English • Deutsch • Español • Français • Italiano • ??? • Polski • Português • ??????? • Svenska • ???(????)? • ???(??)? •

SD/MMC Modification for the Buffalo WHR-G54S and WHR-HP-G54 Wireless Router

Update: 2007-11
The prior edition of this tutorial was attempted, by at least two users, specifically on WHR-HP-G54's. Of a variety of SD and MMC cards, none worked. Since then, the missing tags in the tutorial have been added and it works! Have Fun! Mega-shouts to Iron for an extensive tutorial.

Contents

Introduction

This tutorial guides you through adding a SD or MMC interface to the Buffalo WHR-G54S and Buffalo WHR-HP-G54 router and gives some general information and tips to help you to do the same with other DD-WRT routers. The DD-WRT V2.4 firmware supports this modification, so there is no need to install modules or packages. All configurations can be done from the web interface.

The modification will allow you to add non-volatile memory to your router. I recommend using a SD card up to 1GB. Some forum users reported that they got 2GB working, but both my 2GB cards failed to work. This might be solved in future with an updated MMC driver, but probably my SD cards are to blame. The problem is that there are quite a number of incompatible 2GB cards around.

This modification might also work with MMC cards, but the used communication protocol(SPI) is optional for MMC cards. It is only required for SD and mini-SD cards. **So start with a SD card of 1GB or less to confirm that everything is working.** 1GB does perhaps not sound like a lot, but for this application it is huge!

So what can you do with this added storage capacity:

- Store your own custom programs, scripts and packages (standard or optware)
- Store communication and packet logs
- Store your e-mail database for your e-mail server
- Store your files for your web server
- Store your files for your ftp server
- Store all /jffs content on the card, instead of a network share
- Provide (very slow) swap space

There is another way to add storage capacity to your router, and that is by mounting a shared directory on your computer. DD-WRT supports Samba shares, which is the default Windows sharing mechanism. Samba shares can also be made with Linux. Mounting a shared directory will make the available hard disk space accessible to your router.

Compared to the SD/MMC modification using Samba this has some disadvantages:

- Samba only works over a wired connection (so a wire between the PC and router is required)
- Your PC needs to be powered up for the storage capacity to be available

You can read more about how to mount a directory with Samba here: The Samba Filesystem

Note: The Samba tutorial is outdated. I will add a section to it to show you how to mount a Samba share with the V2.4 firmwares.

The modification is not difficult to implement, but some decent soldering skills are required and knowing how to operate a multi-meter would be very handy.

The SD card

Let's have a look at the requirements to access an SD card via the simplest available protocol: SPI (Serial Peripheral Interface).

SD Pin	SD function(SPI Mode)	Direction	The SD Card pin assignment
1	Chip Select (CS) *	IN	
2	Data In (DI)	IN	
3	Ground	-	
4	Vcc (3.3v)	-	
5	Clock (CLK)	IN	
6	Ground	-	
7	Data Out (DO)	OUT	
8	Reserved	-	
9	Reserved	-	

(*) A *low* level on "Chip Select" selects the chip

From this we can see what is needed:

- Ground
- Power, 3.3 volt
- 3 outputs (that are going to control the 3 inputs of the SD card)
- 1 input (that is going to read the data from the SD card output)

In fact we do not actually need to actively select the SD card via the Chip Select line. We can just always select it by grounding the signal. So in that case we only need 2 outputs. Now let's get to it and open our router to find what we need.

How to open your router

Check this Wiki How to open the Buffalo WHR-HP-G54 to see how to open a WHR-G54S or WHR-HP-G54 router without breaking it. Inside you will find a Printed Circuit Board (PCB).

Finding suitable IO points

Once you have opened the router you will have access to the PCB. We need to find several points on the board that we are going to use to build our SD/MMC interface. To find these points we will need our general purpose friend, the multi-meter.

Finding Ground

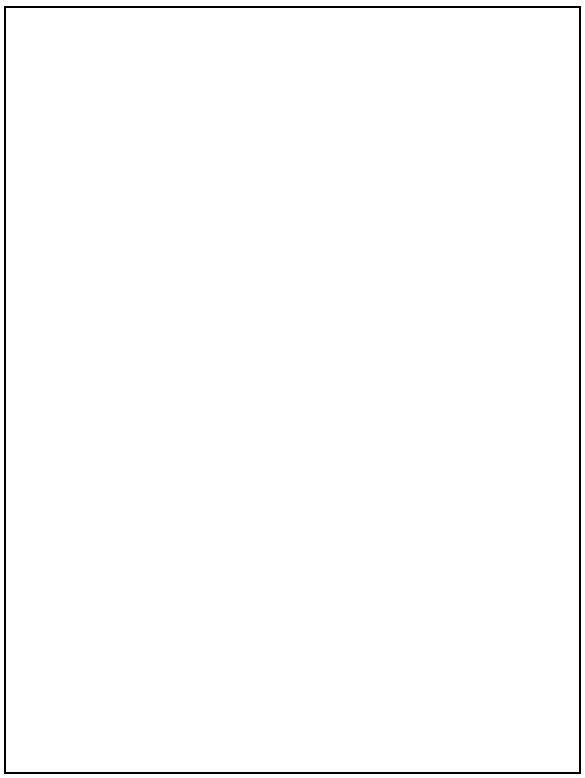
It is very easy to find a grounded point, because ground is present all over the PCB. In most cases the input power ground will also be the ground for the whole PCB. So start looking at the input power connector. This connector has quite big soldering pads, which makes it easy to add an extra wire. Once this point is found, verify it by checking big metal object on the PCB. Usually the metal housing around the transmitter (see pictures below to find it) will be grounded too. To verify that the points are connected use the conductivity check of the multi-meter. Make sure the router is powered down. The connection between the points should give you a very low resistance, about the same resistance as what you would get when just shorting the multi-meter probes. With the Buffalo WHR-G54S I found that the metal mounting pins for the AOSS switch are also grounded(indicated with SD3/6 in the picture below). This gave me an easy ground access point, and is also a nice point for the routing of the wires.

