Department of Electrical and Computing Engineering

UNIVERSITY OF CONNECTICUT

ECE 3411 Microprocessor Application Lab: Fall 2015

Quiz II

There are 5 questions in this quiz. There are 9 pages in this quiz booklet. Answer each question according to the instructions given.

You have **45 minutes** to answer the questions.

Some questions are harder than others and some questions earn more points than others—you may want to skim all questions before starting.

If you find a question ambiguous, be sure to write down any assumptions you make.
Be neat and legible. If we can't understand your answer, we can't give you credit!

Write your name in the space below. Write your initials at the bottom of each page.

THIS IS A CLOSED BOOK, CLOSED NOTES QUIZ.
PLEASE TURN YOUR NETWORK DEVICES OFF.

Any form of communication with other students is considered cheating and will merit an F as final grade in the course.

Do not write in the boxes below

1 (x/12)	**2 (x/16)**	**3 (x/24)**	**4 (x/24)**	**5 (x/24)**	**Total (xx/100)**

Name:

Student ID:

1. [12 points]: Answer the following questions:

a. The compiler will generate an error while compiling the following line of C code. Write the correct version of this line in the space below.

```
const uint8_t my_string PROGMEM = "Hello!";
```

b. Once an interrupt occurs, how does an AVR knows where to find the code for the corresponding Interrupt Service Routine (ISR)?

c. Is the following statement True or False?
"Upon an interrupt, the instruction which is currently being executed in the main code is finished first before executing the Interrupt Service Routine (ISR)."

d. Consider the following push-switch circuit. When this switch is pushed, the logic value passed to AVR (i.e. voltage at node `'To AVR'`) is:

(a) Logic HIGH
(b) Logic LOW
(c) None of the above

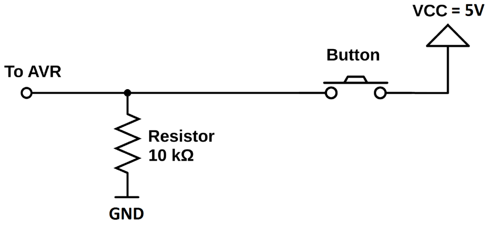
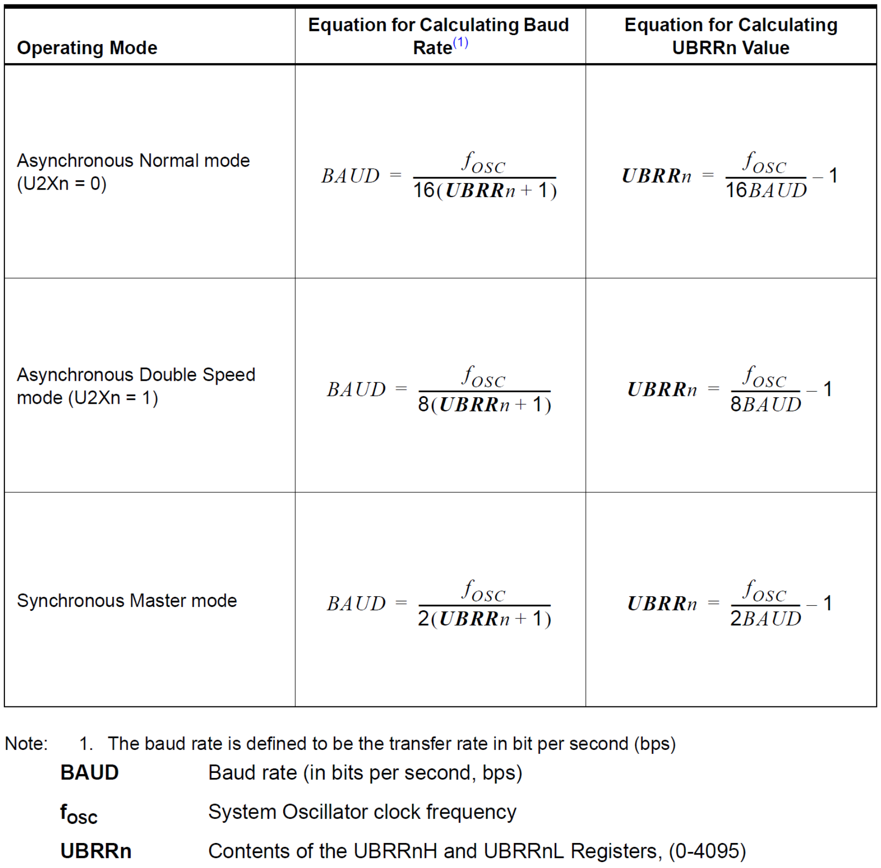


Figure 1: A push switch circuit.

Initials:

2. [16 points]: Using Table 1, calculate the required value of UART Baud Rate Register UBRR0 for a baud rate of 1000 in Asynchronous Normal mode, where the System Oscillator clock frequency of 16MHz. Also, write C code inside `Initialize_UBRR0(uint16_t Value)` function to store the value of argument `Value` into UBRR0 register.

Table 1: Equations for calculating UART Baud Rate Register setting



Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRRn Value
Asynchronous Normal mode (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{16(\boldsymbol{UBRRn} + 1)}$	$\boldsymbol{UBRRn} = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2Xn = 1)	$BAUD = \frac{f_{OSC}}{8(\boldsymbol{UBRRn} + 1)}$	$\boldsymbol{UBRRn} = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master mode	$BAUD = \frac{f_{OSC}}{2(\boldsymbol{UBRRn} + 1)}$	$\boldsymbol{UBRRn} = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps)

BAUD	Baud rate (in bits per second, bps)
f_{OSC}	System Oscillator clock frequency
UBRRn	Contents of the UBRRnH and UBRRnL Registers, (0-4095)

Calculated UBRR0 value =

```
/* Write the code for initializing ‘UBRR0’ here */
void Initialize_UBRR0(uint16_t Value)
{





}
```

Initials:

3. [24 points]: Use LCD Instruction Set table (Table 3) provided on page 5 to fill LCD Commands Table (Table 2) below with the correct bit values of **RS**, **R/W** and **DB7-DB0** signals to configure/control the LCD according the specified desired functionality.

Table 2: LCD Commands Table

No.	Desired Functionality	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	Set interface data length to 8-bit mode, number of display lines to 1, and character font to 5×10 dots.										
2	Turn the display OFF, cursor OFF, and no blinking.										
3	Set the direction of cursor movement towards right and turn the display shift mode ON.										
4	Turn the display ON, cursor ON, and no blinking.										
5	Move the cursor to position $(0, 5)$, i.e. first row and sixth column. **Hint:** The first row starts from DD RAM address 0x00.										
6	Write the character 'A' to the LCD. The ASCII value of 'A' is 0x41.										

Initials:

Table 3: LCD Instruction Set

Instruction	Code										Function	Execution time (max)
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		(f_{OSC} = 250KHz)
Display Clear	0	0	0	0	0	0	0	0	0	1	Clear entire display area, restore display from shift, and load address counter with DD RAM address 00H.	1.64ms
Display/ Cursor Home	0	0	0	0	0	0	0	0	1	*	Restore display from shift and load address counter with DD RAM address 00H.	1.64ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Specify direction of cursor movement and display shift mode. This operation takes place after each data transfer (read/write).	40µs
Display ON/OFF	0	0	0	0	0	0	1	D	C	B	Specify activation of display (D) cursor (C) and blinking of character at cursor position (B).	40µs
Display/ Cursor Shift	0	0	0	0	0	1	S/C	R/L	*	*	Shift display or move cursor.	40µs
Function Set	0	0	0	0	1	DL	N	F	*	*	Set interface data length (DL), number of display line (N), and character font (F).	40µs
RAM Address Set	0	0	0	1	ACG						Load the address counter with a CG RAM address. Subsequent data access is for CG RAM data.	40µs
DD RAM Address Set	0	0	1	ADD							Load the address counter with a DD RAM address. Subsequent data access is for DD RAM data.	40µs
Busy Flag/ Address Counter Read	0	1	BF	AC							Read Busy Flag (BF) and contents of Address Counter (AC).	0µs
CG RAM/ DD RAM Data Write	1	0	Write data								Write data to CG RAM or DD RAM.	40µs
CG RAM/ DD RAM Data Read	1	1	Read data								Read data from CG RAM or DD RAM.	40µs

I/D = 1 : Increment	I/D = 0 : Decrement
S = 1 : Display Shift On	
D = 1 : Display On	
C = 1 : Cursor Display On	
B = 1 : Cursor Blink On	
S/C = 1 : Shift Display	S/C = 0 : Move Cursor
R/L = 1 : Shift Right	R/L = 0 : Shift Left
DL = 1 : 8-Bit	DL = 0 : 4-Bit
N = 1 : Dual Line	N = 0 : Signal Line
F = 1 : 5x10 dots	F = 0 : 5x8 dots
BF = 1 : Internal Operation	
BF = 0 : Ready for Instruction	

DD RAM :	Display Data RAM
CG RAM :	Character Generator RAM
ACG :	Character Generator RAM Address
ADD :	Display Data RAM Address
AC :	Address Counter

Note 1: Symbol "*" signifies an insignificant bit (disregard).

Note 2: Correct input value for "N" is predetermined for each model.

Initials:

4. [24 points]: Let `Task1()` and `Task2()` be two functions from standard C library (`stdlib.h`). Write a C program for your AVR such that it calls `Task1()` every $10ms$ and `Task2()` every $100ms$. You are allowed to use `_delay_ms()` function. Assume that the execution of `Task1()` and `Task2()` virtually takes no time.

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <stdio.h>
#include <stdlib.h>
#include <util/delay.h>

/* Declare any variables here */





int main(void)
{
    /* Write your code below */
```

```
} /* End of main() */
```

Initials:

5. [24 points]: Let `Task1()` and `Task2()` be two functions from standard C library. We want to call `Task1()` once and only once every time a push button is pushed from released state, and we want to call `Task2()` once and only once every time the button is released from pushed state. The function `_button_pushed()` returns TRUE as long as the push button is pressed, and False otherwise.
Implement the above mentioned functionality by extending `Task_PollingButton_Debounce(void)` function given below.

```
/* Debouncing State Machine */
void Task_PollingButton_Debounce(void)
{
    switch (PushState)
    {
        case NoPush:
        if ( _button_pushed() ) PushState=Maybe;
        else PushState=NoPush;
        break;

        case Maybe:
        if ( _button_pushed() ){ PushState=Pushed; PushFlag_Debounce=1; }
        else { PushState=NoPush; PushFlag_Debounce=0; }
        break;

        case Pushed:
        if ( _button_pushed() ) PushState=Pushed;
        else PushState=Maybe;
        break;
    }
}


/* Write your code below */
```

Initials:

```
/* Your code continues here */
```

Initials:

End of Quiz

Please double check that you wrote your name on the front of the quiz.

Initials: