WRT54G EJTAG DeBrick Guide
by
HairyDairyMaid (a.k.a. lightbulb)

hairydairymaid@yahoo.com

Ok – Let's just be clear on a couple things right up front:

1) If you use EJTAG or any form of JTAG to recover (debrick) your WRT54G unit you do so fully ***at your own risk***.
2) I (nor Linksys) will not/can not be held accountable for anything you screw up using this (including your router). [If you want to complain to Broadcom for not releasing chip specs – be my guest.]
3) This ***will void your warrantee***. (Of course – if you already popped the lid off your router it is already voided anyhow)
4) I *hate* writing documentation – so this is it – take it or leave it! (i.e. – don't ask for another guide)

Introduction

I guess if you are reading this you probably either have a bricked WRT54G v2 router or you want the capability to be able to re-flash your Intel Flash chip on your router in an alternate manner in case the normal method(s) do not work.

Well I unfortunately found myself exactly in that spot… trying to upgrade the firmware and it crashed hard and nothing could revive it. So I decided that I must not be alone in this and that there may be a need for JTAG recovery of the flash chip. I started into this project and quickly found Broadcom was about the worst chipmaker out there at releasing info about their chips unless you signed a NDA

and swore to never mention anything to anyone about anything Broadcom related. (Can you tell I don't care for their business stance? Their chips are ok – but their proprietary nature is terrible). [I feel better now]

Anyhow – since Broadcom was unwilling/uncooperative in releasing the Boundary Scan Definition Language (BSDL) file and/or chip specs for the bcm4712 chip there was only one other alternative I knew of and that was EJTAG. This would use the same JTAG port that might be used for JTAG Boundary Scan. Fortunately, Broadcom did add EJTAG 2.0 support to the bcm4712 chip used in the WRT54G v2 router. This is what I decided to use. I did not, however, support everything that EJTAG has to offer.

Since the intent of this project was ONLY to be able to recover/reflash a bad flash (any or all portions of it) or make backups of the flash chip (any or all portions of it) – I only wrote the "WRT54G EJTAG DeBrick Utility" using the bare EJTAG minimums to talk to the flash chip by using DMA routines over EJTAG. Ok with that said let's move on…

Requirements

In order to debrick your WRT54G v2 router's flash using EJTAG/JTAG you need really only two things:

1) A parallel based JTAG cable. I built mine for nothing with spare parts but you could also purchase a Xilinx III parallel JTAG cable or possibly a Wiggler style parallel JTAG cable.
2) Software to do the communications and talk to the flash chip. I wrote my own and am releasing it under the GPL.

Building a JTAG cable

The JTAG cable can actually be of a few different types; however, I decided on a Xilinx parallel type cable since those are very easy to construct for little or no money. You could also use a Wiggler style parallel cable. Since I made the Xilinx cable myself that is what I am going to describe:

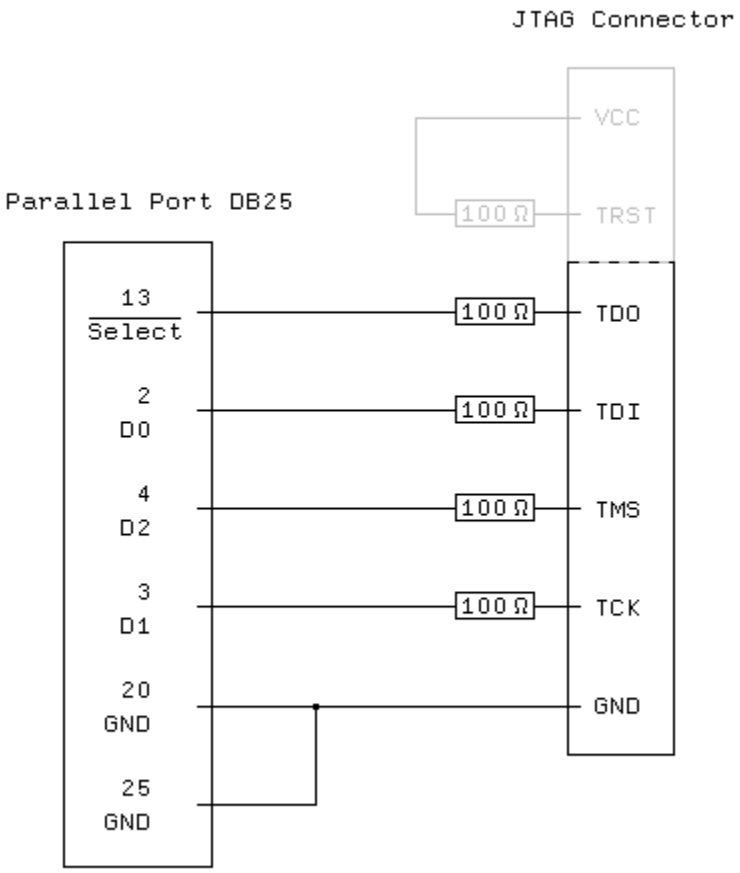
Parts needed:

1) One DB25 parallel connector shell with solder pads
2) Four 100-ohm resistors

3) One old 12 or 14-pin ribbon cable with a connector on the end (from an old PC card) or you can use your own wire and solder your own ends.
4) One 12 pin male header (to solder to the board)

I am not going to write a real detailed explanation as I think pictures will make things much easier.

Here is a simple schematic of how to wire up the parallel connector end and what signals they map to:



Note:
You don't need the VCC and TRST pins on the interface: The TRST is already pulled high on the WRT54G, that's why there is no (need for a) TRST pin on the board. But you might want to build a generic JTAG cable that works with other devices too (as I did – but not needed), even if they actually have a TRST pin. Since this is only a passive cable, make sure that it is not too long. 75cm worked for me.

I was told that this kind of cable may not work with all parallel ports out there, so it is probably better (but a little more work) to build an active cable. You can find

links to schematics for various types of cables on the JTAG-Tools website. However, the passive cable worked fine for me and is probably sufficient for most people.

On the other end of the cable:

CONFIGURING THE EJTAG CONNECTION

The VisionIce JTAG connector has fourteen (14) pins whereas the EJTAG block (J10) has only twelve (12). A twelve-pin jumper block may be needed to mount the VisionIce connector onto the BCM94710AP. Note that pins 13-14 of the VisionIce adapter remain unconnected. The pinout for the EJTAG block (J10) is described in the following table.

Table 8: EJTAG Connections

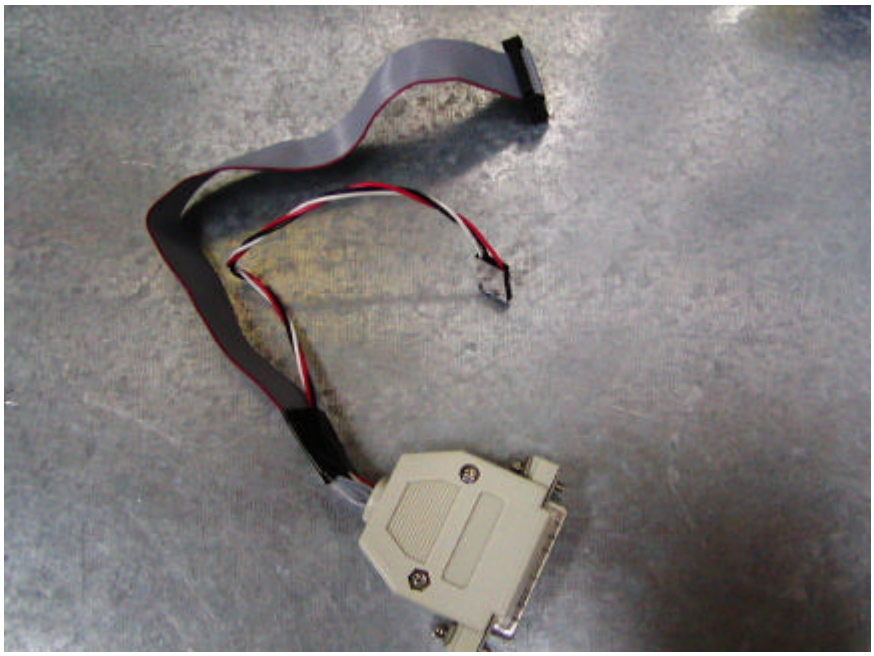
J10 Pin number	*EJTAG SIGNAL*
1	JTAG_TRST_L
3	JTAG_TDI
5	JTAG_TDO
7	JTAG_TMS
9	JTAG_TCK
11	SW1 RESET[a]
2, 4, 6, 8, 10, 12	GROUND

a. Pin 11 is connected to the hardware reset switch (SW1) of the BCM94710AP.

