**MYD-JA5D4X**

**Linux开发手册**

版本 V1.0

2015.05.28

**版本记录**

|  |  |  |
| --- | --- | --- |
| **版本号** | **说明** | **时间** |
| V1.0 | 初始版本 | 2015.05.28 |

# 目 录

[目 录 1](#_Toc420598435)

[第 1 章 概述及软件资源介绍 3](#_Toc420598436)

[1.1 概述 3](#_Toc420598437)

[1.2 软件资源 3](#_Toc420598438)

[第 2 章 Linux开发环境搭建 5](#_Toc420598439)

[2.1 建立工作目录 5](#_Toc420598440)

[2.2 设置交叉编译工具 5](#_Toc420598441)

[2.3 安装工具 5](#_Toc420598442)

[第 3 章 Linux系统编译 6](#_Toc420598443)

[3.1 编译Bootloader 6](#_Toc420598444)

[3.2 编译Linux内核 6](#_Toc420598445)

[第 4 章 Linux系统烧写 8](#_Toc420598446)

[第 5 章 Linux应用程序开发 9](#_Toc420598447)

[5.1 嵌入式开发环境 9](#_Toc420598448)

[5.1.1 配置TFTP服务 9](#_Toc420598449)

[5.1.2 配置NFS服务 10](#_Toc420598450)

[5.1.3 SSH登录 11](#_Toc420598451)

[5.2 应用实例 12](#_Toc420598452)

[5.2.1 GPIO 12](#_Toc420598453)

[5.2.2 Key&LED 12](#_Toc420598454)

[5.2.3 NET 13](#_Toc420598455)

[5.2.4 RTC 13](#_Toc420598456)

[5.2.5 TWI 14](#_Toc420598457)

[5.2.6 RS485 14](#_Toc420598458)

[5.2.7 RS232 14](#_Toc420598459)

[5.2.8 Audio 15](#_Toc420598460)

[5.2.9 EEPROM 15](#_Toc420598461)

[5.2.10 FrameBuffer 16](#_Toc420598462)

[5.2.11 Led\_play 17](#_Toc420598463)

[第 6 章 Qt开发 18](#_Toc420598464)

[6.1 使用光盘提供的Qt SDK 18](#_Toc420598465)

[6.2 交叉编译Qt开发环境 18](#_Toc420598466)

[6.3 移植Qt到开发板 20](#_Toc420598467)

[6.4 交叉编译Qt应用程序 20](#_Toc420598468)

[附录一 联系方式 22](#_Toc420598469)

[附录二 售后服务与技术支持 23](#_Toc420598470)

# 概述及软件资源介绍

## 概述

MYD-JA5D4X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-JA5D4X Linux开发。本手册中开发主机上的命令以Ubuntu为例进行介绍。

## 软件资源

| **类别** | **名称** | **备注** | **源码** |
| --- | --- | --- | --- |
| **引导程序** | MLO（SPL） | 一阶引导 | yes |
| U-boot | 二阶引导 | yes |
| **Linux内核** | Linux 3.18 | 专为MYD-JA5D4X硬件制定的Linux内核 | yes |
| **设备驱动** | USB Host | USB Host驱动 | yes |
| USB Device | USB Device驱动 | yes |
| I2C(TWI) | i2c-dev驱动 | yes |
| Ethernet | 以太网驱动 | yes |
| MMC | MMC/SD/TF卡驱动 | yes |
| eMMC | eMMC驱动 | yes |
| LCD | DSS驱动，支持7寸液晶屏 | yes |
| RTC | 实时时钟驱动 | yes |
| HDMI | SIL9022A驱动 | yes |
| Touch | 电容触摸屏驱动 | yes |
| Button | 按键驱动 | yes |
| USART | 串口驱动 | yes |
| LED | LED驱动 | yes |
| GPIO | GPIO驱动 | yes |
| WDI | 看门狗驱动 | yes |
| Camera | 双摄像头驱动 | yes |
| QSPI | QSPI Flash驱动 | yes |
| EERPOM | EERPOM驱动 | yes |
| ADC | ADC驱动 | yes |
| PWM | PWM驱动 | yes |
| **文件系统** | Ramdisk | 带系统更新工具的轻量级文件系统 | bin |
| Matrix | TI 官方提供的Matrix演示系统 | bin |
| Buildroot | 带Qt 4.8.5库的文件系统 | bin |
| **应用程序** | KEY&LED | 按键指示灯测试程序 | yes |
| NET | TCP/IP Sokect C/S测试程序 | yes |
| RTC | 实时时钟测试程序 | yes |
| I2C(TWI) | i2c-dev应用接口演示程序 | yes |
| RS485 | RS485测试程序 | yes |
| RS232 | RS232测试程序 | yes |
| Audio | Audio测试程序 | yes |
| EEPROM | EEPROM应用接口演示程序 | yes |
| Framebuffer | 显示设备演示程序 | yes |
| **工具** | Cross compiler | Linaro GCC 4.7 | bin |

表格 1‑1

# Linux开发环境搭建

## 建立工作目录

拷贝产品光盘中的源码到Linux开发主机中：

$ mkdir -p <WORKDIR>

$ cp -a <DVDROM>/04-Linux\_Source/\* <WORKDIR>

## 设置交叉编译工具

$ cd <WORKDIR>/Toolchain

$ tar -xvjf \

gcc-linaro-arm-linux-gnueabihf-4.7-2013.03-20130313\_linux.tar.bz2

$ export PATH=$PATH:<WORKDIR>/Toolchain/\

gcc-linaro-arm-linux-gnueabihf-4.7-2013.03-20130313\_linux/bin

$ export CROSS\_COMPILE=arm-linux-gnueabihf-

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。

## 安装工具

此外还需安装一些必要工具，以ubutnu系统为例：

$ 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

# Linux系统编译

## 编译Bootloader

进入Bootloader目录，解压U-boot源码：

$ cd <WORKDIR>/Bootloader

$ tar -xvjf u-boot-2013.10-ti2013.12.01.tar.bz2

$ cd u-boot-2013.10-ti2013.12.01

开始编译：

$ make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf- distclean

$ make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf- <config>

$ make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf-

编译完成后会在当前目录生成所需要的MLO（只有myd\_ja5d4x\_config配置才会生成）、u-boot.bin和u-boot.img。

这里的<config>是配置选项名称，不同的启动模式需使用不同的配置选项。

有如下两种情况：

|  |  |  |
| --- | --- | --- |
| **启动模式** | **编译选项** | **输出文件** |
| Q-SPI Flash | myd\_ja5d4x\_qspiboot\_config | u-boot.bin |
| Micro SD | myd\_ ja5d4x \_config | u-boot.img和MLO |

表格 3‑1

## 编译Linux内核

进入Kernel目录，解压内核源码：

$ cd <WORKDIR>/Kernel

$ tar -xvjf linux-3.12.10-ti2013.12.01.tar.bz2

$ cd linux-3.12.10-ti2013.12.01

开始编译：

$ make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf- distclean

$ make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf- \

myd\_ja5d4x\_defconfig

$ make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf- zImage dtbs

编译完成后在arch/arm/boot 目录会生成内核镜像zImage，在arch/arm/boot/dts 目录会生成设备树的二进制文件myd\_ja5d4x.dtb。

# Linux系统烧写

# Linux应用程序开发

## 嵌入式开发环境

嵌入式开发，通常先在PC上交叉编译后，再通过某种方式（如以太网或串口），将文件复制或传输到目标开发板上执行。为了方便调试，进行应用程序开发前，应该先配置好一个文件共享环境。

### 配置TFTP服务

TFTP简单文件传输协议，可以实现不同主机间的文件发送和接收，另外，U-boot启动时，还可以使用TFTP加载内核。

安装tftp服务端，以ubuntu为例：

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

创建tftp服务器工作目录

$ mkdir <WORKDIR>/tftpboot

$ chmod 777 <WORKDIR>/tftpboot

配置tftp服务器：

$ sudo vi /etc/default/tftpd-hpa

修改或添加如下字段：

TFTP\_DIRECTORY="<WORKDIR>/tftpboot"

TFTP\_OPTIONS="-l -c -s"

