- **Introduction to Programming the 9S12 in C**
- Huang Sectons 5.2 and 5.3
  - Comparison of C and Assembly programs for the HC12
  - How to compile a C program using the GNU-C compiler
  - Using pointers to access the contents of specific addresses in C
  - Using the `iodp256.h` header file

• There are a lot of registers (such as PORTA and DDRA) which you will use when programming in C. Rather than having to define the registers each time you use them, you can include a header file for the HC12 wihch has the registers predefined. The CD in the textbook has such a header file, called hcs12.h. You can find the complete file on the EE 308 homepage.
Here are a few entries from that file:

```
#define        IOREGS_BASE       0x0000
#define        _IO8(off)         *(unsigned char volatile *)(IOREGS_BASE + off)
#define        _IO16(off)        *(unsigned short volatile *)(IOREGS_BASE + off)
#define        PORTA             _IO8(0x00)    //port a = address lines a8 –
 #define PORTB                  _IO8(0x01)    //port b = address lines a0 –
 #define DDRA                   _IO8(0x02)    //port a direction register
#define DDRB                    _IO8(0x03)    //port b direction register
```

Here is a program which uses hcs12.h to write a 0x55 to PORTA:

```
#include "hcs12.h"
main()
{
        DDRA = 0xff;          // Make PORTA output
        PORTA = 0x55;         // write a 0x55 to PORTA
        asm(" swi");
}
```

Some C basics

• Every C program has a function main()
  – The simplest C program is:

```
main()
{
}
```

– Our compiler ends a program by executing an infinite loop – the program never returns to DBug-12. In order to return to DBug-12, include the swi assembly language instruction. Here is how to do that:

```
main()
{
        asm(" swi");
}
```

• Every statement ends with a semicolon

```
x = a+b;
```

• Comment starts with /* ends with */

```
/* This is a comment */
```

• Simple program – increment Port A

```
#include "hcs12.h"
main()
{
        DDRA = 0xff;            /* Make PORTA output */
        PORTA = 0;              /* Start at 0 */
        while(1)                /* Repeat forever */
        {
                PORTA = PORTA + 1;
        }
}
```

• Data Types:

8-bit	16-bit
unsigned char	unsigned int
signed char	signed int

• Need to declare variable before using it:

```
signed char c;
unsigned int i;
```

• Can initialize variable when you define it:

```
signed char c = 0xaa;
signed int i = 1000;
```

– You tell compiler it you are using signed or unsiged numbers; the compiler will figure out whether to use BGT or BHI

• Arrays:

```
unsigned char table[10]; /* Set aside 10 bytes for table */
```

– Can refer to elements table[0] through table[9]

- Can initialize an array when you define it:

```
unsigned char table[] = {0xaa, 0x55, 0xa5, 0x5a};
```

• Arithmetic operators:

```
+ (add)          x = a+b;
- (subtract)     x = a-b;
* (multiply)     x = a*b;
/ (divide)       x = a/b;
% (modulo)       x = a%b;       (Remainder on divide)
```

• Logical operators

```
& (bitwise AND)                 y = x & 0xaa;
| (bitwise OR)                  y = x | 0xaa;
^ (bitwise XOR)                 y = x ^ 0xaa;
<< (shift left)                 y = x << 1;
>> (shift right)                y = x >> 2;
~ (1's complement)              y = ~x;
- (2's complement - negate)     y = -x;
```

Check for equality - use ==

```
if (x == 5)
```

Check if two conditions true:

```
if ((x==5) && (y==10))
```

Check if either of two conditions true:

```
if ((x==5) || (y==10))
```

• Assign a name to a number

```
#define COUNT 5
```

• Include a header file (such as hcs12.h):

```
#include "hcs12.h"
```

• Declare a function: Tell what parameters it uses, what type of number it returns:

```
int read_port(int port);
```

• If a function doesn't return a number, declare it to be type void

```
void delay(int num);
```

Hello, World!

• Here is the standard "hello, world" program:

```
#include <stdio.h>
main()
{
        printf("hello, world\r\n");
}
```

• To write the "hello, world" program, you need to use the printf() function.

• The printf() function is normally a library function

• Our compiler does not have a library which includes the printf() function.

• DBug-12 has a built-in printf, which you can access in the following way:

```
#include "DBug12.h"
main()
{
        DB12FNP->printf("hello, world\r\n");
        asm(" swi");
}
```

• The above program is about 40 bytes long.

• Note that the DBug-12 printf() does not work for floating point numbers.

