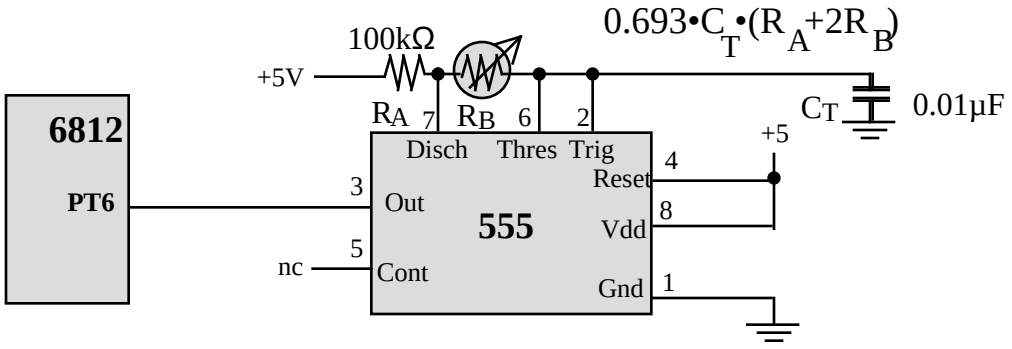
Jonathan W. Valvano First:_____ Last:_____
July 3, 2000, 2:30pm-3:45pm

This is an open book, open notes exam. You must put your answers on these pages only, you can use the back. You have 90 minutes, so please allocate your time accordingly.
Please read the entire quiz before starting.

(50) Question 1. In this problem you will design a digital thermometer. The goal is to measure temperature, 95 ≤ **T** ≤ 105 ˚F. The desired temperature resolution, Δ**T**, is 0.1 ˚F. The temperature transducer, a thermistor, produces a resistance that is linearly related to its temperature. Actually the response is quite nonlinear, but we will assume a linear response for this narrow range. The 555 astable multivibrator produces a period that is also linearly related to temperature, **T**:

T = 174.88 - 0.0528•**P** (where, **P** is in μs and **T** is in ˚F)



Therefore, the period range is 1324 to 1514 μs. You will build this system using the input capture system and **period measurement**. You will use input capture channel PT6, to measure period. You will use output compare channel PT5 to handle the special case when the transducer is disconnected and the period is larger than 3000 μs. The two interrupt service routines (PT5 and PT6) will measure temperature and continuously update the following global variables with the current temperature measurement.

```
unsigned short T; // temperature, range is 950 to 105 and resolution is 0.1 F
int bOK;    // true if T contains a valid measurement, false if T is invalid
```

The following table shows the corresponding temperature, resistance and period values:

T (˚F)	R (kΩ)	p (μs)	Software T
95.0	59.212	1514	950
95.1	59.076	1512	951
100.0	52.377	1419	1000
104.9	45.677	1326	1049
105.0	45.541	1324	1050

(5) Part a) What period measurement resolution do you need?

(5) Part b) Rewrite the equation, **T**=174.88-0.0528•**P**, converting the units of **T** to 0.1 ˚F and converting the units of **P** to the period measurement resolution defined in part a). Then convert the equation so that it can be calculated with integer arithmetic (no floating point will be allowed.) The trick is to convert the equation so the exact calculation is performed, but no intermediate result goes above 32,767.

(15) Part c) Write the ritual subroutine that initializes the interface. Please define and initialize any additional data structures that you require. Initialize the flag, `bOK`, to false, and set it to true after each valid temperature measurement.

(25) Part d) Write the interrupt service routines that measure temperature. Don't worry about the interrupt vectors, just use names like `TC5handler()` and `TC6handler()`. Set `bOK` to false if the period goes above 3000 µs, and set it to true after the next valid measurement. No floating point allowed.

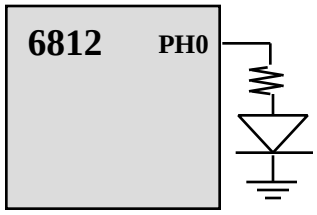
(10) Question 2. In this problem consider the following two C functions.

```
void Ritual0(void){
  DDRH = DDRH | 0x01;   // PORTH bit 0 is an output
}
void Ritual1(void){
  DDRH = DDRH | 0x02;   // PORTH bit 1 is an output
}
```

Part a) Are these two functions friendly? *Be very specific and justify your answer.*

Part b) Do these two functions have critical sections? I.e., consider the sequence where one of these functions is started (running with interrupts enabled), this first function then is interrupted, the other function is called from the interrupt service routine, and after the interrupt service routine finishes the first function is completed. *Be very specific and justify your answer.*

(10) Question 3. Consider the following LED interface to a 6812. Assume the voltage drop across the LED, when active, will be 2 V.



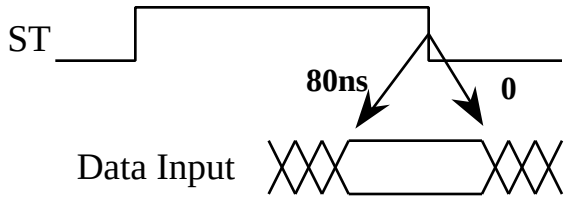
Part a) What is the largest possible LED current that the 6812 can drive in this particular circuit?

Part b) Assuming the LED needs 500 µA to activate, what resistor value should you use?

(40) Question 4. In this problem you will interface a DAC as an output port to the MC68HC812A4. In other words, when software writes to the port address of $0200, the 8-bit data value is copied into the DAC. For example, the following assembly code will set the DAC output voltage to the midrange value:

```
ldaa #$80
staa $0200
```

The 6812 is running at 8 MHz. The DAC control signal, **ST**, will be interfaced to the 6812 using a combination of E, R/W, and CS0. Your interface will be "write-only", participating in write cycles, but not read cycles. The DAC timing is as follows.

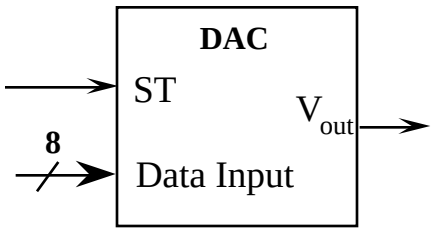


(5) Part a) What 6812 operating mode will you use (e.g., single chip, expanded narrow, expanded wide, or peripheral)?

(5) Part b) Choose which of the following shapes for **ST** will make write data available overlap write data required (See Table 9.8 and Figure 9.28 in Section 9.5.1):

synchronized positive logic
unsynchronized positive logic
synchronized negative logic
unsynchronized negative logic

(10) Part c) Show the digital interface between the 6812 and the DAC. (Don't connect to V_{out})



(10) Part d) How cycle stretches are required? Show your work.

(10) Part e) Show the combined **write cycle** timing diagram. Show **E**, **R/W**, **CS0**, **ST**, **WDA**, and **WDR**. All signals are outputs except Write Data Required. Use arrows to signify causal relations. Show the timing delays as arrows with numbers in nanoseconds. Calculate the actual WDA and WDR intervals.