OpenWrt OpenWrtDocs

Welcome to the new wiki version of the OpenWrt userguide.

To get started, click a link below.

About OpenWrt

- About OpenWrt
- Why should I run OpenWrt?
- OpenWrt Version History

Installing OpenWrt

- Will OpenWrt work on my hardware ?
- Obtaining the firmware
- Installing OpenWrt
  - General instructions (router specific instructions later)
  - Linksys WRT54G and WRT54GS
    - Enabling boot_wait
    - Setting boot_wait from a serial connection
  - ASUS WL-500G and WL-300G
  - ASUS WL-500G Deluxe
  - Siemens Gigaset SE505
  - Motorola WR850G
  - Buffalo Airstation WLA-G54
  - Buffalo AirStation WBR2-G54S
- Using OpenWrt
- Troubleshooting

Using OpenWrt

- Using OpenWrt for the first time
- Firstboot / jffs2
- Editing Files
- ipkg
- Configuration

OpenWrt Configuration

- NVRAM
- Network configuration
  - Sample network configurations
  - The ethernet switch
    - Normal Behavior
    - Using Robocfg
- Wireless configuration
  - Basic settings
  - WEP encryption
  - WPA encryption
  - Wireless Distribution System (WDS) / Repeater / Bridge
  - OpenWrt as client / wireless bridge
- Software configuration
  - System
    - dnsmasq
    - nas
    - wl
    - TimeZone and NTP
    - Crontab
    - PPPoE Internet Connection
    - Access to syslog
  - Applications
    - httpd
    - socks-Proxy

    - JTAG-adaptor Method
    - Shorting Pins Method
- Using the system logs for additional troubleshooting
- Some routers have screws
- Problems going from jffs2 to squashfs or problems booting after reflashing
- Problems accessing the wireless interface
- Source port mismatch with atftp
- Getting help

Deinstalling OpenWrt

- How I remove OpenWrt ?
- Can I use TFTP to go back to the original firmware ?

Glossary

- afterburner
- failsafe
- firstboot
- jffs2
- nvram
- WRT
- SpeedBooster
- squashfs

Almost all of these pages are editable, create an account and click the edit () button at the top of the page.

This is a table of all supported devices as of 2005/10/12. Legend:

- **Supported** - supported in "White Russian"
- **Partial** - partially supported, no support for the wireless card.
- **Untested** - should work in theory but never tested (for additional hardware support, the developers are always happy to accept donations)
- **No** - confirmed that this device is not supported yet
- **WiP** - Work in Progress

Manufacturer	Model	Version	Platform & Frequency	Flash	RAM	Wireless NIC	Switch	boot_wait	Serial	JTAG	USB	Status & Notes
3COM	3CRWE554G72TU-US										No	untested
ALLNet	ALL130DSL (aka Sercomm IP505 ???)		Texas Instruments Sangam/AR7 @150MHZ	2MB	8MB							WiP
ALLNet	ALL0277DSL (aka Sercomm IP806 ???)	v2	Texas Instruments Sangam/AR7 @150MHZ	2MB	16MB	ACX-111	Marvell 88E6060		Yes	No	No	WiP
ALLNet	ALL0277		Broadcom 4710	4MB								Untested
Asus	WL-300G		Broadcom 4710 @ 125MHz	4MB	16MB	integrated Broadcom	None	on			No	Supported
Asus	WL-500B	1	Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI	BCM5325	on			1x v1.1	Supported
Asus	WL-500B	2	Broadcom 4710 @ 125MHz	4MB	16MB	Ralink mini-PCI	BCM5325	on			1x v1.1	Untested
Asus	WL-500G		Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI	BCM5325	on			1x v1.1	Supported
Asus	WL-520G		Broadcom 5350 @ 200MHz	2MB	8MB	integrated Broadcom	integrated into CPU	on			No	Untested
Asus	WL-500G Deluxe		Broadcom 5365 @ 200MHz	4MB	32MB	integrated Broadcom	integrated into CPU	on	Yes	No	2x v2.0	Supported
Asus	WL-700G		Broadcom 4780 @ 300MHz								3x v2.0	Untested
Asus	WL-HDD		Broadcom 4710 @ 125MHz	4MB	16MB	integrated Broadcom	None	on			1x v1.1	Untested
Belkin	F5D7130		Broadcom 4710 @ 125MHz			Broadcom mini-PCI	None					Untested
Belkin	F5D7230-4	pre 1444	Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI						Untested
Belkin	F5D7230-4	from 1444	Broadcom 4712 @ 200 MHz	2MB	8MB	integrated Broadcom	BCM5325		Yes	No		No

Belkin	F5D7231-4	1102	Broadcom 4712 @ 200 MHz	2 MB	8 MB	integrated Broadcom	BCM5325					Untested
Belkin	F5D7231-4P		Broadcom 4712 @ 200 MHz	2 MB	16 MB	integrated Broadcom	ADM6996L				1x v1.1	Untested
Belkin	F5D7330		Broadcom 4710 @ 125MHz	2 MB	8 MB	Broadcom mini-PCI	None					Untested
Belkin	F5D8230-4		Broadcom 4704 @ 300MHz	4MB	16MB	Airgo mini-PCI	BCM5325	on	Yes	No	No	Untested
Buffalo	WBR-B11		Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI	BCM5325	on			No	Supported
Buffalo	WBR2-B11			4MB								Untested
Buffalo	WBR-G54		Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI	BCM5325	on			No	Supported
Buffalo	WBR2-G54		Broadcom 4712 @ 200MHz	4MB	16MB	integrated Broadcom	ADM6996L	on	Yes	Yes	No	Supported
Buffalo	WBR2-G54S		Broadcom 4712 @ 200MHz	4MB	16MB	integrated Broadcom	ADM6996L	on	Yes	Yes	No	Supported
Buffalo	WHR-G54S		Broadcom 5352 @ 200MHz			integrated Broadcom	integrated into CPU		Yes	Yes	No	Untested
Buffalo	WHR-HP-G54		Broadcom 5352 @ 200MHz			integrated Broadcom	integrated into CPU		Yes	Yes	No	Untested
Buffalo	WHR2-G54			4MB								Untested
Buffalo	WHR3-G54			4MB								Untested
Buffalo	WHR3-AG54		Broadcom 4704 @ 300MHz	4MB	64MB	Broadcom mini-PCI						Untested
Buffalo	WLA-G54		Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI	BCM5325	on				Supported
Buffalo	WLA-G54C		Broadcom 4710 @ 125MHz	4MB			None					Untested
Buffalo	WLA2-G54		Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI	None	off				Untested
Buffalo	WLA2-G54C		Broadcom 4712 @ 200MHz	4Mb	16Mb	integrated Broadcom	None		Yes	Yes		Untested
Buffalo	WLA2-G54L		Broadcom 4712 @ 200MHz	4MB	16MB	integrated Broadcom	ADM6996L	on	Yes	Yes		Supported
Buffalo	WLI-TX1-G54		Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI	None					Untested
Buffalo	WLI2-TX1-G54		Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI	None					Untested
Buffalo	WLI2-TX1-AG54		Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI	None					Untested

Buffalo	WZR-G108		Broadcom 4704 @ 300MHz	8Mb		Airgo mini-PCI							Untested
Buffalo	WZR-HP-G54		Broadcom 4704 @ 300MHz	4MB		Broadcom mini-PCI	BCM5325						Untested
Buffalo	WZR-RS-G54		Broadcom 4704 @ 300MHz	8MB	64MB	Broadcom mini-PCI	BCM5325	on					WiP
Dell	Truemobile 2300		Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI		off					Supported
D-Link	DSL-G604T / DSL-G664T		Texas Instruments Sangam/AR7 @ 150MHz	4MB	16MB	ACX111	IP175A	ADAM2	Yes		No		WiP
Linksys	ADSL2MUE		Texas Instruments Sangam/ AR7@150mhz	4MB	16MB	None	None	PSPBoot	Yes		v1.1		WiP
Linksys	WRT54AG		Broadcom 4710 @ 125MHz	4MB	16MB	Prism mini-PCI							Partial
Linksys	WAG54G	2	Texas Instruments Sangam/AR7 @ 150MHz	4MB	16MB	TI ACX111		ADAM2	Yes				WiP
Linksys	WAP54G	1.0	Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI	None	off					WiP
Linksys	WAP54G	1.1	Broadcom 4710 @ 125MHz	4MB	16MB	integrated Broadcom	None	off					WiP
Linksys	WAP54G	2.0	Broadcom 4712 @ 200MHz	2MB	16MB	integrated Broadcom	None	off	Yes	Yes	No		Untested
Linksys	WAP54G	3.0	Broadcom 5352 @ 200MHz			integrated Broadcom	None		Yes	Yes	No		Untested
Linksys	WAP55AG	1.0	Broadcom 4710 @ 125MHz	4MB	16MB	Atheros & Broadcom mini-PCI	None	off					Untested
Linksys	WAP55AG	2.0	Atheros 5312 @ 230MHz			integrated Atheros	None	doesn't exist	Yes	Yes	No		WiP
Linksys	WRE54G	1	Broadcom 4712 @ 200MHz	2MB	8MB	integrated Broadcom	None	off	Yes	No	No		Untested
Linksys	WRT54G	1.0	Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI	ADM6996L	off	No UART				Supported
Linksys	WRT54G	1.1	Broadcom 4710 @ 125MHz	4MB	16MB	integrated Broadcom	ADM6996L	off	No UART	Yes			Supported
Linksys	WRT54G	2.0	Broadcom 4712 @ 200MHz	4MB	16MB	integrated Broadcom	ADM6996L	off	Yes	Yes	No		Supported
Linksys	WRT54G	2.0 rev. XH	Broadcom 4712 @ 200MHz	4MB	16 or 32MB	integrated Broadcom	ADM6996L	off	Yes	Yes	No		Supported

Linksys	WRT54G	2.2	Broadcom 4712 @ 200MHz	4MB	16MB	integrated Broadcom	BCM5325	off	Yes	Yes	No	Supported
Linksys	WRT54G	3.0	Broadcom 4712 @ 200MHz	4MB	16MB	integrated Broadcom	BCM5325	off	Yes	Yes	No	Supported
Linksys	WRT54G	3.1 (AU?)	Broadcom 4712 @ 216MHz	4MB	16MB	integrated Broadcom	BCM5325	off	Yes	Yes	No	Supported
Linksys	WRT54G	4.0	Broadcom 5352 @ 200MHz	4MB	16MB	integrated Broadcom	integrated into CPU	off	Yes	Yes	No	Supported
Linksys	WRT54GC	1.0	Marvell									No
Linksys	WRT54GP2-AT	1.0	Marvell									No
Linksys	WRT54GS	1.0	Broadcom 4712 @ 200MHz	8MB	32MB	integrated Broadcom	ADM6996L	off	Yes	Yes	No	Supported
Linksys	WRT54GS	1.1	Broadcom 4712 @ 200MHz	8MB	32MB	integrated Broadcom	BCM5325	off	Yes	Yes	No	Supported
Linksys	WRT54GS	2.0	Broadcom 4712 @ 200MHz	8MB	32MB	integrated Broadcom	BCM5325	off	Yes	Yes	No	Supported
Linksys	WRT54GS	2.1	Broadcom 4712 @ 200MHz	8MB	32MB	integrated Broadcom	BCM5325	off	Yes	Yes	No	Supported
Linksys	WRT54GS	3.0	Broadcom 5352 @ 200MHz	8MB	32MB	integrated Broadcom	integrated into CPU	off	Yes	Yes	No	Supported
Linksys	WRT54GS	4.0	Broadcom 5352 @ 200MHz	4MB	16MB	integrated Broadcom	integrated into CPU	off	Yes	Yes	No	Supported
Linksys	WRT54GX	1.0	Broadcom 4704 @ 300MHz	4MB	16MB	Airgo mini-PCI	BCM5325	on	Yes	No	No	Partial
Linksys	WRT54GX	2.0	Realtek RTL8651B AGC @ 200MHz	8MB	32MB	Airgo mini-PCI	integrated Realtek	doesn't exist			No	No
Linksys	WRT55AG	1.0	Broadcom 4710 @ 125MHz	4MB	16MB	Atheros & Broadcom mini-PCI	BCM5325	off				Untested
Linksys	WRT55AG	2.0	Atheros 5312 @ 230MHz	4MB	16MB	integrated Atheros	KS8995M	doesn't exist	Yes	Yes	No	WiP
Linksys	WRTP54G		Texas Instruments Sangam/ AR7@150mhz	4MB	16MB	TI ACX111	ADM6996L	PSPBoot	Yes	Yes		WiP
Maxtor	Shared Storage		Broadcom 4780 @ 300Mhz	2MB	32MB	None	None		Yes	No	2x v2.0	Untested
Microsoft	MN-700		Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI	BCM5325	doesn't exist	No	Yes	No	Supported
Motorola	WA840G	1	Broadcom 4710 @ 125Mhz	4MB	16MB	Broadcom mini-PCI	None					Untested
Motorola	WA840G	2	Broadcom 4712 @ 200Mhz	2MB	8MB	integrated Broadcom	None		Yes	No	No	Untested
Motorola	WA840GP		Broadcom 4712 @ 200MHz	2MB	8MB	integrated Broadcom	None		Yes	No	No	Untested

Motorola	WE800G	1	Broadcom 4710 @ 125Mhz	4MB	16MB	Broadcom mini-PCI	None					Untested
Motorola	WE800G	2	Broadcom 4712 @ 200Mhz	2MB	8MB	integrated Broadcom	None		Yes	No	No	Untested
Motorola	WR850G	1	Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI	BCM5325					Supported
Motorola	WR850G	2	Broadcom 4712 @ 200MHz	4MB	16 or 32MB	integrated Broadcom	ADM6996L		Yes	Yes	No	Supported
Motorola	WR850G	3	Broadcom 4712 @ 200MHz	4MB	16MB	integrated Broadcom	ADM6996L		Yes	Yes	No	Supported
Motorola	WR850GP	3	Broadcom 4712 @ 200MHz	4MB	16MB	integrated Broadcom	ADM6996L		Yes	Yes	No	Supported
Netgear	DG834G	2	Texas Instruments Sangam/AR7 @150MHZ	4MB	16MB	ACX111 mini-PCI	Marvell 88E6060		Yes	No	No	WiP
Netgear	FWAG114		Broadcom 4710 @ 125MHz	2MB		Atheros & Broadcom mini-PCI	BCM5325					Untested
Netgear	WG602	3	Broadcom 4712 @ 200MHz	2MB	8MB	integrated Broadcom	None	on	Yes	Yes	No	No
Netgear	WGR614	3	Atheros 2312 @ 180MHz	4MB	16MB	integrated Atheros		doesn't exist			No	WiP
Netgear	WGR614	5	Broadcom 5350 @ 200MHz	1MB	8MB	integrated Broadcom	integrated into CPU	on			No	No
Netgear	WGR614	6	Broadcom 5350 @ 200MHz	1MB	8MB	integrated Broadcom	integrated into CPU	Unknown			No	No
Netgear	WGT624	1	Atheros 2312 @ 180MHz	4MB	16MB	integrated Atheros	Marvell	doesn't exist	Yes	Yes	No	WiP
Netgear	WGT634U		Broadcom 5365P @ 200MHz	8MB	32MB	Atheros mini-PCI	integrated into CPU	doesn't exist	Yes	No	1x v2.0	WiP
Ravotek	W54-AP		Broadcom 4710 @ 125MHz	4MB	16MB		None					Untested
Ravotek	W54-RT		Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI		on				Supported (no leds)
Ravotek	RT210w		Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI	BCM5325	on	No	No	No	Supported
Siemens	SE505	1	Broadcom 4710 @ 125MHz	4MB	16MB	Broadcom mini-PCI		on				Supported
Siemens	SE505	2	Broadcom 4712 @ 200MHz	4MB	8MB	integrated Broadcom	ADM6996L	on	Yes	Yes	1x v1.1 (easy mod)	Supported
Siemens	SX550			4MB								Untested
Simpletech	Simpleshare Office Storage Server		Broadcom 4780 @ 300Mhz		32MB	None	None		Yes	Yes	2x v2.0	Untested
Sitecom	WL-111											Untested

TCOM	Sinus 154 DSL Basic SE	Texas Instruments Sangam/AR7 @150MHZ	2MB	16MB	ACX111 mini-PCI	None		Yes	No	WiP
TCOM	Sinus 154 DSL Basic 3	Texas Instruments Sangam/AR7 @150MHZ	2MB	16MB	ACX111 mini-PCI	None		Yes	No	WiP
Toshiba	WRC-1000	Broadcom 4710 @ 125MHz	4MB	16MB	Prism mini-PCI					Partial, needs hostap
Trendnet	TEW-410APB		2MB							Untested
Trendnet	TEW-410APBplus		2MB							Untested
Trendnet	TEW-411BRP		4MB							Untested
Trendnet	TEW-411BRPplus		4MB							Untested
US Robotics	USR5430		2MB				on			Supported
US Robotics	USR5461	Broadcom 5350 @ 200MHz	2MB	8MB	integrated Broadcom	integrated into CPU	on		1x v2.0	Untested

Almost all of these pages are editable, create an account and click the edit (🗨) button at the top of the page.

