

Mehmet Atas

2304020

Preliminary Work for Exp. #2

1) I write a subroutine, DELAY100, that causes approximately 100 msec delay upon calling.

DELAY100.s		test.s	
1		AREA	DELAY100, READONLY, CODE
2		THUMB	
3		EXPORT	DELAY100
4			
5		__DELAY100	
6	0.250 us	push	{r0,r1}
7	0.083 us	mov	r0,#0
8	0.167 us	mov32	r1,#240000 ;number needed for 100ms delay
9	20.000 ms	count	add r0,#1
10	20.000 ms	cmp	r0,r1
11	60.000 ms	bne	count
12	0.250 us	pop	{r0,r1}
13	0.250 us	bx	lr
14			
15		end	

DELAY100.s		test.s	
1		AREA	main, READONLY, CODE
2		THUMB	
3		EXPORT	__main
4		EXTERN	DELAY100
5			
6		__main	
7	0.167 us	mov32	r2,#0x7fffffff
8			
9	0.250 us	bl	DELAY100
10		;	
11		loop	b loop
12		end	

2) I write a program that continuously detects which key is pressed and outputs the ID of the key through Terminate Window after the key is released.

```

22
23         ldr     r1,=GPIO_PORTB_DIR
24         ldr     r0,[r1]
25         bic     r0,#0xff
26         orr     r0,#IOB           ;define i/o ports
27         str     r0,[r1]
28         ldr     r1,=GPIO_PORTB_AFSEL
29         ldr     r0,[r1]
30         bic     r0,#0xff
31         str     r0,[r1]
32         ldr     r1,=GPIO_PORTB_DEN
33         ldr     r0,[r1]
34         orr     r0,#0xff
35         str     r0,[r1]
36         ldr     r0,=GPIO_PORTB_PUR
37         orr     r1,#0x0f
38         str     r1,[r0]           ;enable pull up resistor for input
39
40
41 getit    ldr     r0,=GPIO_PORTB_DATA
42         ldr     r1,[r0]           ;read inputs
43         mov     r2,r1
44         lsl     r2,#4             ;shift input to output
45         str     r2,[r0]           ;give according output
46         mov     r0,#50
47 delay    bl     DELAY100
48         subs   r0,#1
49         bne    delay             ;wait for 5s
50         b     getit             ;restart the code
51
52         end

```

DELAY100.s Startup.s **prelimQ2.s**

```

1  GPIO_PORTB_DATA EQU 0x400053fc
2  GPIO_PORTB_DIR  EQU 0x40005400
3  GPIO_PORTB_AFSEL EQU 0x40005420
4  GPIO_PORTB_DEN  EQU 0x4000551C
5  GPIO_PORTB_PUR  EQU 0x40005510
6  IOB              EQU 0xF0
7  SYSTCTL_RCGCGPIO EQU 0x400FE608
8
9          AREA      main, READONLY, CODE
10         THUMB
11         EXPORT    __main
12         EXTERN    DELAY100
13
14 __main
15         ldr     r1,=SYSTCTL_RCGCGPIO
16         ldr     r0,[r1]
17         orr     r0,#0x02
18         str     r0,[r1]           ;start clock for pin B
19         nop
20         nop
21         nop
22
23         ldr     r1,=GPIO_PORTB_DIR
24         ldr     r0,[r1]
25         bic     r0,#0xff
26         orr     r0,#IOB           ;define i/o ports
27         str     r0,[r1]
28         ldr     r1,=GPIO_PORTB_AFSEL
29         ldr     r0,[r1]
30         bic     r0,#0xff
31         str     r0,[r1]

```

3) While considering the interface of the 4x4 keypad introduced in Chapter 2, I write a program that constantly detects which key is pressed and gives the key's ID via the Terminate Window after the key is released. I can only assume that one key will be pressed at a time and that no other keys can be pressed before releasing a key. The program is resistant to possible bouncing effects during both pressing and releasing.

a. How can you detect whether any key is pressed?

We can check for inputs, if any is LOW then a key is pressed.

b. How can you detect whether a pressed key is released?

We can check the inputs after a key is pressed, if none of the inputs are LOW then the key is released.

