

MediaTek Inc.

RALINK AP SDK 4.3.0.0 USER'S MANUAL

Copyright © 2014 MediaTek Inc.

All Rights Reserved.

This document is property of MediaTek Inc., receipt, or possession of this document does not express, license, or imply any rights to use, sell, design, or manufacture from this information or the software documented herein. No reproduction, publication, or disclosure of this information, in whole or in part, shall be allowed, unless the prior written consent of MediaTek Inc. is obtained.

NOTE: THIS DOCUMENT CONTAINS SENSITIVE INFORMATION AND HAS RESTRICTED DISTRIBUTION.

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 MediaTek (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. MediaTek 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 MediaTek 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.

MediaTek 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 MediaTek's prior written consent.

MediaTek reserves the right, at its own discretion, to update or revise the Proprietary Information from time to time, of which MediaTek 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.

MediaTek Inc. (Taiwan)

5F, No.5, Tai-Yuen 1st 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	10
2	Version History.....	14
3	Overview of the Ralink AP Demo Board	15
3.1	RT2880.....	15
3.2	RT3052.....	19
3.3	RT3883.....	21
3.4	RT3352.....	24
3.5	RT5350.....	26
3.6	RT6855.....	28
3.7	RT6856.....	29
3.8	MT7620	31
3.9	MT7621	35
3.10	MT7628.....	39
4	AP SDK source code overview	42
5	Tool-chain.....	43
5.1	Install toolchain.....	43
5.2	Install LZMA Utility	43
5.3	Install mksquashfs utility	44
6	Boot loader	46
6.1	Uboot Configuration.....	46
6.2	Build the uboot Image	49
6.3	Burn the uboot image.....	50

7	User Library	51
7.1	Library Configuration	51
7.2	Library Porting	52
7.3	Build user library	53
8	User Application.....	54
8.1	Ralink Proprietary Applications	54
8.2	accel-pptp.....	62
8.3	bigpond.....	62
8.4	bonnie++	62
8.5	bridge-utils	62
8.6	busybox	62
8.7	comgt-0.32	63
8.8	ctorrent-dnh3.2	63
8.9	curl.....	63
8.10	dhcp6.....	63
8.11	dnsmasq-2.40	63
8.12	dropbear-0.52	64
8.13	ebtables	64
8.14	ecmh.....	64
8.15	GoAhead.....	64
8.16	igmpproxy	64
8.17	inadyn.....	64

8.18	iperf.....	65
8.19	iproute2	65
8.20	iptables.....	65
8.21	lighttpd.....	65
8.22	linux-igd.....	65
8.23	lldt(lld2d)	66
8.24	matrixssl-1.8.3.....	66
8.25	miniupnpd-1.6	66
8.26	mkdosfs-2.11.....	66
8.27	mpstat-0.1.1	66
8.28	mtd-utils	66
8.29	ntfs-3g	67
8.30	ntfsprogs	67
8.31	ntpclient.....	67
8.32	nvram library	67
8.33	openl2tp-1.6	67
8.34	openswan-2.6.38.....	67
8.35	p910nd	68
8.36	pciutils-3.0.0	68
8.37	ppp-2.4.2.....	68
8.38	pptp-client	68
8.39	proftpd	68

8.40	radvd-1.0.....	69
8.41	rp-l2tp-0.4	69
8.42	rp-pppoe-3.8.....	69
8.43	samba	69
8.44	sdparm-1.02	69
8.45	strace.....	70
8.46	taskset.....	70
8.47	tcpdump	70
8.48	totd-1.5.....	70
8.49	usb_modeswitch-0.9.5.....	70
8.50	uvc_stream	71
8.51	wireless_tools.....	71
8.52	wpa_supplicant-0.5.7	71
8.53	wsc_upnp.....	71
8.54	zebra-0.95a_ripd.....	71
8.55	Port new user application.....	72
9	Linux Kernel	74
9.1	Linux configuration.....	74
9.2	Change Flash/DRAM Size.....	76
9.3	Change Switch Controller in RT288x SDK	77
9.4	Update User/Kernel default settings	78
9.5	Compile Linux image with root file system.....	79

9.6	Port new Linux kernel module.....	80
9.7	Execute commands at boot up time.....	82
9.8	Add new files in RootFs	82
9.9	Reduce Image size	83
10	Flash Layout and Firmware Upgrade	87
10.1	Flash Layout.....	87
10.2	Firmware Upgrade.....	89
11	FAQ	90
11.1	Default password/UART/networking setting.....	90
11.2	System requirements for the host platform.....	90
11.3	Quickly Build a Default Reference Image	91
11.4	How to configure MT7620 to connect with the external switch	91
11.5	How to configure MT7621 internal switch to get more GPIO pins	92
11.6	How to add new default parameter in flash	93
11.7	Enable Ethernet Converter / AP Client Mode.....	95
11.8	How to change the Ethernet MAC address	98
11.9	How to change the Wireless MAC address	99
11.10	How to configure GPIO ports.....	100
11.11	Use GPIO to turn on LED.....	102
11.12	Use LED firmware to turn on LED	105
11.13	How to start the telnet server.....	107
11.14	11n bit rate derivation	109

11.15	How to build a single image for the flash programmer	112
11.16	How to power down the RT3x5x/RT5350/MT7628 build-in 10/100 PHYs.....	114
11.17	How to power down the RT6855/RT6856/MT7620 build-in 10/100 PHYs	115
11.18	How to enable NFS client.....	117
11.19	How to add a new language to the web UI.....	118
11.20	How to enable watchdog	119
11.21	How to enable USB storage on the RT305x platform.....	120
11.22	How to enable USB automount on the RT305x platform	122
11.23	How to enable software QoS	123
11.24	QoS information.....	128
11.25	How to enable USB Ethernet (example for ASIX AX88XXX)	134
11.26	How to build a single image for the RT2880 8M flash platform.....	135
11.27	How to start a printer server (example for HP officejet 4355).....	136
11.28	How to force the RT3052 link speed	139
11.29	How to verify IGMP snooping function	140
11.30	EHCI/OHCI USB Power Saving	142
11.31	Auto-frequency and Power Saving	143
11.32	Concurrent AP porting Guide	148
11.33	SuperDMZ usage guide	156
11.34	How to support IPv6 Ready Logo	157
11.35	How to enable iPerf tool	161
11.36	How to enable ebtables.....	162

11.37	How to enable IPv6 Rapid Deployment (6rd).....	162
11.38	How to enable IPv6 DS-Lite.....	166
11.39	How to modify flash layout	170
11.40	How to reduce Linux FW size.....	173
11.41	How to change internal GSW PHY Base Address.....	174
11.42	How to support new USB 3G dongle.....	175
11.43	How to enable USB 3G dongle function	175
11.44	How to enable Port Trigger function	179
11.45	Port Trigger information.....	180
11.46	How to enable I2S+Codec function	181
11.47	How to enable SPDIF function	187
11.48	How to enable Dual Image support	192
11.49	How to enable NFC support	198
11.50	How to enable AES Engine.....	203

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 Marvell 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: Intergrated 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 STA version 1.4.0.0	Vitesse Switch Support	Formal: 2007/11/08

	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:	RT2880 platforms	Formal: 2008/10/06

	RT2860 AP driver v2.0.0.0 RT2860 STA driver v1.8.0.0 GreenAP support Busybox 1.12.1 MTD-Based Flash API	RT3050 platforms RT3052 platforms	
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: Ralink Flow Classifier Linux-based Watchdog driver More 3G data card support Video Flow Classification Command User space watchdog daemon	RT2880 platforms RT3050 platforms RT3052 platforms RT3883 platforms RT3662 platforms	Formal: 2010/02/12
3.5 SDK	Feature parity with 3.4 SDK plus: support NAND/SPI/NOR in the same firmware support Hardware NAT on RT3052/RT3883/RT3352 support Software QoS super dmz support support kernel mode pptp/l2tp to improve throughput significantly	RT2880 platforms RT3050 platforms RT3052 platforms RT3883 platforms RT3662 platforms RT3352 platforms RT5350 platforms	Formal: 2010/08/06
3.6 SDK	Feature parity with 3.5 SDK plus: Support IPv6 Ready logo Support IPv6 MLD multicast proxy/snooping Support skb recycling mechanism Support switch packet count debug Support phy register dump Supprot user and kernel mode watchdog module Support kernel mode nvram Support iPerf Support ebttables	RT2880 platforms RT3050 platforms RT3052 platforms RT3883 platforms RT3662 platforms RT3352 platforms RT5350 platforms	Formal: 2011/07/15
4.0 SDK	Feature parity with 3.6 SDK	RT2880 platforms	Formal: 2012/02/22

	support IPv6 Rapid Deployment support IPv6 DS-Lite support two giga phy port display AP Client site_survey	RT3050 platforms RT3052 platforms RT3883 platforms RT3662 platforms RT3352 platforms RT5350 platforms RT6855 platforms RT6856 platforms	
4.1 SDK	Feature parity with 4.0 SDK plus:	RT2880 platforms RT3050 platforms RT3052 platforms RT3883 platforms RT3662 platforms RT3352 platforms RT5350 platforms RT6855 platforms RT6856 platforms MT7620 platforms	
4.2 SDK	Feature parity with 4.1 SDK plus: GCC 4.6.3 compiler. uClibc 0.9.33 Samba 3.6.6 Support software QoS for linux 2.6.36 Add SFQ schedule for SW QoS Shrink memory requirement Support ethtool for linux 2.6.36 Support PPTP/L2TP accelerator Support lighttpd web server (BSD licensed) Support port trigger Support NFC MT6605	RT2880 platforms RT3050 platforms RT3052 platforms RT3883 platforms RT3662 platforms RT3352 platforms RT5350 platforms RT6855 platforms RT6856 platforms MT7620 platforms MT7621 platforms	Formal:2013/10/31
4.3 SDK	Feature parity with 4.2 SDK plus:	MT7628 platforms	Alpha: 2014/05/09

2 VERSION HISTORY

Release	Features	Date	Author
1.2	Initial release		Steven Liu
1.3	WebUI – NTP/DDNS, iNIC I2C, SPI, GPIO Linux driver		Steven Liu
2.0	Squashfs tools installation WebUI - save/restore configure. WPS , factory default WebUI – STA, Ethernet Converter mode		Steven Liu
2.2	WebUI - Operation Mode reorganization How to downsize image		Steven Liu
2.3	How to control GPIO and LED Install mksquashfs Utility Describes Uboot configuration file Add new parameter in default setting		Steven Liu
2.4	WebUI – How to save the configurations to the flash		Winfred Lu
3.0	Updated for RT3052 Chapter Re-organization		Steven Liu
3.1	Update default parameter for LED firmware Update GPIO definition for RT3052 platform Update FAQ		Steven Liu
3.2	Reorganize user manual 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		Steven Liu / Winfred
3.3	Update FAQ -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		Steven
3.4	- How to burn SPI Uboot firmware -How to enable new watchdog -How to verify IGMP snooping		Steven
3.5	- Update "How to enable Software QoS"		YY

3.6	- Update "NVRAM" - Update "How to enable watchdog" - EHCI/OHCI USB Power Saving - Auto-frequency and Power Saving - Concurrent AP porting Guide - SuperDMZ usage guide - How to support IPv6 Ready Logo - How to enable iPerf tool - How to enable ebtables	Red
4.0	- Update concurrent AP porting Guide - How to enable 6RD - How to enable DS-Lite	Roger/Steven/Red
4.1	- Update APSoC chip support	Red
4.2	- Update MT7621 Parts	Steven
4.3	- Update MT7628 Parts	Red

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

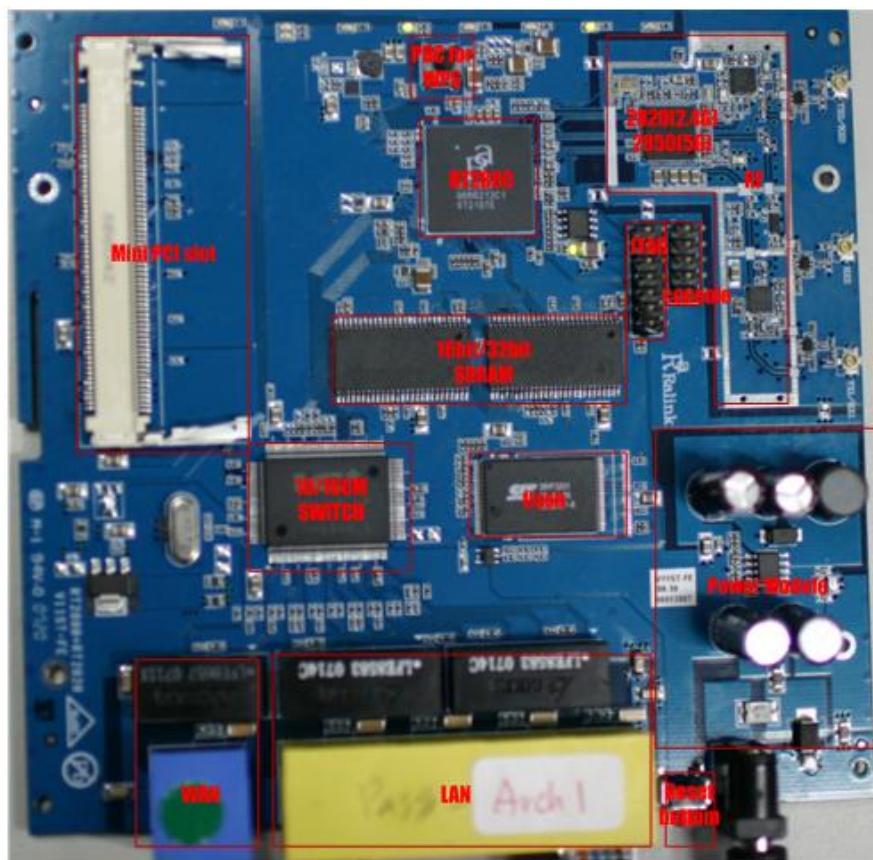


Table 1 RT2880 Memory Mapping

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

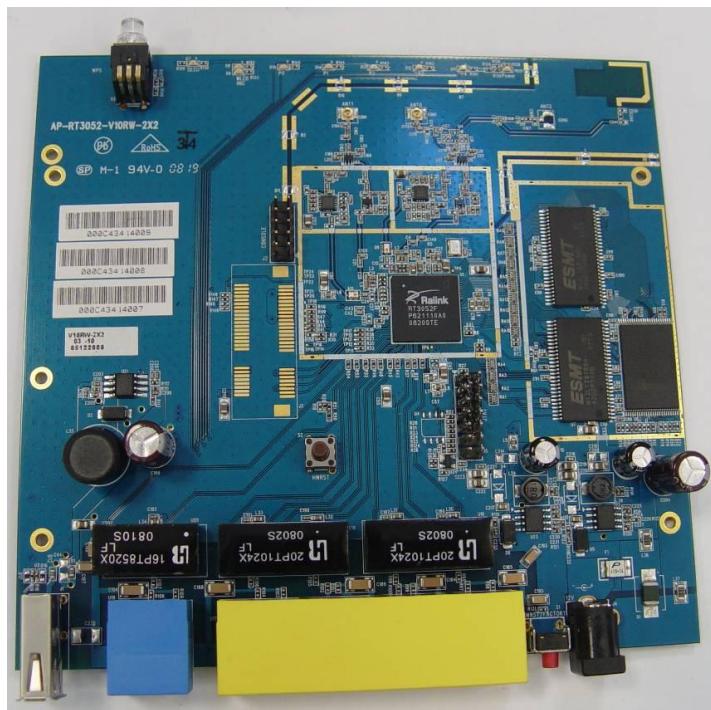


Table 2 RT3052 Memory Mapping

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 RT3883 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

3.4 RT3352

The RT3352 SOC combines Ralink's 802.11n draft compliant 2T2R MAC/BBP/PA/RF, a high performance 400MHz MIPS24KEc CPU core, a Gigabit Ethernet MAC, 5-pors integrated 10/100 Ethernet Swtich/PHY and an USB Host/Device. With the RT3352, there are very few external components required for 2.4GHz 11n wireless products. The RT3352 employs Ralink 2nd generation 11n technologies for longer range and better throughput. The embedded high performance CPU can process advanced applications effortlessly, such as WIFI data processing without overloading the host processor. In addition, the RT3352 has rich hardware interfaces (SPI/ I2S/ I2C/ PCM/ UART/ USB/ GMAC) to enable many possible applications.

Figure 4 The RT3352 Demo Board



Table 4 RT3352 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	<<Reserved>>
1000.0800	-	1000.08FF	256	<<Reserved>>
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	256	MIPS CNT
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	<<Reserved>>
1000.3800	-	1000.3FFF	2 K	<<Reserved>>
1000.4000	-	100F.FFFF		<<Reserved>>
1010.0000	-	1010.FFFF	64 K	Frame Engine
1011.0000	-	1011.7FFF	32 K	Ethernet Swtich
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	<<Reserved>>
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	-	1C00.3FFF	16KB ROM	When system is power on, 16KB internal boot ROM is mapped.

3.5 RT5350

The RT5350 SOC combines Ralink's 802.11n draft compliant 1T1R MAC/BBP/PA/RF, a high performance 360MHz MIPS24KEc CPU core, 5-ports integrated 10/100 Ethernet Switch/PHY and an USB Host/Device. With the RT5350, there are very few external components required for 2.4GHz 11n wireless products. The RT5350 employs Ralink 2nd generation 11n technologies for longer range and better throughput. The embedded high performance CPU can process advanced applications effortlessly, such as WIFI data processing without overloading the host processor. In addition, the RT5350 has rich hardware interfaces (SPI/ I2S/ I2C/ PCM/ UART/ USB) to enable many possible applications.

Figure 5 The RT5350 Demo Board

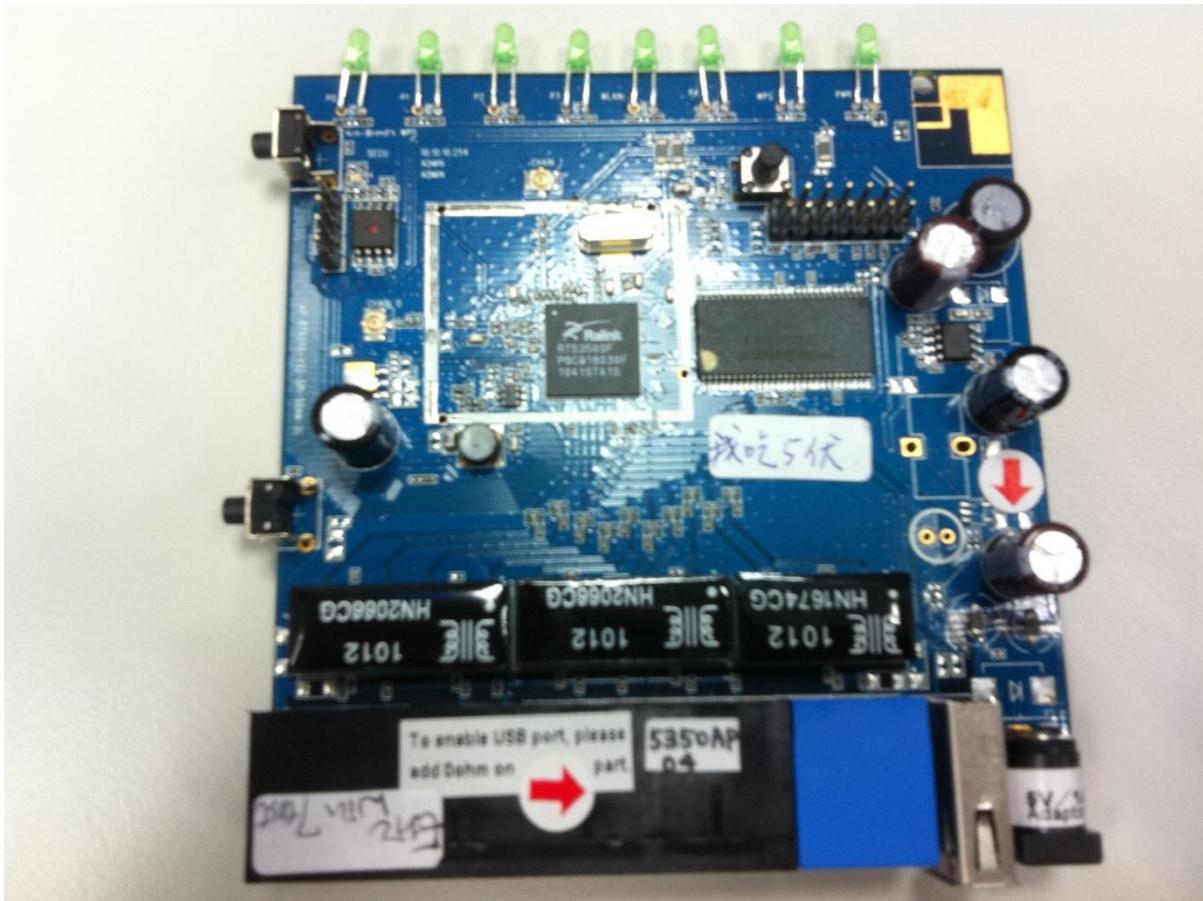


Table 5 RT5350 Memory Mapping

Start		End	Size	Description
0000.0000	-	03FF.FFFF	64 M	SDRAM 64MB
0400.0000	-	0FFF.FFFF	192M	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 (SDR)
1000.0400	-	1000.04FF	256	<<Reserved>>
1000.0500	-	1000.05FF	256	UART
1000.0600	-	1000.06FF	256	PTO
1000.0700	-	1000.07FF	256	Reserved>>
1000.0800	-	1000.08FF	256	Reserved>>
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	256	MIPS CNT
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	Reserved>>
1000.3800	-	1000.3FFF	2 K	Reserved>>
1000.4000	-	100F.FFFF		<<Reserved>>
1010.0000	-	1010.FFFF	64 K	Frame Engine
1011.0000	-	1011.7FFF	32 K	Ethernet Swtich
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	Reserved>>
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	-	1C00.3FFF	16KB ROM	When system is power on, 16KB internal boot ROM is mapped.

3.6 RT6855

Best in Class Network Processors for 802.11n AP/Router

High performance yet cost-effective network processor, that enable scalable Wi-Fi AP/Router designs when combined with Ralink 1x1, 2x2, 3x3 802.11n and 802.11ac wireless chips.

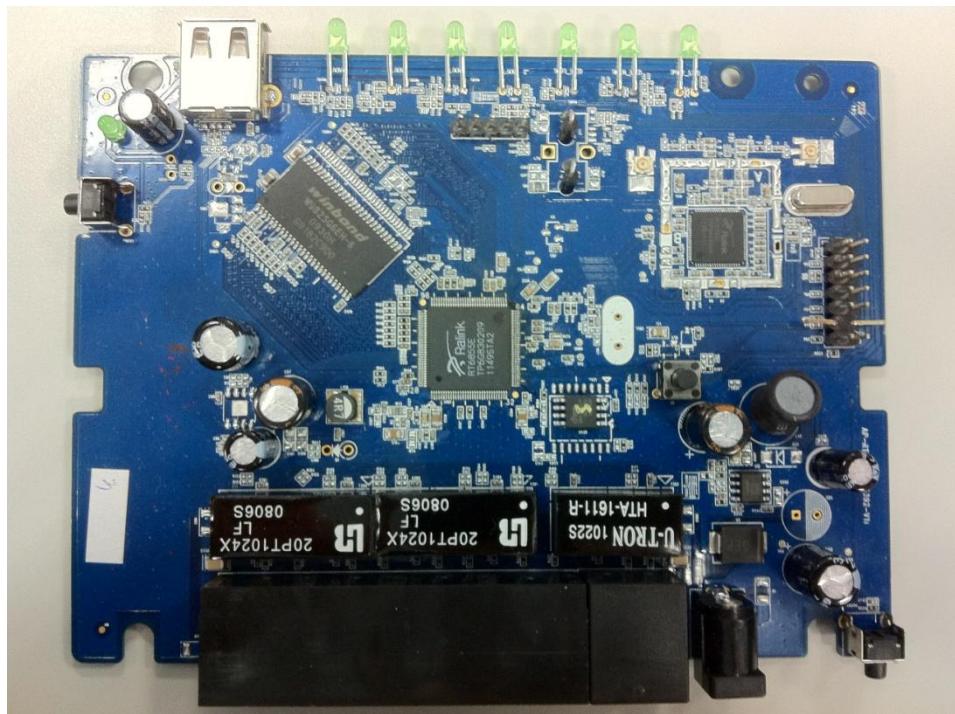
Integrated with a 32-bit MIPS 34Kc CPU, a 5-port 10/100 switch, PCI express port, USB port interface

iNIC (Intelligent NIC) design that provides an easy and ideal solution to add high performance 802.11n/ 802.11ac to any embedded platforms.

Overview:

The RT6855 single chip network processor series contains, an 32-bit MIPS ® 34Kc™ CPU core, a 5-port 10/100 Ethernet switch and a rich array of interfaces to enable interoperability with many possible applications, such as dual PCI express port to connect to 802.11n wireless chip, USB 2.0 port for network storage, 3/4G connectivity, and SPI Flash memory interface to support large bandwidth applications through the AP/router.

Figure 6 The RT6855 Demo Board



3.7 RT6856

Best in Class Network Processors for High Performance 802.11n AP/Router

High performance yet cost-effective network processor, that enable scalable Wi-Fi AP/Router designs when combined with Ralink 1x1, 2x2, 3x3 802.11n and 802.11ac wireless chips.

Integrated with a 32-bit MIPS 34Kc CPU, a 5-port 10/100 switch, dual PCI express ports, USB ports interface

iNIC (Intelligent NIC) design that provides an easy and ideal solution to add high performance 802.11n/ 802.11ac to any embedded platforms.

Overview:

The RT6855 single chip network processor series contains , an 32-bit MIPS ® 34Kc™ CPU core, a 5-port 10/100 Ethernet switch and a rich array of interfaces to enable interoperability with many possible applications, such as dual PCI express port to connect to 802.11n wireless chip, USB 2.0 port for network storage, 3/4G connectivity and printing, PCM interface for analog and VoIP telephony, and an I2S interface for audio streaming, and dual SPI Flash memory interface to support large bandwidth applications through the AP/router.

Figure 7 The RT6856 Demo Board

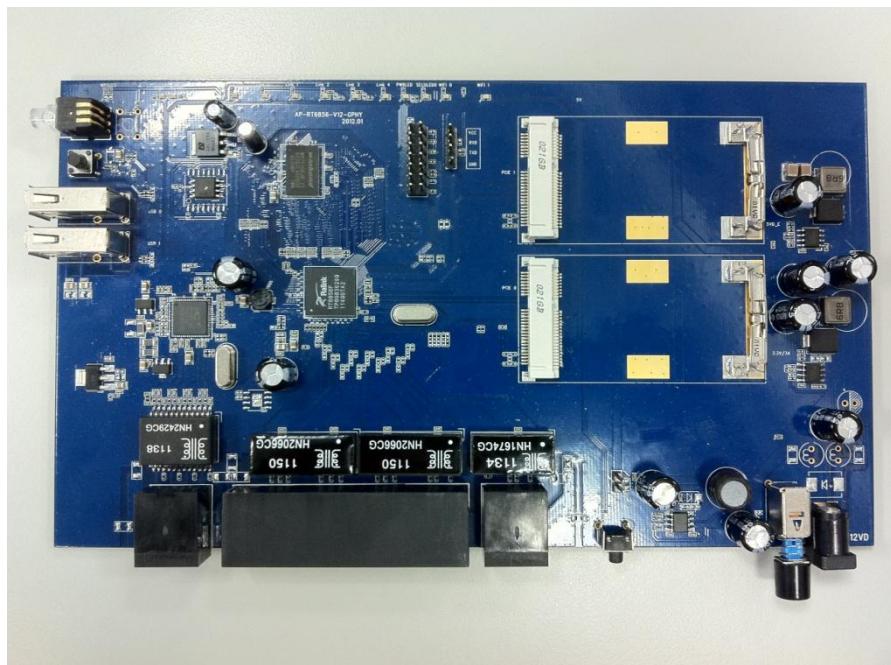


Table 6 RT6856 Memory Mapping

Module	Physical Memory Map
PCIe RC IO	0x1F60_0000 ~ 0x1F60_FFFF
PCIe RC Memory	0x1F70_0000 ~ 0x1F8F_FFFF 0x2000_0000 ~ 0x2FFF_FFFF
System control unit	0x1FB0_0000 ~ 0x1FB0_FFFF
SMC	0x1FB1_0000 ~ 0x1FB1_FFFF
DMC	0x1FB2_0000 ~ 0x1FB2_FFFF
GDMA	0x1FB3_0000 ~ 0x1FB3_FFFF
Interrupt controller	0x1FB4_0000 ~ 0x1FB4_FFFF
Frame Engine	0x1FB5_0000 ~ 0x1FB5_7FFF
Switch	0x1FB5_8000 ~ 0x1FB5_FFFF
ATM SAR	0x1FB6_0000 ~ 0x1FB6_FFFF
Crypto Engine	0x1FB7_0000 ~ 0x1FB7_FFFF
PCIe RC configuration address	0x1FB8_0020
PCIe RC configuration data	0x1FB8_0024
SPI master controller	0x1FBC_0000 ~ 0x1FBC_FFFF
PCM	0x1FBD_0000 ~ 0x1FBD_FFFF
NFC	0x1FBE_0000 ~ 0x1FBE_FFFF
UART	0x1FBF_0000 ~ 0x1FBF_00FF
Timers	0x1FBF_0100 ~ 0x1FBF_01FF
GPIO	0x1FBF_0200 ~ 0x1FBF_02FF
UART2	0x1FBF_0300 ~ 0x1FBF_03FF

3.8 MT7620

The MT7620 router-on-a-chip includes an 802.11n MAC and baseband, a 2.4 GHz radio and FEM, a 580 MHz MIPS® 24K™ CPU core, a 5-port 10/100 switch and two RGMII. The MT7620 includes everything needed to build an AP router from a single chip. The embedded high performance CPU can process advanced applications effortlessly, such as routing, security and VoIP. The MT7620 also includes a selection of interfaces to support a variety of applications, such as a USB port for accessing external storage.

The following table covers the main features offered by the MT7620N and MT7620A. Overall, the MT7620N supports the requirements of an entry-level AP/router, while the more advanced MT7620A supports a number of interfaces together with a large maximum RAM capacity.

Features	MT7620N	MT7620A
CPU	MIPS24KEc (580 MHz)	MIPS24KEc (580 MHz)
Total DMIPs	580 x 1.6 DMIPs	580 x 1.6 DMIPs
I-Cache, D-Cache	64 KB, 32 KB	64 KB, 32 KB
L2 Cache	n/a	n/a
HNAT/HQoS	HNAT	HNAT 2 Gbps forwarding
Memory		
DRAM Controller	16 b	16 b
SDRAM	512 Mb, 120 MHz	512 Mb, 120 MHz
DDR1	512 Mb, 193 MHz	1 Gb, 193 MHz
DDR2	n/a	2 Gb, 193 MHz
NAND	n/a	Small page 512Byte (max 512M bit) Large page 2Kbyte (max 8G bit)
SPI Flash	3B addr mode (max 128Mbit) 4B addr mode (max 512Mbit)	3B addr mode (max 128Mbit) 4B addr mode (max 512Mbit)
SD	n/a	SD-HC class 10 (32GB)
RF	2T2R 802.11n 2.4 GHz	2T2R 802.11n 2.4 GHz
PCIe	n/a	1
USB 2.0	1	1
Switch	5p FE SW	5p FE SW + RGMII(1) 4p FE SW + RGMII(2)

Features	MT7620N	MT7620A
I2S	n/a	1
PCM	n/a	1
I2C	1	1
UART	1 (Lite)	2 (Lite/Full)
JTAG	1	1
Package	DRQFN148- 12 mm x 12 mm	TFBGA265- 11 mm x 11 mm

Figure 8 MT7620N Demo Board



Figure 9 MT7620A Demo Board

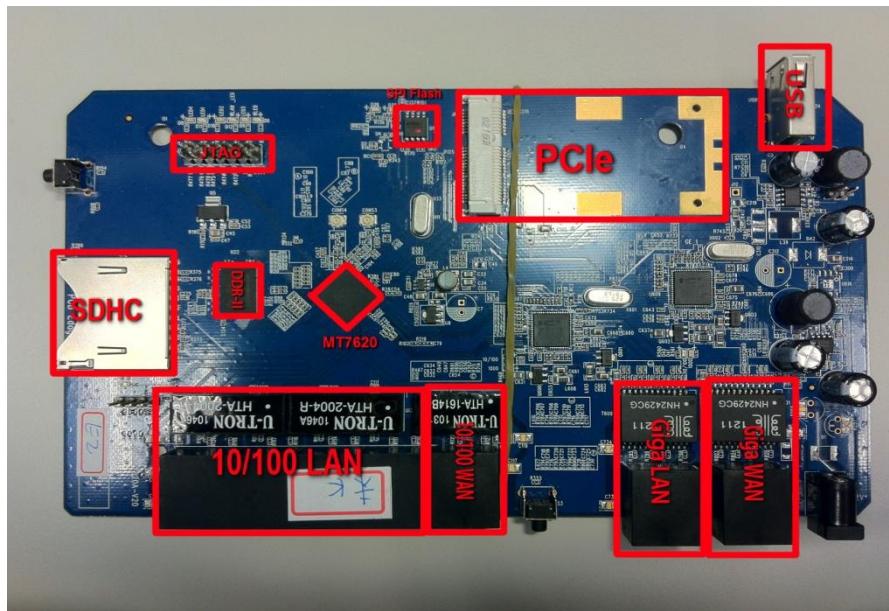


Table 7 MT7620 Memory Mapping

Start	End	Size	Description
0000.0000	-	0FFF.FFFF	256 MBytes DDR2 256 MB/ DDR1 256 MB/SDRAM 128 MB
1000.0000	-	1000.00FF	256 Bytes SYSCTL
1000.0100	-	1000.01FF	256 Bytes TIMER
1000.0200	-	1000.02FF	256 Bytes INTCTL
1000.0300	-	1000.03FF	256 Bytes MEM_CTRL (SDR/DDR)
1000.0400	-	1000.04FF	256 Bytes Rbus Matrix CTRL
1000.0500	-	1000.05FF	256 Bytes UART
1000.0600	-	1000.06FF	256 Bytes PIO
1000.0700	-	1000.07FF	256 Bytes <<Reserved>>
1000.0800	-	1000.08FF	256 Bytes NAND Controller
1000.0900	-	1000.09FF	256 Bytes I2C
1000.0A00	-	1000.0AFF	256 Bytes I2S
1000.0B00	-	1000.0BFF	256 Bytes SPI
1000.0C00	-	1000.0CFF	256 Bytes UARTLITE
1000.0D00	-	1000.0DFF	256 Bytes MIPS CNT
1000.2000	-	1000.27FF	2 KBytes PCM (up to 16 channels)
1000.2800	-	1000.2FFF	2 KBytes Generic DMA (up to 64 channels)

Start		End	Size	Description
1000.3000	-	1000.37FF	2 KBytes	<<Reserved>>
1000.3800	-	1000.3FFF	2 KBytes	<<Reserved>>
1000.4000	-	100F.FFFF		<<Reserved>>
1010.0000	-	1010.FFFF	64 KBytes	Frame Engine
1011.0000	-	1011.7FFF	32 KBytes	Ethernet Swtich
1011.8000		1011.FFFF	32 KBytes	ROM
1012.0000	-	1012.7FFF	32 KBytes	USB Device Control
1012.8000	-	1012.FFFF	32 KBytes	<<Reserved>>
1013.0000	-	1013.3FFF	16 KBytes	SDHC
1013.4000	-	1013.FFFF	48 KBytes	<<Reserved>>
1014.0000	-	1017.FFFF	256 KBytes	PCI Express
1018.0000	-	101B.FFFF	256 KBytes	WLAN BBP/MAC
101C.0000	-	101F.FFFF	256 KBytes	USB Host
1020.0000	-	1023.FFFF	256 KBytes	<<Reserved>>
1024.0000	-	1027.FFFF	256 KBytes	<<Reserved>>
1028.0000	-	1BFF.FFFF		<<Reserved>>
1C00.0000	-	1C00.7FFF	32 KB ROM	When the system is powered on, a 24 KB internal boot ROM is mapped.

