上，甚至设计出 PCB 让厂

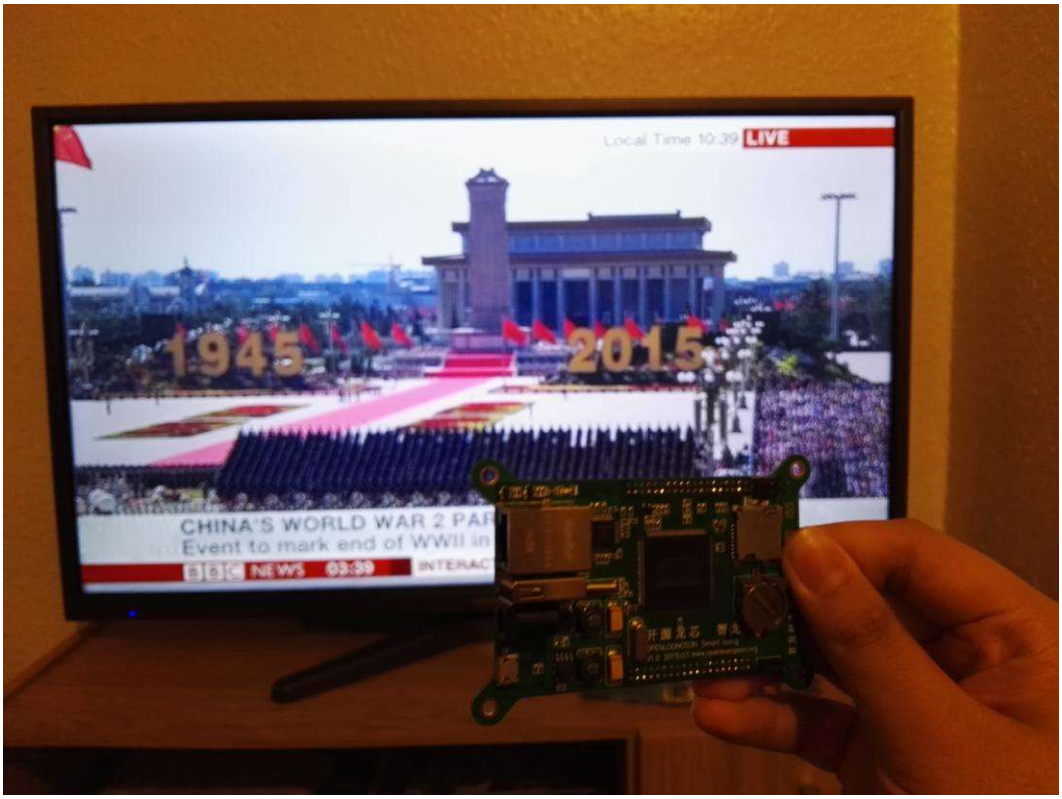
家来生产——不过设计 PCB 也不是件简单事，如果我有时间的话可能会研究下。

此外，我也应该是第一个将开源龙芯主板带出国门的人。可惜我所在的小镇离著名的地标性建筑很远，因此就让龙芯主板与我小镇上的一些标志合影吧。

英国的红绿灯



我所在的小镇(Sutton in Ashfild)的路牌



BBC 直播的阅兵


NOTTINGHAMSHIRE
Fire & Rescue Servic
Ashfield
Creating Safer Communities

消防队

我的学校 我在上 11 年级，相当于国内高一

社区

原文地址：http://www.loongsonclub.com/bbs/portal.php?mod=view&aid=55

3.3 俄罗斯方块

在开源龙芯主板运行俄罗斯方块游戏

1. 从官网下载它的 Linux 的源代码

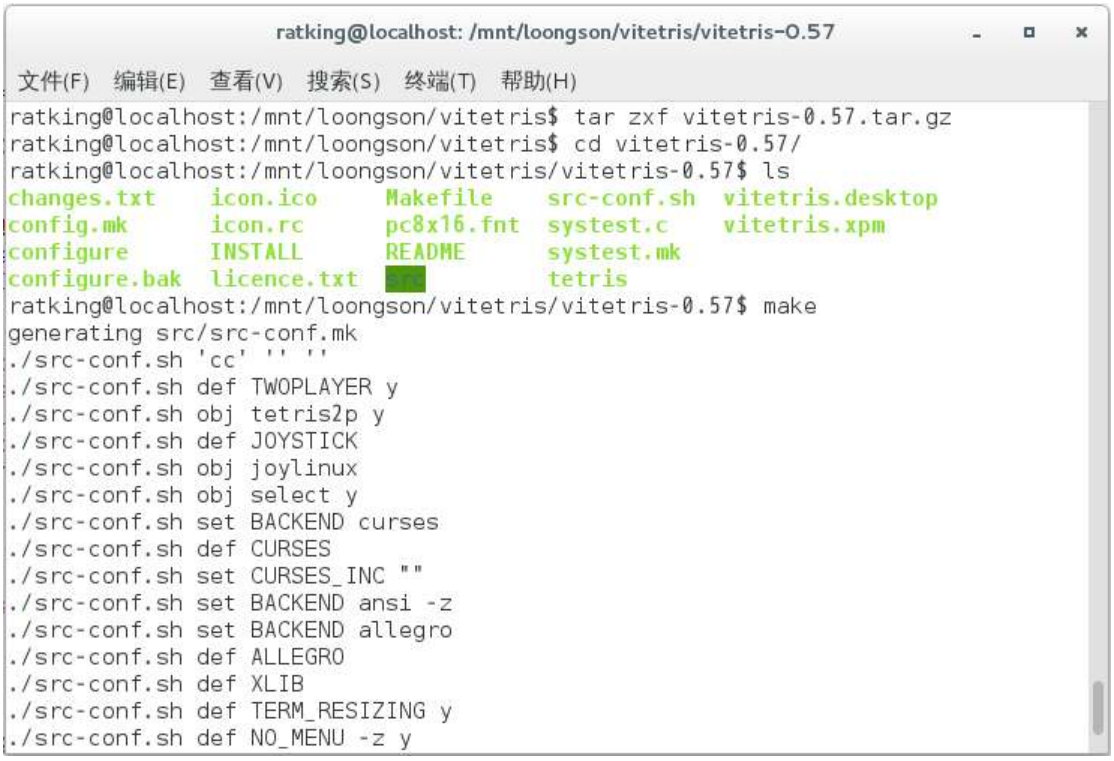
http://www.victornils.net/tetris/vitetris-0.57.tar.gz (约 100KB)

2. 需要在一台普通笔记本电脑的 Linux（我使用的是 Debian 8.1）里用 tar 命令解压缩，然后使用 make 命令编译，生成 tetries。

执 行 以 下 命 令 ：

```
tar zxf vitetris-0.57.tar.gz
或  tar zxvf vitetris-0.57.tar.gz （带参数v的输出解压过程信息）
cd vitetris-0.57/
ls
make
```

复制代码



3.这个 terties 可执行程序实在 linux 中运行的，因此还不能在龙芯 1C 里运行。1C 程序执行以下命令：

```
01.  file tetris
02.  ./tetris
复制代码
```

按方向键与回车是选择菜单，按键盘左上角的[Esc]键可以返回上级或退出游戏。

想在智龙里运行它，需要使用交叉编译工具编译器 mipsel-linux-gcc 把它编译成智龙 V1.0 的龙芯

执行交叉编译命令“make CC=mipsel-linux-gcc”，编译出龙芯 1C 的程序 tetries。

使用 file 命令查看文件信息，显示“MIPS32”，表示生成的已经是龙芯的程序这个程序可以在智龙的嵌入式 Linux 里运行，无法在普通电脑的 Linux 里运行。

