Final Exam

Date: May 13th 2016

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.***

Problem 1	10	
Problem 2	15	
Problem 3	10	
Problem 4	15	
Problem 5	15	
Problem 6	10	
Problem 7	15	
Problem 8	10	
Total	100	

(10) Question 1. Please place **one word or short phrase** that best answers each question in the box.

(1) Part a) Which registers are NOT pushed on the stack when an interrupt occurs?

R4-R11,SP

(1) Part b) Why do we use a resistor when interfacing an LED? ……

To select the LED current

(1) Part c) If the average rate at which you call **Fifo_Put** to enter data into a FIFO is less than the average rate at which you call **Fifo_Get** to remove data, …… could the FIFO ever become full?

Yes, with bursts of fast input

(1) Part d) Why do we add **`static`** to an otherwise local variable? …

Persistence because variable is in permanent RAM

(1) Part e) Why do we add **`const`** to an otherwise global variable? ….

Put in ROM, can't change it

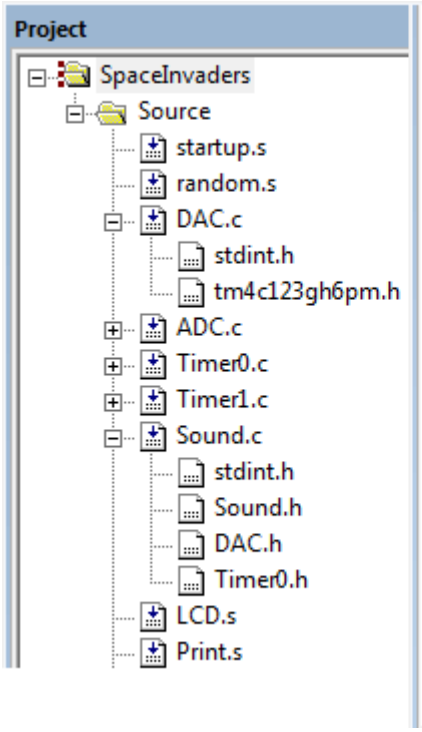
(1) Part f) Why do we add **`static`** to an otherwise global variable? …

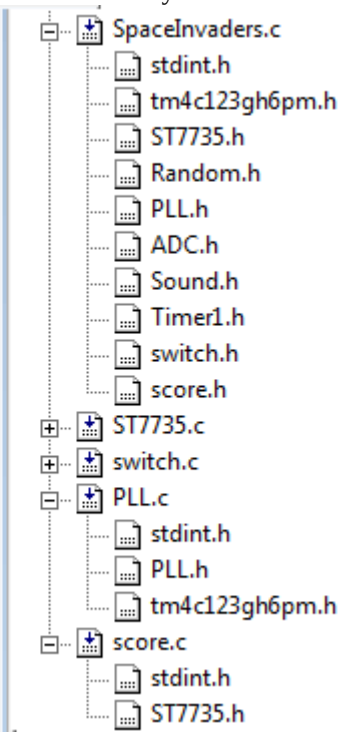
Reduce the scope, private to fil

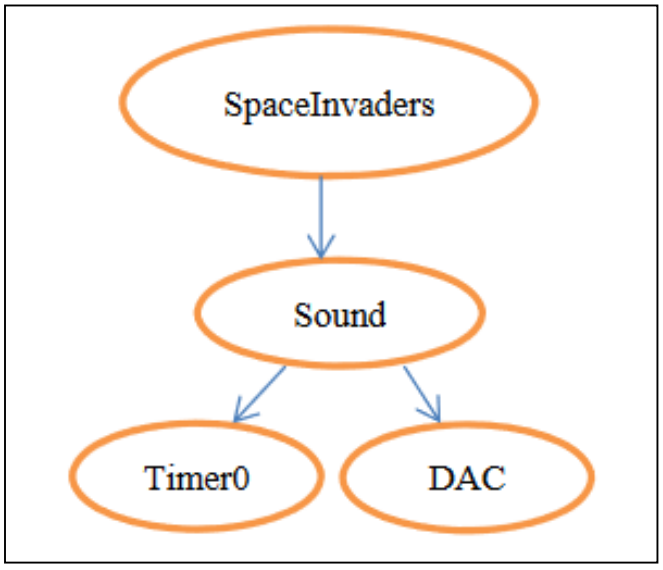
(1) Part g) Why would you ever wish to use the PLL and slow down … the bus clock so the software runs slower?

Save power, make battery run longer

(3) Part h) The following is the project window for my Lab 9 game. Draw a subset of the **call graph** of this system showing all the arrows into and out of the **Sound** module. Put your answer in the box.