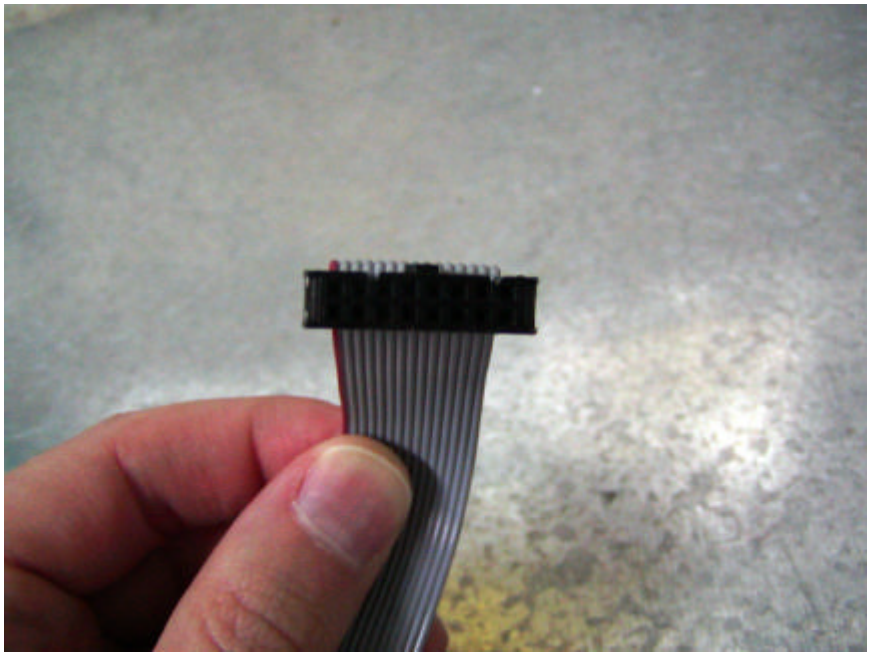
(This was taken out of the Broadcom BCM47XX Reference Guide).

Don't worry about connecting pin 1 (TRST) or pin 11 (RESET) as you can get by without them. I did connect them but don't use them for the WRT54G v2.

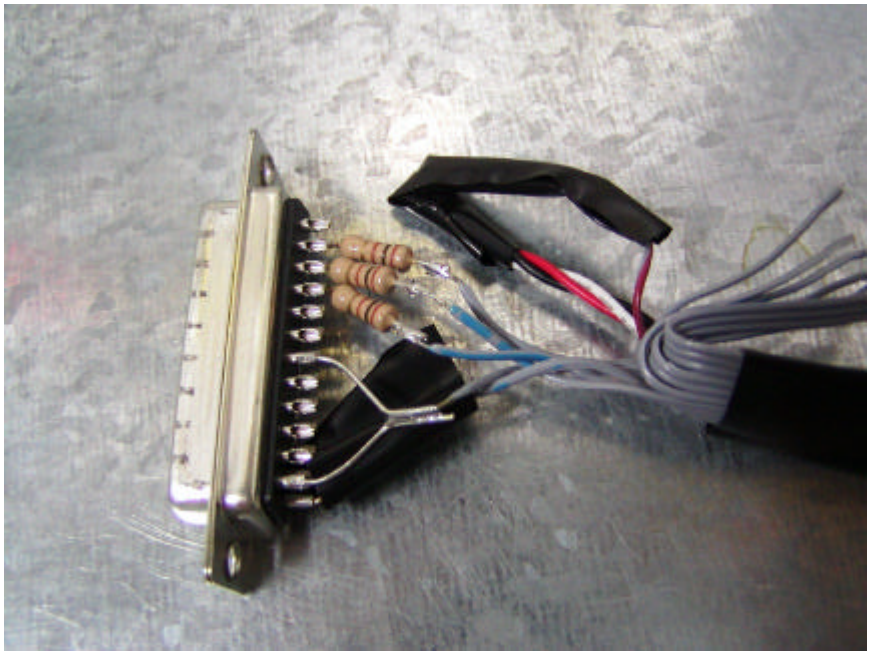
Here are a few pictures of my personally constructed Xilinx Parallel JTAG cable:



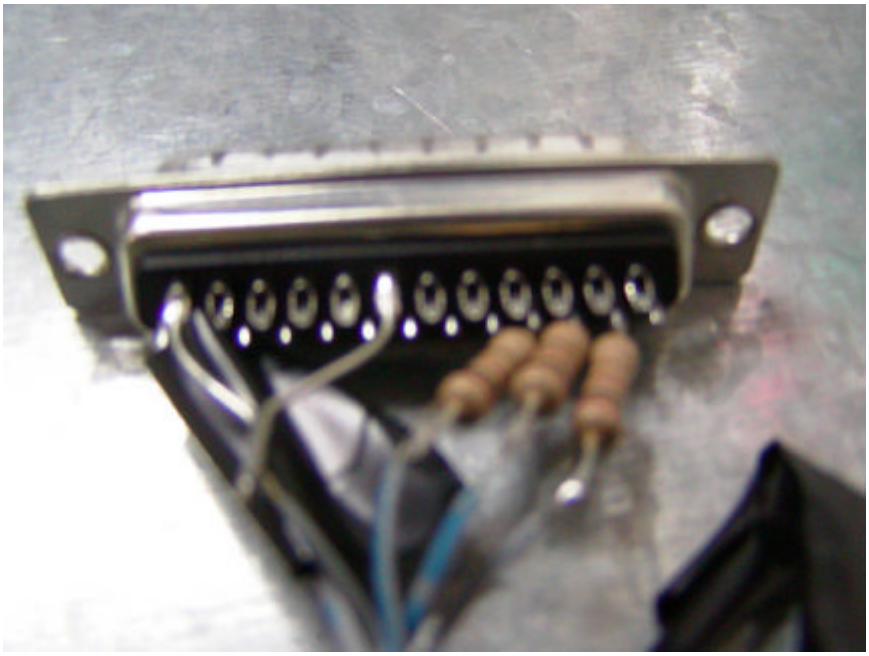
The Finished Cable (the small wire isn't really needed)



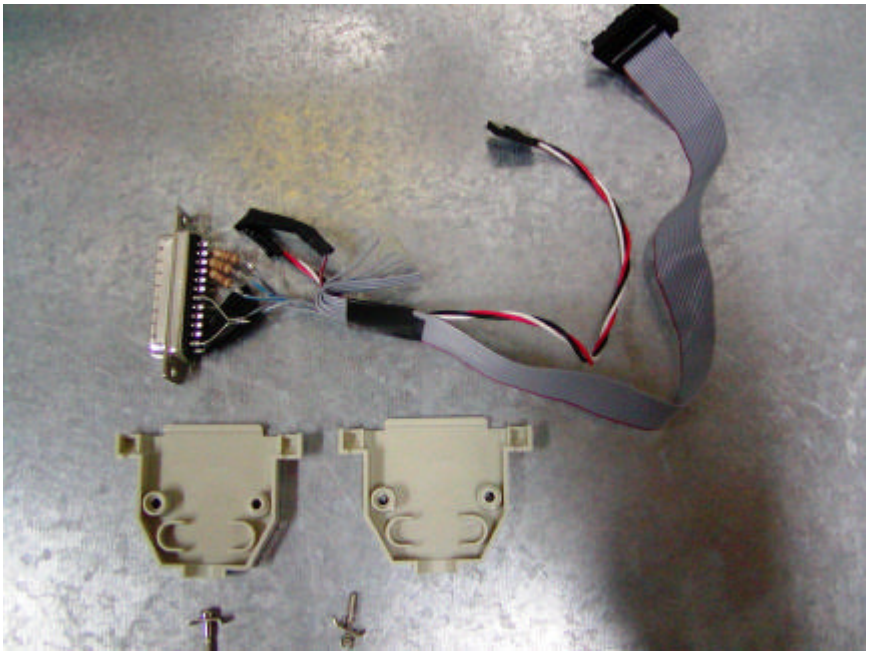
The JTAG connector end (old cable from a PC card)



The DB25 solder end (note the easy/cheap
Way I connected the 100-ohm resistors to the
Connector directly)



Another Picture (kinda blurry – sorry)

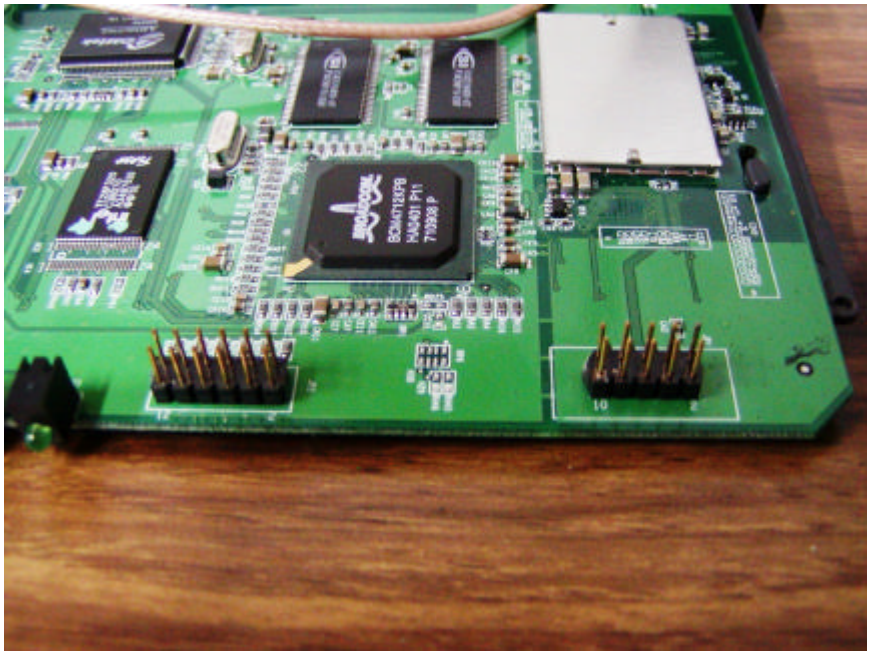


Just Prior to screwing the DB25 case on

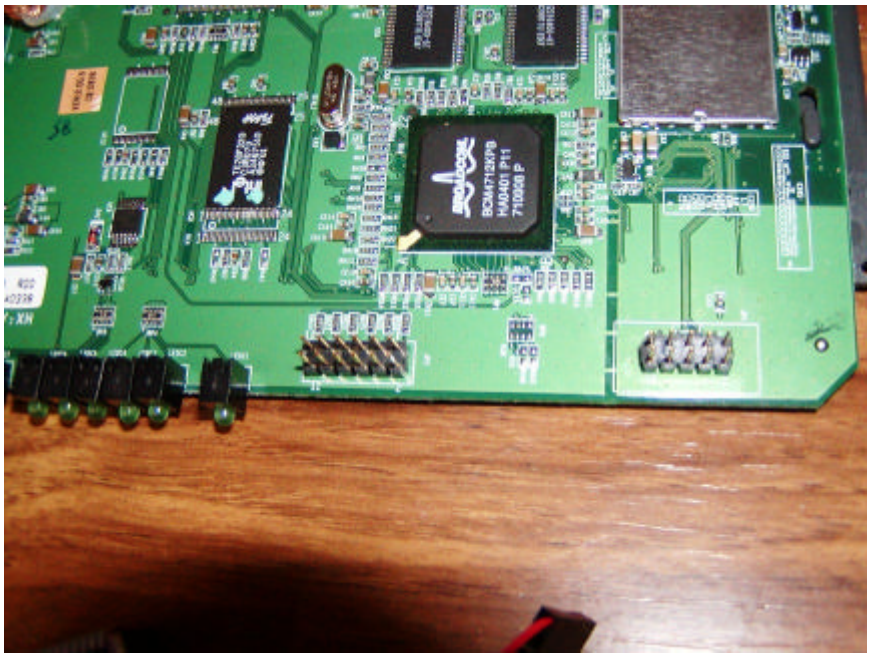
Ok – that should be sufficient… (I *hate* writing documentation)

Once you have the cable assembled you will need to solder a 12-pin header on to the WRT54G v2 board. Mine had all the holes solder-filled so I first had to

remove that existing solder with a solder sucker. Once that was done – it was easy to mount the 12-pin header and solder in place.

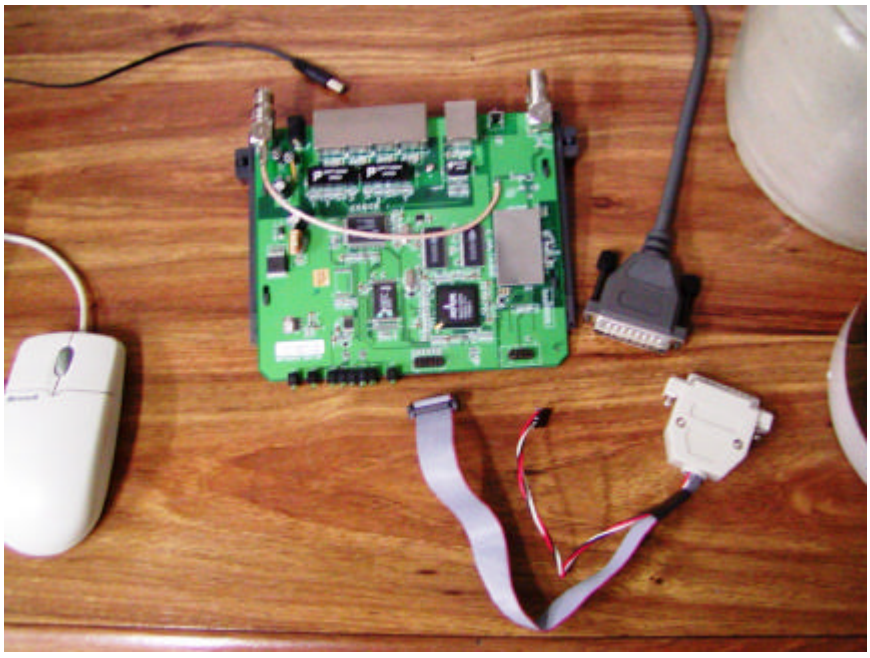


JP2 – is the JTAG header (left)
JP1 – is the serial header (right) (from another project)

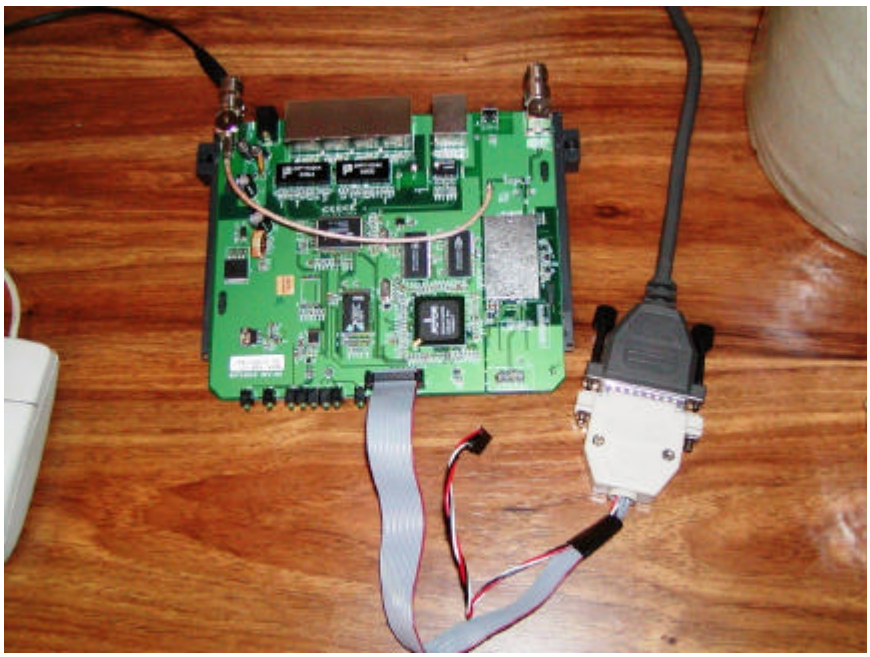


Another picture of the JTAG header (left) soldered in place

Once the cable has been made and the JTAG header is soldered in place you are ready to connect things. Here are a couple sample pictures of mine:



WRT54G v2 & JTAG Cable ready for use



Here is my Xilinx parallel cable hooked up and ready for use on the WRT54G v2 (note: the small 3-wire cable that is not attached carries the TRST signal / VCC which is not needed on the WRT54G v2 and is why it is not connected up).

