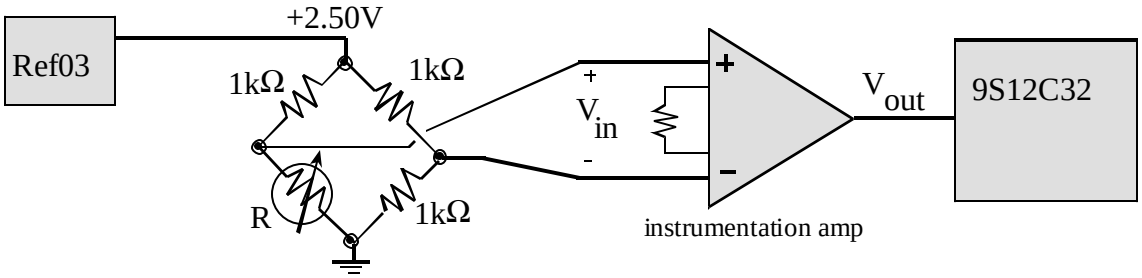
Jonathan W. Valvano First Name: _____ Last Name:_____
December 12, 2004, 7-10pm

This is an open book, open notes exam. You must put answers on these papers; please don't turn in any extra sheets.

The first three questions involve the development of three different solutions to the same problem. The common objective of three systems is to measure resistance with a range of 1000 to 2000 ohms with as good a resolution as possible.

(10) Question 1. The first solution technique involves a resistance bridge and instrumentation amplifier. The ADC is 10-bits and has a range of 0 to +5V.

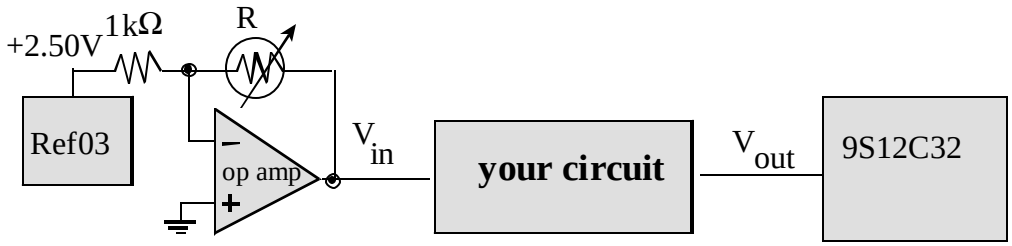


Part a) What is range of voltages appearing on V_{in}?

Part b) What gain is needed on the instrumentation amp?

Part c) What will be resistance measurement resolution?

(10) Question 2. The second solution technique involves a constant current source and linear amplifier. The ADC is again 10-bits and has a range of 0 to +5V.

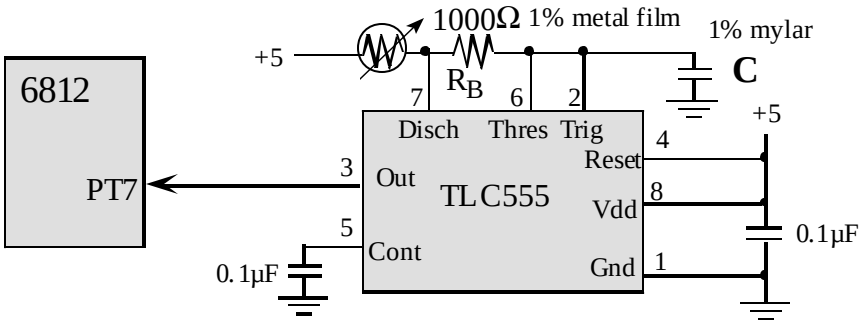


Part a) This is a constant current source. How much current flows across the resistor R?

Part b) What is the linear equation required to map the full scale range of R into the full scale range of the ADC?

Part c) What will be resistance measurement resolution?

(10) Question 3. The third solution technique involves an astable multivibrator and period measurement. If the C is the capacitance in μF and the resistance in Ω, then the period (in μs) of a TLC555 timer is 0.693•C•(R+2R_B). In this circuit R_B is a 1000Ω fixed resistor.



Part a) Assume the TCNT is running at 4MHz, therefore its period is 250ns. What is the largest period that can be measured with a single channel of input capture?

Part b) Choose a value for the capacitor, C, so that the period is that maximum value when the input resistance is 2000 Ω.

Part c) What will be resistance measurement resolution?

(5) Question 4. The 6812 is running in expanded mode with 4 megabytes of extended program page ROM. Fill in the two boxes in the following software that reads physical location $35678.

```
PPAGE = [          ];
data = *((char *)([          ]));
```

(10) Question 5. The motor voltage is +12 V and its current is 0.25 A. Show the hardware interface circuit that allows you to rotate the motor in both directions. PT0 is a pulse-width modulated wave to specify the amount of power, and PT1 controls the direction.

(10) Question 6. A 74C74 CMOS D flip-flop clocks its **Data** input on the rising edge of its **Clk**. For this chip, the setup time is 50 ns, and the hold time is 20 ns. A system using the 74C74 provides valid information to the **Data** input during this interval:

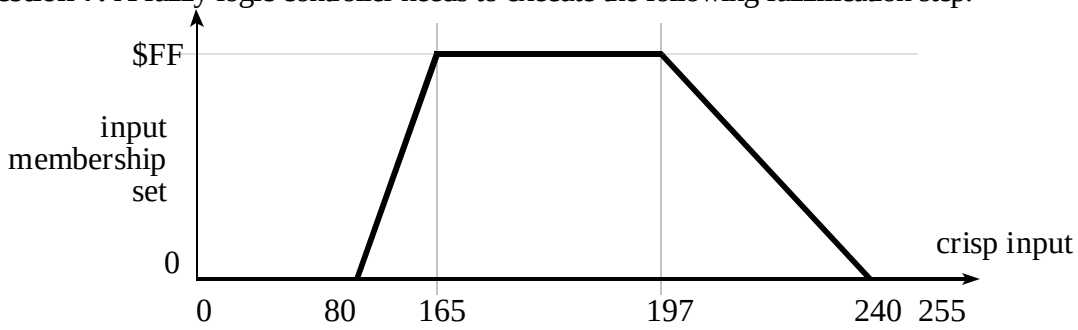
Data = ([80,100], [200,220])

and the rising edge of the **Clk** occurs at

?**Clk** = [170,190]

Will the data be properly stored? Use timing analysis to justify your answer.

(10) Question 7. A fuzzy logic controller needs to execute the following fuzzification step.



Part a) What four constants should be stored in the ROM-based structure to allow the controller to execute this fuzzification with the **`mem`** instruction? Hint: your answer will be approximate, so round to the closest integer. Your answer will be similar in format to Program 13.19 from the book.

Part b) There are three input membership sets: **`Happy`**, **`Healthy`**, and **`Wise`**. There is one output membership set, **`Good`**. Consider the Fuzzy logic equation

```
Good = Happy*(Healthy+Wise)
```

Assume **`Happy`** =100, **`Healthy`** =200, and **`Wise`** =50. Calculate the value of **`Good`** for this case.

Part c) Show the ROM-based data structure to define this Fuzzy logic equation so it could be executed using the **`rev`** instruction. Your answer will be similar in format to Program 13.21 from the book. You may assume the following RAM-based global variables

```
        org  $3800
; membership variables
HAPPY:   ds   1
HEALTHY: ds   1
WISE:    ds   1
GOOD:    ds   1
; input membership variables relative offsets
happy:   equ  0
healthy: equ  1
wise:    equ  2
good:    equ  3
```