(15) Question 2: *SysTick and Fixed-point*

(6) Part a) Write a function in C or assembly that uses SysTick delay for 1 millisecond. Assume that the SysTick is already initialized with **`SysTick->LOAD=0x00FFFFFF; SysTick->CTRL=0x05;`** Assume that the clock is running at 50 MHz, such that each bus cycle is 20 ns. **`SysTick->VAL`** returns the current counter value, 24 bits, counting down.

```
void Delay1ms(){
  SysTick->LOAD = 49999;
  SysTick->VAL = 0;
  while((SysTick->CTRL & 0x00010000) == 0){};
)
```

(6) Part b) Write C code initialization that initializes SysTick to interrupt every 100 us. Assume the bus clock is 80 MHz. You do not need to write the SysTick ISR.

```
void SysTick_Init100us(void){

  SysTick->LOAD = 7999;

  SysTick->CTRL=0x07;
  __enable_irq();
}
```

(3) Part c) Consider the use of decimal fixed-point representation with a resolution of 0.01cm. What is the area of a rectangle in cm^2 whose width and length are represented by fixed-point integer values 250 and 110 respectively?

2.50*1.1 = 2.5+0.25 = 2.75 cm^2

(10) Question 3: *FSM*

(5) Part a) A Moore FSM has 10 states, a 3-bit input (PB5-7), a 5-bit output (PB0-4) and a state dwell (or wait) time that can vary between 300ms to 1800ms. Complete the missing pieces in the definition of the struct for the State and FSM array declaration needed to implement the FSM.

```
#define Init 0
struct State{
  uint8_t out;     // 16 or 1326 bits ok too
  uint16_t wait;   // 32 bit ok too
  uint8_t next[8]; // 16 or 1326 bits ok too
}'
```

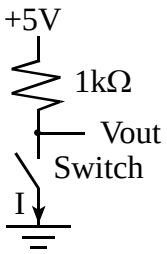
```
}
typedef struct State State_t;
State_t FSM[10]= { … contents are defined for you …}
uint8_t CS; // Index into the FSM array indicating current state
```

(5) Part a) Complete the FSM engine loop inside the main below. Assume you are given a function **`Delay1ms(uint32_t count)`** that delays for count milliseconds. Read from **GPIOB->DIN31_0** to input. Write to **GPIOB->DOUT31_0** to output.

```
int main(void){
  PORTB_Init(); // Port B Initialization is done for you
  CS = Init;
  while(1){
    uint32_t data = GPIOB->DOUT31_0 & ~0x1F;
    data |= FSM[CS].out;
    Delay1ms(FSM[CS].wait);
    uint32_t in = (GPIOB->DIN31_0&0xE0)>>5;
    CS = FSM[CS].next[in];
  }
}
```

(15) Question 4: *Interfacing*

(3) Part a) Consider this negative logic switch circuit used in a +5V digital system. Do not consider the switch to be ideal. Rather, assume the resistance of the switch when the switch is open is 1 MΩ, and the resistance of the switch when the switch is closed is 1 Ω. How much current flows in mA through the switch when the switch is not pressed?



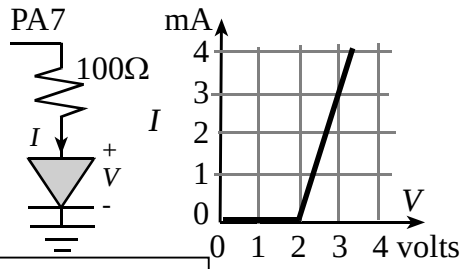
I = V/R = 5V/(1M+1k) = 5V/1M = 5 uA

How much current in mA flows through the switch when the switch is pressed?

I = V/R = 5V/(1+1k) = 5V/1k = 5mA

(6) Part b) The output on PA7 controls this LED. For LED voltages less than 2 volts, the LED current is 0. Assume the output high voltage of PA7 is 3.3V. For voltages above 2 volts, the LED current is

$I = 3 * (V - 2)$, where I is in mA, V is in volts.



What are the **current, voltage,** and **power** to the LED when it is on?

$V_R = 3.3-V$
$I_R = (3.3-V)/0.1 = 33-10*V$ (mA)
$33-10*V = 3 * (V - 2)$
$33+6 = 3 * V + 10*V$
39 = 13V, V=3V, I=3mA, P = 9mW

