

Frequency Counter Using PIC16C5X

Author: Stan D'Souza
Microchip Technology Inc.

INTRODUCTION

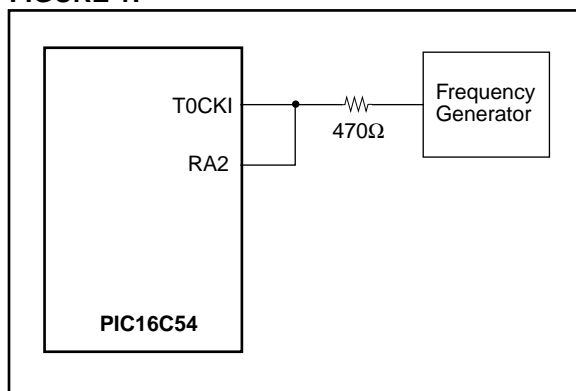
The PIC16C5X has one 8-bit timer (Timer0), which can be used with an 8-bit prescaler. The prescaler runs asynchronously, hence it can count a very high frequency. The minimum rise and fall times of the input frequency are specified to be 10 ns, so the fastest clock rate the TMR0 can count is 50 MHz. The prescaler must be used when measuring high frequency. Since the prescaler can be configured as a divide by 256 counter, the maximum resolution at which the input frequency can be measured is 16-bits. However, the prescaler cannot be directly read like a file register. This application note depicts a unique method by which the user can "extract" the 8-bit value in the prescaler, whereby the resolution of the measurement is 16-bits with the high 8-bits in TMR0 and the low 8-bits in the prescaler.

IMPLEMENTATION

A frequency counter which can read frequencies from 50 Hz to 50 MHz was implemented in this application note in order to demonstrate this method of measuring the 16-bit counter value from the prescaler and TMR0.

The basic hardware for the measurement circuit is depicted in Figure 1. It consists of the frequency input at TMR0 or T0CKI (pin 3 in a PIC16C54). T0CKI is connected to RA2. The input frequency is connected to T0CKI through a 470Ω resistor.

FIGURE 1:



TMR0 is configured to measure the input frequency, at T0CKI of the PIC16C54. The input frequency is "gated" for a precise duration of time. Before starting this precise "gate", TMR0 is cleared (which also clears the prescaler), and the RA2 pin is configured as an input. The precise "gate" is implemented in software as an accurate delay. At the end of the delay, the RA2 pin is configured as an output going low. This will cause the input to TMR0 to be "halted" or "stopped". A 16-bit value of the input frequency is now saved in TMR0 and the 8-bit prescaler. The high 8 bits are in TMR0 and can be easily read. The low 8 bits have to be "shifted out". The 8 bits in the prescaler are "shifted out" by toggling RA2 with a "BSF" and a "BCF" instruction. After every toggle, the value in TMR0 is checked to see if TMR0 has incremented. If the number of toggles required to cause TMR0 to increment by 1 is N, then the 8-bit value in the prescaler can be calculated to be = (256 - N). By concatenating the calculated value and the original value from TMR0, the 16-bit value for the frequency is determined.

To measure a wide range of frequencies, the following intermediate steps were taken:

Frequency Range	Precise "gate" delay	Resolution
50 MHz - 10 MHz	1 ms	±10 kHz
10 MHz - 1 MHz	5 ms	±2 kHz
1 MHz - 100 kHz	50 ms	±200 Hz
100 Hz - 10 kHz	200 ms	±50 Hz
50 Hz - 50 Hz	50 ms (†)	±2 Hz

Note: In this case, TMR0 uses the internal 4 MHz clock and counts the number of instances of the external clock. The maximum time required is 50 ms to make a ± 2 Hz accurate measurement for 10 kHz input frequency.

The check for the correct frequency is performed automatically starting with the high frequency and ending with the low frequency. The maximum time required for each conversion is approximately 310 ms. In other words, three frequency checks are done every second.

CONCLUSION

The PIC16C5X family can be used to make a 16-bit measurement of input frequency with a small overhead of one resistor and one I/O port.

AN592

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: FREQ.ASM

MPASM 01.40 Released

FREQ.ASM 1-16-1997 17:29:41

PAGE 1

```
LOC OBJECT CODE      LINE SOURCE TEXT
VALUE

00001      list p=16C54
00002 ;
00003      include "p16c5x.inc"
00001      LIST
00002 ;P16C5X.INC Standard Header File, Version 3.30 Microchip Technology, Inc.
00224      LIST
00004
00005 #define          _ra0          PORTA,0
00006 #define          _ra1          PORTA,1
00007
00008 ;
00009 ;This program implements the concepts for the frequency counter
00010 ;using a PIC16C54. In this program, RA0 is connected directly
00011 ;to the tmr0 input. Tmr0 input is connected thru a 470 ohm
00012 ;resistor to the freq source. Please note that the
00013 ;the input freq. is required to be a 50% duty cycle, square
00014 ;wave. Though none of the internal calculations are based
00015 ;on this requirement, waveforms which deviate drastically
00016 ;for the one specified were not tested using these routines.
00017 ;The routines written in this program, automatically measure
00018 ;waveforms from 50MHz to 50hz in a period of approx. 300 mS.
00019 ;After a period of approx 300 mS, the 16 bit "measured" value of
00020 ;the freq. is read and saved in the location "flo" and "fhi".
00021 ;A "range" flag is set to indicate if the measurement belongs to
00022 ;the five ranges measured namely:
00023 ;      RANGE:          Flag name
00024 ;      50Mhz to 10Mhz --> Mhz 50 to 10
00025 ;      10Mhz to 1Mhz  --> Mhz 10 to 1
00026 ;      1Mhz to 100Khz --> Khz 1K to 100
00027 ;      100Khz to 10Khz --> Khz 100 to 10
00028 ;      10Khz to 50hz  --> Hz 10K to 50
00029 ;The freq. check is repeated to give approx 3 samples/sec.
00030 ;The "measured" value now has to go through a calculation to
00031 ;get the actual value. Please use the math routines mentioned
00032 ;elsewhere in the Embedded Control Handbook to determine
00033 ;the actual value of the freq.
00034 ;*****
00035 ;Calculations required to determine actual freq. values
00036 ;*****
00037 ;First determine which range flag is set, then calculate as follows:
00038 ;
00039 ;      Mhz50to10:  freq. = (fhi|flo) X 1000
00040 ;      Mhz10to1:   freq. = (fhi|flo) X 200
00041 ;      Khz1Kto100: freq. = (fhi|flo) X 20
00042 ;      Khz100to10: freq. = (fhi|flo) X 5
00043 ;      Hz10Kto50: Please see comments above routine Freq10Kto50
00044 ;
00045 ;
00046 ;      Program:          FREQ.ASM
00047 ;      Revision Date:
00048 ;                      1-16-97          Compatibility with MPASMWIN 1.40
00049 ;
00050 ;*****
00051 ;
```

```

0000000B 00052 fhi      equ          .11          ;high 8 bit value for freq.
0000000A 00053 flo      equ          .10          ;low 8 bit value for freq.
0000000C 00054 tempa     equ          .12
0000000D 00055 tempb     equ          .13
0000000D 00056 limithi   equ          .13
0000000C 00057 limitlo   equ          .12
0000000D 00058 count     equ          .13
0000000E 00059 trisabuf  equ          .14
00000010 00060 InputCounthi equ          .16
0000000F 00061 InputCountlo equ          .15
00000011 00062 #define ddra0 trisabuf,0
00000011 00063 RangeFlag equ          .17
00000011 00064 #define Mhz50to10 RangeFlag,0
00000011 00065 #define Mhz10to1 RangeFlag,1
00000011 00066 #define Khz1Kto100 RangeFlag,2
00000011 00067 #define Khz100to10 RangeFlag,3
00000011 00068 #define Hzl0Kto50 RangeFlag,4
00000011 00069 #define RangeError RangeFlag,5
00000011 00070 ;
00002710 00071 tenMhz     equ          .10000000/.1000
00001388 00072 oneMhz     equ          .1000000/.200
00001388 00073 hndredK equ          .100000/.20
000007D0 00074 tenKhz     equ          .10000/.5
00000001 00075 ;
00000001 00076 Debug     equ          1
00000001 00077 ;
00000001 00078 enabletmr0 macro
00000001 00079         clrfs          TMR0
00000001 00080         bsfs          ddra0
00000001 00081         movfs         trisabuf,W
00000001 00082         tris         PORTA
00000001 00083         endm
00000001 00084 ;
00000001 00085 disabletmr0 macro
00000001 00086         bcf          ddra0
00000001 00087         bcf          _ra0
00000001 00088         movfs         trisabuf,W
00000001 00089         tris         PORTA
00000001 00090         endm
00000001 00091 ;
01FF      00092         org          0x1fff
01FF 0A00 00093         goto         start
0000      00094         org          0
0000      00095 start
0000 0C0F 00096         movlw        0x0f          ;initialize ddra
0001 002E 00097         movwf        trisabuf        ; /
0000      00098         disabletmr0
0002 040E          M         bcf          ddra0
0003 0405          M         bcf          _ra0
0004 020E          M         movfs         trisabuf,W
0005 0005          M         tris         PORTA
0006 0C37 00099         movlw        B'00110111'      ;set the option register
0007 0002 00100         option          ;to measure high freq.
0008 0066 00101         clrfs          PORTB
0009 0040 00102         clrws          PORTB
000A 0006 00103         tris         PORTB
0000      00104
000B      00105 repeat
0000      00106         enabletmr0          ;enable tmr0
000B 0061          M         clrfs          TMR0
000C 050E          M         bsfs          ddra0
000D 020E          M         movfs         trisabuf,W
000E 0005          M         tris         PORTA
000F 09BA 00107         call          delay1mS      ;wait for 1mS
0000      00108         disabletmr0        ;disable tmr0
0010 040E          M         bcf          ddra0

```

AN592

```
0011 0405      M      bcf          _ra0
0012 020E      M      movf        trisabuf,W
0013 0005      M      tris          PORTA
0014 09E1      00109    call          getfreq      ;get freq in fhi and flo
0015 097C      00110    call          check10M     ;check if <= 10 Mhz
0016 0743      00111    btfss        STATUS,Z     ;yes then do lower freq.
0017 0A9F      00112    goto         Freq50Mto10M  ;found 50Mhz to 10Mhz freq.
0018 0061      00113    enabletmr0
0019 050E      M      clr         TMR0
001A 020E      M      bsf         ddra0
001B 0005      M      movf        trisabuf,W
001C 09C3      00114    call          delay5mS     ;wait for 5mS
001D 040E      00115    disabletmr0 ;disable tmr0
001E 0405      M      bcf         ddra0
001F 020E      M      movf        _ra0
0020 0005      M      movf        trisabuf,W
0021 09E1      00116    call          PORTA
0022 0990      00117    call          getfreq      ;get freq in fhi and flo
0023 0743      00118    call          check1M      ;check if <= 1 Mhz
0024 0AA2      00119    btfss        STATUS,Z     ;yes then do lower freq.
0025 0061      00120    goto         Freq10Mto1M   ;else wait for 300 mS
0026 050E      M      enabletmr0 ;enable tmr0
0027 020E      M      clr         TMR0
0028 0005      M      bsf         ddra0
0029 09CD      00121    call          delay50mS    ;wait for 50mS
002A 040E      00122    disabletmr0 ;disable tmr0
002B 0405      M      bcf         ddra0
002C 020E      M      movf        trisabuf,W
002D 0005      M      tris          PORTA
002E 09E1      00123    call          delay50mS    ;wait for 50mS
002F 0995      00124    call          getfreq      ;get freq in fhi and flo
0030 0743      00125    call          check100K    ;check if <= 100 Khz
0031 0AA5      00126    btfss        STATUS,Z     ;yes then do lower freq.
0032 0061      00127    goto         Freq1Mto100K  ;else wait for 250 mS
0033 050E      M      enabletmr0 ;enable tmr0
0034 020E      M      clr         TMR0
0035 0005      M      bsf         ddra0
0036 09D7      00128    call          delay200mS   ;wait for 200 mS
0037 040E      00129    disabletmr0 ;disable tmr0
0038 0405      M      bcf         ddra0
0039 020E      M      movf        trisabuf,W
003A 0005      M      tris          PORTA
003B 09E1      00130    call          delay200mS   ;wait for 200 mS
003C 099A      00131    call          getfreq      ;get freq in fhi and flo
003D 0743      00132    call          check10K     ;check if <= 10Khz
003E 0AA8      00133    btfss        STATUS,Z     ;yes then do lower freq.
00134 ;
00135 ;*****
00136 ;The freq. below 10khz to 50hz is got by using the input freq.
00137 ;to gate the internal 4Mhz clock. The gate is not "opened"
00138 ;until a leading or falling transition is observed at the input.
00139 ;For approx. 50 mS, the internal 1uS clock is sourced to
00140 ;the TMR0 with a divide by 256 prescaler. Every 20uS or so,
00141 ;the transitions on the input line are checked. If a transition
00142 ;is observed, then the "InputCount" is incremented. At the end of 50mS,
00143 ;a last transition is used to close the gate and stop the measurement
00144 ;of the internal freq.
00145 ;Say the input freq to be measured is 1500hz. In 50mS, approx 75
00146 ;cycles will be counted in InputCount. The 16 bit value in flo
00147 ;and fhi is approx. 50,000. Then the freq measured:
```

```

00148 ;
00149 ;          freq. = 75 X 1,000,000/60,000 = 1500 in this case
00150 ; In general  freq. = InputCount X 1,000,000/(fhi|flo).
00151 ;
003F 00152 Freq10Kto50
003F 0070 00153      clrf          InputCountHi      ;0 --> InputCount
0040 006F 00154      clrf          InputCountLo      ;      /
0041 0C17 00155      movlw       B'00010111'        ;start TMR0 with internal
0042 0002 00156      option      ; clk. = 1uS
0043 0C0F 00157      movlw       B'00001111'        ;set RA0 as a input
0044 0005 00158      tris        PORTA             ;      /
0045 0705 00159      btfss      _ra0             ;see if level low
0046 0A49 00160      goto        FirstHigh         ;yes then check leading edge
0047      00161 FirstLow
0047 0605 00162      btfsc      _ra0             ;else look for falling edge
0048 0A47 00163      goto        FirstLow          ;      /
0049      00164 FirstHigh
0049 0705 00165      btfss      _ra0             ;and look for first high
004A 0A49 00166      goto        FirstHigh         ;look for first high
004B 0061 00167      clrf          TMR0             ;      /
004C 0CC3 00168      movlw       high .50000        ;start count
004D 002D 00169      movwf      limithi          ;get high byte of 50000
004E      00170 NextLow
004E 0201 00171      movf        TMR0,W             ;save in RAM
004F 008D 00172      subwf      limithi,W          ;50mS over?
0050 0643 00173      btfsc      STATUS,Z          ;approx. 50
0051 0A65 00174      goto        LastHigh         ;no then skip
0052 0605 00175      btfsc      _ra0             ;look for lasthigh
0053 0A4E 00176      goto        NextLow          ;look for low
0054      00177 NextHigh
0054 0201 00178      movf        TMR0,W             ;      /
0055 008D 00179      subwf      limithi,W          ;50mS over?
0056 0643 00180      btfsc      STATUS,Z          ;approx. 50
0057 0A5E 00181      goto        LastLow         ;no then skip
0058 0705 00182      btfss      _ra0             ;look for lastlow
0059 0A54 00183      goto        NextHigh        ;look for lasthigh
005A 02AF 00184      incf       InputCountLo, F ;look for lastlow
005B 0643 00185      btfsc      STATUS,Z          ;inc count
005C 02B0 00186      incf       InputCountHi, F ;overflow?
005D 0A4E 00187      goto        NextLow         ;inc high value
005E      00188 LastLow
005E 0201 00189      movf        TMR0,W             ;check next
005F 002C 00190      movwf      tempa             ;tmr0 overflow?
0060 02AC 00191      incf       tempa, F          ;      /
0061 0643 00192      btfsc      STATUS,Z          ;      /
0062 0A6C 00193      goto        CloseGate       ;no then skip
0063 0605 00194      btfsc      _ra0             ;overflow then abort
0064 0A5E 00195      goto        LastLow         ;look for low
0065      00196 LastHigh
0065 0201 00197      movf        TMR0,W             ;      /
0066 002C 00198      movwf      tempa             ;tmr0 overflow?
0067 02AC 00199      incf       tempa, F          ;      /
0068 0643 00200      btfsc      STATUS,Z          ;      /
0069 0A6C 00201      goto        CloseGate       ;no then skip
006A 0705 00202      btfss      _ra0             ;overflow then abort
006B 0A65 00203      goto        LastHigh        ;look for high
006C      00204 CloseGate
006C 0C27 00205      movlw      B'00100111'        ;look for high
006D 0002 00206      option      ;stop internal clk
006D      00207 disableTmr0 ;      /
006E 040E      00207      disableTmr0 ;disable tmr0
006E 040E      M      bcf          ddra0
006F 0405      M      bcf          _ra0
0070 020E      M      movf         trisabuf,W
0071 0005      M      tris         PORTA
0072 09E1 00208      call       getfreq          ;get freq
0073 028B 00209      incf       fhi,W           ;out of range?

```

AN592

```
0074 0643    00210      btfsc      STATUS,Z      ; /
0075 0A79    00211      goto      OutofRange  ;yes then set flag
0076 0071    00212      clrf      RangeFlag   ;set Hz10Kto50 flag
0077 0591    00213      bsf      Hz10Kto50    ; /
0078 0AAB    00214      goto      wait50mS
0079          00215 OutofRange
0079 0071    00216      clrf      RangeFlag   ;set error flag
007A 05B1    00217      bsf      RangeError
007B 0AAB    00218      goto      wait50mS
00219 ;
00220 ;Check10M, check if the freq < 10 Mhz if yes then the z bit
00221 ;is set else it is cleared. This routine uses a generic routine
00222 ;checklimit, which check the value in fhi and flo to the ones
00223 ;in limithi and limitlo
007C          00224 check10M
007C 0C27    00225      movlw     high tenMhz   ;get hi value of 10Mhz
007D 002D    00226      movwf     limithi      ;save in limithi
007E 0C10    00227      movlw     low tenMhz   ;get lo value of 10Mhz
007F 002C    00228      movwf     limitlo     ;save in limitlo
00229 ;checklimit, checks if the freq in flo and fhi is lower
00230 ;than the values set in limitlo and limithi. It is a
00231 ;common routine used to check all set limits. If the value
00232 ;is <= the z bit = 0 else z = 1 .
0080          00233 checklimit
0080 020B    00234      movf      fhi,W        ;get high freq value
0081 00AD    00235      subwf     limithi, F    ;and check with high value
0082 0643    00236      btfsc     STATUS,Z     ;if not equal then skip
0083 0A88    00237      goto      chk10Mlo    ;else check low value
0084 0703    00238      btfs     STATUS,C      ;skip if value is < limit
0085 0800    00239      retlw    0            ;value > limit so z = 0.
0086 0040    00240      clr      ;z = 1
0087 0800    00241      retlw    0            ;return with z flag set
0088          00242 chk10Mlo
0088 020A    00243      movf     flo,W        ;get low value
0089 00AC    00244      subwf     limitlo, F   ;and check with low value
008A 0643    00245      btfsc     STATUS,Z     ;not equal then skip
008B 0800    00246      retlw    0            ;else return with z = 1
008C 0703    00247      btfs     STATUS,C      ;skip if value is < limit
008D 0800    00248      retlw    0            ;value > limit so z = 0
008E 0040    00249      clr      ; z = 1
008F 0800    00250      retlw    0            ;return with z flag set
00251 ;
00252 ;Check1M checks if freq is below 1 Mhz
00253 ;
0090          00254 check1M
0090 0C13    00255      movlw     high oneMhz  ;get hi value of 1Mhz
0091 002D    00256      movwf     limithi      ;save in limithi
0092 0C88    00257      movlw     low oneMhz   ;get lo value of 1Mhz
0093 002C    00258      movwf     limitlo     ;save in limitlo
0094 0A80    00259      goto      checklimit
00260 ;
0095          00261 check100K
0095 0C13    00262      movlw     high hndredK ;get hi value of 100Khz
0096 002D    00263      movwf     limithi      ;save in limithi
0097 0C88    00264      movlw     low hndredK  ;get lo value of 100Khz
0098 002C    00265      movwf     limitlo     ;save in limitlo
0099 0A80    00266      goto      checklimit
00267 ;
009A          00268 check10K
009A 0C07    00269      movlw     high tenKhz  ;get hi value of 10Khz
009B 002D    00270      movwf     limithi      ;save in limithi
009C 0CD0    00271      movlw     low tenKhz   ;get lo value of 10Khz
009D 002C    00272      movwf     limitlo     ;save in limitlo
009E 0A80    00273      goto      checklimit
00274 ;
00275 ;
```

```

009F          00276 Freq50Mto10M
009F 0071     00277      clrf          RangeFlag
00A0 0511     00278      bsf           Mhz50to10
00A1 0AAB     00279      goto          wait300mS
00A2          00280 Freq10Mto1M
00A2 0071     00281      clrf          RangeFlag
00A3 0531     00282      bsf           Mhz10to1
00A4 0AAB     00283      goto          wait300mS
00A5          00284 Freq1Mto100K
00A5 0071     00285      clrf          RangeFlag
00A6 0551     00286      bsf           Khz1Kto100
00A7 0AAB     00287      goto          wait250mS
00A8          00288 Freq100Kto10K
00A8 0071     00289      clrf          RangeFlag
00A9 0571     00290      bsf           Khz100to10
00AA 0AAB     00291      goto          wait50mS
              00292 ;
00AB          00293 wait300mS
              00294      If          !Debug
              00295      call          delay50mS
              00296      ENDIF
00AB          00297 wait250mS
              00298      IF          !Debug
              00299      call          delay50mS
              00300      call          delay50mS
              00301      call          delay50mS
              00302      call          delay50mS
              00303      ENDIF
00AB          00304 wait50mS
              00305      IF          !Debug
              00306      call          delay50mS
              00307      ENDIF
              00308 ;
              00309 ;
              00310      IF          Debug
00AB          00311 ;This routine debugs freq. on a PICDEM1 board.
              00312 checkRA1
00AB 0625     00313      btfs          _ral
00AC 0AAB     00314      goto          checkRA1
00AD 09D7     00315      call          delay200mS
00AE 020B     00316      movf          fhi,W
00AF 0026     00317      movwf         PORTB
00B0          00318 chkRA1hi
00B0 0725     00319      btfs          _ral
00B1 0AB0     00320      goto          chkRA1hi
00B2          00321 chkRA1lo
00B2 0625     00322      btfs          _ral
00B3 0AB2     00323      goto          chkRA1lo
00B4 09D7     00324      call          delay200mS
00B5 020A     00325      movf          flo,W
00B6 0026     00326      movwf         PORTB
00B7 0725     00327      btfs          _ral
00B8 0AB7     00328      goto          $-1
              00329      ENDIF
00B9 0A0B     00330      goto          repeat
              00331 ;
              00332 ;delay1mS, is a very accurate 1mS delay for a 4Mhz clock.
00BA          00333 delay1mS
00BA 0CC5     00334      movlw         .197
00BB 002D     00335      movwf         count
00BC 0000     00336      nop
00BD 0ABE     00337      goto          $+1
00BE 0ABF     00338      goto          $+1
00BF          00339 dly1mS
00BF 0AC0     00340      goto          $+1
00C0 02ED     00341      decfsz        count, F

