



MICROCHIP

AN557

Four Channel Digital Voltmeter with Display and Keyboard

Author: Stan D'Souza
Microchip Technology Inc.

INTRODUCTION

The PIC16C71 is a member of the mid-range family of 8-bit, high-speed microcontrollers, namely the PIC16CXXX. The salient features of the PIC16C71 are:

- Improved and enhanced instruction set
- 14-bit instruction word
- Interrupt capability
- On-chip four channel, 8-bit A/D converter

This application note demonstrates the capability of the PIC16C71. This application note has been broken down into four subsections:

Multiplexing Four 7-Segment LED Displays

Multiplexing Four 7-Segment LED Displays and Scanning a 4x4 Keypad

Multiplexing Four 7-Segment LED Displays and the A/D Channel0

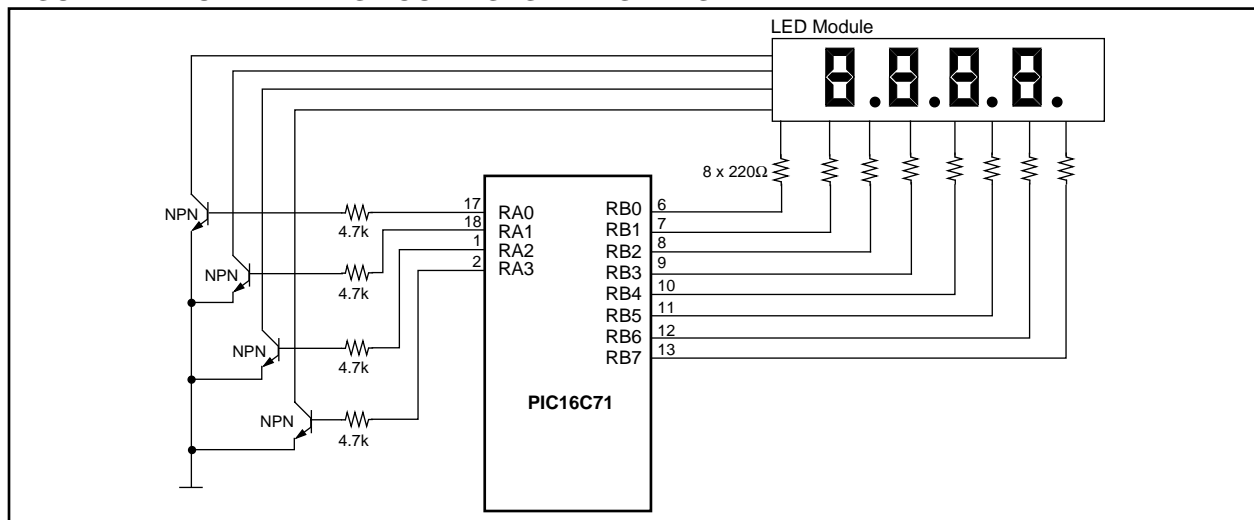
Multiplexing Four 7-Segment LED Displays with a 4x4 Keypad and 4 A/D Channels

MULTIPLEXING FOUR 7-SEGMENT LED DISPLAYS

Hardware

The PIC16C71's I/O ports have an improved sink/source specification. Each I/O pin can sink up to 25 mA and source 20 mA, in addition total PORTB source current is 100 mA and sink current is 150 mA. PORTA is rated for 50 mA source current and 80 mA sink current. This makes the PIC16C71 ideal for driving 7-segment LEDs. Since the total number of I/O pins is limited to 13, the 8-bit PORTB is used to drive the 4 LEDs, while external sink transistors or MOSFETs are used to sink the digit current (Figure 1). Another alternative is to use ULN2003 open collector sink current drivers, which are available in 16-pin DIP or very small SO-16 packages. Each transistor on the ULN2003 can sink a maximum of 500 mA and the base drive can be directly driven from the PORTA pins.

FIGURE 1: MULTIPLEXING FOUR 7-SEGMENTS LEDs



Software

The multiplexing is achieved by turning on each LED for a 5 μ s duration every 20 μ s. This gives an update rate of 50 Hz, which is quite acceptable to the human eye as a steady display. The 5 μ s time-base is generated by dividing the 4.096 MHz oscillator clock. The internal prescaler is configured to be a divide by 32 and assigned to Timer0. TMR0 is pre-loaded with a value = 96. TMR0 will increment to FFh and then roll over to 00h after a period = $(256-96) \cdot (32 \cdot 4 / 4096000) = 5 \mu$ s.

When TMR0 rolls over, flag bit T0IF flag is set, and because bits T0IE and GIE are enabled, an interrupt is generated.

The software implements a simple timer which increments at a 1 second rate. Every second, the 4 nibble (two 8-bit registers MsdTime and LsdTime) are incremented in a BCD format. The lower 4 bits of LsdTime correspond to the least significant digit (LSD) on the display. The high 4 bits of LsdTime correspond to the second significant digit of the display and so on. Depending on which display is turned on, the corresponding 4-bit BCD value is extracted from either MsdTime or LsdTime, and decoded to a 7-segment display. The TMR0 interrupt is generated at a steady rate of 5 μ s and given an instruction time of 1 μ s. The entire display update program can reside in the interrupt service routine with no chance of getting an interrupt within an interrupt. The Code Listing for this section is in Appendix A.

MULTIPLEXING FOUR 7-SEGMENT LED DISPLAYS AND SCANNING A 4x4 KEYPAD

Hardware

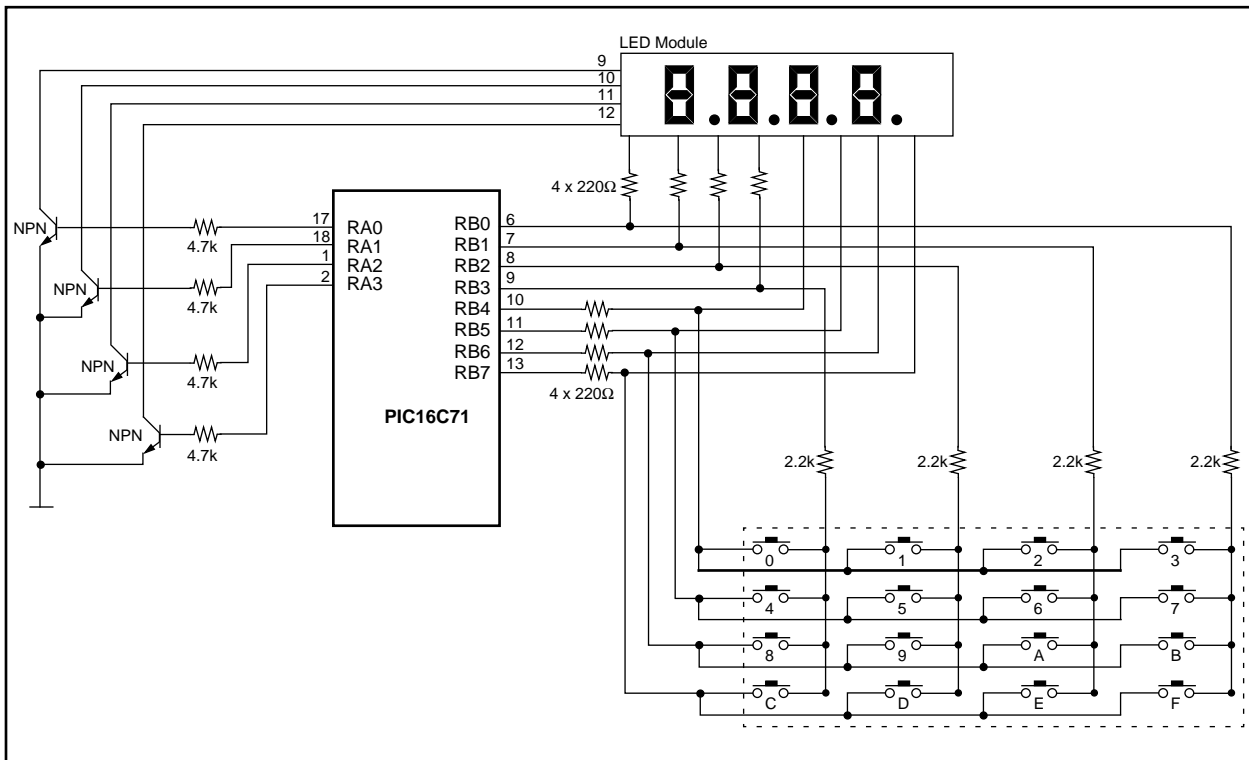
A 4x4 keypad can be very easily interfaced to the PIC16C71's PORTB (Figure 2). Internal pull-ups on pins RB7:RB4 can be enabled/disabled by clearing/setting bit RBPU (OPTION<7>). The internal pull-ups have a value of 20k at 5V (typical). In order to sense a low level at the input, the switch is "connected" to ground through a 2.2 k Ω resistor. A key hit normally lasts anywhere from 50 ms to as long as a person holds the key down. In order not to miss any key hits, the keypad is sampled every 20 μ s (just after the update of the MSD).

Software

To sample the keypad, the digit sinks are first disabled. PORTB is then configured with RB7:RB4 as inputs and RB3:RB0 as outputs driven high. The pull-ups on RB7:RB4 are enabled. Sequentially RB3:RB0 are made low while RB7:RB4 are checked for a key hit (a low level). One key hit per scan is demonstrated in this program. Multiple key hits per scan can very easily be implemented. Once the key hit is sensed, a 40 ms debounce period elapses before key sampling is resumed. No more key hits are sensed until the present key is released. This prevents erroneous key inputs.

The program basically inputs the key hit and displays its value as a hexadecimal character on the multiplexed 7-segment LEDs. The Code Listing for this section is in Appendix B.

FIGURE 2: MULTIPLEXING FOUR 7-SEGMENT LEDS WITH A 4X4 KEYPAD



MULTIPLEXING FOUR 7-SEGMENT LED DISPLAYS AND THE A/D CHANNEL0

Hardware

The four analog channels are connected to RA3:RA0. If any of these pins are used normally as digital I/O, they can momentarily be used as analog inputs. In order to avoid interference from the analog source, it is advisable to buffer the analog input through a voltage follower op-amp, however, it is not always necessary. Figure 3 and Figure 4 show some typical configurations. In this application, the analog input is a potentiometer whose wiper is connected through an RC network to Channel0. The RC is necessary in order to smooth out the analog voltage. The RC does contribute to a delay in the sampling time, however the stability of the analog reading is greatly improved.

Software

The analog input is sampled every 20 ms. The digit sinks and the drivers are turned off (i.e., PORTA is configured as an input and PORTB outputs are made low). A 1 ms settling time is allowed for the external RC network connected to the analog input to settle and then the A/D conversion is started. The result is read then converted from an 8-bit binary value to a 3-digit Binary Code Decimal (BCD) value which is then displayed on the 7-segment LEDs. The Code Listing for this section is in Appendix C.

FIGURE 3: TYPICAL CONNECTION FOR ANALOG/DIGITAL INPUT

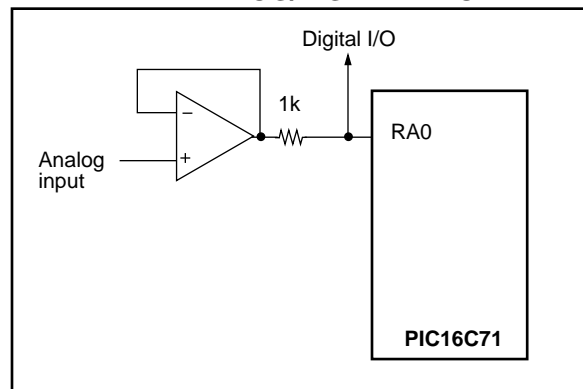
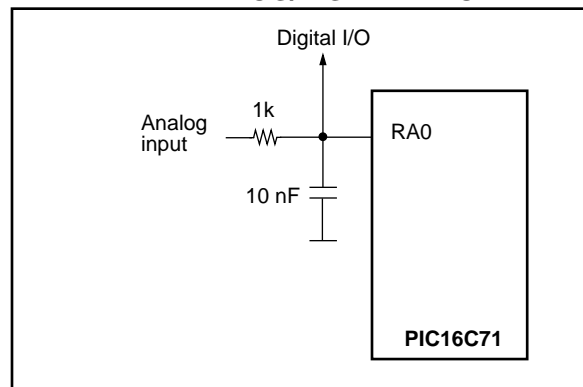


FIGURE 4: TYPICAL CONNECTION FOR ANALOG/DIGITAL INPUT



MULTIPLEXING FOUR 7-SEGMENT LED DISPLAYS WITH A 4x4 KEYPAD AND 4 A/D CHANNELS

Hardware

This section essentially incorporates the previous three sections to give a complete four channel voltmeter. Figure 5 shows a typical configuration. The analog channels are connected through individual potentiometers to their respective analog inputs and are sampled every 20 ms in a round robin fashion. The sampling rate can be increased to as fast as once every 5 μ s if required. The keypad sampling need not be any faster than once every 20 μ s.

Software

The program samples the analog inputs and saves the result in four consecutive locations starting at "ADVALUE", with Channel0 saved at the first location and so on.

KEY 0 \rightarrow Channel0

or

KEY 1 \rightarrow Channel0

Key hits greater than 3 are ignored. The code listing for this section is in Appendix D.

Code Size

Four 7-segment LEDs Program Memory: 139
Data Memory: 6

Four 7-segment LEDs and 4x4 keypad sampling Program Memory: 207
Data Memory: 13

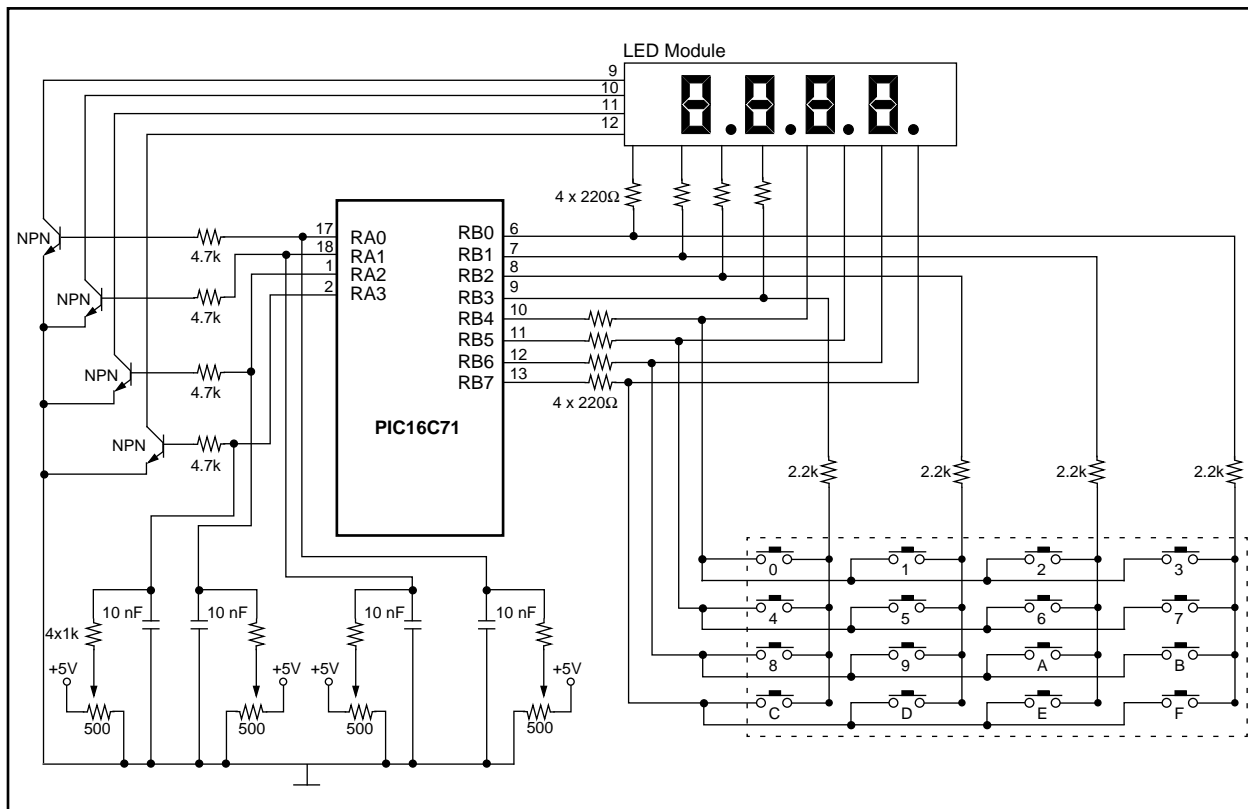
Four 7-segment LEDs and A/D Program Memory: 207
Data Memory: 11

Four 7-segment LEDs, 4x4 keypad sampling, and A/D Program Memory: 207
Data Memory: 13

CONCLUSION

The four A/D channels on the PIC16C71 can be multiplexed with digital I/O, thus reducing overall pin counts and improving I/O pin usage in an analog application.

FIGURE 5: FOUR CHANNEL VOLTMETER WITH DISPLAY AND KEYPAD



Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX A: MPLX.ASM

MPASM 01.40 Released

MPLX.ASM 1-16-1997 16:20:47

PAGE 1

```

LOC  OBJECT CODE      LINE SOURCE TEXT
VALUE

00001 ;*****
00002 ;This program demonstrates how to multiplex four 7 segment LED
00003 ;digits using a PIC16C71. The four digits will start at 0000 and
00004 ;increment at a 1 sec rate up to 9999.
00005 ;The LEDs are updated every 5 mS, for a multiplexing rate of 20 mS.
00006 ;The TMR0 timer is used in internal interrupt mode to generate the
00007 ;5 mS.
00008 ;
00009 ;                               Stan D'Souza 5/8/93
00010 ;
00011 ;       Program:           MPLX.ASM
00012 ;       Revision Date:
00013 ;                               1-15-97       Compatibility with MPASMWIN 1.40
00014 ;
00015 ;*****
00016         LIST P=16C71
00017         ERRORLEVEL  -302
00018 ;
00019         include  <p16c71.inc>
00001         LIST
00002 ; P16C71.INC Standard Header File, Version 1.00 Microchip Technology
00142         LIST
00020 ;
0000000C    00021 TempC equ 0x0c           ;temp general purpose regs
0000000D    00022 TempD equ 0x0d
0000000E    00023 TempE equ 0x0e
0000000F    00024 Count equ 0x0f           ;count
00000010    00025 MsdTime equ 0x10         ;most significant Timer
00000011    00026 LsdTime equ 0x11         ;Least significant Timer
00000001    00027 OptionReg equ 1
00000002    00028 PCL equ 2
00000026    00029 BcdMsdc equ 26
00000027    00030 Bcd equ 27
00031 ;
0000         00032 org 0
0000 2805    00033 goto Start           ;skip over interrupt vector
00034 ;
0004         00035 org 4
0004 281D    00036 goto ServiceInterrupts
00037 ;
0005         00038 Start
0005 2008    00039 call InitPorts
0006 2012    00040 call InitTimers
0007         00041 loop
0007 2807    00042 goto loop
00043 ;
0008         00044 InitPorts
0008 1683    00045 bsf STATUS,RP0           ;select Bank1
0009 3003    00046 movlw 3                   ;make RA0-3 digital I/O
000A 0088    00047 movwf ADCON1           ; /
000B 0185    00048 clrf TRISA              ;make RA0-4 outputs
000C 0186    00049 clrf TRISB              ;make RB0-7 outputs
000D 1283    00050 bcf STATUS,RP0           ;select Bank0
000E 0185    00051 clrf PORTA              ;make all outputs low
000F 0186    00052 clrf PORTB              ; /

```

AN557

```
0010 1585      00053      bsf      PORTA,3      ;enable MSB digit sink
0011 0008      00054      return
00055 ;
00056 ;
00057 ;The clock speed is 4.096Mhz. Dividing internal clk. by a 32 prescaler,
00058 ;the TMR0 will be incremented every 31.25uS. If TMR0 is preloaded
00059 ;with 96, it will take (256-96)*31.25uS to overflow i.e. 5mS. So the
00060 ;end result is that we get a TMR0 interrupt every 5mS.
0012      00061      InitTimers
0012 0190      00062      clr     MsdTime      ;clr timers
0013 0191      00063      clr     LsdTime      ; /
0014 1683      00064      bsf     STATUS,RP0      ;select Bank1
0015 3084      00065      movlw  B'10000100'      ;assign ps to TMR0
0016 0081      00066      movwf  OptionReg      ;ps = 32
0017 1283      00067      bcf     STATUS,RP0      ;select Bank0
0018 3020      00068      movlw  B'00100000'      ;enable TMR0 interrupt
0019 008B      00069      movwf  INTCON      ;
001A 3060      00070      movlw  .96      ;preload TMR0
001B 0081      00071      movwf  TMR0      ;start counter
001C 0009      00072      retfie
00073 ;
001D      00074      ServiceInterrupts
001D 190B      00075      btfsc  INTCON,T0IF      ;TMR0 interrupt?
001E 2822      00076      goto   ServiceTMR0      ;yes then service
001F 3020      00077      movlw  B'00100000'      ;else clr rest
0020 008B      00078      movwf  INTCON
0021 0009      00079      retfie
00080 ;
0022      00081      ServiceTMR0
0022 3060      00082      movlw  .96      ;initialize TMR0
0023 0081      00083      movwf  TMR0
0024 110B      00084      bcf     INTCON,T0IF      ;clr int flag
0025 2028      00085      call  IncTimer      ;inc timer
0026 2050      00086      call  UpdateDisplay      ;update display
0027 0009      00087      retfie
00088 ;
00089 ;The display is incremented every 200*5mS = 1 Sec.
0028      00090      IncTimer
0028 0A0F      00091      incf   Count,W      ;inc count
0029 3AC8      00092      xorlw  .200      ;= 200?
002A 1903      00093      btfsc  STATUS,Z      ;no then skip
002B 282E      00094      goto   DoIncTime      ;else inc time
002C 0A8F      00095      incf   Count, F
002D 0008      00096      return
002E      00097      DoIncTime
002E 018F      00098      clr     Count      ;clr count
002F 0A11      00099      incf   LsdTime,W      ;get lsd
0030 390F      00100      andlw  0x0F      ;mask high nibble
0031 3A0A      00101      xorlw  0x0a      ; = 10?
0032 1903      00102      btfsc  STATUS,Z      ;no then skip
0033 2836      00103      goto   IncSecondLsd      ;inc next lsd
0034 0A91      00104      incf   LsdTime, F      ;else inc timer
0035 0008      00105      return
0036      00106      IncSecondLsd
0036 0E11      00107      swapf  LsdTime,W      ;get hi in low nibble
0037 390F      00108      andlw  0x0F      ;mask hi nibble
0038 3E01      00109      addlw  1      ;inc it
0039 0091      00110      movwf  LsdTime      ;restore back
003A 0E91      00111      swapf  LsdTime, F      ; /
003B 3A0A      00112      xorlw  0x0a      ; = 10?
003C 1903      00113      btfsc  STATUS,Z      ;no then skip
003D 283F      00114      goto   IncThirdLsd      ;else inc next lsd
003E 0008      00115      return
003F      00116      IncThirdLsd
003F 0191      00117      clr     LsdTime
0040 0A10      00118      incf   MsdTime,W      ;get 3rd lsd
```

```

0041 390F          00119          andlw  0x0F          ;mask hi nibble
0042 3A0A          00120          xorlw  0x0a          ;= 10?
0043 1903          00121          btfsc  STATUS,Z      ;no then skip
0044 2847          00122          goto   IncMsd        ;else Msd
0045 0A90          00123          incf   MsdTime, F    ;else inc timer
0046 0008          00124          return
0047              00125          IncMsd
0047 0E10          00126          swapf  MsdTime,W     ;get hi in lo nibble
0048 390F          00127          andlw  0x0F          ;mask hi nibble
0049 3E01          00128          addlw  1              ;inc timer
004A 0090          00129          movwf  MsdTime        ;restore back
004B 0E90          00130          swapf  MsdTime, F    ;
004C 3A0A          00131          xorlw  0x0a          ;= 10?
004D 1903          00132          btfsc  STATUS,Z      ;no then skip
004E 0190          00133          clrf   MsdTime        ;clr msd
004F 0008          00134          return
00135 ;
00136 ;
0050              00137          UpdateDisplay
0050 0805          00138          movf   PORTA,W       ;present sink value in w
0051 0185          00139          clrf   PORTA         ;disable all digits sinks
0052 390F          00140          andlw  0x0f          ;
0053 008C          00141          movwf  TempC         ;save sink value in tempC
0054 160C          00142          bsf   TempC,4        ;preset for lsd sink
0055 0C8C          00143          rrf   TempC, F       ;determine next sink value
0056 1C03          00144          btfss  STATUS,C      ;c=1?
0057 118C          00145          bcf   TempC,3        ;no then reset LSD sink
0058 180C          00146          btfsc  TempC,0       ;else see if Msd
0059 286B          00147          goto   UpdateMsd     ;yes then do Msd
005A 188C          00148          btfsc  TempC,1       ;see if 3rdLsd
005B 2866          00149          goto   Update3rdLsd ;yes then do 3rd Lsd
005C 190C          00150          btfsc  TempC,2       ;see if 2nd Lsd
005D 2861          00151          goto   Update2ndLsd ;yes then do 2nd lsd
005E              00152          UpdateLsd
005E 0811          00153          movf   LsdTime,W     ;get Lsd in w
005F 390F          00154          andlw  0x0f          ;
0060 286F          00155          goto   DisplayOut    ;enable display
0061              00156          Update2ndLsd
0061 2080          00157          call   Chk2LsdZero   ;msd = 0 & 2 lsd 0?
0062 1D03          00158          btfss  STATUS,Z      ;yes then skip
0063 0E11          00159          swapf  LsdTime,W     ;get 2nd Lsd in w
0064 390F          00160          andlw  0x0f          ;mask rest
0065 286F          00161          goto   DisplayOut    ;enable display
0066              00162          Update3rdLsd
0066 2088          00163          call   ChkMsdZero    ;msd = 0?
0067 1D03          00164          btfss  STATUS,Z      ;yes then skip
0068 0810          00165          movf   MsdTime,W     ;get 3rd Lsd in w
0069 390F          00166          andlw  0x0f          ;mask low nibble
006A 286F          00167          goto   DisplayOut    ;enable display
006B              00168          UpdateMsd
006B 0E10          00169          swapf  MsdTime,W     ;get Msd in w
006C 390F          00170          andlw  0x0f          ;mask rest
006D 1903          00171          btfsc  STATUS,Z      ;msd != 0 then skip
006E 300A          00172          movlw  0x0a          ;
006F              00173          DisplayOut
006F 2074          00174          call   LedTable      ;get digit output
0070 0086          00175          movwf  PORTB         ;drive leds
0071 080C          00176          movf   TempC,W       ;get sink value in w
0072 0085          00177          movwf  PORTA         ;
0073 0008          00178          return
00179 ;
00180 ;
0074              00181          LedTable
0074 0782          00182          addwf  PCL, F        ;add to PC low
0075 343F          00183          retlw  B'00111111'   ;led drive for 0
0076 3406          00184          retlw  B'00000110'   ;led drive for 1

```

AN557

```
0077 345B      00185      retlw   B'01011011'      ;led drive for 2
0078 344F      00186      retlw   B'01001111'      ;led drive for 3
0079 3466      00187      retlw   B'01100110'      ;led drive for 4
007A 346D      00188      retlw   B'01101101'      ;led drive for 5
007B 347D      00189      retlw   B'01111101'      ;led drive for 6
007C 3407      00190      retlw   B'00000111'      ;led drive for 7
007D 347F      00191      retlw   B'01111111'      ;led drive for 8
007E 3467      00192      retlw   B'01100111'      ;led drive for 9
007F 3400      00193      retlw   B'00000000'      ;blank led drive
                00194 ;
                00195 ;
0080           00196 Chk2LsdZero
0080 2088      00197      call    ChkMsdZero        ;msd = 0?
0081 1D03      00198      btfsz   STATUS,Z          ;yes then skip
0082 0008      00199      return  ;else return
0083 0E11      00200      swapf   LsdTime,W         ;get 2nd lsd
0084 390F      00201      andlw   0x0f              ;mask of LSD
0085 1D03      00202      btfsz   STATUS,Z          ;0? then skip
0086 0008      00203      return
0087 340A      00204      retlw   .10               ;else return with 10
                00205 ;
0088           00206 ChkMsdZero
0088 0810      00207      movf    MsdTime,W         ;get Msd in w
0089 1D03      00208      btfsz   STATUS,Z          ;= 0? skip
008A 0008      00209      return  ;else return
008B 340A      00210      retlw   .10               ;ret with 10
                00211 ;
                00212
                00213      end
```

MEMORY USAGE MAP ('X' = Used, '-' = Unused)

```
0000 : X--XXXXXXXXXXXXX XXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXX-----
```

All other memory blocks unused.

```
Program Memory Words Used: 137
Program Memory Words Free: 887
Errors : 0
Warnings : 0 reported, 0 suppressed
Messages : 0 reported, 3 suppressed
```


Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX B: MPLXKEY.ASM

MPASM 01.40 Released

MPLXKEY.ASM 1-16-1997 16:24:40

PAGE 1

```

LOC OBJECT CODE      LINE SOURCE TEXT
VALUE

00001 ;*****
00002 ;This program is to demonstrate how to multiplex four 7 segment LED
00003 ;digits and a 4x4 keypad using a PIC16C71.
00004 ;The four digits will start as '0000' and when a key is hit
00005 ;it is displayed on the 7 segment leds as a hex value 0 to F. The last
00006 ;digit hit is always displayed on the right most led with the rest of
00007 ;the digits shifted to the left. The left most digit is deleted.
00008 ;The LEDs are updated every 20mS, the keypad is scanned at a rate of 20
00009 ;mS. The TMR0 timer is used in internal interrupt mode to generate the
00010 ;5 mS.
00011 ;
00012 ;                               Stan D'Souza 5/8/93
00013 ;
00014 ;       Program:           MPLXKEY.ASM
00015 ;       Revision Date:
00016 ;                               1-15-97       Compatibility with MPASMWIN 1.40
00017 ;
00018 ;*****
00019         LIST P=16C71
00020         ERRORLEVEL  -302
00021 ;
00022         include    <p16c71.inc>
00001         LIST
00002 ; P16C71.INC Standard Header File,Ver. 1.00 Microchip Technology, Inc.
00142         LIST
00023 ;
0000000C    00024 TempC   equ    0x0c           ;temp general purpose regs
0000000D    00025 TempD   equ    0x0d
0000000E    00026 TempE   equ    0x0e
00000020    00027 PABuf   equ    0x20
00000021    00028 PBBuf   equ    0x21
0000000F    00029 Count   equ    0x0f           ;count
00000010    00030 MsdTime equ    0x10           ;most significant Timer
00000011    00031 LsdTime equ    0x11           ;Least significant Timer
00000012    00032 KeyFlag equ    0x12           ;flags related to key pad
00000000    00033 keyhit   equ    0              ;bit 0 --> key-press on
00000001    00034 DebnceOn equ    1              ;bit 1 --> debounce on
00000002    00035 noentry equ    2              ;no key entry = 0
00000003    00036 ServKey equ    3              ;bit 3 --> service key
00000013    00037 Debnce   equ    0x13           ;debounce counter
00000014    00038 NewKey   equ    0x14
0000002F    00039 WBuffer  equ    0x2f
0000002E    00040 StatBuffer equ    0x2e
00000001    00041 OptionReg equ    1
00000002    00042 PCL     equ    2
00043 ;
00044 ;
00045 push    macro
00046         movwf   WBuffer           ;save w reg in Buffer
00047         swapf   WBuffer, F         ;swap it
00048         swapf   STATUS,W           ;get status
00049         movwf   StatBuffer         ;save it
00050     endm
00051 ;

```

AN557

```
00052 pop      macro
00053      swapf  StatBuffer,W      ;restore status
00054      movwf  STATUS             ;      /
00055      swapf  WBuffer,W         ;restore W reg
00056      endm
00057 ;
0000      00058      org      0
0000 280D      00059      goto   Start          ;skip over interrupt vector
00060 ;
0004      00061      org      4
00062 ;It is always a good practice to save and restore the w reg,
00063 ;and the status reg during an interrupt.
00064      push
0004 00AF      M      movwf  WBuffer      ;save w reg in Buffer
0005 0EAF      M      swapf  WBuffer, F    ;swap it
0006 0E03      M      swapf  STATUS,W     ;get status
0007 00AE      M      movwf  StatBuffer   ;save it
0008 2036      00065      call   ServiceInterrupts
00066      pop
0009 0E2E      M      swapf  StatBuffer,W  ;restore status
000A 0083      M      movwf  STATUS       ;      /
000B 0E2F      M      swapf  WBuffer,W     ;restore W reg
000C 0009      00067      retfie
00068 ;
000D      00069      Start
000D 2020      00070      call   InitPorts
000E 202A      00071      call   InitTimers
000F      00072      loop
000F 1992      00073      btfsc  KeyFlag,ServKey ;key service pending
0010 2012      00074      call   ServiceKey      ;yes then service
0011 280F      00075      goto   loop
00076 ;
00077 ;ServiceKey, does the software service for a keyhit. After a key
00078 ;service, the ServKey flag is reset, to denote a completed operation.
0012      00079      ServiceKey
0012 0814      00080      movf   NewKey,W      ;get key value
0013 008E      00081      movwf  TempE      ;save in TempE
0014 0E10      00082      swapf  MsdTime,W     ;move MSD out
0015 39F0      00083      andlw  B'11110000'   ;clr lo nibble
0016 0090      00084      movwf  MsdTime      ;save back
0017 0E11      00085      swapf  LsdTime,W     ;get Lsd
0018 390F      00086      andlw  B'00001111'   ;mask off lsd
0019 0490      00087      iorwf  MsdTime, F    ;and left shift 3rd
001A 0E11      00088      swapf  LsdTime,W     ;get Lsd again
001B 39F0      00089      andlw  B'11110000'   ;mask off 2nd
001C 040E      00090      iorwf  TempE,W      ;or with new lsd
001D 0091      00091      movwf  LsdTime      ;make Lsd
001E 1192      00092      bcf   KeyFlag,ServKey ;reset service flag
001F 0008      00093      return
00094
00095 ;
0020      00096      InitPorts
0020 1683      00097      bsf   STATUS,RP0     ;select Bank1
0021 3003      00098      movlw  3              ;make RA0-3 digital I/O
0022 0088      00099      movwf  ADCON1        ;      /
0023 0185      00100      clrf  TRISA          ;make RA0-4 outputs
0024 0186      00101      clrf  TRISB          ;make RB0-7 outputs
0025 1283      00102      bcf   STATUS,RP0     ;select Bank0
0026 0185      00103      clrf  PORTA          ;make all outputs low
0027 0186      00104      clrf  PORTB          ;      /
0028 1585      00105      bsf   PORTA,3        ;enable MSB digit sink
0029 0008      00106      return
00107 ;
00108 ;
00109 ;The clock speed is 4.096Mhz. Dividing internal clk. by a 32 prescaler,
00110 ;the TMR0 will be incremented every 31.25uS. If TMR0 is preloaded
```

```

00111 ;with 96, it will take (256-96)*31.25uS to overflow i.e. 5mS. So the
00112 ;end result is that we get a TMR0 interrupt every 5mS.
00113 InitTimers
002A 002A 0190 00114      clrf    MsdTime          ;clr timers
002B 0191 00115      clrf    LsdTime          ;      /
002C 0192 00116      clrf    KeyFlag        ;clr all flags
002D 1683 00117      bsf    STATUS,RP0      ;select Bank1
002E 3084 00118      movlw  B'10000100'      ;assign ps to TMR0
002F 0081 00119      movwf  OptionReg      ;ps = 32
0030 1283 00120      bcf    STATUS,RP0      ;select Bank0
0031 3020 00121      movlw  B'00100000'      ;enable TMR0 interrupt
0032 008B 00122      movwf  INTCN          ;
0033 3060 00123      movlw  .96             ;preload TMR0
0034 0081 00124      movwf  TMR0          ;start counter
0035 0009 00125      retfie
00126 ;
0036 00127 ServiceInterrupts
0036 190B 00128      btfsc  INTCN,T0IF      ;TMR0 interrupt?
0037 283B 00129      goto   ServiceTMR0      ;yes then service
0038 018B 00130      clrf    INTCN          ;else clr all int
0039 168B 00131      bsf    INTCN,T0IE      ;
003A 0008 00132      return
00133 ;
003B 00134 ServiceTMR0
003B 3060 00135      movlw  .96             ;initialize TMR0
003C 0081 00136      movwf  TMR0          ;
003D 110B 00137      bcf    INTCN,T0IF      ;clr int flag
003E 1805 00138      btfsc  PORTA,0          ;if msb on then do
003F 2042 00139      call   ScanKeys          ;do a quick key scan
0040 20A1 00140      call   UpdateDisplay     ;update display
0041 0008 00141      return
00142 ;
00143 ;
00144 ;ScanKeys, scans the 4X4 keypad matrix and returns a key value in
00145 ;NewKey (0 - F) if a key is pressed, if not it clears the keyhit flag.
00146 ;Debounce for a given keyhit is also taken care of.
00147 ;The rate of key scan is 20mS with a 4.096Mhz clock.
0042 00148 ScanKeys
0042 1C92 00149      btfss  KeyFlag,DebnceOn ;debounce on?
0043 2848 00150      goto   Scan1            ;no then scan keypad
0044 0B93 00151      decfsz Debnce, F      ;else dec debounce time
0045 0008 00152      return                ;not over then return
0046 1092 00153      bcf    KeyFlag,DebnceOn ;over, clr debounce flag
0047 0008 00154      return                ;and return
0048 00155 Scan1
0048 208A 00156      call   SavePorts        ;save port values
0049 30EF 00157      movlw  B'111101111'    ;init TempD
004A 008D 00158      movwf  TempD           ;
004B 00159 ScanNext
004B 0806 00160      movf   PORTB,W          ;read to init port
004C 100B 00161      bcf    INTCN,RBIF      ;clr flag
004D 0C8D 00162      rrf    TempD, F        ;get correct column
004E 1C03 00163      btfss  STATUS,C        ;if carry set?
004F 2862 00164      goto   NoKey           ;no then end
0050 080D 00165      movf   TempD,W          ;else output
0051 0086 00166      movwf  PORTB           ;low column scan line
0052 0000 00167      nop
0053 1C0B 00168      btfss  INTCN,RBIF      ;flag set?
0054 284B 00169      goto   ScanNext        ;no then next
0055 1812 00170      btfsc  KeyFlag,keyhit   ;last key released?
0056 2860 00171      goto   SKreturn        ;no then exit
0057 1412 00172      bsf    KeyFlag,keyhit   ;set new key hit
0058 0E06 00173      swapf  PORTB,W          ;read port
0059 008E 00174      movwf  TempE           ;save in TempE
005A 2064 00175      call   GetKeyValue      ;get key value 0 - F
005B 0094 00176      movwf  NewKey          ;save as New key

```

AN557

```
005C 1592      00177      bsf      KeyFlag,ServKey ;set service flag
005D 1492      00178      bsf      KeyFlag,DebnceOn ;set flag
005E 3004      00179      movlw   4
005F 0093      00180      movwf   Debnce          ;load debounce time
0060           00181      SKreturn
0060 2097      00182      call    RestorePorts   ;restore ports
0061 0008      00183      return
0062           00184      ;
0062           00185      NoKey
0062 1012      00186      bcf      KeyFlag,keyhit ;clr flag
0063 2860      00187      goto    SKreturn
0063           00188      ;
0063           00189      ;GetKeyValue gets the key as per the following layout
0063           00190      ;
0063           00191      ;           Col1      Col2      Col3      Col4
0063           00192      ;           (RB3)     (RB2)     (RB1)     (RB0)
0063           00193      ;
0063           00194      ;Row1(RB4)           0           1           2           3
0063           00195      ;
0063           00196      ;Row2(RB5)           4           5           6           7
0063           00197      ;
0063           00198      ;Row3(RB6)           8           9           A           B
0063           00199      ;
0063           00200      ;Row4(RB7)           C           D           E           F
0063           00201      ;
0064           00202      GetKeyValue
0064 018C      00203      clrfs   TempC
0065 1D8D      00204      btfss   TempD,3          ;first column
0066 286E      00205      goto    RowValEnd
0067 0A8C      00206      incf    TempC, F
0068 1D0D      00207      btfss   TempD,2          ;second col.
0069 286E      00208      goto    RowValEnd
006A 0A8C      00209      incf    TempC, F
006B 1C8D      00210      btfss   TempD,1          ;3rd col.
006C 286E      00211      goto    RowValEnd
006D 0A8C      00212      incf    TempC, F          ;last col.
006E           00213      RowValEnd
006E 1C0E      00214      btfss   TempE,0          ;top row?
006F 2878      00215      goto    GetValCom        ;yes then get 0,1,2&3
0070 1C8E      00216      btfss   TempE,1          ;2nd row?
0071 2877      00217      goto    Get4567          ;yes the get 4,5,6&7
0072 1D0E      00218      btfss   TempE,2          ;3rd row?
0073 2875      00219      goto    Get89ab          ;yes then get 8,9,a&b
0074           00220      Getcdef
0074 150C      00221      bsf      TempC,2          ;set msb bits
0075           00222      Get89ab
0075 158C      00223      bsf      TempC,3          ;           /
0076 2878      00224      goto    GetValCom        ;do common part
0077           00225      Get4567
0077 150C      00226      bsf      TempC,2
0078           00227      GetValCom
0078 080C      00228      movf    TempC,W
0079 0782      00229      addwf   PCL, F
007A 3400      00230      retlw   0
007B 3401      00231      retlw   1
007C 3402      00232      retlw   2
007D 3403      00233      retlw   3
007E 3404      00234      retlw   4
007F 3405      00235      retlw   5
0080 3406      00236      retlw   6
0081 3407      00237      retlw   7
0082 3408      00238      retlw   8
0083 3409      00239      retlw   9
0084 340A      00240      retlw   0a
0085 340B      00241      retlw   0b
0086 340C      00242      retlw   0c
```

```

0087 340D      00243      retlw   0d
0088 340E      00244      retlw   0e
0089 340F      00245      retlw   0f
                00246      ;
                00247      ;SavePorts, saves the porta and portb condition during a key scan
                00248      ;operation.
008A          00249      SavePorts
008A 0805      00250      movf   PORTA,W           ;Get sink value
008B 00A0      00251      movwf  PABuf            ;save in buffer
008C 0185      00252      clrf   PORTA           ;disable all sinks
008D 0806      00253      movf   PORTB,W         ;get port b
008E 00A1      00254      movwf  PBBuf            ;save in buffer
008F 30FF      00255      movlw  0xff            ;make all high
0090 0086      00256      movwf  PORTB           ;on port b
0091 1683      00257      bsf    STATUS,RP0      ;select Bank1
0092 1381      00258      bcf    OptionReg,7     ;enable pull ups
0093 30F0      00259      movlw  B'11110000'     ;port b hi nibble inputs
0094 0086      00260      movwf  TRISB           ;lo nibble outputs
0095 1283      00261      bcf    STATUS,RP0      ;Bank0
0096 0008      00262      return
                00263      ;
                00264      ;RestorePorts, restores the condition of porta and portb after a
                00265      ;key scan operation.
0097          00266      RestorePorts
0097 0821      00267      movf   PBBuf,W         ;get port b
0098 0086      00268      movwf  PORTB           ;
0099 0820      00269      movf   PABuf,W         ;get port a value
009A 0085      00270      movwf  PORTA           ;
009B 1683      00271      bsf    STATUS,RP0      ;select Bank1
009C 1781      00272      bsf    OptionReg,7     ;disable pull ups
009D 0185      00273      clrf   TRISA           ;make port a outputs
009E 0186      00274      clrf   TRISB           ;as well as PORTB
009F 1283      00275      bcf    STATUS,RP0      ;Bank0
00A0 0008      00276      return
                00277      ;
                00278      ;
00A1          00279      UpdateDisplay
00A1 0805      00280      movf   PORTA,W         ;present sink value in w
00A2 0185      00281      clrf   PORTA           ;disable all digits sinks
00A3 390F      00282      andlw  0x0f            ;
00A4 008C      00283      movwf  TempC           ;save sink value in tempC
00A5 160C      00284      bsf    TempC,4         ;preset for lsd sink
00A6 0C8C      00285      rrf    TempC, F        ;determine next sink value
00A7 1C03      00286      btfss  STATUS,C        ;c=1?
00A8 118C      00287      bcf    TempC,3         ;no then reset LSD sink
00A9 180C      00288      btfsc  TempC,0         ;else see if Msd
00AA 28B8      00289      goto   UpdateMsd       ;yes then do Msd
00AB 188C      00290      btfsc  TempC,1         ;see if 3rdLsd
00AC 28B5      00291      goto   Update3rdLsd    ;yes then do 3rd Lsd
00AD 190C      00292      btfsc  TempC,2         ;see if 2nd Lsd
00AE 28B2      00293      goto   Update2ndLsd    ;yes then do 2nd lsd
00AF          00294      UpdateLsd
00AF 0811      00295      movf   LsdTime,W       ;get Lsd in w
00B0 390F      00296      andlw  0x0f            ; /
00B1 28BA      00297      goto   DisplayOut      ;
00B2          00298      Update2ndLsd
00B2 0E11      00299      swapf  LsdTime,W       ;get 2nd Lsd in w
00B3 390F      00300      andlw  0x0f            ;mask rest
00B4 28BA      00301      goto   DisplayOut      ;enable display
00B5          00302      Update3rdLsd
00B5 0810      00303      movf   MsdTime,W       ;get 3rd Lsd in w
00B6 390F      00304      andlw  0x0f            ;mask low nibble
00B7 28BA      00305      goto   DisplayOut      ;enable display
00B8          00306      UpdateMsd
00B8 0E10      00307      swapf  MsdTime,W       ;get Msd in w
00B9 390F      00308      andlw  0x0f            ;mask rest

```

AN557

```
00BA          00309 DisplayOut
00BA 20BF     00310          call   LedTable          ;get digit output
00BB 0086     00311          movwf  PORTB          ;drive leds
00BC 080C     00312          movf   TempC,W       ;get sink value in w
00BD 0085     00313          movwf  PORTA
00BE 0008     00314          return
              00315 ;
              00316 ;
00BF          00317 LedTable
00BF 0782     00318          addwf  PCL, F        ;add to PC low
00C0 343F     00319          retlw  B'00111111'   ;led drive for 0
00C1 3406     00320          retlw  B'00000110'   ;led drive for 1
00C2 345B     00321          retlw  B'01011011'   ;led drive for 2
00C3 344F     00322          retlw  B'01001111'   ;led drive for 3
00C4 3466     00323          retlw  B'01100110'   ;led drive for 4
00C5 346D     00324          retlw  B'01101101'   ;led drive for 5
00C6 347D     00325          retlw  B'01111101'   ;led drive for 6
00C7 3407     00326          retlw  B'00000111'   ;led drive for 7
00C8 347F     00327          retlw  B'01111111'   ;led drive for 8
00C9 3467     00328          retlw  B'01100111'   ;led drive for 9
00CA 3477     00329          retlw  B'01110111'   ;led drive for A
00CB 347C     00330          retlw  B'01111100'   ;led drive for b
00CC 3439     00331          retlw  B'00111001'   ;led drive for C
00CD 345E     00332          retlw  B'01011110'   ;led drive for d
00CE 3479     00333          retlw  B'01111001'   ;led drive for E
00CF 3471     00334          retlw  B'01110001'   ;led drive for F
              00335
              00336          ;
              00337          ;
              00338
00339          end
```

MEMORY USAGE MAP ('X' = Used, '-' = Unused)

```
0000 : X--XXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXXXXXX -----
```

All other memory blocks unused.

Program Memory Words Used: 205
Program Memory Words Free: 819

Errors : 0
Warnings : 0 reported, 0 suppressed
Messages : 0 reported, 6 suppressed

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX C: MPLXCH0.ASM

MPASM 01.40 Released

MPLXCH0.ASM 1-16-1997 16:24:14

PAGE 1

```

LOC OBJECT CODE      LINE SOURCE TEXT
VALUE

00001 ;*****
00002 ;This program is to demonstrate how to multiplex four 7 segment LED
00003 ;and sample ch0 of the a/d in a PIC16C71. The a/d value is displayed
00004 ;as a 3 digit decimal value of the a/d input (0 - 255).
00005 ;The LEDs are updated every 20mS, the a/d is sampled every 20 mS.
00006 ;The TIMER0 timer is used in internal interrupt mode to generate the
00007 ;5 mS.
00008 ;
00009 ;                               Stan D'Souza 5/8/93
00010 ;
00011 ;
00012 ;
00013 ;      Program:          MPLXCH0.ASM
00014 ;      Revision Date:
00015 ;                               1-15-97      Compatibility with MPASMWIN 1.40
00016 ;
00017 ;*****
00018      LIST P=16C71
00019      ERRORLEVEL  -302
00020 ;
00021      include    <p16c71.inc>
00001 LIST
00002 ; P16C71.INC Standard Header File, Ver. 1.00 Microchip Technology, Inc.
00142 LIST
00022 ;
0000026 00023 BcdMsd equ    26
0000027 00024 Bcd      equ    27
0000000C 00025 TempC   equ    0x0c      ;temp general purpose regs
0000000D 00026 TempD   equ    0x0d
0000000E 00027 TempE   equ    0x0e
00000020 00028 PABuf   equ    0x20
00000021 00029 PBBuf   equ    0x21
0000000F 00030 Count   equ    0x0f      ;count
00000010 00031 MsdTime  equ    0x10      ;most significant Timer
00000011 00032 LsdTime  equ    0x11      ;Least significant Timer
00000012 00033 ADFlag   equ    0x12      ;flags related to key pad
00000005 00034 ADOver   equ    5        ;bit 5 --> a/d over
0000002F 00035 WBuffer  equ    0x2f
0000002E 00036 StatBuffer equ  0x2e
00000001 00037 OptionReg equ    1
00000002 00038 PCL     equ    2
00039 ;
00040 push   macro
00041      movwf WBuffer      ;save w reg in Buffer
00042      swapf WBuffer, F   ;swap it
00043      swapf STATUS,W     ;get status
00044      movwf StatBuffer   ;save it
00045      endm
00046 ;
00047 pop   macro
00048      swapf StatBuffer,W ;restore status
00049      movwf STATUS      ; /
00050      swapf WBuffer,W   ;restore W reg
00051      endm

```

AN557

```
0000          00052 ;
0000          00053      org      0
0000 280D     00054      goto     Start          ;skip over interrupt vector
0000          00055 ;
0004          00056      org      4
0004          00057 ;It is always a good practice to save and restore the w reg,
0004          00058 ;and the status reg during an interrupt.
0004 00AF     00059      push
0005 0EAF     M          movwf   WBuffer          ;save w reg in Buffer
0006 0E03     M          swapf   WBuffer, F          ;swap it
0007 00AE     M          swapf   STATUS,W          ;get status
0008 2039     M          movwf   StatBuffer        ;save it
0008          00060      call    ServiceInterrupts
0008          00061      pop
0009 0E2E     M          swapf   StatBuffer,W        ;restore status
000A 0083     M          movwf   STATUS            ; /
000B 0E2F     M          swapf   WBuffer,W          ;restore W reg
000C 0009     00062      retfie
000C          00063 ;
000D          00064 Start
000D 2021     00065      call    InitPorts
000E 202B     00066      call    InitTimers
000F 2036     00067      call    InitAd
0010          00068 loop
0010 1A92     00069      btfsc  ADFlag,ADOver    ;a/d over?
0011 2013     00070      call    UpdateAd        ;yes then update
0012 2810     00071      goto    loop
0012          00072 ;
0013          00073 UpdateAd
0013 1C88     00074      btfss  ADCON0,ADIF     ;a/d done?
0014 0008     00075      return          ;no then leave
0015 0809     00076      movf   ADRES,W          ;get a/d value
0016 00A1     00077      movwf  L_byte
0017 01A0     00078      clrf  H_byte
0018 20AD     00079      call  B2_BCD
0019 0824     00080      movf   R2,W            ;get LSd
001A 0091     00081      movwf  LsdTime         ;save in LSD
001B 0823     00082      movf   R1,W            ;get Msd
001C 0090     00083      movwf  MsdTime         ;save in Msd
001D 1088     00084      bcf   ADCON0,ADIF     ;clr interrupt flag
001E 1008     00085      bcf   ADCON0,ADON     ;turn off a/d
001F 1292     00086      bcf   ADFlag,ADOver   ;clr flag
0020 0008     00087      return
0020          00088 ;
0020          00089 ;
0020          00090 ;
0021          00091 InitPorts
0021 1683     00092      bsf   STATUS,RP0       ;select Bank1
0022 3003     00093      movlw 3                ;make RA0-3 digital I/O
0023 0088     00094      movwf  ADCON1          ; /
0024 0185     00095      clrf  TRISA            ;make RA0-4 outputs
0025 0186     00096      clrf  TRISB            ;make RB0-7 outputs
0026 1283     00097      bcf   STATUS,RP0       ;select Bank0
0027 0185     00098      clrf  PORTA            ;make all outputs low
0028 0186     00099      clrf  PORTB            ; /
0029 1585     00100      bsf   PORTA,3          ;enable MSB digit sink
002A 0008     00101      return
002A          00102 ;
002A          00103 ;
002A          00104 ;The clock speed is 4.096Mhz. Dividing internal clk. by a 32 prescaler,
002A          00105 ;the TMR0 will be incremented every 31.25uS. If TMR0 is preloaded
002A          00106 ;with 96, it will take (256-96)*31.25uS to overflow i.e. 5mS. So the
002A          00107 ;end result is that we get a TMR0 interrupt every 5mS.
002B          00108 InitTimers
002B 0190     00109      clrf  MsdTime          ;clr timers
002C 0191     00110      clrf  LsdTime          ; /
```



```

002D 1683      00111      bsf      STATUS,RP0      ;select Bank1
002E 3084      00112      movlw   B'10000100'     ;assign ps to TMR0
002F 0081      00113      movwf   OptionReg      ;ps = 32
0030 1283      00114      bcf     STATUS,RP0      ;select Bank0
0031 3020      00115      movlw   B'00100000'     ;enable TMR0 interrupt
0032 008B      00116      movwf   INTCN          ;
0033 3060      00117      movlw   .96            ;preload TMR0
0034 0081      00118      movwf   TMR0          ;start counter
0035 0009      00119      retfie
                00120 ;
                00121 ;
0036          00122      InitAd
0036 30C0      00123      movlw   B'11000000'     ;rc osc, ch 0 for a/d
0037 0088      00124      movwf   ADCON0
0038 0008      00125      return
                00126 ;
                00127 ;
0039          00128      ServiceInterrupts
0039 190B      00129      btfsc   INTCN,T0IF     ;TMR0 interrupt?
003A 283E      00130      goto   ServiceTMR0     ;yes then service
003B 018B      00131      clrf   INTCN
003C 168B      00132      bsf     INTCN,T0IE
003D 0008      00133      return
                00134 ;
003E          00135      ServiceTMR0
003E 3060      00136      movlw   .96            ;initialize TMR0
003F 0081      00137      movwf   TMR0
0040 110B      00138      bcf     INTCN,T0IF     ;clr int flag
0041 1C05      00139      btfss   PORTA,0        ;last digit?
0042 2045      00140      call   SampleAd        ;then sample a/d
0043 2071      00141      call   UpdatedDisplay   ;else update display
0044 0008      00142      return
                00143 ;
                00144 ;
0045          00145      SampleAd
0045 205A      00146      call   SavePorts
0046 204C      00147      call   DoAd            ;do a ad conversion
0047          00148      AdDone
0047 1908      00149      btfsc   ADCON0,GO     ;ad done?
0048 2847      00150      goto   AdDone          ;no then loop
0049 1692      00151      bsf     ADFlag,ADOver  ;set a/d over flag
004A 2067      00152      call   RestorePorts    ;restore ports
004B 0008      00153      return
                00154 ;
                00155 ;
004C          00156      DoAd
004C 0186      00157      clrf   PORTB          ;turn off leds
004D 1683      00158      bsf     STATUS,RP0     ;select Bank1
004E 300F      00159      movlw   0x0f          ;make port a hi-Z
004F 0085      00160      movwf   TRISA          ;
0050 1283      00161      bcf     STATUS,RP0     ;select Bank0
0051 1408      00162      bsf     ADCON0,ADON    ;start a/d
0052 307D      00163      movlw   .125
0053 2056      00164      call   Wait
0054 1508      00165      bsf     ADCON0,GO     ;start conversion
0055 0008      00166      return
                00167 ;
                00168 ;
0056          00169      Wait
0056 008C      00170      movwf   TempC          ;store in temp
0057          00171      Next
0057 0B8C      00172      decfsz TempC, F
0058 2857      00173      goto   Next
0059 0008      00174      return
                00175 ;
                00176 ;

```

AN557

```
00177 ;SavePorts, saves the porta and portb condition during a key scan
00178 ;operation.
005A      00179 SavePorts
005A 0805      00180      movf    PORTA,W      ;Get sink value
005B 00A0      00181      movwf   PABuf        ;save in buffer
005C 0185      00182      clrf    PORTA        ;disable all sinks
005D 0806      00183      movf    PORTB,W      ;get port b
005E 00A1      00184      movwf   PBBuf        ;save in buffer
005F 30FF      00185      movlw   0xff         ;make all high
0060 0086      00186      movwf   PORTB        ;on port b
0061 1683      00187      bsf     STATUS,RP0   ;select Bank1
0062 1381      00188      bcf     OptionReg,7  ;enable pull ups
0063 30F0      00189      movlw   B'11110000' ;port b hi nibble inputs
0064 0086      00190      movwf   TRISB        ;lo nibble outputs
0065 1283      00191      bcf     STATUS,RP0   ;Bank0
0066 0008      00192      return
00193 ;
00194 ;RestorePorts, restores the condition of porta and portb after a
00195 ;key scan operation.
0067      00196 RestorePorts
0067 0821      00197      movf    PBBuf,W      ;get port n
0068 0086      00198      movwf   PORTB        ;
0069 0820      00199      movf    PABuf,W      ;get port a value
006A 0085      00200      movwf   PORTA        ;
006B 1683      00201      bsf     STATUS,RP0   ;select Bank1
006C 1781      00202      bsf     OptionReg,7  ;disable pull ups
006D 0185      00203      clrf    TRISA        ;make port a outputs
006E 0186      00204      clrf    TRISB        ;as well as PORTB
006F 1283      00205      bcf     STATUS,RP0   ;Bank0
0070 0008      00206      return
00207 ;
00208 ;
0071      00209 UpdateDisplay
0071 0805      00210      movf    PORTA,W      ;present sink value in w
0072 0185      00211      clrf    PORTA        ;disable all digits sinks
0073 390F      00212      andlw   0x0f         ;
0074 008C      00213      movwf   TempC        ;save sink value in tempC
0075 160C      00214      bsf     TempC,4       ;preset for lsd sink
0076 0C8C      00215      rrf     TempC, F      ;determine next sink value
0077 1C03      00216      btfss   STATUS,C      ;c=1?
0078 118C      00217      bcf     TempC,3       ;no then reset LSD sink
0079 180C      00218      btfsc   TempC,0       ;else see if Msd
007A 288C      00219      goto    UpdateMsd     ;yes then do Msd
007B 188C      00220      btfsc   TempC,1       ;see if 3rdLsd
007C 2887      00221      goto    Update3rdLsd  ;yes then do 3rd Lsd
007D 190C      00222      btfsc   TempC,2       ;see if 2nd Lsd
007E 2882      00223      goto    Update2ndLsd  ;yes then do 2nd lsd
007F      00224 UpdateLsd
007F 0811      00225      movf    LsdTime,W     ;get Lsd in w
0080 390F      00226      andlw   0x0f         ; /
0081 2890      00227      goto    DisplayOut    ;enable display
0082      00228 Update2ndLsd
0082 20A1      00229      call    Chk2LsdZero   ;msd = 0 & 2 lsd 0?
0083 1D03      00230      btfss   STATUS,Z      ;yes then skip
0084 0E11      00231      swapf   LsdTime,W     ;get 2nd Lsd in w
0085 390F      00232      andlw   0x0f         ;mask rest
0086 2890      00233      goto    DisplayOut    ;enable display
0087      00234 Update3rdLsd
0087 20A9      00235      call    ChkMsdZero    ;msd = 0?
0088 1D03      00236      btfss   STATUS,Z      ;yes then skip
0089 0810      00237      movf    MsdTime,W     ;get 3rd Lsd in w
008A 390F      00238      andlw   0x0f         ;mask low nibble
008B 2890      00239      goto    DisplayOut    ;enable display
008C      00240 UpdateMsd
008C 0E10      00241      swapf   MsdTime,W     ;get Msd in w
008D 390F      00242      andlw   0x0f         ;mask rest
```

```

008E 1903          00243          btfsc  STATUS,Z          ;msd != 0 then skip
008F 300A          00244          movlw  0x0a
0090              00245 DisplayOut
0090 2095          00246          call   LedTable          ;get digit output
0091 0086          00247          movwf  PORTB             ;drive leds
0092 080C          00248          movf   TempC,W          ;get sink value in w
0093 0085          00249          movwf  PORTA
0094 0008          00250          return
                    00251 ;
                    00252 ;
0095              00253 LedTable
0095 0782          00254          addwf  PCL, F           ;add to PC low
0096 343F          00255          retlw  B'00111111'      ;led drive for 0
0097 3406          00256          retlw  B'00000110'      ;led drive for 1
0098 345B          00257          retlw  B'01011011'      ;led drive for 2
0099 344F          00258          retlw  B'01001111'      ;led drive for 3
009A 3466          00259          retlw  B'01100110'      ;led drive for 4
009B 346D          00260          retlw  B'01101101'      ;led drive for 5
009C 347D          00261          retlw  B'01111101'      ;led drive for 6
009D 3407          00262          retlw  B'00000111'      ;led drive for 7
009E 347F          00263          retlw  B'01111111'      ;led drive for 8
009F 3467          00264          retlw  B'01100111'      ;led drive for 9
00A0 3400          00265          retlw  B'00000000'      ;blank led drive
                    00266 ;
                    00267 ;
00A1              00268 Chk2LsdZero
00A1 20A9          00269          call   ChkMsZero        ;msd = 0?
00A2 1D03          00270          btfss  STATUS,Z          ;yes then skip
00A3 0008          00271          return                  ;else return
00A4 0E11          00272          swapf  LsdTime,W        ;get 2nd lsd
00A5 390F          00273          andlw  0x0f             ;mask of LSD
00A6 1D03          00274          btfss  STATUS,Z          ;0? then skip
00A7 0008          00275          return
00A8 340A          00276          retlw  .10              ;else return with 10
                    00277 ;
00A9              00278 ChkMsZero
00A9 0810          00279          movf   MsdTime,W        ;get Msd in w
00AA 1D03          00280          btfss  STATUS,Z          ;= 0? skip
00AB 0008          00281          return                  ;else return
00AC 340A          00282          retlw  .10              ;ret with 10
                    00283 ;
                    00284 ;
                    00285 ;
00000026          00286 count  equ        26
00000027          00287 temp   equ        27
                    00288 ;
00000020          00289 H_byte equ        20
00000021          00290 L_byte equ        21
00000022          00291 R0     equ        22          ; RAM Assignments
00000023          00292 R1     equ        23
00000024          00293 R2     equ        24
                    00294 ;
                    00295 ;
00AD 1003          00296 B2_BCD  bcf        STATUS,0      ; clear the carry bit
00AE 3010          00297          movlw  .16
00AF 00A6          00298          movwf  count
00B0 01A2          00299          clrf   R0
00B1 01A3          00300          clrf   R1
00B2 01A4          00301          clrf   R2
00B3 0DA1          00302 loop16 rlf    L_byte, F
00B4 0DA0          00303          rlf    H_byte, F
00B5 0DA4          00304          rlf    R2, F
00B6 0DA3          00305          rlf    R1, F
00B7 0DA2          00306          rlf    R0, F
                    00307 ;
00B8 0BA6          00308          decfsz count, F

```

AN557

```
00B9 28BB      00309      goto    adjDEC
00BA 3400      00310      RETLW  0
                00311      ;
00BB 3024      00312 adjDEC movlw  R2
00BC 0084      00313      movwf  FSR
00BD 20C5      00314      call   adjBCD
                00315      ;
00BE 3023      00316      movlw  R1
00BF 0084      00317      movwf  FSR
00C0 20C5      00318      call   adjBCD
                00319      ;
00C1 3022      00320      movlw  R0
00C2 0084      00321      movwf  FSR
00C3 20C5      00322      call   adjBCD
                00323      ;
00C4 28B3      00324      goto    loop16
                00325      ;
00C5 3003      00326 adjBCD movlw  3
00C6 0700      00327      addwf  0,W
00C7 00A7      00328      movwf  temp
00C8 19A7      00329      btfs   temp,3          ; test if result > 7
00C9 0080      00330      movwf  0
00CA 3030      00331      movlw  30
00CB 0700      00332      addwf  0,W
00CC 00A7      00333      movwf  temp
00CD 1BA7      00334      btfs   temp,7          ; test if result > 7
00CE 0080      00335      movwf  0                ; save as MSD
00CF 3400      00336      RETLW  0
                00337      ;
                00338      ;
                00339
00340      end
```

MEMORY USAGE MAP ('X' = Used, '-' = Unused)

```
0000 : X--XXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXX -----
```

All other memory blocks unused.

Program Memory Words Used: 205
Program Memory Words Free: 819

Errors : 0
Warnings : 0 reported, 0 suppressed
Messages : 0 reported, 7 suppressed

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX D: MPLXAD.ASM

MPASM 01.40 Released

MPLXAD.ASM 1-16-1997 16:23:40

PAGE 1

```

LOC  OBJECT CODE      LINE SOURCE TEXT
VALUE

00001 ;*****
00002 ;This program demonstrates how to multiplex four 7 segment LED
00003 ;digits and a 4X4 keypad along with 4 A/D inputs using a PIC16C71.
00004 ;The four digits will first display the decimal a/d value of ch0.
00005 ;When keys from 0 - 3 are hit the corresponding channel's a/d value
00006 ;is displayed in decimal.
00007 ;The LEDs are updated every 20mS, the keypad is scanned at a rate of 20
00008 ;mS. All 4 channels are scanned at 20mS rate, so each channel gets
00009 ;scanned every 80mS. A faster rate of scanning is possible as required
00010 ;by the users application.
00011 ;Timer0 is used in internal interrupt mode to generate the
00012 ;5 mS.
00013 ;
00014 ;                               Stan D'Souza 5/8/93
00015 ;
00016 ;Corrected error in display routine.
00017 ;                               Stan D'Souza 2/27/94
00018 ;
00019 ;       Program:           MPLXAD.ASM
00020 ;       Revision Date:
00021 ;                               1-15-97       Compatibility with MPASMWIN 1.40
00022 ;
00023 ;*****
00024         LIST P=16C71
00025         ERRORLEVEL -302
00026 ;
00027         include      <p16c71.inc>
00001 LIST
00002 ; P16C71.INC Standard Header File, Ver. 1.00 Microchip Technology, Inc.
00142 LIST
00028 ;
0000000C 00029 TempC equ      0x0c           ;temp general purpose regs
0000000D 00030 TempD equ      0x0d
0000000E 00031 TempE equ      0x0e
00000020 00032 PABuf equ     0x20
00000021 00033 PBBuf equ     0x21
0000000F 00034 Count equ     0x0f           ;count
00000010 00035 MsdTime equ    0x10           ;most significant Timer
00000011 00036 LsdTime equ    0x11           ;Least significant Timer
00037 ;
00000012 00038 Flag equ       0x12           ;general purpose flag reg
00039 #define keyhit Flag,0           ;bit 0 --> key-press on
00040 #define DebnceOn Flag,1         ;bit 1 -> debounce on
00041 #define noentry Flag,2         ;no key entry = 0
00042 #define ServKey Flag,3         ;bit 3 --> service key
00043 #define ADOver Flag,4          ;bit 4 --> a/d conv. over
00044 ;
00000013 00045 Debnce equ     0x13           ;debounce counter
00000014 00046 NewKey equ     0x14
00000015 00047 DisplayCh equ   0x15           ;channel to be displayed
00048 ;
00000016 00049 ADTABLE equ     0x16           ;4 locations are reserved here
00050 ;from 0x16 to 0x19
00051 ;

```

AN557

```
0000002F      00052 WBuffer equ      0x2f
0000002E      00053 StatBuffer equ   0x2e
00000001      00054 OptionReg equ    1
00000002      00055 PCL      equ      2
00056 ;
00057 ;
00058 push      macro
00059      movwf     WBuffer      ;save w reg in Buffer
00060      swapf    WBuffer, F    ;swap it
00061      swapf    STATUS,W      ;get status
00062      movwf    StatBuffer    ;save it
00063      endm
00064 ;
00065 pop        macro
00066      swapf    StatBuffer,W   ;restore status
00067      movwf    STATUS         ;      /
00068      swapf    WBuffer,W     ;restore W reg
00069      endm
00070 ;
0000      00071      org      0
0000 280D      00072      goto     Start      ;skip over interrupt vector
00073 ;
0004      00074      org      4
00075 ;It is always a good practice to save and restore the w reg,
00076 ;and the status reg during a interrupt.
00077      push
0004 00AF      M      movwf    WBuffer      ;save w reg in Buffer
0005 0EAF      M      swapf    WBuffer, F    ;swap it
0006 0E03      M      swapf    STATUS,W      ;get status
0007 00AE      M      movwf    StatBuffer    ;save it
0008 2052      00078      call    ServiceInterrupts
00079      pop
0009 0E2E      M      swapf    StatBuffer,W   ;restore status
000A 0083      M      movwf    STATUS         ;      /
000B 0E2F      M      swapf    WBuffer,W     ;restore W reg
000C 0009      00080      retfie
00081 ;
000D      00082 Start
000D 203B      00083      call    InitPorts
000E 20EE      00084      call    InitAd
000F 2045      00085      call    InitTimers
0010      00086 loop
0010 1992      00087      btfsc   ServKey      ;key service pending
0011 2015      00088      call    ServiceKey    ;yes then service
0012 1A12      00089      btfsc   ADOver      ;a/d pending?
0013 2028      00090      call    ServiceAD     ;yes the service a/d
0014 2810      00091      goto    loop
00092 ;
00093 ;ServiceKey, does the software service for a keyhit. After a key
00094 ;service, the ServKey flag is reset, to denote a completed operation.
00095 ServiceKey
0015 1192      00096      bcf     ServKey      ;reset service flag
0016 0814      00097      movf    NewKey,W     ;get key value
0017 3C03      00098      sublw   3             ;key > 3?
0018 1C03      00099      btfss  STATUS,C      ;no then skip
0019 0008      00100      return    ;else ignore key
001A 0814      00101      movf    NewKey,W     ;load new channel
001B 0095      00102      movwf   DisplayCh
00103 ;
001C      00104 LoadAD
001C 3016      00105      movlw   ADTABLE     ;get top of table
001D 0715      00106      addwf   DisplayCh,W ;add offset
001E 0084      00107      movwf   FSR         ;init FSR
001F 0800      00108      movf    0,W         ;get a/d value
0020 00A1      00109      movwf   L_byte
0021 01A0      00110      clrf   H_byte
```

```

0022 2106      00111      call    B2_BCD
0023 0824      00112      movf   R2,W           ;get LSd
0024 0091      00113      movwf  LsdTime       ;save in LSD
0025 0823      00114      movf   R1,W           ;get Msd
0026 0090      00115      movwf  MsdTime       ;save in Msd
0027 0008      00116      return
00117 ;
00118 ;This routine essentially loads the ADRES value in the table location
00119 ;determined by the channel offset. If channel 0 then ADRES is saved
00120 ;in location ADTABLE. If channel 1 then ADRES is saved at ADTABLE + 1.
00121 ;and so on.
0028      00122 ServiceAD
0028 0808      00123      movf   ADCON0,W       ;get adcon0
0029 008C      00124      movwf  TempC         ;save in temp
002A 3008      00125      movlw  B'00001000'   ;select next channel
002B 0708      00126      addwf  ADCON0,W       ;
002C 1A88      00127      btfsc  ADCON0,5       ;if <= ch3
002D 30C1      00128      movlw  B'11000001'   ;select ch0
002E 0088      00129      movwf  ADCON0
00130 ;now load adres in the table
002F 3016      00131      movlw  ADTABLE
0030 0084      00132      movwf  FSR           ;load FSR with top
0031 0C8C      00133      rrf    TempC, F
0032 0C8C      00134      rrf    TempC, F
0033 0C0C      00135      rrf    TempC,W       ;get in w reg
0034 3903      00136      andlw  3             ;mask off all but last 2
0035 0784      00137      addwf  FSR, F        ;add offset to table
0036 0809      00138      movf   ADRES,W       ;get a/d value
0037 0080      00139      movwf  0             ;load indirectly
0038 1212      00140      bcf    ADOVer        ;clear flag
0039 201C      00141      call  LoadAD        ;load a/d value in display reg.
003A 0008      00142      return
00143
00144
00145
00146 ;
003B      00147 InitPorts
003B 1683      00148      bsf    STATUS,RP0    ;select Bank1
003C 3003      00149      movlw  3             ;make RA0-3 digital I/O
003D 0088      00150      movwf  ADCON1       ;
003E 0185      00151      clrf   TRISA         ;make RA0-4 outputs
003F 0186      00152      clrf   TRISB        ;make RB0-7 outputs
0040 1283      00153      bcf    STATUS,RP0    ;select Bank0
0041 0185      00154      clrf   PORTA        ;make all outputs low
0042 0186      00155      clrf   PORTB        ;
0043 1585      00156      bsf    PORTA,3       ;enable MSB digit sink
0044 0008      00157      return
00158 ;
00159 ;
00160 ;The clock speed is 4.096Mhz. Dividing internal clk. by a 32 prescaler,
00161 ;the TMR0 will be incremented every 31.25uS. If TMR0 is preloaded
00162 ;with 96, it will take (256-96)*31.25uS to overflow i.e. 5mS. So the
00163 ;end result is that we get a TMR0 interrupt every 5mS.
0045      00164 InitTimers
0045 0190      00165      clrf   MsdTime       ;clr timers
0046 0191      00166      clrf   LsdTime       ;
0047 0195      00167      clrf   DisplayCh     ;show channel 0
0048 0192      00168      clrf   Flag          ;clr all flags
0049 1683      00169      bsf    STATUS,RP0    ;select Bank1
004A 3084      00170      movlw  B'10000100'   ;assign ps to TMR0
004B 0081      00171      movwf  OptionReg     ;ps = 32
004C 1283      00172      bcf    STATUS,RP0    ;select Bank0
004D 3020      00173      movlw  B'00100000'   ;enable TMR0 interrupt
004E 008B      00174      movwf  INTCON        ;
004F 3060      00175      movlw  .96           ;preload TMR0
0050 0081      00176      movwf  TMR0         ;start counter

```

AN557

```
0051 0009      00177      retfie
                00178 ;
0052          00179 ServiceInterrupts
0052 190B      00180      btfsc   INTCON,T0IF      ;TMR0 interrupt?
0053 2857      00181      goto    ServiceTMR0     ;yes then service
0054 018B      00182      clrf   INTCON           ;else clr all int
0055 168B      00183      bsf   INTCON,T0IE
0056 0008      00184      return
                00185 ;
0057          00186 ServiceTMR0
0057 3060      00187      movlw  .96              ;initialize TMR0
0058 0081      00188      movwf  TMR0
0059 110B      00189      bcf   INTCON,T0IF     ;clr int flag
005A 1805      00190      btfsc  PORTA,0         ;scan keys every 20 mS
005B 2060      00191      call  ScanKeys         ;when digit 1 is on
005C 1985      00192      btfsc  PORTA,3         ;scan a/d every 20mS
005D 20F1      00193      call  SampleAd         ;when digit 4 is on
005E 20BF      00194      call  UpdateDisplay    ;update display
005F 0008      00195      return
                00196 ;
                00197 ;
                00198 ;ScanKeys, scans the 4x4 keypad matrix and returns a key value in
00199 ;NewKey (0 - F) if a key is pressed, if not it clears the keyhit flag.
00200 ;Debounce for a given keyhit is also taken care of.
00201 ;The rate of key scan is 20mS with a 4.096Mhz clock.
0060          00202 ScanKeys
0060 1C92      00203      btfss  DebnceOn        ;debounce on?
0061 2866      00204      goto   Scan1           ;no then scan keypad
0062 0B93      00205      decfsz Debnce, F      ;else dec debounce time
0063 0008      00206      return                ;not over then return
0064 1092      00207      bcf   DebnceOn        ;over, clr debounce flag
0065 0008      00208      return                ;and return
0066          00209 Scan1
0066 20A8      00210      call  SavePorts        ;save port values
0067 30EF      00211      movlw  B'11101111'    ;init TempD
0068 008D      00212      movwf  TempD
0069          00213 ScanNext
0069 0806      00214      movf   PORTB,W         ;read to init port
006A 100B      00215      bcf   INTCON,RBIF     ;clr flag
006B 0C8D      00216      rrf   TempD, F        ;get correct column
006C 1C03      00217      btfss  STATUS,C       ;if carry set?
006D 2880      00218      goto   NoKey          ;no then end
006E 080D      00219      movf   TempD,W        ;else output
006F 0086      00220      movwf  PORTB          ;low column scan line
0070 0000      00221      nop
0071 1C0B      00222      btfss  INTCON,RBIF     ;flag set?
0072 2869      00223      goto   ScanNext       ;no then next
0073 1812      00224      btfsc  keyhit         ;last key released?
0074 287E      00225      goto   SKreturn       ;no then exit
0075 1412      00226      bsf   keyhit         ;set new key hit
0076 0E06      00227      swapf  PORTB,W        ;read port
0077 008E      00228      movwf  TempE          ;save in TempE
0078 2082      00229      call  GetKeyValue     ;get key value 0 - F
0079 0094      00230      movwf  NewKey         ;save as New key
007A 1592      00231      bsf   ServKey         ;set service flag
007B 1492      00232      bsf   DebnceOn        ;set flag
007C 3004      00233      movlw  4              ;load debounce time
007D 0093      00234      movwf  Debnce
007E          00235 SKreturn
007E 20B5      00236      call  RestorePorts    ;restore ports
007F 0008      00237      return
                00238 ;
0080          00239 NoKey
0080 1012      00240      bcf   keyhit         ;clr flag
0081 287E      00241      goto   SKreturn
                00242 ;
```



```

00243 ;GetKeyValue gets the key as per the following layout
00244 ;
00245 ;                Col1   Col2   Col3   Col4
00246 ;                (RB3)  (RB2)  (RB1)  (RB0)
00247 ;
00248 ;Row1(RB4)        0       1       2       3
00249 ;
00250 ;Row2(RB5)        4       5       6       7
00251 ;
00252 ;Row3(RB6)        8       9       A       B
00253 ;
00254 ;Row4(RB7)        C       D       E       F
00255 ;
0082      00256 GetKeyValue
0082 018C      00257      clrf      TempC
0083 1D8D      00258      btfss     TempD,3      ;first column
0084 288C      00259      goto      RowValEnd
0085 0A8C      00260      incf      TempC, F
0086 1D0D      00261      btfss     TempD,2      ;second col.
0087 288C      00262      goto      RowValEnd
0088 0A8C      00263      incf      TempC, F
0089 1C8D      00264      btfss     TempD,1      ;3rd col.
008A 288C      00265      goto      RowValEnd
008B 0A8C      00266      incf      TempC, F      ;last col.
008C      00267 RowValEnd
008C 1C0E      00268      btfss     TempE,0      ;top row?
008D 2896      00269      goto      GetValCom    ;yes then get 0,1,2&3
008E 1C8E      00270      btfss     TempE,1      ;2nd row?
008F 2895      00271      goto      Get4567      ;yes the get 4,5,6&7
0090 1D0E      00272      btfss     TempE,2      ;3rd row?
0091 2893      00273      goto      Get89ab      ;yes then get 8,9,a&b
0092      00274 Getcdef
0092 150C      00275      bsf      TempC,2      ;set msb bits
0093      00276 Get89ab
0093 158C      00277      bsf      TempC,3      ; /
0094 2896      00278      goto      GetValCom    ;do common part
0095      00279 Get4567
0095 150C      00280      bsf      TempC,2
0096      00281 GetValCom
0096 080C      00282      movf     TempC,W
0097 0782      00283      addwf    PCL, F
0098 3400      00284      retlw   0
0099 3401      00285      retlw   1
009A 3402      00286      retlw   2
009B 3403      00287      retlw   3
009C 3404      00288      retlw   4
009D 3405      00289      retlw   5
009E 3406      00290      retlw   6
009F 3407      00291      retlw   7
00A0 3408      00292      retlw   8
00A1 3409      00293      retlw   9
00A2 340A      00294      retlw   0a
00A3 340B      00295      retlw   0b
00A4 340C      00296      retlw   0c
00A5 340D      00297      retlw   0d
00A6 340E      00298      retlw   0e
00A7 340F      00299      retlw   0f
00300 ;
00301 ;SavePorts, saves the porta and portb condition during a key scan
00302 ;operation.
00A8      00303 SavePorts
00A8 0805      00304      movf     PORTA,W      ;Get sink value
00A9 00A0      00305      movwf    PABuf        ;save in buffer
00AA 0185      00306      clrf     PORTA        ;disable all sinks
00AB 0806      00307      movf     PORTB,W      ;get port b
00AC 00A1      00308      movwf    PBBuf        ;save in buffer

```

AN557

```
00AD 30FF          00309          movlw    0xff          ;make all high
00AE 0086          00310          movwf   PORTB         ;on port b
00AF 1683          00311          bsf     STATUS,RP0    ;select Bank1
00B0 1381          00312          bcf     OptionReg,7   ;enable pull ups
00B1 30F0          00313          movlw   B'11110000'   ;port b hi nibble inputs
00B2 0086          00314          movwf   TRISB         ;lo nibble outputs
00B3 1283          00315          bcf     STATUS,RP0    ;Bank0
00B4 0008          00316          return
00317 ;
00318 ;RestorePorts, restores the condition of porta and portb after a
00319 ;key scan operation.
00B5              00320 RestorePorts
00B5 0821          00321          movf    PBBuf,W       ;get port b
00B6 0086          00322          movwf   PORTB
00B7 0820          00323          movf    PABuf,W       ;get port a value
00B8 0085          00324          movwf   PORTA
00B9 1683          00325          bsf     STATUS,RP0    ;select Bank1
00BA 1781          00326          bsf     OptionReg,7   ;disable pull ups
00BB 0185          00327          clrf   TRISA          ;make port a outputs
00BC 0186          00328          clrf   TRISB         ;as well as PORTB
00BD 1283          00329          bcf     STATUS,RP0    ;Bank0
00BE 0008          00330          return
00331 ;
00332 ;
00BF              00333 UpdateDisplay
00BF 0805          00334          movf    PORTA,W       ;present sink value in w
00C0 0185          00335          clrf   PORTA         ;disable all digits sinks
00C1 390F          00336          andlw  0x0f
00C2 008C          00337          movwf   TempC        ;save sink value in tempC
00C3 160C          00338          bsf     TempC,4       ;preset for lsd sink
00C4 0C8C          00339          rrf     TempC, F     ;determine next sink value
00C5 1C03          00340          btfss  STATUS,C      ;c=1?
00C6 118C          00341          bcf     TempC,3       ;no then reset LSD sink
00C7 180C          00342          btfsc  TempC,0       ;else see if Msd
00C8 28D6          00343          goto   UpdateMsd     ;yes then do Msd
00C9 188C          00344          btfsc  TempC,1       ;see if 3rdLsd
00CA 28D3          00345          goto   Update3rdLsd ;yes then do 3rd Lsd
00CB 190C          00346          btfsc  TempC,2       ;see if 2nd Lsd
00CC 28D0          00347          goto   Update2ndLsd ;yes then do 2nd lsd
00CD              00348 UpdateLsd
00CD 0811          00349          movf    LsdTime,W     ;get Lsd in w
00CE 390F          00350          andlw  0x0f          ; /
00CF 28D8          00351          goto   DisplayOut
00D0              00352 Update2ndLsd
00D0 0E11          00353          swapf  LsdTime,W     ;get 2nd Lsd in w
00D1 390F          00354          andlw  0x0f          ;mask rest
00D2 28D8          00355          goto   DisplayOut    ;enable display
00D3              00356 Update3rdLsd
00D3 0810          00357          movf    MsdTime,W    ;get 3rd Lsd in w
00D4 390F          00358          andlw  0x0f          ;mask low nibble
00D5 28D8          00359          goto   DisplayOut    ;enable display
00D6              00360 UpdateMsd
00D6 0E10          00361          swapf  MsdTime,W     ;get Msd in w
00D7 390F          00362          andlw  0x0f          ;mask rest
00D8              00363 DisplayOut
00D8 20DD          00364          call   LedTable      ;get digit output
00D9 0086          00365          movwf   PORTB        ;drive leds
00DA 080C          00366          movf    TempC,W      ;get sink value in w
00DB 0085          00367          movwf   PORTA
00DC 0008          00368          return
00369 ;
00370 ;
00DD              00371 LedTable
00DD 0782          00372          addwf  PCL, F        ;add to PC low
00DE 343F          00373          retlw  B'00111111'   ;led drive for 0
00DF 3406          00374          retlw  B'00000110'   ;led drive for 1
```

```

00E0 345B      00375      retlw  B'01011011'    ;led drive for 2
00E1 344F      00376      retlw  B'01001111'    ;led drive for 3
00E2 3466      00377      retlw  B'01100110'    ;led drive for 4
00E3 346D      00378      retlw  B'01101101'    ;led drive for 5
00E4 347D      00379      retlw  B'01111101'    ;led drive for 6
00E5 3407      00380      retlw  B'00000111'    ;led drive for 7
00E6 347F      00381      retlw  B'01111111'    ;led drive for 8
00E7 3467      00382      retlw  B'01100111'    ;led drive for 9
00E8 3477      00383      retlw  B'01110111'    ;led drive for A
00E9 347C      00384      retlw  B'01111100'    ;led drive for b
00EA 3439      00385      retlw  B'00111001'    ;led drive for C
00EB 345E      00386      retlw  B'01011110'    ;led drive for d
00EC 3479      00387      retlw  B'01111001'    ;led drive for E
00ED 3471      00388      retlw  B'01110001'    ;led drive for F
00389
00390 ;
00391 ;
00EE          00392 InitAd
00EE 30C0      00393      movlw  B'11000000'    ;internal rc for tad
00EF 0088      00394      movwf  ADCON0         ; /
00395      ;note that adcon1 is set in InitPorts
00F0 0008      00396      return
00397 ;
00F1          00398 SampleAd
00F1 20A8      00399      call   SavePorts
00F2 20F8      00400      call   DoAd           ;do a ad conversion
00F3          00401 AdDone
00F3 1908      00402      btfsc  ADCON0,GO     ;ad done?
00F4 28F3      00403      goto   AdDone        ;no then loop
00F5 1612      00404      bsf    ADOVer        ;set a/d over flag
00F6 20B5      00405      call   RestorePorts  ;restore ports
00F7 0008      00406      return
00407 ;
00408 ;
00F8          00409 DoAd
00F8 0186      00410      clrf   PORTB         ;turn off leds
00F9 1683      00411      bsf    STATUS,RP0    ;select Bank1
00FA 300F      00412      movlw  0x0f          ;make port a hi-Z
00FB 0085      00413      movwf  TRISA         ; /
00FC 1283      00414      bcf    STATUS,RP0    ;select Bank0
00FD 1408      00415      bsf    ADCON0,ADON   ;start a/d
00FE 307D      00416      movlw  .125
00FF 2102      00417      call   Wait
0100 1508      00418      bsf    ADCON0,GO     ;start conversion
0101 0008      00419      return
00420 ;
00421 ;
0102          00422 Wait
0102 008C      00423      movwf  TempC         ;store in temp
0103          00424 Next
0103 0B8C      00425      decfsz TempC, F
0104 2903      00426      goto   Next
0105 0008      00427      return
00428
00429 ;
00430 ;
00000026      00431 count equ 26
00000027      00432 temp  equ 27
00433 ;
00000020      00434 H_byte equ 20
00000021      00435 L_byte equ 21
00000022      00436 R0    equ 22      ; RAM Assignments
00000023      00437 R1    equ 23
00000024      00438 R2    equ 24
00439 ;
00440 ;

```

AN557

```
0106 1003      00441 B2_BCD bcf      STATUS,0      ; clear the carry bit
0107 3010      00442      movlw   .16
0108 00A6      00443      movwf  count
0109 01A2      00444      clrf   R0
010A 01A3      00445      clrf   R1
010B 01A4      00446      clrf   R2
010C 0DA1      00447 loop16 rlf   L_byte, F
010D 0DA0      00448      rlf   H_byte, F
010E 0DA4      00449      rlf   R2, F
010F 0DA3      00450      rlf   R1, F
0110 0DA2      00451      rlf   R0, F
              00452 ;
0111 0BA6      00453      decfsz count, F
0112 2914      00454      goto  adjDEC
0113 3400      00455      RETLW 0
              00456 ;
0114 3024      00457 adjDEC movlw  R2
0115 0084      00458      movwf  FSR
0116 211E      00459      call  adjBCD
              00460 ;
0117 3023      00461      movlw  R1
0118 0084      00462      movwf  FSR
0119 211E      00463      call  adjBCD
              00464 ;
011A 3022      00465      movlw  R0
011B 0084      00466      movwf  FSR
011C 211E      00467      call  adjBCD
              00468 ;
011D 290C      00469      goto  loop16
              00470 ;
011E 3003      00471 adjBCD movlw  3
011F 0700      00472      addwf  0,W
0120 00A7      00473      movwf  temp
0121 19A7      00474      btfsc  temp,3      ; test if result > 7
0122 0080      00475      movwf  0
0123 3030      00476      movlw  30
0124 0700      00477      addwf  0,W
0125 00A7      00478      movwf  temp
0126 1BA7      00479      btfsc  temp,7      ; test if result > 7
0127 0080      00480      movwf  0      ; save as MSD
0128 3400      00481      RETLW 0
              00482 ;
              00483 ;
              00484 ;
              00485 ;
              00486
00487      end
```

MEMORY USAGE MAP ('X' = Used, '-' = Unused)

```
0000 : X---XXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0100 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXX-----
```

All other memory blocks unused.

Program Memory Words Used: 294
Program Memory Words Free: 730

Errors : 0
Warnings : 0 reported, 0 suppressed
Messages : 0 reported, 7 suppressed

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office

Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602-786-7200 Fax: 602-786-7277
Technical Support: 602 786-7627
Web: <http://www.microchip.com>

Atlanta

Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston

Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508-480-9990 Fax: 508-480-8575

Chicago

Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas

Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 972-991-7177 Fax: 972-991-8588

Dayton

Microchip Technology Inc.
Two Prestige Place, Suite 150
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Los Angeles

Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 714-263-1888 Fax: 714-263-1338

New York

Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516-273-5305 Fax: 516-273-5335

San Jose

Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto

Microchip Technology Inc.
5925 Airport Road, Suite 200
Mississauga, Ontario L4V 1W1, Canada
Tel: 905-405-6279 Fax: 905-405-6253

ASIA/PACIFIC

Hong Kong

Microchip Asia Pacific
RM 3801B, Tower Two
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2-401-1200 Fax: 852-2-401-3431

India

Microchip Technology India
No. 6, Legacy, Convent Road
Bangalore 560 025, India
Tel: 91-80-229-0061 Fax: 91-80-229-0062

Korea

Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Shanghai

Microchip Technology
RM 406 Shanghai Golden Bridge Bldg.
2077 Yan'an Road West, Hongjiao District
Shanghai, PRC 200335
Tel: 86-21-6275-5700
Fax: 86 21-6275-5060

Singapore

Microchip Technology Taiwan
Singapore Branch
200 Middle Road
#10-03 Prime Centre
Singapore 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan, R.O.C

Microchip Technology Taiwan
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2-717-7175 Fax: 886-2-545-0139

EUROPE

United Kingdom

Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44-1628-851077 Fax: 44-1628-850259

France

Arizona Microchip Technology SARL
Zone Industrielle de la Bonde
2 Rue du Buisson aux Fraises
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany

Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 München, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy

Arizona Microchip Technology SRL
Centro Direzionale Colleone
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-39-6899939 Fax: 39-39-6899883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shin Yokohama
Kohoku-Ku, Yokohama
Kanagawa 222 Japan
Tel: 81-4-5471- 6166 Fax: 81-4-5471-6122

5/8/97



MICROCHIP

All rights reserved. © 1997, Microchip Technology Incorporated, USA. 6/97

Information contained in this publication regarding device applications and the like is intended for suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.