EE 308 Exam 3
April 24, 2006

Name: _____

You may use any of the Motorola data books, EE 308 notes from the web, 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 HCS12DP256 with a 8 MHz crystal and a 24 MHz bus clock.

Also, assume that `hc12.h` has been included, so you can refer any register in the 9S12 by name rather than by its address in any C code you write.

1. The following problems have to do with the A/D converter on the 9S12.

   (a) On a 9S12, V_{RL} is connected to +2 V, and V_{RH} is connected to +4 V. The A/D converter is set up for 10-bit, right-justified, unsigned mode. The output from the A/D converter is 0x27C. What is input voltage?
   0x27c = 636_{10}

   $$V_{in} = V_{RL} + \frac{V_{RH} - V_{RL}}{2^b} \text{ADvalue} = 2 + \frac{4-2}{1024} \times 636 = 3.24 \text{ V}$$

   (b) The A/D registers of the 9S12 are set up as follows:

ATD1CTL0	ATD1CTL1	ATD1CTL2	ATD1CTL3	ATD1CTL4	ATD1CTL5
0x00	0x00	0x80	0x20	0x05	0x55

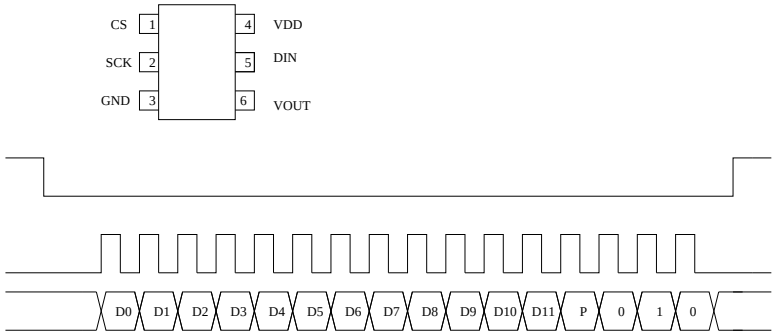
   An 0x55 is written to ATD1CTL5. Explain what the A/D converter will do. Be sure to explain resolution (8 or 10 bits), data justification, number of channels converted, which channels are converted, one scan and stop or continuous scan, and how long it takes to complete one scan.

   - ATD1CTL0 and ATD1CTL1 are unused
   - ATD1CTL2 = 0x80 => AD power on
   - ATD1CTL3 = 0x20 => [S8C S4C S2C S1] = 0100 => 4 conversions/scan
   - ATD1CTL4 = 0x05 => 10-bit mode, 2 MHz A/D clock, 14 cycles/converison
   - ATD1CTL5 = 0x55 => DJM = 0 (left justified), DSGN = 1 (signed), SCAN = 0 (one scan then stop), MULT = 1 (multiple channels), [CC CB CA] = 101, so first channel is 5.
     10 bit mode, left justified, signed, 4 channels, channels are 5, 6, 7, 0 of ATD1 (or inputs PAD13, PAD14, PAD15, PAD8). The result from Channel 5 will go into ATD1DR0, Channel 6 to ATD1DR1, etc.

(c) Write some C code to write 0x55 to ATD1CTL5, wait until the conversion is complete, and read the value of the first channel converted into a 16-bit variable called `ad_value`.

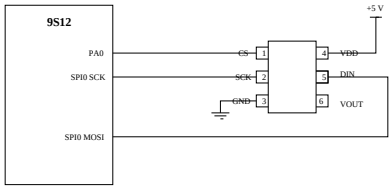
```
unsigned short ad_value;

ATD1CTL5 = 0x55;
while ((ATD1STAT0 & 0x80) == 0) ;  /* Wait for scan to finish */
ad_value = ATD1DR0;
```

2. The figure below shows a 12-bit D/A converter which uses the SPI protocol. It requires a 16-bit input word, sent least significant bit first. Bits 15, 14, and 13 must be 010, or the word will be ignored. Bit 12 is used to power up the D/A converter – a 1 to bit 12 turns on the A/D output, and a 0 to bit 12 turns off the D/A output. The twelve data bits specify the desired output voltage, with 0x000 for 0 V out and 0xFFF for 12 V out. The data sheet for the D/A converter says that the maximum frequency of the SPI clock is 1 MHz.



(a) Sketch how you would connect the D/A to a 9S12. That is, show where the SCK, DIN and $\overline{\text{CS}}$ will connect to the HCS12.

