Building C programs for the *Dragon12*

Version 1.0 – Thu Feb 19 22:16:12 EST 2004

The HC12 GNU C Compiler (*gcc-mc68hc1x*) is able to compile programs for the CPU12, but the *Dragon12* environment requires a bit of custom home-grown help.

1 Compiling and linking

gcc compiles source code to an intermediate format, where the binary code is not committed to a particular set of addresses. The final phase of compilation, called linking, resolves the addresses for variables, constants and text[1], then produces the actual binary file that will be loaded and run on the CPU.

The linking process attempts to connect program variable and function names with real memory addresses. Most of the variables are allocated automatically to data or stack memory. However, it is often necessary to dictate the exact address of variables such as the initial stack pointer, I/O registers and the interrupt vector table.

2 *Dragon12* important addresses

The addresses that must be fixed in memory on the *Dragon12* are described here. The compiler reads the resource file `memory.x` to locate those addresses in memory (figure 1). The `PROVIDE` statement defines the addresses.

2.1 I/O registers

The hardware control and status registers for the MC9S12DP256B MCU at the heart of the *Dragon12* start at address 0x0000, and the block is 0x400 (1024_d) bytes long.

[1]The program instructions executed by the CPU is called "text" because the word "code" is overused and potentially confusing.

```
OUTPUT_FORMAT("elf32-m68hc12", "elf32-m68hc12",
            "elf32-m68hc12")
OUTPUT_ARCH(m68hc12)
ENTRY(_start)
SEARCH_DIR(C:\usr\lib\gcc-lib\m6811-elf\3.0.4\m68hc12\mshort)

MEMORY
{
  ioports (!x)  : org = 0x0000, l = 0x400
  eeprom  (!i)  : org =  0x400, l = 0xc00
  page0   (rwx) : org = 0x1000, l  = 0x100
  data    (rwx) : org = 0x1000, l = 0x1000
  text    (rx)  : org = 0x2000, l = 0x2000
}

PROVIDE (_stack = 0x2000);
PROVIDE (_io_reg = 0x0000);
PROVIDE (_ramvectors = 0x3e00);
```

Figure 1: Linker resource `memory.x` defining the *Dragon12* memory layout

2.2 Stack

The initial value of the stack pointer is 0x2000, corresponding to the start address of the text segment, as defined in the *Dragon12* profile.

2.3 Interrupt vector table

The interrupt service routine addresses are loaded into a RAM table maintained by D-Bug12. In D-Bug12 Ver.4, the table is located at 0x3e00.

3 *Dragon12* hardware registers

The hardware control and status registers for the MCU are described in the *MC9S12DP256B Device User Guide*. The registers are addressed in C as an array `_io_reg` of type `unsigned char`.

3.1 Hardware register header file

The offsets to each register are given in the header file `reg9s12c.h`. The file was translated from the assembler header file `reg9s12.h` supplied in the *Dragon12* support package (note the letter 'c' on the end of the C version).

At the top of each file that needs to address hardware registers, add the line:

```
#include "reg9s12c.h"
```

The file must be in the same directory as the C source file[2].

3.2 Declaring the I/O register array

At the top of each file that needs to address hardware registers, add the line:

```
extern volatile unsigned char _io_reg[];
```

The statement tells the compiler that the array (`[]`) named `_io_reg` contains 8-bit unsigned values (`unsigned char`). The keyword `extern` means that the exact address in memory of the array will be supplied outside this particular file, and the keyword `volatile` prevents the optimiser from removing repetitive read/write operations because the value of the register is outside the program's direct control.

3.3 Writing to a register

To write a value of type `unsigned char` to a register, assign the value to the array element. For example, writing the port B register (PORTB) to turn on the LEDs:

```
unsigned char leds; /* declare before using */
...
_io_reg[portb] = leds;
```

`portb` is the offset from the base of the register block and is supplied by the header file `reg9s12c.h`. The register names are the same as those in the Device User Guide, except they have been changed to lowercase.

[2]The angle brackets (`< >`) used to include standard header files, such as

```
#include <stdio.h>
```

are interpreted as "search the system-wide directories". The standard quotation marks (`"`) mean "search the current directory" .

3.4 Reading from a register

To copy the current value of a register, assign the array element's value to a variable of type `unsigned char`. For example, reading the port H input register (PTIH) to capture the state of the *Dragon12* DIP switches S1:

```
unsigned char switch; /* declare before using */
...
switch = _io_reg[ptih];
```

`ptih` is the offset from the base of the register block and is supplied by the header file `reg9s12c.h`.

4 Interrupts

This section details the mechanics of setting up for interrupts on the *Dragon12* under D-Bug12. For additional instructions on interrupts in C, see EMBS 6100 lab 3 (Interrupts).

4.1 RAM interrupt vector table

D-Bug12 maintains an array of interrupt vectors in RAM to get around the need to program the on-chip flash EPROM every time a new program is downloaded. The offsets are defined in the header file `dbug12c.h`. The names are based on the list given in the D-Bug12 Ver.4 reference guide (p.86).

4.2 Interrupts header file

Include the interrupts definition file with the statement

```
#include "dbug12c.h"
```

The RAM based interrupt table is defined inside `dbug12c.h` as the array

```
extern volatile Address _ramvectors[];
```

where the type `Address` is a synonym for pointer to type `char`. Once you have included `dbug12c.h`, you do not need to further declare or define the array in your program.

5 References

D-Bug12 Ver.4 reference guide
`http://opentech.durhamc.on.ca/bertrandl/embs6100/resources/DB12RG4.pdf`

EMBS 6100 C programming support kit
`http://opentech.durhamc.on.ca/bertrandl/embs6100/resources/embs6100cprograms.zip`

All referenced Motorola PDF documents may be downloaded from the Motorola Semiconductors documentation site for the MC9S12DP256B 16-bit microcontroller:
`http://e-www.motorola.com/webapp/sps/site/prod_summary.jsp?code=MC9S12DP256B&nodeId=0162468636K100`