EE 308 Exam 3
Solutions April 25, 2005

Name: _____

You may use any of the Motorola data books and the notes from the web. 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 4 MHz crystal and a 24 MHz bus clock.

Also, assume that `iodp256.h` has been included, so you can refer any register in the HCS12 by name rather than by its address in any C code you write.

1. A sensor is connected to Channel 4 of the AD0 A/D converter of an HCS12. The sensor measures the relative humidity of an experimental chamber. 0% relative humidity gives a voltage of 0 V. 100% relative humidity gives a voltage of 5 V. The sensor is linear from 0 to 100%. The experimental process requires that the relative humidity remain between 10% and 20%.

   (a) Write some C code to set up the A/D converter to operate in 10 bit mode, and to do a sequence of eight conversions on A/D Channel 4, then stop.

   Need MULT = 0 (one channel), CC CB CA = 100 (Channel 4), S8C S4C S2C S1C = 0000 or 1XXX for eight conversion, SRES8 = 0 for 10-bit mode. Want to do one sequence of 8 conversions, then stop, so need SCAN = 0. I will use right-justified, unsigned data mode.

```
ATD0CTL2 = 0x80;   /* Enable A/D 0 */
ATD0CTL3 = 0x00;   /* 8 conversion */
ATD0CTL4 = 0x05;   /* 10-bit mode, fastest conversion time */
ATD0CTL5 = 0x84;   /* Right justified, unsigned, SCAN=0, MULT=0, Channel 4 */
```

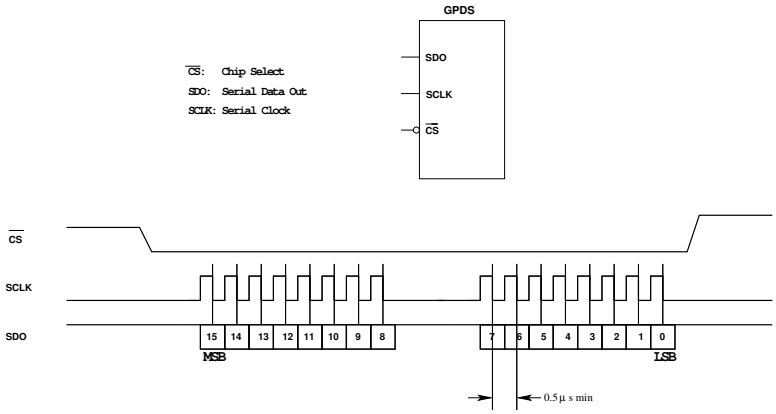
   (b) Write some C code to wait until the above set of 8 conversion has finished.

```
while ((ATD0STAT0 & 0x80) == 0) ;  /* Wait until SCF is set */
```

   (c) The experimental process requires that the relative humidity remain between 10% and 20%. What is the output of the A/D converter when the realtive humidity is 10%? What is the output of the A/D converter when the relative humidity is 20%C?

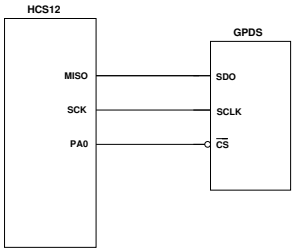
   The ten-bit output goes from 0 (0 V) to 1023 (5 V). 10% RH is 10% of the way to 1024, or 102_{10} = 0x66. 20% RH is 20% of the way to 1024, or 204_{10} = 0xCC.

2. The figure below shows a GPDS (General Purpose Distance Sensor) and its timing diagram. The GPDS works in the folowing manner: When the chip select ($\overline{\text{CS}}$) line of the GPDS goes low, the GPDS sends out a sonic pulse. It measures the time it takes for the echo to return, and converts the distance that represents to millimeters. That unsigned number repesents the distance in millimeters to an object in front of the sensor. (If no echo is detected, it returns an 0xFFFF, indicating no object within its range.) It sends the 16-bit result over the SDO (Serial Data Out) line.



(a) Sketch how you would connect the GPDS to an HCS12. That is, show where the SCLK, SD0 and $\overline{\text{CS}}$ will connect to the HCS12.

Connect SDO (Serial Data Out) to MISO (Master In Slave Out). Connect SCLK to SCK. Connect $\overline{\text{CS}}$ to any general purpose IO. Here I connected it to Bit 0 of Port A.



(b) Write some C code to set up the HCS12 SPI for use with the GPDS. To insure as much credit as possible, explain how you set up each register.

Nee to enable SPI0 in master mode with clock idle low and data valid on 2nd clock edge. Data is sent MSB first. Need to transfer two bytes with CS low, so need to manually control chip select line. The GPDS works with a 2 MHz clock maximum (0.5 μs period) so set baud rate to 2 MHz.