```

AN592

```
00C1 0ABF 00342 goto dly1mS
00C2 0800 00343 retlw 0
00344 ;
00345 ;delay5mS uses delay1mS to get a very accurate 5 mS delay
00C3 00346 delay5mS
00C3 09BA 00347 call delay1mS
00C4 09BA 00348 call delay1mS
00C5 09BA 00349 call delay1mS
00C6 09BA 00350 call delay1mS
00C7 09BA 00351 call delay1mS
00C8 0C04 00352 movlw .4
00C9 002D 00353 movwf count
00CA 00354 tweek5mS
00CA 02ED 00355 decfsz count, F
00CB 0ACA 00356 goto tweek5mS
00CC 0800 00357 return
00358 ;
00359 ;delay50mS uses delay1mS to get a very accurate 50mS delay
00CD 00360 delay50mS
00CD 0C32 00361 movlw .50
00CE 002C 00362 movwf tempa
00CF 00363 dly50mS
00CF 09BA 00364 call delay1mS
00D0 02EC 00365 decfsz tempa, F
00D1 0ACF 00366 goto dly50mS
00D2 0C0E 00367 movlw .14
00D3 002D 00368 movwf count
00D4 00369 tweek50mS
00D4 02ED 00370 decfsz count, F
00D5 0AD4 00371 goto tweek50mS
00D6 0800 00372 retlw 0
00373 ;
00374 ;delay200mS uses delay1mS to get a very accurate 200mS delay.
00D7 00375 delay200mS
00D7 0CC8 00376 movlw .200
00D8 002C 00377 movwf tempa
00D9 00378 dly200mS
00D9 09BA 00379 call delay1mS
00DA 02EC 00380 decfsz tempa, F
00DB 0AD9 00381 goto dly200mS
00DC 0C40 00382 movlw .64
00DD 002D 00383 movwf count
00DE 00384 tweek200mS
00DE 02ED 00385 decfsz count, F
00DF 0ADE 00386 goto tweek200mS
00E0 0800 00387 retlw 0
00388 ;
00389 ;getfreq, toggles the RA0 pin to shift out the value in the
00390 ;prescaler. The number of toggles is kept in count. If the value
00391 ;in tmr0 increments, then the low 8 bit value = !count + 1. The low
00392 ;value of the freq. is loaded in flo and the high in fhi.
00393 getfreq
00E1 00394 movf TMR0,W ;get the tmr0 value
00E2 002B 00395 movwf fhi ;save in fhi
00E3 006D 00396 clrf count ;keep track of the toggles
00E4 00397 toggle
00E4 02AD 00398 incf count, F ;inc for first
00E5 0405 00399 bcf _ra0 ;toggle the input
00E6 0505 00400 bsf _ra0 ; /
00E7 0201 00401 movf TMR0,W ;see if tmr0 incremented
00E8 008B 00402 subwf fhi,W ; /
00E9 0643 00403 btfsz STATUS,Z ;yes then skip
00EA 0AE4 00404 goto toggle ;no then toggle again
00EB 026D 00405 comf count, F ;else complement count
00EC 028D 00406 incf count,W ;and increment
00ED 002A 00407 movwf flo ;save in flo
```



```
00EE 0800    00408      retlw      0                ;return
              00409 ;
              00410      end
```

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

```
0000 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX- -----
01C0 : -----X
```

All other memory blocks unused.

```
Program Memory Words Used: 240
Program Memory Words Free: 272
```

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

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. 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.

Trademarks


The Microchip name and logo, the Microchip logo, FilterLab, KEELOQ, microID, MPLAB, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

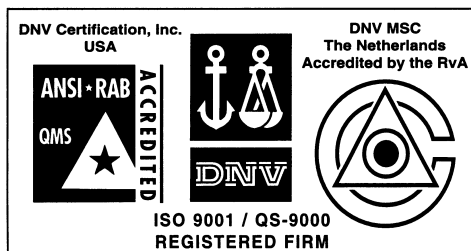
dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2002, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.



Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.



MICROCHIP

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: <http://www.microchip.com>

Rocky Mountain

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

Atlanta

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

Boston

2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago

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

Dallas

4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit

Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo

2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles

18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York

150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose

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

Toronto

6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia

Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing

Microchip Technology Consulting (Shanghai)
Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu

Microchip Technology Consulting (Shanghai)
Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

China - Fuzhou

Microchip Technology Consulting (Shanghai)
Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Shanghai

Microchip Technology Consulting (Shanghai)
Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen

Microchip Technology Consulting (Shanghai)
Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

Hong Kong

Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India

Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan

Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea

Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore

Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan

Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark

Microchip Technology Nordic ApS
Regus Business Centre
Lautrup høj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France

Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - 1er Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany

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

Italy

Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom

Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

01/18/02