

GCCHCS12 Project General Notes

CSE4080: Computer Science Project Faculty of Science & Engineering York University

Course Director: Dr. Uyen T Nguyen
Department of Computer Science & Engineering
Office: Computer Science and Engineering Building, CSE 2024
Phone: (416) 362100 x33274
Email: utn@cse.yorku.ca

Project Director: Dr. Mokhtar Aboelaze
Department of Computer Science & Engineering
Office: Computer Science and Engineering Building, CSE 2026
Phone: (416) 362100 x40607
Email: aboelaze@cse.yorku.ca

Student: Navid Mohaghegh – 206238984
Email: CS231381@cse.yorku.ca

Table of Contents

GNU Development Tool Chain.....	1
Building GNU Development Tool Chain From Source.....	1
1 - Compiling GNU Binutils	1
2. Compiling GCC	2
3. Compiling GDB	3
4. Compiling NewLib	4
Using Binaries of GNU Development Tool Chain.....	5
GNU Development Tool Chain Documentation.....	5
Minicom.....	6..
Installation of Minicom.....	6
Minicom General Notes.....	6
Eclipse Frame Work.....	7
Eclipse C/C++ Development Tools project	7
Installation of Eclipse CDT.....	7
Using Eclipse CDT IDE for HCS12.....	9
Linking External Helper Scripts to Eclipse CDT	11
.....	11
Useful Hints on Eclipse CDT	11
References.....	12

GNU Development Tool Chain

Building GNU Development Tool Chain From Source

This description focuses on building the complete tool chain by hand. The source codes can be obtained from the download section of <http://gcc-hcs12.com> (Linux - [Source Codes](#))

Before you start, you will need:

- The GNU make utility
- GCC Compiler (3.2/3.3)
- GNU assembler, linker and utilities (binutils 2.14/2.15)
- The patch program (version 2.5 at least)
- The Texinfo documentation suite (to produce the documentation)
- Several Unix utilities (sh, cmp, cp, test, ...)

1 - Compiling GNU Binutils

Make sure you have the followings:

- GNU Binutils 2.15: binutils-2.15.tar.gz
- M68HC1x fixes (20040801): binutils-2.15-m68hc1x-20040801.diffs.gz

Type the following commands:

```
gzip -d binutils-2.15-m68hc1x-20040801.diffs.gz
tar xvzf binutils-2.15.tar.gz
mv binutils-2.15 binutils-2.15-m68hc1x
cd binutils-2.15-m68hc1x
patch -p1 < ../binutils-2.15-m68hc1x-20040801.diffs
```

When you build the GNU Binutils, you will get the support for 68HC11 and 68HC12 at the same time. The assembler and linker have a default cpu target based on the option you used during configuration.

```
sh ./configure --target=m6812-elf \  
--program-prefix=m6812-elf-
```

Then, build everything:

```
make
```

This will create: ld, as, ar, ranlib and many other tools. Typing make install will install the binutils files:

```
make install
```

2. Compiling GCC

Make sure you have the followings:

- Compiler GCC 3.3.5: gcc-3.3.5.tar.gz
- M68HC1x fixes: gcc-3.3.5-m68hc1x-20050515.diffs.gz

GCC comes with a C, C++, Objective C, Fortran, Java and Ada compiler. You'll get the cross compiler for all these languages. However, only the C compiler was really ported at this time. The C++ compiler is known to work a little. Other compilers are not checked at all.

Go back to the main folder and run the following commands:

```
gzip -d gcc-3.3.5-m68hc1x-20050515.diffs.gz
tar xvzf gcc-3.3.5.tar.gz
mv gcc-3.3.5 gcc-3.3.5-m68hc1x
cd gcc-3.3.5-m68hc1x
patch -p1 < ../gcc-3.3.5-m68hc1x-20050515.diffs
```

To configure GCC, you must use the same option you have used during configuration of the GNU Binutils.

```
sh ./configure --target=m6812-elf \
--program-prefix=m6812-elf- \
--enable-languages=c,c++
```

Then, build everything:

```
make
```

This will create: xgcc, cpp, cc1 and libgcc.a (as well as many other stuff). Move to the gcc subdirectory and type make install to install the compiler files. They are normally installed under /usr/local/bin, and /usr/local/lib/gcc-lib/m6812-elf. The GNU binutils are assumed to be installed in /usr/local/m6812-elf.

```
cd gcc
make install
```

3. Compiling GDB

Make sure you have the followings:

- GDB sources: `gdb-6.2.tar.gz`
- M68HC1x fixes and improvements (20040829): `gdb-6.2-m68hc1x-20040829.diffs.gz`

Go back to the main folder and run the following commands:

```
gzip -d gdb-6.2-m68hc1x-20040829.diffs.gz
tar xvzf gdb-6.2.tar.gz
mv gdb-6.2 gdb-6.2-m68hc1x
cd gdb-6.2-m68hc1x
patch -p1 < ../gdb-6.2-m68hc1x-20040829.diffs
```

Configure and build GDB for 68HC11/68HC12:

```
sh ./configure --target=m6811-elf \
  --program-prefix=m6811-elf-
make
```

Typing `make install` will install everything.

```
make install
```

This will install `m6811-elf-gdb` and `m6811-elf-run` in the `/usr/local/bin` directory by default. Note that GDB recognizes automatically when the program is for a 68HC11 or for a 68HC12. The simulator is also configured automatically according to the program ELF file.

4. Compiling NewLib

Make sure you have the followings:

- Newlib sources: newlib-1.12.0.tar.gz
- M68HC1x port (20040801): newlib-1.12.0-m68hc1x-20040801.diffs.gz

Go back to the main folder and run the following commands:

```
gzip -d newlib-1.12.0-m68hc1x-20040801.diffs.gz
tar xvzf newlib-1.12.0.tar.gz
mv newlib-1.12.0 newlib-1.12.0-m68hc1x
cd newlib-1.12.0-m68hc1x
patch -p1 < ../newlib-1.12.0-m68hc1x-20040801.diffs
```

The NEWLIB 1.12.0 must be built in a separate directory from the sources. Create such directory before configuring and building NEWLIB. For example, you can type the following commands:

```
cd ..
mkdir build-newlib
cd build-newlib
sh ../newlib-1.12.0-m68hc1x/configure \
  --disable-newlib-io-float --disable-newlib-multithread \
  --target=m6812-elf --program-prefix=m6812-elf-
make CFLAGS="-g -Os -Wall"
```

This will create a sub-directory m6812-elf that will contain the libc, libm and libbcc libraries. These libraries are compiled several times so that you will get 2 sets of 4 versions of them. One set for 68HC11 and one set for 68HC12, each set being composed of libraries compiled for:

- 32-bit integers, 64-bit double
- 32-bit integers, 32-bit double (-fshort-double)
- 16-bit integers, 64-bit double (-mshort)
- 16-bit integers, 32-bit double (-mshort -fshort-double)

Note if you don't specify the CFLAGS options at make time, the default makefile will use `-g -O2 -W -Wall`. You can also avoid the support for source level debugging by using: `CFLAGS="-Os -Wall"`. Also do not pass any `-mshort -m68hc11 -m68hc12 -mlong-calls` or `-fshort-double` option because this will break the multi-lib support.

Typing 'make install' will install the libraries and the includes in `/usr/local/m6812-elf/lib` and in `/usr/local/m6812-elf/include`.

make install

Using Binaries of GNU Development Tool Chain

If you do not feel comfortable to compile everything from scratch, you have the option to use pre-compiled and ready binaries of GNU Development Tool Chain. Please use below to see which one suits you the most and refer to download section of <http://gcc-hcs12.com>.

