EE 308

Exam 3

May 6, 1999

Name: _____

You may use one page of notes and any of the Motorola data books. 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 HC12 with a 16 MHz crystal, resulting in a 8 MHz processor clock.

Also assume that `hc12.h` has been included, so you can refer any register in the HC12 by name rather than by address.

1. Signals from six sensors are connected to bits 0 through 7 of Port AD of a 68HC12. The A/D reference voltages are $V_{RH} = 5$ V and $V_{RL} = 0$ V. You want to set up the A/D converter to convert all of these channels, then stop.

   (a) How do you set up the A/D converter to do this? I.e., what values do you write to which registers?

   i. Enable A/D (`1 -> ADPU in ATDCTL2`)
   ii. Select 8-bit mode (`1 -> S8CM in ATDCTL5`)
   iii. Select multiple channels (`1 -> MULT in ATDCTL5`)
   iv. Select single conversion sequence (`0 -> SCAN in ATDCTL5`)
   v. Always need CD = 0 (`0 -> CD in ATDCTL5`)
   vi. Select clock rate (`0x01 -> ATDCTL4` for fastest convert rate)

   (b) Write some C code which will do the conversions as described above.

```
ATDCTL2 = 0x80;
ATDCTL4 = 0x01;
ATDCTL5 = 0x50;
```

   (c) How can you tell when the sequence of conversions is done?

   - SCF bit of ATDSTAT register is set to 1

   (d) After the conversion is done, the A/D result registers have the following values:

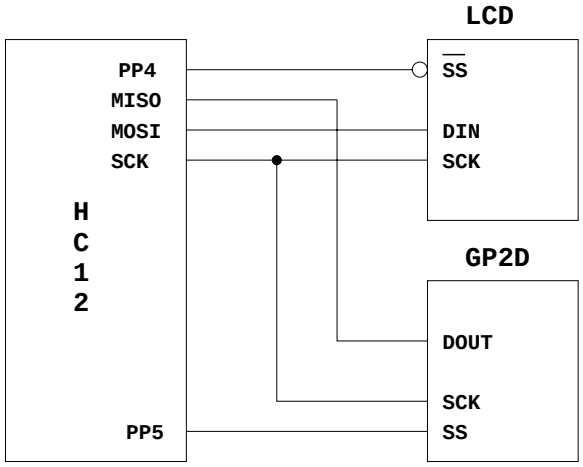
ADR0H	ADR1H	ADR2H	ADR3H	ADR4H	ADR5H	ADR6H	ADR7H
05	F7	6A	42	C2	A5	71	39

   What is voltage on bit 6 of Port A/D?

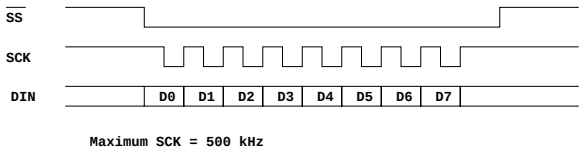
   - ADR6H = 0x71 = 113;

$$V = \frac{ADR}{255} \times (V_{RH} - V_{RL}) + V_{RL} = \frac{113}{255} \times 5V = 2.22V$$

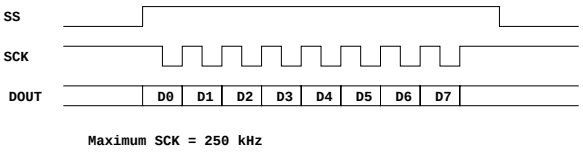
2. A 68HC12 is being used to communicate with two devices over the SPI. The devices are connected as shown below:



Each byte you write to the LCD chip is displayed on an LCD display. The following shows how to write to the LCD chip over the SPI:



The GP2D device is a distance sensor. When the HC12 reads from the GP2D chip, the GP2D sends it a byte which tells it how far it is from an object (such as a wall). The GP2D sends a 0x00 when it is almost touching the object. It sends a 0xFF when it is 2 feet (or more) from the object. The following shows how to read from the GP2D chip over the SPI:



(a) How do you set up the HC12 to communicate with the LCD and the GP2D? Explain what values you need to write to which registers.

