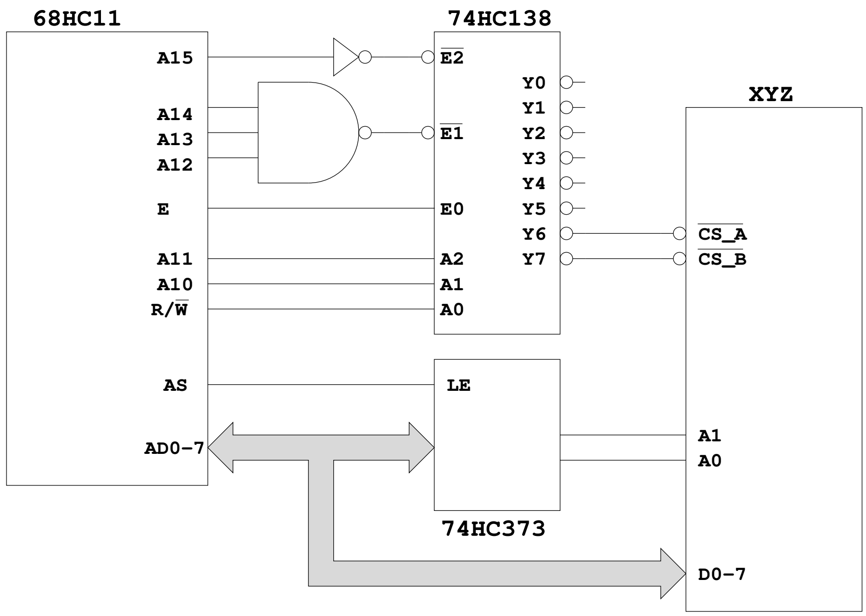
EE 308 - Exam 3

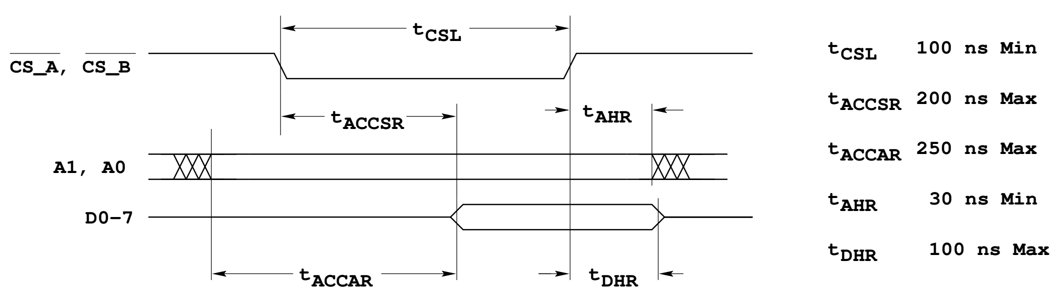
April 13, 1998

Name: ____

You may use any of the Motorola data books, and one page of notes. No calculators allowed. Show all work. Partial credit will be given. No credit will be given if an answer appears with no supporting work.

1. A MC68HC11 with a 2 MHz E-clock is interfaced to a XYZ chip as shown below. Assume the propagation times through the address decoding and demultiplexing chips are all 15 ns. Note that there are two different chip selects on the XYZ chip.





(a) For what range of addresses will XYZ be selected?

`A15-A10 must all be high to select either` $\overline{\texttt{CS\_A}}$ `or` $\overline{\texttt{CS\_B}}$`. Thus, the address range is 0xffc0 to 0xffff.`

(b) What is the difference between $\overline{\texttt{CS\_A}}$ and $\overline{\texttt{CS\_B}}$? I.e., when will $\overline{\texttt{CS\_A}}$ be selected, and when will $\overline{\texttt{CS\_B}}$ be selected?

$\overline{\text{CS\_B}}$ goes low when R/$\overline{\text{W}}$ is high, and $\overline{\text{CS\_B}}$ goes low when R/$\overline{\text{W}}$ is low. Thus, $\overline{\text{CS\_B}}$ is an input port (active on read) and $\overline{\text{CS\_A}}$ is an output port (active on write).

(c) Write a C-language define which will allow you to access XYZ.

```
# define XYZ ((* unsigned char *) 0xffc0)
```

(d) Write a line of C code which will write the number `0xaa` to XYZ using your define from Part c.

```
XYZ = 0xaa
```

(e) Explain in words what t_{ACCSR} means. Will this parameter be met with the interface circuitry shown? Explain, using the circled numbers and the times from the Motorola Technical Data Manual.