Need to connect the SCK line to the SPI0 SCK, connect the DIN to the SPI0 MOSI, and connect the CS line to a general purpose IO line. I chose to use Bit 0 of Port A. Any reasonable choice was acceptable.



(b) Write some C code to set up the 9S12 SPI for use with the D/A converter. To insure as much credit as possible, explain how you set up each register.

```
DDRA = DDRA | 0x01;    /* Make Bit 1 Port A an output */
PORTA = PORTA | 0x01;  /* De-select D/A */
SPI0CR1 = 0x55;  /* 0 1 0 1 0 1 0 1
                    | | | | | | | |
                    | | | | | | | \_ LSB first
                    | | | | | | |___ SSOE = 0 -- multiple byte transfer
                    | | | | | \_____ CPHA = 1 => data valid on 2nd edge
                    | | | | \_______ CPOL = 0 => clock idle low
                    | | | \_________ MSTR = 0 => master mode
                    | | \___________ No INT on transmit
                    | \_____________ Power on SPI0
                    \_______________ No INT on recieve
                 */
SPI0CR2 = 0x00; /* Normal (not bidirectinal) mode */
SPI0BR = 0x22;  /* 1 MHz clock (several other values work too) */
```

(c) Write some C code which will turn on the D/A converter and set the output voltage to 1.25 V.

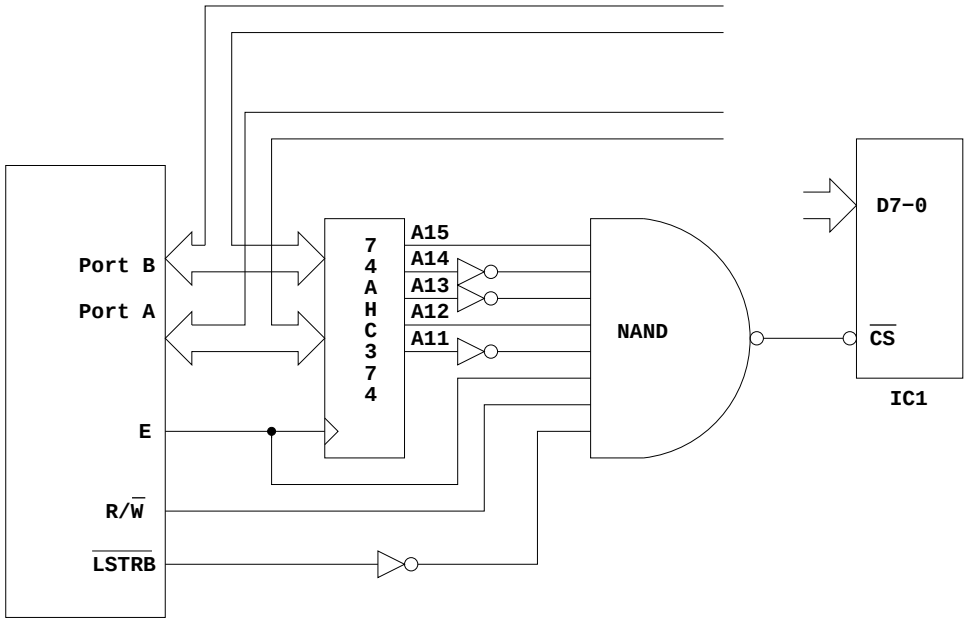
To turn on the D/A, need to make bit P a 1. To set output to 1.25 V, need to send

$$\frac{1.25\,\text{V}}{12\,\text{V}} = \frac{\text{DA value}}{1023} \text{ so DA value} = \frac{1.25}{12} \times 1023 = 427_{10} = 01\text{AB}_{16}$$

This needs to be predeeded by 0101_2, or 5_{16}, so need to send 0x51AB. The SPI transfers are done one byte at a time, so need to break this up into two one-byte transfers. Need to send LSB first, so need to send 0xAB followed by 0x51:

```
unsigned char tmp;

PORTA = PORTA & ~0x01;          /* Select D/A converter */
while ((SPI0SR & 0x20) == 0) ;  /* Wait until okay to transmit */
SPI0DR = 0xAB;                  /* Send low byte */
while ((SPI0SR & 0x80) == 0) ;  /* Wait until transfer finished */
tmp = SPI0DR;                   /* Clear SPIF */
while ((SPI0SR & 0x20) == 0) ;  /* Wait until okay to transmit */
SPI0DR = 0x51;                  /* Send high byte */
while ((SPI0SR & 0x80) == 0) ;  /* Wait until transfer finished */
tmp = SPI0DR;                   /* Clear SPIF */
PORTA = PORTA | 0x01;           /* De-select D/A converter */
```

3. The following shows a 9S12 interfaced to a peripheral chip:



(a) Is the peripheral in input or output chip?Why?

Chip is selected when output of NAND is low. For this to happen, all inputs to NAND must be high. Since R/$\overline{W}$ is an input, R/$\overline{W}$ must be high, so the 9S12 is reading from the peripheral, so the peripheral is an input device.

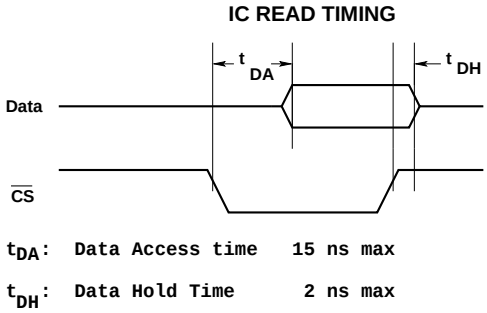
(b) What range of addresses will select the peripheral?

For all inputs to NAND must be high, need A15 = 1, A14 = 0, A13 = 0, A12 = 1, A11 = 0. All the other address lines are not used. Address lines must be $10010XXXXXXXXXXX_2$ to select the chip, so the range is 0x9000 (all X's 0) to 0x97FF (all X's 1). Also, $\overline{\text{LSTRB}}$ must be low, so the chip responds to reads from odd addresses in the range 0x9000 through 0x97FF.

(c) Should you connect the data lines for the peripheral to Port A or Port B? Why?

$\overline{\text{LSTRB}}$ low => low byte (odd address) => data on Port B (AD 7-0).

(d) Assume that the peripheral is an input, with the timing diagram shown below. Answer the following timing questions, assuming that the delay through each of the glue chips is 2 ns.



i. How long is it from the time E goes high until $\overline{\text{CS}}$ goes low? (Assume there are no E-clock stretches.)
E goes low, 2 ns later, A15 through A11 come out of 74AHC374, 2 ns later, inverted A14, A13, A11 come out of NOT gates, 2 ns later, CS goes low. It takes 6 ns from E high to $\overline{\text{CS}}$ low.

ii. How long is it from the time E goes high until the input chip puts its data on the bus? (Assume there are no E-clock stretches.)
E goes low, 6 ns later $\overline{\text{CS}}$ goes low, 15 ns (t_{DA}) later the chip puts its data on the bus, for a total of 21 ns.

iii. From the time the chip puts its data on the bus, how long is it before E goes low? Is this compatible with the 9S12 data setup time? Explain. (Assume there are no E-clock stretches.)
From 9S12 data sheet, E high to E low is 19 ns. E high to data on bus is 21 ns, so data does not get on bus until 2 ns after E goes low. The 9S12 data sheet says data must be on bus 13 ns before E goes low, so this will not work.

iv. From the time the chip puts its data on the bus, how long is it before E goes low? Is this compatible with the 9S12 data setup time? Explain. (Assume there is one E-clock stretch.)
One E-clock stretch increases the time E is high by 40 ns (one clock period). Now, E high to E low is 19 ns + 40 ns = 59 ns. E high to data on bus is 21 ns, so data on bus to E low is 59 ns - 21 ns = 38 ns. The 9S12 data sheet says data must be on bus 13 ns before E goes low, so this will work.

v. From the time E goes low, how long is it before the chip takes its data off the bus? Is this compatible with 9S12 data hold time? Explain. From E low, it takes 2 ns before $\overline{CS}$ goes high, then another 2 ns (t_{DH}) before data is off the bus, so it takes 4 ns to get data off the bus.
From the 9S12 data sheet, the Read Data Hold time (time 11 on diagram) is 0 ns minimum. The chip leaves data on bus for longer than this, so this will work.
From the 9S12 data sheet, the Address Delay Time (time 5 on diagram) is 8 ns maxmium. No more than 8 ns after E goes low, the 9S12 will put the new address on the bus. The chip has its data off the bus in no more than 4 ns, so this should work.
From the 9S12 data sheet, the Data Hold to Address Time (time 9 on diagram) is 2 ns minimum. The 9S12 switches from data to address sometime between 2 ns and 8 ns after E goes low. The chip removes its data in no more than 4 ns, so between 2 ns and 4 ns the 9S12 and the chip may both be trying to drive the bus. This will probably not cause a problem.