ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 351E

MICROCOMPUTER LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	1
EXPERIMENT DATE	:	02.10.2019
LAB SESSION	:	WEDNESDAY - 13.30
GROUP NO	:	G10

GROUP MEMBERS:

150170062	:	Mehmet Fatih YILDIRIM
150180704	:	Cihat AKKİRAZ
150180705	:	Batuhan Faik DERİNBAY
150180707	:	Fatih ALTINPINAR

FALL 2019-2020

Contents

1 INTRODUCTION

One of the microcontrollers from Texas Instruments family is MSP430 which is a mixed signal microcontroller. The MSP430 is designed for low cost and specifically low power consumption. This is the first of the experiments in this period, a simple introduction to the MSP430 was made. In this experiment, MSP430 microcontroller is programmed in order to achieve LED on-off patterns on the experiment booklet.

2 MATERIALS AND METHODS

This experiment is completed by using a MSP430G2553 microprocessor. This microprocessor is programmed by using Code Composer Studio for the desired tasks on the experiment handout. During coding several sources are used:

- MSP430 Education Board Manual [1]
- MSP430 Architecture Chapter 4 [2]
- MSP430 Instruction Set [3]

2.1 Part 1

In the first part of the experiment, a new CCS Project is created as choosing "MSP430G2553" as the target and "Empty Assembly Project" as the project template. Then, code given on the experiment booklet [4](see Figure 2.1) is put to the main.asm file. For better understanding, further examination of the code, line by line if necessary is required:

- Line 1: P1.0 is enabled for future use.
- Line 2: P1.0 is complemented via using XOR instruction to flash.
- Line 3 - 5: This is the wait function. The value 250000_{10} is moved to register 15 and decreased on the 4th line. Taking the 5th line into hand, until the zero flag is raised the program jumps back to 4th line, decreasing the value of R15. When the value in R15 hits 0_{10}, the zero flag is raised causing line 5 to skip execution and continuum of the program.
- Line 6: In order to flash the LED the program jumps to main loop(Seen in 2)

```
SetupP1      bis.b    #001h,   &P1DIR
Mainloop     xor.b    #001h,   &P1OUT
Wait         mov.w    #250000,R15
L1           dec.w    R15
             jnz L1
             jmp Mainloop

```

Figure 1: Assembly Code of Part 1

2.2 Part 2

In the second part of the experiment, the microprocessor was programmed to flash the LEDs as in the given pattern(see Figure 2).

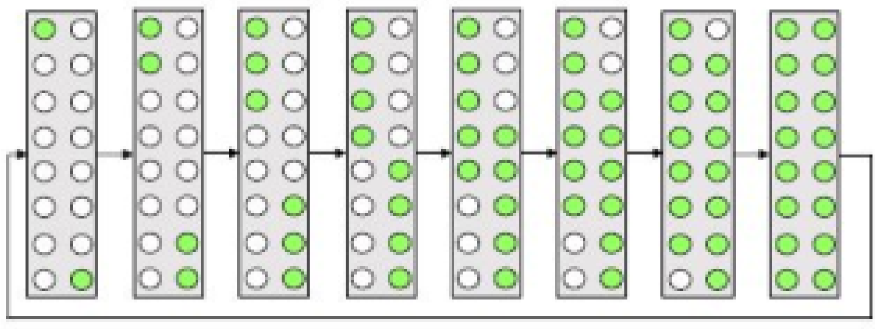

Figure 2: Part 2 LEDs On - Off Pattern

In order to implement the given pattern, team created following piece of code given in Figure 2.2

```
 1  SetupP2         mov.b    #0FFh,   &P1DIR
 2                  mov.b    #0FFh,   &P2DIR
 3  Reset           mov.b    #01h,    &P1OUT
 4                  mov.b    #80h,    &P2OUT
 5                  jmp      Wait
 6
 7  Mainloop        rla.b    &P1OUT
 8                  add.b    #01h,    &P1OUT
 9                  rra.b    &P2OUT
10  Wait            mov.w    #250000,R15
11  L1              dec.w    R15
12                  jnz      L1
13                  cmp.b    #0FFh,   P2OUT
14                  jz       Reset
15                  jmp      Mainloop
16
17
```

