SANTA CLARA UNIVERSITY
Electrical and Computer Engineering Department

ELEN 120 – Embedded Computing Systems

Lab 8 – Matrix I/O

Dylan Thornburg | Sal Martinez | Gigi Patmore

Assignment: In this assignment you will program the segmented LCD display on the DISCO board. Most of the code has been provided – but you must finish the project.

Prelab:

Read Chapters 14.9 and 17 in the book. It is not critical to understand everything but be familiar with what is taught there.

Review the register specifications and pre-supplied code discussed in the lab below and the class presentations related to these topics.

Review this handout and develop a plan for writing and testing your

code.

Problem 1

Your project is to:

1. display the text phrase “SANTA” in positions 1-5 on the LCD display
2. wait 1 second
3. display the text phrase “CLARA” in positions 2-6 on the LCD display
4. wait 1 second
5. Repeat steps 1-4 indefinitely

The illustration on the next page shows the two displays.
In the Lab Handouts\Lab 8 files\ directory on the Google Drive, I have provided a zip file of a project called LCD (redacted). Some critical code is missing.

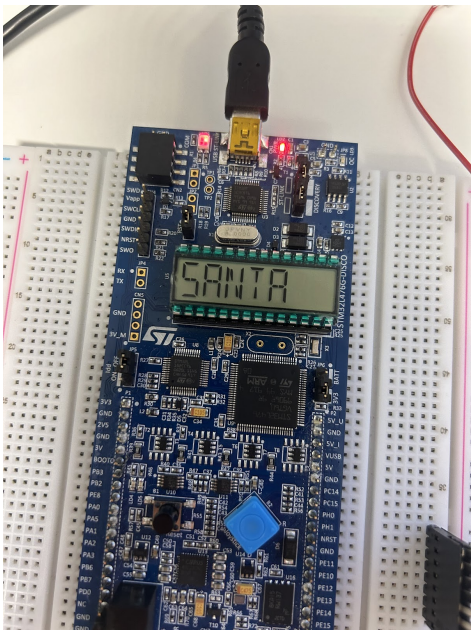
I provide you with a subroutine called lcd_init. This configures and initializes the LCD display.
I provide you with a subroutine called lcd_clear. This clears the LCD display, any time after it has been configured.
I provide you with a subroutine called lcd_draw. It can be used to display a character on the LCD display.

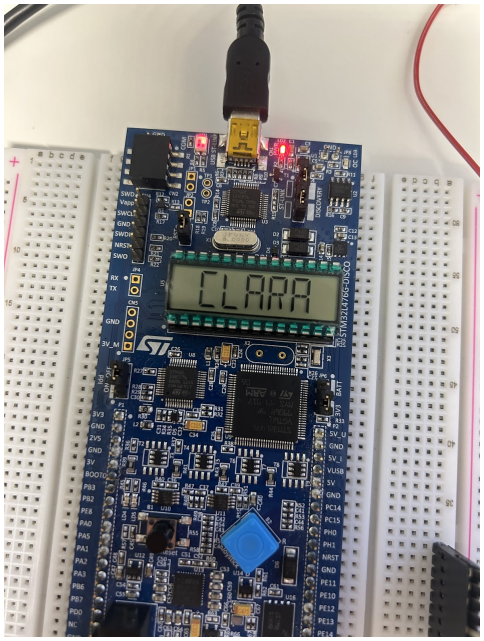
```
;Draws a character into the LCD Frame Buffer based on Zhu's character font encoding
;r0 holds the font encoding (lower 16 bits) Segments G B M E F A C D Q K Col P H J DP
N ;r1 holds the character position (1-6)
```

A subroutine called let2font is supposed to convert an ASCII letter character to its font representation, usable by lcd_draw. You need to complete the code for this routine.

```
; r0 is an ascii letter a-z (0x41-0x5A or 0x61-7A)
; return font in r0
; convert lower to upper - return 0 for out of range
```

Once you have completed and tested let2font, you can add additional code to main.s to accomplish the project requirements.





Demo your project; record a video for your report; turn in your code for let2font and main.s.

MAIN.S:

```
;****************** (C) Andrew Wolfe ************************************
; @file    lcd-r.s
; @author  Andrew Wolfe
; @date    Nov. 19, 2019
; @note
;         This code is for the book "Embedded Systems with ARM Cortex-M
;         Microcontrollers in Assembly Language and C, Yifeng Zhu,
;         ISBN-13: 978-0982692639, ISBN-10: 0982692633 as used at Santa Clara University
;*******************************************************************************
*

        INCLUDE core_cm4_constants.s            ; Load Constant Definitions
```

```
        INCLUDE stm32l476xx_constants.s

        INCLUDE lcd.h


                AREA    main, CODE, READONLY
                EXPORT          __main
                ENTRY




__mainPROC

                bl              lcd_init
endlessbl               lcd_clear

;*************************************

                mov r0, #83;S
                bl let2font
                mov r1, #1
                bl lcd_draw
                mov r0, #65;A
                bl let2font
                mov r1, #2
                bl lcd_draw
                mov r0, #78;N
                bl let2font
                mov r1, #3
                bl lcd_draw
                mov r0, #84;T
                bl let2font
                mov r1, #4
                bl lcd_draw
                mov r0, #65;A
                bl let2font
                mov r1, #5
                bl lcd_draw
```

```
                mov r4, #0x130000;insert some delay. Can't test yet
delay0  subs r4, #1
                bne delay0

                bl              lcd_clear
                mov r0, #67;C
                bl let2font
                mov r1, #2
                bl lcd_draw
                mov r0, #76;L
                bl let2font
                mov r1, #3
                bl lcd_draw
                mov r0, #65;A
                bl let2font
                mov r1, #4
                bl lcd_draw
                mov r0, #82;R
                bl let2font
                mov r1, #5
                bl lcd_draw
                mov r0, #65;A
                bl let2font
                mov r1, #6
                bl lcd_draw

                mov r4, #0x130000;insert some delay. Can't test yet
delay1  subs r4, #1
                bne delay1
;*************************************

                b               endless
                ENDP



                        ALIGN
                        AREA    myData, DATA, READWRITE

                        ALIGN


        END
```

LET2FONT:

```
let2font        PROC
```

```
                    EXPORT let2font
                    ;       r0 is an ascii letter a-z (0x41-0x5A or 0x61-7A)
                    ;       return font in r0
                    ;       convert lower to upper - return 0 for out of range

;*****************************************
                    cmp r0, #97; 'a'
                    subge r0, #32
                    cmp r0, #65; 'A'
                    movlt r0, #0
                    blt done
                    cmp r0, #90; 'Z'
                    movgt r0, #0
                    bgt done
                    sub r0, #65
                    lsl r0, #1
                    ldr r1, =letfont
                    add r1, r0
                    ldrh r0, [r1]
done        bx lr
;*****************************************

                    ENDP
                    ALIGN
```

Problem 2

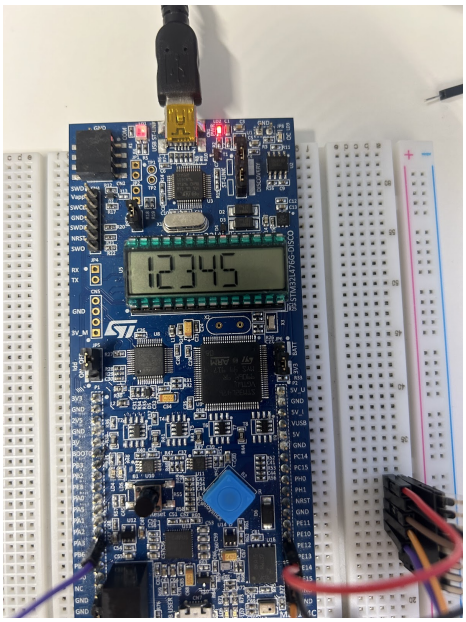
Your project is to:

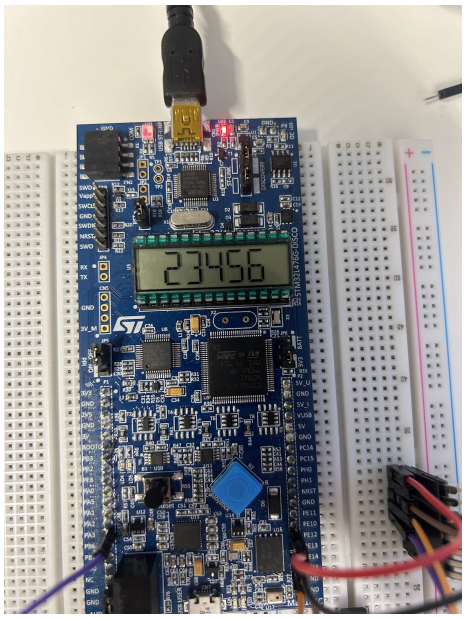
1. display the number “12345” in positions 1-5 on the LCD display
2. wait 1 second
3. display the number “23456” in positions 2-6 on the LCD display
4. wait 1 second
5. Repeat steps 1-4 indefinitely

A subroutine called num2font is supposed to convert an ASCII number character to its font representation, usable by lcd_draw. You need to complete the code for this routine.

```
; r0 is an ascii number 0-9 (0x30-0x39)
; return font in r0
; Only use last hex digit 0-9; zero out A-F
```

Once you have completed and tested num2font, you can add additional code to main.s to accomplish the project requirements.





MAIN.S

Demo your project; record a video for your report; turn in your code for num2font and main.s.

```
;******************** (C) Andrew Wolfe ***************************************
; @file    lcd-r.s
; @author  Andrew Wolfe
; @date    Nov. 19, 2019
; @note
;          This code is for the book "Embedded Systems with ARM Cortex-M
;          Microcontrollers in Assembly Language and C, Yifeng Zhu,
;          ISBN-13: 978-0982692639, ISBN-10: 0982692633 as used at Santa Clara University
;*****************************************************************************
*



        INCLUDE core_cm4_constants.s            ; Load Constant Definitions
        INCLUDE stm32l476xx_constants.s

        INCLUDE lcd.h


                AREA    main, CODE, READONLY
                EXPORT          __main
                ENTRY

__mainPROC

                bl              lcd_init
```

```
endlessbl                lcd_clear

;****************************************

                mov r0, #49;1
                bl num2font
                mov r1, #1
                bl lcd_draw
                mov r0, #50;2
                bl num2font
                mov r1, #2
                bl lcd_draw
                mov r0, #51;3
                bl num2font
                mov r1, #3
                bl lcd_draw
                mov r0, #52;4
                bl num2font
                mov r1, #4
                bl lcd_draw
                mov r0, #53;5
                bl num2font
                mov r1, #5
                bl lcd_draw

                mov r4, #0x130000 ;insert some delay. Can't test yet

delay0  subs r4, #1
                bne delay0

                bl                lcd_clear
                mov r0, #50;2
                bl num2font
                mov r1, #2
                bl lcd_draw
                mov r0, #51;3
                bl num2font
                mov r1, #3
                bl lcd_draw
                mov r0, #52;4
                bl num2font
                mov r1, #4
                bl lcd_draw
                mov r0, #53;5
                bl num2font
                mov r1, #5
                bl lcd_draw
                mov r0, #54;6
```

```
                bl num2font
                mov r1, #6
                bl lcd_draw

                mov r4, #0x130000     ;insert some delay. Can't test yet

delay1  subs r4, #1
                bne delay1

;*****************************************

                b               endless
                ENDP



                        ALIGN
                        AREA    myData, DATA, READWRITE

                        ALIGN


        END
```

NUM2FONT.s

```
num2font        PROC
                        EXPORT num2font
                        ;       r0 is an ascii number 0-9 (0x30-0x39)
                        ;       return font in r0
                        ;       Only use last hex digit 0-9; zero out A-F

;*****************************************
                        sub r0, #48
                        cmp r0, #9
                        movgt r0,#0
                        bgt end
                        lsl r0, #1
                        ldr r1, =numfont
                        add r1, r0
                        ldrh r0, [r1]
end             bx lr
;*****************************************
                        ENDP
```

Problem 3

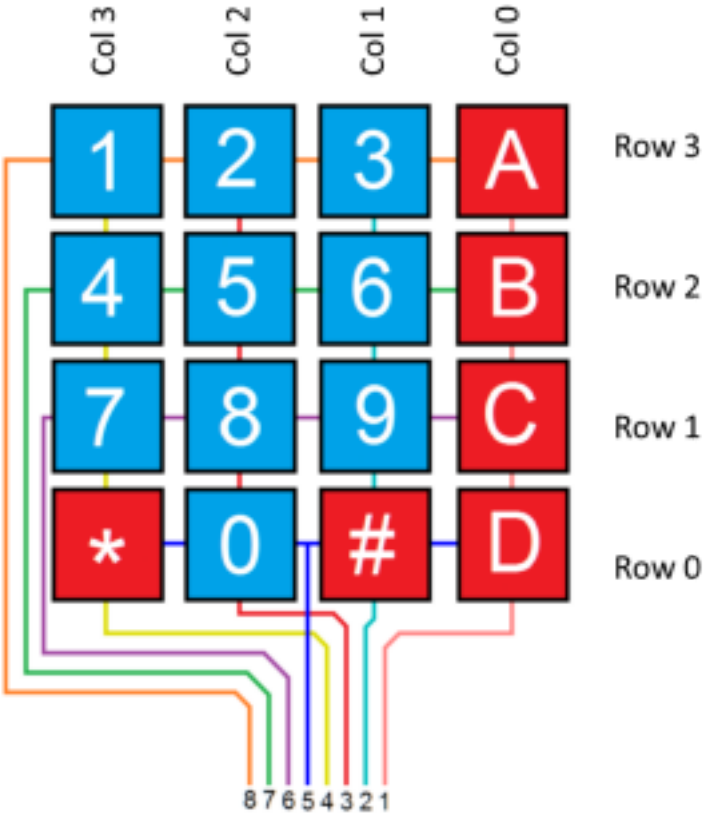
Your project is to:
1. Determine if a keypad button has been pressed
2. If so, scan the keypad rows to figure out which row and column is pressed (won't always work if multiple keys are pressed at once)
3. display the symbol for that key on the LCD in the next position
4. Repeat steps 1-3 indefinitely