(10) Question 8. Design an analog circuit that implements

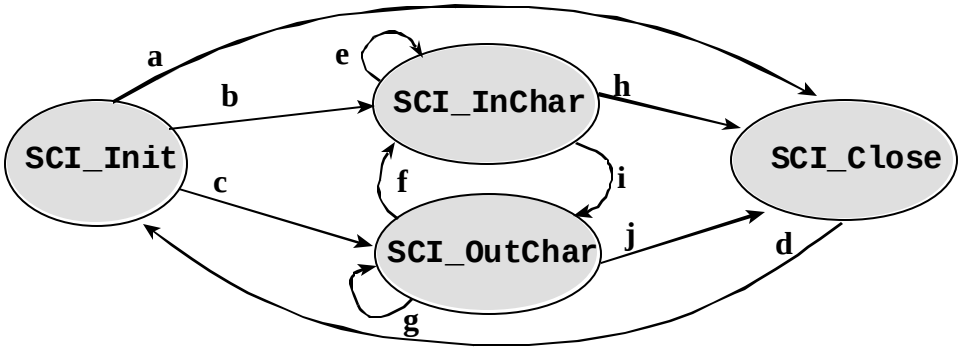
$$V_{out} = 4V_2 + 2V_1 + V_0$$

using one op amp. Show your work.

(25) Question 9. Path expression is a formal mechanism to specify the correct calling order in a group of related functions. Consider a SCI device driver with 4 functions, the prototypes are

```
void SCI_Init(void);          // Initialize Serial port
char SCI_InChar(void);        // Wait for new serial port input
void SCI_OutChar(char data); // Output 8-bit to serial port
void SCI_Close(void);         // Shut down serial port
```

It is obvious that you should not attempt to input/output until the SCI is initialized. In this problem, we will go further and actually prevent the user from executing **`SCI_InChar`** and **`SCI_OutChar`** before executing **`SCI_Init`**. A directed graph is a general method to specify the valid calling sequences. An arrow represents a valid calling sequence within the **path expression**. The system "state" is determined by the function it called last. For this example, we begin in the closed state, because the SCI is initially disabled. The tail of an arrow touches the function we called last, and the head of an arrow points to a function that we are allowed to call next.



In this method, a calling sequence is valid if there is sequence of arrows to define it. E.g.,

```
Init InChar InChar OutChar Close      d b e i j
Init OutChar OutChar OutChar OutChar  d c g g g
Init Close Init InChar Close          d a d b h
```

On the other hand, the following calling sequences are illegal because each has no representative sequence of arrows

```
Init InChar Init OutChar Close    can't initialize twice
Close                             can't close because already disabled
OutChar OutChar OutChar           can't output without initialization
```

A fast, but memory inefficient method, to represent a directed graph uses a square matrix. Since there are four functions, the matrix will be 4 by 4. The row number (0,1,2,3) will specify the current state (the function called last), and the column number (0,1,2,3) will specify the function that might be called next. The values in the matrix are true(1)/false(0) specifying whether or not the next function call is legal. Since there are 10 arrows in the directed graph, there will be exactly 10 true values in the matrix, one for each arrow. The remaining values will be false(0).

Part a) Fill in 15 values for to define the **Path Expression** for this system. E.g.,, **`Path[3][0]`** is arrow "**d**" in the above figure.

```
int Path[4][4]={ /*   Init    InChar  OutChar   Close */
/*           column     0       1       2        3    */
/* Init    row 0*/ {        ,       ,       ,       },
/* InChar  row 1*/ {        ,       ,       ,       },
/* OutChar row 2*/ {        ,       ,       ,       },
/* Close   row 3*/ {   1    ,       ,       ,       }};
```

Part b) You may assume the system is running under a preemptive thread scheduler, and exactly one thread will be calling these four functions, so there are no critical sections to worry about. Add code to the following set of SCI functions that implement **Path Expression**. You will use the following global variable, which defines and initializes the current state:

```
int State=3;  // start in the Closed state
```

E.g., `Path[3][0]`, will be true signifying it is OK to call `SCI_Init` if the SCI is disabled. You may assume there is an operating system function called `OS_Kill()`, which should be called if the thread makes an illegal function call. I.e., you should not write `OS_Kill`. Rather you should call it when needed, which will destroy the thread because it has a serious programming error.

```
void SCI_Init(void){

  SCIBD = 13;

  SCICR1 = 0;

  SCICR2 = 0x0C;

}


char SCI_InChar(void){

  while((SCISR1 & RDRF) == 0){};

  return(SCIDRL);

}


void SCI_OutChar(char data){

  while((SCISR1 & TDRE) == 0){};

  SCIDRL = data;

}


void SCI_Close(void){

  SCICR2 = 0x00;

}
```