重启TFTP服务：

$ sudo service tftpd-hpa restart

假设PC机的IP地址为：192.168.1.111，运行以下命令可以将PC主机tftpboot目录中的文件传输到开发板：

# tftp -l <file-name> -r <file-name> -g 192.168.1.111 69

在开发板的控制终端运行下面的命令可以将开发板上的文件发送到PC主机的tftpboot目录：

# tftp -l <file-name> -r <file-name> -p 192.168.1.111 69

在U-boot命令行，可以使用以下方式加载PC机tftpboot目录中的zImage内核文件：

# setenv serverip 192.168.1.111

# setenv ipaddr 192.168.1.222

# setenv ethaddr 00:01:02:03:04:05

# tftp zImage <mem-addr>

# bootz <mem-addr>

### 配置NFS服务

NFS即网络文件系统，允许主机直接通过网络实现文件共享。另外，除了挂载普通目录外，还可以在Linux启动时将NFS上的目录挂载为开发板根文件系统。

在开发主机上安装NFS服务程序，以ubuntu为例：

$ sudo apt-get install nfs-kernel-server

编辑exports文件，添加nfs文件夹目录：

$ sudo vi /etc/exports

例如设置/home/myir/nfs为NFS目录，将如下内容添加到exports文件中：

/home/myir/nfs \*(rw,subtree\_check,no\_root\_squash,no\_all\_squash,sync)

设置目录权限：

$ chmod 777 –R /home/myir/nfs

修改设置nfs，绑定端口

$ sudo vi /etc/default/nfs-kernel-server

将RPCMOUNTDOPTS修改为：

#RPCMOUNTDOPTS=--manage-gids

RPCMOUNTDOPTS="-p 13100"

重启NFS服务：

$ sudo service nfs-kernel-server restart

在本机上测试NFS服务：

$ sudo mount -t nfs 127.0.0.1:/home/myir/nfs /mnt

若本机NFS挂载成功，接下来在开发板上挂载NFS，将开发板和主机接入同一网络，设置主机服务端IP，例如：

$ sudo ifconfig eth0 192.168.1.111

设置开发板IP，例如：

# ifconfig eth0 192.168.1.222

使用ping命令测试开发板与PC机网络是否连通：

# ping 192.168.1.111

在开发板上新建挂载目录：

# mkdir -p /mnt/nfs

挂载NFS文件夹：

# mount -t nfs -o nolock,rw 192.168.1.111:/home/myir/nfs /mnt/nfs

至此，NFS共享文件夹挂载成功，通过NFS共享文件夹可以方便的实现主机和开发板间的文件共享，并且可以直接在共享文件夹/mnt/nfs下运行目标程序，而免去了文件拷贝的操作。另外，除了挂载目录外，还可以在启动时将NFS上的目录挂载为开发板根文件系统，详细步骤可参考以下网页：

<http://wiki.emacinc.com/wiki/Booting_with_an_NFS_Root_Filesystem>

### SSH登录

使用SSH连接可以在通过网络，像串口终端一样，远程操作Linux命令行系统。

系统启动的时候已经默认启动 dropbear ssh 服务，只要使用SSH协议的软件，输入IP进行连接即可，如下图所示：

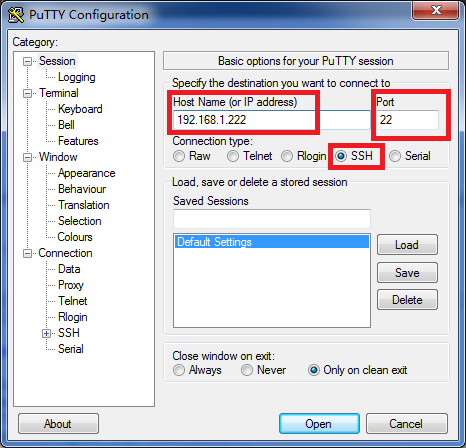


图5-1

在指定位置输入开发板IP，端口号，然后连接，如果弹出安全提示对话框点击YES忽略。在命令行界面输入登录和密码，默认状态登录名为root，密码为空，但某些文件系统密码不能为空，需要先在开发板上设置新密码才能进行SSH登录。