• You can access a few other standard C functions through DBug-12. Look at the DBug12.h include file (on the EE 308 web page) to see which ones.

Setting and Clearing Bits using Assembly and C

• To put a specific number into a memory location or register (e.g., to put 0x55 into PORTA):

- In assembly:

```
ldaa #$55
staa PORTA
```

- In C:

```
PORTA = 0x55;
```

• To set a particular bit of a register (e.g., set Bit 4 of PORTA) while leaving the other bits unchanged:

– In assembly, use the bset instruction with a mask which has 1's in the bits you want to set:

```
bset PORTA,#$10
```

– In C, do a bitwise OR of the register with a mask which has 1's in the bits you want to set:

```
PORTA = PORTA | 0x10;
```

• To clear a particular bit of a register (e.g., clear Bits 0 and 5 of PORTA) while leaving the other bits unchanged:

– In assembly, use the blcr instruction with a mask that has 1's in the bits you want to clear:

```
bclr PORTA,#$21
```

– In C, do a bitwise AND of the register with a mask that has 0's in the bits you want to clear:

```
PORTA = PORTA & 0xDE;
```

or

```
PORTA = PORTA & ~0x21;
```

Waiting for a bit to be set or cleared in Assembly and C

• You often have to wait for an event to occur before taking some action.

• For example, wait for the wash cycle to finish before starting the rinse cycle.

• Usually, when an event occurs, a bit is either set or cleared.

• Here is how to wait until Bit 3 of PORTB is set:

– In assembly:

```
l1:  brclr    PORTB,#$08,l1
```

– In C:

```
while ((PORTB & 0x08) == 0) ;
```

• Here is how to wait until Bit 3 of PORTB is cleared:

– In assembly:

–

```
l1:  brset    PORTB,#$08,l1
```

– In C:

```
while ((PORTB & 0x08) != 0) ;
```

Sum the odd 8-bit numbers in an array

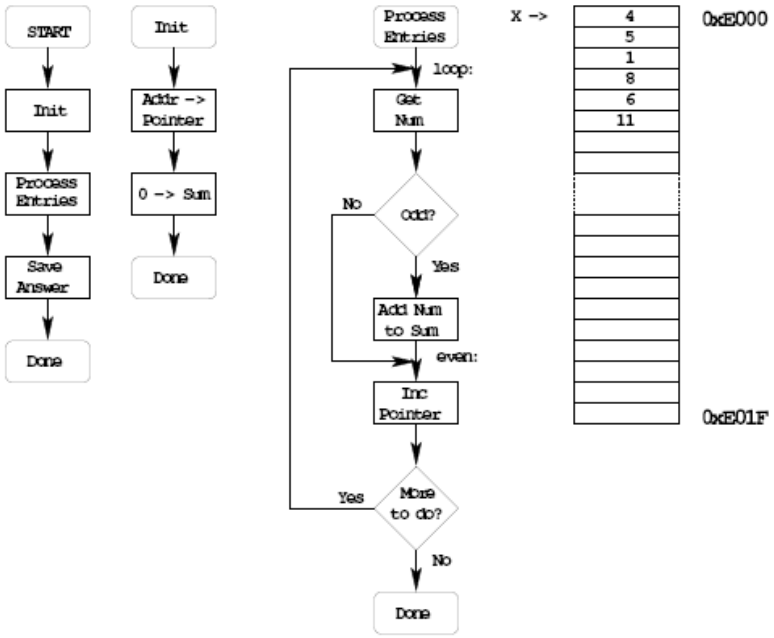
• Write a program to sum all the odd numbers in an array of data.

• The numbers in the array should be treated as unsigned 8-bit numbers.

• The array starts at address 0xE000 and ends at address 0xE01F.

• This is the same program which was done in assembly language on Feb.