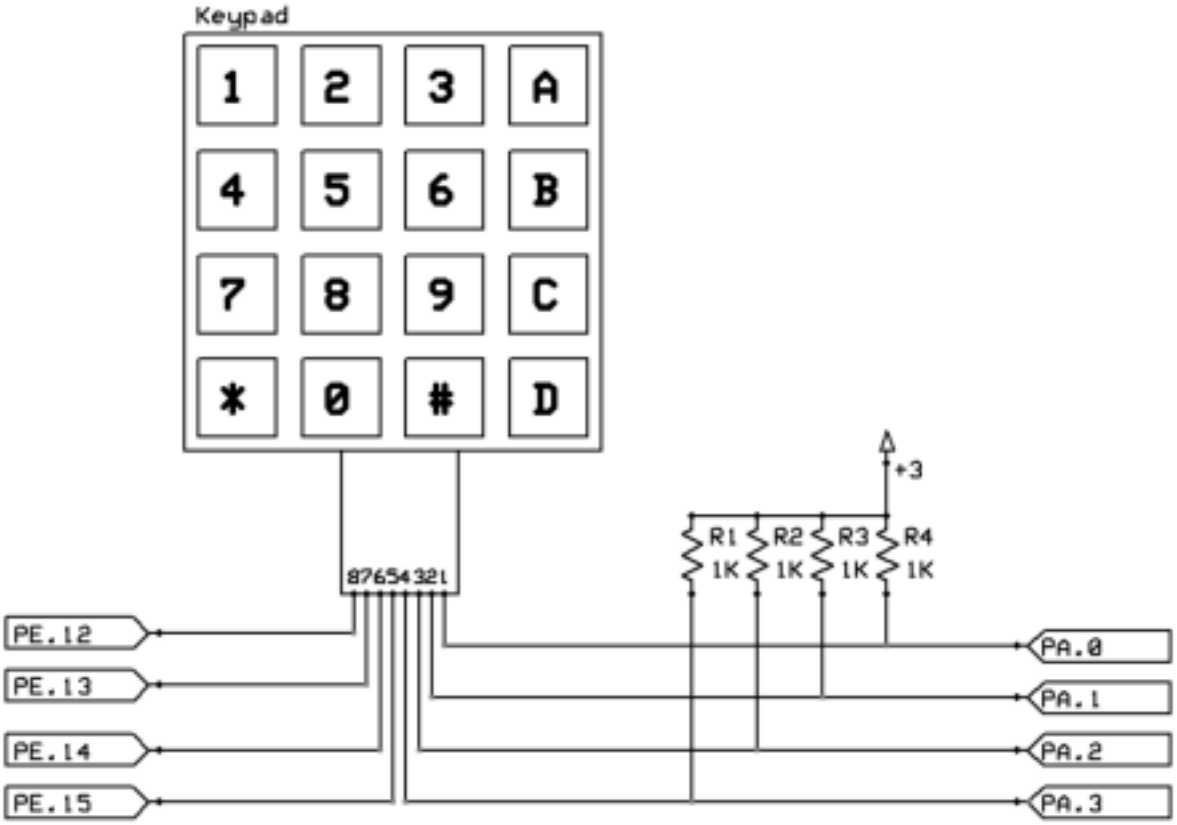
It is important that you display exactly 1 character each time a button is pressed. I have provided

a project called kpad (redacted) that provides most of what you need. You must:

1. Copy the already working/debugged code from your lcd.s file into this project. You can replace lcd.s with yours or just replace num2font and let2font in my code.
2. Write the code for kpad_scan in kpad.s.

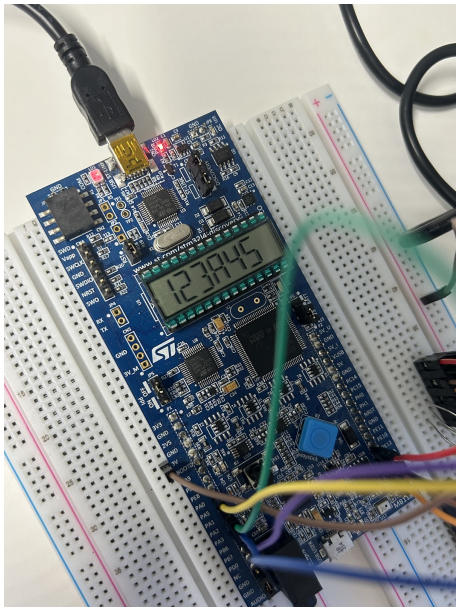
The rows and columns are numbered as shown. A schematic for connecting the keypad to your board and connecting resistors is on the next page. The 8-pin male-male connectors are the most reliable way to connect to your board. Make sure everything is connected exactly as shown. You may want to tape down the keypad to the desk at the top of the tail.

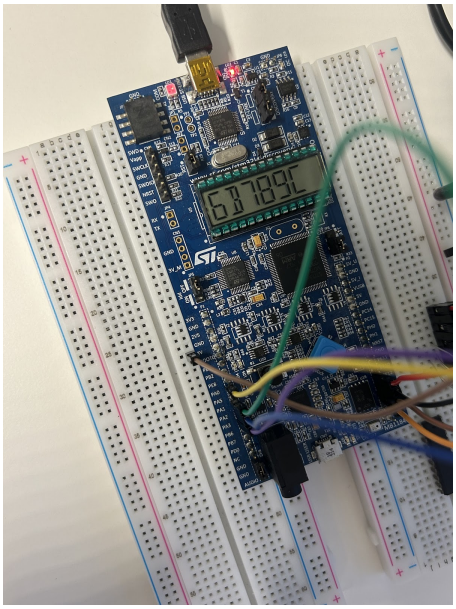

Col 3
Col 2
Col 1
Col 0
1
2
3
A
4
5
6
B
7
8
9
C
*
0
#
D
Row 3
Row 2
Row 1
Row 0
8 7 6 5 4 3 2 1

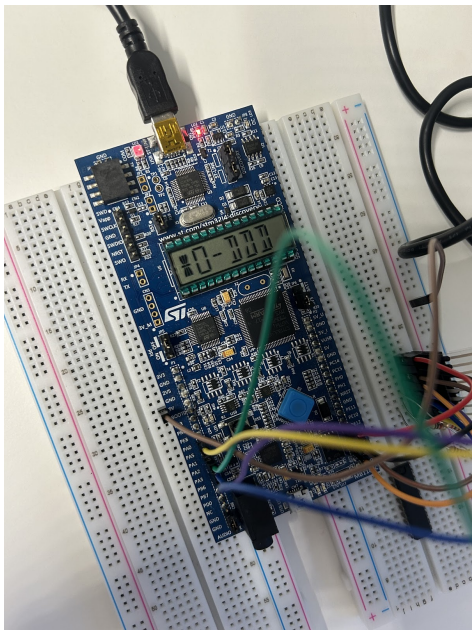

Keypad
1
2
3
A
4
5
6
B
7
8
9
C
*
0
#
D
+3
R1
1K
R2
1K
R3
1K
R4
1K
8 7 6 5 4 3 2 1
PE.12
PE.13
PE.14
PE.15
PA.0
PA.1
PA.2
PA.3

There is no way to display a hashtag on this display, so a hashtag will show up as a dash.

Demo your project; record a video for your report; turn in your code for kpad_scan.







KPAD_SCAN.s

```
kpad_scan       PROC                    ;Scan the 4 rows and return the first
row # pressed in r0 and the first col # pressed in r1 (>3 for none)
                EXPORT      kpad_scan
                push    {lr}
;*****************************
                mov r0, #0x7
                bl kpad_row_read
                cmp r0, #0xf
                movne r0, #0
                bne     end1
                mov r0, #0xb
                bl kpad_row_read
                cmp r0, #0xf
                movne r0, #1
                bne     end1
                mov r0, #0xd
                bl kpad_row_read
                cmp r0, #0xf
                movne r0, #2
                bne     end1
                mov r0, #0xe
                bl kpad_row_read
                cmp r0, #0xf
                movne r0, #3
                bne     end1

end1    push{r0}
                bl kpad_port_read
                cmp r0, #0x7
                moveq r1, #3
                cmp r0, #0xb
                moveq r1, #2
                cmp r0, #0xd
                moveq r1, #1
                cmp r0, #0xe
                moveq r1, #0
                pop {r0}
;*****************************
                pop             {pc}
                ENDP
```