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).***
- 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 120 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) Allocation. Consider this piece of C code. For each item, specify where it is located.
A) Permanent in RAM B) Permanent in nonvolatile ROM C) Temporary in R7
D) Temporary in R0 E) On the stack F) Not stored at all on computer

```
#define SIZE 10 ----------SIZE-
```
F

```
static uint32_t *pt; ----------pt-----
```
A

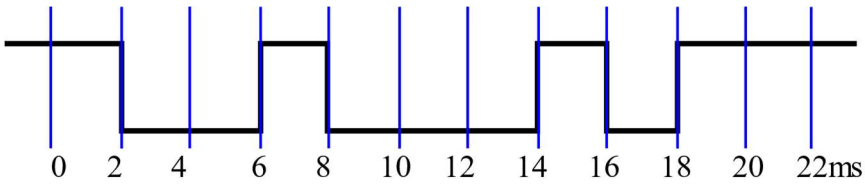
```
uint32_t Operate(uint32_t y){ ----------y-----
```
D

```
  uint32_t buffer[SIZE]; ----------buffer--
```
E

```
  return 0; ----------0-----
```
D

```
}
```

(5) Question 2) UART. The following UART transmission was recorded. It is one frame.
(3) Part a) What is the baud rate? Give units.



500 bps

(2) Part b) What is the 8-bit data value in hex? ----------

0xA2

(15) Question 3) UART output. It is possible to communicate between microcontrollers using the UART protocol even if the microcontroller does not have UART hardware. You will write C language function to implement the transmitter using PB7. You may assume PB7 is already initialized as a regular GPIO output pin. To implement the transmitter, you simply write the **start, b0, b1, b2, b3, b4, b5, b6, b7, stop** bits to the PB7 pin with a fixed delay. This is called bit-banging. Assume the baud rate **BR** is the same as your answer to Q2 part a). To create the baud rate, you are given a **Delay** subroutine that delays exactly 1/**BR**. You will call the **Delay** subroutine 10 times, once for each bit of the frame. Make sure the idle condition (after the stop and before the next start) is correct. How to check your answer: let **z** be the 8-bit data value for your answer to Q2 part b). If you were to call this **UART_OutChar(z)** the output on PB7 should be the same as the figure in Q2. You should NOT be accessing any UART registers, just writing to **GPIOB->DOUT31_0.**

```
// GPIOB->DOUT31_0 outputs to all 32 bits of GPIOB
void UART_OutChar(uint8_t data){
  GPIOB->DOUT31_0 &= ~0x80; // start bit
  Delay(); // 0.5ms
  for(int mask=1; mask <=0x80; mask = mask <<1){
    if(data&mask){
      GPIOB->DOUT31_0 |= 0x80; // data bit high
    }else{
      GPIOB->DOUT31_0 &= ~0x80; // data bit low
    }
    Delay(); // 0.5ms
  }
  GPIOB->DOUT31_0 |= 0x80; // stop bit
  Delay(); // 0.5ms
} // idle is high
  GPIOB->DOUTCLR31_0 = 0x80; // start bit
  Delay(); // 0.5ms
  for(int mask=1; mask <=0x80; mask = mask <<1){
    if(data&mask){
      GPIOB->DOUTSET31_0 = 0x80; // data bit high
    }else{
      GPIOB->DOUTCLR31_0 = 0x80; // data bit low
    }
    Delay(); // 0.5ms
  }
  GPIOB->DOUTSET31_0 = 0x80; // stop bit
  Delay(); // 0.5ms
} // idle is high
```

(10) Problem 4) Equations. Give the functional equations in terms of these parameters:

V_{OL}, output low voltage of MSPM0 in volts
V, voltage across the LED in volts
R, resistance of a smallest resistor of the DAC in ohms
I, current through the LED in amps
n, number of bits in the DAC
b, baud rate of the UART in bits/sec
max, the maximum possible DAC voltage in volts
min, the minimum possible DAC voltage in volts
f_{min}, minimum sound frequency you wish to create with the DAC and interrupts
f_{max}, maximum sound frequency you wish to create with the DAC and interrupts
f, the MSPM0 bus frequency in Hz

(3) Part a) Give the relationship for the **power** dissipated in an LED.

Power =V*I	Units of power =watts

(3) Part b) Largest possible interrupt period; the ISR outputs one value to the DAC, creating sound.

