

RALINK TECHNOLOGY, CORP.

RALINK AP SDK 3.2.0.0 USER'S MANUAL

Copyright © 2008 Ralink Technology, Corp.

All Rights Reserved.

This document is property of Ralink Technology Corporation. Transmittal, 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 Ralink Technology Corporation 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 Ralink (hereinafter "Proprietary Information")

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

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

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

Ralink Technology Corporation (Taiwan)

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

ChuPei City

HsinChu Hsien 302, Taiwan, ROC

Tel +886-3-560-0868

Fax +886-3-560-0818

Sales Taiwan: Sales@ralinktech.com.tw

Technical Support Taiwan: FAE@ralinktech.com.tw

<http://www.ralinktech.com/>

TABLE OF CONTENTS

1	SDK History.....	8
2	Version History.....	11
3	Overview of Ralink AP Demo Board	11
3.1	RT2880	11
3.2	RT3052	14
4	Overview of AP SDK source code	17
5	Tool-chain	18
5.1	Install toolchain.....	18
5.2	Install LZMA Utility	18
5.3	Install mksquashfs Utility	18
6	Boot-loader	19
6.1	Uboot Configuration	19
6.2	Build uboot Image.....	21
6.3	Burn Uboot image.....	21
7	User Library.....	22
7.1	Library Configuration	22
7.2	Library Porting	22
7.3	Build user library	23
8	User Application.....	24
8.1	Ralink Proprietary Applications.....	24

8.1.1	ATED	24
8.1.2	REG	24
8.1.3	FLASH.....	25
8.1.4	GPIO.....	25
8.1.5	MII_MGR	26
8.1.6	MTD	27
8.1.7	NVRAM	27
8.1.8	SPICMD	28
8.1.9	I2CCMD.....	28
8.1.10	Script.....	29
8.2	goahead	29
8.3	nvrn library.....	29
8.4	wsc_upnp.....	29
8.5	iptables	29
8.6	ntpclient.....	30
8.7	mtd-utils.....	30
8.8	ppp-2.4.2.....	30
8.9	bridge-utils	30
8.10	wireless_tools	30
8.11	inadyn	31
8.12	zebra-0.95a_ripd.....	31

8.13	wpa_supplicant-0.5.7.....	31
8.14	totd-1.5	31
8.15	samba-3.0.2	31
8.16	radvd-1.0.....	31
8.17	pptp-client.....	32
8.18	rp-l2tp-0.4	32
8.19	ctorrent-dnh3.2	32
8.20	dhcp6	32
8.21	dnsmasq-2.40.....	32
8.22	igmpproxy	33
8.23	matrixssl-1.8.3.....	33
8.24	rp-pppoe-3.8.....	33
8.25	Port new user application	33
9	Linux Kernel.....	35
9.1	Linux configuration	35
9.2	Change Flash/DRAM Size	37
9.3	Change Switch Controller in RT2880 Platform.....	38
9.4	Update User/Kernel default settings	39
9.5	Compile Linux image	40
9.6	Port new Linux kernel module	40
9.7	Execute commands at boot up time	42

9.8	Add new files in RootFs.....	42
9.9	Image DownSize.....	43
10	Flash Layout and Firmware Upgrade	46
10.1	Flash Layout	46
10.2	Firmware Upgrade	46
10.2.1	By Uboot.....	47
10.2.2	By WebUI.....	47
11	FAQ	49
11.1	RT2880 Default Password/UART/Networking Setting.....	49
11.2	System Requirements for Host Platform	49
11.3	How to add new default parameter in flash	50
11.3.1	Example 1	50
11.3.2	Example 2	50
11.4	Enable Ethernet Converter Feature	52
11.5	Change RF chip from RT2820 to RT2850 in RT2880 Platform.....	53
11.6	How to change Ethernet mac address	54
11.7	How to configure GPIO ports	54
11.8	Use GPIO to turn on LED	56
11.9	Use LED Firmware to turn on LED	60
11.10	How to start telnet server	62
11.10.1	busybox setting	62

11.10.2	Linux setting	62
11.11	11n Bit Rate Derivation	64
11.12	How to build single image for flash programmer	67
11.13	How to Power down rt305x Ethernet ports.....	69
11.14	How to enable NFS client.....	70
11.15	How to add a new language to web ui	71
11.16	How to enable watchdog in RT305x	72
11.17	How to enable USB storage in RT305x platform.....	73
11.18	How to enable USB automount RT305x platform	74

FIGURES

Figure 1	RT2880 Demo Board Diagram	12
Figure 2	RT3052 Demo Board Diagram	15
Figure 3	uClib configuration Menu	22
Figure 4	User Library Configure Menu.....	23
Figure 5	IC+ 10/100 Switch Operation Diagram	39
Figure 6	Ralink SDK Flash Layout (4MB)	46
Figure 7	WebUI Firmware Upgrade	48
Figure 8	Ethernet Converter Operation Diagram	52
Figure 9	WebUI - STA Mode Setting	53
Figure 10	QA – Burn your own EEPROM binary file.....	54
Figure 11	QA – Modify GMAC Mac address	54

Figure 12 LED Definition of WPS Specification	59
Figure 13 Configuration Procedure of Telnet Server	64

TABLES

Table 1 RT2880 Memory Mapping	12
Table 2 RT3052 Memory Mapping	15
Table 3 Networking Setting	49
Table 4 UART Setting	49
Table 5 Web Setting.....	49
Table 6 Requirements of Host Platform	49
Table 7 GPIO Usage of RT2880	57
Table 8 GPIO Usage of RT3052	57
Table 9 RT2880 LED Parameters in Flash.....	60

1 SDK HISTORY

Release	Features	Platform Support	Schedule
1.2 SDK	OS: Linux 2.4.30 Bootloader: Uboot Toolchain : GNU based cross-compiler Driver: UART, Giga Ethernet, Flash, Wi-Fi Driver Application: Bridging, Routing, NAT, PPPoE, Web server, DHCP client, DHCP server Wi-Fi features: WMM, WMM-PS, WEP, WPA/WPA2 personal, WPA/WPA2 Enterprise	RT2880 Shuttle Support IC+ 5 ports 10/100 Switch Support Marvall Giga Single Phy Support	Formal: 2007/3/20

1.3 SDK	<p>Feature parity with 1.2 SDK plus:</p> <p>Application: NTP, DDNS, WebUI enhance, Vista RG (Native IPv6, LLTD), Firewall</p> <p>Driver: I2C, SPI, GPIO driver</p> <p>Wi-Fi features: Intergraded QA, WPS, mBSSID, WDS, STA mode, 802.1x</p> <p>Concurrent AP support</p>	RT2880 MP Support	<p>Beta:</p> <p>2007/04/30</p> <p>Formal:</p> <p>2007/05/25</p>
2.0 SDK	<p>Feature parity with 1.3 SDK plus:</p> <p>File system support ramdisk and squashfs</p> <p>WebUI: save/restore configure. WPS PIN, WPS PBC, factory default, STA mode support</p> <p>Application: push button to load default configuration (GPIO reference design)</p> <p>Wi-Fi features: AP-Client</p> <p>Ethernet Converter Support</p>	None	<p>Beta:</p> <p>2007/07/06</p> <p>Formal :</p> <p>2007/07/20</p>
2.2 SDK	<p>Feature parity with 2.0 SDK plus:</p> <p>AP version 1.6.0.0</p> <p>STA version 1.4.0.0</p> <p>Wi-Fi Certification: 802.11 b/g/n, WPA2, WMM, WMM-PS, WPS</p> <p>Operation Mode reorganization to "Bridge", "Gateway", and "Ethernet Converter"</p> <p>support iNIC driver</p> <p>Support Squash with LZMA filesystem</p>	Vitesse Switch Support	<p>Formal :</p> <p>2007/11/08</p>

2.3 SDK	Feature parity with 2.2 SDK plus: iNIC v1.1.6.1 RT2561 driver v1.1.2.0 Spansion Flash Support RT2860 AP driver v1.7 RT2860 STA driver v1.5 RT2561 WebUI Multi-Language WebUI support	IC+ 100Phy Realtek 100Phy	Formal : 2008/01/16
2.4 SDK	Feature parity with 2.3 SDK plus: iNIC v1.1.7.1 RT2860 AP driver v1.8.1.0 RT2860 STA driver v1.6.0.0 Static/Dynamic Routing Content Filtering	Mii iNIC	Formal : 2008/04/07
3.0 SDK	Feature parity with 2.4 SDK plus: OS: Linux 2.6.21 (Linux2.4 for RT2880, Linux-2.6 for RT3052) 8MB Flash Support – S29GL064N/MX29LV640 Storage Application – FTP/Samba	RT3052 Support	Formal : 2008/06/06
3.1 SDK	Feature parity with 3.0 SDK plus: RT2860 AP driver v1.9.0.0 RT2860 STA driver v1.7.0.0 [RT3052] 16MB/32MB NOR flash support [RT3052] Boot from 0xbf00.0000(MA14=1) [RT3052] Boot from 0xbfc0.0000(MA14=0)	RT2880 platforms RT3052 platforms	Formal : 2008/07/30
3.2 SDK	Feature parity with 3.1 SDK plus: RT2860 AP driver v2.0.0.0 RT2860 STA driver v1.8.0.0 GreenAP support Busybox 1.12.1 MTD-Based Flash API	RT2880 platforms RT3050 platforms RT3051 platforms RT3052 platforms	Formal : 2008/10/06

2 VERSION HISTORY

Release	Features	Author
1.2	Initial	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 OVERVIEW OF 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.

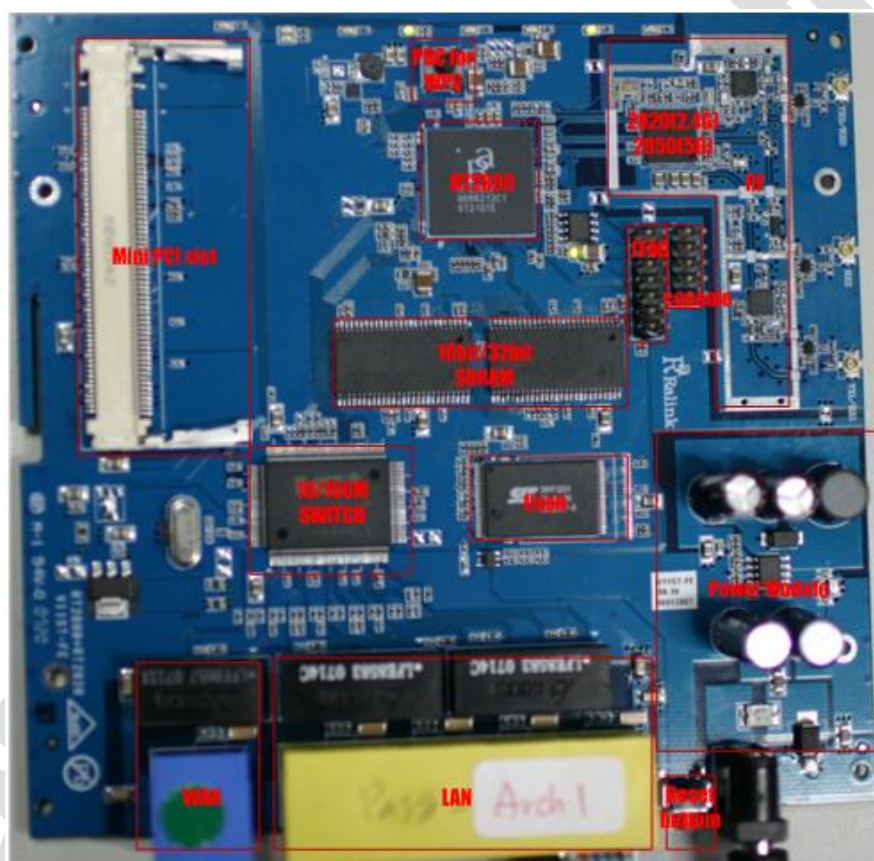


Figure 1 RT2880 Demo Board Diagram

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.

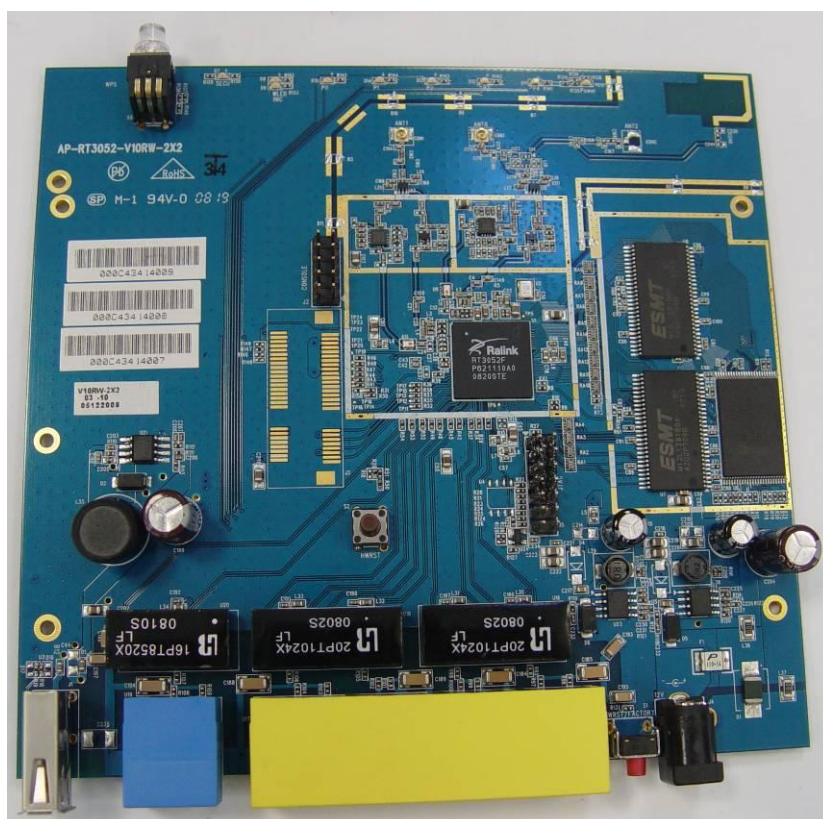


Figure 2 RT3052 Demo Board Diagram

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)	<p>When BOOT_FROM = 2'b00, <16MB external 16-bit flash is mapped.</p> <p>When BOOT_FROM = 2'b01, <8MB external 8-bit flash is mapped.</p> <p>When BOOT_FROM = 2'b10, 4KB internal boot RAM is mapped for boot from NAND application.</p> <p>When BOOT_FROM = 2'b11, 8KB internal boot ROM is mapped for iNIC application.</p>

The remainder of this page is intentionally left blank

4 OVERVIEW OF AP SDK SOURCE CODE

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

The extract process creates directory like `"/home/${user}/RT288x_SDK"`.

a. RT288x_SDK package contains the following directories.

- toolchain : mips toolchain
- source : Linux kernel source
- tools : useful script

b. source directory contains the following directories.

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

The remainder of this page is intentionally left blank

5 TOOL-CHAIN

Ralink AP SDK uses buildroot to build the Linux kernel image. Buildroot is a set of Makefiles and patches. It is simple to generate a cross-compilation toolchain and root filesystem for your target Linux system, using the uClibc C library.

5.1 INSTALL TOOLCHAIN

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

The extract process creates a directory like "/opt/buildroot-gdb"

5.2 INSTALL LZMA UTILITY

lzma is necessary to generate the compressed kernel image. Ralink RT2880 SDK can use lzma to compress the kernel image.

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

Use gzip or lzma to compress kernel image. Modify
RT288x_SDK/source/vendors/Ralink/{Platform}/Makefile

```
COMP = gzip
```

use gzip to compress Linux kernel image

```
COMP = lzma
```

use lzma to compress the Linux kernel image

5.3 INSTALL MKSQUASHFS UTILITY

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

Linux-2.4.x Kernel Version

```
#cd RT288x_SDK/toolchain/mksquash_lzma-3.0
#make
#make install (install mksquashfs-lzma to
/opt/buildroot-gdb/bin/mksquashfs_lzma-3.0)
```

Linux-2.6.21.x Kernel Version

```
#cd RT288x_SDK/toolchain/mksquash_lzma-3.2
#make
#make install (copy mksquashfs/lzma_alone to /opt/buildroot-gdb/bin/)
```

N.B.: lzma_alone is necessary to generate your own ramdisk image, if you turn on “compress ramdisk by lzma” for RT3052.

```
#make menuconfig

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

6 BOOT-LOADER

6.1 UBOOT CONFIGURATION

```
# tar jxvf Uboot_{version}_{BETA/FINAL}_{date}.tar.bz2
#cd Uboot
#make menuconfig
```

1. Configure your DRAM Size

DRAM Component:

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

DRAM Bus: 16bits / 32bits

Example:

- W9825G6EH: 4Mx4Banksx16bits SDRAM:

- Row Address: A0-A12, Column address: A0-A8
- DRAM Component=256Mb
- DRAM Bus =16bits

- W981216DH/W9812G6DH: 2Mx4Banksx16bits SDRAM:

- Row Address: A0-A11, Column address: A0-A8
- DRAM Component=128Mb
- DRAM Bus =16bits

- IS42S32800B: 2Mx4Banksx32bits SDRAM:

- Row Address: A0-A11, Column address: A0-A8
- DRAM Component=128Mb
- DRAM Bus =32bits

2. LAN/WAN Partition

The switch operates in dump switch mode by default when the board powers up. It will cause the clients on the LAN site get the dynamic ip address from the remote DHCP server connected to WAN port.

Set LAN/WAN Partition to avoid the Client's DHCP request forwarded to the WAN side.

6.2 BUILD UBOOT IMAGE

```
# make
```

```
.....
```

uboot.bin is located in Uboot/.

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

6.3 BURN UBOOT IMAGE

Press '9' in Uboot menuconfig, to enter the invisible menu.

Please choose the operation:

- 1: Load system code to SDRAM via TFTP.
 - 2: Load system code then write to Flash via TFTP.
 - 3: Boot system code via Flash (default).
 - 4: Entr boot command line interface.
 - 5: Load ucos code to SDRAM via TFTP.
- You chose 9
- 9: System Load Boot Loader then write to Flash via TFTP.

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

- Input device IP (10.10.10.123) ==:
- Input server IP (10.10.10.99) ==:
- Input Uboot filename (uboot.bin) ==:

The remainder of this page is intentionally left blank

7 USER LIBRARY

7.1 LIBRARY CONFIGURATION

RT288x_SDK uses ulibc 0.9.28 for user applications. Follow these instructions to change the library default setting.

```
# make menuconfig
```

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

```
[ * ] Customize uClibc Settings
```

```
Target Architecture (mips) --->
Target Architecture Features and Options --->
General Library Settings --->
Networking Support --->
String and Stdio Support --->
Big and Tall --->
Library Installation Options --->
uClibc security related options --->
uClibc development/debugging options --->
---
Load an Alternate Configuration File
Save Configuration to an Alternate File
```

Figure 3 uClibc configuration Menu

7.2 LIBRARY PORTING

Follow these instructions to add a new library to 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

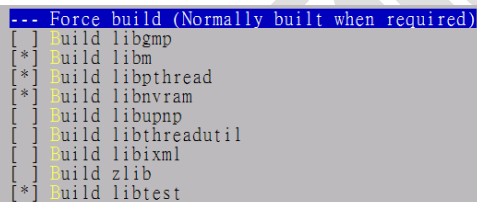
```

bool 'Build libtest'          CONFIG_LIB_LIBTEST_FORCE

# / make menuconfig

```

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



```

--- Force build (Normally built when required)
[ ] build libgmp
[*] build libm
[*] build libpthread
[*] build libnvram
[ ] build libupnp
[ ] build libthreadutil
[ ] build libxml
[ ] build zlib
[*] build libtest

```

Figure 4 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

.....

```

You'll see shared libraries in RT288x_SDK /source/romfs/lib

8 USER APPLICATION

Many useful network applications are provided in RT288x_SDK, to reduce your porting effort. e.g. wan protocol, http server, and debugging tools...etc.

8.1 RALINK PROPRIETARY APPLICATIONS

8.1.1 ATED

Description: for rt2860 v1.4 ATE test program

Usage: ate

N.B.:

- Execute ate on demo board
- Direct connect from LAN port to PC
- Execute QA on PC (wait 30 secs)

8.1.2 REG

Description: register read/write test program

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

N.B.:

- To use system register: reg s 0
- To use wireless register: reg s 1 To use other base address offset: reg s [offset]
- You must insert rt_rdm module first

Example:

```
/ # reg s 0  
  
/ # reg r 18 /* read A0300018 */  
  
/ # reg w 18 12345678 /* write 0x12345678 to A0300018 */
```


8.1.3 FLASH

Description: flash read/write test program

Usage:

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

Example:

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

8.1.4 GPIO

Description: GPIO test program

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

The GPIO testing user application is named *gpio*.

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

Pin sharing scheme

It is important to know what normal function pins are shared with GPIO pins. Only one normal function and GPIO work at a time.

- GPIOMODE: GPIO purpose select)
Configure pins that you would like to use GPIO.
- PIODIR: programmed I/O direction
Configure the direction of all GPIO pins you would like to use GPIO. '1' is an output, and '0' is an input.
- PIODATA: programmed I/O data
You may write data for output GPIO pins, and read data for input GPIO pins. PIOSET, PIORESET, PIOTOG are also used for manipulating GPIO data bits.

- Using GPIO pins as input that raising interrupts, you should also configure PIOINT, PIOEDGE, PIORENA, and PIOFMASK.

8.1.5 MII_MGR

Description: mii register read/write test program

Usage:

- get: mii_mgr -g -p [phy number] -r [register number]
- set: mii_mgr -s -p [phy number] -r [register number] -v [0xvalue]

Example:

- get: mii_mgr -g -p 3 -r 4
- set: mii_mgr -s -p 4 -r 1 -v 0xff11

Kernel Module:

\$SDK/source/\$LINUX/drivers/flash/mii_mgr_main.c

\$SDK/source/\$LINUX/drivers/flash/flash_ioctl.h

- IOCTL Commands
 - FLASH_IOCTL_MII_MGR_GET
 - Get phy register via mdc/mdio interface.
 - FLASH_IOCTL_MII_MGR_SET
 - Set phy register via mdc/mdio interface.
- IOCTL interface

In \$SDK/source/user/rt2880_app/mii_mgr/flash_ioctl.h

```
struct mii_mgr_opt {
    unsigned int    phy_addr;
    unsigned int    reg_addr;
    unsigned int    *fromvalue;
    unsigned int    tovalue;
    unsigned int    result;
};
```

- phy_addr: Address of PHY device
- reg_addr: Register addresses within PHY device
- fromvalue:
 - GET: the phy register data that is read from phy
 - SET: the current register data after MDIO setting
- tovalue: the phy register data that wants to be set

User applications would execute mii_mgr commands through the ioctl interface to the device `/dev/flash0`.

8.1.6 MTD

Description: MTD writing program for firmware update

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

Example: `mtd_write -r write image.bin mtd4`

8.1.7 NVRAM

Description:

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

For configurations in NVRAM, or setting NVRAM values according to a given file

`nvrn_daemon` is a background daemon and register to receive interrupt from GPIO pin 0. If SIGUSR1 is received (user one-clicked GPIO pin 0 button), `nvrn_daemon` will notify GoAhead web server to start WPS PBC process by sending it SIGUSR1. If SIGUSR2 is received (user pressed GPIO pin 0 button for several seconds), `nvrn_daemon` will restore the system configurations to default value.

Usage:

- a. get: `nvrn_get [<2860/inic>] <field>`
- b. set: `nvrn_set [<2860/inic>] <field>`
- c. init: `ralink_init <command> [<platform>] [<file>]`

Commands:

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

Platform:

- `2860` - `rt2860` station
- `inic` - intelligent nic

File:

File name for `renew` command

daemon: `nvr_daemon`

Example:

- `nvr_get 2860 SSID` /* get SSID */
- `nvr_set 2860 SSID ralink` /* set SSID to ralink */
- `ralink_init gen 2860` /* generate `RT2860 .dat` file from `NVRAM` */
- `ralink_init show inic` /* display `INIC` configurations in `NVRAM` */
- `ralink_init renew 2860 ra.dat` /* set `NVRAM` values for `RT2860` paltform according to `ra.dat` file */
- `nvr_daemon` /* start the `nvr_daemon` */

8.1.8 SPICMD

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

Usage: `spicmd read/write parameters`

N.B.:

`spicmd read address`

`spicmd write size address value`

size is 1, 2, 4 bytes

8.1.9 I2CCMD

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

Usage: i2ccmd read/write parameters

N.B.:

i2ccmd read address

i2ccmd write size address value

size is 1, 2, 4 bytes

8.1.10 SCRIPT

Description: WebUI configuration script

Usage: see script help message in detail.

8.2 GOAHEAD

Source code: RT288x_SDK/source/user/goahead/

Description: WebUI reference design of AP/Router Solution.

8.3 NVRAM LIBRARY

Source code: RT288x_SDK/source/lib/libnvram

Description: Library for nvram_get, nvram_set, ralink_init.

8.4 WSC_UPNP

Source code: RT288x_SDK/source/user/WSC_UPNP

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

Required library: libupnp, pthread

8.5 IPTABLES

Source code: RT288x_SDK/source/user/iptables # for Linux-2.4

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

Description: administration tool for IPv4 packet filtering and NAT

8.6 NTPCLIENT

Source code: RT288x_SDK/source/user/ntpclient

Description: ntpclient is an NTP (RFC-1305) client for unix-alike computers. Its functionality is a small subset of xntpd, but IMHO performs better (or at least has the potential to function better) within that limited scope. Since it is much smaller than xntpd, it is also more relevant for embedded computers.

8.7 MTD-UTILS

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

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

8.8 PPP-2.4.2

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

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

8.9 BRIDGE-UTILS

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

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

8.10 WIRELESS_TOOLS

Source code: RT288x_SDK/source/user/ wireless_tools

Description: This package contains the Wireless tools, used to manipulate the Wireless Extensions.

The Wireless Extensions is an interface allowing you to set Wireless LAN specific parameters and get the specific stats.

8.11 INADYN

Source code: RT288x_SDK/source/user/ inadyn

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

8.12 ZEBRA-0.95A_RIPD

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

Description: GNU Zebra is free software that manages various IPv4 and IPv6 routing protocols.

Currently GNU Zebra supports BGP4, BGP4+, OSPFv2, OSPFv3, RIPv1, RIPv2, and RIPng.

8.13 WPA_SUPPLICANT-0.5.7

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

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

8.14 TOTD-1.5

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

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

8.15 SAMBA-3.0.2

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

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

8.16 RADVD-1.0

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

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

8.17 PPTP-CLIENT

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

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

8.18 RP-L2TP-0.4

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

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

8.19 CTORRENT-DNH3.2

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

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

8.20 DHCP6

Source code: RT288x_SDK/source/user/ dhcp6

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

8.21 DNSMASQ-2.40

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

Description: Dnsmasq is a lightweight, easy to configure DNS forwarder and DHCP server. It is

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

8.22 IGMPPROXY

Source code: RT288x_SDK/source/user/ igmpproxy

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

8.23 MATRIXSSL-1.8.3

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

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

8.24 RP-PPPOE-3.8

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

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

8.25 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
OBSJ = hello.o
CFLAGS +=
```

```
all: $(EXEC)

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

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
```

1. Hit 'Select the Product you wish to target' to choice target platform.

```
(RT2880) Ralink Products
(2M/16M) Flash/SDRAM Size
```

2. Select a 'Flash/SDRAM' Size

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

N.B.:

1. Choose the target platform type (RT2880 or RT3052.)
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.

Hit 'Kernel/Library/Defaults Selection' to enter configuration menu, and hit space on the 'Default all settings' item.

```

--- 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

```

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

```

+-----+
| Do you wish to save your new kernel configuration? |
+-----+
| < Yes >      < No > |
+-----+

```

The script retrieves all user/kernel default settings. The following message will be shown after retrieving the default settings.

```

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

```

N.B.: The default configuration file will be stored in a different file depending on the 'Flash/DRAM size' settings. Go to RT288x_SDK/source/vendors/Ralink/{RT2880/RT3052}/config to see all default setting files.

- a. Busybox default configuration files

✓ 2M_16M_config.busybox-2.4.x

- ✓ 4M_16M_config.busybox-2.4.x
 - ✓ 8M_16M_config.busybox-2.4.x
- b. User application default configure file
- ✓ 2M_16M_config.vendor-2.4.x
 - ✓ 4M_16M_config.vendor-2.4.x
 - ✓ 8M_16M_config.vendor-2.4.x
- c. uClibc default configure file
- ✓ 4M_16M_config.uclibc-2.4.x
 - ✓ 2M_16M_config.uclibc-2.4.x
 - ✓ 8M_16M_config.uclibc-2.4.x
- d. Linux kernel 2.4 default configure file
- ✓ 2M_16M_config.linux-2.4.x
 - ✓ 4M_16M_config.linux-2.4.x
 - ✓ 8M_16M_config.linux-2.4.x

9.2 CHANGE FLASH/DRAM SIZE

If you increase/decrease the size of DRAM, please changes the DRAM size setting using “make menuconfig”.

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

● Linux 2.4

```
(RT2880-ASIC) RT2880 Chip Type
(32M) DRAM Size
(4M) Flash Size
```

● Linux 2.6

```
System type (Ralink RT3052 board) --->
Soc Hardware Type (RT3052-ASIC) --->
DRAM Size (32M) --->
Flash Size (8M) --->
Root File System Type (RootFS_in_RAM) --->
```

9.3 CHANGE SWITCH CONTROLLER IN RT2880 PLATFORM

Currently, RT288x_SDK supports IC+ 175C/D switch controller in RT2880 platform. You can use 'make menuconfig' to configure your switch controller setting.

#make menuconfig

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

  [*] Customize Kernel Settings

    Network device support --->

      Ralink Driver --->
```

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

In LAN/WAN Partition item, W/LLLL means P0 is WAN port, and LLLL/W means P4 is WAN Port. The switch is configured by script not Ethernet driver. Please see config-vlan.sh in

RT288x_SDK/source/user/rt2880_app/ scripts.

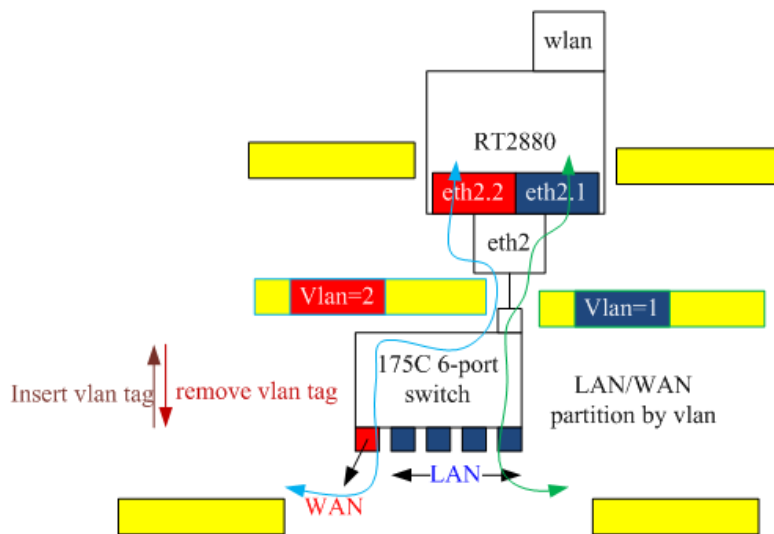


Figure 5 IC+ 10/100 Switch Operation Diagram

9.4 UPDATE USER/KERNEL DEFAULT SETTINGS

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

Hit "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

```
#make dep  
  
#make
```

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

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

N.B.: 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 new image (make = make lib_only user_only modules romfs Linux image)

9.6 PORT NEW LINUX KERNEL MODULE

Example: Port the hello networking module to the RT2880 platform

1. Add the source code to the rt2880 directory

```
# mkdir RT288x_SDK/source/linux-2.4.x/drivers/net/hello  
# vi RT288x_SDK/source/linux-2.4.x/drivers/net/hello/Makefile  
  
O_TARGET := hello.o  
obj-y    := main.o  
obj-m    := $(O_TARGET)  
include $(TOPDIR)/Rules.make  
  
# vi RT288x_SDK/source/linux-2.4.x/drivers/net/hello/main.c  
  
#include <linux/init.h>  
#include <linux/module.h>
```



```
static int hello_init(void)
{
    printk("hello world\n");
    return 0;
}

static void hello_exit(void)
{
    printk("goodbye\n");
}

module_init(hello_init);
module_exit(hello_exit);
MODULE_LICENSE("GPL");

~
```

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

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

3. Modify Config.in

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

4. Turn on the hello module

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

5. Compile the source code

```
#make dep
#make
```

6. Test

```
/ # insmod hello
hello world

/ #
```

9.7 EXECUTE COMMANDS AT BOOT UP TIME

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

```
#!/bin/sh
mount -a
goahead&  <-- add new command here
```

9.8 ADD NEW FILES IN ROOTFS

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

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

Example: add xxx.bin to rootfs

- a. copy xxx.bin to RT288x_SDK/source/vendors/Ralink/{RT2880/RT3052}
- b. edit RT288x_SDK/source/vendors/Ralink/{RT2880/RT3052}/Makefile

```
romfs:
    $(ROMFSINST) /etc_ro/xxx.bin
```

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

The remainder of this page is intentionally left blank

9.9 IMAGE DOWNSIZE

MTD partitions are listed in below.

RootFS in RAM Mode

mtb 0	uboot	0x0
mtb 1	config	0x30000
mtb 2	RF	0x40000
mtb 3	Kernel/RootFS	0x50000 0x400000

RootFS in Flash Mode

mtb 0	Uboot	0x0
mtb 1	Config	0x30000
mtb 2	RF	0x40000
mtb 3	Kernel	0x50000
	Padding	<menuconfig>
mtb 4	Root FS	0x400000

In RootFS in Flash mode, If kernel image size is smaller than the size of mtd3, image builder will add padding bit in the end of kernel image. So, we need to adjust the size of mtd3 to save flash memory.

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

```
#make image
.....

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

# Original Kernel Image Size

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

# Padded Kernel Image Size

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

# Original RootFs Size

4329746 /home/steven/RT288x_SDK/source/romfs

# Compressed RootFs Size

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

# Padded Kernel Image + Compressed Rootfs Size

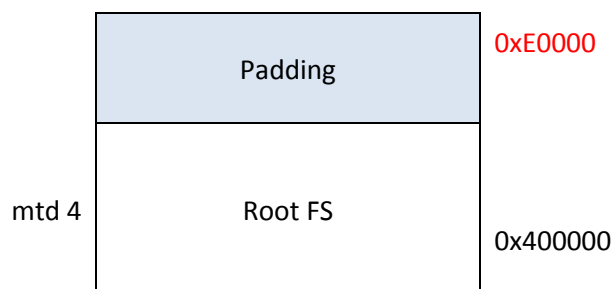
1855424 /home/steven/RT288x_SDK/source/images/zImage.lzma

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

Step2: Change mtddblock size

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

mtd 0	Uboot	0x0
mtd 1	Config	0x30000
mtd 2	RF	0x40000
mtd 3	Kernel	0x50000



host:\$ make menuconfig

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

```
(linux-2.4.x) Kernel Version
[ ] Default all settings (lose changes)
[*] Customize Kernel Settings
[ ] Customize Vendor/User Settings
[ ] Customize Busybox Settings
[ ] Update Default Vendor Settings
```

```
Code maturity level options --->
Loadable module support --->
Machine selection --->
CPU selection --->
General setup --->
```

```
(RT2880-ASIC) RT2880 Chip Type
(32M) DRAM Size
(4M) Flash Size
(RootFS_in_Flash) RT2880 Root File System
(90000) MID Kernel Partition Size (Unit:Bytes)
```

The remainder of this page is intentionally left blank

10 FLASH LAYOUT AND FIRMWARE UPGRADE

10.1 FLASH LAYOUT

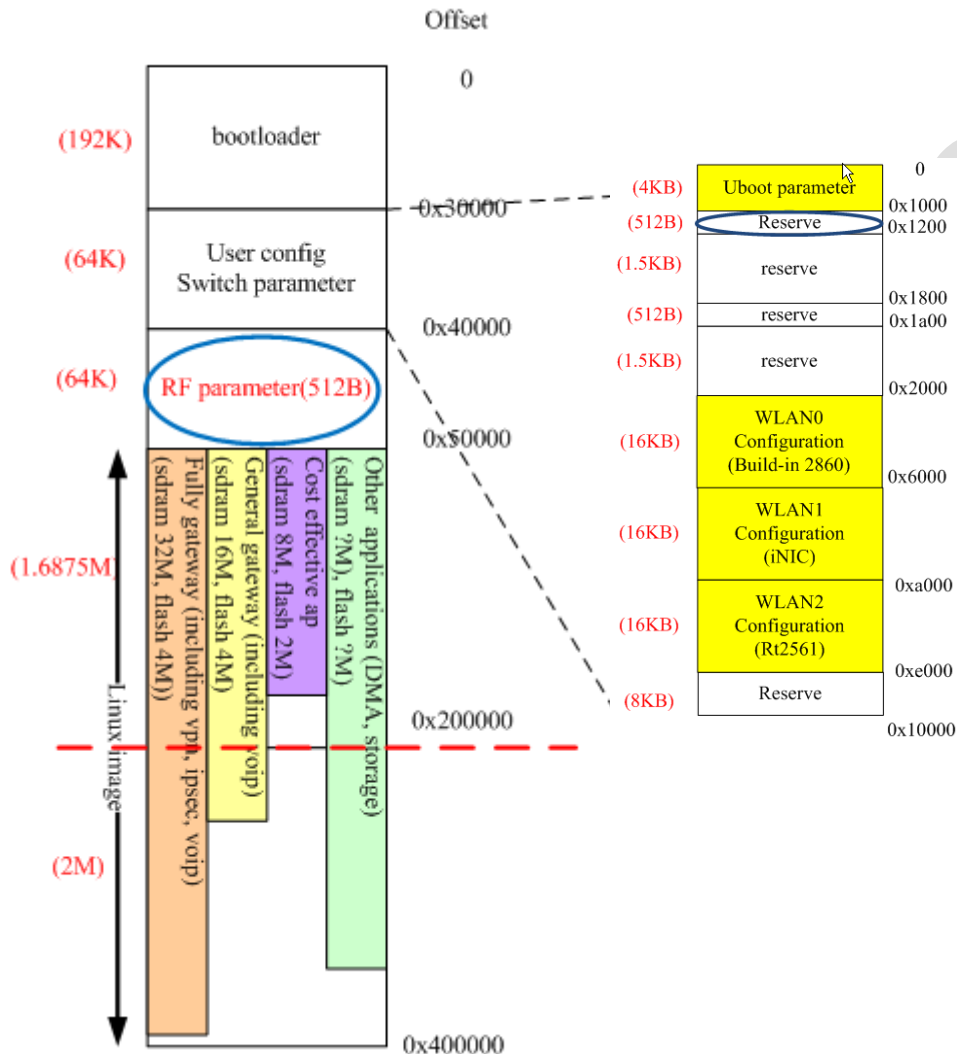


Figure 6 Ralink SDK Flash Layout (4MB)

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

10.2 FIRMWARE UPGRADE

10.2.1 BY UBOOT

```

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

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

SDRAM bus set to 32 bit
SDRAM size =32 Mbytes

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

```

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

10.2.2 BY WEBUI

You can use WebUI to upgrade the Linux image.

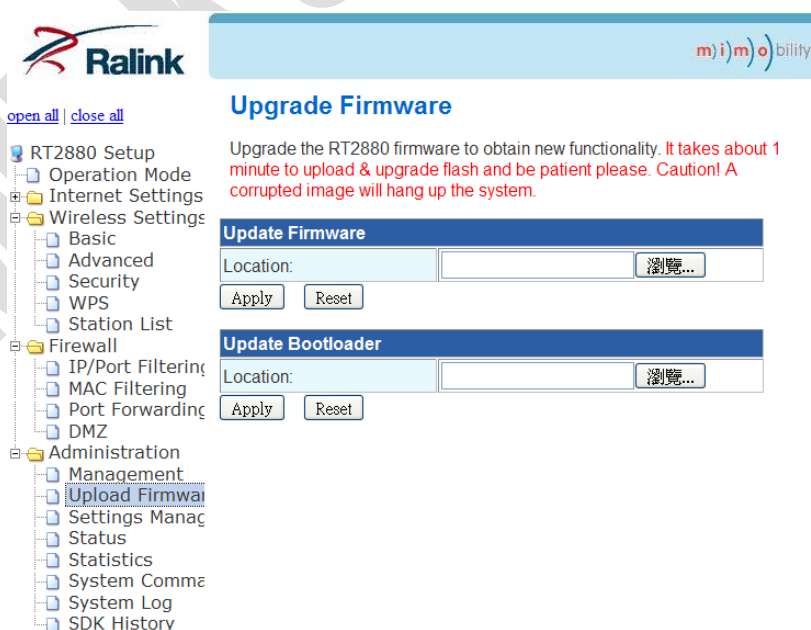


Figure 7 WebUI Firmware Upgrade

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 bytes = MTTD kernel partition size in "make menuconfig")

```
(RT2880-ASIC) RT2880 Chip Type
(32M) DRAM Size
(4M) Flash Size
(RootFS in Flash) RT2880 Root File System
(B0000) MTD Kernel Partition Size (Unit:Bytes) (NEW)
```

The remainder of this page is intentionally left blank

11 FAQ

11.1 RT2880 DEFAULT PASSWORD/UART/NETWORKING SETTING

Table 3 Networking Setting

LAN	IP Address	10.10.10.254
	Subnet	255.255.255.0
WAN	IP Address	DHCP

Table 4 UART Setting

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

Table 5 Web Setting

Item	Default Value
User Name:	admin
Password:	admin

11.2 SYSTEM REQUIREMENTS FOR HOST PLATFORM

RT2880 SDK uses Fedora 6 Host to build the image. Change your Linux distribution If you can't build the image successfully.

Table 6 Requirements of Host Platform

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

HD

40G

11.3 HOW TO ADD NEW DEFAULT PARAMETER IN FLASH

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

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

11.3.1 EXAMPLE 1

Add a new default parameter - WHOAMI for IC+ platform

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

```
WHOAMI=steven
```

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

```
#ralink_init clear 2860  
#reboot
```

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

11.3.2 EXAMPLE 2

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

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

```
RadioOn=1
```

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

```
radio = websGetVar(wp, T("radiohiddenButton"), T("2"));

if (!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

```

11.4 ENABLE ETHERNET CONVERTER FEATURE

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

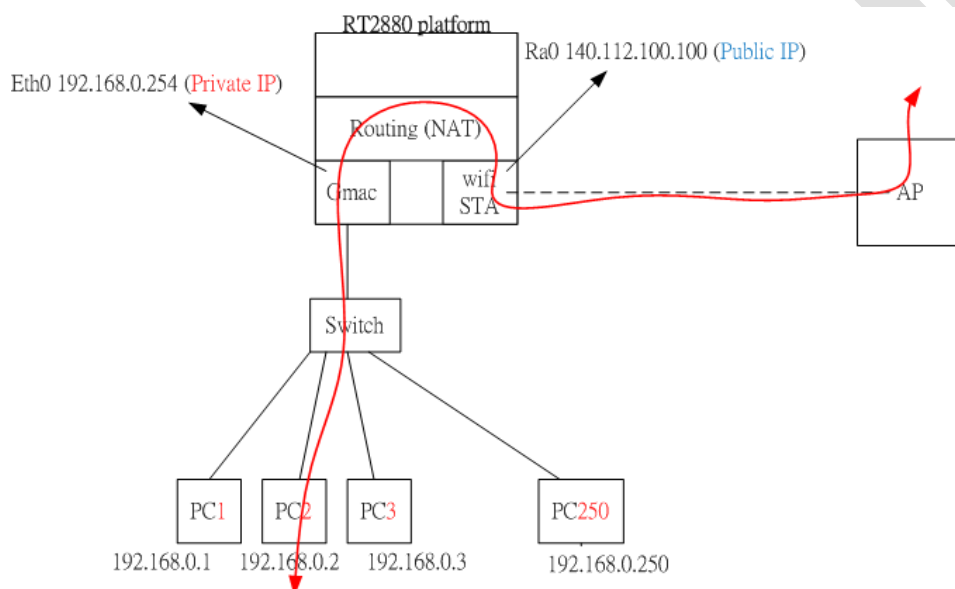


Figure 8 Ethernet Converter Operation Diagram

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

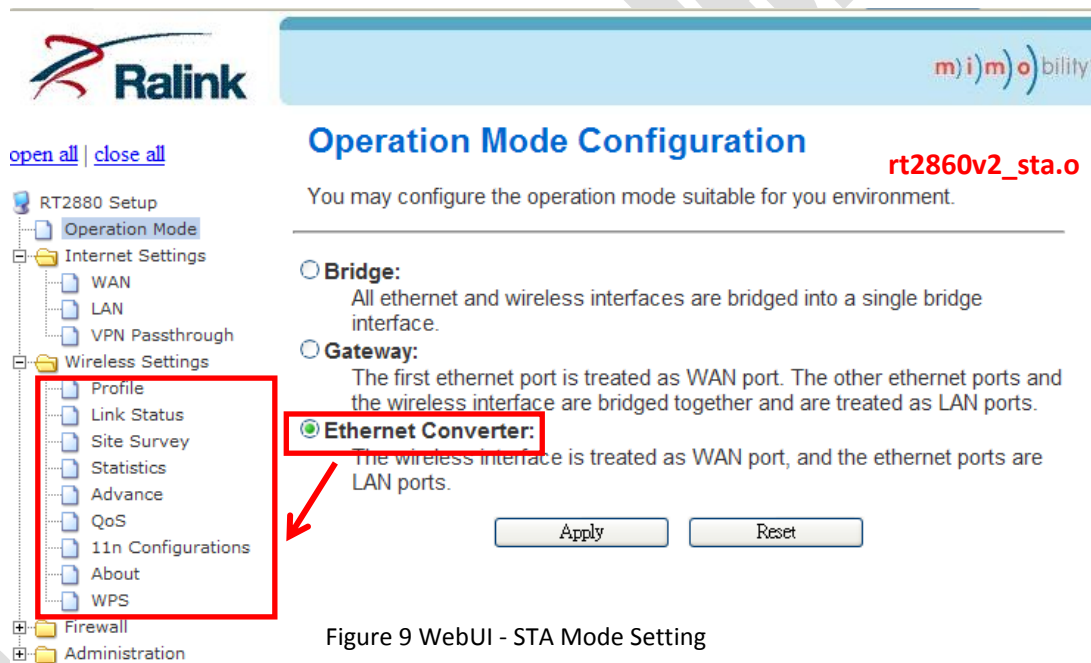
```

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

```

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

Select the operation mode on the "Operation Mode Configuration" web page.



Operation Mode Configuration rt2860v2_sta.o

You may configure the operation mode suitable for you environment.

☐ **Bridge:**
All ethernet and wireless interfaces are bridged into a single bridge interface.

☐ **Gateway:**
The first ethernet port is treated as WAN port. The other ethernet ports and the wireless interface are bridged together and are treated as LAN ports.

☒ **Ethernet Converter:**
The wireless interface is treated as WAN port, and the ethernet ports are LAN ports.

Figure 9 WebUI - STA Mode Setting

11.5 CHANGE RF CHIP FROM RT2820 TO RT2850 IN RT2880 PLATFORM

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

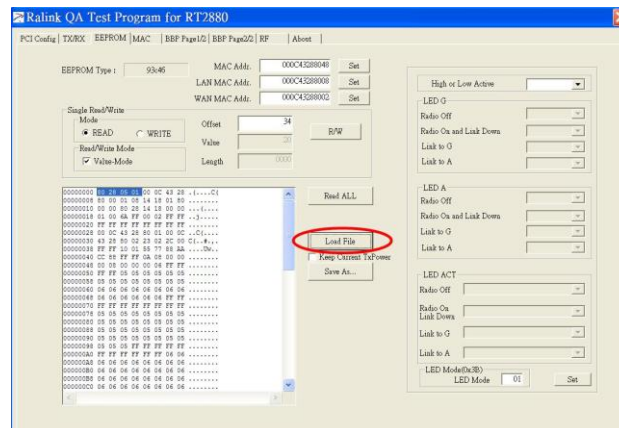


Figure 10 QA – Burn your own EEPROM binary file

11.6 HOW TO CHANGE ETHERNET MAC ADDRESS

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

```
#define GMAC0_ADDR      (RT_EEPROM_BASE + 0x28)

#define GMAC1_ADDR      (RT_EEPROM_BASE + 0x2E)
```

N.B.: If you need the LAN/WAN Ports to have different mac address, modify the Ethernet driver to get GMAC0_ADDR for LAN, and GMAC1_ADDR for WAN.

Please use the QA program to modify your flash content.

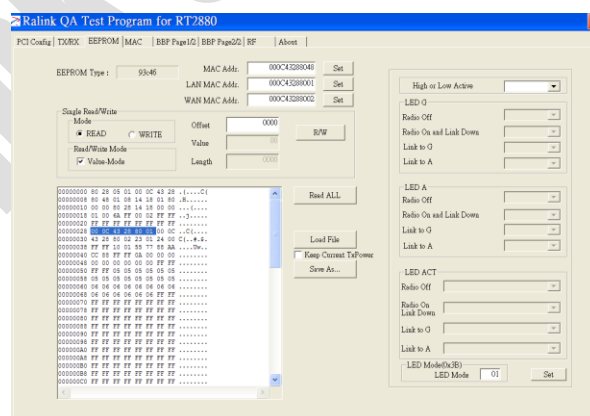


Figure 11 QA – Modify GMAC Mac address

11.7 HOW TO CONFIGURE GPIO PORTS

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

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

- **RALINK_GPIO_SET_DIR** - Configure the direction of GPIO pins using bitmaps. Bit 1 means output, and bit 0 means input. For example, value 0x5 means to configure GPIO pin 0 and 2 as output pins, and 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 means to configure 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 means to configure GPIO pin 0 and 2 as output pins, and other pins are ignored.
- **RALINK_GPIO_READ** - Read the value from GPIO data.
- **RALINK_GPIO_WRITE** - Write a value to GPIO data.
- **RALINK_GPIO_SET** - Set a value with corresponding bits on to GPIO data. For example, value 0x5 means GPIO data bit 0 and 2 will be set to 1, and other bits will be ignored.
- **RALINK_GPIO_CLEAR** - Clear a value with corresponding bits off to GPIO data. For example, value 0x5 means GPIO data bit 0 and 2 will clear to 0, and other bits will be ignored.
- **RALINK_GPIO_READ_BIT** - Read the corresponding bit from GPIO data. For example, bit 2 mean to read the third bit from GPIO data.
- **RALINK_GPIO_WRITE_BIT** - Write a corresponding bit to GPIO data. For example, bit 2 and value 1 means to write value 1 to the third bit of GPIO data.
- **RALINK_GPIO_READ_BYTE** - Read the corresponding byte from GPIO data. For example, byte 2 means to read the third byte from GPIO data.
- **RALINK_GPIO_WRITE_BYTE** - Write a corresponding byte to GPIO data. For example, byte 2 and value 0x33 means to write value 0x33 to the third byte of GPIO data.

- RALINK_GPIO_READ_INT - Same as RALINK_GPIO_READ.
- RALINK_GPIO_WRITE_INT - Same as RALINK_GPIO_WRITE.
- RALINK_GPIO_SET_INT - Same as RALINK_GPIO_SET.
- RALINK_GPIO_CLEAR_INT - Same as RALINK_GPIO_CLEAR.
- RALINK_GPIO_ENABLE_INTP - Enable GPIO input interrupt.
- RALINK_GPIO_DISABLE_INT - Disable GPIO input interrupt.

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

11.8 USE GPIO TO TURN ON LED

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

Table 7 GPIO Usage of RT2880

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

Table 8 GPIO Usage of RT3052

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

Ralink SDK GPIO driver provides an interface to configure the frequency of LEDs connecting to 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 configured 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 your application to configure the 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 currently 4000.

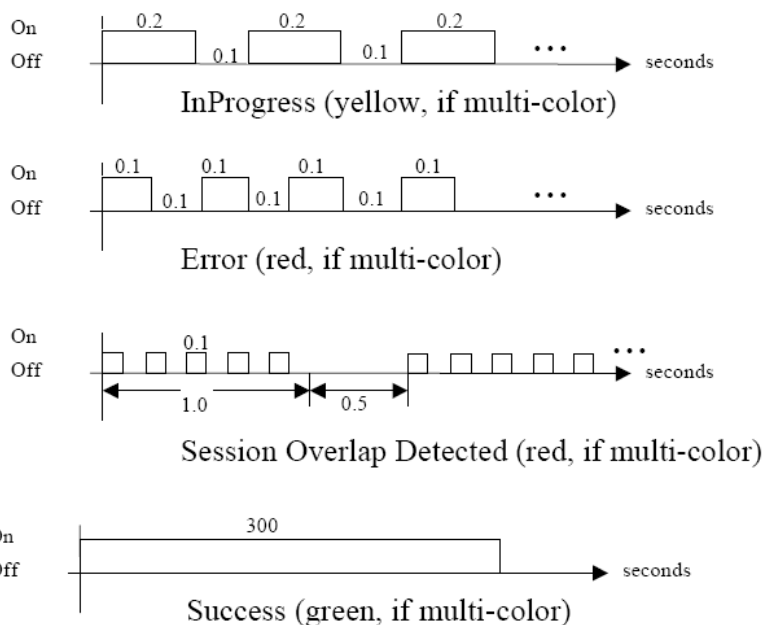


Figure 12 LED Definition of WPS Specification

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

- InProgress: gpio | <gpio> 2 1 4000 0 4000 (i.e. 2 ticks bright, 1 tick dark, blinking forever.)
- Error: gpio | <gpio> 1 1 4000 0 4000 (i.e. 1 tick bright, 1 tick dark, blinking forever.)
- Session Overlap Detected: gpio | <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 | <gpio> 3000 1 1 1 1 (i.e. 3000 ticks bright, 1 tick dark, blinking for one on-offs and one time.)

- To turn the LED on and keep it on: gpio l <gpio> 4000 0 1 0 4000
- To turn the LED off and keep it off: gpio l <gpio> 0 4000 0 1 4000

11.9 USE LED FIRMWARE TO TURN ON LED

1. enable LED firmware

```
#make menuconfig

Kernel/Library/Defaults Selection  --->

[*] Customize Kernel Settings

Network device support  --->

Ralink Driver  --->
```

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

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

Table 9 RT2880 LED Parameters in Flash

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

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

Address	States	Bit	LED-behavior
40h	Radio-off	[3:0]	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
	Radio-on but link down	[7:4]	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
41h	Radio-on and link to G	[11:8]	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
	Radio-on and link to A	[15:12]	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

The current Ralink default flash hex values are as follows:

RT2880 Flash Base Address=0x40000

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

11.10 HOW TO START TELNET SERVER

Check RT288x_SDK/source/user/busybox/.config

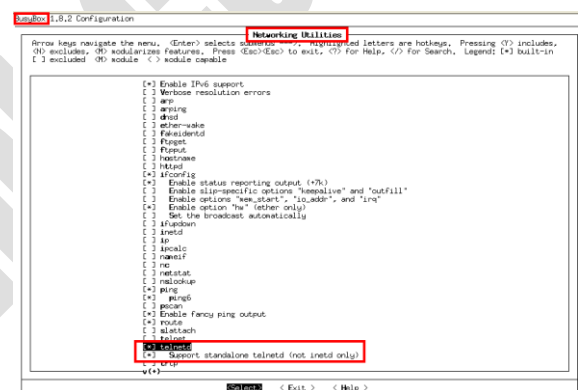
11.10.1 BUSYBOX SETTING

CONFIG_FEATURE_DEVPTS=y → General Configuration
CONFIG_FEATURE_SUID=y → General Configuration
CONFIG_LOGIN=y → Login/Password Management Utilities
CONFIG_TELNETD=y → Networking utilities
CONFIG_FEATURE_TELNETD_STANDALONE=y

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

11.10.2 LINUX SETTING

CONFIG_UNIX98_PTYS=y → Character devices
CONFIG_UNIX98_PTY_COUNT=256
CONFIG_DEVPTS_FS=y → File systems



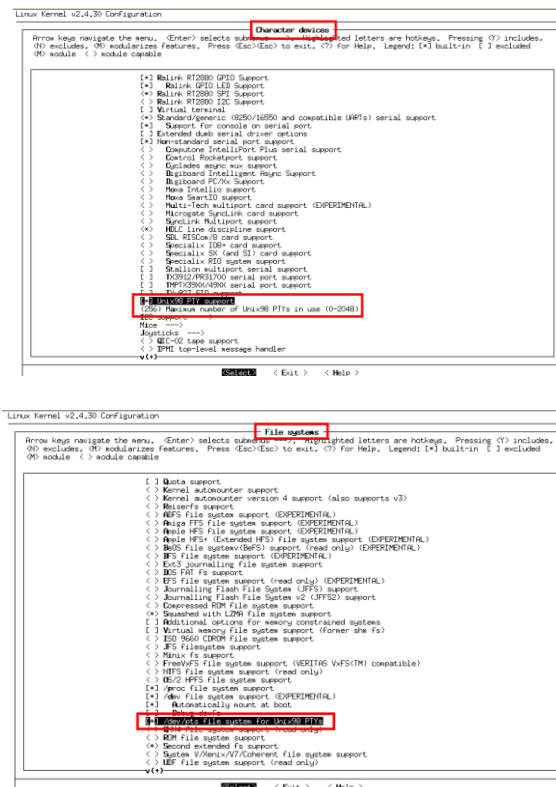


Figure 13 Configuration Procedure of Telnet Server

11.11 11N BIT RATE DERIVATION

- The BitRate of 11n need below information on MAC driver and the real rates will be triggered by PHY layer depends on below three factors.

- MCS
- BW
- GI

- Bandwidth:

Data subcarriers on different bandwidths,
20MHz and 40MHz.

- N_{SD} : Number of data subcarriers.

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

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

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

$$= 2.0769230769230769230769230769231$$

b. Example:

$$\text{MCS}=15, \text{GI}=800\text{ns}, \text{BW}=20\text{MHz}, \text{DataRate} = 130\text{Mbps}$$

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

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

$$= 270\text{Mbps}$$

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

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

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

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

3. Guard Interval.

a. Definition:

$$T_{\text{sym}}: 4\mu\text{s}, \text{ Symbol Interval}$$

$$T_{\text{syms}}: 3.6\mu\text{s}, \text{ Symbol interval of Short GI.}$$

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

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

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.



PHY spec, v1.27

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

3) EWC PHY spec. page 13.



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 $l = 3(\lceil N_{data} \rceil + N_{LTF} + 3) - 3$ 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 $\frac{4}{16}$. N_{LTF} is the number of HT training symbols. The symbol $\lceil x \rceil$ denotes the lowest integer greater or equal to x .

11.12 HOW TO BUILD SINGLE IMAGE FOR FLASH PROGRAMMER

Example: Generate 4M single image for rt2880 platform (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_ulmage
```

```
#
# Single Image Information
#

PACKED_DIR = .
PACKED_IMAGE = steven_ulmage.img
```

```
#cp /tftpboot/uboot.bin .
#cp /tftpboot/steven_ulmage .
#make -f Makefile.4M
```

Flash layout:

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

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

-Original Uboot Image Size

149372 ./uboot.bin

- Original Kernel Image Size

2779348 ./steven_ulmage

- Packed Image Size

4194304 ./steven_ulmage.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_ulmage
-rwxr-xr-x 1 steven users 4194304 Jun 27 13:36 steven_ulmage.img*
-rwxr-xr-x 1 steven users 149372 Jun 27 13:34 uboot.bin*
```

Now, the single image can be burned using flash programmer.

11.13 HOW TO POWER DOWN RT305X ETHERNET PORTS

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

MII control register

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

User Space:

```
# mii_mgr -s -p 0 -r 0 -v 0x3900 //set port 0 register0 bit11
Set: phy[0].reg[0] = 3900

# mii_mgr -s -p 1 -r 0 -v 0x3900 //set port 1 register0 bit11
Set: phy[1].reg[0] = 3900

# mii_mgr -s -p 2 -r 0 -v 0x3900 //set port 2 register0 bit11
Set: phy[2].reg[0] = 3900
```

```
# mii_mgr -s -p 3 -r 0 -v 0x3900 //set port 3 register0 bit11  
  
Set: phy[3].reg[0] = 3900  
  
# mii_mgr -s -p 4 -r 0 -v 0x3900 //set port 4 register0 bit11  
  
Set: phy[4].reg[0] = 3900
```

Kernel Space:

```
extern u32 mii_mgr_read( unsigned int , unsigned int, unsigned int *);  
  
extern u32 mii_mgr_write( unsigned int, unsigned int, unsigned int);  
  
mii_mgr_write( 0, 0, 0x3900) //set port 0 register0 bit11  
  
mii_mgr_write( 1, 0, 0x3900) //set port 1 register0 bit11  
  
mii_mgr_write( 2, 0, 0x3900) //set port 2 register0 bit11  
  
mii_mgr_write( 3, 0, 0x3900) //set port 3 register0 bit11  
  
mii_mgr_write( 4, 0, 0x3900) //set port 4 register0 bit11
```

11.14 HOW TO ENABLE NFS CLIENT

```
#make menuconfig  
  
Kernel/Library/Defaults Selection--->  
Networking options --->  
[*] IP: kernel level autoconfiguration  
File systems --->  
Network File Systems --->  
  
Linux 2.4:  
  
<*> NFS file system support  
[*] Provide NFSv3 client support  
[*] Allow direct I/O on NFS files (EXPERIMENTAL)  
[*] Root file system on NFS
```

```
Linux 2.6
<*> NFS file system support
[*] Provide NFSv3 client support
[*] Provide client support for the NFSv3 ACL protocol extension
[*] Provide NFSv4 client support (EXPERIMENTAL)
[*] Allow direct I/O on NFS files

Kernel/Library/Defaults Selection--->
[*] Customize Kernel Settings (NEW)
[*] Customize Busybox Settings
Linux System Utilities--->
[*] mount
[ ] Support mount helpers
[*] Support mounting NFS file systems
```

Example:

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

# mount

.....

/dev/sda1 on /media/sda1 type vfat
(rw,mask=0000,dmask=0000,codepage=cp437,ioccharset=iso8859-1)

192.168.18.21:/tftpboot on /mnt type nfs
(rw,vers=3,rsize=32768,wsize=32768,hard,nolock,proto=udp,timeo=7,retrans=3,sec=sys,addr=192.16
8.18.21)
```

The remainder of this page is intentionally left blank

11.15 HOW TO ADD A NEW LANGUAGE TO WEB UI

Take Korean for example:

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:

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

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

Add to 'getLangBuilt' function:

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

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

Add to 'initValue' function:

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

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

Add to 'initValue' function:

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

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

Fix 'initLanguage' function

8. make menuconfig

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

11.16 HOW TO ENABLE WATCHDOG IN RT305X


```
#make menuconfig
```

```
Machine selection  --->
```

```
[*] Ralink WatchDog
```

```
[*]   Ralink WatchDog Timer
```

```
[*]     Ralink WatchDog Reset Output
```

```
(10)   WatchDog Timer (Unit:1Sec, Max=30Sec)
```

```
(4)     WatchDog Refresh Interval (Unit:1Sec, Max=30Sec)
```

1. When watchdog detects watchdog module do not update register periodically, it will active low for 3 system clocks on SRAM_CS_N pin if you enable "Ralink WatchDog Reset Output"
2. **WatchDog Timer:** The expired time that watchdog circuit start reset procedure.
3. **WatchDog Refresh Interval:** The update interval of watchdog module

11.17 HOW TO ENABLE USB STORAGE IN 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)

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

Notes: Because you enable many features in kernel, please pay attention that kernel size can't bigger than your MTD kernel partition size in Rootfs in flash mode.

#=====

Original Kernel Image Size

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

Padded Kernel Image Size

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

Original RootFs Size

.....

11.18 HOW TO ENABLE USB AUTOMOUNT IN 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
```