- Linux - DEB Based (Debian, Ubuntu, Knoppix ...)
 - [Have super user access and can install packages](#)
 - [Don not have super user access](#)
- Linux - RPM Based (Redhat, Fedora, CentOS, Suse ...)
 - [Have super user access and can install packages](#)
 - [Don not have super user access](#)
- Linux - Other (Gentoo, Arch ...)
 - Use Alien program along with Linux RPM
 - Or use related distributer package manager (Apt, Yum, Portage ...)
- FreeBSD:
 - Use Linux Emulation along with Linux binaries
- [Windows](#)

GNU Development Tool Chain Documentation

Please refer to <http://gcc-hcs12.com> ([Official Documentation of GNU Development Tools](#))

Minicom

Minicom is a text-based modem control and terminal emulation program for Unix-like operating systems, originally written by Miquel van Smoorenburg, and modeled after the popular MS-DOS program TeliX. Minicom includes a dialing directory, full ANSI and VT100 emulation, an (external) scripting language, and other features. Minicom is a menu-driven communications program. It also has an auto zmodem download. It can easily be used as a replacement of Microsoft HyperTerminal in Unix like operating systems.

Installation of Minicom

In Linux, you need to download the source code of Minicom under the Downloads section of <http://gcc-hcs12.com> and run the followings:

- `tar -zxvf minicom-2.2.tar.gz; cd minicom-2.2;`
- `./configure`
- `make; make install`

Minicom General Notes

To start Minicom, open a shell and type 'minicom'. A help menu screen with the various Minicom commands can be brought up at any time by typing CTRL-A Z. This means hold down the control key and type A, then release the control key and type Z (the a and z are case insensitive). To exit, type CTRL-A X. To set your options type CTRL-A O. To send a file type CTRL-A S.

```
+-----+
|                                     |
|                               Minicom Command Summary                       |
|                                     |
|               Commands can be called by CTRL-A <key>                       |
|                                     |
|               Main Functions           Other Functions                       |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Dialing directory..D | run script (Go)...G | Clear Screen.....C |
| Send files.....S   | Receive files.....R | cOnfigure Minicom..0 |
| comm Parameters...P | Add linefeed.....A | Suspend minicom...J |
| Capture on/off....L | Hangup.....H       | eXit and reset....X |
| send break.....F   | initialize Modem..M | Quit with no reset.Q |
| Terminal settings..T | run Kermit.....K   | Cursor key mode...I |
| lineWrap on/off...W | local Echo on/off..E | Help screen.....Z   |
| Paste file.....Y   |                       | scroll Back.....B   |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|                                     |
|               Select function or press Enter for none.█                       |
|                                     |
|               Written by Miquel van Smoorenburg 1991-1995                   |
|               Some additions by Jukka Lahtinen 1997-2000                   |
|               i18n by Arnaldo Carvalho de Melo 1998                       |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| CTRL-A Z for help | 9600 8N1 | NOR | Minicom 2.2 | VT102 | Offline |
```

Eclipse Frame Work

Eclipse is an open-source software framework written primarily in Java. In its default form it is a Java Integrated Development Environment (IDE), comprising the Java Development Tools (JDT) and compiler (ECJ). Users can extend its capabilities by installing plug-ins written for the Eclipse software framework, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

Eclipse C/C++ Development Tools project

Commonly known as CDT (Eclipse CDT), this is one of the most popular project under the auspices of the Eclipse Foundation. It adds C/C++ project structure and syntax recognition to the Eclipse platform. However, CDT does not include its own set of build tools but uses the GNU tool chain instead. On UNIX-like operating systems, these tools are typically supplied as part of the distribution and might not require the user to perform additional configuration for Eclipse. On Windows, however, users must supply their own build tools. This can be done through a GNU port such as Minimalist GNU for Windows (MinGW) or cygwin, or possibly by supplying tools from another IDE such as Visual Studio.

