## Openwrt版本

版本名称是Barrier Breaker 版本号是14.07

## Openwrt编译环境

编译虚拟机在百度网盘（使用vmware虚拟机）：

链接：https://pan.baidu.com/s/1Ijqsl9xn-6zRkM\_LKq22RA

提取码：gpxg这个是 vmware的磁盘镜像 虚拟机的用户名hlk 密码：12345678

链接：https://pan.baidu.com/s/1RqU9sRh2iZZ-Y3WNPFypdg

提取码：55sy这个是 virtualbox的磁盘镜像 虚拟机的用户名hlk 密码：12345678

虚拟机镜像解压缩后需要使用VirtualBox导入，VirtualBox软件从https://www.virtualbox.org/可以下载。s

虚拟机中自带了Openwrt 14.07 SDK源码，/home/hlk/mtkopenwrt, 已经有默认配置， 可以直接编译。

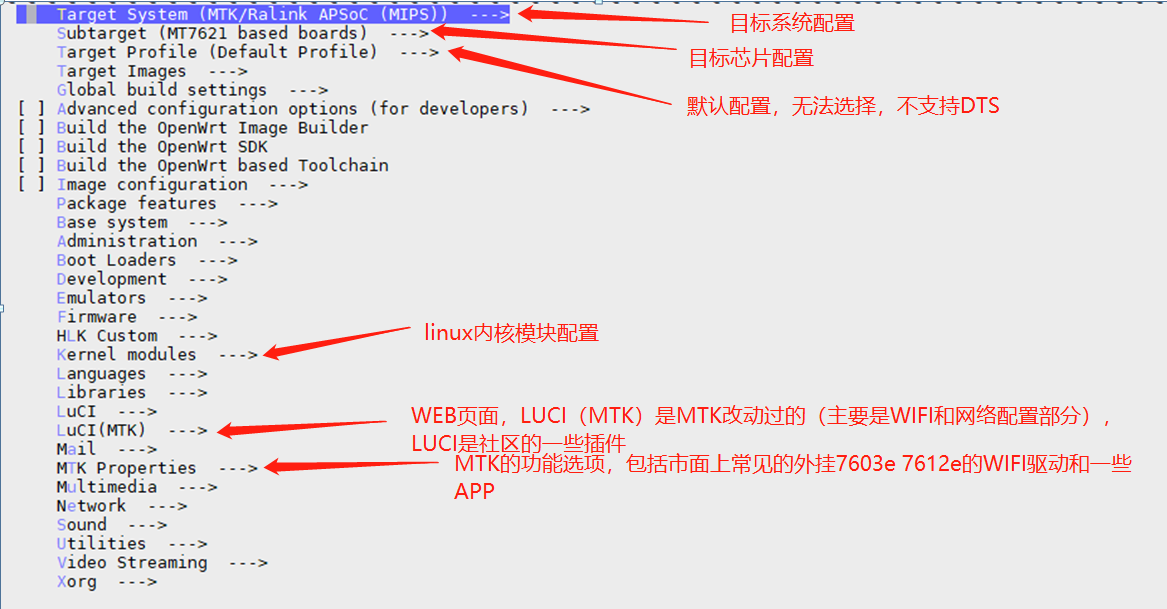
SDK源码包链接：https://pan.baidu.com/s/1X4Om05zAaNAOurfiGuUPxA

提取码：unhb

## Openwrt配置编译

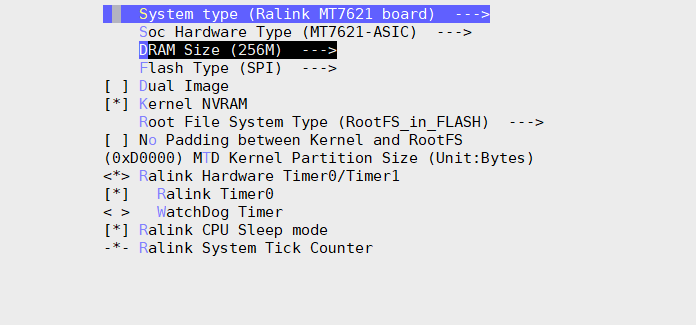
SDK中已经存在一个默认配置，满足路由的基本功能，客户也可以根据自己的需求，进行自定义配置

命令： make menuconfig



修改默认内存大小的配置

make kernel\_menuconfig ->Machine selection ---> DRAM Size (256M) --->



使用命令 make V=99

编译结果保存在bin/ramips/目录下

生成固件名：openwrt-ramips-mt7621-mt7621-squashfs-sysupgrade.bin

## 使用reg 命令控制7621的寄存器

## Openwrt的出厂配置恢复方法

在命令行输入:

umount /dev/mtdblock6; firstboot

firstboot 输入Y 确认恢复默认

Openwrt将会被清除已有的配置信息，恢复为默认出厂配置

## Openwrt中的网口配置

## 在openwrt中添加自己的应用并编译到固件中

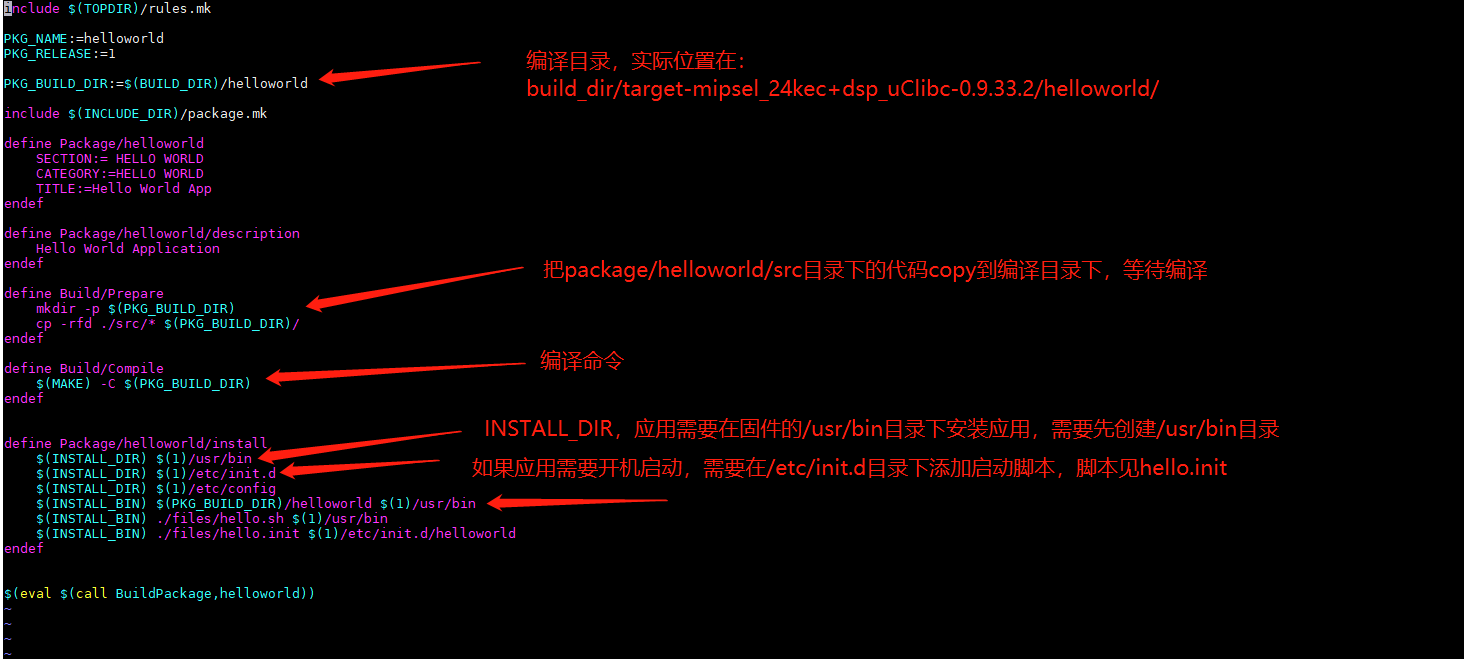
下面以helloworld示例如何在openwrt中添加一个应用

在package下创建一个目录 helloworld

在helloworld目录下创建Makefile文件：文件见附件：



Makefile的规则描述，详细文档可以参考：<https://openwrt.org/docs/guide-developer/packages>



示例代码打包见附件：

直接解压到openwrt的package目录下

Helloworld目录下的src目录是存放源代码的地方：

示例中src目录下的Makefile才是真正用来对代码进行编译的Makefile。

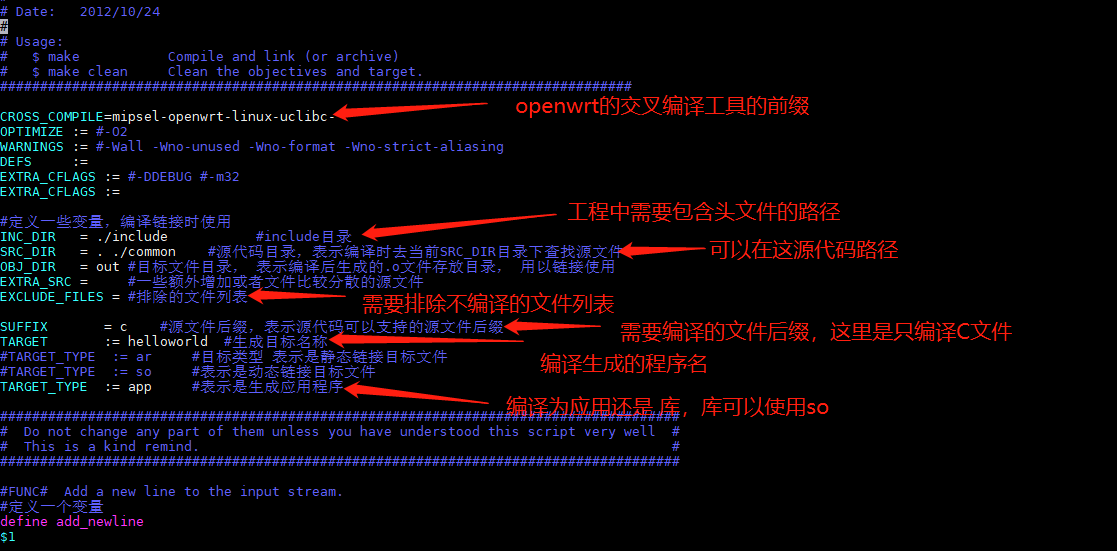
客户也可以根据自己的需求编写makefile ，需要注意的几个地方  
 a. openwrt编译工具的名字是：mipsel-openwrt-linux-uclibc-开头的交叉编译器

不能直接使用mipsel-openwrt-linux-uclibc-ld进行链接。

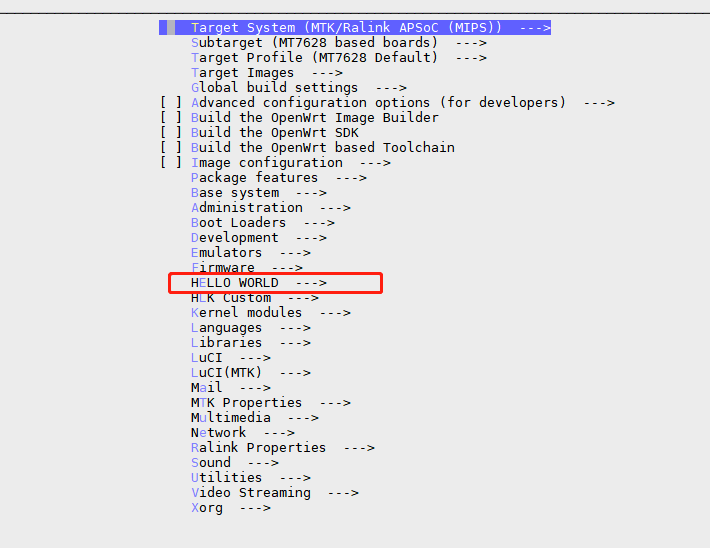
1. 如果使用示例中带的Makefile，对于简单的工程是可以的，它可以遍历src目录

所有的.c文件进行编译

Makefile的描述：



文件创建完成后，使用make menuconfig对openwrt进行配置：

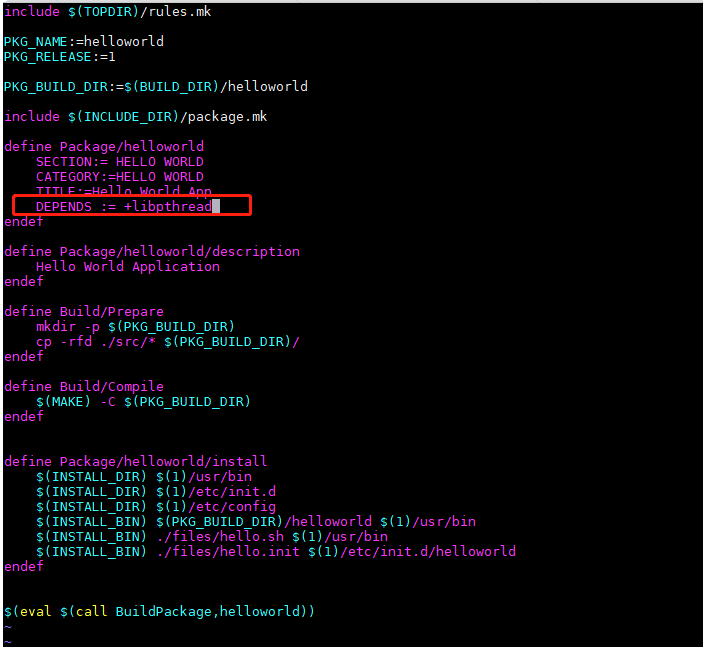


选中后 进入到工程目录下进行make 编译完成后将固件升级后

编译时库的依赖问题：

如果使用了pthread多线程库

可以在外层Makefile中添加如下：



还有一种方法，可以在编译时欺骗openwrt的编译过程

在Makefile中添加：

define Package/helloworld/extra\_provides

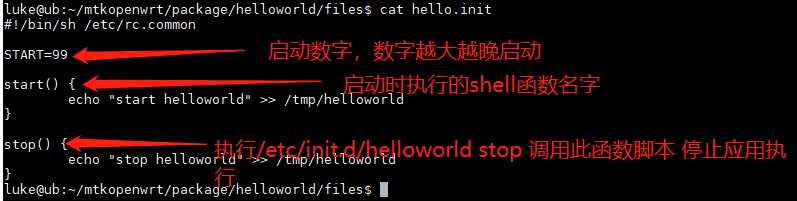
echo "libpthread.so.0"

这样在编译时，就不会出现因为缺少某个库而报错了。但应用实际是运行不了的，需要把相应的库copy到系统库目录下才可以。

上电启动脚本：

Openwrt的上电执行/etc/init.d下的脚本 并按照脚本中的START变量大小依次执行。

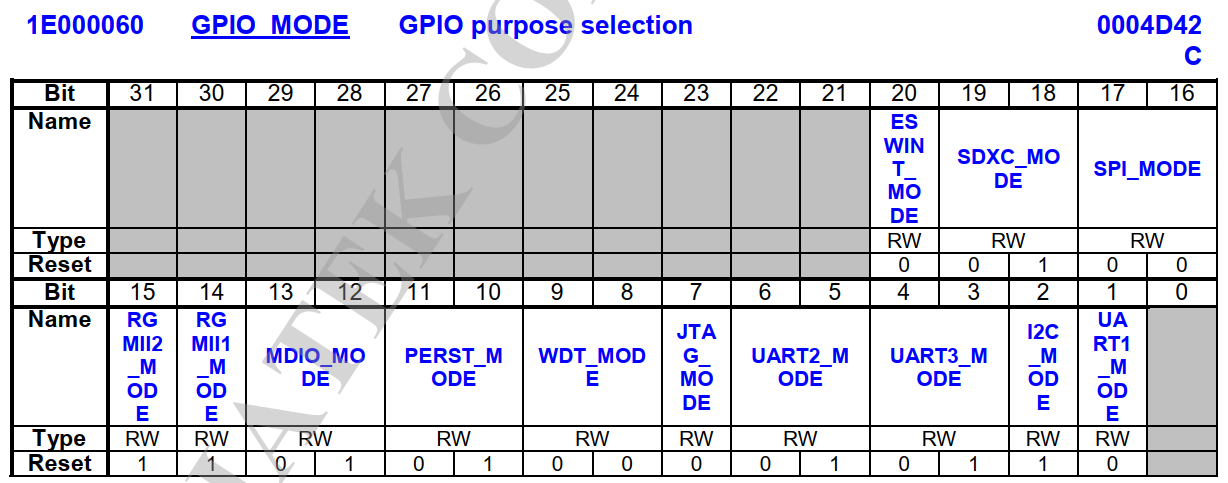
启动脚本的写法参见：



## 控制GPIO

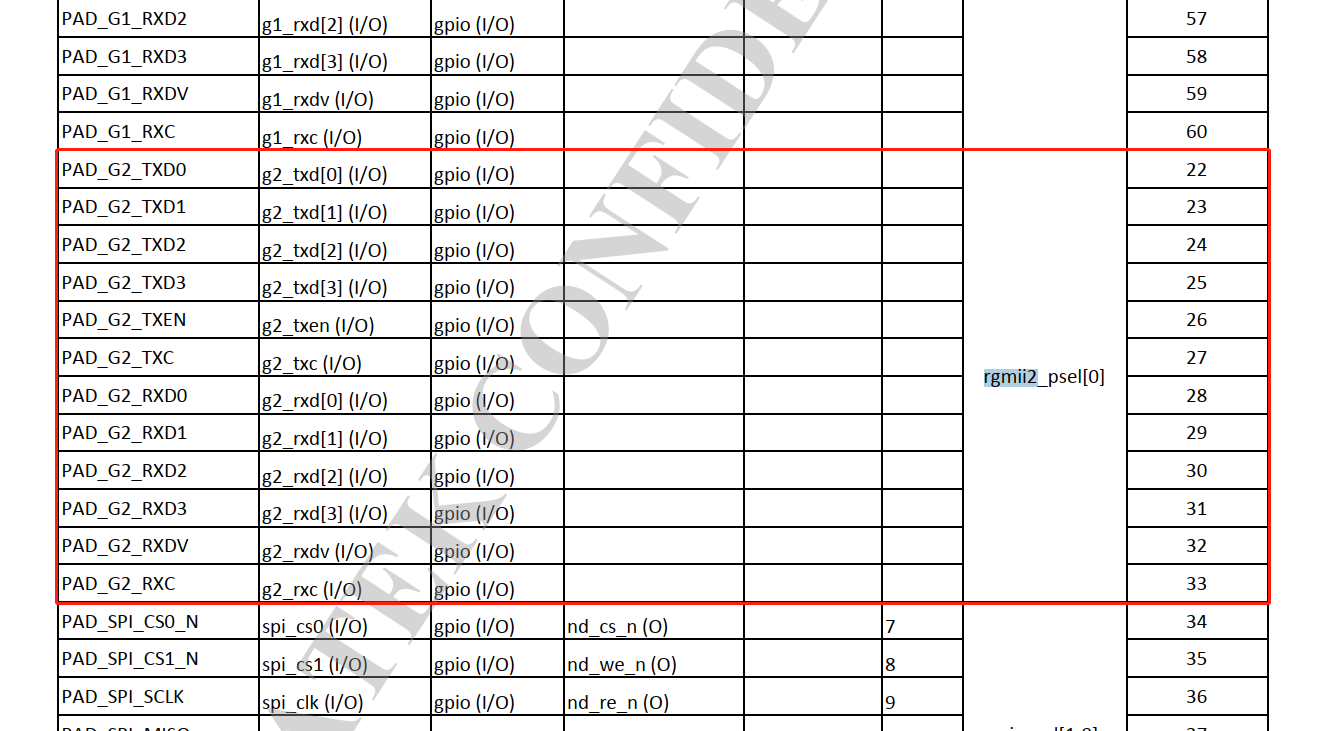
M6721 引脚的 GPIO模式寄存器GPIO\_MODE

见datasheet的32页，可通过控制寄存器的相应bit 来控制引脚是否作为GPIO使用。

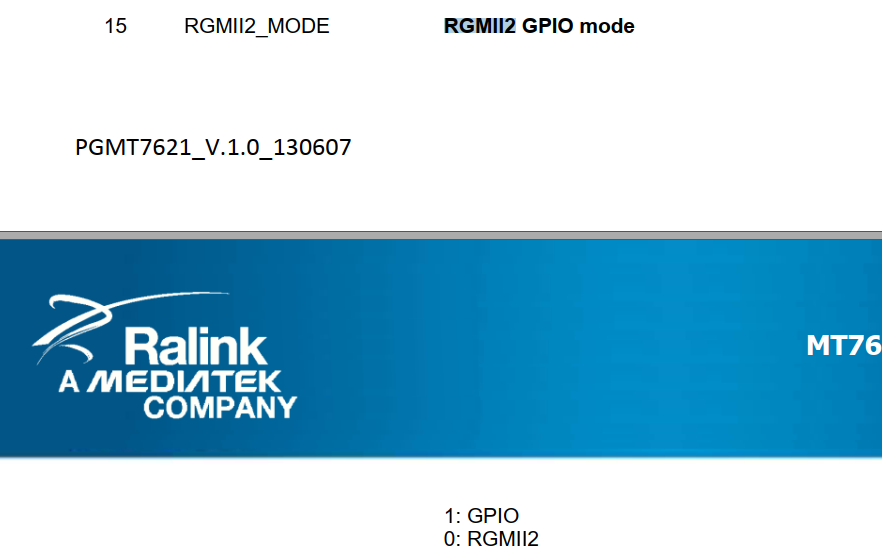


这里已RGMII2的引脚作为示例

datasheet的 66页可以看到 RGMII2\_MODE寄存器实际可以控制 12个引脚，



它在GPIO\_MODE寄存器中的bit 15 RGMII2\_MODE, 把RGMII2\_MODE设置为1时 ，上述的12个引脚既可作为GPIO引脚使用。

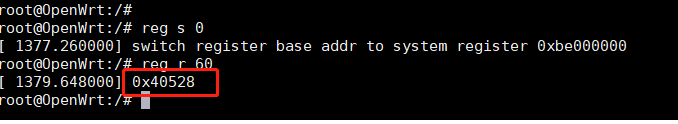


控制方法：

1. 使用reg命令进行控制
2. 将引脚配置为GPIO模式

登陆串口终端，在命令行中使用 reg s 0; reg r 60 读取GPIO\_MODE寄存器的值

固件默认 RGMII2配置为了RGMII2模式，bit15 为0



将bit15 设置为1

通过命令： reg w 60 0x48258

执行完成后：reg r 60 查看是否配置成功

1. 配置GPIO的输入输出方式

把引脚配置为GPIO模式后， 还需要相应GPIO 是输入还是输出

这里MT7621和MT7688是同样的配置寄存器 可以参考MT7688的DataSheet中5.8节 General Purpose IO

GPIO控制寄存器如下图所示：

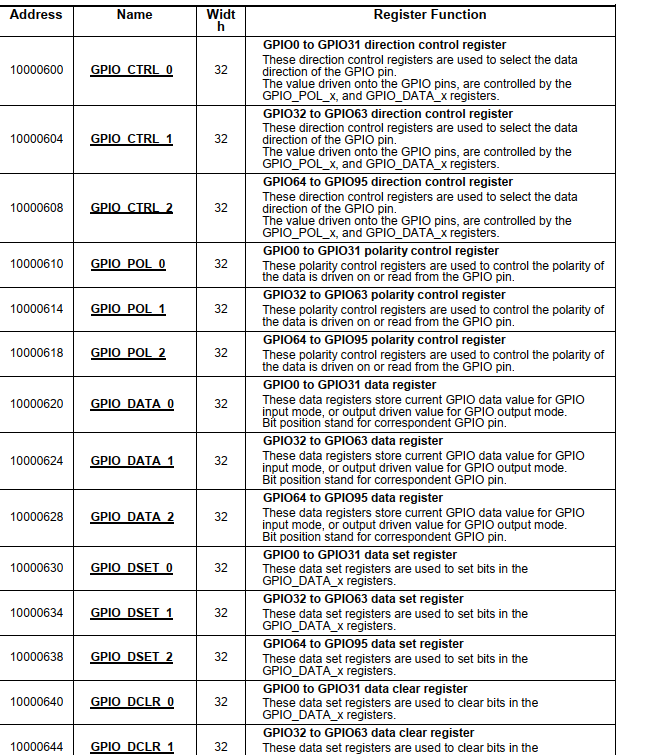
GPIO\_CTRL\_X：是 GPIO方向控制寄存器 其中X=0/1/2 每个寄存器控制32个GPIO

GPIO\_POL\_X： 是GPIO极性控制寄存器 X同上

GPIO\_DATA\_X：是GPIO数据寄存器，可以通过读取这个寄存器来获取当前GPIO的值

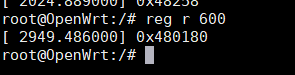
GPIO\_DSET\_X：是GPIO设置寄存器，用以控制相应的GPIO的值是否为1，置1为有效控制

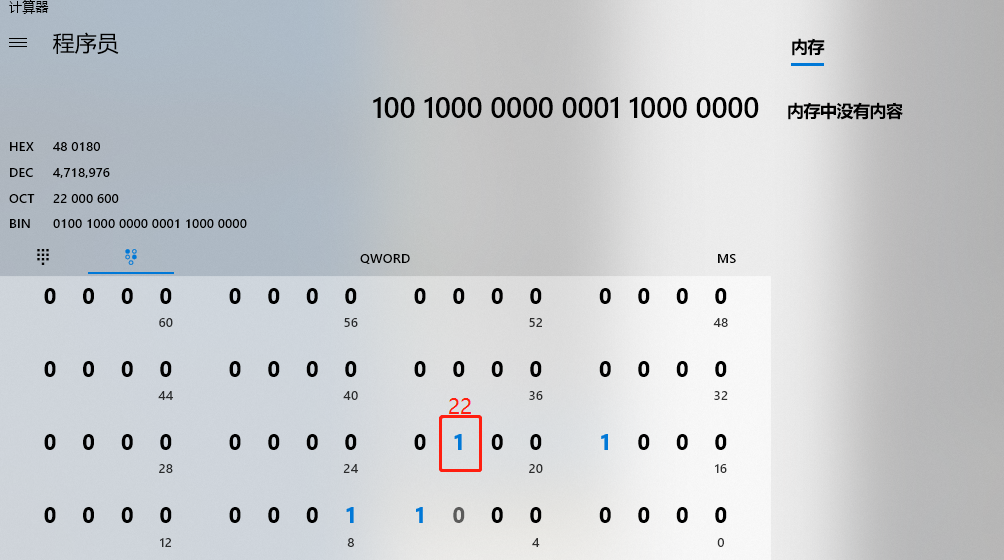
GPIO\_DCLR\_X：是GPIO清除寄存器，用以控制相应的GPIO的值是否为0，置1为有效控制



示例：

控制GPIO22作为GPIO输出引脚，GPIO22在0~31范围内，使用 GPIO\_XXX\_0的寄存器进行控制

读取GPIO\_CTRL\_0寄存器：reg r 600 

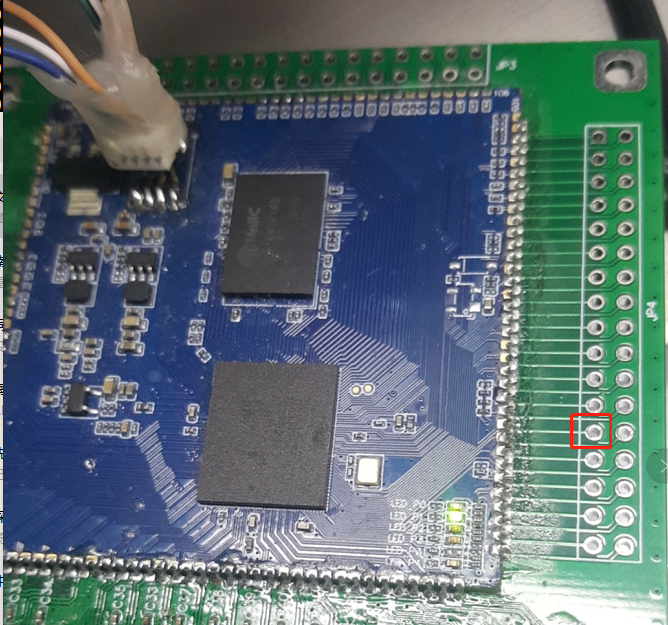


如果bit22是 1 表示 GPIO22 被设置为了GPIO输出模式

通过HLK7621的用户手册可以查到是80pin脚



在开发板上的位置：



设置为高电平：reg w 630 400000

设置为低电平：reg w 640 400000

演示视频：



其他GPIO控制方式 和 RGMII2的控制方式类似。

## SSH登陆问题

Mtkopenwrt 默认使用的ssh时dropbear，不能无密码登陆

1. 可以修改package/base-files/files/etc/shadow文件，把设置了root密码的shadow文件替换掉
2. 启动后使用passwd设置密码

密码设置完成后 ，就可以使用ssh登陆了

## VLAN设置

## 网口设置

### 单WAN口

### 多WAN口

### 无WAN口

### 多网桥配置

## 支持SD卡

需要配置openwrt

make menuconfig

驱动支持

MTK Properties --->

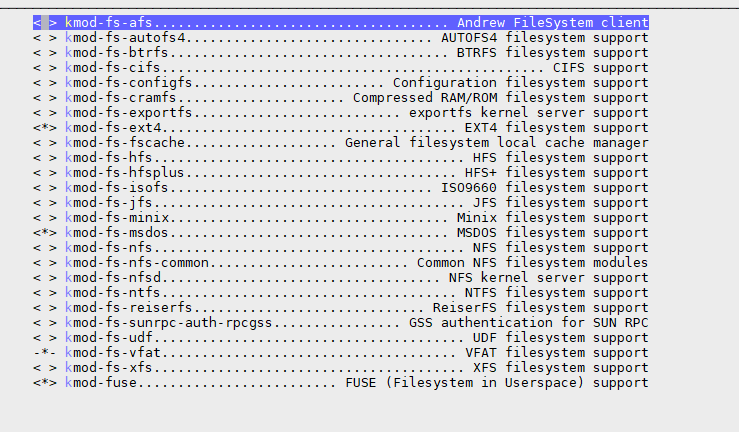
Drivers --->

<\*> kmod-mtk-mmc......................................... MMC/SD card support

配置文件系统支持：

Kernel modules --->

Filesystems --->



## 支持网络共享samba

对openwrt做配置make menuconfig

Kernel modules --->

Filesystems --->

<\*> kmod-fs-cifs................................................ CIFS support

配置LUCI页面：

LuCI --->

Applications --->

<\*> luci-app-samba.................... Network Shares - Samba SMB/CIFS module

配置samba

MTK Properties --->

Applications --->

<\*> samba-server................................................ Samba Server

修改配置文件：package/network/services/samba36/files/samba.config

config samba

option 'name' 'OpenWrt'

