EE 128
Lab 3: Interrupt Handling

Section 021
TA: Mohsen Karimi

Tanish Arora
Leya Zeng

1. Abstract

This Lab's main objectives are to getting familiar with :

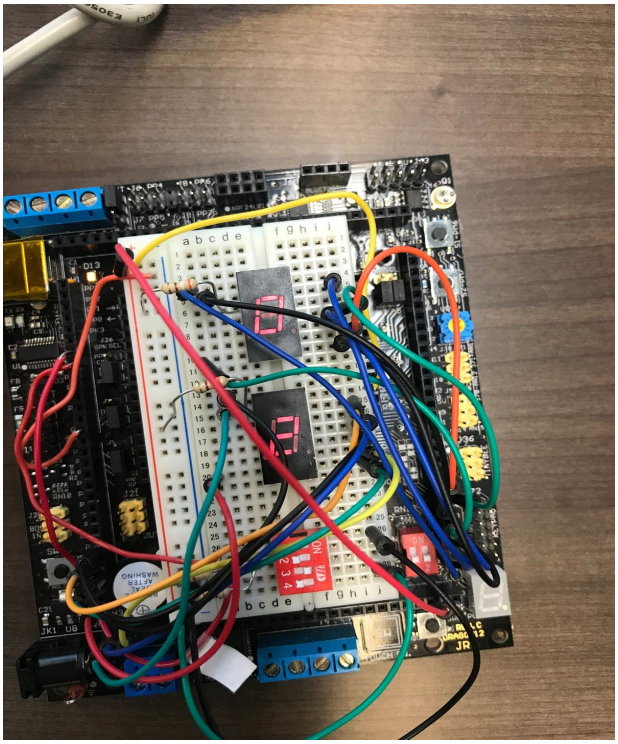
- To verify the concepts we learned during the lecture and become familiar with the interrupt-handling techniques for the 9S12DG256 microcontroller
- This set of exercises will also help in improving skills for designing and debugging the microcontroller based systems.

2. Procedure

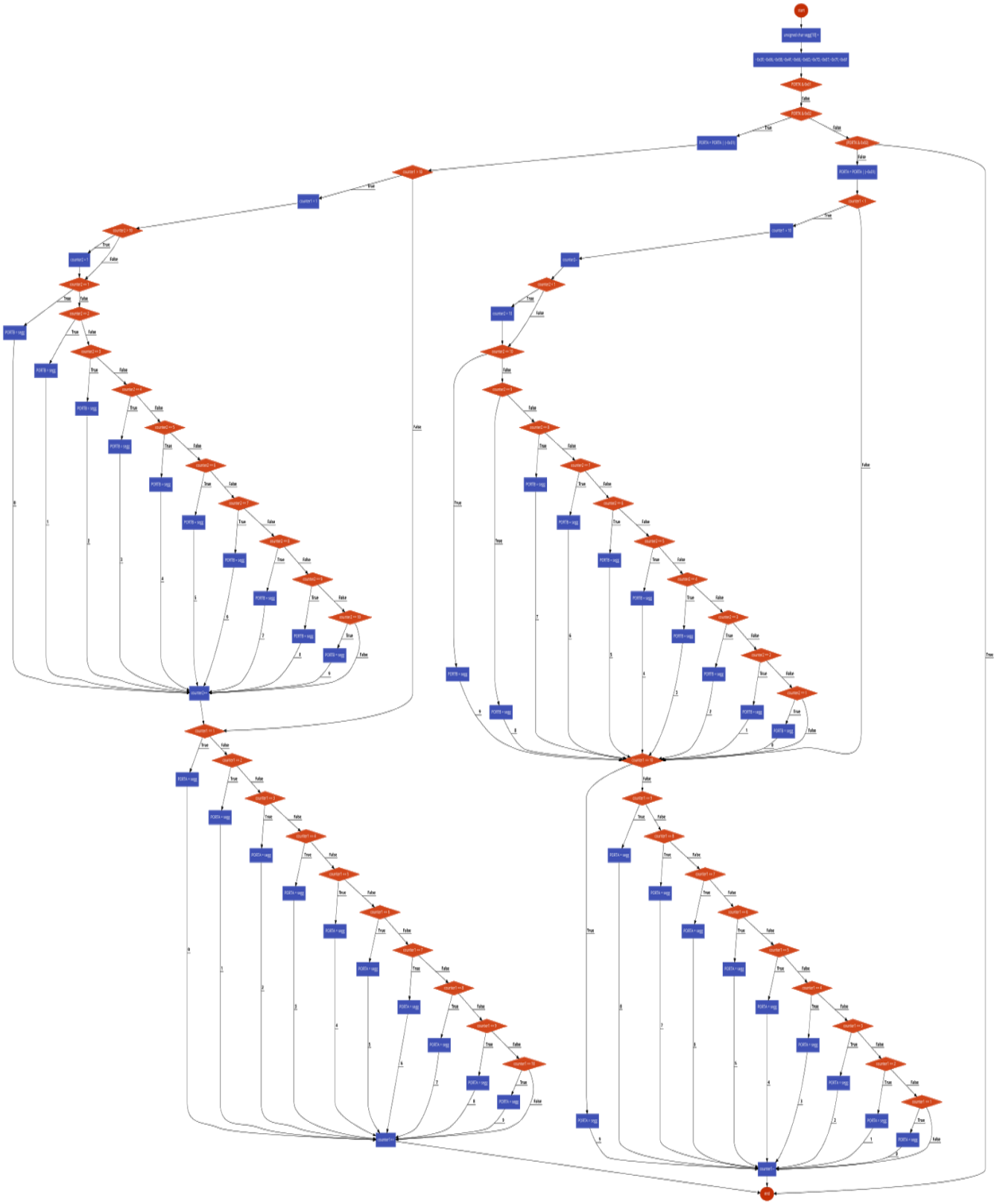
1) Open FreeScale CodeWarrior and start working on the new project. Also, work on the schematic first to make things more clear.
2) Work on the source code to create a two-decade, bi-directional counter, (whose value ranges from 0 through 99 decimal), driven by the same CLK signal will be created. And the operation of the counter works on the digital signals from the dip switch where
   CNT_HLD : 0 for normal operation, and 1 for hold;
   CNT_DIR : 0 for upward counting , and 1 for downward
3) Make sure that the hardware is connected properly i..e, that the LED 7 segment display is connected according to the code to produce different outputs.

3. Experiment System Specification :

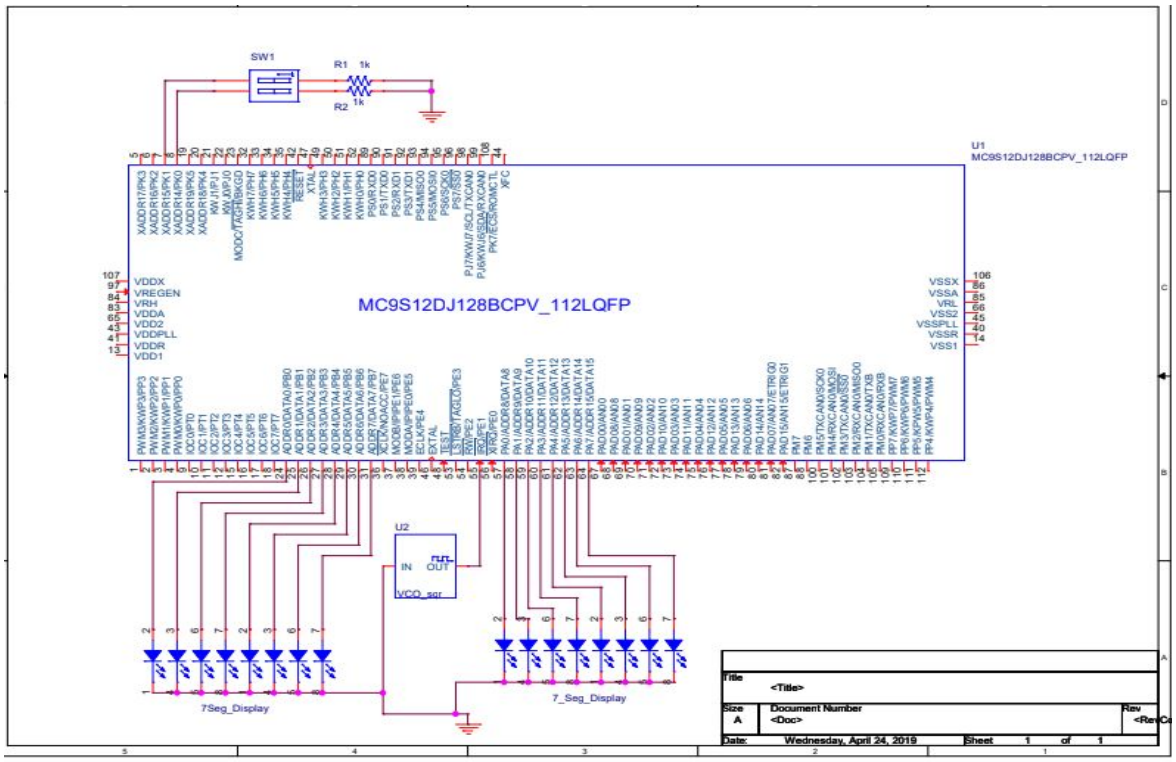
***Lab hardware photo*:**



Flowchart;



4. Hardware Design :



5. Software Design:

High Level Description :

The main point of this program is the interrupt service routine ISR with signal coming from the function generator and detecting the falling edge of the clock signal.

We use two counters for PORTA and PORTB, with PORTK set as switch pin for Hold/Count.

For each interrupt, we check for the input values from PK0 and PK1 using a dip switch, if PK0 is on, we implement a hold on the counting whether in the upward direction or in the downward direction ,which means the 7 segment LED stops at current display number, if PK0 is off which means the bi-directional decade counter will continue normal operation. The direction of the counter depends upon the input from PK1, if on, its in the upward direction, and if off, it continues in the downward direction.

To implement a bi directional two decade counting, We initially set PORTB to be a higher number and PORTA to be the lowest number.

Seg array is defined in which we set the 10 different hex values for the 7 segment LED number display, since we are using common anode LED, we had to negate all defined values.

Each time the program performs an interrupt, counter1 is added by 1, and the corresponding PORTA's value is changed. Whenever the counter1 value reaches over 10, counter2 will increase by 1, and each time counter1 falls below 1, counter2 will decrease by 1, PORTB will then display the result accordingly. (We set the counter values between 1 ~ 10 to

avoid negative counter values for which counter1 < 0 would occur). When PORTB, the higher number reach over 10 or below 1, we set counter2 to 1 or 10 again, so the number would display 0 from 9 and 9 from 0.

Program Listing :

```
#include <hidef.h>      /* common defines and macros */
#include "derivative.h"      /* derivative-specific definitions */
#include <mc9s12dg256.h>

typedef void (*near tIsrFunc)(void); /* Type of interrupt vector
entry */
__interrupt void IRQ_ISR(void); /* Declaration of ISR for IRQ */
const tIsrFunc _vect @0x3E72 = IRQ_ISR; /* 0x3E72: entry for IRQ */
// Let counter be a static variable initialized to 0
static unsigned char counter1 = 1;
static unsigned char counter2 = 1;

__interrupt void IRQ_ISR(void)
{
  //Update the clock tick and its display;
  //Read Port K;
  unsigned char segg[10] =
{~0x3F,~0x06,~0x5B,~0x4F,~0x66,~0x6D,~0x7D,~0x07,~0x7F,~0x6F};

  //if (CNT_HLD is 1) return; /* rotate hold */
  if (PORTK & 0x01) return;
  //if (CNT_DIR is 1) /* rotate left */
  if (PORTK & 0x02) {
  PORTB = PORTB | (~0x01);
    //rotate left -- wrap around if end is reached;
    if (counter1 > 10) {
      //counter2++;
      counter1 = 1;

      if (counter2 > 10) counter2 = 1;
      if (counter2 == 1) PORTB = segg[0];
      else if(counter2 == 2) PORTB = segg[1];
      else if(counter2 == 3) PORTB = segg[2];
      else if(counter2 == 4) PORTB = segg[3];
      else if(counter2 == 5) PORTB = segg[4];
      else if(counter2 == 6) PORTB = segg[5];
      else if(counter2 == 7) PORTB = segg[6];
      else if(counter2 == 8) PORTB = segg[7];
      else if(counter2 == 9) PORTB = segg[8];
      else if(counter2 == 10) PORTB = segg[9];
```

```
    counter2++;
  }
  if (counter1 == 1) PORTA = segg[0];
  else if(counter1 == 2) PORTA = segg[1];
  else if(counter1 == 3) PORTA = segg[2];
  else if(counter1 == 4) PORTA = segg[3];
  else if(counter1 == 5) PORTA = segg[4];
  else if(counter1 == 6) PORTA = segg[5];
  else if(counter1 == 7) PORTA = segg[6];
  else if(counter1 == 8) PORTA = segg[7];
  else if(counter1 == 9) PORTA = segg[8];
  else if(counter1 == 10) {
    PORTA = segg[9];
  }
  counter1++;

} else if (!(PORTK & 0x02)) {
  PORTB = PORTB | (~0x01);
  //rotate right ?wrap around if end is reached;
  //Update counter displays;
   if (counter1 < 1) {
    counter1 = 10;
    counter2--;
    if (counter2 < 1) counter2 = 10;
   if (counter2 == 10) PORTB = segg[9];
    else if(counter2 == 9) PORTB = segg[8];
   else if(counter2 == 8) PORTB = segg[7];
   else if(counter2 == 7) PORTB = segg[6];
   else if(counter2 == 6) PORTB = segg[5];
   else if(counter2 == 5) PORTB = segg[4];
   else if(counter2 == 4) PORTB = segg[3];
   else if(counter2 == 3) PORTB = segg[2];
   else if(counter2 == 2) PORTB = segg[1];
   else if(counter2 == 1) PORTB = segg[0];
  }
  if (counter1 == 10) PORTA = segg[9];
  else if(counter1 == 9) PORTA = segg[8];
  else if(counter1 == 8) PORTA = segg[7];
  else if(counter1 == 7) PORTA = segg[6];
  else if(counter1 == 6) PORTA = segg[5];
  else if(counter1 == 5) PORTA = segg[4];
  else if(counter1 == 4) PORTA = segg[3];
  else if(counter1 == 3) PORTA = segg[2];
  else if(counter1 == 2) PORTA = segg[1];
  else if(counter1 == 1) PORTA = segg[0];
  counter1--;
```

```
  }
}

void main(void) {
  // config PORTA and PORTB for output
  DDRA = 0xFF;
  PORTA = 0x00;
  DDRB = 0xFF;
  PORTB = 0x00;
  // config PORTK for input
  DDRK = 0x00;
  PORTK = 0x03;
  // Enable IRQ interrupts;
  PORTE = 0x2; /* IRQ PIN PE1 PULL HIGH */
  INTCR = 0xC0; /* IRQ to be falling edge triggered */

  /* and enable IRQ Interrupt */
  EnableInterrupts;
  for (;;){}
}
```

6. Problems Encountered:

There were no major problems with this lab. Everything was pretty straight forward and we were able to complete the lab on time. The only hiccup we had was at the beginning when we were trying to add the 7 segment display to the board and were not able to find the right datasheet and hence we were connecting all to the wrong pins.

During coding, we were stuck on how to make the counter work correctly as we are not using any loops, only if/else, and kept forgetting the decimal point between the higher and lower counts.

8. Questions:

1) **Measure the time duration from the falling edge of the CLK signal to the change of the decimal point (from on to off, or vice versa) on a 7-segment display.**
   Ans) About 0.81ms per clock cycle
2) **Based on your experiment, estimate the highest frequency of the CLK signal under which the system can operate correctly**
   Ans) 12.3 Mhz
3) **For this lab, we use a falling-edge generated interrupt. If you use level-sensitive interrupts, what extra steps (and/or hardware) would you need? (don't write the entire code or schematic; answer in a brief manner)**

Ans) We would need to use an “A/D converter” as for the “Level Sensitive Interrupts”, the Microcontroller will keep updating the ISR as long as the pin input is low.

4) **What are the factors contributing to the interrupt latency? Answer briefly**
   Ans) The few factors which lead to interrupt latency is the clock speed, the type of interrupt controller used in this case the function generator, the frequency and the amplitude of the same.

9. Conclusion:

In this lab, we worked with Codewarrior in order to get started with concept of interrupt handling with the 9S12DG256 microcontroller and the system, and also gained experience in working with the 7 segment display and encoding a bi-directional two decade counter. It was fun learning new concepts about both, and astonishing to see so many uses of the same. For this Lab, Leya worked on the software design part of it and Tanish worked on the hardware part of it.