Jonathan W. Valvano First Name: _____ Last Name:_____
May 16, 2000, 9 am to 12 noon

This is an open book, open notes exam. You may put answers on the backs of the pages, but please don't turn in any extra sheets.

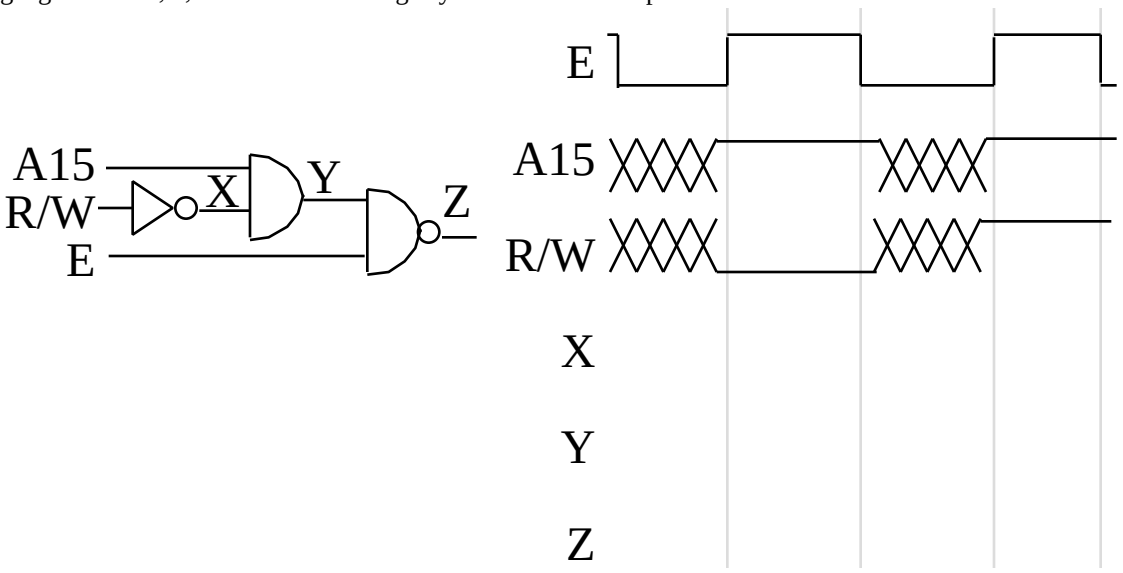
(25) Question 1. In this problem, you will design a 6812-based current meter. The current range is 0 to 1000µA. Because you are using the 8-bit ADC and a linear analog circuit, the measurement precision will also be 8 bits. You will design a linear analog circuit that converts the 0 to 1000µA current into a 0 to +5V signal for the internal ADC. Use at least one op amp so the input impedance will be close to zero.
Part a) What will be the measurement resolution in µA?

Part b) Design the linear analog circuit that converts the current, I, into the 6812 ADC voltage, V_2. Specify chip numbers and component values. You may use any op amp with any power configuration, but there is a way to it with just +5V power.

Part c) Show the ritual that initializes the ADC system (no interrupts, just simple one-time measurements)

Part d) Show the function that samples the ADC and calculates current in µA. No interrupts, just a function.

(10) Question 2. Consider the following digital circuit connected to a 6812 in expanded mode. Assume a 10 ns gate delay through each gate. You may assume the signal E, A15, and R/W are actual outputs from the 6812. Draw the timing signals for X,Y,Z. Use arrows to signify causal relationships.



(15) Question 3. The MC68HC912B32 has hardware support for pulse width modulation. You will be writing software to set up the system in 8-bit left-aligned mode to generate a PWM signal on Port P bit 0 (PP0). The frequency of the PCLK is 8 MHz. You will set up clock source A to have a period of 8 µs (125 kHz) using the prescaler for clock A. You will then select clock source A for channel 0. The output will be high at the beginning of the cycle, and go low when the duty count is reached. The overall period of the waveform will be determined by the value in the register PWPER0. I.e., the period is 8µs•(PWPER0+1). The time duration of the signal being high is determined by the value in the register PWDTY0. I.e., Therefore, the duty cycle will be (PWDTY0+1)/(PWPER0+1).
Part a) Write the ritual that initializes the system with a 1kHz 50% duty cycle waveform on PP0.
Part b) Write a function that changes the duty cycle of the waveform on PP0. To change the duty cycle, all you have to do is write a new value to PWDTY0. The format of the input parameter to this function is 8-bit unsigned binary fixed-point. 0 means 0%, 128 means 50%, and 255 means 99.6%.