option 'workgroup' 'WORKGROUP'

option 'description' 'OpenWrt'

option 'homes' '1'

option 'interface' 'wan wwan lan' #允许访问网口

config sambashare

option browseable 'yes'

option name 'Share'

option path '/mnt/sda1'

option users 'root,nobody' #允许用户

option read\_only 'no'

option guest\_ok 'yes'

option create\_mask '0700'

option dir\_mask '0700'

## 支持UVC网络摄像头

## 策略路由

## 静态路由

## 防火墙设置

## 串口使用

## 添加SFTP服务

添加sftp配置 使用make menuconfig

Network --->SSH ---><\*> openssh-sftp-server.................................. OpenSSH SFTP server

## UCI语法

UCI配置分为两种 有名和匿名

### 匿名配置添加方法：

比如wifi接口的配置就属于匿名配置：

uci add wireless wifi-iface

uci set [wireless.@wifi-iface[-1].device=mt7628](mailto:wireless.@wifi-iface[-1].device=mt7628)

uci set [wireless.@wifi-iface[-1].ifname=ra1](mailto:wireless.@wifi-iface[-1].ifname=ra1)

uci set [wireless.@wifi-iface[-1].network=lan1](mailto:wireless.@wifi-iface[0].network=lan1)

uci set [wireless.@wifi-iface[-1].mode=ap](mailto:wireless.@wifi-iface[0].mode=ap)

uci set [wireless.@wifi-iface[-1].ssid=HLK-A905](mailto:wireless.@wifi-iface[0].ssid=HLK-A905)

uci set [wireless.@wifi-iface[-1].encryption=psk2](mailto:wireless.@wifi-iface[0].encryption=psk2)

uci set [wireless.@wifi-iface[-1].key=12345678](mailto:wireless.@wifi-iface[-1].key=12345678)

### 有名配置添加方法：

比如network中的网络接口配置就属于有名配置：

uci set network.lan1=interface

uci set network.lan1.ifname=eth0.1

uci set network.lan1.force\_link=1

uci set network.lan1.type=bridge

uci set network.lan1.proto=static

uci set network.lan1.ipaddr=192.168.17.254

uci set network.lan1.netmask=255.255.255.0

## WEB

Mtkopenwrt有一个默认的LUCI web界面

LuCI(MTK) --->1. Collections ---><\*> luci 勾选上即可，不需要的话不用勾选

## 添加串口登陆功能

## 调试串口添加密码登陆

为了安全性考虑，进行串口登入的时候也希望像ssh那样要求输入用户名和密码才能进入控制台。

一、客制化busybox

make menuconfig

Base system --->

  <\*> busybox ......

         [\*] Customize busybox option

               Login/Password Management Utilities --->

                     [\*] login (NEW)

二、修改启动脚本

vim /etc/inittab

::askconsole:/bin/ash --login

改为

::askconsole:/bin/login

三、root密码默认设置

package/base-files/files/etc/shadow

可以通过webui设置好，查看/etc/shadow文件，再写到源码。

注：系统启动会执行到/etc/inittab，最后一行::askconsole:/bin/ash --login

/bin/ash 附带 --login 参数, ash 则会在进入 cmdloop 之前, 先去载入 /etc/profile

————————————————

版权声明：本文为CSDN博主「zjf30366」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/zjf30366/article/details/85000690

## 使用调试串口作为普通串口

需要关闭调试信息：

修改mtkopenwrt的内核代码：

内核补丁：

## USB口

# USB2.0的1-4Pin的定义和USB3.0一致,不同的是增加了2对TX,RX信号线及1根Gnd线，后部分的5根Pin脚才是USB3.0的关键Pin脚

# 当在主板上的USB3.0母口上插入USB3.0设备时，公口前端的4 pin(USB2.0)信号线会与母口的4 pin(USB2.0)信号线先接触，接着再是5pin的信号线接触，

# 如果系统侦测设备的短暂时间里，如果我们还没有让后5Pin(USB3.0)信号线接通的话，设备就被系统判定为USB2.0设备了

# 即：设备插入时间>系统识别设备的时间，则USB3.0设备被系统提前识别为USB2.0设备

# 设备插入时间指的是从设备插入开始到后5Pin信号线接通之间的一小段非常短暂的时间

# 疑问：设备只是开始插慢了一点，但是设备最后还是被完全插入了，为何设备还是被认成USB2.0设备呢，这是因为，系统把设备侦测为USB2.0设备后，

# 当公口，母口的USB2.0 pin脚接触后，没有再次启动设备侦测，我们只需要重启OS，或者在设备管理器下disable USB3.0 Controller，再Enable USB3.0 Controller即可 通过这样的操作后，我们可以通过运行HD-Tach来验证突其发传输速度，可以达到100MB/s以上

USB3.0 口

使用USB3.0 U盘的测试数据

写入100M数据的时间：7.17 6.95 6.08 7.22 8.32 4.45 4.50 5.68 7.58 5.73 4.12 6.72 3.96 9.45 5.65 4.98 5.23 6.26 7.03 4.83 7.21 4.50 5.60 7.57 4.79 4.44 4.17 9.53 7.00 6.15 4.08 6.14

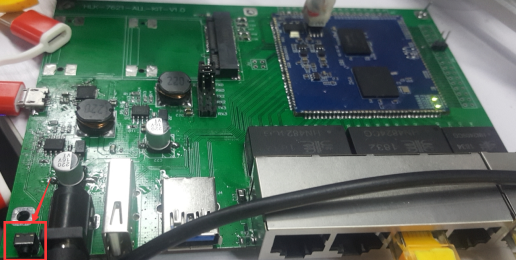
读取100M数据的时间：1.99 1.97 2.09 2.00 1.95 1.98 1.97 2.15 1.97 2.02 1.98 2.18 2.03 2.03 1.97 2.09 1.98 2.00 1.99 2.00 2.10 2.00 2.06 2.00 2.15 2.00 1.97 2.01 2.05 2.12 2.00 2.03

# 7621固件升级帮助指导

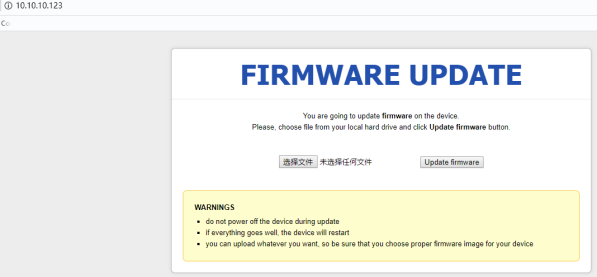
## 固件升级

固件有几种升级方式：

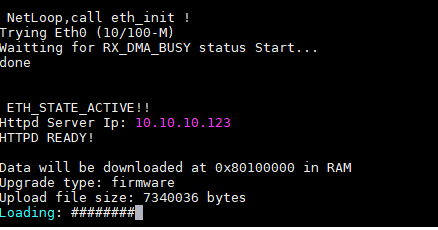
### 1. 启动时按住按键进行升级

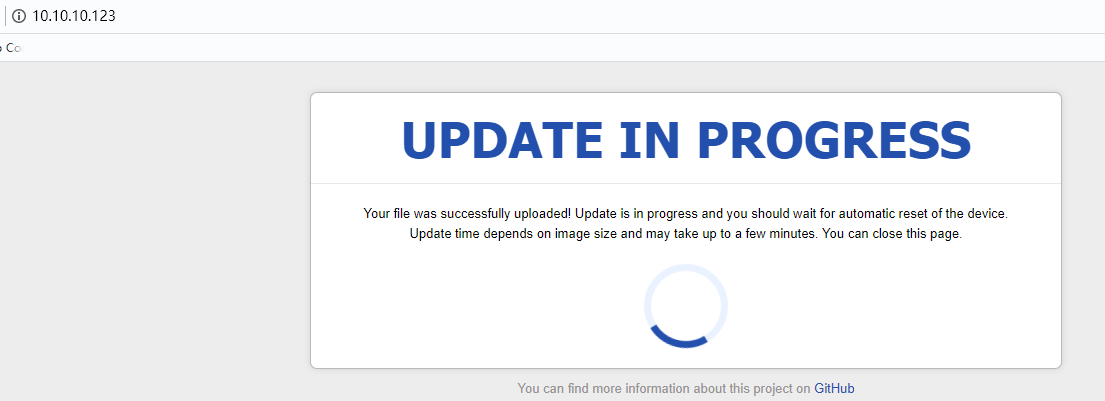


页面地址：10.10.10.123

升级页面如图所示：

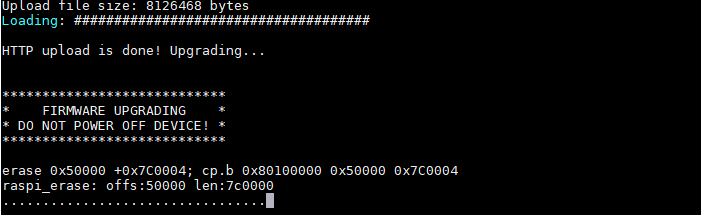
选择文件，然后点击Update Firmware ，注意，在上传固件时，页面并不会有变化，只能注意网口灯是不是再闪来判断有没有开始下载。