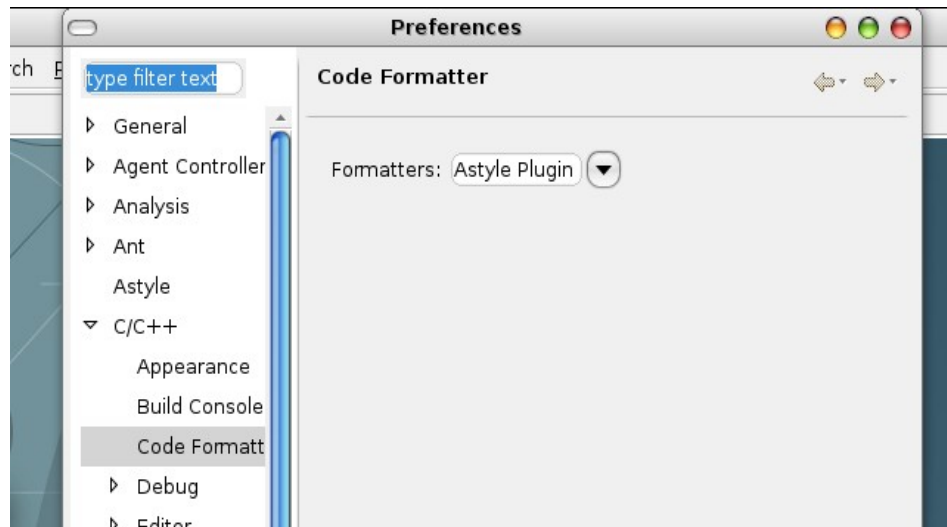
Being able to change the compiler command and its settings in an Eclipse IDE provides the possibility to develop codes almost for any processor and platforms using Eclipse CDT. In GCC-HCS12 project, Eclipse CDT is being used along with HC12 GNU Tool Chain Suit to provide an IDE for HCS12 platforms. In addition, capability of Eclipse in supporting external plugins provides the option to easily make external scripts and link them to Eclipse CDT IDE. These scripts can be used to ease the process of uploading and debugging the code on the HC12 platforms. We will explain how mentioned procedures can be done below. For more details, please refer to tutorial section of <http://gcc-hcs12.com>.

Installation of Eclipse CDT

Please visit <http://www.eclipse.org> and refer to 'Downloads' section and try to download Eclipse IDE for C/C++ Developers. Remember in Linux, a '.tar.gz' file can be opened using 'tar -zxvf filename.tar.gz'. Usually the executable file is './eclipse' once you are in Eclipse CDT directory.

If your Eclipse CDT does not have a source code formatter and beautifier, you can use <http://astyleclipse.sourceforge.net>. This plugin can be easily installed using Eclipse update mechanism. To do so, you need to refer to Software Update section which is commonly located under Help menu. You can use Find and Install option to add a new channel (<http://astyleclipse.sourceforge.net/update>) and

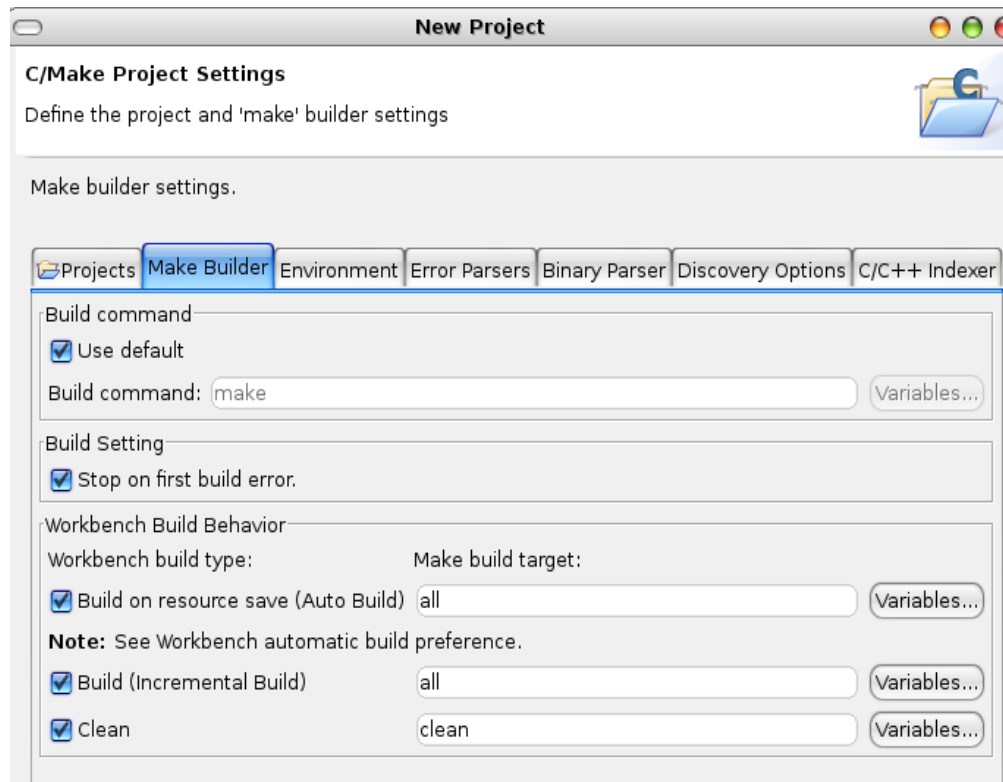
follow from there. After installation make sure your Eclipse is using Astyle plugin by going to Preferences section which is commonly located under Window menu bar. Once you are in Preferences, you should check the options for your C/C++ code formatter and make sure its using Astyle.



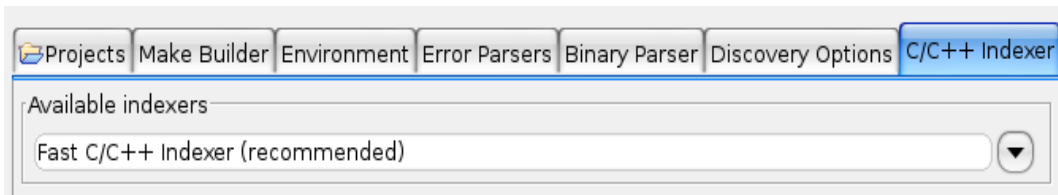
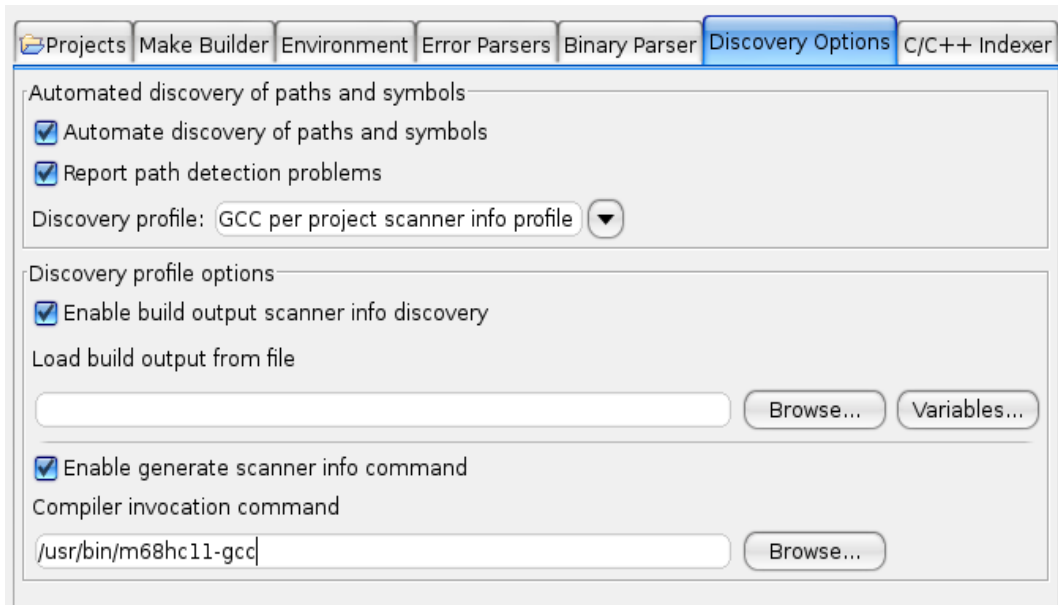
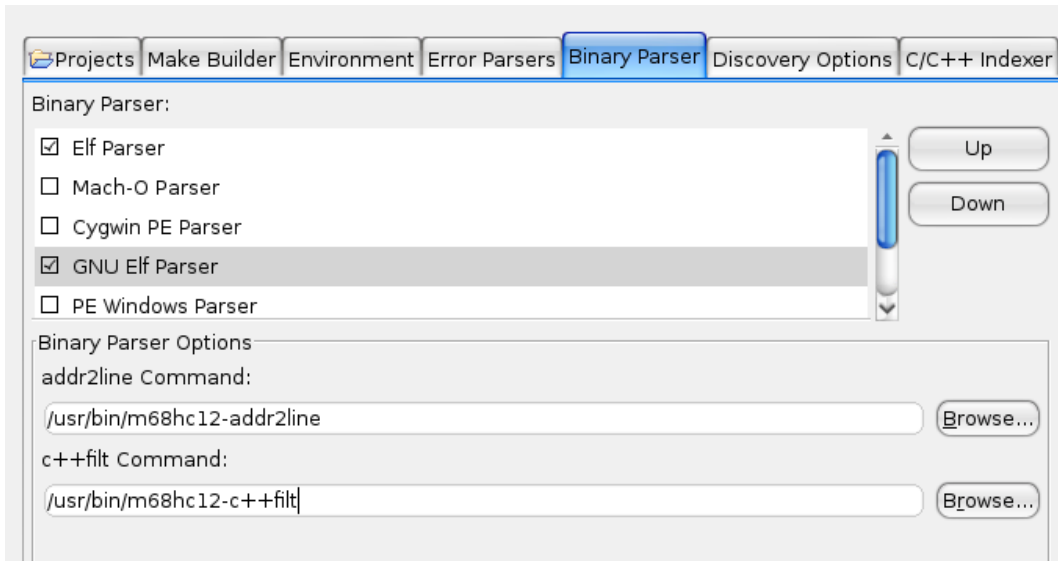
Using Eclipse CDT IDE for HCS12

