EE 308

Exam 2

March 29, 2010

Name: _____

You may use any of the class handouts and one page of notes. Show all work. Partial credit will be given. No credit will be given if an answer appears with no supporting work.

For all the problems in this exam, assume you are using an MC9S12DG256 with an 8 MHz oscillator clock and a 24 MHz bus clock.

Also, assume that `derivative.h` has been included, so you can refer any register in the MC9S12 by name rather than by its address in any C code you write.

1. The following questions concern writing C code.

   (a) Write some C code which will read the unsigned byte at address 0x1000, and assign it to a variable called `x1`. Be sure to define the variable `x1`.

   ```
   char x1;     // Define variable x1

   x1 = *(char *) 0x1000;
   ```

   (b) Write some C code which will do the following: If bits 3, 2, 1 and 0 of PTH have the value 1001 (binary), write a `0xff` to PORTB. Otherwise, write a `0x00` to PORTB. (Assume that all bits of PTH have been set up for input, and all bits of PORTB have been set up for output.)

   ```
   if ((PTH & 0x0f) == 0x09) PORTB = 0xff;
   else PORTB = 0x00;
   ```

   (c) Write some C code which sets bits 1 and 3, and clears bits 0 and 5 of the TIOS register. It should leave the other bits of TIOS unchanged.

   ```
   TIOS = TIOS | 0x0A & ~0x21;
   ```

2. The following question concerns interrupts and resets. Assume the MC9S12 has the following in its memory:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2000	10	23	3B	7C	10	04	86	80	B7	10	25	3B	FC	10	18	F3
2010	12	50	FD	10	18	86	40	B7	10	23	3B	FC	10	12	DD	02
2020	86	02	B7	10	23	3B	7C	10	03	86	40	B7	10	25	3B	86
FFC0	CC	05	9F	CD	99	03	84	9C	01	9B	CC	90	66	FC	93	30
FFD0	7E	E3	4B	7E	E5	38	21	54	05	83	09	34	2A	38	3C	03
FFE0	41	38	66	F2	7C	13	37	0C	25	F2	0C	38	5F	1B	42	1A
FFF0	7A	26	21	13	6A	AA	20	1F	4B	38	33	38	45	38	20	29

(a) Explain what happens to the Program Counter when the MC9S12 is powered up or reset. What will be the value in the MC9S12's Program Counter immediately after a reset?

The program counter is loaded with the contents of the reset vector, 0xFFFE and 0xFFFF. Immediately after reset, the program counter will have a 0x2029.

(b) What is the address of the instruction the MC9S12 will execute (i.e., the first instruction of the interrupt service routine) when it gets the RTI interrupt?

The RTI interrupt vector is at location 0xFFF0-0xFFF1, so the address of the first instruction of the RTI interrupt service routine is 0x7A26.

(c) Write some C code to set up the MC9S12 to generate an RTI interrupt about once every 4 ms. Be sure to enable the interrupt.

```
RTICTL = 0x60;  // Set rate to 4.096 ms (other values do this too)
CRGINT = 0x80;  // Enable RTI interrupt
CRGFLG = 0x80;  // Clear RTI flag
UserRTI = (unsigned short) &rti_isr;  // Set interrupt vector
__asm(cli);          // Enable interrupts in general
```

(d) Write an RTI interrupt service routine which increments PORTA every time the RTI interrupt occurs.

```
interrupt void rti_isr(void)
{
    PORTA = PORTA + 1;    // Increment PORTA
    CRGFLG = 0x80;        // Clear RTI flag
}
```

3. The MC9S12 registers have the following values when an enabled RTI interrupt occurs:

Reg	–	–
	S X H I N Z V C	
CCR	1 1 0 0 1 0 0 1	
A:B	A3	92
X	AABB	
Y	1234	
SP	18A3	
PC	2956	

(a) What will be the value of the MC9S12 stack pointer when the MC9S12 begins executing the first instruction of the RTI interrupt service routine?

The stack pointer is decremented by 9 because the MC9S12 pushes 9 bytes onto the stack. The new value will be 0x18A3 - 9 = 0x189A.

(b) Explain what happens to the MC9S12 stack when the MC9S12 gets the RTI interrupt. Show how the stack will be changed when the interrupt occurs – that is, show what bytes will be put into the stack area of memory, and what locations are.

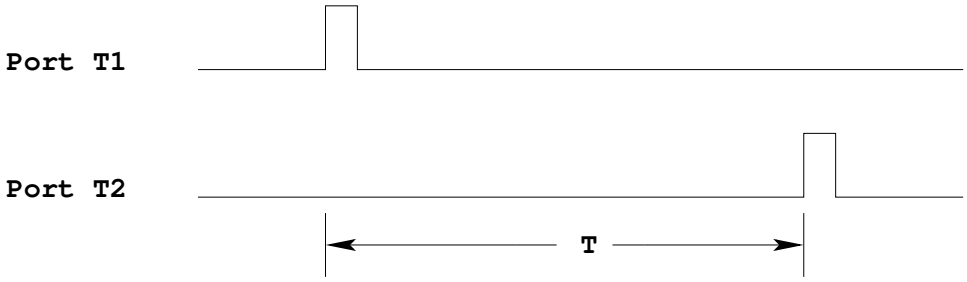
The MC9S12 completes the current instruction, pushes CCR, B, A, X, Y, and PC onto the stack, and loads the PC with the RTI interrupt vector.

