



(Matrix I/O)  
(Curtis Robinson, Khondakar Mujtaba)  
(ELEN 120)  
(Wednesday 2:15)

**Objective:** Our goal throughout this lab project is to program the segmented LCD display on the DISCO board. We must know the register specifications in the STM32L4x6 advanced ARM®-based 32-bit MCUs reference manual and pre-supplied code discussed in the lab below. With the tools provided our goal is to display the text phrase “SANTA” wait one second “CLARA” wait one second and repeat indefinitely.

**Procedure:**

**Problem 1:** In the subroutine locations below we are going to utilize the let2font file to convert the chosen ASCII letter characters to their corresponding font representation. After that, we are supposed to return the font to the output positions with the implementation of the stack to then display the correct wording on the LCD display.

**Problem 2:**

For this problem, we are supposed to carry out the same additions to the file to allow the font conversion and displaying of the numbers 1234 and 5678 on their respective positions on the display.

**Problem 3:**

For this problem, we add the functionality of a keypad into the program. We must determine if one of the 16 keypad buttons is pressed and display that respective input on the LCD screen. Below we implement a kpad\_scan function in kpad.s that grabs the information from the input registers.

## Problem 1:

### let2font:

```
776 let2font      PROC
777               EXPORT let2font
778               ;   r0 is an ascii letter a-z (0x41-0x5A or 0x61-7A)
779               ;   return font in r0
780               ;   convert lower to upper - return 0 for out of range
781
782 ;*****
783 ;   Put your code here for this subroutine
784 ;*****
785               push    {r1}; saves values
786               push    {r2}
787               push    {r3}
788               CMP     r0,#0x40;checks if invalid
789               BLE     invalid
790
791               CMP     r0,#0x7B;
792               BGE     check1
793
794               CMP     r0,#0x5B;
795               BGE     check1
796
797
798               LDR     r1,=letfont; load address for letfont
799               CMP     r0,#0x60;
800               BLE     skip
801
802               SUB     r0,r0,#0x20; converts to uppercase if lower
803
804 skip
805               SUB     r0,r0,#0x41;find out which address corresponds with the letter
806               MOV     r2,#0x02;
807               MUL     r3,r0,r2;
808               ADD     r1,r1,r3;
809               LDRH    r0,[r1]; returns font
810               pop     {r3}
811               pop     {r2}
812               pop     {r1}
813               bx      lr
814
815 check1         CMP     r0,#0x60;
816               BLE     invalid
817 invalid
818               MOV     r0,#0x00;
819
820               pop     {r3}
821               pop     {r2}
822               pop     {r1}
823               bx      lr
```

## main.s

```
25 __main PROC
26
27     bl      lcd_init
28 endless bl      lcd_clear
29
30 ;*****
31 ; Put your code here to display Santa Clara
32 ;*****
33     MOV r3,#0x05
34     MOV r1,#0x01
35     LDR r2,=santa
36 loop
37     LDRB r0,[r2];
38     bl let2font
39     PUSH {r1}
40     PUSH {r2}
41     PUSH {r3}
42     bl lcd_draw
43     pop {r3}
44     pop {r2}
45     pop {r1}
46     ADD r2,r2,#0x01;
47     ADD r1,r1,#0x01;
48
49     SUBS r3,r3,#0x01;
50     CMP r3,#0x00;
51     BNE loop
52
53     LDR r3,=count
54     bl delay1
55
56
57     MOV r3,#0x06
58     MOV r1,#0x01;
59
60 loop2
61     LDRB r0,[r2];
62     bl let2font
63     PUSH {r1}
64     PUSH {r2}
65     PUSH {r3}
66     bl lcd_draw
67     pop {r3}
68     pop {r2}
69     pop {r1}
70     ADD r2,r2,#0x01;
71     ADD r1,r1,#0x01;
72
73     SUBS r3,r3,#0x01;
74     CMP r3,#0x00;
75     BNE loop2
76
77     LDR r3,=count
78     bl delay1
```

```
79
80     LDR r2,=santa
81
82     b     endless
83     ENDP
84
85
86     ALIGN
87     AREA    myData, DATA, READWRITE
88
89     santa    dcb     "S", "A", "N", "T", "A", " ", "C", "L", "A", "R", "A"
90     count    dcd     1333333
91     ALIGN
92
93
94     END
95
```

## Problem 2

Main.s:

```
25 __main PROC
26
27     bl      lcd_init
28 endless bl      lcd_clear
29
30 ;*****
31 ;   Put your code here to display Santa Clara
32 ;*****
33     MOV r3,#0x05
34     MOV r1,#0x01
35     LDR r2,=santa
36 loop
37     LDRB r0,[r2];
38     bl  num2font
39     PUSH {r1}
40     PUSH {r2}
41     PUSH {r3}
42     bl  lcd_draw
43     pop {r3}
44     pop {r2}
45     pop {r1}
46     ADD r2,r2,#0x01;
47     ADD r1,r1,#0x01;
48
49     SUBS r3,r3,#0x01;
50     CMP r3,#0x00;
51     BNE loop
52
53     LDR r3,=count
54     bl  delay1
55
56
57     MOV r3,#0x06
58     MOV r1,#0x01;
59
60 loop2
61     LDRB r0,[r2];
62     bl  num2font
63     PUSH {r1}
64     PUSH {r2}
65     PUSH {r3}
66     bl  lcd_draw
67     pop {r3}
68     pop {r2}
69     pop {r1}
70     ADD r2,r2,#0x01;
71     ADD r1,r1,#0x01;
72
73     SUBS r3,r3,#0x01;
74     CMP r3,#0x00;
75     BNE loop2
76
77     LDR r3,=count
78     bl  delay1
```

```

79
80         LDR r2,=santa
81
82         b      endless
83     ENDP
84
85
86         ALIGN
87         AREA    myData, DATA, READWRITE
88
89     santa    dcb    "1","2","3","4","5"," ","2","3","4","5","6"
90     count    dcd    1333333
91         ALIGN
92
93
94     END
95

```

## numfont2.s

```

763 num2font    PROC
764             EXPORT num2font
765             ;   r0 is an ascii number 0-9 (0x30-0x39)
766             ;   return font in r0
767             ;   Only use last hex digit 0-9; zero out A-F
768
769 ;*****
770 ;   Put your code here for this subroutine
771 ;*****
772             push    {r1}; saves values
773             push    {r2}
774             push    {r3}
775             CMP r0,#0x30;checks if invalid
776             BLE invalidl
777
778             CMP r0,#0x39;
779             BGE invalidl
780             LDR r1,=numfont; load address for letfont
781             SUB r0,r0,#0x30;find out which address corresponds with the letter
782             MOV r2,#0x02;
783             MUL r3,r0,r2;
784             ADD r1,r1,r3;
785             LDRH r0,[r1]; returns font
786             pop {r3}
787             pop {r2}
788             pop {r1}
789             bx lr
790
791 invalidl
792             MOV r0,#0x00;
793
794             pop {r3}
795             pop {r2}
796             pop {r1}
797             bx lr
798
799
800             ENDP

```

### Problem 3:

```
58 kpad_scan      PROC                               ;Scan t
59      EXPORT    kpad_scan
60      push      {lr}
61      MOV r3,#0xF; checks first row
62      push {r0}
63      MOV r0,#0x7;
64      bl kpad_row_read
65      CMP r3,r0;
66      POP {r1}
67      MOV r2,#0
68      BGT ret
69
70      MOV r0,#0xB; checks second row
71      push {r1}
72      bl kpad_row_read
73      CMP r3,r0;
74      POP {r1}
75      MOV r2,#1
76      BGT ret
77
78      MOV r0,#0xD;checks third row
79      push {r1}
80      bl kpad_row_read
81      CMP r3,r0;
82      POP {r1}
83      MOV r2,#2
84      BGT ret
85
86      MOV r0,#0xE;checks last row
87      push {r1}
88      bl kpad_row_read
89      CMP r3,r0;
90      POP {r1}
91      MOV r2,#3
92      BGT ret
93 ret
94      push {r2}
95      CMP r0,#0x7; checks which column in the row is pressed
96      BNE nxt1
97
98      MOV r1,#0x3
99      b    ans
100
101 nxt1      CMP r0,#0xB;
102      BNE nxt2
103
104      MOV r1,#0x2
105      b    ans
106
107 nxt2      CMP r0,#0xD;
108      BNE nxt3
109
110      MOV r1,#0x1
111      b    ans
112
113 nxt3      CMP r0,#0xE;
114      BNE ans
115
116      MOV r1,#0x0
117      b    ans
118 ans
119      pop {r0}; restore row value
120      pop      {pc}
121      ENDP
```



## Demo Videos:

Problem 1:

[https://drive.google.com/file/d/1vbmB37prATh0jty4OchS\\_PwtPw3hdaXD/view?usp=sharing](https://drive.google.com/file/d/1vbmB37prATh0jty4OchS_PwtPw3hdaXD/view?usp=sharing)

Problem 2:

[https://drive.google.com/file/d/1G4euDIDRwhTT0hbo3z7dyvxv\\_Gcbi2uM/view?usp=sharing](https://drive.google.com/file/d/1G4euDIDRwhTT0hbo3z7dyvxv_Gcbi2uM/view?usp=sharing)

Problem 3:

<https://drive.google.com/file/d/1Vh36kh6vAKjRmibtZV6D2az8wxruCEEY/view?usp=sharing>

