



# **RTL819X OpenWRT SDK Application Note**

Confidential

© 2014 Realtek Semiconductor Corporation

All Rights reserved

No. 2, Innovation Road II, Hsinchu Science Park,

Hsinchu 300, Taiwan

[www.realtek.com](http://www.realtek.com)

## Change History

Version	Date	Remarks
2.0	2014/03/07	Initial Release
2.1	2014/03/14	
2.2	2014/06/30	1. add 512M support section 2. add HW Watchdog section 3. add Nand flash section
2.3	2014/11/21	1. Add Proprietary High performace filesystem module section 2. Add Bluetooth support section 3. Add Appendix 10.1 "Modified files in the original OpenWRT" 4. refine ApplicationNote

## Table of Contents

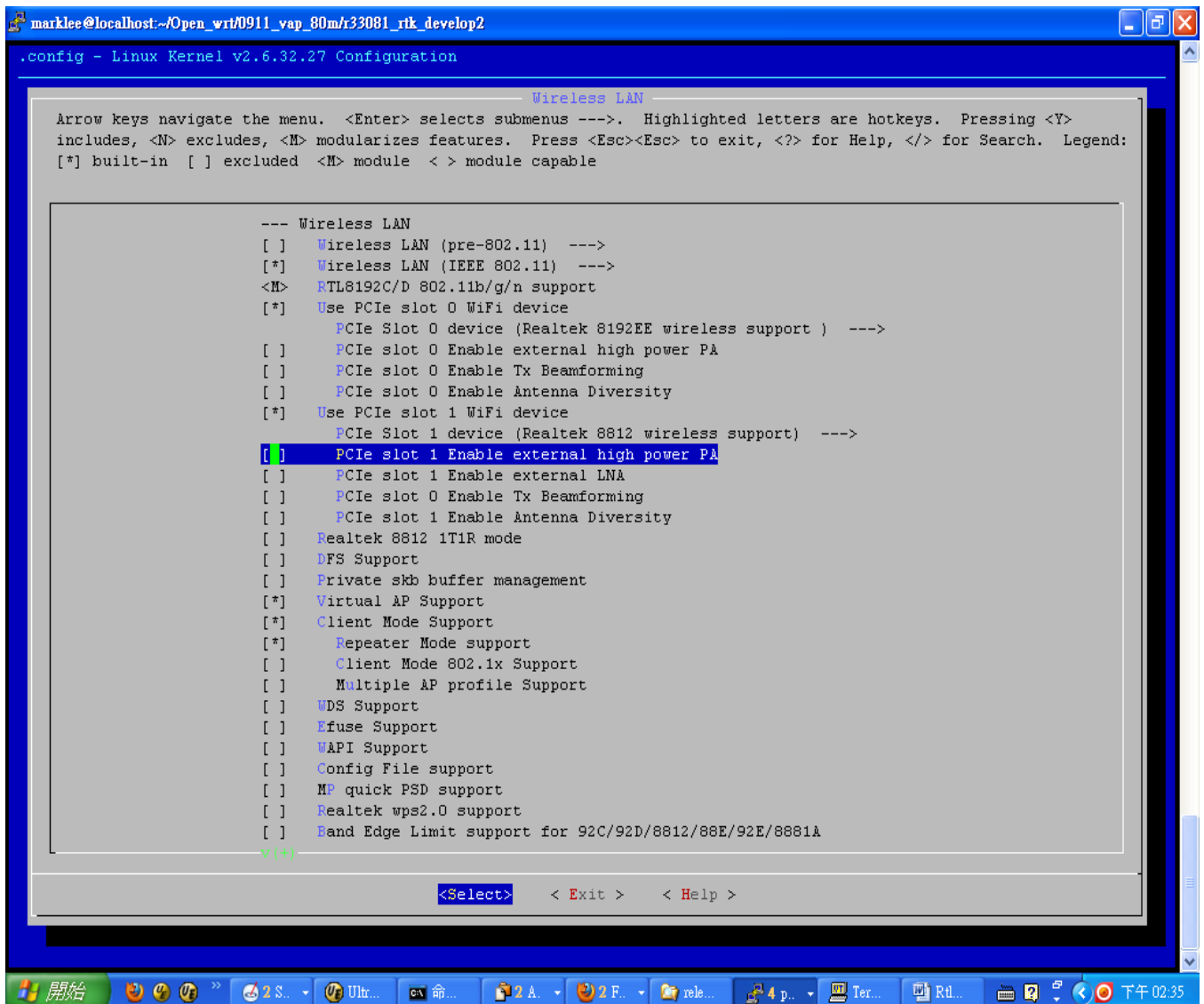
1.	UCI Support .....	4
2.	BSP Kernel options .....	4
2.1.	Wireless driver options .....	4
2.2.	Switch driver options .....	5
2.3.	USB storage support .....	6
2.4.	Nand Flash support .....	6
2.5.	HW Watchdog support .....	7
2.6.	512MB DRAM support(98C only) .....	7
3.	Wireless API Support .....	8
3.1.	Wireless Configuration File .....	8
3.2.	Wireless scripts .....	9
3.3.	Wireless tools .....	9
3.4.	WPS utility .....	10
3.4.1.	AP mode WPS.....	10
3.4.2.	Client mode WPS .....	10
4.	GPIO API .....	11
5.	swconfig support.....	13
5.1.	swconfig menuconfig .....	13
5.2.	UCI switch option .....	13
5.3.	swconfig utility .....	14
6.	Firmware upgrade.....	15
7.	WEB GUI support .....	15
8.	Realtek proprietary features .....	15
8.1.	5M/10M bandwidth control .....	16
8.2.	Repeater mode(bridged AP-STA) .....	16
8.3.	WDS support.....	17
8.4.	Dual Image support.....	19
8.5.	High performance filesystem module.....	21
8.6.	Tools for access hardware settings in FLASH .....	22
9.	Bluetooth Support .....	23
9.1.	Enable Bluetooth Support .....	23
9.2.	Enable USB Bluetooth device support .....	26
9.3.	Enable UART Bluetooth device support.....	27
9.4.	BT functions .....	28
10.	Appendix .....	37
10.1.	Modified files in the original OpenWRT .....	37

# 1. UCI Support

Realtek provides WiFi Router SDK based on OpenWRT framework and also adopt its powerful configuration interface “UCI” to configure the whole system. You can get all UCI information at: <http://wiki.openwrt.org/doc/uci>

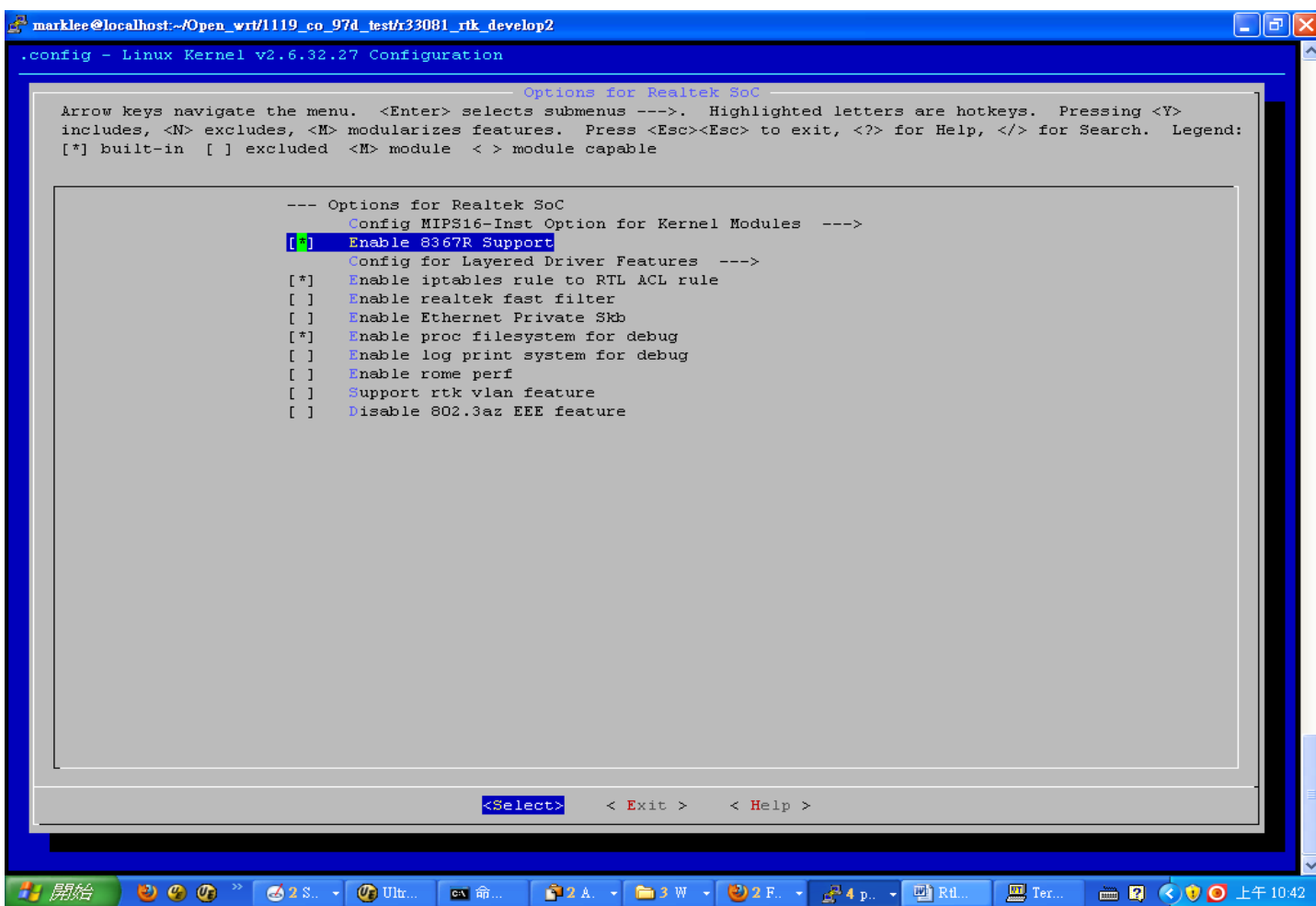
## 2. BSP Kernel options

### 2.1. Wireless driver options



- make kernel\_menuconfig , and enter the sub-menu for realtek's wireless driver.
- Select the correct wifi device to the PCI slot for you platform.
- Important HW related options
  - "external LNA" need your HW platform support
  - "external High power PA" need your HW platform support
  - "Efuse support", if your wifi tx calibration data need to be retrieve from EEPROM of wifi device

## 2.2. Switch driver options



- make kernel\_menuconfig , and enter the sub-menu for realtek's SOC switch driver options in "Network Device" support.
- Important HW related options
  - If your HW platform has "8367R" Giga switch on board. Please select "Enable 8367R Support".

## 2.3. USB storage support

- If your platform is USB supported, then you can select the following options in kernel\_menuconfig to enable USB storage support for different platforms.

### **(1) For 96E (OTG only)**

```
Device Drivers --->
[*] USB support --->
<*> Support for Host-side USB
...
<*> Synopsys DWC_OTG support
[*] enable debug mode
```

### **(2) For 98C/97D/8881A : USB related options has been enabled in default.**

- Select filesystems and language options for your application.

## 2.4. Nand Flash support

- If your platform is Nand Flash supported, then you can follow below steps to configure kernel\_menuconfig to enable your nand flash.

Step1: **“Disable”** SPI flash support in kernel\_menuconfig

Device Drivers →

Memory Technology Device (MTD) support →

RAM/ROM/Flash chip drivers →

[ ] RTL819x SPI flash support **(Disable)**

Step2: Enable Nand flash support in kernel\_menuconfig

Device Drivers →

Memory Technology Device (MTD) support →

NAND Device Support →

[\*] Realtek NAND Flash Driver **(Enable)**

## 2.5. HW Watchdog support

- Follow below steps to configure kernel\_menuconfig to enable HW watchdog support.

Step1: Enable Rtl819x watchdog in kernel\_menuconfig

*Device Drivers* →

*Watchdog Timer Support* →

[\*] *Rtl819x SoC watchdog* **(Enable)**

## 2.6. 512MB DRAM support(98C only)

- If you platform is 512M DRAM supported, then you can follow below steps to configure kernel\_menuconfig to enable your nand flash

Step1: Enable High memory support in kernel\_menuconfig

*Kernel type* →

[\*] *High Memory Support* **(Enable)**

[ ] *Enable bounce buffers* **(Disable)**

## 3. Wireless API Support

The nl80211 netlink interface is the linux standard interface for wireless user application, in order to support various nl80211-based wifi application (for example, iw, hostapd, wpa\_supplicant..etc) on the OpenWRT framework, we implement the cfg80211 interface (nl80211 interface in kernel space) in our wifi driver and base on the compat-wireless package of OpenWRT SDK.

The standard wireless UCI file for configuring the wifi function in OpenWRT is also supported in this SDK. There are some OpenWRT scripts already designed for cfg80211-based wifi drivers, these scripts will response for reading the wireless UCI file and bring up related application to control the cfg80211 wifi device. We use these scripts to handle the standard OpenWRT's wireless UCI file.

### 3.1. Wireless Configuration File

- File: /etc/config/wireless
- It is the standard wireless UCI file in OpenWRT.
- 80211ac VHT80 option is supported
- Below is a simple AP example for wireless UCI file.

```
config 'wifi-device' 'radio0'
    option 'type' 'mac80211'
    option 'macaddr' '00:e0:4c:81:88:cc'
    option 'hwmode' '11na'
    list 'ht_capab' 'SHORT-GI-20'
    list 'ht_capab' 'SHORT-GI-40'
    option 'channel' '7'
    option 'htmode' 'VHT80'
    option 'country' '00'
    option 'disabled' '0'
config 'wifi-iface'
    option 'device' 'radio0'
    option 'network' 'lan'
    option 'mode' 'ap'
    option 'ssid' 'OpenWrt_AP'
    option 'encryption' 'none'
```

- Get full information at <http://wiki.openwrt.org/doc/uci/wireless>



## 3.2. Wireless scripts

→ **/sbin/wifi:**

This script is built in the OpenWRT system for the main wifi device operation control.

→ **/lib/wifi/mac80211.sh & /lib/netifd/wireless/mac80211.sh:**

These script are built in OpenWRT system for init/reinit/disable cfg80211-based wifi device. They will retrieve the Wifi UCI file and bring up some nl80211-based wifi applications(iw,hostapd..etc). Some functions are modified or added to support realtek's proprietary features.

## 3.3. Wireless tools

→ **iw:**

→ iw is a new nl80211-based CLI configuration utility for wireless devices.

→ iw "survey dump" is supported to report channel information. (only **channel time, channel busy time, channel tx time, channel rx time of current channel are supported**)

→ Get full iw information at <http://wireless.kernel.org/en/users/Documentation/iw>

→ **wireless extensions tools: iwconfig, iwlist**

→ cfg80211 interface is compatible with legacy wireless-extension, hence the tools based on wireless-extension interface are also supported in our SDK.

→ **hostapd:**

→ cfg80211-based device driver is supported in Hostapd driver interface, hence we use hostapd to control the wifi interface running for AP mode. We also use

→ **hostapd\_cli:**

→ **hostapd\_cli** utility is a text-based frontend program for interacting with hostapd. WPS pin code and push button are both supported.

→ **wpa\_supplicant:**

→ cfg80211-based device driver is supported in Wpa\_supplicant driver interface, hence we use wpa\_supplicant to control the wifi interface running for Client mode.

## 3.4. WPS utility

### 3.4.1. AP mode WPS

→ The WPS configuration are also stored in wifi UCI file(/etc/config/wireless)

→ These options are already defined in OpenWRT.

```
option 'wps_pushbutton' '1'
option 'wps_label' '1'
option 'wps_config' 'keypad'
option 'wps_device_type' '6-0050F204-1'
option 'wps_device_name' 'Realtek HAPD_DEV AP'
option 'wps_manufacturer' 'www.realtek.com'
```

→ Reference <http://wiki.openwrt.org/doc/uci/wireless#wps.options> for more detail information.

→ We use hostapd\_cli to control the WPS function in AP mode, below are the examples for push button and Pin code method.

→ Pin code:

Command: `hostapd_cli -p /var/run/hostapd -i wlan0 -B wps_pin any 44332211 120`

→ Push button

Command: `hostapd_cli -p /var/run/hostapd -i wlan0 -B wps_pbc .`

→ `hostapd_cli -h` for more information

### 3.4.2. Client mode WPS

→ We use wpa\_cli to control the WPS function in Client mode, below are the examples for push button and Pin code method.

→ Pin code:

Command: `wpa_cli -p /var/run/wpa_supplicant -i wlan0 wps_pin any 4433221100 120`

→ Push button

Command: `wpa_cli -p /var/run/wpa_supplicant -i wlan0 wps_pbc`

→ `wpa_cli -h` for more information

## 4. GPIO API

### ■ **Simple GPIO Button Hotplug driver support on OpenWRT**

- select “Kernel modules” and enter
- select “Other modules” and enter
- <\*> kmod-gpio-button-hotplug

### ■ **Linux general GPIO control**

The realtek's gpio driver is based on standard Linux general/input/leds device driver framework. Hence the users can use the standard /sys/ to control defined gpio pin depend on its class.

→ **/sys/class/gpio and /sys/class/leds** :For example, we test some command below in console.(8197D platform)

#### **(1) Show all defined gpios**

```
root@rtl819xd:/sys/class/gpio# ls
export      gpio3       gpio5       gpio6       gpiochip0   unexport
```

#### **(2) Trigger led to light (suppose “gpio6 is connected to led and low active )**

```
root@rtl819xd:/sys/class/gpio# echo 0 > gpio6/value
```

#### **(3) Press WPS button (gpio3) and get it's status**

```
root@rtl819xd:/sys/class/gpio# cat gpio3/value
1
root@rtl819xd:/sys/class/gpio# cat gpio3/value      → ( press button )
0
```

#### **(4) Show all defined leds**

```
root@rtl819xd:/sys/class/leds# ls
rtl819x:green:wps
```

#### **(5) Trigger led to light**

```
root@rtl819xd:/sys/class/leds# echo 1 > rtl819x\:green\:wps/brightness
```

### ■ ***Customize your GPIO driver***

Since the GPIO pin definition could be different for different platform or different HW board design, You can modify or extend the GPIO driver to customize the GPIO pin. Below show the main source file for different platforms if you need to customize it.

- (1) 8198C
  - target/linux/rtkmips/files/arch/mips/realtek/mach\_rtl8198c.c
- (2) 8197DL/8197DN :
  - target/linux/realtek/files/arch/rlx/soc-rtl819xd/mach\_rtl819xd.c
- (3) 8881A:
  - target/linux/realtek/files/arch/rlx/soc-rtl8881a/mach\_8881a.c
- (4) 8196E
  - target/linux/realtek/files/arch/rlx/soc-rtl8196e/mach\_rtl8196e\_v110\_11n.c

## 5. swconfig support

Swconfig is a very popular utility in OpenWRT to configure WiFi Router's switch function, it includes standard vlan function and also provide a easy way to extend swconfig commad for proprietary switch function(ex: mib\_counter, port link status...etc).

### 5.1. swconfig menuconfig

- Enable swconfig in OpenWRT menuconfig

→ select **"Base system"** and enter  
→ select **"swconfig"**

### 5.2. UCI switch option

- *The UCI option for switch is in the UCI file : /etc/config/network*
- *Switch UCI option example : (it's also default config when you enable swconfig in Realtek OpenWRT SDK)*

```
config 'switch' 'eth0'
    option 'name' 'eth0'
    option 'reset' '1'
    option 'enable_vlan' '1'
config 'switch_vlan' 'eth0_1'
    option 'device' 'eth0'
    option 'vlan' '1'
    option 'ports' '0 1 2 3 6t'
config 'switch_vlan' 'eth0_2'
    option 'device' 'eth0'
    option 'vlan' '2'
    option 'ports' '4 6t'.
```

*NOTE: port6 is CPU port in realtek's SOC .*

- Get full switch option information at <http://wiki.openwrt.org/doc/uci/network/switch>
- The switch config web page on "LUCI" is "Network" → "Switch" on web page.

### 5.3. swconfig utility

- Get some information about swconfig information at <http://wiki.openwrt.org/doc/techref/swconfig>
- Since the vlan related configuration were already bind to UCI in OpenWRT, the users can setup vlan by UCI option instead of executing swconfig command directly.
- Below list some command examples for realtek's SOC switch function support in swconfig.

1. *Show all information about port0 :*

→ Command: "swconfig dev eth0 port 0 show"

2. *get all mibCounter about port0 :*

→ Command: "swconfig dev eth0 port 0 get mib"

3. *reset all mibCounter about port0 :*

→ Command: "swconfig dev eth0 port 0 set reset\_mib"

4. *get link information about port0*

→ Command: *swconfig dev eth0 port 0 get link*

```
config 'switch' 'eth0'
    option 'name' 'eth0'
    option 'reset' '1'
    option 'enable_vlan' '1'
config 'switch_vlan' 'eth0_1'
    option 'device' 'eth0'
    option 'vlan' '1'
    option 'ports' '0 1 2 3 6t'
config 'switch_vlan' 'eth0_2'
    option 'device' 'eth0'
    option 'vlan' '2'
    option 'ports' '4 6t'.
config 'switch_port '
    option 'port' 4
    option 'pvid' 2
```

**NOTE:** *port6 is CPU port in realtek's SOC .*

- Get full switch option information at <http://wiki.openwrt.org/doc/uci/network/switch>
- The switch config web page on "LUCI" is "Network" → "Switch" on web page.

## 6. Firmware upgrade

### ■ *Script API*

The user can use the general firmware upgrade command “/sbin/sysupgrade” provided by OpenWRT to do the image upgrade or backup configuration files. Below list two examples (Assume image file are put at /tmp/firmware.img). You can get more information at <http://wiki.openwrt.org/doc/techref/sysupgrade>.

→ Upgrade firmware and save related configuration files

Command : **sbin/sysupgrade /tmp/firmware.img**

→ Upgrade firmware without saving any configuration file

Command : **sbin/sysupgrade -n /tmp/firmware.img**

*NOTE: The firmware upgrade in LUCI WEB page is also supported.*

### ■ *DualImage support : see “Section8.5”*

## 7. WEB GUI support

- LUCI is the most popular WEB GUI system based on uhttpd in the OpenWRT SDK.
- LCUI is adopted in this SDK as the web server reference design.
- LUCI can support read/write UCI files to configure the OpenWRT Wifi Router.
- The Web server default login IP : 192.168.1.1 with ID = root, and no password need.

## 8. Realtek proprietary features

## 8.1. 5M/10M bandwidth control

The original wifi UCI file of OpenWRT does not support 5M/10M bandwidth. We add new options in wifi UCI file for each radio interface to support these features.

### → New UCI options

option	value : function	Comment
rtk_priv_bw	5M : set bandwidth to 5M 10M : set bandwidth to 10M	1. This option needs HW capability support in wifi chip, for example rtl8812. 2. When this option is set, the setting of option "htmode" will be ignored by wifi driver.

### → UCI Example: Set radio0 interface to 10M bandwidth

```
config 'wifi-device' 'radio0'
option 'type' 'mac80211'
option 'channel' '36'
option 'macaddr' '00:e0:4c:81:89:ee'
option 'hwmode' '11n'
option 'htmode' 'HT20'
list 'ht_capab' 'SHORT-GI-20'
list 'ht_capab' 'SHORT-GI-40'
option 'disabled' '0'
option 'country' 'US'
option rtk_priv_bw 10M
```

```
config 'wifi-iface'
option 'device' 'radio0'
option 'network' 'lan'
option 'mode' 'ap'
option 'ssid' 'OpenWrt_AC_HT80'
option 'encryption' 'none'
```

## 8.2. Repeater mode(bridged AP-STA)

The original OpenWRT does not support for binding Client interface to bridge with another AP interface due to the wifi driver need to support some IP/MAC address translation protocols to handle the bridging traffic. The nat2.5 is implemented in our wifi driver for the address



translation mechanism, hence the wifi device can be configured as Repeater mode in our solution.

→ **Repeater mode UCI sample:**

```
config 'wifi-device' 'radio0'
    option 'type' 'mac80211'
    option 'macaddr' '00:e0:4c:81:88:cc'
    option 'hwmode' '11ng'
    option 'channel' '7'
    option 'htmode' 'HT40+'
    list 'ht_capab' 'SHORT-GI-20'
    list 'ht_capab' 'SHORT-GI-40'
config 'wifi-iface'
    option 'device' 'radio0'
    option 'ssid' 'OpenWrt_AP'
    option 'network' 'lan'
    option 'mode' 'ap'
    option 'encryption' 'none'

config 'wifi-iface'
    option 'device' 'radio0'
    option 'ssid' 'OpenWrt_Client'
    option 'network' 'lan'
    option 'mode' 'sta'
    option 'encryption' 'none'
```

**#NOTE:** AP interface must be listed before STA interface in the UCI file and they need to be configured to the same network(for bridging) , for example they all belong to “lan” in this sample.

→ **Repeater mode WEB GUI :** Reference to [RTK\\_Repeater\\_WEB\\_guide.pdf](#) to see how to configure Repeater from LUCI.

## 8.3. WDS support

If there are 2 APs, named AP-1 & AP-2, if

1. Both AP-1 & AP-2 enable WDS function,
2. AP-1 adds AP-2's mac into its WDS entry, AP-2 adds AP-1's mac into its WDS

entry,

3. AP-1 & AP-2 use the same WDS encryption type & key.
4. AP-1 & AP-2 use the same channel.

Then AP-1 & AP-2 can transfer packets via WDS function.

### → **New UCI options**

option	value : function	Comment
rtk_wds	0 – disable, 1 – enable	WDS enable/disable
rtk_wds_macaddr0 rtk_wds_macaddr1 rtk_wds_macaddr2 ...	xxxxxxxxxxxx (12 digits mac address). ex: 00e04c8188bb	Set mac address of WDS AP  rtk_wds_macaddr0 = first MAC of WDS AP rtk_wds_macaddr1 = first MAC of WDS AP rtk_wds_macaddr2 = first MAC of WDS AP ... Currently support MAX=8 WDS AP
rtk_wds_privacy	0 – disabled, 1 – WEP64, 2 – TKIP, 4 – AES (CCMP), 5 – WEP128	WDS encryption mode
rtk_wds_wepkey	Type of byte array	WDS WEP default key
rtk_wds_passphrase	Type of string	WDS PSK key

### → **WDS mode UCI sample:**

Add additional WDS options (black parts) to your root-AP config. (Only AP mode support WDS function).

**Example1: Set mac 00:e0:4c:aa:03:21 & 00:e0:4c:aa:03:22 into WDS entries,in None-security.**

```
config 'wifi-iface'
    option 'device' 'radio0'
    option 'ssid' 'OpenWrt_AP'
    option 'network' 'lan'
    option 'mode' 'ap'
    option 'encryption' 'none'
```

```
option 'rtk_wds' '1'  
option 'rtk_wds_macaddr0' '00e04caa0321'  
option 'rtk_wds_macaddr1' '00e04caa0322'  
option 'rtk_wds_privacy' 0'
```

***Example2: Set mac 00:e0:4c:aa:03:21 & 00:e0:4c:aa:03:22 into WDS entries,  
in WEP mode, Wep Key = 12345***

```
config 'wifi-iface'  
    option 'device' 'radio0'  
    option 'ssid' 'OpenWrt_AP'  
    option 'network' 'lan'  
    option 'mode' 'ap'  
    option 'encryption' 'none'  
    option 'rtk_wds' 1  
    option 'rtk_wds_macaddr0' '00e04caa0321'  
    option 'rtk_wds_macaddr1' '00e04caa0322'  
    option 'rtk_wds_privacy' '1'  
    option 'rtk_wds_wepkey' '3132333435'
```

***Example3: Set mac 00:e0:4c:aa:03:21 & 00:e0:4c:aa:03:22 into WDS entries,  
in PSK-CCMP mode, Key = 12345678***

```
config 'wifi-iface'  
    option 'device' 'radio0'  
    option 'ssid' 'OpenWrt_AP'  
    option 'network' 'lan'  
    option 'mode' 'ap'  
    option 'encryption' 'none'  
    option 'rtk_wds' '1'  
    option 'rtk_wds_macaddr0' '00e04caa0321'  
    option 'rtk_wds_macaddr1' '00e04caa0322'  
    option 'rtk_wds_privacy' '4'  
    option 'rtk_wds_passphrase' '12345678'
```

## **8.4. Dual Image support**

Realtek provide DualImage support in this SDK to keep system more safe from upgrading a unstable image(system crash) or a damaged one(burning flash error). The system could be recovered to previous image stored in the backup bank.

