WRT54GL I2C Bus and Real-Time Clock

Author: Kenny Saltiel

Table of Contents

Introduction

This document will describe the steps taken to build the hardware and software required to add an I2C bus and a Real-Time Clock to the Linksys WRT54GL router. The I2C bus is very versatile and can interface with many components such as eeprom memory chips, GPS location detectors, solid-state compasses, LCD displays, etc and of course the Real-Time clock chip described here.

Disclaimer

Credits

The information presented here was partially gathered from sources on the Internet including the following forums:

http://www.linksysinfo.org/ and http://openwrt.org/

Firmware:

OpenWrt Whiterussian 0.9

Kernel driver i2c-mips-gpio:

Based on original i2c-mips-gpio by John Newbigin,

which in turn was based on i2c-philips-par.c by Simon G. Vogl.

Changed to sb_gpio-Interface by Torsten Landschoff.

4-wire logic and final version by niclas at datenritter dot de.

User space programs are modified versions of utilities provided in the lm_sensors package: http://www.lm-sensors.org/

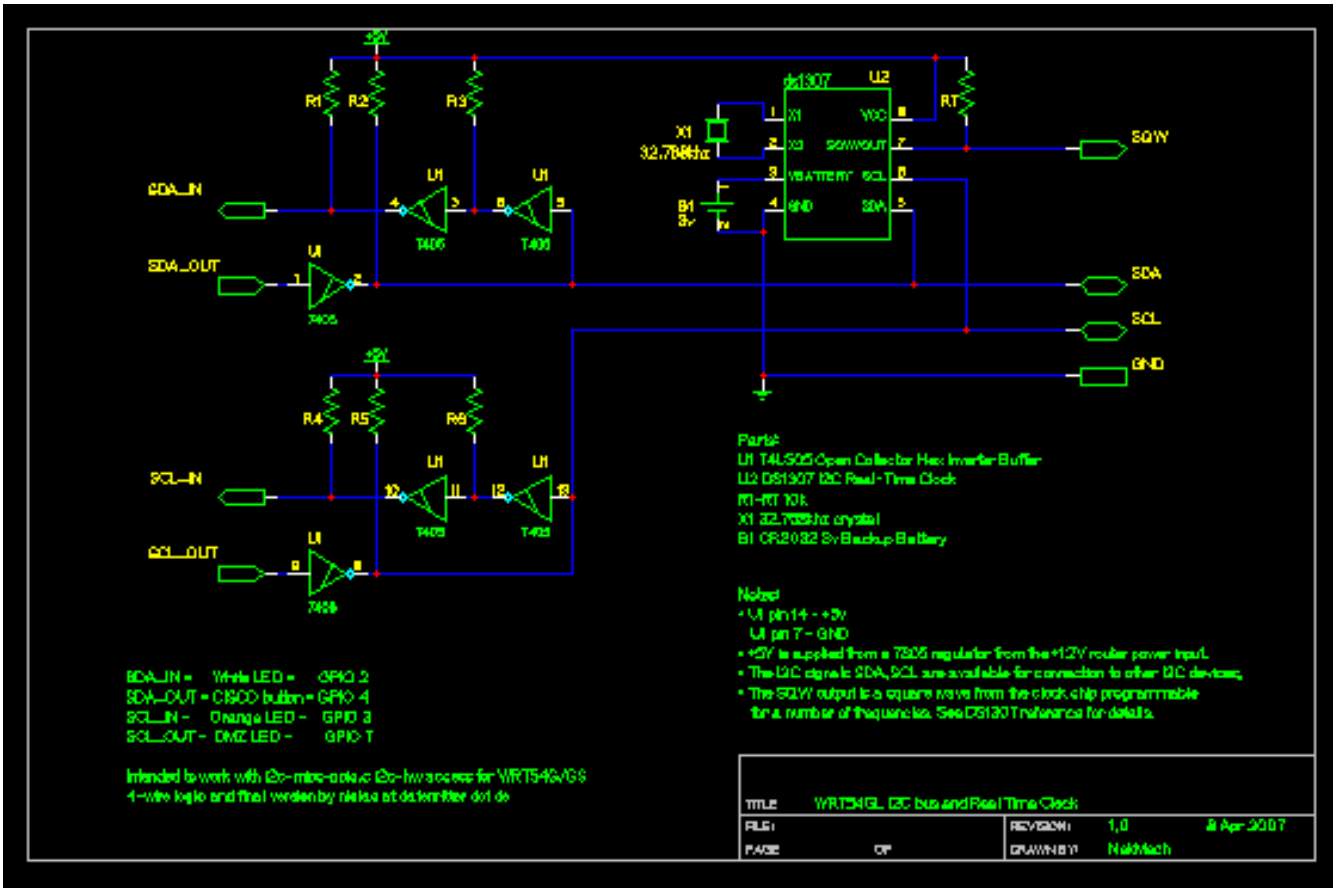
Schematics were created with gEDA: http://www.geda.seul.org/

