

Getting Started with GCC for Motorola 68HC11 Using EVBPlus2 with Buffalo

Application Note AN0005

Author : Lin Zhao

Date : September 2, 2003

Abstract: This application note shows you how to write a “Hello World” C program, how to have it compiled using EmbeddedGNU and GCC, and how to load and run the application using EmbeddedGNU. This program was tested on the EVBPlus2 and FOX11 boards. It should also work on most HC11 evaluation boards that boot from Buffalo.

Since the MC6850 UART, an Asynchronous Communication Interface Adapter (ACIA), was directed to P2 on the EVBPlus2 or FOX11 board, we have to redirect the SCI serial output from 68HC11 microcontroller to P2.

In this application note, the ACIA registers of 68hc11 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.

If you are using a FOX11 board, please replace the definitions of ACIA in Listing_1 with:

```
/* define ACIA */  
#define ACIA_CTRL 0x2400  
#define ACIA_STAT 0x2400  
#define ACIA_XMIT 0x2401  
#define ACIA_RCV 0x2401
```

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\gcc_hc11\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 "hello11", and click OK. Navigate to the folder where your C source file is saved (c:\EmbeddeGNU\gcc_hc11) and click "Save".

In Project Options dialog frame, select "EVPlus2" 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. hello11, 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.

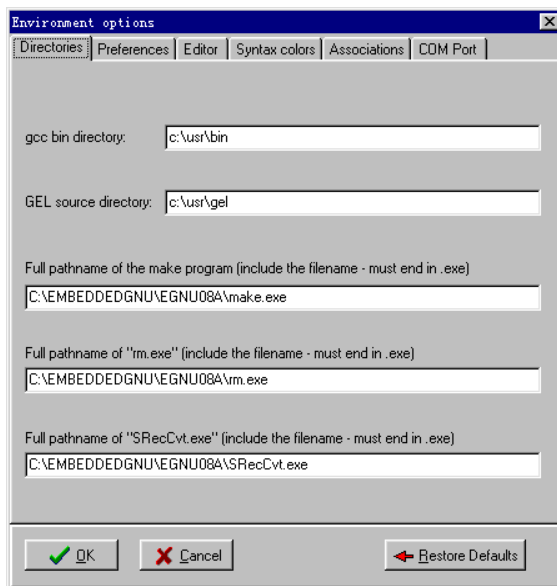


Figure 1 Settings of Directories

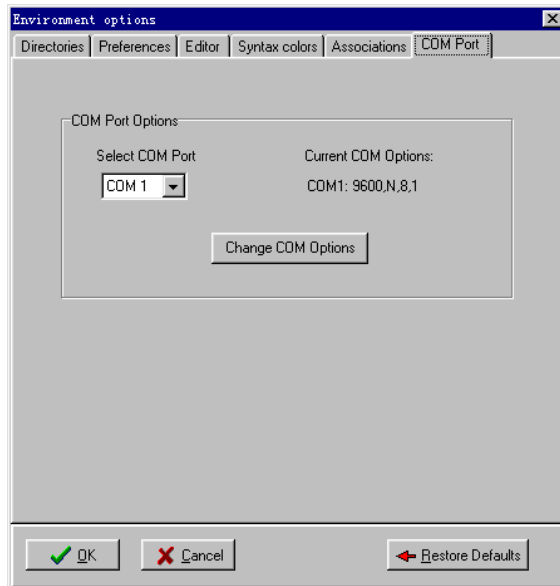


Figure 2 Serial Port Configuration

6. Build Your Project:

Choose Build -> Make from the menu system to build your project. You will get one error in the Compiler window that is located in the lower half part of the IDE window.

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 ACIA registers of EVBPlus2 were 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 was defined in the “memory.x” file.

Now, open the “memory.x” file that is located in your project folder, c:\EmbeddeGNU\gcc_hc11\, 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.

If you still get errors after compiling, check your “memory.x” file and make it have the same contents as in Listing_2.

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 EVBPlus board once. In the terminal window, you should see that Buffalo monitor is started. (If you can see nothing in the terminal window after you pressed the reset button, hit “Enter” on your keyboard.)

Type “load t” in the terminal window and hit “Enter” on your keyboard. Choose Build -> Download from the menu system. Select the S19 file, i.e. hello11.s19, and click “Open” to have the S19 file downloaded to your board.

After the S19 file is downloaded, type “g c000” in the terminal window and hit “Enter” on your keyboard. You should get the same result as shown in Figure 3.

Note:

1. “c000” is the start address of the program. So, you have to type “g c000” 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.

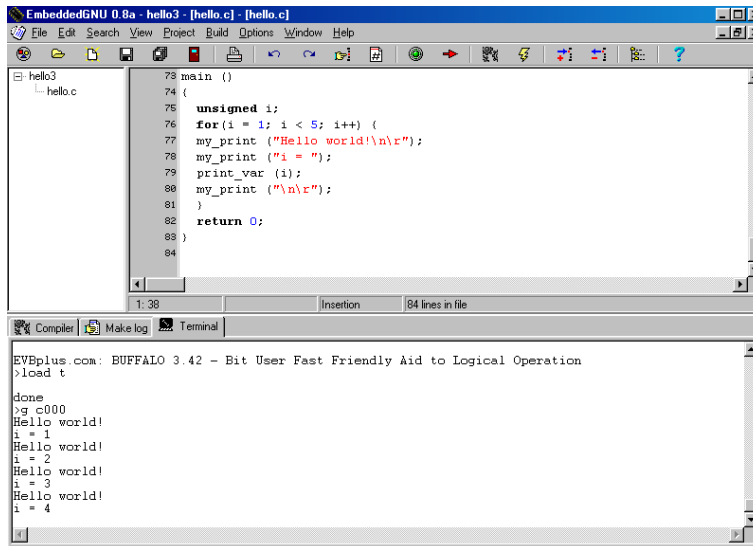


Figure 3 Load and Run Hello World

Listing 1. C Source File for “Hello World”

```

/*
 * Name: hello.c
 * Author: Lin Zhao
 * Date: 9/2/2003
 *
 * Description: Simple Hello World for EVBPlus2 board. This example
 *              program prints "Hello World!" 5 times and variable "i"
 *              5 times.
 *              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 68HC11/12.
 */

/* define ACIA */
#define ACIA_CTRL 0x6400
#define ACIA_STAT 0x6400
#define ACIA_XMIT 0x6401
#define ACIA_RCV 0x6401

/*
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[];

/* Send a character on ACIA */
static inline void

```

```

putchar (char c)
{
    /* Wait until the Transmit Data Register is empty */
    while (!(io_reg[ACIA_STAT] & 0x02));

    io_reg[ACIA_XMIT] = c;
}

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

/* Prints variables on the serial device. */

static void
print_var (unsigned i)
{
    char buff[15];
    char* p;
    unsigned char c;
    unsigned int value;

    p = &buff[15];
    *--p = 0;
    value = i;

    do {
        c = value % 10;
        value = value / 10;
        *--p = c + '0';
    } while (value != 0);

    my_print (p);
}

int
main ()
{
    unsigned i;
    for(i = 1; i < 5; i++) {
        my_print ("Hello world!\n\r");
        my_print ("i = ");
        print_var (i);
        my_print ("\n\r");
    }
    return 0;
}

```

Listing 2. “memory.x” file

```
MEMORY
{
  page0 (rwx) : ORIGIN = 0x0, LENGTH = 256
  text (rx) : ORIGIN = 0xC000, LENGTH = 6000
  data : ORIGIN = 0x100, LENGTH = 800
}

/* Setup the stack on the top of the data internal ram (not used). */
PROVIDE (_stack = 0x01FF-1);
PROVIDE (io_reg = 0x0000);
```

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>