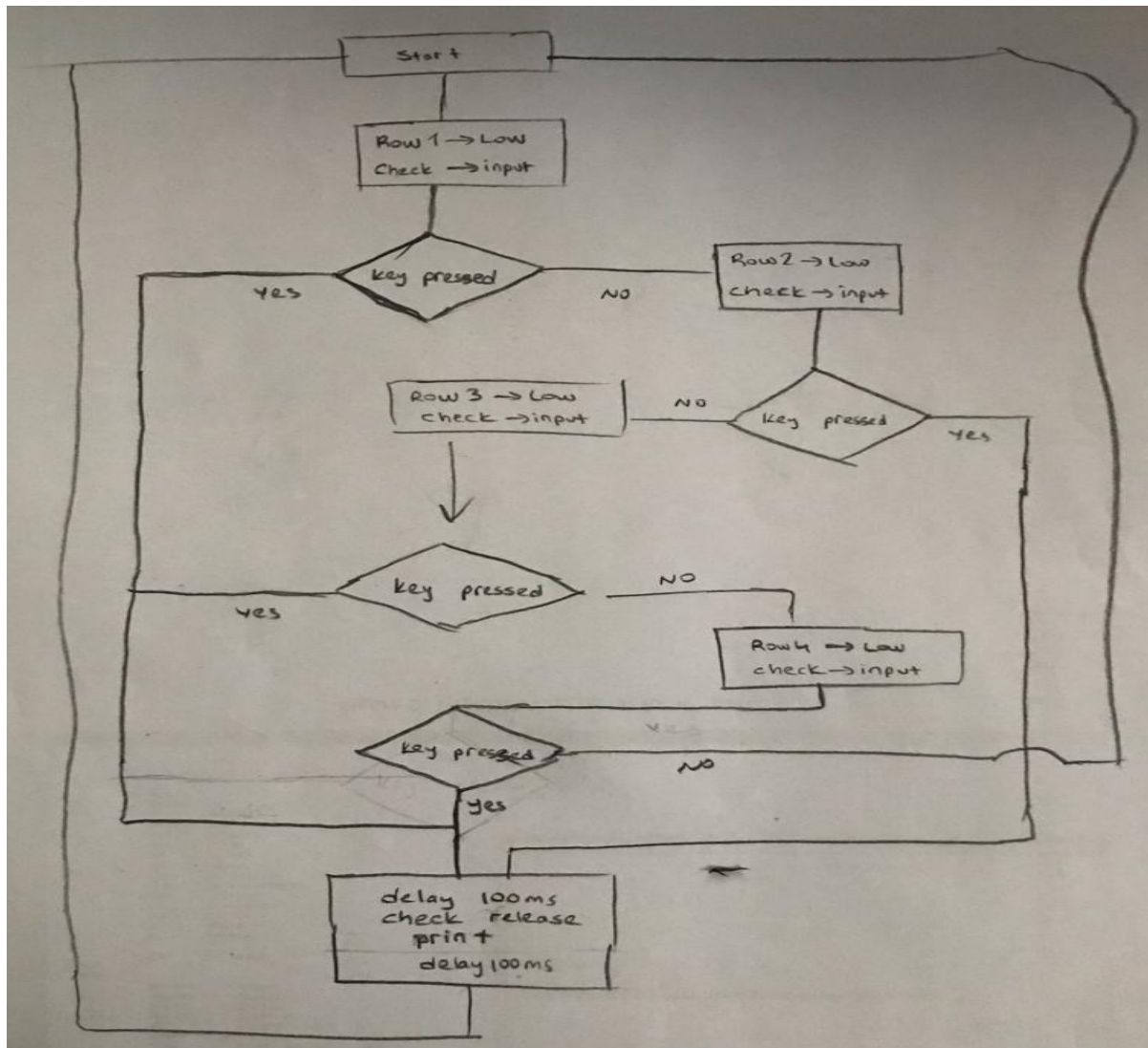
c. Assuming that you have detected that a key is pressed. Explain your algorithm to determine which one is pressed.

We can make one row's output LOW at a time and check the inputs. If any key is pressed in that row then we can see the column from the inputs. So, we can locate any key.

d. Discuss what can happen due to bouncing. How can you avoid bouncing effects?

If we check the output immediately after it is pressed, we can see that the output is HIGH even if the key is not yet released due to bouncing effect. We can wait 100ms before reading the input, so that bouncing effect disappears.

e. Now, develop your overall end-to-end algorithm that outputs ID of the pressed key to the terminal window and draw its flow chart.



f. Implement the developed algorithm in part-e by using assembly language.

```

1      GPIO_PORTB_DATA      EQU 0x400053fc
2      GPIO_PORTB_DIR       EQU 0x40005400
3      GPIO_PORTB_AFSEL     EQU 0x40005420
4      GPIO_PORTB_DEN       EQU 0x4000551C
5      GPIO_PORTB_PUR       EQU 0x40005510
6      IOB                  EQU 0xF0
7      SYSTCL_RCGCGPIO      EQU 0x400FE608
8
9      AREA                 main, READONLY, CODE
10     THUMB
11     EXPORT               _main
12     EXTERN               DELAY100
13     EXTERN               OutChar
14
15     __main
16     ldr                  r1,=SYSTCL_RCGCGPIO
17     ldr                  r0,[r1]
18     orr                  r0,#0x02
19     str                  r0,[r1]           ;start clock for pin B
20     nop
21     nop
22     nop
23
24     ldr                  r1,=GPIO_PORTB_DIR
25     ldr                  r0,[r1]
26     bic                  r0,#0xff
27     orr                  r0,#IOB           ;define i/o ports
28     str                  r0,[r1]
29     ldr                  r1,=GPIO_PORTB_AFSEL
30     ldr                  r0,[r1]
31     bic                  r0,#0xff
32     str                  r0,[r1]
33     ldr                  r1,=GPIO_PORTB_DEN
34
35     ldr                  r0,[r1]
36     orr                  r0,#0xff
37     str                  r0,[r1]
38     ldr                  r0,=GPIO_PORTB_PUR
39     orr                  r1,#0x0f
40     str                  r1,[r0]           ;enable pull up resistor for input
41
42     firstRow             ldr                  r0,=GPIO_PORTB_DATA
43     mov                  r5,#0
44     mov                  r1,#0xe0
45     str                  r1,[r0]           ;make the output "0" for first row
46     nop
47     nop
48     nop                   ;wait for output to stablize
49     ldrb                 r1,[r0]
50     cmp                  r1,#0xee
51     moveq                 r5,#0x30
52     cmp                  r1,#0xed
53     moveq                 r5,#0x31
54     cmp                  r1,#0xeb
55     moveq                 r5,#0x32
56     cmp                  r1,#0xe7
57     moveq                 r5,#0x33           ;check for each column
58     cmp                  r5,#0
59     bne                  print             ;start print operation if any key is detected
60     mov                  r1,#0xd0
61     str                  r1,[r0]           ;make the output "0" for second row
62     nop
63     nop
64     nop                   ;wait for output to stablize
65     ldrb                 r1,[r0]
66     cmp                  r1,#0xde

```

```

67      moveq    r5,#0x34
68      cmp     r1,#0xdd
69      moveq    r5,#0x35
70      cmp     r1,#0xdb
71      moveq    r5,#0x36
72      cmp     r1,#0xd7
73      moveq    r5,#0x37      ;check for each column
74      cmp     r5,#0
75      bne     print          ;start print operation if any key is detected
76      mov     r1,#0xb0
77      str     r1,[r0]        ;make the output "0" for third row
78      nop
79      nop
80      nop                  ;wait for output to stabilize
81      ldrb    r1,[r0]
82      cmp     r1,#0xbe
83      moveq    r5,#0x38
84      cmp     r1,#0xbd
85      moveq    r5,#0x39
86      cmp     r1,#0xbb
87      moveq    r5,#0x41
88      cmp     r1,#0xb7
89      moveq    r5,#0x42      ;check for each column
90      cmp     r5,#0
91      bne     print          ;start print operation if any key is detected
92      mov     r1,#0x70
93      str     r1,[r0]        ;make the output "0" for fourth row
94      nop
95      nop
96      nop                  ;wait for output to stabilize
97      ldrb    r1,[r0]
98      cmp     r1,#0x7e
99      moveq    r5,#0x43

```

```

90      cmp     r5,#0
91      bne     print          ;start print operation if any key is detected
92      mov     r1,#0x70
93      str     r1,[r0]        ;make the output "0" for fourth row
94      nop
95      nop
96      nop                  ;wait for output to stabilize
97      ldrb    r1,[r0]
98      cmp     r1,#0x7e
99      moveq    r5,#0x43
100     cmp     r1,#0x7d
101     moveq    r5,#0x44
102     cmp     r1,#0x7b
103     moveq    r5,#0x45
104     cmp     r1,#0x77
105     moveq    r5,#0x46      ;check for each column
106     cmp     r5,#0
107     bne     print          ;start print operation if any key is detected
108     b       firstRow       ;return to first row if no key is detected
109
110     print    check         ;wait 100ms for debouncing (for pressing)
111     ldrb    r1,[r0]
112     and     r1,#0x0f
113     cmp     r1,#0x0f
114     bne     check          ;wait until key is released
115     bl      OutChar
116     mov     r5,#0x0d
117     bl      OutChar        ;print the key value and a newline
118     bl      DELAY100       ;wait 100ms for debouncing (for releasing)
119     b       firstRow       ;restart the code
120
121     end
122

```



Termite 3.4 (by CompuPhase)



COM3 9600 bps, 8N1, no handshake

Settings

Clear

About

Close

0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F

|



L1 L2 L3 L4

R1 ee ed eb e7 0123

R2 de dd db d7 4567

R3 be bd bb b7 89AB

R4 7e 7d 7b 77 CDEF

B0->R1

B1->R2

B2->R3

B3->R4

B4->L1

B5->L2

B6->L3

B7->L4