

EE 2004
2016-2017: Semester B
Assignment 2
Due: Apr 7, 2017
Solution

Instructions:

You will submit the following 5 files: (1) ConvertHexArrayToASCIIArray.asm, (2) ConvertHexArrayToASCIIArray.lst, (3) InterruptProgram.asm, (4) InterruptProgram.lst and (5) The Word file showing your calculation in Question 2(a). All files should be zipped and submitted as a single zip file.

Students will submit the assignment through Canvas. Click on the item "Assignment" on the left panel. You should see a row with title "Assignment 2". Click on the "Assignment 2" label and find the "Submit Assignment" label on the right panel. Click on it and upload the requested zip file.

Question 1 (40 marks)

- (a) (10 marks) Write a subroutine ConvertHexToASCII that implements a lookup table converting a hexadecimal digit ranging from 0 to F to its corresponding ASCII representation. Please use the tblrd mechanism (instead of the computed goto method). Store the output in WREG. The lookup table is provided below. (Hint: Use the program LookupTable.asm I provided as a blueprint and make appropriate modifications. Also recall what you did in the lab when you implemented the SVN_SEG subroutine.)

Hexadecimal	ASCII
0	0x30
1	0x31
2	0x32
3	0x33
4	0x34
5	0x35
6	0x36
7	0x37
8	0x38
9	0x39
A	0x41
B	0x42
C	0x43
D	0x44
E	0x45
F	0x46

- (b) (30 marks) Given an array of 8 bytes stored in data memory location 0x040 through 0x047 as shown in Fig. 1, write a **loop** that writes the ASCII codes corresponding to the **least** significant nibble stored in the data memory address 040 + *i* if the address is even to the data memory address 050 + *i*, and writes the ASCII codes corresponding to the **most** significant nibble if the address 040 + *i* is odd, where *i* = 0, 1, ... 7 (See Fig. 2). Use ConvertHexToASCII you wrote in (a) to convert a hexadecimal digit to its corresponding ASCII representation. You can choose to use the program template shown in Fig. 3. Your program **must** involve a loop and you must use FSRs to point to the arrays starting from data memory with address 0x040 and 0x050. **For this question, you will submit a single .asm file, named ConvertHexArrayToASCIIArray.asm, containing the subroutine ConvertHexToASCII you implemented in (a) and the program you write in (b). Please also submit the .lst file generated when you assemble your .asm file.**

Data Memory Address	Value
0x040	0xF1
0x041	0x54
0x042	0xAC
0x043	0x3F
0x044	0x45
0x045	0x67
0x046	0x98
0x047	0xF4

Figure 1 Array storing hexadecimal digits

Data Memory Address	Value
0x050	0x31
0x051	0x35
0x052	0x43
0x053	0x33
.....	

Figure 2 Expected result after your program completes

```

Main:      ORG      0x000000
           movlw 0xF1; Initialize array starting from 0x040
           ; according to Fig. 1.
           movwf 0x40, A
           movlw 0x54
           movwf 0x41, A
           movlw 0xAC
           movwf 0x42, A
           movlw 0x3F
           movwf 0x43, A

           ; Other initializations

MainLoop:

           bra $

ConvertHexToASCII: ;Subroutine you wrote in Part (a)

           END

```

Figure 3 Program template for Question 1

```

; EE 2004
; Instructor: Bernard Chiu
; Assignment 2 Q1
;-----
LIST    P=18F4520                ;directive to
define processor                  #include <P18F4520.INC>          ;processor
specific variable definitions
;-----
                                cblock 0x00
                                InputConvertHexToASCII
                                COUNTSub
                                COUNT
                                This
                                endc

                                ORG 0x000000
                                goto Main

Main:  movlw 0xF1; Initialize array starting from 0x040
                                ; according to Fig. 1.

                                movwf 0x40, A
                                movlw 0x54
                                movwf 0x41, A
                                movlw 0xAC
                                movwf 0x42, A
                                movlw 0x3F
                                movwf 0x43, A
                                movlw 0x45
                                movwf 0x44, A
                                movlw 0x67
                                movwf 0x45, A
                                movlw 0x98
                                movwf 0x46, A
                                movlw 0xF4
                                movwf 0x47, A

                                lfsr 0, 0x040
                                lfsr 1, 0x050

                                movlw d'8'
                                movwf COUNT, A

MainLoop:  movff POSTINC0, This
                                btfss FSR0L, 0; Jump if This is stored in an even address (note: FSR0
is odd, FSR0 + 1 is even)
ExtractUpper:  swapf This, F, A
ExtractLower:  movlw 0x0f
                                andwf This, F, A
                                movff This, InputConvertHexToASCII
                                call ConvertHexToASCII
                                movff WREG, POSTINC1

                                decfsz COUNT, F, A
                                bra MainLoop

                                bra$

```

; ConvertHexToASCII is the subroutine required in Part (a).

```
ConvertHexToASCII:  movlw  upper ASCIIList
                   movwf  TBLPTRU
                   movlw  high ASCIIList
                   movwf  TBLPTRH
                   movlw  low  ASCIIList
                   movwf  TBLPTRL

                   movf    InputConvertHexToASCII, W, A
                   addlw   0x01
                   movwf   COUNTSub, A

Loop:              tblrd*+
                   decf    COUNTSub, F, A
                   bnz     Loop
                   movf     TABLAT, W    ;Result in WREG
                   return

                   org 0x000200
ASCIIList db 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39, 0x41, 0x42,
0x43, 0x44, 0x45, 0x46

                   end
```

Question 2 (60 marks)

- (a) (5 marks) Determine the initial value of TMR0 to generate a 5 ms time delay without using a prescaler (equivalently, prescaler = 1). The clock frequency of the PIC18 microcontroller is 4MHz. **Submit a Word file showing your calculation.**

(FFFF- initial value + 1) = d'5000'

initial value = 0xEC78.

- (b) (55 marks) Suppose Port C is connected to the 8-button keypad and Ports B and D are connected to the 4-digit 7 segment LED as specified in Lab 2, Task 2. Write an interrupt-based program to detect the button pressed and display the corresponding digit in the 4-digit 7 segment LED in the following manner. In the following discussion, U, V, X and Y could be any integer from 1 to 8.

- Initially, "0000" is displayed in the 4-digit 7 segment LED.
- After the button PX in the 8-button keypad is pressed, the display would change to "X000".
- After the second button PY is pressed, the display would change to "XY00".
- After the third and fourth buttons, PU and PV respectively, are pressed, the display would become "XYUV".
- After the fifth button PY is pressed, the display would change to "YYUV". That is, the label of the current button Y, will replace the digit X originally placed in the first position of the 4-digit 7 segment LED.

A few hints:

- Use the main program to scan continuously in order to identify the button that has been pressed.
- Set up the Timer 0 so that it triggers an interrupt every 5 ms. When the interrupt occurs, switch to display the next digit and reinitialize TMR0. You can use the results you obtained in Part (a). For simplicity, you are suggested not to use the priority scheme.
- Consider using FSR0 to point to the file register in which the button number should be stored when a button is pressed.

You can make use of the following template. **You will submit a .asm file, named InterruptProgram.asm and the .lst file generated when you assemble your .asm file.**

```

; 4 variables storing the numbers displayed in the 4-digit 7 segment
; LED
        cblock 0x00
        Num1
        Num2
        Num3
        Num4
        endc

        ORG 0x000000
        bra Main
        ORG 0x000008
        bra Timer0_ISR
        ; Check which interrupt flag is raised, branch to the
        ; appropriate ISR.

        ORG 0x000100
Main:
        ; Initialize Timer 0 and specify the initial numbers
        ; displayed in the 4-digit 7 segment LED.

        ; Specify data flow direction of various ports.

MainLoop:  call KeyScan

        ; if no button is pressed, go back to MainLoop to scan again.
        ; if a button is pressed, store it in the appropriate variable
        ; among Num1 to Num4 and then go back to the MainLoop to scan
        ; again.

Keyscan:

Timer0_ISR:

        retfie

SVN_SEG:

        END

```

Figure 4 Program template for Question 2

Important notes to students:

The solution I provide here works in hardware. For a working program, a debouncing mechanism has to be implemented (Line 95 of my code) and there is also a practical need to preserve and restore contents of the WREG and STATUS registers in the ISR (Line 109, 131). Since there is no dedicated laboratory session for this exercise, I do not expect you to produce a program that really works in hardware.

In marking your assignment, I only have the following three expectations:

1. Proper implementation of the Timer 0 ISR.
2. Proper initialization of the LED display (Lines 69-83) and Timer interrupt mechanism (Lines 86-90).
3. Proper implementation of the functions of KeyScanLoop in Main and ChangeDigit subroutine (Lines 95-98, 102-107)

:\Users\Bernard\Documents\EE2004\2017\Assignment2\EE20041617Assgmt2\EE20041617Assgmt2\14_KeyLock\KeyScanold.as

```
1  ;*****
2  ;   EE 2004 16/17 Sem B Assignment 2
3  ;   Instructor: Bernard Chiu
4  ;
5  ; Question Statement:
6  ; Write an
7  ;interrupt-based program to detect the button pressed and display the corresponding
8  ;digit in the 4-digit 7 segment LED in the following manner. In the following discussion,
9  ;U, V, X and Y could be any integer from 1 to 8.
10 ;· Initially, "0000" is displayed in the 4-digit 7 segment LED.
11 ;· After the button PX in the 8-button keypad is pressed, the display would change
12 ;to "X000".
13 ;· After the second button PY is pressed, the display would change to "XY00".
14 ;· After the third and fourth buttons, PU and PV respectively, are pressed, the
15 ;display would become "XYUV".
16 ;· After the fifth button PY is pressed, the display would change to "YYUV". That
17 ;is, the label of the current button Y, will replace the digit X originally placed in
18 ;the first position of the 4-digit 7 segment LED.
19 ;
20 ;*****
21 ;
22 ;   Files Required: P18F4520.INC
23 ;   Port C is connected to the 8-button keypad and Ports B and D are
24 ;   connected to the 4-digit 7 segment LED as specified in Lab 2, Task 2.
25 ;
26 ;*****
27
28         LIST      P=18F4520          ;directive to define processor
29         #include <P18F4520.INC>      ;processor specific variable definitions
30
31 ;-----
32 ;Configuration bits, see above .inc file for details on notation.
33 ;
```

:\Users\Bernard\Documents\EE2004\2017\Assignment2\EE20041617Assgmt2\EE20041617Assgmt2\14_KeyLock\KeyScanold.as

```
34          CONFIG  OSC = XT          ;XT oscillator
35          CONFIG  WDT = OFF         ;WDT disabled
36          CONFIG  LVP = OFF         ;Low Voltage ICSP disabled
37          CONFIG  PBADEN = OFF      ;PortB Digital port
38
39          ;-----
40          ;Variable definitions
41          ;
42          CBLOCK  0x000
43          PAT_NUM; output of the subroutine SVN_SEG
44          DIGIT_0; Four digits displayed
45          DIGIT_1
46          DIGIT_2
47          DIGIT_3
48          DIGIT_POS; Position in the 4-digit 7-segment LED currently being displayed
49          KEY_POS; Position of the digit that will be replaced in the next button
push
50          KEY_CODE; Button number that has been pushed (1 to 8) or 0 if no button
is pressed (see Keyscan)
51          TMP_WREG; Preserve the content of WREG and STATUS registers in an
interrupt-based program
52          TMP_STATUS; i.e., contents would be the same before and after ISR
execution
53          COUNT; used in lookup table
54          DELAY_H; used in Delay loop. Delay loop is necessary for handling debounce
55          DELAY_L
56          ENDC
57
58          ;-----
59          ;Reset vector
60          ;
61          ORG      0x000000
62          bra      Main          ;go to start of main code
```

:\Users\Bernard\Documents\EE2004\2017\Assignment2\EE20041617Assgmt2\EE20041617Assgmt2\14_KeyLock\KeyScanold.as

```
63
64          ORG      0x000008
65          bra      Timer0_ISR
66
67          org      0x000100
68          ;-----
69 Main:      clrfs DIGIT_0, A
70          clrfs DIGIT_1, A
71          clrfs DIGIT_2, A
72          clrfs DIGIT_3, A
73
74          clrfs TRISB ; digit position
75          clrfs TRISD ; digit pattern
76          setfs TRISC ; 8-button keypad
77
78          ; Initialize Display: 0 in the 0th position
79          clrfs PORTB
80          clrfs DIGIT_POS, A
81          movffs DIGIT_0, PAT_NUM
82          calls SVN_SEG
83          movwf PORTD
84
85          ; Initialize Timer
86          movlw 0x88 ;Timer0 on, FOSC/4, no prescalar, 16-bit
87          movwf TOCON
88          bcf INTCON, TMR0IF ;Clear Timer0 Interrupt Flag
89          bsf INTCON, TMR0IE ;Enable Timer0 Interrupt
90          bsf INTCON, GIE ;Enable Global Interrupt
91
92          clrfs KEY_POS, A
93          clrfs KEY_CODE, A
94
95 KeyscanLoop: calls KeyscanWithDebounce; Students can just call Keyscan without handling
```

:\Users\Bernard\Documents\EE2004\2017\Assignment2\EE20041617Assgmt2\EE20041617Assgmt2\14_KeyLock\KeyScanold.as

```
debounce
96      tstfsz KEY_CODE, A
97      call ChangeDigit
98      bra KeyscanLoop
99
100     bra $
101
102 ChangeDigit: movlw    0x03                ;Key position is KEY_POS mod 4 (remainder of
KEY_POS divided by 4) -- see https://en.wikipedia.org/wiki/Modulo_operation
103     andwf    KEY_POS, W                ;These two lines are implementing KEY_POS mod
4
104     lfsr     1, DIGIT_0                ;FSR1 = address of DIGIT_0
105     movff    KEY_CODE, PLUSW1          ;Move detected KEY_CODE to register FSR1 +
(KEY_POS mod 4)
106     incf     KEY_POS
107     return
108
109 Timer0_ISR: movff    WREG,    TMP_WREG    ;Preserve the contents of WREG and STATUS
registers
110     movff    STATUS, TMP_STATUS ;Students are not required to do this.
111
112     btfss    INTCON, TMR0IF            ;Check if Timer0 Interrupt
113     bra      INT_Exit
114     bcf      INTCON, TMR0IF
115
116     movlw    0xEC                      ;Reinitize TMR0 to generate the next 5ms delay
117     movwf    TMR0H
118     movlw    0x78
119     movwf    TMR0L
120
121     incf     DIGIT_POS
122     movlw    0x03
123     andwf    DIGIT_POS, W
```

4

:\Users\Bernard\Documents\EE2004\2017\Assignment2\EE20041617Assgmt2\EE20041617Assgmt2\14_KeyLock\KeyScanold.as

```
124          movwf    PORTB
125
126          lfsr     0, DIGIT_0
127          movff    PLUSW0, PAT_NUM
128          call     SVN_SEG
129          movwf    PORTD
130
131 INT_Exit:  movff    TMP_WREG,    WREG    ;Preserve the contents of WREG and STATUS
registers
132          movff    TMP_STATUS, STATUS
133          retfie
134
135 KeyscanWithDebounce:  call     Delay                ;Handle Key Debounce
136                      call Keyscan
137                      movwf KEY_CODE, A
138
139 Wait:      call Keyscan; Debounce: makes sure only one digit is changed per push
140          tstfsz WREG; if true, skip out
141          bra Wait
142          call Delay ;Handle Key Debounce
143          return
144
145 Delay:     movlw    0x10 ; Here for handling debounce
146          movwf    DELAY_H
147          clrf     DELAY_L
148 DelayLoop: decfsz   DELAY_L
149          bra      DelayLoop
150          decfsz   DELAY_H
151          bra      DelayLoop
152          return
153
154 Keyscan:   btfss   PORTC, 0
155          retlw    d'1'
```

:\Users\Bernard\Documents\EE2004\2017\Assignment2\EE20041617Assgmt2\EE20041617Assgmt2\14_KeyLock\KeyScanold.as

```
156          btfss    PORTC, 1
157          retlw     d'2'
158          btfss    PORTC, 2
159          retlw     d'3'
160          btfss    PORTC, 3
161          retlw     d'4'
162          btfss    PORTC, 4
163          retlw     d'5'
164          btfss    PORTC, 5
165          retlw     d'6'
166          btfss    PORTC, 6
167          retlw     d'7'
168          btfss    PORTC, 7
169          retlw     d'8'
170          retlw     d'0'
171
172
173
174  SVN_SEG:      movlw   upper array
175               movwf   TBLPTRU
176               movlw   high array
177               movwf   TBLPTRH
178               movlw   low array
179               movwf   TBLPTRL
180               movf    PAT_NUM, W, A
181               addlw   0x01
182               movwf   COUNT, A
183  Loop:         tblrd*+
184               decf    COUNT, F, A
185               bnz     Loop
186               movf    TABLAT, W      ;Result in WREG
187               return
188
```

:\Users\Bernard\Documents\EE2004\2017\Assignment2\EE20041617Assgmt2\EE20041617Assgmt2\14_KeyLock\KeyScanold.as

```
189             org 0x000500
190
191 array db B'00111111', B'00000110', B'01011011', B'01001111', B'01100110', B'01101101',
192         B'01111101', B'00000111', B'01111111';
193 ;*****
194 ;End of program
195 ;
196             END
197
```