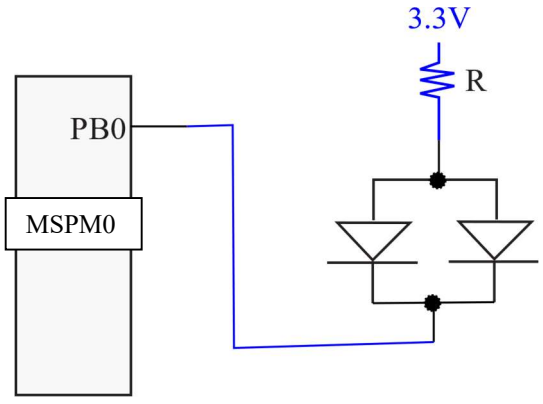
Period = $1/(2*f_{max})$	Units of period = sec

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

Resolution =$(max-min)/2^n$	Units of resolution = volts

(10) Question 5) LED interface.

KCL states LEDs currents will add. KVL states LED voltages will be equal. Effective operating point of the two LEDs will be 1V, 4mA. This is negative logic, so voltage across resistor will be 3.3-1-0.3V = 2V. Current through resistor will be 2V/R=4mA. So R = 500Ω.



(10) Question 6) Assembly array access. Write an assembly language subroutine called **CountIf** that reads all elements of an array. Each element of the array is a signed 8-bit integer. A pointer to the array is passed in R0, and the size of the array passed in R1. Count the number of elements that are less than or equal to 42. The size may be 0. Follow AAPCS rules. Return the count in R0.

```
CountIf:
     MOVS  R2,#0 // Count
loop: CMP  R1,#0 // size
     BEQ  done
     LDRB R3,[R0]  // fetch signed 8 bit
     CMP  R3,#42
     BGT  skip     // signed branch
     ADDS  R2,#1    // found one
skip ADDS  R0,#1    //next pointer
     SUBS  R1,R1,#1 //size
     B    loop
done MOVS  R0,R2 // return parameter
     BX   LR
```

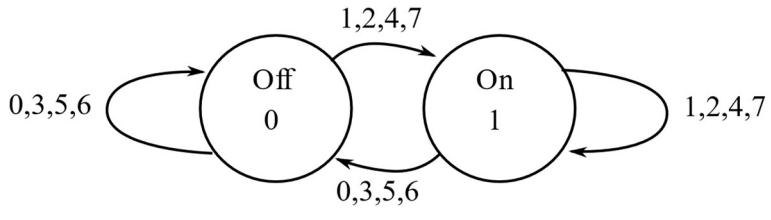
(5) Question 7) Local variables. The subroutine **mySub** has no input or output parameters. The subroutine allocates one 32-bit local variable **j**, and uses R7 stack frame addressing to access the local variable. Fill in the binding for the local variable.

```
.equ j,  [ 4 ]          //binding for 32-bit local variable

mySub: SUB  SP,SP,#4   //allocate j
       PUSH {R7}
       MOV  R7,SP      //frame pointer using R7
//---------start of body--------------------
       MOVS R0,#42
       STR  R0,[R7,#j] //set j = 42
;---------end of body---------------------
       POP  {R7}
       ADD  SP,SP,#4 //deallocate j
       BX   LR
```

(10) Problem 8) FSM STG..

Input	Output
000	0
001	1
010	1
011	0
100	1
101	1
110	0
111	1



(15) Question 9) FIFO. You are asked to finish the FIFO implementation that can store up to 4 values.

```
// Gets a character from the FIFO,
// Input: none
// Output: return the character or
//         return 0 if the FIFO was empty
char Fifo_Get(void){
  if(n == 0) return 0;     // empty
  char data=Buffer[0];     // [0] is oldest
  for(int i=0; i<n; i++){ // shift data down
    Buffer[i] = Buffer[i+1];
  }
  n--;
  return data;
}
```

(15) Problem 10) Sprites. Consider a game with many balls.

```
void SysTick_Handler(void){
  for(int i=0; i<32; i++){
    if(Balls[i].life == alive){
      Balls[i].vy += Balls[i].ay; // vy is integral of ay
      Balls[i].y += Balls[i].vy;  // y is integral of y
      if((Balls[i].y > 159)|| (Balls[i].y < 0)){
        Balls[i].life = dying;
      }
    }
  }
  Flag = 1;
}
```