EE 308

Exam 3

May 4, 2009

Name: _____

You may use any of the Freescale data books, the class lecture notes, and a calculator. 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 MS9S12 with an 8 MHz crystal, resulting in a 24 MHz bus clock.

Also, assume that `hcs12.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. I^2C Bus

   (a) The schematic for the Dragon12-Plus board shows that the SDA and SCL lines of the I2C bus are connected to VCC through pull-up resistors. What are the purpose of these pull-up resistors?

   The SDA and SCL lines are normally held high via the pull-up resistors. Any device on the I2C bus can pull either of the lines low. When the device releases the line, it goes back to a high because of the pull-up resistors.

   (b) Suppose you accidentally connected the pull-up resistors on SDA and SDL to ground rather than VCC. What would the signals on the SDA and SCL lines look like?

   With the pull-up resistors connected to ground, the SDA and SCL lines will always be low. Since the devices connected to the I2C lines can only pull them low, there is no way these lines can go high. The I2C bus will not work.

   (c) Describe a feature of the START condition that can be used to distinguish it from any other type of activity on the I2C bus.

   The START condition occurs when SDA goes from high to low while SCL is high. This can never occur any other time. (The STOP condition occurs when SDA goes from low to high which SCL is high.)

   (d) The ADS7830 for Texas Instruments is an 8-bit, 8-channel A/D converter with an I2C bus. The data sheet for the ADS7830 says that, in standard mode, the SCL clock frequency should be less than 100 kHz. What value should you write to what register in order the set up the MC9S12 to run the I2C bus as fast as possible such that it is still compatible with the ADS7830? Explain.

   24 MHz/100 kHz = 240. Need the SCL Divider to be 240 or bigger. The best value to write to IBFD (IIC Bus Frequency Divider Register) is 0x1F, which divides the bus clock by 240 exactly. The values 0x23, 0x5B or 0x92, all of which divide the bus clock by 256, are also pretty good.

2. A/D Converter

(a) Suppose the A/D converter on the MC9S12 is set up in 10-bit mode, with V_{RL} = 1.0 V and V_{RH} = 3.0 V. ATD0 of the MC9S12 is set up to convert all eight inputs, and store the results in right-justified mode. After a series of conversions, the following results are obtained:

ATD0DR0	ATD0DR1	ATD0DR2	ATD0DR3	ATD0DR4	ATD0DR5	ATD0DR6	ATD0DR7
0x0049	0x008D	0x036B	0x0116	0x0158	0x019E	0x01E2	0x026B

i. What was the input voltage on Channel 0?
0x0049 = 73_{10}, so

$$V_0 = V_{RL} + \frac{V_{RH} - V_{RL}}{2^{10} - 1} \times \text{ADvalue} = 1.0 + \frac{3.0 - 1.0}{1023} \times 73 = 1.143 \text{ V}$$

ii. What was the input voltage on Channel 4?
0x0158 = 344_{10}, so

$$V_0 = V_{RL} + \frac{V_{RH} - V_{RL}}{2^{10} - 1} \times \text{ADvalue} = 1.0 + \frac{3.0 - 1.0}{1023} \times 344 = 1.673\text{V}$$

(b) Write some C code to set up ATD0 to operate in 10-bit mode, and to convert Channels 0 through 3 one time, then stop.

```
ATD0CTL2 = 0x80;     // Power up A/D 0
ATD0CTL3 = 0x20;     // 0x20 = 0b0 0 1 0 0 0 0 0
                     //            | | | |
                     //            \ \ \ \_________ 4 conversions per sequen
ATD0CTL4 = 0x05;     // 10 bit conversions, fastest rate possible
ATD0CTL5 = 0x90;     // 0x90 = 0b 1 0 0 1 0 0 0 0
                     //           | | | |   | | |
                     //           | | | |   \_\_\____ Start Channel 0
                     //           | | | \_____________ Multiple channels
                     //           | | \_______________ One set then top
                     //           \_\_________________ Right justified, unsig
```

(c) Write some C code to wait until the above set of 4 conversion has finished.

```
while ((ATD0STAT0 & 0x80) == 0) ; // Wait for SCF flag to become set
```

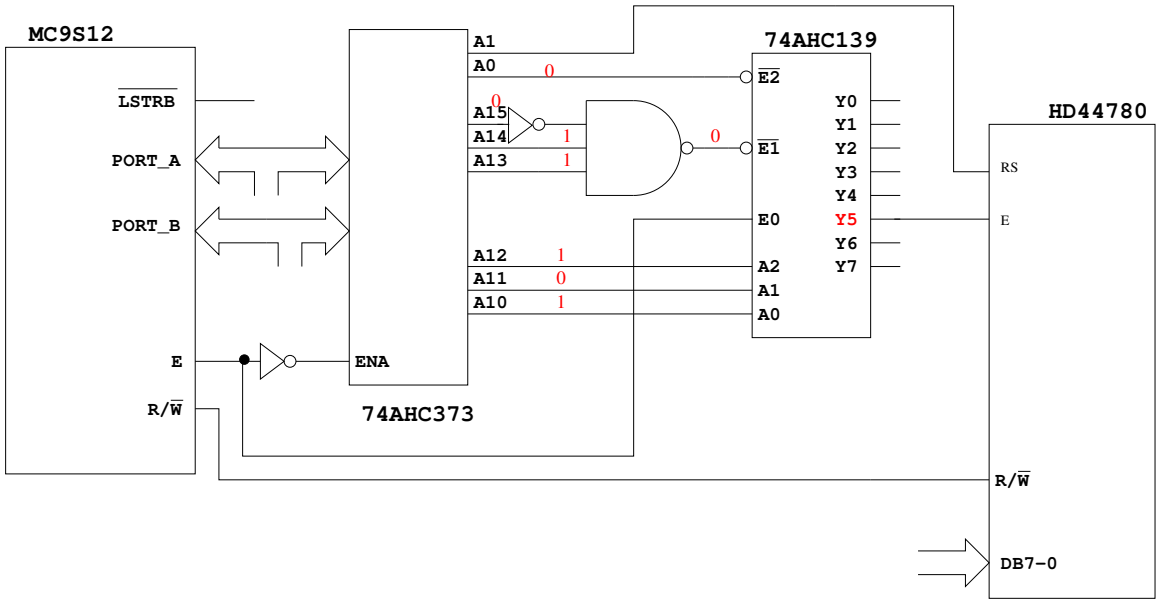
(d) Given your setup above, how long did it take the MC9S12 to convert the four channels?
With ATD0CTR4 = 0x05, it takes 14 cycles/conversion with a 2 MHz A/D clock, so it takes 14 $\times$ 0.5 μs = 7 μs/conversion, and it takes 28 μs for the set of four conversions.

3. Interfacing

The Hitachi HD44780 is a chip to control LCD displays. The figure below shows a possible way to connect the HD44780 to an MC9S12.

The HD44780 is enabled when its E input is high. (This is not the same as the E clock for the MC9S12.) The HD44780 has two internal register, Register 0 and Register 1. The RS input to the HD44780 is a register select. If RS is low, Register 0 is selected. If RS is high, Register 1 is selected. R/$\overline{W}$ is the Read/Write line. If R/$\overline{W}$ is high, the chip is in read mode. R/$\overline{W}$ is low, the chip is in write mode.

The 74AHC139 is the same as the 74AHC138, except the selected output is active high rather than low. That is, when the chip is not enabled, all eight outputs are low. when the chip is enabled, The output corresponding to A_2 A_1 A_0 is high, and the other seven outputs are low. The 74AHC373 is an eight-bit transparent latch. When ENA is high, the data inputs are transfered into the data outputs. When ENA goes low, the data doesn't change even if the data inputs change.



(a) For what range of addresses will the HD44780 be selected? Explain.

To select the 74AHC373 chip, need A15 low, A14 and A13 high. To select output Y5, need A12 A11 A19 = 1 0 1, so need address lines to be `011101XXXXXXXXXX`. With all the X's as 0, this gives 0x7400. With all the X's as 1, this give 0x77FF. To select the 74AHC139, also need A0 to be low, so need even addresses withing the range 0x7400 through 0x77FF.

(b) Should the data lines of the HC44780 be connected to Port A or Port B? Explain.

New A0 to be low, so use even addresses, high byte, Port A.

(c) Explain how you can access Register 0, and how you can access Register 1.

To access Register 0, need A1 to be low. Thus, even addresses in the range of 0x7400 through 0x77FF for which A1 is low (such as 0x7400) will access Register 0. To access Register 0, use an address of the form: `0111 01XX XXXX XX00`.

To access Register 1, need A1 to be high. Thus, even addresses in the range of 0x7400 through 0x77FF for which A1 is high (such as 0x7402) will access Register 1. To access Register 0, use an address of the form: `0111 01XX XXXX XX10`.

(d) Write some C code which will write the 8-bit value 0xAA to Register 0.

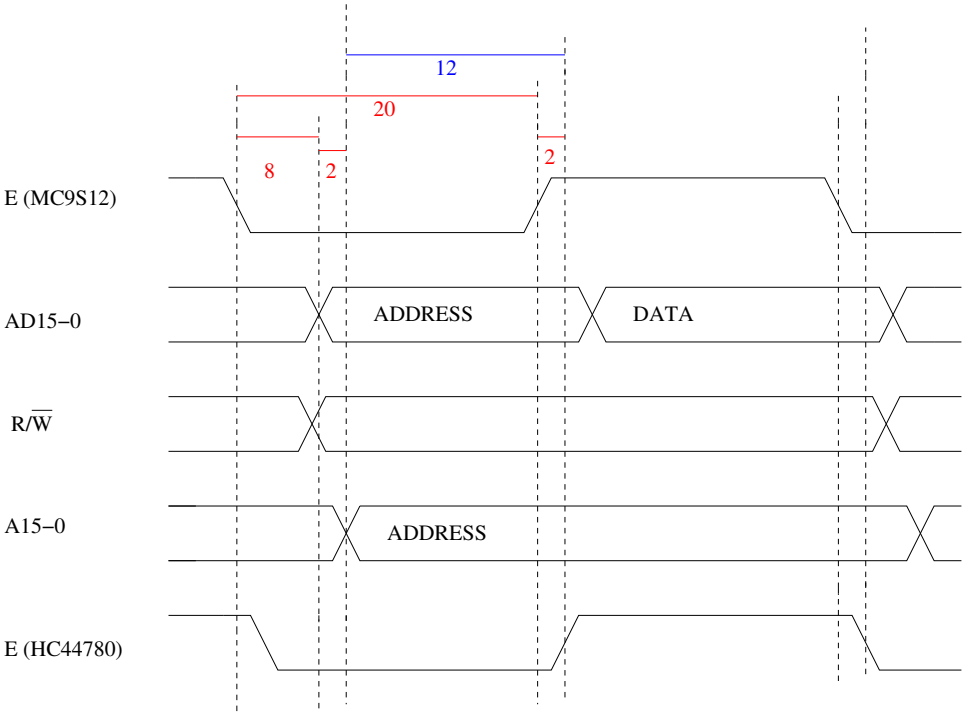
```
*(char *) 0x7400 = 0xaa;
```

(e) Write some C code which will read the 8-bit value from Register 1, and save it in a variable called `Reg_1`.

```
char Reg_1;

Reg_1 = *(char *) 0x7402;
```

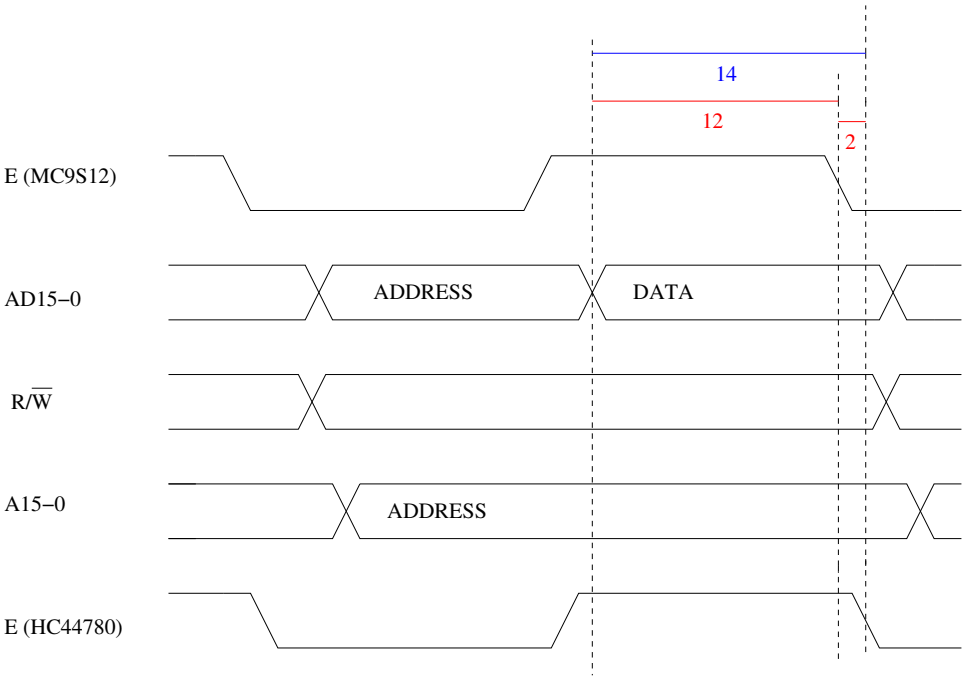
(f) The timing diagrams for the HC44780 chip are attached. Assume that the propagation delays through each glue logic chip is 2 ns. Is the time t_{AS} on the HC44780 data sheet compatible with an MC9S12 with a 24 MHz bus clock, and no E-clock stretches? If the time is not compatible, can you make it compatible by changing the bus clock frequency or adding E-clock stretches? Explain.



t_{AS} is the time from RS (A1) and R/$\overline{W}$ being stable until E (on the HC44780) goes high. E (on the HC44780) goes high 2 ns after E (on the MC9S12) goes high. From the MC9S12 data sheet, it takes time 5 (t_{AD} on the MC9S12, 8 ns) from E (MC9S12) low to new address. It takes 2 ns longer for A1 to get through the 74HC373, so it takes 10 ns from E (MC9S12) low to A1 valid. Since E (MC9S12) is low for about 20 ns, it is 10 ns from A1 valid to E (MC9S12) high, then another 2 ns for E (HC44780) to go high. Thus, it is 12 ns from A1 valid to E (HC44780) high. The HC44780 needs 40 ns, so this will not work.

Can make it work by slowing down the bus clock so E (MC9S12) is low for 50 ns and high for 50 ns. This adds an extra 30 ns to t_{AD} (MC9S12), so now from A1 established to E (HC44780 high) is 42 ns. This means E is low for 50 ns, and high for 50 ns, which is a 10 MHz bus clock.

(g) Is the time t_{DSW} on the HC44780 data sheet compatible with an MC9S12 with a 24 MHz bus clock, and no E-clock stretches? If the time is not compatible, can you make it compatible by changing the bus clock frequency or adding E-clock stretches? Explain.



t_{DSW} is the time from new data on the bus to E (HC44780) goes low. The MC9S12 puts new data on the bus 12 ns before E (MC9S12) goes high (time 14 on the MC9S12 data sheet). E (HC44780) goes high 2 ns later, so the data is on the bus 14 ns before E (HC44790) goes high. The HC44780 needs data on the bus 80 ns before its E goes low, so this is 66 ns too fast. If you use a 10 MHz bus clock, this gives another 30 ns, so the data will be on the bus 44 ns before E (HC44780) goes high. This is still too short. Can add an E-clock strech, which adds 100 ns to the time E (MC9S12) is high. It will work then.

The HC44780 will work with the MC9S12 if the MC9S12 bus clock is 10 MHz or slower. With a 10 MHz bus clock, still need an E-clock strech to meet the t_{DSW} time of the HC44780.