## 应用实例

MYD-JA5D4X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-gnueabihf-

$ 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 <program-to-be-execute>

### GPIO

本例介绍如何使用Linux命令行操作GPIO。

这里以GPIO5\_5为例，对应MYD-JA5D4X开发板扩展接口J4的Pin-6。

GPIO5\_5对应的Linux GPIO编号为GPIO165，即5×32+5=165。

导出GPIO 165：

# echo 165 > /sys/class/gpio/export

查看新导出的GPIO端口：

# ls /sys/class/gpio/gpio165

设置GPIO方向为输出：

# echo "out" > /sys/class/gpio/gpio165/direction

使GPIO口输出高电平：

# echo 1 > /sys/class/gpio/gpio165/value

使GPIO口输出低电平：

# echo 0 > /sys/class/gpio/gpio165/value

释放GPIO口：

# echo 165 > /sys/class/gpio/unexport

### 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，执行程序，程序将点亮LED0～LED3，按下开发板上的SW3、SW4按键将打印出相关的按键信息。

# ./key\_led

status led0 on

status led0 off

status led1 on

status led1 off

status led2 on

status led2 off

status led3 on

status led3 off

Hit any key on board ......

key 102 Released

key 102 Pressed

key 158 Released

key 158 Pressed

### NET

本例使用 TCP/IP sokect API，实现一个简单的C/S程序，详情请参考源码。将目录*<WORKDIR>/Examples/network*中的可执行程序arm\_client拷贝至开发板，pc\_server拷贝至PC，将开发板和PC接入网络，设置IP：

$ sudo ifconfig eth0 192.168.1.111

# ifconfig eth0 192.168.1.222

在PC上运行服务程序：

$ ./pc\_server

再在开发板上运行客户程序。将看到所发送的信息：

# ./arm\_client 192.168.1.111

form server: Make Your idea Real!

同时可以看到PC端看到连接的客户端IP：

$ ./pc\_server

REC FROM： 192.168.1.222

### 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。执行程序设置当前时间，例如：

# ./rtc\_test -s 11 03 40 2014 08 14

date/time is updated to: 14-8-2014, 11:03:40.

然后读取时间，观察时间变化：

# ./rtc\_test

Current RTC date/time is 14-8-2014, 11:03:50.

# ./rtc\_test

Current RTC date/time is 14-8-2014, 11:03:51.

### TWI

### 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U16中rs485对应的接口A、 B分别与另一个开发板U16中的 A、 B 相连；

将调试串口J12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；然后打开光盘，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：

# ./rs485 -d /dev/ttyO1 -b 9600

SEND:0123456789

RECV:0123456789, total:10

SEND:0123456789

RECV:0123456789, total:10

SEND:0123456789

RECV:0123456789, total:10

SEND:0123456789

RECV:0123456789, total:10

SEND:0123456789

RECV:0123456789, total:10

SEND:0123456789

RECV:0123456789, total:10

### RS232

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

将开发板上的串口J13与另一个开发板的串口J13相连，调试串口J12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。然后打开光盘，将目录*<WORKDIR>/Examples/rs232*中的可执行程序rs232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：

# ./rs232 -d /dev/ttyO2 -b 115200

SEND:0123456789

RECV:0123456789, total:10

SEND:0123456789

RECV:0123456789, total:10

SEND:0123456789

RECV:0123456789, total:10

SEND:0123456789

RECV:0123456789, total:10

SEND:0123456789

RECV:0123456789, total:10

SEND:0123456789

RECV:0123456789, total:10

### 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。

打开光盘，将目录*<WORKDIR>/Examples/audio*中的可执行程序audio拷贝至开发板，把调试串口J12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。将耳机插到音频输出口J10，将音频输入到音频输入口J11，执行audio 程序开始录音，按Ctrl-c结束录音并开始播放刚才的所录的声音，如下所示：

# ./audio

recording... press 'Ctrl-c' to stop

record stopped, playing...

play done!

### EEPROM

本例程演示如何使用EEPROM驱动，对开发板上的EERPOM进行读写操作，详情请参考源码。

禁用EEPROM写保护：

# echo 103 > /sys/class/gpio/export"

# echo "out" > /sys/class/gpio/gpio103/direction"

# echo 0 > /sys/class/gpio/gpio103/value"

注：假如gpio103已经被export，则以上第一条export命令将会失败，但不影响程序运行，继续往下操作即可。

执行程序：

# ./eeprom\_test

eeprom size: 32 KB

write 'eeprom write/read test!' to eeprom

get the following string from eeprom:

eeprom write/read test!