This document was written with OpenOffice2: http://www.openoffice.org/

Thanks to all who have posted their knowledge on the Internet and to the OpenWrt team for their great firmware.

The Hardware

Schematic description



This is a png image of the I2C and clock schematic. The full size png and gEDA schematic is bundled with this document.

Basically, the circuit is comprised of 2 IC's and support components.

The 74LS05 buffer is open collector and requires pull up resistors on the outputs. All incoming and outgoing signals are buffered.

The DS1307 clock circuit is just the basic reference design specified in the data sheet and it's I2C signals are just connected to the I2C bus.

The SQW signal is just a square wave programmable as 1Hz, 4kHz, 8kHz, 32kHz or as a constant high or low output. Can be used for anything you want.

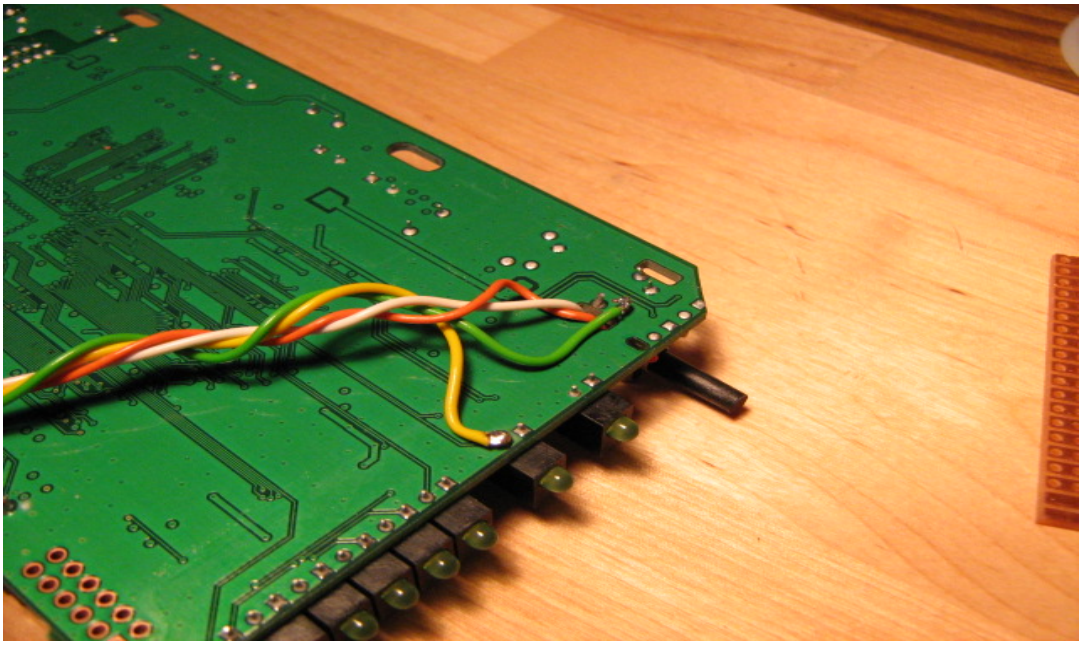
The battery backup mechanism of the clock chip ensures time is maintained also when the router is powered off.

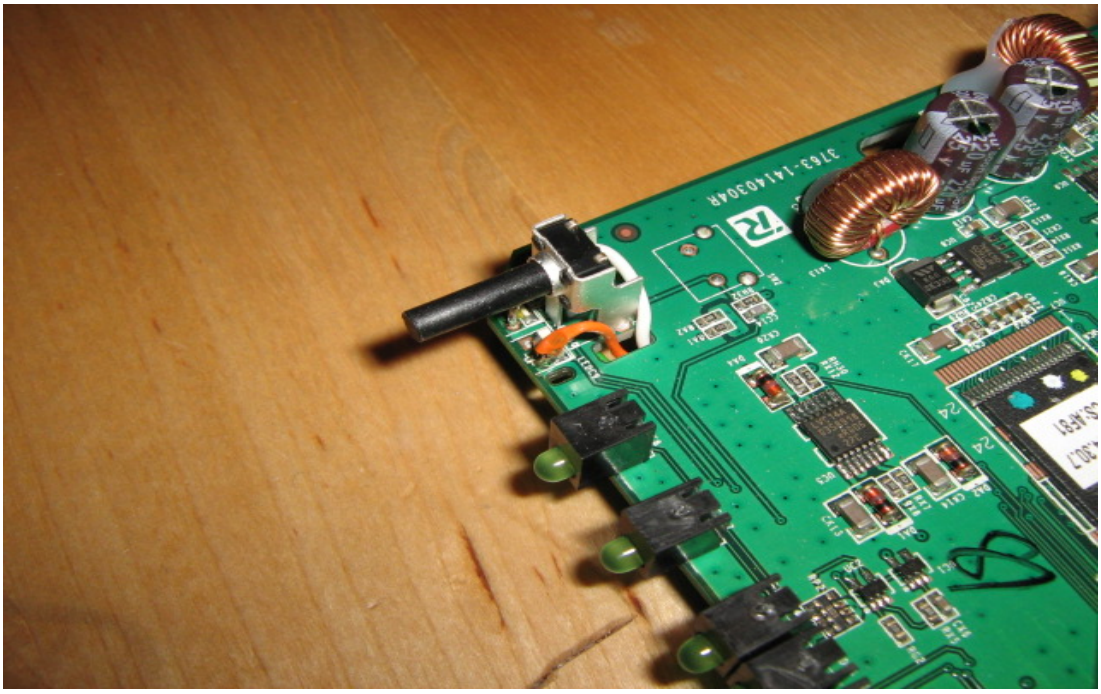
+5V power was obtained from the 7805 regulator used in the Serial Console circuit which is not described here but the gEDA schematic and png for it should accompany this document.

I2C GPIO signals:

SDA_OUT :	GPIO 4 (Cisco switch)
SDA_IN:	GPIO 2 (White LED)
SCL_OUT:	GPIO 7 (DMZ LED)
SCL_IN:	GPIO 3 (Orange LED)

GPIO Pins

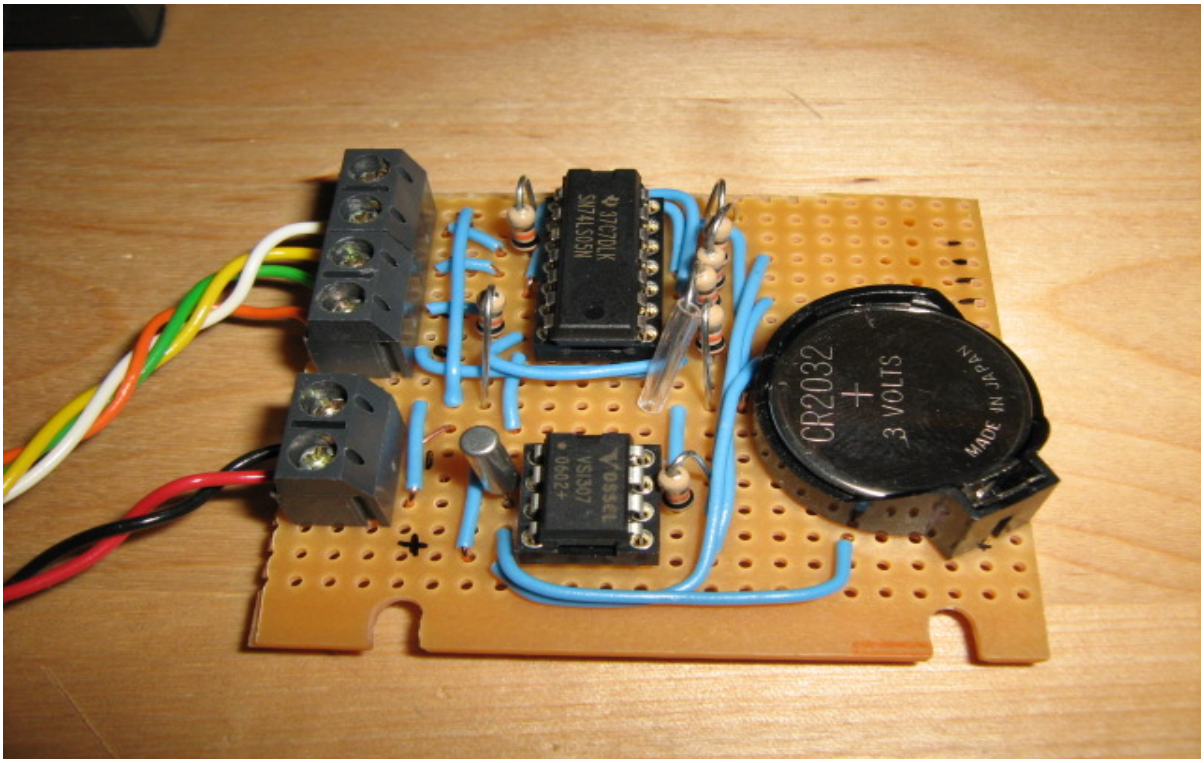




Yellow wire = GPIO 7 (DMZ LED)
Green wire = GPIO 4 (Cisco button)
White wire = GPIO 2 (White LED)
Orange wire = GPIO 3 (Orange LED)

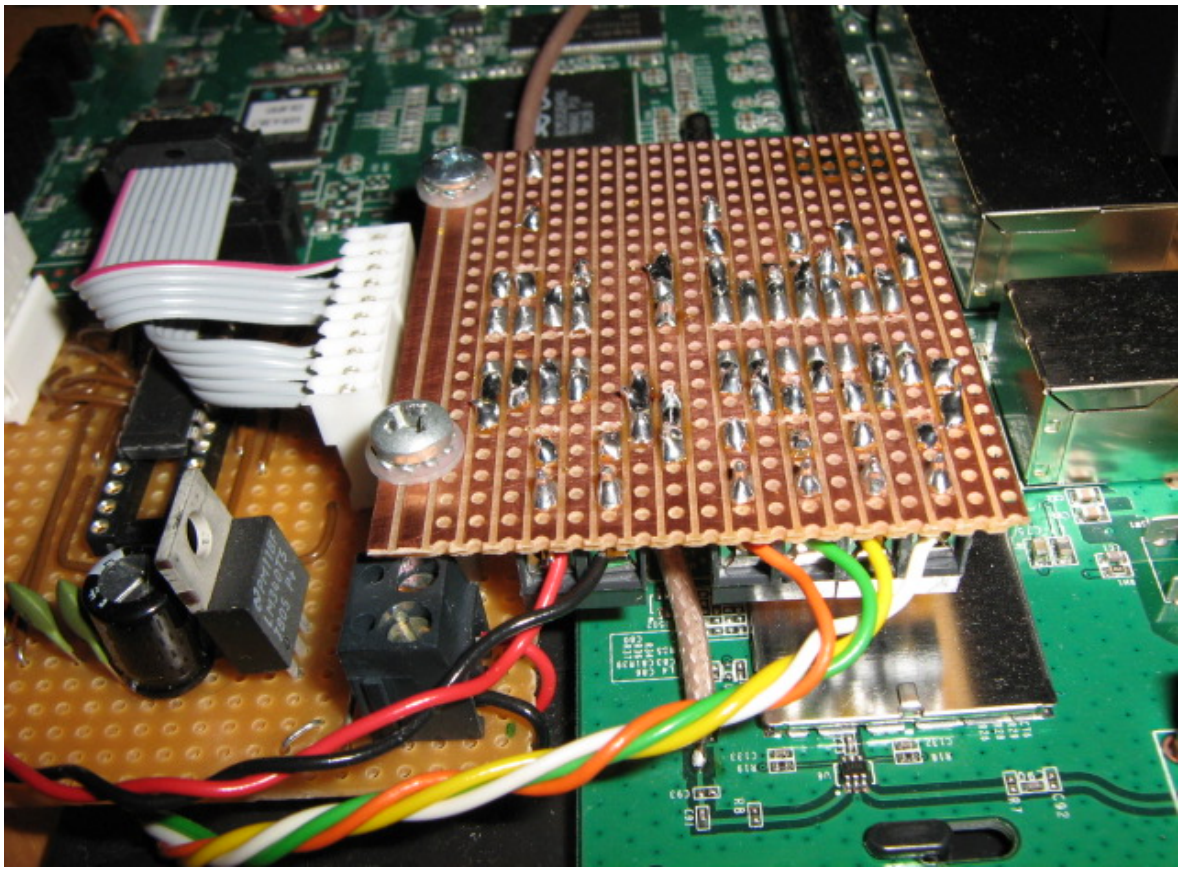
Construction

The circuit was built on a small piece of perforated board.



The upper right hand corner is for the I2C extension connector I haven't added yet.

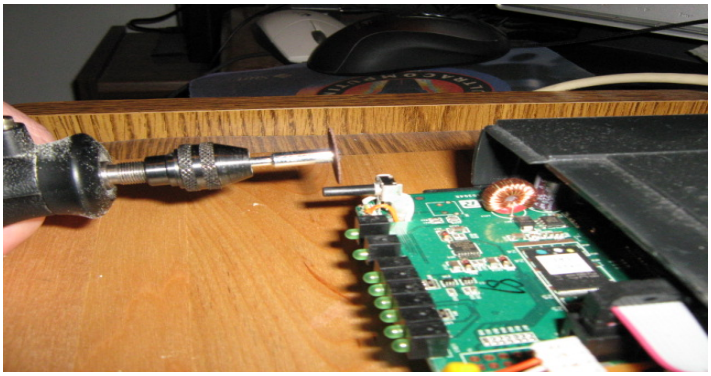
The 4 GPIO signals as you see are connected with the upper terminal block and +5V connected with the lower terminal block.



Mounted in the router together with the serial console adapter and 5v regulator.

Although it doesn't look like it, there's about 10mm clearance above the main board.

Since the gpio pin for the Cisco button is used, I decided to shorten the length of the buttons lever to avoid it getting pressed accidentally. This probably wouldn't cause any problems but better to be safe.



The Software

Prerequisites

1. OpenWrt Whiterussian 0.9 flashed to the router and running normally. Available from http://downloads.openwrt.org/whiterussian/0.9/default/
2. Microperl ipkg installed on the router. Available from http://downloads.openwrt.org/backports/0.9/microperl_5.8.6-1_mipsel.ipk
3. A Linux workstation if you plan on compiling the binaries.

WRT54GL OS

Not much to be said here besides flash your router with Whiterussian 0.9 as stated in the prerequisites. No other router or Whiterussian version has been tested.

Kernel Modules

These kernel modules were used:

i2c-core

i2c-algo-bit

i2c-proc

i2c-dev

i2c-mips-gpio (compiled separately from the regular kernel modules)

User Space programs and scripts - Overview

There are 3 binaries:

i2cset, i2cread, i2cdump

And 3 scripts:

i2c-load.sh, gethwclock.pl, S99i2c

In all these examples the device used was the DS1307 I2C clock chip.

The clock chip is wired to I2C bus “0” (the only one), and has a device address of “104” decimal or “68H” hex. These programs were all installed under /usr/share/i2c on the router and are run from there besides S99i2c which is installed under /etc/init.d.

Scripts

There are 3 scripts.

i2c-load.sh – bash script for loading the kernel modules in the correct order with "insmod".

gethwclock.pl – microperl script which performs all the tasks of reading and writing to the clock chip. You will need the microperl ipkg installed as stated in the prerequisites.

S99i2c – bash script which is executed at boot to load the kernel modules, and update the system time from the hardware clock.

Binaries

i2cset – Sends any command over the i2c bus to any device.

i2cread – Reads any number of characters from any device on the i2c bus.

i2cdump – Provided as a diagnostic tool which can read all the available from any i2c device.

User Space programs and scripts – Description

i2cset

Used to send data/commands to an i2c device.

Running with no command line options displays an error message and help syntax as well as the available i2c busses.

Example:

```
root@OpenWrt:/usr/share/i2c# ./i2cset
Syntax: i2cset I2CBUS CHIP-ADDRESS VALUES
  I2CBUS is an integer
  Installed I2C busses:
    i2c-0       i2c             WRT54G GPIO                     Bit-shift algorithm
```

The gethwclock.pl script uses it to set the time, the control register and to move the register pointer of the hardware clock to the correct position for reading the time.

i2cread

Used to read back data from an i2c device.

Running with no command line options, displays an error message and help syntax as well as the available i2c busses.

Example:

```
root@OpenWrt:/usr/share/i2c# ./i2cread
Syntax: i2cread I2CBUS CHIP-ADDRESS COUNT
  I2CBUS is an integer
  Installed I2C busses:
   i2c-0       i2c           WRT54G GPIO                   Bit-shift algorithm
```

The gethwclock.pl script uses it to read the hardware clock during boot (to set the system clock) and any time requested manually.

i2cdump

Used as a general purpose diagnostic tool, it performs a dump of any i2c devices registers/memory.

Running with no command line options displays an error message and help syntax.

Example:

```
root@OpenWrt:/usr/share/i2c# ./i2cdump
Error: No i2c-bus specified!
Syntax: i2cdump I2CBUS ADDRESS [MODE] [BANK [BANKREG]]
  MODE is 'b[yte]', 'w[ord]', 's[mbusblock], or 'i[2cblock]' (default b)
  Append MODE with 'p' for PEC checking
  I2CBUS is an integer
  ADDRESS is an integer 0x00 - 0x7f
  BANK and BANKREG are for byte and word accesses (default bank 0, reg 0x4e)
  BANK is the command for smbusblock accesses (default 0)
  Installed I2C busses:
   i2c-0       i2c           WRT54G GPIO                   Bit-shift algorithm
```

Running the utility with the bus number “0” and device address “104” will dump all the registers for the clock chip. Note that addresses 08H – 3FH are general purpose ram registers which can be used for anything you want. Do not use ram register 3FH since it will be overwritten when reading the date from the clock chip by the gethwclock.pl script.

Example:

```
root@OpenWrt:/usr/share/i2c# ./i2cdump 0 104
No size specified (using byte-data access)
  WARNING! This program can confuse your I2C bus, cause data loss and worse!
  I will probe file /dev/i2c/0, address 0x68, mode byte
  You have five seconds to reconsider and press CTRL-C!
```

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f                  0123456789abcdef
00: 04 48 16 01 09 04 07 10 10 10 6a 50 52 c3 4d 22    ?H????????jPR?M"
10: cb 6a 68 00 e2 7c 50 41 03 01 f9 55 12 0c 69 0c    ?jh.?|PA???U??i?
20: 70 1e 02 42 a8 2c 02 2a bc 48 28 3e 80 2c b7 84    p??B?,?*?H(>?,??
30: 1f 00 68 42 c6 5e 34 d4 42 8a 20 28 c7 90 fe ff    ?.hB?^4?B? (???.
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
```

i2c-load.sh

Bash script which will load the kernel modules in the following order:

1. i2c-core
2. i2c-algo-bit
3. i2c-proc
4. i2c-dev
5. i2c-mips-gpio

It normally gets run by S99i2c during router boot.

When modules are successfully loaded you should see the device under the devices directory /dev:

```
root@OpenWrt:/usr/share/i2c# ls -l /dev/i2c/0
crw-------    1 root     root      89,   0 Jan  1  1970 /dev/i2c/0
```

And under /proc:

```
root@OpenWrt:/usr/share/i2c# cat /proc/bus/i2c
i2c-0   i2c             WRT54G GPIO                     Bit-shift algorithm
```

When examining the output of the “dmesg” command, you should see:

```
i2c-core.o: i2c core module version 2.6.1 (20010830)
```

i2c-algo-bit.o: i2c bit algorithm module
i2c-proc.o version 2.6.1 (20010830)
i2c-dev.o: i2c /dev entries driver module version 2.6.1 (20010830)
i2c-core.o: driver i2c-dev dummy driver registered.
i2c-mips-gpio.o: i2c WRT54G GPIO module version 1.5 2005-12-16
i2c-dev.o: Registered 'WRT54G GPIO' as minor 0
i2c-core.o: adapter WRT54G GPIO registered as adapter 0.

lsmod will show:

```
root@OpenWrt:/usr/share/i2c# lsmod
Module                  Size  Used by    Tainted: P
i2c-mips-gpio           1728   0
i2c-dev                 4252   0
i2c-proc                7020   0 (unused)
i2c-algo-bit            8948   1 [i2c-mips-gpio]
i2c-core               17944   0 [i2c-dev i2c-proc i2c-algo-bit]
```

gethwclock.pl

This is a microperl script which writes to the clock chip using i2cset, and reads from it using i2cread. It reads the clock chip and sets the system time by converting the registers from the clock chip to a format usable by the date command, and runs date.

On the reverse side, it updates the clock chip (on demand only) by reading system time and converting it to the register values which are programmed in the clock with i2cset.

When run with no command line options, it will display a help message with syntax.

Example – help message:

```
root@OpenWrt:/usr/share/i2c# ./gethwclock.pl

gethwclock.pl <sethw|gethw|setos|inithw>

sethw - set the hardware clock from system time
gethw - display the date stored in the hardware clock
setos - set the system time from the hardware clock
inithw - initialize hardware clock control register
        and start counter. This should be done once before
        using the clock for the first time
```

Example – display time from clock chip:
root@OpenWrt:/usr/share/i2c# ./gethwclock.pl gethw
Mon Apr 09 13:31:07 2007

Example – showing how the clock chip is used to set the system time:
root@OpenWrt:/usr/share/i2c# date 123023392005.22
Fri Dec 30 23:39:22 UTC 2005
root@OpenWrt:/usr/share/i2c# date
Fri Dec 30 23:39:24 UTC 2005
root@OpenWrt:/usr/share/i2c# ./gethwclock.pl setos
root@OpenWrt:/usr/share/i2c# date
Mon Apr 9 13:25:38 UTC 2007

Example – showing the system date getting saved to the clock chip:
root@OpenWrt:/usr/share/i2c# date 123018232005.20
Fri Dec 30 18:23:20 UTC 2005
root@OpenWrt:/usr/share/i2c# ./gethwclock.pl sethw
root@OpenWrt:/usr/share/i2c# ./gethwclock.pl gethw
Fri Dec 30 18:23:32 2005

Example – running the inithw command to set the SQW output and start the clock:
root@OpenWrt:/usr/share/i2c# ./gethwclock.pl inithw
hw clock init succeeded

S99i2c

Located in /etc/init.d and runs during boot of the router, it performs 2 functions.

1. Load the I2C kernel modules by running /usr/share/i2c/i2c-load.sh
2. Sets the system time from the hardware clock using /usr/share/i2c/gethwclock.pl

Installing

The easy way

An ipkg is provided: linksys-wrt54gl-i2c-rtc_1_mipsel.ipk

Transfer this package to the router and install it. All the files will be installed in the correct locations:

/etc/init.d/S99i2c

/lib/modules/2.4.30/i2c-algo-bit.o

/lib/modules/2.4.30/i2c-core.o

/lib/modules/2.4.30/i2c-dev.o

/lib/modules/2.4.30/i2c-mips-gpio.o

/lib/modules/2.4.30/i2c-proc.o

/usr/share/i2c/ i2cread

/usr/share/i2c/i2cdump

/usr/share/i2c/i2c-load.sh

/usr/share/i2c/gethwclock.pl

/usr/share/i2c/i2cset

Set the routers date:

root@OpenWrt:/# date MMDDhhmm[[CC]YY][.ss]

Example:

root@OpenWrt:/# date 041520302007.30

For April 15 20:30:30 2007

Set the hardware clock chip from the system time:

root@OpenWrt:/# cd /usr/share/i2c

root@OpenWrt:/usr/share/i2c#./gethwclock.pl sethw

The hard way

Compile kernel modules.

Compiled using the buildroot kit and selecting the i2c character device support when running “make menuconfig” under the “build_mipsel/linux” directory. After that they were just copied over to the router.

Compile the binaries

All binaries (i2cset,i2cread,i2cdump) were compiled under the buildroot tree for Whiterussian 0.9. Binaries, source files and the Makefile are included as a tar file with this document. The tar file should be extracted under OpenWrt root directory.

Note that a complete firmware build has already been performed to create the necessary kernel modules. These binary tools are not dependent on the kernel modules during compilation. See OpenWrt BuildRoot documentation to see how thats done.

The extracted tar file appears as directory “i2cread” marked in blue below.

```
/home/kenny/whiterussian/openwrt $ls -l
total 96
-rw-r--r--   1 kenny users  1339 2007-03-09 17:33 Config.in
-rw-r--r--   1 kenny users  2013 2007-03-09 17:33 Config.in.devel
-rw-r--r--   1 kenny users 17992 2007-03-09 17:33 LICENSE
-rw-r--r--   1 kenny users  4689 2007-03-09 17:33 Makefile
-rw-r--r--   1 kenny users   728 2007-03-09 17:33 README
drwxr-xr-x   3 kenny users  4096 2007-03-31 15:15 bin/
drwxr-xr-x  30 kenny users  4096 2007-03-31 03:12 build_mipsel/
drwxr-xr-x   2 kenny users  4096 2007-03-09 23:27 dl/
drwxr-xr-x   4 kenny users  4096 2007-03-09 17:32 docs/
drwxr-xr-x   2 kenny users  4096 2007-04-07 00:21 i2c-mips-gpio/
drwxr-xr-x   2 kenny users  4096 2007-04-09 19:48 i2cread/
drwxr-xr-x 135 kenny users  4096 2007-03-09 17:33 package/
-rw-r--r--   1 kenny users  3636 2007-03-09 17:33 rules.mk
drwxr-xr-x   3 kenny users  4096 2007-03-09 17:32 scripts/
drwxr-xr-x  10 kenny users  4096 2007-03-09 23:25 staging_dir_mipsel/
drwxr-xr-x   6 kenny users  4096 2007-03-09 17:32 target/
drwxr-xr-x  11 kenny users  4096 2007-03-09 17:32 toolchain/
drwxr-xr-x  13 kenny users  4096 2007-03-09 18:16 toolchain_build_mipsel/
```

Typing the “Make” command will build the binaries from source files.

Example:

```
/home/kenny/whiterussian/openwrt/i2cread $make

/home/kenny/whiterussian/openwrt/staging_dir_mipsel/bin/mipsel-linux-uclibc-gcc -O2 -W -Wall
-Wstrict-prototypes -Wmissing-prototypes -fno-strict-aliasing -mips32 -isystem
/home/kenny/whiterussian/openwrt/build_mipsel/linux-2.4-brcm/linux-2.4.30/include -I
/home/kenny/whiterussian/openwrt/build_mipsel/linux-2.4-brcm/linux-
2.4.30/arch/mips/bcm947xx/include  -c i2cset.c -o i2cset.ro

i2cset.c:33: warning: no previous prototype for 'help'

/home/kenny/whiterussian/openwrt/staging_dir_mipsel/bin/mipsel-linux-uclibc-gcc -o i2cset
i2cset.ro

/home/kenny/whiterussian/openwrt/staging_dir_mipsel/bin/mipsel-linux-uclibc-gcc -O2 -W -Wall
-Wstrict-prototypes -Wmissing-prototypes -fno-strict-aliasing -mips32 -isystem
/home/kenny/whiterussian/openwrt/build_mipsel/linux-2.4-brcm/linux-2.4.30/include -I
/home/kenny/whiterussian/openwrt/build_mipsel/linux-2.4-brcm/linux-
2.4.30/arch/mips/bcm947xx/include  -c i2cread.c -o i2cread.ro

i2cread.c:33: warning: no previous prototype for 'help'

/home/kenny/whiterussian/openwrt/staging_dir_mipsel/bin/mipsel-linux-uclibc-gcc -o i2cread
i2cread.ro

/home/kenny/whiterussian/openwrt/staging_dir_mipsel/bin/mipsel-linux-uclibc-gcc -O2 -W -Wall
-Wstrict-prototypes -Wmissing-prototypes -fno-strict-aliasing -mips32 -isystem
/home/kenny/whiterussian/openwrt/build_mipsel/linux-2.4-brcm/linux-2.4.30/include -I
/home/kenny/whiterussian/openwrt/build_mipsel/linux-2.4-brcm/linux-
2.4.30/arch/mips/bcm947xx/include  -c i2cdump.c -o i2cdump.ro

i2cdump.c:47: warning: no previous prototype for 'help'

/home/kenny/whiterussian/openwrt/staging_dir_mipsel/bin/mipsel-linux-uclibc-gcc -o i2cdump
i2cdump.ro

rm *.ro
```

These binaries can be copied over to the router with scp, set the executable bit for each one with chmod +x <file name>, and test by running with no command line options. See section #5.5.User Space programs and scripts – Description for further information on each one.

As mentioned previously, you probably won't need to build the binaries yourself unless you want to, since precompiled binaries are supplied in the tar file for the Broadcom BCM5352 CPU used in the WRT54GL. Note that the binaries are not “stripped”.

Compile the i2c-mips-gpio kernel module

Extracting the tar file will also create the i2c-mips-gpio directory which contains the source and Makefile for this module.

Type “make” to compile it.

The result will be i2c-mips-gpio.o which needs to be copied over to the router and installed under the module directory /lib/modules/2.4.30

Scripts

Copy the scripts described in section #5.5.User Space programs and scripts – Description to their correct locations and set the executable bit for each.

Conclusion

Reboot to check that it works. Note that there's a couple seconds delay from the time the router has finished booting till the date gets set. This is due to the time it takes to load the kernel modules and run the gethwclock.pl script.

Don't forget that a number of I2C devices can be connected to the same bus since each has a unique address, and the tools installed here to control the hardware clock can be used for other devices as well.