

Building the H8300 gcc cross compiler from scratch

Adam Wulf
Travis R. Fischer

January 20, 2006

This document is a modified version of one presented by John Regehr at <http://www.cs.utah.edu/~regehr/research/BrickOS-Cyclone/INSTALL.html>.

Step 1

- Create a folder to place everything in, for example:
`mkdir ~/RCX`

Step 2

- Goto: `ftp://ftp.gnu.org/pub/gnu/gcc`
- Find the folder here called: "releases" and click it.
- Download the: "gcc-3.3.3.tar.gz" file into your RCX folder.

INFO: We use one of the gcc versions from the releases folder to get a stable version of the gcc, which we can reliably build our cross compiler with. (Using some of the currently worked on compilers such as the gcc 4.0, caused problems).

The gcc provides: GNU C and C++ compilers

Step 3

- Goto: `ftp://ftp.gnu.org/pub/gnu/binutils`
- Find the newest version to download, in my case it was:
`binutils-2.16.tar.gz`
- Download it to your RCX folder.

INFO: Binutils is a collection of binary utilities, including `ar`, `nm`, `objcopy`, `objdump`, `ranlib`, `size`, `strings`, `strip`, `c++filt`, `addr2line`, `nlmconv`.

We need the `objcopy` to create `.srec` files which are the type of file that we upload to the RCX brick.

Step 4

- Goto: `http://sources.redhat.com/newlib/`
- Click on "download" in the left menu.
- Stand in the RCX folder when typing:
`cvs -z 9 -d :pserver:anoncvs@sources.redhat.com:/cvs/src login`
- When asking for a password enter: "anoncvs"
- Although it doesn't appear to do anything, it enables you to do the following command:
`cvs -z 9 -d :pserver:anoncvs@sources.redhat.com:/cvs/src co newlib`

INFO: This will download all the files into a folder called: `/src` Inside it there will be several files, including folders such as `newlib`, `libgloss` etc.

The `newlib` provides a C library to be used when developing for embedded systems. It makes the files created smaller, since embedded systems are limited in resources such as memory.

Step 5

- Change the folder name from `src` to `newlib-src` (so we know what it is the source for).

Step 6

- Standing in the RCX folder type:

```
tar -zxvf gcc-3.3.3.tar.gz
```

(If you downloaded a different version this filename will of course be different.)

- This will create a folder called `gcc-3.3.3`

INFO: This line will unpack the tar-ball file that you downloaded with the files used to build our new gcc compiler in a few steps.

Step 7

- Standing in the RCX folder type:

```
tar -zxvf binutils-2.16.tar.gz
```

(Again, if you downloaded a different version of the bin utils, you the filename will be different).

- This will unpack the file into a folder called: `binutils-2.16`

Step 8

- Preparing to build our native gcc, we setup a few symbolic links to make sure we use the right folders.
- Standing in the RCX folder type:

```
- cd gcc-3.3.3
- ln -s ../binutils-2.16/bfd
- ln -s ../binutils-2.16/binutils
- ln -s ../binutils-2.16/gas
- ln -s ../binutils-2.16/gprof
- ln -s ../binutils-2.16/ld
- ln -s ../binutils-2.16/opcodes
- rm -rf libiberty
- ln -s ../binutils-2.16/libiberty
- ln -s ../newlib-src/newlib
- ln -s ../newlib-src/libgloss
```

Step 9

We now build the native gcc, which we need to build the cross compiler later.

- Create a temporary folder, standing in the RCX folder type:

```
mkdir Temp
```

- Standing in the RCX folder type:

(NOTE: change "dyrby" to your own user account name.)

```
- mkdir build-native
- cd build-native
- ../gcc-3.3.3/configure --enable-languages=c --prefix=/home/dyrby/RCX/Temp
- make bootstrap
- make install
```

Step 10

It is now time to build the cross-compiler. Standing in the RCX folder type:
(NOTE: change "dyrby" to your own user account name.)

- `mkdir build-h8300`
- `cd build-h8300`
- `../gcc-3.3.3/configure --enable-languages=c --target=h8300-hms --prefix=/home/dyrby/RCX/Temp --with-newlib`
- `make`
- `make install`

Step 11

- Go to: <http://erika.sssup.it/download.shtml>
- Under "Loader" halfway down the page, download firmdl3, by clicking the link "firmdl3"
- Download the zip file. Place it in your RCX folder.
- In the RCX folder, enter the following commands:
 - `unzip firmdl`
 - `cd firmdl`
 - `make`

INFO: You must have gcc installed on your computer to compile it. Also, note that you have to do this step before you setup your path, since this file needs to be compiled with the regular gcc and not the new one that we create!

Step 12

You now have a cross-compiler. To use it, it must be added to your path.

- Go to your environment settings and add these lines to the front of your path:
(If you do not add them to the front, they will be shadowed/masked by the systems already installed gcc, assembler, linker etc.)

```
– ~/RCX/build-h8300/gcc/  
– ~/RCX/build-h8300/gas/  
– ~/RCX/build-h8300/ld/  
– ~/RCX/build-h8300/binutils/  
– ~/RCX/firmdl/
```

- Your path will now look something like:

```
default_path=~/RCX/build-h8300/gcc/:~/RCX/build-h8300/gas/:~/RCX/build-h8300/ld/:~/RCX/build-h8300/t  
PATH=$default_path
```

Step 13

It is time to download a link script.

- Goto: <http://cvs.sourceforge.net/viewcvs.py/brickos/brickos/>
- Click on h8300.rcx
- On the top entry, right click on "download" and choose save link as.
- Save this file in: `~/RCX/build-h8300/ld/ldscripts`

NOTE: In all scripts there is a command called: `ENTRY(.....)`. When you run the linker it will look for what is inside the entry command.

When writing programs for the RCX brick, you want to use `kmmain` instead of `main` that you are used to in regular c programs.

In the scripts we use the `ENTRY("_kmmain")`

Step 14

We now must set up our working libraries. Standing in your RCX folder type:

- `mkdir include` (this is for our .h files)
- `mkdir lib` (this is files written by other people than RAP)
- `mkdir src` (our own .c files)

Step 15

Now that the cross-compiler is installed, we need to create a Makefile for building RCX uploadable files.

NOTE: when you see `¡TAB¿` press the tab key, spaces are NOT the same as pressing tab. Spaces instead of tab will cause an error. Use tab in make files.

Also Note: Its important to keep the `-c` flag when compiling with our `gcc`, because it disable automatic linking, we need to do it with our own linker with the right options and arguments.

- Standing in the RCX folder, type:

```
emacs Makefile
```

- In the file write:

```
CC = gcc-cross
COFF = ld-new
SREC = objcopy
```

```
CCFLAGS = -O2 -I ./include -fno-builtin-fomit-frame-pointer
```

```
LIBPATH = ./lib/
SRCPATH = ./src/
```

```
COFFFLAGS = -T ./build-h8300/ld/ldscripts/h8300.rcx -relax
SRECFLAGS = -I coff-h8300 -O srec out.coff kernel.srec
```

```
compile:
```

```
<TAB>\$(CC) \$(CCFLAGS) \$(SRCPATH)*.c
```

```

<TAB>rm -f kernel.srec
<TAB>\$(CC) \$(CCFLAGS) -c \$(LIBPATH)*.c
<TAB>\$(COFF) \$(COFFFLAGS) -o out.coff *.o
<TAB>\$(SREC) \$(SRECFLAGS)
<TAB>make clean

clean:
<TAB>rm -f *.o
<TAB>rm -f out.coff

```

INFO: The names `gcc-cross`, `ld-new`, and `objcopy` only works if the path is setup correctly in the `.bashrc`, (or your other environment variables depending on your operating system). I suppose it could be worked around by having the complete path on these lines, in that case those would be:

```

CC = ~/RCX/build-h8300/gcc/gcc-cross
COFF = ~/RCX/build-h8300/ld/ld-new
SRED = ~/RCX/build-h8300/binutils/objcopy

```

To create a file straight up, without using the makefile, type the following: (on my `test.c` file)

- `gcc-cross -O2 -I -fno-builtin-fomit-frame-pointer -c test.c`
- `ld-new -T * ~/RCX/build-h8300/ld/ldscripts/h8300.rcx -relax -o out.coff test.o`
- `objcopy -I coff-h8300 -O srec out.coff kernel.srec`

Step 16

You can now create a new `kernel.srec` file by typing "make" while standing in the RCX folder.

Step 17

To upload the `kernel.srec` file to the Tower:

- Insert the Tower in the USB port on the computer.
- Turn on the RCX robot.
- Make sure you have write access to the serial port on the computer you are using. (might need the system admin to open it for you, depending on what machine you are working on on campus.)
- Stand in the RCX folder and type:

```
firmdl3 kernel.srec
```
- Congratulations, you have now uploaded your software to the RCX brick.