执行以下命令：

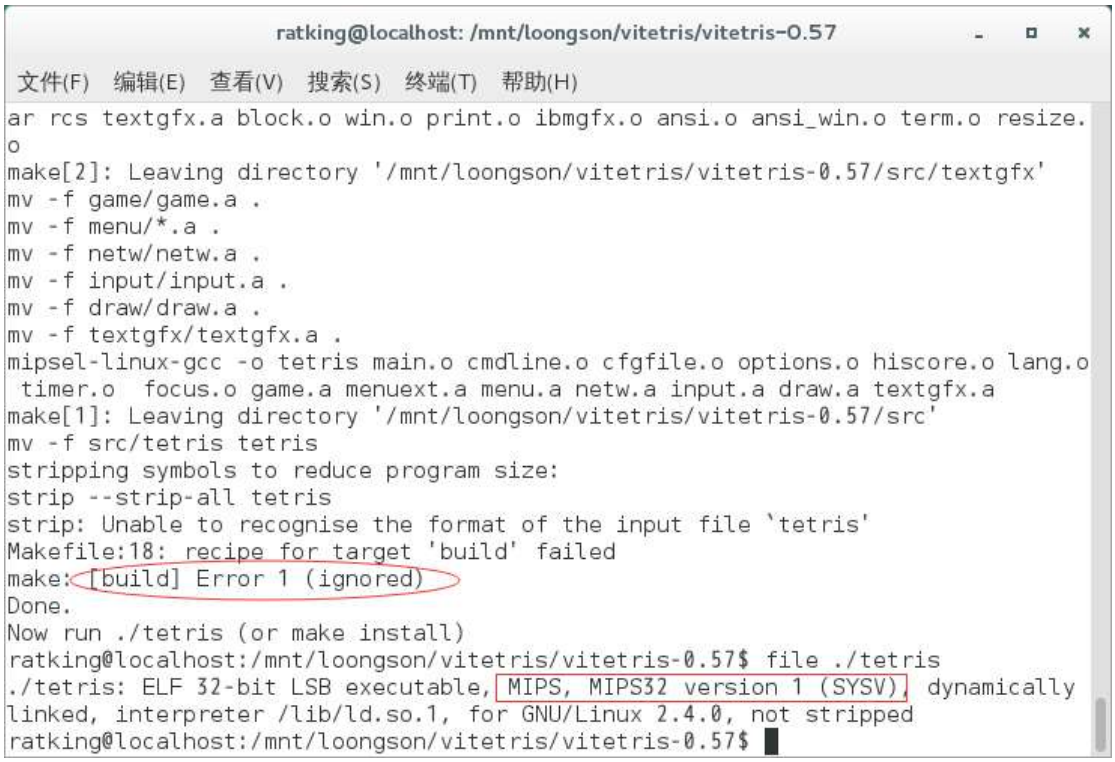
01.tar zxf vitetris-0.57.tar.gz

02.cd vitetris-0.57/

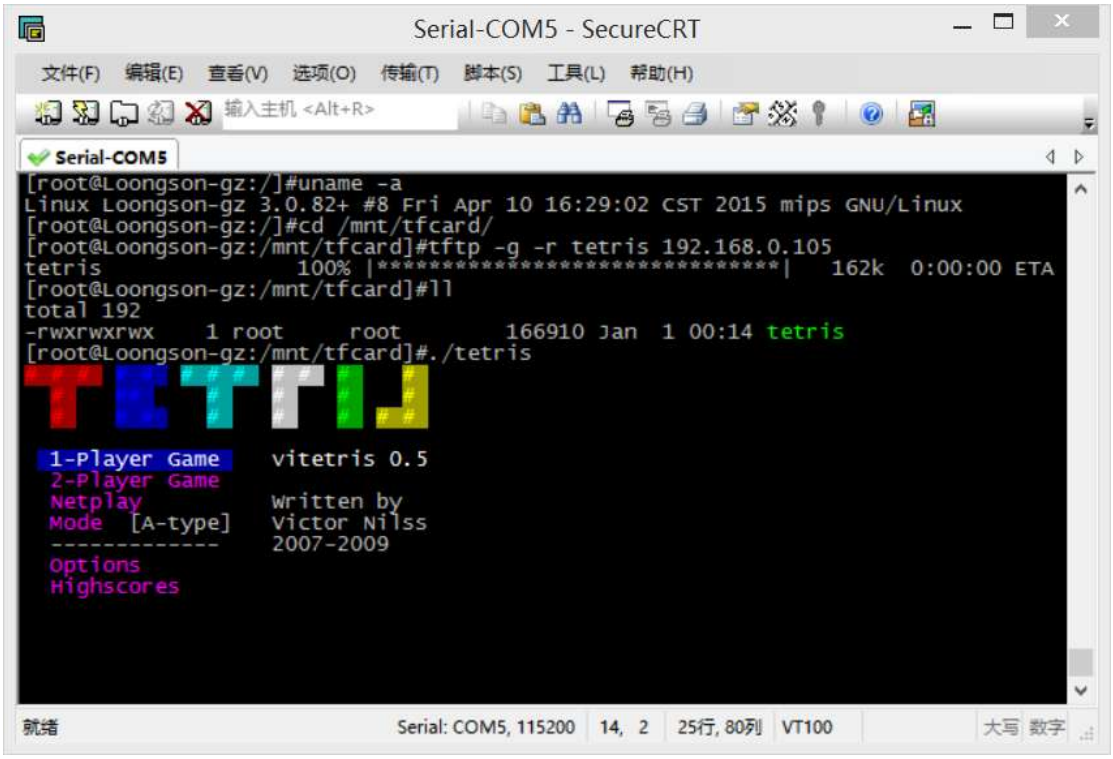
03.make CC=mipsel-linux-gcc

04.file ./tetris

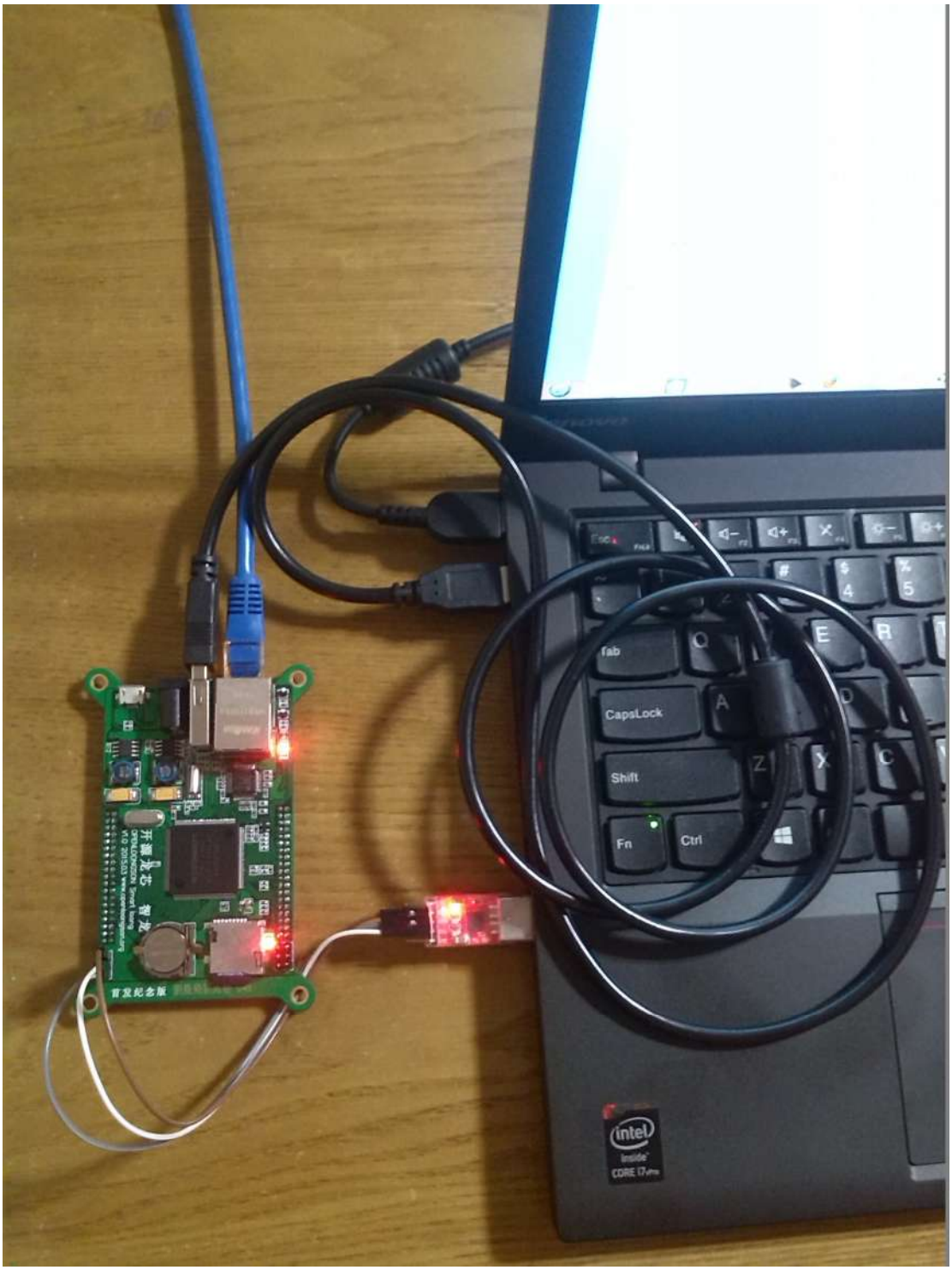
复制代码



5.通过 tftp 把这个龙芯 1C 程序下载到智龙里。



6. 使用的硬件：智龙 V1.0、USB 转 TLL 数据线、智龙自带的 USB 数据线(两个标准 USB 口，用于给智龙提供 5V 直流电源)、RJ45 水晶头网线、笔记本电脑、路由器、宽带。



7. 使用的软件：Windoz 8.1、VirtualBox 5.0、Debian 8.1、gcc-4.3-ls232.tar.gz、SecureCRT 6.7.0、Tftpd64 4.52、智龙里的嵌入式 Linux、vitetris-0.57：

原文地址：
http://www.openloongson.org/forum.php?mod=viewthread&tid=100&extra=page%3D2

3.4 智龙连接物联网平台智城云

3.5 智龙连接微信公众号

3.6 3D 打印机主板

附录

龙芯 1C 引脚复用表

Linux 常用命令

命令			
ls	显示目录/档案	reboot	重启
cd	打开档案	pwd	显示当前所在目录
chmod	改变档案权限	chgrp	改变档案所属群组
chown	改变档案拥有者	mkdir	新建目录
cp	复制档案/目录	cat	由第一行开始显示档案内容