For example, if the current running bank is “bank0” and the user want to upgrade the firmware image, the image will be upgraded into “bank1” and reboot by using image stored in “bank1”. There are two possible conditions will happen after rebooting, below we show how the dualimage function work,

(1) If the system is booting correctly and bring up the user application successfully, a retry counter stored in flash HW information block should be reset to indicate the image is a good one for running and “bank0” become the backup bank for next time firmware upgrade.

(2) If the system could not be booting correctly due to image checksum error or it keep rebooting continuously (the default limitation of rebooting count is “3” times ) due to a unstable firmware release, the system will decide to boot from the previous image stored in “bank0”.

The following section will guide you how to enable the DualImage support in both bootcode part and kernel part.

### ■ **Dual Image (Bootcode part)**

The Dual Image function need bootcode support, you can read the “README” (under bootcode\_patch/ ) to learn how to obtain a bootcode that is DualImage supported and how to enable it in bootcode menuconfig option. The keys are,

(1) Get the bootcode that is dual image supported.

(2) Enable below option in bootcode menuconfig

[\*] Support DualImage(OpenWRT SDK)

\*\*\* Second Bank Offset \*\*\*

(0x400000) flash offset

*NOTE: Second bank offset means the start point for second bank in the flash . for example you have a 8M flash , then you can set the offset to 4M(0x400000), this will split the flash into bank0(0~4M) and bank1(4M~8M).*

### ■ **Dual Image (Kernel part)**

(1) Enable kernel options

→ make kernel\_menuconfig (Machine selection -> BSP configurations->)

[\*] DualImage Support(OpenWRT SDK)

\*\*\* Second Bank Offset \*\*\*

(0x400000) flash offset

*NOTE: Second bank offset means the start point for second bank in the flash .For example you have a 8M flash , then you can set the offset to 4M(0x400000), this will split the flash into bank0(0~4M) and bank1(4M~8M).*

(2) DualImage Firmware upgrade related scriptes

→ **/sbin/sysupgrade**

The script command for user application interface to do dualimage firmware upgrade is exactly the same with **Section5**.

→ **/lib/upgrade/platform.sh.**

The dualimage firmware upgrade procedure was implemented in the script (/lib/upgrade/platform.sh) to hidden the detail from user application interface( /sbin/sysupgrade).

(3) Boot information: /proc/flash

The files under this directory were stored in flash HW block to record the boot information for dualimage operations.

→ **/proc/flash/ bootcnt**

*This file represents the current rebooting count. The value should be reset to “0” once the system was bringing up successfully to indicate bootcode that this image is a good one, otherwise the bootcode will always increase this counter everytime booting and switch to another bank if the counter reach the max counter limitation define by user (/proc/flash/bootmaxcnt). Currently, we put the action for reset counter in /etc/init.d/swdone.*

→ **/proc/flash/bootmaxcnt**

*This file represents the maxium rebooting count. The default value is “3”. The user can overwrite the setting by echo value to this proc file.*

→ **/proc/flash/bootbank**

*This file represents the bank that is running currently.*

## 8.5. High performance filesystem module

Realtek provides a kernel-mode file-system module which can full access to volumes of the popular file systems: “**NTFS**”, “**exFAT**” and “**HFS+**” with high

performance.

### (1) *Select kmod-rtl\_fs*

make menuconfig

Kernel modules --->

Filesystems --->

<\*> kmod-rtl\_fs..... Realtek High Performance NTFS/exFAT/HFS+ filesystem

```
< > kmod-fs-afs..... Andrew FileSystem client
-- kmod-fs-autofs4..... AUTOFS4 filesystem support
< > kmod-fs-btrfs..... BTRFS filesystem support
< > kmod-fs-cifs..... CIFS support
< > kmod-fs-configfs..... Configuration filesystem support
< > kmod-fs-cramfs..... Compressed RAM/ROM filesystem support
< > kmod-fs-exportfs..... exportfs kernel server support
< > kmod-fs-ext4..... EXT4 filesystem support
< > kmod-fs-fscache..... General filesystem local cache manager
< > kmod-fs-hfs..... HFS filesystem support
< > kmod-fs-hfsplus..... HFS+ filesystem support
< > kmod-fs-isofs..... ISO9660 filesystem support
< > kmod-fs-jfs..... JFS filesystem support
< > kmod-fs-minix..... Minix filesystem support
< > kmod-fs-msdos..... MSDOS filesystem support
< > kmod-fs-nfs..... NFS filesystem support
< > kmod-fs-nfs-common..... Common NFS filesystem modules
< > kmod-fs-nfsd..... NFS kernel server support
< > kmod-fs-ntfs..... NTFS filesystem support
< > kmod-fs-reiserfs..... ReiserFS filesystem support
< > kmod-fs-udf..... UDF filesystem support
< > kmod-fs-vfat..... VFAT filesystem support
< > kmod-fs-xfs..... XFS filesystem support
-- kmod-fuse..... FUSE (Filesystem in Userspace) support
<*> kmod-rtl_fs..... Realtek High Performance NTFS/exFAT/HFS+ filesystem
```

### (2) *Manual insert module and mount device*

insmod jnl.ko

insmod ufsd.ko

mkdir -p <directory path>

mount -t ufsd -o nls=utf8 <device> <directory path>

## 8.6. Tools for access hardware settings in FLASH

Realtek provides a user program, *flash*, to read/write hardware MIB settings in FLASH. These settings should be written during mass production procedure. They are stored permanently unless over-written. Part of them is relative to power index for

radio performance normalization. Those most customers concerned are the MAC addressed for Ethernet/Wireless interface. These settings take effect during system boot-up.

Ps. In case that store wireless relative settings at EFUSE of wireless module, any modification through *flash* is not applied. And settings with token, "WLAN", are relative to wireless.

There are three types of sub-command supported :

- flash allhw

```
root@rtkmips:/# flash allhw
read_hw_setting_offset = 20000
HW_BOARD_VER=1
HW_NIC0_ADDR=00e04c819809
HW_NIC1_ADDR=00e04c819808
HW_WLAN0_WLAN_ADDR=00e04c819888
HW_WLAN0_WLAN_ADDR1=00e04c819889
HW_WLAN0_WLAN_ADDR2=00e04c81988a
HW_WLAN0_WLAN_ADDR3=00e04c81988b
HW_WLAN0_WLAN_ADDR4=00e04c81988c
HW_WLAN0_WLAN_ADDR5=00e04c81988d
HW_WLAN0_WLAN_ADDR6=00e04c81988e
HW_WLAN0_WLAN_ADDR7=00e04c81988f
HW_WLAN0_TV_POWER_CCK_A=00000000000000000000000000000000
```

- flash get \$(parameter name)  
Ex. flash get HW\_WLAN0\_WLAN\_ADDR then return:  
HW\_WLAN0\_WLAN\_ADDR=00e04c819888  
  
note. \$(parameter name) is the value from allhw displayed
- flash set \$(parameter name)  
Ex. flash set HW\_WLAN0\_WLAN\_ADDR 00e04c819887

## 9. Bluetooth Support

### 9.1. Enable Bluetooth Support

#### ■ *Kernel menuconfig*



```
[*] Enable the block layer --->
    Bus options (PCI, PCMCIA, EISA, ISA, TC) --->
    Executable file formats --->
    Power management options --->
[*] Networking support --->
    Device Drivers --->
    Firmware Drivers --->
    File systems --->
```

```
--- Networking support
    Networking options --->
[ ] Amateur Radio support --->
[ ] CAN bus subsystem support --->
[ ] IrDA (infrared) subsystem support --->
[*] Bluetooth subsystem support --->
[ ] RxRPC session sockets
-* Wireless --->
[ ] WiMAX Wireless Broadband support --->
[ ] RF switch subsystem support --->
[ ] Plan 9 Resource Sharing Support (9P2000) --->
[ ] CAIF support --->
[ ] Ceph core library
[ ] NFC subsystem support --->
```

```
--- Bluetooth subsystem support
[*] RFCOMM protocol support
[*] RFCOMM TTY support
[*] BNEP protocol support
[*] Multicast filter support
[*] Protocol filter support
[*] Bluetooth device drivers --->
```





```
[ ] HCI USB driver
[*] RTK HCI USB driver
[*] HCI UART driver
[ ]   UART (H4) protocol support
[*]   BCSP protocol support
[ ]   Atheros AR300x serial support
[*]   Realtek H5 protocol support
[ ]   HCILL protocol support
[ ]   Three-wire UART (H5) protocol support
[ ] HCI BCM203x USB driver
[ ] HCI BPA10x USB driver
[ ] HCI BlueFRITZ! USB driver
[ ] HCI VHCI (Virtual HCI device) driver
[ ] Marvell Bluetooth driver support
```

### ■ Users menuconfig

Select bluez and the bluetooth chip

```
Target System (Realtek mips SOC) --->
Subtarget (RTL8198c based boards) --->
Target Profile (AP package) --->
Target Images --->
Global build settings --->
[ ] Advanced configuration options (for developers) --->
[ ] Build the OpenWrt Image Builder
[ ] Build the OpenWrt SDK
[ ] Build the OpenWrt based Toolchain
[ ] Image configuration --->
Base system --->
Boot Loaders --->
Development --->
Firmware --->
Kernel modules --->
Languages --->
Libraries --->
LuCI --->
Network --->
[ ] utilities --->
```

```
< > avrusbboot..... USB bootloader for Atmel AVR controllers
< > bandwidthd..... Bandwidthd
< > bash..... The GNU Bourne Again SHell
< > bash-completion..... bash completion scripts
< > bc..... Arbitrary precision calculator language
< > bluelog..... Bluetooth scanner and logger
< > bluez..... Realtek Bluetooth utility --->
< > bluez-hcidump..... Bluetooth packet analyzer
< > bluez-utils..... Bluetooth utilities
< > boblight-client
< > boblight-daemon
< > bonniexx..... Bonnie++ - hard drive bottleneck testing program.
< > bsdiff..... Tools for building and applying patches to binary files
< > byobu
```

```

- bluez..... Realtek Bluetooth utility
  Serial Chip Selection --->
  USB Chip Selection --->

```

■ **Add mdev in by busybox menconfig if usb bluetooth use**

```

[ ] Build the OpenWrt SDK
[ ] Build the OpenWrt based Toolchain
[ ] Image configuration --->
  Package features --->
  [*] Base system --->
    Administration --->
    Boot Loaders --->
    Development --->

```

```

<*> base-files..... Base filesystem for OpenWrt
< > block-mount..... Block device mounting and checking
< > bridge..... Ethernet bridging configuration utility
<*> busybox..... Core utilities for embedded Linux --->
< > ca-certificates..... System CA certificates
< > dash..... Debian Almquist shell

```

```

--- busybox..... Core utilities for embedded Linux
[*] Customize busybox options
  Busybox Settings --->
  *** Applets ***
  Archival Utilities --->

```

```

Linux Ext2 FS Progs --->
Linux Module Utilities --->
Linux System Utilities --->
Miscellaneous Utilities --->
Networking Utilities --->
Print Utilities --->

```

```

[ ] lpcs
[ ] losetup
[*] mdev
[*] Support /etc/mdev.conf
[*] Support subdirs/symlinks
[*] Support regular expressions substitutions when renaming device
[*] Support command execution at device addition/removal
[*] Support loading of firmwares
[ ] mswap

```

## 9.2. Enable USB Bluetooth device support

**[\*] USB support --->**

```
-- USB support
<*> Support for Host-side USB
[ ] USB verbose debug messages
[ ] USB announce new devices
*** Miscellaneous USB options ***
[*] Enable USB persist by default
[ ] Dynamic USB minor allocation
[ ] Rely on OTG Targeted Peripherals List
[ ] Disable external hubs
< > USB Monitor
< > Support WUSB Cable Based Association (CBA)
*** USB Host Controller Drivers ***
< > Cypress C67x00 HCD support
<*> xHCI HCD (USB 3.0) support
[*] Debugging for the xHCI host controller
< > OXU210HP HCD support
< > ISP116X HCD support
< > ISP 1760 HCD support
< > ISP1362 HCD support
< > SL811HS HCD support
< > R8A66597 HCD support
*** USB Device Class drivers ***
< > USB Modem (CDC ACM) support
< > USB Printer support
< > USB Wireless Device Management support
< > Functions for loading firmware on EZUSB chips
<*> Synopsys DWC OTG support
[ ] enable debug mode
```

## 9.3. Enable UART Bluetooth device support

Device Drivers --->

Character devices --->

Serial drivers --->

```
[*] 8250/16550 and compatible serial support
[*]   Console on 8250/16550 and compatible serial port
(2)   Maximum number of 8250/16550 serial ports
(2)   Number of 8250/16550 serial ports to register at runtime
[ ]   Extended 8250/16550 serial driver options
*** 8250 compatible port support ***
[ ] 819x RTL UART1 support
[ ] SC16IS7x0 series (I2C bus) support
```

**Note:** Number of 8250/16550 serial port to register at runtime should be changed to 2

## 9.4. BT functions

→If usb bluetooth chip used ,the following command should be issued before bring the device up.

```
#echo /bin/mdev > /proc/sys/kernel/hotplug
#mount -t sysfs sysfs /sys
#echo 1 > /sys/class/firmware/timeout
```

→If uart bluetooth chip used, the following command should be issued to bring up the device.

```
#hciattach -n -s 115200 /dev/ttyS1 rtk_h5 115200 &
```

→ verify by bluez command

```
#hciconfig hci0 up
#hciconfig hci0 reset
#hciconfig hci0 iscan /*can be seen by other bluetooth device */
#hcidtool scan /*scan other bluetooth device*/.
#l2ping xx:xx:xx:xx:xx:xx /*if can ping through,you may should click “pair” on cellphone */
```

→ how to pair (“#” indicate run on dut console)

```
#hciconfig hci0 piscan
#bluetoothctl /*enter bluetoothctl interface*/
#help /*show bluetoothctl support command*/
#scan on
#pairable on
#discoverable on
#agent on
#default-agent
```

Open bluetooth on cellphone, search bluetooth, and click the bluetooth device you found  
# [agent] Confirm passkey 059627 (yes/no): **yes** /\*input yes\*/  
click “pair” on cellphone

## 10. HW NAT support.

### 10.1. Enable hw nat Support.

#### ■ *Kernel menuconfig*

```
CONFIG_RTL_HW_NAPT=y
make kernel_menuconfig
Device Drivers --->
  [*] Network device support --->
    [*] Options for Realtek SoC --->
      Config for Layered Driver Features --->
        Hardware Features Selection (Enable RTL Hardware NAPT)
```

```
CONFIG_RTL_HW_NAPT_REFINE_KERNEL_HOOKS=y
make kernel_menuconfig
Device Drivers --->
  [*] Network device support --->
    [*] Options for Realtek SoC --->
      [*] Hw Nat Refine Kernel Hooks
```

### 10.2. How to enable/disable hw nat on console.

enable: echo 1 > /proc/hw\_nat

disable: echo 0 > /proc/hw\_nat

(note: if use pppoe dial, please use interface name "ppp0" on luci, you can check with console command "ifconfig", get name with "pppoe-ppp0" interface)

### 10.3. Hardware NAT characteristic & limitation.

Following is the characteristic & limitation about hardware nat

- 1) support static/dhcp/pppoe wan type, not support vpn tunnel (PPTP/L2TP/multi-PPPoE wan type).
- 2) not support IP conntracks which need do ALG.
- 3) when URL filter enabled, must disable hardware nat.
- 4) when enable software qos, must disable hardware nat.

- 5) when enable rtk vlan, must disable hardware nat.
  - 6) not support IP fragment packets.
  - 7) support ipv4, not support ipv6.
  - 8) support tcp/udp, not support icmp....
  - 9) not support encryption packets (e.g.: IPSec).
  - 10) not support A or B class network address.
  - 11) support server port / trigger port / DMZ / port mapping etc, and hardware NAT need not do anything special for the features related to iptables rules because hardware NAT is independent of the iptables rules.
  - 12) won't affect alg when enable hardware nat.
  - 13) total 1024 hardware nat entries, bi-direction support 512 connection.
- Note: If wan type changes to PPTP/L2TP/multi-PPPoE or URL filter/QoS function is enabled, hardware NAT must be manually disabled by echo correct value to "/proc/hw\_nat". Others like ALG or IP fragment will be automatically processed, no other settings involved.

## **11. fastpath support.**

### **11.1. Enable fastpath Support.**

- ***user menuconfig***
  - PACKAGE\_kmod-fastpath [=y]
  - make menuconfig
  - Kernel modules --->
  - Network Support --->
  - <\*> kmod-fastpath..... fastpath driver

### **11.2. How to enable/disable fastpath on console.**

enable: echo 1 > /proc/fast\_nat  
disable: echo 0 > /proc/fast\_nat

## 12. HW Qos support.

### 12.1. Enable Qos support.

#### ■ *user menuconfig*

make menuconfig

-> LuCI

-> Applications

-> luci-app-qos..... Quality of Service (PACKAGE\_luci-app-qos [=y])

```

^@^@^@^@^@^@^@^@^@^@^@^@ 3. Applications ^@^@^@^@^@^@^@^@^@^@^@
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:
^@^@^@^@ (-) ^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@
  < > luci-app-ntpc..... NTP time synchronisat
  < > luci-app-ocserv... OpenConnect VPN server configu
  < > luci-app-olsr..... OLSR configu
  < > luci-app-olsr-services
  < > luci-app-olsr-viz.....
  < > luci-app-p2pblock..... LuCI Support for the
  < > luci-app-p910nd..... p910nd
  < > luci-app-pbx.....
  < > luci-app-polipo..... LuCI Supp
  <+> luci-app-qos..... Quality of Serv
  < > luci-app-radvd.....
^@^@^@ (+) ^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@

```

#### ■ *Kernel menuconfig*

make kernel\_menuconfig

-> Device Drivers

-> Network device support (NETDEVICES [=y])

-> Options for Realtek SoC (RTL\_819X\_SWCORE [=y])

-> Support HW Qos (RTL\_HW\_QOS\_SUPPORT [=y])

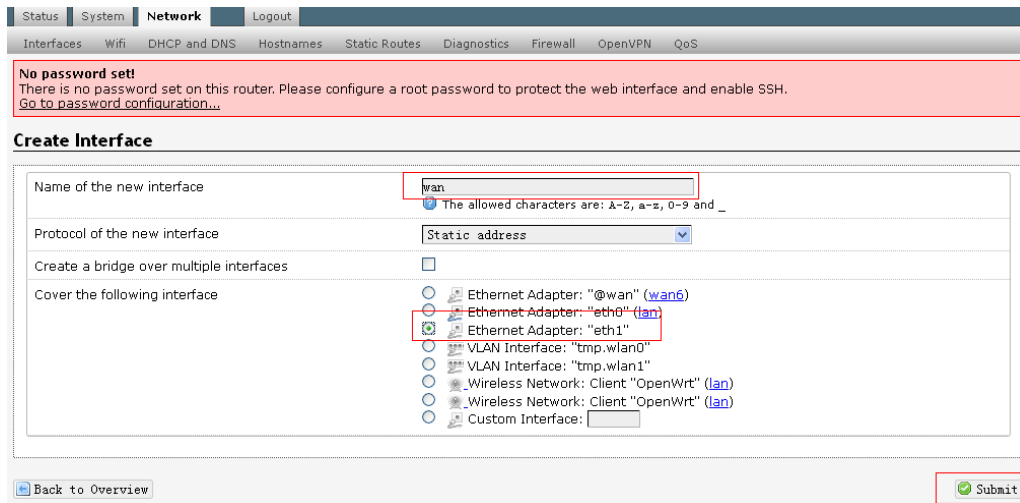


## 12.2. How to configure by ui.

Network->interfaces    add new interface







**No password set!**  
There is no password set on this router. Please configure a root password to protect the web interface and enable SSH.  
[Go to password configuration...](#)

**Create Interface**

Name of the new interface:

Protocol of the new interface:

Create a bridge over multiple interfaces: ☐

Cover the following interface:

- ☐ Ethernet Adapter: "@wan" ([wan6](#))
- ☒ Ethernet Adapter: "eth0" ([lan](#))
- ☐ Ethernet Adapter: "eth1"
- ☐ VLAN Interface: "tmp.wlan0"
- ☐ VLAN Interface: "tmp.wlan1"
- ☐ Wireless Network: Client "OpenWrt" ([lan](#))
- ☐ Wireless Network: Client "OpenWrt" ([lan](#))
- ☐ Custom Interface:

[Back to Overview](#) [Submit](#)

Notice: If need configure rtl hw qos by ui, the interface name in qos config file need to be the same name as the interface name in network config file. And the default name in qos config file is wan.

Network config file:

```
root@rtkmips:/# cat etc/config/network

config interface 'loopback'
    option ifname 'lo'
    option proto 'static'
    option ipaddr '127.0.0.1'
    option netmask '255.0.0.0'

config interface 'lan'
    option ifname 'eth0'
    option type 'bridge'
    option proto 'static'
    option ipaddr '192.168.1.1'
    option netmask '255.255.255.0'
    option ip6assign '60'

config interface 'wan6'
    option ifname '@wan'
    option proto 'dhcpv6'

config globals 'globals'
    option ula_prefix 'fd4f:7258:2e1f::/48'

config interface 'wan'
    option proto 'static'
    option ifname 'eth1'
    option ipaddr '192.168.2.123'
    option netmask '255.255.255.0'
    option gateway '192.168.2.200'
    option broadcast '192.168.2.255'
```

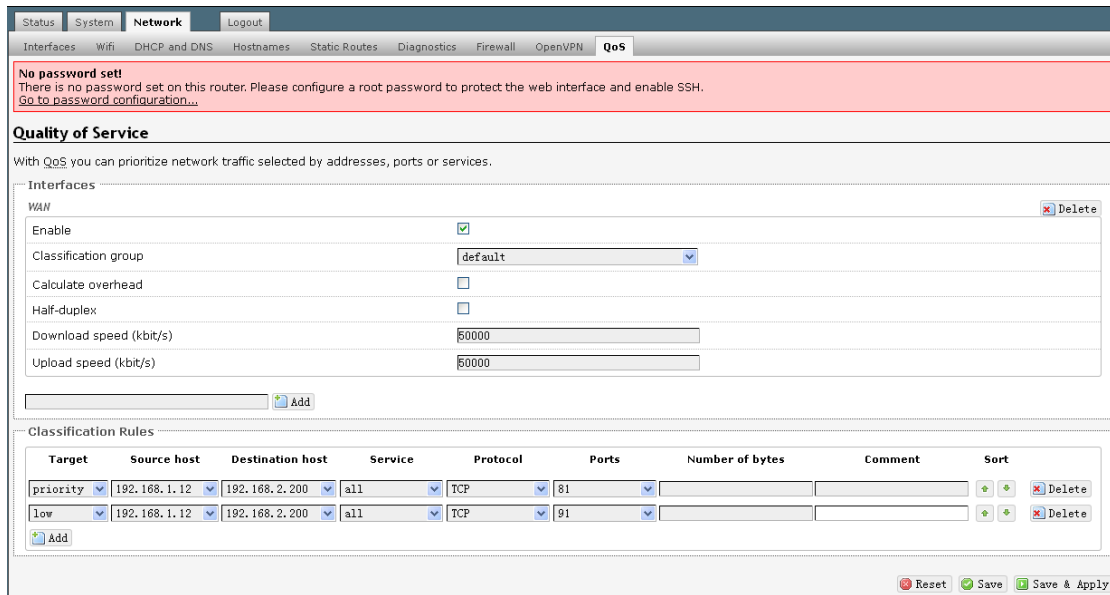
Qos config file:

```
root@rtkmips:/# cat etc/config/qos
# QoS configuration for OpenWrt

# INTERFACES:
config interface wan
    option classgroup "Default"
    option enabled 0
    option upload 128
    option download 1024

# RULES:
config classify
    option target "Priority"
    option ports "22,53"
    option comment "ssh, dns"
```

## 13.2.2 set qos rule.

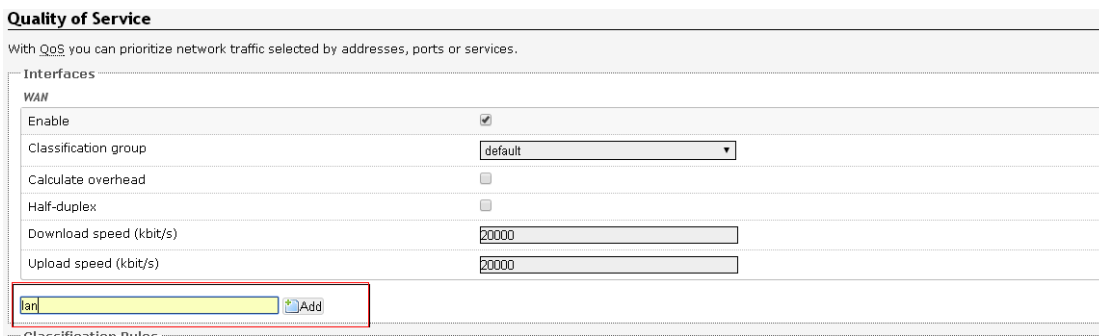


Notice:

- 1) the name in qos config file and network config file need to be the same
- 2) RTL hw qos only support qos rules by ip/port/tcp/udp.
- 3) the priority of each rule : priority> express> normal> low

## 13.2.3 wan2lan test.

- 1) Firstly add lan Interface on Qos web:



- 2) Then config downlink qos rules:

<b>LAN</b>								
Enable	<input checked="" type="checkbox"/>							
Classification group	default							
Calculate overhead	<input type="checkbox"/>							
Half-duplex	<input type="checkbox"/>							
Download speed (kbit/s)	20000							
Upload speed (kbit/s)	20000							
<input type="text"/> <input type="button" value="Add"/>								
<b>Classification Rules</b>								
Target	Source host	Destination host	Service	Protocol	Ports	Number of bytes	Comment	Sort
priority	all	192.168.1.110	all	TCP	10000			
express	all	192.168.1.110	all	TCP	20000			

## 12.3. How to configure by tc command.

Example: lan->wan limit total rate =102400kbit /s, src port 81~port 89 guaranteed min rate =51200kbit/s, and others guaranteed min rate =25600kbit/s

```
tc qdisc add dev eth1 root handle 2:0 htb default 11 r2q 64
```

```
//lan->wan limit total rate =102400kbit /s
```

```
tc class add dev eth1 parent 2:0 classid 2:1 htb rate 102400kbit ceil 102400kbit
```

```
//default class guaranteed min rate =25600kbit/s, strict max rate =102400kbit/s
```

```
tc class add dev eth1 parent 2:1 classid 2:11 htb rate 25600kbit ceil 102400kbit prio 2
```

```
tc qdisc add dev eth1 parent 2:11 handle 211: sfq perturb 10
```

```
// src port 81~port 89 guaranteed min rate =51200kbit/s, strict max rate =102400kbit/s
```

```
tc class add dev eth1 parent 2:1 classid 2:12 htb rate 51200kbit ceil 102400kbit prio 2
```

```
tc qdisc add dev eth1 parent 2:12 handle 212: sfq perturb 10
```

```
iptables -t mangle -A PREROUTING -p tcp -m tcp --source-port 81:89 -j MARK --set-mark 2
```

```
tc filter add dev eth1 parent 2:0 protocol ip prio 100 handle 2 fw classid 2:12
```

## 12.4. Limitation.

- 1) RTL hw qos Only support tc command + iptables set mark to set qos rule.
- 2) Iptablese match rule only support by ip/port/tcp/udp.
- 3) schedule support hfsc and htb, filter only support fw, output queue can't exceed 4 queues.
- 4) Only support wan type of static ip and dhcp.
- 5) the max qos rules is 8, do not exceed 8 qos rules.

## 13. Appendix

### 13.1. Modified files in the original OpenWRT

■ ***Below are the files/dir list “modified” by Realtek in the original OpenWRT***

- target/Config.in
- include/version.mk
- tools/Makefile
- scripts/feeds
- package/network/config/netifd/patches
- package/network/service/hostapd/files/netifd.sh
- package/network/service/hostapd/files/patches
- package/kernel/mac80211/files/lib/netifd/wireless/mac80211.sh
- package/kernel/mac80211/patches
- package/kernel/mac80211/Makefile
- package/system/fstools/patches
- package/system/mountd/patches
- package/network/utls/iwinfo/src/iwinfo\_lua.c

■ ***Below are the files list that “added” by Realtek in the original OpenWRT***

- rtk\_deconfig/
- rtk\_scripts/
- tools/rtk\_tools/
- target/linux/realtek/
- target/linux/rtkmips/
- staging\_dir/rsdk-4.6.4-5281-EB-3.10-0.9.33-m32ub-20141001/
- staging\_dir/rsdk-4.6.4-4181-EB-3.10-0.9.33-m32u-20141001/
- include/site/mips-rlx5281-linux/
- include/site/mips-rlx4181-linux/
- package/utls/rtk\_app
- package/kernel/rtl\_nf
- package/kernel/rtl\_fs
- package/utls/bluez/
- package/kernel/fastpath/
- package/kernel/rtl\_dev\_stats/