RALINK TECHNOLOGY, CORP.

RALINK AP SDK 3.4.0.0 USER'S MANUAL

Proprietary Notice and Liability Disclaimer

The confidential Information, technology or any Intellectual Property embodied therein, including without limitation, specifications, product features, data, source code, object code, computer programs, drawings, schematics, know-how, notes, models, reports, contracts, schedules and samples, constitute the Proprietary Information of Ralink (hereinafter "Proprietary Information")

All the Proprietary Information is provided "AS IS". No Warranty of any kind, whether express or implied, is given hereunder with regards to any Proprietary Information or the use, performance or function thereof. Ralink hereby disclaims any warranties, including but not limited warranties of non-infringement, merchantability, completeness, accuracy, fitness for any particular purpose, functionality and any warranty related to course of performance or dealing of Proprietary Information. In no event shall Ralink be liable for any special, indirect or consequential damages associated with or arising from use of the Proprietary Information in any way, including any loss of use, data or profits.

Ralink retains all right, title or interest in any Proprietary Information or any Intellectual Property embodied therein. The Proprietary Information shall not in whole or in part be reversed, decompiled or disassembled, nor reproduced or sublicensed or disclosed to any third party without Ralink's prior written consent.

Ralink reserves the right, at its own discretion, to update or revise the Proprietary Information from time to time, of which Ralink is not obligated to inform or send notice. Please check back if you have any question. Information or items marked as "not yet supported" shall not be relied on, nor taken as any warranty or permission of use.

Ralink Technology Corporation (Taiwan)

5F, No.36, Tai-Yuen Street,

ChuPei City

HsinChu Hsien 302, Taiwan, ROC

Tel +886-3-560-0868

Fax +886-3-560-0818

Sales Taiwan: Sales@ralinktech.com.tw

Technical Support Taiwan: FAE@ralinktech.com.tw

http://www.ralinktech.com/

TABLE OF CONTENTS

1 SDK HISTORY

Release	Features	Platform Support	Schedule
1.2 SDK	OS: Linux 2.4.30 Bootloader: Uboot Toolchain: GNU based cross-compiler Driver: UART, Giga Ethernet, Flash, Wi-Fi Driver Application: Bridging, Routing, NAT, PPPoE, Web server, DHCP client, DHCP server Wi-Fi features: WMM, WMM-PS, WEP, WPA/WPA2 personal, WPA/WPA2 Enterprise	RT2880 Shuttle Support IC+ 5 ports 10/100 Switch Support Marvall Giga Single Phy Support	Formal: 2007/03/20
1.3 SDK	Feature parity with 1.2 SDK plus: Application: NTP, DDNS, WebUI enhance, Vista RG (Native IPv6, LLTD), Firewall Driver: I2C, SPI, GPIO driver Wi-Fi features: Intergraded QA, WPS, mBSSID, WDS, STA mode, 802.1x Concurrent AP support	RT2880 MP Support	Beta: 2007/04/30 Formal: 2007/05/25
2.0 SDK	Feature parity with 1.3 SDK plus: File system support ramdisk and squashfs WebUI: save/restore configure. WPS PIN, WPS PBC, factory default, STA mode support Application: push button to load default configuration (GPIO reference design) Wi-Fi features: AP-Client Ethernet Converter Support	None	Beta: 2007/07/06 Formal: 2007/07/20
2.2 SDK	Feature parity with 2.0 SDK plus: AP version 1.6.0.0	Vitesse Switch Support	Formal: 2007/11/08

	STA version 1.4.0.0 Wi-Fi Certification: 802.11 b/g/n, WPA2, WMM, WMM-PS, WPS Operation Mode reorganization to "Bridge", "Gateway", and "Ethernet Converter" support iNIC driver Support Squash with LZMA file system		
2.3 SDK	Feature parity with 2.2 SDK plus: iNIC v1.1.6.1 RT2561 driver v1.1.2.0 Spansion Flash Support RT2860 AP driver v1.7 RT2860 STA driver v1.5 RT2561 WebUI Multi-Language WebUI support	IC+ 100Phy Realtek 100Phy	Formal: 2008/01/16
2.4 SDK	Feature parity with 2.3 SDK plus: iNIC v1.1.7.1 RT2860 AP driver v1.8.1.0 RT2860 STA driver v1.6.0.0 Static/Dynamic Routing Content Filtering	Mii iNIC	Formal: 2008/04/07
3.0 SDK	Feature parity with 2.4 SDK plus: OS: Linux 2.6.21 (Linux2.4 for RT2880, Linux-2.6 for RT3052) 8MB Flash Support – S29GL064N/MX29LV640 Storage Application – FTP/Samba	RT3052 Support	Formal: 2008/06/06
3.1 SDK	Feature parity with 3.0 SDK plus: RT2860 AP driver v1.9.0.0 RT2860 STA driver v1.7.0.0 [RT3052] 16MB/32MB NOR flash support [RT3052] Boot from 0xbf00.0000(MA14=1) [RT3052] Boot from 0xbfc0.0000(MA14=0)	RT2880 platforms RT3052 platforms	Formal: 2008/07/30

3.2 SDK	Feature parity with 3.1 SDK plus: RT2860 AP driver v2.0.0.0 RT2860 STA driver v1.8.0.0 GreenAP support Busybox 1.12.1 MTD-Based Flash API	RT2880 platforms RT3050 platforms RT3052 platforms	Formal: 2008/10/06
3.3 SDK	Feature parity with 3.2 SDK plus: RT2860 AP driver v2.2.0.0 RT2860 STA driver v2.1.0.0	RT2880 platforms RT3050 platforms RT3052 platforms	Formal: 2009/04/27
3.4 SDK	Feature parity with 3.3 SDK plus: RT2860 AP driver v2.4.0.0 RT2860 STA driver v2.3.0.0	RT2880 platforms RT3050 platforms RT3052 platforms RT3350 platforms RT3883 platforms RT3662 platforms	Formal: 2010/02/12

2 VERSION HISTORY

Release	Features	Date	Author
1.2	Initial release		Steven Liu
1.3	WebUI – NTP/DDNS, iNIC		Steven Liu
	I2C, SPI, GPIO Linux driver		
2.0	Squashfs tools installation		Steven Liu
	WebUI - save/restore configure. WPS , factory default		
	WebUI – STA, Ethernet Converter mode		
2.2	WebUI - Operation Mode reorganization		Steven Liu
	How to downsize image		
2.3	How to control GPIO and LED		Steven Liu
	Install mksquashfs Utility		
	Describes Uboot configuration file		
	Add new parameter in default setting		
2.4	WebUI – How to save the configurations to the flash		Winfred Lu
3.0	Updated for RT3052		Steven Liu
	Chapter Re-organization		
3.1	Update default parameter for LED firmware		Steven Liu
	Update GPIO definition for RT3052 platform		
	Update FAQ		
3.2	Reorganize user manual		Steven Liu / Winfred
	Update FAQ		
	-How to enable NFS Client		
	-How to add new language to webUI		
	- How to Power down rt305x Ethernet ports		
	- How to enable USB storage in RT305x platform		
	-How to enable USB automount in RT305x platform		
3.3	Update FAQ		Steven
	-How to enable software QoS		
	- How to enable USB Ethernet		
	- How to build a single image for the RT2880 8M flash platform		
	- How to start printer server		
	-How to force link speed		
3.4	- How to burn SPI Uboot firmware		Steven
	-How to enable new watchdog		
	-How to verify IGMP snooping		

3 OVERVIEW OF THE RALINK AP DEMO BOARD

3.1 RT2880

The RT2880 SOC combines Ralink's 802.11n draft compliant 2T3R MAC/BBP, a high performance 266-MHz MIPS4KEc CPU core, a Gigabit Ethernet MAC and a PCI host/device, to enable a multitude of high performance, cost-effective 802.11n applications. The RT2880 has two RF companion chips: The RT2820, for 2.4G-band operation; and the RT2850, for dual band 2.4G or 5G operations. In addition to traditional AP/router applications, the chipset can be implemented as a WLAN "intelligent" NIC, drastically reducing the load on the host SOC, such as DSL/Cable or Multimedia Applications processors. Users can treat the WLAN iNIC as a simple Ethernet device for easy porting and guaranteed 802.11n WLAN performance without the need to upgrade to an expensive host SOC.

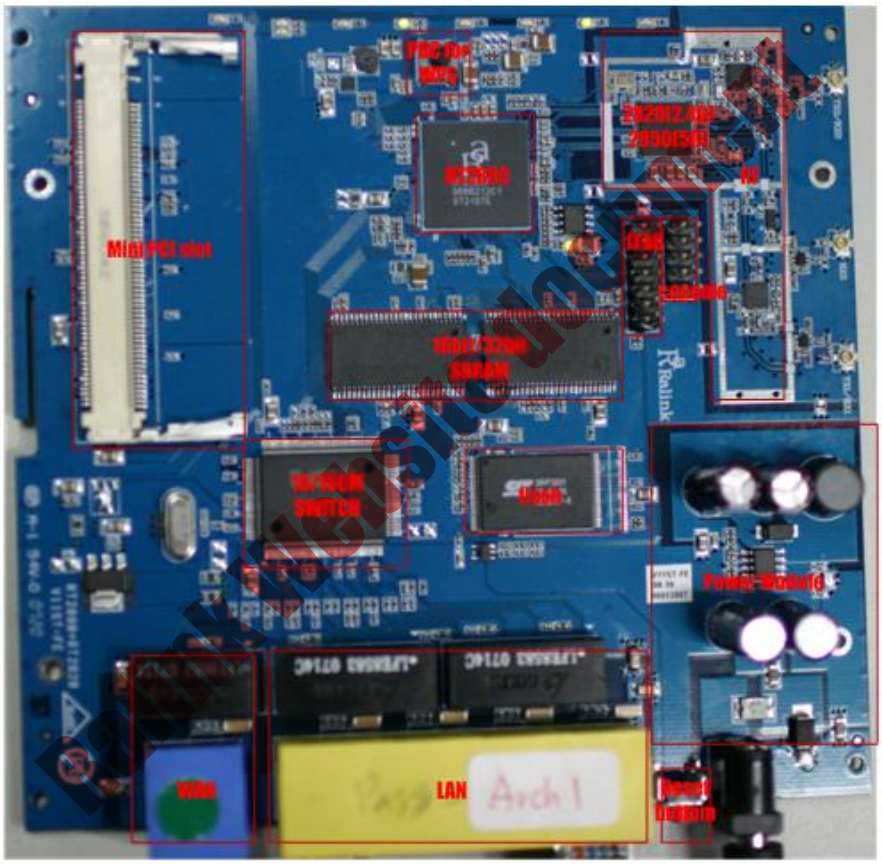


Figure 1 The RT2880 Demo Board

Table 1 RT2880 Memory Mapping

Address Range (hex)			Size	Block Name
0000.0000	-	001F.FFFF	2M	Reserved
0020.0000	-	0020.1FFF	8K	Reserved

0020.2000	-	0020.3FFF	8K	Reserved
0020.2000	-	0020.5FFF	8K	Reserved
0020.6000	-	002F.FFFF	1024K	Reserved
0030.0000	-	0030.00FF	256	System Control
0030.0100	-	0030.01FF	256	Timer
0030.0200	-	0030.02FF	256	Interrupt Controller
0030.0300	-	0030.03FF	256	Memory Controller
0030.0400	-	0030.04FF	256	Reserved
0030.0500	-	0030.05FF	256	UART
0030.0600	-	0030.06FF	256	Programmable I/O
0030.0700	-	0030.07FF	256	Reserved
0030.0800	-	0030.08FF	256	Reserved
0030.0900	-	0030.09FF	256	I2C
0030.0A00	-	0030.0AFF	256	Reserved
0030.0B00	-	0030.0BFF	256	SPI
0030.0C00	-	0030.0CFF	256	UART Lite
0030.0D00	-	0030.0DFF	256	Reserved
0030.0F00	-	0030.0FFF	256	Reserved
0030.1000	-	0030.FFFF	1020K	Reserved
0040.0000	-	0040.FFFF	64K	Frame Engine
0041.0000	-	0041.FFFF	64K	Embedded 16KB ROM (wrap-around in the 64KB space)
0042.0000	-	0042.FFFF	64K	PCM Controller
0043.0000	-	0043.FFFF	64K	Reserved
0044.0000	-	0047.FFFF	256K	PCI Host/Device Controller
0048.0000	-	004B.FFFF	256K	802.11n MAC/BBP
004C.0000	-	004F.FFFF	256K	Reserved
0050.0000	-	0053.FFFF	256K	Reserved
0054.0000	-	007F.FFFF	2816K	Reserved
0080.0000	-	0080.7FFF	32K	Reserved
0080.8000	-	0080.FFFF	32K	Reserved
0081.0000	-	0081.FFFF	64K	Reserved
0082.0000	-	0082.FFFF	64K	Reserved
0083.0000	-	0083.FFFF	64K	Reserved
0084.0000	-	0088.FFFF	256K	Reserved
0100.0000	-	01FF.FFFF	16M	External SRAM
0800.0000	-	0BFF.FFFF	64M	SDRAM
0C00.0000	-	0FFF.FFFF	64M	SDRAM

1000.0000	-	1003.FFFF	256K	Reserved
1004.0000	-	1007.FFFF	256K	Reserved
1008.0000	-	100B.FFFF	256K	Reserved
100C.0000	-	100F.FFFF	256K	Reserved
1010.0000	-	1BFF.FFFF	192M	Reserved
1C00.0000	-	1FFF.FFFF	64M	External Flash
2000.0000	-	2FFF.FFFF	256M	PCI Memory Space
3000.0000	-	FFFF.FFFF	3.25G	Reserved

3.2 RT3052

The RT3052 SOC combines Ralink's 802.11n draft compliant 2T2R MAC/BBP/RF, a high performance 384MHz MIPS24KEc CPU core, 5-port integrated 10/100 Ethernet switch/PHY, an USB OTG and a Gigabit Ethernet MAC. There are very few external components required for 2.4GHz 11n wireless products with the RT3052. It employs Ralink's 2nd generation 11n technologies for longer range and better throughput. The embedded high performance CPU can process advanced applications effortlessly, such as routing, security and VOIP. The USB port can be configured to access external storage for Digital Home applications. The RT3052 also has rich hardware interfaces (SPI/I2S/I2C/UART/GMAC) to enable many possible applications.

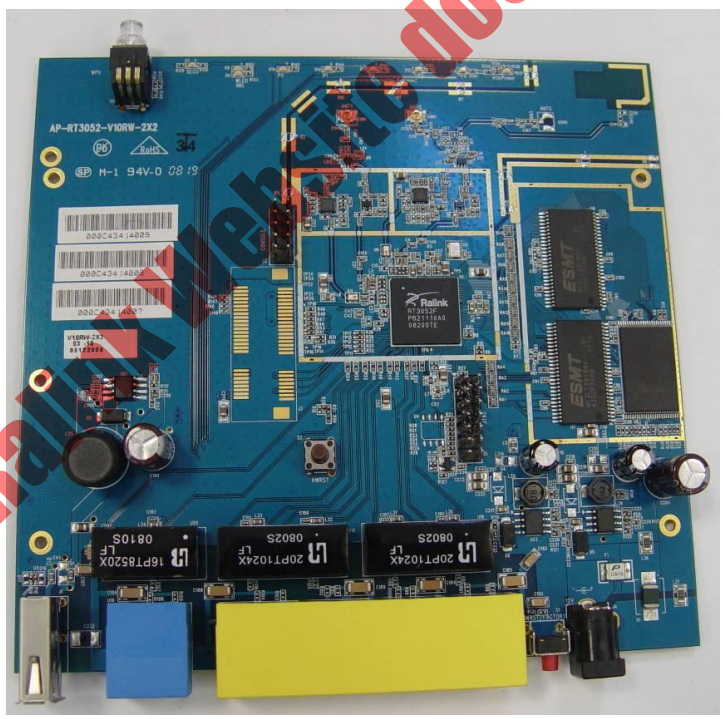


Figure 2 The RT3052 Demo Board

Table 2 RT3052 Memory Mapping

0000.0000	-	03FF.FFFF	64M	SDRAM
0400.0000	-	0FFF.FFFF		<<Reserved>>
1000.0000	-	1000.00FF	256	SYSCTL
1000.0100	-	1000.01FF	256	TIMER
1000.0200	-	1000.02FF	256	INTCTL
1000.0300	-	1000.03FF	256	MEM_CTRL (SDRAM & Flash/SRAM)
1000.0400	-	1000.04FF	256	PCM
1000.0500	-	1000.05FF	256	UART
1000.0600	-	1000.06FF	256	PIO
1000.0700	-	1000.07FF	256	Generic DMA
1000.0800	-	1000.08FF	256	NAND Flash Controller
1000.0900	-	1000.09FF	256	I2C
1000.0A00	-	1000.0AFF	256	I2S
1000.0B00	-	1000.0BFF	256	SPI
1000.0C00	-	1000.0CFF	256	UARTLITE
1000.0D00	-	100F.FFFF		<<Reserved>>
1010.0000	-	1010.FFFF	64K	Frame Engine
1011.0000	-	1011.7FFF	32K	Ethernet Switch
1011.8000		1011.9FFF	8K	ROM
1011_a000		1011_FFFF		<<Reserved>>
1012.0000	-	1012.7FFF	32K	<<Reserved>>
1012.8000		1012.FFFF	32K	<<Reserved>>
1013.0000	-	1013.7FFF	32K	<<Reserved>>
1013.8000	-	1013.FFFF	32K	<<Reserved>>
1014.0000	-	1017.FFFF	256K	<<Reserved>>
1018.0000	-	101B.FFFF	256K	802.11n MAC/BBP
101C.0000	-	101F.FFFF	256K	USB OTG
1020.0000	-	1AFF.FFFF		<<Reserved>>
1B00.0000	-	1BFF.FFFF	16MB	External SRAM/Flash
1C00.0000	-	1EFF.FFFF		<<Reserved>>
1F00.0000	-	1FFF.FFFF	16MB(flash) or 4KB(ram) or 8KB(rom)	When BOOT_FROM = 2'b00, <16MB external 16-bit flash is mapped. When BOOT_FROM = 2'b01, <8MB external 8-bit flash is mapped. When BOOT_FROM = 2'b10, 4KB internal boot RAM is mapped for boot from NAND application.

				When BOOT_FROM = 2'b11, 8KB internal boot ROM is mapped for iNIC application.

3.3 RT3883

The RT3883 SOC combines Ralink's 802.11n draft compliant 3T3R MAC/BBP/RF, a high performance 500MHz MIPS74Kec CPU core, a Gigabit Ethernet MAC, and a USB Host/Device. With the RT3883, there are very few external components required for 2.4/5GHz 11n wireless products. The RT3883 employs Ralink 2nd generation 11n technologies for longer range and better throughput. The embedded high performance CPU can process advanced applications effortlessly, such as WI-FI data processing without overloading the host processor. In addition, the RT3883 has rich hardware interfaces (SPI/ I2S/ I2C/ PCM/ UART/ USB/ PCI/ PCIe/ RGMII/ MII) to enable many possible applications.

Figure 3 The RT3883 Demo Board

Table 3 RT3052 Memory Mapping

Start		End	Size	Description
0000.0000	-	0FFF.FFFF	256 M	DDR2 256MB/SDRAM 128MB
1000.0000	-	1000.00FF	256	SYSCTL
1000.0100	-	1000.01FF	256	TIMER
1000.0200	-	1000.02FF	256	INTCTL
1000.0300	-	1000.03FF	256	MEM_CTRL (SDR/DDR)
1000.0400	-	1000.04FF	256	<<Reserved>>
1000.0500	-	1000.05FF	256	UART
1000.0600	-	1000.06FF	256	PIO
1000.0700	-	1000.07FF	256	Flash Controller (NOR/SRAM)
1000.0800	-	1000.08FF	256	NAND Controller
1000.0900	-	1000.09FF	256	I2C
1000.0A00	-	1000.0AFF	256	I2S
1000.0B00	-	1000.0BFF	256	SPI
1000.0C00	-	1000.0CFF	256	UARTLITE
1000.0D00	-	1000.0DFF		<<Reserved>>
1000.2000	-	1000.27FF	2 K	PCM (up to 16 channel)
1000.2800	-	1000.2FFF	2 K	Generic DMA (up to 64 channel)
1000.3000	-	1000.37FF	2 K	CODEC 1
1000.3800	-	1000.3FFF	2 K	CODEC 2
1000.4000	-	100F.FFFF		<<Reserved>>
1010.0000	-	1010.FFFF	64 K	Frame Engine
1011.0000	-	1011.7FFF	32 K	<<Reserved>>
1011.8000		1011.BFFF	16 K	ROM
1011.C000	-	1011.FFFF	16 K	<<Reserved>>
1012.0000	-	1012.7FFF	16 K	USB Device
1012.8000	-	1012.FFFF	16 K	<<Reserved>>
1013.0000	-	1013.7FFF	32 K	<<Reserved>>
1013.8000	-	1013.FFFF	32 K	<<Reserved>>
1014.0000	-	1017.FFFF	256 K	PCI/ PCI Express
1018.0000	-	101B.FFFF	256 K	802.11n MAC/BBP
101C.0000	-	101F.FFFF	256 K	USB Host
1020.0000	-	1023.FFFF	256 K	<<Reserved>>
1024.0000	-	1027.FFFF	256 K	<<Reserved>>

1028.0000	-	1BFF.FFFF		<<Reserved>>
1C00.0000	-	1DFF.FFFF	16KB ROM or 32MB 16-bit Flash or 16MB 8-bit Flash	When BOOT_FROM = 3'b000, up-to 32MB external 16-bit flash is mapped. When BOOT_FROM = 3'b001, up-to 16MB external 8-bit flash is mapped. When BOOT_FROM = 3'b010/3'b011/3'b100, 16KB internal boot ROM is mapped.
1E00.0000	-	1FFF.FFFF		External SRAM/Flash
2000.0000	-	2FFF.FFFF	256 M	PCI/PCIe Memory Space

4 AP SDK SOURCE CODE OVERVIEW

The subsequent command is used in the development environment. It makes a directory equivalent to "/home/${user}/RT288x_SDK".

#tar jxvf RT288x_SDK_{version}_{date}.tar.bz2

- The RT288x_SDK package contains the subsequent directories.
  - toolchain : mips toolchain
  - source : Linux kernel source
  - tools :useful script
- The source directory contains the subsequent directories.
  - config : auto-configuration files
  - images : Linux image
  - lib : uClibc 0.9.28
  - linux-2.4.x : Linux kernel source for RT2880
  - linux-2.6.21.x : Linux kernel source for RT3052
  - romfs : root file system (uncompressed)
  - tools : useful script to generate rootfs
  - user : user applications
  - vendor : init scripts of target platform (inittab, rcS...etc)

5 TOOL-CHAIN

The Ralink AP SDK uses buildroot to make the Linux kernel image. Buildroot is a set of Makefiles and patches. It is easy to make a cross-compilation toolchain and root file system for the target Linux system. Use the uClibc C library.

5.1 Install toolchain

#cp RT288x_SDK/toolchain/buildroot-gcc342.tar.bz2 /opt

#$ tar jxvf buildroot-gcc342.tar.bz2

The extract procedure makes a directory equivalent to "/opt/buildroot-gdb"

5.2 Install LZMA Utility

lzma is necessary to make the compressed kernel image. The Ralink RT2880 SDK uses lzma to compress the kernel image.

#cd RT288x_SDK/toolchain/lzma-4.32.0beta3
#./configure
#make
#make install (install lzma to /usr/local/bin)

Use gzip or lzma to compress the kernel image.

Make changes to RT288x_SDK/source/vendors/Ralink/{Platform}/Makefile

```
COMP = gzip
```

Use gzip to compress the Linux kernel image.

```
COMP = lzma
```

Use lzma to compress the Linux kernel image.

5.3 Install mksquashfs utility

mksquashfs-lzma is necessary to make the compressed rootfs. The Ralink AP SDK uses mksquashfs with lzma to compress the root filesystem.

Linux-2.4.x Kernel Version

#cd RT288x_SDK/toolchain/mksquash_lzma-3.0

#make

#make install (install mksquashfs-lzma to /opt/buildroot-gdb/bin/mksquashfs_lzma-3.0)

Linux-2.6.21.x Kernel Version

#cd RT288x_SDK/toolchain/mksquash_lzma-3.2

#make

#make install (copy mksquashfs/lzma_alone to /opt/buildroot-gdb/bin/)

LZMA_ALONE IS NECESSARY TO MAKE YOUR OWN RAMDISK IMAGE, IF YOU TURN ON "COMPRESS RAMDISK BY LZMA" FOR RT3052.

#make menuconfig

Kernel/Library/Defaults Selection --->

Machine selection --->

[*] Compress ramdisk by lzma instead of gzip

6 BOOT LOADER

6.1 Uboot Configuration

tar jxvf Uboot_{version}_{BETA/FINAL}_{date}.tar.bz2

#cd Uboot

#make menuconfig

1. Set the DRAM Size

DRAM Component:

	Row	Column
64Mb	12	8
128Mb	12	9
256Mb	13	9

DRAM Bus: 16bits / 32bits

Example:

- W9825G6EH: 4Mx4Banksx16bits SDRAM:
    - Row Address: A0-A12, Column address: A0-A8
    - DRAM Component=256Mb
    - DRAM Bus =16bits
- W981216DH/W9812G6DH: 2Mx4Banksx16bits SDRAM:
    - Row Address: A0-A11, Column address: A0-A8
    - DRAM Component=128Mb
    - DRAM Bus =16bits
- IS42S32800B: 2Mx4Banksx32bits SDRAM:
    - Row Address: A0-A11, Column address: A0-A8
    - DRAM Component=128Mb
    - DRAM Bus =32bits

2. LAN/WAN Partition

The switch automatically operates in dump switch mode when the board turns on. Clients on the LAN get the dynamic IP address from the remote DHCP server connected to the WAN port.

Set the LAN/WAN partition to prevent the Client's DHCP request being sent to the WAN side.

6.2 Build the uboot Image

```
# make
.............
```

NOR Flash: **uboot.bin** is located in Uboot/.

```
# cp uboot.bin /tftpboot
```

SPI Flash: **uboot.img** is located in Uboot/

```
# cp uboot.img /tftpboot
```

NAND Flash: **uboot.img** is located in Uboot/

```
# cp uboot.img /tftpboot
```

6.3 Burn the uboot image

Press '9' on the Uboot menuconfig, to open the invisible menu.

Set the operation:

1: Load system code to SDRAM via TFTP

2: Load system code then write to Flash via TFTP

3: Boot system code via Flash (default)

4: Enter boot command line interface

5: Load ucos code to SDRAM via TFTP

You chose 9

9: System Load Boot Loader then write to Flash via TFTP.

Warning! Erase Boot Loader in Flash then burn new one. Are you sure? (Y/N) Please Input new ones /or Ctrl-C to discard

Input device IP (10.10.10.123) ==:

Input server IP (10.10.10.99) ==:

Input Uboot filename (uboot.bin) ==:

7 USER LIBRARY

7.1 Library Configuration

RT288x_SDK uses ulibc 0.9.28 for user applications. The subsequent instructions show how to change the default library setting.

```
# make menuconfig
Kernel/Library/Defaults Selection    --->
[ *] Customize uClibc Settings
```

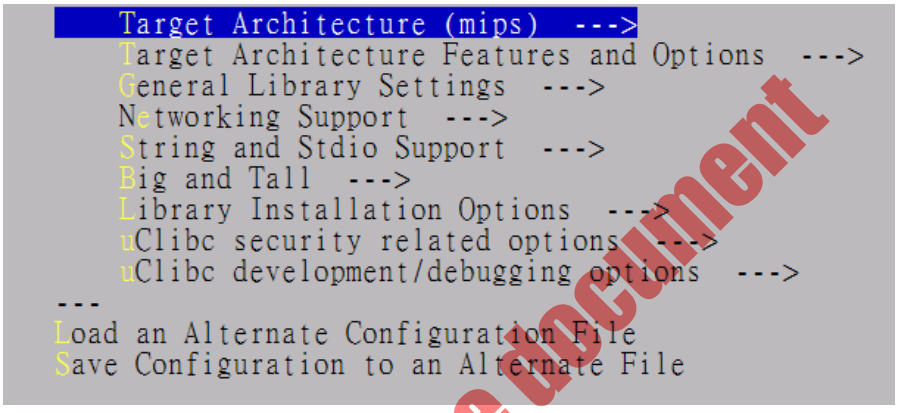


Figure 4 uClib configuration Menu

7.2 Library Porting

The subsequent instructions show how to add a new library to the RT288x_SDK.

Example: Port libtest to RT288x_SDK

1. *#/ cp -r libtest to RT288x_SDK/source/lib*
2. modify RT288x_SDK/source/lib/libtest/Makefile
   [you can reference to libnvram/Makefile]
3. modify RT288x_SDK/source/lib/Makefile

ifeq ($(CONFIG_LIB_LIBTEST_FORCE),y)

DIRS += libtest

endif

ifeq ($(CONFIG_LIB_LIBTEST_FORCE),y)

@$(MAKE) -C libtest shared

endif

4. modify RT288x_SDK/source/config/config.in

bool 'Build libtest' CONFIG_LIB_LIBTEST_FORCE

#/ make menuconfig

You can see the “Build libtest” on the menu.

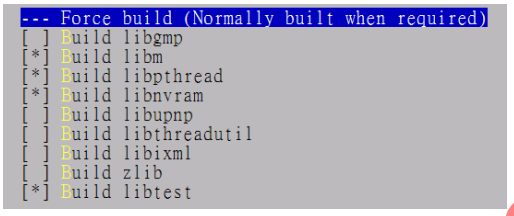


Figure 5 User Library Configure Menu

5. Compile your new library

#make dep

#make lib_only

7.3 Build user library

 # cd RT288x_SDK/source

 # make lib_only

 # make romfs

.....

The shared libraries are shown in RT288x_SDK /source/romfs/lib

8 USER APPLICATION

Many useful network applications (e.g. wan protocol, http server, debugging tools, etc.) are supplied with the RT288x_SDK to make porting easier.

8.1 Ralink Proprietary Applications

8.1.1 ATED

Description: for rt2860 v1.4 ATE test program

Usage: ate

Note:

- Execute ate on the demo board
- Connect directly from the LAN port to the PC
- Execute QA on the PC (wait 30 seconds)

8.1.2 REG

Description: register the read/write test program

Usage: reg [r/w/s] [offset] [value]

Note:

- To use system register: reg s 0
- To use wireless register: reg s 1 To use other base address offset: reg s [offset]
- The rt_rdm module must be put in first

Example:

/ # reg s 0

/ # reg r 18 / read A0300018 */*

/ # reg w 18 12345678 / write 0x12345678 to A0300018 */*

8.1.3 FLASH

Description: flash read/write test program

Usage:

a. read: flash -r [offset(hex)] -c [num of bytes]
b. write: flash -w [offset(hex)] -o [value(hex)] -c [num of bytes]
c. erase: flash -f [first sector_num] -l [last sector_num

Example:

a. read: flash -r 370000 -c 4
b. write: flash -w 370000 -o 1234 -c 4
c. erase: flash -f 60 -l 61

8.1.4 GPIO

Description: GPIO test program

Usage: GPIO [r/w/i/l]

The name of the GPIO testing user application is "*gpio*".

- gpio w: writing test (output)
- gpio r: reading test (input)
- gpio i (<gpio>): interrupt test for GPIO number
- gpio l <gpio> <on> <off> <blinks> <rests> <times>: set led on <gpio>(0~24) on/off interval, no. of blinking/resting cycles, blinking time

Pin sharing scheme

It is important to know what normal function pins are shared with the GPIO pins. Only one normal function and GPIO can operate at the same time.

- GPIOMODE: GPIO purpose select)
  Configure the pins to use as GPIO.
- PIODIR: programmed I/O direction
  Configure the direction of all GPIO pins to use as GPIO.
  an output is set as '1', and an input pin is set as '0'.
- PIODATA: programmed I/O data
  Write data for output GPIO pins, and read data for input GPIO pins. PIOSET, PIORESET, PIOTOG are also used for adjusting GPIO data bits.
- PIOINT, PIOEDGE, PIORENA, and PIOFMASK should be set when using GPIO pins for input that causes an interruption.

8.1.5 MII_MGR

Description: mii register read/write test program

Usage:

a. get: mii_mgr -g -p [phy number] -r [register number]

b. set: mii_mgr -s -p [phy number] -r [register number] -v [0xvalue]

Example:

a. get: mii_mgr -g -p 3 -r 4

b. set: mii_mgr -s -p 4 -r 1 -v 0xff11

Kernel Module:

$SDK/source/$LINUX/drivers/net/raeth/mii_mgr.c

$SDK/source/$LINUX/drivers/net/raeth/ra_ioctl.h

- IOCTL Commands
  - RAETH_MII_READ
    - Get phy register via the mdc/mdio interface.
  - RAETH_MII_WRITE
    - Set phy register via the mdc/mdio interface.
- IOCTL interface

```
typedef struct ralink_mii_ioctl_data {
    __u32    phy_id;
    __u32    reg_num;;
   __u32    val_in;
   __u32    val_out;
};
```

  - phy_id: Address of PHY device
  - reg_num: Register addresses within PHY device
  - val_ine:
    - GET: the phy register data that is read from phy
    - SET: the current register data after MDIO setting
  - Val_out: the phy register data that wants to be set
  -

User applications run mii_mgr commands through the ioctl interface to the raeth driver.

8.1.6 MTD

Description: MTD writing program for firmware update

Usage: mtd_write -r write [file] [device]

Example: mtd_write -r write image.bin mtd4

8.1.7 NVRAM

Description:

a. get value in NVRAM for RT2860 or INIC platform
b. set value in NVRAM for RT2860 or INIC platform
c. display all configurations in NVRAM, or generate .dat files

nvram_daemon is a daemon and register for NVRAM settings, or setting NVRAM values referring to a given file. It receives interruptions from GPIO pin 0. If SIGUSR1 is received (user one-clicked GPIO pin 0 button), nvram_daemon tells the GoAhead web server to start the WPS PBC procedure by sending it SIGUSR1. If SIGUSR2 is received (user pressed GPIO pin 0 button for several seconds), nvram_daemon will restore the system configuration to the default values.

Usage:

a. get: nvram_get [<2860/inic>] <field>
b. set: nvram_set [<2860/inic>] <field>
c. init: ralink_init <command> [<platform>] [<file>]

Commands:

- rt2860_nvram_show (display rt2860 values in nvram)
- inic_nvram_show (display inic values in nvram)
- show (display values in nvram for <platform>)
- gen (generate config file from nvram for <platform>)
- renew (replace nvram values for <platform> with <file>)

Platform:

- 2860 - rt2860 station
- inic - intelligent nic

File: File name for renew command

daemon: nvram_daemon

Example:

a. nvram_get 2860 SSID /* get the SSID */
b. nvram_set 2860 SSID ralink /* set the SSID to ralink */
c. ralink_init gen 2860 /* generate the RT2860 .dat file from NVRAM */
d. ralink_init show inic /* display the INIC configurations in NVRAM */
e. ralink_init renew 2860 ra.dat /* set NVRAM values for RT2860 platform according to ra.dat file */
f. nvram_daemon /* start the nvram_daemon */

8.1.8 SPICMD

Description: SPI Toolkit for SPI EEPROM Read/Write Program...

Usage: spicmd read/write parameters

Note:

- spicmd read the address
- spicmd writes the size address value
- size is 1, 2, 4 bytes

8.1.9 I2CCMD

Description: I2C Toolkit for EEPROM Read/Write via I2C Interface...

Usage: i2ccmd read/write parameters

Note:

- i2ccmd read the address
- i2ccmd write the size address value
- size is 1, 2, 4 bytes

8.1.10 Script

Description: WebUI configuration script.

Usage: Refer to the script help message.

8.2 goahead

Source code: RT288x_SDK/source/user/goahead/

Description: WebUI reference design of the AP/Router Solution.

8.3 nvram library

Source code: RT288x_SDK/source/lib/libnvram

Description: Library for nvram_get, nvram_set and ralink_init.

8.4 wsc_upnp

Source code: RT288x_SDK/source/user/WSC_UPNP

Description: Ralink WPS (Wi-Fi Protected Setup) UPNP Daemon

Required library: libupnp, pthread

8.5 iptables

Source code:

RT288x_SDK/source/user/iptables # for Linux-2.4

RT288x_SDK/source/user/ iptables-1.4.0rc1 #for Linux-2.6

Description: Administration tool for IPv4 packet filtering and NAT.

8.6 ntpclient

Source code: RT288x_SDK/source/user/ntpclient

Description: ntpclient is an NTP (RFC-1305) client for Unix-like computers. Its functionality is a small subset of xntpd, but it appears to perform better (or at least has the ability to function better) within that limited scope. It is much smaller than xntpd and is more applicable to embedded computers.

8.7 mtd-utils

Source code: RT288x_SDK/source/user/ mtd-utils

Description: for jffs2 file system support erase/format…etc. example: mkfs.jffs2, erase, eraseall

8.8 ppp-2.4.2

Source code: RT288x_SDK/source/user/ ppp-2.4.2

Description: a package which uses the Point-to-Point Protocol (PPP) to supply Internet connections over serial lines.

8.9 bridge-utils

Source code: RT288x_SDK/source/user/ bridge-utils

Description: brctl is used to set up, maintain, and inspect the Ethernet bridge configuration in the Linux kernel. An Ethernet bridge is a device commonly used to connect different networks of the Ethernet together, so that the Ethernets will appear as one Ethernet to the participants. Each of the Ethernets being connected corresponds to one physical interface in the bridge. These individual Ethernets are bundled into one bigger ('logical') Ethernet. This bigger Ethernet corresponds to the bridge network interface.

8.10 wireless_tools

Source code: RT288x_SDK/source/user/ wireless_tools

Description: This package contains the Wireless tools. The wireless tools are used to control the Wireless Extensions. The Wireless Extensions is an interface that lets you set the Wireless LAN specific parameters and get the specific stats.

8.11 inadyn

Source code: RT288x_SDK/source/user/ inadyn

Description: INADYN is a dynamic DNS client. It maintains the IP address of a host name. It periodically checks if the IP address stored by the DNS server is the real current address of the machine that is running INADYN

8.12 zebra-0.95a_ripd

Source code: RT288x_SDK/source/user/ zebra-0.95a_ripd

Description: GNU Zebra is free software that manages various IPv4 and IPv6 routing protocols. Currently GNU Zebra supports BGP4, BGP4+, OSPFv2, OSPFv3, RIPv1, RIPv2, and RIPng.

8.13 wpa_supplicant-0.5.7

Source code: RT288x_SDK/source/user/ wpa_supplicant-0.5.7

Description: WPA Supplicant (Supported WPA/IEEE 802.11i)

8.14 totd-1.5

Source code: RT288x_SDK/source/user/ totd-1.5

Description: Totd is a small DNS proxy nameserver that supports IPv6 only hosts/networks that communicate with the IPv4 world using some translation mechanism.

8.15 samba-3.0.2

Source code: RT288x_SDK/source/user/ samba-3.0.2

Description: Samba is an Open Source/Free Software suite that has, since 1992, provided file and print services to all manner of SMB/CIFS clients, including the numerous versions of Microsoft Windows operating systems. Samba is freely available under the GNU General Public License.

8.16 radvd-1.0

Source code: RT288x_SDK/source/user/ radvd-1.0

Description: The router advertisement daemon (radvd) is run by Linux or BSD systems acting as IPv6 routers. It sends Router Advertisement messages, specified by RFC 2461, to a local Ethernet LAN periodically and when requested by a node sending a Router Solicitation message. These messages are required for IPv6 stateless auto configuration.

8.17 pptp-client

Source code: RT288x_SDK/source/user/ pptp-client

Description: pptp is an implementation of the PPTP protocol for Linux and other Unix systems.

8.18 rp-l2tp-0.4

Source code: RT288x_SDK/source/user/ rp-l2tp-0.4

Description: This is a user-space implementation of L2TP (RFC 2661) for Linux

8.19 ctorrent-dnh3.2

Source code: RT288x_SDK/source/user/ ctorrent-dnh3.2

Description: CTorrent is a BitTorrent Client program written in C/C++ for FreeBSD and Linux. CTorrent is fast and small.

8.20 dhcp6

Source code: RT288x_SDK/source/user/ dhcp6

Description: DHCPv6 is a stateful address auto-configuration protocol for IPv6, a counterpart to IPv6 stateless address auto-configuration protocol. It can be used independently or coexist with its counterpart protocol. This protocol uses client/server mode of operation but also provides support through a Relay Agent. It is currently being defined by IETF DHC WG. The specification is still in the draft form.

8.21 dnsmasq-2.40

Source code: RT288x_SDK/source/user/ dnsmasq-2.40

Description: Dnsmasq is a lightweight, easy to configure DNS forwarder and DHCP server. It is designed to provide DNS and, optionally, DHCP, to a small network. It can serve the names of local machines which are not in the global DNS. The DHCP server integrates with the DNS server and allows machines with DHCP-allocated addresses to appear in the DNS with names configured either in each host or in a central configuration file. Dnsmasq supports static and dynamic DHCP leases and BOOTP/TFTP for network booting of diskless machines.

8.22 igmpproxy

Source code: RT288x_SDK/source/user/ igmpproxy

Description: IGMPproxy is a simple mulitcast router for Linux that only uses the IGMP protocol.

8.23 matrixssl-1.8.3

Source code: RT288x_SDK/source/user/ matrixssl-1.8.3

Description: MatrixSSL is an embedded SSL implementation designed for small footprint applications and devices. It is an open-source software package available under the GNU license. It consists of a single library file with a simple API set that an application writer can use to secure their application.

8.24 rp-pppoe-3.8

Source code: RT288x_SDK/source/user/ rp-pppoe-3.8

Description: pppoe is a user-space redirector which permits the use of PPPoE (Point-to-Point Over Ethernet) with Linux. PPPoE is used by many DSL service providers.

8.25 usb_modeswitch-0.9.5

Source code: RT288x_SDK/source/user/ usb_modeswitch-0.9.5

Description: USB_ModeSwitch is (surprise!) a small mode switching tool for controlling "flip flop" (multiple device) USB gear. Several new USB devices (especially high-speed WAN stuff, they're expensive anyway) have their MS Windows drivers onboard; when plugged in for the first time they act like a flash storage and start installing the driver from there. After that (and on every consecutive plugging) this driver switches the mode internally, the storage device vanishes (in most cases), and a new device (like an USB modem) shows up. Some call that feature "ZeroCD".

8.26 Port new user application

Example: Add hello application to /bin

(a) Create hello directory in RT288x_SDK/source/user

#mkdir RT288x_SDK/source/use/hello

(b) Add Makefile to RT288x_SDK/source/user/hello

```
EXEC = hello

OBJS = hello.o

CFLAGS +=

all: $(EXEC)

$(EXEC): $(OBJS)
        $(CC) $(LDFLAGS) -o $@ $(OBJS)

romfs:
        $(ROMFSINST) /bin/$(EXEC)

clean:
        -rm -f $(EXEC) *.elf *.gdb *.o
```

(c) Add hello.c to RT288x_SDK/source/user/hello

```
main()
{
        printf("hello world\n");
}
```

(d) Edit RT288x_SDK/source/config/config.in

```
mainmenu_option next_comment
comment 'XXX Add-on Applications'
bool 'hello_world'                      CONFIG_USER_HELLO_WORLD
endmenu
```

(e) Edit RT288x_SDK/source/user/Makefile

```
dir_$(CONFIG_USER_HELLO_WORLD)        += hello
```

(f) Turn on hello application

```
#make menuconfig

[*] hello_world (NEW)
```

(g) Build new image

#make dep

#make

(h) check file is correct

#cd RT288x_SDK/source/romfs/bin

#file hello

#hello: ELF 32-bit LSB executable, MIPS, MIPS-II version 1 (SYSV), dynamically linked (uses shared libs), stripped

(i) Testing

BusyBox v1.4.2 (2007-05-04 11:15:35 CST) Built-in shell (ash)

Enter 'help' for a list of built-in commands.

/ #

/ # hello

hello world

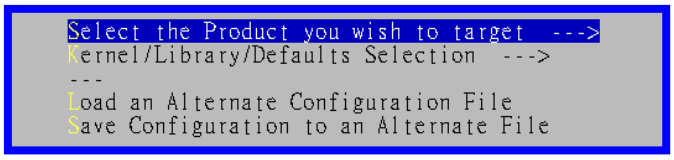
/ #

9 LINUX KERNEL

9.1 Linux configuration

 # cd RT288x_SDK/source

 # make menuconfig



1. Use 'Select the Product you wish to target' to set the target platform.

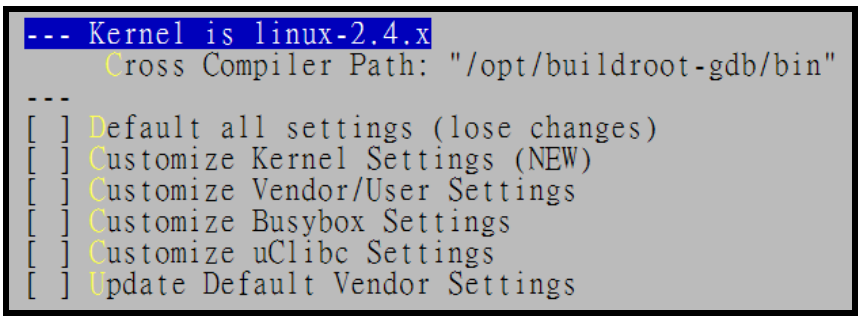
2. Use the 'Flash/SDRAM Size'

- 2M/16M: 2M Flash and 16M DRAM for pure AP solution (pass Vista basic logo and Wi-Fi certification b/g/n logo)
- 4M/16M: 4M Flash and 16M DRAM for complete AP solution, including AP, STA mode)
- 8M/32M: 8M Flash and 32M DRAM for complete AP/NAS solution, including USB applications)

Note:

1. Choose the target platform type (RT2880 or RT3052 or RT3883.)
2. Modify the User/Kernel Configuration or Load/Save User/Kernel Default setting.
3. Load the target platform setting from a file.
4. Save the target platform setting to a file.

Use 'Kernel/Library/Defaults Selection' to open the configuration menu. Use 'Default all settings'.



3. Go out of the configuration menu and save the new kernel configuration.

The script gets all user/kernel default settings back. The subsequent message is shown after getting the default settings back.

> **** End of Linux kernel configuration.*
>
> **** Check the top-level Makefile for additional configuration.*
>
> **** Next, you must run 'make dep'.*

Note: The default configuration file is stored in a different file, referring to the 'Flash/DRAM size' settings. Go to RT288x_SDK/source/vendors/Ralink/{RT2880/RT3052/RT3883}/config to see all the default setting files.

a. Busybox default configuration files

- ✓ 2M_16M_config.busybox-2.4.x/2M_16M_config.busybox-2.6.21.x
- ✓ 4M_16M_config.busybox-2.4.x/4M_16M_config.busybox-2.6.21.x
- ✓ 8M_16M_config.busybox-2.4.x/8M_16M_config.busybox-2.6.21.x

b. User application default configure file

- ✓ 2M_16M_config.vendor-2.4.x/2M_16M_config.vendor-2.6.21.x
- ✓ 4M_16M_config.vendor-2.4.x/4M_16M_config.vendor-2.6.21.x
- ✓ 8M_16M_config.vendor-2.4.x/8M_16M_config.vendor-2.6.21.x

c. uClibc default configure file

- ✓ 4M_16M_config.uclibc-2.4.x/4M_16M_config.uclibc-2.6.21.x
- ✓ 2M_16M_config.uclibc-2.4.x/2M_16M_config.uclibc-2.6.21.x
- ✓ 8M_16M_config.uclibc-2.4.x8M_16M_config.uclibc-2.6.21.x

d. Linux kernel 2.4/2.6 default configure file

- ✓ 2M_16M_config.linux-2.4.x/2M_16M_config.linux-2.6.21.x
- ✓ 4M_16M_config.linux-2.4.x/4M_16M_config.linux-2.6.21.x

- ✓ 8M_16M_config.linux-2.4.x/8M_16M_config.linux-2.6.21.x
- ✓

9.2 Change Flash/DRAM Size

Change the DRAM size setting using “make menuconfig” if you increase or decrease the size of DRAM.

#make menuconfig

Kernel/Library/Defaults Selection ---->

[*] Customize Kernel Settings (NEW)

Machine selection ---->

- Linux 2.4

(RT2880-ASIC) RT2880 Chip Type

(32M) DRAM Size

(4M) Flash Size

- Linux 2.6

System type (Ralink RT3052 board) --->

Soc Hardware Type (RT3052-ASIC) --->

DRAM Size (32M) --->

Root File System Type (RootFS_in_RAM) --->

9.3 Change Switch Controller in RT2880 Platform

The RT288x_SDK supports the IC+ 175C/D switch controller on the RT2880 platform at this time. You can use ‘make menuconfig’ to adjust the switch controller settings.

#make menuconfig

Kernel/Library/Defaults Selection --->

[*] Customize Kernel Settings

Network device support --->

Ralink Driver --->

```
(IC+)   GMAC is connected to
[*]    Partition LAN/WAN on IC+
(W/LLLL)   LAN/WAN Partition
```

W/LLLL in the LAN/WAN Partition item means P0 is a WAN port, and LLLL/W means P4 is WAN Port. The switch is configured by the script, not the Ethernet driver. Please see config-vlan.sh in RT288x_SDK/source/user/rt2880_app/ scripts.

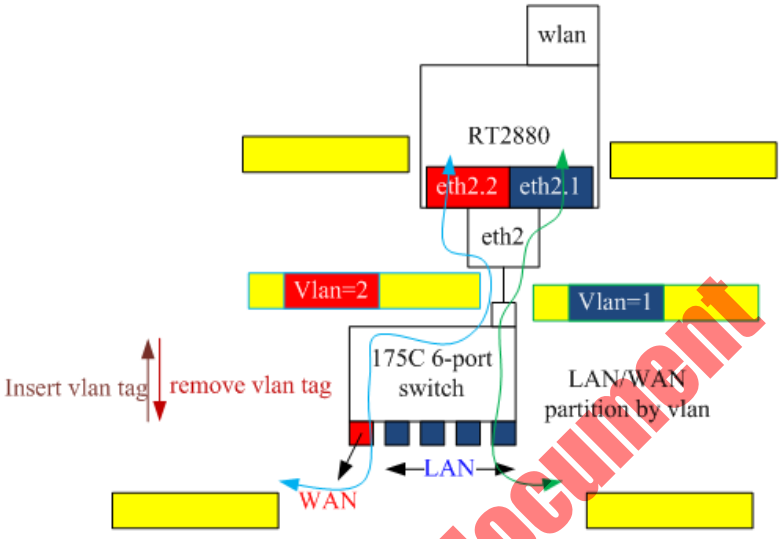


Figure 6 IC+ 10/100 Switch Operation Diagram

9.4 Update User/Kernel default settings

Modify the default setting if necessary. Select the 'Kernel/Library/Defaults Selection' item to enter the kernel/application configuration menu. After entering the menu, select the 'Update Default Vendor Settings' item to update the User/Kernel default settings. (Note: the new default setting will be saved in RT288x_SDK/source/vendors/Ralink/{RT2880/RT3052/RT3883}/config)

```
---  Kernel is linux-2.4.x
     Cross Compiler Path: "/opt/buildroot-gdb/bin"
---
[ ] Default all settings (lose changes)
[ ] Customize Kernel Settings (NEW)
[ ] Customize Vendor/User Settings
[ ] Customize Busybox Settings
[ ] Customize uClibc Settings
[*] Update Default Vendor Settings
```

Select “Exit” to leave the configuration menu. Select “Yes” to save the new kernel configuration.

The script updates the User/Kernel default settings.

9.5 Compile Linux image

#make dep

#make

The following files in RT288x_SDK/images, and ${user}_uImage will be copied to /tftpboot by default.

a. ramdisk.gz - root file system
b. ${user}_uImage - Linux image (Linux kernel+rootfs)
c. zImage.{gz/lzma} - compressed Linux kernel

Note: What kinds of "make" can be used?

a. make Linux image if you modify kernel source files
b. make modules romfs Linux image if you modify the kernel module source files
c. make user_only romfs Linux image if you modify application source files
d. You can execute "make" to generate a new image (make = make lib_only user_only modules romfs Linux image)

9.6 Port new Linux kernel module

Example: Port the hello networking module to the RT2880 platform

1. Add the source code to the rt2880 directory

 # mkdir RT288x_SDK/source/linux-2.4.x/drivers/net/hello

#vi RT288x_SDK/source/linux-2.4.x/drivers/net/hello/Makefile

O_TARGET := hello.o

obj-y := main.o

obj-m := $(O_TARGET)

include $(TOPDIR)/Rules.make

#vi RT288x_SDK/source/linux-2.4.x/drivers/net/hello/main.c

```
#include <linux/init.h>
#include <linux/module.h>
static int hello_init(void)
{
        printk("hello world\n");
        return 0;
}
static void hello_exit(void)
{
        printk("goodbye\n");
}
module_init(hello_init);
module_exit(hello_exit);
MODULE_LICENSE("GPL");
~
```

2. Modify RT288x_SDK/source/linux-2.4.x/drivers/net/Makefile

```
subdir-$(CONFIG_RT2880_HELLO) += hello
```

3. Modify Config.in

```
tristate '    Ralink hello module' CONFIG_RT2880_HELLO
```

4. Turn on the hello module

```
#make menuconfig
<M>      Ralink hello module
```

5. Compile the source code

```
#make dep
#make
```

6. Test

/ # insmod hello

hello world

/ #

9.7 Execute commands at boot up time

Edit RT288x_SDK/source/vendors/Ralink/RT2880/rcS

#!/bin/sh

mount –a

goahead& <-- add new command here

9.8 Add new files in RootFs

If you execute the "make clean" script, it will delete RT288x_SDK/source/romfs directory.

You cannot copy the file to RT288x_SDK/source/romfs manually because it will disappear after executing "make clean".

Example: add xxx.bin to rootfs

a. copy xxx.bin to RT288x_SDK/source/vendors/Ralink/{RT2880/RT3052/RT3883}

b. edit RT288x_SDK/source/vendors/Ralink/{RT2880/RT3052/RT3883}/Makefile

romfs:

$(ROMFSINST) /etc_ro/xxx.bin

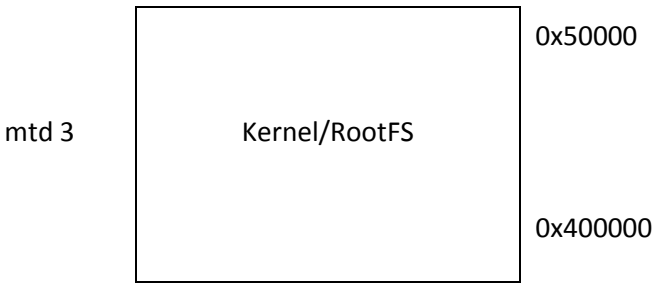
The script will copy xxx.bin to RT288x_SDK/source/romfs/etc_ro after executing "make romfs"

9.9 Image DownSize

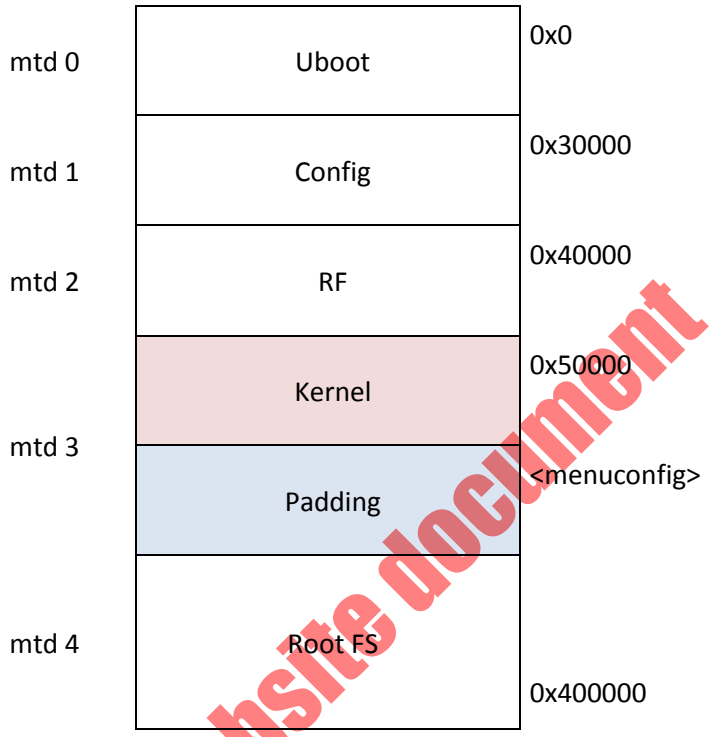
The MTD partitions are subsequently shown.

RootFS in RAM Mode

mtd 0	uboot	0x0
mtd 1	config	0x30000
mtd 2	RF	0x40000



RootFS in Flash Mode



In RootFS in Flash mode, the image builder will add a padding bit to the end of kernel image if the kernel image size is smaller than the size of mtd3. The size of mtd3 must be adjusted to save flash memory.

Step1: Check the original kernel image size (ex: 446603)

#make image

.....

#==========<SquashFS Info>=================

 # Original Kernel Image Size

576110 /home/steven/RT288x_SDK/source/images/zImage.lzma

 # Padded Kernel Image Size

786368 /home/steven/RT288x_SDK/source/images/zImage.lzma

 # Original RootFs Size

4329746 /home/steven/RT288x_SDK/source/romfs

 # Compressed RootFs Size

1069056 /home/steven/RT288x_SDK/source/images/ramdisk

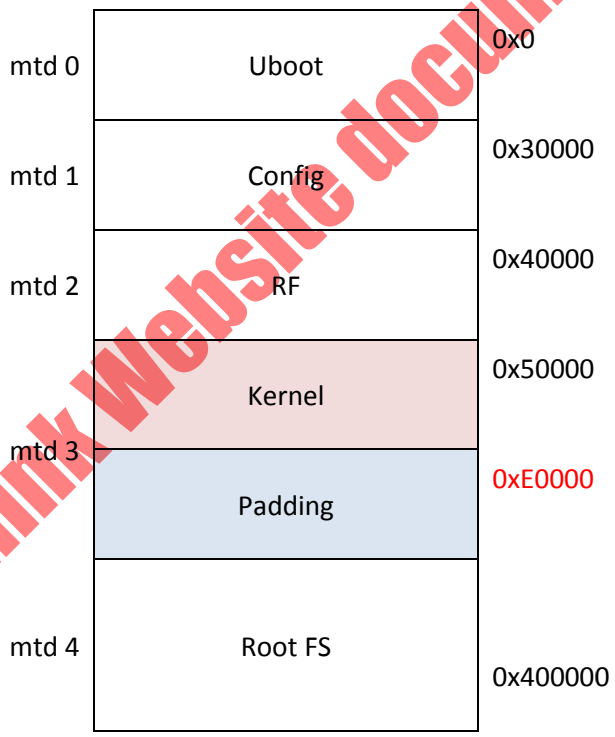
 # Padded Kernel Image + Compressed Rootfs Size

1855424 /home/steven/RT288x_SDK/source/images/zImage.lzma

#== .....

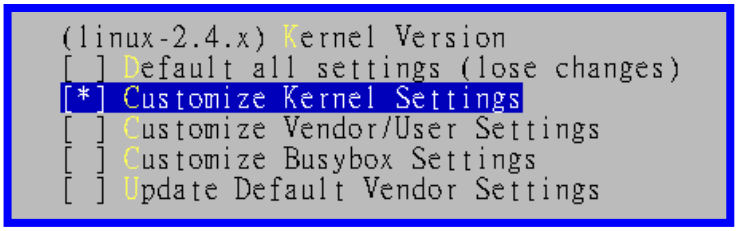
Step2: Change mtdblock size

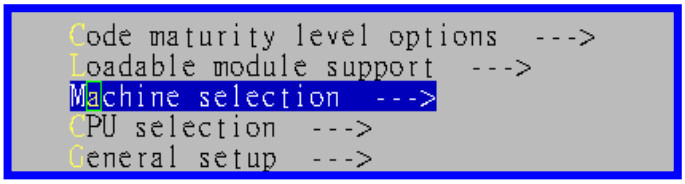
576110=0x8CA6E -> 0x90000 (multiple of 0x10000 because the flash sector size=64KB)

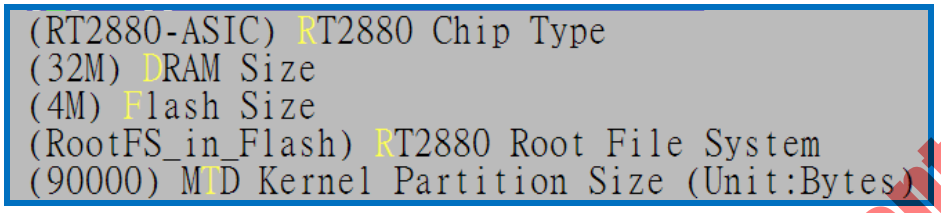


host:$ make menuconfig

Hit 'Kernel/Library/Defaults Selection' to enter configuration menu.







10 FLASH LAYOUT AND FIRMWARE UPGRADE

10.1 Flash Layout

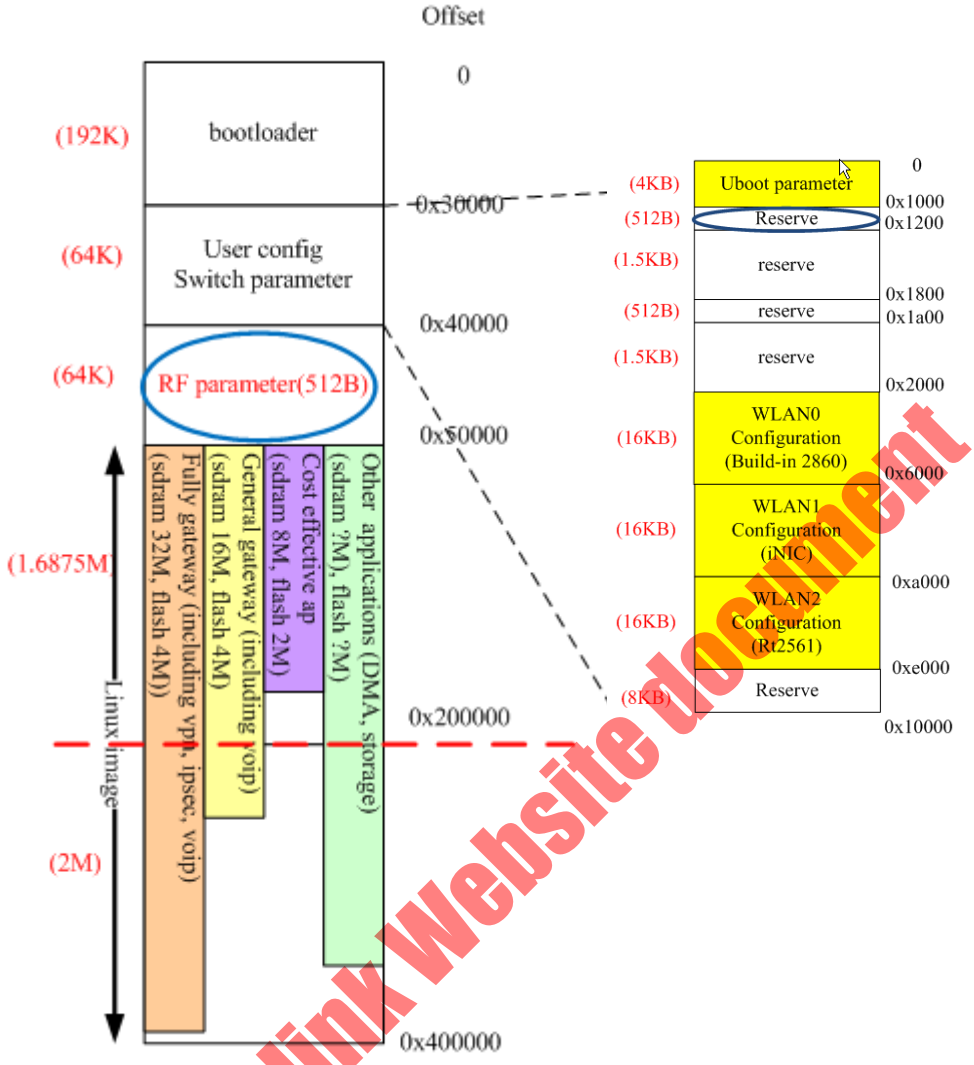


Figure 7 Ralink SDK Flash Layout (4MB)

In the 'user configure switch parameter' partition, the WLAN0 configuration is for built-in RT2860 parameters, the WLAN1 configuration is for iNIC parameters, and the WLAN2 configuration is for RT2561 parameters. Use the free space to save your own parameters if you don't need to support iNIC or RT2561 on your product.

10.2 Firmware Upgrade

10.2.1 By Uboot

```
============================================
Ralink UBoot Version: 2.0
--------------------------------------------
ASIC 2880_MP (MAC to 100PHY Mode)
DRAM COMPONENT: 128Mbits
DRAM BUS: 32BIT
Total memory: 32Mbytes
Date:May  9 2008  Time:11:14:00
============================================
 D-CACHE set to 4 way
 I-CACHE set to 4 way

 ##### The CPU freq = 266 MHZ ####

 SDRAM bus set to 32 bit
 SDRAM size =32 Mbytes

Please choose the operation:
   1: Load system code to SDRAM via TFTP.
   2: Load system code then write to Flash via TFTP.
   3: Boot system code via Flash (default).
   4: Entr boot command line interface.
   5: Load ucos code to SDRAM via TFTP.
```

1. Select option 2 on the UBoot menu to burn the Linux image from 0x50000 to 0x400000.
2. Select option 9 on the Uboot menu to update your uboot from 0x0 to 0x30000.

10.2.2 By WebUI

You can use WebUI to upgrade the Linux image.

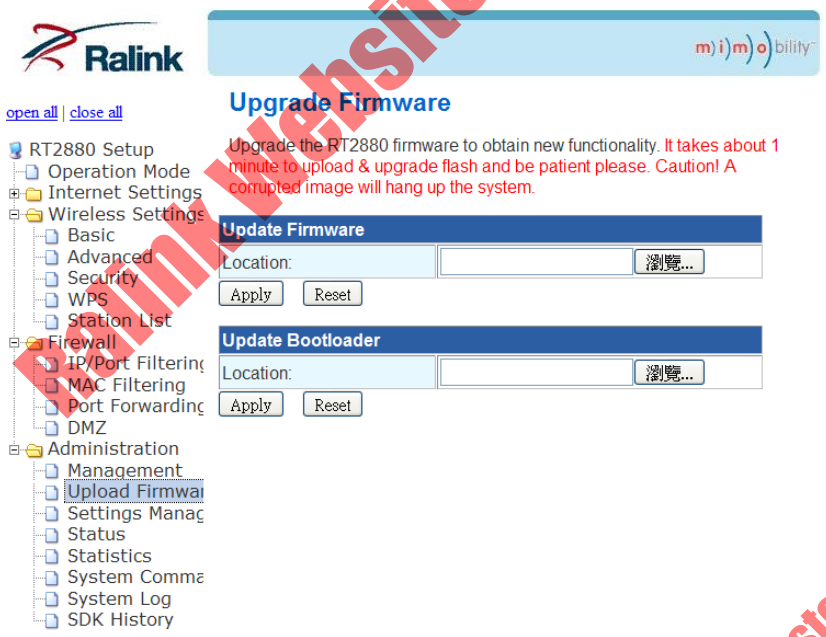
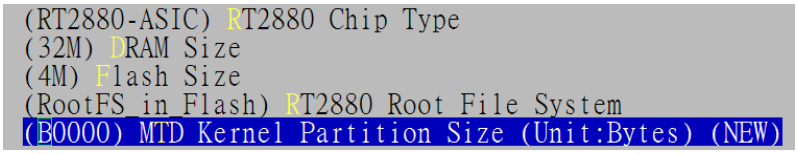


Figure 8 WebUI Firmware Upgrade

CGI uses the mtd_write command to burn a Linux image.

- **File system in RAM** - Burn Linux image to mtdblock3 (Kernel)
- **File system in Flash** - Burn first x bytes to mtdblock3, and others to mtdblock4 (ps. X byes = MTTD kernel partition size in “make menuconfig”



11 FAQ

11.1 RT2880 Default password/UART/networking setting

Table 4 Networking Setting

LAN	IP Address	10.10.10.254
	Subnet	255.255.255.0
WAN	IP Address	DHCP

Table 5 UART Setting

Item	Value
Baud Rate	57600
Data bits	8
Parity	None
Stop Bit	1
Flow Control	None

Table 6 Web Setting

Item	Default Value
User Name:	admin
Password:	admin

11.2 System requirements for the host platform

RT2880 SDK uses Fedora 6 Host to build the image. Change your Linux distribution if you cannot successfully build the image.

Table 7 Requirements of Host Platform

Item	Value
Linux Distribution	Fedora 6
Kernel version	2.6.18-1.2798.fc6
RAM	512MB
HD	40G

11.3 How to add new default parameter in flash

There are four default settings In RT288x_SDK/source/vendors/Ralink/RT2880, based on different platforms.

- RT2860_default_vlan: IC+ (gateway mode)/Vitesse Platform
- RT2860_default_novlan: IC+ (bridge mode)/Marvell 1000 Phy platform
- RT2860_default_oneport: IC+ 100 Phy platform
- RT2561_default: RT2561 PCI NIC (RT2860+RT2561 concurrent)

11.3.1 Example 1

Add a new default parameter - WHOAMI for IC+ platform

1. Edit RT288x_SDK/source/vendors/Ralink/RT2880/ RT2860_default_vlan, and add the following line.

WHOAMI=steven

2. Push “wps/load_default” button or execute the following commands

#ralink_init clear 2860

#reboot

3. Use nvram_get to retrieve WHOAMI parameter in script file (RT288x_SDK/source/user/rt2880_app/scripts), or nvram_bufset, nvram_bufget, nvram_commit in your CGI(RT288x_SDK/source/user/goahead/src) to use your feature.

11.3.2 Example 2

Save the RADIO ON/OFF button in WebUI to flash:

1. Add a line to RT288x_SDK/source/vendors/Ralink/RT2880/ RT2860_default_vlan for the default value:

RadioOn=1

2. Modify RT288x_SDK/source/user/goahead/src/wireless.c, function wirelessBasic() to save the radio on/off value to flash:

```
radio = websGetVar(wp, T("radiohiddenButton"), T("2"));
if (!strncmp(radio, "0", 2)) {
    nvram_bufset(RT2860_NVRAM, "RadioOn", radio);
```

```
        doSystem("ifconfig ra0 down");
        websRedirect(wp, "wireless/basic.asp");
        return;
    }
    else if (!strncmp(radio, "1", 2)) {
        nvram_bufset(RT2860_NVRAM, "RadioOn", radio);
        doSystem("ifconfig ra0 up");
        websRedirect(wp, "wireless/basic.asp");
        return;
    }
```

3. Modify the RT288x_SDK/source/user/rt2880_app/scripts/internet.sh script not to bring ra0 up if RadioOn value stored in flash is not 1. Change “ifconfig ra0 0.0.0.0” to…

```
radio=`nvram_get 2860 RadioOn`
if [ "$radio" = "1" ]
        ifconfig ra0 0.0.0.0 up
else
        ifconfig ra0 0.0.0.0 down
fi
```

11.4 Enable Ethernet Converter Feature

The Wi-Fi Interface on the RT2880 platform should be configured for STA mode. All PCs under the RT2880 GMAC port connect to the AP via the RT2880 platform.

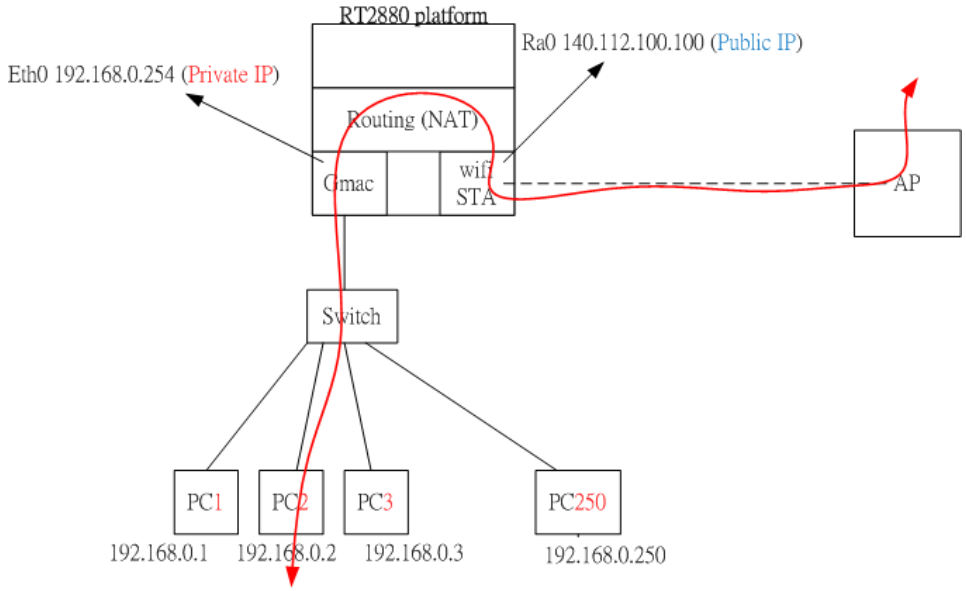


Figure 9 Ethernet Converter Operation Diagram

If the RT2880 platform can be operated as an AP or Ethernet converter by WebUI Configuration, make sure STA support and AP support as a Linux module is on in the rt2860v2 driver.

```
<M>   Ralink RT2860 802.11n AP support - 2860v2, (RBUS and PCI)
(RBUS) Bus Type
[ ]    IED SUPPORT
[*]    WSC (WiFi Simple Config)
[ ]    Nintendo
[ ]    LLTD (Link Layer Topology Discovery Protocol)
[*]    ATE
[*]    WDS
[*]    MBSSID
[ ]    AP-CLient Support
[ ]    IGMP snooping support
[ ]    NETIF Block
<M>   Ralink RT2860 802.11n STA support - 2860v2, (RBUS and PCI)
(RBUS) Bus Type
[ ]    IED SUPPORT
[ ]    WPA Supplicant
[ ]    WSC (WiFi Simple Config)
```

Turn on the rt2860v2 STA support if the RT2880 platform is an Ethernet converter only.

Select the operation mode on the “Operation Mode Configuration” web page.

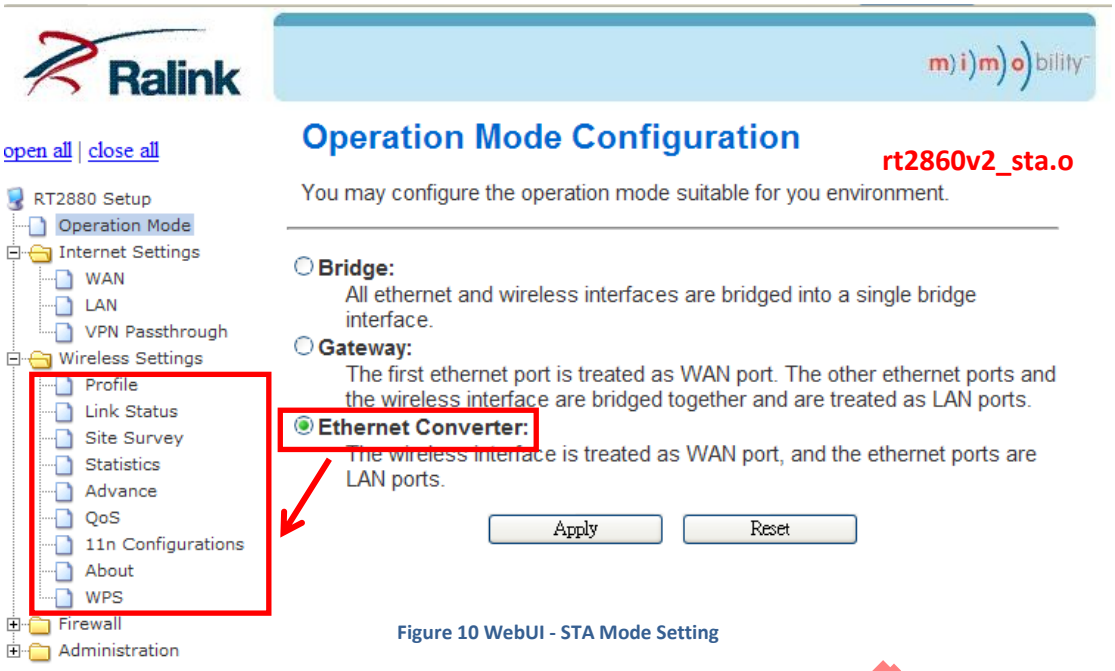


Figure 10 WebUI - STA Mode Setting

11.5 Change RF chip from RT2820 to RT2850 on the RT2880 platform

The QA program can burn an RT2850 EEPROM binary file. Click the "Load File" button and choose your own EEPROM binary file. The QA program will immediately burn the binary file to flash.

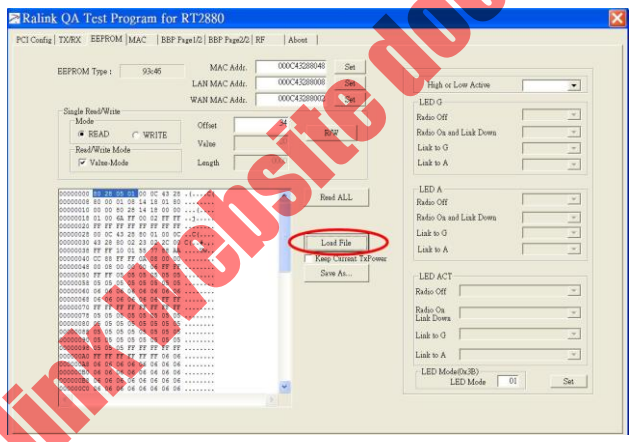


Figure 11 QA – Burn your own EEPROM binary file

11.6 How to change the Ethernet MAC address

The Ralink Ethernet driver uses GMAC0_ADDR to save its LAN/WAN mac address. If GMAC0_ADDR is empty, it will generate a random mac address instead.

#define GMAC0_ADDR (RT_EEPROM_BASE + 0x28)

#define GMAC1_ADDR (RT_EEPROM_BASE + 0x2E)

Note: If you need the LAN/WAN Ports to have different MAC addresses, adjust the Ethernet driver to get GMAC0_ADDR for LAN, and GMAC1_ADDR for WAN.

Use the QA program to modify your flash content.

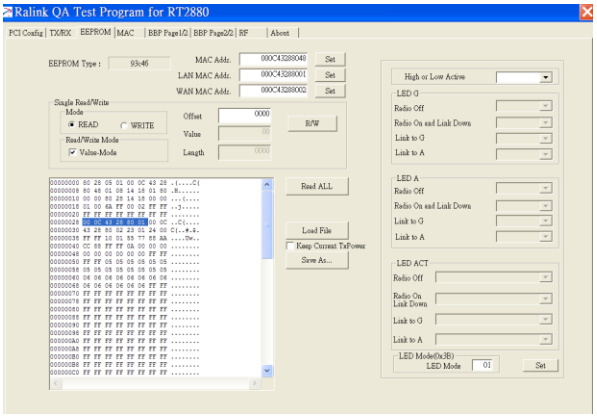


Figure 12 QA – Modify GMAC Mac address

11.7 How to configure GPIO ports

$SDK/source/linux-2.4.x/drivers/char/ralink_gpio.c

$SDK/source/linux-2.4.x/drivers/char/ralink_gpio.h

- RALINK_GPIO_SET_DIR - Configure the direction of the GPIO pins using bitmaps. Bit 1 is for output, and bit 0 is for input. For example, value 0x5 is for configuring GPIO pin 0 and 2 as output pins, and the other pins as input pins.
- RALINK_GPIO_SET_DIR_IN - Configure one or several GPIO pins as input pins using bitmaps. For example, value 0x5 is for configuring GPIO pin 0 and 2 as input pins, and other pins are ignored.
- RALINK_GPIO_SET_DIR_OUT - Configure one or several GPIO pins as output pins using bitmaps. For example, value 0x5 is for configuring GPIO pin 0 and 2 as output pins, and other pins are ignored.
- RALINK_GPIO_READ - Read the value from the GPIO data.
- RALINK_GPIO_WRITE - Write a value to the GPIO data.
- RALINK_GPIO_SET - Set a value with corresponding bits on to the GPIO data. For example, value 0x5 means GPIO data bit 0 and 2 will be set to 1, and the other bits will be ignored.
- RALINK_GPIO_CLEAR - Clear a value with corresponding bits off the GPIO data. For example, value 0x5 means GPIO data bit 0 and 2 will clear to 0, and other bits will be ignored.

- RALINK_GPIO_READ_BIT - Read the corresponding bit from the GPIO data. For example, bit 2 means read the third bit from GPIO data.
- RALINK_GPIO_WRITE_BIT - Write a corresponding bit to the GPIO data. For example, bit 2 and value 1 mean to write value 1 to the third bit of GPIO data.
- RALINK_GPIO_READ_BYTE - Read the corresponding byte from the GPIO data. For example, byte 2 means to read the third byte from GPIO data.
- RALINK_GPIO_WRITE_BYTE - Write a corresponding byte to the GPIO data. For example, byte 2 and value 0x33 mean to write value 0x33 to the third byte of the GPIO data.
- RALINK_GPIO_READ_INT - Same as RALINK_GPIO_READ.
- RALINK_GPIO_WRITE_INT - Same as RALINK_GPIO_WRITE.
- RALINK_GPIO_SET_INT - Same as RALINK_GPIO_SET.
- RALINK_GPIO_CLEAR_INT - Same as RALINK_GPIO_CLEAR.
- RALINK_GPIO_ENABLE_INTP - Enable GPIO input interrupt.
- RALINK_GPIO_DISABLE_INT - Disable GPIO input interrupt.

RALINK_GPIO_REG_IRQ - Register to receive an interruption from a GPIO pin. When the GPIO pin is interrupted, a signal SIGUSR1 or SIGUSR2 will be sent to the registered user process id. SIGUSR1 is sent when the GPIO pin has been clicked once, and SIGUSR2 is send when the GPIO pin has been pressed for several seconds.

11.8 Use GPIO to turn on LED

The following tables show the current GPIO definition for RT2880/RT3052/RT3883.

Table 8 GPIO Usage of RT2880

RT2880 Pin #	Pin name	GPIO define	Board version		Description
			2.4G	Dual	
			V30RW-FE	V11RW-GB	
K20	GPIO0	WPS/ Reset to default	●	●	Low Active signal input for Wi-Fi protection setup function and restore the setting to default value when push bottom for 3 second
P17	GPIO8/**DTR_N**		●	●	Reserved
R17	GPIO10/**DCD_N**	Giga PHY Reset		●	Low Active output for GigaPHY reset
T18	GPIO11/**DSR_N**		●	●	Reserved
P20	GPIO12/**CTS_N**	System Status/ Power LED	●	●	Low Active output for system ready LED display
N19	GPIO13/**RIN**	Security LED	●	●	Low Active output for security LED indicates when wireless security is enabled, display security status on panel
R20	GPIO14/**RXD**		●	●	Reserved for system reboot, Low Active output

Table 9 GPIO Usage of RT3052

RT3052 Pin #	Pin name	GPIO define	Board version	Description
			AP-RT3052-V20RW-2X2	
U10	GPIO0	WPS PBC	●	Low Active signal input for WPS function when push bottom over 3 second
T10	GPIO1/**I2C_SD**			
R10	GPIO2/**I2C_SCLK**			
U9	GPIO3/**SPI_EN**	RX_SW	●	GPIO3/GPIO5 ANT diversity 10: ANT2 01: ANT0
T9	GPIO4/**SPI_CLK**			
U8	GPIO5/**SPI_DOUT**	RX_SWN	●	
R9	GPIO6/**SPI_DIN**	iNIC mode select	●	Resistor strapping input 1: load code mode 0: dump switch mode
G2	GPIO7/**RTS_N**			
F2	GPIO8/-**TXD**-			
G1	GPIO9/-**CTS_N**-	System/ Power LED	●	Low Active output System status/Power display
J3	GPIO10/-**RXD**-	SW RST/ Factory	●	1. SW RST: Low Active signal input 2.Factory default: push bottom over 3 second
J4	GPIO11/-**DTR_N**-			
H3	GPIO12/-**DCD_N**-			
F1	GPIO13/-**DSR_N**-	Security LED	●	Low Active output security mode display
K4	GPIO14/-**RIN**-	WPS LED	●	Low Active output Indicate WPS PBC status

Table 10 GPIO Usage of RT3883/RT3662

RT3883/RT3662 Ball #	Ball name	Function	Description
K9	GPIO0	WPS LED	Use for WPS LED on Reference board.
K8	GPIO1	GPHYRST_N	Use for Giga Switch reset on Reference board.
L9	GPIO2	Band selection	RF 2.4GHz/5GHz Band selection.
L8	GPIO3	WPS_PB	WPS Push Button.
G14	GPIO4	SWRST_N_PB	Factory Default Push Button.
H14	GPIO5	Boot Strapping	Boot Strapping
H12	GPIO6	Boot Strapping	Boot Strapping
H13	GPIO7	Boot Strapping	Boot Strapping
G12	GPIO8	NC	Reserved for internal use.

The Ralink SDK GPIO driver gives an interface to set the frequency of the LEDs connected to the GPIOs.

Define RALINK_GPIO_LED_LOW_ACT to 1 at $SDK/linux-2.4.x/drivers/char/ralink_gpio.h if the LEDs are inactive. Otherwise, define it as 0.

```
#make menuconfig
Kernel/Library/Defaults Selection  --->
[*] Customize Kernel Settings (NEW)
Character devices  --->
[*] Ralink RT2880 GPIO Support
[*]     Ralink GPIO LED Support
```

The LED can be set to blink in different ways if RALINK_GPIO_LED has been built enabled. The argument for RALINK_GPIO_LED_SET is ralink_gpio_led_info structure:

```
typedef struct {
        int gpio
        unsigned int on
        unsigned int off
```

unsigned int blinks

unsigned int rests;

unsigned int times;

} ralink_gpio_led_info;

Write the application to set the LED frequency through the ioctl interface of the GPIO device. Use the example application, gpio.

#make menuconfig

Kernel/Library/Defaults Selection ---->

[*] Customize Vendor/User Settings

Ralink RT288x Application ---->

[ ] RT2880 GPIO Test

Usage:

gpio *l* <gpio> <on> <off> <blinks> <rests> <times>

- gpio: GPIO number of the board
- on: number of ticks that the LED will be bright
- off: number of ticks that the LED will be dark
- blinks: number of on-offs that the LED will blink
- rests: number of on-offs that the LED will rest
- times: number of blinks before the LED stops

Note: 1 tick is equal to 100ms. The maximum number is 4000 at this time.

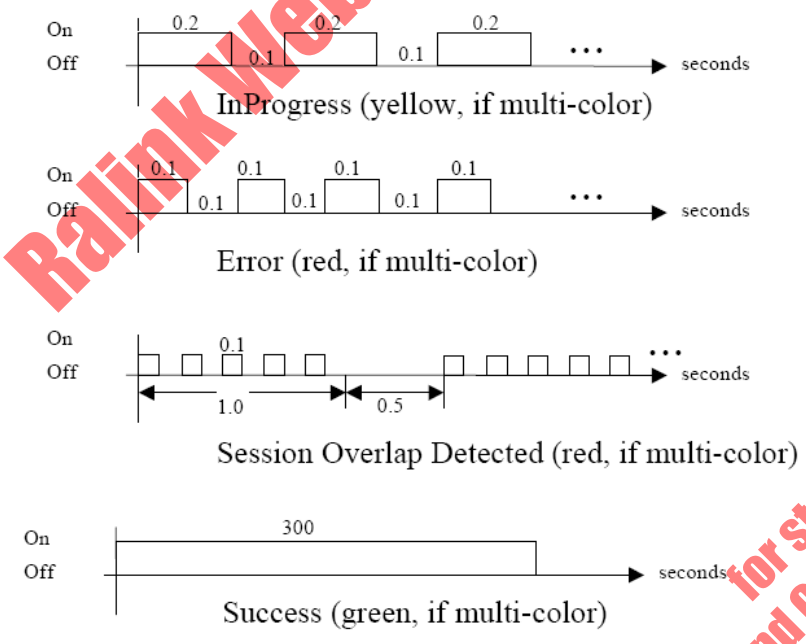


Figure 13 LED Definition of WPS Specification

Using the WPS PBC status LED as an example, the configurations would be:

- InProgress: gpio l <gpio> 2 1 4000 0 4000 (i.e. 2 ticks bright, 1 tick dark, blinking forever.)
- Error: gpio l <gpio> 1 1 4000 0 4000 (i.e. 1 tick bright, 1 tick dark, blinking forever.)
- Session Overlap Detected: gpio l <gpio> 1 1 10 5 4000 (i.e. 1 tick bright, 1 tick dark, blinking for 10 on-offs, resting for 5 on-offs, and never stops.)
- Success: gpio l <gpio> 3000 1 1 1 1 (i.e. 3000 ticks bright, 1 tick dark, blinking for one on-offs and one time.)
- To turn the LED on and keep it on: gpio l <gpio> 4000 0 1 0 4000
- To turn the LED off and keep it off: gpio l <gpio> 0 4000 0 1 4000

11.9 Use LED firmware to turn on LED

1. enable LED firmware

#make menuconfig

Kernel/Library/Defaults Selection ---->

[*] Customize Kernel Settings

Network device support ---->

Ralink Driver ---->

```
<M>   Ralink RT2860 802.11n AP support - 2860v2, (RBUS and PCI)
(RBUS) Bus Type
[ ]    Dual Band
[*]    LED Support
[*]    WSC (WiFi Simple Config)
[*]    LLTD (Link Layer Topology Discovery Protocol)
[*]    ATE
[*]    WDS
[*]    MBSSID
[ ]    AP-CLient Support
[ ]    IGMP snooping Support
[ ]    NETIF Block
[ ]    DFS Support
[ ]    Carrier Detect Support
```

2. Fill out flash content to control the LED behavior because the LED firmware will read the configuration from flash.

Table 11 RT2880 LED Parameters in Flash

Address	Bit	LED Mode	Mode Description	Comment
3Bh	[6:0]	0	HW control	The default mode. Driver sets MAC register and MAC controls LED.
		1	FW default mode	The firmware controls how LED blinks.
		2	8sec scan	Same as LED mode 1 except that fast blink for 8sec when doing scanning.
		3-63	-	Reserved for future
		64	Signal strength setting	Besides mode 1, additionally set LED signal strength. LedParam1[0] = GPIO polarity (0 is negative) LedParam0 = Signal strength (Valid value are 0, 1,3,7,15,31. 0 is the weakest.)
	7	GPIO Polarity		

Address	States	Bit	RT2860 Pin-127_LED behavior
3Eh	Radio off	[1:0]	00: Reserved 01: Solid on 10: Blink when transmitting data and management packet 11: Blink when transmitting data, management packet and beacon
		2	0: Solid on when no traffic 1: Slow blink when no traffic
		3	Reserved
	Radio on but link down	[5:4]	00: Reserved 01: Solid on 10: Blink when transmitting data and management packet 11: Blink when transmitting data, management packet and beacon
		6	0: Solid on when no traffic 1: Slow blink when no traffic
		7	Reserved
3Fh	Radio on and link to G	[9:8]	00: Reserved 01: Solid on 10: Blink when transmitting data and management packet 11: Blink when transmitting data, management packet and beacon
		10	0: Solid on when no traffic 1: Slow blink when no traffic
		11	Reserved
	Radio on and link to A	[13:12]	00: Reserved 01: Solid on 10: Blink when transmitting data and management packet 11: Blink when transmitting data, management packet and beacon
		14	0: Solid on when no traffic 1: Slow blink when no traffic
		15	Reserved

Address	States	Bit	LED behavior	
40h	Radio off	[3:0]	bit 0: LED G	1: Positive polarity 0: Negative polarity
			bit 1: LED A	
			bit 2: LED Act	
			bit 3: 0: Reserved	1: LED ACT polarity inversion when link to A
	Radio on but link down	[7:4]	bit 0: LED G	1: Positive polarity 0: Negative polarity
			bit 1: LED A	
			bit 2: LED Act	
			bit 3: 0: Reserved	1: LED ACT polarity inversion when link to A
41h	Radio on and link to G	[11:8]	bit 0: LED G	1: Positive polarity 0: Negative polarity
			bit 1: LED A	
			bit 2: LED Act	
			bit 3: 0: Reserved	1: LED ACT polarity inversion when link to A
	Radio on and link to A	[15:12]	bit 0: LED G	1: Positive polarity 0: Negative polarity
			bit 1: LED A	
			bit 2: LED Act	
			bit 3: 0: Reserved	1: LED ACT polarity inversion when link to A

The current Ralink default flash hex values are subsequently shown.

RT2880 Flash Base Address=0x40000

- 4003B: 1 controlled by firmware
- 4003C: 55 LED A/G don't care
- 4003D: 77 LED A/G don't care
- 4003E: A8 LED ACT radio off = solid on/off
- 4003F: AA LED ACT blink when transmitting data & management packet
- 40040: 8C LED Act positive polarity when radio off -> solid off
- 40041: 88 LED Act negative polarity when link to A/G -> blink

11.10 How to start the telnet server

Check RT288x_SDK/source/user/busybox/.config

11.10.1 busybox setting

CONFIG_FEATURE_DEVPTS=y ➔ General Configuration

CONFIG_FEATURE_SUID=y ➔ General Configuration

CONFIG_LOGIN=y ➔ Login/Password Management Utilities

CONFIG_TELNETD=y ➔ Networking utilities

CONFIG_FEATURE_TELNETD_STANDALONE=y

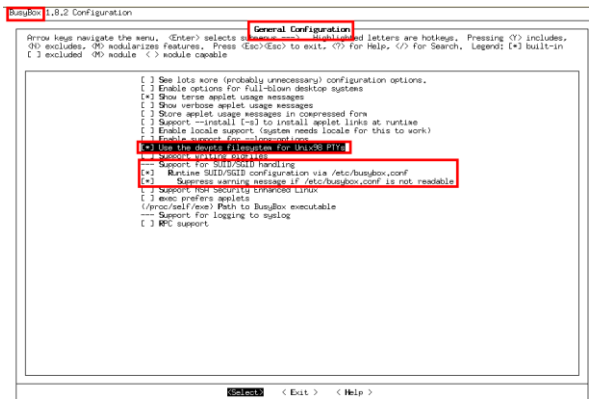
Check RT288x_SDK/source/linux-2.4.x/.config

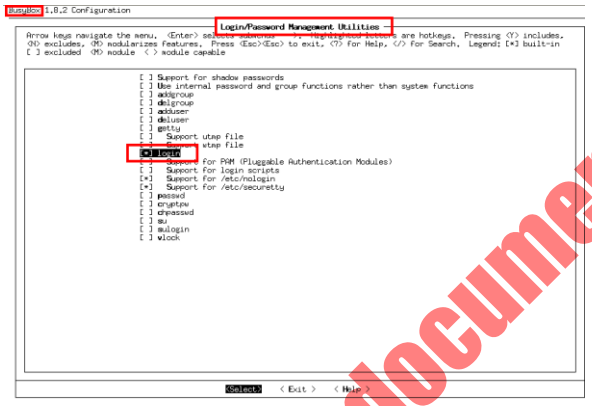
11.10.2 Linux setting

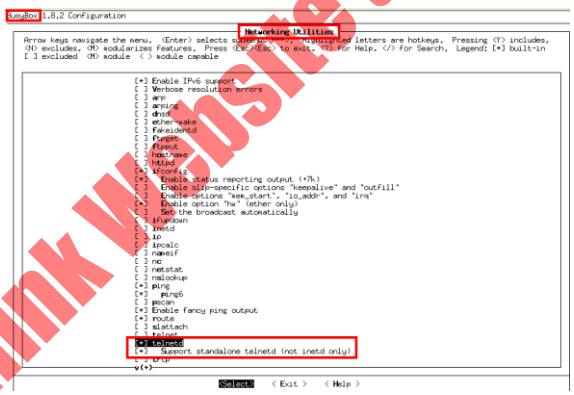
CONFIG_UNIX98_PTYS=y ➔ Character devices

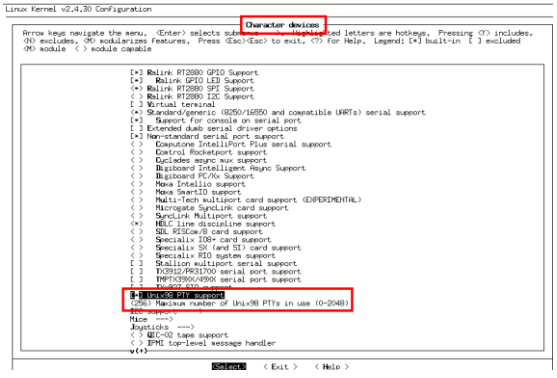
CONFIG_UNIX98_PTY_COUNT=256

CONFIG_DEVPTS_FS=y ➔ File systems









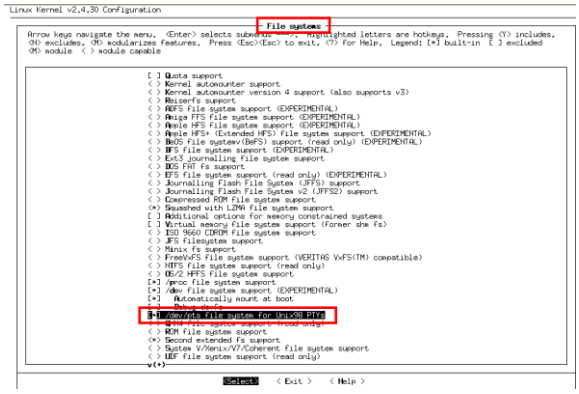


Figure 14 Configuration Procedure of Telnet Server

11.11 11n bit rate derivation

1. The 11n bit rate is calculated by the MAC driver. The MAC driver refers to the three subsequent factors.
   a. MCS
   b. BW
   c. GI

Note: the bit rate is primarily given by the PHY layer.

2. Bandwidth: Data subcarriers on different bandwidths, 20MHz and 40MHz.

   a. N_{SD}: Number of data subcarriers.

      N_{SD}[40Mhz] = 108

      N_{SD}[20Mhz] = 52

      N_{SD}[40Mhz]/N_{SD}[20MHz] = 108/52

      = 2.076923076923076923076923076923

   b. Example:

      MCS=15, GI=800ns, BW=20MHz, DataRate = 130Mbps

      MCS=15, GI=800ns, BW=40MHz, DataRate = 130 * [$N_{sd(40Mhz)}$ / $N_{sd(20Mhz)}$]

      = 130 * [108 / 52]

      = 270Mbps

   c. Please refer to “IEEE P802.11n/D2.04, June 2007” on page 314 for subsequent table.

Table 207—MCS parameters for optional 20 MHz, N_{SS} = 2, N_{ES} = 1, EQM (#665)

MCS Index	Modulation	R	$N_{BPSCS(iSS)}$	N_{SD}	N_{SP}	N_{CBPS}	N_{DBPS}	Data rate (Mb/s)	
								800 ns GI	400 ns GI See NOTE
8	BPSK	1/2	1	52	4	104	52	13.0	14.4
9	QPSK	1/2	2	52	4	208	104	26.0	28.9
10	QPSK	3/4	2	52	4	208	156	39.0	43.3
11	16-QAM	1/2	4	52	4	416	208	52.0	57.8
12	16-QAM	3/4	4	52	4	416	312	78.0	86.7
13	64-QAM	2/3	6	52	4	624	416	104.0	115.6
14	64-QAM	3/4	6	52	4	624	468	117.0	130.0
15	64-QAM	5/6	6	52	4	624	520	130.0	144.4

NOTE—The 400 ns GI rate values are rounded to 1 decimal place

3. Guard Interval:

   a. Definition:

      T_{sym}: 4μs ;Symbol Interval

      T_{syms}: 3.6μs ;Symbol interval of Short GI.

   b. Ratio of symbol interval on GI, refer to below EWC PHY Sepc.

      Tsym / Tsyms = 4μsec / 3.6μsec = 10/9

   c. Example:

      MCS=15, 40MHz Bandwidth, and 400ns Short Guard Interval.

      270.0 * (10/9) = 300.0 for Short GI.

   d. Reference:

      1) IEEE 802.11n draft 2.04, page 316 and

Table 211—MCS parameters for optional 40 MHz, NSS = 2, NES = 1, EQM (#665)									
MCS Index	Modulation	R	NBPSCS(iSS)	NSD	NSP	NCBPS	NDBPS	Data rate (Mb/s)	
								800 ns GI	400 ns GI
8	BPSK	1/2	1	108	6	216	108	27.0	30.0
9	QPSK	1/2	2	108	6	432	216	54.0	60.0
10	QPSK	3/4	2	108	6	432	324	81.0	90.0
11	16-QAM	1/2	4	108	6	864	432	108.0	120.0
12	16-QAM	3/4	4	108	6	864	648	162.0	180.0
13	64-QAM	2/3	6	108	6	1296	864	216.0	240.0
14	64-QAM	3/4	6	108	6	1296	972	243.0	270.0
15	64-QAM	5/6	6	108	6	1296	1080	270.0	300.0

2) EWC PHY spec. page 13.

Parameter	Value in legacy 20MHz channel	Value in 20MHz HT channel	Value in 40MHz channel	
			HT format	Legacy Duplicate
frequency spacing				
T_{FFT}: IFFT/FFT period	3.2µsec	3.2µsec	3.2µsec	
T_{GI}: Guard Interval length	0.8µsec= $T_{FFT}/4$	0.8µsec	0.8µsec	
T_{GI2}: Double GI	1.6µsec	1.6µsec	1.6µsec	
T_{GIS}: Short Guard Interval length	0.4µsec= $T_{FFT}/8$	0.4µsec	0.4µsec	
T_{L-STF}: Legacy Short training sequence length	8µsec=10× $T_{FFT}/4$	8µsec	8µsec	
T_{L-LTF}: Legacy Long training sequence length	8µsec=2× $T_{FFT}+T_{GI2}$	8µsec	8µsec	
T_{SYM}: Symbol Interval	4µsec= $T_{FFT}+T_{GI}$	4µsec	4µsec	
T_{SYMS}: Short GI Symbol Interval	3.6µsec= $T_{FFT}+T_{GIS}$	3.6µsec	3.6µsec	
T_{L-SIG}	4µsec= T_{SYM}	4µsec	4µsec	

Tsym/Tsyms = 4u/3.6u = 10/9

3) EWC PHY spec. page 13.

transmission for a period of corresponding to the length of the rest of the packet. When L-SIG TXOP Protection is not used (see "L-SIG TXOP Protection" section of the EWC MAC spec), the value to be transmitted is $l = 3\left(\lceil N_{data} \rceil + N_{LTF} + 3\right) - 3$ where N_{data} is the number of **4usec** symbols in the data part of the packet. While using short GI N_{data} is equal to the actual number of symbols in the data part of the packet multiplied by $\frac{9}{10}$. N_{LTF} is the number of HT training symbols. The symbol $\lceil x \rceil$ denotes the lowest integer greater or equal to x.

11.12 How to build a single image for the flash programmer

Example: Make a 4M single image for the rt2880 platform (the Uboot partition is 192K, user configuration partition is 64K, and RF partition is 64K)

 # RT288x_SDK/tools/single_img

#vi Makefile.4M

#

 # Change uboot/kernel size if necessary

#

UBOOT_SIZE = 0x50000

KERNEL_SIZE = 0x3B0000

#--

USER_NAME = $(shell whoami)

#

 # Uboot Image Information

#

UBOOT_DIR = .

UBOOT_IMAGE = uboot.bin

#

 # Linux Kernel Image Information

#

KERNEL_DIR = .

KERNEL_IMAGE = steven_uImage

#

 # Single Image Information

#

PACKED_DIR = .

PACKED_IMAGE = steven_uImage.img

#cp /tftpboot/uboot.bin .

#cp /tftpboot/steven_uImage .

#make –f Makefile.4M

Flash layout:

```
+-----------+--------+--------+-----------------------------+
| Uboot | UsrCfg |    RF  |    Linux Kernel Image |
+-----------+--------+--------+-----------------------------+

|<--------0x50000-------->|<-------0x3B0000------>|
```

-Original Uboot Image Size

149372 ./uboot.bin

- Original Kernel Image Size

2779348 ./steven_uImage

- Packed Image Size

4194304 ./steven_uImage.img

#ls –l

-rw-r--r-- 1 steven users 3831 Jun 24 19:00 Makefile.16M

-rw-r--r-- 1 steven users 2865 Jun 27 13:27 Makefile.4M

-rw-r--r-- 1 steven users 3744 Jun 24 19:00 Makefile.8M

-rw-r--r-- 1 steven users 2779348 Jun 27 13:34 steven_uImage

*-rwxr-xr-x 1 steven users 4194304 Jun 27 13:36 steven_uImage.img**

*-rwxr-xr-x 1 steven users 149372 Jun 27 13:34 uboot.bin**

The single image can now be burned using the flash programmer.

11.13 How to power down the rt305x Ethernet ports

Port	0	1	2	3	4
Map	W	L	L	L	L

MII control register

Bit	Name	Description	Read/Write	Default
15	mr_main_reset	1=Reset: 0=Normal, reset all digital logic, except phy_reg	R/ W; SC	1'h0
14	loopback_mii	Mii loop back	R/W	1'h0
13	force_speed	1 = 100Mbps: 0=10Mbps, when mr_autoneg_enable = 1'b0	R/W	1'h1
12	mr_autoneg_enable	1= Enabled: 0=Normal	R/W	1'h1
11	powerDown	phy into power down (power down analog TX analog RX, analog AD)	R/W	1'h0
10	reserved		RO	1'h0
9	mr_restart_negotiation	1 = Restart Auto-Negotiation: 0 = Normal	R/W; SC	1'h0
8	force_duplex	1 = Full Duplex: 0 = Half Duplex, when mr_autoneg_enable = 1'b0	R/W;PC	1'h1
7:0	RESERVED		RO	8h00

User Space:

 # mii_mgr -s -p 0 -r 0 -v 0x3900 //set port 0 register0 bit11

Set: phy[0].reg[0] = 3900

 # mii_mgr -s -p 1 -r 0 -v 0x3900 //set port 1 register0 bit11

Set: phy[1].reg[0] = 3900

 # mii_mgr -s -p 2 -r 0 -v 0x3900 //set port 2 register0 bit11

Set: phy[2].reg[0] = 3900

 # mii_mgr -s -p 3 -r 0 -v 0x3900 //set port 3 register0 bit11

Set: phy[3].reg[0] = 3900

 # mii_mgr -s -p 4 -r 0 -v 0x3900 //set port 4 register0 bit11

Set: phy[4].reg[0] = 3900

Kernel Space:

*extern u32 mii_mgr_read(unsigned int , unsigned int, unsigned int *);*

extern u32 mii_mgr_write(unsigned int, unsigned int, unsigned int);

mii_mgr_write(0, 0, 0x3900) //set port 0 register0 bit11

mii_mgr_write(1, 0, 0x3900) //set port 1 register0 bit11

mii_mgr_write(2, 0, 0x3900) //set port 2 register0 bit11

mii_mgr_write(3, 0, 0x3900) //set port 3 register0 bit11

mii_mgr_write(4, 0, 0x3900) //set port 4 register0 bit11

You also need to set POC[27:23] to disable Phy port.

RT288x_SDK/source/linux-2.6.21.x/drivers/net/raeth/rather.c)

```
*(unsigned long *)(0xb0110090) = 0x0??07f7f;
```

POC1: Port Control 0 (offset: 0x90)

Bits	Type	Name	Description	Initial value
31:30	R/W	HASH_ADDR_SHIFT	Address table hashing algorithm option for member set index	2'b0
29	R/W	DIS_GMII_PORT_1	Disable port 6 1: port disable (if dumb mode, default = 0)	1'b1
28	R/W	DIS_GMII_PORT_0	Disable port 5 1: port disable (if dumb mode, default = 0)	1'b1
27:23	R/W	DIS_PORT	Disable phy port 1: port disable (if dumb mode, default = 0)	5'h1f
22:16	R/W	DISRMC2_CPU	1: disable RMC packet to cpu	7'h0
15	RO	-	Reserved	1'b0
14:8	R/W	EN_FC	Enable pause flow control enable 802.3x flow control	7'h7f
7	RO	-	Reserved	1'b0
6:0	R/W	Reserved	Enable back pressure 1: enable back pressure (but need to qualify BP_mode)	7'h7f

11.14 How to enable NFS client

#make menuconfig

Kernel/Library/Defaults Selection--->

Networking options --->

[*] IP: kernel level autoconfiguration

File systems --->

Network File Systems --->

Linux 2.4:

<*> NFS file system support

[*] Provide NFSv3 client support

[*] Allow direct I/O on NFS files (EXPERIMENTAL)

[*] Root file system on NFS

Linux 2.6

<> NFS file system support*

[*] Provide NFSv3 client support

[*] Provide client support for the NFSv3 ACL protocol extension

[*] Provide NFSv4 client support (EXPERIMENTAL)

[*] Allow direct I/O on NFS files

Kernel/Library/Defaults Selection--->

[*] Customize Kernel Settings (NEW)

[*] Customize Busybox Settings

Linux System Utilities--->

[*] mount

[ ] Support mount helpers

[*] Support mounting NFS file systems

Example:

```
# mount -o nolock 192.168.18.21:/tftpboot /mnt

# mount

……

/dev/sda1 on /media/sda1 type vfat
```

```
(rw,fmask=0000,dmask=0000,codepage=cp437,iocharset=iso8859-1)

192.168.18.21:/tftpboot on /mnt type nfs

(rw,vers=3,rsize=32768,wsize=32768,hard,nolock,proto=udp,timeo=7,retrans=3,sec=sys,addr=192.168.18.21)
```

11.15 How to add a new language to the web UI

The following instructions are an example and show how to add the Korean language to the web UI.

1. Copy all the xml files under RT288x_SDK/source/user/goahead/web/lang/en to RT288x_SDK/source/user/goahead/web/lang/kr and translate the “msgstr” part in those files. (Note: the translation should be UTF-8 encoded)

2. Add an entry to RT288x_SDK/source/config/config.in:

```
dep_bool '   language pack - Korean'     CONFIG_USER_GOAHEAD_LANG_KR
$CONFIG_USER_GOAHEAD_HTTPD
```

3. Add an entry toRT288x_SDK/source/user/goahead/Makefile:

```
ifneq ("$(CONFIG_USER_GOAHEAD_LANG_KR)", "y")
    rm -rf $(ROMFSDIR)/$(ROOT_DIRECTORY)/lang/kr
endif
```

4. RT288x_SDK/source/user/goahead/src/utils.c:

   Add to ‘getLangBuilt’ function:

```
    else if (!strncmp(lang, "kr", 5))
#ifdef CONFIG_USER_GOAHEAD_LANG_KR
    return websWrite(wp, T("1"));
#else
    return websWrite(wp, T("0"));
#endif
```

5. RT288x_SDK/source/user/goahead/web/overview.asp

   Add to ‘initValue’ function:

```
var lang_kr = "<% getLangBuilt("kr"); %>";
if (lang_kr == "1")
     lang_element.options[lang_element.length] = new Option('Korean', 'kr');
```

6. RT288x_SDK/source/user/goahead/web/adm/management.asp

   Add to ‘initValue’ function:

```
var lang_kr = "<% getLangBuilt("kr"); %>";
if (lang_kr == "1")
     lang_element.options[lang_element.length] = new Option('Korean', 'kr');
```

7. RT288x_SDK/source/user/goahead/web/home.asp

   Fix 'initLanguage' function

8. make menuconfig

   Customize Vendor/User Settings ---> Network Applications ---> select Korean language pack

11.16 How to enable watchdog in RT305x

#make menuconfig

Kernel/Library/Defaults Selection --->

[*] Customize Kernel Settings

Device Drivers --->

Character devices --->

Watchdog Cards --->

<M> Ralink APSoC Hardware Watchdog

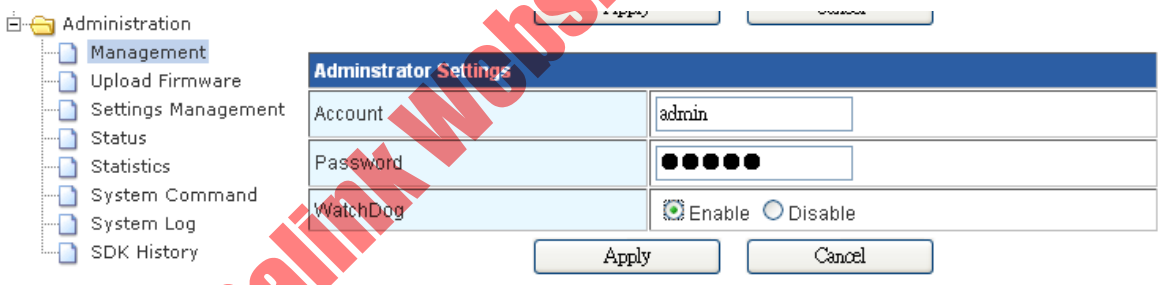
[*] Ralink WatchDog Reset Output

[*] Customize Vendor/User Settings

Miscellaneous Applications --->

[*] watchdog

Finally, Enable watchdog in WebUI.



11.17 How to enable USB storage on the RT305x platform

#make menuconfig

Kernel/Library/Defaults Selection --->

[*] Customize Kernel Settings (NEW)

Device Drivers --->

SCSI device support --->

<*> SCSI device support

<*> SCSI disk support

USB support ---->

<*> Support for Host-side USB

[*] USB verbose debug messages

[*] USB device filesystem

<*> USB Mass Storage support

[*] USB Mass Storage verbose debug

File systems ---->

<*> Filesystem in Userspace support

DOS/FAT/NT Filesystems ---->

<*> VFAT (Windows-95) fs support

(437) Default codepage for FAT (NEW)

(iso8859-1) Default iocharset for FAT (NEW)

Partition Types ---->

[*] Advanced partition selection

[*] PC BIOS (MSDOS partition tables) support (NEW)

Native Language Support ---->

(iso8859-1) Default NLS Option

<*> Codepage 437 (United States, Canada)

<*> Traditional Chinese charset (Big5)

<*> NLS ISO 8859-1 (Latin 1; Western European Languages)

<*> NLS UTF-8

Ralink Module ---->

<M> RALINK DWC_OTG support

[ ] enable debug mode

[*] HOST ONLY MODE

[ ] DEVICE ONLY MODE

CAUTION: THE KERNEL SIZE CANNOT BE BIGGER THAN THE MTD KERNEL PARTITION SIZE IN ROOTFS IN FLASH MODE.

#==

Original Kernel Image Size

1033369 /home/steven/rt3052/RT288x_SDK/source/images/zImage.lzma

Padded Kernel Image Size

1048512 /home/steven/rt3052/RT288x_SDK/source/images/zImage.lzma

Original RootFs Size

……

11.18 How to enable USB automount on the RT305x platform

#make menuconfig

Kernel/Library/Defaults Selection --->

[*] Customize Busybox Settings

Linux System Utilities --->

[*] mdev

[*] Support /etc/mdev.conf

[ ] Support subdirs/symlinks (NEW)

[*] Support command execution at device addition/removal

[*] Customize Vendor/User Settings

Miscellaneous Applications --->

[*] ntfs-3g

11.19 How to enable software QoS

Kernel/Library/Defaults Selection --->

[*] Customize Vendor/User Settings

Ralink RT288x Application --->

[*] RT2880 QoS

[*] Customize Kernel Settings

Networking --->

Networking options --->

[*] Network packet filtering framework (Netfilter) --->

Core Netfilter Configuration --->

<*> Netfilter connection tracking support

Netfilter connection tracking support (Layer 3 Independent Connection tracking)

[*] Connection tracking flow accounting

<*> "DSCP" target support

<*> "MARK" target support

<*> "TCPMSS" target support

<*> "conntrack" connection tracking match support

<*> "DSCP" match support

<*> "helper" match support

<*> "length" match support

<*> "state" match support

<*> "layer7" match support

<*> "webstr" match support

<*> "tcpmss" match support

IP: Netfilter Configuration ---->

<*> IP tables support (required for filtering/masq/NAT)

<*> Packet filtering

<*> REJECT target support

<*> Packet mangling

QoS and/or fair queueing ---->

[*] QoS and/or fair queueing

<*> Hierarchical Token Bucket (HTB)

<*> Random Early Detection (RED)

<*> Stochastic Fairness Queueing (SFQ)

<*> Generic Random Early Detection (GRED)

<*> Differentiated Services marker (DSMARK)

<*> Elementary classification (BASIC)

<*> Traffic-Control Index (TCINDEX)

<*> Netfilter mark (FW)

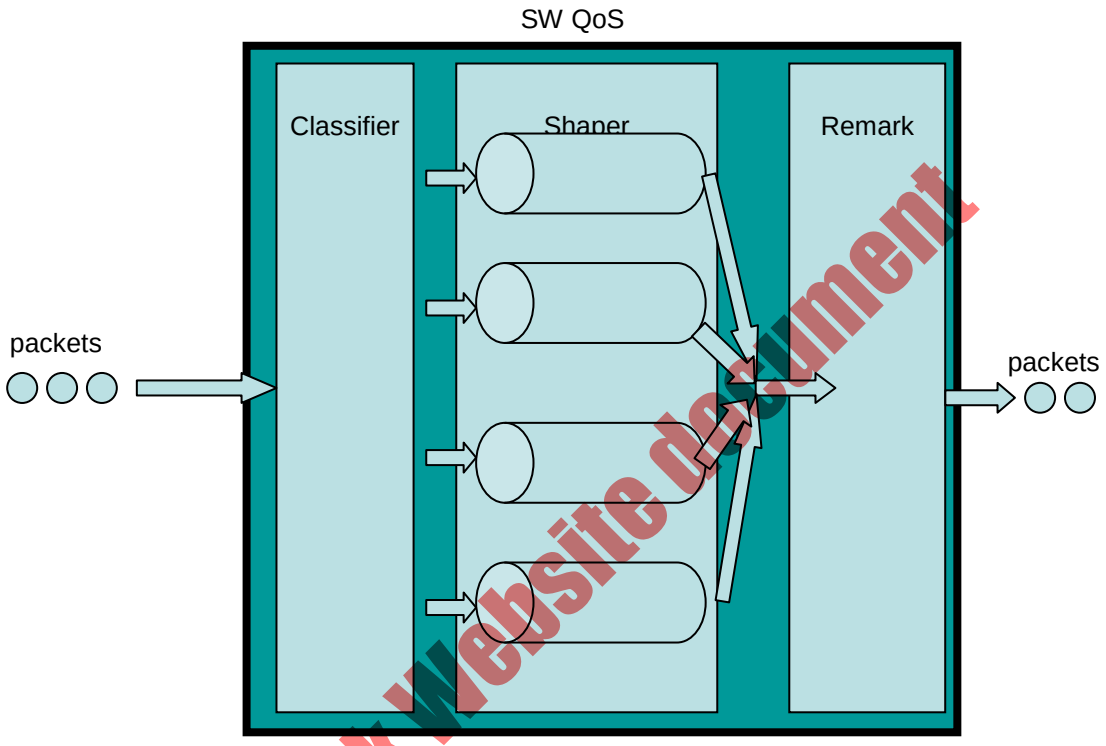
11.20 Software QoS information

11.20.1 Software QoS – Preface

The Ralink SoC SW QoS supports many different types of classification, rate limitation, and DSCP remarking. Ralink SoC SW QoS is based on the Linux Qdiscs, TC, and iptables. Ralink SoC SW QoS only supports upload stream on a WAN interface at this time.

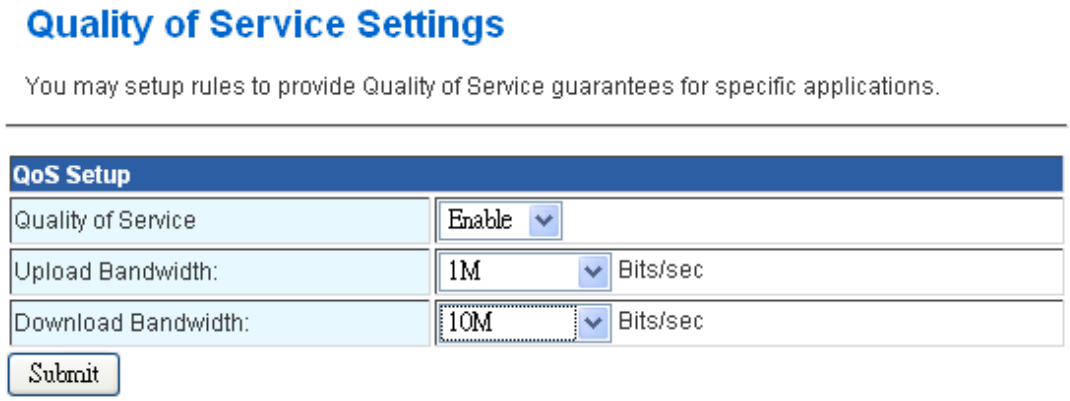
11.20.2 Software QoS – Concept

The Ralink SoC SW QoS architecture is shown in the subsequent figure. The Classifier module classifies incoming packets into the Shaper module. The Shaper module has 4 queues (groups) to do rate limitation, and then the Remark module rewrites the DSCP field of the packet if it is necessary.

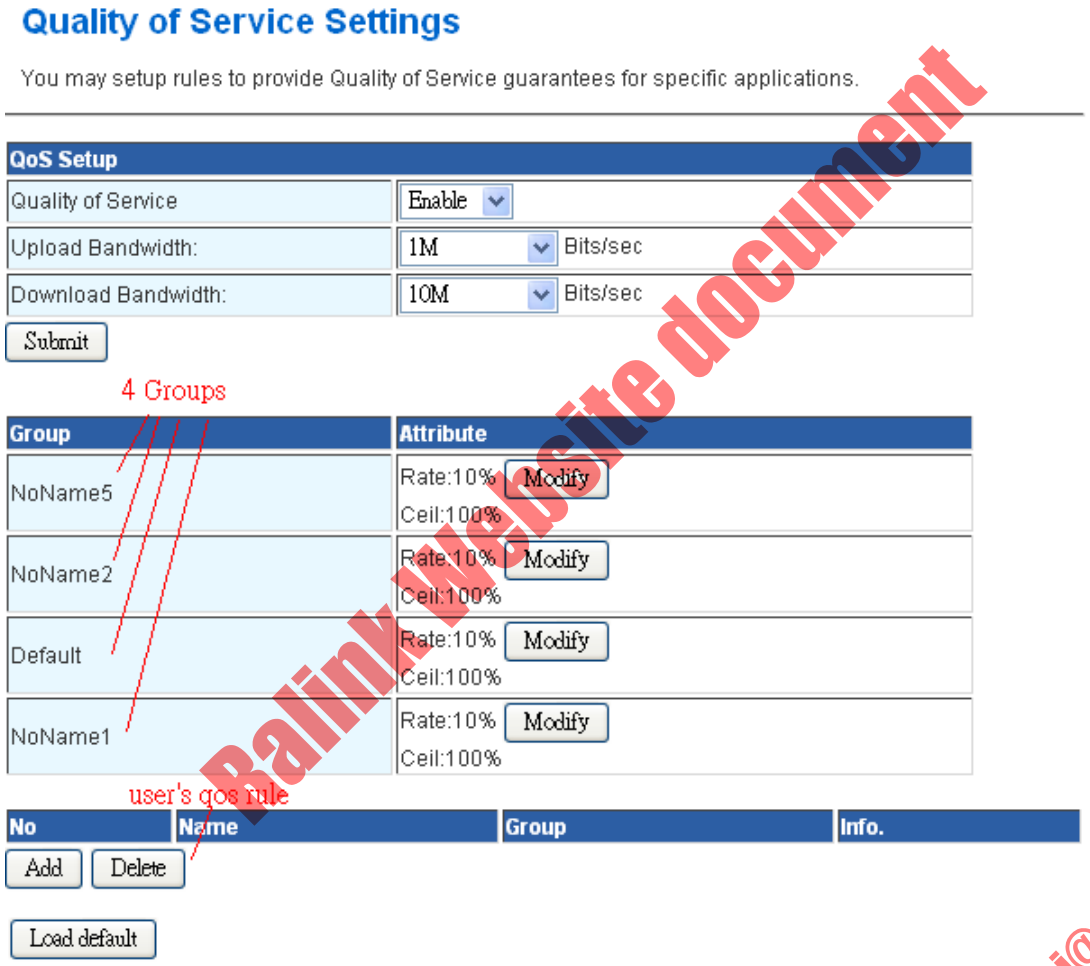


11.20.3 Software QoS – Usage

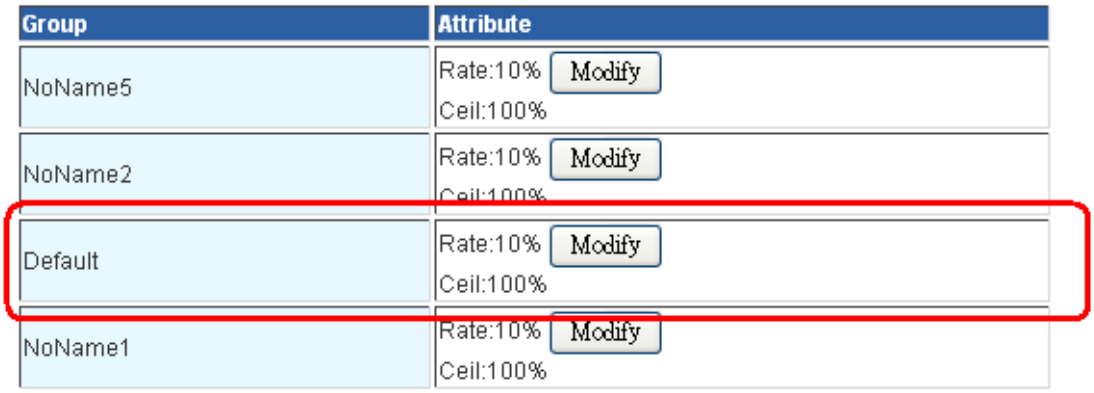
1. Select “Enable” on the web UI to enable the Software QoS.
2. Enter the upload and download bandwidth details to make a good fit with the user’s network environment (e.g. ADSL 512k/64k, Cable Modem 10M/10M….)



4 groups would are shown after enabling QoS. Now all packets through this gateway are classified into these 4 groups according to the user's QoS rules. The 4 groups are subsequently shown.



IF ONE PACKET DOES NOT AGREE WITH ANY RULES, THEN THE PACKET IS CLASSIFIED INTO THE DEFAULT GROUP (THE THIRD GROUP).

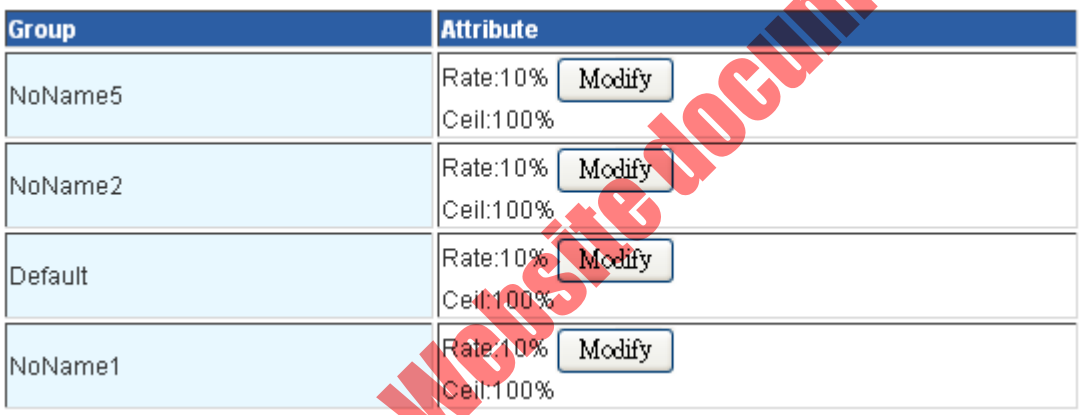


There are 3 attributes in every group, as shown in the subsequent figure.

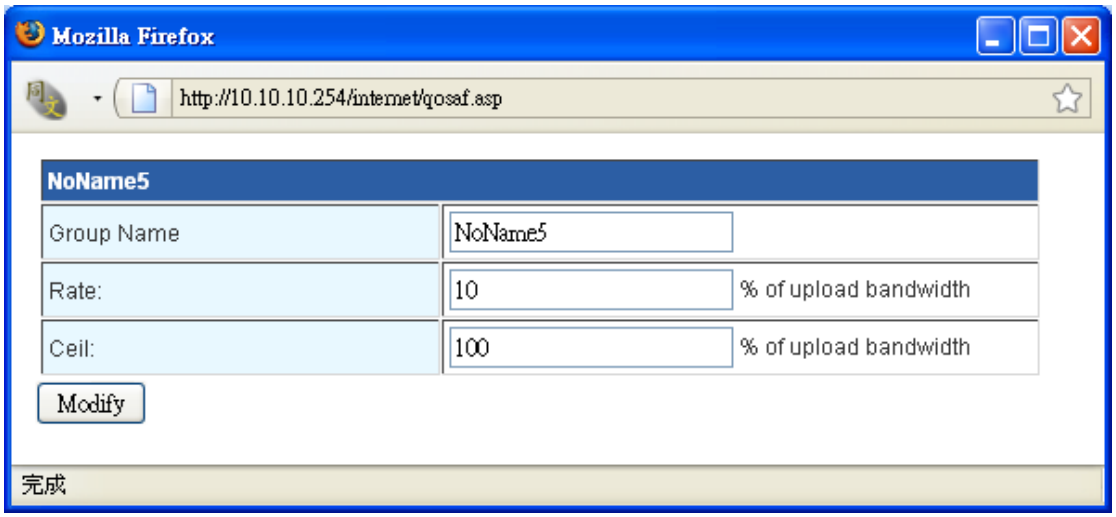
a. Name: The user can define the group name.

b. Rate: Set the guaranteed minimum bandwidth that a group can use.

c. Ceil: Set the maximum bandwidth that a group can use.



The user can press “Modify” to adjust the 3 attributes in every group.



In these groups, the first group has the highest priority. The next group has the second priority, the third group is the default group. The last group has the lowest priority.

Group	Attribute
NoName5	Rate:10% Modify Highest Ceil:100%
NoName2	Rate:10% Modify Middle Ceil:100%
Default	Rate:10% Modify Default Ceil:100%
NoName1	Rate:10% Modify Lowest Ceil:100%

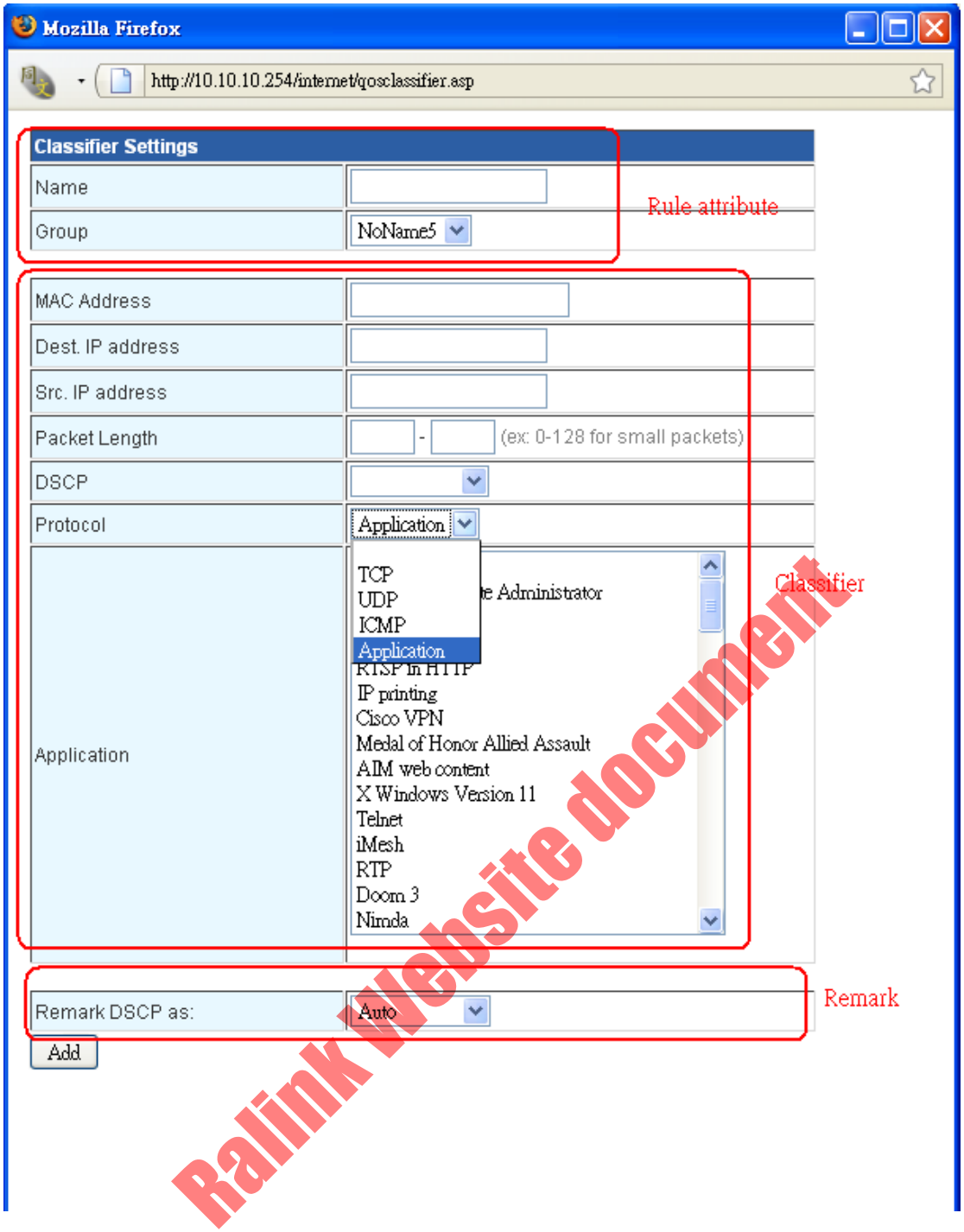
"Highest priority" means the available bandwidth will serve the group first, but rules about guaranteed rate and ceil in every group are still met. For example, people often hope VoIP traffic is classified as the highest priority group for short latency, and P2P traffic to be classified as the lowest priority.

The QoS rule has now been made to do classification, and remarking.

No	Name	Group	Info.

Add Delete

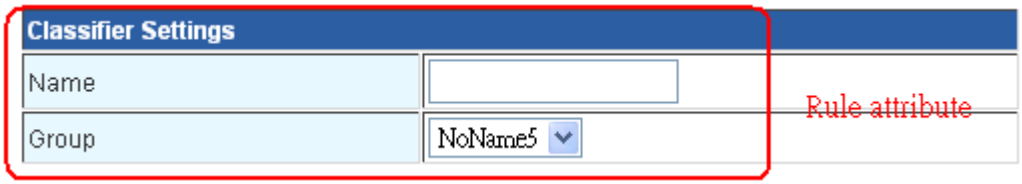
There are 3 areas in every QoS rule ("rule attribute", "classifier", and "remark").



1) Rule attribute:

   a) Name: specifies this rules name

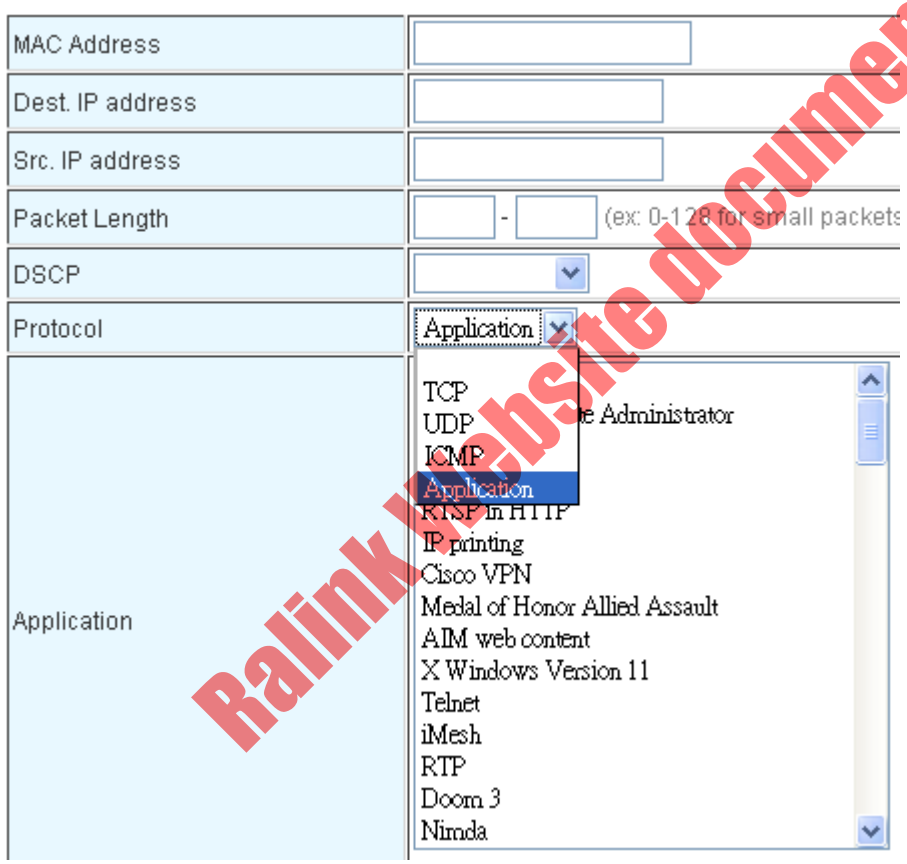
   b) Group: specifies which group this rule is belongs to.



2) Classifier:

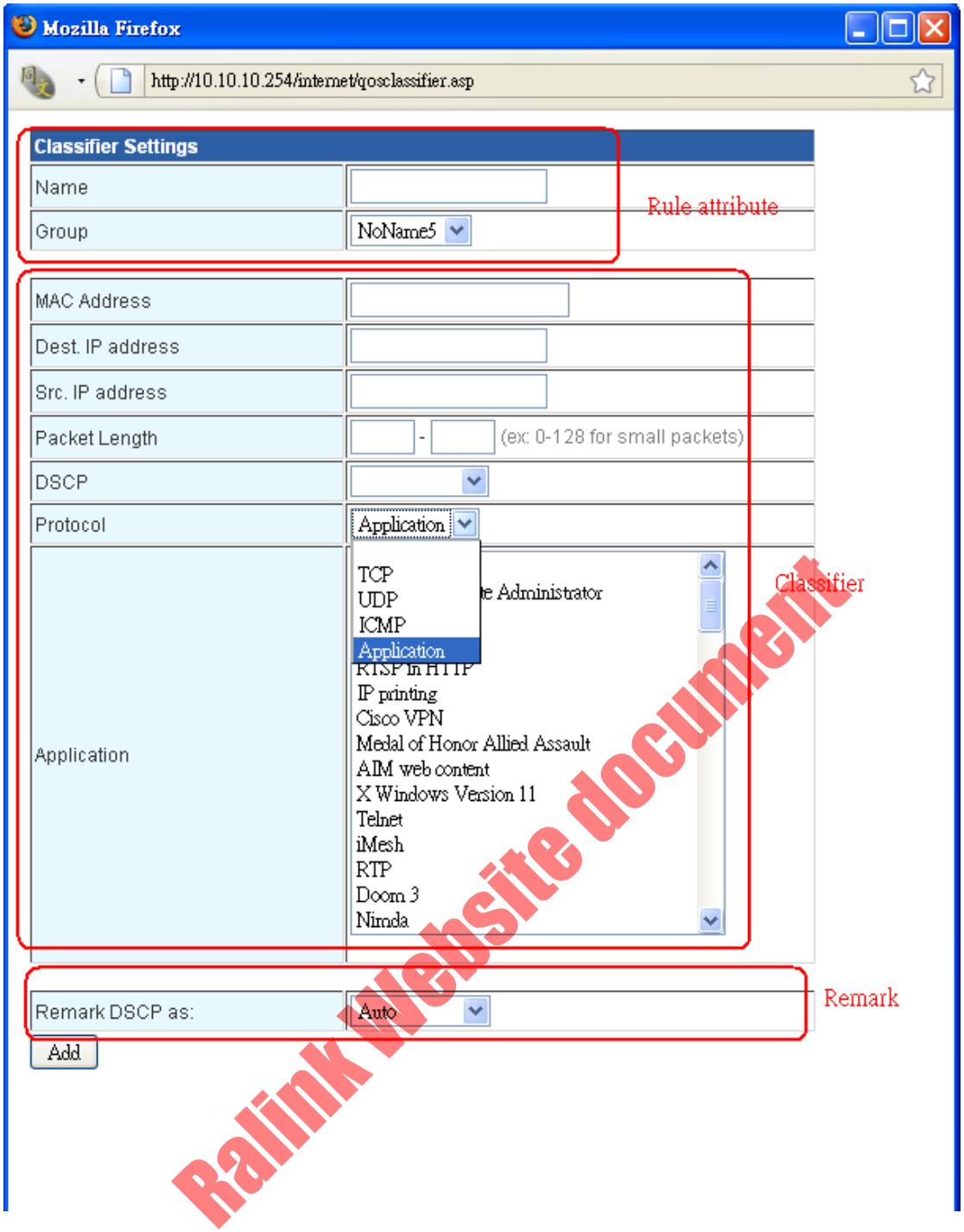
RT2880 QoS supports these classifiers at this time:

a) SRC MAC Address
b) SRC/DSP IP address (with netmask)
c) Packet length
d) DSCP field
e) ICMP, TCP/UDP port range
f) Layer 7 (content inspection)

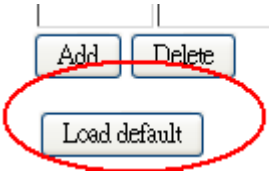


3) Remark:

This argument specifies what DSCP value would be added to the packet as a remark which matches the rule.



There is a set of QoS rules saved as default settings. They can be used by choosing “Load Default”.



The default QoS settings are shown subsequently. These settings are an example.

Group	Attribute
High	Rate:30% Modify Ceil:100%
Middle	Rate:20% Modify Ceil:100%
Default	Rate:5% Modify Ceil:100%
Low	Rate:10% Modify Ceil:100%

No	Name	Group	Info.
1	ICMP_HIGH	High	Protocol: ICMP Remark DSCP :EF
2	Small_Packet_HIGH	High	Packet Length: 0 - 128 Remark DSCP :EF
3	VoIP_H323_HIGH	High	Protocol: Application Application: h323 Remark DSCP :EF
4	VoIP_SIP_HIGH	High	Protocol: Application Application: sip Remark DSCP :EF
5	VoIP_Skype1_HIGH	High	Protocol: Application Application: skypeout Remark DSCP :EF
6	VoIP_Skype2_HIGH	High	Protocol: Application Application: skypetoskype Remark DSCP :EF
7	RTP_HIGH	High	Protocol: Application Application: rtp Remark DSCP :EF
8	SSH_HIGH	High	Protocol: Application Application: ssh Remark DSCP :EF
9	MSN_Messenger_MIDDLE	Middle	Protocol: Application Application: msnmessenger Remark DSCP :AF21
10	Yahoo_MIDDLE	Middle	Protocol: Application Application: yahoo Remark DSCP :AF21
11	POP3_LOW	Low	Protocol: Application Application: msnmessenger Remark DSCP :AF11
12	SMTP_LOW	Low	Protocol: Application Application: smtp Remark DSCP :AF11
13	P2P_eMule_LOW	Low	Protocol: Application Application: edonkey Remark DSCP :AF11
14	P2P_BT_LOW	Low	Protocol: Application Application: bittorrent Remark DSCP :AF11

As shown previously, all ICMP, small packets, VoIP related, and SSH packets are classified into the highest group named “High” which has 30% minimal guaranteed bandwidth. That means all these packets share at least 30% upload bandwidth.

In this example, the left bandwidth can be evaluated after satisfying every group is 100 – 30 – 20 – 5 – 10 = 35 (%).

The 4 groups do not share (or fight for) the 35% bandwidth remaining. The design of the system recognizes the priority. In this design, the “High” group has the highest priority. After the other 3 groups are satisfied, the best case that “High” group can have is 30 + 35 = 65 (%) bandwidth. (Or 100 – 20 – 5 – 10 = 65%).

VoIP packets have the highest priority, but if someone floods ICMP packets to consume all bandwidth of the group (ICMP packets also have the highest priority), it is still possible that VoIP packets can be delayed. Therefore, the smallest guaranteed bandwidth is per group, but not per rule.

11.21 How to enable USB Ethernet (exampel for ASIX AX88XXX)

Kernel/Library/Defaults Selection --->

[*] Customize Kernel Settings

Device Drivers --->

USB support --->

USB Network Adapters --->

<M> Multi-purpose USB Networking Framework

<M> ASIX AX88xxx Based USB 2.0 Ethernet Adapters

<M> CDC Ethernet support (smart devices such as cable modems)

CONFIG_USB_RTL8150=m

 # insmod usbnet

 # insmod cdc_ether

usbcore: registered new interface driver cdc_ether

 # insmod asix.ko

usbcore: registered new interface driver asix

usb 1-1: new high speed USB device using dwc_otg and address 2

usb 1-1: Product: USB2.0

usb 1-1: Manufacturer: ASIX Elec. Corp.

usb 1-1: SerialNumber: 01

usb 1-1: configuration #1 chosen from 1 choice

eth0: register 'asix' at usb-lm0-1, ASIX AX8817x USB 2.0 Ethernet, 00:0e:2e:41:72:9e

 # brctl addif br0 eth0

device eth0 entered promiscuous mode

 # brctl show br0

bridge name	bridge id	STP enabled	interfaces
br0	8000.000c43414367	no	ra0
			eth2.1
			eth0

 # ifconfig eth0 up

ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready

br0: port 3(eth0) entering learning state

eth0: link up, 100Mbps, full-duplex, lpa 0xC5E1

br0: topology change detected, propagating

br0: port 3(eth0) entering forwarding state

 # ping 10.10.10.3

PING 10.10.10.3 (10.10.10.3): 56 data bytes

64 bytes from 10.10.10.3: seq=0 ttl=128 time=3.381 ms

64 bytes from 10.10.10.3: seq=1 ttl=128 time=1.038 ms

64 bytes from 10.10.10.3: seq=2 ttl=128 time=1.067 ms

64 bytes from 10.10.10.3: seq=3 ttl=128 time=1.069 ms

11.22 How to build a single image for the RT2880 8M flash platform

#cd Uboot

#make menuconfig

```
(128Mb) DRAM Component

(32bits) DRAM Bus

(8M) Flash Size
```

#cd RT288x_SDK/source

#make menuconfig

```
Kernel/Library/Defaults Selection    --->
   [*] Customize Kernel Settings
```

```
        Machine selection    --->
         (8M) Flash Size
```

#cd RT288x_SDK/tools/single_img/RT2880

#vi Makefile.8M

```
UBOOT_IMAGE = rt2880_100phy_128Mbx16_8Mflash.uboot

KERNEL_IMAGE = rt2880_100phy_128Mbx16_8Mflash.linux

PACKED_IMAGE = rt2880_100phy_128Mbx16_8Mflash.uboot
```

#make –f Makefile.8M

Flash layout:

```
+----------------------+---------+---------+----+-------------------------------+
| KERNEL PartII    | Uboot |UsrCfg| RF| Kernel PartI                |
+----------------------+-------+------+---+-----------------------------------+

|<----0x400000-->|<----0x50000->|←---------0x3B0000    --------->|
```

11.23 How to start a printer server (example for HP officejet 4355)

Step1: SDK Configuration

#make menuconfig

Kernel/Library/Defaults Selection --->

[*] Customize Kernel Settings

Device Drivers --->

USB support --->

<*> USB Printer support

[*] Customize Vendor/User Settings

Network Applications --->

[*] p910nd (small printer daemon)

Step2: Plug in USB Printer

```
# usb 1-1: new full speed USB device using dwc_otg and address 2
```

```
usb 1-1: Product: Officejet 4300 series

usb 1-1: Manufacturer: HP

usb 1-1: SerialNumber: CN864GZ1S004GR

usb 1-1: configuration #1 chosen from 1 choice

drivers/usb/class/usblp.c: usblp0: USB Bidirectional printer dev 2 if 1 alt 0 proto 2 vid 0x03F0 pid

 0x5411
```

Step3: run the printer daemon

p910nd -f /dev/lp0

Step4: Setup the printer in Windows

11.24 How to force the RT3052 link speed

There are two kinds of force mode that refer to the configuration of the remote peer.

1. Force Mode (Both RT305x and remote peer disable auto negotiation algorithm)

- **10MB/Full:** Set bit13=0, bit12=0,bit8=1 (reg_addr=0)
- **10MB/Half**:Set bit13=0,bit12=0,bit8=0 (reg_addr=0)
- **100MB/Full**:Set bit13=1,bit12=0,bit8=1 (reg_addr=0)
- **100MB/Half**:Set bit13=1,bit12=0,bit8=0 (reg_addr=0)

CR → Address:00(d00) → Reset State:3100

Bit	Read/Write	Name	Description	Default
15	R/W; SC	MR_MAIN_RESET	1=Reset: 0=Normal, reset all digital logic, except phy_reg	1'h0
14	R/W	LOOPBACK_MII	Mii loop back	1'h0
13	R/W	FORCE_SPEED	1 = 100Mbps: 0=10Mbps, when mr_autoneg_enable = 1'b0	1'h1
12	R/W	MR_AUTONEG_ENABLE	1= Enabled: 0=Normal	1'h1
11	R/W	POWERDOWN	phy into power down (power down analog TX analog RX, analog AD)	1'h0
10	RO	-	Reserved	1'h0
9	R/W; SC	MR_RESTART_NEGOTIATION	1 = Restart Auto-Negotiation: 0 = Normal	1'h0
8	R/W	FORCE_DUPLEX	1 = Full Duplex: 0 = Half Duplex, when mr_autoneg_enable = 1'b0	1'h1
7:0	RO	-	Reserved	8h00

2. Auto negotiation (Both RT305x and remote peer enable auto negotiation algorithm)

- **10MB/Full: Set bit6=1 (reg_addr=4)**
- **10MB/Half: Set bit5=1 (reg_addr=4)**
- **100MB/Full: Set bit8=1 (reg_addr=4)**
- **100MB/Half: Set bit7=1 (reg_addr=4)**

Auto-Negotiation advertisement register

CR → Address:04(d04) → Reset State: → 05e1

Bit	Read/Write	Name	Description	Default
15	R0	Next Page Enable	1=Set to use Next Page: 0=Not to use Next Page	1'h0
14	RO	-	Reserved	1'h0
13	R/W	Remote Fault Enable	1 = Auto Negotiation Fault Detected 0 = No Remote Fault	1'h0
12:11	RO	Not Implemented	Technology Ability A7-A6	2'h0
10	R/W	Pause	Technology Ability A5	1'h1
9	RO	Not Implemented	Technology Ability A4	1'h0
8	R/W	100Base-TX Full Duplex Capable	1 = Capable of Full Duplex 0 = Not Capable	1'h1
7	R/W	100 Base-TX Half Duplex Capable	1 = Capable of Half Duplex 0 = Not Capable	1'h1
6	R/W	10 Base-T Full Duplex Capable	1 = Capable of Full Duplex 10BASE-T 0 = Not Capable	1'h1
5	R/W	10 Base-T Half Duplex Capable	1 = Capable of Half Duplex 10BASE-T 0 = Not Capable	1'h1
4:0	R/W	Selector Field	Identifies type of message	5'h01

User Mode:

 # mii_mgr -s -p [port_no] -r [reg_addr] -v [Value]

Kernel Space:

extern u32 mii_mgr_write(unsigned int, unsigned int, unsigned int);

mii_mgr_write([port_no], [reg_addr], [value])

NOTES: IF BOTH RT305X SWITCH AND REMOTE PEER DO NOT USE THE SAME CONFIGURATION (I.E. AUTO-NEGOTIATION OR FORCE MODE) IT CAN CAUSE A PROBLEM.

11.25 How to verify IGMP snooping function

Step1: Compiling IGMP proxy application.

#make menuconfig

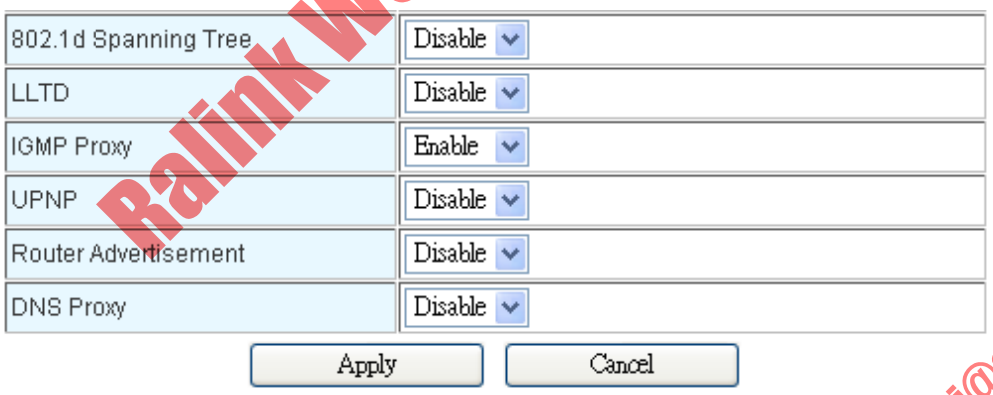
Kernel/Library/Defaults Selection --->

[*] Customize Vendor/User Settings (NEW)

Network Applications --->

[*] igmp proxy (RFC4605)

Step2: Enable IGMP Proxy in WebUI.



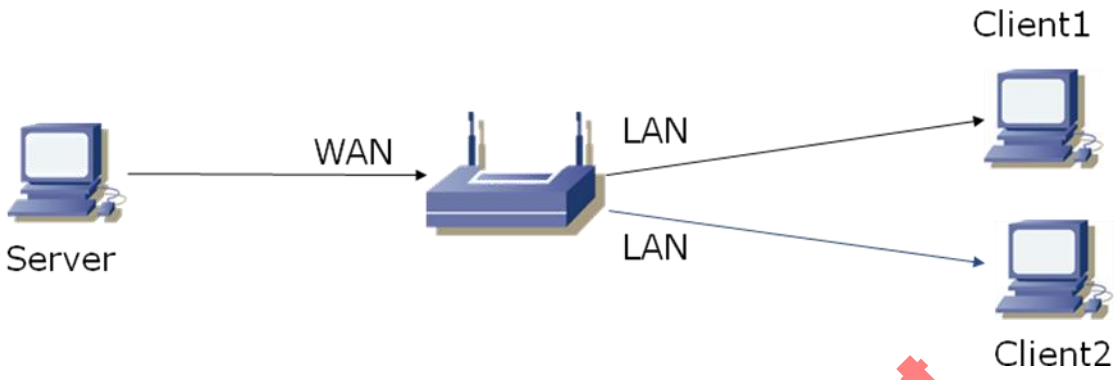
Step3: Install windows server 2003 resource kit tools in your PCs.

You can get the test application from the following link or Ralink SDK.

- HTTP://WWW.MICROSOFT.COM/DOWNLOADS/DETAILS.ASPX?FAMILYID=9D467A69-57FF-4AE7-96EE-B18C4790CFFD&DISPLAYLANG=EN

- RT288x_SDK/source/user/igmpproxy/tools/rktools.exe.

Step4: Start Multicast test



Mcast server:

C:\>mcast /GRPS:239.1.1.1 /SRCS:10.10.10.3 /NUMPKTS:1000 /INTVL:50 /SEND

(Please use "/intf" argument to specify an interface to receive or send if you have multiple network interfaces.)

Now, you can see the multicast packets will be generated by Mcast Server.

```
Frame 42 (290 bytes on wire, 290 bytes captured)
Ethernet II, Src: Msi_9f:da:b7 (00:16:17:9f:da:b7), Dst: IPv4mcast_01:01:01 (01:00:5e:01:01:01)
  Destination: IPv4mcast_01:01:01 (01:00:5e:01:01:01)
  Source: Msi_9f:da:b7 (00:16:17:9f:da:b7)
  Type: IP (0x0800)
Internet Protocol, Src: 10.10.10.3 (10.10.10.3), Dst: 239.1.1.1 (239.1.1.1)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
  Total Length: 276
  Identification: 0x5ae1 (23265)
  Flags: 0x00
  Fragment offset: 0
  Time to live: 5
  Protocol: Unknown (0xff)
  Header checksum: 0x54fb [correct]
  Source: 10.10.10.3 (10.10.10.3)
  Destination: 239.1.1.1 (239.1.1.1)
Data (256 bytes)
  Data: FFFFFFFFFFFFFFFF0102030405060708090A0B0C0D0E0F10...
  [Length: 256]
```

Mcast Client1:

C:\>mcast /GRPS:239.1.1.1 /RECV

Step5: Starting network sniffer on Client1 and Client2.

The right behavior is only Client1 can receive multicast packets.