OpenWrt OpenWrtDocs/About

Redirected from page "OpenWrt"

Clear message

OpenWrtDocs

1. About OpenWrt
2. Why should I run OpenWrt?
3. OpenWrt Version History

1. About OpenWrt

With the release of the Linux sources for the Linksys WRT54G/GS series of routers came a number of modified firmwares to extend functionality in various ways. Each firmware was 99% stock sources and 1% added functionality, and each firmware attempted to cater to a certain market segment with the functionality that they provided. The downsides were twofold, one - it was often difficult to find a firmware with the combination of functionality desired (leading to forks and yet more custom firmwares) and two - all the firmwares were based on the original Linksys sources which were far behind mainstream Linux development.

OpenWrt takes a different route, instead of starting out with the Linksys sources, the development started with a clean slate. Piece by piece software was added to bring the functionality back to that of the stock firmware, using the most recent versions available. What makes OpenWrt really unique though is the fact it employs a writable filesystem so the firmware is no longer a static compilation of software but can instead be dynamically adjusted to fit the particular needs of the situation. In

short, the device is turned into a mini linux PC with OpenWrt acting as the distribution, complete with almost all traditional linux commands and a package management system for easily loading on extra software and features.

2. Why should I run OpenWrt?

Because Linux gives us the power to do what we need with cheap hardware while avoiding proprietary, rigid software. OpenWrt isn't for the faint of heart, but if you honestly need a barebones Linux and are prepared to do some work OpenWrt is the fastest Linux based firmware for the Linksys WRT's. At the moment the distribution contains more than 100 software packages. Furthermore the OpenWrt community provide more add-on packages. For developers the project provides a build system, which may be used to create modified firmware from source. Porting of new software packages is simplified with the use of the SDK.

If you aren't up to the task or simply don't need a real, full blooded Linux, here are some alternatives:

Seattle Wireless's Customized Firmware Listing

3. OpenWrt Version History

The project started in January 2004. The first OpenWrt versions were based on Linksys original GPL sources for WRT54g and a buildroot from the uclibc project. This version is known as OpenWrt "stable" and was widely in use. There are still many OpenWrt applications, like the Freifunk-Firmware or Sip@Home, which are based on this version.

In the beginning of 2005 some new developers has joined the small developer team. After some month of closed development the team decided to publish the first "experimental" versions of OpenWrt. The experimental version use a heavily customized buildsystem based on buildroot2 from the uclibc project. OpenWrt uses official linux kernel sources (2.4.30) and only add patches for the system on chip and drivers for the network interfaces. The developer team try to reimplement most of the proprietary code inside the GPL tarballs of the different vendors. There are free tools for writing new firmware images directly into the flash (mtd), for configuring the wireless lan chip (wlcompat/wificonf) and to program the VLAN capable switch (robocfg). The codename of the first OpenWrt release is "White Russian" a popular cocktail. OpenWrt 1.0 is

planned for the end of 2005.

Almost all of these pages are editable, create an account and click the edit () button at the top of the page.

DeleteCache (cached 2005-10-17 17:23:53)

Immutable page (last edited 2005-10-09 14:53:26 by JayStrauss)

Or try one of these actions: Like Pages, Local Site Map, Spell Check

OpenWrtDocs

1. Will OpenWrt work on my hardware ?

See TableOfHardware

2. Obtaining the firmware

Stable Release

At the moment we have no stable supported release. You can get release candidates for the next stable OpenWrt release in binary format: http://downloads.openwrt.org/whiterussian/

Stable Source

The stable source code can be found in the above directory or from our CVS repository. This is not recommended for beginners; we will not troubleshoot failed compiles.

```
cvs -d:pserver:anonymous@openwrt.org:/openwrt -z3 co -r whiterussian openwrt
```

Development

Development take place in CVS. You get the source via:

```
cvs -d:pserver:anonymous@openwrt.org:/openwrt -z3 co openwrt
```

If you find any bugs, please use our forum or irc channel to report.

You may find some binary snapshots in the directories of our developers http://downloads.openwrt.org/people/

3. Installing OpenWrt

LOADING AN UNOFFICIAL FIRMWARE WILL VOID YOUR WARRANTY

OpenWrt is an unofficial firmware which is neither endorsed or supported by the vendor of your router. OpenWrt is provided "AS IS" and without any warranty under the terms of the GPL. You can always flash back your original firmware, so please be sure you have it downloaded and saved locally.

To avoid potentially serious damage to your router caused by an unbootable firmware you should read the documentation for your specific router model.

We strongly suggest you also read OpenWrtDocs/Troubleshooting before installing

The jffs2 versions of may take several minutes for the first bootup and will require a reboot before being usable

3.1. General instructions (router specific instructions later)

Although you can install the firmware through more traditional means (via webpage), we recommend that you use TFTP for your first install. This is **NOT** a requirement, simply a damned good idea; if anything goes wrong you can just TFTP the old firmware back.

When the device boots it runs a bootloader. It's the responsibility of this bootloader to perform basic system initialization along with validating and loading the actual firmware; think of it as the BIOS of the device before control is passed over to the operating system. Should the firmware fail to pass a CRC check, the bootloader will presume the firmware is corrupt and wait for a new firmware to be uploaded over the network. The type of the preinstalled bootloader depends on your model. Broadcom based routers use CFE - Common Firmware Environment (older Boards use PMON), Texas Instruments based routers use ADAM2.

The basic procedure of using a tftp client to upload a new firmware to your router:

- unplug the power to your router
- start your tftp client
  - give it the router's address (usually 192.168.1.1)
  - set mode to octet
  - tell the client to resend the file, until it succeeds.
  - put the file
- plug your router, while having the tftp client running and constantly probing for a connection
- the tftp client will receive an ack from the bootloader and starts sending the firmware

Please be patient, the reflashing occurs AFTER the firmware has been transferred. DO NOT unplug the router, it will automatically reboot into the new firmware.

On routers with a DMZ led, OpenWrt will light the DMZ led while booting, after bootup it will turn the DMZ led off. Sometimes automatic rebooting does not work, so you can safely reboot after 5 minutes.

The tftp commands vary across different implementations. Here are two examples, netkit's tftp client and Advanced TFTP (available from:

ftp://ftp.mamalinux.com/pub/atftp/)

netkit's tftp commands:

```
tftp 192.168.1.1
tftp> binary
tftp> rexmt 1
tftp> timeout 60
tftp> trace
Packet tracing on.
tftp> put openwrt-xxx-x.x-xxx.bin
```

Setting "rexmt 1" will cause the tftp client to constantly retry to send the file to the given address. As advised above, plug in your box after typing the commands, and as soon as the bootloader starts to listen, your client will successfully connect and send the firmware. You can try to run "ping -f 192.168.1.1" (as root) in a separate window and enter the line "put openwrt-xxx-x.x-xxx.bin" as the colons stop running over your terminal when you power-recycle your router.

Advanced TFTP commands:

```
atftp
tftp> connect 192.168.1.1
tftp> mode octet
tftp> trace
tftp> timeout 1
tftp> put openwrt-xxx-x.x-xxx.bin

Or use the command-line:
atftp --trace --option "timeout 1" --option "mode octet" --put --local-file openwrt-xxx-x.x-xxx.
bin 192.168.1.1
```

MacTFTP Client commands: (For use with OS X. tftp in terminal did not work for me)

```
Download MacTFTP Client [http://www.mactechnologies.com/pages/downld.html]
 * Choose Send
 * Address: 192.168.1.1
 * Choose the openwrt-xxx-x.x-xxx.bin file
 * Click on start while applying power to the WRT54G
```

Please note, netkit tftp has failed to work for some people. Try to use Advanced TFTP. Don't forget about your firewall settings, if you use one. It is best to run the "put" command and then immediately apply power to the router, since the upload window is extremely short and very early in boot.

TFTP Error	**Reason**
Code pattern is incorrect	The firmware image you're uploading was intended for a different model.
Invalid Password	The firmware has booted and you're connected to a password protected tftp server contained in the firmware, not the bootloader's tftp server.

Timeout	Ping to verify the router is online Try a different tftp client (some are known not to work properly)

Some machines will disable the ethernet when the router is powered off and not enable it until after the router has been powered on for a few seconds. If you're consistantly getting "Invalid Password" failures try connecting your computer and the router to a hub or switch. Doing so will keep the link up and prevent the computer from disabling its interface while the router is off.

Windows 2000 and Windows XP have a TFTP client, and it can be used to flash with OpenWrt firmware.

Windows 2000/XP TFTP Client short Instructions

- Open two command windows (Start-Run-Enter "cmd")
- In one window, type "ping -t 192.168.1.1" and press enter. 192.168.1.1 is the router IP.
- Ping will continuously try to contact the wrt. Keep this running
- In the other window, prepare the tftp command "tftp -i 192.168.1.1 PUT OpenWrt-gs-code.bin". Do not press enter yet!
- Now you may plug in the router (unplug it first if it was plugged).
- In the ping window it will start saying "Hardware Error"
- Return to the tftp window. As soon as the ping window starts to answer again, press enter in the tftp window.
- The image should now be flashed without multiple tries.
- If ping starts with "Hardware Error", then starts to answer, and then returns to "Hardware Error" again for a short moment, you waited too long.

3.2. Linksys WRT54G and WRT54GS

To use the TFTP method above you need to enable boot_wait. Plug your ethernet cable into one of the LAN ports. Once enabled, the router will wait for ~3 seconds for a firmware before booting. While in boot_wait the router is **always 192.168.1.1, regardless of configuration** -- you'll have to force your computer to use 192.168.1.x (netmask 255.255.255.0) address for the purpose of reflashing.

Do not use the Linksys TFTP program. IT WILL NOT WORK.

Model	Firmware (JFFS2)	Firmware (SQUASHFS)
WRT54G	openwrt-wrt54g-jffs2.bin	openwrt-wrt54g-squashfs.bin
WRT54GS	openwrt-wrt54gs-jffs2.bin	openwrt-wrt54gs-squashfs.bin

Squashfs files:

- The firmwares with "squashfs" in the filename use a combination of a readonly squashfs partition and a writable jffs2 partition. This gives you a /rom with all the files that shipped with the firmware and a writable root containing symlinks to /rom. This is considered the standard install.

Jffs2 files:

- The firmwares with "jffs2" in the name are jffs2 only; all of the files are fully writable. The "4M" and "8M" in the filenames is a reference to the flash block size; most 4M flash chips use a block size of 64k while most 8M chips tend to use a 128k block size -- there are some exceptions. The jffs2 partition needs to be formatted for the correct block size and hence the two versions.

  The jffs2 versions are for experienced users only -- these firmwares only have minimal support for failsafe mode.

For more information, see the README file that comes with the release.

3.2.1. Enabling boot_wait

If the boot_wait variable is set, the bootup process is delayed by few seconds allowing a new firmware to be installed through the bootloader using tftp. Setting of the boot_wait variable is done through a bug in the Ping.asp administration page by pinging the certain "addresses" listed below. **You find ping.asp by navigating through the administration page and selecting diagnostics.**.

First, for this to work the **internet port must have a valid ip address**, either from dhcp or manually configured from the main page - the port itself doesn't need to be connected unless using dhcp. Next, navigate to the Ping.asp page and enter exactly each line listed below, one line at a time into the "IP Address" field, pressing the Ping button after each entry.

⚠ **The last versions of the firmware to support the Ping.asp bug described below are 3.01.3 for the WRT54G (up to/including v3.0) and 3.37.2 for the WRT54GS (up to/including v2.0). Downgrading to these firmwares is required to enable boot_wait.**

⚠ **I have a WRT54GS 1.1 with a firmware version 4.50 - the Ping.asp trick worked for me without downgrading the firmware! This seems to work as well with the WRT54G V3.1 with firmware V4.01.2. If you are lazy, like me, it may worth a try.**

⚠ **Ping bug still exists in firmware 4.20.6 on hardware 4.0 and firmware 4.20.7 on hardware 2.0, but it is necessary to use the ping_times variation on the hack, plus some stripping of javascript. Details here.**

```
;cp${IFS}*/*/nvram${IFS}/tmp/n
;*/n${IFS}set${IFS}boot_wait=on
;*/n${IFS}commit
;*/n${IFS}show>tmp/ping.log
```

When you get to the last command the ping window should be filled with a long list of variables including **boot_wait=on** somewhere in that list.

This ping exploit definitely works with ALL WRT54G/GS VERSIONS. You must have an address on the WAN port. In the Setup/Basic Setup/ Internet Setup section you may wish to select Static IP and set IP=10.0.0.1, Mask=255.0.0.0, Gateway=10.0.0.2. Those values are meaningless; you'll be overwriting them soon with new firmware. Note: flashing a Linksys WRT54GS v1.1 by using TFTP is only possible using the Port 1 of the switch!

You can also use the takeover script to make ping hack in a single command (need a shell command line interpreter). This script expects to find the to-be flashed firmware in a file called **openwrt-g-code.bin**, which is in the *current* directory.

There is another bug still present in Ping.asp (firmware revision 3.03.1) where you can put your shell code into the ping_times variable. See http://www.linksysinfo.org/modules.php?name=Forums&file=viewtopic&t=448 This means you don't have to downgrade your firmware first and it removes the input size restrictions so you can use more obvious shell commands like:

```
`/usr/sbin/nvram set boot_wait=on`
`/usr/sbin/nvram commit`
`/usr/sbin/nvram show > /tmp/ping.log`
```

3.2.2. Setting boot_wait from a serial connection

With a serial connection to your WRT, you don't have to use the ping bug or change your Linksys firmware. You can set boot_wait from the console, using the commands

```
#nvram set boot_wait=on
#nvram get boot_wait           (just to confirm, should respond with "on")
#nvram commit                  (takes a few seconds to complete)
```

You can also set boot_wait from the CFE boot loader (to enter CFE, reboot the router with "# reboot" while hitting "Ctrl C" continously)

```
CFE> nvram set boot_wait=on
CFE> nvram get boot_wait           (just to confirm, should respond with "on")
CFE> nvram commit                  (takes a few seconds to complete)
```

3.3. ASUS WL-500G and WL-300G

Pull the plug, press and hold the reset button, plug the device and wait until the PWR LED starts flashing slowly (almost immediately). Now release the reset button and upload the firmware by TFTP using the following commands:

TFTP commands:

```
tftp 192.168.1.1
tftp> binary
tftp> trace
tftp> get ASUSSPACELINK\x01\x01\xa8\xc0 /dev/null
tftp> put openwrt-xxx-x.x-xxx.trx ASUSSPACELINK
```

After this, wait until the PWR LED stops flashing and the device to reboot and you should be set. There's also nice shell script doing this work for you to be at http://openwrt.org/downloads/utils/flash.sh.

As an alternative (or if this installation routine doesn't do the trick for you) you can always use the ASUS Recovery tool from your utilities CD to upload your openwrt firmware.

Another thing is that the ASUS WL-500G/WL-300G doesn't seem to revert to the 192.168.1.1 address when starting the bootloader, but seems to use the LAN IP address set in NVRAM, so try this address or use the recovery tool if you've got problems flashing your firmware.

There are several helpful tutorials especially for the ASUS routers at http://www.macsat.com.

3.4. ASUS WL-500G Deluxe

This device is based on the Broadcom chipset so the openwrt-brcm-x image is required. Pull the plug, press and hold the reset button, plug the device and wait until the PWR LED starts flashing slowly (almost immediately). Now release the reset button and upload the firmware by TFTP using the following commands:

TFTP commands:

```
tftp 192.168.1.1
tftp> binary
tftp> trace
tftp> put openwrt-xxx-x.x-xxx.trx
```

After this, wait until the PWR LED stops flashing and the device to reboot and you should be set. There's also nice shell script doing this work for you to be at http://openwrt.org/downloads/utils/flash.sh. This script is also included in the source under scripts/flash.sh.

As an alternative (or if this installation routine doesn't do the trick for you) you can always use the ASUS Recovery tool from your utilities CD to upload your openwrt firmware.

Another thing is that the ASUS WL-500GD doesn't revert to the 192.168.1.1 address when starting the bootloader, but use the LAN IP address set in NVRAM, so try this address or use the recovery tool if you've got problems flashing your firmware.

3.5. Siemens Gigaset SE505

The installation procedure is essentially the same as the generic one described above. The only differences are that the bootloader listens based on nvram lan_ipaddr= variable (default: 192.168.2.1) and the IP of the machine sending the new firmware has to be 192.168.x.100 or the router will only accept the first packet. boot_wait is enabled by default on these devices.

You can erase nvram settings by pressing reset button while powering on the router.

Starting with WHITE RUSSIAN (RC2) the bug is fixed. So from this release of OpenWrt everything works just fine.

3.6. Motorola WR850G

Flashing the Motorola WR850G is fairly easy. Just follow these easy steps!

1. Use the web interface to set the router's IP address to 192.168.1.1. This will mitigate the issue where dnsmasq doesn't properly read the subnet from the configuration.
2. Download the motorola firmware image (either the squashfs or the jffs2-8mb version) from the website. (Note: The motorola has 4mb flash, but requires the 8mb version. This is due to the paging size of the flash rom that is used, and is not related to the ignominously confusing names used for the files. At the moment the motorola-jffs2-4mb is entirely useless [64k page size, 8mb is 128k page size].)
3. Change the extension of the firmware image to .trx, because the Motorola web interface will not accept files with different extensions.
4. Use the Control Panel -> Firmware page of the Motorola web interface to upload OpenWRT. The power light on the WR850G will flash between red and green. DO NOT INTERRUPT THE POWER TO THE WR850G WHILE THIS IS HAPPENING. Doing so has been shown by the state of California to cause birth defects such as low birth weight, miscarriage, and the Black Lung.
5. You will receive a message in your browser telling you the flash is complete and that you should restart the router. Do so, either using the web interface or power cycling the router.
6. When you're finished, telnet to 192.168.1.1, issue the 'reboot' command if you're using jffs2, and change your password to activate dropbear.
7. If you're having trouble getting an IP, try setting your IP manually to 192.168.1.2. Sometimes dnsmasq doesn't work properly with the WR850G routers. An nvram reset ((('mtd erase nvram; reboot'))) may solve this issue (Note: erasing nvram resets the router's IP to 192.168.10.1) ⚠ **NOTE:** It has been reported that v2 of the WR850G will **NOT** reset the nvram after erasing it, leaving the unit bricked. So proceed with caution!

⚠ **If you're using TFTP to flash the firmware, put to the host 192.168.10.1.**

3.7. Buffalo Airstation WLA-G54

This device is based on the Broadcom chipset so the openwrt-brcm-x image is required. The web interface will not allow you to install the openwrt firmware so you will need to use tftp. Pull the power plug, press and hold the reset button, plug the device and wait until the PWR LED starts flashing slowly (almost immediately). Now release the reset button and upload the firmware. This unit keeps the IP address that it was set to whilst in this mode. Factory setting is 192.168.11.2.

TFTP commands:

```
tftp 192.168.11.2
tftp> binary
tftp> trace
tftp> rexmt 1
tftp> timeout 60
tftp> put openwrt-xxx-x.x-xxx.trx
```

After this, wait until the PWR LED stops flashing and the device to reboot and you should be set. You should be able to telnet to 192.168.11.2 or whatever the unit was set to prior to the installation.

3.8. Buffalo AirStation WBR2-G54S

Here too you need an openwrt-brcm-*.trx image. The device has boot_wait=on by default, so you can just begin sending the file from your TFTP client, power up the device, and let it install. The TFTP loader uses the IP address to which you've configured the device; 192.168.11.1 by default. If you ping the device, the TFTP loader will respond with TTL=100, but both the Buffalo firmware and OpenWRT will respond with TTL=64.

The firmware provided by Buffalo has some extra headers at the beginning. If you load it via TFTP, you must first remove the extras so that the file begins with "HDR0". Otherwise, it won't boot (but you can still replace it via TFTP).

With the Buffalo firmware (at least version 2.30), if you save the settings to a file, it will obfuscate the output by inverting each bit. To undo this and see the NV-RAM settings, filter the file through: perl -pe 's/(.)/chr(ord($1)^0xFF)/seg; tr/\0/\n/'

4. Using OpenWrt

Please see OpenWrtDocs/Using

5. Troubleshooting

If you have any trouble flashing to OpenWrt please refer to OpenWrtDocs/Troubleshooting

Almost all of these pages are editable, create an account and click the edit (🗨) button at the top of the page.

OpenWrt OpenWrtDocs/Using

OpenWrtDocs

1. Using OpenWrt for the first time

OpenWrt uses the DMZ led to signal bootup, turning the led on while booting and off once completely booted. Once booted, you should be able to telnet into the router using the last address it was configured for.

```
Trying 192.168.1.1...
Connected to 192.168.1.1.
Escape character is '^]'.

BusyBox v1.00 (2004.12.24-03:19+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.


  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
```

```
         |__| W I R E L E S S   F R E E D O M


root@OpenWrt:/#
```

The firmware itself is designed to occupy as little space as possible while still providing a reasonably friendly commandline interface. With no packages installed, the firmware will simply configure the network interfaces, setup a basic NAT/firewall and load the telnet server and dnsmasq (a combination dns forwarder and dhcp server).

Why no telnet password? Telnet is an insecure protocol with no encryption, we try to make a point of this insecurity by not enabling a password. If you're in an environment that requires password protection we suggest setting a password with the *passwd* command, which will disable the telnet server and enable the Dropbear SSH server.

What if I can't get in? The problem is caused when the jffs2 partition (see below) is detected but unusable, either the result of previous OpenWrt installation or occasionally just caused by a brand new router. Simply boot into failsafe mode and run firstboot to reformat the jffs2 partition.

What if I can not access telnet when first booting? This may very well be a problem with your firewall settings in linux or windows. If you have any firewalls, disable them. *WARNING* Do this only if you know what you are doing and can restore your iptables rules effortlessly. In linux, you can flush iptables firewall settings by issuing the following series of commands:

```
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -F
```

2. Firstboot / jffs2

The following applies only to the squashfs images of OpenWrt. The jffs2 images just need an extra reboot, so you don't have to run *firstboot* yourself.

The OpenWrt firmware contains two pieces, a kernel and a readonly filesystem embeded in the

firmare known as squashfs. The job of the firstboot script is to make a secondary, writable jffs2 filesystem out of the free space in flash.

When OpenWrt boots it will check for the existance of a jffs2 partition and attempt to boot from that, otherwise it will boot from the squashfs filesystem. The lack of a jffs2 partition will *automatically* trigger the firstboot script which will run in the background, creating a jffs2 filesystem and populating it with symbolic links to the squashfs filesystem (which is now remounted to the /rom directory). The effect of this is that the root filesystem will suddenly appear to be writable shortly after bootup, this can be verified with the mount command:

```
/dev/mtdblock/4 on / type jffs2 (rw)
```

If you do not see this line you may need to run firstboot manually (just type "firstboot"). The firstboot script can also be used to erase all changes to the jffs2 partition, particularly useful when upgrading from previous OpenWrt versions with different filesystem layouts.

3. Editing Files

On jffs2 firmware images, the whole root filesystem is writable, however on the squashfs builds you need some extra steps to edit the default configuration files:

The use of symlinks by the firstboot script saves space on the jffs2 partition, but it does have some interesting side effects such as edititing files. Since all files are by default symlinks to a readonly filesystem, you will not be able to edit files directly -- you'll get a readonly error if you try. Instead you have to delete the symlink and copy the file to the jffs2 partition to be able to edit it.

```
rm /etc/ipkg.conf
cp /rom/etc/ipkg.conf /etc/ipkg.conf
vim /etc/ipkg.conf
```

4. ipkg

The ipkg utility is a lightweight package manager used to download and install OpenWrt packages from the internet. (Linux users familiar with apt-get will recognise the similarities)

`ipkg update`	Download a list of packages available
`ipkg list`	View the list of packages
`ipkg install dropbear`	Install the dropbear package
`ipkg remove dropbear`	Remove the dropbear package

More options can be found via *`ipkg --help`*.

Additional packages can be found through Nico's package tracker; these packages can be installed using *`ipkg install http://example.com/package.ipk`* or by adding the the source repository to your */etc/ipkg.conf*.

If you have USB storage, or install packages to a destination other than root, the shell script ipkg-link will create automatic symlinks to the root filesystem for those packages. See the info on ipkg-link on the UsbStorageHowto

5. Configuration

See OpenWrtDocs/Configuration

Almost all of these pages are editable, create an account and click the edit (🗨) button at the top of the page.

OpenWrt OpenWrtDocs/Configuration

1. NVRAM

NVRAM stands for Non-Volatile RAM, in this case the last 64K of the flash chip used to store various configuration information in a *name=value* format.

Command	Description
nvram show | sort | less	Display everything in nvram
nvram get boot_wait	Get a specific variable
nvram set boot_wait=on	Set a value
nvram set lan_ifnames="vlan0 vlan1 vlan2"	set multiple values to one param
nvram unset foo	Delete a variable
nvram commit	Write changes to the flash chip (otherwise only stored in RAM)

A complete list of nvram options can be found at OpenWrtNVRAM.

2. Network configuration

Quick overview of the router architecture:

The WRT54G is made up of an Ethernet switch, a wireless access point and a router chip that connects them together.

http://upload.wikimedia.org/wikipedia/commons/0/0f/WRT54G_internal_architecture.png

The names of the network interfaces will depend largely on what hardware OpenWrt is run on.

Manufacturer	Model	Hardware version	LAN	WAN	WIFI	Comments
Linksys	WRT54G	v1.x	vlan2	vlan1	eth2	
Linksys	WRT54G	v2.x/v3.0	vlan0	vlan1	eth1	
Linksys	WRT54GS	v1.x/v2.0/v3	vlan0	vlan1	eth1	
Asus	WL-300g		eth0	None	eth2	
Asus	WL-500g		eth0	eth1	eth2	
Asus	WL-500g Deluxe		vlan0	vlan1	eth1	eth0 is the whole switch, with lan and wan ports

Buffalo	WBR-G54		eth0	eth1	eth2	
Buffalo	WBR2-G54S		vlan0	vlan1	eth1	eth0 is the whole switch, with lan and wan ports
Buffalo	WLA-G54		eth0	N/A	eth2	No WAN port on this device
Motorola	WR850G	v3	vlan0	vlan1	eth1	eth0 is the whole switch, with lan and wan ports
Microsoft	MN700	v.x	eth0	eth1	eth2	
Siemens	SE505	v1	eth0	eth1	eth2	
Siemens	SE505	v2	vlan0	vlan1	eth1	eth0 is the whole switch, with lan and wan ports

Please update to include other models.

NOTE: LAN and WIFI are bridged together in br0 by default, on some devices WAN can be eth1 and LAN eth0.

In general, the switch works the following way: It recieves data (physical) on either lan port or on the wan port. Then the switch tags the packages (with VLAN), so Linux is able to see the difference and that way we have the vlan devices, which describe wan or lan port.

The basic (802.3) network configuration is handled by a series of NVRAM variables:

NVRAM	**Description**
<name>_ifname	The name of the linux interface the settings apply to
<name>_ifnames	Devices to be added to the bridge (only if the above is a bridge)
<name>_proto	The protocol which will be used to configure an IP
	static: Manual configuration (see below)
	dhcp: Perform a DHCP request
	pppoe: Create a ppp tunnel (requires pppoecd package)
<name>_ipaddr	ip address (x.x.x.x)
<name>_netmask	netmask (x.x.x.x)
<name>_gateway	Default Gateway (x.x.x.x)
<name>_dns	DNS server (x.x.x.x)

The command *ifup* <*name*> will configure the interface defined by <name>_ifname according to the above variables. As

an example, the /etc/init.d/S40network script will automatically run the following commands at boot:

```
ifup lan
ifup wan
ifup wifi
```

The *ifup lan* command will bring up the interface specified by lan_ifname. Normally the lan_ifname is set to br0 which will cause it to create the bridge br0 and add the the interfaces from lan_ifnames to the bridge; lan_proto is usually static which means that br0 will have the ip address from lan_ipaddr, and so on for the rest of the variables listed above.

It's important to remember that it's the <name>_ifname that specifies the interfaces, the <name> compontent itself has almost no value. This means that if you changed lan_ifname to be the internet port, vlan1, then *ifup lan* would bring up the internet port, not the lan ports (despite using the command *ifup lan* and using the lan_ variables). Also, it means that you can create any <name> variables you want, foo_ifname, foo_proto .... and they would be used by *ifup foo*.

The only <name> with any significance is **wan**, used by the /etc/init.d/S45firewall script. The firewall script will NAT traffic through the wan_ifname, blocking connections to wan_ifname.

Further information about the variables used can be found at OpenWrtNVRAM

Don't forget to check the OpenWrtFaq for information about howto setup PPPOE etc.

2.1. Sample network configurations

For client mode configuration (rather than AP mode), see this page: ClientModeHowto

(note these examples use wrt54g v2.x/wrt54gs v1.x interface names)

The default network configuration: (lan+wireless bridged as 192.168.1.1/24, wan as dhcp)

```
lan_ifname=br0
lan_ifnames="vlan0 eth1"
lan_proto=static
lan_ipaddr=192.168.1.1
lan_netmask=255.255.255.0

wan_ifname=vlan1
wan_proto=dhcp
```

If you just want to use OpenWrt as an access point you can avoid the WAN interface completely: (lan+wireless bridged as 192.168.1.25/24, routed through 192.168.1.1, wan ignored)

```
lan_ifname=br0
lan_ifnames="vlan0 eth1"
lan_proto=static
lan_ipaddr=192.168.1.25
lan_netmask=255.255.255.0
lan_gateway=192.168.1.1
lan_dns=192.168.1.1

wan_proto=none
```

You can also have the lan interface fetch its configuration via dhcp, but to do so, you'll have to comment out the line:

```
# linksys bug; remove when not using static configuration for lan
nvram set lan_proto="static"
```

in /etc/init.d/S05nvram (The usual story about replacing the symlink with a copy of the file before editting applies). After doing this, you need to set the appropriate nvram variable:

```
lan_proto=dhcp
```

To separate the LAN from the WIFI: (lan as 192.168.1.25/24, wireless as 192.168.2.25/24, wan as dhcp, remove your wifi interface (eth1 on v2/3 linksys routers) from the lan_ifnames variable)

```
lan_ifname=vlan0
lan_proto=static
lan_ipaddr=192.168.1.25
lan_netmask=255.255.255.0

wifi_ifname=eth1
wifi_proto=static
wifi_ipaddr=192.168.2.25
wifi_netmask=255.255.255.0

wan_ifname=vlan1
wan_proto=dhcp

lan_ifnames=vlan0 eth2 eth3
```

2.2. The ethernet switch

The WRT54G is essentially a WAP54G (wireless access point) with a 6 port switch. There's only one physical ethernet connection and that's wired internally into port 5 of the switch; the WAN is port 0 and the LAN is ports 1-4. The separation of the WAN and LAN interfaces is done by the switch itself. The switch has a vlan map which tells it which vlans can be accessed through which ports.

The vlan configuration is based on two variables (per vlan) in nvram.

```
vlan0ports="1 2 3 4 5*" (use ports 1-4 on the back, 5 is the wrt54g itself)
vlan0hwname=et0
```

2.2.1. Normal Behavior

This is only the case if the nvram variable boardflags is set. On the WRT54G V1.1 and earlier, it's not set.

When the et module (ethernet driver) loads it will read from vlan0ports to vlan15ports, behind the scenes the ethernet driver is using these variables to generate a more complex configuration which will be sent to the switch. When packets are recieved from external devices they need to be assigned a vlan id, and when packets are sent to those external devices the vlan tags need to be removed.

PVID represents the primary vlan id, in other words if a packet doesn't have a vlan tag, which vlan does it belong to? The ethernet driver handles this rather trivially, in the case of vlan0ports="1 2 3 4 5*", ports 1-4 are set to PVID 0 (vlan0). Since the wrt needs to recieve packets from both the LAN (vlan0) and the WAN (vlan1), port 5 is a special case appearing in both vlan0ports and vlan1ports. This is where the '*' is used -- it determines the PVID of port 5, which is also the only port not to untag packets (for hopefully obvious reasons).

The second variable, vlan0hwname is used by the network configuration program (or script in the case of openwrt) to determine the parent interface. This should be set to "et0" meaning the interface matching et0macaddr.