mv	移动/重命名	touch	修改档案时间/新建档案
mr	删除	man	查看命令解析
cal	显示日历	info	查看命令解析
bc	计算器	tar -xvfz	解压一个 gzip 格式的压缩包
shutdown	关机	tar -cvfz	创建一个 gzip 格式的压缩包

PMON 命令

命令			
devcp	copy form src to dst	cp0s	access cp0
d1	dump address byte	d2	dump address half word
d4	dump address word	d8	dump address double word
m1	modify address byte	m2	mofify address half word
m4	modify address word	m8	modify address double word

xmodem	xmodem [base=baseaddr] [file=filename]	ymodem	ymodem base=baseaddr] [file=filename]
sysinfo	硬件测试	info	硬件测试
setup	设置 boot loader	|	run cmd and return 0
mtd_erase	yaffs write and read	losetup	losetup
md5sum	md5sum file	erase_all	擦除 Nand 芯片上的所有数据

网络			
ifconfig	ifconig fx0 [up|down|remove|stat| setmac|readrom|setrom|addr netmask]	ifup	ifup fxp0
ifdown	ifdown fxp0	ifaddr	Configure Network Interface
ping	ping IP		

Boot and Load			
boot	boot	oload	load memory from hostport
load	load file		

Misc			
call	call function	devls	罗列所有设备
flush	flush caches	reboot	重启系统
poweroff	重启系统	halt	重启系统
flash	烧写 flash	tr	transparent mode
rz	zmodem download		

Debugger			
c	continue execution	t	trace (single step)
to	trace (step over)	db	删除断点(s)
b	设置断点	g	start execution (go)
sym	define symbol	ls	list symbols
r	display/set register	l	list (disassemble) memory
bt	stack backtrace		

Shell			
more	paginator	h	帮助信息
sh	命令 shell	vers	打印版本信息
eval	evaluate and print result	hi	显示命令历史
date	获得或设置日期和时间		

grub like command			
initrd	load initrd/initramfs image		

Pci			
pcicfg	PCI 配置	pciscan	扫描 PCI 总线

环境变量			
env	显示环境变量	set	显示或设置环境变量
unset	取消环境变量设置	eset	修改环境变量

rays			
bl	从配置文件中加载 boot 菜单		