## Openwrt版本

版本名称是Barrier Breaker 版本号是14.07

## Openwrt编译环境

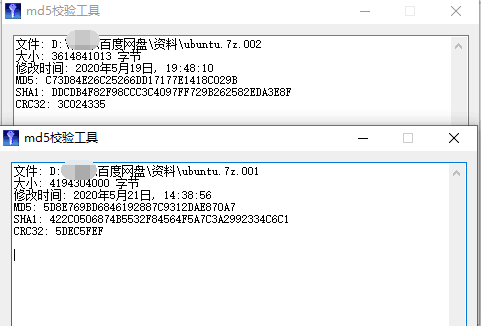
（1）编译虚拟机在百度网盘（使用vmware虚拟机）：

链接: https://pan.baidu.com/s/1fE5BrAIC8I3tjR1i6tv6cw 提取码: n86v

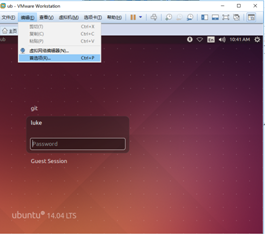
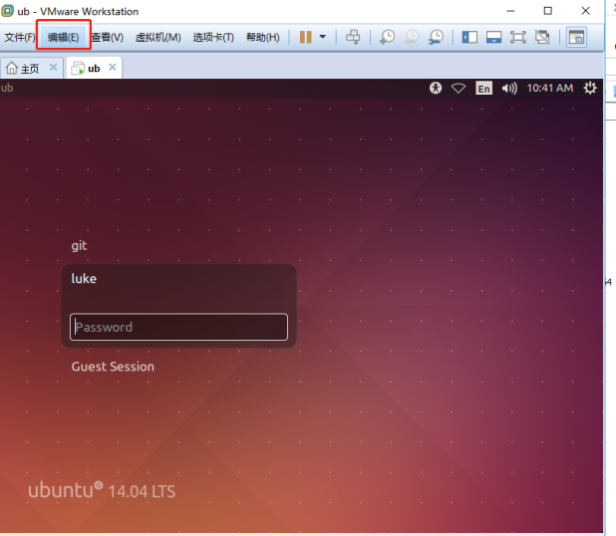
虚拟机的用户名luke 密码：luke

虚拟机版本 14.1.1

1. 如遇到下载完百度网盘中文件后，解压不了请使用md5工具对比压缩包大小。可正常解压运行的压缩包信息如下:

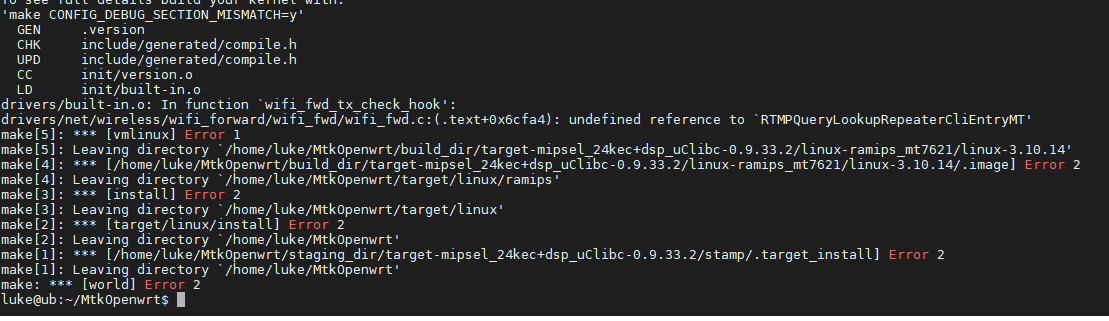


（3）如果遇到鼠标无法点击 可以进行下面的配置：

在

（4）虚拟机中自带了Mtk Openwrt的源码，/home/luke/MtkOpenwrt, 已经有默认配置， 可以直接编译。

编译时会遇到问题



make kernel\_menuconfig->

Ralink Module --->

[\*] WiFi Driver Support --->

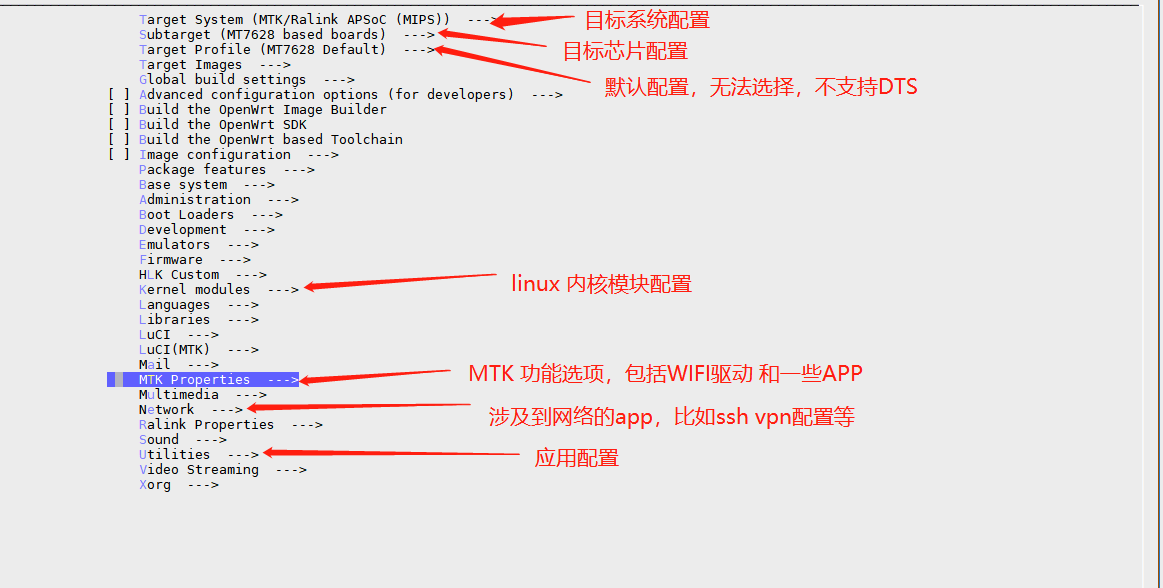
[] WiFi packet forwarding

把WiFi packet forwarding 的\*去掉 重新编译

## Openwrt配置编译

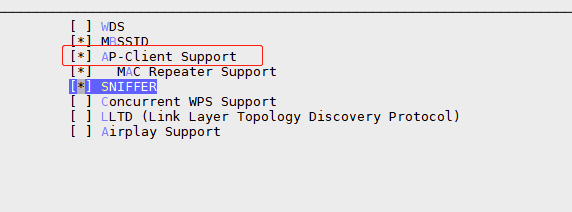
SDK中已经存在一个默认配置，满足路由的基本功能，客户也可以根据自己的需求，进行自定义配置

命令： make menuconfig



WIFI驱动配置在 MTK Properties-> Drivers --->kmod-mt7628 下

如需要STA功能可以选择：AP-Client Support



其他配置项，我们也没有尝试过，不了解具体功能

注意：

Kmod-mt7628sta 无法使用，请不要选择，如果需要sta功能 请选择kmod-mt7628驱动下的ap-client support



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-mt7628-mt7628-squashfs-sysupgrade.bin

## 使用reg 命令控制7688/7628的寄存器

MtkOpenwrt可以使用reg命令对7688/7628的寄存器进行配置

在需要在配置固件时加入reg命令：

make menuconfig ->MTK Properties->Applications-> <\*> reg 勾选上reg， 升级固件后重启。

1. reg s 0 设置寄存器地址的偏移量 0 表示 寄存器地址设置为MT7688的datasheet 第五章中描述的sysctl base address。
2. reg r 60 读取GPIO1\_MODE寄存器的值
3. reg w 60 xx 设置GPIO1\_MODE寄存器的值

示例：默认代码编译完成后 GPIO1\_MODE的寄存器默认设置UART1为GPIO模式

可以进行如下操作把UART1\_MODE设置为UART1-Lite模式

**reg s 60**

**reg r 60** 读取GPIO1\_MODE的寄存器值为0x55044410

通过读取7688的datasheet可以确认GPIO1\_MODE的24：25bit 是用于控制UART1的引脚的模式，0表示把UART1引脚配置为uart1-lite，1表示把UART1引脚配置为GPIO模式，使用命令 **reg w 60 0x54044410** （把24：25bit置为0），

验证UART1是否是串口：使用echo aaaaaaa >/dev/ttyS0(UART1 在系统中的设备)