Finding power, 3.3 volt

Once a grounded point is found we can start looking for the 3.3 volt power supply that we need. Check your power supply specification that is written on it. Check if it provides AC or DC at the output. Often AC is indicated with a wavy line: ~. DC is indicated like this: ⎓. Verify the information on the power supply with a multi-meter.

If the power supply output is 3.3 volt DC (WHR-G54S), then check the power socket connection at the PCB for 3.3 volt. To verify this you need to power up the router. Be very careful with this, as it is very easy to short circuit the board with some parts lying around. Switch the multi-meter to DC volt measurement, range 0-20 volt. The actual value that you find might be slightly higher or lower than 3.3 volt(3.1-3.5 volt). If you power supply states an output voltage higher than 3.3 volt (WHR-HP-G54, +5v, others usually 5 or 12 volts) then the PCB contains a power regulator to lower the input voltage to 3.3 volt DC. You will need to find this power converter, and it's 3.3 volt DC output. The power converter is a big component, usually close to the input power socket. Power up the router and connect one multimeter probe to a grounded point. Then carefully use the other probe to check the voltage at big pads of the big components close to the power socket until you find the 3.3 volt DC power supply. Below you can see what is printed on a WHR-G54S power supply. It states that it delivers 3.3 volts DC, and maximal 2 amps. How convenient!

Image of a WHR-G54S Power Supply (a WHR-HP-G54 is different, supplying +5v).



Finding general IO output points

To control the SD card we need some outputs, which are connected to the inputs of the SD card. Routers usually contain LED's to signal their status. This gives us a easy opportunity to find some output points. The only disadvantage of this is that we will lose the signalling functionality of the LED's because they are now used for the SD card communication. What we first should establish is which LED's we can control by software. To do this login to the router and use the "gpio" command. So under Windows type in your dos box:

```
telnet <router IP address>
```

The default IP address is 192.168.1.1. Telnet needs to be enabled for this, which it is by default. You can enable or disabled it in the services tab of the web interface.

The login name is: "root", even if you changed the router name. The default password is "admin".

The syntax of the gpio command is like this:

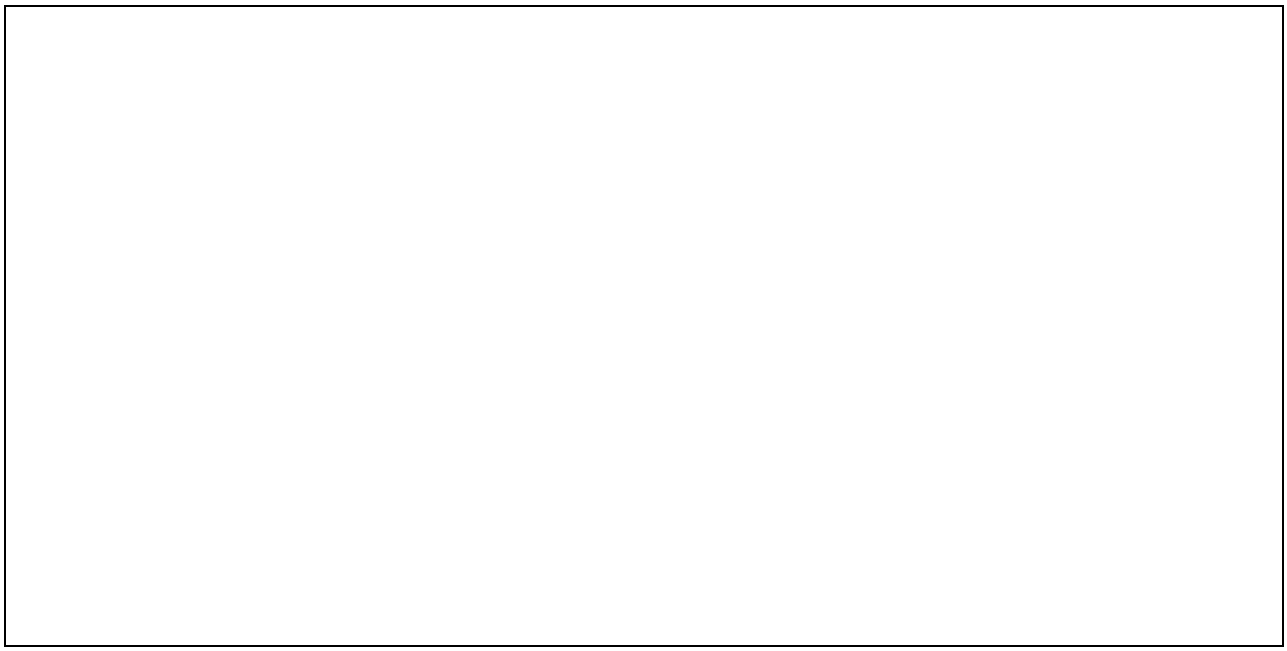
```
gpio enable  <IO pin number>      # to  enable a IO pin, which switches the LED off
gpio disable <IO pin number>      # to disable a IO pin, which switches the LED on
gpio poll    <IO pin number>      # to read the status of a switch
```

For the *<IO pin number>* use numbers between 0 and 14. So watch the LED's closely while giving the gpio commands. You can also use this little command line to make your life easier:

```
while true; do gpio disable 1; sleep 3; gpio enable 1; sleep 3; done
```

It will switch on the LED for 3 seconds, then switch if off for 3 seconds, and so on. You will need to press <CTRL> + "c" to exit from this command.

Replace the "1" in two places to change it for other IO pin numbers. In the screen capture below you can see how it should look.



There is one important thing to consider! The IO pins can be used for input and for output purposes, and as you might know, there is usually a reset switch on a router. So what would happen if you write the status of the pin that is used to see if the reset button was pressed!? Right, you create a reset. You lose your settings as if you pressed the reset button. You will lose the telnet connection and see the LED's flashing. When the router restarts it can happen that the reset button is still "pressed", which will cause the router to reset the NVRAM to factory default values. It should be clear that you cannot use this IO pin for the SD card interface.

Now go through all the IO numbers and record accurately which LED's are software controllable by which IO pin number. In many cases you will not see a LED changing. This means that the IO is not in use, or perhaps is used for input purposes. A WHR-G54S yielded this:

```
GPIO 1 - Bridge LED (Green, 3rd LED from top on front panel
GPIO 2 - WLAN LED (Green)
GPIO 3 - Extra or Missing LED, between bridge and WLAN (Green if present, not visible when case i
GPIO 6 - AOSS LED (Orange, on top)
GPIO 7 - Diag LED (Red)
```

And I found that the reset button is using GPIO 4:

```
GPIO 4 - Reset Button
```

The power LED is not software controllable in my routers. We need 2 or 3 outputs and we 5 found, so let's make some choices. Let's use the extra LED and the AOSS LED, the Bridge LED is optional.

So now we have found some LED's that we can control by software. Each LED has 2 connections, a positive site (anode) and a negative site (cathode). So an important question arises. **Which side of the LED do we need to connect to?**

To answer this we need to use the multi-meter again. Send the command described above to make the LED that needs to be analyzed flash on-off-on. Ground one probe and measure the voltage on each side of the LED. We need to use the side of the LED that that changes between about 0.2 volt when it is on, and 3.3 volt when it is off. The **wrong** side will be at about 2 volts when the LED is on, and 3.3 volt when it is off.

In the Buffalo router one side of the LED is marked with a "+", we will need to use the **other side** of the LED.

Finding a general IO input point

We need to find 1 input. The switches on the router are the candidates for this. The Buffalo WHR-G54S or WHR-HP-G54 routers have 3 switches:

- A reset switch, which is not suitable to use as it has some side effects ;)
- The AOSS switch
- The Auto/Bridged switch

We can use the "gpio" command again, this time to read the IO status:

```
gpio poll <IO pin number>      # to read the status of a switch
```

This command will wait until a change in the switch position is detected and signal this by printing "00" or "01". So execute the command and press all (except reset) switches and check if you see some output on the screen.

We will need to press <CTRL> + "c" to exit from this command. Try this with all IO numbers that were not connected to a LED.

In my case this yielded:

```
GPIO 0 - AOSS button on top (State 00 is down, state 01 is up)
GPIO 5 - Auto/Bridge Switch (State 00 is "bridge", state 01 is "Auto")
```

I already knew that the reset switch was behind GPIO 4. I choose to use the Auto/Bridge switch because this switch is not very useful. With the DD-WRT firmware it not used anyway; the router mode is defined by the software settings, not by the switch.

The switch has several leads going into the PCB. So we need to find out which lead/soldering pad we need to access so the system can read the IO status from the pin. To find this pad we need a multi-meter again. We need to find the soldering pad that is at about 3.3 volt when the switch is in 1 position, and close to 0 volt when it is the other position. See the pictures below to see which solder pad I found.

It is very important to leave the switch in the position where the multi-meter reads 3.3 volt, otherwise the switch is forcing the IO line to a certain level, which makes it unusable. Leave the switch in the "Auto" position.

The wiring layout

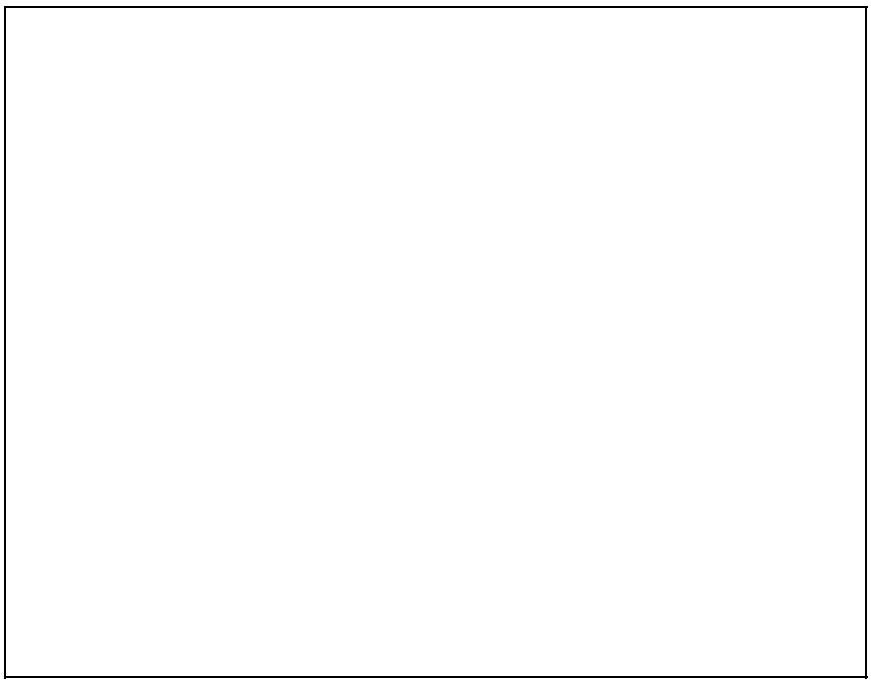
In total this yielded the following wiring layout:

SD CARD			Router			The SD card pin assignment
Pin	Function(SPI Mode)	Direction	IO	Function	Direction	
1	Chip Select (CS) *	IN	Ground (or GPIO 1)	(Bridge LED)	OUT	
2	Data In (DI)	IN	GPIO 6	AOSS LED	OUT	
3	Ground	-	Ground	Ground	-	
4	Vcc (3.3v)	-	Vcc (3.3v)	Vcc (3.3v)	-	
5	Clock (CLK)	IN	GPIO 3	Extra LED	OUT	
6	Ground	-	Ground	Ground	-	
7	Data Out (DO)	OUT	GPIO 5	Bridge/Auto switch	IN	
8	Reserved	-	-	-	-	
9	Reserved	-	-	-	-	
(*) A *low* level on "Chip Select" selects the chip						

The use of GPIO 1 for the "Chip Select" signal is optional. If you don't use it then just ground SD pin 1. You might swap around the LED's any way you like, but make sure to reflect this also in the DD-WRT Gui(see below). I've seen that some people build this modification and connect router outputs to the SD card output. **This is confirmed to work too, but is not recommended!** It might destroy your router and/or your SD card. The reason why it works is because the IO's are general purpose IO's, which means that they can be reconfigured in software to work as an input or output. But what happens during boot up, when the LED's are flashing!? Just don't do this!

miniSD/microSD cards

In comparison with a traditional SD card, the pin layouts for miniSD and microSD cards are shown below.



Differences from SD:

- microSD has only one ground pin (6)
- miniSD has two additional pins (10 and 11)

Note that while both SD and miniSD cards support the SPI protocol, the support in microSD is only optional. Even if you buy a microSD card with an accompanied SD adapter, it may be incompatible.

Implementing the modification

Requirements & Parts

Now it's time to implement the modification. For this we need:

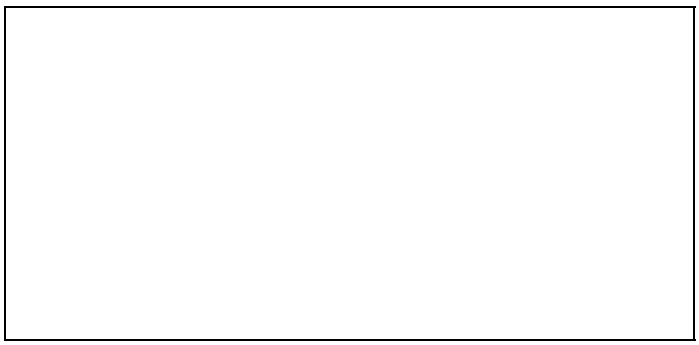
- A multi-meter
- some pieces of thin wire (stranded ethernet wire, or wire diameter about 0.18mm)
- some soldering tin (fine electrical solder)
- and a soldering iron with a small tip. Recommended soldering irons of 15-20 watts, or better irons with a controllable temperature. If only "lead free" solder is available, use 400 degree setting, which is quite high, but needed because the solder used has a higher melting point.
    - ♦ a small piece of natural sponge, wet, to clean the tip of the soldering iron, and any other normal soldering accessories. See these links if you are new to soldering electronics [1] [2].
- Some glue to fixate the wires and parts (I recommend rubber glue)
- An SD card (1GB or less is recommended at first)
- An SD card socket or adapter (see below for some suggestions)

The modification is not difficult to implement, but some decent soldering skills are required and knowing how to operate a multi-meter would be very handy.

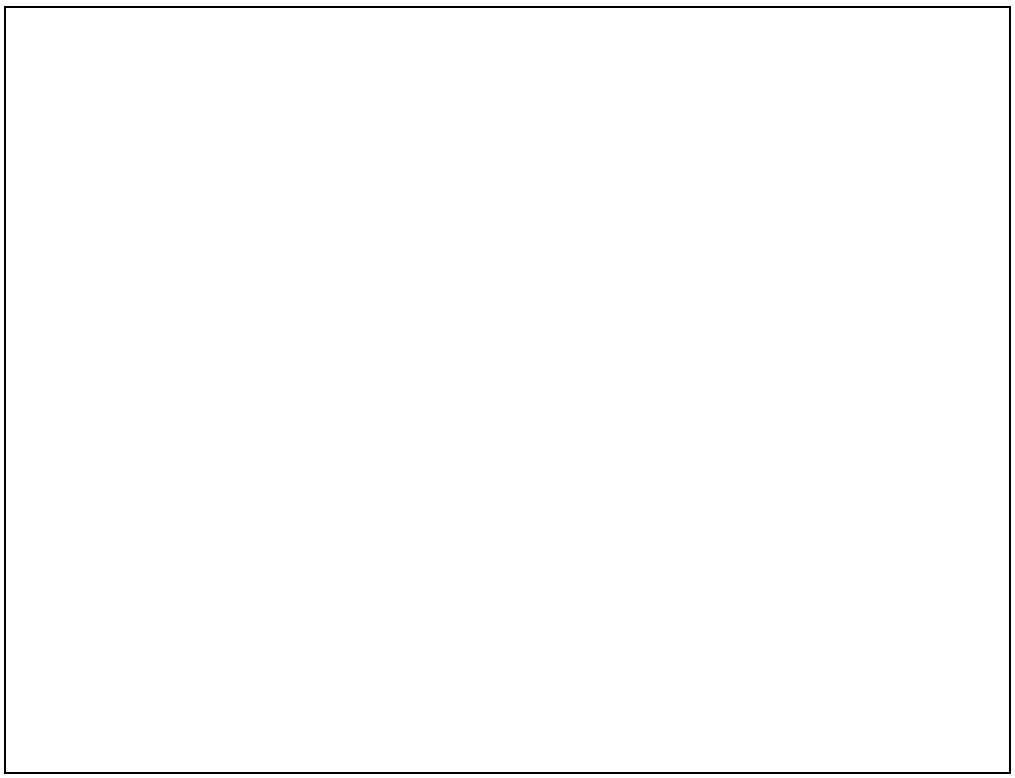
Getting an SD Card socket

It is possible to directly solder the wires to the SD card, but I recommend against this as it reduces flexibility a lot. I recommend getting a SD card adapter. Of course one can buy a socket only, however there are (cheaper) alternatives:

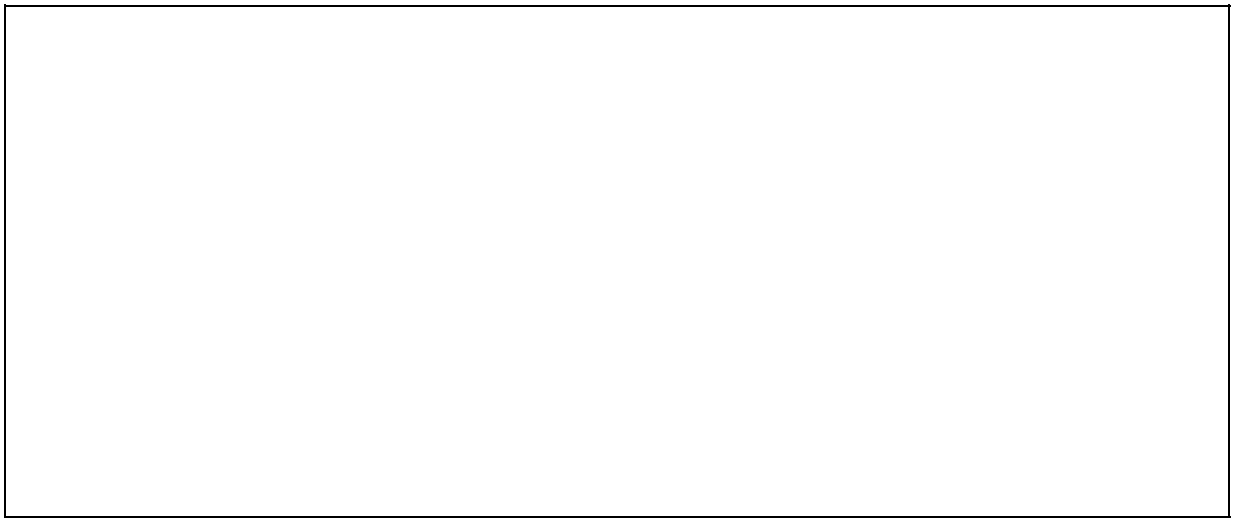
1) Use a mini-SD to SD card adapter. In many cases this adapter comes for free with a mini-SD card, but one sometimes can buy them separately too. The wires can be soldered to the adapter. This is not a very sturdy solution.



2) Disassemble a 'USB-to-SD Card' adapter. These are usually less expensive than buying only the socket. This is what I did. I bought a tiny adapter for 15 rmb (about 1.5 euro or 2 US$) and disassembled it. The adapter is so tiny that I did not need to remove the adapter from the circuit board at all. I just removed all components that were not needed(smd resistors, LED, USB connector, x-tal). The USB-SD card adapter I used is shown below.



3) Use a 5.25 inch floppy disk cable connector. The pitch of this cable is the same as what is used with SD cards. This is quite a bulky solution, but it works ok. I'm not sure if you will be able to close your router when choosing this solution.

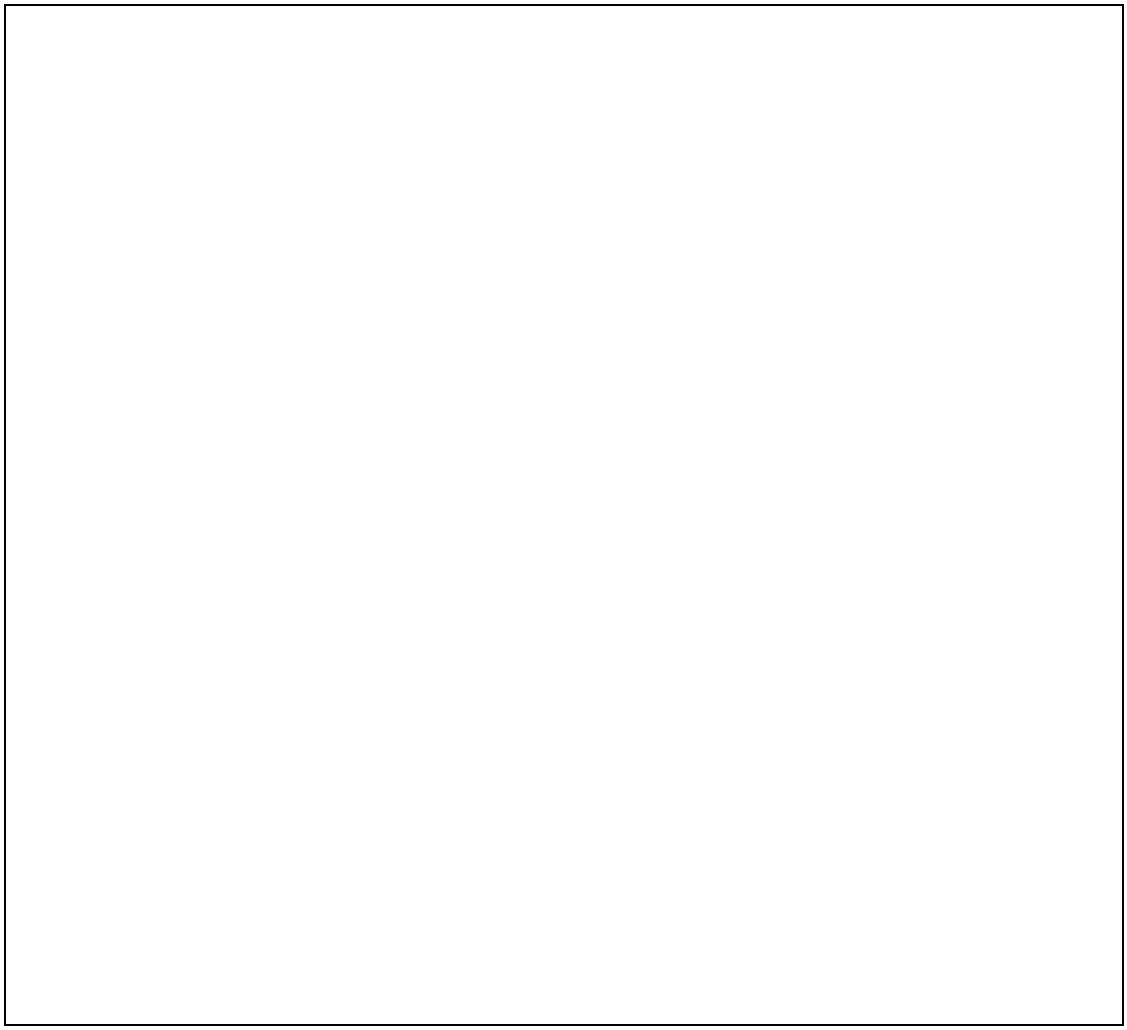


Wiring Points

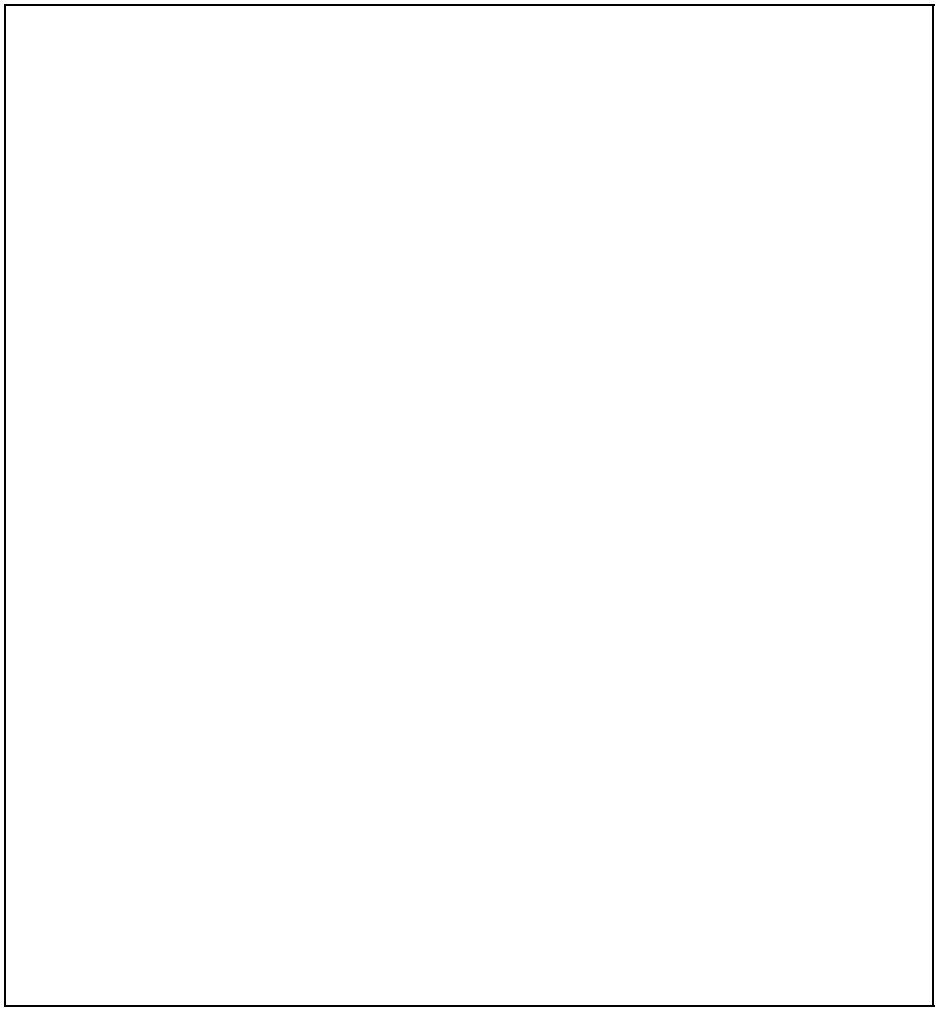
- Note: **The WHR-G54S wiring points are different than the WHR-HP-G54.** The WHR-G54S uses a +3.3v power supply, and it's LED's do not require resistors to drop the voltage. The WHR-HP-G54 uses a +5.0v power supply. The wiring points for the WHR-HP-G54 are **before** the resistor, on the side away from the LED. Refer to the WHR-HP-G54 image below, it is very accurate.

The pictures below show which points need to be soldered to the SD adapter. SD1 refers to SD card pin 1. Be very careful when soldering, especially when soldering the LED's/resistors. They drop off easily if one heats them too long. So go for a short accurate soldering action of about 1 second. If it fails then let the LED/resistor cool down and try again. Once the connection is established you can use some rubber glue to fixate the wires so they don't come of so easily anymore. I also used rubber glue to fixate the SD card adapter to some large components present on the PCB.

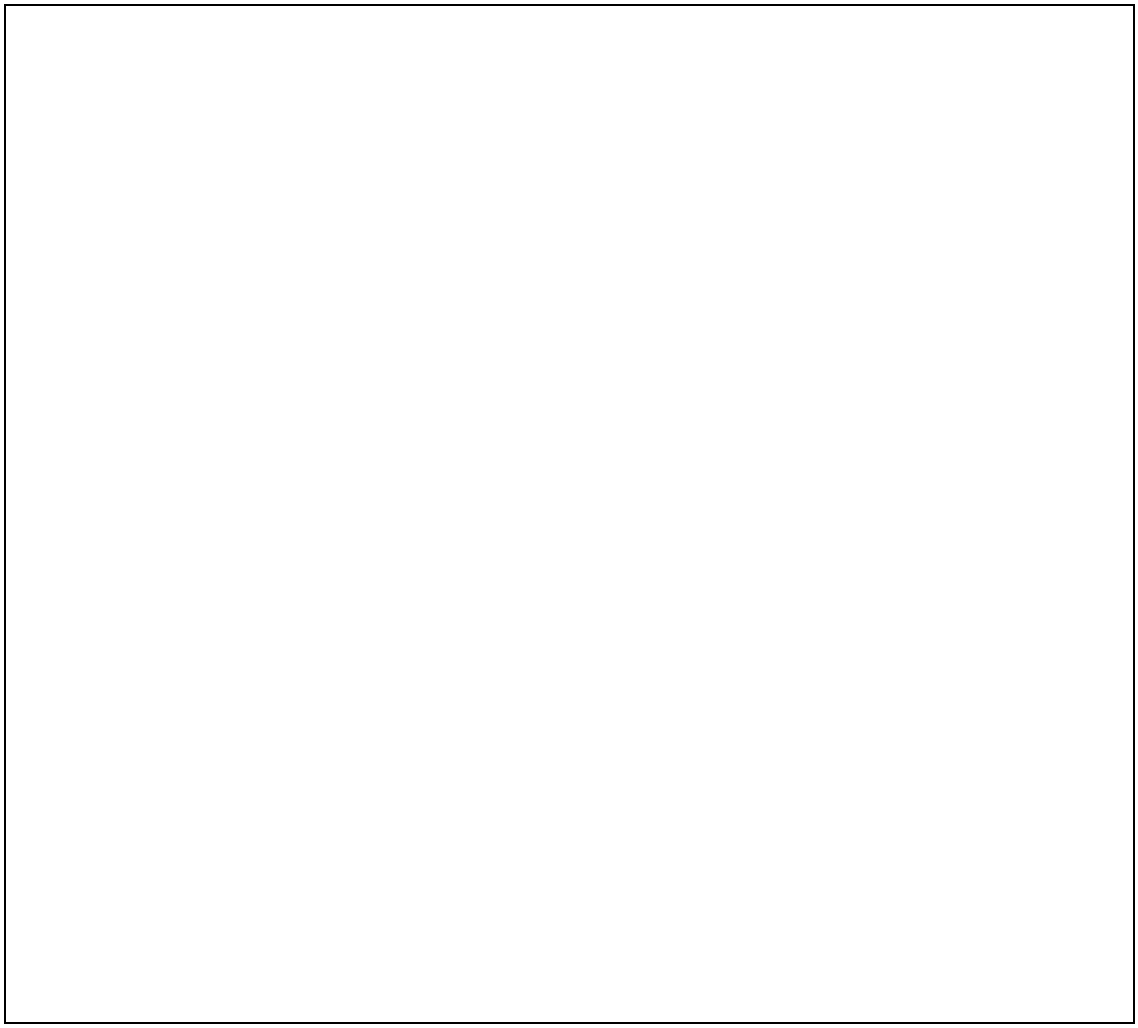
WHR-G54S wiring points, Component side (Not WHR-HP-G54!):



WHR-G54S wiring points, Back-side (Not WHR-HP-G54!):

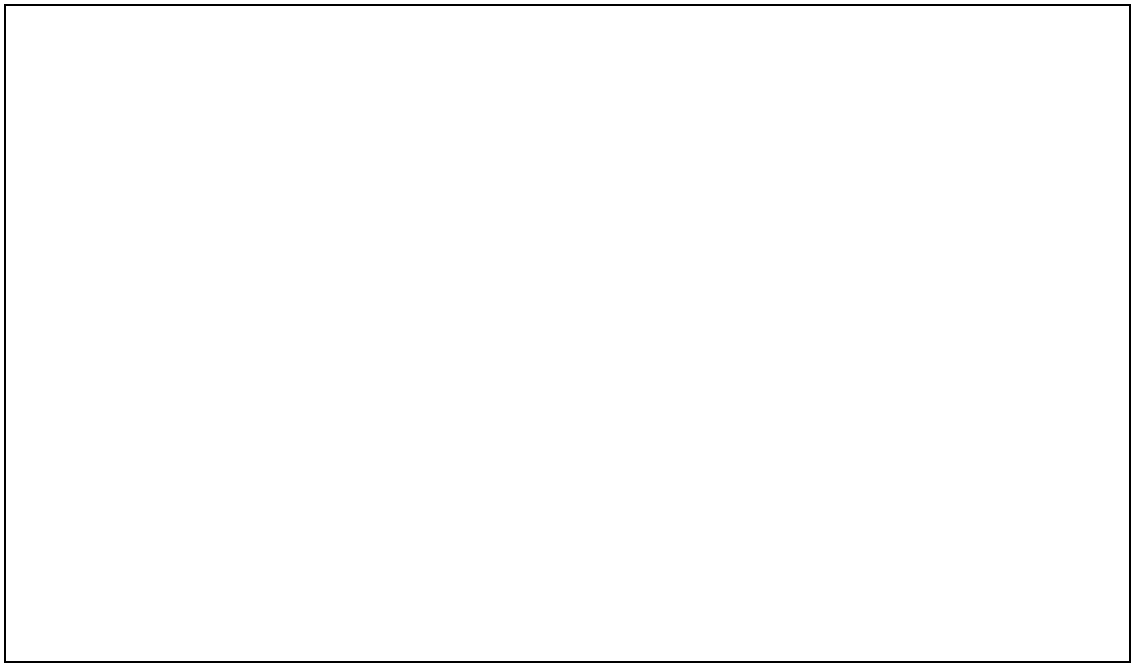


WHR-HP-G54 wiring points:
Note - All the WHR-HP-G54 wiring points are on the component side.



Setting up DD-WRT to support the modification

We will need to configure the MMC interface in the DD-WRT GUI (firmware V2.4). Set it up as indicated in the screenshot of the Administration screen below. If you soldered SD Pin 1(Chip Select) to ground then you can use any(0-9) **unused** GPIO output number in the CS field. The screenshot also indicates that modification is working, the available disk space is shown. In case something is wrong I usually found that 3,008.00 KB is reported. This also happens if no SD card is inserted.



Be careful now, all data on the SD card will be erased without warning once you boot the router. If everything works correctly then DD-WRT will format the SD card with the EXT2 file system if this file system in not present on the card. This will take some time, so be patient. If everything works, then the DD-WRT gui will report the available disk space. An extra reboot might be required for this.

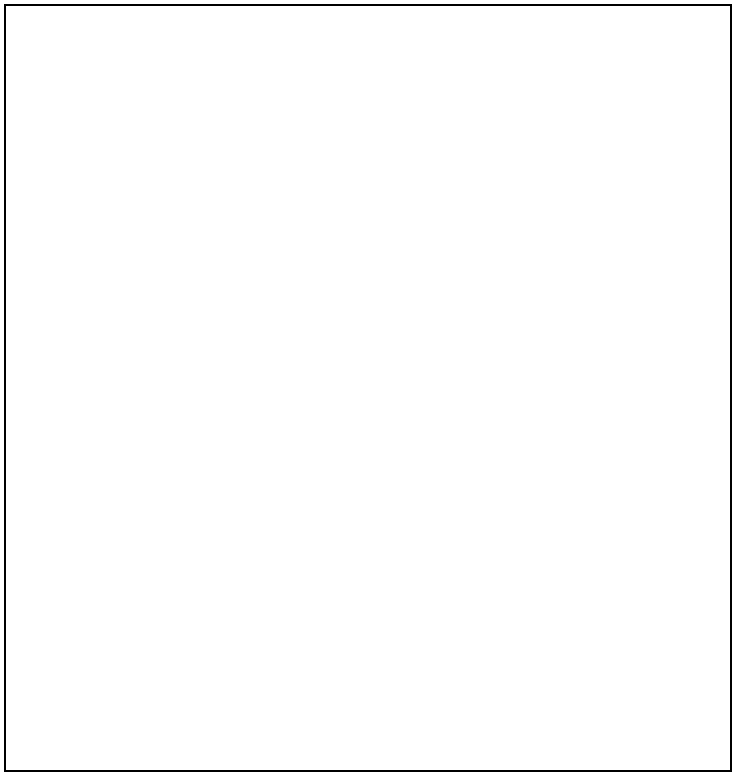
- User Mcta Comments: If, for some reason the DD-WRT format process doesn't work, you can try formatting the card externally. To do this you'll need a Linux package. For Windows only users you can download a CD bootable version of Knoppix linux. When you format the card make sure the block size is 1024 bytes or smaller. For a 1 GB SD flash card Knoppix chooses a default block size of 4096 bytes. To format the card with a 1024 byte block size use the following command:

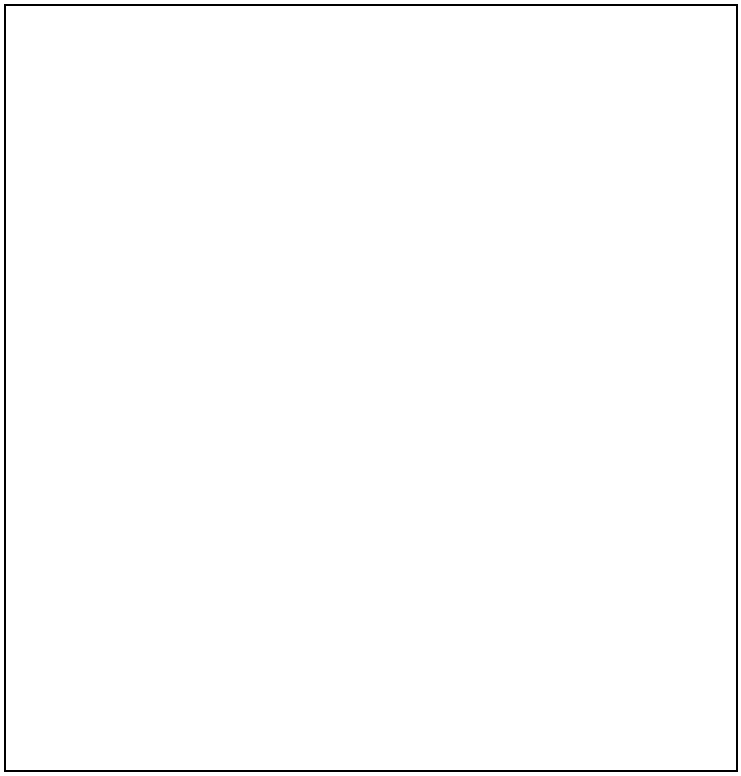
```
sudo mke2fs -b 1024 /dev/sda1  (change the /dev/sda1 as appropriate)
```

Editing files and folders on the card

- The SD card is automatically mounted at: "/mmc".
- If CIFS/SAMBA doesn't work, or you just want to store all the JFFS/IPKG content on the Router, JFFS space can be moved onto the card. To do this, turn on JFFS in the Web Administration page, *mkdir /mmc/jffs* and put the command in your startup: *mount --bind /mmc/jffs /jffs*. Now anything written into /jffs is actually being stored on the SD card, in /mmc/jffs. JFFS now also shows the same free space as the SD card in the Web interface.
- It is possible to automatically run a script from the SD card. To do this create a executable script in: "/mmc/etc/config". See Startup_Scripts#Save_the_Script and Script_Execution
- Windows does not support the EXT2 file system, but this can be solved with a driver that can be found here: http://www.fs-driver.org
- To access the router file system and SD card from any computer which can SSH into the router, you can use SCP. For Windows XP (or 2K) users, WinSCP.

Some pictures of the modification on a WHR-G54S





Pictures of the modification on a WHR-HP-G54

Credits

Your feedback, questions and remarks are welcome, just drop Iron a message in the forum.

Missing tags/info/additional photos/very minor structural changes by sdoboze

Kudos to JohnS

Other Resources

Look here at the "Porting to other platforms" Section, very bottom of page. Forum links: [3]. Yes. I know it's OpenWRT. Before anyone gets juvenile or hostile, DD-WRT is based on OpenWRT.