Once you install and run Eclipse CDT, you can make a new project by referring to File->New->Project. You should choose Standard C Make Project or any option that lets you manually control the Make file. Make file should not be controlled automatically as you are going to change the compiler commands name and options to GCC for HC12 later.

Eclipse usually use a Workplace directory to save the projects files and settings. This folder can be easily changed to whatever you prefer. Also make sure that on the new project wizard, you choose Build on Resource Save and also Stop on First Build Error options. As you will see later, we use 'make all' to build our projects and 'make clean' to clean and erase all the compiled files.



In Binary Parser section make sure that you have chosen GNU ELF Parser and you IDE is pointing to HC11/12 addr2line and c++filt commands. Also make sure your Compiler Invocation Command is pointing to GCC for HC11/12. Since GCC for HC11/12 is a uniform compiler, you can decide your target platform through the compiling flags in Makefile and the command name of the compiler itself does not play any role there. Keep in mind, it is recommended to use Fast C/C++ Indexer option.



Linking External Helper Scripts to Eclipse CDT

Once you made a proper C project that can control the Make file manually, you can edit the Make file according to your needs and be able to compile your codes for HC12 platforms. But somehow you need to transfer, upload and debug the results. Obviously, Eclipse CDT does not have any feature on HC12, but you can link your helper scripts using External Tools options.

External Tools which is usually located under Run menu will give the option to be able to specify a command or script and set its working directory. You can choose that Eclipse will refresh your project files after running the scripts. Also you have the option to run the script in background. You can find helper scripts in 'bin' folder of each examples on <http://gcc-hcs12.com>.

In Linux, you can set the baud rate of a terminal by issuing 'stty' command. Also it is possible to echo a sentence (set of characters) to a character device like serial lines or even use 'cat' command to transfer the whole or part of a file to serial device. Sleep command is another useful tool to make sure your serial device got the command you sent to it. Most scripts that you see in the examples are using the above commands.

For each project you have to double check your helper scripts on External Tools section to make sure External Tools is pointing to the current project scripts and working directory. You can add as many scripts as you would like. Scripts can be organized using Organize Favorites option. Please refer to <http://gcc-hcs12.com> to see tutorial clips for more details.

Useful Hints on Eclipse CDT

One of the most useful features of Eclipse IDE is syntax and code assistance. By pressing CTRL+SPACE or ALT+\ you will code completion. The other feature is source code formatter and beautifier which can be done using CTRL+SHIFT+F. By going to Window->Preferences you can change the editor options such as colors, font size, line numbering and many more.

References

- <http://gcc-hcs12.com>
- <http://www.wikipedia.org>
- <http://m68hc11.servftp.org>
- <http://www.eclipse.org>
- <http://astyleclipse.sourceforge.net>