### FrameBuffer

本例演示对Linux的FrameBuffer操作，实现颜色和彩色栅格测试。

拷贝*<WORKDIR>/Examples/framebuffer*目录下的测试程序开发板，如果使用的是matrix图形系统，使用下面的命令关闭matrix界面：

# /etc/init.d/matrix-gui-2.0 stop

执行程序：

# ./framebuffer\_test

终端将显示屏幕信息，显示设备会先后出现8种背景色，然后显示颜色栅栏，以下是HDMI输出信息：

The framebuffer device was opened successfully.

vinfo.xres=1024

vinfo.yres=768

vinfo.bits\_per\_bits=32

vinfo.xoffset=0

vinfo.yoffset=0

finfo.line\_length=4096

The framebuffer device was mapped to memory successfully.

......

使用下面的命令可重启matrix界面：

# /etc/init.d/matrix-gui-2.0 start

### Led\_play

本例演示如何在用户空间读取按键以及如何控制用户led灯。

拷贝*<WORKDIR>/Examples/led\_play*目录下的测试程序led\_play到开发板，在开发板的控制终端上执行程序：

# ./led\_play

长按按键SW3三秒即可触发，触发后三个LED灯：D24，D25和D26开始闪烁。详细操作过程请见源码。

注：led\_play已嵌入到matrix图形系统并设置为自启动，如果使用该文件系统则可跳过拷贝和执行步骤，直接长按按键SW3三秒即可触发。如需取消led\_play的自启动，删除matrix文件系统下的/etc/rc5.d/S99led\_play文件即可。

# Qt开发

本小节描述在MYD-JA5D4X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。

## 使用光盘提供的Qt SDK

⑴ 解压编译好的tslib到PC：

$ sudo tar xvjf \

/media/cdrom/05-Linux\_Source/Qt\_Arm/tslib-prebuild.tar.bz2 -C /opt

⑵ 解压Qt SDK到PC：

$ sudo tar xvjf \

/media/cdrom/05-Linux\_Source/Qt\_Arm/qt-4.8.5-sdk.tar.bz2 -C /opt

⑶ 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](#_1.10.4_交叉编译Qt应用程序)。

## 交叉编译Qt开发环境

⑴ 参考本文之前章节安装交叉编译工具，设置环境变量。

⑵ 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

$ sudo 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/opt*/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：

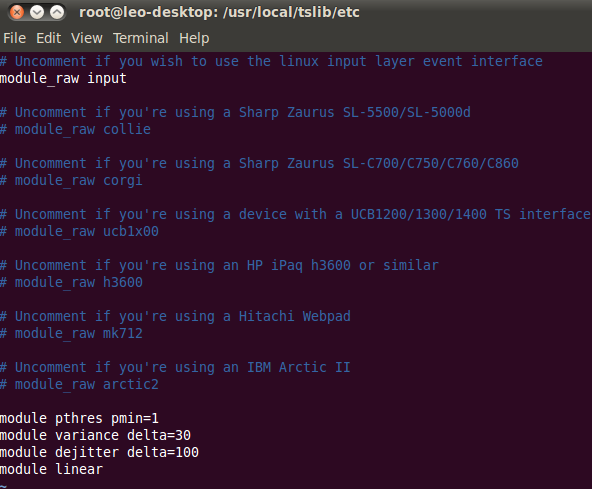


图 6‑1

⑶ 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。

## 移植Qt到开发板

⑴ 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/usr/local/目录下：

$ cp -ra /opt/tslib <WORKDIR>/Filesystem/rootfs/opt/

⑵ 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rootfs文件夹的opt/目录下：

$ cp -r /opt/qt-4.8.5 <WORKDIR>/Filesystem/rootfs/opt/

⑶ 进入<WORKDIR>/Filesystem/rootfs/etc/init.d/目录，编写脚本qt.sh，如下：

export TSLIB\_CONSOLEDEVICE=none

export TSLIB\_FBDEVICE=/dev/fb0

export TSLIB\_TSDEVICE=/dev/input/touchscreen0

export LD\_LIBRARY\_PATH=/lib:/usr/lib:/opt/tslib/lib:/opt/qt-4.8.5/lib

export QT\_QWS\_FONTDIR=/opt/qt-4.8.5/lib/fonts

export QWS\_USB\_KEYBOARD=/dev/input/event2

export PATH=/bin:/sbin:/usr/bin/:/usr/sbin:/opt/tslib/bin

if grep "display1" /proc/cmdline > /dev/null ; then

export QWS\_MOUSE\_PROTO="Tslib:/dev/input/touchscreen0 MouseMan:/dev/input/mice"

else

export QWS\_MOUSE\_PROTO="Tslib:/dev/input/touchscreen0"

fi

export QWS\_DISPLAY=:1

修改<WORKDIR>/Filesystem/rootfs/etc/init.d/目录下的rcS文件，在该文件最后加上：

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

/etc/init.d/qt.sh

fi

⑷ 然后重新制作烧写包含Qt库和tslib的文件系统。

## 交叉编译Qt应用程序

⑴ 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++

⑵ 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();