- Enable SPI (`1 -> SPE in SP0CR1`)
- Put SPI into master mode (`1 -> MSTR in SP0CR1`)
- Set clock phase and polarity – need CPOL = 1 (clock idle high) and CPHA = 0 (data valid on first edge) in SP0CR1
- Data on both devices comes LSB first (`1 -> LSBF in SP0CR1`)
- Select normal mode (`0 -> SPC0 in SP0CR2`)
- Select clock speed. One device works at 500 kHz, the other at 250 kHz, so set speed to 250 kHz to be compatible with both (`0x04 -> SP0BR`)
- Make SCK and MOSI outputs (can make SS an output also) `0xE0 or 0x60 -> DDRS`
- Make slave select lines PP4 and PP5 outputs (`1 -> DDP4 and DDP5 in DDRP`)
- Deselect slaves by making PP4 high and PP5 low (`1 -> PP4 and 0 -> PP5 in PORTP`)

(b) Write some C code to set up the HC12 to communicate with the LCD chip and the GP2D chip.

```
DDRS = DDRS | 0xE0;    /* or DDRS = DDRS | 0x60; */
DDRP = DDRP | 0x30;
PORTP = (PORTP | 0x10) & ~0x20;  /* Deselect slaves */
SP0CR1 = 0x59;         /* or SP0CR1 = 0x5B; */
SP0CR2 = 0x00;         /* Normal mode */
SP0BR = 0x04;          /* 250 kHz clock */
```

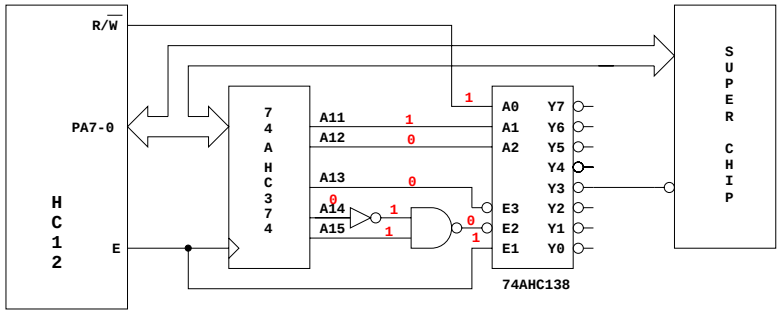
(c) Write some C code to read the distance from the GP2D. Make sure the LCD chip is deselected while you are doing this.

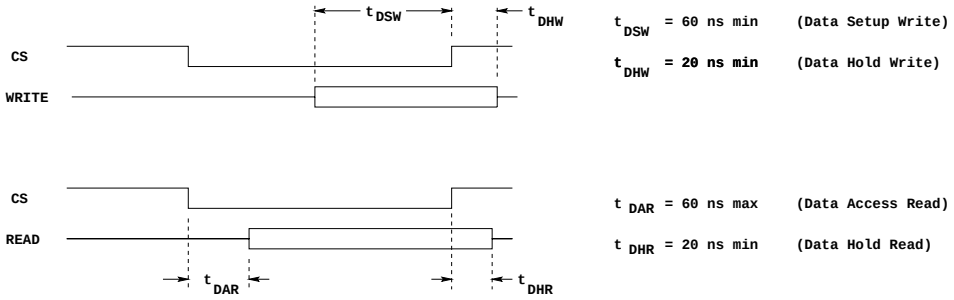
```
char distance;           /* Variable to save distance value */

PORTP = PORTP | 0x20;    /* Select GP2D */
SP0DR = 0x00;            /* Write anything to SP0DR to make SCK go */
while ((SP0SR & 0x80) != 0x80) ;   /* Wait for transfer to finish */
PORTP = PORTP & ~0x20;   /* Deelect GP2D */
distance = SP0DR;        /* Read data from GP2D */
```

(d) Explain how the SPIF (SPI Flag) is set. Also, explain how to clear this flag.

- SPIF flag is set after 8 SCK cycles – in master mode, must write data to SP0DR to make SCK go; in slave mode, SPIF set after master has clocked all eight bits into slave.
- Read SP0SR (with SPIF set), then access (read from or write to) SP0DR.

3. An HC12 is connected to a peripheral chip as shown:





(a) Explain the purpose of the 74AHC374 chip.

- Port A acts as a multiplexed address and data bus. While E is low, the HC12 puts address lines A15-A8 on Port A. While E is high, Port A acts as D15-D8 – when writing, the HC12 puts the data to write on Port A, the peripheral chip should latch the data on the falling edge of E; when reading, the peripheral chip puts the data to read on Port A, and the HC12 reads it on the falling edge of E.

- The 74AHC374 chip acts as an address demultiplexer – it latches the address lines A15-A8 on the rising edge of E so the address will be available while E is high

(b) Explain the purpose of the 74AHC138 chip.

- The 74AHC138 decodes the address, E, and R/W to select the appropriate chip – when (on the 138) E1 = 1, and E2 = E3 = 0, one of the outputs of the 138 will go low. Which output goes low depends on A2, A1 and A0 of the 138. Thus, the 138 acts as a partial address decoder to select the chip the HC12 wants to talk to, and to select that chip only when E is high.

(c) For what range of addresses will the Super Chip be selected?