如果接了串口0 会显示

升级过程中会弹出 ，注意 这里直到升级完成重启 页面都不会再改变，

如果未插调试串口，只能通过ping 192.168.16.254 来判断固件有没有启动起来。

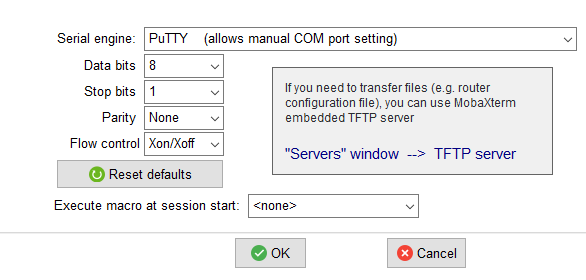
升级时通过调试串口可以查看进度

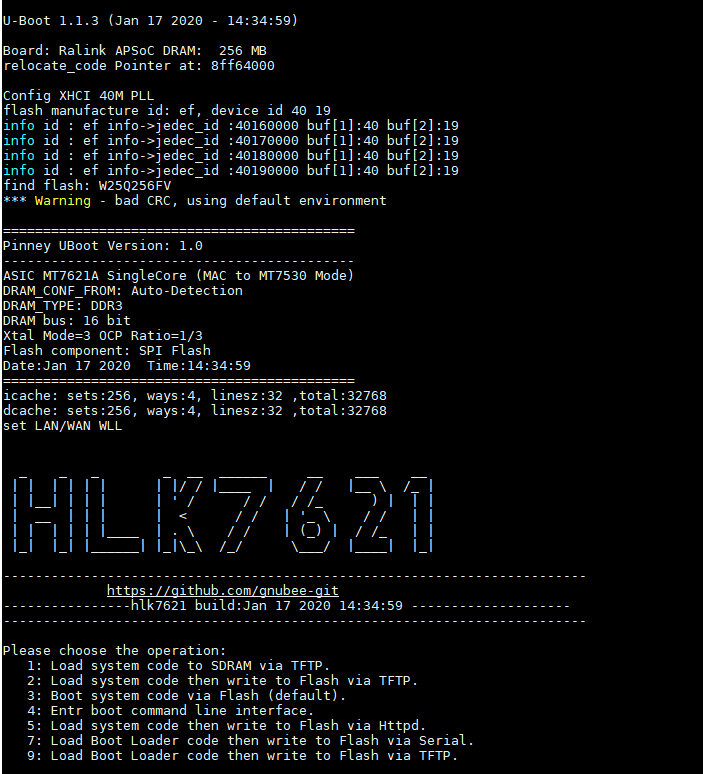


### 2. 通过串口命令行进行升级

需要接入调试串口：

用短接线把TX RX 连到USB转串口芯片上：芯片型号CP210X，使用USB线连接到模块的USB口上

串口配置

上电后出现打印：

会停顿3秒 ，让用户选择相应功能

### 3. 固件启动后，通过固件的LUCI页面进行升级

### 4. 固件启动后，通过命令行进行升级

## Openwrt固件版本

### 社区版本git地址： <https://github.com/openwrt/openwrt.git>

Openwrt v18.06.6 可以正常启动，网口，调试串口可以正常使用，其他功能未测试，社区版本的openwrt是不提供任何技术支持的。

配置见附件：

已编译好的固件：

添加到 openwrt SDK的目录下 文件名 改为.config

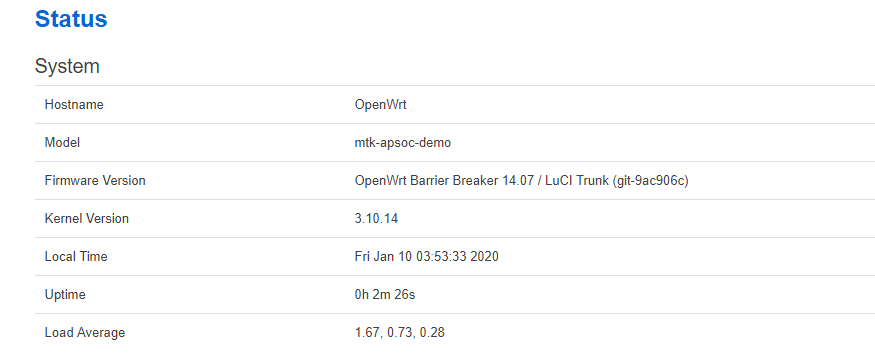
下载编译方法：

Git clone <https://github.com/openwrt/openwrt.git;cd> openwrt; git checkout v18.06.6; ./scripts/feeds update -a;./scripts/feeds install –a

已编译好的固件：



### 2．Openwrt 14.07版本



已编译好的固件：

## Uboot

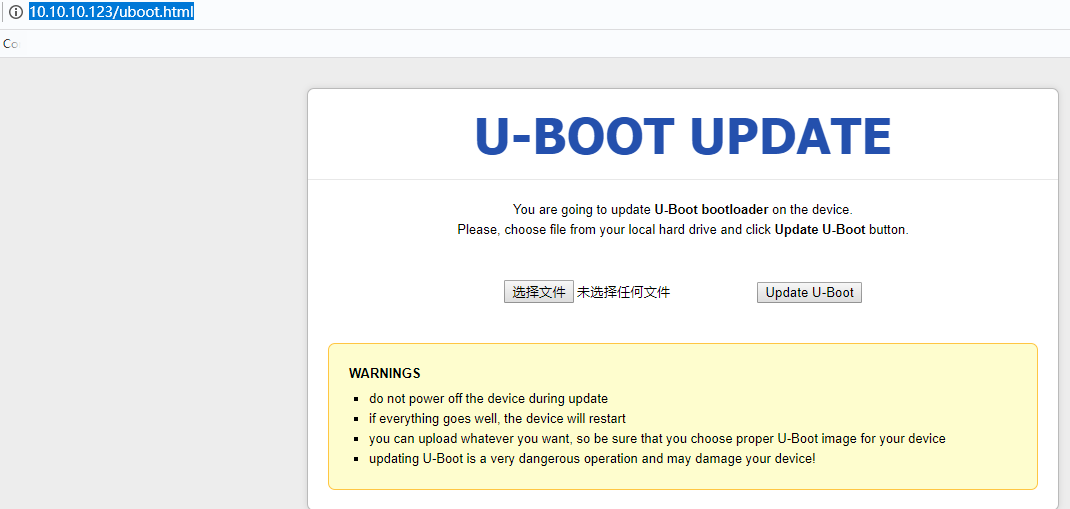
我司出厂的uboot源码是从github下载的开源版本，地址：<https://github.com/gnubee-git/GnuBee-MT7621-uboot.git>

已编译好的uboot： 

## Uboot升级方式：

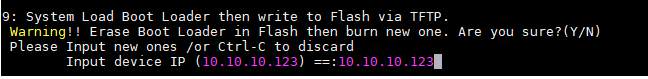
### 1.启动时按住WPS按键，uboot进入升级模式：

注意：PC端连接模块的网口的IP需要设置一个和10.10.10.123同网段的IP

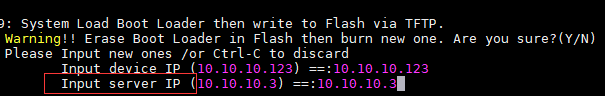
页面地址：<http://10.10.10.123/uboot.html> 

### 2.通过串口命令行进行升级

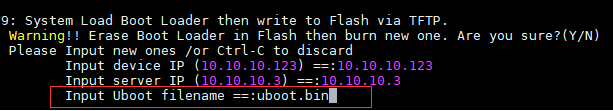
键入数字9 进入到 TFTP升级uboot模式：

键入Y 进入 参数配置：

Input device IP : 配置7621模块的IP

回车：

Input Server IP :配置连接到7621的PC网口上的IP



Input Uboot filename： 输入需要升级的uboot的文件名称

### 3. 旧版本出厂的7621硬件升级

旧版本出厂的7621 中flash中带有一个uboot，但是无法通过键入数字选择 手动进入到固件升级模式

只能通过再启动时按住按键 5秒 使uboot进入固件TFTP升级模式

PC上必须的操作：

1. PC网口连接到模块的WAN网口 必须有10.10.10.3这个Ip地址
2. 把需要升级的固件的名称改为tim\_uImage
3. 打开TFTP server，把文件目录改为tim\_uImage固件所在的目录

# 7621的分区情况

## openwrt社区版本通过查看DTS文件：

这里使用的是AP-MT7621A-V60.dts，可以看到它的一个flash分区配置情况



### openwrt14.07版本：

这个版本无法通过DTS配置flash分区

只能通过修改源码的方式：

源码在 SDK的build\_dir/target-mipsel\_24kec+dsp\_uClibc-0.9.33.2/linux-ramips\_mt7621/linux-3.10.14/drivers/mtd/ralink目录下

# 7621 刷机flash镜像

