

## Frequency Counter Using PIC16C5X

*Author: Stan D'Souza  
Logic Products Division*

### INTRODUCTION

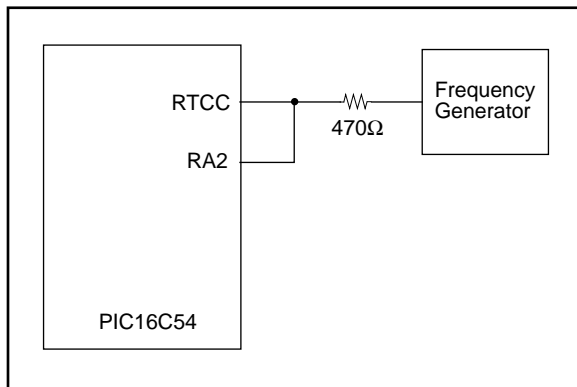
The PIC16C5X has one 8-bit timer (RTCC), which can be used with an 8-bit prescaler. The prescaler runs asynchronously, hence it can count a very high frequency. The minimum rise and fall times of the input frequency are specified to be 10 ns, so the fastest clock rate the RTCC can count is 50 MHz. The 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 the RTCC 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 RTCC.

The basic hardware for the measurement circuit is depicted in Figure 1. It consists of the frequency input at RTCC or RA4 (pin 3 in a PIC16C54). RA4 is connected to RA2. The input frequency is connected to RTCC through a 470Ω resistor.

**FIGURE 1:**



The RTCC is configured to measure the input frequency, at RA4 of the PIC16C54. The input frequency is "gated" for a precise duration of time. Before starting this precise "gate", RTCC 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 RTCC to be "halted" or "stopped". A 16-bit value of the input frequency is now saved in RTCC and the 8-bit prescaler. The high 8 bits are in RTCC 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 RTCC is checked to see if RTCC has incremented. If the number of toggles required to cause RTCC 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 RTCC, 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 kHz - 10 kHz	200 ms	±50 Hz
50 kHz - 50 Hz	50 ms†	±2 Hz

† In this case, the RTCC 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.

