LAB 6

EXTERNAL INTERRUPT

1. INTRODUCTION

This lab gives information about external interrupt mechanism of CSM12C32. What are the differences between polling and interrupt method? Read the following documents about interrupts and polling.

Reading Material:

Interrupts_Polling.pdf (Read interrupts and polling slides)

CSM12C32 Schematic.pdf (Find SW1 and its connection)

MC9S12C32 Datasheet.pdf (Read interrupt chapter, understand interrupt vectors)

2. INTERRUPT

External interrupt is commonly used for asynchronous communication. For example, your microprocessor could wait signal from another chip, processor etc. When this signal comes, your processor begins processing. There are two ways to detect these types of external signals.

a. Polling Method

This method continuously checks the signal. Until the signal comes, the processor cannot execute any process except checking for the signal. If the processor tries to execute other processes too long, it could miss the signal. This method may be inefficient if the external signal is not stationary (i.e. external process does not wait and keep the signal to be detected).

```
while(1){
      if( CheckSignal()==1 ){
            //Your Program
      }
      //if some program is here,
      //the processor could miss the signal depending on the length
      //of the program
}
```

b. Interrupt

This method does not check the signal. When signal comes, your microprocessor immediately jumps to the interrupt service routine. After finishing the interrupt service routine, processor continues execution from where it left off

```
while(1){
      //Some programs could be here and its size do not affect
      //anything.
}
void ISR(void){ //When interrupt occurs, processor executes this
                 //function

      //Your Program
}
```

3. EXPERIMENT

a. Interrupt

Write a program according to the following specifications.

Initially, LED1 is off and LED is on. When SW1 is pressed, LED states are inverted.

LED1	LED2
On	Off

After pressing SW1 (Initially, SW1 is released)

Off	On

After releasing SW1 (LED states are not changed)

Off	On

After pressing SW1 again

On	Off

After releasing SW1 (LED states are not changed)

On	Off

.....

LED states are changed, when released-pressed transition occurs.

Draft assembly code is given on the next page. Draft code initializes interrupt on SW1. Please write your main program and interrupt service routine.

(Hint: CSM12C32 enters interrupt service routine, when released-pressed transition occurs)

b. Polling

Write a program according to same specifications as above. But, do not use interrupt. Try to use polling method. (Hint: Firstly, initialize PE0 as input using DDRE. To read PE0, store PORTE value and mask all bits except 0th bit).

In your report, explain interrupt working mechanism and each step of your code.

```
; export symbols
           XDEF Entry       ; export 'Entry' symbol
           ABSENTRY Entry   ; for absolute assembly: mark this as
                            ; application entry point

; include derivative specific macros
           INCLUDE 'mc9s12c32.inc'

;absolute address to place my code/constant data
ROMStart   EQU  $4000

; variable/data section
           ORG RAMStart
Counter1   DC.W $FFFF
Counter2   DC.W $0010

; code section
           ORG   ROMStart
Entry:

           LDS     #RAMEnd+1    ; initialize the stack pointer
           ADDCC   #$BF         ; Enable XIRQ interrupt on SW1

           /////////////////////
           //YOUR MAIN PROGRAM//
           /////////////////////


ISR:       //////////////////////////
           //YOUR INTERRUPT PROGRAM//
           //////////////////////////

           RTI

           ORG   $FFF4 //XIRQ interrupt vector
           DC.W  ISR   //Beginning address of ISR
```