The Software

Once the cable is made (or purchased) and is ready for use then the software is the last piece. I have written my own software which talks across the JTAG port via EJTAG 2.0 standards.

One thing to note is that JTAG (at least over parallel) is rather slow. This is a very important thing to remember as it could take a very long time to flash the entire flash chip at once or even the kernel image. I recommend flashing the CFE (as needed) and the NVRAM spaces but then use the normal “tftp method” over Ethernet to recover/flash another kernel image.

There are two source files included which are written to compile under Linux:

- wrt54g.c
- wrt54g.h

Compile these as you might any other Linux source. I have included a simple “makefile” as well. Once this is done you are almost ready to use the software.

A couple things first:

1) Make sure the parallel port is “accessible” – i.e. – issue an “rmmod lp” command if needed.
2) Always plug the cable into your parallel port on your PC and to the board with the WRT54G’s power off.
3) Be smart - make a backup before flashing or erasing anything. (don’t come crying to me if you don’t)
4) Again – flashing over parallel JTAG is slow – (yes a good deal slower than flashing over Ethernet) – so don’t get impatient.

To run the software you will type:

./wrt54g <and hit RETURN>

and you will get the following displayed:

```
wrt54g: read/write flash memory via EJTAG
 usage: wrt54g [option]
        -backup:cfe
        -backup:nvram
        -backup:kernel
        -backup:wholeflash
        -erase:cfe
        -erase:nvram
        -erase:kernel
        -erase:wholeflash
        -flash:cfe
        -flash:nvram
        -flash:kernel
        -flash:wholeflash
```

These are the options you can run the program with. Starting the program with any of the options will start it immediately and run until completion (so get the option correct!)

A couple notes here:

- Backing up the Kernel or Wholeflash will take a really long time (I think I already mentioned that – why yes - I believe I did!)
- The image to flash must reside in the same directory as the program
- The image to flash must be named one of the following: ***CFE.BIN*** or ***NVRAM.BIN*** or ***KERNEL.BIN*** or ***WHOLEFLASH.BIN***
- Anytime you backup an image the image is saved with the name ***CFE.BIN.SAVED*** or ***NVRAM.BIN.SAVED*** or ***KERNEL.BIN.SAVED*** or ***WHOLEFLASH.BIN.SAVED***
- Anytime you flash a portion of the flash chip using this utility – it will first erase that portion of the chip to flash.
- Issuing a Flash command will *not* automatically backup what is there first. That is up to you to first issue a backup command.

> ***Take the time to make a backup of each section of the flash before doing anything else. (This is only smart in case you roast things later – and you may want to put those backups in a safe location. I believe if I recall correctly the CFE.BIN.SAVED image (the CFE portion of the flash) contains a MAC Address embedded specific to the router.)***

Ok – now that you have seen the options you need to know one thing first before running the program for real…

You need to type the requested command line option in completely and just before hitting <ENTER> plug in the power cable to the WRT54G. In other words – have the JTAG cable hooked up to both the PC and router with the router's power off and then type the command line you wish and plug in the router and hit <ENTER>. The command should start working and progress will be seen on screen.

*** IMPORTANT NOTE ***

Anytime you re-run the program, follow the above step – it is important since the WRT54G v2 has a Watchdog Timer built into it that will reset things at a very

inappropriate time in the flash process if it cannot be disabled quickly by the software.

One last comment – if you erase the NVRAM portion of the flash – many times that is all that is needed to un-hose the flash. Try this first. Also – if the CFE and KERNEL is intact then it will post a refreshed copy of the NVRAM to the NVRAM space once it is cleared (on the next boot).

It is a very good idea to re-cycle the power to the unit between operations (i.e. – backup/erase/flash) and doing the step mentioned above about running the program quickly after plugging in the router.

Ok – I am really over writing documentation now – so I hope this has been helpful! Good luck on debricking your busted WRT54G v2 router.

Enjoy!
- hairydairymaid (a.k.a. lightbulb)