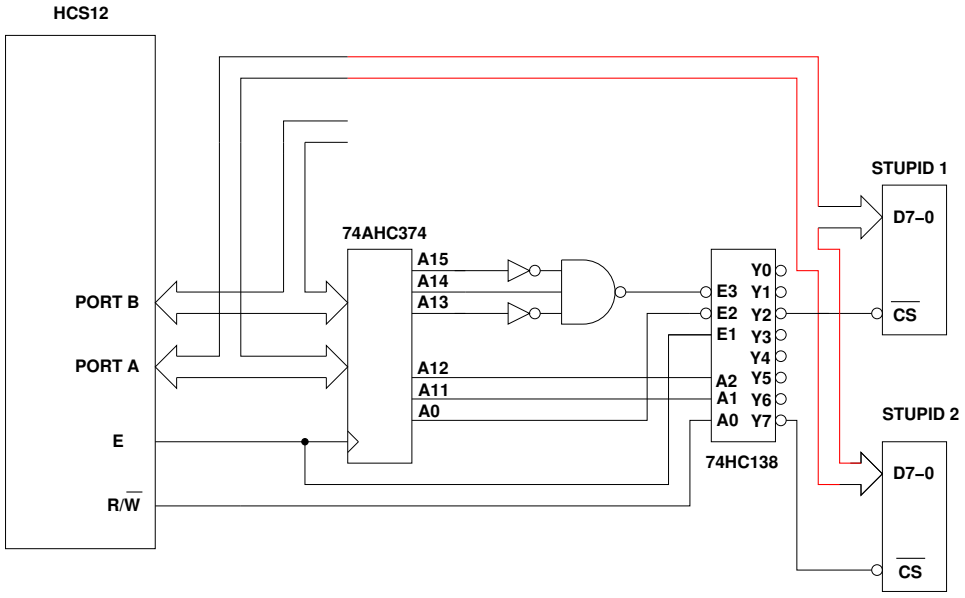
```
SPI0CR1 = 0x54;    /* No interrupts, enable SPI, Master Mode, Clock idle low,
                    * Data on 2nd clock edge, MSB first, control SS manually */
SPI0CR2 = 0x00;    /* Normal (not bidirectional) mode */
SPI0BR = 0x50;     /* 2 MHz clock */
DDRA = 0xFF;       /* PORT A output */
PORTA = 0xFF;      /* Deselect slave */
```

(c) Write some C code to read the distance, and assign it to a variable called `distance`.
The only tricky thing here is that the distance is a 16-bit number which is read as two 8-bit numbers. The data needs to be converted from two 8-bit numbers to the 16-bit distance.

```
unsigned int distance;

PORTA = PORTA & ~0x01;           /* Select GPDS */
while ((SPI0SR & 0x20) == 0) ;   /* Wait until SPI0TEF flag set */
SPI0DR0 = 0;                     /* Write junk byte to start transmission */
while ((SPI0SR & 0x80) == 0) ;   /* Wait for transmission of finish */
distance = SPI0DR << 8;          /* Read 8 MSB of distance */
while ((SPI0SR & 0x20) == 0) ;   /* Wait until SPI0TEF flag set */
SPI0DR0 = 0;                     /* Write junk byte to start transmission */
while ((SPI0SR & 0x80) == 0) ;   /* Wait for transmission of finish */
distance = distance | SPI0DR;    /*
PORTA = PORTA | 0x01;            /* Deselect GPDS */
```

3. A new startup company designs a series of peripheral chips they call the Super Terrific Universal Peripheral Interface Device (STUPID) chips. The figure below shows two STUPID chips connected to an HC12. One is an input chip, the other is an output chip.



(a) What range of addresses will select STUPID 1 chip? Is it an input chip or an output chip.

To enable the 74AHC138, need A15=0, A14=1 and A13=0, E=1 and A0=0. To select STUPID1 (Y2), need address inputs to 74AHC138 to be 010, so need A12=0, A11=1 and R/$\overline{\text{W}}$=0. This means that means that address lines must be `01001XXXXXXXXXX0`, or any even address between 0x4800 to 00x4FFE. Because R/$\overline{\text{W}}$=0, this is an output chip — the HCS12 uses it for output.

(b) Should `D7-0` of STUPID 1 be connected to Port A or Port B? Why?

From (a), STUPID1 is selected for even address, so it is the high byte, and should be connected to Port A (AD15-8).

(c) What range of addresses will select STUPID 2 chip? Is it an input chip or an output chip.

To enable the 74AHC138, need A15=0, A14=1 and A13=0, E=1 and A0=0. To select STUPID1 (Y2), need address inputs to 74AHC138 to be 111, so need A12=1, A11=1 and R/$\overline{\text{W}}$=1. This means that means that address lines must be `01011XXXXXXXXXX0`, or any even address between 0x5800 to 00x5FFE. Because R/$\overline{\text{W}}$=1, this is an input chip — the HCS12 uses it for input.

(d) Should `D7-0` of STUPID 2 be connected to Port A or Port B? Why?

From (c), STUPID2 is selected for even address, so it is the high byte, and should be connected to Port A (AD15-8).

(e) Write some C code to write the value 0x55 to the output chip.

```
*(unsigned char *) 0x4800 = 0x55;
```

(f) Show what will be on the bus when you write an 0x55 to the output port.

Note: When writing to an even byte, $\overline{\text{LSTRB}}$ is a don't care. The instruction in (b) writes a single byte to an even address, so $\overline{\text{LSTRB}}$ will be high.