

MYD-AM335X-Y Linux 开发手册

版本 V1.0

版本记录

版本号	说明	时间
V 1.0	初始版本	2014.9
V 1.1	补充 SDIO Wi-Fi 内容	2014.9.29

目 录

第 1 章 概述及软件资源介绍	1
1.1 概述	1
1.2 软件资源	1
第 2 章 Linux 开发环境搭建	3
2.1 建立工作目录	3
2.2 设置交叉编译工具	3
2.3 安装工具	3
第 3 章 Linux 系统编译	4
3.1 Bootloader	4
3.2 编译 Linux 内核	5
3.3 制作文件系统	5
第 4 章 Linux 系统烧写	7
4.1 TF 卡系统映像更新	7
4.2 NAND Flash 更新/恢复	8
第 5 章 Linux 应用程序	10
5.1 CAN	10
5.2 Key&LED	10
5.3 NET	11
5.4 RTC	11
5.5 NAND Flash	12
5.6 LCD	13
5.7 RS485	13

5.8 Audio	13
5.8 SDIO Wi-Fi	14
第 6 章 Qt 开发	15
6.1 使用光盘提供的 Qt SDK	15
6.2 交叉编译 Qt 开发环境	15
6.3 移植 Qt 到开发板	16
6.4 交叉编译 Qt 应用程序	17

第 1 章 概述及软件资源介绍

1.1 概述

MYD-AM335X-Y 提供了丰富的系统资源和软件资源，本手册将从环境搭建开始，一步步介绍如何进行 MYD-AM335X-Y Linux 开发。本手册中开发主机上的命令以 Ubuntu 为例进行教授。

1.2 软件资源

类别	名称	描述	源码
Boot loader	U-boot	二级引导程序，负责系统初始化和引导内核 1.支持 NAND Flash 擦除读写 2.支持网络下载映像 3.支持设置、保存环境变量 4.支持内存内容显示、对比、修改 5.支持 bootcmd、bootargs 等设置	YES
内核	Linux 3.2.0	专为 MYD-AM335X-Y 的硬件制定的 Linux 内核	YES
驱动	USB Host	USB Host 驱动，支持 OHCI 和 EHCI 两种传输模式	YES
	USB Device	USB Device 驱动 (Gadget)	YES
	Ethernet	以太网驱动	YES
	MMC/SD	MMC/SD 卡驱动	YES
	NAND Flash	NAND Flash/SmartMedia 驱动	YES
	I2C	I2C 驱动	YES
	SPI	SPI 驱动	YES
	Audio	SGTL5000 音频驱动	YES
	LCD	LCD 屏驱动，可支持 4.3 寸, 7 寸液晶屏(后续支持)	YES
	RTC	内置 RTC 时钟驱动	YES

类别	名称	描述	源码
	TouchScreen	4 线电阻触摸屏驱动	YES
	PWM	PWM (脉宽调制) 驱动	YES
	UART	串口驱动	YES
	CAN	CAN 驱动	YES
	PMU	电源管理驱动	YES
	LED	LED 驱动, 包括 GPIO LED 和 PWM LED 驱动	YES
	GPIO	GPIO 驱动	YES
	SDIO Wi-Fi	SDIO 无线网卡驱动	NO
文件 系统	rootfs	基于 buildroot 定制的文件系统	二进制
	rootfs-qt	Qt 文件系统	二进制
应用 程序	Aduio	Aduio 测试程序	YES
	CAN	CAN 测试程序	YES
	Key&LED	按键 LED 测试程序	YES
	NET	网络测试程序	YES
	RTC	RTC 时钟测试实验	YES
	NAND Flash	NAND Flash 时钟测试实验	YES
	GPIO	GPIO 测试程序	YES
	RS485	RS485 测试程序	YES
	Qt	Qt 环境以及演示程序	YES

表 1-1

第 2 章 Linux 开发环境搭建

2.1 建立工作目录

拷贝 MYD-AM335X-Y 光盘资料到 Linux 开发主机中:

```
$ mkdir -p <WORKDIR>
$ cp -a <DVDROM>/05-Linux_Source/* <WORKDIR>
```

2.2 设置交叉编译工具

```
$ cd <WORKDIR>/Toolchina
$ tar -xvjf \
gcc-linaro-arm-linux-gnueabi-hf-4.7-2013.03-20130313_linux.tar.bz2
$ export PATH=$PATH:<WORKDIR>/Toolchain/\
gcc-linaro-arm-linux-gnueabi-hf-4.7-2013.03-20130313_linux/bin
$ export CROSS_COMPILE=arm-linux-gnueabi-hf-
```

执行完“export”命令后输入 arm 按 Tab 键来检查是否设置成功，该设置只对当前终端有效，如需永久修改，请修改用户配置文件。

2.3 安装工具

安装其他必要工具:

```
$ sudo apt-get install build-essential git-core libncurses5-dev
$ sudo apt-get install flex bison texinfo zip unzip zlib1g-dev gettext
$ sudo apt-get install gperf libSDL-dev libesd0-dev libwxgtk2.6-dev
$ sudo apt-get install uboot-mkimage
$ sudo apt-get install g++ xz-utils
```

第 3 章 Linux 系统编译

3.1 Bootloader

注：目前暂不支持 7.0 寸 LCD，将会在后续添加该支持。

(1) 编译选项：

进入 Bootloader 目录，解压 U-boot 源码：

```
$ cd <WORKDIR>/Bootloader
$ tar -jxvf u-boot.tar.bz2
$ cd u-boot
```

开始编译：

```
$ make ARCH=arm distclean
$ make ARCH=arm myd_am335x_lcd4.3_config
$ make ARCH=arm
```

编译完成后，在 u-boot 目录下会生成 MLO 和 u-boot.img 文件。这里需要注意的是，第二个 make 中的“myd_am335x_lcd4.3_config”表示将 4.3 寸 LCD 触摸屏的启动选项加入到 U-boot 的默认参数中，这里可以配置成如下几种默认显示：

显示模式	配置项
LCD 4.3 inch (480 x 272)	myd_am335x_lcd4.3_config
LCD 7.0 inch (800 x 480, 电阻式)	myd_am335x_lcd7.0r_config
LCD 7.0 inch (800 x 480, 电容式)	myd_am335x_lcd7.0c_config

表 3-1

(2) 启动选项：

MYD-AM335X 有多种硬件配置，需要根据实际的硬件情况设置相应的参数。下面列出了 u-boot 参数与硬件的关系：

参数	值	含义
mem	256M	DDR 256MB
	512M	DDR 512MB
board-am335xevm.display_mode	lcd4i3	4.3 寸电阻式触摸屏
	lcd7ir	7 寸电阻式触摸屏

参数	值	含义
	lcd7ic	7 寸电容式触摸屏
sdio_wifi	0	不加载 SDIO 无线网卡驱动
	1	加载 SDIO 无线网卡驱动

表 3-2

例如, 在 U-boot 命令行中输入如下命令, 将硬件设置为 7.0 寸电阻式触摸屏和 256M 内存, 且不加载 SDIO 无线网卡驱动:

```
# setenv optargs board-am335xevm.display_mode=lcd7ic mem=256M sdio_wifi=0
# saveenv
```

3.2 编译 Linux 内核

进入 Kernel 目录, 解压内核源码:

```
$ cd <WORKDIR>/Kernel
$ tar -xvjf linux-3.2.0-psp04.06.00.08.sdk.tar.bz2
$ cd linux-3.2.0-psp04.06.00.08.sdk
```

开始编译:

```
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- distclean
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- \
myd_am335x_y_defconfig
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- uImage
```

