EE2016 Lab

Experiment 2: Interrupts and Timers in Atmel AVR Atmega

Manomukil T EE20B075

27th September 2021

Contents

1 Aim:

Using Atmel AVR assembly language programming, implement interrupts and timers in Atmel Atmega microprocessor. To implement the blinking of an LED 10 times once switch is pressed in assembly and C programming using :

1. INT1 Interrupt
2. INT0 Interrupt

2 Questions:

1. Fill in the blanks in the assembly code
2. Use int0 to redo the same in the demo program (duely filled in). Once the switch is pressed the LED should blink 10 times (ON (or OFF) - 1 sec, duty cycle could be 50 %). Demonstrate both the cases.
3. Rewrite the program in 'C' (int1). Rewrite the C program for int0.
4. Demonstrate both the cases (of assembly and C).

3 Solution:

3.1 Question1:Fill in the blanks in the assembly code.

A textfile was given containing a program to enable interrupt1 with some of the code incomplete. The code has been filled and the entire code is displayed below:

```
;
; INT1.asm
;
; Created: 26-09-2021 20:45:54
; Author : ManomukilT
;


; Replace with your application code
;Start
.org 0;
rjmp reset;

.org 0x0002;
rjmp int1_ISR;

.org 0x0100;
```

```
reset:
        LDI R16,0x70; Loading Stacker Point Address
        OUT SPL,R16;
        LDI R16,0x00;
        OUT SPH,R16;

        LDI R16,0x01; Interface PortB pin0 to be output
        OUT DDRB,R16; so to view LED Blinking

        LDI R16,0x00;
        OUT DDRD,R16;

        OUT MCUCR, R16;Set MCUCR register to enable low level interrupt

                LDI R16, 0x80;Set GICR Resgister to enable interrupt 1
        OUT GICR, R16;

        LDI R16, 0x00;
        OUT PORTB, R16;

        SEI;

ind_loop:
    RJMP ind_loop;

int1_ISR:
    IN R16, SREG;
    PUSH R16;

    LDI R16, 0x0A;
    MOV R0, R16;

    c1:  LDI R16, 0x01;
    OUT PORTB, R16

    LDI R16, 0x05
    a1:  LDI R17, 0xFF
    a2:  LDI R18,0xFF
         a3:      DEC R18
         BRNE a3
         DEC R17
    BRNE a2
    DEC R16
    BRNE a1
```

```
    LDI R16, 0x00
    OUT PORTB, R16

    LDI R16, 0x05
    b1:  LDI R17, 0xFF
    b2:  LDI R18, 0xFF
         b3:  DEC R18
         BRNE b3
         DEC R17
    BRNE b2
    DEC R16
    BRNE b1

    DEC R0
    BRNE c1
    POP R16
    OUT SREG, R16

    RETI
```

3.2 Question2:Redoing the above program with INT0

The same program has been implemented with interrupt0 and the code is displayed below:

```
;
; INT0_ASM.asm
;
; Created: 26-09-2021 20:45:54
; Author : ManomukilT
;


; Replace with your application code
;Start
.org 0;
rjmp reset;

.org 0x0001;
rjmp int0_ISR;

.org 0x0100;

reset:
        LDI R16,0x70; Loading Stacker Point Address
        OUT SPL,R16;
```

```
        LDI R16,0x00;
        OUT SPH,R16;

        LDI R16,0x01; Interface PortB pin0 to be output
        OUT DDRB,R16; so to view LED Blinking

        LDI R16,0x00;
        OUT DDRD,R16;

        OUT MCUCR, R16;Set MCUCR register to enable low level interrupt

                LDI R16, 0x40;Set GICR Resgister to enable interrupt 1
        OUT GICR, R16;

        LDI R16, 0x00;
        OUT PORTB, R16;

        SEI;

ind_loop:
    RJMP ind_loop;

int0_ISR:
    IN R16, SREG;
    PUSH R16;

    LDI R16, 0x0A;
    MOV R0, R16;

    c1:  LDI R16, 0x01;
    OUT PORTB, R16

    LDI R16, 0x05
    a1:  LDI R17, 0xFF
    a2:  LDI R18,0xFF
         a3:      DEC R18
         BRNE a3
         DEC R17
    BRNE a2
    DEC R16
    BRNE a1

    LDI R16, 0x00
    OUT PORTB, R16

    LDI R16, 0x05
```

```
b1:   LDI R17, 0xFF
b2:   LDI R18, 0xFF
      b3:   DEC R18
      BRNE b3
      DEC R17
BRNE b2
DEC R16
BRNE b1

DEC R0
BRNE c1
POP R16
OUT SREG, R16

RETI
```

3.3 Rewrite the program in 'C' (int1). Rewrite the C program for int0.

3.3.1 C code with INT1

The code for the program with interrupt1:

```
/*
 * main.c
 *
 * Created: 27-09-2021 08:36:50
 * Author : ManomukilT
 */

#define F_CPU 1000000  // clock frequency

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

ISR (INT1_vect)
{
        int i;
        for (i=1;i<=10;i++) // for 10 times LED blink

        {
                PORTB=0x01;
                _delay_ms(1000);    // delay of 1 sec
                PORTB=0x00;
                _delay_ms(1000);
```

```
        }


}
int main(void)
{
        //Set the input/output pins appropriately
        //To enable interrupt and port interfacing
        //For LED to blink
        DDRD=0x00;    //Set appropriate data direction for D
        DDRB=0x00;    //Make PB0 as output
        MCUCR=0x00;   //Set MCUCR to level triggered
        GICR=0x80;    //Enable interrupt 1
        PORTB=0x00;
        sei();        // global interrupt flag

        while (1) //wait
        {

        }
}
```

3.3.2 C code with INT0

The code for the program with interrupt0:

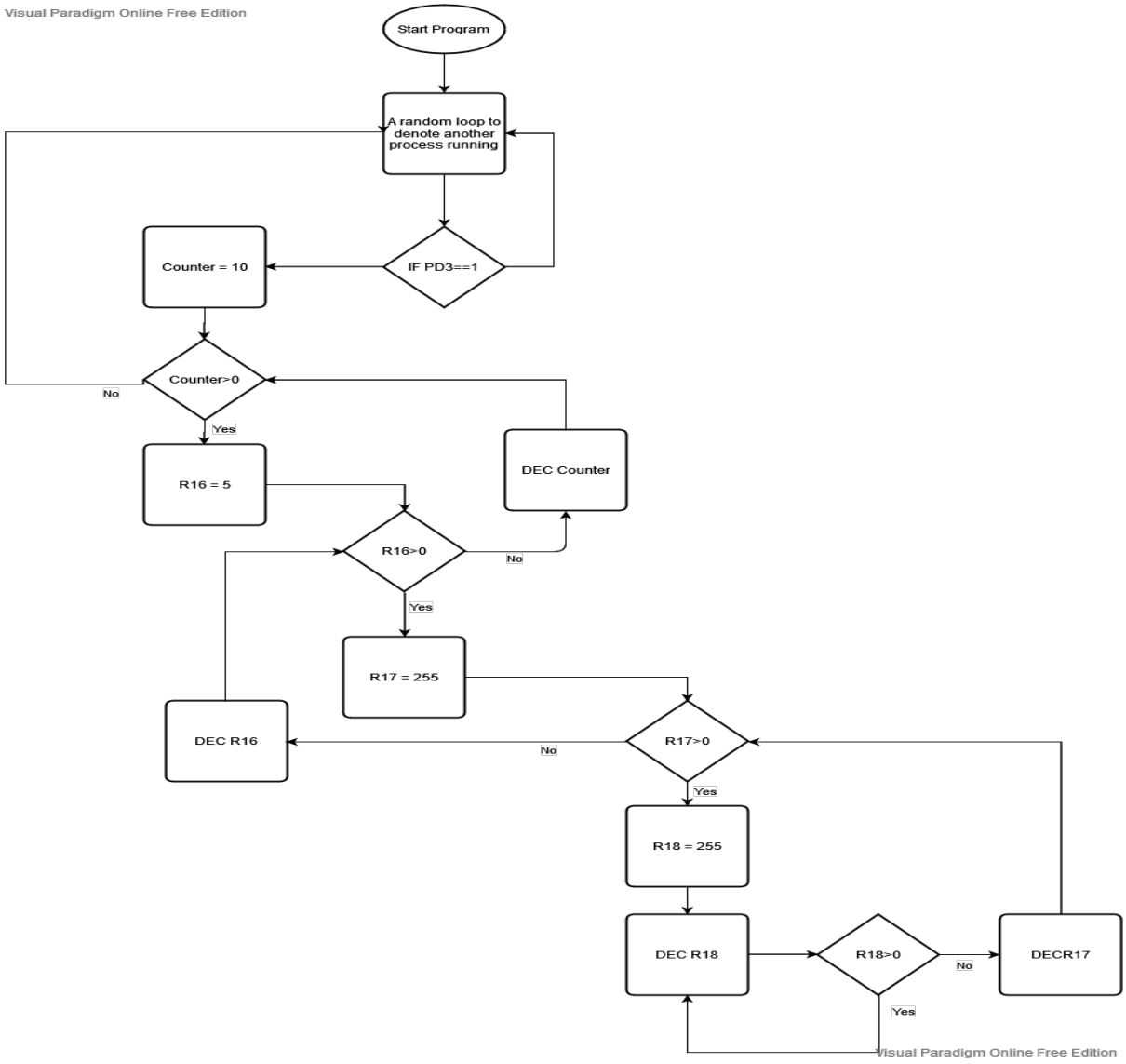
```
/*
 * main.c
 *
 * Created: 27-09-2021 08:36:50
 * Author : ManomukilT
 */

#define F_CPU 1000000  // clock frequency

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

ISR (INT1_vect)
{
        int i;
        for (i=1;i<=10;i++) // for 10 times LED blink

        {
                PORTB=0x01;
```

```
                _delay_ms(1000);    // delay of 1 sec
                PORTB=0x00;
                _delay_ms(1000);

        }


}
int main(void)
{
        //Set the input/output pins appropriately
        //To enable interrupt and port interfacing
        //For LED to blink
        DDRD=0x00;    //Set appropriate data direction for D
        DDRB=0x00;    //Make PB0 as output
        MCUCR=0x00;   //Set MCUCR to level triggered
        GICR=0x80;    //Enable interrupt 1
        PORTB=0x00;
        sei();        // global interrupt flag

        while (1) //wait
        {

        }
}
```

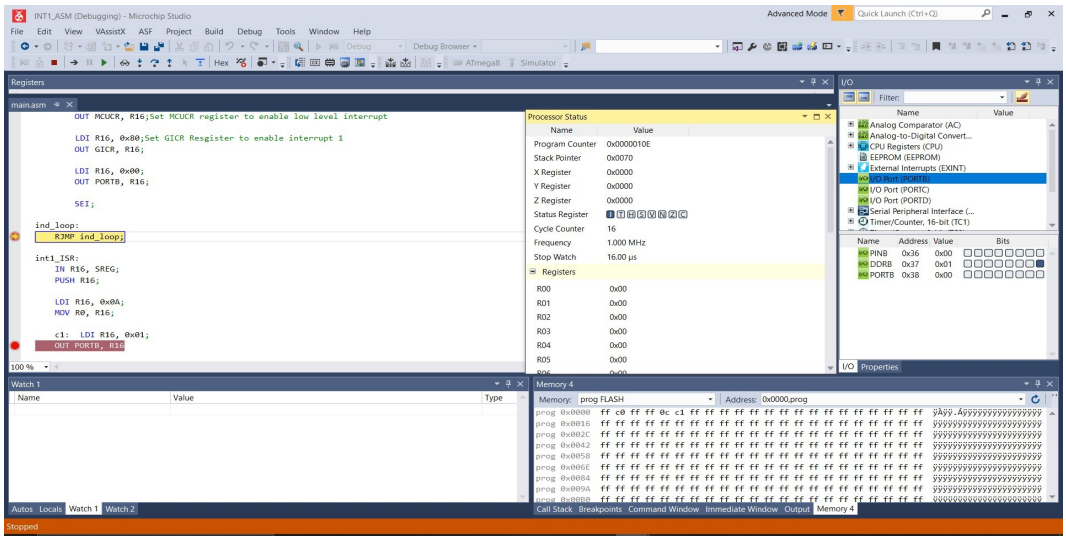
3.4 Demonstrate both the cases (of assembly and C).

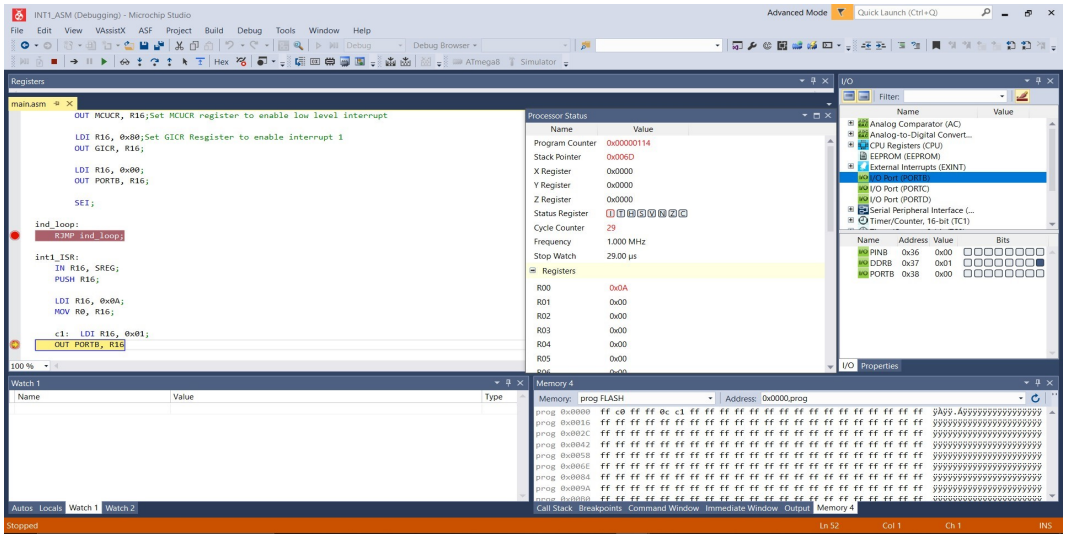
All the four programs have the same logic and flow. Only the functions for each program and language differs. The logic and flow of all the four programs is shown in the next page:

3.4.1 Flowchart:

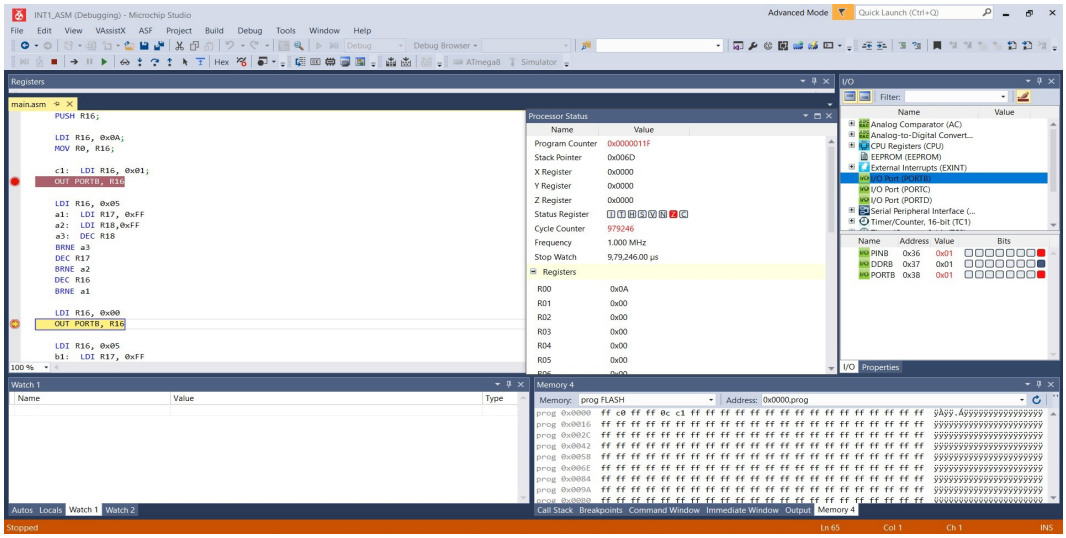


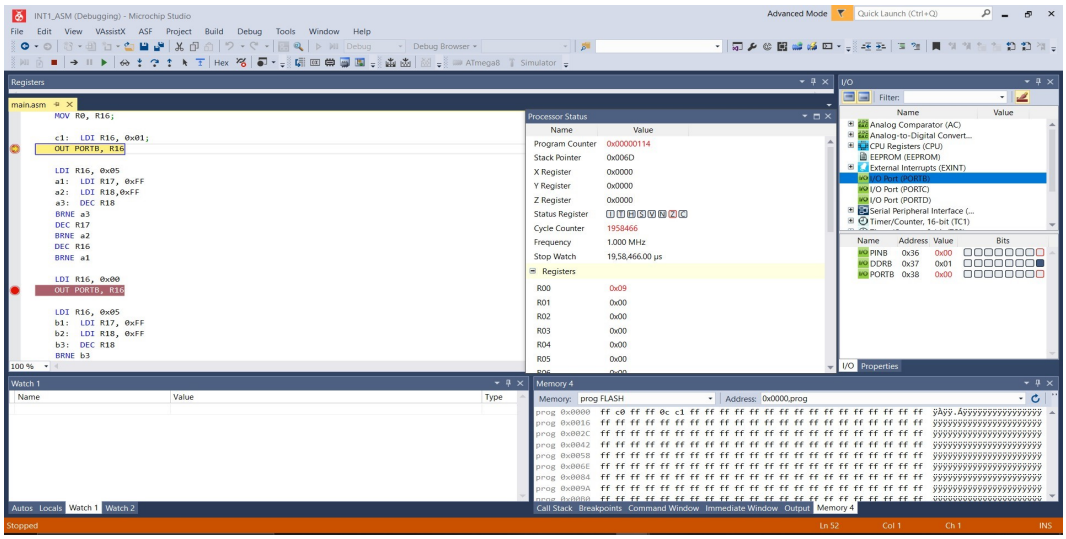
3.4.2 Outputs:



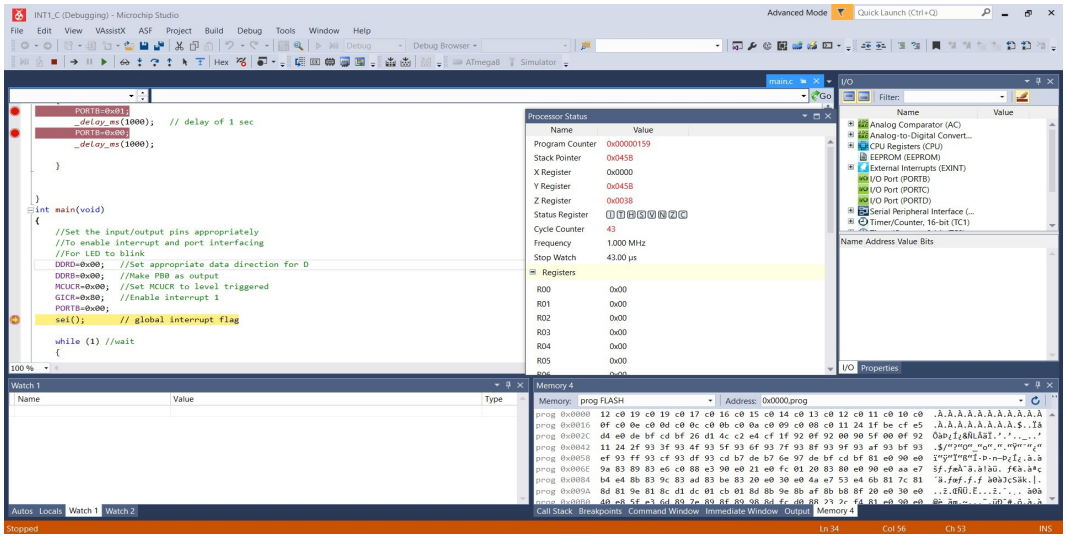


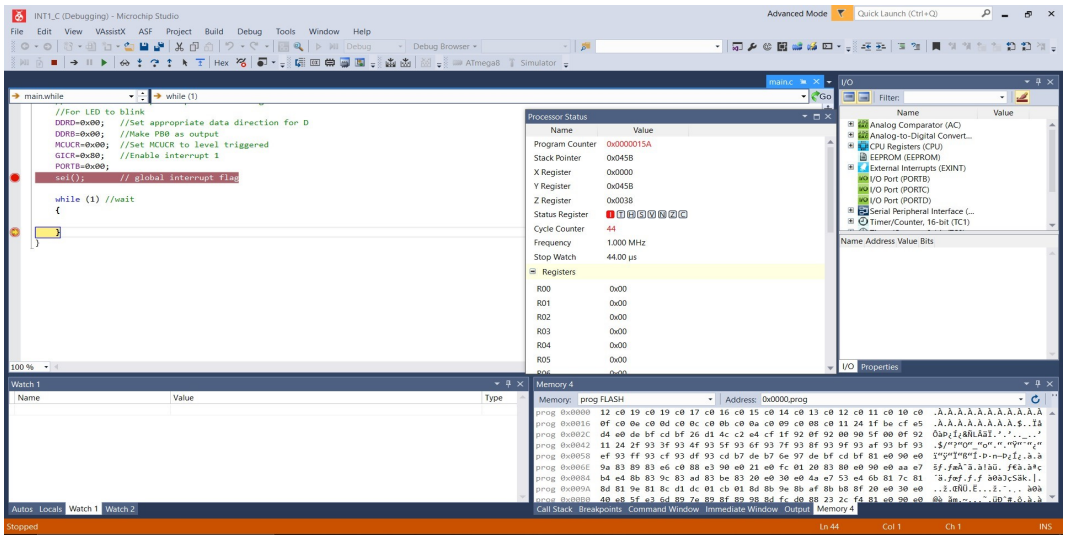
The above outputs are for INT1.asm



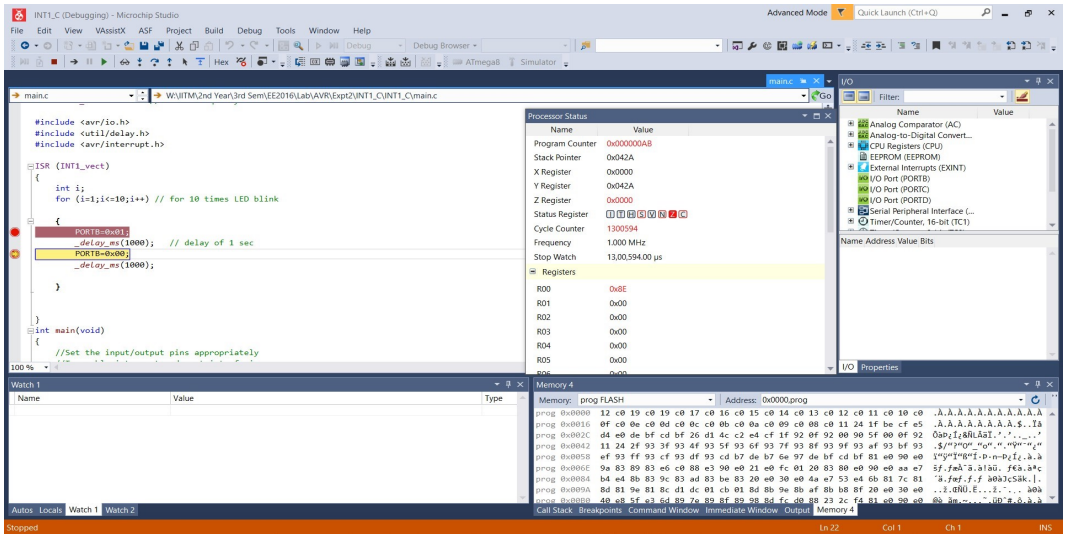


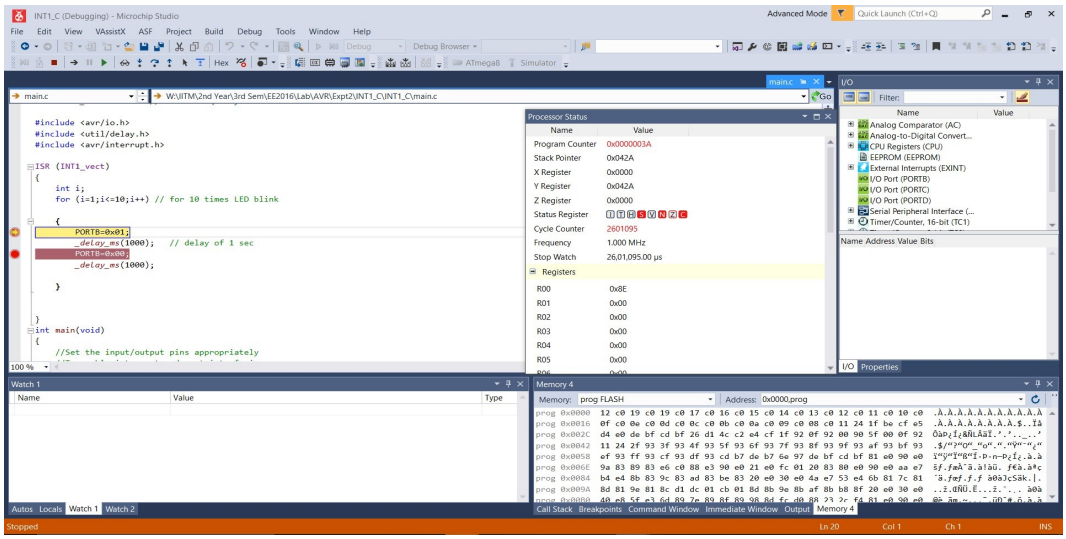
The above outputs are for INT1.asm



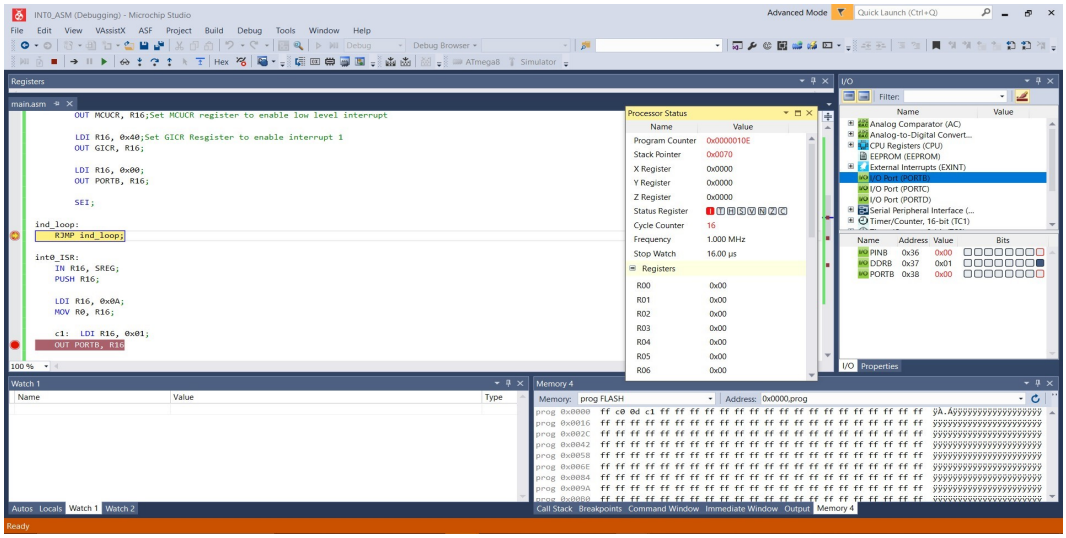


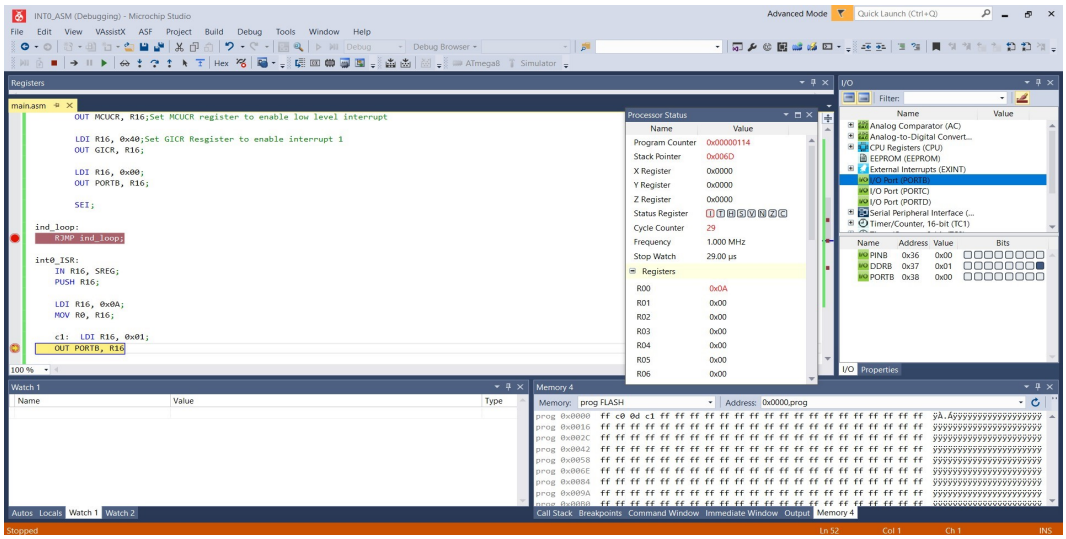
The above outputs are for INT1.c



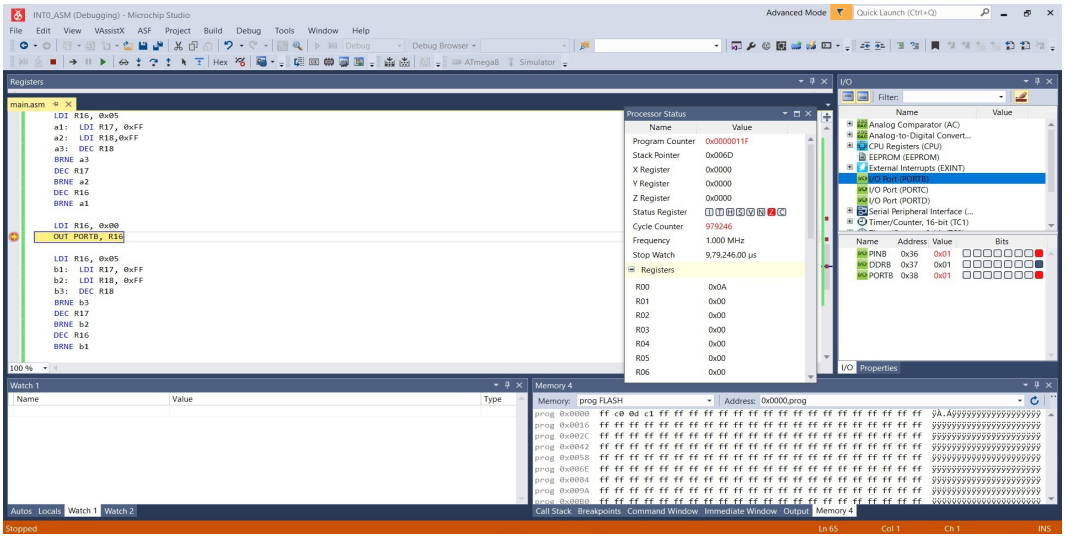


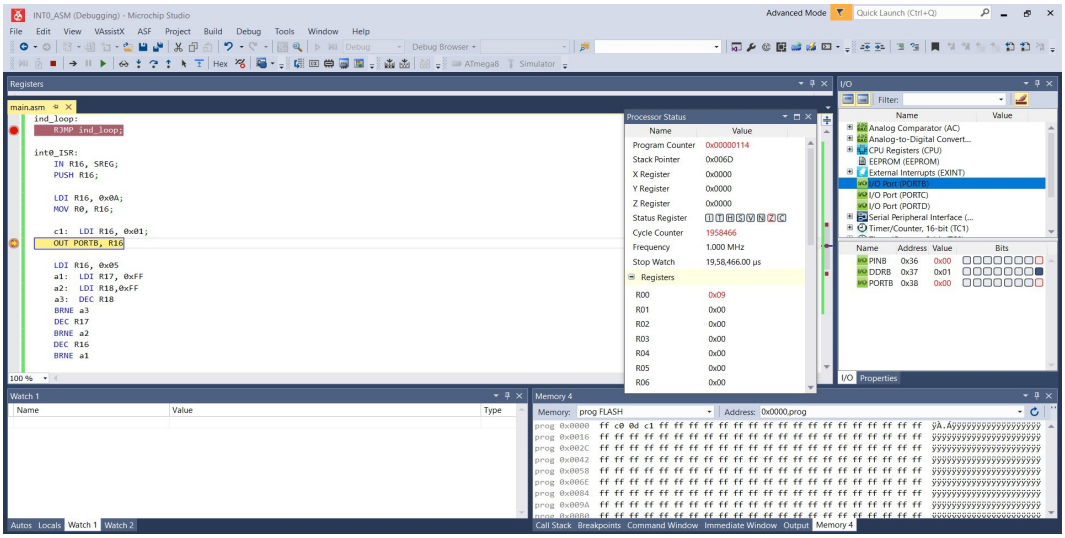
The above outputs are for INT1.c



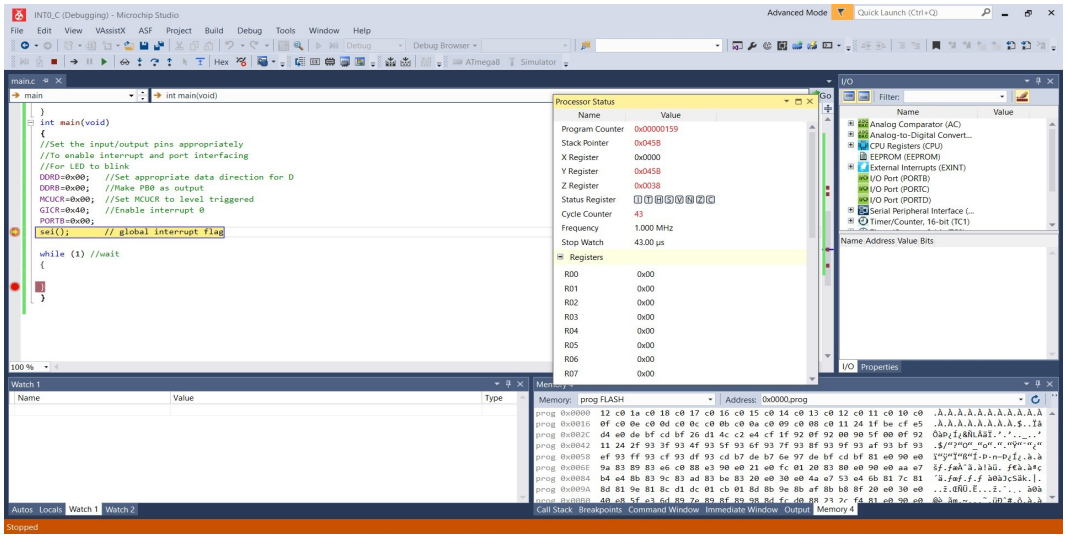


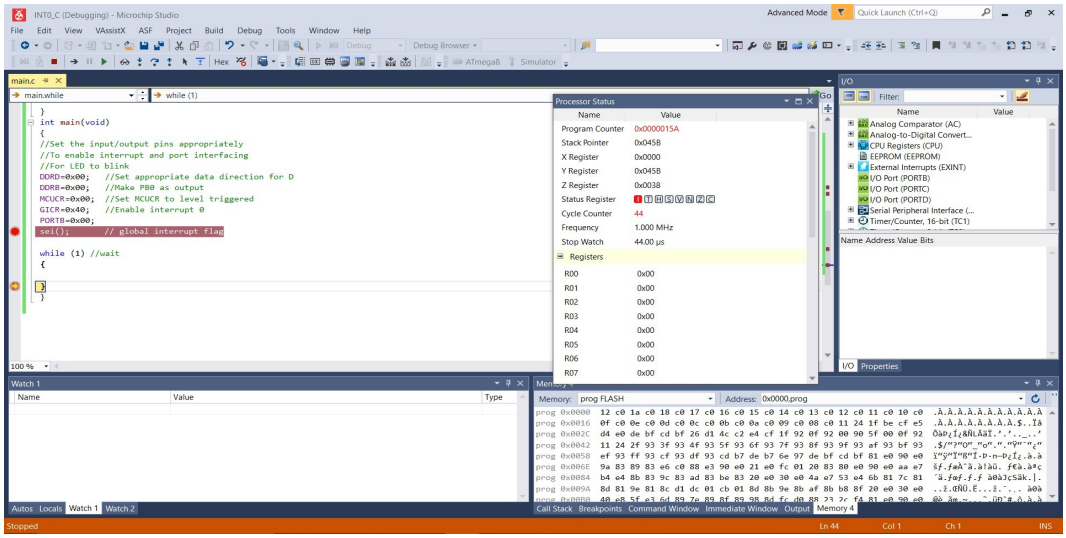
The above outputs are for INT0.asm



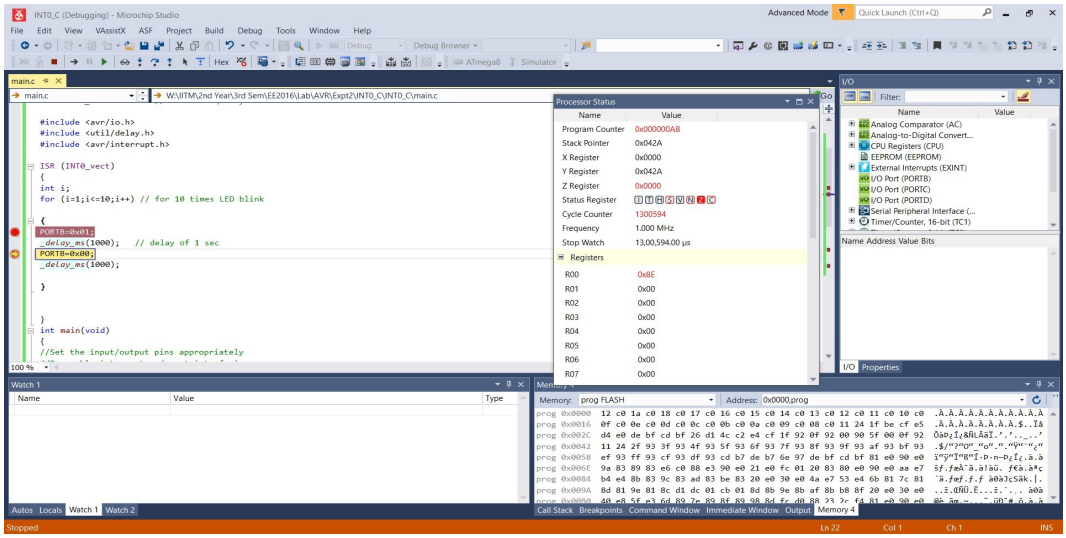


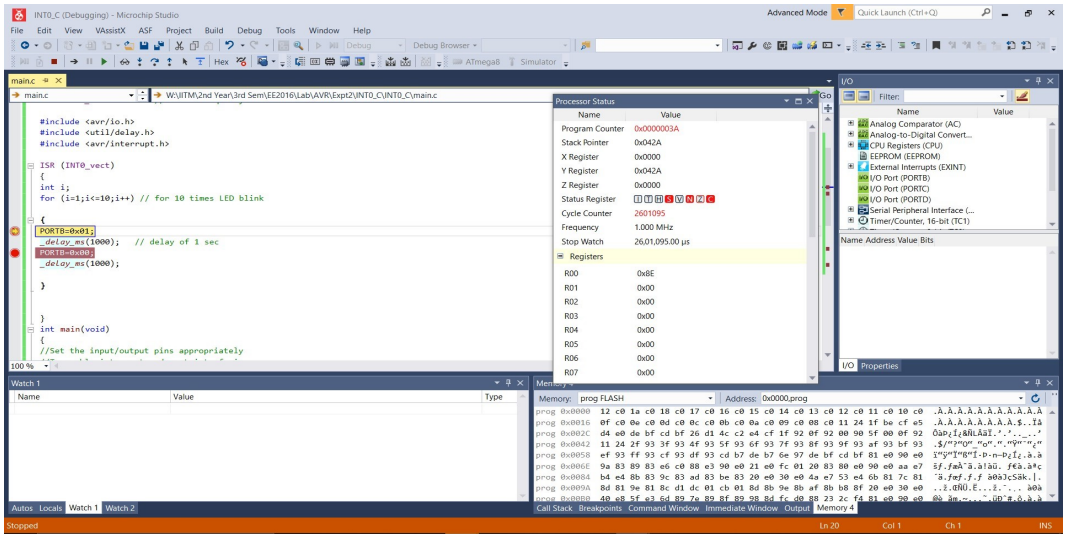
The above outputs are for INT0.asm





The above outputs are for INT0.c





The above outputs are for INT0.c

4 Conclusion:

The following was learnt about through this experiment:

1. Using External Inputs with MCUCR and GICR
2. Interfacing Inputs and Outputs
3. Generating delays without the use of timers using nested loops
4. Programming Assembly code in C

Further an idea was established on how these concepts could be implemented for use in the real-world