其他命令选项，可以通过reg --help 学习用法

设置功能脚的引脚作为GPIO也是类似用法

使用reg 命令控制作为GPIO引脚的输入输出功能

通过reg命令可以控制寄存器使相应引脚进入到GPIO模式

reg命令 还用来控制GPIO引脚作为输入还是输出

示例1：GPIO0引脚默认既作为GPIO引脚，通过reg命令控制寄存器设置GPIO0引脚作为GPIO的输出模式并输出高低点评：(GPIO0的GPIO索引号是11)

reg s 0

参看MT7688\_Datasheet\_v1\_4.pdf手册的5.8一节，描述了7688的GPIO寄存器

GPIO\_CTRL\_0寄存器用以控制0~31号gpio的方向，设置为0表示输入，设置1表示输出

reg r 600 读取GPIO\_CTRL\_0寄存器的值为0

把gpio11设置为输出模式 可以使用命令 :

reg w 600 0x800

设置gpio11为高电平：写入GPIO\_DSET\_0寄存器的bit11控制gpio11为高电平

reg w 630 800

设置gpio11为低电平：写入GPIO\_DSET\_0寄存器的bit11 控制gpio11为低电平

reg w 634 800

设置引脚为GPIO的输入类似， 对于GPIO11 只需要控制GPIO\_CTRL\_0的bit11

为0即可

## 配置WIFI的STA功能

在这个版本的openwrt的WIFI驱动是一个AP+STA共存的apclient功能，源码对于APClient的支持目前仅限于命令行 和 配置文件，暂不支持在luci页面中进行配置，只能通过命令行对周围的AP进行扫描并配置。

扫描AP命令：iwpriv ra0 set SiteSurvey=1;sleep 3;iwpriv ra0 get\_site\_survey

配置STA

1. 通过修改/etc/config/wireless配置文件

config wifi-iface

option device mt7628

option ifname ra0

option network lan

option mode ap

option ssid mt7628-1912

option encryption psk2

option key 12345678

添加下面的配置参数

option ApCliEnable '1'

option ApCliSsid 'WIFI-mark'

option ApCliAuthMode 'WPA2PSK'

option ApCliEncrypType 'AES'

option ApCliWPAPSK '13590297795'

完成后 配置WAN口到apcli0

修改/etc/config/network

config interface 'wan'

option proto 'dhcp'

option ifname 'apcli0'

重启网络 /etc/init.d/network restart

## Openwrt的出厂配置恢复方法

在命令行输入:

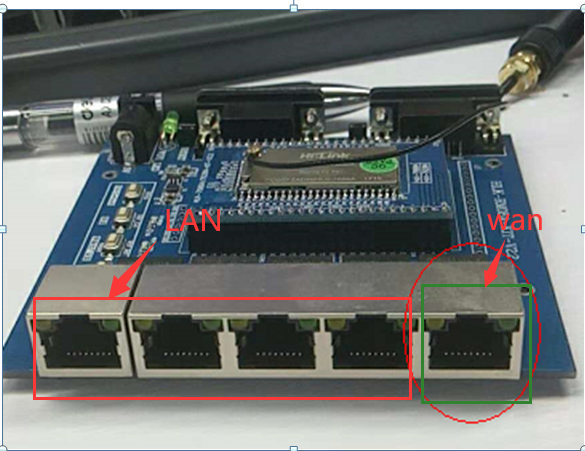
umount /dev/mtdblock6; firstboot

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

出厂固件默认的网口配置对照底板示意：



网口从左到右分别对应7688/7628的P0 P1 P2 P3 P4引脚

Openwrt下的所谓的LAN WAN口是通过脚本进行配置的

./build\_dir/target-mipsel\_24kec+dsp\_uClibc-0.9.33.2/switch/ipkg-ramips\_24kec/switch/lib/network/switch.sh中

setup\_switch()

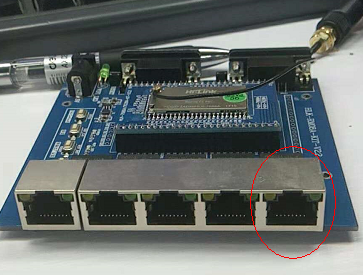
{

#configEsw LLLLW 配置P4为WAN口

configEsw WLLLL 配置P0为WAN口

}

LLLLW （P4为WAN口）在底板上的位置如图：



.

WLLLL (P0为WAN口)在底板上的位置如图：



如临时需将WAN口更换为LAN口 ， 可以单板上修改/lib/network/switch.sh文件，setup\_switch()

{

configEsw WLLLL 如果W在左边 则改到右侧， 如果在右侧则改到左侧

}

## 在openwrt中添加自己的应用并编译到固件中

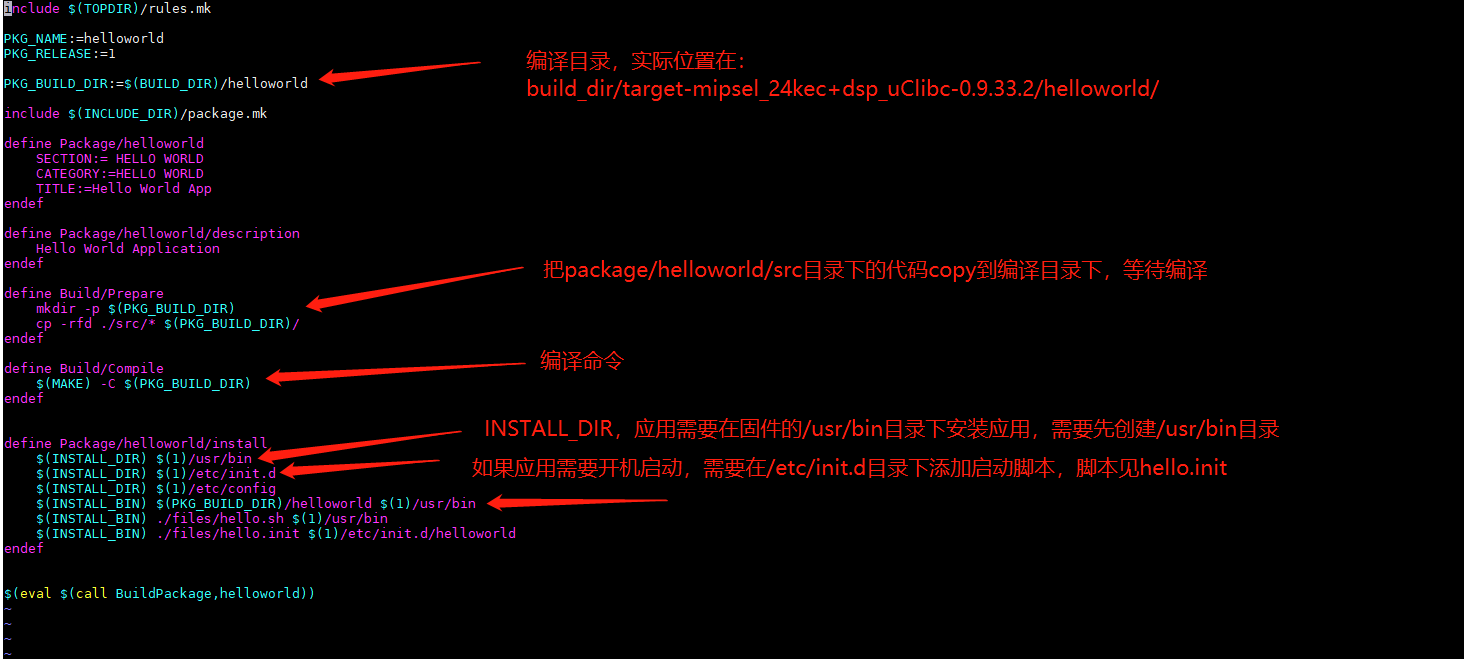
下面以helloworld示例如何在openwrt中添加一个应用

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。

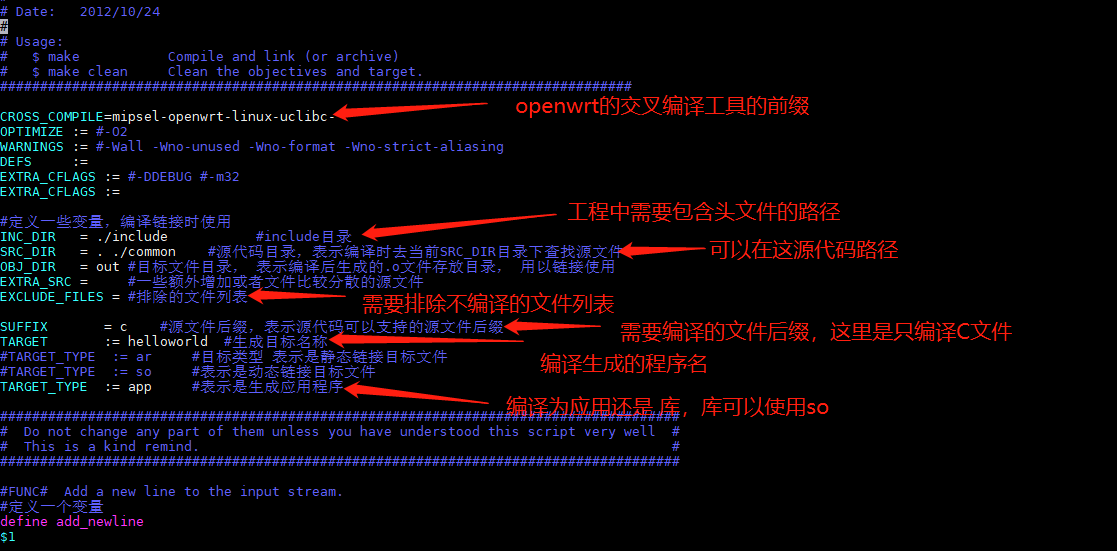
客户也可以根据自己的需求编写makefile ，需要注意的几个地方  
 a. openwrt编译工具的名字是：mipsel-openwrt-linux-uclibc-开头的交叉编译器