## APPENDIX A

MPASM 01.30 Released

FREQ.ASM 12-12-1995 22:47:23

PAGE 1

```
LOC OBJECT CODE      LINE SOURCE TEXT
VALUE

00001          list p=16C54,f=inhx8m
00002 ;
00003          include "pic5X.h"
00001 ;This is the common header file for all PIC16C5X parts.
00002 ;
00003 ;
00004          CBLOCK 0x00
00000000      00005          _indf, _tmr0, _pcl, _status, _fsr
00000005      00006          _porta, _portb
00007          ENDC
00008 ;
00009 ; Porta Bits
00010 #define          _ra0          _porta,0
00011 #define          _ra1          _porta,1
00012 #define          _ra2          _porta,2
00013 #define          _ra3          _porta,3
00014
00015
00016 ; Portb bits
00017 #define          _rb0          _portb,0
00018 #define          _rb1          _portb,1
00019 #define          _rb2          _portb,2
00020 #define          _rb3          _portb,3
00021 #define          _rb4          _portb,4
00022 #define          _rb5          _portb,5
00023 #define          _rb6          _portb,6
00024 #define          _rb7          _portb,7
00025 ;
00026 ; STATUS Reg Bits
00027 #define          _carry        _status,0
00028 #define          _c           _status,0
00029 #define          _dc          _status,1
00030 #define          _z           _status,2
00031 #define          _pd         _status,3
00032 #define          _to         _status,4
00033 #define          _pa0        _status,5
00034 #define          _pal        _status,6
00035 ;
00004 ;
00005 ;This program implements the concepts for the frequency counter
00006 ;using a PIC16C54. In this program, RA0 is connected directly
00007 ;to the tmr0 input. Tmr0 input is connected thro a 470 ohm
00008 ;resistor to the freq source. Please note that the
00009 ;the input freq. is required to be a 50% duty cycle, square
00010 ;wave. Though none of the internal calculations are based
00011 ;on this requirement, wave forms which deviates drastically
00012 ;for the one specified were not tested using these routines.
00013 ;The routines written in this program, automatically measure
00014 ;waveforms from 50MHz to 50hz in a period of approx. 300 mS.
00015 ;After a period of approx 300 mS, the 16 bit "measured" value of
00016 ;the freq. is read and saved in the location "flo" and "fhi".
00017 ;A "range" flag is set to indicate if the measurement belongs to
00018 ;the five ranges measured namely:
```

```

LOC  OBJECT CODE      LINE SOURCE TEXT
VALUE

00019 ;      RANGE:                Flag name
00020 ;      50Mhz to 10Mhz --> Mhz50to10
00021 ;      10Mhz to 1Mhz  --> Mhz10to1
00022 ;      1Mhz to 100Khz --> Khz1Kto100
00023 ;      100Khz to 10Khz --> Khz100to10
00024 ;      10Khz to 50hz  --> Hz10Kto50
00025 ;The freq. check is repeated to give approx 3 samples/sec.
00026 ;The "measured" value now has to go through a calculation to
00027 ;get the actual value. Please use the math routines mentioned
00028 ;elsewhere in the Embedded Control Handbook to determine
00029 ;the actual value of the freq.
00030 ;*****
00031 ;Calculations required to determine actual freq. values
00032 ;*****
00033 ;First determine which range flag is set, then calculate as follows:
00034 ;
00035 ;      Mhz50to10: freq. = (fhi|flo) X 1000
00036 ;      Mhz10to1:  freq. = (fhi|flo) X 200
00037 ;      Khz1Kto100: freq. = (fhi|flo) X 20
00038 ;      Khz100to10: freq. = (fhi|flo) X 5
00039 ;      Hz10Kto50: Please see comments above routine Freq10Kto50
00040 ;
00041 ;
00042 ;      Program:          FREQ.ASM
00043 ;      Revision Date:
00044 ;          12-12-95      Compatibility with MPASMWIN 1.30
00045 ;
00046 ;*****
00047 ;
0000000B 00048 fhi    equ    .11          ;high 8 bit value for freq.
0000000A 00049 flo    equ    .10          ;low 8 bit value for freq.
0000000C 00050 tempa  equ    .12
0000000D 00051 tempb  equ    .13
0000000D 00052 limithi equ    .13
0000000C 00053 limitlo equ    .12
0000000D 00054 count  equ    .13
0000000E 00055 trisabuf equ    .14
00000010 00056 InputCounthi equ    .16
0000000F 00057 InputCountlo equ    .15
00058 #define ddra0 trisabuf,0
00000011 00059 RangeFlag equ    .17
00060 #define Mhz50to10 RangeFlag,0
00061 #define Mhz10to1  RangeFlag,1
00062 #define Khz1Kto100 RangeFlag,2
00063 #define Khz100to10 RangeFlag,3
00064 #define Hz10Kto50 RangeFlag,4
00065 #define RangeError RangeFlag,5
00066 ;
00002710 00067 tenMhz    equ    .10000000/.1000
00001388 00068 oneMhz     equ    .1000000/.200
00001388 00069 hndredK    equ    .100000/.20
000007D0 00070 tenKhz    equ    .10000/.5
00071 ;

```

# AN592

```
LOC OBJECT CODE      LINE SOURCE TEXT
VALUE

00000001            00072 Debug  equ    1
00073 ;
00074 enabletmr0    macro
00075             clrfs  _tmr0
00076             bsfs  ddra0
00077             movfs trisabuf,w
00078             tris  _porta
00079             endm
00080 ;
00081 disabletmr0    macro
00082             bcsfs ddra0
00083             bcsfs _ra0
00084             movfs trisabuf,w
00085             tris  _porta
00086             endm
00087 ;
01FF                00088             org    0x1fff
01FF 0A00            00089             goto   start
0000                00090             org    0
0000                00091 start
0000 0C0F            00092             movlw  0x0f           ;initialize ddra
0001 002E            00093             movwf  trisabuf       ; /
00094             disabletmr0
0002 040E            M             bcsfs  ddra0
0003 0405            M             bcsfs  _ra0
0004 020E            M             movfs  trisabuf,w
0005 0005            M             tris  _porta
0006 0C37            00095             movlw  B'00110111'   ;set the option register
0007 0002            00096             option ;to measure high freq.
0008 0066            00097             clrfs  _portb
0009 0040            00098             clrw
000A 0006            00099             tris  _portb
00100
000B                00101 repeat
00102             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            00103             call  delay1mS       ;wait for 1mS
00104             disabletmr0         ;disable tmr0
0010 040E            M             bcsfs  ddra0
0011 0405            M             bcsfs  _ra0
0012 020E            M             movfs  trisabuf,w
0013 0005            M             tris  _porta
0014 09E1            00105             call  getfreq        ;get freq in fhi and flo
0015 097C            00106             call  check10M       ;check if <= 10 Mhz
0016 0743            00107             btfs  _z             ;yes then do lower freq.
0017 0A9F            00108             goto  Freq50Mto10M   ;found 50Mhz to 10Mhz freq.
00109             enabletmr0           ;enable tmr0
0018 0061            M             clrfs  _tmr0
0019 050E            M             bsfs  ddra0
001A 020E            M             movfs  trisabuf,w
```

```

LOC  OBJECT CODE      LINE SOURCE TEXT
VALUE

001B 0005           M      tris   _porta
001C 09C3           00110  call   delay5mS      ;wait for 5mS
                                00111  disabletmr0        ;disable tmr0

001D 040E           M      bcf   ddra0
001E 0405           M      bcf   _ra0
001F 020E           M      movf  trisabuf,w
0020 0005           M      tris   _porta
0021 09E1           00112  call   getfreq      ;get freq in fhi and flo
0022 0990           00113  call   check1M      ;check if <= 1 Mhz
0023 0743           00114  btfss  _z            ;yes then do lower freq.
0024 0AA2           00115  goto   Freq10Mto1M  ;else wait for 300 mS
                                00116  enabletmr0        ;enable tmr0

0025 0061           M      clrf  _tmr0
0026 050E           M      bsf   ddra0
0027 020E           M      movf  trisabuf,w
0028 0005           M      tris   _porta
0029 09CD           00117  call   delay50mS    ;wait for 50mS
                                00118  disabletmr0        ;disable tmr0

002A 040E           M      bcf   ddra0
002B 0405           M      bcf   _ra0
002C 020E           M      movf  trisabuf,w
002D 0005           M      tris   _porta
002E 09E1           00119  call   getfreq      ;get freq in fhi and flo
002F 0995           00120  call   check100K    ;check if <= 100 Khz
0030 0743           00121  btfss  _z            ;yes then do lower freq.
0031 0AA5           00122  goto   Freq1Mto100K ;else wait for 250 mS
                                00123  enabletmr0        ;enable tmr0

0032 0061           M      clrf  _tmr0
0033 050E           M      bsf   ddra0
0034 020E           M      movf  trisabuf,w
0035 0005           M      tris   _porta
0036 09D7           00124  call   delay200mS   ;wait for 200 mS
                                00125  disabletmr0        ;disable tmr0

0037 040E           M      bcf   ddra0
0038 0405           M      bcf   _ra0
0039 020E           M      movf  trisabuf,w
003A 0005           M      tris   _porta
003B 09E1           00126  call   getfreq      ;get freq in fhi and flo
003C 099A           00127  call   check10K     ;check if <= 10Khz
003D 0743           00128  btfss  _z            ;yes then do lower freq.
003E 0AA8           00129  goto   Freq100Kto10K ;else wait 50mS
                                00130  ;
                                00131  ;*****
00132 ;The freq. below 10khz to 50hz is got by using the input freq.
00133 ;to gate the internal 4Mhz clock. The gate is not "opened"
00134 ;until a leading or falling transition is observed at the input.
00135 ;For approx. 50 mS, the internal 1uS clock is sourced to
00136 ;the TMR0 with a divide by 256 prescaler. Every 20uS or so,
00137 ;the transitions on the input line are checked. If a transition
00138 ;is observed, then the "InputCount" is incremented. At the end of 50mS,
00139 ;a last transition is used to close the gate and stop the measurement
00140 ;of the internal freq.
00141 ;Say the input freq to be measured is 1500hz. In 50mS, approx 75

```

# AN592

```
LOC OBJECT CODE      LINE SOURCE TEXT
VALUE

00142 ;cycles will be counted in InputCount. The 16 bit value in flo
00143 ;and fhi is approx. 50,000. Then the freq measured:
00144 ;
00145 ;          freq. = 75 X 1,000,000/60,000 = 1500 in this case
00146 ; In general  freq. = InputCount X 1,000,000/(fhi|flo).
00147 ;
003F      00148 Freq10Kto50
003F 0070      00149      clrf   InputCounthi   ;0 --> InputCount
0040 006F      00150      clrf   InputCountlo   ; /
0041 0C17      00151      movlw  B'00010111'      ;start TMR0 with internal
0042 0002      00152      option ; clk. = 1uS
0043 0C0F      00153      movlw  B'00001111'      ;set RA0 as a input
0044 0005      00154      tris   _porta          ; /
0045 0705      00155      btfss  _ra0            ;see if level low
0046 0A49      00156      goto   FirstHigh       ;yes then check leading edge
0047
0047 0605      00157      btfsc  _ra0            ;else look for falling edge
0048 0A47      00159      goto   FirstLow        ; /
0049
0049 0705      00160      FirstHigh              ;and look for first high
0049 0705      00161      btfss  _ra0            ;look for first high
004A 0A49      00162      goto   FirstHigh       ; /
004B 0061      00163      clrf   _tmr0           ;start count
004C 0CC3      00164      movlw  high .50000     ;get high byte of 50000
004D 002D      00165      movwf  limithi         ;save in RAM
004E
004E 0201      00166      NextLow
004E 0201      00167      movf   _tmr0,w         ;50mS over?
004F 008D      00168      subwf  limithi,w       ;approx. 50
0050 0643      00169      btfsc  _z              ;no then skip
0051 0A65      00170      goto   LastHigh        ;look for lasthigh
0052 0605      00171      btfsc  _ra0            ;look for low
0053 0A4E      00172      goto   NextLow         ; /
0054
0054 0201      00173      NextHigh
0054 0201      00174      movf   _tmr0,w         ;50mS over?
0055 008D      00175      subwf  limithi,w       ;approx. 50
0056 0643      00176      btfsc  _z              ;no then skip
0057 0A5E      00177      goto   LastLow         ;look for lastlow
0058 0705      00178      btfss  _ra0            ; /
0059 0A54      00179      goto   NextHigh
005A 02AF      00180      incf   InputCountlo, F ;inc count
005B 0643      00181      btfsc  _z              ;overflow?
005C 02B0      00182      incf   InputCounthi, F ;inc high value
005D 0A4E      00183      goto   NextLow         ;check next
005E
005E 0201      00184      LastLow
005E 0201      00185      movf   _tmr0,w         ;tmr0 overflow?
005F 002C      00186      movwf  tempa           ; /
0060 02AC      00187      incf   tempa, F       ; /
0061 0643      00188      btfsc  _z              ;no then skip
0062 0A6C      00189      goto   CloseGate       ;overflow then abort
0063 0605      00190      btfsc  _ra0            ;look for low
0064 0A5E      00191      goto   LastLow         ; /
0065
0065 0201      00192      LastHigh
0065 0201      00193      movf   _tmr0,w         ;tmr0 overflow?
0066 002C      00194      movwf  tempa           ; /
```

```

LOC  OBJECT CODE      LINE SOURCE TEXT
VALUE

0067 02AC          00195      incf   tempa, F      ;      /
0068 0643          00196      btfsc  _z           ;no then skip
0069 0A6C          00197      goto   CloseGate    ;overflow then abort
006A 0705          00198      btfss  _ra0         ;look for high
006B 0A65          00199      goto   LastHigh
006C          00200      CloseGate
006C 0C27          00201      movlw  B'00100111'  ;stop internal clk
006D 0002          00202      option ;           ;      /
          00203      disable tmr0       ;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          00204      call   getfreq      ;get freq
0073 028B          00205      incf   fhi,w        ;out of range?
0074 0643          00206      btfsc  _z           ;      /
0075 0A79          00207      goto   OutofRange   ;yes then set flag
0076 0071          00208      clrf   RangeFlag    ;set Hz10Kto50 flag
0077 0591          00209      bsf    Hz10Kto50    ;      /
0078 0AAB          00210      goto   wait50mS
0079          00211      OutofRange
0079 0071          00212      clrf   RangeFlag    ;set error flag
007A 05B1          00213      bsf    RangeError
007B 0AAB          00214      goto   wait50mS
          00215      ;
          00216      ;Check10M, check if the freq < 10 Mhz if yes then the z bit
          00217      ;is set else it is cleared. This routine uses a generic routine
          00218      ;checklimit, which check the value in fhi and flo to the ones
          00219      ;in limithi and limitlo
007C          00220      check10M
007C 0C27          00221      movlw  high tenMhz  ;get hi value of 10Mhz
007D 002D          00222      movwf  limithi      ;save in limithi
007E 0C10          00223      movlw  low tenMhz   ;get lo value of 10Mhz
007F 002C          00224      movwf  limitlo      ;save in limitlo
          00225      ;checklimit, checks if the freq in flo and fhi is lower
          00226      ;than the values set in limitlo and limithi. It is a
          00227      ;common routine used to check all set limits. If the value
          00228      ;is <= the z bit = 0 else z = 1 .
0080          00229      checklimit
0080 020B          00230      movf   fhi,w        ;get high freq value
0081 00AD          00231      subwf  limithi, F   ;and check with high value
0082 0643          00232      btfsc  _z           ;if not equal then skip
0083 0A88          00233      goto   chk10Mlo     ;else check low value
0084 0703          00234      btfss  _c           ;skip if value is < limit
0085 0800          00235      retlw  0            ;value > limit so z = 0.
0086 0040          00236      clrw   ;           ;z = 1
0087 0800          00237      retlw  0            ;return with z flag set
0088          00238      chk10Mlo
0088 020A          00239      movf   flo,w        ;get low value
0089 00AC          00240      subwf  limitlo, F   ;and check with low value
008A 0643          00241      btfsc  _z           ;not equal then skip
008B 0800          00242      retlw  0            ;else return with z = 1
008C 0703          00243      btfss  _c           ;skip if value is < limit

```

# AN592

```
LOC OBJECT CODE      LINE SOURCE TEXT
VALUE

008D 0800            00244      retlw  0          ;value > limit so z = 0
008E 0040            00245      clrw           ; z = 1
008F 0800            00246      retlw  0          ;return with z flag set
00247 ;
00248 ;Check1M checks if freq is below 1 Mhz
00249 ;
0090                00250      check1M
0090 0C13            00251      movlw  high oneMhz ;get hi value of 1Mhz
0091 002D            00252      movwf  limithi     ;save in limithi
0092 0C88            00253      movlw  low oneMhz  ;get lo value of 1Mhz
0093 002C            00254      movwf  limitlo     ;save in limitlo
0094 0A80            00255      goto   checklimit
00256 ;
0095                00257      check100K
0095 0C13            00258      movlw  high hndredK ;get hi value of 100Khz
0096 002D            00259      movwf  limithi     ;save in limithi
0097 0C88            00260      movlw  low hndredK ;get lo value of 100Khz
0098 002C            00261      movwf  limitlo     ;save in limitlo
0099 0A80            00262      goto   checklimit
00263 ;
009A                00264      check10K
009A 0C07            00265      movlw  high tenKhz ;get hi value of 10Khz
009B 002D            00266      movwf  limithi     ;save in limithi
009C 0CD0            00267      movlw  low tenKhz  ;get lo value of 10Khz
009D 002C            00268      movwf  limitlo     ;save in limitlo
009E 0A80            00269      goto   checklimit
00270 ;
00271 ;
009F                00272      Freq50Mto10M
009F 0071            00273      clrfs  RangeFlag
00A0 0511            00274      bsfs  Mhz50to10
00A1 0AAB            00275      goto  wait300mS
00A2                00276      Freq10Mto1M
00A2 0071            00277      clrfs  RangeFlag
00A3 0531            00278      bsfs  Mhz10to1
00A4 0AAB            00279      goto  wait300mS
00A5                00280      Freq1Mto100K
00A5 0071            00281      clrfs  RangeFlag
00A6 0551            00282      bsfs  Khz1Kto100
00A7 0AAB            00283      goto  wait250mS
00A8                00284      Freq100Kto10K
00A8 0071            00285      clrfs  RangeFlag
00A9 0571            00286      bsfs  Khz100to10
00AA 0AAB            00287      goto  wait50mS
00288 ;
00AB                00289      wait300mS
00290      If      !Debug
00291      call   delay50mS
00292      ENDIF
00AB                00293      wait250mS
00294      IF      !Debug
00295      call   delay50mS
00296      call   delay50mS
```



```

LOC  OBJECT CODE      LINE SOURCE TEXT
VALUE

                                00297      call    delay50mS
                                00298      call    delay50mS
                                00299      ENDIF
00AB                                00300  wait50mS
                                00301      IF      !Debug
                                00302      call    delay50mS
                                00303      ENDIF
                                00304      ;
                                00305      ;
                                00306      IF      Debug
                                00307 ;This routine debugs freq. on a PICDEM1 board.
00AB                                00308  checkRA1
00AB 0625                        00309      btfsc   _ra1
00AC 0AAB                        00310      goto    checkRA1
00AD 09D7                        00311      call    delay200mS
00AE 020B                        00312      movf   fhi,w
00AF 0026                        00313      movwf  _portb
00B0                                00314  chkRA1hi
00B0 0725                        00315      btfss  _ra1
00B1 0AB0                        00316      goto    chkRA1hi
00B2                                00317  chkRA1lo
00B2 0625                        00318      btfsc  _ra1
00B3 0AB2                        00319      goto    chkRA1lo
00B4 09D7                        00320      call    delay200mS
00B5 020A                        00321      movf   flo,w
00B6 0026                        00322      movwf  _portb
00B7 0725                        00323      btfss  _ra1
00B8 0AB7                        00324      goto    $-1
                                00325      ENDIF
00B9 0A0B                        00326      goto    repeat
                                00327      ;
                                00328 ;delay1mS, is a very accurate 1mS delay for a 4Mhz clock.
00BA                                00329  delay1mS
00BA 0CC5                        00330      movlw  .197
00BB 002D                        00331      movwf  count
00BC 0000                        00332      nop
00BD 0ABE                        00333      goto  $+1
00BE 0ABF                        00334      goto  $+1
00BF                                00335  dly1mS
00BF 0AC0                        00336      goto  $+1
00C0 02ED                        00337      decfsz count,F
00C1 0ABF                        00338      goto  dly1mS
00C2 0800                        00339      retlw  0
                                00340      ;
                                00341 ;delay5mS uses delay1mS to get a very accurate 5 mS delay
00C3                                00342  delay5mS
00C3 09BA                        00343      call    delay1mS
00C4 09BA                        00344      call    delay1mS
00C5 09BA                        00345      call    delay1mS
00C6 09BA                        00346      call    delay1mS
00C7 09BA                        00347      call    delay1mS
00C8 0C04                        00348      movlw  .4
00C9 002D                        00349      movwf  count

```

# AN592

```
LOC OBJECT CODE      LINE SOURCE TEXT
VALUE

00CA          00350 tweek5mS
00CA 02ED      00351      decfsz  count, F
00CB 0ACA      00352      goto    tweek5mS
00CC 0800      00353      return
                00354 ;
                00355 ;delay50mS uses delay1mS to get a very accurate 50mS delay
00CD          00356 delay50mS
00CD 0C32      00357      movlw  .50
00CE 002C      00358      movwf  tempa
00CF          00359 dly50mS
00CF 09BA      00360      call   delay1mS
00D0 02EC      00361      decfsz tempa, F
00D1 0ACF      00362      goto   dly50mS
00D2 0C0E      00363      movlw  .14
00D3 002D      00364      movwf  count
00D4          00365 tweek50mS
00D4 02ED      00366      decfsz count, F
00D5 0AD4      00367      goto   tweek50mS
00D6 0800      00368      retlw  0
                00369 ;
                00370 ;delay200mS uses delay1mS to get a very accurate 200mS delay.
00D7          00371 delay200mS
00D7 0CC8      00372      movlw  .200
00D8 002C      00373      movwf  tempa
00D9          00374 dly200mS
00D9 09BA      00375      call   delay1mS
00DA 02EC      00376      decfsz tempa, F
00DB 0AD9      00377      goto   dly200mS
00DC 0C40      00378      movlw  .64
00DD 002D      00379      movwf  count
00DE          00380 tweek200mS
00DE 02ED      00381      decfsz count, F
00DF 0ADE      00382      goto   tweek200mS
00E0 0800      00383      retlw  0
                00384 ;
                00385 ;getfreq, toggles the RA0 pin to shift out the value in the
                00386 ;pre-scaler. The number of toggles is kept in count. If the value
                00387 ;in tmr0 increments, then the low 8 bit value = !count + 1. The low
                00388 ;value of the freq. is loaded in flo and the high in fhi.
00E1          00389 getfreq
00E1 0201      00390      movf   _tmr0,w           ;get the tmr0 value
00E2 002B      00391      movwf  fhi              ;save in fhi
00E3 006D      00392      clrf   count           ;keep track of the toggles
00E4          00393 toggle
00E4 02AD      00394      incf   count, F        ;inc for first
00E5 0405      00395      bcf   _ra0             ;toggle the input
00E6 0505      00396      bsf   _ra0             ;
00E7 0201      00397      movf   _tmr0,w        ;see if tmr0 incremented
00E8 008B      00398      subwf  fhi,w           ;
00E9 0643      00399      btfsc  _z              ;yes then skip
00EA 0AE4      00400      goto   toggle         ;no then toggle again
00EB 026D      00401      comf   count, F       ;else complement count
00EC 028D      00402      incf   count,w        ;and increment
```



---

---

# 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: 708 285-0071 Fax: 708 285-0075

### Dallas

Microchip Technology Inc.  
14651 Dallas Parkway, Suite 816  
Dallas, TX 75240-8809  
Tel: 214 991-7177 Fax: 214 991-8588

### Dayton

Microchip Technology Inc.  
Suite 150  
Two Prestige Place  
Miamisburg, OH 45342  
Tel: 513 291-1654 Fax: 513 291-9175

### Los Angeles

Microchip Technology Inc.  
18201 Von Karman, Suite 1090  
Irvine, CA 92715  
Tel: 714 263-1888 Fax: 714 263-1338

## AMERICAS (continued)

### 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 Technology  
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

### Korea

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

### Singapore

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

### Taiwan

Microchip Technology  
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 1 628 850303 Fax: 44 1 628 850178

### 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 Muenchen, Germany  
Tel: 49 89 627 144 0 Fax: 49 89 627 144 44

### Italy

Arizona Microchip Technology SRL  
Centro Direzionale Colleoni  
Palazzo Taurus 1 V. Le Colleoni 1  
20041, Agrate Brianza, Milan Italy  
Tel: 39 39 689 9939 Fax: 39 39 689 9883

## JAPAN

Microchip Technology Intl. Inc.  
Benex S-1 6F  
3-18-20, Shin Yokohama  
Kohoku-Ku, Yokohama  
Kanagawa 222 Japan  
Tel: 81 45 471 6166 Fax: 81 45 471 6122

5/10/96



**MICROCHIP**

All rights reserved. © 1996, Microchip Technology Incorporated, USA.

Information contained in this publication regarding device applications and the like is intended through 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. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.