Sample configurations (unless otherwise specified, vlan variables not shown are assumed to be unset)

Default:

```
vlan0ports="1 2 3 4 5*"
vlan0hwname=et0
vlan1ports="0 5"
vlan1hwname=et0
```

All ports lan (vlan0):

```
vlan0ports="0 1 2 3 4 5*"
vlan0hwname=et0
```

LAN (vlan0), WAN (vlan1), DMZ (vlan2):

```
vlan0ports="1 2 5*"
vlan0hwname=et0
vlan1ports="0 5"
vlan1hwname=et0
vlan2ports="3 4 5"
vlan2hwname=et0
```

It's a good idea when choosing a vlan layout to keep port 1 in vlan0. At least the WRT54GS v1.0 will not accept new firmware via tftp if port 1 is in another vlan.

2.2.2. Using Robocfg

Robocfg is a utility written by Oleg Vdovikin to enable the hardware configuration of the Broadcom BCM5325E/536x VLAN enabled 6-port ethernet switch. When used properly, it can configure the switch in such a way that enables each of the five exposed ports of the switch to be treated as a separate, individual ethernet interface. Using robocfg, the switch can also be configured to tag packets for use in VLAN enabled networks, and to configure each port's MDI, duplex, and speed settings. Robocfg options can be issued individually, or strung together on one line, each new option and parameter separated by a space. See the bottom of this section for a copy of Robocfg's own stated parameters.

Sample Command Uses

Show current switch configuration:

```
robocfg show
```

Enable or disable a port(note: tx/rx_disabled can be useful for traffic monitoring):

```
robocfg port X state <enabled|disabled|rx_disabled|tx_disabled>
```

Set port speed and duplex:

```
robocfg port X media <auto|10HD|10FD|100HD|100FD>
```

Set port crossover state:

```
robocfg port X mdi-x <auto|on|off>
```

Advanced Configuration

When changing port assignments for VLANs, the switch should be disabled before changing the settings, and then re-enabled after the settings have been entered. Of course, the configuration should also be done using a serial console or executed as a script, since reconfiguration of the switch will disconnect any current telnet or ssh session. Port numbers followed by a "t" will pass tagged packets(necessary for port 5), while port numbers with a "u", or no "t", will untag packets when passing them through the interface. The following example(which configures each physical port with it's own VLAN) has been stretched out to better show each action:

```
robocfg switch disable
robocfg vlans enable reset
robocfg vlan 0 ports "0 5t"
robocfg vlan 1 ports "1 5t"
robocfg vlan 2 ports "2 5t"
robocfg vlan 3 ports "3 5t"
robocfg vlan 4 ports "4 5t"
robocfg switch enable
```

Now that the switch has been configured to tag the appropriate packets, the VLANs can be created using the vconfig command:

```
vconfig add eth0 0
vconfig add eth0 1
vconfig add eth0 2
vconfig add eth0 3
vconfig add eth0 4
```

Now VLANs 0-4 have been created, and these can be seen with the "ifconfig -a" command. Each VLAN now needs to be assigned a unique hardware MAC address:

```
ifconfig vlan0 hw ether XX:XX:XX:XX:XX:00
ifconfig vlan1 hw ether XX:XX:XX:XX:XX:01
ifconfig vlan2 hw ether XX:XX:XX:XX:XX:02
ifconfig vlan3 hw ether XX:XX:XX:XX:XX:03
ifconfig vlan4 hw ether XX:XX:XX:XX:XX:04
```

An IP address can be assigned to each VLAN interface now, if desired:

```
ifconfig vlanX xx.xx.xx.xx netmask xx.xx.xx.xx
```

Finally, each interface can be brought up:

```
ifconfig vlanX up
```

Alternately, all ports can be placed on vlan0:

```
robocfg switch disable
robocfg vlans enable reset
robocfg vlan 0 ports "0 1 2 3 4 5t"
robocfg switch enable
vconfig eth0 0
ifconfig vlan0 xx.xx.xx.xx netmask xx.xx.xx.xx
ifconfig vlan0 up
```

Original Robocfg Parameter List

```
Usage: robocfg <op> ... <op>
Operations are as below:
        show
        switch <enable|disable>
        port <port_number> [state <enabled|rx_disabled|tx_disabled|disabled>]
                [stp none|disable|block|listen|learn|forward] [tag <vlan_tag>]
                [media auto|10HD|10FD|100HD|100FD] [mdi-x auto|on|off]
        vlan <vlan_number> [ports <ports_list>]
        vlans <enable|disable|reset>

        ports_list should be one argument, space separated, quoted if needed,
        port number could be followed by 't' to leave packet vlan tagged (CPU
        port default) or by 'u' to untag packet (other ports default) before
        bringing it to the port, '*' is ignored

Samples:
1) ASUS WL-500g Deluxe stock config (eth0 is WAN, eth0.1 is LAN):
robocfg switch disable vlans enable reset vlan 0 ports "0 5u" vlan 1 ports "1 2
3 4 5t" port 0 state enabled stp none switch enable
2) WRT54g, WL-500g Deluxe OpenWRT config (vlan0 is LAN, vlan1 is WAN):
robocfg switch disable vlans enable reset vlan 0 ports "1 2 3 4 5t" vlan 1 ports
 "0 5t" port 0 state enabled stp none switch enable
```

3. Wireless configuration

3.1. Basic settings

NVRAM variable	Description
wl0_mode	**ap** = Access Point (master mode), **sta** Client mode
wl0_ssid	ESSID
wl0_infra	**0** = Ad Hoc mode, **1** = normal AP/Client mode
wl0_closed	**0** = Broadcast ESSID, **1** Hide ESSID
wl0_channel	1 / 2 / 3 /.../ 11 vhannel

See OpenWrtNVRAM for more NVRAM settings.

3.2. WEP encryption

NVRAM variable	Description
wl0_wep	**disabled** = disabled WEP, **enabled** = enable WEP
wl0_key	**1** .. **4** = Select WEP key to use
wl0_key[1..4]	WEP key in hexadecimal format (allowed hex chars are 0-9a-f)

Avoid using WEP keys with 00 at the end, otherwise the driver won't be able to detect the key length correctly.

A 128-Bit WEP key must be 26 hex digits long.

Setting up WPA will override any WEP settings.

3.3. WPA encryption

For enabling WPA, you need to install the nas package. When you enable or disable WPA settings, you should make sure that the NVRAM variable **wl0_auth_mode** is unset, because it is obsolete.

More information is on OpenWrtDocs/nas. (solve problem with WhiteRussian RC2 and client mode)

NVRAM variable	Description
wl0_akm	**open** = No WPA
	psk = WPA Personal/PSK (Preshared Key)
	wpa WPA with a RADIUS server

	psk2 = WPA2 PSK
	wpa2 WPA2 with RADIUS
	"psk psk2" or **"wpa wpa2"** = support both WPA and WPA2
wl0_crypto	**tkip** = RC4 encryption
	aes = AES encryption
	aes+tkip = support both
wl0_wpa_psk	Password to use with WPA/WPA2 PSK (at least 8, up to 63 chars)
wl0_radius_key	Shared Secret for connection to the Radius server
wl0_radius_ipaddr	IP to connect...
wl0_radius_port	Port# to connect...

3.4. Wireless Distribution System (WDS) / Repeater / Bridge

OpenWrt supports the WDS protocol, which allows a point to point link to be established between two access points. By default, WDS links are added to the br0 bridge, treating them as part of the lan/wifi segment; clients will be able to seamlessly connect through either access point using wireless or the wired lan ports as if they were directly connected.

Configuration of WDS is simple, and depends on one of two variables

NVRAM	Description
wl0_lazywds	Accept WDS connections from anyone (0:disabled 1:enabled)
wl0_wds	List of WDS peer mac addresses (xx:xx:xx:xx:xx:xx, space separated)

For security reasons, it's recommended that you leave wl0_lazywds off and use wl0_wds to control WDS access to your AP. wl0_wds functions as an access list of peers to accept connections from and peers to try to connect to; the peers will either need the mac address of your AP in their wl0_wds list, or wl0_lazywds enabled.

Easy steps for a successfull WDS:

First do it without wireless protection and then activate the protection. If you activate both you will double the pain to find a problem.

1. Configure the IPs of each AP - don't use the same! For easier maintenance you can use the same subnet.
2. Add the **other** APs MAC address to the list of allowed peers to each AP. With OpenWRT it's the variable wl0_wds.
3. Disable all the unneeded services like DHCP, port forwarding, firewalling etc. **except** on the AP the has the internet connection. Remember: The other APs only act as the extended arm of the internet connected AP.
4. Configure the WLAN parameters on all APs identical. That is SSID, channel, etc. - keep it simple. If you want to

try boosters etc. do this later.

5. Have you commited your values? Do it. And reboot.
6. Now connect a lan cable to each AP and try to ping the internet AP. It should answer. Else start checking the settings.
7. You are done. Now activate security on the devices. Optionally hide the SSID (wl0_closed=1). If WPA-PSK doesn't work chances are that a peer partner doesn't support it. Try WEP.

 Note: if you broke up your bridge as detailed in "To separate the LAN from the WIFI" above, this will not just work, since you no longer have a br0 device. You will have to add a bridge to one of your devices again, and create appropriate firewall rules, to make things work. There are currently no detailed instructions on how to set this up, so you better no what you are doing...

3.5. OpenWrt as client / wireless bridge

Starting with RC2 WhiteRussian basically the only thing you have to do is to switch the WL mode like with the bridge:

```
nvram set wl0_mode=sta
```

For more information, see ClientModeHowto

4. Software configuration

4.1. System

4.1.1. dnsmasq

Dnsmasq is lightweight, easy to configure DNS forwarder and DHCP server.

Documentation can be found at OpenWrtDocs/dnsmasq

4.1.2. nas

nas is the binary, Broadcom proprietary, tool that sets up security connection on wireless device.

Documentation with discovered feature can be found at OpenWrtDocs/nas .

4.1.3. wl

wl is a proprietary Linksys binary tool for setting the parameters of the wireless interface.

Documentation with discovered feature can be found at OpenWrtDocs/wl .

4.1.4. TimeZone and NTP

To set a Time Zone type something like the following line in /etc/profile:

```
export TZ="CET-1CETDST"
```

note: This sets TimeZome to GMT+1

If you want to use a TimeClient to Syncronize, use rdate for that (Note: rdate uses port 37/tcp on remote host). Create a file in /etc/init.d/ called S51rdate, with the contents:

```
#!/bin/sh
/usr/sbin/rdate 128.138.140.44
```

save it, and then type this at a prompt to make it executable:

```
chmod a+x /etc/init.d/S51rdate
```

Putting a TimeZone entry for the Systemlogger could also be an good idea simply put a line like something in /etc/TZ:

```
CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
```

note: this sets TimeZone for CET/CEST (Central European Time UTC+1 / Central European Summer Time UTC+2) and the starting (5th week of March at 02:00) and endtime (5th week of October at 03:00) of DST (Daylight Saving Time).

More can be found here: http://leaf.sourceforge.net/doc/guide/buci-tz.html#id2594640 and: http://openwrt.org/forum/viewtopic.php?id=131

Examples:

Australia	Melbourne,Canberra,Sydney	EST-10EDT-11,M10.5.0/02:00:00,M3.5.0/03:00:00
	Perth	WST-8
	Brisbane	EST-10
	Adelaide	CST-9:30CDT-10:30,M10.5.0/02:00:00, M3.5.0/03:00:00
	Darwin	CST-9:30
	Hobart	EST-10EDT-11,M10.1.0/02:00:00,M3.5.0/03:00:00

Europe	Amsterdam, Netherlands	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Athens, Greece	EET-2EEST-3,M3.5.0/03:00:00,M10.5.0/04:00:00
	Barcelona, Spain	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Berlin, Germany	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Brussels, Belgium	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Budapest, Hungary	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Copenhagen, Denmark	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Dublin, Ireland	GMT+0IST-1,M3.5.0/01:00:00,M10.5.0/02:00:00
	Geneva, Switzerland	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Helsinki, Finland	EET-2EEST-3,M3.5.0/03:00:00,M10.5.0/04:00:00
	Lisbon, Portugal	WET-0WEST-1,M3.5.0/01:00:00,M10.5.0/02:00:00
	London, Great Britain	GMT+0BST-1,M3.5.0/01:00:00,M10.5.0/02:00:00
	Madrid, Spain	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Oslo, Norway	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Paris, France	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Prague, Czech Republic	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Roma, Italy	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Stockholm, Sweden	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
New Zealand	Auckland,Wellington	NZST-12NZDT-13,M10.1.0/02:00:00, M3.3.0/03:00:00
USA & Canada	Hawaii Time	HAW10
	Alaska Time	AKST9AKDT
	Pacific Time	PST8PDT
	Mountain Time	MST7MDT
	Central Time	CST6CDT
	Eastern Time	EST5EDT
	Atlantic Time	AST4ADT
Asia	Jakarta	WIB-7

Please update and include your Time Zone.

You can find more on timezones on timeanddate.com.

4.1.5. Crontab

HowtoEnableCron

4.1.6. PPPoE Internet Connection

Be sure you have the pppoe modules/packages installed!

Set the nvram according to

```
nvram set wan_ifname=ppp0
nvram set wan_proto=pppoe
nvram set ppp_mtu=1492 # The MTU of your ISP
nvram set pppoe_ifname=vlan1 # For WRT54GS only. This should be your wan port.
nvram set ppp_username=your_isp_login
nvram set ppp_passwd=your_isp_password
nvram commit
```

and reboot.

Use ifconfig (device ppp0) or ping to determine the link is up. If there is no link enable **debug** in */etc/ppp/options* and use **logread** to check the error messages.

If you have services (vpn, ntpd) that should be started after the link is up, put the start-scripts in */etc/ppp/ip-up* (don't forget to chmod +x). For example:

```
#!/bin/sh

sh /etc/init.d/S55ntpd start
/usr/sbin/openvpn /etc/openvpn.conf
```

4.1.7. Access to syslog

If you want to read the syslog messages, use the **logread** tool.

4.2. Applications

4.2.1. httpd

httpd is the binary, part of BusyBox, tool that start http daemon.

Documentation can be found at OpenWrtDocs/httpd .

4.2.2. socks-Proxy

There is a socks-proxy available for OpenWRT, it is called **srelay** (Find via the package tracker). However, there is no documentation for this package. So, here is a quick guide:

srelay comes with a configuration file: /etc/srelay.conf (surprise surprise). It has some examples, but basically you will want to do this:

```
192.168.1.0/24 any -
```

This should give every computer in the 192.168.1-Subnet access to srelay while keeping everything else out.

Then start srelay: **srelay -c /etc/srelay.conf -r -s**. Find out more about the available options with **srelay -h**.

Keep in mind that this information was found using trial-and-error-methods, so it might still be faulty or have unwanted side effects.

4.2.3. uPnP

uPnP is Universal Plug and Play. You can use either the LinkSys binary from the original firmware or the compiled version.

Documentation and the background of uPnP can be found at OpenWrtDocs/upnp

4.2.4. CUPS - Printing system with spooling

You can not print a testpage on the local cups, because this would need to have ghostscript installed on your embedded system.

If you have a special Postscript Printer Description (ppd) file for your printer, copy it to /usr/share/cups/model/ and restart cupsd. Cups will install it in /etc/cups/ppd and you can choose it via the web interface. (192.168.1.1:631)

If you have problems with permissions, try to change /etc/cups/cupsd.conf to fit your local TCP/IP network:

```
<Location />
Order Deny,Allow
Deny From All
Allow from 127.0.0.1
Allow from 192.168.1.0/24 #your ip area.
</Location>
```

MacOS X tip: Configure your extended printer settings. If you use the standard printer settings and add an ipp printer, macosx will add after the server adress /ipp . But this class etc. does not exist on your cupsd.

5. Hardware

5.1. LED

Document can be found at wrtLEDCodes

Almost all of these pages are editable, create an account and click the edit (🗨) button at the top of the page.

1. Disclaimer

The contents of this section of the wiki can have serious consequences. While every effort has been made to test and verify the items herein, if executed incorrectly, or if you just happen to have a bad day you COULD SERIOUSLY DAMAGE YOUR HARDWARE. Neither I (inh) nor anyone else will be held responsible for anything you do.

2. Hardware

2.1. Serial Console

Serial ports allow you to do a myriad of things, including connect to your computer, connect to other devices such as LCDs and GPSes, etc... With a little programming, you could even connect a bunch of routers together.. This mod doesn't *add* serial ports; those are already there. This just makes them much easier to use with just about any hardware you want.

2.1.1. Finding Serial Console

(stealed from the AR7Port page) The method used to find the serial port was suggested to me on irc; use a piezo buzzer and attach it's ground (usually black) wire to a ground point on the router - the back of the power regulators are usually good candidates, but check this with a multimeter/voltmeter... Use the other wire to probe any of the header pins which may be pre-installed, or any of the component holes which look like they could have header pins installed into. Once you get the right pin, the piezo should make a screeching sound much like that of a 56kbps connection.