3.9 MT7621

The MT7621 SoC includes a high performance 880 MHz MIPS1004Kc CPU core and high speed USB3.0/PCIe/SDXC interfaces, which is designed to enable a multitude of high performance, cost-effective IEEE 802.11n/ac applications with a MediaTek (Ralink) WiFi client card.

There are several masters (MIPS 1004KEc, USB, PCI Express, SDXC, FE) in the MT7621 SoC on a high performance, low latency Rbus, (Ralink Bus). In addition, the MT7621 SoC supports lower speed peripherals such as UART Lite, GPIO, NFI and SPI via a low speed peripheral bus (Pbus). The DDR2/DDR3 controller is the only bus slave on the Rbus. It includes an Advanced Memory Scheduler to arbitrate the requests from bus masters, enhancing the performance of memory access intensive tasks.

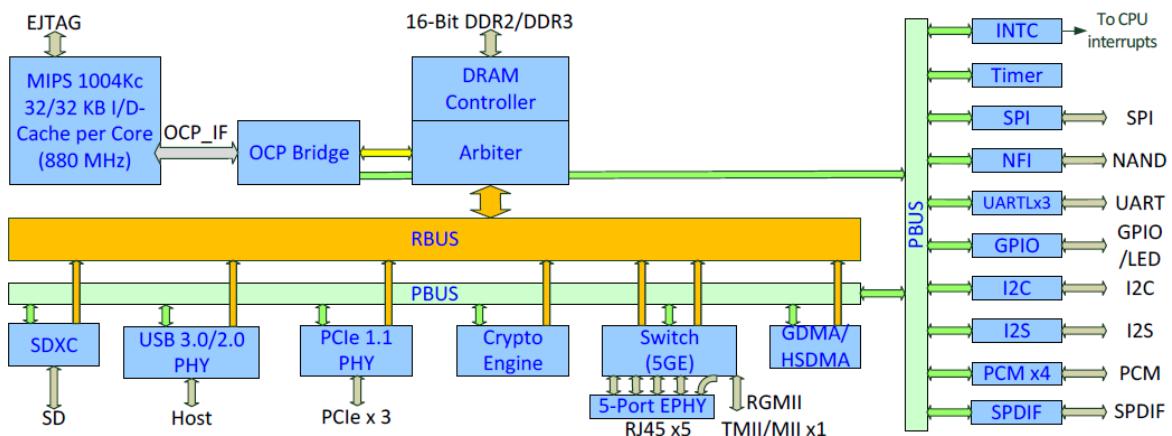


Figure 10 MT7621A Demo Board

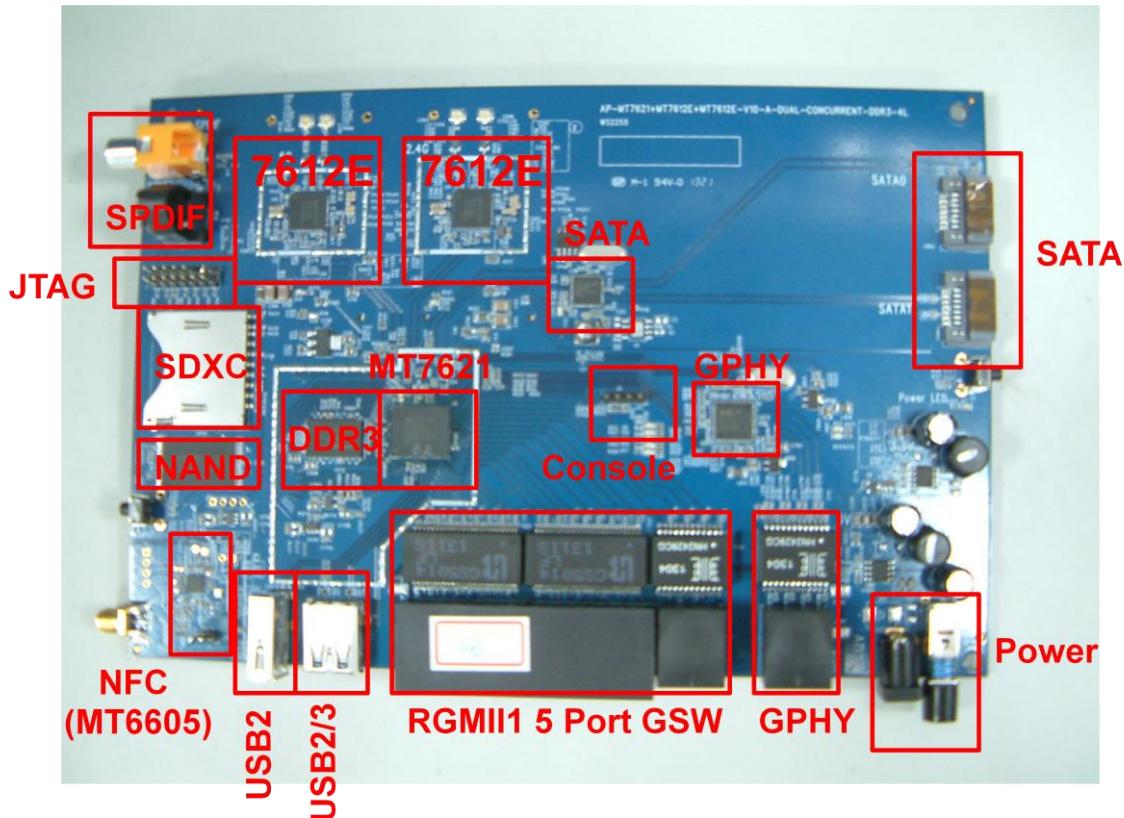


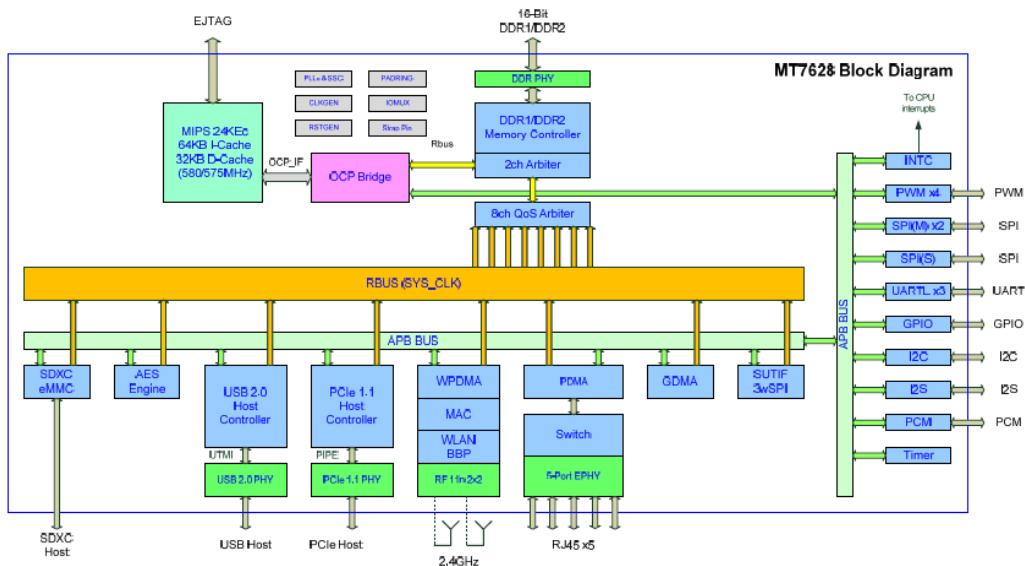
Table 8 MT7621 Memory Mapping

Start	End	Size	Description
0	1BFFFFFF	448M	DRAM Direct Map
1C000000	1DFFFFFF	32M	<<Reserved>>
1E000000	1E0000FF	256	SYSCTL
1E000100	1E0001FF	256	TIMER
1E000200	1E0002FF	256	INTCTL
1E000300	1E0003FF	256	Flash Controller (NOR/SRAM/SDRAM)
1E000400	1E0004FF	256	Rbus Matrix CTRL
1E000500	1E0005FF	256	MIPS CNT
1E000600	1E0006FF	256	GPIO
1E000700	1E0007FF	256	S/PDIF
1E000800	1E0008FF	256	DMA_CFG_ARB
1E000900	1E0009FF	256	I2C
1E000A00	1E000AFF	256	I2S
1E000B00	1E000BFF	256	SPI CSR
1E000C00	1E000CFF	256	UARTLITE 1
1E000D00	1E000DFF	256	UARTLITE 2
1E000E00	1E000EFF	256	UARTLITE 3
1E000F00	1E000FFF	256	ANACTL
1E001000	1E0017FF	2K	<<Reserved>>
1E001800	1E001FFF	2K	<<Reserved>>
1E002000	1E0027FF	2K	PCM (up to 16 channel)
1E002800	1E002FFF	2K	Generic DMA (up to 64 channel)
1E003000	1E0037FF	2K	NAND Controller *(actually 1K in Module)
1E003800	1E003FFF	2K	NAND_ECC Controller *(actually 3K in module)
1E004000	1E004FFF	4K	Crypto Engine
1E005000	1E005FFF	4K	MEM_CTRL (DDRII/DDRIII)
1E006000	1E006FFF	4K	EXT_MC_ARB
1E007000	1E007FFF	4K	HS DMA
1E008000	1E00FFFF	32K	<<Reserved>>
1E010000	1E0FFFFF	960K	<<Reserved>>
1E100000	1E10DFFF	56K	Frame Engine (FE SRAM: 0x1E108000~0x1E10DFFF)
1E10E000	1E10FFFF	8K	PCIe SRAM
1E110000	1E117FFF	32K	Ethernet GMAC
1E118000	1E11FFFF	32K	ROM
1E120000	1E12FFFF	64K	<<Reserved>>
1E130000	1E137FFF	32K	SDXC
1E138000	1E13FFFF	32K	<<Reserved>>
1E140000	1E17FFFF	256K	PCI Express
1E180000	1E1BFFFF	256K	<<Reserved>>
1E1C0000	1E1FFFFF	256K	USB Host (U2+U3)
1E200000	1E23FFFF	256K	<<Reserved>>
1E240000	1E24FFFF	64K	<<Reserved>>

1E250000	1E7FFFFF	5824K	<<Reserved>>
1E800000	1EBFFFFFF	4M	PCIE Direct Access for iNIC
1EC00000	1FBBFxFFFF	16128K	<<Reserved>>
1FBC0000	1FBDFxFFFF	128	CM_GIC
1FBE0000	1FBEEFFFF	64K	<<Reserved>>
1FBF0000	1FBF7FFFF	32K	CM_CPC
1FBF8000	1FBFFFFF	32K	CM_GCR
1FC00000	1FFFFFFF	4M	ROM/SPI FLASH Direct Access
20000000	23FFFFFF	64M	DRAM Re-Map
24000000	5FFFFFFF	960M	<<Reserved>>
60000000	6FFFFFFF	256M	PCIE Direct Access
70000000	7FFFFFFF	256M	<<Reserved>>

3.10 MT7628

The MT7628 SoC includes a high performance 580/575 MHz MIPS24KEc CPU core and high speed USB2.0/PCIe interfaces, which is designed to enable a multitude of high performance, cost-effective IEEE 802.11n applications with a MediaTek WiFi client card.



There are several masters (MIPS 24KEc, USB, PCI Express, SDXC, FE) in the MT7628 SoC on a high performance, low latency Rbus. In addition, the MT7628 SoC supports lower speed peripherals such as UART Lite, GPIO, I2C and SPI via a low speed peripheral bus (Pbus). The DDR/DDR2 controller is the only bus slave on the Rbus. It includes an Advanced Memory Scheduler to arbitrate the requests from bus masters, enhancing the performance of memory access intensive tasks.

Figure 11. The MT7628 Demo Board

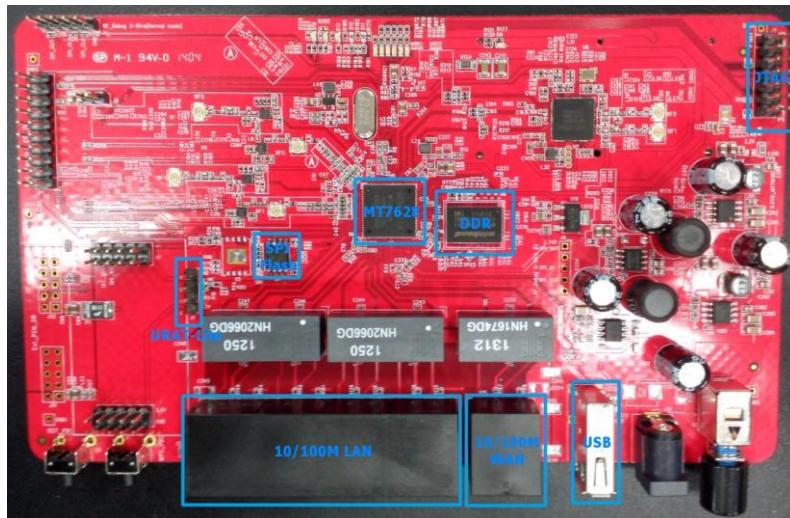


Table 9. MT7628 Memory Mapping

Start		End	Size	Description
0000.0000	-	0FFF.FFFF	256 MBytes	DDR256MB
1000.0000	-	1000.00FF	256 Bytes	SYSCTL
1000.0100	-	1000.01FF	256 Bytes	TIMER
1000.0200	-	1000.02FF	256 Bytes	INTCTL
1000.0300	-	1000.03FF	256 Bytes	EXT_MC_ARB(DDR/DDRII)
1000.0400	-	1000.04FF	256 Bytes	Rbus Matrix CTRL
1000.0500	-	1000.05FF	256 Bytes	MIPS CNT
1000.0600	-	1000.06FF	256 Bytes	GPIO
1000.0700	-	1000.07FF	256 Bytes	SPI Slave
1000.0800	-	1000.08FF		<>Reserved>>
1000.0900	-	1000.09FF	256 Bytes	I2C
1000.0A00	-	1000.0AFF	256 Bytes	I2S
1000.0B00	-	1000.0BFF	256 Bytes	SPIMaster
1000.0C00	-	1000.0CFF	256 Bytes	UARTLITE1
1000.0D00	-	1000.0DFF	256Bytes	UARTLITE 2
1000.0E00	-	1000.0EFF	256Bytes	UARTLITE 3
1000.0F00	-	1000.0FFF		<>Reserved>>
1000.1000	-	1000.17FF	2KBytes	RGCTL
1000.1800	-	1000.1FFF		<>Reserved>>
1000.2000	-	1000.27FF	2 KBytes	PCM (up to 16 channels)
1000.2800	-	1000.2FFF	2 KBytes	Generic DMA (up to 16channels)
1000.3000	-	1000.3FFF		<>Reserved>>
1000.4000	-	1000.4FFF	4KBytes	AES Engine
1000.5000	-	1000.5FFF	4 Kbytes	PWM
1000.6000	-	100F.FFFF		<>Reserved>>
1010.0000	-	1010.FFFF	64 Kbytes	Frame Engine
1011.0000	-	1011.7FFF	32KBytes	Ethernet Switch
1011.8000	-	1011.FFFF		<>Reserved>>
1012.0000	-	1012.7FFF	32 KBytes	USB PHY
1012.8000	-	1012.FFFF		<>Reserved>>
1013.0000	-	1013.7FFF	32 KBytes	SDXC / eMMC
1013.8000	-	1013.FFFF		<>Reserved>>
1014.0000	-	1017.FFFF	256KBytes	PCI Express

Start		End	Size	Description
1018.0000	-	101B.FFFF		<<Reserved>>
101C.0000	-	101F.FFFF	256 KBytes	USB Host Controller
1020.0000	-	102F.FFFF	1 MBytes	<<Reserved>>
1030.0000	-	103F.FFFF	1 MBytes	WLAN MAC/BBP
1040.0000	-	1BFF.FFFF		<<Reserved>>
1C00.0000	-	1C3F.FFFF	4 MBytes	SPI Flash Direct Access
1C40.0000	-	1FFF.FFFF		<<Reserved>>
2000.0000	-	2FFF.FFFF	256 MBytes	PCIE Direct Access
3000.9999	-	3FFF.FFFF		<<Reserved>>

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.
 - doc : User manual and useful documents.
 - 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/RT3883/RT3352/RT3883
 - linux-2.6.36MT.x: Linux kernel source for RT6855/RT6856
 - linux-2.6.36.x : Linux kernel source for MT7620/MT7621/MT7628
 - linux-3.10.14.x : Linux kernel source for MT7628
 - rootfs : 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

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

For RT series/ MT7620,

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

For MT7621/MT7628,

- Decompress buildroot-gcc463_32bits.tar.bz2 (for 32/64 bits Host Linux) or buildroot-gcc463_64bits.tar.bz2(for 64 bits Host Linux only) to /opt in order to build your own Linux image.

2. Decompress mips-2012.03.tar.bz2 to /opt (for 32/64 bits Host Linux) in order to build your own Uboot image.

5.2 Install LZMA Utility

Izma is necessary to make the compressed kernel image. The Ralink RT288x SDK uses Izma to compress the kernel image.

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

Use gzip or Izma 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.

Notes: lzma algorithm is not backward-compatible, so you have to use this lzma-4.32.0beta3 tool to compress your kernel image otherwise you will encounter lzma decompression issue.

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-gcc342/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 to /opt/buildroot-gcc342/bin/mksquashfs_lzma-3.2 & lzma_alone  
to /opt/buildroot-gcc342/bin/)
```

Linux-2.6.36.x/Linux-3.10.14.x Kernel Version

```
#cd RT288x_SDK/toolchain/mksquash_lzma-3.2  
#make  
#make install (copy lzma_alone to /opt/buildroot-gcc463/usr/bin/)  
  
#tar jxf squashfs4.2.tar.bz2  
#cd squashfs4.2/squashfs-tools$  
#make  
#cp mksquashfs /opt/buildroot-gcc342/bin/mksquashfs_lzma-4.2
```

LZMA_ALONE IS NECESSARY TO MAKE YOUR OWN RAMDISK IMAGE, IF YOU TURN ON
“COMPRESS RAMDISK BY LZMA” ON LINUX 2.4/2.6.21/2.6.36/3.10.14 KERNEL.

Linux-2.4.x /Linux-2.6.21.x Kernel Version

```
#make menuconfig  
Kernel/Library/Defaults Selection --->  
Machine selection --->  
[*] Compress ramdisk by lzma instead of gzip
```

Linux-2.6.36.x/Linux-3.10.14.x Kernel Version

```
#make menuconfig  
Kernel/Library/Defaults Selection --->  
General setup --->  
[*] Support initial ramdisks compressed using LZMA
```

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

1.1 DRAM Component (MT7621):

For reference board, please choose 512Mb for DDR2 , 1024Mb for DDR3.

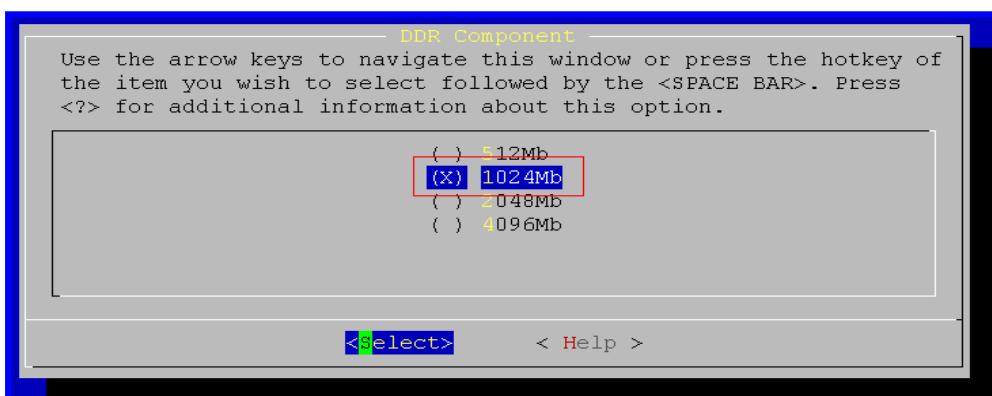
For DDR2 1066Mhz component or other dram size, please follow the below table:

	DRAM Speed	DDR AC Timing	
DDR2	800Mz	512Mb	No need to choose
		1024Mb	DDR2_Default(1Gb)
		W971GG6KB25 (1024Mb)	DDR2_W971GG6KB25(1Gb)
DDR2-1066	1066Mz	W9751G6KB(512Mb)	DDR2_1066_W9751G6KB(512Mb)
		W971GG6KB18(1024Mb)	DDR2_1066_W971GG6KB18(1Gb)
DDR3	1200Mz	1024Mb	No need to choose
		2048Mb	DDR3_Default(2Gb)
		4096Mb	DDR3_Default(4Gb)

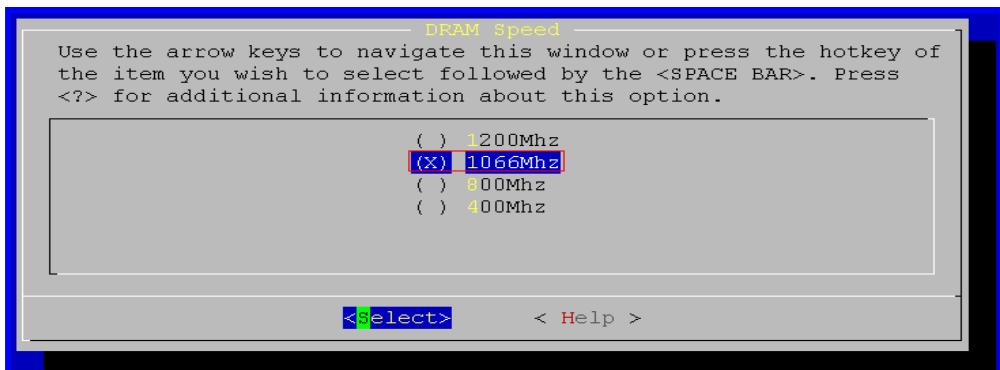
For example, choose DDR2-1066 1Gb (W971GG6KB18) :

Step 1 : Choose DRAM component (size) to 1024Mb (1Gb)

```
Cross Compiler Path: "/opt/mips-2012.03/bin/"
---
(ASIC) Chip Type
(MT7621) Chip ID
(GMAC1) Use GE1 or GE2
(GE_RGMII_FORCE_1000) GE1 connected to
(NAND) Flash Type
(1024Mb) DDR Component
(1200Mhz) DRAM Speed
(880Mhz) CPU Frequency
---
[ ] Dual Image
[ ] Dual Core Support
[ ] Partition LAN/WAN
[ ] DDR ACTiming Setting
---
Load an Alternate Configuration File
Save Configuration to an Alternate File
```



Step 2 : Choose DRAM speed (1066Mhz) :



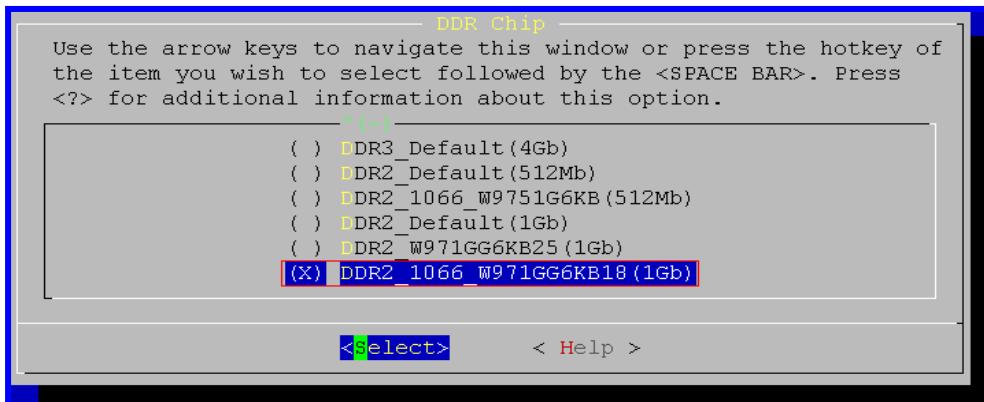
Step 3 : Choose DDR AC Timing Setting:

```

    Cross Compiler Path: "/opt/mips-2012.03/bin/"
---
(ASIC) Chip Type
(MT7621) Chip ID
(GMAC1) Use GE1 or GE2
(GE_RGMII_FORCE_1000) GE1 connected to
(NAND) Flash Type
(1024Mb) DR Component
(1066Mhz) DRAM Speed
(880Mhz) CPU Frequency
---
[ ] Dual Image
[ ] Dual Core Support
[ ] Partition LAN/WAN
 DDR ACTiming Setting
---
Load an Alternate Configuration File
Save Configuration to an Alternate File

```

Step 4 : Select DDR AC Timing Setting by different DDR component (W971GG6KB18):



1.2 DRAM Component (RT series, MT7620 and MT7628):

	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
```

```
.....
```

1. RT2880/RT3052/RT3883/RT3352/RT5350:
 - NOR Flash: **uboot.bin** is located in Uboot/.
 - SPI Flash: **uboot.img** is located in Uboot/
 - NAND Flash: **uboot.img** is located in Uboot/
2. RT6855/RT6856/MT7620/MT7621:
 - SPI Flash: **uboot.bin** is located in Uboot/.
 - NAND Flash: **uboot.img** is located in Uboot/

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.
- 7: Load Boot Loader code then write to Flash via Serial.
- 9: Load Boot Loader code then write to Flash 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 (10.10.10.3) ==:

Input Uboot filename (uboot.bin) ==:

7 USER LIBRARY

7.1 Library Configuration

RT288x_SDK uses uClibc 0.9.28/0.9.33.2 for user applications. The subsequent instructions show how to change the default library setting.

RT series/MT7620 use uClibc 0.9.28

MT7621/MT7628 use uClibc 0.9.33.2

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

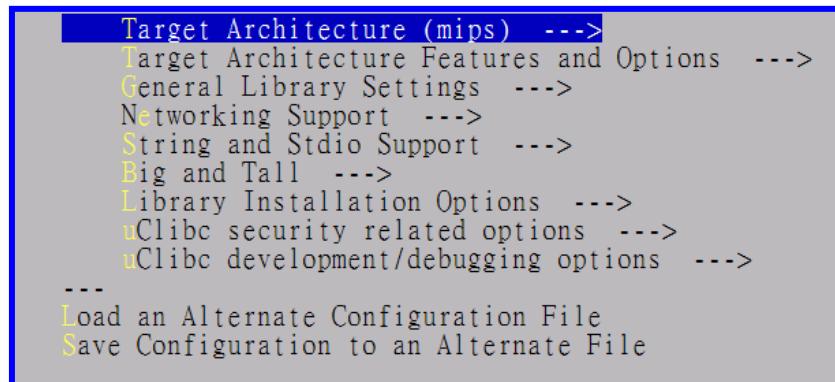


Figure 8.1 uClib 0.9.28 configurations Menu

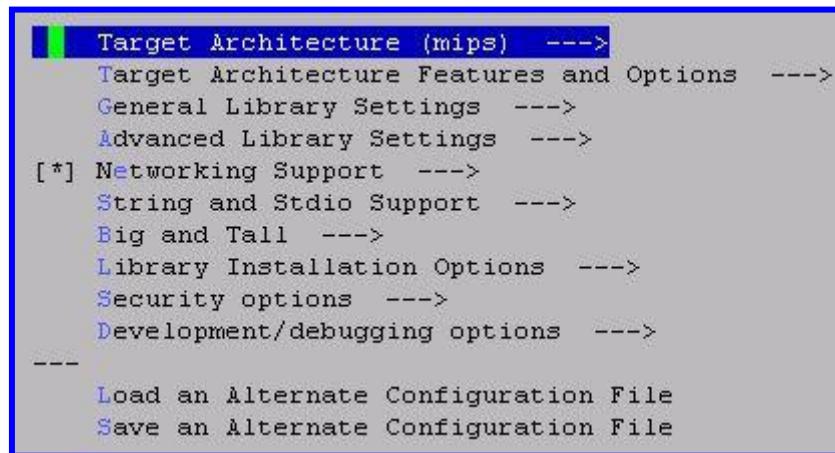


Figure 8.2 uClib 0.9.33.2 configurations Menu

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.

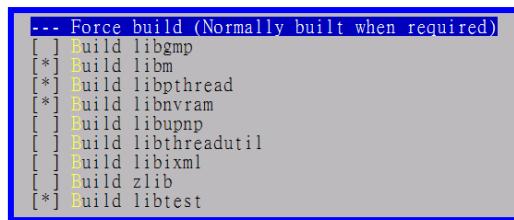


Figure 9 User Library Configure Menu

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. Except for the Ralink Proprietary Application described in the section 8.1, most applications are ported from open source for reference only. Developers can upgrade/modify/change applications by themselves for customize purpose.

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 b0000000
```

```
/#reg r 18 /*read b0000018 */
```

```
/#reg w 18 12345678 /* write 0x12345678 to b0000018 */
```

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 ETH_MAC

Description: flash read/write program to update Ethernet MAC address.

Usage:

- a. read: eth_mac r <lan|wan>
- b. write: eth_mac w <lan|wan> <MACADDR[0]> <MACADDR[1]> <MACADDR[2]> <MACADDR[3]> <MACADDR[4]> <MACADDR[5]>

Example:

- a. read: eth_mac r lan
- b. write: eth_mac w lan 00 0c 43 76 21 01

8.1.5 GPIO

Description: GPIO test program

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

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

- gpio w: write test (Note that all GPIO pins will be changed to output direction when writing)
- gpio r: read test (Note that all GPIO pins will be changed to input direction when reading)
- gpio g <gpio#>: read the target GPIO pin. (Note that the target GPIO pin will be changed to input direction when reading)

- 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.

8.1.6 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.

8.1.7 MTD

Description: MTD writing program for firmware update

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

Example: mtd_write -r write image.bin mtd4

8.1.8 NVRAM

8.1.8.1 Basic feature

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/Lighttpd 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/rtdev>] <field>
- b. set: nvram_set [<2860/rtdev>] <field>
- c. init: ralink_init <command> [<platform>] [<file>]

<Commands>:

- rt2860_nvram_show (display rt2860 values in nvram)
- rtdev_nvram_show (display rtdev 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 or the first Wi-Fi interface
- rtdev - intelligent nic or the second Wi-Fi interface

<File>: File name for renew command

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 rtdev /* 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 */

8.1.8.2 Kernel NVRAM

To avoid accessing NVRAM inconsistently, sdk also supports NVRAM in Kernel mode.

How to use:

```
$ make menuconfig
```

```
Kernel/Library/Defaults Selection -->
```

[*] Customize Kernel Settings ---> Exit--->Yes

Machine selection --->

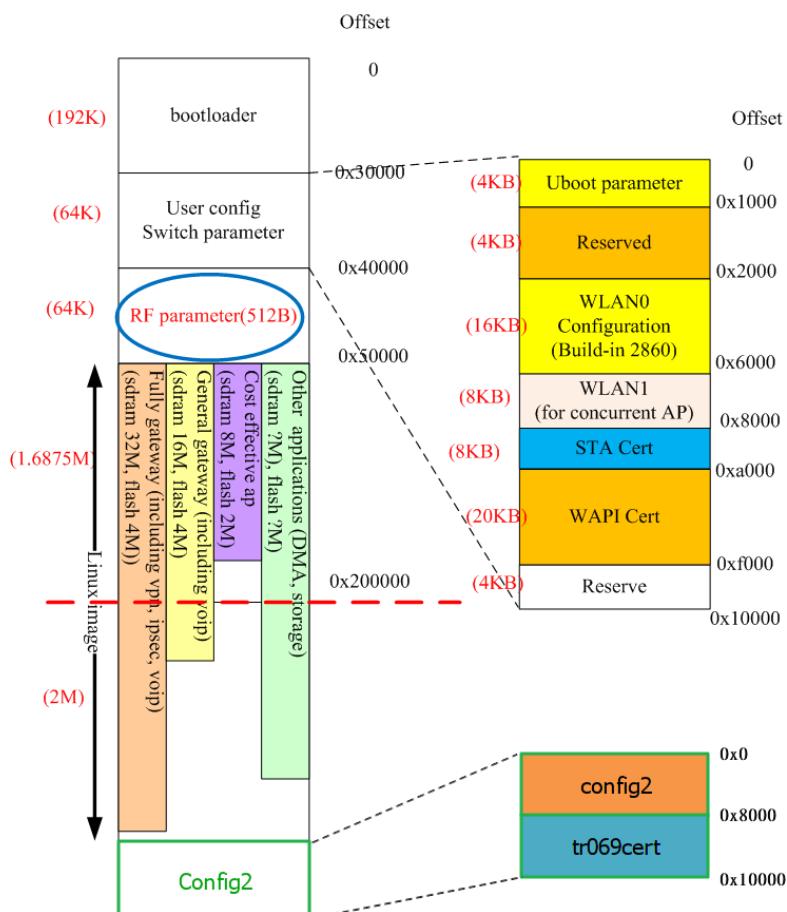
```

System type (Ralink MT7620 board) --->
Soc Hardware Type (MT7620-ASIC) --->
DRAM Size (64M) --->
Flash Type (SPI) --->
[ ] Dual Image
[*] Kernel NVRAM
  Root File System Type (RootFS_in_RAM) --->
  (16384) Default RAM disk size
  < > Ralink Hardware Timer0/Timer1
[*] Ralink CPU Sleep mode
--*- Ralink System Tick Counter
[ ] Kprofile

```

8.1.8.3 Extend NVRAM

If developer needs another flash block as 2nd nram, SDK already provided sample solution, like "Config2" block.



How to use:

```
$ make menuconfig  
  
Kernel/Library/Defaults Selection --->  
  
[*] Customize Kernel Settings ---> Exit--->Yes  
  
Machine selection --->
```

```
[ ] Dual Image  
[*] Kernel NVRAM  
[*] NVRAM Extension
```

The sample MTD partition, “Config2”, implements **MTD_CONFIG_PART_SIZE** bytes and it begins from **MTD_CONFIG2_PART_OFFSET**. Developer can configure two definitions to assign the maximum size and start address of 2nd NVRAM.

8.1.9 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

8.1.10 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

8.1.11 I2SCMD

Description: I2S Toolkit for raw playback/record via I2S Interface...

Usage: i2scmd [cmd] [srate] [vol] < playback files

Note:

- cmd = 0|1 - i2s raw playback|record
- srate = 8000|16000|32000|44100|48000 Hz playback sampling rate
- vol = -10~2 db playback volume

Example:

- i2scmd 0 48000 2 </etc_ro/test_sound.snd

8.1.12 SPDIFCMD

Description: SPDIF Toolkit for raw playback via SPDIF Interface...

Usage:

[fmt=0] [srate] [wordlen] [pathname]

[fmt=1] [srate] [rawtype] [pathname]

fmt = 0|1 - spdif pcm| raw data

srate = 22050| 24000| 32000|44100|48000|88200|96000|176400|192000 Hz sampling frequency

rawtype = for raw data (fmt = 1) -- (0: Null data; 1: AC3 data; 3: Pause)

wordlen = 16| 24 bits per sample

downsample = 1: no down sample; 2: 2x down sample; 4: 4x down sample

[fmt=2] [pathname]

Example: (for PCM data, 16 bit)

spdifcmd 0 48000 16 </etc_ro/test_sound.snd

8.1.13 Script

Description: WebUI configuration script.

Usage: Refer to the script help message.

8.2 accel-pptp

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

Description: ACCEL-PPTP is client/server solution for Linux. This project is based on POPTOP (<http://www.poptop.org>) and PPTPCLIENT (<http://pptpclient.sourceforge.net>). It uses kernel module to increase performance and decrease system usage.

8.3 bigpond

Source code: RT288x_SDK/source/user/bigpond

Description: This is a client to connect to Telstra's Big Pond Broadband powered by Cable.

8.4 bonnie++

Source code: RT288x_SDK/source/user/bonnie++

Description: Bonnie++ is a free file system benchmarking tool for Unix-like operating systems. Bonnie++ is a benchmark suite that is aimed at performing a number of simple tests of hard drive and file system performance.

8.5 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.

8.6 busybox

Source code: RT288x_SDK/source/user/busybox

Description: BusyBox combines tiny versions of many common UNIX utilities into a single small executable.

8.7 comgt-0.32

Source code: RT288x_SDK/source/user/comgt-0.32

Description: Comgt is a command line tool for controlling, configuring and interacting with Option Wireless 3G and 2G (HSDPA, UMTS, EDGE, GPRS, GSM) data devices within the Linux environment.

8.8 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.

8.9 curl

Source code: RT288x_SDK/source/user/curl

Description: curl is a command line tool for transferring data with URL syntax

8.10 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.

8.11 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.

8.12 dropbear-0.52

Source code: RT288x_SDK/source/user/dropbear-0.52

Description: Dropbear is a relatively small SSH server and client. It runs on a variety of POSIX-based platforms.

8.13 ebtables

Source code: RT288x_SDK/source/user/ebtables-v2.0.9-2

Description: The ebtables program is a filtering tool for a Linux-based bridging firewall. It enables transparent filtering of network traffic passing through a Linux bridge.

8.14 ecmh

Source code: RT288x_SDK/source/user/ecmh

Description: Easy Cast du Multi Hub (ecmh) is a networking daemon that acts as a full IPv6 MLDv1 and MLDv2 Multicast "Router".

8.15 GoAhead

Source code: RT288x_SDK/source/user/goahead

Description: The GoAhead Web Server is a simple, low end, compact web server that has been widely ported to many embedded operating systems with limited memory. Old WebUI reference design of the AP/Router Solution.

8.16 igmproxy

Source code: RT288x_SDK/source/user/igmproxy

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

8.17 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.

8.18 iperf

Source code: RT288x_SDK/source/user/iperf-2.0.5

Description: iPerf was developed by NLANR/DAST as a modern alternative for measuring maximum TCP and UDP bandwidth performance. iPerf allows the tuning of various parameters and UDP characteristics. iPerf reports bandwidth, delay jitter, datagram loss.

8.19 iproute2

Source code:

```
RT288x_SDK/source/user/iproute2-2.6.24-rc7    #for Linux-2.6.21  
RT288x_SDK/source/user/iproute2-2.6.38        #for Linux-2.6.36
```

Description: iproute2 is a collection of utilities for controlling TCP / IP networking and traffic control in Linux.

8.20 iptables

Source code:

```
RT288x_SDK/source/user/iptables           # for Linux-2.4  
RT288x_SDK/source/user/ iptables-1.4.0rc1 #for Linux-2.6.21  
RT288x_SDK/source/user/ iptables-1.4.10   #for Linux-2.6.36
```

Description: Administration tool for IPv4 packet filtering and NAT.

8.21 lighttpd

Source code: RT288x_SDK/source/user/lighttpd-1.4.20

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

8.22 linux-igd

Source code: RT288x_SDK/source/user/linux-igd

Description: It is a deamon that emulates Microsoft's Internet Connection Service (ICS). It implements the UPnP Internet Gateway Device specification (IGD) and allows UPnP aware clients, such as MSN Messenger to work properly from behind a NAT firewall.

8.23 llDt(llD2d)

Source code: RT288x_SDK/source/user/llDt

Description: The llD2d daemon allows linux systems to display in Vista's Network Map.

8.24 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.

8.25 miniupnpd-1.6

Source code: RT288x_SDK/source/user/miniupnpd-1.6

Description: New reference design to perform Wi-Fi WPS with wired external Registerar.

8.26 mkdosfs-2.11

Source code: RT288x_SDK/source/user/mkdosfs-2.11

Description: mkdosfs is used to create an MS-DOS file system under Linux on a device (usually a disk partition).

8.27 mpstat-0.1.1

Source code: RT288x_SDK/source/user/mpstat-0.1.1

Description: mpstat is intended to be a utility for Linux to monitor SMP machine's CPU statistics. It records %usr, %idl, major & minor faults, and number of interrupts that have occurred. mpstat is similar to vmstat.

8.28 mtd-utils

Source code: RT288x_SDK/source/user/mtd-utils-1.5.0

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

8.29 ntfs-3g

Source code: RT288x_SDK/source/user/ntfs-3g

Description: The NTFS-3G driver is an open source, freely available read/write NTFS driver for Linux, FreeBSD, Mac OS X, NetBSD, and Haiku. It provides safe and fast handling of the Windows XP, Windows Server 2003, Windows 2000 and Windows Vista file systems. Most POSIX file system operations are supported, and full file ownership and permission support is also coming along fast.

8.30 ntfsprogs

Source code: RT288x_SDK/source/user/ntfsprogs

Description: The Linux-NTFS project aims to bring full support for the NTFS filesystem to the Linux operating system.

8.31 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.

8.32 nvram library

Source code: RT288x_SDK/source/lib/libnvram

Description: Library for nvram_get, nvram_set and ralink_init.

8.33 openl2tp-1.6

Source code: RT288x_SDK/source/user/openl2tp-1.6

Description: OpenL2TP is a complete implementation of RFC2661 - Layer Two Tunneling Protocol Version 2, able to operate as both a server and a client.

8.34 openswan-2.6.38

Source code: RT288x_SDK/source/user/openswan-2.6.38

Description: Openswan is an IPsec implementation for Linux. It has support for most of the extensions

(RFC + IETF drafts) related to IPsec, including IKEv2, X.509 Digital Certificates, NAT Traversal, and many others.

8.35 p910nd

Source code: RT288x_SDK/source/user/p910nd-0.91

Description: p910nd is a small non-spooling printer daemon that is suitable for diskless hosts, particularly hosts that have been booted via the network, or run an embedded OS, but have a printer attached. It accepts jobs from a spooling host over a TCP connection.

8.36 pciutils-3.0.0

Source code: RT288x_SDK/source/user/pciutils-3.0.0

Description: The PCI Utilities package contains a library for portable access to PCI bus configuration registers and several utilities based on this library.

8.37 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.

8.38 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.

8.39 proftpd

Source code: RT288x_SDK/source/user/proftpd

Description: ProFTPD grew from a desire for a secure and configurable FTP server. It was inspired by a significant admiration of the Apache web server. Unlike most other Unix ftp servers, it has not been derived from the old BSD ftpd code base, but is a completely new design and implementation.

8.40 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.

8.41 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

8.42 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.

8.43 samba

Source code:

RT288x_SDK/source/user/ samba-3.0.2

RT288x_SDK/source/user/ samba-3.6.6

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.

8.44 sdparm-1.02

Source code: RT288x_SDK/source/user/sdparm-1.02

Description: This utility fetches and potentially changes SCSI device (e.g. disk) mode pages. Inquiry data including Vital Product Data (VPD) pages can also be displayed. Commands associated with starting and stopping the medium; loading and unloading the medium; and other housekeeping function may also be issued by this utility.

8.45 strace

Source code: RT288x_SDK/source/user/strace

Description: strace is a system call tracer, i.e. a debugging tool which prints out a trace of all the system calls made by another process/program.

8.46 taskset

Source code: RT288x_SDK/source/user/taskset

Description: taskset is used to set or retrieve the CPU affinity of a running process given its PID or to launch a new COMMAND with a given CPU affinity.

8.47 tcpdump

Source code: RT288x_SDK/source/user/tcpdump

Description: A powerful command-line packet analyzer.

8.48 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.

8.49 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".

8.50 uvc_stream

Source code: RT288x_SDK/source/user/uvc_stream

Description: "uvc_stream" is a command line application to stream JPEG files over an IP-based network from the webcam to a viewer like Firefox, Cambozola, Videolanclient or even to a Windows Mobile device running the TCPMP-Player

8.51 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.

8.52 wpa_supplicant-0.5.7

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

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

8.53 wsc_upnp

Source code: RT288x_SDK/source/user/WSC_UPNP

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

Required library: libupnp, pthread

8.54 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.

8.55 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)

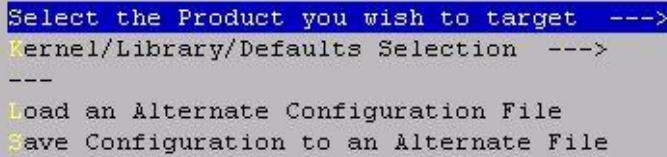
Enter 'help' for a list of built-in commands.

```
/ #
/ # hello
hello world
/ #
```

9 LINUX KERNEL

9.1 Linux configuration

```
# cd RT288x_SDK/source
# make menuconfig
```



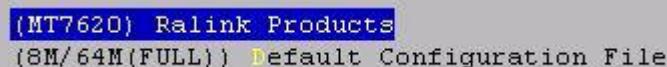
```
Select the Product you wish to target --->
Kernel/Library/Defaults Selection --->
---
Load an Alternate Configuration File
Save Configuration to an Alternate File
```

Note:

1. Choose the target platform type (RT series, MT7620, MT7621, and MT7628)
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.

Steps:

1. Use 'Select the Product you wish to target' to set the target platform (RT series, MT7620, MT7621, and MT7628).



```
(MT7620) Ralink Products
(8M/64M(FULL)) Default Configuration File
```

2. Use the 'Flash/SDRAM Size'
 - 2M/16M(AP): 2M Flash and 16M DRAM for pure AP solution (pass Vista basic logo and Wi-Fi certification b/g/n logo)
 - 4M/32M(AP+NAS): 4M Flash and 32M DRAM for complete AP/NAS solution, including USB applications)
 - 8M/64M(VOIP/TR069): 8M Flash and 64M DRAM for VOIP/TR069 solution.
 - 8M/64M(FULL): 8M Flash and 64M DRAM for full reference setting.
3. Use 'Kernel/Library/Defaults Selection' to open the configuration menu. Use 'Default all settings'.

```
--- uClibc version: 0.9.33.2
--- Cross Compiler Path: /opt/buildroot-gcc463/usr/bin
---
[*] Default all settings (lose changes)
[ ] Customize Kernel Settings (NEW)
[ ] Customize Vendor/User Settings
[ ] Customize Busybox Settings
[ ] Customize uClibc Settings
[ ] Customize uClibc++ Settings
[ ] Update Default Vendor Settings
```

4. 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/{Platform}/config to see all the default setting files.

For example, following are the default configuration files of MT7620

a. Busybox default configuration files

- ✓ 2M_16M_config.busybox-2.6.36.x
- ✓ 4M_32M_config.busybox-2.6.36.x
- ✓ 8M_32M_config.busybox-2.6.36.x //for 8M/64M(VOIP/TR069) setting
- ✓ 8M_64M_config.busybox-2.6.36.x

b. User application default configure file

- ✓ 2M_16M_config.vendor-2.6.36.x
- ✓ 4M_32M_config.vendor-2.6.36.x
- ✓ 8M_32M_config.vendor-2.6.36.x //for 8M/64M(VOIP/TR069) setting
- ✓ 8M_64M_config.vendor-2.6.36.x

c. uClibc default configure file

- ✓ 2M_16M_config.uclibc-2.6.36.x
- ✓ 4M_32M_config.uclibc-2.6.36.x
- ✓ 8M_32M_config.uclibc-2.6.36.x //for 8M/64M(VOIP/TR069) setting
- ✓ 8M_64M_config.uclibc-2.6.36.x

d. Linux kernel 2.6.36 default configure file

- ✓ 2M_16M_config.linux-2.6.36.x
- ✓ 4M_32M_config.linux-2.6.36.x
- ✓ 8M_32M_config.linux-2.6.36.x //for 8M/64M(VOIP/TR069) setting
- ✓ 8M_64M_config.linux-2.6.36.x

Each platform supports Linux kernel:

Platform	Linux Kernel
RT2880	linux-2.4.x
RT3052	linux-2.6.21.x
RT3883	linux-2.6.21.x
RT3352	linux-2.6.21.x
RT5350	linux-2.6.21.x
RT6855	linux-2.6.36MT.x
RT6856	linux-2.6.36MT.x
MT7620	linux-2.6.36.x (default)
MT7621	linux-2.6.36.x (default)
MT7628	linux-2.6.36.x (default) / linux-3.10.14.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 --->
```

- MT7620 example

```
System type (Ralink MT7620 board) --->
Soc Hardware Type (MT7620-ASIC) --->
[ ] DRAM Size (64M) --->
[ ] Flash Type (SPI) --->
[ ] Dual Image
[*] Kernel NVRAM
Root File System Type (RootFS_in_RAM) --->
(16384) Default RAM disk size
```

9.3 Change Switch Controller in RT288x SDK

The RT288x_SDK can configure the WAN/LAN partition of the internal switch in the SoC. You can use 'make menuconfig' to adjust the switch controller settings.

```
#make menuconfig
```

Kernel/Library/Defaults Selection --->

[] Customize Kernel Settings*

Ralink Module --->

```
[ ] GMAC is connected to
[*] Ralink Embedded Switch
[*] LAN/WAN Partition
Switch Board Layout Type (LLLL/W) --->
```

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.

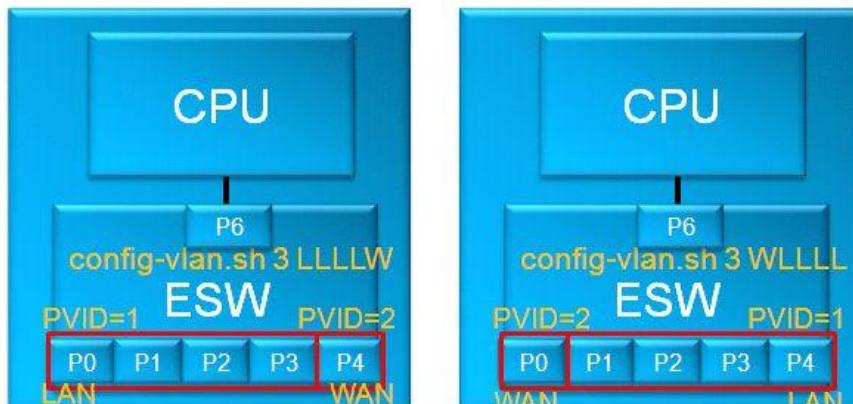


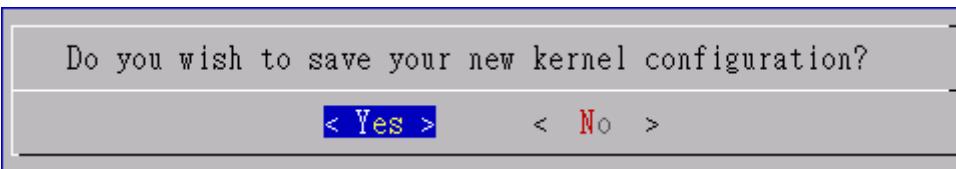
Figure 10 MT7620 10/100 Switch Operation Diagram

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/{Platform}/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.



```
Do you wish to save your new kernel configuration?
```

```
< Yes >      < No >
```

The script updates the User/Kernel default settings.

9.5 Compile Linux image with root file system

```
#make dep  
#make
```

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

- a. \${user}_ulimage - Linux image (Linux kernel+rootfs)
- b. zImage.{gz/lzma} - compressed Linux kernel+rootfs

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 MT7620 platform

1. Add the source code to the module directory

```
# mkdir RT288x_SDK/source/linux-2.6.36.x/drivers/net/hello  
#vi RT288x_SDK/source/linux-2.6.36.x/drivers/net/hello/Makefile  
  
obj-$(CONFIG_MT7620_HELLO) += hello.o  
  
#vi RT288x_SDK/source/linux-2.6.36.x/drivers/net/hello/hello.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-2.6.36.x/drivers/net/Makefile

```
obj-$(CONFIG_MT7620_HELLO) += hello/
```

3. Modify RT288x_SDK/source/linux-2.6.36.x/ralink/Kconfig

```
#vi RT288x_SDK/source/linux-2.6.36.x/ralink/Kconfig
```

```
menu "Ralink Module"
```

```
config MT7620_HELLO
    tristate "MT7620 hello module"
```

4. Turn on the hello module

```
#make linux_menuconfig
```

```
Ralink Module --->
```

```
<M> MT7620 hello module
```

5. Compile the source code

```
#make dep
```

```
#make
```

6. Load the image to MT7620 RFB

7. Test in MT7620 console

```
/ # insmod hello
```

```
hello world
```

```
/ # rmmod hello
```

```
goodbye
```

9.7 Execute commands at boot up time

Edit RT288x_SDK/source/vendors/Ralink/{Platform}/rcS

```
#!/bin/sh
mount -a
mkdir -p /var/run
cat /etc_ro/motd
nvram_daemon&
goahead&

#for telnet debugging
telnetd

#for syslogd
mkdir -p /var/log <-- add new command after this line
```

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/{Platform}
- b. edit RT288x_SDK/source/vendors/Ralink/{ Platform }/Makefile

romfs:

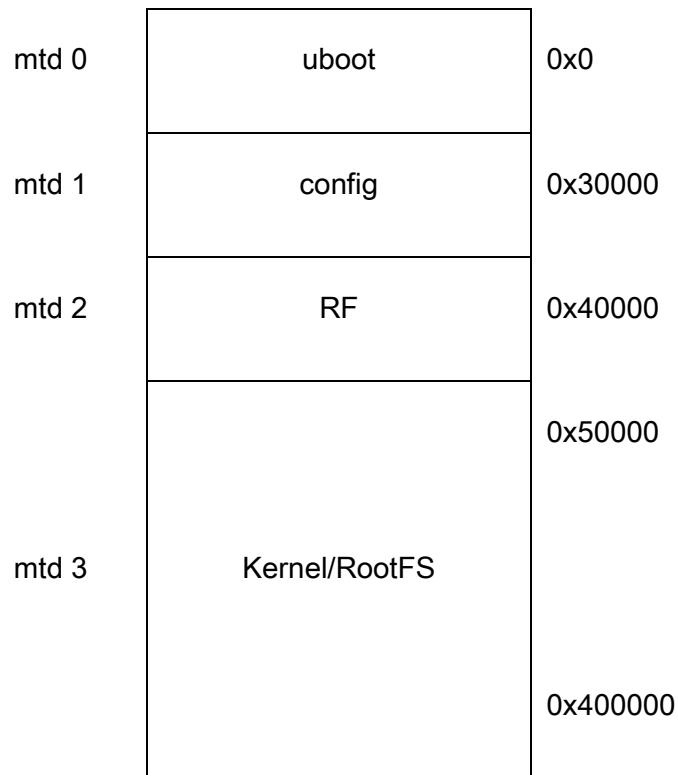
\$(ROMFSINST) /etc_ro/xxx.bin

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

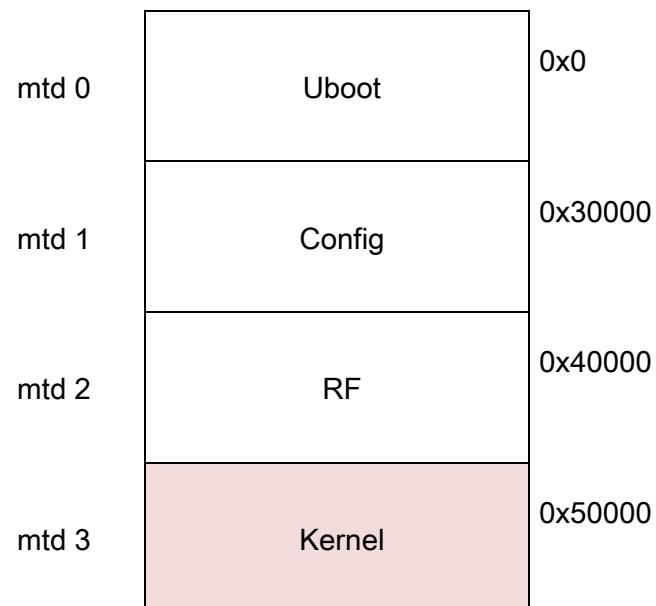
9.9 Reduce Image size

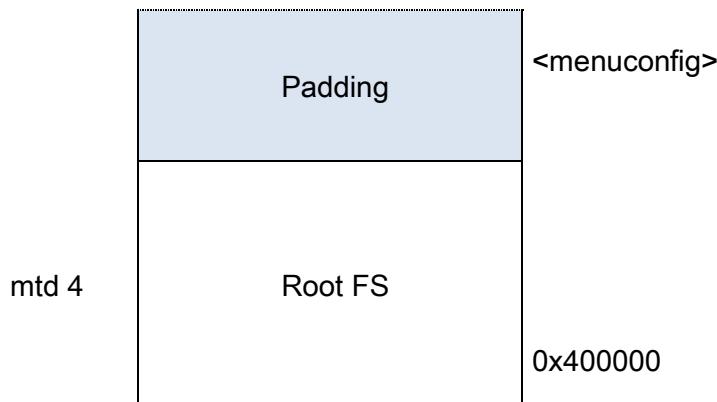
The MTD partitions are subsequently shown.

1. RootFS in RAM Mode



2. RootFS in Flash Mode with Padding





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.izma
```

Padded Kernel Image Size

```
786368 /home/steven/RT288x_SDK/source/images/zImage.izma
```

Original RootFs Size

```
4329746 /home/steven/RT288x_SDK/source/romfs
```

Compressed RootFs Size

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

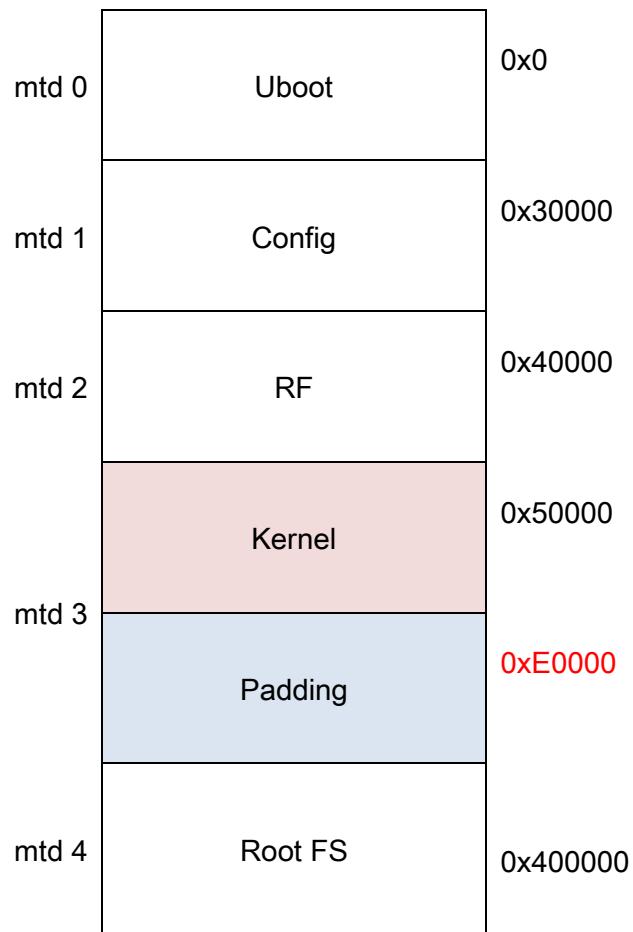
Padded Kernel Image + Compressed Rootfs Size

```
1855424 /home/steven/RT288x_SDK/source/images/zImage.izma
```

```
#===== .....
```

Step2: Change mtdblock size

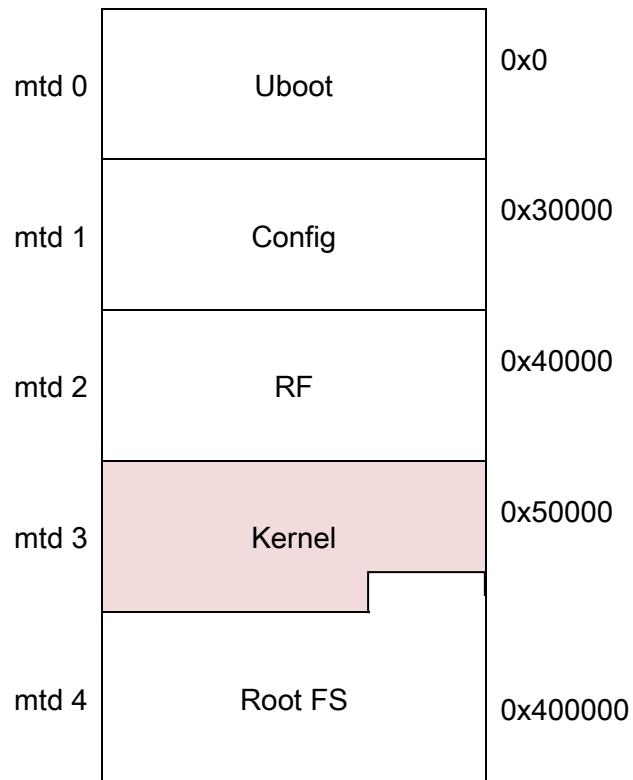
$576110 = 0x8CA6E \rightarrow 0x90000$ (multiple of $0x10000$ because the flash sector size=64KB)



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

```
System type (Ralink MT7620 board) --->
Soc Hardware Type (MT7620-ASIC) --->
DRAM Size (64M) --->
Flash Type (SPI) --->
[ ] Dual Image
[*] Kernel NVRAM
Root File System Type (RootFS_in_FLASH) --->
[ ] No Padding between Kernel and RootFS (NEW)
(0x90000) MTD Kernel Partition Size (Unit:Bytes)
```

3. RootFS in Flash Mode with no Padding



#make menuconfig

Kernel/Library/Defaults Selection --->

[] Customize Kernel Settings*

Machine selection --->

```

System type (Ralink MT7620 board) --->
Soc Hardware Type (MT7620-ASIC) --->
DRAM Size (64M) --->
Flash Type (SPI) --->
[ ] Dual Image
[*] Kernel NVRAM
Root File System Type (RootFS in FLASH) --->
[!] No Padding between Kernel and RootFS

```

10 FLASH LAYOUT AND FIRMWARE UPGRADE

10.1 Flash Layout

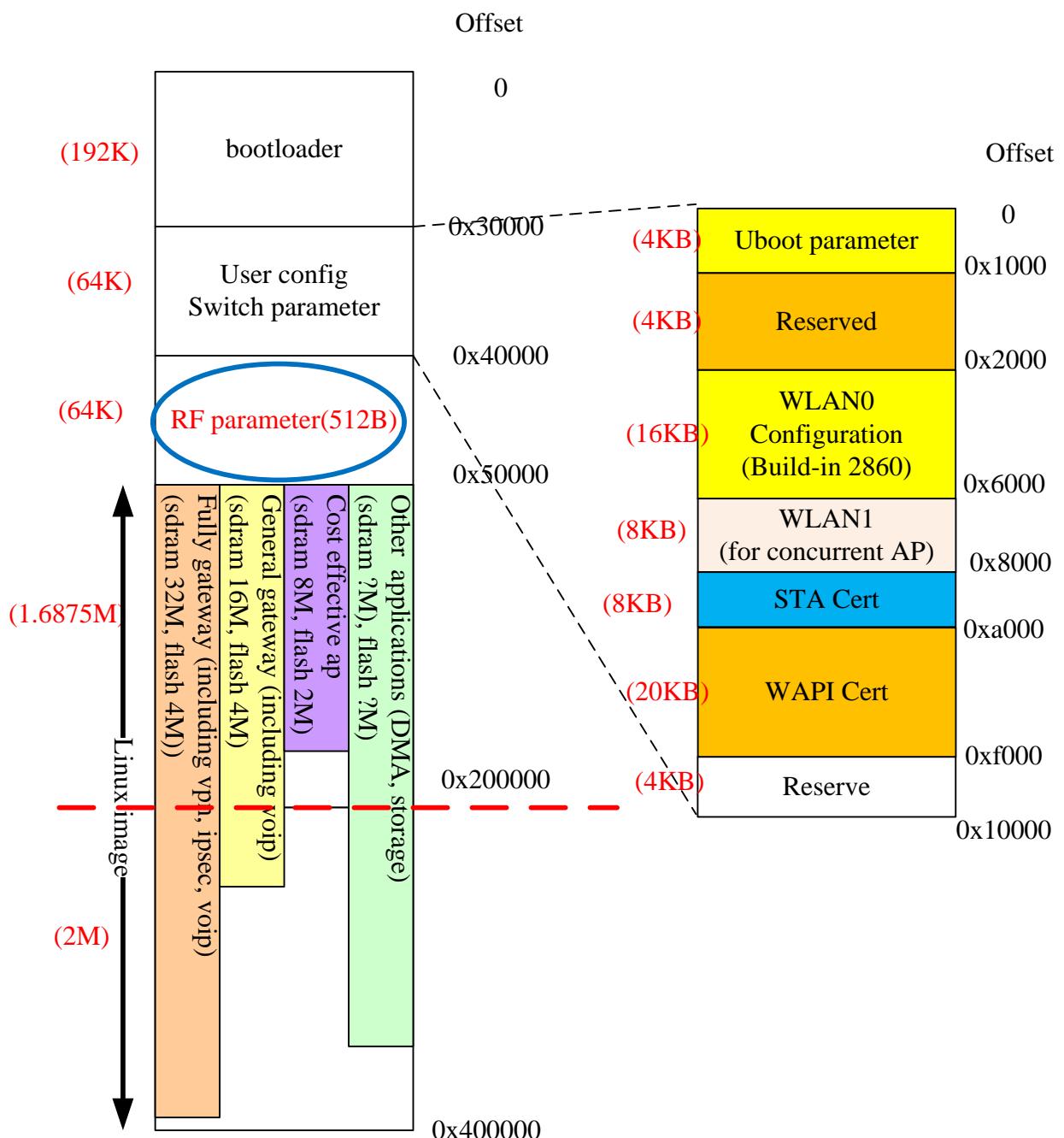


Figure 11 Ralink SDK Flash Layout (4MB)

In the 'user configure switch parameter' partition, the WLAN0 configuration is for built-in first wifi interface parameters, the WLAN1 configuration is for second wifi interface parameters, and the STA

cert configuration is stored certificate for station, and the WAPI cert configuration is stored certificate for WAPI. Beside Uboot and WLAN0 blocks, you may use the free space to save your own parameters.

Another, RT6855, RT6856 and MT7621 are standalone solutions without WiFi. So, their LAN/WAN MAC address is stored in 0xE000-0xE00b in RF Parameter block of flash.

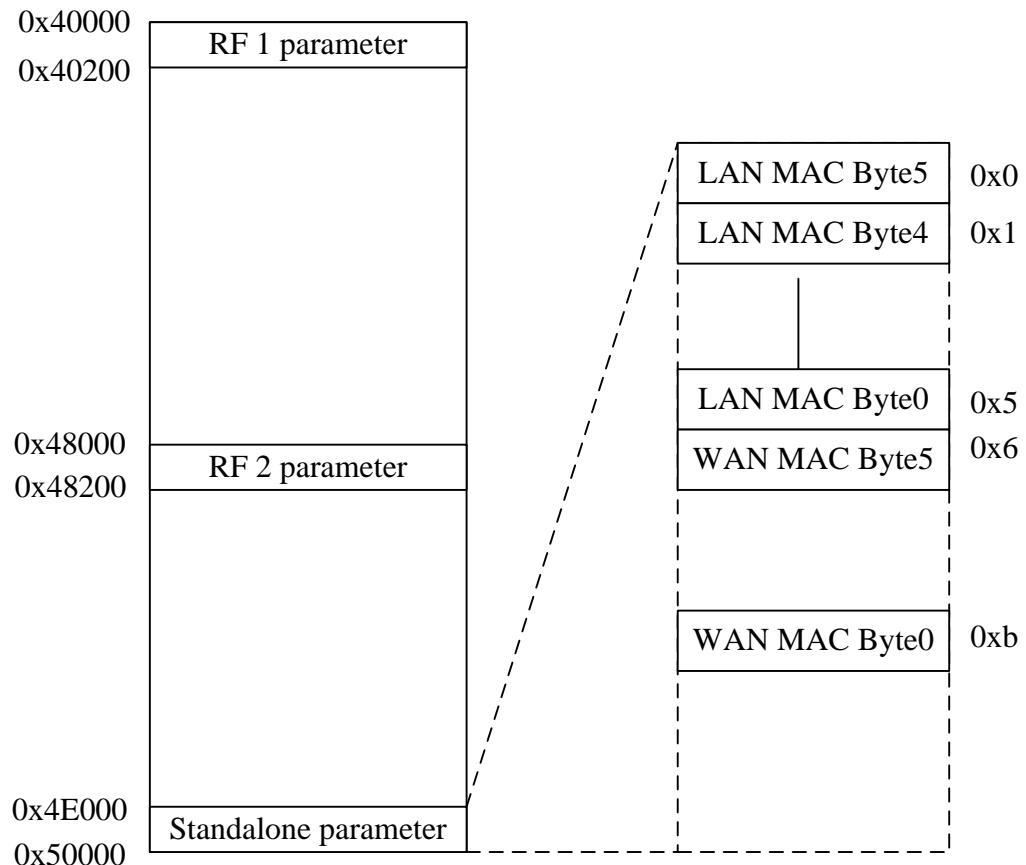


Figure 12 RF parameter block of Ralink SDK Flash Layout

10.2 Firmware Upgrade

10.2.1 By Uboot

A TFTP server is needed for upgrading firmware.

```
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.
7: Load Boot Loader code then write to Flash via Serial.
9: Load Boot Loader code then write to Flash 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 13 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 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 Ubuntu 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	Ubuntu
Kernel version	2.6.38
RAM	512MB
HD	40G

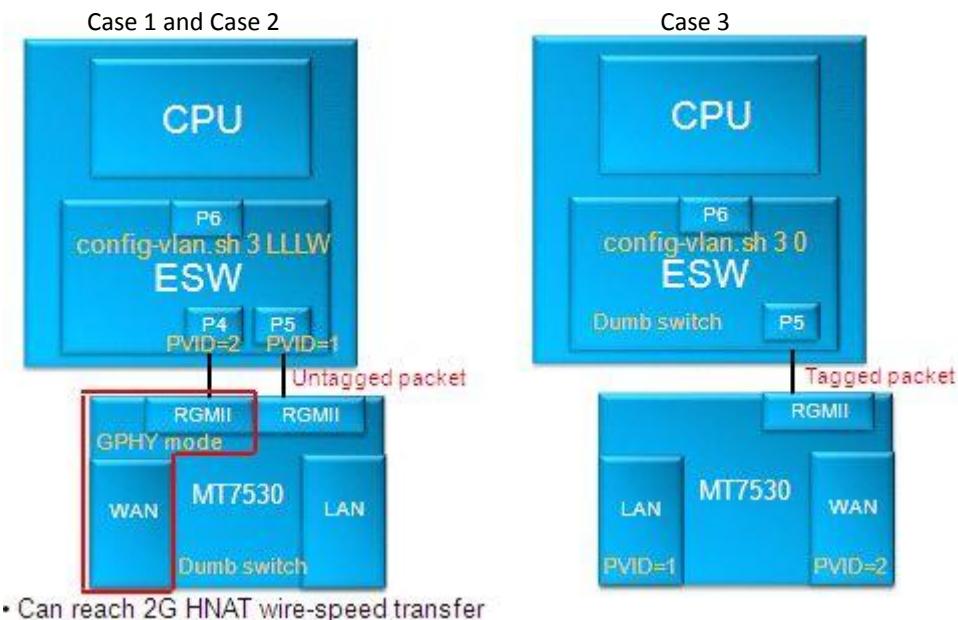
11.3 Quickly Build a Default Reference Image

Following steps can help you to setup your own development environment, and build a default firmware image of reference board for verification.

1. Untar the SDK source.
2. Install the toolchain by Chapter 5.
3. Do Wi-Fi driver porting by “Section 11.32 Concurrent AP porting Guide” if needed.
4. Choose the platform and use the default setting by “Section 9.1 Linux configuration”
5. Compile and get a new firmware image with root file system by “Section 9.5 Compile Linux image with root file system”
6. Upgrade the new firmware image on reference board to verify the new firmware image by “Section 10.2 Firmware Upgrade”

11.4 How to configure MT7620 to connect with the external switch

There are three cases of configuration:



Case 1: MT7620+MT7530 (2 RGMII and P0 as WAN port)

```

GMAC is connected to
[*] Balink Embedded Switch
[*] LAN/WAN Partition
    Switch Board Layout Type (LLLL/W) --->
[*] Port 4 Support
    Target Mode (GE_RGMII_MT7530_P0_AN (MT7530 Internal GigaPhy)) --->
[*] Port 5 Support
    Target Mode (MT7530 Giga_SW (RGMII)) --->

```

Case 2: MT7620+3-party GSW (2 RGMII and P0 as WAN port)

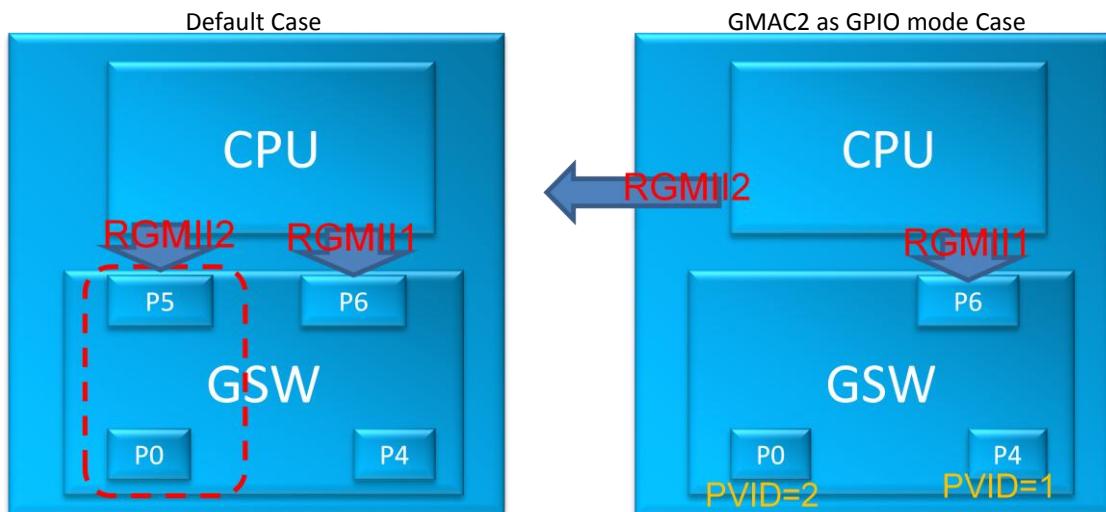
```
[*] Port 4 Support
  Target Mode (Giga Phy (RGMII)) --->
  (0x0) Port4 Phy Address
[*] Port 5 Support
  Target Mode (Giga SW/iNIC (RGMII)) --->
```

Case 3: MT7620+3-party GSW (1 RGMII and P0 as WAN port)

```
[*] LAN/WAN Partition
  Switch Board Layout Type (W/LLLL) --->
  [ ] Port 4 Support
  [*] Port 5 Support
  [ ] Target Mode (Giga SW/iNIC (RGMII)) --->
```

11.5 How to configure MT7621 internal switch to get more GPIO pins

MT7621 default configuration sets GMAC1(RGMII1) connecting to GSW, and GMAC2(RGMII2) connecting to GSW P0 as Phy Port mode. In the default case, it can reach 2G HNAT wire-speed transfer between LAN and WAN. If more GPIO pins are needed, the GMAC2(RGMII2) can be configured into GPIO mode. In this case, it only can reach 1G HNAT wire-speed.



Default Case:

```
-*- GMAC is connected to (RGMII_FORCE_1000 (GigaSW, CPU)) --->
-*- Ralink Embedded Switch
-*- LAN/WAN Partition
Switch Board Layout Type (W/LLLL) --->
[*] GMAC2 Support
  GMAC2 is connected to (Internal Gigaphy) --->
```

GMAC2 as GPIO mode Case:

```
-*- GMAC is connected to (RGMII_FORCE_1000 (GigaSW, CPU)) --->
-*- Ralink Embedded Switch
[*] LAN/WAN Partition
Switch Board Layout Type (W/LLLL) --->
[!] GMAC2 Support
```

11.6 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)

11.6.1 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.

11.6.2 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 (!strcmp(radio, "0", 2)) {

    nvram_bufset(RT2860_NVRAM, "RadioOn", radio);

    doSystem("ifconfig ra0 down");

    websRedirect(wp, "wireless/basic.asp");

    return;

}

else if (!strcmp(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
```

11.7 Enable Ethernet Converter / AP Client Mode

The Wi-Fi Interface should be configured into STA mode or AP client enabled. All PCs under the Ethernet port connect to the AP via NAT mechanism.

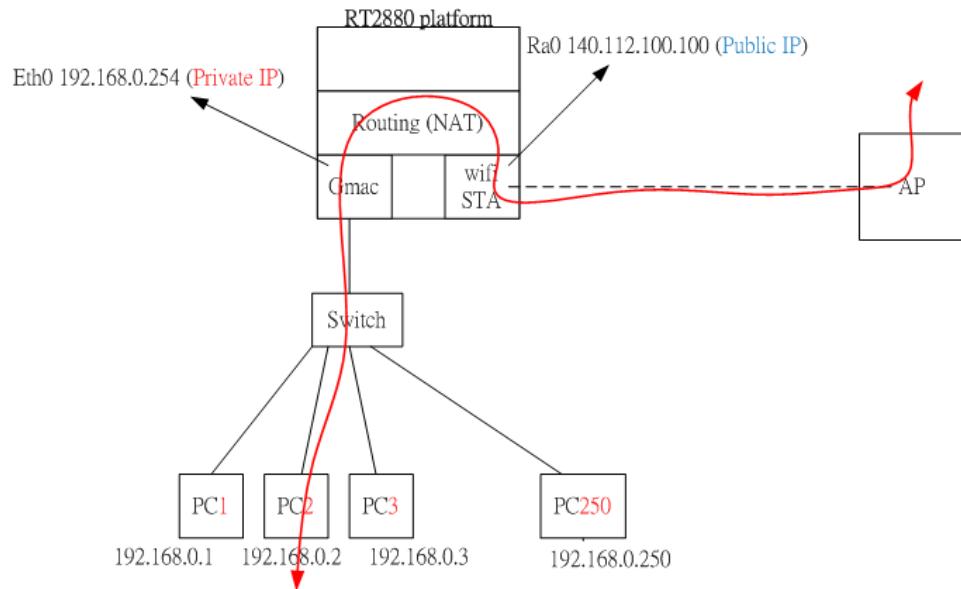


Figure 14 Ethernet Converter / AP Client Mode Operation Diagram

The detail configuration and Web GUI setting are described starting from next page.

1. Turn on the Wi-Fi STA support for Ethernet Converter.

```
#make menuconfig
Kernel/Library/Defaults Selection --->
[*] Customize Kernel Settings
Ralink Module --->
[*] WiFi Driver Support --->
```

```
--- WiFi Driver Support
    Choose First WiFi Interface (RT2860 for rt2860v2 driver) --->
    Choose Second WiFi Interface (None) --->
<*> Ralink RT2860 802.11n STA support
[ ] WPA Supplicant
[*] LED Support
[*] WSC (WiFi Simple Config)
[*]     WSC 2.0(WiFi Simple Config 2.0)
[ ] DLS ((Direct-Link Setup) Support
[ ] Video Turbine support
[ ] TSSI Compensation
[*] 802.11n Draft3
[*] Wireless Direct (P2P)
```

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

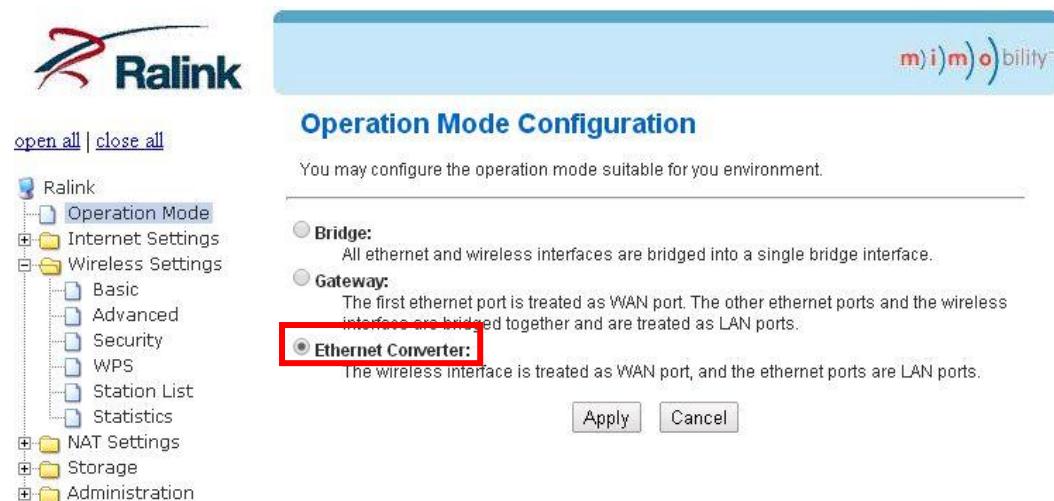


Figure 15.1 WebUI - STA Mode and Ethernet Converter Mode Setting

2. Turn on the Wi-Fi AP Client Mode

#make menuconfig

Kernel/Library/Defaults Selection --->

[*] Customize Kernel Settings

Ralink Module --->

[*] WiFi Driver Support --->

```
--- WiFi Driver Support
    Choose First WiFi Interface (RT2860 for rt2860v2 driver) --->
    Choose Second WiFi Interface (None) --->
<*> Ralink RT2860 802.11n AP support
[ ] LED Support
[*] WSC (WiFi Simple Config)
[*] WSC 2.0(WiFi Simple Config 2.0)
[ ] LLTD (Link Layer Topology Discovery Protocol)
[ ] WDS
[ ] MESSID
[*] AP-Client Support
[ ] MAC Repeater Support (NEW)
[ ] IGMP snooping
[ ] NETIF Block
```

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

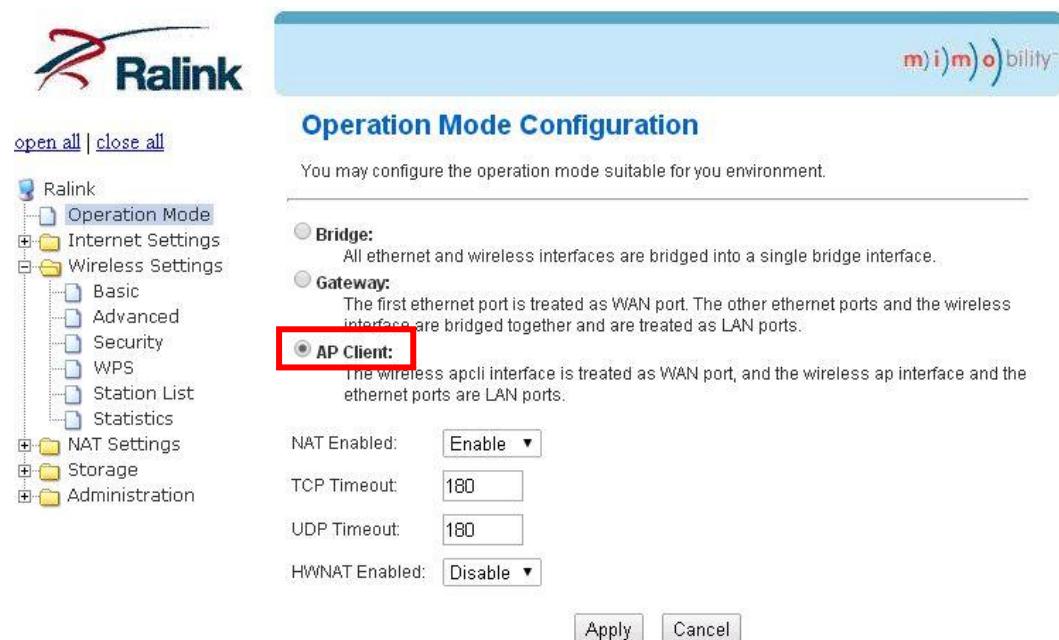


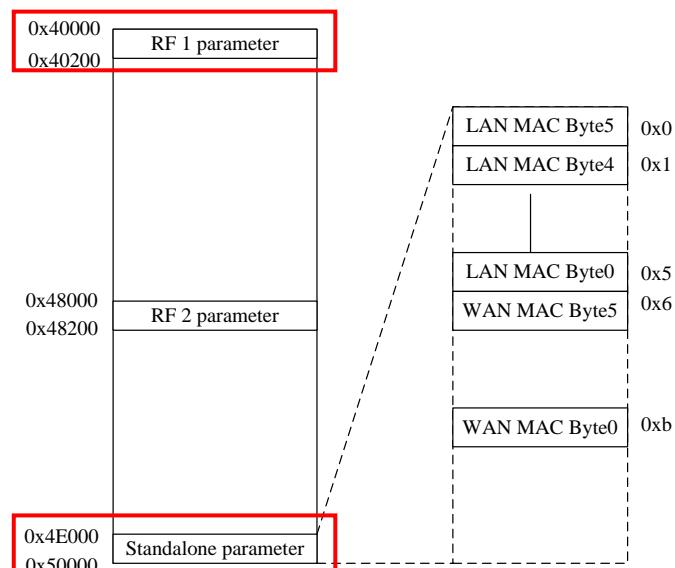
Figure 15.2 WebUI – AP Client Mode Setting

11.8 How to change the Ethernet MAC address

The Ralink Ethernet uses Factory MTD block to save its LAN/WAN MAC address. If the setting is empty, it will generate a random MAC address instead.

(Factory MTD Base Address: 0x40000)

Platform	LAN MAC Address	WAN MAC Address
MT7620/ MT7628	0x28	0x2E
MT7621/RT6856	0xE000	0xE006
Other RT series	0x28	0x2E



The following tools can help to modify the MAC address setting in the flash.

1. eth_mac
2. flash

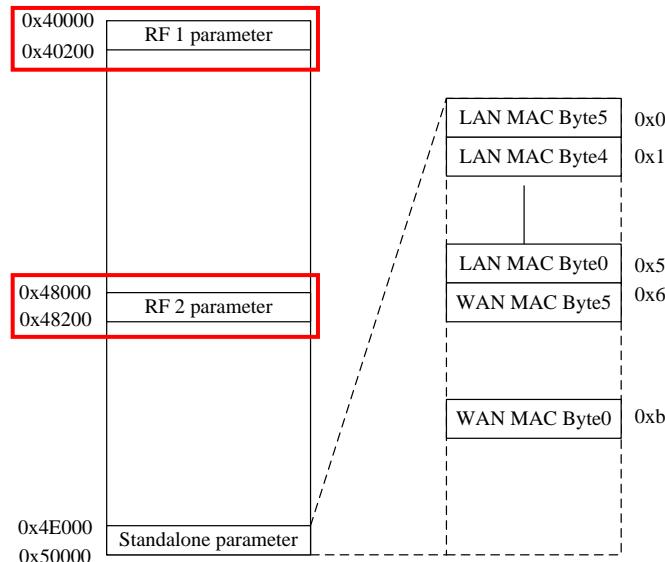
For example, Set the LAN (eth2.1) MAC address of MT7620 as 00:0c:43:76:21:01

1. # eth_mac w lan 00 0c 43 76 21 01
2. # flash -w 40028 -o 00
 # flash -w 40029 -o 0c
 # flash -w 4002A -o 43
 # flash -w 4002B -o 76
 # flash -w 4002C -o 21
 # flash -w 4002D -o 01

11.9 How to change the Wireless MAC address

The MAC address of wireless interface are set in the EEPROM content offset 0x0004, 0x0006, and 0x0008.

If the EEPROM type of the wireless interface is FLASH mode, the EEPROM content is stored in the factory MTD block. The first/second wireless interface uses the RF 1 parameter/RF 2 parameter in the factory MTD block.



Platform	1st Wireless Interface	2nd Wireless Interface
RF parameter Base Address	0x40000	0x48000
MAC Addr[0]	0x0004	0x0004
MAC Addr[1]	0x0005	0x0005
MAC Addr[2]	0x0006	0x0006
MAC Addr[3]	0x0007	0x0007
MAC Addr[4]	0x0008	0x0008
MAC Addr[5]	0x0009	0x0009

The following commands can help to modify the MAC address setting in the flash.

1. flash
2. iwpriv ra0 e2p / iwpriv rai0 e2p (Please refer to the SoftAP programming guide)

For example, set the MAC address of 1st Wi-Fi interface of MT7620 as 00:0c:43:76:21:16.

```
# flash -w 40004 -o 00
# flash -w 40005 -o 0c
# flash -w 40006 -o 43
# flash -w 40007 -o 76
# flash -w 40008 -o 21
# flash -w 40009 -o 16
```

11.10 How to configure GPIO ports

`$SDK/source/linux-2.6.21.x/drivers/char/ralink_gpio.c`

`$SDK/source/linux-2.6.21.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_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_INTP - 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 sent when the GPIO pin has been pressed for several seconds.

11.11 Use GPIO to turn on LED

The following tables show the current GPIO definition for RT2880/RT3052/RT3883/RT3352/RT5350.

Table 6 GPIO Usage of RT2880

RT2880- Pin # ^②	Pin name ^③	GPIO- define ^④	Board version ^⑤		Description ^⑥
			2.4G ^⑦	Dual ^⑧	
			V30RW-FE ^⑨	V11RW-GB ^⑩	
K20 ^⑪	GPIO0 ^⑫	WPS/ Reset to default ^⑬	● ^⑭	● ^⑮	Low Active signal input for Wi-Fi protection setup function and restore the setting to default value when push bottom for 3 second ^⑯
P17 ^⑰	GPIO8/DTR_N ^⑱	^⑲	● ^⑳	● ^㉑	Reserved ^㉒
R17 ^㉓	GPIO10/DCD_N ^㉔	Giga PHY Reset ^㉕	^㉖	● ^㉗	Low Active output for GigaPHY reset ^㉘
T18 ^㉙	GPIO11/DSR_N ^㉚	^㉛	● ^㉜	● ^㉝	Reserved ^㉞
P20 ^㉟	GPIO12/CTS_N ^㉟	System- Status/ ^㉟ Power- LED ^㉟	● ^㉟	● ^㉟	Low Active output for system-ready LED display ^㉟
N19 ^㉟	GPIO13/RIN ^㉟	Security- LED ^㉟	● ^㉟	● ^㉟	Low Active output for security LED indicates when wireless security is enabled; display security status on panel ^㉟
R20 ^㉟	GPIO14/RXD ^㉟	^㉟	● ^㉟	● ^㉟	Reserved for system reboot, Low Active output ^㉟

Table 7 GPIO Usage of RT3052

RT3052- Pin # ^①	Pin name ^②	GPIO- define ^③	Board version ^④	Description ^⑤
^⑥	^⑦	^⑧	AP-RT3052-V20RW-2X2 ^⑨	^⑩
U10 ^⑪	GPIO0 ^⑫	WPS- PBC ^⑬	● ^⑭	Low Active signal input for WPS function when push-bottom over 3 second ^⑮
T10 ^⑯	GPIO1/I2C_SD ^⑰	^⑱	^⑲	^⑳
R10 ^㉑	GPIO2/I2C_SCLK ^㉒	^㉓	^㉔	^㉕
U9 ^㉖	GPIO3/SPI_EN ^㉗	RX_SW ^㉘	● ^㉙	GPIO3/GPIO5 ANT diversity ^㉚ 10: ANT2 ^㉛ 01: ANT0 ^㉜
T9 ^㉟	GPIO4/SPI_CLK ^㉟	^㉟	^㉟	
U8 ^㉟	GPIO5/SPI_DOUT ^㉟	RX_SWN ^㉟	● ^㉟	
R9 ^㉟	GPIO6/SPI_DIN ^㉟	iNIC- mode- select ^㉟	● ^㉟	Resistor-strapping input ^㉟ 1: load code mode ^㉟ 0: dump switch mode ^㉟
G2 ^㉟	GPIO7/RTS_N ^㉟	^㉟	^㉟	^㉟
F2 ^㉟	GPIO8/TXD ^㉟	^㉟	^㉟	^㉟
G1 ^㉟	GPIO9/CTS_N ^㉟	System/- Power- LED ^㉟	● ^㉟	Low Active output ^㉟ System status/Power display ^㉟
J3 ^㉟	GPIO10/RXD ^㉟	SW_RST/- Factory ^㉟	● ^㉟	1. SW_RST ^㉟ 2. Low Active signal input ^㉟ 2. Factory default: push bottom over 3 second ^㉟
J4 ^㉟	GPIO11/DTR_N ^㉟	^㉟	^㉟	^㉟
H3 ^㉟	GPIO12/DCD_N ^㉟	^㉟	^㉟	^㉟
F1 ^㉟	GPIO13/DSR_N ^㉟	Security- LED ^㉟	● ^㉟	Low Active output security mode display ^㉟
K4 ^㉟	GPIO14/RIN ^㉟	WPS- LED ^㉟	● ^㉟	Low Active output ^㉟ Indicate WPS PBC status ^㉟

Table 8 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 /<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.

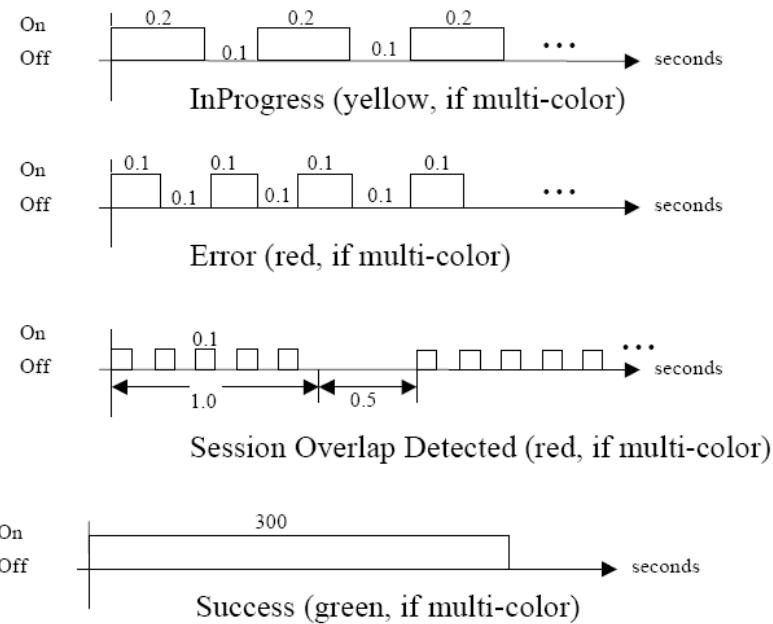


Figure 18 LED Definition of WPS Specification

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

11.12 Use LED firmware to turn on LED

1. Enable LED firmware

[RT2880]

```
#make menuconfig
```

```
Kernel/Library/Defaults Selection --->
```

```
[*] Customize Kernel Settings
```

```
Network device support --->
```

```
Ralink Driver --->
```

[MT7620]

#make linux_menuconfig

Ralink Module --->

[*] WiFi Driver Support --->

```
--- WiFi Driver Support
    Choose First WiFi Interface (MT7620 for rlt_wifi driver) --->
    Choose Second WiFi Interface (None) --->
<*> Ralink RT2860 802.11n AP support
    [*] LED Support
    [*] WSC (WiFi Simple Config)
    [*]     WSC 2.0(WiFi Simple Config 2.0)
    [ ] LLTD (Link Layer Topology Discovery Protocol)
```

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

Table 18 RT2860 LED Parameters in Flash

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

Address	States ^o	Bit ^o	RT2860_Pin-127_LED_behavior ^o
3Eh ^o	Radio off ^o	[1:0] ^o	00: Reserved ^o 01: Solid on ^o 10: Blink when transmitting data and management packet ^o 11: Blink when transmitting data, management packet and beacon ^o
		2 ^o	0: Solid on when no traffic ^o 1: Slow blink when no traffic ^o
		3 ^o	Reserved ^o
3Fh ^o	Radio on but link down ^o	[5:4] ^o	00: Reserved ^o 01: Solid on ^o 10: Blink when transmitting data and management packet ^o 11: Blink when transmitting data, management packet and beacon ^o
		6 ^o	0: Solid on when no traffic ^o 1: Slow blink when no traffic ^o
		7 ^o	Reserved ^o
		[9:8] ^o	00: Reserved ^o 01: Solid on ^o 10: Blink when transmitting data and management packet ^o 11: Blink when transmitting data, management packet and beacon ^o
		10 ^o	0: Solid on when no traffic ^o 1: Slow blink when no traffic ^o
3Fh ^o	Radio on and link to G ^o	11 ^o	Reserved ^o
		[13:12] ^o	00: Reserved ^o 01: Solid on ^o 10: Blink when transmitting data and management packet ^o 11: Blink when transmitting data, management packet and beacon ^o
		14 ^o	0: Solid on when no traffic ^o 1: Slow blink when no traffic ^o
		15 ^o	Reserved ^o

Address	States ^②	Bit ^③	LED behavior ^④
40h ^⑤	Radio off ^⑥	[3:0] ^⑦	bit0: LED G .. bit1: LED A .. bit2: LED Act .. bit3: 0: Reserved ..
			1: Positive polarity .. 0: Negative polarity
			1: LED ACT polarity inversion when link to A ..
			bit0: LED G .. bit1: LED A .. bit2: LED Act .. bit3: 0: Reserved ..
41h ^⑤	Radio on but link down	[7:4] ^⑧	1: Positive polarity .. 0: Negative polarity
			1: LED ACT polarity inversion when link to A ..
			bit0: LED G .. bit1: LED A .. bit2: LED Act .. bit3: 0: Reserved ..
			1: Positive polarity .. 0: Negative polarity
	Radio on and link to G	[11:8] ^⑨	1: Positive polarity .. 0: Negative polarity
			1: LED ACT polarity inversion when link to A ..
			bit0: LED G .. bit1: LED A .. bit2: LED Act .. bit3: 0: Reserved ..
			1: Positive polarity .. 0: Negative polarity
	Radio on and link to A	[15:12] ^⑩	1: Positive polarity .. 0: Negative polarity
			1: LED ACT polarity inversion when link to A ..
			bit0: LED G .. bit1: LED A .. bit2: LED Act .. bit3: 0: Reserved ..

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

11.13 How to start the telnet server

Check RT288x_SDK/source/user/busybox/.config

11.13.1 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

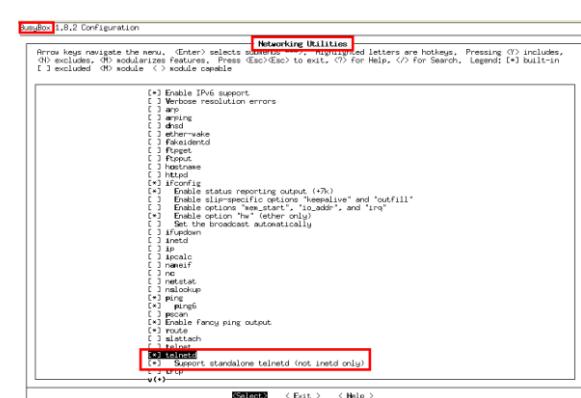
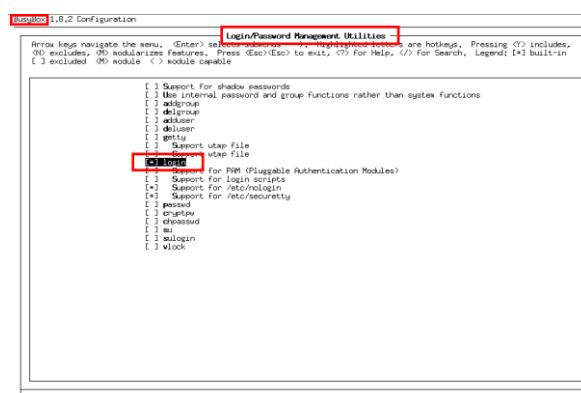
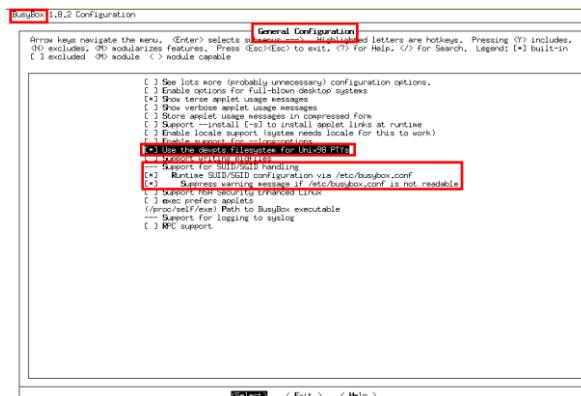
Check RT288x_SDK/source/linux-2.4.x/.config

11.13.2 Linux setting

CONFIG_UNIX98_PTYS=y → Character devices

CONFIG_UNIX98_PTY_COUNT=256

CONFIG_DEVPTS_FS=y → File systems



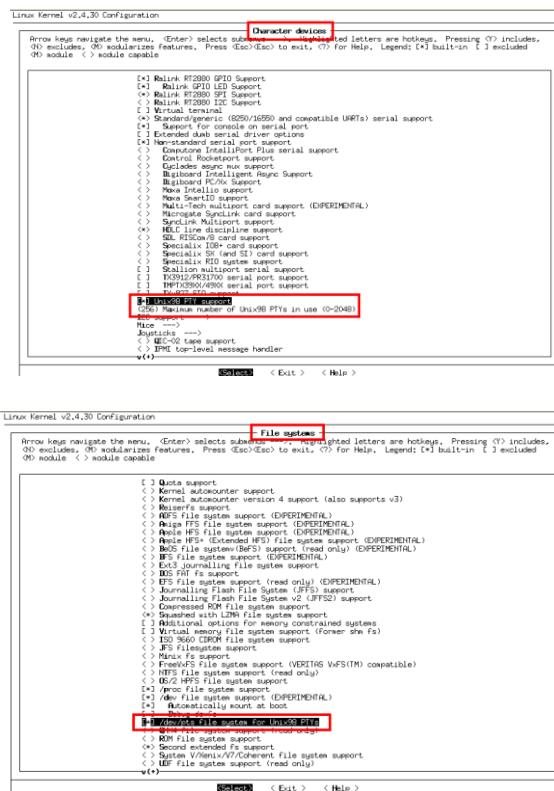


Figure 20 Configuration Procedure of Telnet Server

11.14 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.

$$N_{SD}[40\text{Mhz}] = 108$$

$$N_{SD}[20\text{Mhz}] = 52$$

$$N_{SD}[40\text{MHz}]/N_{SD}[20\text{MHz}] = 108/52$$

$$= 2.0769230769230769230769231$$

- b. Example:

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

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

$$= 130 * [108 / 52]$$

= 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	NBPSCS(iSS)	NSD	NSP	NCBPS	NDBPS	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.

$$T_{sym} / T_{syms} = 4\mu\text{sec} / 3.6\mu\text{sec} = 10/9$$

c. Example:

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

$$270.0 * (10/9) = 300.0 \text{ for Short GI.}$$

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.

EWC®

PHY spec, v1.27

Parameter	Value in legacy 20MHz channel	Value in 20MHz HT channel	Value in 40MHz channel	
			HT format	Legacy Duplicate
T _{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 _{GI: Double GI}	1.6μsec	1.6μsec	1.6μsec	
T _{GI: Short Guard Interval length}	0.4μsec = T _{FFT} /8	0.4μsec	0.4μsec	
T _{L-ST: Legacy Short training sequence length}	8μsec = 10 × T _{FFT} /4	8μsec	8μsec	
T _{L-LTE: Legacy Long training sequence length}	8μsec = 2 × T _{FFT} + T _{GI}	8μsec	8μsec	
T _{SYM: Symbol Interval}	4μsec = T _{FFT} + T _{GI}	4μsec	4μsec	
T _{SYMS: Short GI Symbol Interval}	3.6μsec = T _{FFT} + T _{GI}	3.6μsec	3.6μsec	
T _{L-SIG}	4μsec = T _{SYM}	4μsec	4μsec	

$T_{sym}/T_{syms} = 4\mu/3.6\mu = 10/9$

3) EWC PHY spec. page 13.

EWC®

PHY spec, v1.27

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 $I = 3(\lceil N_{data} \rceil + N_{LTE} + 3) - 3$ 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 $\frac{1}{10} \cdot N_{LTE}$ is the number of HT training symbols. The symbol $\lceil x \rceil$ denotes the lowest integer greater or equal to x .

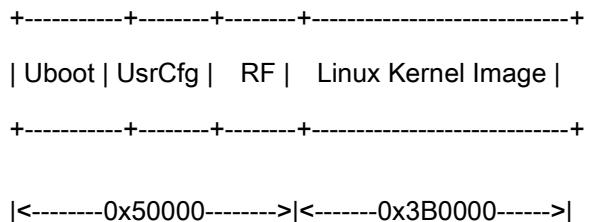
11.15 How to build a single image for the flash programmer

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_vlimage  
#  
# Single Image Information  
#  
PACKED_DIR = .  
PACKED_IMAGE = steven_vlimage.img
```

```
#cp /tftpboot/u-boot.bin .
#cp /tftpboot/steven_ulimage .
#make -f Makefile.4M
```

Flash layout:



-Original Uboot Image Size

149372 ./u-boot.bin

- Original Kernel Image Size

2779348 ./steven_ulimage

- Packed Image Size

4194304 ./steven_ulimage.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_ulimage
-rwxr-xr-x 1 steven users 4194304 Jun 27 13:36 steven_ulimage.img*
-rwxr-xr-x 1 steven users 149372 Jun 27 13:34 u-boot.bin*
```

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

11.16 How to power down the RT3x5x/RT5350/MT7628 build-in 10/100 PHYs

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 POC1[27:23] to disable Phy port.

<code>*(unsigned long *)(0xb0110090) = 0x0??07f7f;</code>

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

The Ralink Ethernet driver can be referenced.

[RT3x8x]: RT288x_SDK/source/linux-2.6.21.x/drivers/net/raeth/raether.c

[MT7628]: RT288x_SDK/source/linux-2.6.36.x/drivers/net/raeth/raether.c

11.17 How to power down the RT6855/RT6856/MT7620 build-in 10/100 PHYs

Please modify GPC1[29:24] to disable PHY ports by following command in console.

```
# switch reg s 7014 ??e0000c
```

GPC1: GIGA Port-I Control (offset: 0x7014)

Bits	Type	Name	Description	Initial value
31:30	-	-	Reserved	0x0
29:24	RW	PHY_DIS	Disable Internal 5-port EPHY.	0x0

Furthermore, MT7620 can save more power to stop the clock inside switch by modifying CKGCR[0] and CKGCR[1] via following command.

```
# switch reg s 3ff0 1e03
```

394. CKGCR: Clock Gating Control Register (offset: 0x3FF0)

Bits	Type	Name	Description	Initial Value
31:16	-	-	Reserved	0x0000
15:8	RW	LPI_TXIDLE THD	Low Power Idle (LPI) Tx Idle Threshold When there is no packet to be transmitted and exceeds time period specified by LPI_TXIDLE THD, the TXMAC will automatically enter LPI mode and send EEE LPI frame to link partner. Default: 30 ms (unit: 1 ms)	0x1e
7:6	-	-	Reserved.	0x0
5	RW	CKG_TXIDLE	Tx Global Clock Idle Stop 0: Keep Tx port clock ticking. 1: Stop Tx port clock ticking when the corresponding port has no traffic to send and Rx port blocks have been idle for <LPI_TXIDLE THD> ms.	0x0
4	RW	CKG_RXLPI	Rx Global Clock Idle 0: Keep Rx port clock ticking 1: Stop Rx port clock ticking when the corresponding port enters LPI mode and Rx port blocks are idle.	0x0
3:2	-	-	Reserved.	0x0
1	RW	CKG_LNKDN_PORT	Global Clock Link-Down Port Stop Port clock: clocks for GMAC, PORT_CTRL, and SCH blocks 0: Keep Rx and Tx port clock ticking 1: Stop both Rx and Tx port clock ticking when the corresponding port enters link-down status for 7 seconds.	0x1
0	RW	CKG_LNKDN_GLB	Global Clock Link-Down Stop Global clock: Clock for BMU, PB_CTRL, and ARL blocks 0: Keep the global clock ticking. 1: Stop the global clock ticking when all ports enter link-down status for 7 seconds.	0x1

11.18 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,wszie=32768,hard,nolock,proto=udp,timeo=7,retrans=3,sec=sys,addr=192.16  
8.18.21)
```

11.19 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 to RT288x_SDK/source/user/goahead/Makefile:

```
ifeq ("$(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 (!strcmp(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

11.20 How to enable watchdog

- User mode Watchdog:

\$ make menuconfig

Kernel/Library/Defaults Selection --->

[*] Customize Kernel Settings

Device Drivers --->

[*] Watchdog Timer Support --->

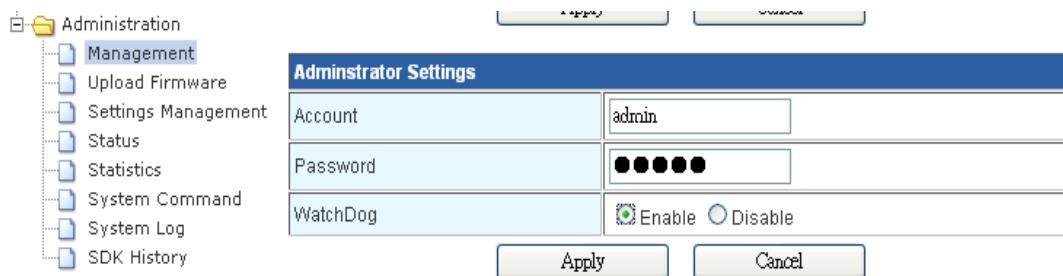
```
--- Watchdog Timer Support
[ ] Disable watchdog shutdown on close
    *** Watchdog Device Drivers ***
< > Software watchdog
< > ALi M7101 PMU Computer Watchdog
<*> Ralink APSoC Hardware Watchdog
[ ] Ralink WatchDog Reset Output (NEW)
    *** PCI-based Watchdog Cards ***
< > Berkshire Products PCI-PC Watchdog
< > PCI-WDT500/501 Watchdog timer
    *** USB-based Watchdog Cards ***
< > Berkshire Products USB-PC Watchdog
```

[*] Customize Vendor/User Settings

Miscellaneous Applications --->

```
[ ] lsusb
[ ] usb_modeswitch
[ ] comgt
[ ] sdparm
[*] watchdog
```

Finally, Enable watchdog in WebUI.



- Kernel mode Watchdog:

```
$ make menuconfig
```

Kernel/Library/Defaults Selection --->

[*] Customize Kernel Settings

Machine selection --->

```
<*> Ralink Hardware Timer0/Timer1
[ ]   Ralink Timer0
<*> WatchDog Timer
[ ]   Ralink WatchDog Reset Output (NEW)
(10)  WatchDog Timer (Unit:1Sec, Max=30Sec) (NEW)
(4)   WatchDog Refresh Interval (Unit:1Sec, Max=30Sec) (NEW)
```

NOTES: PLEASE ENABLE ONE MODE AT A TIME.

11.21 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

.....

11.22 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

11.23 How to enable software QoS

To support the Ralink SW QoS, many menuconfig options in Ralink SDK must be enabled, including in kernel and application configs. Following are the configuration of Kernel linux-2.6.21.x and Kernel linux-2.6.36.x.

- Kernel linux-2.6.21.x config :

Kernel IMQ config:

Since the Intermediate Queueing (IMQ) pseudo device are used to support Ralink SW QoS, it must be enabled first, or some needed options in Netfilter configs won't show up due to dependency.

Networking --->

Device Drivers --->

Network device support --->

<*> IMQ (intermediate queueing device) support

IMQ behavior (PRE/POSTROUTING) (IMQ AB)

(2) Number of IMQ devices

Kernel Netfilter configs:

In order to support Ralink SW QoS, several necessary Netfilter modules are used, including Netfilter match and target modules. These modules must be enabled to let Ralink SW QoS work correctly. But first of all, a proprietary Ralink option in Netfilter has to be enabled.

To completely fit the requirement of Ralink SW QoS some changes are made in Linux Netfilter architecture. For this changes, a Ralink proprietary Netfilter option **Netfilter Ralink SWQoS support** is introduced. This Ralink proprietary Netfilter option must be enabled to support Ralink SW QoS, or the classification of IP address may not work properly. If the Ralink SW QoS is not required, of course, it is recommended to leave this option blank to keep the Linux Netfilter architecture unchanged and expected.

-> Networking

-> Networking support (NET [=y])

-> Networking options

-> Network packet filtering framework (Netfilter) (NETFILTER [=y])

-> Core Netfilter Configuration

[*] **Netfilter Ralink SWQoS support(Marking after NAT)**

Then please enable the following necessary netfilter and iptables modules to support Ralink SW QoS:

-> Networking

-> Networking support (NET [=y])

-> Networking options

-> Network packet filtering framework (Netfilter) (NETFILTER [=y])

-> Core Netfilter Configuration

<*> Netfilter connection tracking support

<*> "conntrack" connection tracking match support

<*> "DSCP" target support

<*> "MARK" target support

<*> "DSCP" match support

<*> "helper" match support

<*> "length" match support

<*> "mac" address match support

<*> "state" match support

<*> "layer7" match support

<*> "Ethernet port for incoming packets" match support

And,

-> Networking

-> Networking support (NET [=y])

-> Networking options

-> Network packet filtering framework (Netfilter) (NETFILTER [=y])

->IP: Netfilter Configuration --->

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

<*> Packet mangling

<*> IMQ target support

Application configs:

Besides kernel configs, there are also several application menuconfigs which has to be enabled to support Ralink SW QoS.

[*] Customize Vendor/User Settings

Library Configuration --->

[*] Build libresolv

Network Applications --->

[*] iptables

[*] iproute2

[*] tc

Ralink Proprietary Application --->

[*] Software QoS

- **Kernel linux-2.6.36.x config :**

Networking support →

Networking options →

Network packet filtering framework (Netfilter)→

```
--- Network packet filtering framework (Netfilter)
[ ] Network packet filtering debugging
[*] Advanced netfilter configuration
[*] Bridged IP/ARP packets filtering
    Core Netfilter Configuration --->
<*> IP virtual server support --->
    IP: Netfilter Configuration --->
< > Ethernet Bridge tables (ebtables) support --->
```

QoS and/or fair queueing ---> all selected

IP: Netfilter Configuration:

```
<*> IPv4 connection tracking support (required for NAT)
<*> IP tables support (required for filtering/masq/NAT)
<*> "addrtype" address type match support
<*> "ah" match support
<*> "ecn" match support
<*> "ttl" match support
<*> Packet filtering
<*> REJECT target support
<*> LOG target support
<*> ULOG target support
<*> Packet mangling
<*> ECN target support
<*> "TTL" target support
<*> raw table support (required for NOTRACK/TRACE)
<*> ARP tables support
<*> ARP packet filtering
<*> ARP payload mangling
```

Core Netfilter Configuration (EXPERIMENTAL =y):

```

[NAT Types (Restricted Cone (EXPERIMENTAL))  --->
[*] Netfilter Ralink SWQoS support
<*> Netfilter NFQUEUE over NFNETLINK interface
-- Netfilter LOG over NFNETLINK interface
<*> Netfilter connection tracking support
-**- Connection mark tracking support
[ ] Connection tracking zones
[*] Connection tracking events
<*> DCCP protocol connection tracking support (EXPERIMENTAL)
<*> SCTP protocol connection tracking support (EXPERIMENTAL)
< > UDP-Lite protocol connection tracking support
< > Amanda backup protocol support
<*> FTP protocol support
< > H.323 protocol support
< > IRC protocol support
< > NetBIOS name service protocol support
<*> PPtP protocol support
< > SANE protocol support (EXPERIMENTAL)
< > SIP protocol support
<*> TFTP protocol support
< > RTSP protocol support
<*> Connection tracking netlink interface
< > Transparent proxying support (EXPERIMENTAL)
-- Netfilter Xtables support (required for ip_tables)
    *** Xtables combined modules ***
-- nfmark target and match support
-- ctmark target and match support
    *** Xtables targets ***
<*> CHECKSUM target support
<*> "CLASSIFY" target support
<*> "CONNMARK" target support
<*> "CT" target support
<*> "DSCP" and "TOS" target support
-- "HL" hoplimit target support
<*> IDLETIMER target support
<*> "IMQ" target support
<*> "MARK" target support

```

```

<*> "NFLOG" target support
<*> "NFQUEUE" target Support
<*> "NOTRACK" target support
-- "RATEEST" target support
<*> "TEE" - packet cloning to alternate destination
<*> "TRACE" target support
<*> "TCPMSS" target support
<*> "TCPOPTSTRIP" target support (EXPERIMENTAL)
    *** Xtables matches ***
<*> "cluster" match support
<*> "comment" match support
<*> "connbytes" per-connection counter match support
<*> "connlimit" match support"
<*> "connmark" connection mark match support
<*> "conntrack" connection tracking match support
<*> "cpu" match support
<*> "dccp" protocol match support
<*> "dscp" and "tos" match support
<*> "esp" match support
<*> "hashlimit" match support
<*> "helper" match support
-- "hl" hoplimit/TTL match support
<*> "iprange" address range match support
<*> "ipvs" match support
<*> "length" match support
<*> "limit" match support
<*> "mac" address match support
<*> "mark" match support
<*> "multiport" Multiple port match support
<*> "osf" Passive OS fingerprint match
<*> "owner" match support
<*> "physdev" match support
<*> "pkttype" packet type match support
<*> "quota" match support
<*> "rateest" match support
<*> "realm" match support
<*> "recent" match support
<*> "sctp" protocol match support (EXPERIMENTAL)
<*> "state" match support
<*> "layer7" match support
[*]     Layer 7 debugging output
<*> "statistic" match support
<*> "string" match support
<*> "webstr" match support
<*> "tcpmss" match support
<*> "time" match support
<*> "u32" match support

```

Device drivers →

Network device support →

```
-#- IMQ (intermediate queueing device) support
    IMQ behavior (PRE/POSTROUTING) (IMQ AB) --->
(2) Number of IMQ devices
```

Customize uClibc Settings

Target Architecture Features and Options --->

[*] Enable full C99 math library support

Customize Vendor/User Settings

Library Configuration --->

[*] Build libresolv

Network Applications --->

[*] iptables

[*] iproute2

[*] tc

Ralink Proprietary Application --->

[*] Software QoS

11.24 QoS information

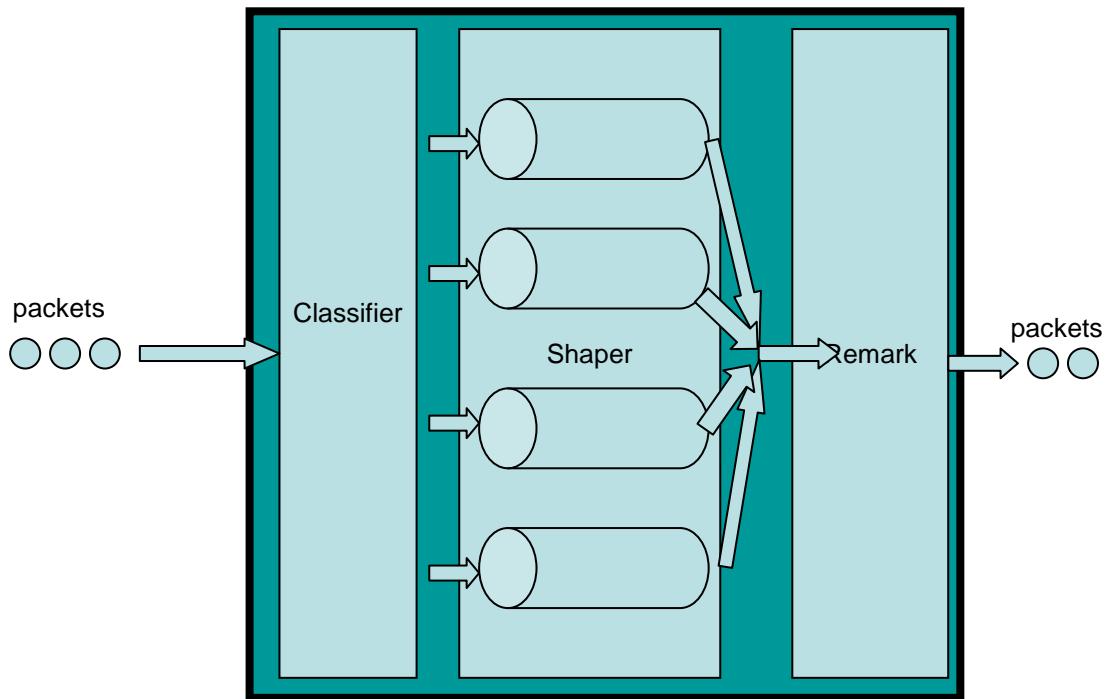
11.24.1 Software QoS – Preface

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 supports download and upload stream on a WAN interface.

11.24.2 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.

SW QoS



11.24.3 Software QoS – Usage

Conceptually, there are three main settings in Ralink SW QoS have to be specified : Global settings, Group settings, and Rule settings.

Global settings:

Quality of Service Settings

You may setup rules to provide Quality of Service guarantees for specific applications.

QoS Setup	
Quality of Service	Download from Internet
Upload Bandwidth:	32M Bits/sec
Download Bandwidth:	32M Bits/sec
QoS Model:	DRR
Reserved bandwidth:	0% (10% is recommended)

1. Select "upload to Internet " or "download from Internet" on the web UI to enable the Ralink SW 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....)
3. Select a QoS model: AUTO QoS, DRR (Deficit Round Robin), SPQ(Strict Priority Queue), DRR+SPQ.

4. Select reserved bandwidth. The reserved bandwidth is out of the control of Ralink SW QoS.

Group settings:

Four QoS groups are shown after specifying Global settings in Ralink SW QoS. Now all packets through this gateway are classified into these four QoS groups according to the user's QoS rules settings. The four QoS groups are subsequently shown.

Quality of Service Settings

You may setup rules to provide Quality of Service guarantees for specific applications.

QoS Setup	
Quality of Service	Download from Internet
Upload Bandwidth:	32M Bits/sec
Download Bandwidth:	32M Bits/sec
QoS Model:	DRR
Reserved bandwidth:	0% (10% is recommended)
QoS Download Settings	
Highest	Rate: 10% Cell: 100%
High	Rate: 10% Cell: 100%
Default	Rate: 10% Cell: 100%
Low	Rate: 10% Cell: 100%
<input type="button" value="Submit"/>	

The default group is the group named Default(the third group), that means the packet would be classified into this group if it doesn't match with any rules.

QoS Download Settings	
Highest	Rate: 10% Cell: 100%
High	Rate: 10% Cell: 100%
Default	Rate: 10% Cell: 100% Default group
Low	Rate: 10% Cell: 100%
<input type="button" value="Submit"/>	

In each QoS group there are two attributes Rate and Ceil as shown in the subsequent figure.

QoS Download Settings			
Highest	Rate: <input type="text" value="0%"/>	Cell: <input type="text" value="100%"/>	
High	Rate: <input type="text" value="10%"/>	Cell: <input type="text" value="100%"/>	
Default	Rate: <input type="text" value="20%"/>	Cell: <input type="text" value="100%"/>	
Low	Rate: <input type="text" value="40%"/>	Cell: <input type="text" value="100%"/>	
	50%		
	60%		
	70%		
	80%		
	90%		
	100%		
<input type="button" value="Submit"/>			

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

b. Ceil: Set the maximum bandwidth that this group can use.

The first group named Highest has the highest priority. The next group named High has the second priority. The third group named Default is the default group. The last group named Low has the lowest priority.

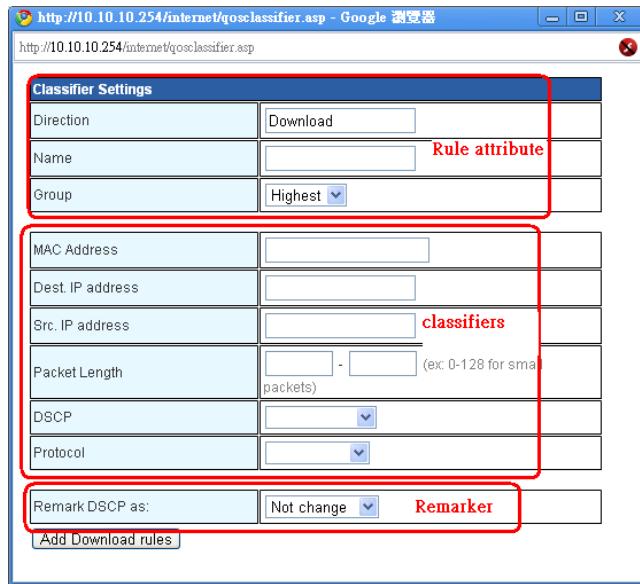
QoS Download Settings			
Highest	Highest group	Rate: <input type="text" value="10%"/>	Cell: <input type="text" value="100%"/>
High	High group	Rate: <input type="text" value="10%"/>	Cell: <input type="text" value="100%"/>
Default	Default	Rate: <input type="text" value="10%"/>	Cell: <input type="text" value="100%"/>
Low	Lowest group	Rate: <input type="text" value="10%"/>	Cell: <input type="text" value="100%"/>
<input type="button" value="Submit"/>			

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

Rules settings:

The QoS rules are made to do classification, and remarking. One QoS rules are made of 3 parts: attributes, classifiers, and remaker.

No	Name	Group	Info.
	Add Download rules		<input type="button" value="Delete"/>



1) Rule attribute:

- a) Name: specifies this rule's name
- b) Group: specifies which group this rule belongs to.

2) Rule classifiers:

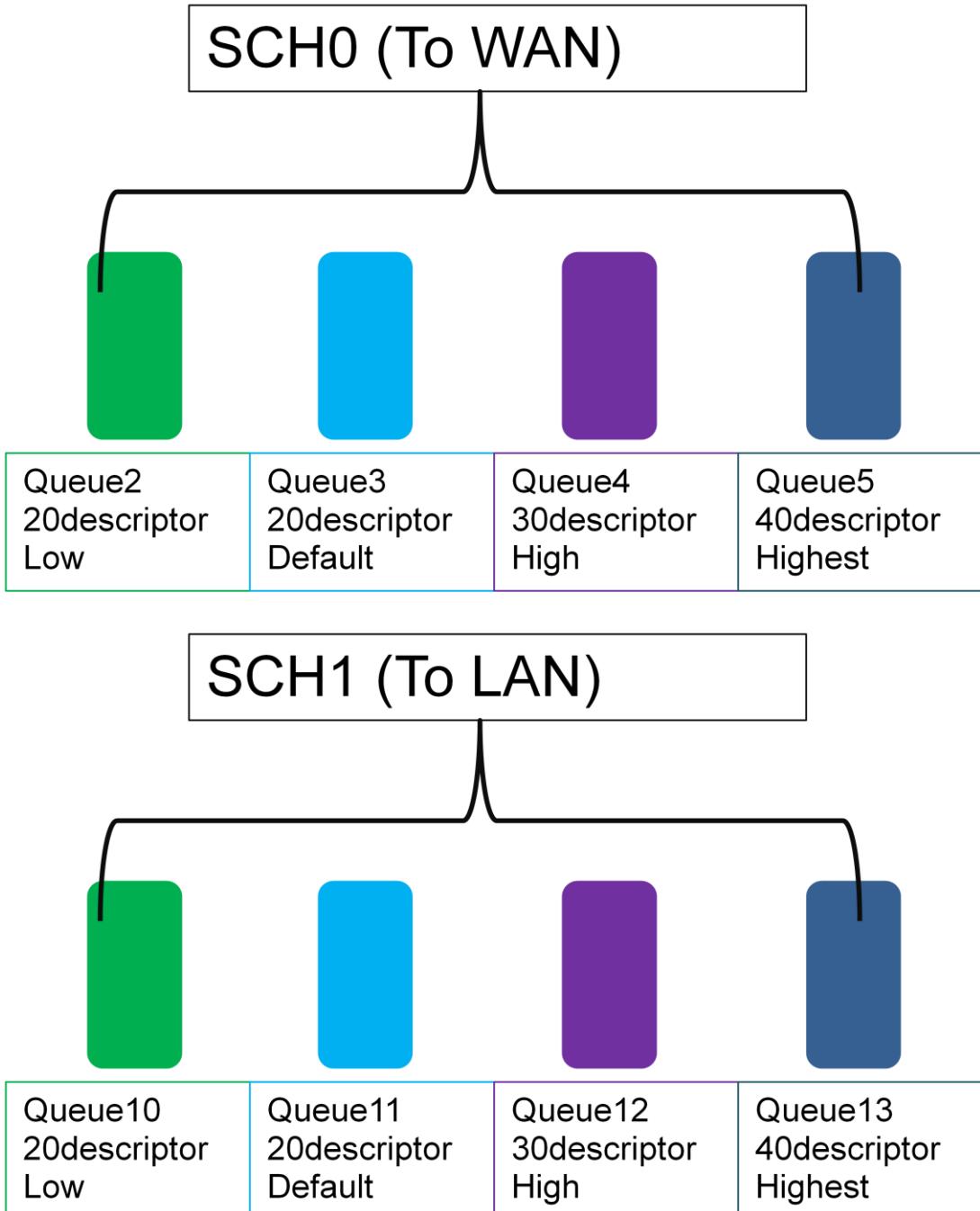
Ralink SW QoS supports these classifiers currently:

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

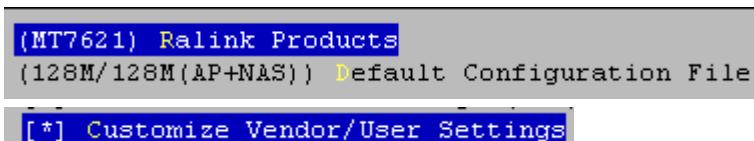
3) Rule Remarker: This argument specifies what DSCP value would be added to the packet as a remark which matches the rule.

11.24.4 Hardware QoS – Usage

The Ralink SoC HW QoS architecture is shown in the subsequent figure.



```
# make menuconfig
```



Proprietary Application --->

```
[*] QoS Support
(Hardware) QoS
```

11.25 How to enable USB Ethernet (example 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

11.26 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

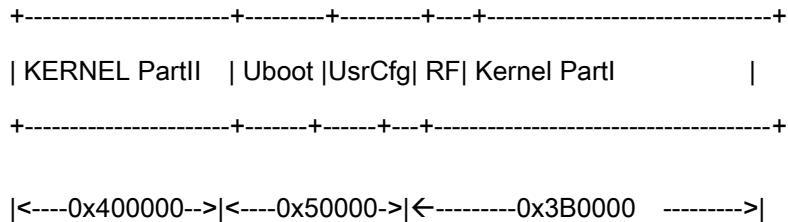
UBOOT_IMAGE = rt2880_100phy_128Mbx16_8Mflash.uboot

KERNEL_IMAGE = rt2880_100phy_128Mbx16_8Mflash.linux

PACKED_IMAGE = rt2880_100phy_128Mbx16_8Mflash.pack

```
#make -f Makefile.8M
```

Flash layout:



11.27 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

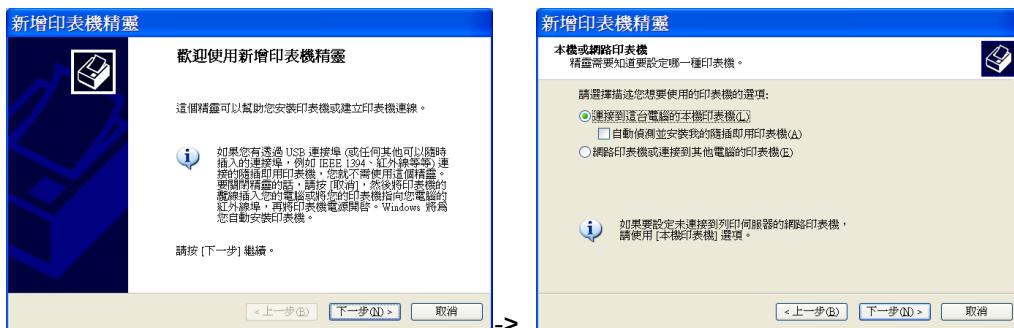
```
# usb 1-1: new full speed USB device using dwc_otg and address 2
usb 1-1: Product: Officejet 4300 series
usb 1-1: Manufacturer: HP
usb 1-1: SerialNumber: CN864GZ1S004GR
usb 1-1: configuration #1 chosen from 1 choice
drivers/usb/class/usblp.c: usblp0: USB Bidirectional printer dev 2 if 1 alt 0 proto 2 vid
0x03F0 pid
```

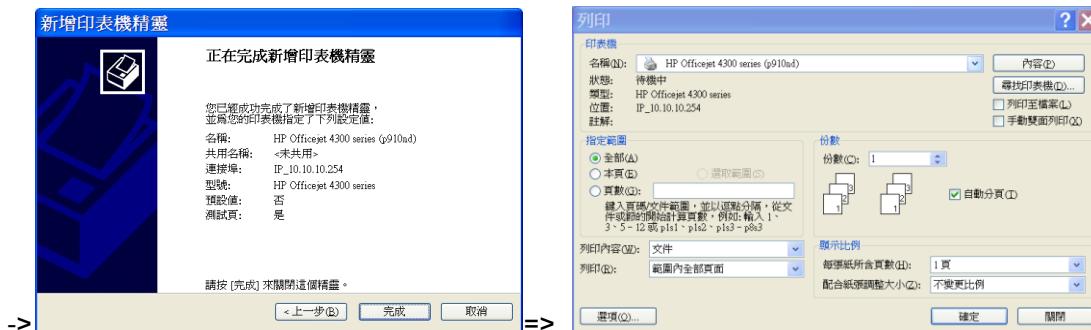
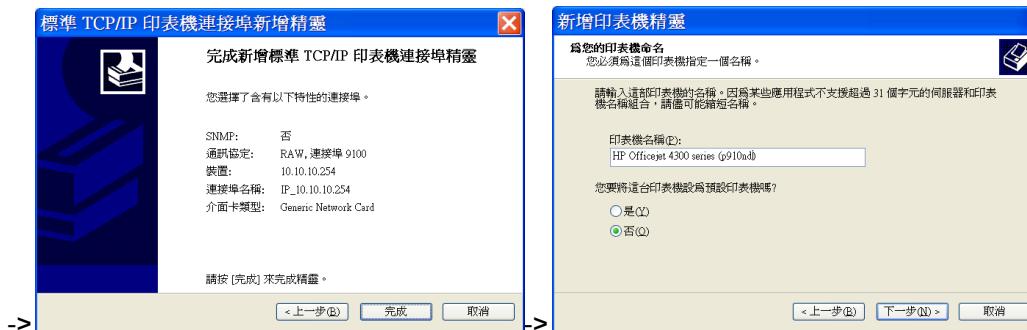
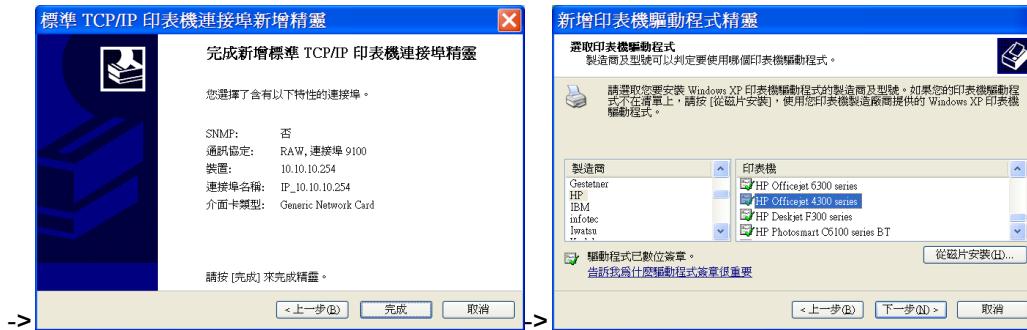
0x5411

Step3: run the printer daemon

```
# p910nd -f /dev/lp0
```

Step4: Setup the printer in Windows





11.28 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 <u>mr_autoneg_enable=1'b0</u>	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 <u>mr_autoneg_enable=1'b0</u>	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 ⁺	RO ⁺	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.

11.29 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.

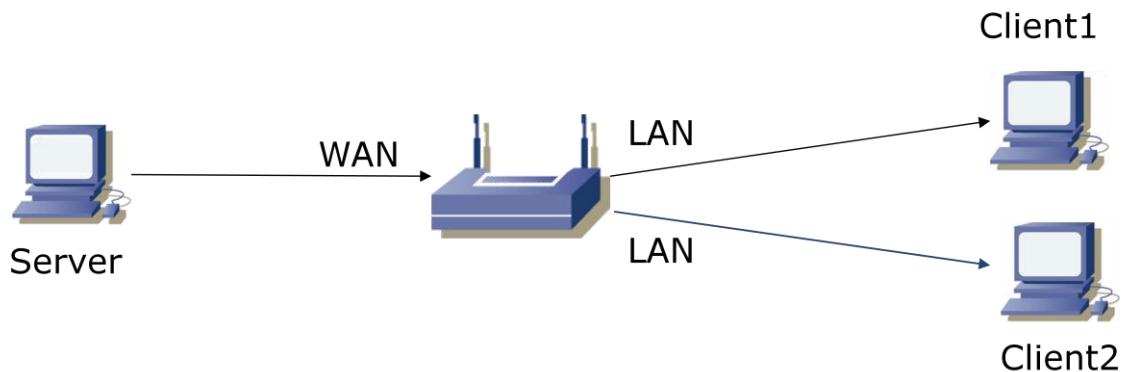
802.1d Spanning Tree	Disable <input type="button" value="▼"/>
LLTD	Disable <input type="button" value="▼"/>
IGMP Proxy	Enable <input type="button" value="▼"/>
UPNP	Disable <input type="button" value="▼"/>
Router Advertisement	Disable <input type="button" value="▼"/>
DNS Proxy	Disable <input type="button" value="▼"/>

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 Version 4, 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: FFFFFFFFFFFFFF0102030405060708090A0B0C0D0E0F10...
  [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.

11.30 EHCI/OHCI USB Power Saving

A potential issue may happen if user chooses a mixed version pair of SDK Linux and Uboot. A confirmed pair is RT3883/RT3662 SDK3.5 Uboot + SDK 3.4 Linux, this pair may cause system to freeze during boot up.

To reduce power consumption and lower the working temperature, SDK3.5 and later versions disable the USB power and clock gating during the boot-loader initialization stage. The advantage is more power-saving. The detail is SDK 3.5 Uboot would disable the USB HW module by default. And then the SDK 3.5 EHCI/OHCI Linux driver have to re-enable USB HW module before accessing USB related registers.

However, if user chooses an unexpected pair, ex. SDK3.5 Uboot + SDK 3.4 Linux, the system may freeze at OHCI initialization during boot up as following log. This is because the SDK 3.5(or later) Uboot would disable the USB HW module to save power, but then the older SDK Linux(SDK 3.4) EHCI/OHCI driver does not re-enable the USB HW module before accessing USB function.

```

...
rt3xxx-ohci rt3xxx-ohci: RT3xxx OHCI Controller
rt3xxx-ohci rt3xxx-ohci: new USB bus registered, assigned bus number 2
rt3xxx-ohci rt3xxx-ohci: irq 18, io mem 0x101c1000
      <<<< freeze at here

```

To solve the issue(freeze at OHCI init), please disable the USB(EHCI/OHCI) power saving feature in SDK 3.5(and later) Uboot as following:

In Uboot/lib_mips/board.c

```

void board_init_r(gd_t *id, ulong dest_addr)
{
...
//void config_usb_ehciohci(void);
//config_usb_ehciohci();
...
}

```

And then rebuild Uboot.

11.31 Auto-frequency and Power Saving

The RT3352/RT5350 SOC has the capability of auto-frequency and power saving.

- CPU Auto-Frequency (RT3352/RT5350)
- SDR Power Pre-charge Power Down (RT3352/RT5350)
- DDR self Refresh Power Save (RT3352)
- Ethernet Power Down (RT3352/RT5350)
- USB Power Down (RT3352/RT5350)
- WIFI Power Down (RT3352/RT5350)

Notice: Those new features are supported by SDK 3.5.2.0 and later version.

1. Setup

- How to turn on CPU Auto-Frequnecy

For RT3352/RT5350, We can turn on CPU auto frequency function by:

Modifying config.mk in Uboot and rebuild uboot firmware

```
...
```

```
RALINK_DDR_CONTROLLER_OPTIMIZATION = OFF
```

```
RALINK_CPU_AUTO_FREQUENCY = ON
```

```
RALINK_SDR_PRECHARGE_POWER_DOWN = OFF
```

```
RALINK_DDR_SELF_REFRESH_POWER_SAVE_MODE = OFF
```

```
...
```

```
$make
```

Set Linux Kernel Configuration and then rebuild linux firmware

```
$make menuconfig --->
```

Machine selection ---> [*] Ralink External Timer

```
.....  
$make dep; make
```

- How to turn on SDR Pre-charge Power Down

For RT3352/RT5350, We can turn on SDR power save by:

Modifying config.mk in Uboot and rebuild uboot firmware

```
...
```

```
RALINK_DDR_CONTROLLER_OPTIMIZATION = OFF
```

```
RALINK_CPU_AUTO_FREQUENCY = OFF
```

```
RALINK_SDR_PRECHARGE_POWER_DOWN = ON
```

```
RALINK_DDR_SELF_REFRESH_POWER_SAVE_MODE = OFF
```

...

\$make

- How to turn on DDR Self Refresh Power Save

For RT3352 , We can turn on DDR power save by:

Modifying config.mk in Uboot and rebuild uboot firmware

...

```
RALINK_DDR_CONTROLLER_OPTIMIZATION = OFF
```

```
RALINK_CPU_AUTO_FREQUENCY = OFF
```

```
RALINK_SDR_PRECHARGE_POWER_DOWN = OFF
```

```
RALINK_DDR_SELF_REFRESH_POWER_SAVE_MODE = ON
```

...

\$make

2. Setup in script

...

```
/sbin/config-powersave.sh cpu 1      - enable CPU autofrequency
```

```
/sbin/config-powersave.sh cpu 0      - disable CPU autofrequency
```

```
/sbin/config-powersave.sh sdr 1      - enable SDR precharge powersave
```

```
/sbin/config-powersave.sh sdr 0      - disable SDR precharge powersave
```

```
/sbin/config-powersave.sh ddr 1      - enable DDR self auto refresh
```

```
/sbin/config-powersave.sh ddr 0      - disable DDR self auto refresh
```

```
/sbin/config-powersave.sh ethernet 1 [port] - enable Ralink ESW PHY powerdown
```

```
/sbin/config-powersave.sh ethernet 0 [port]      - disable Ralink ESW PHY powerdown  
/sbin/config-powersave.sh usb 1                  - enable usb powerdown  
/sbin/config-powersave.sh usb 0                  - disable usb powerdown  
/sbin/config-powersave.sh wireless 1            - enable wireless powerdown  
/sbin/config-powersave.sh wireless 0            - disable wireless powerdown  
...  
...
```

- How to turn on CPU Auto-Frequency

For RT3352/RT5350, We can turn on CPU auto frequency function by:

```
#config-powersave.sh cpu 1
```

- How to turn on SDR Pre-charge Power Down

For RT3352/RT5350, We can turn on SDR power save by:

```
#config-powersave.sh sdr 1
```

- How to turn on DDR Self Refresh Power Save

For RT3352 , We can turn on DDR power save by:

```
#config-powersave.sh ddr 1
```

- How to turn on Ethernet Power Down

For RT3352 /RT5350, We can turn on Ethernet port#3 power down by:

```
#config-powersave.sh ethernet 1 3
```

- How to turn on USB Power Down

For RT3352 /RT5350, We can turn on USB power down by:

```
#config-powersave.sh usb 1
```

- How to turn on WIFI Power Down

For RT3352 /RT5350, We can turn on WIFI power down by

```
#config-powersave.sh wifi 1
```

3. Check Function

- CPU Auto-Frequency

Turn off:

```
#  
# reg s b0000000  
switch register base addr to 0xb0000000  
# reg r 40  
0x34501  
#
```

Turn on:

```
# reg s b0000000  
switch register base addr to 0xb0000000  
# reg r 40  
0x80035f41  
#
```

- SDR Pre-charge Power Save

Turn off:

```
# reg s b0000300  
switch register base addr to 0xb0000300  
# reg r 1c  
0x3ffff  
# reg r 4  
0xe1110600  
#
```

Turn on:

```
# reg s b0000300  
switch register base addr to 0xb0000300  
# reg r 1c  
0x1  
# reg r 4  
0xf1110600  
#
```

- DDR Self Refresh Power Save

Turn off:

```
# reg s b0000300  
switch register base addr to 0xb0000300  
# reg r 1c  
0x3ffff  
# reg r 18  
0x3  
#
```

Turn on:

```
# reg s b00000300
switch register base addr to 0xb00000300
# reg r 1c
0x6d000001
# reg r 18
0x13
#
```

11.32 Concurrent AP porting Guide

The APSOC has the capability of working 1st wireless interface and 2nd wireless interfaces concurrently.

