Getting Started with GCC for Motorola 68HC12 Using DRAGON12 and MiniDRAGON+

Application Note AN0004

Author : Lin Zhao
Date : August 12, 2003

Abstract: This application note shows you how to write a “Hello World” C program, how to have it compiled by using EmbeddedGNU and GCC, and how to load and run the application using EmbeddedGNU. This program was tested on DRAGON12 and MiniDRAGON+ boards. It should also work on most HC12 and HCS12 boards.

In this application note, the SCI registers of MC9S12DP256 were defined in the C program. Therefore, any include, makefile or configuration file of GEL was not used. The I/O registers were represented by a volatile array, *io_reg*. In this example, the *io_reg* address was defined in the “memory.x” file.

The following steps are applicable to both DRAGON12 and MiniDRAGON+. It is assumed that you have installed Eric Engler’s EmbeddedGNU IDE version 08.a on your C drive at “c:\EmbeddedGNU\” and have installed GNU C for 68HC11/68HC12 on your C drive at “c:\usr\”.

1. Start EmbeddedGNU IDE:

Navigate to “c:\EmbeddedGNU\egnu08a”, double click on “EmbeddedGNU” application to start EmbeddedGNU IDE.

2. Edit C Source File:

Choose File -> New Source File from the menu system. You can either type in your C code as shown in Listing_1 or use the C file for this application note that can be downloaded from Wytec Company’s web site. Then, save it to your folder, such as “c:\EmbeddeGNU\hello12\hello.c”.

3. Create Your Project:

Choose File -> New Project from the menu system. Type in your project name in the New Project dialog frame, say “hello12”, and click OK. Navigate to the folder where your C source file is saved (c:\EmbeddeGNU\hello12\) and click “Save”.

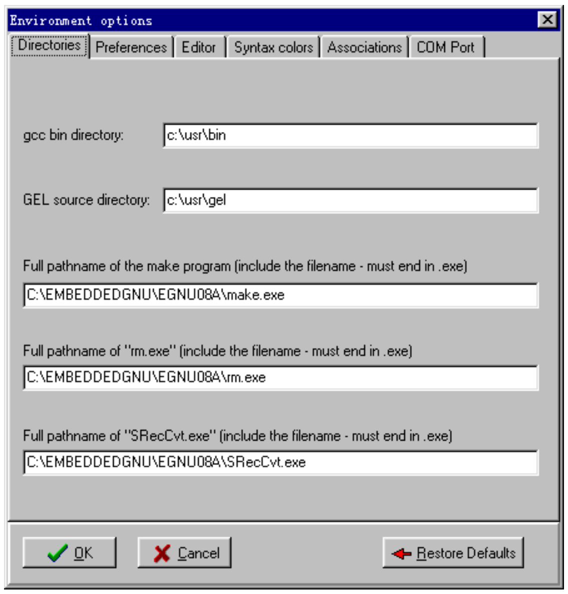
In Project Options dialog frame, select “Dragon12” under Hardware Profile and Click OK.

4. Add C Source File to Your Project:

In the project window, which is on the left side of the IDE window, right click on the project name, i.e. hello12, and choose “Add to Project”. Select your C source file (“hello.c” here) and click “Open”. The C source file is added to your project.

5. Setup Options:

Choose Options -> Environment Options from the menu system. Under “Directories” tab, use all the default setting as shown in Figure 1. Then, click on “COM Port” tab. Select the serial port you want to use, such as COM1 and set COM options as 9600, N, 8, 1 as shown in Figure 2. Click OK to save the options.



gcc bin directory: c:\usr\bin

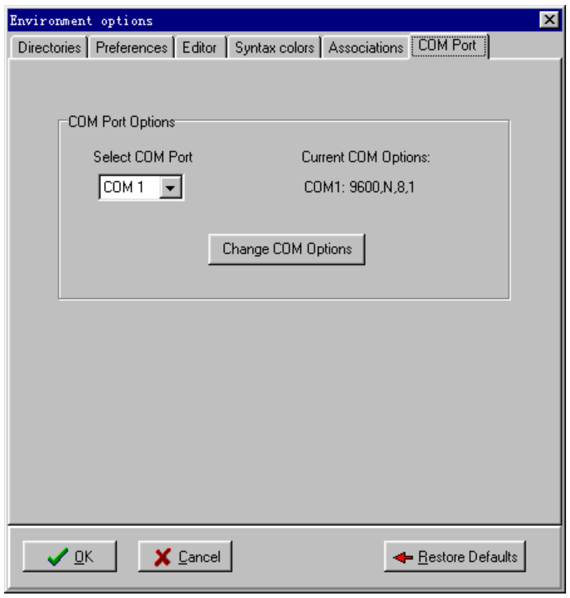
GEL source directory: c:\usr\gel

Full pathname of the make program (include the filename - must end in .exe)
C:\EMBEDDEDGNU\EGNU08A\make.exe

Full pathname of "rm.exe" (include the filename - must end in .exe)
C:\EMBEDDEDGNU\EGNU08A\rm.exe

Full pathname of "SRecCvt.exe" (include the filename - must end in .exe)
C:\EMBEDDEDGNU\EGNU08A\SRecCvt.exe

Figure 1 Settings of Directories



COM Port Options

Select COM Port: COM 1

Current COM Options: COM1: 9600,N,8,1

Figure 2 Serial Port Configuration

6. Build Your Project:

Choose Build -> Make from the menu system to build your project. You will get 3 warnings and one error in the Compiler window that is located in the lower half part of the IDE window. The warnings are not critical. But we must deal with the error.

The reason for getting an error is that we have not defined the address of I/O registers array, *io_reg,* in the “memory.x” file. As stated at the beginning of this application note, the SCI registers of MC9S12DP256 are defined in the C program. Any include, makefile or configuration file of GEL is not used. The I/O registers are represented by a volatile array, *io_reg*. The *io_reg* address is defined in the “memory.x” file.

Now, open the “memory.x” file that is located in your project folder, c:\EmbeddeGNU\hello12\, by using any text editor. Add

```
PROVIDE (io_reg = 0x0000);
```

at the end of “memory.x” file and save the file.

Rebuild your project by choosing Build -> Make. You should have your project built with no problem.

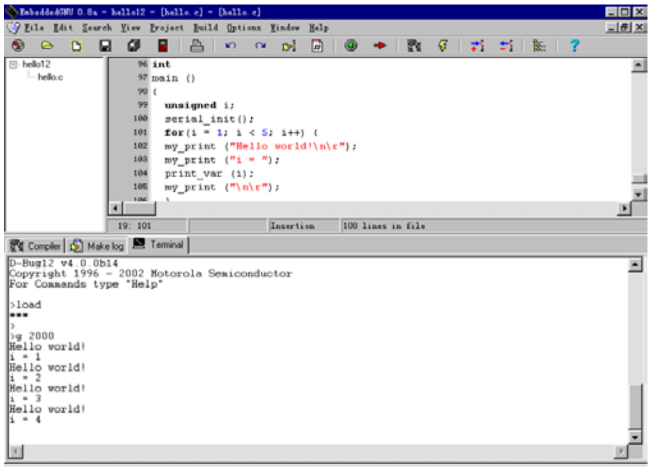
7. Load and Run:

Click on the “Terminal” tab, which is located in the lower half part of the IDE window, to activate the terminal window.

Press the reset button on your DRAGON12 or MiniDRAGON+ board once. In the terminal window, you should see that D-Bug12 is started. (If you can see nothing in the terminal window after you pressed the reset button, hit “Enter” on your keyboard.)

Type “load” in the terminal window and hit “Enter” on your keyboard. Choose Build -> Download from the menu system. Select the S19 file, i.e. hello12.s19, and click “Open” to have the S19 file downloaded to your board.

After the S19 file is downloaded, type “g 2000” in the terminal window and hit “Enter” on your keyboard. You should get the same result as shown in Figure 3.


EmbeddedGNU 0.8a - hello12 - [hello.c] - [hello.c]
File Edit Search View Project Build Options Window Help
hello12
hello.c
19: 101
Insertion
100 lines in file
Compiler
Make log
Terminal

```
int
main ()
{
  unsigned i;
  serial_init();
  for(i = 1; i < 5; i++) {
  my_print ("Hello world!\n\r");
  my_print ("i = ");
  print_var (i);
  my_print ("\n\r");
```

```
D-Bug12 v4.0.0b14
Copyright 1996 - 2002 Motorola Semiconductor
For Commands type "Help"

>load
***
>
>g 2000
Hello world!
i = 1
Hello world!
i = 2
Hello world!
i = 3
Hello world!
i = 4
```

Figure 3 Load and Run Hello World

Note:

1. “2000” is the start address of the program. So, you have to type “g 2000” to run the program. You can find the start address in the .DMP file.
2. This program displays “Hello World” four times. “i” is used as a counter. Refer to Listing_1 for detail.

Listing_1. C Source File for "Hello World"

```
/*
* Name:   hello.c
* Author: Lin Zhao
* Date:   8/10/2003
*
* Description: Simple Hello World for DRAGON12 and MiniDRAGON+ that
*                   operates in EVB mode.
*                   This example program prints "Hello World!" on the
*                   9S12DP256 serial line. since it is self contained, it
*                   does not use any include, makefile or configuration
*                   file of GEL.
*
*                  This program was compiled and tested using EmbeddedGNU
*                  and GCC for 68HC12.
*/

/* define SCI registers */
#define SCI0BDH 0xC8    /* SCI Baud register        */
#define SCI0BDL 0xC9
#define SCI0CR1 0xCA    /* SCI Control register 1 */
#define SCI0CR2 0xCB    /* SCI Control register 2 */
#define SCI0SR1 0xCC    /* SCI Status register       */
#define SCI0DRL 0xCF    /* SCI Data register        */

/* Flags of the SCI0CR2 register.  */
#define TE          0x08      /* Transmitter Enable Bit */
#define RE          0x04      /* Receiver Enable bit      */

/* Flags of the SCI0SR1 register.  */
#define TDRE      0x80      /* Transmit Data Register Empty Flag */

#define BAUD 163  /* SCI Baud Rate 9600 */

/*
The I/O registers are represented by a volatile array.
In this example, the io_reg address is defined in the
'memory.x' file.
*/
extern volatile unsigned char io_reg[];

/* Prints variables on the serial device.  */
static void
print_var (unsigned i)
{
  char buf[15];
  char* p;
  unsigned char c;
  unsigned int value;

  p = &buf[15];
  *--p = 0;
  value = i;
  do {
    c = value % 10;
    value = value / 10;
    *--p = c + '0';
  } while (value != 0);

  my_print (p);

}

/* Send a character on the serial line */
static inline void
putchar (char c)
{
  /* Wait until the Transmit Data Register is empty  */
  while (!(io_reg[SCI0SR1] & TDRE))
    continue;

  io_reg[SCI0DRL] = c;
  io_reg[SCI0CR2] |= TE;
}

void
my_print (const char *msg)
{
  while (*msg != 0)
    putchar (*msg++);
}

void
serial_init (void)
{
  /* Set SCI baudrate at M6812_DEF_BAUD    */
  io_reg[SCI0BDL] = BAUD;

  /* Set character format 1 start, 8-bits, 1 stop */
  io_reg[SCI0CR1] = 0;

  /* Enable receiver and transmitter.  */
  io_reg[SCI0CR2] = TE | RE;
}

int
main ()
{
  unsigned i;
  serial_init();
  for(i = 1; i < 5; i++) {
  my_print ("Hello world!\n\r");
  my_print ("i = ");
  print_var (i);           /* Print the content of a variable */
  my_print ("\n\r");
  }
  return 0;
}
```

References:

1. Eric Engler. *Embedded Tools*. http://www.geocities.com/englere_geo/

2. "Using the GNU Development Tools for 68HC11 and 68HC12," http://stephane.carrez.free.fr/doc/guide.html

3. *MC9S12DP256 Advanced Information, Revision 1.1*. December 1, 2000. Motorola.