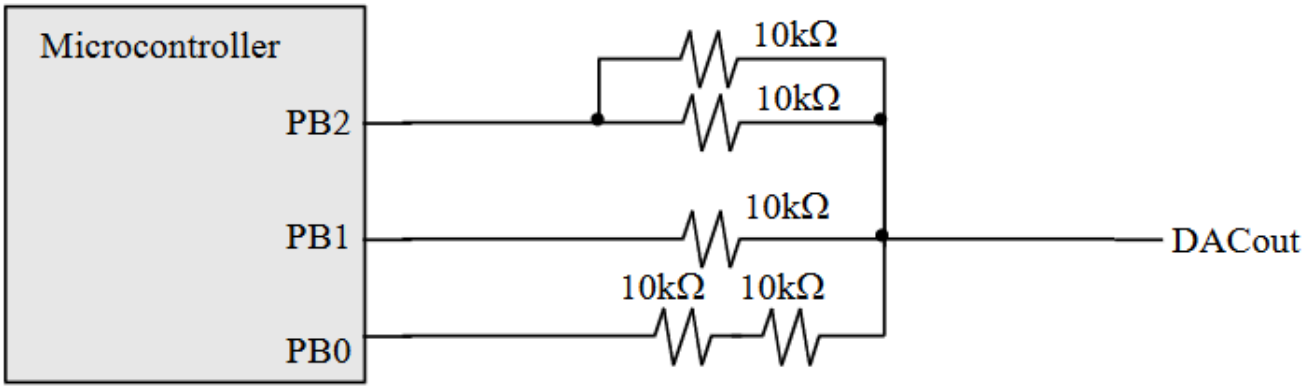
V =

I =

P =

(6) Part c) Design a 3-bit DAC using multiple 10k resistors. No values other than 10k are allowed.

R2 is 5k made with two 10k in parallel, R1 is 10k, R0 is 20k made with two 10k in series



(15) Question 5: *FIFO queue*
You are asked to implement a FIFO that can handle up to **SIZE** elements. You cannot add additional global variables. You cannot change the function prototypes.
(3) Part a) Write the routine that initializes the FIFO.

```
#define SIZE 10
uint8_t FIFO[SIZE],GetI,PutI,Count;
void Fifo_Init(void){ // Initialize FIFO
 Count = GetI = PutI = 0;

}
```

(6) Part b) Write the routine that puts data into the FIFO. If the FIFO is full, this routine should spin (i.e., wait) until there is room in FIFO for the data. You can add local variables.

```
// enter one byte into the FIFO
void Fifo_Put(uint8_t data){
  while(Count == SIZE){};
  Count++;
  FIFO[PutI] = data;
  PutI = (PutI+1)%SIZE;
}
```

(6) Part c) Write the routine that gets data from the FIFO. If the FIFO is empty, this routine should spin (i.e., wait) until there is data in FIFO to return. You can add local variables.

```
// return one byte of data
// if empty spin until there is data in FIFO
uint8_t Fifo_Get(void){ uint8_t data;
  while(Count == 0){};
  data = FIFO[GetI];
  Count--;
  GetI = (GetI+1)%SIZE;
  return data;

}
```

(10) Question 6: *Local variables*

(3) Part a) What does **LCD_OutDec** print to screen?

```
void add1(uint32_t *in){
  *in = (*in)+1;
}
void func(void){
  uint32_t a = 10;
  add1(&a);
  LCD_OutDec(a);
}
```

11

Answer (select one option):

a) 11
b) 10
c) Some address of the stack region
d) Address of **a**
e) Unknown, unable to determine

(7) Part b) Implement both subroutines **func** and **add1** in assembly. For the subroutine **func**, use binding, allocation, access and deallocation of the local variable **a** on the stack. Implement each line of C explicitly in assembly. Use AAPCS. You must clearly identify each of the different stages. Use C statements as comments.

```
add1:
  LDR  R1,[R0]
  ADDS R1,R1,#1
  STR  R1,[R0]
  BX   LR


 .equ a,0
func:
  PUSH {LR}
  SUB  SP,#4
  MOVS R0,#10
  STR  R0,[SP,#a]
  MOV  R0,SP
  BL   add1
  LDR  R0,[SP,#a]
  BL   LCD_OutDec
  ADD  SP,#4
  POP  {PC}
```

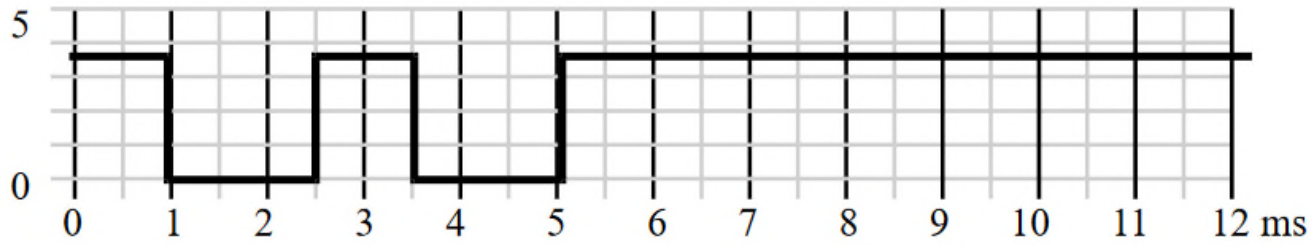
(15) Question 7: *UART and ADC*

(3) Part a) What is the basic idea of **`UART0->IBRD`** and **`UART0->FBRD_R`**? Pick one answer.

i. IBRD sets the baud rate
ii. FBRD sets the baud rate
iii. IBRD+FBRD sets the baud rate
iv. IBRD*FBRD sets the baud rate
v. IBRD+FBRD/64 sets the baud rate

v.

(6) Part b) Assume a serial port operating with a baud rate of 2000 bits per second. Draw the UART waveform when the decimal value 140 is transmitted. You may assume the channel is idle before and after the frame. Time flows from left to right. The dark vertical black lines correspond to 1-ms boundaries. Assume the frame begins at time = 1 ms, and show the waveform from 0 to 12 ms.
140 = 128+8+4 = 0x8C



(3) Part c) You have a 12-bit, 0 to 3.3V range ADC. If the ADC input is 3.3/4V = 0/825V, what will be the digital value **in decimal** returned by this ADC?

4095*1.25/3 = 1706 = 0x6AA

0x6AA

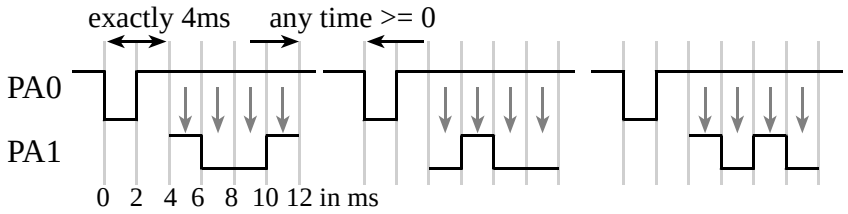
(3) Part d) You are attempting to capture a sinusoidal sound with a frequency of 8 kHz. Using the 12-bit ADC and periodic interrupt, you have programmed the SysTick to interrupt at a frequency of 12 kHz. During the SysTick ISR you collect one ADC sample. Is it possible to recreate the original signal 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?

No, The sampling rate in this case that satisfies the Nyquist theorem is 16kHz (2*8kHz). The chosen sampling frequency of 12 kHz is insufficient and will cause **aliasing**.

(10) Question 8: *Design Problem*

You are to implement one-directional communication between two microcontrollers (sender and receiver) using two pins PA0 and PA1. The sender microcontroller programs these pins as output and the receiver programs them as input. PA0 is used as the control and PA1 is used to transfer the actual data. The bit-level protocol is as follows: when it has data to transmit, the sender sends a pulse on PA0, which is a low for 2 ms, followed by a high for 2 ms. The first bit of transmission (on PA1) immediately follows, with each bit sent lasting for exactly 2 ms. Each 4-bit transmission is preceded by a pulse on PA0. The time between transmissions can be any value greater than or equal to zero. See in the timeline below that the first byte transferred is 0x29. Note that each byte of data is transmitted as two 4-bit nibbles with the bit order as 0,1,2,3 then 4,5,6,7. The vertical arrows mark when you should read the input data. You may assume PA1 and PA0 are initialized to inputs.

(8) Part a) The overall goal of the communication is to transfer data from the sender to the receiver. In this section you will write a function that receives one byte of data. You may assume that you are given a function **`Delay1ms()`.**



You may assume the software execution time is negligible compared to the Delay function

```
uint8_t ReceiveChar(void){

uint8_t data;
while((GPIOA->DIN31_0&0x01)==1){}
while((GPIOA->DIN31_0&0x01)==0){}
Delay1ms(); Delay1ms(); Delay1ms();
data = (GPIOA->DIN31_0&0x02)>>1; Delay1ms(); Delay1ms();
data = data|(GPIOA->DIN31_0&0x02); Delay1ms(); Delay1ms();
data = data|(( GPIOA->DIN31_0&0x02)<<1); Delay1ms(); Delay1ms();
data = data|(( GPIOA->DIN31_0&0x02)<<2);
while((GPIOA->DIN31_0&0x01)==1){}
while((GPIOA->DIN31_0&0x01)==0){}
Delay1ms(); Delay1ms(); Delay1ms();
data = data|(( GPIOA->DIN31_0&0x02)<<3); Delay1ms(); Delay1ms();
data = data|(( GPIOA->DIN31_0&0x02)<<4); Delay1ms(); Delay1ms();
data = data|(( GPIOA->DIN31_0&0x02)<<5); Delay1ms(); Delay1ms();
data = data|(( GPIOA->DIN31_0&0x02)<<6);
return data;
```

(2) Part b) What is maximum achievable **bandwidth** in bits/sec for this communication scenario?

4bits/12ms = 4000bits/12s = 1000/3 =333bits/sec, 4/6*(500bps) = 333 bps