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

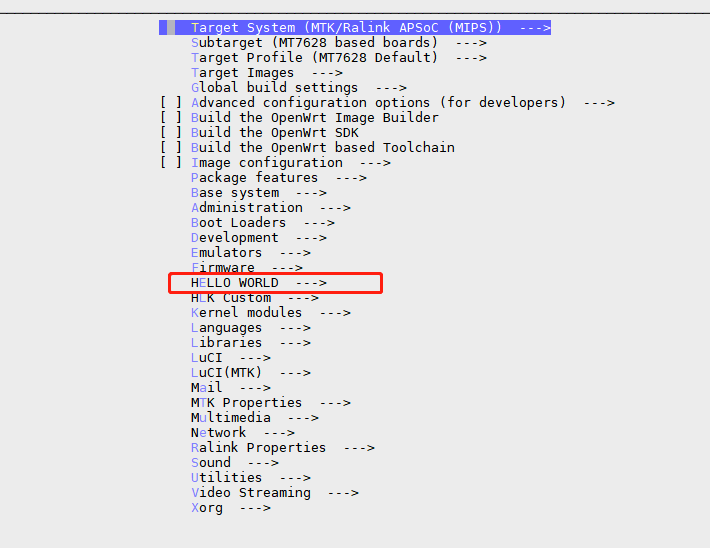
1. 如果使用示例中带的Makefile，对于简单的工程是可以的，它可以遍历src目录

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：

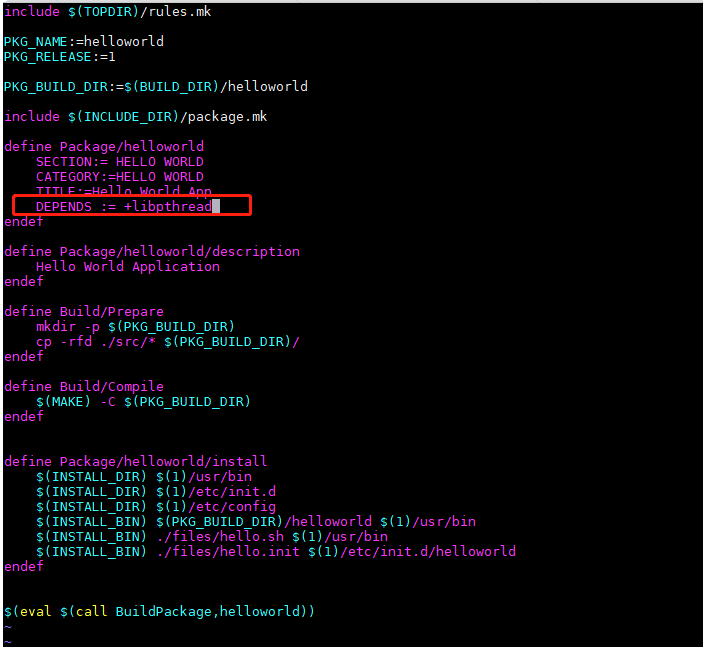


选中后 进入到工程目录下进行make 编译完成后将固件升级后

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

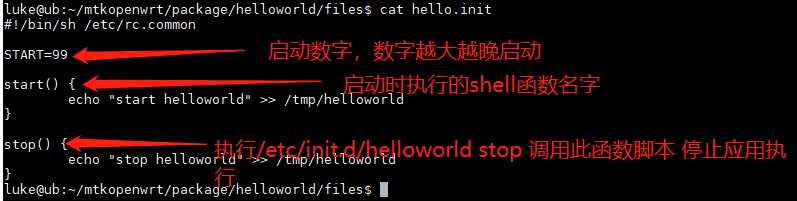
echo "libpthread.so.0"

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：



## 使用C语言控制GPIO

代码示例：

直接替换package/ramips/applications/gpio下的文件即可，替换完成后使用make menuconfig 添加gpio：

MTK Properties --->Applications ---> <\*> gpio.............................................. Command to config gpio

内核的补丁：

把补丁放到target/linux/ramips/patches/目录下

gpio getdir [gpio]

gpio setdir [gpio] [dir] dir----0:IN 1:OUT

gpio read [gpio]

gpio write [gpio] [val] val----0/1

gpio notify [gpio] [pid] notify pid and proc signal

gpio led [gpio] [on] [off] [blink] [reset] [times]

MTK openwrt的GPIO驱动中实现了三个功能

1. 输入输出

操作示例：

设置GPIO 11方向：gpio setdir 11 1 (0: IN 1: OUT)

获取GPIO 11方向：gpio getdir 11

读取GPIO 11值：gpio read 11

写入GPIO 11为 1 ：需要把GPIO 11的方向设置为OUT

gpio setdir 11 1; gpio write 11 1

具体代码可以见

1. 中断通知

代码示例：

GPIO驱动实现了一套中断通知机制，设置给定GPIO触发中断，绑定GPIO到某个进程上，进而使用信号通知进程，用于按键处理，根据按键按住的时间长短有三种信号

SIGUSR1 按键按住大于6S

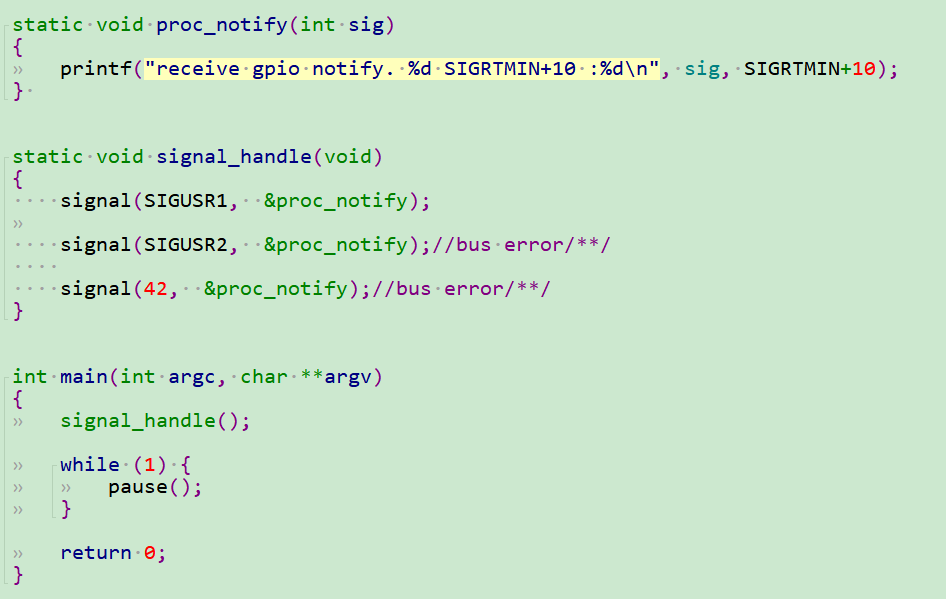
SIGUSR2 按键大于6S

42 按住时间上小于250ms

使用方法：

gpio notify 38 `pidof gpio\_notify` 绑定GPIO38 到 gpio\_notify进程上

Gpio\_notify的源码附件：



1. LED控制

由内核定时器控制LED的闪烁，控制代码见：build\_dir/target-mipsel\_24kec+dsp\_uClibc-0.9.33.2/linux-ramips\_mt7628/linux-3.10.14/drivers/char/ralink\_gpio.c：ralink\_gpio\_led\_do\_timer函数，实际控制代码宏#if defined (RALINK\_GPIO\_HAS\_9532)的范围内

