

EE 308 – Homework 5

Due Feb. 22, 2010

For the homework problems which follow assume you have included the file `derivative.h` in your C program. Thus, you can refer to `PORTB` when you want to access a byte at address `0x0001`. Where I ask for "some code" just write that part of a C program which will do the task. Where I ask for "a program" write a complete program, include the `#include "derivative.h"` line, the declaration of variables, the `main()` function, etc.

1. Write some C code which will clear bits 4 and 2 of the eight-bit register at address `0x0075` while leaving the other bits unchanged.
2. Write some C code which will set bits 12, 10, 5 and 1 of the sixteen-bit register at address `0x0076` while leaving the other bits unchanged.
3. Write some C code which will set bit 0 of `PORTB` if the 16-bit value at address `0x0099` is less than 72, and will clear bit 0 of `PORTB` if the value is greater than or equal to 72.
4. Write some C code which will wait until Bit 5 of the eight-bit number at address `0x00cc` becomes clear.
5. Write a C program which will find the largest and smallest 16-bit number in memory locations `0x8000` to `0x80ff`. Store the maximum in address `0x1000` and the minimum in address `0x1002`. Treat the numbers as signed.
6. An MC9S12 has the following in some of its registers:

TSCR1	TSCR2	TFLG2
0x80	0x05	0x00

- (a) Is the timer subsystem enabled? How can you tell this?
 - (b) How long will it take (in ms) for the timer to overflow? How can you tell this?
 - (c) Has the timer overflowed since the last time the TOF flip-flop was reset? How can you tell this?
7. Write a C function which does the following: the function will be called with one eight-bit argument, and will return one eight-bit value. Write the argument to `PORTB`, then increment its value and return that new value.
 8. Write a C function which does the following: the function will be called with one argument and will return one eight-bit value. Write the argument to `PORTB`, then generate the next pattern in the sequence for an eight-bit Johnson counter. The procedure to do this is as follows: Shift the present pattern to the right by one bit. The most significant bit of the next pattern is the inverse of the least significant bit of the present pattern. The function return the eight-bit value of the next pattern.

9. Write a C function to take the next entry out of a table, write it to `PORTB`, and update the index into the table. Here is an example of what the table might look like:

```
const char table[] = {0x00, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80};
```

The function is called with one argument, which is the index of the pattern to write to `PORTB`. Your code should write the table entry corresponding to the index to `PORTB`. (For example, if `table_index` were 5, you would write the fifth element of the table, `0x10`, to `PORTB`.) The function should return the index of the next element of the table. Make sure that index stays between 0 and the end of the table.

10. Write a C function to do the following: the function will be called with one eight-bit argument, and will return one eight-bit value. Bits 2 through 0 of the argument tell which bit of `PORTB` to turn on. Bit 3 of the argument represents the direction to shift the bit for the next pattern (1 for left, 0 for right). The upper four bits of the argument are ignored. The bit shifts until it reaches the end, then changes direction. The function returns an eight-bit value, which indicates the bit of `PORTB` to set and the shift direction for next time. For example, if the argument contained `%00001100`, this would indicate that you should turn on bit 4 of `PORTB`. The function should return a `%00001101`, indicating that next time Bit 5 should be lit up, and you should continue shifting left. When you reach the left end (bit 7) change the direction bit from 1 to 0. When you reach the right end (bit 0) change the direction bit from 0 to 1.
11. Write a C program for Part 1 of Lab 3. The program will display four different patterns on the LED display connected to Port B. You will use the state of bits 1 and 0 of the onboard DIP switch to select which of the four patterns to display. Write a program to set up Port B as an eight bit output port (be sure to disable the seven-segment displays, and to enable the individual LEDs), and to implement (i) a binary up counter, (ii) a shifting bit, (iii) a Johnson counter, and (iv) a Ford Thunderbird style turn signal based on the state of the DIP switches. (These are the four subroutines from Problems 6 to 9.) Insert a 100 ms delay between updates of the display. Write the delay as a subroutine. Be sure to initialize the stack pointer in you program.

Use four variables to hold information on the four patterns. Initialize these four variables to the first pattern in the sequence.

You should have a loop which checks the DIP switches connected to Port H. If bit 7 of the DIP switches is high, end the loop and exit back to DBug-12 with a `SWI` instruction. If bit 7 of the DIP switches is low, check bits 0 and 1 to determine what pattern to display:

PH1	PH0	Pattern
0	0	Binary Up Counter
0	1	Shifting Bit
1	0	Johnson Counter
1	1	TBird Turn Signal

For example, if bits 1 and 0 of Port H are 10, call the Johnson Counter function with the Johnson Counter variable as its argument, and save the returned value into the Johnson Counter variable. Call the Delay subroutine, then loop back to check the DIP switches again.