Final Exam

Date: December 14, 2019

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. *You will not reveal the contents of this exam to others who are taking the makeup thereby giving them an undue advantage*:

Signature: _____

Instructions:

- ***Write your UT EID on all pages (at the top) and circle your instructor's name at the bottom.***
- Closed book and closed notes. No books, no papers, no data sheets (other than the last four 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. *Anything outside the boxes will be ignored in grading*.
- You have 180 minutes, so allocate your time accordingly.
- For all questions, unless otherwise stated, find the most efficient (time, resources) solution.
- Unless otherwise stated, make all I/O accesses friendly.
- *Please read the entire exam before starting.* ***See supplement pages for Device I/O registers.***

(5) Problem 1. Variables. Consider the following C program.

```
uint8_t A=5;
const uint8_t B=5;
static uint8_t C=5;
volatile uint8_t D=5;
void func(const int32_t E, int32_t F){
  int32_t G=5;
  int32_t static H=5;
}
```

For each question list all possible variable names. Specify names **`A B C D E F G`** and/or **`H`**. If there are no possible answers, specific NONE

(1) Part a) Which variable is allocated in R1? (for this question give the one answer) ……

(1) Part b) Which variables may be allocated on the stack? (for this question give NONE, one, or more answers) ……

(1) Part c) Which variables are private to (have scope limited to) the function **`func`**? (for this question give NONE, one, or more answers) ……

(1) Part d) Which variables are initialized to 5 when the you **download** object code to the MSPM0, before any software has started? (for this question give NONE, one, or more answers) ……

(1) Part e) Which variable is the best one to use to share information between the main program and software running in an ISR? (for this question give the one answer) ……

(15) Problem 2. Equations. Give the relationships in terms of these parameters: (V_{OL}, output low voltage of MSPM0 in volts), (V, voltage in volts), (R, resistance in ohms), (n, number of bits in the ADC, e.g., 12 bits), (b, baud rate of the UART in bits/sec, e.g., 115200 bps), (*max*, the maximum possible ADC voltage in volts, e.g., 3.3V), (*min*, the minimum possible ADC voltage in volts, e.g., 0V), (r, rate at which one moves the slide pot in oscillations per sec, e.g., 10 Hz), (R, the SysTick LOAD value), (f, the MSPM0 bus frequency in Hz, e.g., 80,000,000 Hz).

(4) Part a) Give the relationship for the **power** dissipated in a resistor.

P =

Units of power =

(4) Part b) Give the relationship for the maximum **bandwidth** possible on a UART.

BW =

Units of bandwidth =

(4) Part c) Give the relationship for the ADC **resolution**.

Resolution =

Units of resolution =

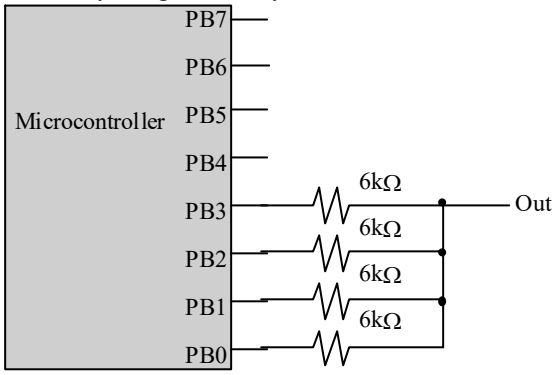
(3) Part d) Give the relationship for SysTick **interrupt period**.

Period =

Units of period =

Integer (binary)	Out (volts)
0000	0.0V
0001	
0010	
0011	
0100	
0101	
0110	
0111	
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	4.0 V

(10) Problem 3. Circuit. Consider this interface circuit. Assume PB3, PB2, PB1, PB0 are digital output representing a binary integer from 0 to 15. Notice all the resistors are the same value. To make the math easier, assume V_{OH} of the microcontroller is 4V, and assume V_{OL} is 0V. Some of the values are filled in. Complete the table showing the relationship between output voltage Out, and the binary integer. Show your work



(5) Question 4. Write an assembly language subroutine that stores a value into an array. The array is a global called **`Buffer`**. Each element of the array is a signed 16-bit integer. The size of the array is 256 elements. There are two parameters to your subroutine. R0 contains the signed 16-bit integer, and R1 contains the index (0 to 255). If R1 is less than or equal to 255, then store the one value into the array at that index. If R1 is greater than 255, do not store into the array.

(5) Problem 5. LED interface. The LED has an operating point of 1.2V, 1 mA. Assume V_{OL} is 0.3V and V_{OH} is 3.2V. Interface the LED to PB0 using negative logic. Show the circuit and label all resistors, and capacitors needed. Show the circuit and the math required to select any resistors needed. No software is required.

(10) Problem 6. Draw the state transition graph for a Moore FSM used to control an LED. There are two inputs and one output. Consider the two inputs as a binary integer, *I*, from 0 to 3. The input will determine the brightness of the LED. More specifically, the duty cycle of the LED should be 100**I*/3 in percent. The time constant of the human's visual processing is about 100 ms. The switch input and LED output are both in positive logic. Each state has a name, an output, a dwell time, and multiple arrows to next states. Just show the graph, no software is required. You may use X to specify 0,1,2 or 3.

(10) Question 7: You are asked to implement a FIFO queue using the following variables. These variable names and types are fixed and cannot be changed. You cannot add additional global or static variables. You can add local variables.

```
int16_t *GetPt;     // pointer to oldest (next to Get)
int16_t *PutPt;     // pointer to free space (next place to Put)
int16_t Buffer[10]; // FIFO can store up to 9 elements in this Buffer
void Fifo_Init(void){
  GetPt = PutPt = Buffer;
}
```

```
// Gets an element from the FIFO
// Input: Pointer to a place that will get
// Output: 1 for success and 0 for failure
//  failure is when the FIFO is empty
uint32_t Fifo_Get(int16_t *pt){
```

```
// Adds an element to the FIFO
// Input: value to be inserted
// Output: 1 for success and 0 for failure
//  failure is when the FIFO is full
uint32_t Fifo_Put(int16_t data){
```

(5) Problem 8. Assume the UART0 has been initialized for busy-wait synchronization. Design **an assembly function** to implement OutChar with these two steps

1) Wait for UART TxFifo to have room, bit 7 of UART0_STAT
2) Write data to the UART_TXDATA to send data

The C prototype for the function is **`void OutChar(char data);`**

(5) Question 9. Translate the following C code to assembly

```
void (*Task)(void);

void SysTick_Init(void(*t)(void)){
  Task = t;
  SysTick->LOAD = 79999
  SysTick->CTRL = 7;
  __enable_irq(); // I=0
}

void SysTick_Handler(void){
  (*Task)();
}
```

```
      .data
      .align 2


      .text
      .align 2
```

(10) Question 10. The subroutine **mySub** has one call by value parameter. There are no return parameters. The one call by value input parameter is AAPCS compliant. A typical calling sequence is

```
        .text
        .align 2
stuff: .long  123       //32-bit constant
start: LDR  R0,=stuff
       LDR  R0,[R0]
       BL   mySub
```

The subroutine allocates two 32-bit local variables, **i** and **j** and uses SP stack pointer addressing to access the local variables. The binding for these two are

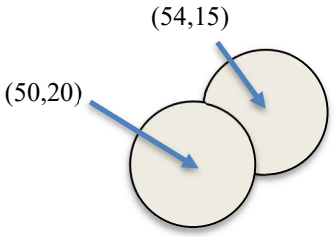
```
.equ i, [          ]  //binding for 32-bit local variable
.equ j, [          ]  //binding for 32-bit local variable
mySub:
      [                    ]//allocate i,j
      PUSH {LR}
//---------start of body-------------------
      [                    ]//set i = input parameter
    LDR  R3,[SP,#i]  //Reg R4 is the input parameter value
    STR  R3,[SP,#j]  //save parameter into local j
//---------end of body---------------------
    POP  {R3}
      [                    ]//deallocate i,j
    BX R3
```

In the boxes provided, show the binding for the two local variables, the assembly code to allocate the two local variables, the assembly code to set i equal to the input parameter, and the assembly code to deallocate the two local variables.

(5) Question 11: You are attempting to capture a sinusoidal sound with a frequency of 1 kHz. The ADC is initializes to take one 12-bit sample. Using the 12-bit ADC and periodic interrupt, you have programmed the SysTick to interrupt at a frequency of 12 kHz. During the SysTick ISR you collect one ADC sample. Is it possible to recreate the original signal from the captured samples? If your answer is *yes*, explain how. If your answer is *no*, what is the term used to refer to this loss of information?

(15) Problem 12. Consider a game that has 32 circles. There is an array of sprites (**`Balls`**) specifying the current status of each circle. Each circle has a radius of 4 pixels, and has an (x,y) coordinate of the center of the circle, two velocities, and a life parameter. The circles are moving according to the two velocities. You may assume the **`Balls`** array has been populated with data before your function is called. Two circles are touching if the distance from one center to the other center is less than or equal to 8 pixels. The figure on the right shows one example with two circles at (x,y)=(50,20) and (54,15). These circles are touching because sqrt(4*4+5*5) = sqrt(41) is less than or equal to 8 pixels. Hint: you do not need floating point or square root to solve this problem.

```
typedef enum {dead,alive} status_t;
struct sprite {
  int16_t x;        // x coordinate, in pixels
  int16_t y;        // y coordinate, in pixels
  int16_t vx;       // x velocity, in pixels/frame
  int16_t vy;       // y velocity, in pixels/frame
  status_t life;};  // dead or alive
typedef sprite sprite_t;
sprite_t Balls[32]
```



Implement a C function that searches to see if two alive circles are touching. If two **`alive`** circles are touching, invert the sign of the x velocities of both circles. Do not worry about 3 or more circles touching at the same time.

```
void Collisions(void){
```