SUM ODD 8−BIT NUMBERS IN ARRAY FROM 0xE000 TO 0xE01f



Convert to C

How to check if odd?
Divide by 2, if REM = 1 odd
Modulo (%) in C returns REM

```
main(){
        ptr = (unsigned char *) 0xe000;
        sum = 0;
        do{
                x = *ptr;
                if ((x % 2) = = 1) {
                        sum = sum + x;
                }
                ptr = ptr + 1;
        }
        while (ptr <= (unsigned char *) 0xe01f);
}
```

```
/* Program to sum the odd numbers in an array
* The numbers are unsigned characters
* The array starts at address 0xE000 and
* is 0x20 bytes long
*/
#include "DBug12.h"

#define     START 0xE000
#define     LEN 0x20
#define     END (START+LEN-1)

main()
{
        unsigned int sum;       /* Keep the running sum (need 16-bit number) */
        unsigned char *ptr;     /* Pointer to array element */
        unsigned char x;        /* Character from array */

        ptr = (unsigned char *) START;
        sum = 0;
        do
        {
                x = *ptr;               /* Get entry */
                if ((x % 2) == 1)       /* Is number odd? */
                {
                        sum = sum + x; /* Odd: add to sum */
                                        /* C automatically makes x 16-bits */
                }
                ptr = ptr + 1;          /* Advance to next */
        }
        while (ptr <= (unsigned char *) END); /* Done? */
        DB12FNP -> printf("sum = %d\r\n",sum);
        asm(" swi");
}
```

A software delay

• To enter a software delay, put in a nested loop, just like in assembly.
– Write a function delay(num) which will delay for num milliseconds

```
void delay(unsigned int num)
{
        unsigned int i;
        while (num > 0)
        {
```

```
        i = XXXX;
                                /* ------------------------------ */
        while (i > 0)           /*                                */
        {                       /* Want inner loop to delay */
                i = i - 1;      /* for 1 ms                       */
        }                       /*                                */
                                /* ------------------------------ */
        num = num - 1;
    }
}
```

• What should XXXX be to make a 1 ms delay?

• Look at assembly listing generated by compiler:

```
                0000206f <delay>:
                    206f:   34              pshx
                    2070:   7c 10 02        std   1002 <_.z>
                    2073:   27 1e           beq   2093 <delay+0x24>
                    2075:   18 00 80 03     movw  #XXX <__bss_size+0xXXX>, 0,SP
                    2079:   e8
                    207a:   ec 80           ldd   0,SP
                    207c:   27 0a           beq   2088 <delay+0x19>
                   ------------------------------------------------
inner           | 207e:   ed 80           ldy   0,SP
loop            | 2080:   1a 5f           leax  -1,Y
takes           | 2082:   6e 80           stx   0,SP
13 cycles       | 2084:   ec 80           ldd   0,SP
                | 2086:   26 f6           bne   207e <delay+0xf>
                   ------------------------------------------------
                    2088:   fc 10 02        ldd   1002 <_.z>
                    208b:   c3 ff ff        addd  #ffff <_stack+0xc3ff>
                    208e:   7c 10 02        std   1002 <_.z>
                    2091:   26 e2           bne   2075 <delay+0x6>
                    2093:   30              pulx
                    2094:   3d              rts
```

• Inner loop takes 13 cyles.

• One millisecond takes 24,000 cycles

(24,000,000 cycles/sec × 1 millisecond = 24,000 cycles)

• Need to execute inner loop 24,000/13 = 1,846 times to delay for 1 millisecond

```
void delay(unsigned int num)
{
        unsigned int i;
        while (num > 0)
        {
                i = 1846;
                                /* ------------------------------- */
                while (i > 0)   /*                                 */
```

```
                {                       /* Inner loop takes 13 cycles */
                i = i - 1;              /* Execute 1836 times to      */
                }                       /* delay for 1 ms             */
                                        /* ------------------------------ */
                num = num - 1;
        }
}
```

Program to increment LEDs connected to PORTA, and delay for
50 ms between changes

```
#include "hcs12.c"
#define D_1MS (24000/13)            // Inner loop takes 13 cycles
                                    // Need 24,000 cycles for 1 ms

void delay(unsigned int num);
main()
{
        DDRA = 0xff;                /* Make PORTA output */
        PORTA = 0;                  /* Start with all off */
        while(1)
        {
                PORTA = PORTA + 1;
                delay(50);
        }
}

void delay(unsigned int num)
{
        unsigned int i;
        while (num > 0)
        {
                i = D_1MS;
                while (i > 0)
                {
                        i = i - 1;
                }
                num = num - 1;
        }
}
```

Program to display a particular pattern of lights on PORTA

```
#include "hcs12.c"
#define D_1MS (24000/13)    // Inner loop takes 13 cycles
                            // Need 24,000 cycles for 1 ms
#define TABLEN 8

void delay(unsigned int num);
main()
{
        const char table[] = {0x80,0x40,0x20,0x10,
                              0x08,0x04,0x02,0x01};
        int i;

        DDRA = 0xff;          /* Make PORTA output */
        PORTA = 0;            /* Start with all off */
        i = 0;
        while(1)
        {
                PORTA = table[i];
                delay(50);
                i = i + 1;
                if (i >= TABLEN) i = 0;       /* Start over when */
                                              /* end is reached */
        }
}
```