Times are for the 2.0 MHz 68HC11E9, and come from Column 2 of Table A-7 on Page A-18 of the Technical Data Manual.

Time from device selected ($\overline{\text{CS\_B}}$ going low) until XYZ puts data on data bus during read. $\overline{\text{CS\_B}}$ goes low one propogation time after E goes high, so this time should not exceed $\boxed{35}$ - t_{prop}, or 192 ns - 15 ns = 178 ns. t_{ACCSR} of 200 ns exceeds this time, so is not compatible with the HC11.

(f) Explain in words what t_{ACCAR} means. Will this parameter be met with the interface circuitry shown? Explain, as above.

Time from address lines $A1$, $A0$ valid until XYZ puts data on data bus during read. Address lines are valid on outputs of 74HC373 one propogation delay after they come out of the HC11, so this time should not exceed $\boxed{29}$ - t_{prop} = 307 ns - 15 ns = 292 ns. t_{ACCAR} does meet this requirement.

(g) Explain in words what t_{CSL} means. Will this parameter be met with the interface circuitry shown? Explain, as above.

Time $\overline{\text{CS\_A}}$ or $\overline{\text{CS\_B}}$ must be low on an access. These chip selects become low one propogation delay after E goes high, and go back high one propogation delay after E goes low, so the time is at least $\boxed{3}$ - t_{prop} + t_{prop} = 222 ns. These times must be at least 100 ns, so this requirement is met.

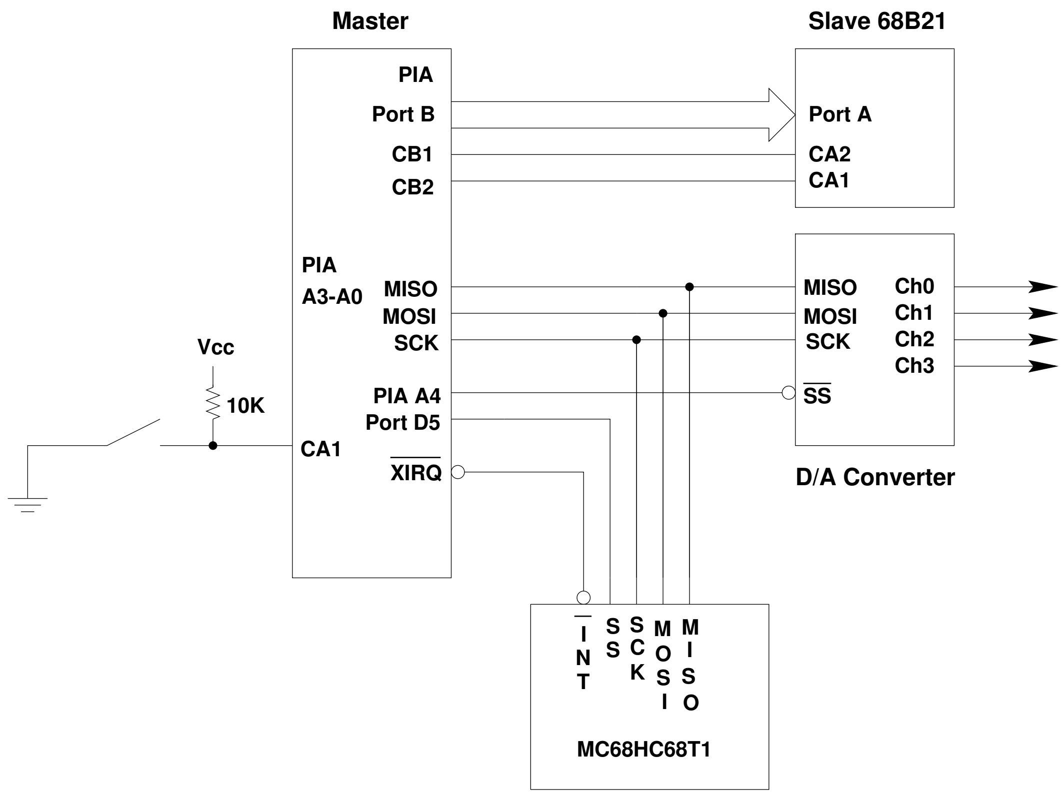
(h) Explain in words what t_{AHR} means. Will this parameter be met with the interface circuitry shown? Explain, as above.

Time address $A1$, $A0$ are held on the address bus after the chip is deselected. $A1$ and $A0$ will not change until one propogation delay after the latch enable input on the 74HC373 goes high. Thus the time is at least $\boxed{26}$ + t_{prop} = 53 ns + 15 ns = 68 ns. Thus, the actual t_{AHR} exceeds the 30 ns minimum, and the requirement is met.

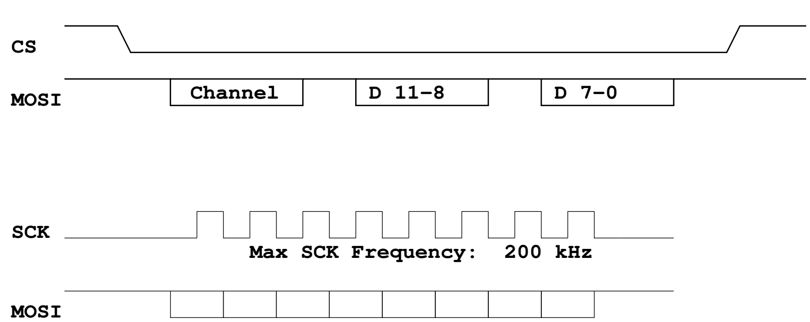
(i) Explain in words what t_{DHR} means. Will this parameter be met with the interface circuitry shown? Explain, as above.

Time the XYZ holds the data on the data bus after the chip is deselected. The HC11 needs the data to be removed from the bus no longer than $\boxed{18}$, or 83 ns after E goes low. One propogation delay after E goes low, $\overline{\text{CS\_B}}$ goes high, and t_{DHR} later, the XYZ removes the data from the bus. Thus, the XYZ chip may hold data the bus t_{prop} + t_{DHR} = 15 ns + 100 ns = 115 ns after E goes low. Since the HC11 needs the bus released in 83 ns, this time is too long, and is not compatible.

2. An HC11 is connected to several other devices through its 68B21 PIA and its SPI interfaces. Assume the Master HC11 has a 2 MHz E-clock. Besides the 68HC68T1 Real Time Clock and another 68B21 PIA, the HC11 is connected to a four-channel, 12-bit Digital-to-Analog converter. To send data to the D/A converter the HC11 has to send three bytes – the two-bit address, the four most significant bits of the data, and the eight least significant bits of the data. The RTC uses a crystal frequency of 1.048576 MHz. You want the RTC to generate a periodic interrupt 16 times a second, and to generate a 1 Hz square wave on its CLKOUT pin.



SPI Timing for D/A Converter



(a) Write some C code to set up PIA Port B: All eight bits of Port B are outputs, using pulse-mode handshaking, with no interrupt enabled on CB1.

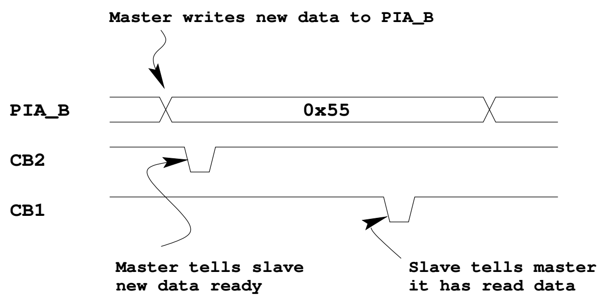
CB5=1 (CRB2 output), CB4=0 (handshaking), CB3=1 (pulse), CB2=1 (access data port), CB0=0 (no interrupt on CB1). CB7, CB6 and CB1 are don't cares. Thus, PIA_CRB should be XX1011X0. 0x2c and 0x2e are acceptable answers.

```
PIA_CRB = 0x00;   /* Access DDRB */
PIA_DDRB = 0xff;  /* All bits of Port B output */
PIC_CRB = 0x2c;   /* Access Port B, set up as described above */
```

(b) How is the interrupt flag for CB1 cleared?

Read PIA Port B

(c) Draw a simple timing diagram showing what is on PIA Port B, CB2, and CB1 when the HC11 sends the slave PIA the byte `0x55`, and the slave acknowledges that it has read the byte.



(d) Write some C code to set up PIA Port A: Port A Bits 7-4 outputs, Bits 3-0 inputs, CA1 an input with interrupt enabled on the rising edge, and CA2 a general-purpose output with a value of 1.

CA5=1 (CRA2 output), CA4=1 (general purpose), CA3=1 (CB2 high), CB2=1 (access data port), CB1=1 (CA1 active on rising edge), CB0=1 (interrupt on CA1). CB7 and CB6 are don't cares. Thus, PIA_CRA should be XX111111.

```
PIA_CRA = 0x00;   /* Access DDRA */
PIA_DDRA = 0xf0;  /* Bits 7-4 outputs, 3-0 inputs */
PIC_CRA = 0x3f;   /* Access Port A, set up as described above */
```

(e) When a rising edge occurs on CA1, what bit of what PIA register will be set?

