Experiment 2

Sevim Eftal Akşehirli, 150190028
Hacer Yeter Akıncı, 150200006
Büşra Özdemir, 150200036
Aslı Yel, 150200054

***Abstract*—In this experiment, MSP430 - The General Purpose Input Output is done by configuring the registers of ports of the MSP430G2553 microcontroller board as input/output. Also, the existence of an issue called dirty bouncing is introduced. This problem is resolved by waiting for a short period of time after each time the button is pressed. In Part 1, a specific bit of one of the ports is specified as a button, and a register is used to count how many times that button is pressed. In Part 2, two buttons are set to increase the values of two different variables when they are pressed (separately), and another button to reset these values when it is pressed. In Part 3, one more button is added and it triggers a multiplication operation on the variables whose values are set by pushing their corresponding buttons repeatedly as many times as their desired values.**

***Index Terms*—Input, port, push button, press, release, bit operation, conditional branch, status bits, dirty bounces, switch debouncing, count, multiply, debug, etc.**

I. Introduction

This experiment is planned to gain more knowledge about and experience in the MSP430 education board. In the meantime, it is aimed to reinforce the use of the MSP430G2553's ports and to observe the input value we receive with the help of the buttons on the board. Creating a "debounce" part in the code has been reinforced in order to prevent dirty bouncing that may occur in the physical environment when receiving input with the help of a button. It is aimed to understand the usages and differences of bit, bis, bic, and mov instructions in the instruction set. Checking the steps by examining the registers in Debug mode is one of the achievements of the experiment.

II. Materials and Methods

PART 1

In this part, we set the button as the least significant bit of the P2 register and calculate how many times the button is pressed. We also considered debouncing.

Listing 1. Assembly Code for Part1

```
Setup
        bis.b    #00000000b, &P2DIR
        bis.b    #00000000b, &P2IN
        mov.b    #00000000b, R6
        mov.w    #05000000, R15

MainLoop
        bit.b    #00000001b, &P2IN
        jne      Increment
        jmp      MainLoop

Increment
        inc      R6
        mov.w    #05000000, R15
        jmp      Debounce

Debounce
        dec.w    R15
        jnz      Debounce
        jmp      MainLoop
```

Setup: This section sets the P2 port as inputs, initializes R6 (the counter) with a value of 0 and sets the initial value of R15 to 05000000. R15 is used to provide switch debouncing.

Mainloop: This section checks whether the last bit of the P2 is pressed using bit.b. If it is pressed, the program proceeds with the Increment label. Otherwise, the program returns to the Mainloop.

Increment: It increments the counter (R6) by one, and R15 is set to a large value for debouncing. Then, the program jumps to the Debounce part.

Debounce: This part is a delay after the press. It decreases R15 until R15 is equal to zero. Then it returns to MainLoop.

PART 2

In this part, we set X as register R6 and Y as register R7. We assigned 3 buttons: he first button increases X, the second button increases Y, the last button is used to reset both.

Listing 2. Assembly Code for Part2

```
Setup
          bis.b     #00000000b, &P2DIR
          bis.b     #00000000b, &P2IN
          mov.b     #00000000b, R6 ;X
          mov.b     #00000000b, R7 ;Y
          mov.w     #05000000, R15

MainLoop
          bit.b     #10000000b, &P2IN
          jne       IncX
          bit.b     #01000000b, &P2IN
          jne       IncY
          bit.b     #00000001b, &P2IN
          jne       Reset
          jmp       MainLoop

IncX
          inc       R6
          mov.w     #05000000, R15
          jmp       Debounce

IncY
          inc       R7
          mov.w     #05000000, R15
          jmp       Debounce

Reset
          mov.b     #00000000b, R6
          mov.b     #00000000b, R7
          mov.w     #05000000, R15
          jmp       Debounce

Debounce
          dec.w     R15
          jnz       Debounce
          jmp       MainLoop
```

Setup: This section sets the P2 ports as input, R6 and R7 initial value as 00000000b, and R15 initial value as #05000000 for delay and provides switch debouncing.

Mainloop: This section checks whether P2 is pressed using bit operartion. If the first bit of P2 is pressed, it jumps to the label IncX and increments X. If the second bit of P2 is pressed, it jumps to the label IncY and increments Y. If the last bit of P2 is pressed, it jumps to the label Reset and X and Y are reset. Otherwise, the program returns to the Mainloop.

