Final Exam

Date: May 12, 2017

UT EID: ________________________ ___________________

Printed Name: ____
Last, First

Your signature is your promise that you have not cheated and will not cheat on this exam, nor will you help others to cheat on this exam:

Signature: ____

Instructions:

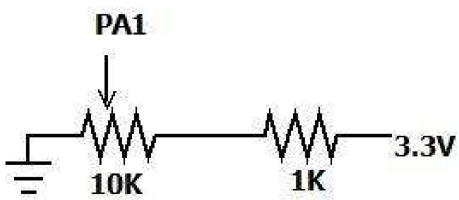
- Closed book and closed notes. No books, no papers, no data sheets (other than the last two pages of this Exam)
- No devices other than pencil, pen, eraser (no calculators, no electronic devices), please turn cell phones off.
- Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space (boxes) provided. Do Not write answers on back of pages
- You have 180 minutes, so allocate your time accordingly.
- Unless otherwise stated, make all I/O accesses friendly.
- Please read the entire exam before starting.

Problem 1	15	
Problem 2	15	
Problem 3	15	
Problem 4	10	
Problem 5	15	
Problem 6	15	
Problem 7	15	
Total	100	

[15 points] Problem 1: Fundamentals. Answer the following short questions in the boxes provided.

(i) (**2pt**) Explain what happens when you execute the following Stack instructions.
PUSH {R0, R2}
POP {R2, R0}

(ii) (**3pt**) Given the following series configuration of a 10kΩ potentiometer and a 1kΩ fixed resistor, what is the range of voltages at the input pin PA1?



(iii) (**1pt**) Complete the following statement: A ___________ local variable is allocated permanent space in the RAM.

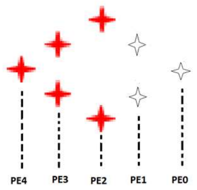
(iv) (**2pt**) The input signal of the ADC has frequency components ranging from 200Hz to 2kHz. What must the sampling frequency of the ADC be to faithfully reproduce the signal?

(v) (**4pt**) Let the bus clock frequency be 80MHz. Calculate the SysTick timer LOAD value so that the timer triggers an interrupt every 1 μs?

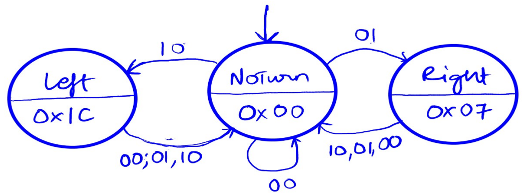
(vi) (**3pt**) A Moore FSM that you implemented in Lab4 repeated a sequence of 4 steps over and over. Complete the procedure below.

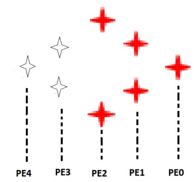
1.
2. Wait (optional)
3.
4.

(15) Problem 2: Finite State Machine. You may recall the bicycle with turn indicators from the first midterm. The outputs are the five LEDs on PB4 to PB0 which flash when indicating a turn:



Left Indication





Right Indication

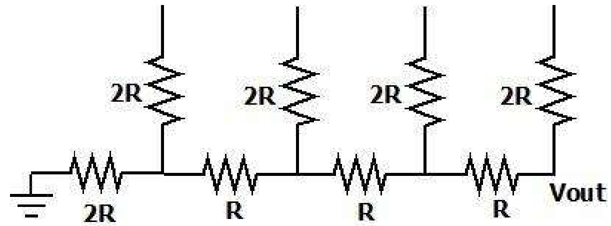
The input was an accelerometer reading which is now abstracted, so you can call the function: `uint8_t get_direction()`, which returns `00, 01 or, 10` to indicate to stay straight, turn right or turn left respectively. You are required to flash the LEDS at 5Hz with 50% duty-cycle. The state-transition graph for a Moore FSM implementation is given above (without the wait times).

Complete the code below by adding state `#defines`, blanks in the struct, FSM array size and entries, state initialization, and the FSM loop.

```
//#defines here


struct State{
  uint8_t out;   // output produced in this state
  uint32_t wait; // delay in ms units;
                 // Can call delayms(count) to wait for count milliseconds
  uint8_t next[___]; // list of next states
};
typedef struct State State_t;
State_t FSM[__] = {






}
uint8_t curState = ________; //set the initial state here
int main() {
      // All Port Initialization done for you
      // Complete the FSM loop below
      …
      while(1){











      }
}
```

[15 points] Problem 3: Hardware.

Part a(10 pt): You are given the following R-2R ladder DAC circuit for a 4-bit DAC connected to the microcontroller port pins PB3 (MSB), PB2, PB1, PB0 (LSB). The output of the DAC is connected to a speaker.



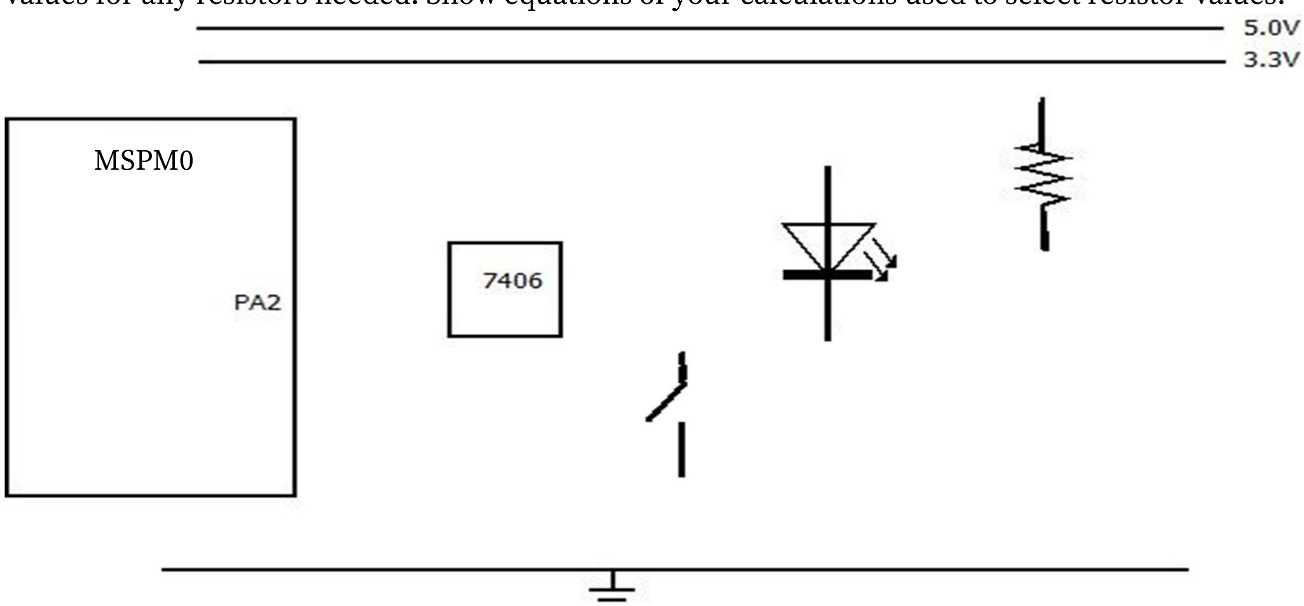
(i) (**4p**t) Mark the microcontroller port pins on the circuit schematic.

(ii) (**3pt**) A few rows in the table below have been completed for you. Complete the rest of the table. (Note that the values of Vout are rounded, and that for an R-2R ladder circuit the exact values of Vout are slightly different).

PB3	PB2	PB1	PB0	Vout (V)
0	0	0	0	0.0
0	0	0	1	
0	0	1	0	0.4
0	0	1	1	
0	1	0	0	
1	1	1	1	

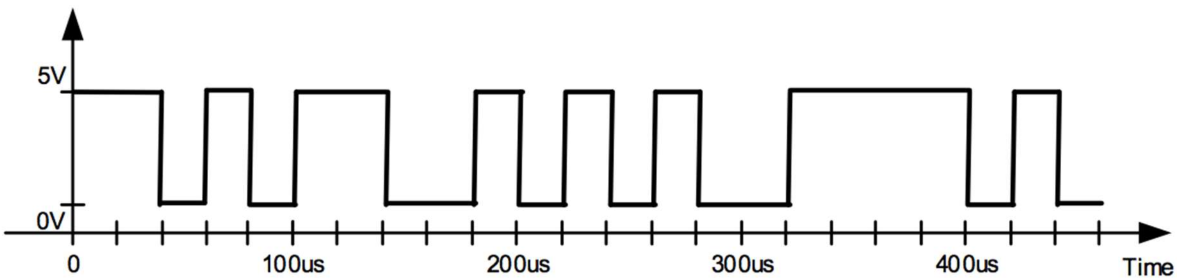
(iii) (**3pt**) What is the range, resolution, and precision of this DAC?

Part b(5pt): You are given an MSPM0 microcontroller, an LED whose desired operating point is 1.6V and 1.5mA, and resistors (of your choice). Interface this LED to PA2 using ***positive*** logic. Show your connections clearly. Assume the microcontroller output voltages are V_{OH} = 3.1V and V_{OL} = 0.1V. Specify values for any resistors needed. Show equations of your calculations used to select resistor values.



[10 points] Problem 4: UART.

Part a (6pt): Assuming start, stop, and data bits only, mark the frame boundaries and data bits. Assume no breaks during transmission (frames are sent back-to-back).



Part b (4pt): What is the baud rate and bandwidth of this channel?

[15 points] Problem 5: FIFO.

Consider a struct named `fix_pt_t` that contains two 8-bit integer fields `whole` and `frac`. Implement a FIFO of `fix_pt_t` elements. The FIFO supports the basic interface: `put`, `get`, `is_empty`, and `is_full`, which are all functions. The function `put` is passed a `fix_pt_t` element by value and `get` removes a value from the FIFO and places it in a parameter passed by reference. Let the maximum FIFO size be a defined constant `MAX_SZ`.

In addition, `get` returns the delay, in terms of number of elements added or removed from the FIFO, between when the element itself was added to the fifo with a `put` and until it is taken out with the `get`. That is, the delay represents how many times `put` or `get` were called between the insertion and removal of an element. Fill out the FIFO code below.

Note: *A correct implementation of the FIFO with struct elements is worth 10 points; implementing the delay feature will get you the full 15.*

```
#define MAX_SZ 32
// Define your struct fix_pt_t here. 1 point.


// global variables for the FIFO. 1 point.
```

// implementations of `put(fix_pt_t elem), uint8_t get(fix_pt_t *elem), is_empty(), is_full().`
// 1 point each for `is_full` and `is_empty`. 4 points each for `put` and `get`.

[15 points] ***Problem 7*: Variable Fundamentals.** The code below is AAPCS compliant. The relevant code given in ARM assembly is complete (main is not given). The C part is very incomplete, but you need not fully complete it nor should you worry about fully understanding the ASM. The question is about variables and types rather than directly on what the code does. Hint bar is called first. Recall that each variable can be local or global and permanent or temporary (e.g., local temporary, global permanent, local permanent).

```
VAR_a: .space 4

Baz:
  cmp  r0, #0
  bge  BazRet
  ldr  r1,= 0xffffffff
  eors r0, r0, r1
  adds  r0, r0, #1
BazRet:
  bx   lr

.equ w,  ???     <---- PartC
.equ z, 4
Foo:
  push {lr,r7,r4,r5}
  sub  sp, sp, #8
  mov  r7, sp

  mov  r4, r0
  mov  r5, #0
LabelFooA
  cmp  r1, r5
  bls  LabelFooB
  ldrsb r0, [r4,r5]
  str  r5, [r7,#z]
  ldr  r5, [r7,#w]
  blx  r5
  ldr  r5, [r2]
  add  r5, r5, r0
  str  r5, [r2]
  ldr  r5, [r7,#z]
  add  r5, r5, #1
  b    LabelFooA
LabelFooB:
  ldr  r4, =VAR_a
  ldr  r0, [r4]
  add  r0, r0, #1
  str  r0, [r4]
  cmp  r3, #0
  bge  LabelFooC
  mov  r0, #0
LabelFooC:
  add  sp, sp, #8
  pop  {lr,r7,r4,r5}
  bx   lr



bar:
  push {lr,r7}
  sub  sp, sp, #8
  ldr  r7, =baz
  str  r7, [sp]
  movs r1,#0x0f
  ands r3, r2, r1  <---- PartA
  bl   foo
  add  sp, sp, #8
  pop  {r7,pc}
```

```
uint32_t baz(???     a) {
   // does something useful and returns
   // see ASM, but do not convert to C
}
???     foo(int8_t A[8],
            ??? // more parameters here
            ) {
  // Code in foo that you need not
  // complete in C. But, line below is
  // part of the code that is important
   ++a;
 // More code
}
uint32_t bar(int8_t A[8],
             uint32_t len,
             // another param
            ) {
 // code that you don't need to complete
}
```

Part a (2pt): What **sort of variable** is r3 marked **PartA** in ASM (e.g., global permanent)?

Part b (3pt): How many parameters does `foo` take?

Part c (2pt): What is ??? marked **PartC** in ASM?

Part d (2pt): What **sort of variable** is **a** inside foo (look at C side, this one is harder)?

Part e(2pt): What is the C99 type of `baz`'s input param?

Part f (2pt): Which of foo's parameters are pass-by-reference (zero or more) -- indicate by parameter number.

Part g (2pt): What is the type of `foo`'s last input param? Use C syntax but explain in words otherwise (this one is harder).