编译完成后, 会在 arch/arm/boot 目录下生成 uImage 文件。

3.3 制作文件系统

(1) 安装必要工具:

```
$ sudo apt-get install mtd-utils
```

(2) 进入文件系统目录:

```
$ cd <WORKDIR>/Filesystem/
```

(3) 解压文件系统:

```
$ tar xvjf myd-am335x_rootfs.tar.bz2
```

(4) 制作适用于 256M NandFlash 的文件系统:

```
$ ./mk-ubi-256mb myd-am335x_rootfs
```

执行完上述操作后, 当前目录下生成的 ubi.img 文件, 就是我们需要的适用于 256M

NandFlash 的 ubi 文件系统。

(5) 制作适用于 512M NandFlash 的文件系统:

```
$ ./mk-ubi-512mb myd-am335x_rootfs
```

执行完上述操作后，当前目录下生成的 ubi.img 文件，就是我们需要的适用于 512M NandFlash 的 ubi 文件系统。

第 4 章 Linux 系统烧写

注：启动开发板之前需要注意 JP1 跳线的连接，连接 JP1 的 1-2 脚时将会从 SD 卡启动系统，连接 JP1 的 2-3 脚时将会从 NandFlash 启动系统。改变 JP1 跳线的连接之后需要给开发板重新上电启动才能生效。

4.1 TF 卡系统映像更新

(1) TF 卡格式化

请使用光盘目录 03-Tools 目录下的 HP USB Disk Storage Format Tool 2.0.6 工具来格式化 TF 卡。

- ① 把 MMC/SD 卡插入 USB 读卡器，然后将读卡器跟电脑连接
- ② 打开 HP USB Disk Storage Format Tool，出现类似提示如下：

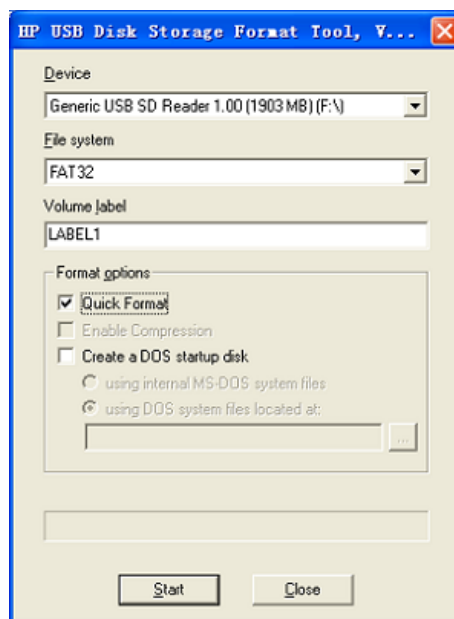


图 4-1

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

注意：HP USB Disk Storage Format Tool 会将清除 TF 卡的分区。若需要保留分区，请使用电脑系统自带的格式软件。

(2) 映像更新

将 02-Images\Linux-image\256M_NandFlash\LCD_4.3 目录下的所有文件拷贝到 TF 卡上，然后将 TF 卡插入到开发板上的 SD 卡插槽，连接 JP1 跳线的 1-2 脚，上电重启开发板。

4.2 NAND Flash 更新/恢复

Nand 启动映像的更新需要借助于 u-boot 来完成。不管 NAND Flash 是否有数据，都可以利用 TF 卡启动的 u-boot 对 NAND Flash 更新映像。

(1) 准备

① 用光盘目录 03-Tools 目录下的 HP USB Disk Storage Format Tool 2.0.6 工具将 TF 卡格式化为 FAT 或 FAT32 文件系统

② 将光盘目录 02-Images\Linux-image\256M_NandFlash\LCD_4.3 目录下的映像文件拷贝到 TF 卡中

(2) 更新

① 将带有系统映象的 TF 卡插入开发板，连接 JP1 的 1-2 脚，上电启动，在 u-boot 的提示读秒处，按下键盘上任意键进入 u-boot:

```
U-Boot SPL 2011.09 (Jan 13 2014 - 17:02:44)
Texas Instruments Revision detection unimplemented
Booting from MMC...
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img

U-Boot 2011.09 (Jan 13 2014 - 17:02:44)

I2C: ready
DRAM: 256 MiB
WARNING: Caches not enabled
Did not find a recognized configuration, assuming General purpose EVM in
Profile 0 with Daughter board
NAND: HW ECC Hamming Code selected
256 MiB
MMC: OMAP SD/MMC: 0
*** Warning - bad CRC, using default environment
```

```
Net: cpsw
Hit any key to stop autoboot: 0
```