IncX: It increments the X (R6) by one, and R15 is set to a large value for debouncing. Then, the program jumps to the Debounce part.

IncY: It increments the Y(R7) by one, and R15 is set to a large value for debouncing. Then, the program jumps to the Debounce part.

Reset: It resets X(R6) and Y(R7), and R15 is set to a large value for debouncing. Then, the program jumps to the Debounce part.

PART 3

In this part, we calculated the multiplication of X and Y and assigned it to Z. For that we set Z to 0 at first and then added X to it Y times. To not change the original value of Y, R9 is a temporary register that keeps Y. After every addition operation of X, R9 will be decremented by one. When R9 is equal to 0, Y times X will be added and it is equal to the multiplication of X and Y.

Listing 3. Assembly Code for Part3

```
Setup
          bis.b     #00000000b, &P2DIR
          bis.b     #00000000b, &P2IN
          mov.b     #00000000b, R6   ; X
          mov.b     #00000000b, R7   ; Y
          mov.b     #00000000b, R8   ; Z
          mov.b     #00000000b, R9   ; temp
          mov.w     #05000000, R15

MainLoop
          mov.b     R7, R9
          bit.b     #10000000b, &P2IN
          jne       IncX
          bit.b     #01000000b, &P2IN
          jne       IncY
          bit.b     #00000001b, &P2IN
          jne       Reset
          bit.b     #00000010b, &P2IN
          jne       Multiply
          jmp       MainLoop

IncX
          inc       R6
          mov.w     #05000000, R15
          jmp       Debounce

IncY
          inc       R7
          mov.w     #05000000, R15
          jmp       Debounce

Reset
          mov.b     #00000000b, R6
          mov.b     #00000000b, R7
          mov.w     #05000000, R15
          jmp       Debounce

Multiply
          add       R6, R8
          dec       R9
          cmp       #0d, R9
          jeq       Setup
          jmp       Multiply

Debounce
          dec.w     R15
          jnz       Debounce
          jmp       MainLoop
```

Setup: This section sets the P2 port as input. It initializes R6 as X, R7 as Y, R8 as Z (X*Y) and R9 as 0. R9 is an iterator, and will be used in multiply section. R15 is set the initial value to 05000000 for providing switch debouncing

MainLoop: In this section, R9 is set as Y. X and Y could be increased and reset by buttons as we implemented in the part 2. X and Y could be multiplied with an additional button.

IncX and IncY: It increments the registers(X and Y) by one, and R15 is set to a large value for debouncing. Then, the program jumps to the Debounce part.

Reset: It resets X (R6) and Y (R7), and R15 is set to a large value for debouncing. Then, the program jumps to the Debounce part.

Multiply: X is added to Z and R9 is decreased by one. The addition operation is repeated until R9 reaches 0.

Debounce: This part is a delay after the press. It decreases R15 until R15 is equal to zero. Then it returns to MainLoop.

III. Results

For the first part, we created a counter register that counts how many times the assigned button is pressed. We tested our code several times. It works correctly, just as expected. For the second part, we created two registers (X and Y) that are increased each time the assigned button is pressed and we assigned one button to reset the registers. We tested our code several times. It work correctly, just as expected. For the third part, we created two registers (X and Y) that are increased each time the assigned button is pressed, one register to keep the value of multiplication of X and Y and one register as temporary to keep the value of Y. We assigned one button to reset the registers and one button to make multiplication. We tested our code several times. It works correctly, just as expected. For all parts of the experiment, in order to check whether it works correctly or not, we run the code in debug mode, put some breakpoints at meaningful lines, and observed the relevant registers at each step of debugging session.

IV. Discussion and Summary

In the first part, we wanted to use Port 1, but for some reason some bits appeared as 1 although we set it to 0 initially, and the code did not work as we wanted. We solved this problem by setting Port 2 as input and continued the other parts with P2.

All 3 parts were interconnected. After completing the first part, we did not have much difficulty with the others.

There is the same redundancy in our codes for all three parts of the experiment: the last line of Setup label which initializes R15. This line has no effect and is not necessary since we already have this line in labels Increment, IncX, IncY, Reset, Multiply, R15 will set again whenever is is needed.

During the experiment we had some problems using CC-Studio, but we handled them afterwards.