A. The interface1 (ra0)

B. The interface 2 (rai0)

Station can associate and execute WPS connection for any wireless interface. Moreover, user can configure the settings of any wireless interface by Web GUI.

You can refer to Ralink_AP_SDK_User's_Manual for the Detail information.

1. Setup:

If your SDK does not include RT309x/RT539x/RT3572/RT5572/RT5592/RT3593 support, please refer the following steps to install it.

Requirement:

- RT288x_SDK
- RT3090/RT5392/RT3572/RT5572/RT5592/RT3593 WiFi driver
- RT3090/RT5392/RT3572/RT5572/RT5592/RT3593 EEPROM binary files

Procedure: (RT3883+RT3090 as example)

Step1.

Please copy RT309x WiFi driver to RT288x_SDK/linux-2.6.xx.x/drivers/net/wireless

ex:

```
$cp RT3090_ap RT288x_SDK/linux-2.6.xx.x/drivers/net/wireless
```

Step2.

Please modify RT288x_SDK/linux-2.6.xx.x/drivers/net/wireless/Makefile

ex:

```
...
```

```

ifneq ($(CONFIG_RT2860V2_AP),)
obj-$(CONFIG_RT2860V2_AP)      += rt2860v2_ap/
endif
ifneq ($(CONFIG_RT2860V2_STA),)
obj-$(CONFIG_RT2860V2_STA)      += rt2860v2_sta/
endif
ifneq ($(CONFIG_RT3090_AP),)
obj-$(CONFIG_RT3090_AP)          += RT3090_ap/
endif
...

```

Step3.

Please modify RT288x_SDK/linux-2.6.xx.x/ralink/Kconfig

ex:

```

...
#source "drivers/net/wireless/rt2860v2_st/Kconfig"

#source "drivers/net/wireless/rt2860v2_apsta/Kconfig"
source "drivers/net/wireless/RT3090_ap/Kconfig"

config RT3090_AP_RF_OFFSET
    depends on RT3090_AP
    hex
    default 0x48000
...

```

Step4.

If wifi driver support ***FLASH_SUPPORT***, please copy EEPROM binary file to
RT288x_SDK/source/vendors/Ralink/RT3883

ex:

```
$cp RT3092_PCIE_LNA_2T2R_ALC_V1_2.bin
RT288x_SDK/source/vendors/Ralink/{RT3883/RT3352/RT5350}
```

Step5.

Please modify RT288x_SDK/source/vendors/Ralink/RT3883/Makefile

ex:

```

...
$(ROMFSINST) -e CONFIG_RALINK_RT3883_3T3R RT2860_default_novlan_3s
/etc_ro/Wireless/RT2860AP/RT2860_default_novlan
$(ROMFSINST) -e CONFIG_RALINK_RT3883_3T3R RT2860_default_vlan_3s
/etc_ro/Wireless/RT2860AP/RT2860_default_vlan
```

```

$(ROMFSINST) -e CONFIG_RALINK_RT3662_2T2R
/etc_ro/Wireless/RT2860AP/RT2860_default_novlan
$(ROMFSINST) -e CONFIG_RALINK_RT3662_2T2R
/etc_ro/Wireless/RT2860AP/RT2860_default_vlan

$(ROMFSINST) -e CONFIG_RT3090_AP /etc_ro/Wireless/iNIC/RT2860AP.dat
$(ROMFSINST) -e CONFIG_RT3090_AP
/etc_ro/Wireless/RT2860AP/RT3092_Pcie_LNA_2T2R_ALC_V1_2.bin
...

```

Step6.

Please modify RT288x_SDK/source/user/rt2880_app/scripts/internet.sh

ex:

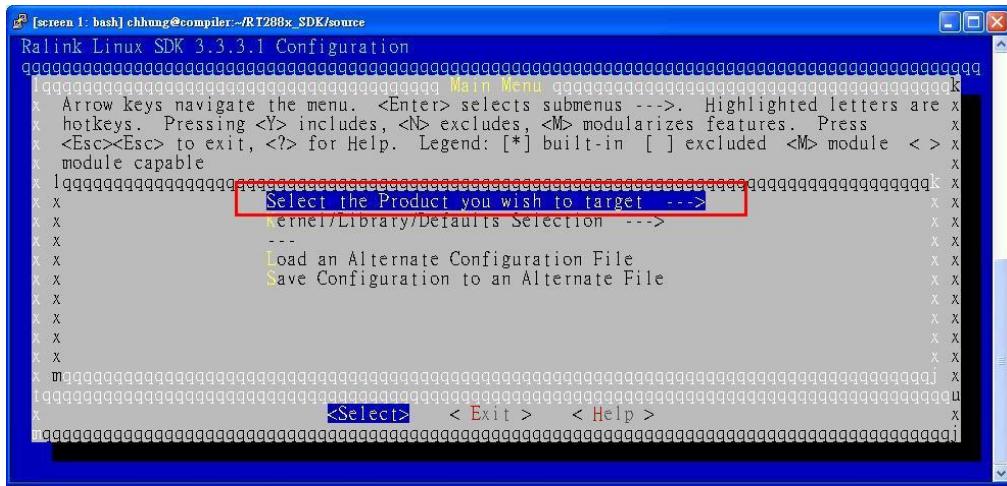
```

...
ifRaxWdsxDown
if [ "$CONFIG_RTDEV" != "" -o "$CONFIG_RT2561_AP" != "" ]; then
    ifRaixWdsxDown
fi
if [ "$CONFIG_RT2860V2_AP" != "" ]; then
    rmmod rt2860v2_ap_net
    rmmod rt2860v2_ap
    rmmod rt2860v2_ap_util
fi
if [ "$CONFIG_RT2860V2_STA" != "" ]; then
    rmmod rt2860v2_sta_net
    rmmod rt2860v2_sta
    rmmod rt2860v2_sta_util
fi
if [ "$RT2880v2_INIC_PCI" != "" ]; then
    rmmod iNIC_pci
fi
if [ "$CONFIG_RT3090_AP" != "" ]; then
    rmmod RT3090_ap_net
    rmmod RT3090_ap
    rmmod RT3090_ap_util
fi
...
# RTDEV_PCI support
if [ "$RT2880v2_INIC_PCI" != "" ]; then
    insmod -q iNIC_pci
fi
if [ "$CONFIG_RT3090_AP" != "" ]; then
    insmod -q RT3090_ap_util
    insmod -q RT3090_ap
    insmod -q RT3090_ap_net
fi
...

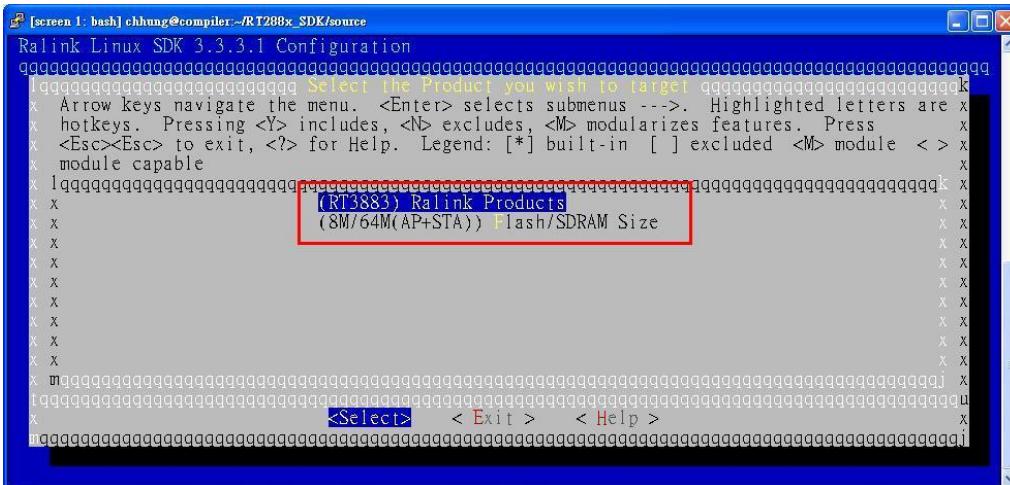
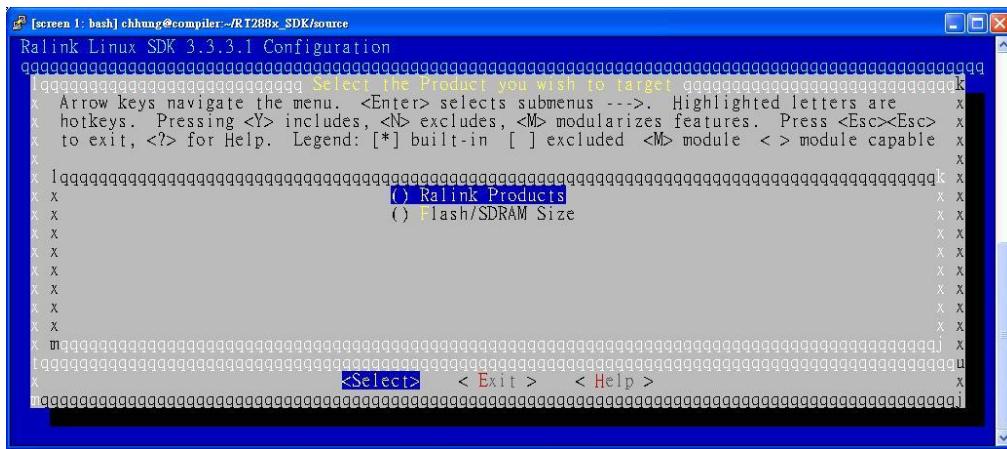
```

Step7.

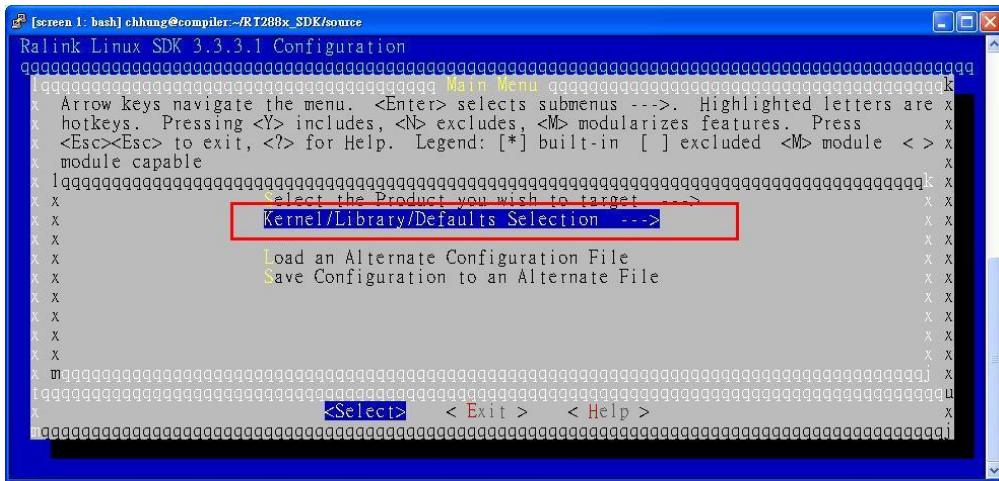
You must switch directory to RT2888x_SDK/source and execute “make menuconfig,” like below:



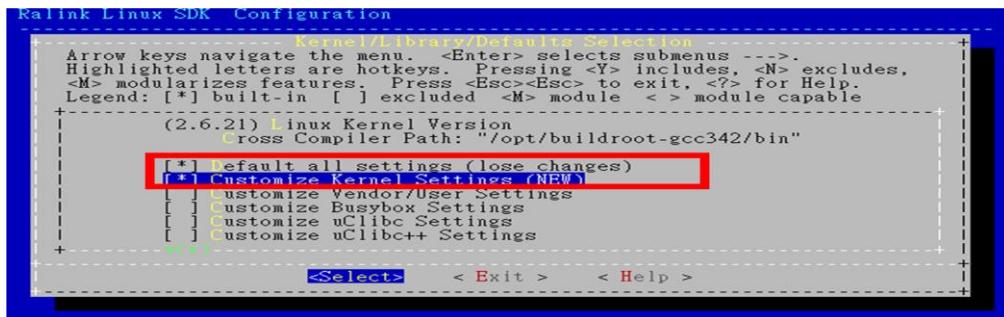
Please choose “Select the Product you wish to target” option to configure the main settings of your target platform. **<<Please select 4M/32M or 8M/64M Flash/SDRAM size>>**



And then, please exit “Select the Product you wish to target” option and enter “Kernel/Library/Defaults Selection” option.



You must select “Default all settings” option to load default configuration first and select “Customize Kernel Settings” options to turn on 2nd interface.



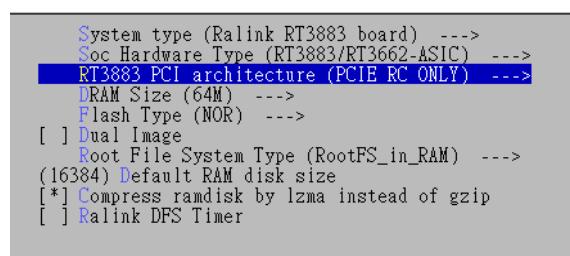
Exit ---> yes.

Step8.

After load default, you can enter kernel configured main menu.

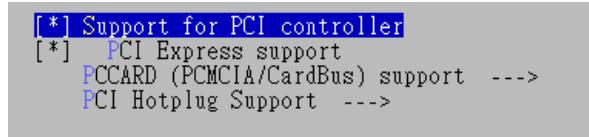
If 2nd wireless uses PCIE interface:

Please enter “Machine selection” and choice “RT3883 PCI architecture” to “PCIE RC ONLY” mode.



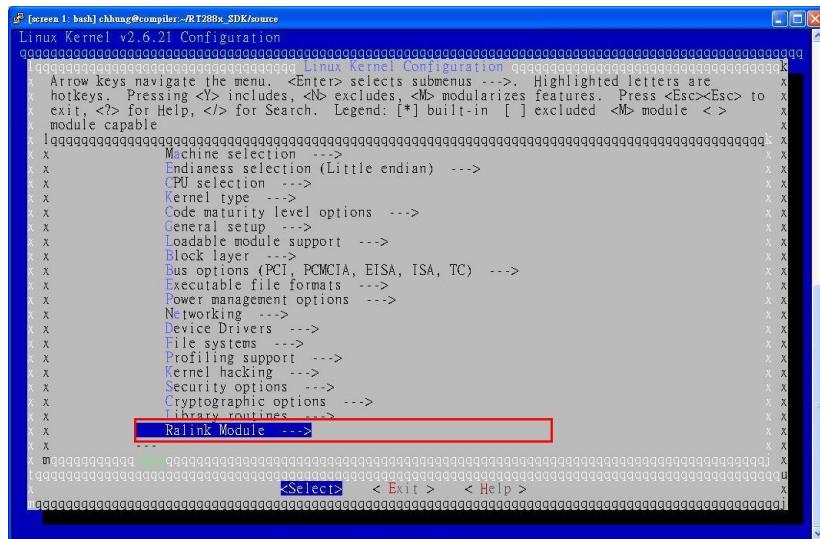
Leave “Machine selection” option.

Please enter “Bus options (PCI, PCMCIA, EISA, ISA, TC)” option and check whether PCI/PCIE support or not, like below:

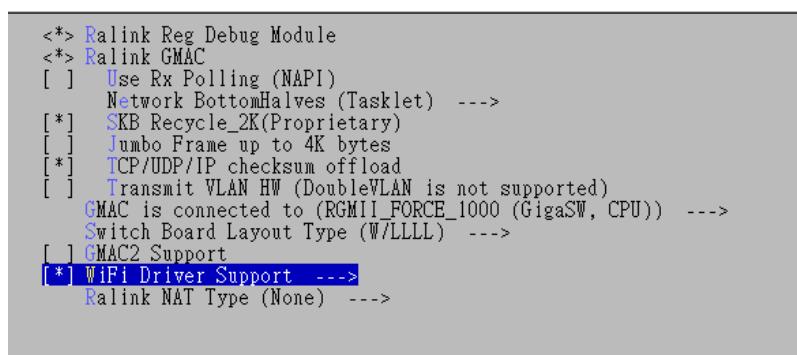


Leave “Bus options (PCI, PCMCIA, EISA, ISA, TC)” option.

Please enter “Ralink Module” option



You must enter “WiFi Driver Support” and select RT3090 module to act 2nd WiFi interface, but one of them could be selected.



```

--- WiFi Driver Support
    RF Type (2T3R (RT3662)) --->
<*>   Ralink RT2860 802.11n AP support
[*]     LED Support
[*]     WSC (WiFi Simple Config)
[*]         WSC 2.0(WiFi Simple Config 2.0)
[*]     LLTD (Link Layer Topology Discovery Protocol)
[*]     WDS
[*]     Nintendo (NEW)
[*]     MBSSID
[*]         New MBSSID MODE (NEW)
[*]     AP-Client Support
[*]     IGMP snooping
[*]     NETIF Block (NEW)
[*]     DFS (NEW)
[*]     Carrier Detect (NEW)
[*]     DLS ((Direct-Link Setup) Support
[*]     IDS (Intrusion Detection System) Support (NEW)
[*]     MESH Support (NEW)
[*]     WAPI Support (NEW)
[*]     Green AP Support (NEW)
[*]     Memory Optimization (NEW)
[*]     Video Turbine support (NEW)
[*]     802.11n Draft3
[*]     Intelligent Rate Adaption (NEW)
[*]     Tx Beam Forming Support (Only 3883) (NEW)
<M>   Ralink RT3090 802.11n AP support
[*]     Flash Support
[*]     LED Support
[*]     WSC (WiFi Simple Config)
[*]     LLTD (Link Layer Topology Discovery Protocol)
[*]     WDS
[*]     MBSSID
[*]         NETIF Block (NEW)
[*]     DLS ((Direct-Link Setup) Support
[*]     IDS (Intrusion Detection System) Support (NEW)
[*]     802.11n Draft3

```

Leave “Ralink Module” option and then exit “Linux Kernel Configuration”.

Exit ---> yes

Step9.

Final, you can execute “**make dep**” and “**make**” to build image under the RT288x_SDK/source.

```
$make dep
```

```
$make
```

2. Usage:

If the firmware is built successfully, you can upgrade it into your RT3052/RT3883/RT3352/RT5350/RT6855/RT6856 reference board by TFTP Server or Web GUI.

After firmware upgrade, you can use Web GUI or command line to check if two wifi interfaces are successfully inserted and brought up or not.

- Web GUI



- Command line:

1st WiFi interface name: ra0

2nd WiFi interface name: rai0

ra0's profile is located on /etc/Wireless/RT2860/RT2860.dat and rai0's profile is located on /etc/Wireless/iNIC/iNIC_ap.dat. To change rai0's wireless settings, you can edit its profile and re-bring up it, like ra0.

Certainly, ra0 and rai0 are shown their detail information or configured settings by iwpriv command, for example:

ra0:

```
#iwprive ra0 set SSID=RT3883AP
```

```
#iwprive ra0 stat
```

```
...
```

rai0:

```
#iwprive rai0 set SSID=RTDEV_AP
```

```
#iwprive rai0 stat
```

```
...
```

11.33 SuperDMZ usage guide

Usage:

```
super_dmz [-f] [-l lan_ifname] [-w wan_ifname] [-t tcp_port] [-t tcp_port1:tcp_port2] [-u udp_port] [-u  
udp_port1:udp_port2]
```

-f : flush & clear super_dmz functions from system.

-l lan_ifname: Explicitly assign the LAN interface name, ex. “br0” or “eth2.2”. In Ralink SDK this argument is assigned automatically based on the current operation mode if it is not assigned explicitly.

-w wan_ifname: Explicitly assign the WAN interface name, ex “eth2.2” or “ppp0”. In Ralink SDK this argument is assigned automatically based on the current WAN mode if it is not assigned explicitly.

-t tcp_port: TCP port tcp_port is the exception of DMZ forwarding, ex “80” or “23”. The most case here is “80” for AP web remote access.

-t tcp_port1:tcp_port2 : TCP port from tcp_port1 to tcp_port2 is the exception of DMZ forwarding.

-u udp_port: UDP port udp_port is the exception of DMZ forwarding.

-u udp_port1:udp_port2 : UDP port from udp_port1 to udp_port2 is the exception of DMZ forwarding.

Example:

1) # super_dmz -f

Clear Super DMZ function from system.

2) # super_dmz

Enable Super DMZ function.

3) # super_dmz -l eth0 -t 80

Enable Super DMZ function. Assign “eth0” as LAN interface. Avoid tcp port 80 is forwarding.(To make web server on router reachable from WAN side)

4) # super_dmz -w eth2 -t 45:123 -t 3128 -u 10000 -u 500:600

Enable Super DMZ function. Assign “eth2” as WAN interface. Avoid tcp port 45 to 123, tcp port 3128, udp port 10000, and udp port 500 to 600 are forwarding.

Implementation note:

1. When

1) system boot up

2) WAN IP is acquired or changed (Ex. PPPoE or DHCP on WAN)

3) Virtual Server(Port forwarding) settings change

the super_dmz have to re-run:

```
# super_dmz -f
```

```
# super_dmz
```

11.34 How to support IPv6 Ready Logo

The IPv6 Forum (<http://www.ipv6forum.com>) IPv6 Ready Logo Program is a conformance and interoperability testing program intended to increase user confidence by demonstrating that IPv6 is available now and is ready to be used.

To pass Ipv6 Ready Logo (Phase-2), (<https://www.ipv6ready.org/?page=phase-2-tech-info>),

IPv6 Ready Logo Phase-2

(Test Specifications and Test Tools)

Target

- IPv6 Core Protocols
 - Router
 - Host
- DHCPv6
 - Client
 - Server
 - Relay agent
- IPsec

please enable additional three applications:

- ecmh

Easy Cast du Multi Hub (ecmh) is a networking daemon that acts as a full IPv6 MLDv1 and MLDv2 Multicast "Router".

\$ make menuconfig

[*] Customize Vendor/User Settings

Miscellaneous Applications --->

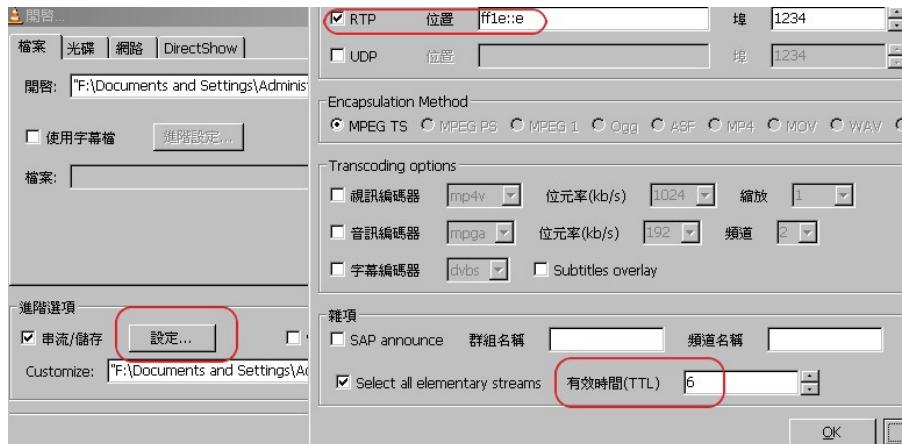
```
[*] language pack - Simplified Chinese
[*] ecmh (IPv6 multicast forwarding/MLD daemon)
[*] igmp proxy (RFC4605)
[*] inadyn (DDNS Client)
```

Exit ---> Yes

[Example]

1. #ecmh -e -i eth2.2

2. VLC server 0.8.6d



3. VLC client 2.0.5 =>open network streaming



4. dump switch table and see join port is at port#3

```
# switch dump
hash port(0:6) fid vid age mac-address
068: ---1 --- 0 1 149 f0def16a81d5
104: ----1--- 0 2 149 002215be842e
544: ----1-1- 0 2 255 3333000000001
068: ---1 --- 0 1 149 f0def16a81d5
104: ----1--- 0 2 149 002215be842e
544: ----1-1- 0 2 255 3333000000001
548: 1111 ---| 0 1 255 3333000000001
574: ---1 --1- 0 1 255 333300000000e
578: ----1-1- 0 2 255 333300000000e
600: -1-1 --1- 0 1 255 01005e7ffffa
60c: ----1-1- 0 2 255 01005e7ffffa
```

snooping is working

- **ip** command in iproute2

to flush neighbor cache during running test log

\$ make menuconfig

[*] Customize Vendor/User Settings

Network Applications --->



Exit ---> Yes

- **radvd**

radvd, the Router Advertisement Daemon, is an open-source software product that implements link-local advertisements of IPv6 router addresses and IPv6 routing prefixes using the Neighbor Discovery Protocol (NDP) as specified in RFC 2461.^[2] The Router Advertisement Daemon is used by system administrators in stateless autoconfiguration methods of network hosts on Internet Protocol version 6 networks.

```
$ make menuconfig
```

[*] Customize Vendor/User Settings

Network Applications --->

```
[ ] ip-12tp
[*] radvd (Router Advertisement Daemon)
[ ] radvd dump
[*] rt2860apd (802.1x Authenticator)
[ ] rt61apd (Legacy 802.1x Authenticator)
```

Exit ---> Yes

11.35 How to enable iPerf tool

iPerf was developed by NLANR/DAST as a modern alternative for measuring maximum TCP and UDP bandwidth performance. iPerf allows the tuning of various parameters and UDP characteristics. iPerf reports bandwidth, delay jitter, datagram loss.

```
$ make menuconfig
```

[*] Customize Vendor/User Settings

Miscellaneous Applications --->

```
[ ] ixia Endpoint
[*] iperf
[ ] lmbench
[*] mtd write
[ ] mpstat
[ ] netcat
[ ] netstat-nat
```

Exit ---> Yes

Usage:

Server side: iperf -s

Client side: iperf -c [server's ip] -w 128k -t 30 -i 10

11.36 How to enable ebtables

The ebtables program is a filtering tool for a Linux-based bridging firewall. It enables transparent filtering of network traffic passing through a Linux bridge.

```
$ make menuconfig
```

```
[*] Customize Vendor/User Settings
```

```
Network Applications -->
```

```
[*] dnsmasq (DNS forwarder, DHCP server)
[ ] disktype(detect format of a disk)
[ ] echo server
[*] ebtables
[ ] storage(enable chmod, fdisk in busybox)
[*] go-ahead webserver
[ ]   enable IPv6 support
[ ]   enable SSL support
[ ]   enable hostname support
[ ]   enable GreenAP support (enable crond in busybox)
[ ]   enable Wizard support
```

Exit --> Yes

Usage:

If router would like to block all packets of a host from intranet to internet:

```
# ebtables -A FORWARD -s [host' MAC address] -j DROP
```

Or

```
# ebtables -A FORWARD -p IPv4 --ip-src [host' IP address] -j DROP
```

11.37 How to enable IPv6 Rapid Deployment (6rd)

To enable IPv6 Rapid Deployment (6rd), please include ipv6 6rd feature support in the kernel:

```
# make menuconfig
```

[*] Customize Kernel Settings

In the kernel settings, find “The IPv6 protocol” by select the following options:

[*] Networking support --->

 Networking options --->

 <*> The IPv6 protocol --->

 <*> IPv6: IPv6-in-IPv4 tunnel (SIT driver)

 [*] IPv6: IPv6 Rapid Deployment (6RD) (EXPERIMENTAL)

Please check both “IPv6: IPv6-in-IPv4 tunnel (SIT driver)” and “IPv6: IPv6 Rapid Deployment (6RD) (EXPERIMENTAL)”.

To enable Ipv6 6rd, the firmware should also support iproute2 utility:

[*] Customize Vendor/User Settings

Network Applications --->

[*] iproute2

[*] ip



After compile and download the firmware, please use iproute2's ip command to configure the IPv6 6rd function:

```
ip tunnel add <6rd if name> mode sit local <WAN ipv4 address> ttl <ttl>
```

```
ip tunnel 6rd dev <6rd if name> 6rd-prefix <ISP's 6rd prefix>
```

```
ip addr add <6rd ipv6 address> dev <6rd if name>
```

```
ip link set <6rd if name> up
```

```
ip route add ::/0 via ::<ISP's 6rd border router ipv4 address> dev <6rd if name>
```

Note: the <6rd ipv6 address> should be generated from <ISP's 6rd prefix> and <WAN IPv4 address>, for example, if ISP's prefix is 2001:aaaa/32, and WAN ipv4 address is 100.1.1.1, then the 6rd address could be

2001:aaaa:6401:101::1/32

to add LAN ipv6 address, you can use the following command:

```
ip addr add <LAN ipv6 addr> dev <LAN if name>
```

Note: the LAN ipv6 address should be same as 6rd's ipv6 address, except address mask, for example, in above case, the LAN ipv6 address will be

2001:aaaa:6401:101::1/64

to enable ipv6 forwarding, please use this command:

```
echo "1" > /proc/sys/net/ipv6/conf/all/forwarding
```

The following figure shows an example that configures IPv6 6rd:

```
# ip tunnel add 6rdtun mode sit local 111.80.78.220 ttl 64
# ip tunnel 6rd dev 6rdtun 6rd-prefix 2001:e41::/32
# ip addr add 2001:e41:6f50:4edc::1/32 dev 6rdtun
# ip link set 6rdtun up
# ip route add ::/0 via ::61.211.224.125 dev 6rdtun
# ip addr add 2001:e41:6f50:4edc::1/64 dev br0
# echo "1" > /proc/sys/net/ipv6/conf/all/forwarding
#
```

```
#
#
# ifconfig 6rdtun
6rdtun      Link encap:IPv6-in-IPv4
              inet6 addr: 2001:e41:6f50:4edc::1/32 Scope:Global
              inet6 addr: ::111.80.78.220/128 Scope:Compat
              UP RUNNING NOARP MTU:1480 Metric:1
              RX packets:0 errors:0 dropped:0 overruns:0 frame:0
              TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:0
              RX bytes:0 (0.0 B)   TX bytes:0 (0.0 B)

#
# ifconfig br0
br0          Link encap:Ethernet HWaddr 00:0C:43:43:63:F3
              inet addr:10.10.10.254 Bcast:10.10.10.255 Mask:255.255.255.0
              inet6 addr: 2001:e41:6f50:4edc::1/64 Scope:Global
              inet6 addr: fe80::20c:43ff:fe43:63f3/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
              RX packets:81 errors:0 dropped:0 overruns:0 frame:0
              TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:0
              RX bytes:8771 (8.5 KiB)   TX bytes:1072 (1.0 KiB)
#
```

This example has a WAN IPv4 address=111.80.78.220 and 6rd-prefix=2001:e41::/32, and ISP's border server ipv4 address is 61.211.224.125

User also configures IPv6 RD settings via Web GUI:

IPv6 Setup

IPv6 Connection Type	
IPv6 Operation Mode	Tunneling Connection (6RD)
Tunneling Connection (6RD) Setup	
ISP 6rd Prefix / Prefix Length	2001:e41 / 32
ISP Border Relay IPv4 Address	61.211.224.125
<input type="button" value="Apply"/> <input type="button" value="Cancel"/>	

11.38 How to enable IPv6 DS-Lite

To enable IPv6 DS-Lite, please include ipv6 6rd feature support in the kernel:

```
# make menuconfig
```

[*] Customize Kernel Settings

In the kernel settings, find “The IPv6 protocol” by select the following options:

[*] Networking support --->

 Networking options --->

 <*> The IPv6 protocol --->

 <*> IPv6: IP-in-IPv6 tunnel (RFC2473)

Please check “IPv6: IPv6: IP-in-IPv6 tunnel (RFC2473)”.

To enable Ipv6 DS-Lite, the firmware should also support iproute2 utility:

[*] Customize Vendor/User Settings

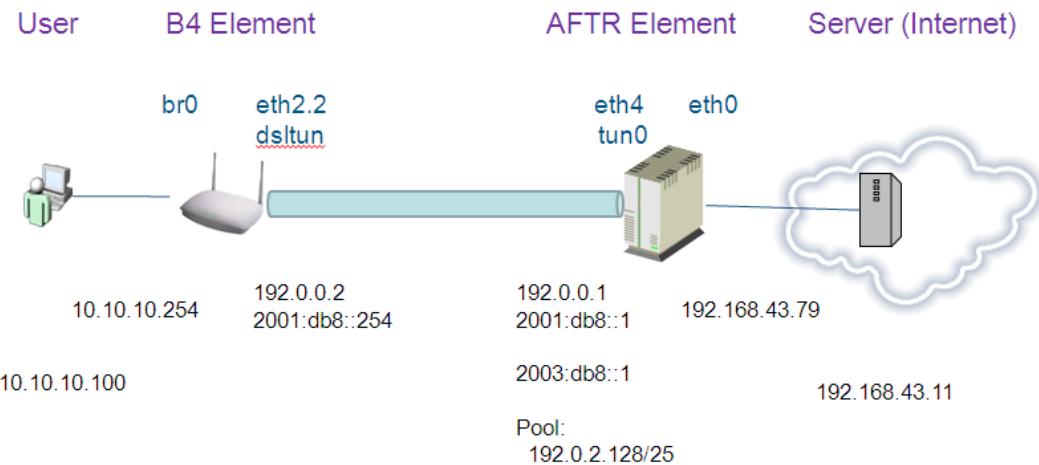
 Network Applications --->

 [*] iproute2

 [*] ip



After compiling and downloading the firmware, please use iproute2's ip command to configure the IPv6 DS-Lite function:



- Configuration on B4 Element

#IPv6 Address

```
ip -6 addr add 2001:db8::254/32 dev eth2.2
```

#IPv6 Routing

```
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

#Create DS-Lite Interface

```
ip -6 tunnel add dsltun mode ipip6 remote 2001:db8::1 local 2001:db8::254 dev eth2.2
```

```
ip link set dev dsltun up
```

adds the IPv4 default route to the server to forward all IPv4 packets to the ds-lite interface dsltun

```
ip route add default dev dsltun
```

#IPv6 Default Route

```
ip -6 route add default dev eth2.2
```

#Static IPv6 Route

```
ip -6 route add 2001:db8::1/128 via 2003:db8::1
```

- Configuration on AFTR (<http://www.isc.org/software/aftr>)

#IPv6 Address & Routing

```
ip -6 addr add 2003:db8::1/32 dev eth4
```

```
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

#Tunnel Interface Configuration (aftr.conf)

```
ip addr add 192.0.0.1 peer 192.0.0.2 dev tun0
```

```
ip route add 192.0.2.128/25 dev tun0
```

```
ip -6 addr add fe80::1 dev tun0
```

```
ip -6 route add 2001:db8::1 dev tun0
```

#Routing to B4 Element

```
ip -6 route add 2001:db8::254/128 dev eth4
```

#NAT

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 192.168.43.79
```

```
iptables -t nat -A PREROUTING -i eth0 -j DNAT --to-dest 192.0.2.1
```

aftr.conf

```
default tunnel mss on
defmtu 1450
address endpoint 2001:db8::1
address icmp 198.18.200.10
pool 192.0.2.128
acl6 ::0/0
```

aftr-script

```
aftr_start() {  
  
    set -x  
  
    ip link set tun0 up  
  
    ip addr add 192.0.0.1 peer 192.0.0.2 dev tun0  
  
    ip route add 192.0.2.128/25 dev tun0  
  
    ip -6 addr add fe80::1 dev tun0  
  
    ip -6 route add 2001:db8::1 dev tun0  
  
}  
  
aftr_stop() {  
  
    set -x  
  
    ip link set tun0 down  
  
}
```

Another, user could use Web GUI to set DS-Lite:

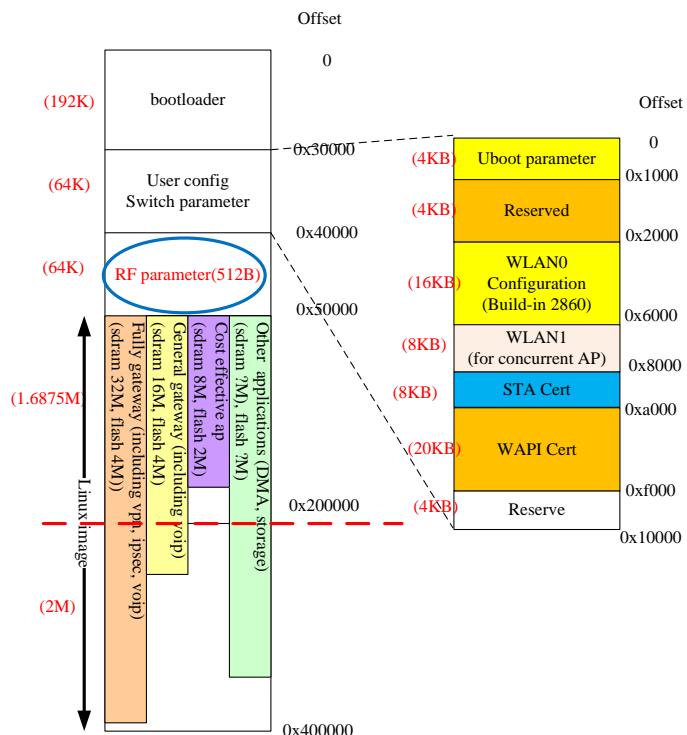
IPv6 Setup

IPv6 Connection Type	
IPv6 Operation Mode	Tunneling Connection (DS-Lite) <input type="button" value="▼"/>
Tunneling Connection (DS-Lite) Setup	
WAN IPv6 Address	2001:db8::254
AFTR Server IPv6 Address	2001:db8::1
Gateway IPv6 Address	2003:db8::1
<input type="button" value="Apply"/> <input type="button" value="Cancel"/>	

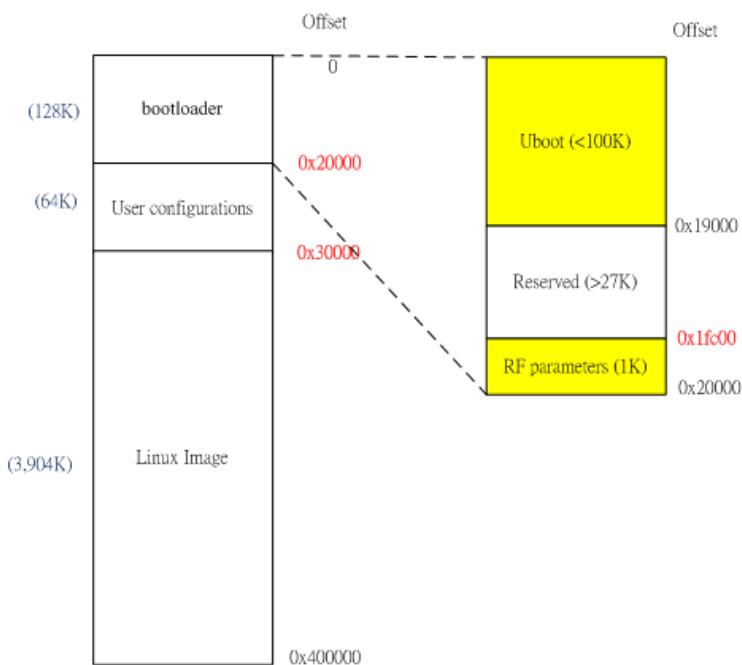
11.39 How to modify flash layout

Basically, you can make uboot and RF parameters use same flash sector t and it can save up to 3 flash sector compared to original design.

Default Flash layout:



New flash layout:



1. modify rt2860v2/ os/linux/rt_linux.c to set flash partition name.

```
ra_mtd_read_nm("Bootloader", a&0xFFFF, (size_t) b, p);
```

2. Modify rt2860v2/include iface/rtmp_rbs.h to set the offset of RF parameter.

```
#define RF_OFFSET 0x1FC00 //last 1Kbyte in flash sector 1
```

3. Modify raeth/raether.c

```
i = ra_mtd_read_nm("Bootloader", GMAC0_OFFSET, 6, addr.sa_data);
i = ra_mtd_read_nm("Bootloader", GMAC2_OFFSET, 6, addr.sa_data);
```

4. Modify raeth/raether.h

```
#define GMAC2_OFFSET 0x1FC22
#if ! defined (CONFIG_RALINK_RT6855A)
#define GMAC0_OFFSET 0x1FC28
#else
#define GMAC0_OFFSET 0x1FC00
#endif
#define GMAC1_OFFSET 0x1FC2E
```

5. Modify lib/libnvram/flash_api.c

```
int flash_read_mac(char *buf)
{
    int fd, ret;
    if (!buf)
        return -1;
    fd = mtd_open("Bootloader", O_RDONLY);
    if (fd < 0) {
        fprintf(stderr, "Could not open mtd device\n");
        return -1;
    }
#ifndef NO_WIFI_SOC
    lseek(fd, 0x1FC2E, SEEK_SET);
#else
    lseek(fd, 0x1FC06, SEEK_SET);
#endif
}
```

```
    ret = read(fd, buf, 6);
    close(fd);
    return ret;
}
```

6. Modify drivers/mtd/maps/ralink-flash.h

```
#define MTD_BOOT_PART_SIZE      0x20000
#define MTD_CONFIG_PART_SIZE     0x10000
#define MTD_FACTORY_PART_SIZE    0x00000
```

7. Modify drivers/mtd/ralink/ralink_spi.c , drivers/mtd/maps/ralink-flash.c, drivers/mtd/ralink/ralink_nand.c, or drivers/mtdralink/ralink_nand_rt3052.c which depended on your flash type.

```
#if 0
{
    name:      "Factory",
    size:      MTD_FACTORY_PART_SIZE,
    offset:    MTDPART_OFS_APPEND
#endif
```

8. Modify Uboot/ include/configs/rt2880.h

```
#define CFG_BOOTLOADER_SIZE      0x20000
#define CFG_FACTORY_SIZE          0x000
```

11.40 How to reduce Linux FW size.

1. Modify vendors/config/mips/config.arch

```
CFLAGS := $(if $(LOPT),$(LOPT),-Os) -fomit-frame-pointer  
CFLAGS := $(if $(UOPT),$(UOPT),-Os) -fomit-frame-pointer
```

2. Modify Kernel configuration to save Linux FW size.

General setup --->

[*] Optimize for size (Look out for broken compilers!)

--- Configure standard kernel features (for small systems) --->

[] Enable printk function in the kernel

3. Use miniupnpd instead of linux-igd & wscd.

[] Customize Vendor/User Settings (NEW)

Network Applications --->

[*] miniupnpd

[] linux-igd

[] wscd (WSC/WPS)

4. Modify user/rt2880_app/switch/switch.c or gsw_switch.c

Change all keyword from CONFIG_RT2860V2_AP_MEMORY_OPTIMIZATION to
CONFIG_CC_OPTIMIZE_FOR_SIZE

5. Modify user/rt2880_app/nvram/ralink_init.c

Change all keyword from CONFIG_RT2860V2_AP_MEMORY_OPTIMIZATION to
CONFIG_CC_OPTIMIZE_FOR_SIZE

6. Modify user/wireless_tools/Makefile

```
BUILD_STATIC = y  
BUILD_WE_ESSENTIAL = y
```

11.4.1 How to change internal GSW PHY Base Address.

1. Please change internal PHY base address to > 5

RT6855/6: set 0xbfb5f014 = 0x10000c

RT7620: set 0xb0117014= 0x10000c

GPC: GIGA Port Control (offset: 0x7014)

			Internal EPHY Based Address	
20:16 ^w	RW ^w	PHY_BASE ^w	The base PHY address of the internal 5-port EPHY can be assigned by this register value. When you change the default value, you need to reset EPHY again to get the new register value. ^w	0x0 ^w

2. Reset PHY

RT6855/6: set 0xbfb00834[24]=1 then set 0xbfb00834[24]=0

RT7620: set 0xb0000034[24]=1 then set 0xb0000034[24]=0

RSTCTRL2 (offset:0x834)

24	RW	EPHY_RST	Write 1 to this bit will reset Ethernet PHY block Write 0 to de-assert reset.	0x0
----	----	----------	--	-----

11.42 How to support new USB 3G dongle

- Step1: Switch USB 3G dongle mode

In the general case, the 3G dongle will be recognized as a USB mass storage device when you plug it into USB port. You need to switch it to modem mode and then you can start 3G dial up. SDK uses open source application “usb_modeswitch” to accomplish this work. “usb_modeswitch” needs a configuration file for each 3G dongle. So, what you need to do is add a configuration file into SDK for the new 3G dongle. “usb_modeswitch” keeps updating its configuration file database to support more new 3G dongle. You can download the latest “usb_modeswitch” version and find corresponded configuration file.

Example:

Support Huawei E169u 3G dongle

1. Download usb_modeswitch database “usb-modeswitch-data”
2. Get the vendor ID and product ID of the new 3G dongle

```
# cat /proc/bus/usb/devices
```

11.43 How to enable USB 3G dongle function

The RT288x_SDK supports USB 3G dongle to work as WAN interface. This requires Kernel drivers to support USB stack and dongle device, and also user-space application to establish 3G PPP connection.

Configuration:

Step1: User-space applications configuration

```
#cd RT288x_SDK/source  
#make menuconfig
```

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

Network Applications --->

[*] 3G connection

```
#cd RT288x_SDK/source
```

```
#make menuconfig
```

Kernel/Library/Defaults Selection --->

[*] Customize Busybox Settings

 Linux System Utilities --->

 [*] mdev

 [*] Support /etc/mdev.conf

 [*] Support subdirs/symlinks

 [*] Support regular expressions substitutions when renaming device

 [*] Support command execution at device addition/removal

Step2: Kernel configuration

USB Host driver:

```
#cd RT288x_SDK/source
```

```
#make menuconfig
```

Kernel/Library/Defaults Selection --->

[*] Customize Kernel Settings

 Device Drivers --->

 [*] USB support --->

 <*> Support for Host-side USB

 [*] USB device filesystem

 <*> EHCI HCD (USB 2.0) support

[*] Ralink EHCI HCD support

<*> OHCI HCD support

[*] Ralink OHCI HCD support

USB 3G dongle driver:

```
#cd RT288x_SDK/source
```

```
#make menuconfig
```

Kernel/Library/Defaults Selection --->

[*] Customize Kernel Settings

Device Drivers --->

[*] USB support --->

[*] USB Serial Converter support --->

[*] USB driver for GSM and CDMA modems

PPP driver:

```
#cd RT288x_SDK/source
```

```
#make menuconfig
```

Kernel/Library/Defaults Selection --->

[*] Customize Kernel Settings

Device Drivers --->

[*] Network device support --->

[*] PPP (point-to-point protocol) support

[*] PPP support for async serial ports

Device hot plugging :

```
#cd RT288x_SDK/source  
#make menuconfig
```

```
Kernel/Library/Defaults Selection --->  
[*] Customize Kernel Settings  
    General setup --->  
        [*] Configure standard kernel features (for small systems) --->  
            [*] Support for hot-pluggable devices
```

Sysfs filesystem :

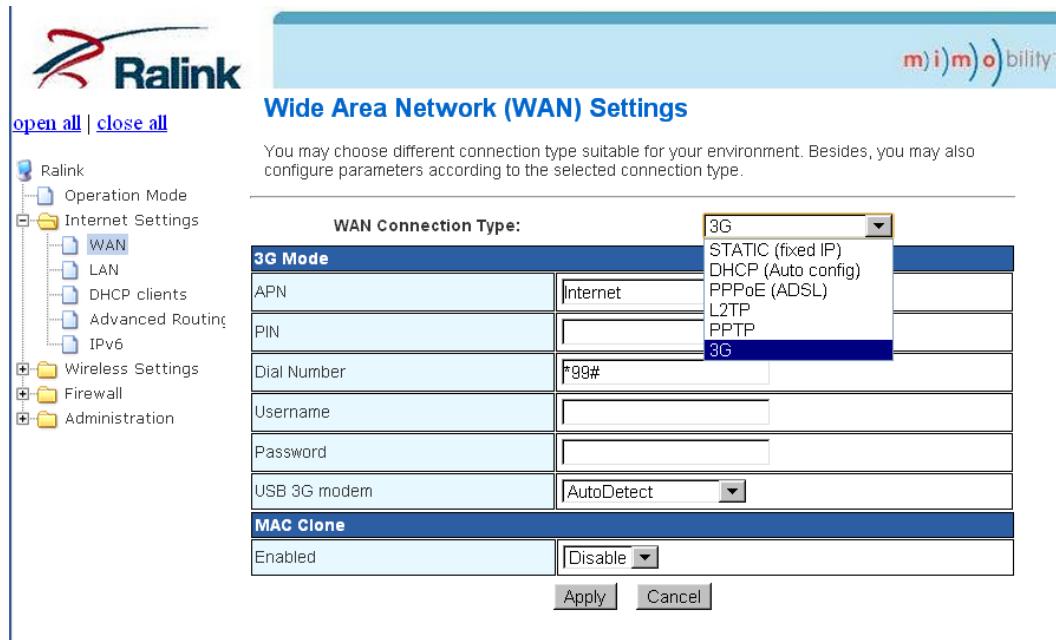
```
#cd RT288x_SDK/source  
#make menuconfig
```

```
Kernel/Library/Defaults Selection --->  
[*] Customize Kernel Settings  
    File systems --->  
        Pseudo filesystems --->  
            [*] sysfs file system support
```

Start 3G dial up

You can start 3G dial up through Web GUI or command line. Some parameters such like APN, Dial number are needed for connection establishment. These parameters should be provided by the ISP.

Web GUI:



Note: 3G Web GUI page doesn't exist in 2M/16M Default Configuration File.

Command line:

Usage: 3g.sh [3G_dongle_model]

Example:

```
#/ 3g.sh HUAWEI-E169
```

3G dongle model supported by current SDK are "HUAWEI-E169", "BandLuxe-C270", "MobilePeak-Titan", and "DATANG-M5731".

11.44 How to enable Port Trigger function

To support the Port Trigger function, the menuconfig options in SDK must be enabled.

Networking Support →

Networking Options →

Network packet filtering framework (Netfilter) →

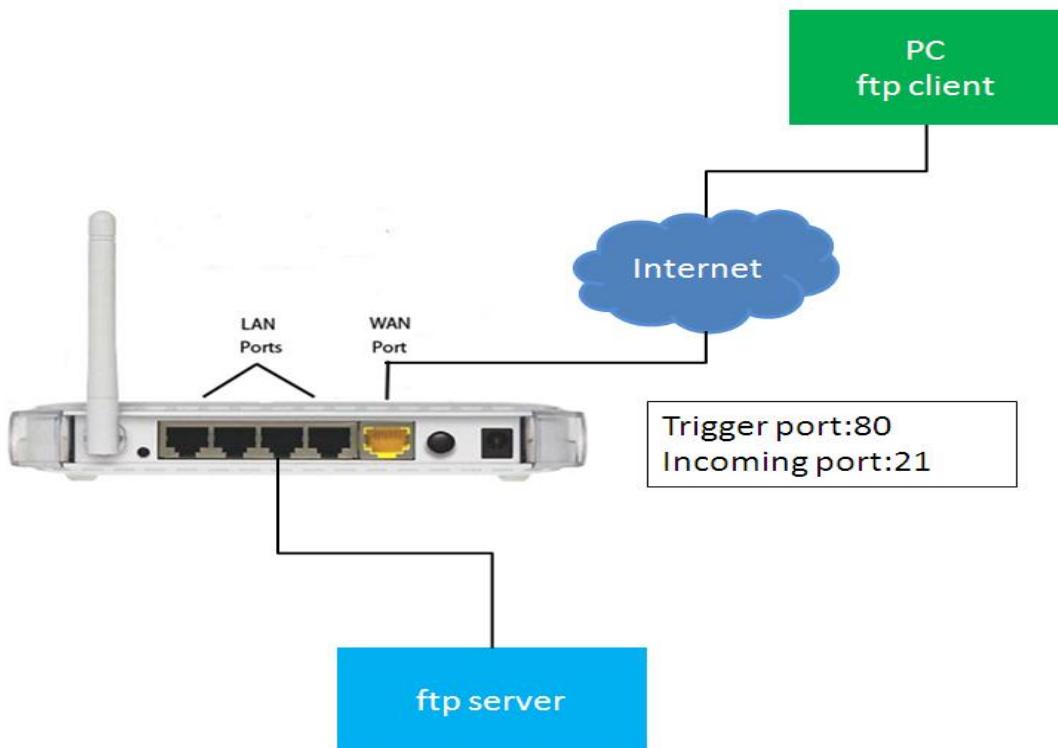
IP:Netfilter configuration →

<*> TRIGGER target support

11.45 Port Trigger information

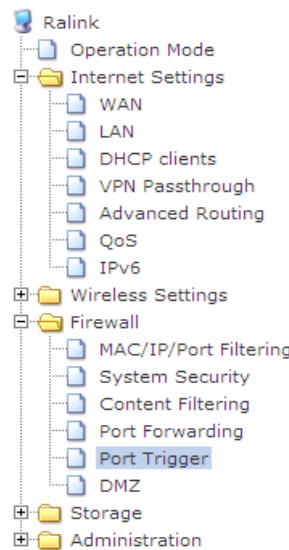
11.45.1 Port Trigger Concept

Port trigger concept is shown in the subsequent figure. Port triggering is a way to trigger port forwarding in which outbound traffic on predefined ports ('called trigger port') causes inbound traffic to specific incoming ports to be dynamically forwarded to the initiating host in a limited period of time.



11.45.2 Port Trigger Usage

WEB UI :


[open all](#) | [close all](#)


Port Trigger Setting

You may setup Port Trigger services on Internet.

Port Trigger

Port Trigger	<input type="button" value="Enable"/>
Trigger Protocol	<input type="button" value="TCP"/>
Trigger Port	<input type="text"/>
Incoming Protocol	<input type="button" value="TCP"/>
Incoming Port	<input type="text"/>
Comment	<input type="text"/>

(The maximum rule count is 32.)

Current Port Trigger in system:

No.	Current Trigger Protocol	Current Trigger Port	Current Incoming Protocol	Current Incoming Port	Comment
1	<input type="checkbox"/>	TCP	80	TCP	21

11.46 How to enable I2S+Codec function

11.46.1 I2S concept

I2S is an Audio interface. It can provide “PLAYBACK” and “RECORD” function with proper codec. This SDK support I2S+WM8960 codec. I2S is in slave mode while WM8960 codec is in master mode. This SDK also provide internal REFCLK to codec as its MCLK.

11.46.2 How to enable I2S+WM8960 codec

1. Enter menuconfig. Choose “Kernel/Library/Defaults Selection” and select “Customize Kernel Settings”

```
----- Kernel/Library/Defaults Selection -----
Enter> selects submenus --->. Highlighted letters are hotkeys
*. Press <Esc><Esc> to exit, <?> for Help. Legend: [*] buil
----- Cross Compiler Path: "/opt/buildroot-gcc342/bin"
--- [ ] Default all settings (lose changes)
[*] Customize Kernel Settings (NEW)
[ ] Customize Vendor/User Settings (NEW)
[ ] Customize Busybox Settings
[ ] Customize uClibc Settings
[ ] Customize uClibc++ Settings
[ ] Update Default Vendor Settings
```

2. Enter “Device Drivers”

```
----- Linux Kernel Configuration -----
Menu. <Enter> selects submenus --->. Highlighted letters
* features. Press <Esc><Esc> to exit, <?> for Help, </> fo
ipable
----- Machine selection --->
[*] Enable FPU emulation
    Endianess selection (Little endian) --->
    CPU selection --->
    Kernel type --->
    General setup --->
[ ] Enable loadable module support --->
[*] Enable the block layer --->
    Bus options (PCI, PCMCIA, EISA, ISA, TC) --->
    Executable file formats --->
    Power management options --->
[*] Networking support --->
[!] Device Drivers --->
    File systems --->
    Kernel hacking --->
    Security options --->
[ ] Cryptographic API --->
    Library routines --->
    Ralink Module --->
--- Load an Alternate Configuration File
Save an Alternate Configuration File
```

3. Enter “Character devices”

```
Configuration
  └── Device Drivers
    u. <Enter> selects submenus --->. Highlighted letters are hotkeys.
      Features. Press <Esc><Esc> to exit, <?> for Help, </> for Search
      Help

      └── Generic Driver Options --->
        [ ] Connector - unified userspace <-> kernelspace
        [*] Memory Technology Device (MTD) support --->
        [ ] Parallel port support --->
        [*] Block devices --->
        [ ] Misc devices --->
        [ ] ATA/ATAPI/MFM/RLL support (DEPRECATED) --->
          SCSI device support --->
        [ ] Serial ATA and Parallel ATA drivers --->
        [ ] Multiple devices driver support (RAID and LVM)
        [*] Network device support --->
        [ ] ISDN support --->
        [ ] Telephony support --->
        Input device support --->
        [ ] Character devices --->
        [ ] I2C support --->
        [ ] SPI support --->
        PPS support --->
        [ ] Dallas's 1-wire support --->
        [ ] Power supply class support --->
        [ ] Hardware Monitoring support --->
```

4. Choose “Ralink I2S Support”

```
Character devices
  u. <Enter> selects submenus --->. Highlighted letters are hotkeys.
    Features. Press <Esc><Esc> to exit, <?> for Help, </> for Search
    Help

    └── Ralink I2S Support
      [*] Ralink GPIO Support
      [*] Ralink GPIO LED Support
      [*] Ralink GDMA Support
        GDMA Channel Allocation Mode (All for Everybody) --->
      [ ] Ralink RT2880 SPI Support
      [*] Ralink RT2880 I2C Support
      [ ] Ralink PCM Support
      [ ] Ralink I2S Support
      [ ] Virtual terminal
      [ ] /dev/kmem virtual device support
      [ ] Non-standard serial port support
        Serial drivers --->
      [*] Unix98 PTY support
      [ ] Support multiple instances of devpts
      [*] Legacy (BSD) PTY support
      (256) Maximum number of legacy PTY in use
      [ ] TPMI top-level message handler --->
      [ ] Hardware Random Number Generator Core support
      [ ] Siemens R3964 line discipline
      [ ] RAW driver (/dev/raw/rawN)
      [ ] Log panic/oops to a RAM buffer
```

5. After choosing “Ralink I2S Support”, you can see “Audio Selection”. The default setting is “Select WM8960”.

And you can also see “Use Internal REFCLK AS MCLK Source” is chosen as default config

```
Character devices
nu. <Enter> selects submenus --->. Highlighted letters are hotkey
features. Press <Esc><Esc> to exit, <?> for Help, </> for Search.
able

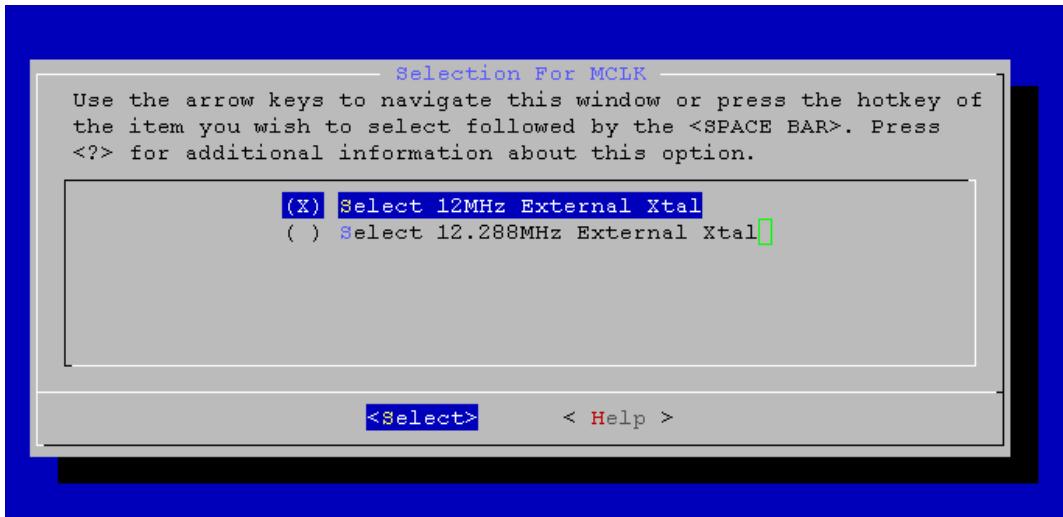
[*] Ralink GPIO Support
[*] Ralink GPIO LED Support
-**- Ralink GDMA Support
    GDMA Channel Allocation Mode (All for Everybody) --->
[ ] Ralink RT2880 SPI Support
-**- Ralink RT2880 I2C Support
[ ] Ralink PCM Support
[*] Ralink I2S Support
    Audio Code Selection (Select WM8960) --->
[*] Use Internal REFCLK As MCLK Source
    Selection For MCLK (Select 12MHz Internal REFCLK) --->
[ ] Virtual terminal
[ ] /dev/kmem virtual device support
[ ] Non-standard serial port support
    Serial drivers --->
[*] Unix98 PTY support
[ ] Support multiple instances of devpts
[*] Legacy (BSD) PTY support
```

6. If you want to use external Xtal to provide MCLK to codec, you can un-choose “Use Internal REFCLK AS MCLK Source” as the following figure. And you can see “Selection For MCLK (Select 12MHz External Xtal)”

```
Character devices
u. <Enter> selects submenus --->. Highlighted letters are hotkey
eatures. Press <Esc><Esc> to exit, <?> for Help, </> for Search.
ble

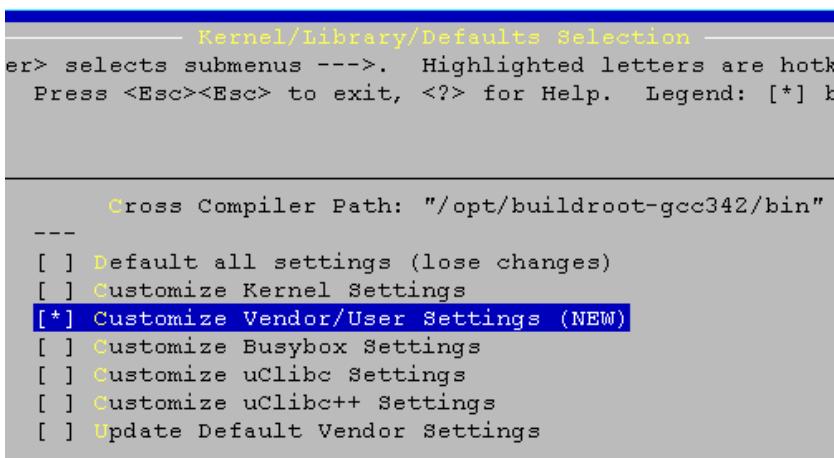
[*] Ralink GPIO Support
[*] Ralink GPIO LED Support
-**- Ralink GDMA Support
    GDMA Channel Allocation Mode (All for Everybody) --->
[ ] Ralink RT2880 SPI Support
-**- Ralink RT2880 I2C Support
[ ] Ralink PCM Support
[*] Ralink I2S Support
    Audio Code Selection (Select WM8960) --->
[!] Use Internal REFCLK As MCLK Source
    Selection For MCLK (Select 12MHz External Xtal) --->
[ ] Virtual terminal
[ ] /dev/kmem virtual device support
[ ] Non-standard serial port support
    Serial drivers --->
```

7. If you want to use 12.288MHz External Xtal, you can enter “Selection For MCLK (Select 12MHz External Xtal)” as the following figure

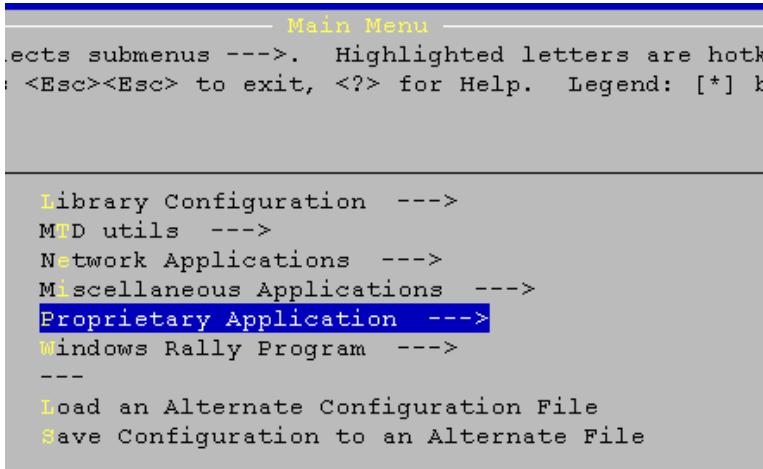


11.46.3 How to enable I2S command in user space application

1. Enter menuconfig. Choose “Kernel/Library/Defaults Selection” and select “Customize Vendor/User Settings”



2. Enter “Proprietary Application”



3. Choose I2S command

```

Proprietary Application
ects submenus --->. Highlighted lett
<Esc><Esc> to exit, <?> for Help. L

[*] Proprietary Application
[*] ATE Agent
[*] Register R/W
[ ] CSR
[ ] Flash
[ ] HW NAT
[ ] SW NAT
[*] MiIMgr
[*] NVRAM
[ ] Layer2 Management
[ ] GPIO
[ ] SPI Command
[ ] I2C Command
[ ] Memory usage
[ ] CoS Support
[ ] Software QoS
[ ] Super DMZ
[*] Embedded Switch Command
[ ] QDMA Command
[*] I2S Command
[ ] ECM Command
[ ] SPDIF Command

```

11.46.4 I2S user command for “PLAYBACK” and “RECORD”

```

#
#
#
# i2scmd
Usage: [cmd] [sratel] [vol] < playback file
      [cmd] [sratel] [vol] [size]
      cmd = 0|1 - i2s raw playback|record
      sratel = 8000|16000|32000|44100|48000 Hz playback sampling rate
      vol = -10^2 db playback volume
i2scmd ...quit
#
#
#

```

The diagram shows two red arrows. One arrow points from the word "playback" to the part of the usage string where "playback file" is specified. Another arrow points from the word "record" to the part of the usage string where "[size]" is specified.

1. Command of “PLAYBACK” function

Example: **i2scmd 0 48000 100 </etc_ro/test_sound.snd**

2. Command of “RECORD” function

Example: **i2scmd 1 48000 100 5000000**

<NOTE> Owing to MT7628 support not only 16 bit, but 24 bit word length, and little/big Endian, please use the following command as MT7628 I2S command:

```
# i2scmd
This is Ralink II2S: i2s_release succeeds
2S Command Program...
Usage: [cmd] [srate] [vol] [wordlen] [Endian fmt]< playback file
       [cmd] [srate] [vol] [size] [wordlen] [Endian fmt]
       cmd = 0|1 - i2s raw playback|record
       srate = 8000|16000|32000|44100|48000 Hz playback sampling rate
       vol = -10~2 db playback volume
       wordlen = 16|24 bit
       endian fmt = 1|0 - little|big endian
i2scmd ...quit
```

1. Command of “PLAYBACK” function:

Example:

16 bit/little Endian: **i2scmd 0 48000 100 16 1 </etc_ro/test_sound_16b_little.snd**

16 bit/big Endian: **i2scmd 0 48000 100 16 0 </etc_ro/test_sound_16b_big.snd**

24 bit/little Endian: **i2scmd 0 48000 100 24 1 </etc_ro/test_sound_24b_little.snd**

24 bit/big Endian: **i2scmd 0 48000 100 24 0 </etc_ro/test_sound_24b_big.snd**

2. Command of “RECORD” function

16 bit/little Endian: **i2scmd 1 48000 100 5000000 16 1**

16 bit/big Endian: **i2scmd 1 48000 100 5000000 16 0**

24 bit/little Endian: **i2scmd 1 48000 100 5000000 24 1**

24 bit/big Endian: **i2scmd 1 48000 100 5000000 24 0**

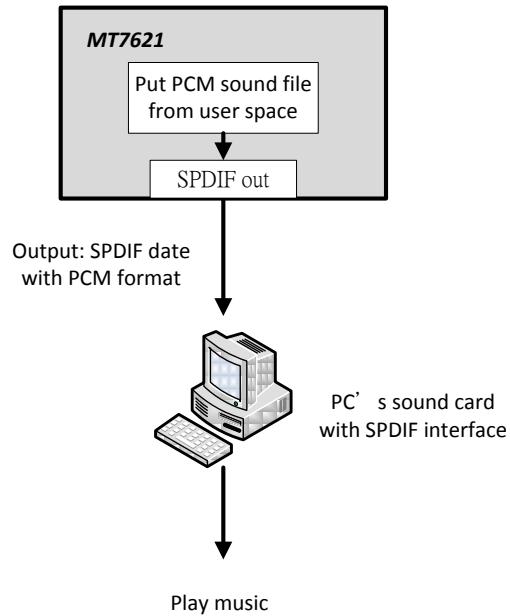
11.47 How to enable SPDIF function

11.47.1 SPDIF concept

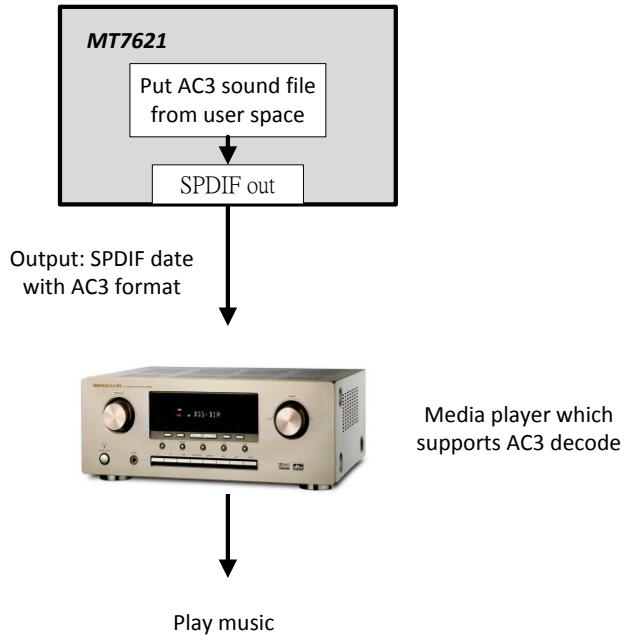
SPDIF is an audio interface. So far, only MT7621 support SPDIF interface and only support “Playback” function.

MT7621 SPDIF support two data format: PCM and AC3 and support sampling rate: 22.05KHz, 24KHz, 32KHz, 44.1KHz, 48KHz. To test SPDIF, you can set the test environments as below:

- PCM test environment:



- AC3 test environment:



11.47.2 How to build ulmage with SPDIF enabling

- Enter menuconfig. Choose “Kernel/Library/Defaults Selection” and select “Customize Kernel Settings”

```
Kernel/Library/Defaults Selection
selects submenus --->. Highlighted letters are hotkeys. Press
to exit, <?> for Help. Legend: [*] built-in [ ] excluded <M>

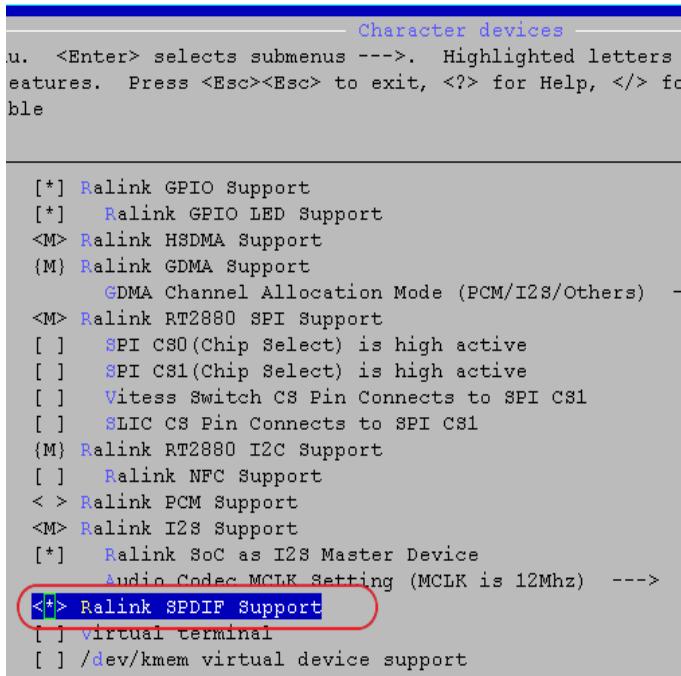
Cross Compiler Path: "/opt/buildroot-gcc342/bin"
---
[*] Default all settings (lose changes)
[*] Customize Kernel Settings (NEW) (highlighted)
[ ] Customize Vendor/User Settings (NEW)
[ ] Customize Busybox Settings
[ ] Customize uClibc Settings
[ ] Customize uClibc++ Settings
[ ] Update Default Vendor Settings
```

2. Then, enter “Device Driver”

```
Linux Kernel Configuration
ter> selects submenus --->. Highlighted letters are hotkeys. P
<Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-
---

Machine selection --->
[*] Enable FPU emulation
  Endianess selection (Little endian) --->
  CPU selection --->
  Kernel type --->
  General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
  Bus options (PCI, PCMCIA, EISA, ISA, TC) --->
  Executable file formats --->
  Power management options --->
[*] Networking support --->
  Device Drivers ---> (highlighted)
    File systems --->
    Kernel hacking --->
    Security options --->
  -*- Cryptographic API --->
    Library routines --->
    Ralink Module --->
  ---
    Load an Alternate Configuration File
    Save an Alternate Configuration File
```

3. After entering “Device Driver”, select “Ralink SPDIF Support”:



```

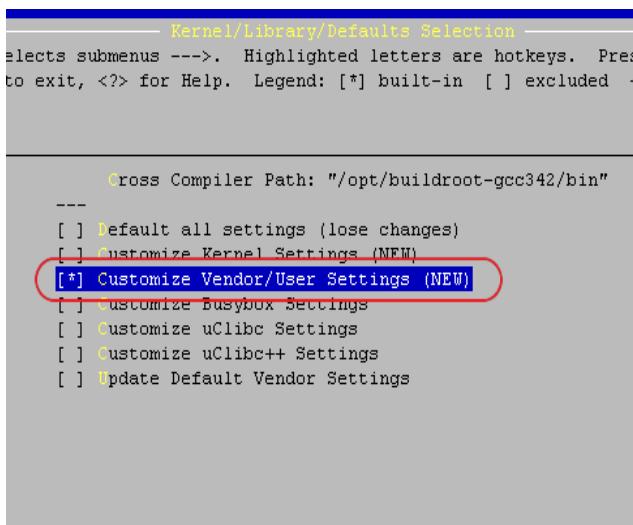
-- Character devices --
u. <Enter> selects submenus --->. Highlighted letters
eatures. Press <Esc><Esc> to exit, <?> for Help, </> fo
ble

[+] Ralink GPIO Support
[+] Ralink GPIO LED Support
<M> Ralink HSDMA Support
(M) Ralink GDMA Support
    GDMA Channel Allocation Mode (PCM/I2S/Others) -
<M> Ralink RT2880 SPI Support
[ ] SPI CS0(Chip Select) is high active
[ ] SPI CS1(Chip Select) is high active
[ ] Vitess Switch CS Pin Connects to SPI CS1
[ ] SLIC CS Pin Connects to SPI CS1
(M) Ralink RT2880 I2C Support
[ ] Ralink NFC Support
< > Ralink PCM Support
<M> Ralink I2S Support
[*] Ralink SoC as I2S Master Device
    Audio Codec MCLK Setting (MCLK is 12Mhz) --->
<+> Ralink SPDIF Support
[ ] virtual terminal
[ ] /dev/kmem virtual device support

```

11.47.3 How to enable SPDIF command in user space application

1. Enter menuconfig. Choose “Kernel/Library/Defaults Selection” and select “Customize Vendor/User Settings”



```

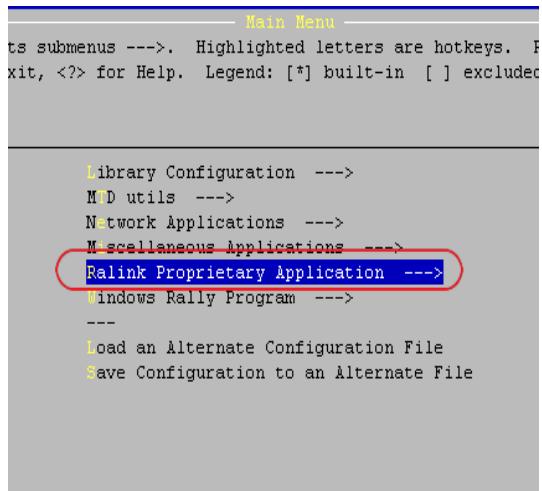
-- Kernel/Library/Defaults Selection --
elected submenus --->. Highlighted letters are hotkeys. Pres
to exit, <?> for Help. Legend: [*] built-in [ ] excluded <

Cross Compiler Path: "/opt/buildroot-gcc342/bin"
---

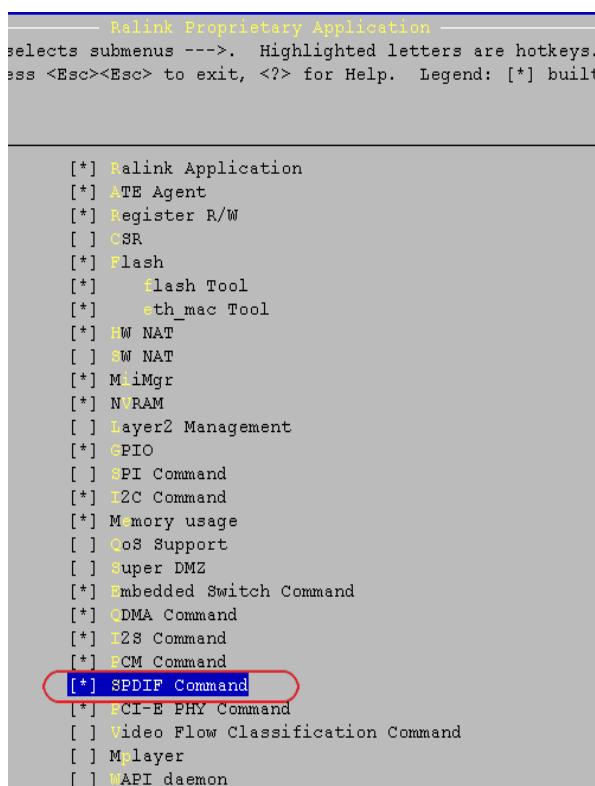
[ ] Default all settings (lose changes)
[ ] Customize Kernel Settings (NEW)
<+> Customize Vendor/User Settings (NEW)
[ ] Customize Busybox Settings
[ ] Customize uClibc Settings
[ ] Customize uClibc++ Settings
[ ] Update Default Vendor Settings

```

2. Then, enter “Ralink Proprietary Application”



3. Select “SPDIF command”



11.47.4 SPDIF user command for “PLAYBACK”

```
#  
# spdifcmd  
This is Mediatek SPDIF Command Program...  
Usage: [fmt=0] [srate] [wordlen] [pathname]  
       [fmt=1] [srate] [rawtype] [pathname]  
       fmt = 0|1 - spdif pcm| raw data  
       srate = 22050| 24000| 32000| 44100| 48000 Hz sampling frequency  
       rawtype = for raw data (fmt = 1) -- (0: Null data;) 1: AC3 data; (3: Pause)  
       wordlen = 16| 24 bits per sample  
       downsample = 1: no down sample; 2: 2x down sample; 4: 4x down sample  
       [fmt=2] [pathname]  
spdifcmd ...quit  
#
```

1. Command for PCM data, 16 bit word length
spdifcmd 0 48000 16 </etc_ro/test_sound.snd
2. Command for PCM data, 24 bit word length
spdifcmd 0 48000 24 </etc_ro/test_sound.snd
3. Command for AC3 data
spdifcmd 1 48000 1 </etc_ro/test_sound.snd

11.48 How to enable Dual Image support

11.48.1 Dual Image function objective

This function is used to prevent the device from being out of use because of firmware upgrade failure.

11.48.2 How to configure Dual Image function

Kernel:

1. Enter menuconfig. Choose “Kernel/Library/Defaults Selection” and select “Customize Kernel Settings”

```
----- Kernel/Library/Defaults Selection -----
<Enter> selects submenus --->. Highlighted letters are hotkeys
. Press <Esc><Esc> to exit, <?> for Help. Legend: [*] built-in
[ ] Default all settings (lose changes)
[*] Customize Kernel Settings (NEW)
[ ] Customize Vendor/User Settings (NEW)
[ ] Customize Busybox Settings
[ ] Customize uClibc Settings
[ ] Customize uClibc++ Settings
[ ] Update Default Vendor Settings
```

2. Enter "Machine selection"

```
----- Linux Kernel Configuration -----
navigate the menu. <Enter> selects submenus --->.
Pressing <Y> includes, <N> excludes, <M> modularize
for Help, </> for Search. Legend: [*] built-in [ able

[*] Machine selection --->
[*] Enable FPU emulation
    Endianess selection (Little endian) --->
    CPU selection --->
    Kernel type --->
    General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
    Bus options (PCI, PCMCIA, EISA, ISA, TC) --->
    Executable file formats --->
    Power management options --->
[*] Networking support --->
    Device Drivers --->
    File systems --->
    Kernel hacking --->
    Security options --->
< > Cryptographic API --->
    Library routines --->
    Ralink Module --->
```

3. Choose "Dual Image"

```

System type (Ralink MT7620 board) --->
Soc Hardware Type (MT7620-ASIC) --->
DRAM Size (8M) --->
Flash Type (SPI) --->
Flash Size (4M) --->
[*] Dual Image
[ ] Kernel NVRAM
Root File System Type (RootFS_in_RAM) --->
(8192) Default RAM disk size
< > Ralink Hardware Timer0/Timer1
[ ] Ralink CPU Sleep mode
[*] Ralink System Tick Counter
[ ] Kprofile

```

4. Select “Flash Type” for your platform

```

System type (Ralink MT7620 board) --->
Soc Hardware Type (MT7620-ASIC) --->
DRAM Size (8M) --->
Flash Type (SPI) --->
Flash Size (4M) --->
[*] Dual Image
[ ] Kernel NVRAM
Root File System Type (RootFS_in_RAM) --->
(8192) Default RAM disk size
< > Ralink Hardware Timer0/Timer1
[ ] Ralink CPU Sleep mode
[*] Ralink System Tick Counter
[ ] Kprofile

```

5. Select “Flash Size” for your platform

```

System type (Ralink MT7620 board) --->
Soc Hardware Type (MT7620-ASIC) --->
DRAM Size (8M) --->
Flash Type (SPI) --->
Flash Size (8M) --->
[*] Dual Image
[ ] Kernel NVRAM
Root File System Type (RootFS_in_RAM) --->
(8192) Default RAM disk size
< > Ralink Hardware Timer0/Timer1
[ ] Ralink CPU Sleep mode
[*] Ralink System Tick Counter
[ ] Kprofile

```

Uboot:

1. Enter menuconfig. Choose “Dual Image”

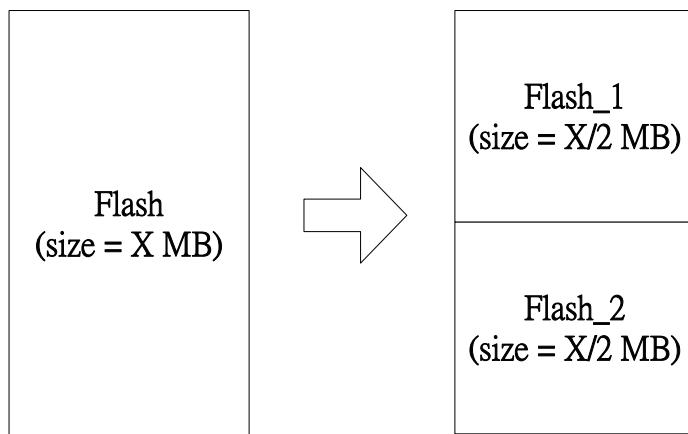
```
---  
(ASIC) Chip Type  
(MT7620) Chip ID  
(None) Port 5 Connect to  
(None) Port 4 Connect to  
(SPI) Flash Type  
(4M) Flash Size  
(DDR1) DRAM Type  
(256Mb) DDR Component  
(16bits) DDR Width  
---  
(ROM) Ram/Rom version  
[*] Dual Image (NEW)  
[ ] Partition LAN/WAN (NEW)  
---  
Load an Alternate Configuration File  
Save Configuration to an Alternate File
```

2. Select “Flash Size” for your platform

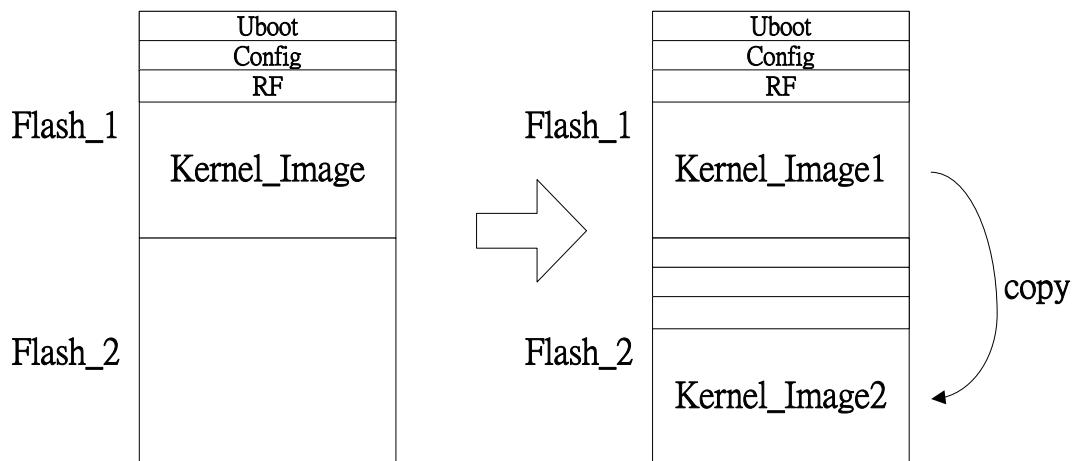
```
---  
(ASIC) Chip Type  
(MT7620) Chip ID  
(None) Port 5 Connect to  
(None) Port 4 Connect to  
(SPI) Flash Type  
(8M) Flash Size  
(DDR1) DRAM Type  
(256Mb) DDR Component  
(16bits) DDR Width  
---  
(ROM) Ram/Rom version  
[*] Dual Image (NEW)  
[ ] Partition LAN/WAN (NEW)  
---  
Load an Alternate Configuration File  
Save Configuration to an Alternate File
```

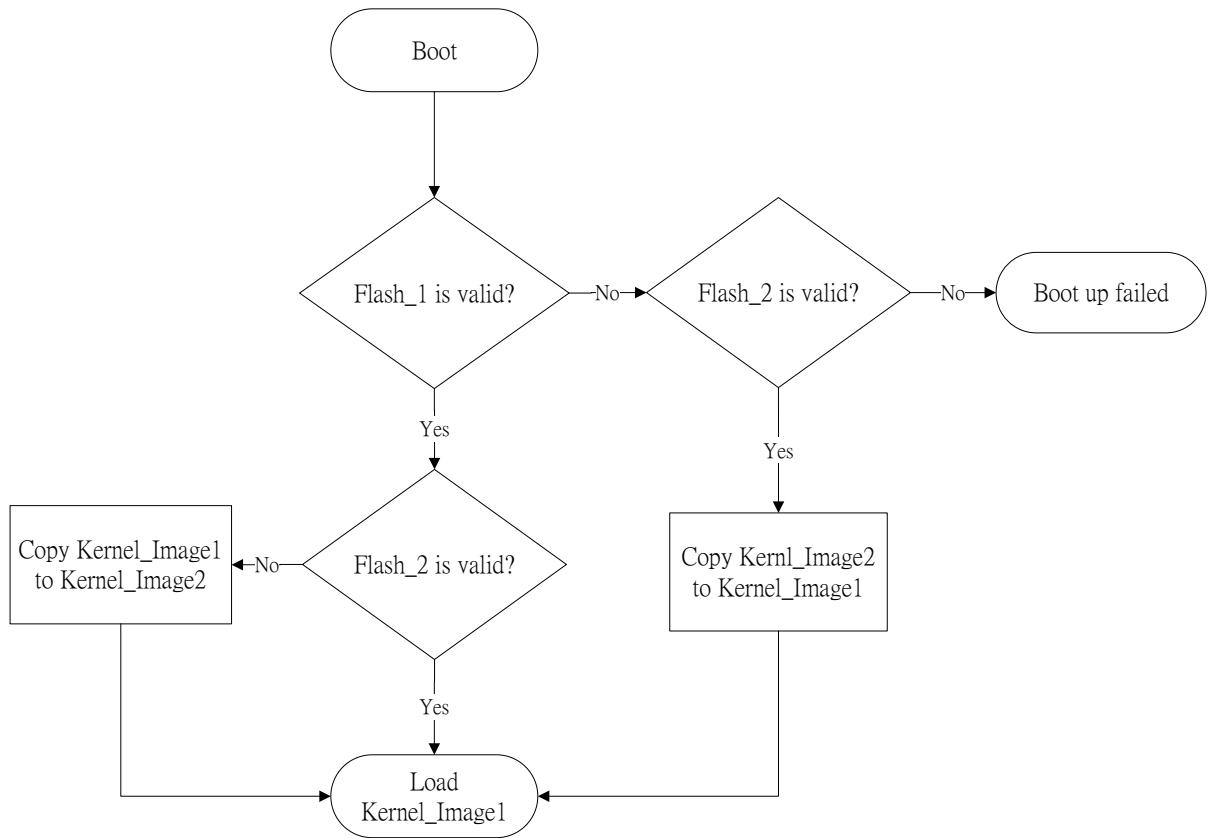
11.48.3 Dual Image function flowchart

- After enabling Dual Image function, the Flash will be split to partitions (Flash_1, Flash_2), and the size of each partition will be the half of original Flash size.



2. When boot up, both Flash_1 and Flash_2 will be verified to see if the partition is valid or not. If Flash_1 is valid but Flash_2 is not, then the Kernel_Image1 will be copied to Flash_2. If Flash_2 is valid but Flash_1 is not, then Kernel_Image2 will be copied to Flash_1. If both partitions are valid, no action will be taken.





11.49 How to enable NFC support

11.49.1 How to compile NFC code

- a. Enable Wifi config in 'make menuconfig' (CONFIG_WSC_NFC_SUPPORT=y)
 - b. config_menuconfig ---> Proprietary Application --->
- [*] NFC Service Daemon
- (nfcsd will communicates with nfc middleware , nfc middleware will communicates with MT6605 driver)
- [*] NFC Handover Daemon
- (nfchodis communicates with nfcsd via socket , get data from MT6605 and pass to WiFi driver via ioctl)
- c. linux_menuconfig --->
- Device Drivers --->
- Character devices --->
- <M> Ralink RT2880 I2C Support (CONFIG_RALINK_I2C)
- [*] NFC Support (CONFIG_MTK_NFC_SUPPORT)
- [] MT6605 NFC Simulation (CONFIG_MTK_NFC_MT6605_SIM)optional
- Device Drivers --->
- [*] Network device support --->
- <*> MediaTek(R) NFC support (CONFIG_MT6605_NFC)

11.49.2 Where to buy NFC tag

Customer can buy NFC tag from below website.

<http://www.amazon.com/Tags-Chip-Pack-Keychain-Bonus/dp/B00CE6QGLU>

<http://search.taobao.com/search?q=nfc+tag>

Sag : Ada Chu : +886-4-2492-5298

Dynacard : Eddie Lee : +886-3-4834868

Basically, we need to use MT6595-based phone to do the test. (MeiZu phone will be MP in 2014.8.)

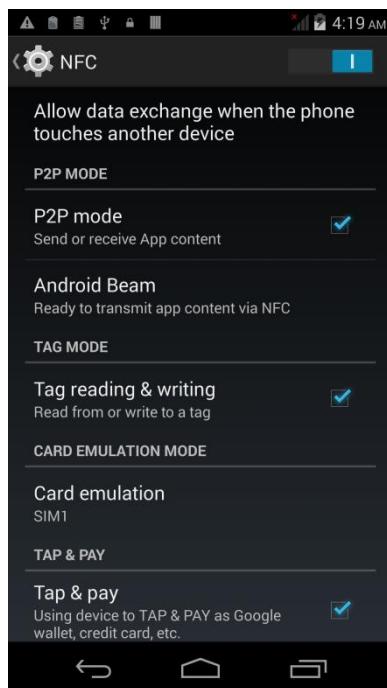
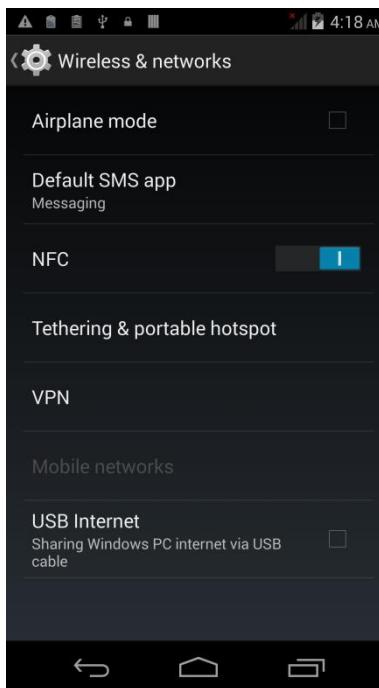
11.49.3 NFC command and How to test

DUT:

MT7621 (AP) + MT6595 (Phone) + MT6605 (NFC)



Step 1 : MT6595 Enable NFC, P2P mode open



Step 2 : MT7621 WPS Enable

Web browser address <http://10.10.10.254>, id: admin pwd:admin

WPS Config "Enable"



Step 3 : MT7621 NFC Enable

Web browser address <http://10.10.10.254>, Administration → System Command

Command Line: nfchod -s ra0 & Apply

Command Line: iwpriv ra0 set NfcStatus=1 Apply



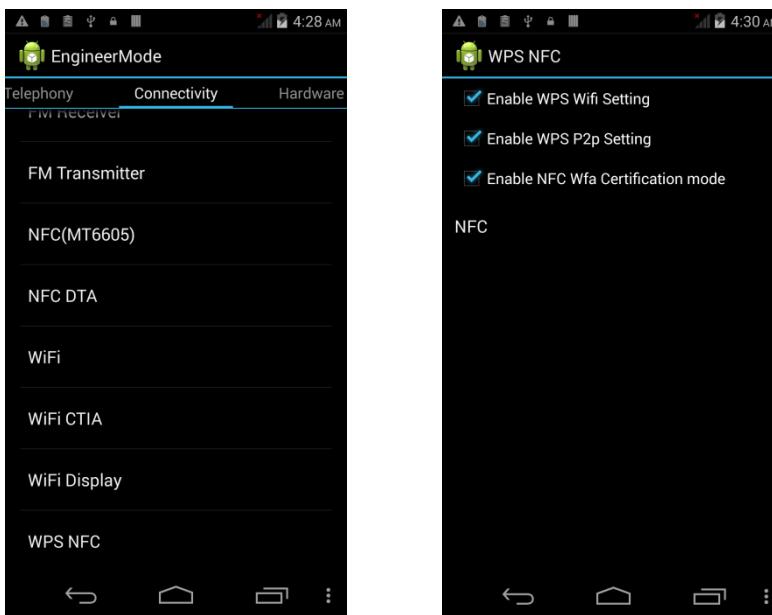
PHONE: (MT6595)

Step 4 : MT6595 WPS start

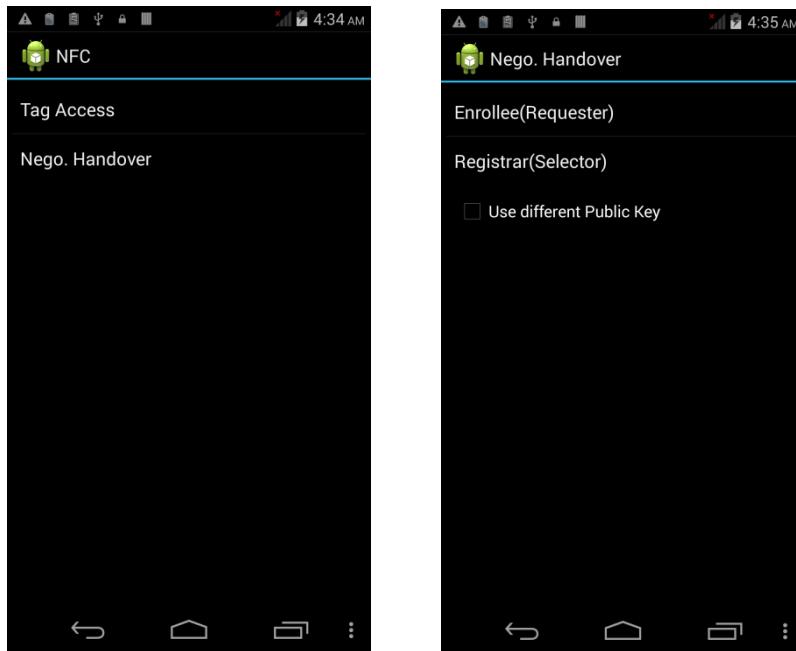
On home screen tap , Key *#*#3646633#*#*



In Connectivity, select WPS NFC, make 3 items been checked (if need, please phone reboot),
then Tap NFC

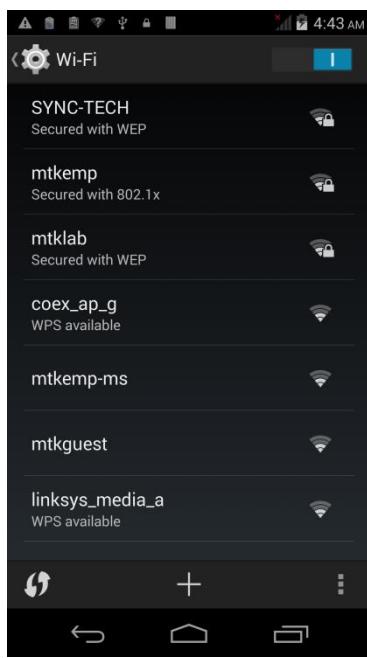


select Nego. Handover, then select Enrollee(Requester)



Take MT6595(Phone) close MT7621(AP) NFC Antenna, if P2P detection please tap phone screen.

Finally, check Wi-Fi link status.



11.50 How to enable AES Engine

11.50.1 AES Engine Support

AES Engine provides encryption and decryption operations with AES-CBC and AES-ECB algorithm.

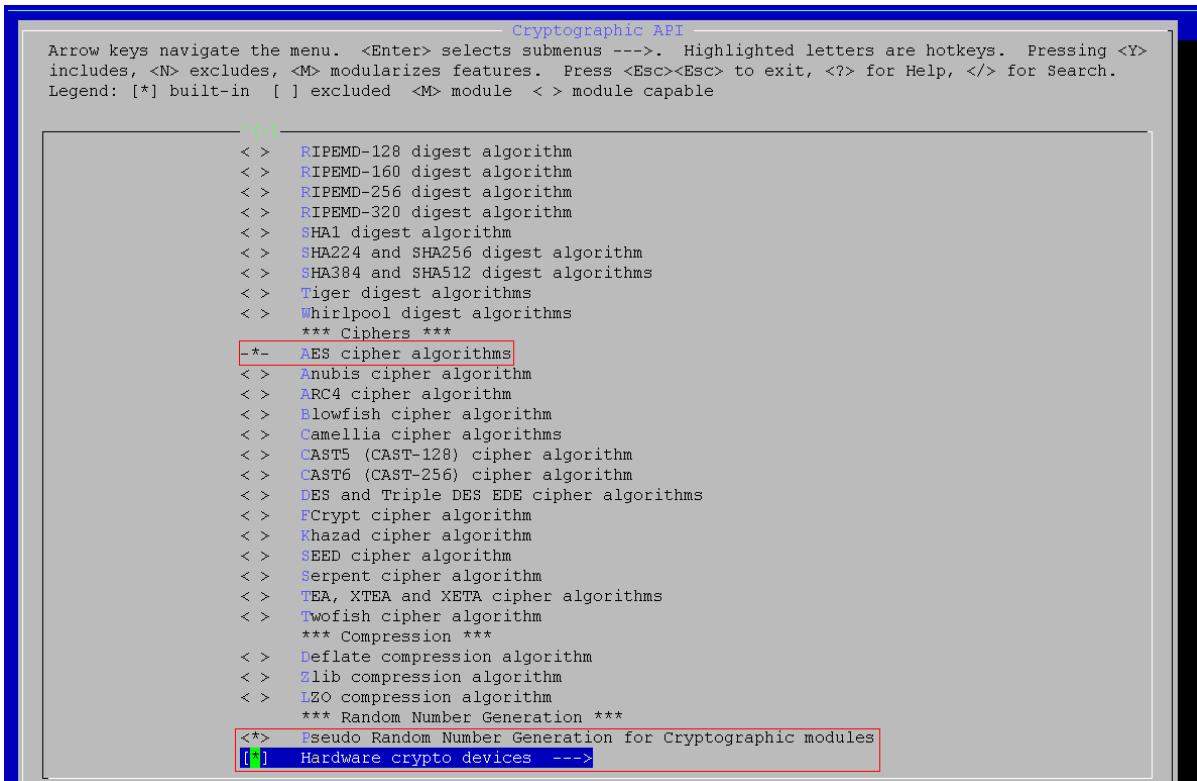
11.50.2 How to enable AES Engine and Run test Program

1. Enter menuconfig. Choose “Kernel/Library/Defaults Selection” and select “Customize Kernel Settings”
2. Enter Cryptographic API then select “Testing Module”, “CBC Support”, and “ECB Support”.

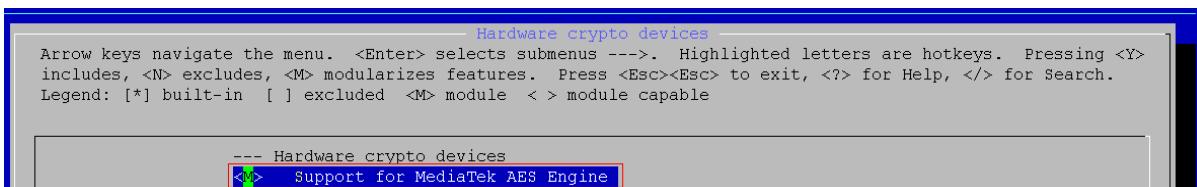
```
Cryptographic API
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y>
includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search.
Legend: [*] built-in [ ] excluded <M> module < > module capable

--- Cryptographic API
      *** Crypto core or helper ***
[ ]  FIPS 200 compliance
-* Cryptographic algorithm manager
[*] Disable run-time self tests
<*> Null algorithms
< > Software async crypto daemon
< > Authenc support
*> Testing module
      *** Authenticated Encryption with Associated Data ***
< > CCM support
< > GCM/GMAC support
< > Sequence Number IV Generator
      *** Block modes ***
<*> CBC support
< > CTR support
< > CTS support
*> ECB support
< > PCBC support
      *** Hash modes ***
< > HMAC support
```

3. Select “AES cipher algorithm”, “Psedu Random Number Generation”, and “Hardware crypto devices”.



- Enter "Hardware crypto devices", select "Support for MeidaTek AES Engine".



- Run AES Engine Driver and Kernel Test Module:

```
# insmod mtk_aes b=16
```

```
# insmod tcrypt mode=200
```

Then test result will show the performance of aes-cbc and aes-ecb.

```
# insmod tcrypt mode=200
testing speed of ecb(aes) encryption
test 0 (128 bit key, 16 byte blocks): 1 operation in 22 usec (64 bytes)
test 1 (128 bit key, 64 byte blocks): 1 operation in 22 usec (256 bytes)
test 2 (128 bit key, 256 byte blocks): 1 operation in 22 usec (1024 bytes)
test 3 (128 bit key, 1024 byte blocks): 1 operation in 44 usec (4096 bytes)
test 4 (128 bit key, 8192 byte blocks): 1 operation in 120 usec (16384 bytes)
test 5 (192 bit key, 16 byte blocks): 1 operation in 22 usec (64 bytes)
test 6 (192 bit key, 64 byte blocks): 1 operation in 22 usec (256 bytes)
test 7 (192 bit key, 256 byte blocks): 1 operation in 22 usec (1022 bytes)
test 8 (192 bit key, 1024 byte blocks): 1 operation in 44 usec (4096 bytes)
test 9 (192 bit key, 8192 byte blocks): 1 operation in 130 usec (16384 bytes)
test 10 (256 bit key, 16 byte blocks): 1 operation in 22 usec (64 bytes)
test 11 (256 bit key, 64 byte blocks): 1 operation in 22 usec (256 bytes)
test 12 (256 bit key, 256 byte blocks): 1 operation in 22 usec (1024 bytes)
test 13 (256 bit key, 1024 byte blocks): 1 operation in 44 usec (4096 bytes)
test 14 (256 bit key, 8192 byte blocks): 1 operation in 140 usec (16384 bytes)
```

- Build OpenSSL:

Edit source/user/Makefile

Add this line:

```
dir_y += openssl-1.0.1f
```

7. Run OpenSSL with AES Engine:

```
# insmod mtk_aes b=16
#insmod cryptodev
#mdev -s
#openssl speed -evp aes-256-cbc -elapsed -engine cryptodev

# openssl speed -evp aes-256-cbc -elapsed -engine cryptodev
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
engine "cryptodev" set.
You have chosen to measure elapsed time instead of user CPU time.
Doing aes-256-cbc for 3s on 16 size blocks: 140259 aes-256-cbc's in 3.00s
Doing aes-256-cbc for 3s on 64 size blocks: 140034 aes-256-cbc's in 3.00s
Doing aes-256-cbc for 3s on 256 size blocks: 139504 aes-256-cbc's in 3.01s
Doing aes-256-cbc for 3s on 1024 size blocks: 137592 aes-256-cbc's in 3.00s
Doing aes-256-cbc for 3s on 8192 size blocks: 36100 aes-256-cbc's in 3.01s
OpenSSL 1.0.1f 6 Jan 2014
built on: Mon Mar 31 13:33:06 CST 2014
options:bn(64,32) rc4(ptr,char) des(idx,cisc,16,long) aes(partial) idea(int) blowfish(ptr)
compiler: /opt/buildroot-gcc342/bin/mipsel-linux-gcc -O2 -fomit-frame-pointer -pipe -Dlinux -D__linux__ -Dlinux -DEMbed -I/home/qwert/perforce/ws_qwert_chin_temp/WIFI_SOC/TRUNK/RT288x_SDK/source/lib/include -I/home/qwert/perforce/ws_qwert_chin_temp/WIFI_SOC/TRUNK/RT288x_SDK/source -fPIC -DOPENSSL_PIC -DOPENSSL_THREADS -D_REENTRANT -DDSO_DLFCN -DHAVE_DLFCN_H -DHAVE_CRYPTODEV -DUSE_CRYPTODEV_DIGESTS -DTERMIO -O3 -fomit-frame-pointer -Wall
The 'numbers' are in 1000s of bytes per second processed.
type      16 bytes    64 bytes   256 bytes   1024 bytes   8192 bytes
aes-256-cbc    748.05k    2987.39k   11864.79k   46964.74k   98249.57k
```