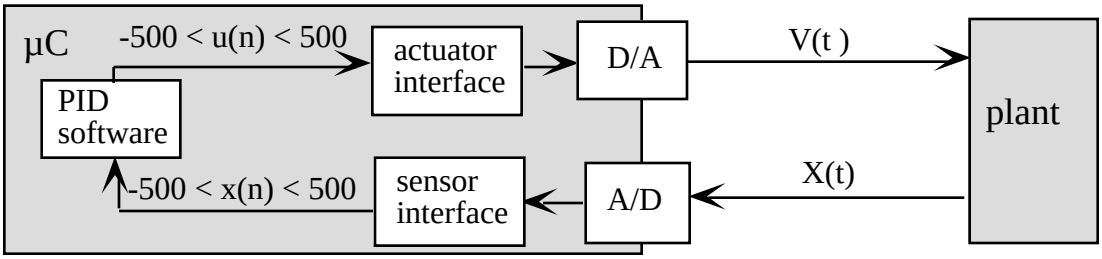
(20) Question 4. The objective of this problem is to develop the fixed-point equations that implement a PID position controller. You are to implement the following control system

X(t) is the state variable, position (µm)
X* is the desired state, position (µm)
e(t)=X*-X(t) is the error, position (µm)
V(t) is the actuator command (mV)

$$V(t)=K_P e(t) + K_I\int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt}$$

where K_P is 2.54 (mV/µm)
K_I is 123.2 (mV/µm-sec)
K_D is 0.00125 (mV-sec/µm)

The state estimator and actuator output hardware/software interfaces are given. Your PID software is given a signed 16-bit integer input, `x(n)`, that represents the current state variable. Let `xstar` be the desired position. Your PID software will calculate a signed 16-bit integer output, `u(n)`, that will be fed to the actuator interface. Note that `u(n)` and V(t) both have units of mV. Similarly, `x(n)` `xstar` and X(t) have units of µm.



Assuming the digital controller is executed every 1ms (1000Hz), show the control equation to be executed in the periodic interrupt handler. In this process you will convert from floating to fixed-point numbers, and convert from continuous to discrete time. Explain how you would deal with overflow/underflow. You should assume some noise exists on the sensor input. No C code is required, but please give every explicit fixed-point equations showing how to calculate `u(n)` in terms of `x(n)` and `xstar`. You may define additional variables as you need them.

(20) Question 5. When the 6812 instruction set was designed, Motorola studied software written for the 6811 and searched for common operations. In this way, they could develop new instructions to optimize these operations. In most cases, the goal was not to provide new operations that couldn't be done in software, but rather to reduce execution speed. In this same manner, your goal is to design two new instructions that will improve performance for real time schedulers using blocking counting semaphores. You will design one instruction for **signal** and one for **wait**.

Part a) First, show the assembly syntax for your two new instructions. In particular, give a couple of assembly language examples showing the new instructions and their operand format(s). E.g., these are assembly language instructions.

```
bne  loop     ; branch to location loop if Z bit is 0
bsr  function ; call subroutine "function"
cmpb 2,x      ; compare RegB to the 8-bit memory contents at x+2
adda [2,x]    ; 2+x is an address. add 8-bit contents at address to RegA
```

Part b) Next, give the very explicit functions performed by the instructions. `addr` is the effective address of the instruction. Use pseudo code to define the instructions. For example, the pseudo code for the above 6812 instructions is

`bne addr`	If Z = 0, then `addr` => PC
`bsr addr`	SP-2 => SP, store return address on stack, then `addr` => PC
`cmpb addr`	B – (`addr`), then sets CCR bits N,Z,V,C
`adda addr`	A + (`addr`) => A, then sets CCR bits N,Z,V,C

Part c) Illustrate how your two new instructions could be used to implement OS_Wait and OS_Signal functions using either flowcharts or C code. Just give a rough idea, don't write the entire blocking OS.

(10) Question 6. The 6812 is running in expanded mode with 512K of extended data page RAM. One particular virtual address is page number $19, byte number $126.
Part a) Fill in the two boxes in the following software that clears this location.

```
DPAGE = [                ];
*((char *)([                ])) = 0;
```

Part b) What is the physical memory address of this byte?

Part c) In this example there is extended data RAM at physical addresses 0 to $7FFFF. How can there be extended data RAM at physical address $0F000 and internal EEPROM at $0F000?