驱动需要 6个参数：

GPIO： GPIO号

On：

Off： On+off表示闪烁比例，具体闪烁时间和 多个参数有关。

on off的大致时间单位是100ms 既：on=1 off=1表示亮100ms，灭100ms

Blinks：

Rests： blinks/rests 这两个参数表示闪烁/熄灭次数

Blinks表示闪烁的次数（一个亮灭为一次）

比如 on=1 off=1 blink=10 rests=10表示 闪烁10/2次。

Times：控制次数 1

例如：on = 1 off = 1 blink = 10 rests = 10 times =1

表示亮100ms 灭100ms 一共循环10/2 共5个周期 然后灭10/2 5个周期，循环一次

这个驱动的LED控制比较复杂，如果需求较简单的话 ，可以参照下面的设置：

长亮：on >= 4000

长灭：off >= 4000 或 rests >= 4000

快闪：on = 1 off =1 blink=4000

慢闪：on = 5 off = 5 blink = 4000

## 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

MTK openwrt的默认vlan配置

这里大致介绍一下交换芯片的VLAN管理：

### Port和vlan相关属性

#### PORT属性：

PVID：端口的缺省vlan ID，当收到的数据包不懈怠vlan tag的時候，芯片会给数据包打上PVID，然后进行转发

#### Vlan属性：

vlan有三個重要的属性：VID，member port 和 untag port

VID：唯一标识一个vlan

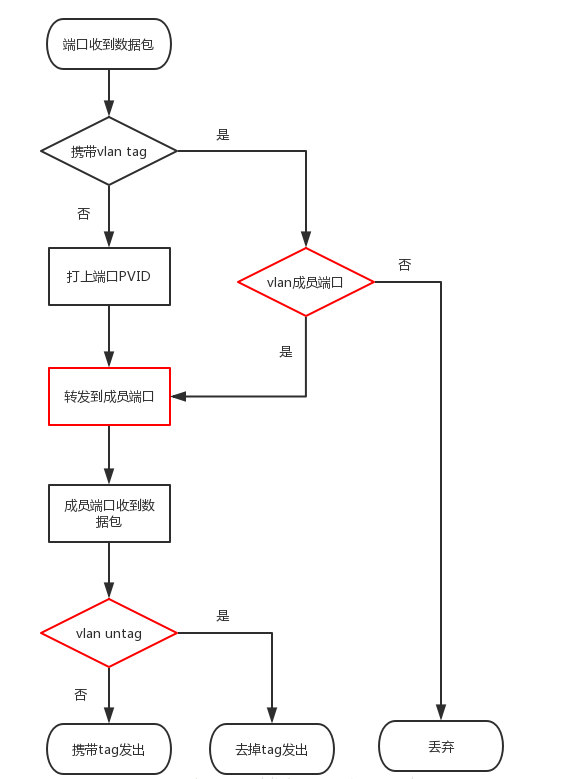
Member Port：Vlan的成员端口

当端口收到一個带vlan的数据包的時候，芯片会首先判断该端口是否是数据包所属vlan的成员端口，如果不是，直接丢弃，反之通过，当芯片要转发一个数据包的時候，只会把数据包转发到所属vlan的成员端口。

untag port：需要去除vlan tag的端口

当端口要发出某一数据包的時候，芯片会判断该数据包所属vlan在本端口是否是untag的，如果是，就去掉vlan tag，反之保留。

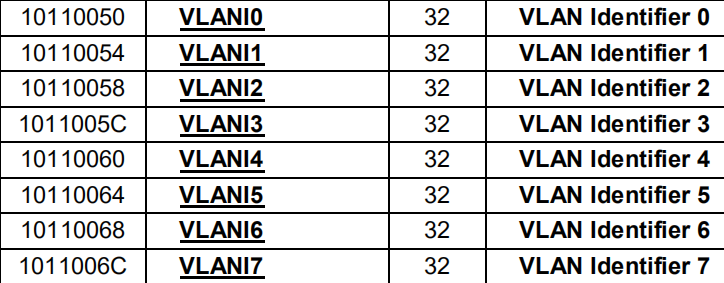
### 数据收发流程

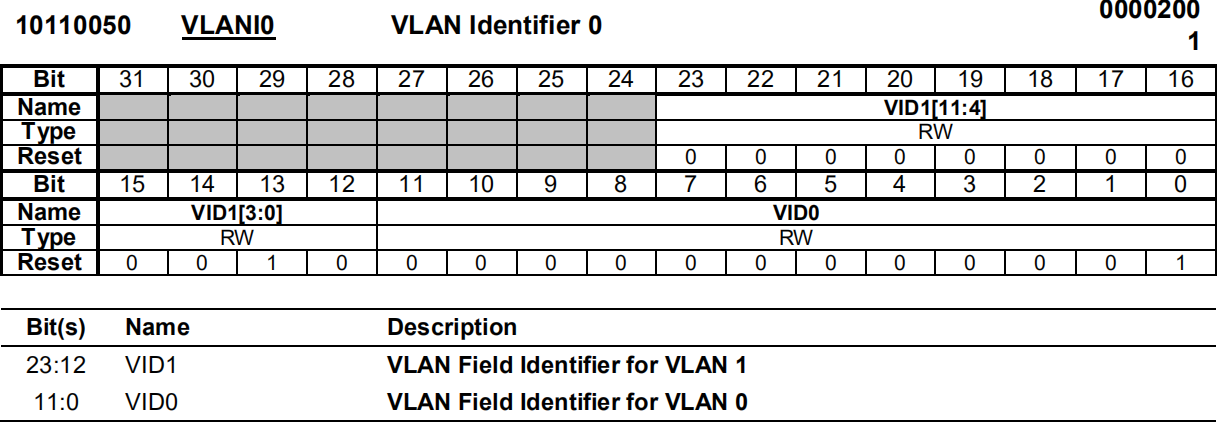


### 芯片配置

#### 配置vlan的VID

       7628/7688一共有8个寄存器，用来记录VID，每个寄存器可以记录两个VID，一共可记录16个VID



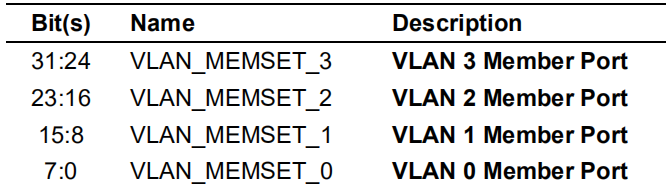
       每个VID 12位，VID的范围是0 - 4095  
 

Mtkopenwrt的默认vlan设置命令：switch reg w 50 2001

既 VID0设置为VLAN1，VID1设置为VLAN2

#### 配置vlan的member port

为1则对应port是该vlan的成员端口  

当设置P0为WAN口时，是这么设置MEM PORT的：

switch reg w 70 ffff417e

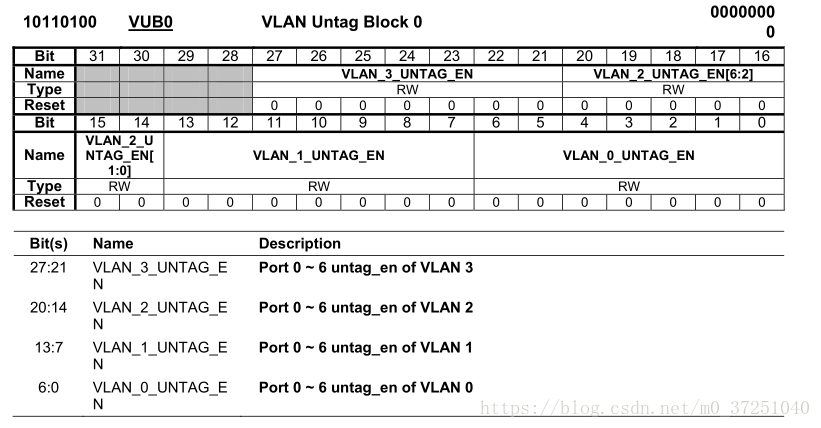
Vlan0 mem port： 7e的二进制形式：0111 1110 表示把端口P1 P2 P3 P4 P5 P6都加入到VLAN1中，

Vlan1 mem port： 41的二进制形式：0100 0001 表示把端口 P0 P6加入到VLAN2中

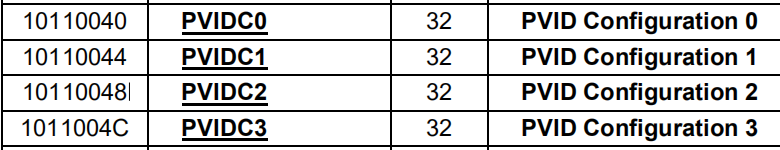
P6是交换芯片连接CPU的端口

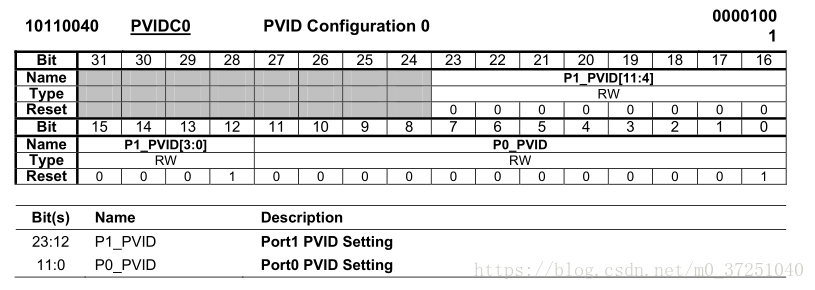
#### 配置vlan的untag port

为1则该vlan在对应port是untag的



#### 配置端口的PVID





每个寄存器控制两个端口的PVID，默认PVID为1

配置Port 0的PVID为2 可以使用命令switch reg w 40 1002 (1002是16进制，高位的1是Port 1的PVID设置为1)

## 网口设置

MT7628/7688共有5个网口，可以通过软件配置任意一个网口为WAN口

默认openwrt只能通过修改配置的方式

### 单WAN口

Mtkopenwrt默认支持两种配置方式 P0作为WAN口或P4作为WAN口

修改工程中的脚本：./build\_dir/target-mipsel\_24kec+dsp\_uClibc-0.9.33.2/switch/ipkg-ramips\_24kec/switch/lib/network/switch.sh中

setup\_switch()

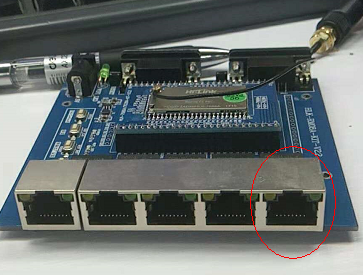
{

#configEsw LLLLW 配置P4为WAN口

configEsw WLLLL 配置P0为WAN口

}

LLLLW （P4为WAN口）在底板上的位置如图：



.

