Experiment 7

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

***Abstract*—This experiment focuses on understanding and utilizing the 16x2 dot matrix LCD by implementing various programs to display dynamic strings. The initial section of the experiment involved configuring the LCD using specific initialization settings, laying the foundation for subsequent programming tasks. Through this experimentation, different programs were developed to dynamically display strings on the LCD, demonstrating practical applications of interfacing with the 16x2 dot matrix LCD. The report details the initialization process and presents the implementation of diverse programs aimed at showcasing the versatility and functionality of the LCD in real-time applications.**

***Index Terms*—LCD, Set of LCD Instruction, Cursor, Display, Nibble**

I. Introduction

In this experiment, we delve into the practical application of 16x2 dot matrix LCDs, vital components in contemporary electronic displays. Our focus is on programming techniques that leverage these LCDs effectively. Beginning with a provided initialization setup in Part 0, this report explores dynamic string displays. Through a series of programming exercises, we aim to grasp various methods for showcasing changing text on the LCD screen. This experiment offers hands-on learning, aiming to equip individuals with the skills needed to manipulate and present dynamic content on a 16x2 dot matrix LCD

II. Materials and Methods

A. Part-1

In the first part of the experiment, we analyzed the display.asm code to understand the initialization and configuration process of the LCD, and then we changed the code and added: ”ITU Comp Eng MicroLab. We tried printing 2022”.

```
Main      clr R5
          mov.b #00001111b, R5
          call #SendCMD
          mov &Delay50us, R15
          call #Delay


          mov #string, R7

Loop      clr R6
          mov.b @R7+, R6
          cmp.b #0Dh, R6
          jeq YeniSatir
          cmp.b #00h, R6
          jeq Fin
          call #SendData
          mov &Delay50us, R15
          call #Delay
          jmp Loop

YeniSatir   clr R5
          mov.b #11000000b, R5
          call #SendCMD
          mov &Delay50us, R15
          call #Delay
          jmp Loop


Fin       jmp Fin


TrigEn    bis.b #01000000b, &P2OUT
          bic.b #01000000b, &P2OUT
          ret

SendCMD   mov.b R5, &P1OUT
          call #TrigEn
          rla R5
          rla R5
          rla R5
          rla R5
          mov.b R5, &P1OUT
          call #TrigEn
          ret

SendData  bis.b #10000000b, &P2OUT
          mov.b R6, &P1OUT
          call #TrigEn
          rla R6
          rla R6
          rla R6
          rla R6
          mov.b R6, &P1OUT
          call #TrigEn
          bic.b #10000000b, &P2OUT
          ret

Delay     dec.w R15
          jnz Delay
          ret
```

Main: Within the function, we prepared R5 to send data to P1OUT and assigned the specified delay time to R15 to use the delay function as we wished. We sent the string value that we will print on the screen to R7.

Loop: In this function, we aimed to throw the string value that will be our output to R6. The code loads the string value we put into R7 in the Main function into R6 as a character. Meanwhile, if it reaches 0Dh, which is the character we need to move to the bottom line, it jumps to the YeniSatir function. If it reaches the 00h character, which indicates that we have loaded the entire text, it finishes the code by entering the Fin function. At the end of each character, we go to the SendData function and load R6 into P1OUT, and with this function, we access all characters.

YeniSatir: With this function, we update R5 to move to the bottom row and manipulate P1OUT by calling the SendCMD function. Afterward, we return to the loop again to read the remaining characters.

Fin: We ensure the end of the code by putting it in an endless loop.

TrigEn: By manipulating P2OUT with this function, we create the necessary signal for the LCD to operate following the desired part of the experiment.

SendCMD: Since the LCD, which normally works with 8-bit by default, is connected to the MSP430 only with the upper nibble connection, we make a mov instruction to P1OUT twice with R5. Using the Rla method, we shift the number to the left and in this way we also update the MSB bits of P1OUT. Meanwhile, we update P2OUT twice to use the LCD as we want by using the TrigEn function.

SendData: Since the LCD, which works as in SENDCmd, is connected to the MSP430 only with the upper nibble connection, we make the mov instruction to P1OUT twice with R6, which holds the R6 value that holds our character. Using the Rla instruction, we update the MSB bits of P1OUT and send our character. Meanwhile, we manipulate P2OUT with bis, bic instructions and TrigEn.

Delay: We create the time we need to obtain by decreasing the value we loaded into R15 in the main function with the dec.w instruction.

B. Part-2

In the second part of the experiment, we printed “ITU Comp Eng” on the top line, and “MicroLab” printed on the bottom line. The code was modified to replace the 2023” strings with the 2hz frequency. To swap the two lines as we wish, we assigned a new register and created “MicroLab. We managed which line 2023” would be on and changed “ITU Comp Eng” to be written on the other line.

```
MainLoop
          clr R5
          mov.b #00001111b, R5
          call #SendCMD
          mov &Delay50us, R15
          call #Delay
          mov.b #00000001b, R5
        call #SendCMD
        mov &Delay2ms, R15
        call #Delay

        cmp #0b, R8
        jeq FirstSatir
        mov.b #11000000b, R5
        call #SendCMD
        mov &Delay50us, R15
        call #Delay
        mov #string, R7
        jmp Loop

FirstSatir     clr R5
        mov.b #10000000b, R5
        call #SendCMD
        mov &Delay50us, R15
        call #Delay
        mov #0, R8
        mov #string, R7

Loop
          clr R6
          mov.b @R7+, R6
          cmp.b #0Dh, R6
          jeq compare
          cmp.b #00h, R6
          jeq Fin
          call #SendData
          mov &Delay50us, R15
          call #Delay
          jmp Loop

compare   cmp #0b, R8
        jeq AltSatir
        jmp UstSatir

AltSatir     clr R5
      mov.b #11000000b, R5
      call #SendCMD
      mov &Delay50us, R15
      call #Delay
      mov #1b, R8
      jmp Loop

UstSatir        clr R5
      mov.b #10000000b, R5
      call #SendCMD
```

```
        mov &Delay50us, R15
        call #Delay
        mov #0, R8
        jmp Loop

Fin         mov &Delay100ms, R15
        call #Delay
        mov &Delay100ms, R15
        call #Delay
        mov &Delay100ms, R15
        call #Delay
        mov &Delay100ms, R15
        call #Delay
        mov &Delay100ms, R15
        call #Delay
        jmp MainLoop



TrigEn      bis.b #01000000b, &P2OUT
        bic.b #01000000b, &P2OUT
        ret

SendCMD     mov.b R5, &P1OUT
        call #TrigEn
        rla R5
        rla R5
        rla R5
        rla R5
        mov.b R5, &P1OUT
        call #TrigEn
        ret

SendData    bis.b #10000000b, &P2OUT
        mov.b R6, &P1OUT
        call #TrigEn
        rla R6
        rla R6
        rla R6
        rla R6
        mov.b R6, &P1OUT
        call #TrigEn
        bic.b #10000000b, &P2OUT
        ret

Delay       dec.w R15
        jnz Delay
        ret
```

MainLoop: We made additions to the Main function we wrote in Part 1. We aimed to control which line the character sequences we need to print will be on by using the R8 register. When the code ran for the first time, we ensured that R8 was set to 0 and entered the FirstSatir function. While the code sends the string to R7 in the FirstLine function in the first display, it does this in the MainLoop function after the first display and passes to the Loop function.

FirstSatir: We created this function to use in the first display of the code. By updating R5, we enabled the LCD to write to the first line, and we completed the function by updating R8 to 0 again.

Loop: We made some changes to the loop function in Part 1 for this part of the experiment. While loading, in the same way, using R6 and the SendData function, we aimed to check which line the string would be written to by jumping to the compare function when we read the first part of the string, that is, the part up to the 0Dh character.

compare: We intended to process the value in the R8 register in other functions. In this function, we aimed to direct the string to the AltSatir or UstSatir function according to the value of R8.

AltSatir: In this function, we aimed to update R5 so that the LCDs the string on the bottom line. By updating R8, we aimed for the code to enter the UstSatir Function when it returns to the Loop function and the compare function.

UstSatir: The function we prepared with the same working logic as AltSatir aims to update R5 so that the LCD prints the string on the top line. Again, by updating R8, the aim was to enter the code into the AltSatir function in the next compare function.

Fin: We modified the delay time with R15 to achieve the desired 2 Hz frequency in the Fin function. For the code to be in constant repetition, add the ”jmp MainLoop” line, which is the transition line to the MainLoop.

TrigEn: No changes were made to the function, it was used as it was in the first part.

SendCMD: No changes were made to the function, it was used as it was in the first part.

SendData: No changes were made to the function, it was used as it was in the first part.

Delay: No changes were made to the function, it was used as it was in the first part.