Figure 3: Assembly Code of Part 2

For better understanding, further examination of the code, line by line if necessary, is required:

- Line 1: The first column of LEDs are set up. Notice that all LEDs in the first column will be needed, as clearly seen at the last state in Figure 2.2. The binary representation for this case would be 11111111_2. Therefore, in this line, the value 11111111_2 $(0xFF)$ is moved to the absolute address $P1DIR$ in order to activate all LEDs in the first column.

- Line 2: For the exact same reasons, namely, because of the need for all LEDs to light up, in this line, the value 11111111_2 $(0xFF)$ is moved to the absolute address $P2DIR$ in order to activate all LEDs in the second column.

- Line 3-5: This is the Reset function that will later be used to reach the first state in Figure 2.2. In line 3, the value 00000001_2 $(0x01)$ is moved to the absolute address $P1OUT$ in order to light the first LED in the first column. Similarly, in line 4, the value 10000000_2 $(0x80)$ is moved to the absolute address $P2OUT$ in order to light the last LED in the second column. Note that the least significant bit (LSB) is located at the top of the column and the most significant bit (MSB) is located at the bottom of the column. In line 5, the program jumps to the Wait function.

- Line 7-9: This is the Mainloop that will be executed in order to achieve the next state. In line 7, the value at the address $P1OUT$ is arithmetically shifted to left in order to light the next LED. In line 8, this value in $P1OUT$ is incremented by 1 in order to light the first LED in first column again which went of in line 7. In line 9, the value at the address $P2OUT$ is arithmetically shifted to right in order to light the next LED. Since "arithmetically" shifted, the leftmost digit corresponding to the last LED in second column will automatically filled with 1 which was the value here before shifting. Therefore, an operation like the one in line 8 is not needed.

- Line 10-12: This is the Wait function that will be executed in order to provide the delay between each 2 consecutive states. In line 10, the value 250000_{10} is moved to register 15 and in line 11, it is decreased. In line 12, until the zero flag is raised the program jumps back to 11th line to decrease the value in R15. When the value in R15 hits 0_{10}, the zero flag is raised and the program skips to the next line.

- Line 13-15: In line 13, the value at $P2OUT$ is compared with 11111111_2 $(0xFF)$ in order to know whether all LEDs in second column are lit which means the last state is reached. If they are equal, the zero flag is raised. In line 14, if it is raised, the program jumps to Reset function to reach the first state again. In line 15, the program jumps to Mainloop to achieve the next state because if this line is reached, it is known that the last state is not reached yet.

2.3 Part 3

In the part 3 of the first experiment the LED's were lit in a descending order, changing the column every two steps as shown in Figure 4.

In order to achieve the given sequence the team decided on the following piece of code given in Figure 5. For better understanding, further examination of the code, line by line if necessary, is required:

- Line 1: The first of line of LEDs are set up. Notice the pattern that starting with two LEDs, every second doublet of LEDs will be lit. When sketched, "O" denoting LEDs that are needed to be lit and "X" that don't need to be lit, the pattern can be expressed as "$XXOOXXOO$" for the first line of LEDs. Hence the binary representation of such expression will be "00110011_2". Note that the least significant bit (LSB) is located at the top of the column and the most significant bit (MSB) is located at the bottom of the column.
  Therefore in this line the value 00110011_2 ($0x33$) is moved to the absolute address $P1DIR$ in order to activate aforementioned LEDs.

- Line 2:Since the LEDs that will be lit in the second column follow the "$OOXXOOXX$", 11001100_2 ($0xCC$) value is moved the absolute address $P2DIR$ in the second line.

- Line 3: This is the reset function. 00000001_2 ($0x01$) is stored in register 14 in order to set the initial state of the LEDs.

- Line 4-5: This is the beginning of the main loop. The value in R14 is moved the absolute addresses of LEDs in order to light them. Notice that every value of R14 is moved to both LED columns but each LED will light up according to the "SetupP3" when certain values are present.

- Line 6:The value in the register 14 is arithmetically shifted to left in order to light the next LED. For a better representation binary expressions can be checked up on. For example the very first value stored in R14 (00000001_2) results in (00000010_2) after an arithmetic shift left.

- Line 7-9: This is the wait function. The value 250000_{10} is moved to register 15 and decreased on the 8th line. Taking the 9th line into hand, until the zero flag is raised the program jumps back to 8th line, decreasing the value of R15. When the value in R15 hits 0_{10}, the zero flag is raised causing line 9 to skip execution and continuum of the program.

- Line 10-12: 10th line checks whether the 8bit value in register 14 is 0_{10} ($0x00$) or not. If that's the case, the zero flag is raised and the program jumps to "Reset" function as depicted on line 11. If the zero flag is not raised this means that not all 8 rows of LEDs were lit, hence the program jumps to main loop (Seen in line 12) in order to light up the next LED in the sequence.

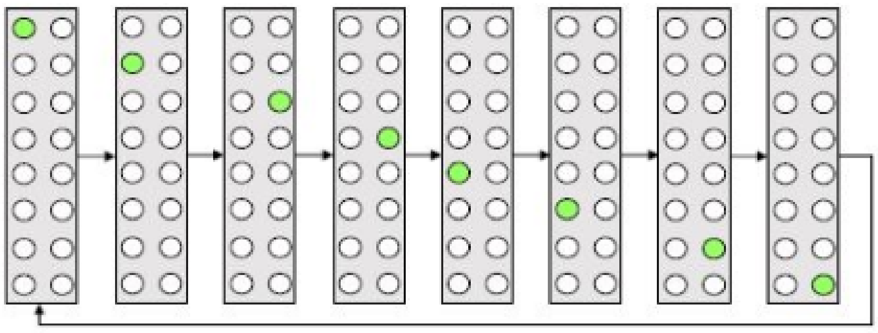

Figure 4: Part 3 LEDs On - Off Pattern

```
1   SetupP3        mov.b   #033h,   &P1DIR
2                  mov.b   #0CCh,   &P2DIR
3   Reset          mov.b   #01h,    R14
4   Mainloop       mov.b   R14,     &P1OUT
5                  mov.b   R14,     &P2OUT
6                  rla.b   R14
7   Wait           mov.w   #250000,R15
8   L1             dec.w   R15
9                  jnz     L1
10                 cmp.b   #000h,   R14
11                 jz      Reset
12                 jmp     Mainloop
```