② 进入 u-boot 命令行后，在命令行上输入 “run updatesys”，开始自动更新系统：

```
# run updatesys

NAND erase.chip: device 0 whole chip
Erasing at 0x1ffe0000 -- 100% complete.
OK
reading MLO

38167 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x0, size 0x9517
38167 bytes written: OK
reading u-boot.img

232892 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x80000, size 0x38dbc
232892 bytes written: OK
reading uImage

3506488 bytes read
HW ECC BCH8 Selected

NAND write: device 0 offset 0x280000, size 0x358138
3506488 bytes written: OK
reading ubi.img

14942208 bytes read
SW ECC selected

NAND write: device 0 offset 0x780000, size 0xe40000
14942208 bytes written: OK
```

③ 系统更新完后，核心板上的 LED 灯 D3 将会闪烁，此时拨出 TF 卡，连接 JP1 跳线的 2-3 脚，给开发板重新上电后即可从 Nand Flash 启动。

第 5 章 Linux 应用程序

MYD-AM335X-Y 提供了常用外设的演示程序，程序以及源码都位于“<WORKDIR>/Examples/”，请根据目录内的 Makefile 或 README 文件进行编译：

```
$ cd <WORKDIR>/Examples/<app dir>
$ export CROSS_COMPILE=arm-linux-gnueabi-
$ make
```

在开发板上运行程序时需要注意权限，如果无法执行请使用如下命令：

```
# chmod +x *
```

5.1 CAN

本例程演示使用 Linux API 让开发板上的 CAN 发送和接收数据，详情请参考源码。

将目录“<WORKDIR>/Examples/can”中的可执行程序 can_send 和 can_receive 拷贝至开发板。执行以下步骤：

(1) 将开发板上 CON2 的 1、2 脚分别跟另一开发板上 CON2 的 1、2 脚相连，把调试串口 J10 连到 PC 上并将 PC 串口的波特率设为 115200，数据位为 8，停止位为 1，无奇偶校验。

(2) 分别将两块开发板上的 can0 通信波特率都设置位 50KBps，并使能 can0 设备。

```
# canconfig can0 bitrate 50000 ctrlmode triple-sampling on
# canconfig can0 start
```

(3) 两块开发板一个作为发送端另一个作为接收端，先在一块开发板执行以下命令发送数据：

```
# ./can_send -d can0 -i 0x01 0x11 0x22 0x33 0x44 0x55 0x66
```

再在另一开发板执行以下命令接收数据：

```
# ./can_receive -d can0
can0 0x1 [6] 0x11 0x22 0x33 0x44 0x55 0x66
can0 0x1 [6] 0x11 0x22 0x33 0x44 0x55 0x66
can0 0x1 [6] 0x11 0x22 0x33 0x44 0x55 0x66
```

5.2 Key&LED

本例程演示如何使用 Linux API 操作开发板上的 USER 按键与 LED，详情请参考源码。

将目录 “<WORKDIR>/Examples/key_led” 中的可执行程序 key_led 拷贝至开发板，把调试串口 J10 连到 PC 上并将 PC 串口的波特率设为 115200，数据位为 8，停止位为 1，无奇偶校验。执行以下命令后按下开发板上的 K3 或 K4 按键将打印出相关的按键信息，同时 D2 点亮，松开按键后 D2 熄灭：

```
# ./key_led /dev/input/event0
type:1, code:1, value:1
type:1, code:1, value:0
# ./key_led /dev/input/event0
type:1, code:102, value:1
type:1, code:102, value:0
```

5.3 NET

本例程演示使用 Linux API 让开发板上的网口发送和接收数据，详情请参考源码。

将目录“<WORKDIR>/Examples/network”中的可执行程序 client 拷贝至开发板，server 拷贝至 PC，首先配置好 PC 和开发板的 ip 地址，以 eth0 为例，这里 PC 的 ip 是 192.168.1.1，开发板的 ip 是 192.186.1.2。配置好网络后，将开发板上 eth0 用网线连到 PC，把调试串口 J10 连到 PC 上并将 PC 串口的波特率设为 115200，数据位为 8，停止位为 1，无奇偶校验。以 PC 为服务器端，开发板为客户端，先在 PC 执行以下命令：

```
$ ./server 192.168.1.2
```

再在开发板执行以下命令可看到所发送的信息：

```
# ./client 192.168.1.1
send messages: 1234567890 to 192.168.1.1
send messages: 1234567890 to 192.168.1.1
send messages: 1234567890 to 192.168.1.1
```

同时可以看到 PC 接收到开发板发送的数据：

```
$ ./server 192.168.1.2
received messages: 1234567890 from 192.168.1.2
received messages: 1234567890 from 192.168.1.2
received messages: 1234567890 from 192.168.1.2
```

5.4 RTC

本例程演示如何使用 Linux API 对开发板上的 RTC 进行时间设置与读取，详情请参考源码。

将目录 “<WORKDIR>/Examples/rtc” 中的可执行程序 `rtc_test` 拷贝至开发板，把调试串口 J10 连到 PC 上并将 PC 串口的波特率设为 115200，数据位为 8，停止位为 1，无奇偶校验。执行以下命令将 rtc 当前时间设置为 2014/8/14 11:03:40 并读取当前时间：

```
# ./rtc_test -s 11 03 40 2014 08 14
    date/time is updated to: 14-8-2014, 11:03:40.
# ./rtc_test
    Current RTC date/time is 14-8-2014, 11:03:50.
```

5.5 NAND Flash

本例程演示如何使用 Linux API 擦除、写入和读取开发板上的 nandflash，详情请参考源码。

将目录 “<WORKDIR>/Examples/mtd” 中的可执行程序 `nand_mtd` 拷贝至开发板，把调试串口 J10 连到 PC 上并将 PC 串口的波特率设为 115200，数据位为 8，停止位为 1，无奇偶校验。执行以下命令后可对 nandflash 进行擦除、写入和读取数据：

```
# ./nand_mtd /dev/mtd7

MTD Type: 4
MTD total size: 529006592 bytes
MTD erase size: 131072 bytes
MTD write size: 2048 bytes

erase the last block at 528875520

erase done!

writing 16 bytes data to flash...
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e
0x0f
write done!

reading data from flash...
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e
0x0f
read done!
```


5.6 LCD

通过对 Linux 的 FrameBuffer 操作，实现 LCD 的彩色栅格测试。

(1) 编译并拷贝<WORKDIR>\Examples\lcd 目录下的测试程序开发板。在 Linux 终端输入如下命令：

```
# ./lcd_test
```

(2) 运行程序，终端显示屏幕信息，LCD 屏幕会先后出现 8 种背景色，然后显示颜色渐变的调色板图案：

```
vinfo.xres=800
vinfo.yres=480
vinfo.bits_per_bits=16
vinfo.xoffset=0
vinfo.yoffset=0
finfo.line_length=1600
```

5.7 RS485

本例程演示如何使用 Linux API 配置开发板上的 RS485 并使其发送和接收数据，详情请参考源码。

将开发板上 CON2 中 rs485 对应的接口 4、5 脚分别与另一个开发板 CON2 的 4、5 脚相连；将调试串口 J10 连到 PC 上并将 PC 串口的波特率设为 115200，数据位为 8，停止位为 1，无奇偶校验；将目录 “<WORKDIR>/Examples/rs485” 中的可执行程序 rs485 拷贝至两块开发板并分别执行以下命令，PC 串口终端打印如下信息：

```
root@MYD-AM335X:~# ./rs485 -d /dev/ttyO1 -b 9600
SEND:0123456789
RECV:0123456789, total:10
SEND:0123456789
RECV:0123456789, total:10
SEND:0123456789
RECV:0123456789, total:10
```

5.8 Audio

本例程演示如何使用 Linux API 控制音频的输入与输出，详情请参考源码。

将 “<WORKDIR>/Examples/audio” 中的可执行程序 audio 拷贝至开发板，将耳机插

到音频输出口 J8，将音频输入到音频输入口 J9，执行 `audio` 程序开始录音，该程序会打开一个音频采集节点，将其配置为立体声，16 位，44.1kHz 的，交错读/写。然后它开始录音。

按“`Ctrl-C`”结束，如下所示：

```
# ./audio
snd_pcm_readi successful
```

5.9 SDIO Wi-Fi

开发板上还集成了一块 SDIO 无线网卡，按前文提到的方法加载网卡驱动，然后按如下步骤设置网络。

(1) 启动 `wpa_supplicant` 后台服务：

```
# wpa_supplicant -iwlan0 -Dwext -B -c/etc/wpa_supplicant.conf
```

(2) 使用 `wpa_cli` 配置网络：

```
# wpa_cli -iwlan0 scan
# wpa_cli -iwlan0 scan_r
# wpa_cli -iwlan0 remove_n all
# wpa_cli -iwlan0 add_n
# wpa_cli -iwlan0 set_n 0 ssid '"MYIR-LINK "'
# wpa_cli -iwlan0 set_n 0 psk '"Make Your Idea Real"'
```

注意：这里的设置是针对 WPA-PSK/WPA2-PSK 加密方式，其他加密方式请参考《[AN] SDIO Wi-Fi 应用笔记》。

(3) 使能网络连接：

```
# wpa_cli -iwlan0 enable_n 0
```

(4) 保存网络配置到 `conf` 文件：

```
# wpa_cli -iwlan0 save_c
```

(5) 通过 DHCP 获取 IP 地址：

```
# udhcpc -n -i wlan0
```

(6) 通过 `ping` 命令检查网络是否联通

```
# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.13: seq=1 ttl=64 time=0.401 ms
64 bytes from 192.168.1.13: seq=2 ttl=64 time=0.412 ms
```

第 6 章 Qt 开发

本小节描述在 MYD-AM335X-Y 上使用 Qt 进行 GUI 程序开发的方法和步骤，包括两大部分，第一部分讲述光盘中提供的 Qt SDK 的使用方法，一般的 Qt 程序开发使用光盘中提供的 Qt SDK 即可；第二部分讲述如何从 Qt 源码中编译生成 Qt 开发环境，当光盘中提供的 Qt 库不能满足 Qt 开发程序需求时才需要自己制定 Qt 开发环境。

6.1 使用光盘提供的 Qt SDK

(1) 解压编译好的 tslib 到 PC:

```
$ sudo tar xvfj \  
/media/cdrom/05-Linux_Source/Qt_Arm/tslib-prebuild.tar.bz2 -C /opt
```

(2) 解压 Qt SDK 到 PC:

```
$ sudo tar xvfj \  
/media/cdrom/05-Linux_Source/Qt_Arm/qt-4.8.5-sdk.tar.bz2 -C /opt
```

(3) 配置环境变量和交叉编译 Qt 应用程序

配置环境变量和交叉编译 Qt 应用程序请参考：[6.4 交叉编译 Qt 应用程序](#)。

6.2 交叉编译 Qt 开发环境

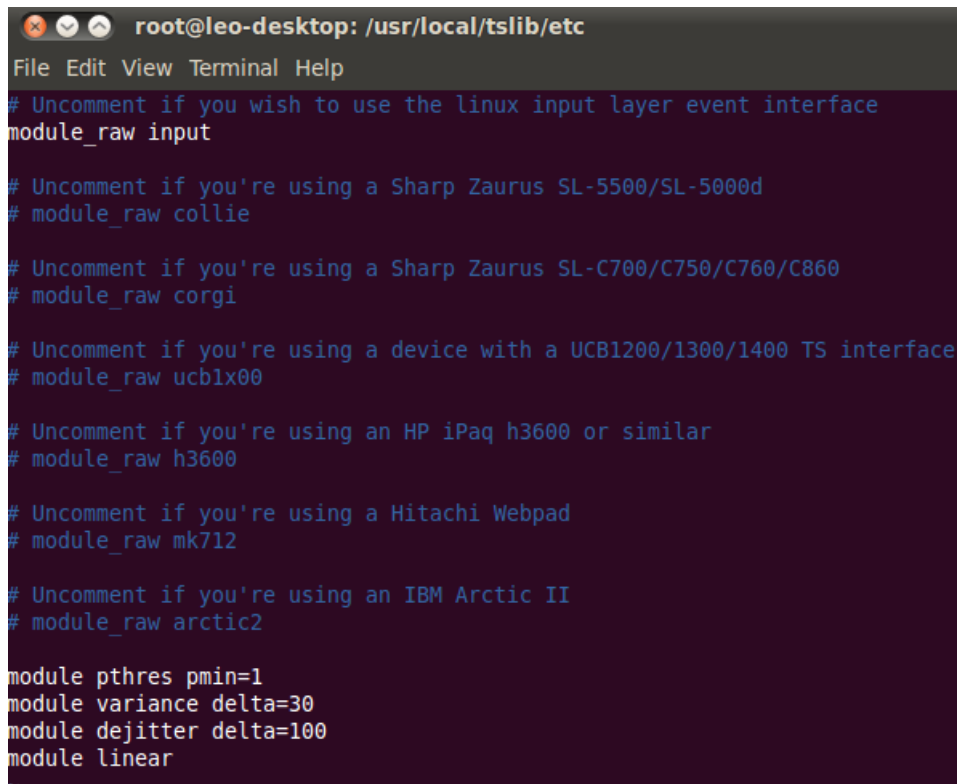
(1) 参考本文之前章节安装交叉编译工具，设置环境变量。

(2) 安装 automake、libtool、autoconf 包，编译安装 tslib:

```
$ sudo apt-get install automake libtool autoconf  
$ cd <WORKDIR>  
$ cp /media/cdrom/05-Linux_Souce/Qt_Arm/tslib-1.4.tar.bz2 ./  
$ tar -jxf tslib-1.4.tar.bz2  
$ cd tslib  
$ ./ts-build  
$ make install
```

注意：若执行 ts-build 出错时，编辑 ts-build 修改其中的交叉编译工具路径。

编译完成后，tslib 将被安装到 `/usr/local/tslib` 目录，此时需要将此目录下 `tslib/etc/ts.conf` 文件第二行 “`#module_raw input`” 的注释去掉，变为 “`module_raw input`”，注意一定要顶格，如图 6-1 所示：



```
root@leo-desktop: /usr/local/tslib/etc
File Edit View Terminal Help
# Uncomment if you wish to use the linux input layer event interface
module_raw input

# Uncomment if you're using a Sharp Zaurus SL-5500/SL-5000d
# module_raw collie

# Uncomment if you're using a Sharp Zaurus SL-C700/C750/C760/C860
# module_raw corgi

# Uncomment if you're using a device with a UCB1200/1300/1400 TS interface
# module_raw ucblx00

# Uncomment if you're using an HP iPaq h3600 or similar
# module_raw h3600

# Uncomment if you're using a Hitachi Webpad
# module_raw mk712

# Uncomment if you're using an IBM Arctic II
# module_raw arctic2

module pthres pmin=1
module variance delta=30
module dejitter delta=100
module linear
```

图 6-1

(3) 编译 qt

拷贝 Qt 源码到工作目录:

```
$ cd <WORKDIR>
$ cp \
/media/cdrom/05-Linux_Source/Qt_Arm/qt-everywhere-opensource-src-4.8.5
.tar.bz2\ ./
```

安装必要的工具包:

```
$ sudo apt-get install xorg-dev libfontconfig1-dev
libfreetype6-dev libx11-dev libxcursor-dev libxext-dev
libxfixes-dev libxft-dev libxi-dev libxrandr-dev libxrender-dev
```

开始编译:

```
$ tar -jxf qt-everywhere-opensource-src-4.8.5.tar.bz2
$ cd qt-everywhere-opensource-src-4.8.5
$ ./qt-build
$ sudo make install
```

编译完成后, qt-4.8.5 将被安装到/opt/qt-4.8.5 目录。

6.3 移植 Qt 到开发板

(1) 把安装到/usr/local 中的 tslib 目录下的文件拷贝到用于编译开发板文件系统的

myd-am335x_rootfs 文件夹的 usr/local/目录下:

```
$ cp -ra /usr/local/tslib <WORKDIR>/myd-am335x_rootfs/usr/local/
```

(2) 把安装到/opt 中的 qt-4.8.5 目录文件拷贝到用于编译开发板文件系统的

myd-am335x_rootfs 文件夹的 opt/目录下:

```
$ cp -r /opt/qt-4.8.5 <WORKDIR>/myd-am335x_rootfs/opt/
```

(3) 进入开发板上的/etc/init.d, 编写脚本 qt.sh, 如下:

```
export TSLIB_CONSOLEDEVICE=none
export TSLIB_FBDEVICE=/dev/fb0

export TSLIB_TSDEVICE=/dev/input/event1
export TSLIB_CONFFILE=/usr/local/tslib/etc/ts.conf
export TSLIB_PLUGININDIR=/usr/local/tslib/lib/ts
export TSLIB_CALIBFILE=/etc/pointercal
export
LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/tslib/lib:/opt/qt-4.8.5/lib
export QT_QWS_FONTDIR=/opt/qt-4.8.5/lib/fonts
export PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/tslib/bin

if grep "hdmi" /proc/cmdline > /dev/null ; then
    export QWS_MOUSE_PROTO="Tslib:/dev/input/event1
MouseMan:/dev/input/mice"
else
    export QWS_MOUSE_PROTO="Tslib:/dev/input/event1"
fi

export QWS_DISPLAY=:1
```

修改开发板上/etc/init.d/目录下的 rc 文件, 在该文件最后加上:

```
if [ -e /etc/init.d/qt.sh ];then
/etc/init.d/qt.sh
fi
```

(4) 参照【1.4.5】重新编译文件系统, 生成包含 QT 库和 tslib 的 ubi.img 文件

6.4 交叉编译 Qt 应用程序

(1) 配置 PC 机的 Qt 编译环境:

```
$ export QT_PREFIX=/opt/qt-4.8.5
$ export PATH=${QT_PREFIX}/bin:$PATH
$ export QMAKESPEC=${QT_PREFIX}/mkspecs/qws/linux-arm-g++
```

(2) 创建代码:

```
$ mkdir hellomyir
$ cd hellomyir
$ gedit hellomyir.cpp
```

输入如下代码:

```
#include <QApplication>
#include <QLabel>
int main(int argc, char **argv)
{
    QApplication app(argc,argv);
    QLabel label("Make Your idea Real!");
    label.show();
    return app.exec();
}
```

(3) 编译:

```
$ qmake -project
$ qmake
$ make
```

(4)将生成的 `hellomyir` 拷贝到开发板, 并在开发板上执行:

```
# ./hellomyir -qws
```

将编译生成的可执行文件拷贝到开发板中运行, 将会在 LCD 屏幕上看到 “*Make Your idea Real!*” 的 Qt 窗口。