WLLLL (P0为WAN口)在底板上的位置如图：



如临时需将WAN口更换为LAN口 ， 只需要在已启动单板上修改/lib/network/switch.sh文件，setup\_switch()

{

configEsw WLLLL 如果W在左边 则改到右侧， 如果在右侧则改到左侧

}

### 多WAN口

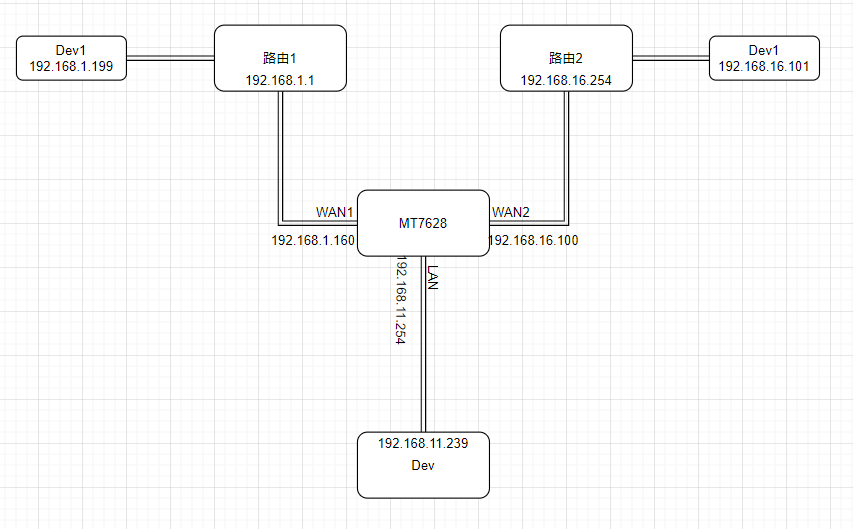
双WAN口配置举例：

把P3 P4口作为WAN口 P0 P1 P2口作为LAN口

P3配置静态IP 11.11.11.254

P4配置静态IP 10.10.10.254

组网图：



Dev 访问192.168.1 .199时，报文经由WAN1到达目的地

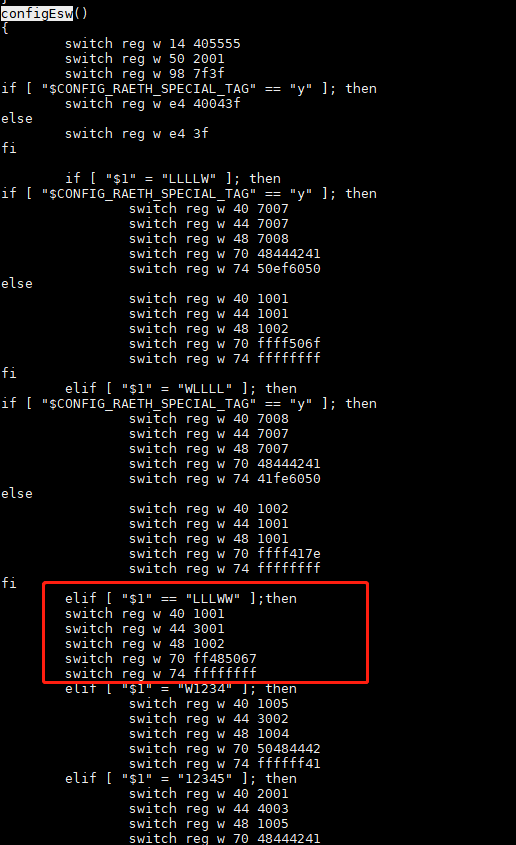
Dev 访问192.168.16.101时，报文经由WAN2到达目的地

双WAN口的配置分为几个步骤

1. Vlan配置

修改switch.sh(文件位于：package/ramips/applications/switch/files/switch-7628.sh)

在configEsw函数中添加：



相对于LLLLW的设置不同点

Switch reg w 44 3001 把PORT3的PVID 设置为3

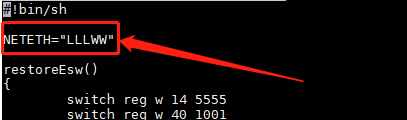
Switch reg w 70 ff485067 设置vlan memport，

VLAN1：0x67既0110 0111 把P0 P1 P2 P5 P6添加到VLAN1中

VLAN2：0x50既0101 0000 把P4 P6添加到VLAN2

VLAN3：0x48既0100 1000 把P3 P6添加到VLAN3

修改NETETH="LLLLW"-> NETETH="LLLWW"



1. 配置/etc/config/network文件修改

添加wan1

在/etc/config/network下添加 wan1配置

config interface 'wan1'

option ifname 'eth0.3'

option proto 'dhcp'

也可以通过UCI命令配置

uci set network.wan1=interface

uci set network.wan1.ifname=eth0.3

uci set network.wan1.proto=dhcp

uci commit

1. 把wan1添加到wan的防火墙域中

uci add\_list firewall.@zone[1].network=wan1

uci commit

重启网络/etc/init.d/network restart

设置完成后， 在Dev上ping 192.168.1.199 和 192.168.16.101，看可否ping通

三WAN口 和四WAN口的配置和双WAN口类似, 这里就不再举例说明

### 无WAN口

所有网口全设置为LAN口

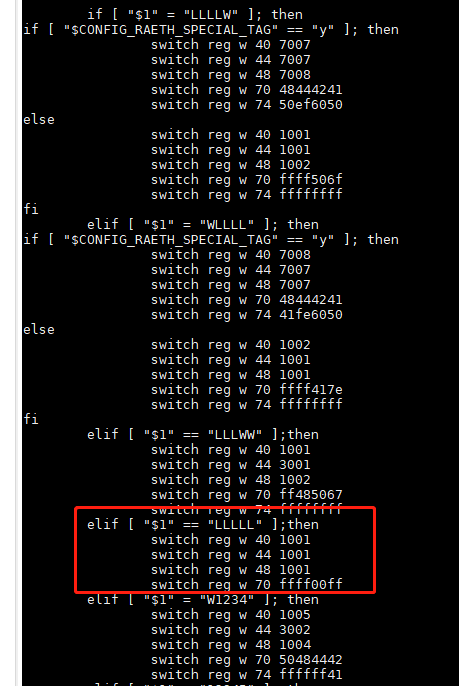
elif [ "$1" == "LLLLL" ];then

switch reg w 40 1001

switch reg w 44 1001

switch reg w 48 1001

switch reg w 70 ffff00ff



setup\_switch()

{

configEsw LLLLL

}

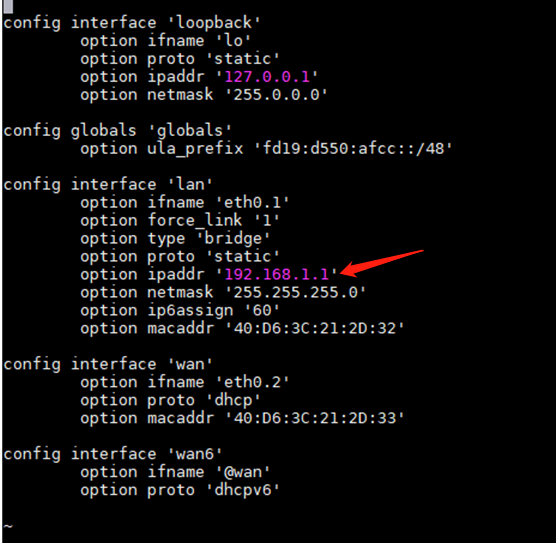
修改/etc/config/network

删除config interface 'wan'配置，重启网络 /etc/init.d/network restart

### 修改LAN口地址

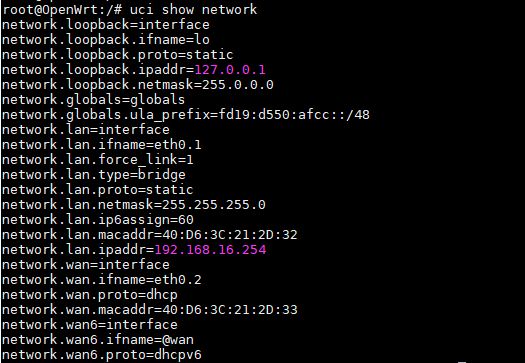
1. 直接修改/etc/config/network配置文件，打开/etc/config/network