Bit 7 of Control Regester A (PIA_CRA)

(f) How are the flags cleared in the 68B21 PIA? Write some C code to clear the interrupt flag associated with CA1.

Read the associated data port.

```
temp = PIA_A;
```

(g) Write some C code to set up the HC11 to talk to the D/A converter and the Real Time Clock over the SPI. (Note: no SPI interrupt should be generated in the master.)

A little tricky. The D/A converter runs at a maximum clock frequency of 200 kHz, so the SPI clock frequency should be set to 125 kHz. The D/A uses a clock polarity of 0 and a clock phase of 0. The RTC can use either polarity (so set to 0), but needs a clock phase of 1. Thus,

```
it is necessary to change CPHA before selecting the device you want
to talk to.  I accepted either a 1 or a 0 for CPHA. Thus we want 0101X110
for the RTC and 01010010 for the D/A converter:
   DDRD |= 0x38;
   SPCR = 0x56;  /* For talking to the RTC */
   or
   DDRD |= 0x38;
   SPCR = 0x52;  /* For talking to the D/A */
```

(h) How is the SPI flag cleared? Write some C code to clear the SPI flag.

```
Read the SPSR, then access (read or write) the SPDR:
temp = SPSR;
temp = SPDR;
```

(i) Write some C code to set up the RTC as described: using a crystal frequency of 1.048576 MHz, with a periodic interrupt 16 times a second, and a 1 Hz square wave on its CLKOUT pin.

```
   Turn on RTC: Clock Control Register D7=1; RTC uses xtal:  D6=0;
1.048576 MHz xtal:  D5-D4 = 10; CLKOUT of 1 Hz:  D2-D1-D0=101.  Thus,
Clock Control Register = 1010X101.  For a periodic interrupt with a
16 Hz rate, the Interrupt Control Register should be 8.
   SPCR = 0x56;      /* Make sure CPHA = 1 on SPI */
   PORTD |= 0x20;    /* Select RTC */
   SPDR = 0xB1;      /* Select RTC Clock Control Register */
   while((SPSR&0x80)==0) ;   /* Wait for transfer to finish */
   SPDR = 0xa5;      /* Clock Control Register = 1010x101 */
   while((SPSR&0x80)==0) ;   /* Wait for transfer to finish */
   SPDR = 0x08;      /* Interrupt Control Register = 0x08 */
   while((SPSR&0x80)==0) ;   /* Wait for transfer to finish */
   PORTD &= ~0x20;   /* Deselect RTC */
```

(j) How does the XIRQ interrupt differ from the IRQ interrupt? Write some code to enable the XIRQ interrupt on the HC11.

```
   I accepted any of the following:  The XIRQ mask is the X bit of
the CCR, while the IRQ mask is the I bit of the CCR. The XIRQ is generated
by the XIRQ pin (pin 18) of the HC11, the IRQ by the IRQ pin (pin 19)
of the HC11.  The XIRQ is the highest priority interrupt, so it will
be executed before the IRQ interrupt if the two are asserted simultaneously.
Once enabled, the XIRQ interrupt cannot be disabled.
   To enable XIRQ, clear the X bit of the CCR:
   asm("  tpa");
   asm("  anda #0xbf");
   asm("  tap");
```

(k) Write an XIRQ interrupt service routine. In your routine you should send the data 0x123 to Channel 2 of the D/A converter.

```
SPCR = 0x52;     /* Make sure CPHA = 0 on SPI */
PIA_A &= ~0x10;  /* Select D/A */
SPDR = 0x02;     /* Select Channel 2 of the D/A */
while((SPSR&0x80)==0) ;   /* Wait for transfer to finish */
SPDR = 0x01;     /* Send 4 MSB of data */
while((SPSR&0x80)==0) ;   /* Wait for transfer to finish */
SPDR = 0x23;     /* Send 8 LSB of data */
while((SPSR&0x80)==0) ;   /* Wait for transfer to finish */
PIA_A |= 0x10;   /* Deselect D/A */
```

(l) Write an IRQ interrupt service routine for the PIA. In the routine you should send a 0xaa to the Slave PIA. You do not need to wait for the slave PIA to respond.

Since the interrupt came on the PIA, and only CA1 was set up to generate an interrupt, the interrupt should be cleared by reading PIA_A. The data is written to the slave on PIA_B:

```
#pragma interrupt_handler irq_isr
void irq_isr(void)
{
   char temp;
   temp = PIA_A;
   PIA_B = 0xaa;
}
```