Figure 5: Assembly Code of Part 3

2.4 Part 4

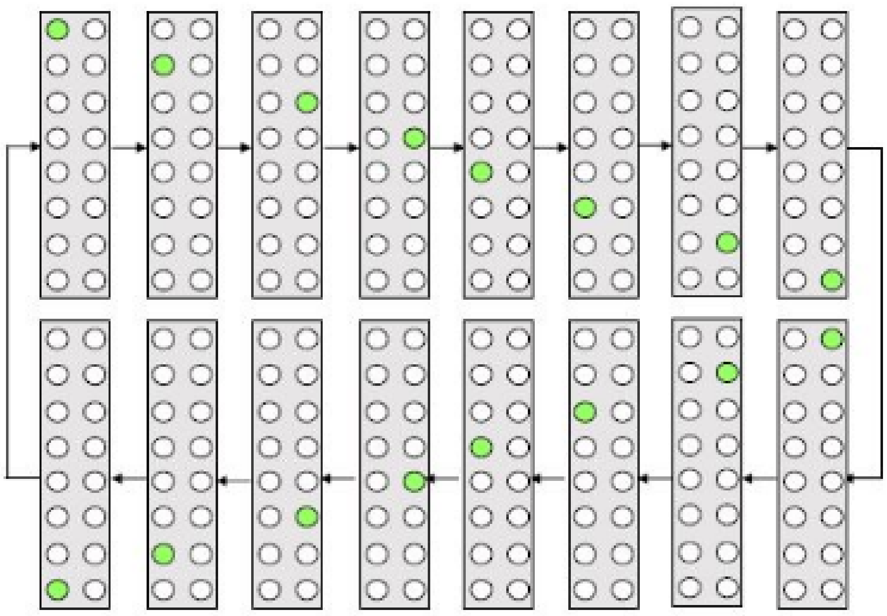

Figure 6: Part 4 LEDs On - Off Pattern

In the final part of the experiment, a modification is required to be applied on Part 3's code in order to achieve the sequence in Part 4.

As can be seen in Figure 6, in the second part of the sequence only the LEDs that are on the opposite side will light up. To achieve this, the values of $P1DIR$ and $P2DIR$ have to be swapped. This can be done by adding lines from 4 to 7 to the code and changing line 16. Rather than jumping back to Reset like in Figure 5, jumping to SetupB is required to initiate the second part of the sequence.

Line by line explanation of the SetupF sequence is required in order to have a better understanding:

- Line 4-5: Since first and second parts of the sequence given in Figure 6 should cycle over time, the code should be returning to first part of the sequence after completing the second. This is achieved by these lines. $P2DIR$ is checked, if its equal to 00110011_2 which means that it is in the second part of the sequence, then zero flag will be up. Line 5, will initiate the SetupF.

- Line 6-5; Since LEDs have to lit on the opposite way, values of $P1DIR$ and $P2DIR$ are swapped.

```
SetupF          mov.b   #033h,  &P1DIR
                        mov.b   #0CCh,  &P2DIR
                        jmp     Reset
SetupB          cmp.b   #033h,  &P2DIR
                        jz              SetupF
                        mov.b   #0CCh,  &P1DIR
                        mov.b   #033h,  &P2DIR
Reset           mov.b   #01h,   R14
Main            mov.b   R14,    &P1OUT
                        mov.b   R14,    &P2OUT
                        rla.b   R14
Wait            mov.w   #250000,R15
L1                      dec.w   R15
                        jnz     L1
                        cmp.b   #000h,  R14
                        jz              SetupB
                        jmp     Main
```

Figure 7: Assembly Code of Part 4

3 RESULTS

There wasn't any problem or any unexpected situations. Results were observed the way the team has predicted them. Desired LEDs on - off patterns and the codes to implement these patterns are given in the previous section.

4 DISCUSSION

Preparing the code before going to the lab session speeds up the team's work. In the lab session, the code was debugged to correct minor errors. There was no contradiction between the theoretical knowledge of the team and the results of the experiment.
Please refer to section 2 "MATERIALS AND METHODS" for exclusively detailed information, tables, images, analysis, interpretation and results, covering all the required material under other sections.

5 CONCLUSION

Team didn't face any particular difficulties that were worth noting throughout the experiment. Team was able to complete the given tasks quite swiftly. All in all, this experiment helped team members to improve their abilities to program MSP430 microprocessor for given conditions situations.

REFERENCES

[1] Texas Instruments. Msp430 education board document. 2009.

[2] Texas Instruments. Msp430 architecture. 2009.

[3] Texas Instruments. Msp430 architecture. December 2004 - Revised July 2013.

[4] Istanbul Technical University Department of Computer Engineering. Blg 351e wednesday experiment 1: Basic assembly coding, Spring 2019.

[5] Overleaf documentation https://tr.overleaf.com/learn.

[6] Detailed info on writing reports https://projects.ncsu.edu/labwrite/res/res-studntintro-labparts.html.