可以看到 lan的地址默认为192.168.1.1，可以修改这个字段，改完后使用/etc/init.d/network restart



1. 通过UCI命令配置

使用uci show network, 可以看到network.lan.ipaddr=192.168.16.254



通过uci命令更改：uci set network.lan.ipaddr=192.168.1.1

保存：uci commit

保存完成后 重启网络：/etc/init.d/network restart

网络重启完成后，确认地址是否改成功：

Ifconfig br-lan



## 支持SD卡

需要配置openwrt

make menuconfig

驱动支持

MTK Properties --->

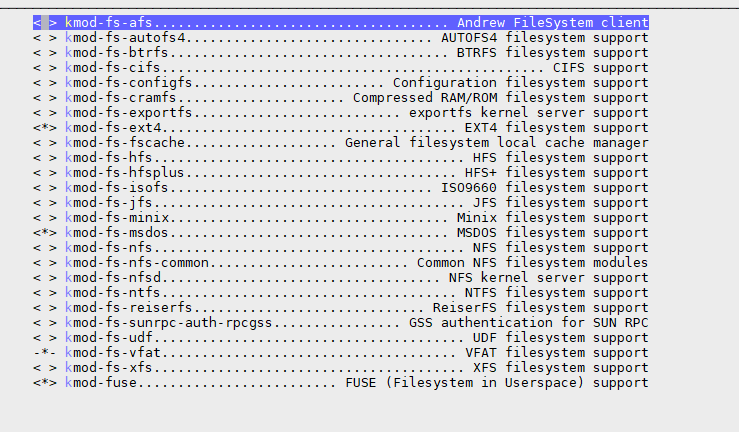
Drivers --->

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

配置文件系统支持：

Kernel modules --->

Filesystems --->



make kernel\_menuconfig:

Ralink Module --->

[\*] One Port Only

## 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

## 看门狗WATCHDOG

Openwrt提供了两个看门狗

硬件：MT7628/7688提供了WDT\_RST\_N引脚

可以通过设置寄存器，使得看门狗定时器超时 输出低电平，控制模块启动

模块复位得两种方式：

1. 设置WDT2SYSRST\_EN 使得watchdog超时 复位系统

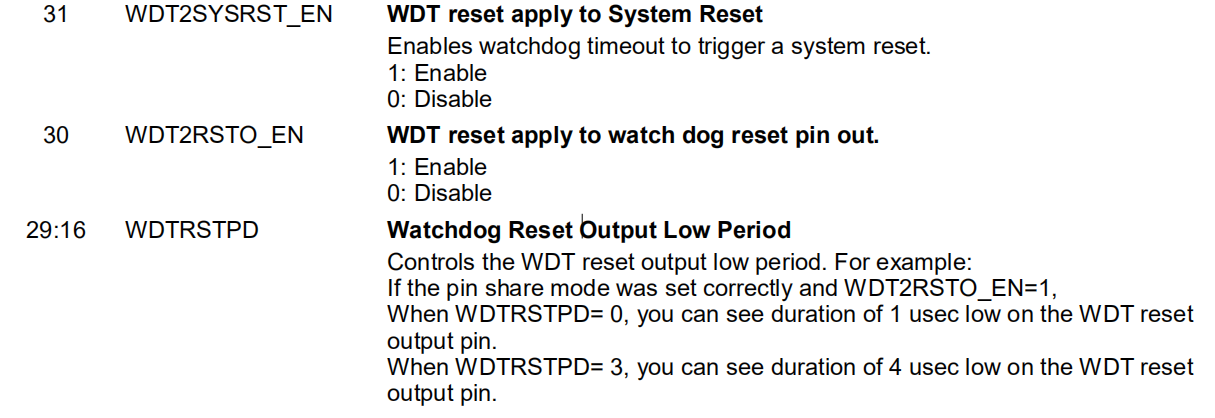
reg s 0; reg r 38; reg w 38 80030000

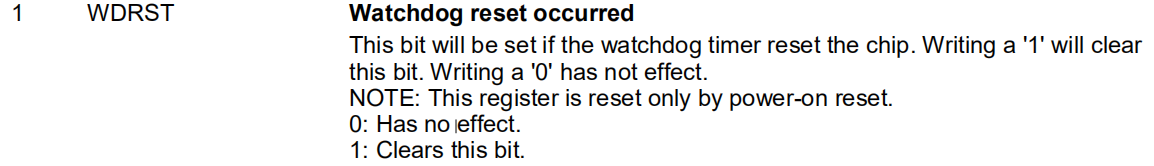
1. 设置WDT2RSTO\_EN 定时器超时 WDT\_RST\_N输出低电平（需要进行验证 WDT接到REST引脚上可否引起系统重启）

可通过设置WDTRSTPD 设置低电平输出时间

reg s 0; reg w 38 40030000

**Reset Status Register寄存器**





使用方法：

Mtkopenwrt中提供了定时器使用示例：package/ramips/applications/watchdog/

可参照示例实现 移植到自己的代码中。

## 定时器

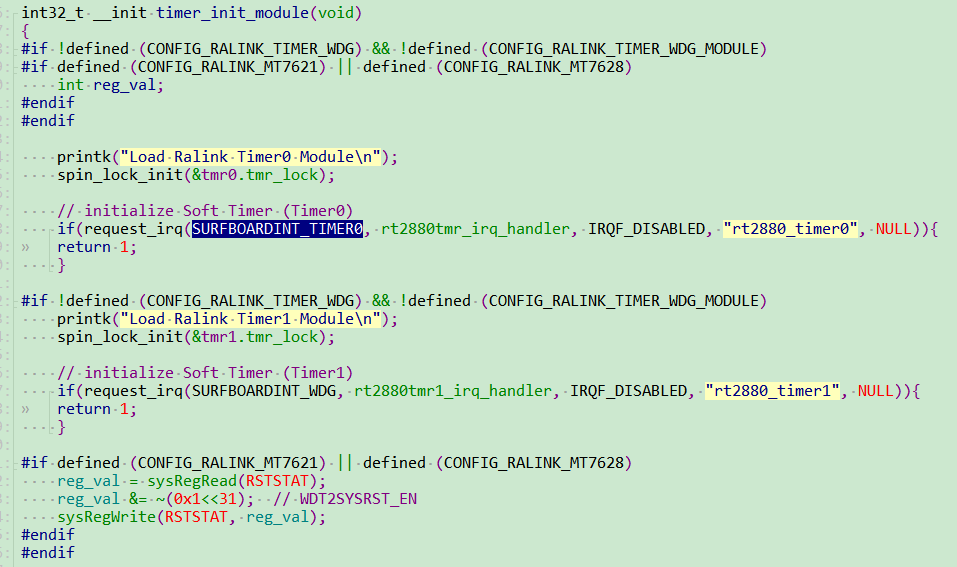
MT7628/7688提供了两个硬件定时器

特性：

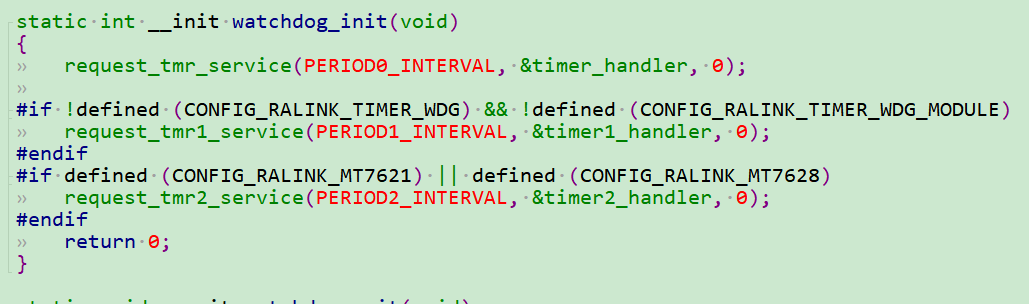
1. 每个定时器最小定时时间1us.
2. 每个定时器有独立中断

可设置为周期模式和超时模式

Mtkopenwrt的linux 内核提供了定时器控制的源码：build\_dir/target-mipsel\_24kec+dsp\_uClibc-0.9.33.2/linux-ramips\_mt7628/linux-3.10.14/arch/mips/ralink/rt\_timer.c



定时器使用示例：target-mipsel\_24kec+dsp\_uClibc-0.9.33.2/linux-ramips\_mt7628/linux-3.10.14/drivers/char/ralink\_period.c



用户如有需要，可参照内核代码实现相应的功能。

## 策略路由

## 静态路由

## WIFI设置

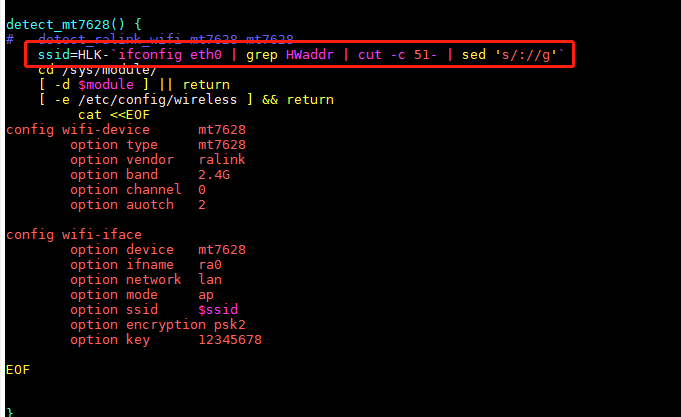
### 设置默认的AP名称/密码

编译固件时修改：

修改package/ramips/drivers/mt7628/files/mt7628.sh

修改红框处的SSID字符串即可实现修改默认AP的SSID

option key 设置AP的密码



### 设置STA功能

MT7628 Mtkopenwrt的WIFI驱动是ApCli驱动，既可以作为AP 同时也可以作为STA连接其他无线路由器

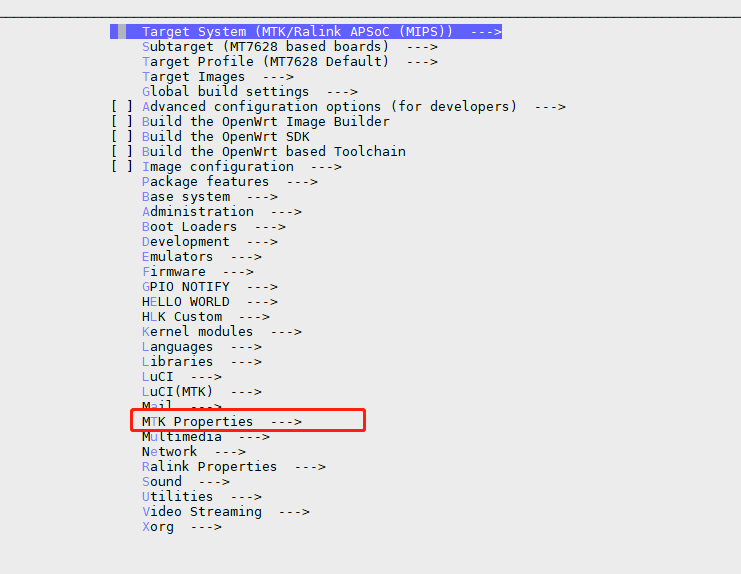
需要配置驱动：

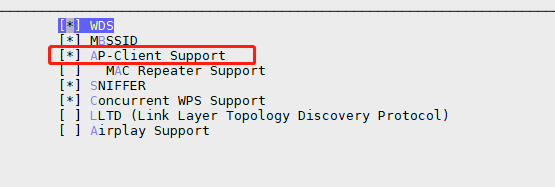
make menuconfig

MTK Properties --->Drivers ---><\*> kmod-mt7628.................................... MTK MT7628 wifi AP driver --->

WiFi Operation Modes --->[\*] AP-Client Support

选中AP-Client Support 后 退出 编译固件





这里介绍WIFI的配置命令：

扫描附近AP：iwpriv ra0 set SiteSurvey=1; sleep 3; iwpriv ra0 get\_site\_survey

设置STA：

#### 直接修改/etc/config的配置文件

示例：

config wifi-iface

option device mt7628

option ifname ra0

option network lan

option mode ap

option ssid mt7628-1912

option encryption psk2

option key 12345678

添加下面的配置参数

option ApCliEnable '1'

option ApCliSsid 'Hi-Link\_PLC'

option ApCliAuthMode 'WPA2PSK'

option ApCliEncrypType 'AES'

option ApCliWPAPSK '12345678'

修改option channel 为实际连接的AP的信道

config wifi-device mt7628

option type mt7628

option vendor ralink

option band 2.4G

option channel 6

option auotch 2

修改后保存。

再对/etc/config/network进行修改

添加：

config interface 'wwan'

option proto 'dhcp'

option ifname 'apcli0' #修改网口为apcli0

修改/etc/config/firewall:

在wan域中添加list network 'wwan'

config zone

option name 'wan'

option input 'REJECT'

option output 'ACCEPT'

option forward 'REJECT'

option masq '1'

option mtu\_fix '1'

list network 'wan'

list network 'wan6'

list network 'wwan'

list network 'wwan'

重启网络 /etc/init.d/network restart

* + - * 1. 当然也可以通过UCI命令来设置：

设置STA参数：

uci set wireless.@wifi-iface[-1].ApCliEnable=1

uci set wireless.@wifi-iface[-1].ApCliSsid=Hi-Link\_PLC

uci set wireless.@wifi-iface[-1].ApCliAuthMode=WPA2PSK

uci set wireless.@wifi-iface[-1].ApCliEncrypType=AES

uci set wireless.@wifi-iface[-1].ApCliWPAPSK=hi-link123

uci set wireless.mt7628.channel=11

uci commit

设置网络：

uci set network.wwan=interface

uci set network.wwan.ifname=apcli0

uci set network.wwan.proto=dhcp

uci commit

设置防火墙：

uci add\_list [firewall.@zone[1].network=wwan](mailto:firewall.@zone[1].network=wwan)

uci commit

设置完成后重启网络

iwpriv apcli0 set ApCliAutoConnect=1

/etc/init.d/network restart

也可以使用纯命令进行设置：

iwpriv apcli0 set ApCliEnable=0

ifconfig apcli0 0.0.0.0

iwpriv apcli0 set ApCliSsid= Hi-Link\_PLC

iwpriv apcli0 set ApCliWPAPSK= hi-link123

iwpriv apcli0 set ApCliAuthMode=WPA2PSK

iwpriv apcli0 set ApCliEncrypType=AES //如果只配置一遍AutoMode 和EncType 则会造成 在关联AP时 出现 加密模式不匹配的问题

iwpriv apcli0 set ApCliAuthMode=WPA2PSK

iwpriv apcli0 set ApCliEncrypType=AES

iwpriv apcli0 set Channel=6

iwpriv apcli0 set ApCliAutoConnect=1

iwpriv apcli0 set ApCliEnable=1

查询是否连接到AP上 可以使用命令 iwconfig

iwpriv apcli0 set ApCliAutoConnect=1

这个命令是用来 当AP信道切换了 ，sta会主动搜索信道连接到新的信道上

可以用来防止有些ap会切换信道 导致连不上的问题

不过连上之后最好关掉 否则可能会一直切换信道 导致其他设备连不上模块的AP

iwpriv apcli0 set ApCliAutoConnect=0

修改参数：

uci set wireless.@wifi-iface[1].ApCliEnable=1

uci set wireless.@wifi-iface[1].ApCliSsid=360WiFi-BB

uci set wireless.@wifi-iface[1].ApCliAuthMode=WPA2PSK

uci set wireless.@wifi-iface[1].ApCliEncrypType=AES

uci set wireless.@wifi-iface[1].ApCliWPAPSK=12345678

uci set wireless.mt7628.channel=11

uci commit

### 多SSID配置

Mtkopenwrt的驱动最多支持4个SSID

示例：多添加一个SSID

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)

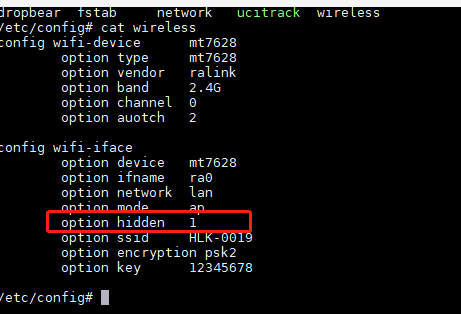
多SSID的一个使用场景：访客WIFI

通过配置多个网桥，把主网和访客网络隔开，保护主网的安全

多网桥配置参见 网口配置中的多网桥配置

注意：目前多SSID和 AP+STA的配置有冲突，不能在多SSID配置下 再去设置Apcli连接指定路由，此问题还未解决

### 隐藏SSID



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

uci commit

### 扫描周围AP

iwpriv ra0 set SiteSurvey=1;sleep 3; iwpriv ra0 get\_site\_survey

### 禁用WIFI

iwpriv ra0 set RadioOn=0

### 7）隐藏WIFI

iwpriv ra0 set HideSSID=1

## 防火墙设置

详细参见：<https://openwrt.org/zh-cn/doc/uci/firewall>

Openwrt支持两种途径配置 iptables ,一种就是 Openwrt 自己的 UCI 方式,另一种就是传统的 Linux 方式

这个章节里介绍一下UCI的配置

## QA测试

文档：

## AES使用

## 串口使用

Mt7628/7688共有3个串口，mtkopenwrt使用了串口UART0 Lite作为调试串口，默认波特率是57600

设置Uart1 Lite为串口

1）修改源码：build\_dir/target-mipsel\_24kec+dsp\_uClibc-0.9.33.2/linux-ramips\_mt7628/linux-3.10.14/drivers/char/ralink\_gpio.h

的514行，#define RALINK\_GPIOMODE\_DFT (RALINK\_GPIOMODE\_UART2 | RALINK\_GPIOMODE\_UART3) | (RALINK\_GPIOMODE\_SPI\_CS1) | (RALINK\_GPIOMODE\_WDT)

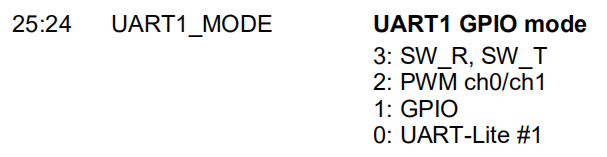
7688的默认把UART2 UART3 SPI\_CS1 WDT功能脚作为GPIO，把RALINK\_GPIOMODE\_UART2 去掉，编译完成后升级固件即可

UART2默认的波特率是9600

UART2对应的硬件上的UART1 Lite引脚

1. 启动后通过命令配置为串口：

设置GPIO1\_MODE寄存器，设置25:24bit为 0



命令设置方法：

reg s 0

reg r 60 #读取GPIO1\_MODE寄存器的值，读取到GPIO1\_MODE的寄存器值为0x55444410，把27:26 bit设置为0

reg w 60 0x54444410

设置UART3 Lite为串口：

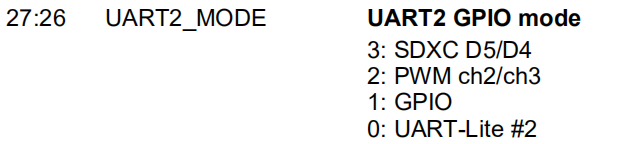
Mtkopenwrt的串口驱动默认只初始化了uart1 uart2两个串口需要修改代码

1）添加代码补丁：

放到target/linux/ramips/patches/目录下

1. 源码同uart1 lite的修改

通过命令设置：



reg s 0

reg r 60 #读取GPIO1\_MODE寄存器的值，读取到GPIO1\_MODE的寄存器值为0x55444410，把27:26 bit设置为0

reg w 60 0x51444410

设置串口波特率的命令：

## 添加SFTP服务

添加sftp配置 使用make menuconfig

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

如果没有这个配置，是因为SFTP是需要通过feed下载的软件

feed更新命令： ./scripts/feeds update -a;./scripts/feeds install –a， 这个过程可能会因为网络原因失败，可以多试几次s

## UCI语法

<https://openwrt.org/docs/guide-user/base-system/uci> openwrt社区对于UCI命令的帮助文档

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

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

## mtkopenwrt自带命令

mii\_mgr:用来设置查询phy寄存器

i. 获取port的端口状态

可以查看<https://blog.csdn.net/subfate/article/details/44958597>介绍的PHY寄存器

port4的phy 编号为4 读取寄存器1

**mii\_mgr -g -p 4 -r 1**

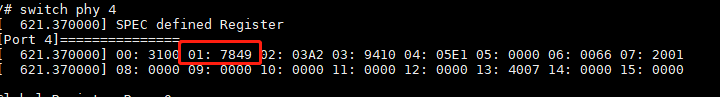
其中 bit2 表示端口链接状态Link Status，bit5表示Auto-Negotiation Complete 自动协商是否完成

一般插拔网线 只是这两个bit在改变

它与reg命令获取ESW Base address+0x80 既读取POA寄存器的29bit是相同的

**reg s 0;reg r 110080;**

也可以通过**switch phy 4**读取到 端口4的所有寄存器的值，其中01：7849和**mii\_mgr -g -p 4 -r 1读取到的值一致**



## 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

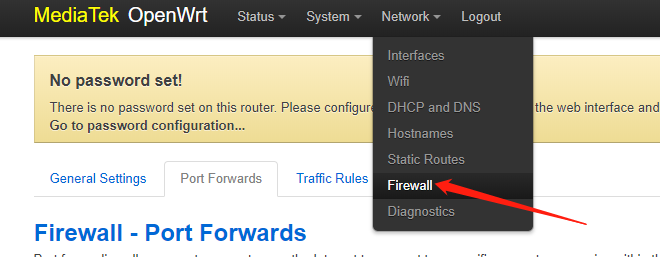
可以通过webui设置好，查看/etc/shadow文件，再写到源码。

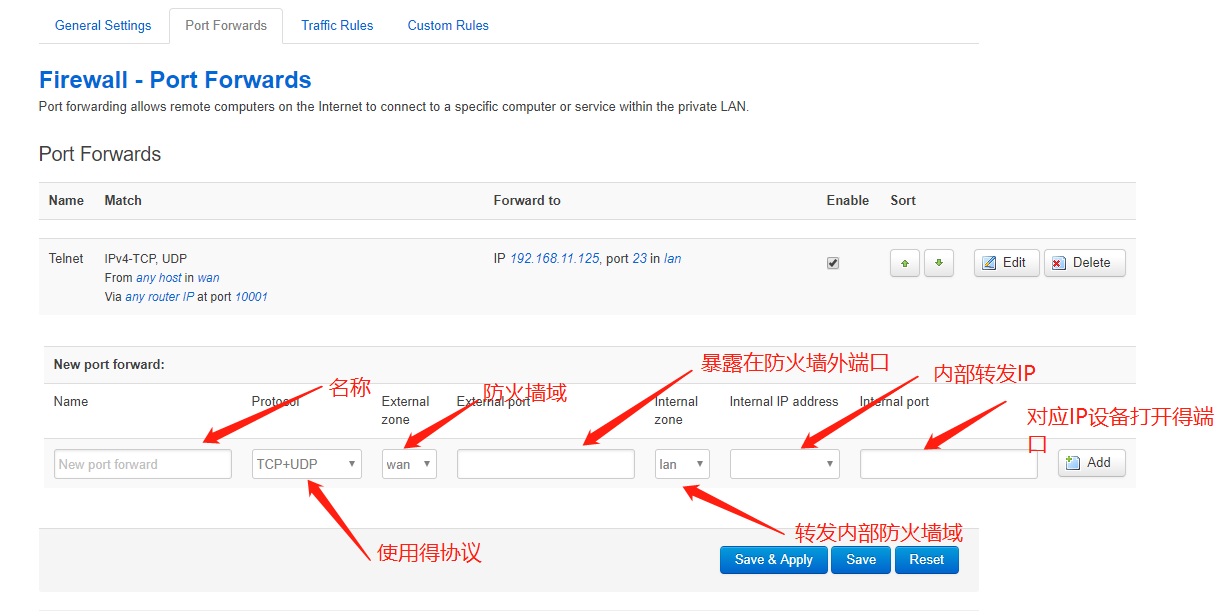
注：系统启动会执行到/etc/inittab，最后一行::askconsole:/bin/ash --login

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

## 网络端口映射

Mtkopenwrt得LUCI上提供了端口映射得配置页面：





# 硬件限制及注意事项

sSPI

SPI 不支持全双工模式

# 客户问题

## 打印 mmc : error-22 whilst initialising MMC card

如果出现这种提示， 是因为openwrt配置了SD卡， 但模块的SD卡引脚接的不是SD卡槽导致的

（例如，用户可能出现，配置了SD卡，但是模块缺插在海凌科的底板上导致出现这种错误）

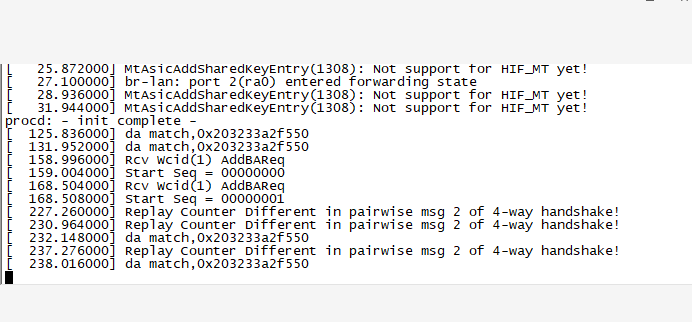


## 打印 DDR Calibration DQS reg = 00008889

如果出现打印后直接重启

请确认，电源问题 如果电源电压不足可能会造成内存频率不稳导致打印这个信息后重启。

## APCLI模式下，电脑经常会连接不上AP，串口提示da match



# 客户需求

## WAN口和STA出口自动切换

如果需要在wan口联网时使用wan口作为默认出口，wan口断开时 使用wwan作为默认出口

Mtkopenwrt的网口驱动默认在WAN口连接断开后，没有通知用户态的netifd做断开链接的处理

所以需要修改驱动代码：

build\_dir/target-mipsel\_24kec+dsp\_uClibc-0.9.33.2/linux-ramips\_mt7628/linux-3.10.14/drivers/net/raeth/raether.c

esw\_interrupt函数：

内核补丁： 

内核的配置需要和脚本配置相符

修改脚本：

package/network/config/netifd/files/lib/netifd/dhcp.script

修改deconfig\_interface函数为：

deconfig\_interface() {

proto\_init\_update "\*" 0

proto\_send\_update "$INTERFACE"

pid=`ps w|grep udhcpc-apcli0.pid|grep -v "grep"|awk '{print $1}'`;

if [ ! $pid ];then

return

fi

if [ "$INTERFACE" = "wan" ];then

kill -SIGUSR2 $pid

sleep 1

kill -SIGUSR1 $pid

fi

}

修改renew|bound, 因为wifi比 以太网口启动慢，会造成在启动时后获取到wifi的IP , 会把默认路由刷新为WIFI路由，需要以太网口renew一次IP:

renew|bound)

setup\_interface

if [ "$INTERFACE" = "wan" ];then

return

fi

pid=`ps w|grep udhcpc-eth0.2.pid|grep -v "grep"|awk '{print $1}'`

if [ ! $pid ];then

return

fi

if [ "$INTERFACE" = "wwan" ];then

kill -SIGUSR2 $pid

sleep 1

kill -SIGUSR1 $pid

fi

已修改文件：

修改package/ramips/applications/switch/files/switch-7628.sh：

setup\_switch()

{

configEsw $NETETH

echo $NETETH > /tmp/neteth

}

内核会解析/tmp/neteth文件 的网口配置

## 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：

## 文档修订记录：

|  |  |  |
| --- | --- | --- |
| 版本号 | 修订内容 | 日期 |
| V1.0 | Draft版本 | 2019-12-19 |
| V1.1 | 增加常见问题 | 2020-5-21 |