BUILDING M68HC11 PROJECTS WITH THE GNU ASSEMBLER

Microsoft's 64-Bit windows 7 breaks compatibility with several old DOS based applications, including AS11 and other assemblers. The GNU project's GCC compiler tools will work, however using them is a bit more complicated and will probably require you to modify your source code.

The GNU assembler requires a : after labels and either * or ; before comments. These were optional under as11.

This example uses BATCH files under windows to build an assembly language project for the Axiom CME-11E9-EVBU board using the GNU Compiler. We're using the GCC Compiler instead of the GAS Assembler to make this easier.

You can also use MAKE files for this if you prefer.

INSTALLING GNU FOR HC11/12

The GNU software for the HC11/12 is available on the Axiom HC11/12 support CD's under: **\Shareware\GNU**

It can also be downloaded here:
http://www.gnu-m68hc11.org/m68hc11_download.php

The batch files in this example assume it's installed at: **C:\user**
so the compiler and other utilities will be at: **C:\user\bin**

If you install to a different path you must modify the files: **build.bat** and **hc11_srec.bat** changing the "c:\user\bin\" to your path.

BUILDING M68HC11 PROJECTS WITH THE GNU ASSEMBLER

Included in this example is a batch file called: HELLO.BAT

```
:: uncomment one of these
:: Single Chip
call build SC_11e9evbu.ls hello.s
:: External RAM
::call build ExRAM_11e9evbu.ls hello.s
:: External EEPROM
::call build ExEEPROM_11e9evbu.ls hello.s
```

This will build a simple "Hello World" project for 3 different memory models:

- Single Chip
- External RAM
- External EEPROM

All 3 methods call the batch file: BUILD.BAT, which requires 2 parameters:

- A GNU linker script file
- An assembly source file

You can see in the HELLO.BAT example the only difference is the **Linker Script** file that's passes as a parameter. In this example the Single Chip version is uncommented, which is the target that will be build when HELLO.BAT is executed. Try running it.

To build one of the other targets, simply add :: before the line that uses SC_11e9evbu.ls and remove the :: before the other one.

MODIFYING THE BUILD FILE

This example was created to make the build process as simple as possible using only 1 source file. However your projects will probably require multiple source files. For this you'll need to modify the BUILD.BAT file:

```
:: Example Build Batch file for a GCC Assembler project
::    %1 = linker script file name
::    %2 = assembly source file name
@echo off

:: Check if input files specified
IF [%1]==[] GOTO Syntax
IF [%2]==[] GOTO Syntax

:: delete previously generated output files
call cleanup.bat

:: assemble files here
c:\usr\bin\m6811-elf-gcc -m68hc11 -c -O1 -g -Wa,--gdwarf2 -o %~n2.o %2 2> error.out
@IF ERRORLEVEL 1 GOTO End

:: link files here
c:\usr\bin\m6811-elf-gcc -m68hc11 -Wl,--script,%1 -nostartfiles -o %~n2.elf %~n2.o 2>> error.out
@IF ERRORLEVEL 1 GOTO End

:: convert elf output to other types
call hc11_srec.bat %2
EXIT /B

:End
start NOTEPAD error.out
EXIT /B

:Syntax
echo Usage: build linkerscript filename
```

This batch file takes a linker script and a source file from the command line. You'll probably want to make a copy of this batch file and modify it to add your file names directly, in place of %2.

After verifying input parameters, CLEANUP.BAT is called to delete any output files generated from previous builds. Highly recommended you do this to save yourself debugging grief.

The source code is then assembled with the first call to m6811-elf-gcc, then linked with the 2nd call.

If any errors were found in assembly or linking they are sent to a text file called ERROR.OUT. The batch file will then jump to the end and displays the errors in windows Notepad.

If no errors were found, HC11_SREC.BAT is called to convert the .ELF output file to a Motorola SREC fie, as well as other useful formats. Look at these output files in your text editor, they will come in handy during debugging.

To build a project using multiple source files, you'll need to call GCC to assemble each source file to object (.o) files individually. Then call GCC to link all the object files into one final executable (.elf) file.

For example, if we were to add a serial.s file to our hello.s example we could do it like this:

```
:: delete previously generated output files
call cleanup.bat

:: assemble files here
c:\usr\bin\m6811-elf-gcc -m68hc11 -c -O1 -g -Wa,--gdwarf2 -o serial.o serial.s 2> error.out
@IF ERRORLEVEL 1 GOTO End
c:\usr\bin\m6811-elf-gcc -m68hc11 -c -O1 -g -Wa,--gdwarf2 -o hello.o hello.s 2> error.out
@IF ERRORLEVEL 1 GOTO End

:: link files here
c:\usr\bin\m6811-elf-gcc -m68hc11 -Wl,--script,%1 -nostartfiles -o hello.elf hello.o serial.o 2>> error.out
@IF ERRORLEVEL 1 GOTO End

:: convert elf output to other types
call hc11_srec.bat %2
EXIT /B

:End
start NOTEPAD error.out
```

If we saved this as HELLO2.BAT, we could build it for External RAM by calling:

```
HELLO2.BAT ExRAM_11e9evbu.ls
```

MORE INFORMATION

The GNU Software Tools are extremely powerful and provide MANY more options than simple assemblers like AS11. Most of the online examples you'll find are for C code or mixed C and Assembly however.

For more information visit the projects website at:
http://www.gnu-m68hc11.org/wiki/index.php/Main_Page

Documentation for the GCC tools used in this example can be found here:
http://gcc.gnu.org/onlinedocs/gcc-3.3.6/gcc/

And here:
http://sourceware.org/binutils/docs/

There is also an active support group for this HC11/12 port here:
http://tech.groups.yahoo.com/group/gnu-m68hc11/