}

⑶ 编译：

$ qmake -project

$ qmake

$ make

⑷将生成的hellomyir拷贝到开发板，并在开发板上执行：

# ./hellomyir

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。

# 附录一 联系方式

销售联系方式

* **网址：** [www.myir-tech.com](http://www.myir-tech.com)
* **邮箱：**[sales.cn@myirtech.com](mailto:sales.cn@myirtech.com)

深圳总部

* 负责区域：广东 / 四川 / 重庆 / 湖南 / 广西 / 云南 / 贵州 / 海南 / 香港 / 澳门
* 电话：0755-25622735 0755-22929657
* 传真：0755-25532724
* 邮编：518020
* 地址：深圳市罗湖区文锦北路1010号文锦广场文盛中心1306

上海办事处

* 负责区域：上海 / 湖北 / 江苏 / 浙江 / 安徽 / 福建 / 江西
* 电话：021-60317628 15901764611
* 传真：021-60317630
* 邮编：200062
* 地址：上海市普陀区中江路106号北岸长风I座1402

北京办事处

* 负责区域：北京 / 天津 / 陕西 / 辽宁 / 山东 / 河南 / 河北 / 黑龙江 / 吉林 / 山西 / 甘肃 / 内蒙古 / 宁夏
* 电话：010-84675491 13269791724
* 传真：010-84675491
* 邮编：102218
* 地址：北京市昌平区东小口镇中滩村润枫欣尚2号楼1009

技术支持联系方式

* **电话：**0755-25622735

**邮箱：**[support@myirtech.com](mailto:support@myirtech.com)

# 附录二 售后服务与技术支持

凡是通过米尔科技直接购买或经米尔科技授权的正规代理商处购买的米尔科技全系列产品，均可享受以下权益：

1、6个月免费保修服务周期

2、终身免费技术支持服务

3、终身维修服务

4、免费享有所购买产品配套的软件升级服务

5、免费享有所购买产品配套的软件源代码，以及米尔科技开发的部分软件源代码

6、可直接从米尔科技购买主要芯片样品，简单、方便、快速；免去从代理商处购买时，漫长的等待周期

7、自购买之日起，即成为米尔科技永久客户，享有再次购买米尔科技任何一款软硬件产品的优惠政策

8、OEM/ODM服务

**如有以下情况之一，则不享有免费保修服务：**

1、超过免费保修服务周期

2、无产品序列号或无产品有效购买单据

3、进液、受潮、发霉或腐蚀

4、受撞击、挤压、摔落、刮伤等非产品本身质量问题引起的故障和损坏

5、擅自改造硬件、错误上电、错误操作造成的故障和损坏

6、由不可抗拒自然因素引起的故障和损坏

**产品返修：**用户在使用过程中由于产品故障、损坏或其他异常现象，在寄回维修之前，请先致电米尔科技客服部，与工程师进行沟通以确认问题，避免故障判断错误造成不必要的运费损失及周期的耽误。

**维修周期：**收到返修产品后，我们将即日安排工程师进行检测，我们将在最短的时间内维修或更换并寄回。一般的故障维修周期为3个工作日（自我司收到物品之日起，不计运输过程时间），由于特殊故障导致无法短期内维修的产品，我们会与用户另行沟通并确认维修周期。

**维修费用：**在免费保修期内的产品，由于产品质量问题引起的故障，不收任何维修费用；不属于免费保修范围内的故障或损坏，在检测确认问题后，我们将与客户沟通并确认维修费用，我们仅收取元器件材料费，不收取维修服务费；超过保修期限的产品，根据实际损坏的程度来确定收取的元器件材料费和维修服务费。

**运输费用：**产品正常保修时，用户寄回的运费由用户承担，维修后寄回给用户的费用由我司承担。非正常保修产品来回运费均由用户承担。