Addr	Value	What
0x189A	C9	CCR
0x189B	92	B
0x189C	A3	A
0x189D	AA	X High
0x189E	BB	X Low
0x189F	12	Y High
0x18A0	34	Y Low
0x18A1	29	PC High
0x18A2	56	PC Low

(c) What happens to the condition code register when the MC9S12 gets an RTI interrupt? Why did the Motorola engineers have the MC9S12 do this?

The I bit of the condition code register is set to 1. This disables interrupts so the MC9S12 does not respond to another interrupt while it is in an interrupt service routine. The interrupts are re-enabled when the MC9S12 leaves the interrupt with the `rti` (Return from Interrupt) instruction, and the CCR (with a 0 in the I bit) is reloaded with its original value from the stack.

4. You are doing an experiment where you need to measure the the speed of an object. You do this by measuring the time it takes an object to travel between two points. When the object passes the first point it breaks a light beam which creates a pulse on a signal connected to Port T1. When it passes the second point it creates a pulse on a signal connected to Port T2. You know before the experiment that the time difference will be between 100 ms and 200 ms.



(a) What value should you write to the timer prescaler? Why? Write some C code to do this.
With a prescaler of 0, the overflow rate is 65,536 cycles/24,000,000 cycles/second = 2.73 ms. Need to increase this by a factor of 128 to get an overflow rate of 350 ms. To do this, write a 7 to the prescaler.

```
TSCR2 = 0x07;
```

(b) How do you set up the MC9S12 to capture the times of the rising edges of the two signals? Write some C code to do this.

```
TSCR1 = 0x80;    // Enable timer subsystem
TSCR2 = 0x07;    // Set prescaler to 350 ms overflow rate
TIOS = TIOS & ~0x06;  // Set up channels 1 and 2 for input capture
TCTL4 = TCTL4 | 0x14 & ~0x28;  // Capture rising edges
TFLG1 = 0x06;  // Clear flags for channels 1 and 2
```

(c) Write some C code which will wait until the object passes the second point.
Wait until timer channel 2 flag is set.

```
while ((TFLG1 & 0x04) == 0) ;
```

(d) Write some C code which will clear the flag for Timer Channel 2.
Write a 1 to flag for channel 2, and a 0 to all other bits of TFGL1.

```
TFLG1 = 0x04;
```

(e) After both edges have been captured, the following is in the MC9S12 timer registers:

TC0	TC1	TC2	TC3	TC4	TC5	TC6	TC7
681C	C25F	1B25	A29C	F49A	F902	18AC	0059

How long (in seconds) did it take for the object to traverse the distance?
TC2 - TC1 = 0x1B25 - 0xC25F = 0x58C6 = $22{,}726_{10}$ cycles.
(Note that there was a borrow on the subtraction, which is ignored.)
Normally, one clock cycle is 1/24,000,000 seconds. The prescaler is 7, which increases this time by $2^7 = 128$, so one cycle of the timer subsystem is 128/24,000,000 seconds.
22,726 cycles $\times 128/24,000,000$ cycles/sec = 221 ms.

5. The phaser control system on the Enterprise has burned out. Mr. Scott asks you to design a new control system using the MC9S12. The phaser needs a PWM signal with a 10 kHz frequency. The stun setting requires a 10% duty cycle. The vaporize setting requires an 80% duty cycle.

   (a) Set up the MC9S12 to produce a 10 kHz PWM signal with a 10% duty cycle on Bit 0 of Port P.

   24,000,000/10,000 = 2,400 cycles.
   Can use either clock mode 0 or clock mode 1.
   For clock mode 1:
   2,400 = PWMPER0 $\times$ $2^{PCKA+1}\times$ PWMSCLA
   Can do this with PWMPER0 = 200, PCKA = 1, PWMSCLA = 3 (and many other ways) This is what I do below.
   For clock mode 0,
   2,400 = PWMPER0 $\times 2^{PCKA}$
   Can do this with PWMPER0 = 150 and PCKA = 4 (and other ways)

```
PWMCTL = 0x00;                          // 8 bit mode
PWMPOL = 0xFF;                          // high polarity
PWMCAE = 0x00;                          // left aligned
PWMCLK = PWMCLK | 0x01;                 // clock mode 1
PWMPRCLK = PWMPRCLK | 0x01 & ~0x06;     // PCKA = 1 (for channels 0, 1, 4, 5)
PWMSCLA = 0x03;                         // scale for channels 0, 1, 4, 5
PWMPER0 = 200;                          // period for channel 0
PWME = PWME | 0x01;                     // enable channel 0
PWMDTY0 = 20;                           // 10% duty cycle for channel 0
```

   (b) Set up the MC9S12 to produce a 10 kHz PWM signal with an 80% duty cycle on Bit 1 of Port P.

   Same frequency as for Channel 0. Channel 0 and 1 share PCKA and PWMSCLA, so those do not need to be reset.
   Need to select clock mode 1 for channel 1, set the period and duty cycle, and enable channel 1.

```
PWMCLK = PWMCLK | 0x02;   // clock mode 1 for channel 1
PWMPER1 = 200;            // period for channel 1
PWME = PWME | 0x02;       // enable channel 1
PWMDTY1 = 160;            // 80% duty cycle for channel 1
```