C. Part-3

In the third part of the experiment, we aimed to display the pattern given before the experiment on the LCD in the same way. In this part, where we understood and implemented the LCD instruction in more detail, we obtained the pattern by arranging the Cursor/Display shift instruction as cursor-off.

```
MainLoop
            clr R9
            clr R8
               clr R5
            mov.b #00001100b, R5
            call #SendCMD
            mov &Delay50us, R15
            call #Delay
            mov.b #00000010b, R5
            call #SendCMD
            mov &Delay50us, R15


            mov #string, R7

Loop
            clr R6
            mov.b #00000001b, R5
          mov &Delay2ms, R15
            mov.b @R7, R6
            mov.b R9, R10
            call #shift
            call #SendData
            mov &Delay100ms, R15
            call #Delay
            mov &Delay100ms, R15
            call #Delay
            mov &Delay100ms, R15
            call #Delay
            mov &Delay100ms, R15
            call #Delay
            mov &Delay100ms, R15
            call #Delay

            inc.b R9
          mov.b #00000001b, R5
          call #SendCMD
          mov &Delay100ms, R15
          call #Delay
          mov &Delay100ms, R15
          call #Delay
            cmp #16d, R9
            jeq Fin
            cmp #0b, R8
            jeq YeniSatir
            jmp Satir

shift     cmp #0d, R10
        jeq return
        mov #00010100b, R5
        call #SendCMD
        mov &Delay50us, R15
        call #Delay
        dec R10
        jmp shift

return      ret

YeniSatir     clr R5
        mov.b #11000000b, R5
        call #SendCMD
        mov &Delay50us, R15
        call #Delay
        mov.b #1b, R8
        jmp Loop

Satir       clr R5
        mov.b #10000000b, R5
        call #SendCMD
        mov &Delay50us, R15
        call #Delay
        ;mov  #00010100b, R5
        ;call #SendCMD
        ;mov &Delay50us, R15
        ;call #Delay
        mov.b #0b,  R8
        jmp Loop

Fin       jmp MainLoop




TrigEn      bis.b #01000000b, &P2OUT
        bic.b #01000000b, &P2OUT
        ret

SendCMD     mov.b R5, &P1OUT
        call #TrigEn
        rla R5
        rla R5
        rla R5
        rla R5
        mov.b R5, &P1OUT
        call #TrigEn
        ret

SendData    bis.b #10000000b, &P2OUT
        mov.b R6, &P1OUT
        call #TrigEn
        rla R6
        rla R6
        rla R6
        rla R6
        mov.b R6, &P1OUT
        call #TrigEn
        bic.b #10000000b, &P2OUT
        ret

Delay     dec.w R15
        jnz Delay
        ret
```

MainLoop: In this part, we aimed to reset R8 and R9 on each display when the code comes to the MainLoop again. We sent the necessary binary value to manipulate R5 and P1OUT in line with the instructions we wanted. With this binary value, we provided both shift and cursor-off.

Loop: This time, we used R9 in the loop to control the printing process * to the pattern 16 times. We loaded R10 with R9 to send the shift operation instruction to the LCD. When the ”*” number of the code reached 16, we made it start again by going to the fin function. As in the second part, we checked which line we would print on with R8.

shift: It is used to shift the cursor forward, it counts the stars and shifts the cursor accordingly.

return: We used the return function to return from a subroutine.

YeniSatir: We updated R5 for the line change. We updated R8 so that the function moves to the next line in the next operation.

Satir: We updated R5 for the line change. We updated R8 so that the function moves to the next line in the next operation.

Fin: When we reached 16 ”*” in the desired pattern, the code was edited to write the pattern again in the same way from the MainLoop.

TrigEn: No changes were made to the function, it was used as it was in the first part.

SendCMD: No changes were made to the function, it was used as it was in the first part.

SendData: No changes were made to the function, it was used as it was in the first part.

Delay: No changes were made to the function, it was used as it was in the first part.

III. Results

First of all, we analyzed the given code and investigated how the LCD works, what modes the LCD has, and how we should use them. We examined the given code and updated part 1 to give the desired output. Since we were asked to make an alternating pattern between two sators for Part 2, we analyzed the problem mathematically. We created a loop to support the pattern and set the change time of the text.3. In the section, we found the equivalent of the “*” character that we need to print in binary format. We examined the LCD’s instruction set and adjusted the value in the most accurate and useful way. Finally, we completed the project by turning off the cursor mode.

IV. Discussion and Summary

During this experiment, we experienced some difficulties because we were working on an LCD with which we had no previous experience. In the first part of the experiment, we ran the code provided to us as desired. In the second part, we tried to create a regular pattern for strings to change lines. We could only achieve this by defining a different function when the code first made the display, and then we obtained the output we wanted by checking the register value we assigned with the values 1-0. In the third part of the experiment, we had to look at the LCD instructions in more detail and it took a long time to find the mode we needed. We had trouble with the shift operation. When the code reached the pattern we wanted, we turned off the cursor mode, did not display the cursor on the display, and completed the experiment. We completed all three parts successfully and gained experience in LCD control.