Student no: _2020-01144-MN-0 **Date Started: Mar. 2, 2023**

Name: _Jewel Anne Reyes____ **Date Submitted: Mar. 2, 2023**

Machine Problem # 3

Interrupts

Objective:

- To know the Benefit of Interrupt Request in MCU
- To determine the components of Interrupt Request.
- To enhance skills of Assembly Programming using MPLAB and Proteus.
- To create a code that will provide Interrupt routine.
- To be familiar with Input/Output Interfacing technique of PIC16F84A
- To configure the PORTS of MCU for I/O Interfacing.

Background:

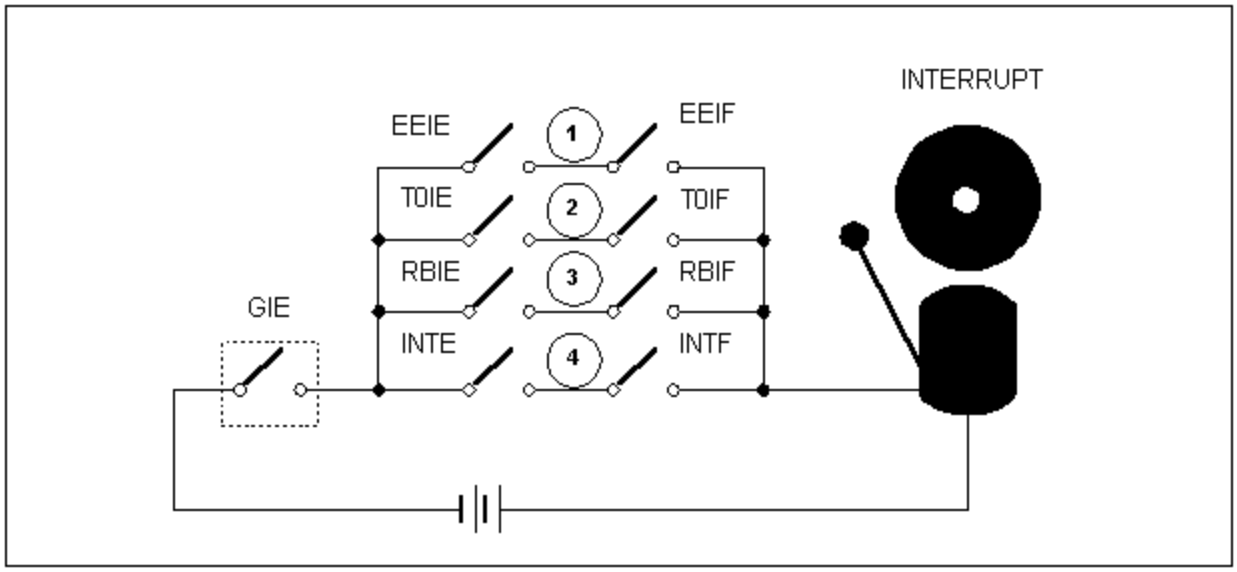
Interrupts are a mechanism of a microcontroller which enables it to respond to some events at the moment they occur, regardless of what microcontroller is doing at the time. This is a very important part, because it provides connection between a microcontroller and environment which surrounds it. Generally, each interrupt changes the program flow, interrupts it and after executing an interrupt subprogram (interrupt routine) it continues from that same point on.

Control register of an interrupt is called INTCON and can be accessed regardless of the bank selected. Its role is to allow or disallowed interrupts, and in case they are not allowed, it registers single interrupt requests through its own bits. Bit 7 of INTCON is called GIE. This is the Global Interrupt Enable. Setting this to 1 tells the PIC that we are going to use an interrupt. Bit 4 of INTCON is called INTE, which means INTerrupt Enable. Setting this bit tells the PIC that RB0 will be an interrupt pin. Setting bit 3, called RBIE, tells the PIC that we will be using Port B bits 4 to 7. Now the PIC knows when this pin goes high or low, it will need to stop what its doing and get on with an interrupt routine.

INTCON Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
bit7							

Legend:
R = Readable bit **W** = Writable bit
U = Unimplemented bit, read as '0' - n = Value at power-on reset



Simplified outline of PIC16F84 microcontroller interrupt

PIC16F84 has four interrupt sources:

1. Termination of writing data to EEPROM.
2. TMR0 interrupt caused by timer overflow.
3. Interrupt during alteration on RB4, RB5, RB6 and RB7 pins of port B.
4. External interrupt from RB0/INT pin of microcontroller.

Materials:

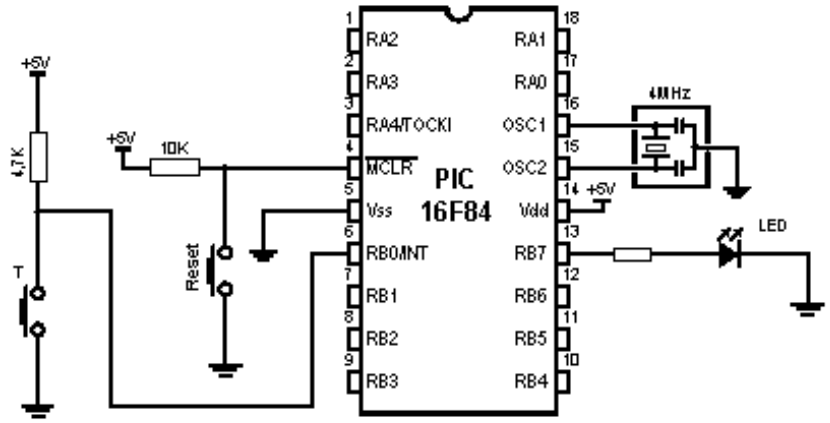
PIC16F84A	PC
PIC Programmer	Power Supply
Test Jig	

Procedure:

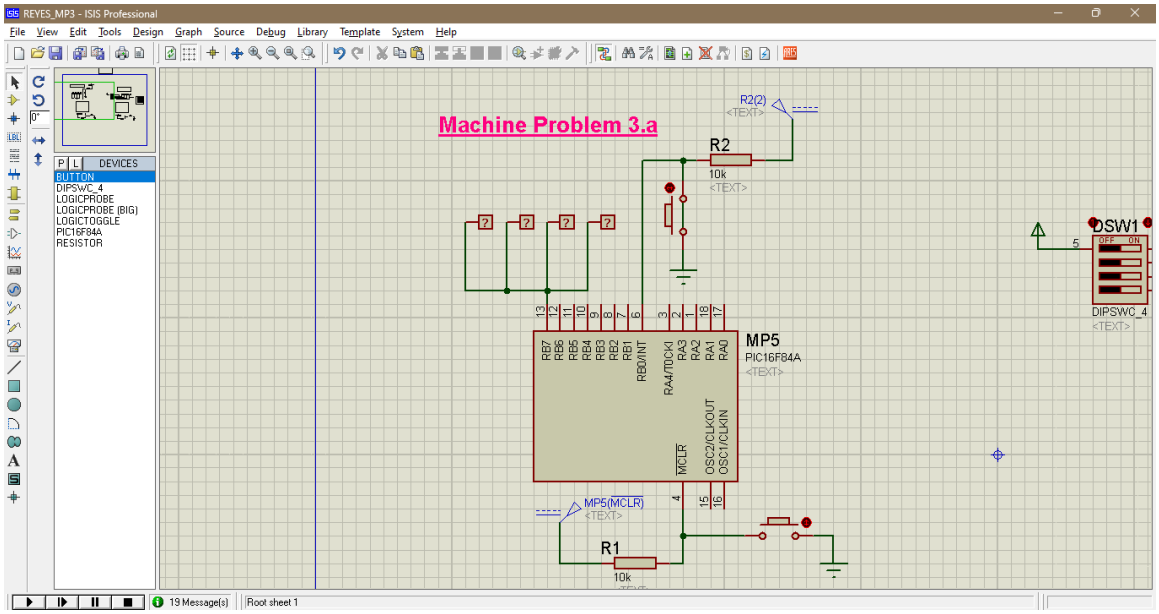
Perform the following Setup and provide a code for each setup.

SETUP A: Processing interrupt caused by change on pin RB0/INT

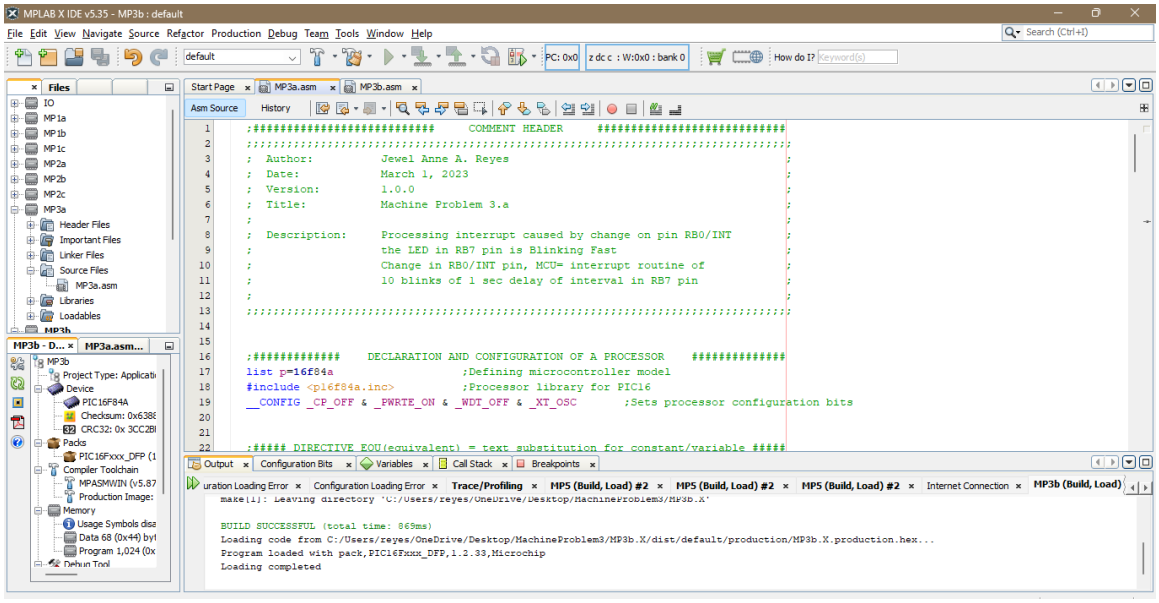
A. Configure the Jig according to the setup showed by the figure below. The normal operation is the LED in RB7 pin is Blinking Fast. The moment there is a change in RB0/INT pin, the MCU will cause an interrupt routine of 10 blinks of 1 sec delay of interval in RB7 pin, and will return to the normal operation.



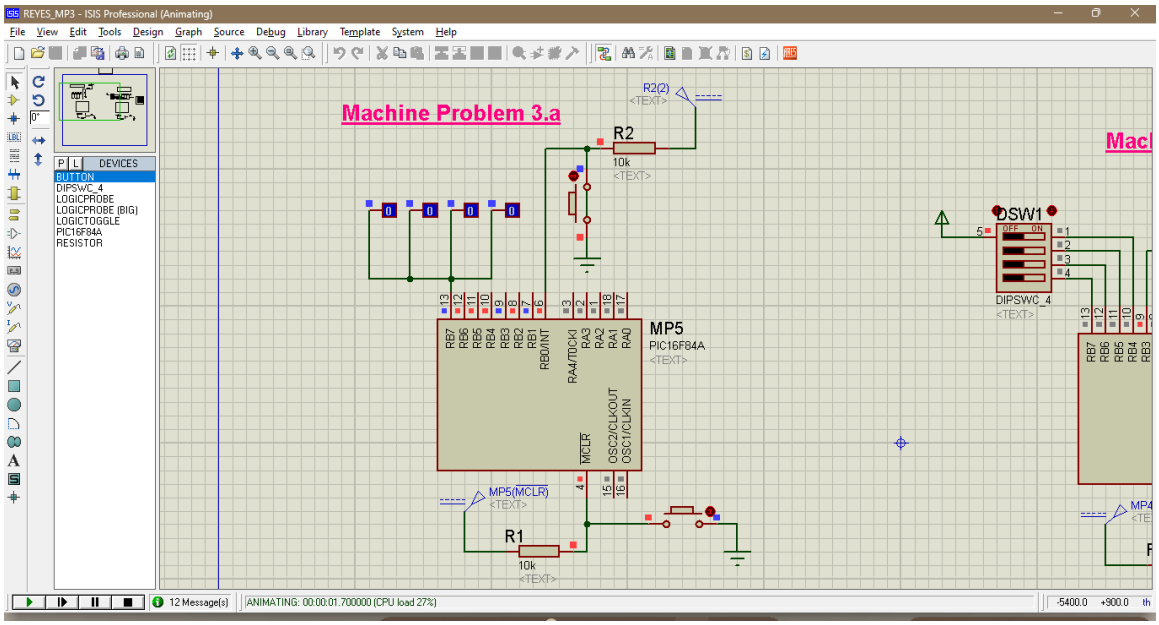
Schematic:

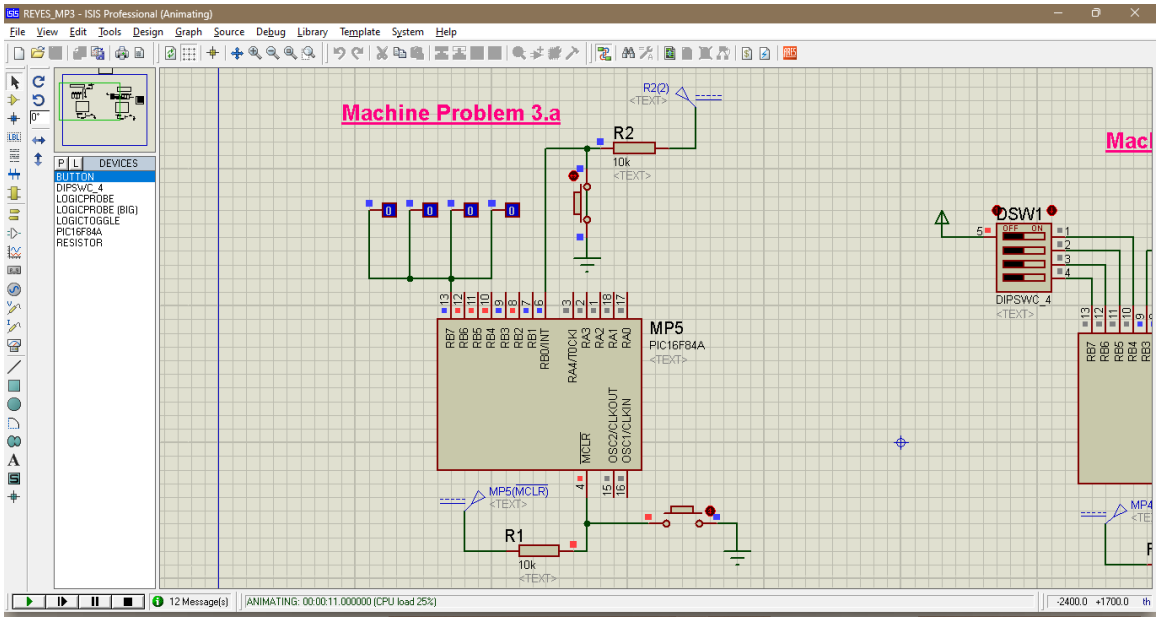


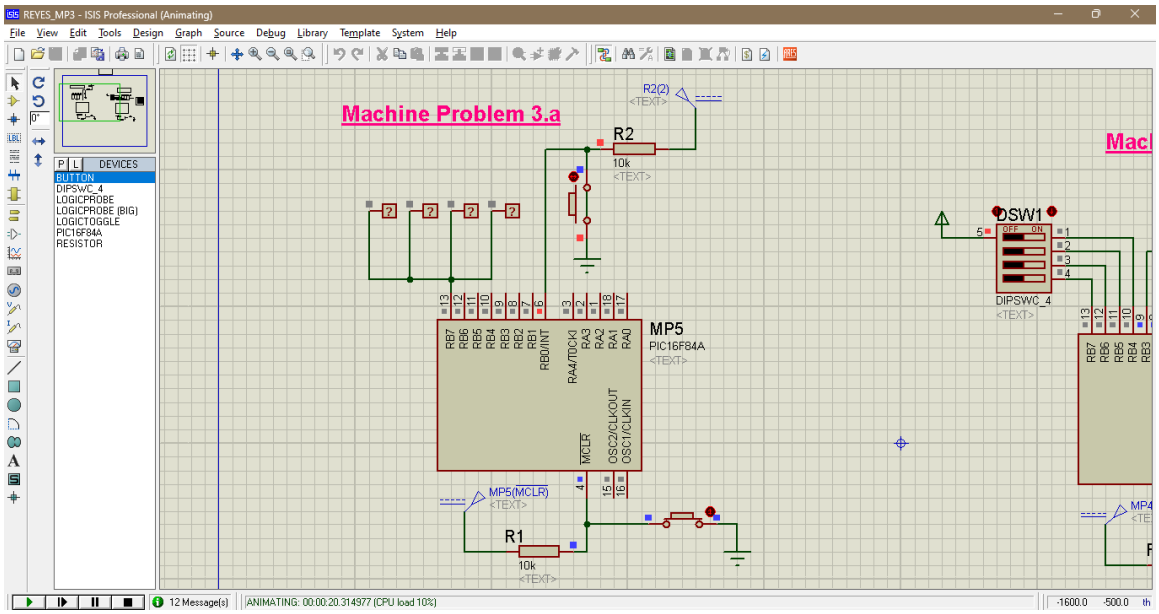
Program Build Successful:



Outputs:

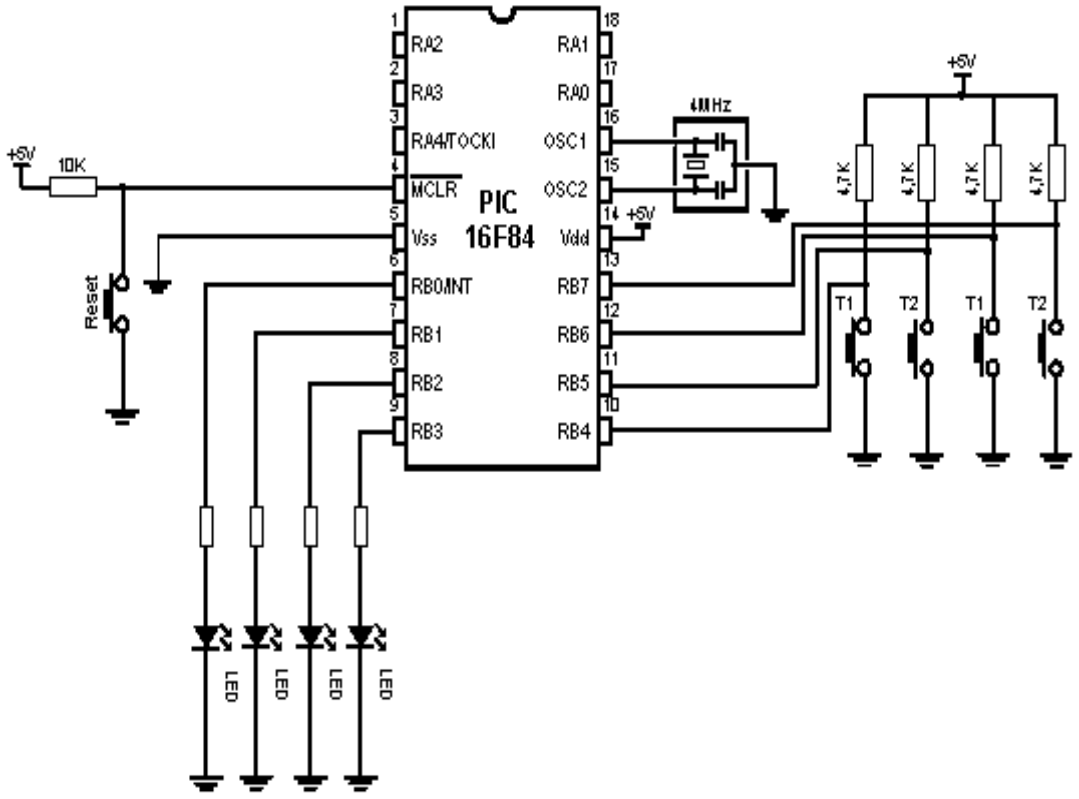

REYES_MP3 - ISIS Professional (Animating)
Machine Problem 3.a
R2
10k
DSW1
DIPSWC_4
MP5
PIC16F84A
MP5(MCLR)
R1
10k
12 Message(s)
ANIMATING: 00:00:01.700000 (CPU load 27%)


REYES_MP3 - ISIS Professional (Animating)
Machine Problem 3.a
R2
10k
DSW1
DIPSWC_4
MP5
PIC16F84A
MP5(MCLR)
R1
10k
12 Message(s)
ANIMATING: 00:00:11.000000 (CPU load 25%)

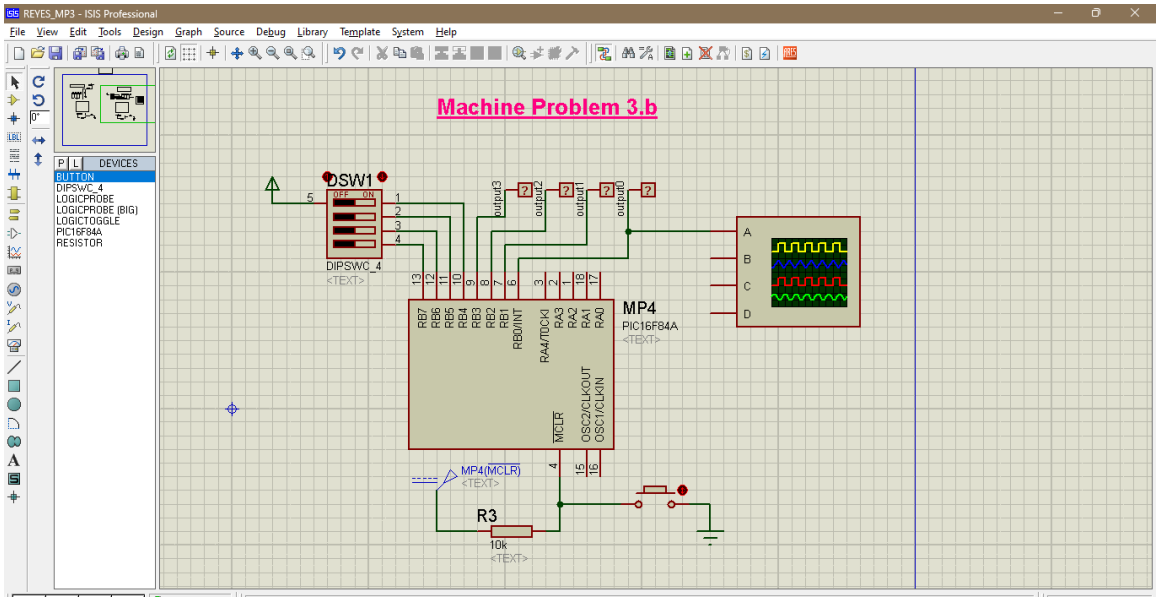

REYES_MP3 - ISIS Professional (Animating)
File View Edit Tools Design Graph Source Debug Library Template System Help
Machine Problem 3.a
R2
10k
R1
10k
MP5
PIC16F84A
MP5(MCLR)
DSW1
DIPSWC_4
DEVICES
BUTTON
DIPSWC_4
LOGICPROBE
LOGICPROBE (BIG)
LOGICTOGGLE
PIC16F84A
RESISTOR
12 Message(s)
ANIMATING: 00:00:20.314977 (CPU load 10%)

SETUP B: Processing interrupt caused by changes on pins RB4-RB7

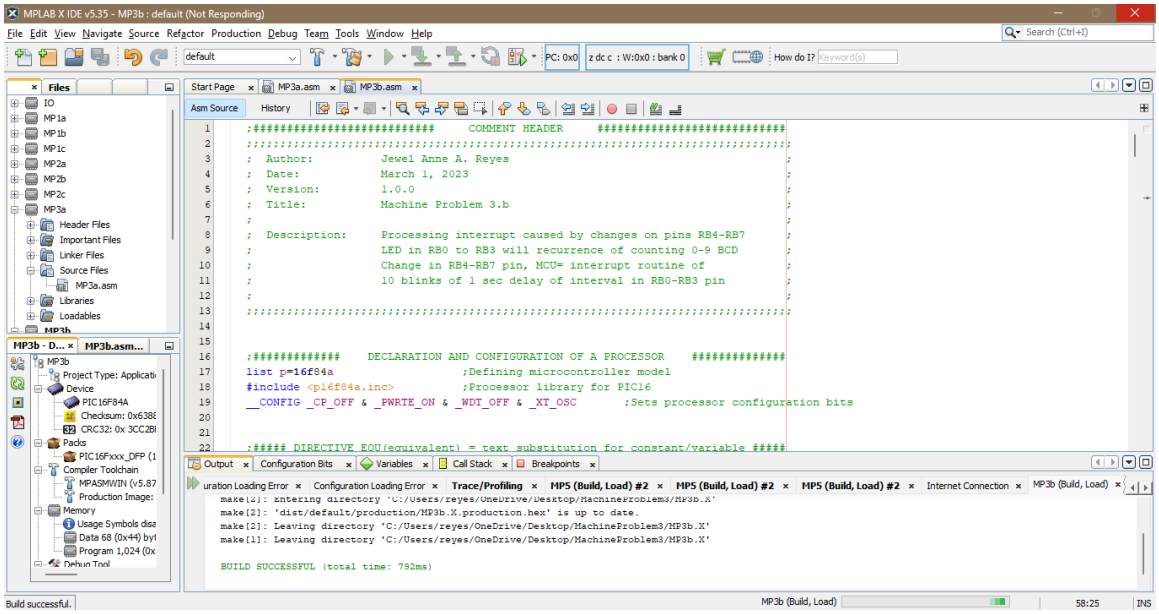
B. Configure the Jig according to the setup showed by the figure below. The normal operation is the LED in RB0 to RB3 will recurrence of counting 0 – 9 BCD. Any changes in pins RB4 – RB7, the MCU will cause an interrupt routine of 10 blinks of 1 sec delay of interval in pins RB0 – RB3, and will return to the normal operation.



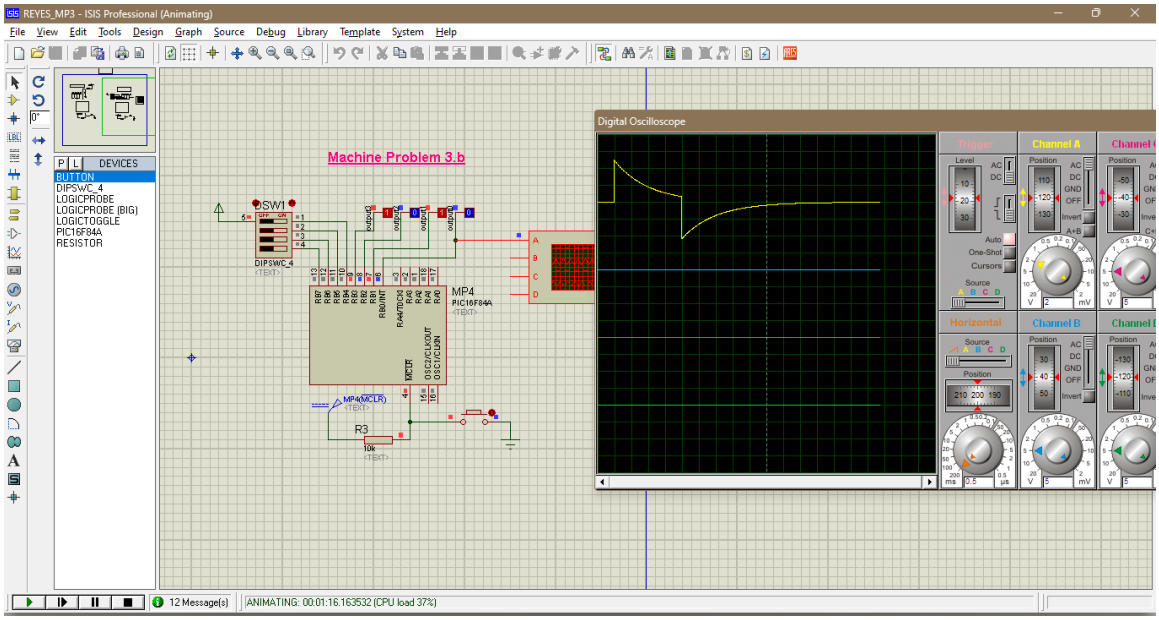
Schematic:

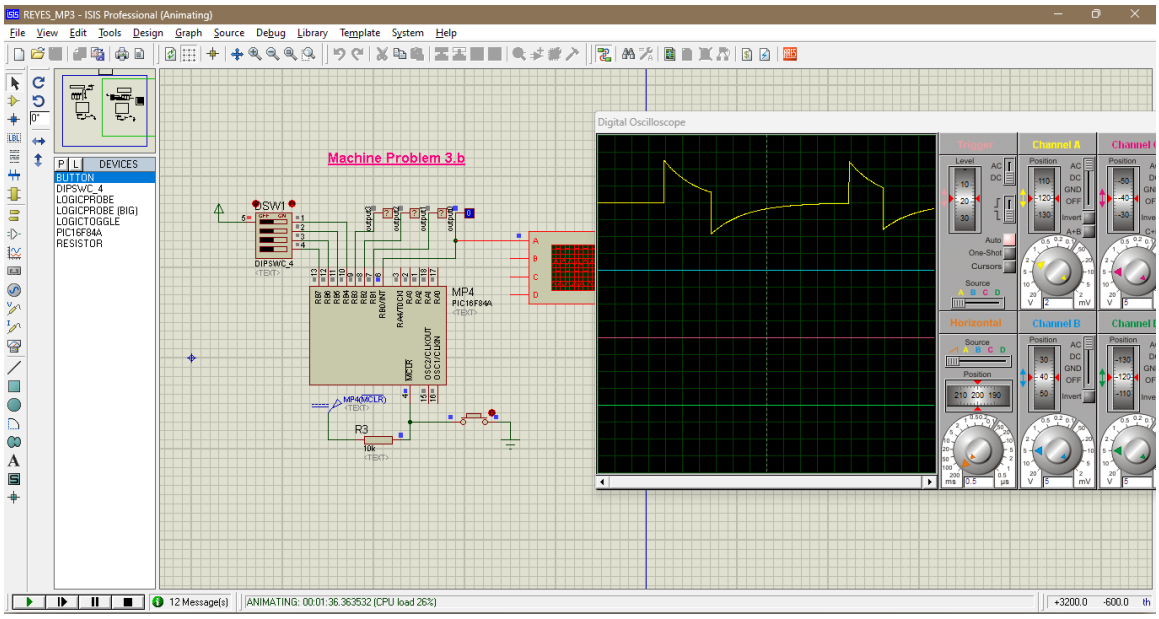


Program Build Successful:



Outputs:


REYES_MP3 - ISIS Professional (Animating)
File View Edit Tools Design Graph Source Debug Library Template System Help
Machine Problem 3.b
Digital Oscilloscope
DEVICES
BUTTON
DIPSWC_4
LOGICPROBE
LOGICPROBE (BIG)
LOGICTOGGLE
PIC16F84A
RESISTOR
DSW1
MP4
PIC16F84A
R3
10k
12 Message(s)
ANIMATING: 00:01:16.163532 (CPU load 37%)


REYES_MP3 - ISIS Professional (Animating)
File View Edit Tools Design Graph Source Debug Library Template System Help
Machine Problem 3.b
Digital Oscilloscope
DEVICES
BUTTON
DIPSWC_4
LOGICPROBE
LOGICPROBE (BIG)
LOGICTOGGLE
PIC16F84A
RESISTOR
DSW1
MP4
PIC16F84A
R3
10k
12 Message(s)
ANIMATING: 00:01:36.363532 (CPU load 26%)

Observation:

As I have observed, it is now obvious that the timer raises the interrupt when the count goes from FFh to 00h. To show that a overflow has occured, the timer sets the second bit of the INTCON register. But this bit has to be cleared in the coming ISR(Interrupt service routine) before the interrupt can be re-enabled. Interrupts are really advantageous and luckily, PIC16F84A has it. There are so many purpose of it. Here is an example of how interrupts can be used in a real-world application:

Suppose we are building a system that monitors the temperature of a room and turns on a fan when the temperature exceeds a certain threshold. We can use an external temperature sensor to detect the temperature, and an interrupt to trigger the fan when needed.

First, we need to set up the interrupt source. We can use the RB0/INT pin as the interrupt source, and configure it to trigger on a rising edge. This is done by setting the appropriate bits in the INTCON and OPTION_REG registers.

Next, we write an ISR that reads the temperature from the sensor and turns on the fan if the temperature is above the threshold. The ISR should be as short and simple as possible, as interrupts can occur at any time and can disrupt the normal flow of the program.

Finally, we enable interrupts by setting the GIE bit in the INTCON register, and start the main program loop. The main program loop should continuously monitor the temperature and update the fan status as needed.

In summary, interrupts are a useful feature of the PIC16F84A microcontroller that allow it to respond to external events quickly and efficiently. By following the steps outlined above, interrupts can be implemented in a variety of real-world applications, from temperature monitoring to motor control to user input detection.

Conclusion/ Recommendation:

The PIC16F84A is a popular 8-bit microcontroller that is commonly used in embedded systems. One of its useful features is its ability to handle interrupts. Interrupts allow the microcontroller to respond to external events without having to constantly poll for them in the main program loop. This can be useful in situations where a quick response is needed, such as in real-time applications.

It is recommended to explore more and use different setups for interrupts especially that it has so many use in real world problems. It is suggested to use different inputs and outputs and maybe different crystal oscillator or timer to really appreciate the digital signals.

Source Code: (2 .asm)

(Please Attach the asm file)

```
;##########################    COMMENT HEADER         ###########################
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  Author:        Jewel Anne A. Reyes                           ;
;  Date:    March 1, 2023                           ;
;  Version:       1.0.0                                 ;
;  Title:    Machine Problem 3.a                        ;
;                                                       ;
;  Description:   Processing interrupt caused by change on pin RB0/INT  ;
;                 the LED in RB7 pin is Blinking Fast               ;
;                 Change in RB0/INT pin, MCU= interrupt routine of     ;
;                 10 blinks of 1 sec delay of interval in RB7 pin   ;
;                                                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;#############   DECLARATION AND CONFIGURATION OF A PROCESSOR   ##############
list p=16f84a                ;Defining microcontroller model
#include <p16f84a.inc>             ;Processor library for PIC16
__CONFIG _CP_OFF & _PWRTE_ON & _WDT_OFF & _XT_OSC      ;Sets processor configuration bits


;##### DIRECTIVE EQU(equivalent) = text substitution for constant/variable #####
; Special Purpose Registers (SFR):
STATUS          EQU 03H          ;STATUS equivalent to 03H File Address
PORTA           EQU 05H
PORTB           EQU 06H
TRISA    EQU 85H
TRISB    EQU 86H
OPTION_REG  EQU 81H
INTCON          EQU         0BH
; General Purpose Registers (GPR):
cblock 0x20                  ;start of general purpose registers
        CounterA
        CounterB
        CounterC
        W_TEMP
        STATUS_TEMP
endc                         ;end of general purpose registers

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ORG 00H                  ;Start assembling lines below this statement at RESET vector (00h)
          GOTO Init
        ORG 04H                  ;Start assembling lines below this statement at Peripheral Interrupt Vector (04H)
          PUSH
                BCF INTCON, 0
                MOVWF W_TEMP
```

```
        SWAPF STATUS, W
        MOVWF STATUS_TEMP
ISR
        BCF INTCON, INTF
        BSF PORTB,7
        call Delay
        BCF PORTB,7
        call Delay

        BSF PORTB,7
        call Delay
        BCF PORTB,7
        call Delay

        BSF PORTB,7
        call Delay
        BCF PORTB,7
        call Delay

        BSF PORTB,7
        call Delay
        BCF PORTB,7
        call Delay

        BSF PORTB,7
        call Delay
        BCF PORTB,7
        call Delay

        BSF PORTB,7
        call Delay
        BCF PORTB,7
        call Delay

        BSF PORTB,7
        call Delay
        BCF PORTB,7
        call Delay

        BSF PORTB,7
        call Delay
        BCF PORTB,7
        call Delay

        BSF PORTB,7
        call Delay
```

```
                BCF PORTB,7
                call Delay
            POP
                SWAPF STATUS_TEMP,W
                MOVWF STATUS
                SWAPF W_TEMP, F
                SWAPF W_TEMP, W

        RETFIE                  ;Return from interrupt
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Init
        BSF STATUS, 5
        MOVLW b'00000001'
        MOVWF TRISB
        BSF OPTION_REG,7
        BSF OPTION_REG,6
        BCF STATUS,5
        BSF INTCON,INTE
        BSF INTCON,GIE
        BCF PORTB,7
        goto Blink
Blink
        BSF PORTB,7
        call Delay2

        BCF PORTB,7
        call Delay2
        goto Blink
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Delay
   MOVLW D'6'                   ;1 us
   MOVWF CounterC       ;1 us
   MOVLW D'24'                  ;1 us
   MOVWF CounterB       ;1 us
   MOVLW D'167'         ;1 us
   MOVWF CounterA               ;1 us
loop
   DECFSZ CounterA,1            ;167 x 1 cycle + 1 cycle = 168 us
   GOTO loop            ;166 * 2 us = 332 us
   DECFSZ CounterB,1            ;24 x 1 cycle + 1 cycle = 25 us
   GOTO loop            ;24 * 2 us = 48 us
   DECFSZ CounterC,1            ;6 x 1 cycle + 1 cycle = 7 us
   GOTO loop            ;5 * 2 us = 10 us
   NOP                          ;1 us
                        ;TOTAL = 597 us
   RETURN               ;2 us
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Delay2
    MOVLW D'1'                  ;1 us
    MOVWF CounterC          ;1 us
    MOVLW D'24'                 ;1 us
    MOVWF CounterB          ;1 us
    MOVLW D'167'            ;1 us
    MOVWF CounterA              ;1 us
loop2
    DECFSZ CounterA,1           ;167 x 1 cycle + 1 cycle = 168 us
    GOTO loop2                  ;166 * 2 us = 332 us
    DECFSZ CounterB,1           ;24 x 1 cycle + 1 cycle = 25 us
    GOTO loop2                  ;24 * 2 us = 48 us
    DECFSZ CounterC,1           ;6 x 1 cycle + 1 cycle = 7 us
    GOTO loop2                  ;5 * 2 us = 10 us
    NOP                         ;1 us
                            ;TOTAL = 597 us
                            ;3A + 770B + 197,122C - 197,879
                            ;3(167) + 770(24) + 197,122(6) - 197,879
                            ;501 + 18,480 + 1,182,732 - 197,879
                            ;1,201,713 - 197,879
                            ;DELAY = 1,003,834 us == 1.003 seconds
    RETURN                  ;2 us
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
end
```

```
;############################     COMMENT HEADER           ###############################
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  Author:          Jewel Anne A. Reyes                                              ;
;  Date:    March 1, 2023                                        ;
;  Version:         1.0.0                                                  ;
;  Title:    Machine Problem 3.b                                           ;
;                                                                          ;
;  Description:     Processing interrupt caused by changes on pins RB4-RB7          ;
;                   LED in RB0 to RB3 will recurrence of counting 0–9 BCD           ;
;                   Change in RB4-RB7 pin, MCU= interrupt routine of                ;
;                   10 blinks of 1 sec delay of interval in RB0-RB3 pin             ;
;                                                                          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;##############    DECLARATION AND CONFIGURATION OF A PROCESSOR    ###############
list p=16f84a                    ;Defining microcontroller model
#include <p16f84a.inc>                  ;Processor library for PIC16
__CONFIG _CP_OFF & _PWRTE_ON & _WDT_OFF & _XT_OSC        ;Sets processor configuration bits


;##### DIRECTIVE EQU(equivalent) = text substitution for constant/variable #####
; Special Purpose Registers (SFR):
STATUS          EQU 03H             ;STATUS equivalent to 03H File Address
PORTA           EQU 05H
PORTB           EQU 06H
TRISA    EQU 85H
TRISB    EQU 86H
OPTION_REG  EQU 81H
INTCON          EQU         0BH
; General Purpose Registers (GPR):
cblock 0x20                      ;start of general purpose registers
         CounterA
         CounterB
         CounterC
         W_TEMP
         STATUS_TEMP
endc                             ;end of general purpose registers

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
         ORG 00H                        ;Start assembling lines below this statement at RESET vector (00h)
            GOTO Init
         ORG 04H                        ;Start assembling lines below this statement at Peripheral Interrupt Vector (04H)
         PUSH
            MOVWF W_TEMP
            MOVF STATUS, 0
```

```
    MOVWF STATUS_TEMP
ISR
    BCF INTCON, 0

    BSF PORTB, 0
    BSF PORTB, 1
    BSF PORTB, 2
    BSF PORTB, 3
    CALL Delay

    BCF PORTB, 0
    BCF PORTB, 1
    BCF PORTB, 2
    BCF PORTB, 3
    CALL Delay

    BSF PORTB, 0
    BSF PORTB, 1
    BSF PORTB, 2
    BSF PORTB, 3
    CALL Delay

    BCF PORTB, 0
    BCF PORTB, 1
    BCF PORTB, 2
    BCF PORTB, 3
    CALL Delay

    BSF PORTB, 0
    BSF PORTB, 1
    BSF PORTB, 2
    BSF PORTB, 3
    CALL Delay

    BCF PORTB, 0
    BCF PORTB, 1
    BCF PORTB, 2
    BCF PORTB, 3
    CALL Delay

    BSF PORTB, 0
    BSF PORTB, 1
    BSF PORTB, 2
    BSF PORTB, 3
    CALL Delay
```

```
BCF PORTB, 0
BCF PORTB, 1
BCF PORTB, 2
BCF PORTB, 3
CALL Delay

BSF PORTB, 0
BSF PORTB, 1
BSF PORTB, 2
BSF PORTB, 3
CALL Delay

BCF PORTB, 0
BCF PORTB, 1
BCF PORTB, 2
BCF PORTB, 3
CALL Delay

BSF PORTB, 0
BSF PORTB, 1
BSF PORTB, 2
BSF PORTB, 3
CALL Delay

BCF PORTB, 0
BCF PORTB, 1
BCF PORTB, 2
BCF PORTB, 3
CALL Delay

BSF PORTB, 0
BSF PORTB, 1
BSF PORTB, 2
BSF PORTB, 3
CALL Delay

BCF PORTB, 0
BCF PORTB, 1
BCF PORTB, 2
BCF PORTB, 3
CALL Delay

BSF PORTB, 0
BSF PORTB, 1
BSF PORTB, 2
```

```
            BSF PORTB, 3
            CALL Delay

            BCF PORTB, 0
            BCF PORTB, 1
            BCF PORTB, 2
            BCF PORTB, 3
            CALL Delay

            BSF PORTB, 0
            BSF PORTB, 1
            BSF PORTB, 2
            BSF PORTB, 3
            CALL Delay

            BCF PORTB, 0
            BCF PORTB, 1
            BCF PORTB, 2
            BCF PORTB, 3
            CALL Delay

            BSF PORTB, 0
            BSF PORTB, 1
            BSF PORTB, 2
            BSF PORTB, 3
            CALL Delay

            BCF PORTB, 0
            BCF PORTB, 1
            BCF PORTB, 2
            BCF PORTB, 3
            CALL Delay
          POP
            MOVF STATUS_TEMP, 0
            MOVWF STATUS
            MOVF W_TEMP, 0
            goto Start
          retfie
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Init
    BSF STATUS,5
    MOVLW 0xF0 ; Set TRISB<3:0> = 0 (output) and TRISB<7:4> = 1 (input)
    MOVWF TRISB
    BSF OPTION_REG, 6
    BSF OPTION_REG, 7
    BCF STATUS, 5
```

```
    BSF INTCON, RBIE ; change to 3
    BSF INTCON, GIE
    goto Start
Start
    ;0
    BCF PORTB, 0
    BCF PORTB, 1
    BCF PORTB, 2
    BCF PORTB, 3
    CALL Delay

    ;1
    BCF PORTB, 0
    BCF PORTB, 1
    BCF PORTB, 2
    BSF PORTB, 3
    CALL Delay

    ;2
    BCF PORTB, 0
    BCF PORTB, 1
    BSF PORTB, 2
    BCF PORTB, 3
    CALL Delay

    ;3
    BCF PORTB, 0
    BCF PORTB, 1
    BSF PORTB, 2
    BSF PORTB, 3
    CALL Delay

    ;4
    BCF PORTB, 0
    BSF PORTB, 1
    BCF PORTB, 2
    BCF PORTB, 3
    CALL Delay

    ;5
    BCF PORTB, 0
    BSF PORTB, 1
    BCF PORTB, 2
    BSF PORTB, 3
    CALL Delay
```

```
    ;6
    BCF PORTB, 0
    BSF PORTB, 1
    BSF PORTB, 2
    BCF PORTB, 3
    CALL Delay

    ;7
    BCF PORTB, 0
    BSF PORTB, 1
    BSF PORTB, 2
    BSF PORTB, 3
    CALL Delay

    ;8
    BSF PORTB, 0
    BCF PORTB, 1
    BCF PORTB, 2
    BCF PORTB, 3
    CALL Delay

    ;9
    BSF PORTB, 0
    BCF PORTB, 1
    BCF PORTB, 2
    BSF PORTB, 3
    CALL Delay
    goto Start
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Delay
    MOVLW D'6'                  ;1 us
    MOVWF CounterC          ;1 us
    MOVLW D'24'                 ;1 us
    MOVWF CounterB          ;1 us
    MOVLW D'167'            ;1 us
    MOVWF CounterA              ;1 us
loop
    DECFSZ CounterA,1           ;167 x 1 cycle + 1 cycle = 168 us
    GOTO loop               ;166 * 2 us = 332 us
    DECFSZ CounterB,1           ;24 x 1 cycle + 1 cycle = 25 us
    GOTO loop               ;24 * 2 us = 48 us
    DECFSZ CounterC,1           ;6 x 1 cycle + 1 cycle = 7 us
    GOTO loop               ;5 * 2 us = 10 us
    NOP                         ;1 us
    RETURN                  ;2 us
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
end
```

Additionals:

(Please attach any changes in the setup including images and schematics)