Make sure you reset the router after probing each pin. The bootloader/linux bootup messages will only happen for a few seconds, after that the serial console will be silent - so even if you have the right pin you will not hear anything.

A more accurate method would be to use either a logic analyzer or an oscilloscope, but these are expensive and for the basic task of locating a serial pin a little overkill.

Unfortunately, BCM4702KPB SoC shares some UART pins with Ethernet0 port, so UART is disabled via GPIO. You have to use external NS16c550-compatible UART for devices based on this chip (i.e. WAP54Gv1 or Asus WL-300G). There is the large 20-pin jumper block connected to the CPU I/O data lines, which can be connected to an external UART.

ASUS WL-500b/g: http://wl500g.info/showthread.php?t=587&page=1&pp=15

2.1.2. Home-made RS-232 kit

Background

Most OpenWrt compatible devices have one or two serial ports on the router's pcb (printed circuit board.) The problem is they operate on 3.3v, which means **they will get fried if you connect them to your computer's serial port**, which operates at 12v. Luckily, this is more common a thing than you would think, and as such, Maxim (no, not the magazine) has made a few handy little ICs for us to use. The newest (and IMHO best) is the MAX233, or more specifically, the MAX233a, which has a higher speed capacity and uses less power. This guide will tell you how to solder everything together to get a pc-compatible serial port on your OpenWrt router.

http://jdc.parodius.com/wrt54g/serial.html

http://www.nslu2-linux.org/wiki/HowTo/AddASerialPort

2.1.3. USB Kit

A USB based data cable for a mobile cell phone is another possibility.

http://www.nslu2-linux.org/wiki/HowTo/AddASerialPort

- note: For the serial console on a WRT54G with a USB cell phone cable, the following pins are used: 4(tx), 6(rx), 10(gnd)

2.1.4. Adding Dual Serial Ports

http://www.rwhitby.net/wrt54gs/serial.html

2.1.5. Terminal software

Hyperterm

Minicom

2.2. Adding an MMC/SD Card

This is one very cool mod! Credit goes to kiel.kool.dk for this awesome work. They have also pioneered some other interesting mods as well. Check out http://duff.dk/wrt54gs/ for info. They created this mod for the wrt54g version 2, then I (INH) ported it to version 3. If you have another version, you are going to have to figure out how to port it.. but it shouldn't be too hard.

Introduction

This mod allows you to read and write from a MMC/SD card. This is awesome as it can literally give you 555 time the storage space. You can now have over one gigabyte of memory to store and run programs from, store packet logs, etc etc.. It's not a very hard mod to do, unless you have something other than a wrt54g version 2 or 3. If thats the case, please read on, as I go over how I ported this mod to my version 3.

2.2.1. Installing on a wrt54g version 2 and 2.2

The following is the guide from kiel.kool.dk by Rasmus Rohde and Mads Ulrik Kristoffersenon about installing an MMC/SD card reader/writer in a wrt54g version 2, with added commentary where I feel is appropriate

I added now comments for WRT HW-version 2.2 where the GPIO locations are different, but the general procedure is the same.

This project is for people who would like to add a little storage to their Linksys WRT54G router besides the builtin 4MB flash ram. What we will do is connect an SD card reader to some of the GPIO pins of the CPU found inside the Linksys and with the help of a little driver we can use as a block device from Linux. This means that if you compile your kernel for the Linksys with e.g. support for MSDOS partitions and VFAT you will be able to mount, read, write, partition and so on your normal SD cards. The speed obtainable for reading and writing seems to be about 200 KB/s.

Pictures

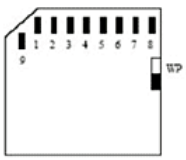
- The insides of the router with SD card reader installed
- The finished product with SD card reader installed

What you need

- A soldering iron and a bit of tin solder (and a little bit of soldering skills)
- An SD card reader unless of course you want to solder directly on the card
- 6 pieces of thin wire
- A Linksys WRT54G (hardware version 2)

How to proceed

1. For the SD card to work we need to attach 6 wires inside the router. This drawing of the SD card should give an idea of the pins that come into play:

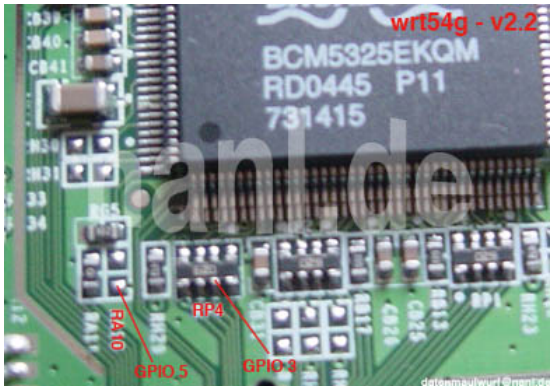


1. CS - Chip Select for the SD card	GPIO7
2. DI - Data in on the SD card.	GPIO5
3. VSS - Ground is a good thing	GND
4. VDD - We need power of course. 3.3V will do the job	3.3v
5. CLK - The clock we generate for the SD card	GPIO3
6. VSS2 - Another ground is also a good thing	GND
7. DO - Data out from the SD card	GPIO4

- We will be driving the SD card in SPI mode, meaning that only one of the four data out pins are used (pin 7). Obtaining the specs for driving the card in the native SD mode is VERY costly and furthermore the limited number of GPIO pins available inside the router also mandates the use of some sort of serial protocol. The two VSS pins can simply be wired together for this project (VSS2 is used to control the sleep mode of the card). With this in mind lets look at the solder points in the router.
    1. The first three solder points are located at RP3
    2. The next two solder points are located at JP1
    3. The last solder point is at the DMZ LED

For Version 2.2 hardware:

- GPIO 3 can be found on Pin 3 of RP4 (near the BCM switch IC), just left of it you can find GPIO 5 next to the RA10 Text label. GPIO 4 is located near the RA13 Text label (near to the Power LED)



- This is a picture of the GPIO 3+5 for wrt-Version 2.2 taken from http://nanl.de/nanl/

Proceed by soldering a wire to each of the 6 solder points. Pay special attention not to short circuit the pins of RP3 - even though these solder points were chosen because they provide the most spacious access point to the GPIO lines needed, it's still pretty tight quarters, so watch out!

1. By now the wires should be attached nicely inside the router, so that we may continue to connect them to the SD card (reader). This picture shows the SD card reader. It is pretty easy to solder on that one.
2. Mount the card reader somewhere inside your router. We chose the right hand side of the top cover, using double sided duct tape to make it stick and drilled a small slot to allow cards to be inserted and removed with the cover closed. See the picture links at the top of the page to see what this looks like and check this picture of the actual hole.
3. That was easy. We are now ready for the software part.

Software

First of all we suggest that you configure a kernel with support for MSDOS partitions and VFAT. Partition support must be built into the kernel whereas VFAT can be built both as a module or into the kernel. These are some things you may want to include in your .config:

```
CONFIG_PARTITION_ADVANCED=y
```

```
CONFIG_MSDOS_PARTITION=y
CONFIG_FAT_FS=y
CONFIG_MSDOS_FS=y
CONFIG_VFAT_FS=y
```

Now get the driver and the Makefile. You will need to modify the Makefile to point to where your OpenWRT linux kernel headers are and also the mipsel compiler location. When that is done just type make (ignore the warnings - they are OK).

The module is now ready to be inserted. Make sure a card is placed in the reader and then load the module. Check with dmesg that everything went OK, and hopefully you should now have some new devices in /dev/mmc/... Here is a little snippet of a "conversation" with the router

```
root@radio:~# ls -al /lib/modules/2.4.20/
drwxr-xr-x    1 root     root            0 Jan  1 00:08 .
drwxr-xr-x    1 root     root            0 Jan  1 00:01 ..
lrwxrwxrwx    1 root     root           28 Jan  1 00:01 et.o -> /rom/lib/modules/2.4.20/et.o
-rw-r--r--    1 root     root        50616 Jan  1 00:02 fat.o
-rw-r--r--    1 root     root        12780 Jan  1 00:08 mmc.o
-rw-r--r--    1 root     root        11244 Jan  1 00:03 msdos.o
-rw-r--r--    1 root     root        19156 Jan  1 00:05 vfat.o
lrwxrwxrwx    1 root     root           28 Jan  1 00:01 wl.o -> /rom/lib/modules/2.4.20/wl.o
root@radio:~# insmod mmc
Using /lib/modules/2.4.20/mmc.o
root@radio:~# dmesg | tail -7
mmc Hardware init
mmc Card init
mmc Card init *1*
mmc Card init *2*
Size = 249856, hardsectsize = 512, sectors = 499712
Partition check:
 mmca: p1
root@radio:~# insmod fat
Using /lib/modules/2.4.20/fat.o
root@radio:~# insmod msdos
Using /lib/modules/2.4.20/msdos.o
root@radio:~# mount /dev/mmc/disc0/part1 /mnt -tmsdos
root@radio:~# ls -al /mnt
drwxr-xr-x    2 root     root        16384 Jan  1  1970 .
drwxr-xr-x    1 root     root            0 Jan  1 00:01 ..
-rwxr-xr-x    1 root     root            0 Jan  1 00:07 bossepr0.pic
-rwxr-xr-x    1 root     root        22646 Jan  1 00:02 ld-uclib.so
-rwxr-xr-x    1 root     root        12780 Jan  1  2000 mmc.o
-rwxr-xr-x    1 root     root      1048576 Jan  1  2000 temp.bin
-rwxr-xr-x    1 root     root     16777216 Jan  1  2000 temp2.bin
-rwxr-xr-x    1 root     root     16777216 Jan  1  2000 temp3.bin
-rwxr-xr-x    1 root     root          693 Jan  1  2000 temp4.bin
root@radio:~# df
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/root                  896       896         0 100% /rom
/dev/mtdblock/4           2176      1580       596  73% /
/dev/mmc/disc0/part1    249728     33856    215872  14% /mnt
```

A little help with kernel compilation

The easiest way to get a kernel running with the needed fs support is probably by downloading OpenWRT and building the flash image. When you are familiar with this process, it is quite easy to change the settings for your kernel. Just go to buildroot/build_mipsel/linux and type make menuconfig. Go to file systems -> Partition Types and check "Advanced partition selection" and "PC BIOS (MSDOS partition tables) support". In "File systems" you should also check "DOS FAT fs support" and optionally "VFAT (Windows 95) fs support". When done just exit saving the changed and type make dep zImage to force a rebuild of the kernel. Then you can just rebuild your OpenWRT image and the new kernel will be included automatically. GPIO pins, eh?

The integrated Broadcom CPU BCM4712 used in the WRT54G provides a number of General Purpose Input/Output pins (or GPIO pins) that are used for various purposes in the router. We have been able to identify 8 such pins until now and these are assigned as follows:

Pin	Direction	Name
GPIO 0	(Output)	WLAN LED
GPIO 1	Output	POWER LED
GPIO 2	Output	ADM_EECS
GPIO 3	Output	ADM_EESK

GPIO 4	Input	ADM_EEDO
GPIO 5	Output	ADM_EEDI
GPIO 6	Input	Reset button
GPIO 7	Output	DMZ LED

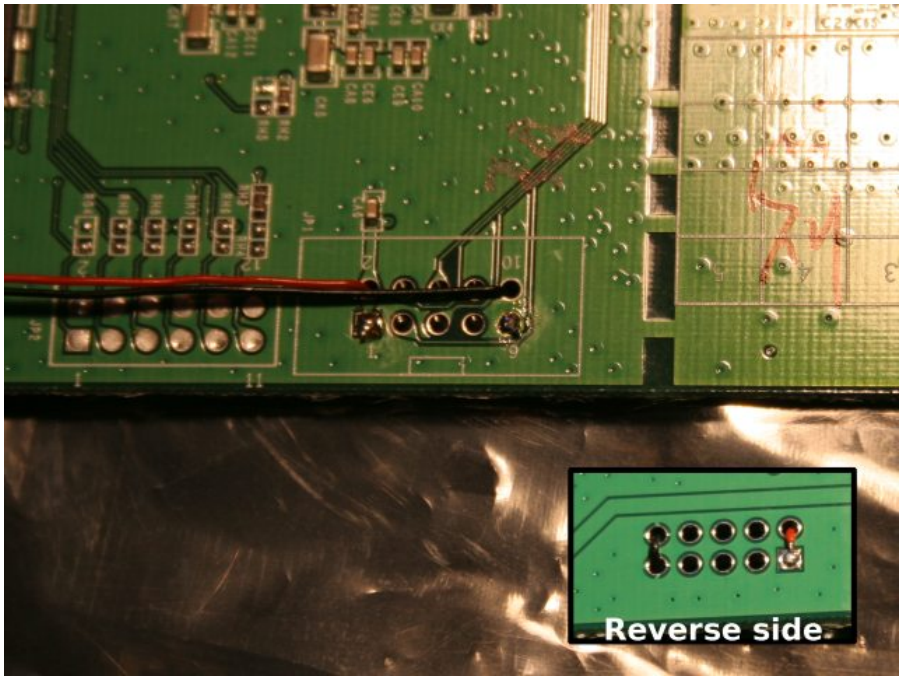
The pins used in this project are the ADM_EESK, ADM_EEDO, ADM_EEDI and DMZ LED pins. The ADM_* pins constitute an interface used to configure the ADMTek switch chip. Since this only happens during the boot process, we are free to use these pins to our likings afterwards (the corresponding pins on the switch chip will be tri-state after configuration). The names of the other pins should be self explanatory. The direction of the pins can be individually programmed (even though this of course does not make sense for every pin).

2.2.2. Installing on a wrt54g version 3 and 3.1

*to be written, in the meantime you can find version 3 info here.

Basically the same as above, but different GPIO points on the board.

Power - 3.3v (red), and GND (black). I looped through the board for strength of connection:



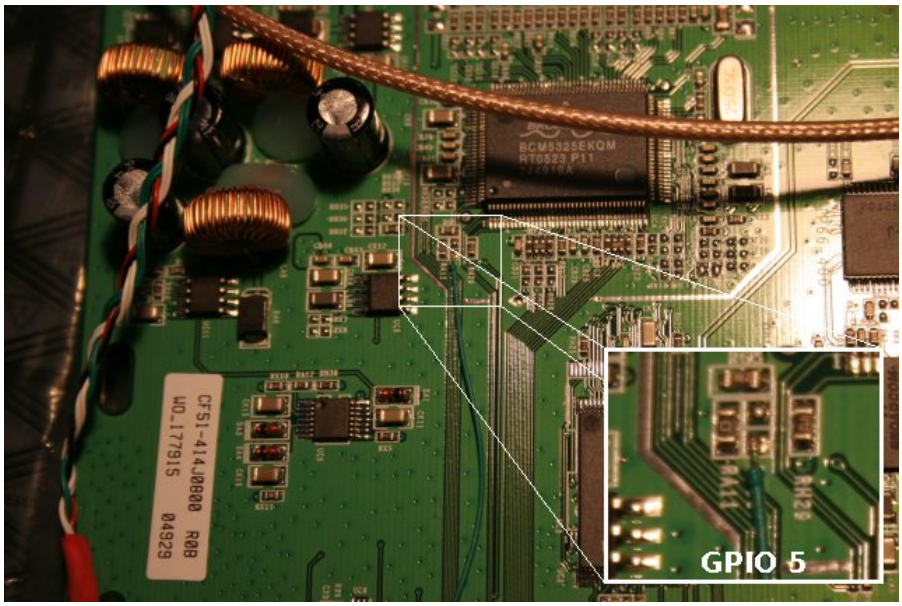
GPIO 3, as mentioned in the URL above, on the right hand side of the amber LED:

http://www.ethernal.org/wrt54g/pics/button.jpg

GPIO 4 and 7:

http://www.ethernal.org/wrt54g/pics/underside.jpg

GPIO 5 - definitely right next to the "RA10" label:



2.2.3. Porting to other platforms

*almost done being written

2.3. USB

If your WRT* has a USB port, you could attach a lot of USB devices.

- http://www.linux-usb.org/
- USBDeviceSupport @NSLU2 Linux

2.3.1. add USB to your Siemens SE505

On the side with the powerplug you will find some 'C's

- add C906 with 100µF 16Volt

- add C986 with 10µF 16Volt

- add U981 with an LM7805

This will support the +5 Volt to your USB-Port.

Go to the other side of the PCB wehre the antenna is placed.

- add wire to F51 as Fuse

- add to 'R' about 15kOhm to R723 and R724

- shorten R733 R734

- put an USB-Plug to J51

Thats all.

2.3.2. USB Hard Drive

Already done, see UsbStorageHowto.

All "USB Mass Storage" class devices will work too: USB-to-IDE, some cellphones, come digital cams e.t.c.

2.3.3. USB Serial port/Modem

It is possible to connect a USB HUB and up to 127 USB-to-RS232 convertors.

Some USB cellphone datacables are dirt cheap and contains a USB-to-RS232 convertor (i.e. Prolific PL2303).

2.3.4. USB Keyboard/Joystick

Hmmm...

2.3.5. USB Sound devices

http://wiki.openwrt.org/UsbAudioHowto

http://www.nslu2-linux.org/wiki/HowTo/SlugAsAudioPlayer

Logitech USB Headset for PlayStation 2

Micronas UAC355xB USB Codec

2.3.6. USB Webcam

Check out this page: http://www.nslu2-linux.org/wiki/HowTo/AddUsbWebcam

Asus WL-500G-Deluxe has a Webcam support and Motion Detection software in the default firmware.

If you have a Philips-based cam (Philips and many Logitechs, also others) and a USB port, you can try the following (tested by me on an ASUS WL500GX):

1. Grab the "Tom" package from here: http://wl500g.info/showpost.php?p=8610&postcount=17 (and be sure to read through some of the posts) and install it, then erase the /lib/modules/2.4.20/pwc.o file
2. Download the kmod-videodev and kmod-pwc packages from http://downloads.openwrt.org/people/nico/whiterussian/packages/
3. Install them
4. Plug in you camera and enjoy! You can use camsrv to stream images, mvc as a simple motion detector... or compile your own programs.

Note: the video device will most likely be /dev/v4l/video0 instead of the common /dev/video0, because of devfs. Just use the correct parameters when you invoke the programs.

2.3.7. USB Ethernet

If you need one (2..3..127) additional Ethernet ports, it is possible to use USB-to-Ethernet adaptor.

As example, Genius (KYE) GF3000U, Linksys USB100TX, D-Link DSB-650TX which are based on the ADMtek Pegasus AN986.

Most of this devices has 10/100Mbit/s Full-Duplex Ethernet interface, but transfer rate is about 10Mbit/s only.

2.3.8. USB Bluetooth

It is possible, see this thread in the Forum.

2.3.9. USB VGA

http://www.winischhofer.at/linuxsisusbvga.shtml

2.4. Mini PCI and PCI

According to PCI-SIG: *The Mini PCI specification defines an alternate implementation for small form factor PCI cards referred to in this specification as a Mini PCI card. This specification uses a qualified subset of the same signal protocol, electrical definitions, and configuration definitions as the Conventional PCI Specification.*

In other words it is a compact 3.3V version of venerable PCI. Many Mini PCI devices are available today: sound cards, IDE/ATA and SATA controllers, and even accelerated SVGA cards. For example: miniPCI and miniAGP Cards.

It is possible to remove a Wi-Fi Mini PCI card and insert another device. Fortunately, some A/G dual-standart WRT* models have two Mini PCI slots.

Because Mini PCI and PCI are cousins, you can use **regular PCI cards** with your Mini PCI-equipped hardware using Mini PCI-to-PCI converter. Information on some Mini PCI-to-PCI converters can be found here:

- IM300 Mini PCI Type III to PCI Adapter Card
- IM380 Mini PCI Type III to PCI Adapter Card with **two** PCI slots, one 3.3V and one 5V --- check out juicy pictures!
- Costronic's Mini PCI-to-PCI CV09MP-P series.

2.5. Adding a GPS

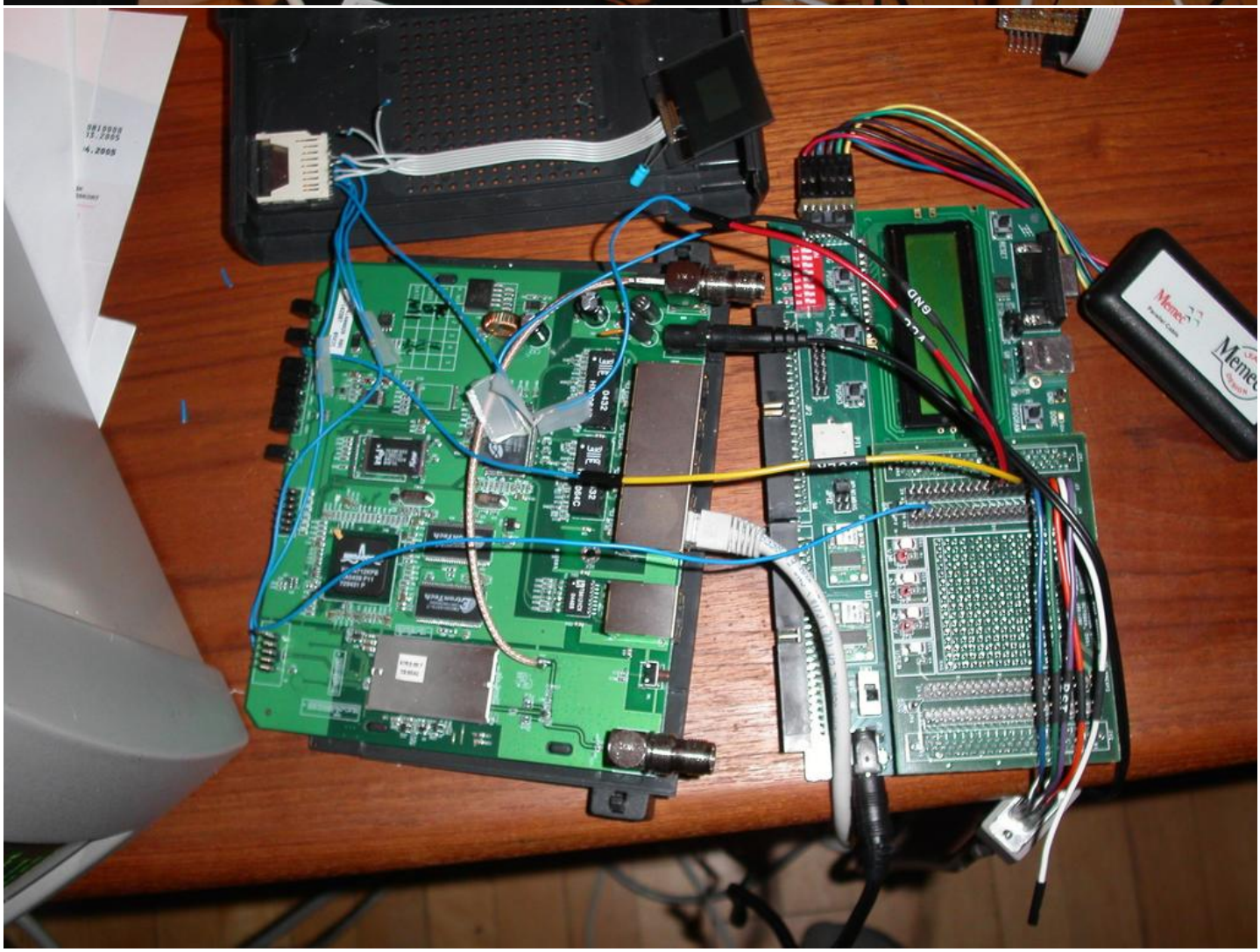
Adding a GPS to your router may seem like an odd idea, but it does have it's uses. If you like to war drive, this combined with the SD card mod would let you simply plug in the router to your cigarette lighter and go, logging the networks to the sd card. It also isn't a hard mod to do. Depending on your GPS, this may be as simple as soldering 3 wires to your router. In my case it was a little more complicated, but by no means hard. It was just like adding a serial port, but instead of adding the serial port, I added the GPS.

2.6. Adding a Weather Station

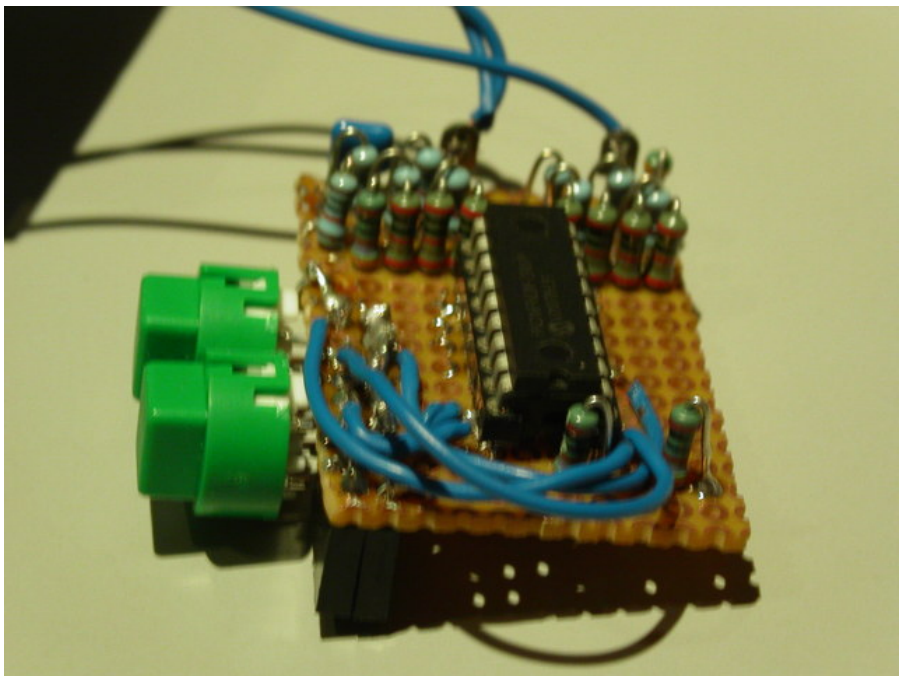
Connect a WX200 / WM918 / WMR918 / WMR968 weather station http://david.zope.nl/hardware/wl500g/wx200wl500g/

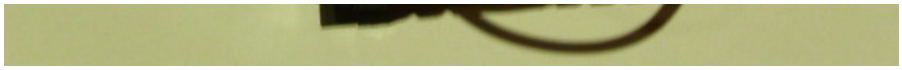
2.7. Adding an LCD

2.8. Adding VGA Output



2.9. Adding Second Reset Button (v2.2 only)





2.10. Adding Sound Output

See also: http://wiki.openwrt.org/UsbAudioHowto

2.11. Adding a Power Button

2.12. Adding a Power Reset Button

2.13. Making it Mobile

http://yasha.okshtein.net/wrt54g/4m.jpg

http://yasha.okshtein.net/wrt54g/ How to Mobilize a WRT54g

2.14. Adding i2c bus

i2c bus allows you to extend the IO ability beyond just 8 bits of IO.

Inital docs are here http://www.byteclub.net/wiki/index.php?title=Wrt54g

2.15. Power Over Ethernet/Power Requirements

The internal voltage regulators on the WRT54g (version 3) are made by Anachip Corp. http://www.anachip.com

The 1509-33 is actually a AP1509 part, and is the 3.3V regulator.

The 1509-50 is actually a AP1509 part as well, and is the 5.0V regulator.

The minimum operating voltage of the 5V regulator is 7VDC, absolute maximum is 24VDC, per the datasheet http://www.anachip.com/downloads/datasheets/power/sw_reg/AP1509.pdf

3. Software

3.1. Things not to compile in

When you change things in the configs yourself, only active the following if you realy know what you are doing.

Busybox Configuration - General Configuration - Support NSA Security Enhanced Linux = N

Busybox Configuration - Init Utilities - halt = N

Busybox Configuration - Init Utilities - poweroff = N

Busybox Configuration - Networking Utilities - ifupdown = N

3.2. Software Tools

3.2.1. Networking

WRTbwlog	A tool that shows internet traffic on all wired and wireless interfaces, as well as many other useful and related functions

3.2.2. System

3.2.3. Wireless

WiViz	A very nice wireless network visualization tool

3.3. Software Guides

3.3.1. Wireless

3.3.1.1. Client Mode

See ClientModeHowto

3.3.2. System

3.3.2.1. LED System Load Monitor

Credit goes to SeRi for starting this mod. He had it use the wrt's white and amber LEDs (version 3 only) to show system load. I thought it was a very nifty mod, but I couldn't use it, as the white and amber LEDs are used for the read/write lines on the SD card mod. So what did I do? I modded the mod of course! Now anyone with a spare LED can use this mod. you just need to set the correct GPIO pin. For wrt54g's version 2-3, gpio 7 is for the DMZ LED, which is what I use. You can modify the source accordingly. This will flash the LED once per second under normal useage, twice per second under medium load, and when there is a high load on the system, the LED flashes 3 times per second.

NOTE: You will need to compile your kernel with the Busybox option for usleep enabled. This is what is used for the LED strobing

Installing Necessary Software

First of, grab the loadmon.sh script, and [mbm]'s GPIO tool. Then untar the gpio tool, and copy the files to your /usr/sbin directory. A typical way to do this on a jffs2 install would go as follows. If you are using squash fs, then you shoudl know what to do.

```
cd /tmp
wget http://downloads.openwrt.org/inh/programs/loadmon.sh
wget http://downloads.openwrt.org/gpio.tar.gz
gzip -d gpio.tar.gz
tar xvf gpio.tar
mv gpio /usr/sbin
mv loadmon.sh /usr/sbin
```

Now that everything is in place, you need to edit your configuration files to start up the script manually when the router boots. To do this, create a script in /etc/init.d to start loadmon.sh. Here's a simple way to do that:

```
echo "#!/bin/sh" > /etc/init.d/S60loadmon
echo "/usr/sbin/loadmon.sh &" >> /etc/init.d/S60loadmon
chmod +x /etc/init.d/S60loadmon
```

Now reboot and test it out

If you dont want to build your own firmware, and you own a router with the white and orange lights, you can try this script. It will show the white light if load is low, white and orange at medium load, and orange at high load:

```
#!/bin/sh

#Set GPIO to the GPIO of the LED you wish to use.
# Default is 7 for DMZ LED on most routers..
GPIOG=2
GPIOR=3
DELAY=2
HIGHLOAD="70"
```

```
MEDLOAD="30"

while sleep $DELAY; do
   load=$(cat /proc/loadavg | cut -d " " -f1 | tr -d ".")
   #echo $load

    if [ "$load" -gt "$HIGHLOAD" ]; then
        gpio enable $GPIOG
        gpio disable $GPIOR
    elif [ "$load" -gt "$MEDLOAD" ]; then
        gpio disable $GPIOG
        gpio disable $GPIOR
    else
        gpio disable $GPIOG
        gpio enable $GPIOR
    fi
done
```

3.3.2.2. Transparent Firewall

See TransparentFirewall

4. Firmware

4.1. Overclocking

Overclocking the WRT has been a very sought-after mod. Many people overclock their home PCs, and now I will tell you how to overclock your OpenWrt router. Please read the "troubleshooting" section at the bottom of this document, it contains important information on things you should do before trying to overclock.

Background Info

Many people know that by setting the nvram variable clkfreq, you can overclock your router. Many people also know that Linksys actually released a beta firmware, changing clkfreq to 216 to fix stability issues. That quick fix actually works quite well, as many people can tell you. Linksys also released a lesser-known beta firmware that set clkfreq to 240. There are also a few things discovered by [mbm] and I that seem to affect performance. First off, you can NOT set clkfreq to any number you want. It is very selective, and only certain values work. Also, there are 2 clocks you can adjust. This was previously unknown (read: another OpenWrt first.)

NOTE: While many people have had success with this, some have not. It is HIGHLY recommended that you flash the modified CFE images I (inh) provide at http://downloads.openwrt.org/inh/cfe/ in case something goes wrong. Otherwise you will have to setup a JTAG cable to debrick. Even the moderate/simple overclocking suggested here has been reported to fail. Even though the clock rate is valid (like the 216 stability fix), it has caused a router to constantly reboot.

Simple Overclocking

As stated earlier, Linksys released firmware that made their routers run at 216 MHz instead of 200 to fix stability issues. You too can do this simple overclock to make your router run much more solidly. Here is all you have to do.

At the OpenWrt prompt, type:

```
nvram set clkfreq=216
nvram commit
```

Thats it! Reboot your router by either unplugging it and plugging it back in, or by typing:

```
reboot
```

Simple enough! If your router was unstable with high traffic loads before, you should be much more stable now

Moderate Overclocking

While a 16mHz increase doesn't seem like much, it works wonders for the router. But what if you want to go faster? Setting clkfreq to 220 locks up the router, and then you are stuck with having to use the JTAG method to de-brick. That is, of course, assuming you didn't change the default values in the CFE file, in which case all you have to do is reboot with the reset button held in... see the 'changing cfe defaults' guide)

Anyways, back on topic.. More speed! The trick with making it run faster is setting the right clkfreq values. The wrong ones turn your router into a brick. Here is a list of values that are known to work: 192,200,216,228,240,252,264,272,280,288,300

I've personally tested all of them on my wrt54g version 3, and they all worked. There is one caveat however; values above 264 seem to have no change. By checking the cpuinfo, it still reports the BogoMIPS as 264, even if clkfreq is set above that. To check your cpuinfo, type:

```
cat /proc/cpuinfo
```

Try the values, test your performance, or just bask in all your overclocking glory

Advanced Overclocking

This is the good stuff, especially if you have done the MMC/SD card mod, as it boosts the read/write speeds from 200 kilobytes a second to over 330

In addition to setting clkfreq to a higher number, there is also another clock that can be controlled. This is called the sb clock, and is believed to be the clock that controls the speed of the data transfer between different areas of the Broadcom CPU. To set it, you set clkfreq like this:

```
nvram set clkfreq=MIPSclock,SBclock
```

For example, the following does the same as if you were to set clkfreq to 264:

```
nvram set clkfreq=264,132
```

MIPSclock is the standard clock you change when setting clkfreq with one value. The second number you set it to is the aforementioned SBclock. The SBclock, just like the MIPSclock, only has certain values that can be used, or it will brick your router. Here's a table:

MIPSclock	SBclock
192	96
200	100
216	108
228	101333333
228	114
240	120
252	126
264	132
272	116571428
280	120
288	123428571
300	120

Some users have reported problems going above 240; you will need a JTAG cable to erase nvram if the clkfreq setting doesn't work.

Those are all values known to work. You can either set just the MIPSclock by using that value, or set both MIPS and SB clocks by using:

```
nvram set clkfreq=MIPSclock,sbclock
```

You can also mix and match values. I've personally found that setting MIPSclock to 300 and SBclock to 96, I get much better performance.

Conclusion

The clock seems to still remain somewhat of a mystery. With the recently discovered SBclock, and table of usable values, overclocking is a much more feasible and safe mod than it used to be.

Troubleshooting

Possible NEW Recovery Method (no jtag from RawDigits): I set my wrt to 264 MHz today (didn't read well enough before trying this) which resulted in a router constantly rebooting. I noticed that the number of seconds I could ping the router got progressively shorter after each boot, so I thought heat might be at fault here. I placed my WRT in the freezer, and then ran a power cord and cat5 to it. This resulted in a stable WRT at the higher frequency, allowing me to reset the NVRAM var for clkfreq. The WRT is now running at its friendly 200 MHz with no issues whatsoever. Let me know if this works for you (rawdigits@hotmail.com).

One another method (also no jtag and no freezer need): With my Linksys WRT54G2 I downloaded my complet nvram to my desktop computer first. I try to overclock with clkfreq=216. I realized that the router make booting again and again. But I managed to debrick it. I unplug the router's power supply, press and hold down the reset button, then plug in. In the next seconds the nvram will be erased and fill up with default settings. My router boots after normally. After all I have to do only to set up the original nvram settings from my downloaded one. (szimszon@integrity.hu)

Setting an invalid clkfreq value can have a very undesirable effect: complete router lockup, AKA 'bricking.' Normally bricking isn't that bad of a thing. You can simply use the JTAG method to de-brick. When setting clkfreq values, however, you must take extra care. If you set an invalid window, you have a VERY VERY small time frame to get the jtag to erase the nvram before the CPU locks up. Rough estimates give a window of 1/2 second. If you have ever had to do this, it is a very big annoyance. A better solution is to add the nvram value reset_gpio to the default nvram stored in the cfe. By setting the right value to reset_gpio, and flashing the modified cfe back on to your router, if you do set a wrong clkfreq value, all you have to do is reset with the reset button held in, and everything will reset back to defaults. Details on this method can be found here.

4.2. Changing CFE defaults

The following is a guide from http://wl500g.dyndns.org/wrt54g.html that I've copied here, with added commentary. I am not the original author, that credit goes to Oleg.

 IMPORTANT: This information provided AS IS, without any warranties. If in doubt leave this page now. This information applies to WRT54G hw rev 2.0, 2.2, 3.0. No other units were tested, but most likely WRT54GS units should be the same. WRT54G hw rev 1.x use different layout, so you need to adjust things accordingly.

The wrt54g v.2.2 unit was kindly donated to me by maxx, the member of the forum.chupa.nl forum. I would like to publically say thank you to him.

Extracting default values

Telnet/ssh to your router running your favorite firmware and type the following

```
dd if=/dev/mtdblock/0 bs=1 skip=4116 count=2048 | strings > /tmp/cfe.txt
dd if=/dev/mtdblock/0 of=/tmp/cfe.bin
```

Copy both cfe.bin and cfe.txt to your linux box (this is required).

To copy files from your router to your computer, make sure the Dropbear package is installed, and type:

```
scp root@<router ip>:/tmp/cfe.bin /directory/on/your/computer
scp root@<router ip>:/tmp/cfe.txt /directory/on/your/computer
```

Check cfe.txt, it should look like this (this is from v.2.2):

```
boardtype=0x0708
boardnum=42
boardrev=0x10
boardflags=0x0118
boardflags2=0
sromrev=2
clkfreq=200
sdram_init=0x000b
sdram_config=0x0062
sdram_refresh=0x0000
sdram_ncdl=0x0
et0macaddr=00:90:4C:00:00:00
et0phyaddr=30
et0mdcport=0
gpio5=robo_reset
vlan0ports=1 2 3 4 5*
vlan0hwname=et0
vlan1ports=0 5
vlan1hwname=et0
wl0id=0x4320
il0macaddr=00:90:4C:00:00:00
aa0=3
ag0=255
pa0maxpwr=0x4e
pa0itssit=62
pa0b0=0x15eb
pa0b1=0xfa82
pa0b2=0xfe66
wl0gpio2=0
wl0gpio3=0
cctl=0
ccode=0
dl_ram_addr=a0001000
os_ram_addr=80001000
os_flash_addr=bfc40000
lan_ipaddr=192.168.1.1
lan_netmask=255.255.255.0
scratch=a0180000
boot_wait=off
watchdog=5000
bootnv_ver=2
```

Changing defaults

Open cfe.txt using text editor and change defaults in the way you like (but be extremely careful, as some changes could prevent device from booting and you will need to use JTAG cable to bring it back to life). For me I've decided to enable both Afterburner (Speedbooster) and set boot_wait to on by default, so reset to default no longer messes the things, so I've applied this pseudo-patch (please note, that I've added bit

0x200 to boardflags to enable afterburner):

```
-boardflags=0x0118
-boot_wait=off
+boardflags=0x0318
+boot_wait=on
```

To make life easier for me, I added "reset_gpio=6" to the cfe.txt file. This way, if I do set something wrong, like clkfreq, and the router just locks up, I wont have to try over and over again to hit a very slim window with the JTAG to erase the nvram. I can just hold reset when the router powers on, and it will use the default nvram values stored in the cfe.

If you do not understand some things in this file, do not try to edit it. This is also applies to afterburner. I've also tried to change default lan_ipaddr, but this does not work in the way I expect: CFE started to answer to ping request to new lan_ipaddr, but it does not accept tftp transfers...

Creating new CFE image

You will need a nvserial utility which comes with several GPL tarballs. Linksys supplies it in the wrt54g.1.42.3, wrt54g.1.42.2, wap55ag.1.07, wap54gv2.2.06. Launch nvserial in the way like this on your x86 linux box: *You can get nvserial from http://downloads.openwrt.org/people/inh/programs/nvserial*

```
nvserial -i cfe.bin -o cfe_new.bin -b 4096 -c 2048 cfe.txt
```

It works really slow, but it should finally create cfe_new.bin file for you, which has new embedded nvram.

Recompiling kernel with writable pmon partition

By default most firmwares has pmon partition write protected, i.e. you can't flash anything to this first 256k of flash. This is to prevent corrupting PMON/CFE. To remove this "lock" you will need to compile your own firmare with the following patch, you will need to copy the patch into "target/linux/linux-2.4/patches/brcm". (This patch works with WHITERUSSIAN RC3)

```
--- linux-2.4.30/arch/mips/bcm947xx/setup.c.orig        2005-09-21 11:24:09.000000000 -0400
+++ linux-2.4.30/arch/mips/bcm947xx/setup.c     2005-09-21 13:48:46.853425632 -0400
@@ -174,7 +174,7 @@
 #ifdef CONFIG_MTD_PARTITIONS

 static struct mtd_partition bcm947xx_parts[] = {
-       { name: "pmon", offset: 0, size: 0, mask_flags: MTD_WRITEABLE, },
+       { name: "pmon", offset: 0, size: 0 /*, mask_flags: MTD_WRITEABLE,*/ },
        { name: "linux", offset: 0, size: 0, },
        { name: "rootfs", offset: 0, size: 0, },
        { name: "nvram", offset: 0, size: 0, },
```

Flashing new CFE image

So, once you've recompiled and flashed your new firmware you need you upgrade CFE. This process is dangerous, as flash failure during it will prevent your unit from booting. Copy cfe_new.bin to your wrt54g and flash it. The exact commands are dependent on the firmware. With OpenWrt I've used the following:

```
mtd unlock pmon
mtd write -f /tmp/cfe_new.bin pmon
```

I recommend using the JTAG cable method for re-flashing your CFE. If something were to go wrong, you would end up needing the JTAG cable anyways. It's really cheap and easy to build, and makes it possible to recover from almost any error you make when writing to the flash. Check out http://openwrt.org/OpenWrtDocs/Troubleshooting

Checking it

Embedded nvram is only used, when real nvram is either corrupted or empty (CRC/magic checks fails), so you will need to erase nvram or to reset to defaults. With OpenWrt type this:

```
mtd erase nvram
```

Then cross your fingers and reboot your unit. And remember - I'm not responsible for any damage to your unit, as this information is provided AS IS for my own pleasure. oleg@cs.msu.su Posted: 2005-04-03

4.3. Customizing Firmware Image

It is relatively easy to create a custom firmware image which is pre-loaded with particular software packages and your own files. For example, it's easy to move the root home directory to /root, pre-load an .ssh/authorized_keys file, and modify /etc/passwd to include a stock password and point the root home directory at /root instead of the default /tmp. To do this you will need a Linux system and to download the source tar file. Extract this tar file, cd into the "openwrt" directory and look in the "docs" subdirectory. Documentation for customizing the image is located there.

The short form is that you first run "make" and will be presented with a configuration like the normal Linux kernel menuconfig. Use this to select various software packages and configurations. When done, customize the base file-system by modifying the filesystem image under "target/default/target_skeleton", and then run "make" again. This will run quite a while, well over an hour on a Pentium M 1.8 system. When complete, your customized firmware will be in the "bin" subdirectory, ready to install. If you make changes to the filesystem image, you'll need to regenerate the firmware with "make target_prepare

target_install". If you remove files, you will need to remove them from "build_mipsel/root" as well, or they will persist across new firmware image builds.

5. Downloads

5.1. Programs

Almost all of these pages are editable, create an account and click the edit (🗨) button at the top of the page.

OpenWrtDocs

1. Failsafe mode

If you've broken one of the startup scripts, firewalled yourself or corrupted the jffs2 partition, you can get back in by using OpenWrt's failsafe mode. To get into failsafe, plug in the router and wait for the DMZ led to light then immediately press and hold the reset button for 2 seconds. If done right the DMZ led will quickly flash 3 times every second.

(The act of switching between a normal boot and failsafe mode could change your mac address! This will invalidate the arp cache of the workstation you're using to access OpenWRT with. If you can't ping OpenWRT at 192.168.1.1, check the cache with 'arp -a' and delete it with 'arp -d 192.168.1.1'.)

(holding the reset button before the DMZ led can reset NVRAM)

(the ASUS WL-300G does not have a DMZ led : press the reset button for 2 seconds just after the AIR led lights, or maybe the LAN led. At some point it works, anyway.)

When in failsafe, the system will boot using only the files contained within the firmware (the squashfs partition) ignoring any changes made to the jffs2 partition. Additionally, various network settings will be overridden forcing the router to 192.168.1.1.

If you want to completely erase the jffs2 partition, removing all packages you can run firstboot.

If you want to attempt to fix the jffs2 partition, mount it with the following commands:

```
mtd unlock /dev/mtd/4
mount -t jffs2 /dev/mtdblock/4 /jffs
```

After the partition is mounted, you can edit the files in /jffs. If you run firstboot with the jffs2 partition mounted, it will not format the partition, but it will overwrite files with symlinks. (Packages will be preserved, changes to scripts will be lost)

Note: if you cannot figure out how to put your device into failsafe mode then remember that you can always modify the boot scripts in the source. So if you want to boot failsafe mode, you might edit your buildroot/build_mipsel/root/etc/preinit to something like this:

```
#!/bin/sh
mount none /proc -t proc
mount none /tmp -t ramfs
export FAILSAFE=true
exec /sbin/init
```

Build a new image by typing make in the buildroot directory, install the modified firmware and boot the device. This forces your device to boot in FAILSAFE every time. So in order to boot in normal mode, you'll have to undo the changes you've made to the preinit file.

ASUS WL-500G units seem to respond only on the WAN port when booted in failsafe mode. ASUS WL-300G responds only on the LAN port in failsafe mode.

2. Resetting to defaults

( resetting the NVRAM is not a good idea on every model, for example wl500g (and the wr850g) bootloader will not recreate default values, avoid deleting nvram)

If you're having trouble setting up some feature of your router (wireless, lan ports, etc) and for some reason all of the documentation here just isn't working for you, it's sometimes best to start from scratch with a default configuration. Sometimes the various firmwares you try will add conflicting settings to NVRAM that will need to be flushed. Erasing NVRAM ensures there aren't any errant settings confusing your poor confused router. Run this command to restore your NVRAM to defaults:

```
mtd erase nvram
reboot
```

This will clear out the NVRAM partition and reboot the router, the bootloader will create a new NVRAM partition with default settings after the reboot. Remember to set boot_wait back on after you reboot your router -- trying to do it before rebooting will just write your old settings (cached in memory) back to the flash.

To reset changes you've made to the OpenWRT filesystem, run

```
firstboot
```

If firstboot is run while the jffs2 filesystem is mounted (eg. non-failsafe mode) it will skip formatting and only reset changed files to their defaults. (Files are overwritten with symlinks to their original copy in /rom; extra files and packages are left intact)

After these two steps, you'll have a router with a pristine unchanged configuration. Everything should work now.

3. Recovering from bad firmware

3.1. Software based method

If you've followed the instructions and warnings you should have boot_wait set to on. With boot_wait on, each time the router boots you will have roughly 3 seconds to send a new firmware using tftp. Use a standard tftp client to send the firmware in binary mode to 192.168.1.1. Due to limitations in the bootloader, this firmware will have to be under 3MB in size.

See OpenWrtDocs/Installing
"3.2. Using boot_wait to upload the firmware"

 The Motorola WR850G may wait for an image on 192.168.10.1

 the ASUS WL-300G does not have a DMZ led : press the reset button for 2 seconds just after the AIR led lights, or maybe the LAN led. At some point it works, anyway.

3.2. JTAG-adaptor Method

you are now leaving the safe grounds of warranty coverage

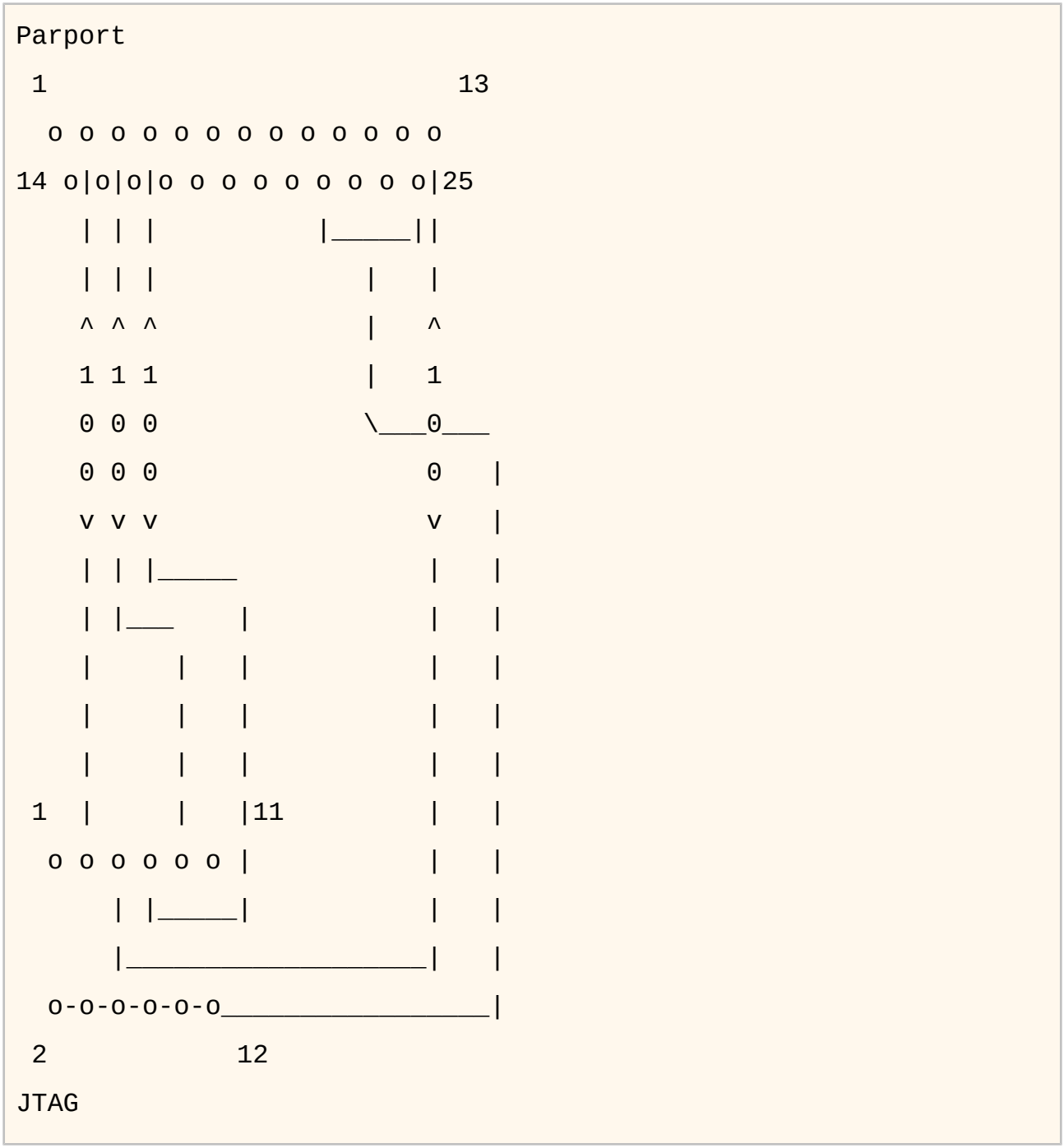
You still don't want to short any pins on your precious router. Thats nasty disgusting behaviour. A lot better way to get a Flash into your wrecked piece of hardware, is to build your own JTAG-adaptor. It's easy, you can make it in a jiffy using spare parts from the bottom of your messy drawer. You need:

1. 4 100R resistors
2. 1 male SUB-D 25 plug
3. If you want to do it right, a 12-way IDC-Connector plug (these are the ones who look like the HDD-Cables)
4. A 12-way ribbon cable for above
5. The boyfriend of that IDC-Connector for the PCB
6. HairyDairyMaids "zip-file" with a debrick utility and instructions how to connect everything together (mirror).
7. A Linksys WRT54G with a broken flash and the desperate feeling that you can't make it

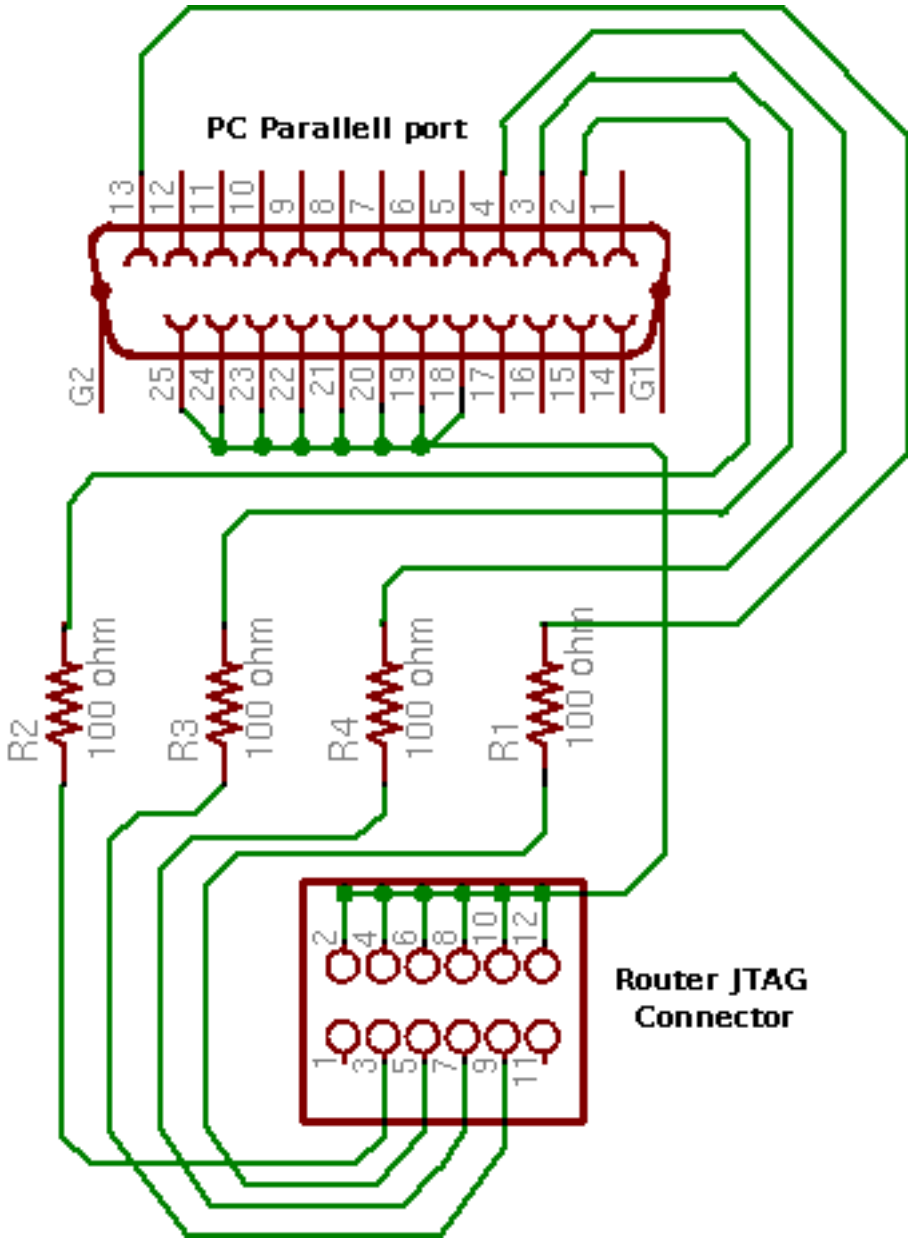
any worse.

It is basically like this:

NOTE: The diagram below is as if you were looking at your computer's parallel port head on. If you are going to solder directly to a male connector, pay close attention to the pin numbers as they will be in a different orientation on the male connector. When looking at the back of the male connector (where you solder wires to) pin 13 is on the far left, while 1 is on the right.



Or a more modern version if you prefer:



Use the pin numbers on the parallel port connector, and the pin numbers on the WRT pcb, as they are all correct. Note: Pin 12 is assumed to be grounded. If it is not grounded on your WRT, you may safely connect the wire indicated on Pin 12 to any grounded even-numbered pin on the WRT's JTAG connector.

Oh, and by the way, this cable is a good thing to have anyway, because many embedded devices feature that JTAG-interface e.g. HP's IPAQ has one as well, so if you dare to open it, you can do lots of "funky things with your IPAQ"

Openwince/JTAG calls this cable as "Xilinx DLC5 JTAG Parallel Cable III" but since this variant isn't buffered, the length of this cable must not exceed 15..20cm.

Since the JTAG adaptor gives you full access to your Flash, I wonder if that nasty thing about shorting pins shouldn't be removed altogether.

Note: I had to enable ppdev in the kernel to use the program by hairydairymaid with linux. Working versions of the CFE can be found at http://downloads.openwrt.org/people/inh/cfe/, information about changing the CFE are available at http://wiki.openwrt.org/OpenWrtDocs/Customizing.

Note2: I had to disable i2c-parport support in my kernel - because i always got the kernel message "all devices in use" when trying to access the parport.

3.3. Shorting Pins Method

If you didn't set boot_wait and don't build a JTAG, you'll have to resort to opening the router and shorting pins on the flash chip to recover.

4M flash chip (WRT54G v1.0, v1.1, v2.0)	Use pins 15&16
4M flash chip (WRT54G v2.2)	Use pins 16&17
4M flash chip (Buffalo WBR2-G54)	Use pins 16&17
4M flash chip (Motorola WR850Gv2)	Use pins 5&6
8M flash chip (WRT54GS v1.0, v1.1)	Use pins 5&6

Be very careful with the flash chip, short only the pins shown in the instructions and do not bend or break any pins; shorting the wrong pins can cause serious damage.

Open the router and locate the flash chip, while the router is off use a straight pin or small screwdriver to connect the pins shown and plug in the router. The bootloader will be unable to load the firmware and instead it will run a tftp server on 192.168.1.1 as described above. On a WRT54G/WRT54GS the power led will be flashing (diag led on a WRT54G v1.0) and all other leds will be normal, when you see this led pattern you can stop shorting the pins and tftp a firmware to 192.168.1.1.

See http://voidmain.is-a-geek.net/redhat/wrt54g_revival.html

Note1: With my 1.1 wrt54g device, there was no way to make it work with atftp, tftp or even windows tftp.. I was about to trash the device when I managed to put back linksys official firmware using the short pin and the official uploader tool and then puted back the openwrt using the administration web upgrade tool.. Ouf!

Note2: Observed on a WRT54GS V1.0: it is *sometimes* necessary to hit the reset button **AFTER** having shorted the pins and letting the lights come to their steady-state as mentioned above. This has been observed multiple times by at least two OpenWRT users, but no obvious pattern has emerged as to why it sometimes works as advertised above, while other times requires the reset button to be hit. If you're stuck here, it can't hurt to try this.

 The Motorola WR850G may wait for an image on 192.168.10.1

What the hell does shorting the pins do / how do you know what pins?

The pins listed are address lines, if you grab the datasheet for any of the flash chips they'll be shown as a0, a1, a2 ...

Each address line represents 1 bit -- Suppose you wanted the 12th byte off the chip, 12 translates to 1100 in binary which means you'd need 4 address lines and they'd be set on or off (voltage, no voltage) depending on if the bit is 1 or 0.

If you short the pins, that changes the address the chip sees as requested. Continuing with the earlier example, suppose of those 4 address lines, the middle two were shorted:

-XX-

The requested address, 1100 gets seen as 1110; a request for address 12 got turned into a request for address 14. Likewise 3 (0011) becomes 7 (0111), 4 (0100) becomes 6 (0110) .. etc.

Result: It's actually impossible to read the value at 12 in this case, and it's likely that address 14 holds a different value. If this were a firmware, the bootloader would attempt to verify the firmware on bootup with a CRC check, mangling the addresses would change the data read and the CRC wouldn't match.

In the end, there's nothing really magical about pins 15-16; you can pick any address lines and short them and *something* will happen; if you didn't short the addresses of the bootloader there's a good chance it'll boot up and wait for a firmware.

4. Using the system logs for additional troubleshooting

Modern versions of OpenWRT use S10boot to start a syslogd. If a daemon is misbehaving and you can't figure out why use the *logread* tool to access the messages sent to syslog. Often the solution makes itself evident.

5. Some routers have screws

At least Linksys WRT54GS v2.0 and Linksys WAG54G have screws hidden under the two front feet!

If you're having trouble popping open your router to get at the internals, it's probably because there are screws hidden under the the two front feet in the blue part of the case. DO NOT apply extra force to open these models without checking for the prescence of screws!

Gently use your nails or a flat object to pry all the edges of the front feet up, then simply remove them. The feet are plugs, not just a thin rubber covering, so careful removal will not harm the feet.

From there you will have access to two small Phillips-head screws. Remove and enjoy.

6. Problems going from jffs2 to squashfs or problems booting after reflashing

Important note! This section assumes you have taken care of backup - follow this procedure without backing up properly first, and your jffs2 files are gone!

There are only two times when the jffs2 partition gets formatted:

- If you flash to a jffs2 firmware, the jffs2 partition is always formatted the first time the device boots (hence the extra reboot)

- If you use squashfs and /sbin/mount_root is unable to pivot the root to the jffs2 filesystem

In all other instances (with the exception of failsafe), OpenWrt will assume that the jffs2 partition is valid and attempt to use it. This creates a problem when either the filesystem layout changes and the jffs2 symlinks are invalid, or when the jffs2 partition has been overwritten due to a larger firmware.

There's two ways to avoid the above issue:

- If you haven't yet reflashed, reflash using the command "mtd -e linux -r write openwrt-xxxx.trx linux". The "-e linux" tells mtd to erase any existing data; OpenWrt will be unable to find a jffs2 partition at bootup and the firstboot script will be called to create a jffs2 partition.
- If you have reflashed with squashfs and the device is unbootable then what's happened is OpenWrt has detected the jffs2 partition and attempted to bootit and crashed. Booting into failsafe mode will allow you into the device where you can run "firstboot" manually.

7. Problems accessing the wireless interface

After clearing you nvram and rebooting, the wireless interface has disappeared. In fact, the wl.o kernel module does not load anymore, due to the lack of some nvram variables, and you will find this message in your log :

```
eth%d: 3.90.37.0 driver failed with code 23
```

If you have a WRT54GS v1.1, you may try to add the following variables :

```
wl0id=0x4320
wl0gpio2=0
wl0gpio3=0
```

and try to load the wl.o module :

```
insmod wl.o
```

Another way to fix this problem should be to flash a "working" linksys firmware, configure your router and revert to openwrt.

8. Source port mismatch with atftp

If you get 'source port mismatch, check bypassedtimeout: retrying...' error when trying to upload firmware, there is probably something wrong with your arp table. First try clearing it by using 'arp -d 192.168.1.1' and retry. You can check which mac address your computer sees with 'arp -a'. If clearing didn't help, you can also try setting MAC address (under MAC address clone in basic setup) to mac address that your computer sees. Upload should work afterwards. I had this problem with wrt54gs.

9. Getting help

Still stuck? see http://openwrt.org/support for information on where to get help.

Almost all of these pages are editable, create an account and click the edit (🗨) button at the top of the page.

OpenWrt OpenWrtDocs/Deinstalling

OpenWrtDocs

1. How I remove OpenWrt ?
2. Can I use TFTP to go back to the original firmware ?

1. How I remove OpenWrt ?

If you are unhappy with our distribution you can go back to the original firmware you have used before.

For that please use the following commands:

```
cd /tmp
wget http://www.example.org/original.trx
mtd -e linux -r write original.trx linux
```

Replace the file "original.trx" with the filename of the firmware you like to install. "linux" is the partition you need to write. -e linux will remove old stuff, before writing the new image. -r will reboot the router after successfully writing the file.

If you only have a .BIN formatted firmware file, this is not a problem, simply cut of the header before using the commands above:

```
dd bs=32 skip=1 if=original.bin of=original.trx
```

2. Can I use TFTP to go back to the original firmware ?

Sure, the other way you can use is the TFTP method.

 Note: On many versions, the TFTP server is only able to accept smaller firmwares at bootup. Firmwares that are over approximately 3M will fail with a "no space" error. If this happens you'll need to either use a different firmware or reflash using the mtd commands above instead.

Almost all of these pages are editable, create an account and click the edit () button at the top of the page.

OpenWrt Glossary

1. afterburner

Also known as speedbooster, this is the broadcom enhancement that claims faster 802.11 speeds. On most routers it breaks WDS.

2. failsafe

A special recovery mode in OpenWrt triggered by holding the reset button for ~2 seconds immediately after the DMZ led lights durring boot. While in failsafe OpenWrt will ignore files in jffs2 and will override the lan ip address with 192.168.1.1

3. firstboot

The firstboot run to initialize the jffs2 partition. It is normally run automatically the first time OpenWrt is booted, but can be run manually to reset the jffs2 partition.

4. jffs2

This is the name of the writable filesystem used as the root in OpenWrt.

5. nvram

The term nvram is used to both describe the partition where configuration is stored and the utility used to access the configuration.

6. WRT

Generic term used to describe the WRT54G/WRT54GS series of routers.
(wrt is also the name of the bot used on the IRC channel to display news)

7. SpeedBooster

See afterburner.

8. squashfs

A readonly filesystem normally mounted as /rom.

Almost all of these pages are editable, create an account and click the edit () button at the top of the page.