- From the diagram, you must have A15 high, A14 low and A13 low to select the 138. You must have A12 low, A11 high and R/W high to select Y3. The address range is

```
1000 1xxx xxxx xxxx = 0x8800 to 0x8fff
```

(d) Is the Super Chip an input or output chip? How can you tell?

- Since R/W must be high, the chip is select for reads only, and hence is an input chip (remember, input and output are always from the point of view of the HC12)

(e) If the Super Chip is an input chip write some C code which will read a byte of data from the chip and store it in a variable called `data`. If the Super Chip is an output chip write some C code which will write a `0xa5` to it.

- The HC12 is an input chip, so you need to read a byte from address 0x8800 - 0x8fff (even addresses only, since even addresses are accessed on Port A, odd addresses on Port B). You need to do this with a pointer in C:

```
char data;
data = *(char *)0x8800;
```

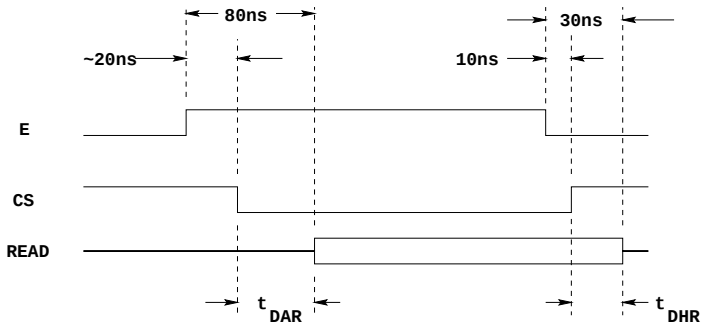
or

```
#define SUPER_CHIP (*(char *)0x8800)
char data;

data = SUPER_CHIP;
```

or

```
char data, *ptr;

ptr = (char *)0x8800;
data = *ptr;
```

(f) On the above figure two timing diagrams are shown. Only one of them applies to the Super Chip – the upper one if the Super Chip is an output device; the lower one if the Super Chip is an input device. Consider the relevant diagram from your answer to the previous part. Based on the timing diagram is the Super Chip compatible with the HC12? Explain in detail – draw a timing diagram which shows how the chip select is generated from the HC12 bus signals, and explain why the chip is or is not compatible based on the times in the figure above. Assume the propagation delays for the 74AHC374 and 74AHC138 chips are 10 ns.



- After E goes high it takes one delay for address to get through 374, and one more delay for 138 to select chip, so after E goes high, it takes 2 delays (about 20 ns) before CS goes low. From E going high until SuperChip puts data on bus is 2 delays + t_{DAR}, or 20ns + 60ns or 80ns. HC12 needs data on bus 30ns after E goes high (circled time 23 on HC12 timing diagram). The SuperChip cannot get its data on bus fast enough, so SuperChip will not work. If you add one E-clock stretch, circled time 23 goes to 30ns + t_{cyc} = 30ns + 125ns = 155ns. Now the SuperChip will work.
- After E goes low, it takes one delay for 138 to become deselected, so CS goes high 10 ns after E goes low. SuperChip holds data on bus for t_{DHR} after CS goes high, so data will stay on bus for 10ns + 20ns = 30ns after E goes low.
  HC12 needs data to stay on bus for circled time 12, or 0ns, after E goes low (i.e., SuperChip cannot remove data before E goes low). Since SuperChip holds data for 30ns after E goes low, SuperChip will work with HC12 based on this time.

4. The following problem deals with interrupts.

   (a) How do you set up the IRQ interrupt to respond to a falling edge, and enable the IRQ interrupt? I.e., what values to you write to which registers?

   - To respond to interrupts on the falling edge, `1 -> IRQE of INTCR`
   - To enable specific interrupt, `1 -> IRQEN of INTCR`
   - To enable maskable interrupts in general, clear I bit of CCR

     ```
     INTCR = 0xC0;
     enable();
     ```

   (b) Explain the differences between the IRQ interrupt and the XIRQ interrupt.

   - IRQ has a specific mask (IRQEN bit of INTCR), XIRQ does not
   - IRQ can respond to an edge or a level (IRQE bit of INTCR); XIRQ responds only to a level
   - XIRQ is enable by X bit of CCR; IRQ is enabled by I bit of CCR (as well as IRQEN bit of INTCR)
   - Once X bit of CCR is cleared, it cannot be set again – i.e., once XIRQ is enable, you cannot disable it. Once I bit of CCR is cleared to enable maskable interrupts, it can be set back to 1 to disable maskable interrupts.
   - XIRQ has a higher priority than IRQ – if the HC12 receives IRQ and XIRQ at the same time, it will respond to the XIRQ interrupt first.