Department of Electrical and Computing Engineering

UNIVERSITY OF CONNECTICUT

ECE 3411 Microprocessor Application Lab: Fall 2015

Quiz III

There are 5 questions in this quiz. There are 11 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/14)	2 (x/14)	3 (x/24)	4 (x/24)	5 (x/24)	Total (xx/100)

Name:

Student ID:

1. [14 points]: Answer the following questions:
(Encircle the correct answer for Multiple Choice Questions)

A. Which timer register chooses the type of timer-based interrupt vector?

(a) TCCRnA
(b) TCCRnB
(c) TIMSK
(d) OCRnA/OCRnB

B. Timer n, operating in CTC mode clears the value stored in the output compare register OCRnX, when TCNTn reaches the value stored in OCRnX.

(a) True
(b) False

C. What register stores the values of external interrupt flags and will trigger an external interrupt if the I-bit in SREG is set?

(a) EICRA
(b) EIMSK
(c) EIFR
(d) PCIFR

D. Given below is an ISR for external interrupt INT1 that toggles a LED whenever a switch connected to INT1 pin is pushed.

```
/* External Interrupt INT1 ISR. Interrupt triggered at Falling Edge */
ISR(INT1_vect)
{
    EIMSK &= ˜(1<<INT1);     // Disable External Interrupt INT1
    PORTB ˆ= (1<<PORTB5);    // Toggle a LED

    /* Enable External Interrupt INT1 again later in main() code */
}
```

What is the purpose of disabling INT1 in the ISR? What could go wrong if INT1 is not disabled immediately?

Initials:

E. The figure below shows Input Capture Unit block diagram for Timer 1.

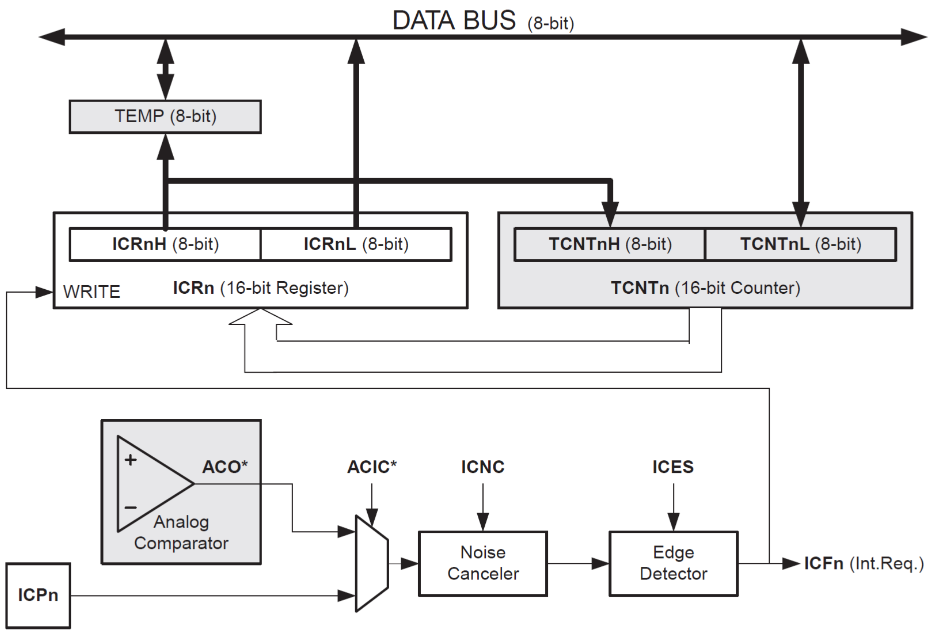

Figure 1: Input Capture Unit Block Diagram.

List the two sources (shown in the block diagram) that can be configured to generate an “Input Capture Interrupt”.

Initials:

2. [14 points]: The ISR given below triggers periodically every $1ms$ and implements a simple Finite State Machine (FSM).

```
// Timer 0 Compare Match ISR
ISR (TIMER0_COMPA_vect)
{
    /* FSM Implementation */
    switch (Current_State)
    {
        case State_A:
        if(Flag == 0)    Current_State = State_B;
        else             Current_State = State_D;
        break;

        case State_B:
        if(Flag != 0)    Current_State = State_A;
        break;

        case State_C:
        Current_State = State_A;
        break;

        case State_D:
        if(Flag != 0)    Current_State = State_C;
        break;
    }
}
```

(a) Draw the state transition diagram of this FSM.

(b) Fill in the state transition table given below for this FSM.

Table 1: FSM State Transition Table

Time (ms)	**Flag**	**Current_State**
0	0	State_B
1	0	
2	1	
3	1	
4	0	
5	1	
6	0	
7	0	

Initials:

3. [24 points]: The code given below uses Timer 1 'Compare Match A' ISR to blink a LED connected to PB5. If the clock frequency ($clk_{I/O}$) is 16MHz, complete the "`initialize_all()`" function below such that the LED toggles after every 250 milliseconds.
You may use Timer 1 data sheet provided at the end of this booklet.

```
/* Initialization function */
void initialize_all(void)
{
    // Set the LED pin as Output here


    // Configure Timer 1 here.










     // Enable Global Interrupts here.


} /* End of initialize_all() */
//-----------------------------------------------------------------------

/* Timer 1 Compare Match ISR */
ISR (TIMER1_COMPA_vect)
{
    PORTB ^= (1<<PORTB5);    // Toggle the LED
}
//-----------------------------------------------------------------------

/* Main Function */
int main(void)
{
    // Initialize everything
    initialize_all();

    while(1);    /* Nothing to do */

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

Initials:

4. [24 points]: You want to toggle a LED connected to PB5 after every 250 milliseconds. One way to do it is by using Timer 1 'Overflow' ISR and a software counter. If the clock frequency ($clk_{I/O}$) is 16MHz, complete the "`initialize_all()`" function and "`ISR(TIMER1_OVF_vect)`" below such that the error in LED toggling period is **less than 1 millisecond**.
You may use Timer 1 data sheet provided at the end of this booklet.
Hint: Running the Timer on higher frequencies provides more accurate results.
Hint: Overflow occurs when the counter reaches its maximum 16-bit value (MAX = 0xFFFF).

```
/* Global variable declarations */
volatile uint8_t software_counter;
volatile uint8_t counter_reset_value;

/* Initialization function */
void initialize_all(void)
{
    // Set the LED pin as Output here



    // Configure Timer 1 here.




    // Initialize 'counter_reset_value' with appropriate value here.



    // Initializing 'software_counter'
    software_counter = counter_reset_value;


     // Enable Global Interrupts here.


} /* End of initialize_all() */
//----------------------------------------------------------------------
```

Initials:

```
/* Timer 1 Overflow ISR */
ISR(TIMER1_OVF_vect)
{
    /* Your code for ISR goes here */















}
//------------------------------------------------------------------------

/* Main Function */
int main(void)
{
    // Initialize everything
    initialize_all();

    while(1)
    {
        if( software_counter == 0 )
        {
            PORTB ^= (1<<PORTB5);    // Toggle the LED
            software_counter = counter_reset_value;
        }
    }

} /* End of main() */

//------------------------------------------------------------------------
```

Initials:

5. [24 points]: You need to design a system such that whenever a certain internal condition (checked by the function '`is_condition_true()`') is true, a small function executes **atomically** and with **the highest priority** over any other code in your software. One way to do it is by using External Interrupt INT0 ISR (INT0 is at pin `PD2`).
Complete the "`initialize_all()`" and "`main()`" functions such that INT0 ISR gets triggered everytime the function "`is_condition_true()`" returns true. State clearly if yo need to make any hardware connections between any two pins etc.
You may use External Interrupts data sheet provided at the end of this booklet.

State hardware connections (if any):

```
//-------------------------------------------------------------------------
/* Initialization function */
void initialize_all(void)
{
    // Configure INT0 and perform any other initializations here.









    // Enable Global Interrupts here.


} /* End of initialize_all() */
//-------------------------------------------------------------------------
/* External Interrupt INT0 ISR */
ISR(INT0_vect)
{
    /* Function that needs to be executed atomically */
    Some_Atomic_Code();

    /* Any other code that you want to include in ISR goes here. */




}
//-------------------------------------------------------------------------
```

Initials:

```
/* Main Function */
int main(void)
{
    // Initialize everything
    initialize_all();

    while(1)
    {
        if( is_condition_true() )
        {
            /* Your code to trigger INT0 ISR goes here */









        }
    }

} /* End of main() */

//----------------------------------------------------------------------
```

Initials:

End of Quiz

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

Initials: