```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00001           title     "DC/AC RMS Voltage/Current, true/apparent Power, Freq.and Kf meter"
                     00002           list      P=PIC16F684, ST=OFF, R=dec, T=on, n=45  ;Turn off symbol table in list file
                     00003           #include  <P16F684.INC>
                     00001         LIST
                     00002 ; P16F684.INC  Standard Header File, Version 1.03    Microchip Technology, Inc.
                     00379         LIST
                     00004 ;_____________________________________________________________________________________________
                     00005 ;Program to take 2500 Voltage and Current samples 200 µs apart. It also measures
                     00006 ; averige period time over as many cycles within maximal 0.25 second. It calculates
                     00007 ; DC average voltage and current, RMS Voltage and current, Voltage Form-factor,
                     00008 ; true and apparent power and Frequency. Required time ±500ms.
                     00009 ;Reference voltage is set to 2.5 V to allow positive and negative readings.
                     00010 ; Input voltage is reduced in hardware by a scaling switch by 1/2, 1/20 or 1/200
                     00011 ;to get readings of ±0 to 5.00, 50.0 or 500 Volt, Input current is measured across a
                     00012 ; 25 mOhm resistor by an OpAmp and a ladder resistor network switch selector to get
                     00013 ; readings of ±0 to 90.0 mA, 900 mA, 9.00 A or 11.0 A
                     00014 ; Scaled readings are digitized bij an A/D converter and their ref, real,
                     00015 ;rectified, squared, multiplied values are summed up bij an interrupt routine.
                     00016 ; The main program then takes the mean and square root of these sums
                     00017 ;and converts the result to scale. A four digits display presents the result.
                     00018 ; Aditional the frequency can be displayed from 5 upto 99 KHz
                     00019 ;_____________________________________________________________________________________________
2000   000C 0002 0008 00020           __idlocs  0xC281    ;Checksum
       0001
2007   3FF2          00021           __config  _HS_OSC & _WDT_OFF  ;High speed crystal, Watchdog timer off
                     00022 ;________1_________2_________3_________4_________5_________6_________7_________8_________9
                     00023 ; Register memory definition 32 bytes
                     00024 ; Special function registers shared in all banks
                     00025 ;INDF     equ       00h       ;indirect address provided by FSR
                     00026 ;PCL      equ       02h       ;program counter low 8 address bits
                     00027 ;STATUS   equ       03h       ;program status register (bits 6-5= Page 0-3 Select)
                     00028 ;FSR      equ       04h       ;IA select (bit 7= Bank Select, bits 6-0= INDF address)
                     00029 ;PCLATH equ         0Ah       ;program counter 5 high address bits.
                     00030 ;INTCON   equ       0Bh       ;interrupt control register
                     00031
                     00032 ; Special function registers in Bank 0 only
                     00033 ;TMR0     equ       01h       ;Timer 0
                     00034 ;PORTA    equ       05h       ;General IO Port A register (see user bit definitions)
                     00035 ;PORTC    equ       07h       ;General IO Port C register (see user bit definitions)
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      00036 ;PIR1     equ       0Ch       ;peripheral interrupt request reg.
                      00037 ;TMR1L    equ       0Eh       ;Timer 1 low byte
                      00038 ;TMR1H    equ       0Fh       ;Timer 1 high byte
                      00039 ;T1CON    equ       10h       ;Timer 1 control register: GE=0, CKPS=00 OSCEN=0, CS=0
                      00040 ;TMR2     equ       11h       ;Timer 2 counter prescaler
                      00041 ;T2CON    equ       12h       ;Timer 2 control register
                      00042 ;CCPR1L   equ       13h       ;Time capture low register
                      00043 ;CCPR1H   equ       14h       ;Time capture high register
                      00044 ;CMCON0   equ       19h       ;Comparator control register
                      00045 ;CMCON1   equ       1Ah       ;Comparator control register
                      00046 ;ADRESH   equ       1Eh       ;A/D result high bits
                      00047 ;ADCON0   equ       1Fh       ;A/D control register
                      00048
                      00049 ; Special function registers  Bank 1 only
                      00050 ;OPTION_REG equ     81h       ;Timer 0 Control register, ~GPPU=0, INTEDG
                      00051 ;TRISA    equ       85h       ;Tristate control Port A
                      00052 ;TRISC    equ       87h       ;Tristate control Port C
                      00053 ;PIE1     equ       8Ch       ;Peripheral interrupt enable reg.
                      00054 ;PCON     equ       8Eh       ;Power control register
                      00055 ;OSCCON   exu       8Fh       ;Oscilator control register
                      00056 ;OSCTUNE equ       90h       ;Int.HF Oscilator calibration register
                      00057 ;ANSEL    equ       91h       ;Analog selection register
                      00058 ;PR2      equ       92h       ;Timer 2 period register
                      00059 ;WPUA     equ       95h       ;PORTA weak pull-up register
                      00060 ;IOCA     equ       96h       ;PORTA Interrupt on change
                      00061 ;VRCON    equ       99h       ;Reference voltage control register
                      00062 ;EEDAT    equ       9Ah       ;EEPROM data register
                      00063 ;EEADR    equ       9Bh       ;EEPROM adress register
                      00064 ;EECON1   equ       9Ch       ;EEPROM control register
                      00065 ;ADRESL   equ       9Eh       ;A/D result low bits
                      00066 ;ADCON1   equ       9Fh       ;A/D control register
                      00067
                      00068 ; General-IO PORTA/TRISA bit assignment
                      00069 ;Vss                           ;[14], GND
  00000000            00070 ARef      equ       0         ;[13], ANS0(/RA0/C1IN+/ICSPDAT) analog input, Reference point
  00000001            00071 AIsrc     equ       1         ;[12], ANS1(/RA1/C1IN-/ICSPCLK) analog input, Source current
  00000002            00072 AVsrc     equ       2         ;[11], ANS2(/RA2/INT/C1OUT) analog input, Source voltage
                      00073 ;_MCLR    equ       3         ;[4], ~MLCR(/RA3/Vpp) input (wpu)
                      00074 ;Xtal1    equ       4         ;[3], OSC2(RA4/AN3/CLKOUT) output
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00075 ;Xtal2    equ       5         ;[2], OSC1(RA5) input
                     00076 ;Vdd                          ;[1], +5V
                     00077
                     00078 ; General-IO PORTC/TRISC bit assignment
  00000000           00079 Vscale    equ       0         ;[10] C2IN+(/RC0/ANS4) input, 0V=*.**,1.2V=**.*,2.4V=***
  00000001           00080 Iscale    equ       1         ;[9] C2IN-(RC1/ANS5) input, 0V=**.*m,1.2V=***m,2.4V=*.**,5V=**.*
  00000002           00081 TM_CLK    equ       2         ;[8] RC2, TM1637 CLK, output
  00000003           00082 TM_DIO    equ       3         ;[7] RC3, TM1637 DIO, input/output
                     00083 ;         equ       4         ;[6] (unused)
                     00084 ;CCP1     equ       5         ;[5] CCP1(/RC5)
                     00085
                     00086 ; User data registers in RAM bank 0
                     00087           CBLOCK    0x20
  00000020           00088 SumVsqr:4 ;32 bit sum of 25*100 squared Vsrc-Ref A/D samples (max: 9C400000h)
  00000024           00089 SumIsqr:4 ;32 bit sum of 25*100 squares Isrc-Ref A/D samples (max: 9C400000h)
  00000028           00090 SumWatt:4 ;32 bit sum of (Vsrc-Ref)*(Isrc-Ref) signed (max: 9C400000h)
  0000002C           00091 Square:4  ;32 bit input for Sqrt_32 subroutine
  00000030           00092 Watt:4              ;20 bit (uses 32 bits) product of (Vsrc-Ref)*(Isrc-Ref) unsigned
  00000034           00093 Product:4 ;32 bit product from multiply (also Dividend and Quotient for division)
  00000038           00094 SumRef:3  ;24 bit sum 25*100 actual Ref A/D samples (max: 271000h)
  0000003B           00095 SumVsrc:3 ;24 bit sum 25*100 actual Vsrc A/D samples (max: 271000h)
  0000003E           00096 SumIsrc:3 ;24 bit sum 25*100 actual Isrc A/D samples (max: 271000h)
  00000041           00097 SumVrec:3 ;24 bit sum 25*100 rectified Vsrc-Ref A/D voltage samples (max: 271000h)
  00000044           00098 SumIrec:3 ;24 bit sum 25*100 rectified Isrc-Ref A/D current samples (max: 271000h)
  00000047           00099 Vsqr:3              ;20 bit result from square routine to be used by int
  0000004A           00100 Isqr:3              ;20 bit square of Irec to be used in int
  0000004D           00101 CCP_cnt             ;8 bit count of captured cycles
  0000004E           00102 CCPtime:2 ;16 bit elapsed time for CCP_cnt cycles
  00000050           00103 Divisor:2 ;16 bit Divisor
  00000052           00104 Remain:2  ;16 bit remainder from square Root (also remainder from division)
  00000054           00105 Counter             ;8 bit multiplier (also bit counter for square root and division)
  00000055           00106 Cycles,Samples      ;cycles covered, sample counter per cycle to be used by int          ;
  00000057           00107 sw_stat             ;status of function buttons attached to TM1637 and Scale settings
  00000058           00108 prefun              ;previous displayed function (allows display of Kf for V and I)
  00000059           00109 Scale               ;decimal point location and scale indicators
  0000005A           00110 Sign,TenK,OneK,Huns,Tens,Ones ;decimals for display
                     00111           ENDC
                     00112 ;variables used by other subroutines
  00000034           00113 Dividen   equ       Product
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

  00000034            00114 Root      equ       Product
  00000034            00115 NumL      equ       Product   ;source for bin2dec
  00000035            00116 NumH      equ       Product+1 ;high byte for same
  0000005A            00117 OV_flag   equ       Sign      ;flag to handle over-voltage condition
                      00118 ;sw_stat(3:0) value assigment for PB switches
  00000000            00119 P         equ       0         ;value in sw_stat for display true power
  00000001            00120 S         equ       1         ;value in sw_stat for display apparent power
  00000002            00121 Freq      equ       2         ;value in sw_stat for display Freq
  00000003            00122 Kf        equ       3         ;value in sw_stat for display Kf
  00000008            00123 Va        equ       8         ;value in sw_stat for display DC voltage
  00000009            00124 Vrms      equ       9         ;value in sw_stat for display Vrms
  0000000A            00125 Ia        equ       10        ;value in sw_stat for display DC current
  0000000B            00126 Irms      equ       11        ;value in sw_stat for display RMS current
                      00127 ;other bit assignments in sw_stat
  00000002            00128 adjust    equ       2         ;bit nr in sw_stat for adjust (0= no adjust, 1= adjust)
  00000003            00129 adj_dec   equ       3         ;bit nr in sw_stat for adjust divisor (0= increment, 1= decrement)
                      00130 ;Scale bit asignment in sw_stat, if both are 0 then Vscale0 (5 V) or Iscale0 (100 mA)
  00000004            00131 Vscale1   equ       4         ;bit nr for voltage scale 1 (50 V)
  00000005            00132 Vscale2   equ       5         ;bit nr for voltage scale 2 (500 V)
  00000006            00133 Iscale1   equ       6         ;bit nr for ampere scale 1,if 0 (99.9 mA), if 1 (999 mA)
  00000007            00134 Iscale2   equ       7         ;bit nr for ampere scale 2, if 1 (9.99 A), if 12 on (99.9 A)
                      00135 ;bit assignment in prefun
  00000000            00136 KfI       equ       0         ;if 1 then Kf for Current, else Kf for Voltage
                      00137 ;Display Scale bit assignment in Scale register
  00000000            00138 Scale1    equ       0         ;decimal point for 1 decimal place
  00000001            00139 Scale2    equ       1         ;decimal point for 2 decimal places
  00000002            00140 Scale3    equ       2         ;decimal point for 3 decimal places (for 10 kHz)
  00000004            00141 milli     equ       4         ;m- indicator, set in D6-SB
  00000005            00142 kilo      equ       5         ;k- indicator, set in D6-SG
                      00143 ;7 segment display bit assignments
  00000000            00144 Seg_A     equ       0
  00000001            00145 Seg_B     equ       1
  00000002            00146 Seg_C     equ       2
  00000003            00147 Seg_D     equ       3
  00000004            00148 Seg_E     equ       4
  00000005            00149 Seg_F     equ       5
  00000006            00150 Seg_G     equ       6
  00000007            00151 Seg_DP    equ       7
                      00152
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00153 ; User data in RAM bank 0 shared with bank 1
                     00154           CBLOCK    0x70      ;->7Fh, 16 bytes in RAM bank 0 shared with bank 1 and 2
  00000070           00155 int_W               ;interrupt save byte for W reg
  00000071           00156 int_S               ;interrupt save byte for STATUS reg
                     00157 ;int_PCH  ;interrupt save byte for PCLATH reg
                     00158 ;int_FSR  ;interrupt save byte for FSR reg (unused)
  00000072           00159 Ref:2               ;10 bit saved value of Ref to be used by int
  00000074           00160 Vsrc:2              ;10 bit saved value of Vsrc(-Ref) to be used by int
  00000076           00161 Isrc:2              ;10 bit saved value of Isrc(-Ref) to be used by int
  00000078           00162 Vrec:2              ;10 bit rectified voltage
  0000007A           00163 Irec:2              ;10 bit rectified current
  0000007C           00164 TM_dat,TM_cnt       ;data byte and bit count variables for TM1637 routine
                     00165           ENDC
                     00166 ;variables used for Frequency measure interrupt routine
  00000074           00167 FrstCCP   equ       Vsrc      ;16 bit timer value for first rise
  00000078           00168 LastCCP   equ       Vrec      ;16 bit time for last since first rise
                     00169 ;converting TM1637 scan codes to below 16 and switch states
                     00170 ;K1-Sw    Scancd    com        -8       switch    K2-Sw     Scancd    com        -8       switch
                     00171 ;         0xFF      0x00      0xF8      ---
                     00172 ;K1-S1    0xF7      0x08      0x00      P         K2-S1     0xEF      0x10      0x08      V=
                     00173 ;K1-S2    0xF6      0x09      0x01      S         K2-S2     0xEE      0x11      0x09      V~
                     00174 ;K1-S3    0xF5      0x0A      0x02      Freq      K2-S3     0xED      0x12      0x0A      I=
                     00175 ;K1-S4    0xF4      0x0B      0x03      Kf        K2-S4     0xEC      0x13      0x0B      I~
                     00176 ;K1-S5    0xF3      0x0C      0x04      iSclV=    K2-S5     0xEB      0x14      0x0C      dSclV=
                     00177 ;K1-S6    0xF2      0x0D      0x05      iSclV~    K2-S6     0xEA      0x15      0x0D      dSclV~
                     00178 ;K1-S7    0xF1      0x0E      0x06      iSclI=    K2-S7     0xE9      0x16      0x0E      dSclI=
                     00179 ;K1-S8    0xF0      0x0F      0x07      iSclI~    K2-S8     0xE8      0x17      0x0F      dSclI~
                     00180 ;_____________________________________________________________________________________
                     00181 ;EEPROM data        256 bytes of 8 bits
2100                 00182           org       0x2100              ;EEPROM preload address
2100   0043 004F 0050 00183           de        "COPYRIGHT 2025 Antonie (A.C.M.) Daas, Wormer Netherlands, "
       0059 0052 0049
       0047 0048 0054
       0020 0032 0030
       0032 0035 0020
       0041 006E 0074
       006F 006E 0069
       0065 0020 0028
       0041 002E 0043
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

       002E 004D 002E
       0029 0020 0044
       0061 0061 0073
       002C 0020 0057
       006F 0072 006D
       0065 0072 0020
       004E 0065 0074
       0068 006
213A   0020 003C 0068 00184           de        " <http://daas.info>"
       0074 0074 0070
       003A 002F 002F
       0064 0061 0061
       0073 002E 0069
       006E 0066 006F
       003E
  0000004D            00185 Vsc0cal   equ       low $               ;4.99 V   (±5% over range -127 > +127)
214D   00C6            00186           de        0xC6                ;calibration value for Voltage Scale 0
  0000004E            00187 Vsc1cal   equ       low $               ;49.9 V
214E   00BA            00188           de        0xBA                ;calibration value for Voltage Scale 1
  0000004F            00189 Vsc2cal   equ       low $               ;499 V
214F   00C9            00190           de        0xC9                ;calibration value for Voltage Scale 2
  00000050            00191 Isc0cal   equ       low $               ;99.9 mA
2150   00F1            00192           de        0xF1                ;calibration value for Current Scale 0
  00000051            00193 Isc1cal   equ       low $               ;999 mA
2151   00F4            00194           de        0xF4                ;calibration value for Current Scale 1
  00000052            00195 Isc2cal   equ       low $               ;9.99 A
2152   00EB            00196           de        0xEB                ;calibration value for Current Scale 2
  00000053            00197 Isc3cal   equ       low $               ;99.9 A   (limited to max. 11 A effective)
2153   0005            00198           de        0x05                ;calibration value for Current Scale 3
                       00199 ;_____________________________________________________________________________
                       00200 ; Macro to Square a 10 bit unsigned integer by Ton Daas
                       00201 ; Input: p<9:0>, Output: q<19:0> (20 bits)
                       00202 ; 53 instructions, 53 steps
                       00203 Sqr10     macro     p,q                 ;q has to be cleared
                       00204 ;q+1:q= p<7:0>^2
                       00205           clrc
                       00206           movf      p,W                 ;get p in W
                       00207           btfsc     p,0
                       00208            addwf    q+1,F
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00209           rrf       q+1,F
                     00210           rrf       q,F
                     00211           btfsc     p,1
                     00212            addwf    q+1,F
                     00213           rrf       q+1,F
                     00214           rrf       q,F
                     00215           btfsc     p,2
                     00216            addwf    q+1,F
                     00217           rrf       q+1,F
                     00218           rrf       q,F
                     00219           btfsc     p,3
                     00220            addwf    q+1,F
                     00221           rrf       q+1,F
                     00222           rrf       q,F
                     00223           btfsc     p,4
                     00224            addwf    q+1,F
                     00225           rrf       q+1,F
                     00226           rrf       q,F
                     00227           btfsc     p,5
                     00228            addwf    q+1,F
                     00229           rrf       q+1,F
                     00230           rrf       q,F
                     00231           btfsc     p,6
                     00232            addwf    q+1,F
                     00233           rrf       q+1,F
                     00234           rrf       q,F
                     00235           btfsc     p,7
                     00236            addwf    q+1,F
                     00237           rrf       q+1,F
                     00238           rrf       q,F       ; no carry out
                     00239 ; p*(p+1)*2^9       (W still contains p)
                     00240           rrf       q+1,F     ;align q+1 at bit 9, produces carry
                     00241           rrf       q+2,F     ;save carry bit
                     00242           btfsc     p+1,0     ;if p+1<0> bit is set, then:
                     00243            addwf    q+1,F     ;add p*2, produces carry
                     00244           rrf       q+1,F     ;save carry bit
                     00245           rrf       q+2,F     ;save bit in q+2
                     00246           btfsc     p+1,1     ;if p+1<1> bit is set, then:
                     00247            addwf    q+1,F     ;add p*4, produces carry
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00248           rlf       q+2,F     ;carry into q+2
                     00249           rlf       q+1,F     ;restore q+1
                     00250           rlf       q+2,F     ;
                     00251           rlf       q+1,F     ;restore q+1
                     00252           rlf       q+2,F     ;shift q+2 in place
                     00253 ;+p+1^2*2^16
                     00254           movf      p+1,W
                     00255           btfsc     p+1,0
                     00256            addwf    q+2,F     ;no carry
                     00257           rlf       p+1,W     ;no carry out
                     00258           btfsc     p+1,1
                     00259            addwf    q+2,F
                     00260           endm
                     00261 ;_____________________________________________________________________________
                     00262 ; Macro to Multiply two 10 bit unsigned integers by Ton Daas
                     00263 ; Input: x<9:0>, y<9:0>, Output: p<19:0> (20 bits), Local: p<31:24>
                     00264 ; 62 instructions, 62 steps
                     00265 Mul_10    macro     x,y,p
                     00266           clrc                          ;make sure C=0
                     00267 ;p1:p0= x0*y0
                     00268           movf      y,W       ;get y in W
                     00269           btfsc     x,0       ;if x<0> is 1, then:
                     00270            addwf    p+1,F     ;add y to p1
                     00271           rrf       p+1,F     ;shift p1 right, bit 0 into C
                     00272           rrf       p,F       ;shift p0 right, including carry
                     00273           btfsc     x,1       ;etc.
                     00274            addwf    p+1,F
                     00275           rrf       p+1,F
                     00276           rrf       p,F
                     00277           btfsc     x,2
                     00278            addwf    p+1,F
                     00279           rrf       p+1,F
                     00280           rrf       p,F
                     00281           btfsc     x,3
                     00282            addwf    p+1,F
                     00283           rrf       p+1,F
                     00284           rrf       p,F
                     00285           btfsc     x,4
                     00286            addwf    p+1,F
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00287           rrf       p+1,F
                     00288           rrf       p,F
                     00289           btfsc     x,5
                     00290            addwf    p+1,F
                     00291           rrf       p+1,F
                     00292           rrf       p,F
                     00293           btfsc     x,6
                     00294            addwf    p+1,F
                     00295           rrf       p+1,F
                     00296           rrf       p,F
                     00297           btfsc     x,7
                     00298            addwf    p+1,F
                     00299           rrf       p+1,F
                     00300           rrf       p,F       ;C=0, p1:p0 holds product of x0 and y0, p2=0
                     00301 ;p2:p1=+ x[9:8]*y   ;y is still in W
                     00302           btfsc     x+1,0
                     00303            addwf    p+1,F     ;produces new bit 16 into C
                     00304           rrf       p+1,F     ;get bit 16 into bit 15, bit 8 into C
                     00305           rrf       p+2,F     ;get bit 8 into bit 23, C=0
                     00306           btfsc     x+1,1
                     00307            addwf    p+1,F     ;produces new bit 17 into C
                     00308           rlf       p+2,F     ;move C (bit 17) into bit 16, bit 8 into C
                     00309           rlf       p+1,F     ;move C (bit 8) into bit 8, bit 16 into C
                     00310           rlf       p+2,F     ;move C (bit 16) into 16, C=0
                     00311 ;p2:p1=+x*y[9:8]
                     00312           clrf      p+3
                     00313           movf      x,W
                     00314           btfsc     y+1,0
                     00315            addwf    p+1,F     ;produces new bit 16 into C
                     00316           rrf       p+1,F     ;get bit 16 into bit 15, bit 8 into C
                     00317           rrf       p+3,F     ;get bit 8 into bit 23, C=0, p+3= ?000 0000
                     00318           btfsc     y+1,1
                     00319            addwf    p+1,F     ;produces new bit 17 into C
                     00320           rlf       p+3,F     ;move C (bit 17) into bit 16, bit 8 into C, p+3= 0000 000?
                     00321           rlf       p+1,F     ;move C (bit 8) into bit 8, bit 16 into C
                     00322           rlf       p+3,W     ;move C (bit 16) into 16, C=0, p+3= 0000 000?
                     00323           addwf     p+2,F
                     00324 ;p2=+x[9:8]*y[9:8]
                     00325           movf      y+1,W     ;get high 2 bits in W
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00326           btfsc     x+1,0     ;if bit 8 is 1, then:
                     00327            addwf    p+2,F     ;add high 2 bits into high byte, C=0
                     00328           rlf       y+1,W     ;get high 2 bits *2
                     00329           btfsc     x+1,1     ;if bit 9 is 1, then:
                     00330            addwf    p+2,F     ;add high 2 bits *2 into high byte
                     00331           endm
                     00332 ;_______end of data definitions_____________________________________________________
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00333           page
                     00334 ;_____________________________________________________________________________
                     00335 ; Page 0, Flash Program memory (0-3FFh= 1K words)
0000                 00336           org       0                   ;Page 0, PC load at PO or MCLR
0000   018A          00337           clrf      PCLATH              ;make sure we are on page 0
0001   0183          00338           clrf      STATUS              ;make sure we address bank 0
0002   2BB5          00339           goto      init
                     00340 ;_____________________________________________________________________________
                     00341 ;Program stack has 8 entries for interrupt and subroutine CALLs
                     00342 ;--------------------------------
                     00343 ; Global interrupt handler     time from int=      3µs
0004                 00344           org       4                   ;PC load at global interrupt
0004   00F0          00345           movwf     int_W               ;save content of W (status unchanged)
0005   0E03          00346           swapf     STATUS,W  ;get current STATUS register swapped
0006   00F1          00347           movwf     int_S               ;save interrupted STATUS swapped
0007   0183          00348           clrf      STATUS              ;make sure we address bank 0
0008   1B0C          00349           btfsc     PIR1,ADIF ;if interrupt from A/D
0009   2833          00350            goto     AD_int              ;go and handle interrupt
000A   188C          00351           btfsc     PIR1,TMR2IF         ;if no interrupt from Timer 2, then:
000B   282C          00352            goto     T2_int              ;go and handle interrupt
000C   1A8C          00353           btfsc     PIR1,CCP1IF         ;if interrupt from CCP, then:
000D   2815          00354            goto     CCP_int             ;go and handle interrupt
000E   1C0C          00355           btfss     PIR1,TMR1IF         ;if no interrupt from Timer 1, then:
000F   2989          00356            goto     int_ret             ;nothing to do, so quit
                     00357 ; Timer 1 interrupt handler for frequency. Overfows after 65536 counts= ±262 ms
0010   100C          00358           bcf       PIR1,TMR1IF         ;clear our interrupt flag
0011   1010          00359 T1_stop   bcf       T1CON,TMR1ON        ;stop Timer 1, also flags finished
0012   0195          00360           clrf      CCP1CON             ;reset CCP module
0013   128C          00361           bcf       PIR1,CCP1IF         ;clear any pending CCP int
0014   2989          00362           goto      int_ret             ;done
                     00363
                     00364 ; Capture interrupt handler for frequency (maximum 255 cycles, FFFFh maximum last CCP)
0015   128C          00365 CCP_int   bcf       PIR1,CCP1IF         ;clear interrupt flag
0016   0874          00366           movf      FrstCCP,W
0017   0475          00367           iorwf     FrstCCP+1,W         ;check for first CCP received
0018   1D03          00368           skpz                          ;if not first, then:
0019   281F          00369            goto     CCP_nxt             ;go and handle next
001A   0813          00370           movf      CCPR1L,W  ;get time captured low byte
001B   00F4          00371           movwf     FrstCCP             ;and save it
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

001C   0814           00372           movf      CCPR1H,W  ;get time captured high byte
001D   00F5           00373           movwf     FrstCCP+1 ;and save it
001E   2989           00374           goto      int_ret
                      00375
001F   0874           00376 CCP_nxt   movf      FrstCCP,W ;load first time captured low byte
0020   0213           00377           subwf     CCPR1L,W  ;calc time elapsed since first low byte
0021   00F8           00378           movwf     LastCCP             ;and save it
0022   0875           00379           movf      FrstCCP+1,W         ;load first time captured high byte
0023   1C03           00380           skpc                          ;if borrow from low byte, then:
0024   0F75           00381            incfsz   FrstCCP+1,W         ;take care of borrow
0025   0214           00382            subwf    CCPR1H,W  ;calc time elapsed since first high byte
0026   00F9           00383           movwf     LastCCP+1 ;and save it
0027   0FD5           00384           incfsz    Cycles,F  ;count cycles captured, if < 256, then:
0028   2989           00385            goto     int_ret             ;done, continue counting, else:
0029   2811           00386           goto      T1_stop             ;go and stop counting (CCP values remains valid)
                      00387
                      00388 ;Timer 2 interrupt handler to trigger A/D sampling (execution time= 21 µs)
                      00389 ; Trigger A/D sampling 210 µs apart
002A   1C8C           00390 T2_wait   btfss     PIR1,TMR2IF         ;if no interrupt yet from Timer 2, then:
002B   282A           00391            goto     $-1                 ;wait for it
002C   108C           00392 T2_int    bcf       PIR1,TMR2IF         ;clear interrupt flag
002D   1B0C           00393           btfsc     PIR1,ADIF ;if AD interrupt pending, then:
002E   2833           00394            goto     AD_int              ;go handle AD interrupt directly, else:
002F   149F           00395           bsf       ADCON0,GO ;start A/D conversion for Ref sample (56 steps)
0030   159F           00396           bsf       ADCON0,CHS1         ;select AN2 (Vsrc) to allow acquisition time
                      00397 ;A/D interrupt handler (total execution time= Ref:50, Vsrc: 120, Isrc: 212)
                      00398 ; Collects 25 x 100 voltage samples from A/D.
                      00399 ; After AD_GO it takes 23 µs (=46 steps) to complete. In the mean time previous result
                      00400 ; can be read and input selection may be changed. Allow at least 8 µs after input change.
0031   1F0C           00401 AD_wait   btfss     PIR1,ADIF ;if no interrupt yet from AD, then:
0032   2831           00402            goto     $-1                 ;wait for it (might take 17 cycles)
0033   130C           00403 AD_int    bcf       PIR1,ADIF ;clear interrupt
0034   1683           00404           bsf       STATUS,RP0          ;select register bank 1
0035   081E           00405           movf      ADRESL,W  ;get lower 8 bits of A/D result
0036   1283           00406           bcf       STATUS,RP0          ;reselect register bank 0
0037   191F           00407           btfsc     ADCON0,CHS0         ;if Iscr is selected, then Vsrc is finished
0038   2856           00408            goto     ADVsrc              ;process Vsrc value
0039   1D9F           00409           btfss     ADCON0,CHS1         ;if Vsrc is not selected, then Isrc is finished
003A   28C3           00410            goto     ADIsrc              ;process Isrc value
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00411 ;read Ref and add to SumRef with reference to ground
003B   149F          00412           bsf       ADCON0,GO ;restart A/D conversion for Vsrc or Isrc sample
003C   119F          00413           bcf       ADCON0,CHS1         ;deselect AN2 (Vsrc)
003D   151F          00414           bsf       ADCON0,CHS0         ;select AN1 (Isrc)
003E   00F2          00415           movwf     Ref                 ;save Ref value
003F   07B8          00416           addwf     SumRef,F  ;add low byte to sum
0040   081E          00417           movf      ADRESH,W  ;get upper 2 bits of A/D result
0041   00F3          00418           movwf     Ref+1               ;save upper 2 bits of Ref
0042   1803          00419           skpnc                         ;if carry from add to SumRef
0043   0F1E          00420            incfsz   ADRESH,W  ;add carry to W
0044   07B9          00421            addwf    SumRef+1,F          ;add upper 2 bits to SumRef+1
0045   1803          00422           skpnc                         ;if carry, then:
0046   0ABA          00423            incf     SumRef+2,F          ;increment high byte of SumRef
0047   01C7          00424           clrf      Vsqr                ;clear Vsqr, used in next routine
0048   01C8          00425           clrf      Vsqr+1
0049   01C9          00426           clrf      Vsqr+2
004A   01CA          00427           clrf      Isqr                ;clear Isqr, used in next routine
004B   01CB          00428           clrf      Isqr+1
004C   01CC          00429           clrf      Isqr+2
004D   01B0          00430           clrf      Watt                ;clear Watt, used in next routine
004E   01B1          00431           clrf      Watt+1
004F   01B2          00432           clrf      Watt+2
0050   0BD6          00433           decfsz    Samples,F ;if not all 100 samples collected, then:
0051   2989          00434            goto     int_ret             ;wait for next AD int
0052   3064          00435           movlw     100                 ;new sample count
0053   00D6          00436           movwf     Samples             ;preset Sample count
0054   03D5          00437           decf      Cycles,F  ;count cycles
0055   2989          00438           goto      int_ret             ;done, expect to wait ±14 to 18 steps
                     00439
                     00440 ;read Vsrc A/D and add to SumVsrc with reference to ground
0056   149F          00441 ADVsrc    bsf       ADCON0,GO ;restart A/D conversion for Isrc sample
0057   111F          00442           bcf       ADCON0,CHS0         ;deselect AN1 (Isrc), select AN0 (Ref)
0058   00F4          00443           movwf     Vsrc                ;save lower byte of Vsrc
0059   07BB          00444           addwf     SumVsrc,F ;sum Source A/D for averaging DC offset
005A   081E          00445           movf      ADRESH,W  ;get upper 2 bits of A/D result
005B   00F5          00446           movwf     Vsrc+1              ;and save it
005C   1803          00447           skpnc                         ;if carry from add Vsrc, then:
005D   0F75          00448            incfsz   Vsrc+1,W  ;add carry to W, if not 0, then:
005E   07BC          00449            addwf    SumVsrc+1,F         ;add high byte
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

005F   1803           00450           skpnc                           ;if carry, then:
0060   0ABD           00451            incf     SumVsrc+2,F           ;increment 3rd byte
                      00452 ;calculate voltage with reference to Ref, signed result replaces Vsrc
0061   0872           00453           movf     Ref,W                  ;get Ref low byte
0062   02F4           00454           subwf    Vsrc,F                 ;calculate offset from Ref low byte
0063   0873           00455           movf     Ref+1,W                ;get Ref high 2 bits
0064   1C03           00456           skpc                            ;if borrow, then:
0065   0A73           00457            incf     Ref+1,W               ;add borrow
0066   02F5           00458           subwf    Vsrc+1,F  ;calculate offset from Ref high byte
                      00459 ;Rectify (Vsrc-Ref) into Vrec
0067   0874           00460           movf     Vsrc,W                 ;copy Vsrc
0068   00F8           00461           movwf    Vrec                   ;  into Vrec
0069   0875           00462           movf     Vsrc+1,W  ;and its high byte
006A   00F9           00463           movwf    Vrec+1
006B   1803           00464           skpnc                           ;if positive Vsrc (no borrow from subwf), then:
006C   2872           00465            goto    Vabs                   ;skip complementing
006D   09F8           00466           comf     Vrec,F                 ;complement low byte
006E   09F9           00467           comf     Vrec+1,F  ;and its high byte
006F   0AF8           00468           incf     Vrec,F                 ;make 2's complement
0070   1903           00469           skpnz                           ;if overflow, then:
0071   0AF9           00470            incf     Vrec+1,F  ;propagate carry
                      00471 ;check for over-voltage
0072   0C79           00472 Vabs      rrf      Vrec+1,W  ;get bit 8 in C
0073   0C78           00473           rrf      Vrec,W                 ;get bits 8 to 1 in W
0074   39F0           00474           andlw    0xF0                   ;isolate bits 8 to 5
0075   1D03           00475           skpz                            ;if any bit is set, then:
0076   145A           00476            bsf     OV_flag,0 ;flag for over-voltage
0077   1CF9           00477           btfss    Vrec+1,1  ;if Vrec < 512, then:
0078   01DA           00478            clrf    OV_flag                ;clear flag for over-voltage
                      00479 ;add result to SumVrec
0079   0878           00480           movf     Vrec,W                 ;get low byte to add
007A   07C1           00481           addwf    SumVrec,F ;add low byte
007B   0879           00482           movf     Vrec+1,W  ;get high byte to add
007C   1803           00483           skpnc                           ;if carry from add low byte, then:
007D   0F79           00484            incfsz   Vrec+1,W  ;increase byte to add, if not 0, then:
007E   07C2           00485            addwf    SumVrec+1,F           ;add high byte
007F   1803           00486           skpnc                           ;if carry, then:
0080   0AC3           00487            incf     SumVrec+2,F           ;increment 3rd byte
                      00488 ;Square value and add to SumVsqr
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00489           Sqr10     Vrec,Vsqr ;square Vrec, result in Vsqr
                         M ;q+1:q= p<7:0>^2
0081   1003              M           clrc
0082   0878              M           movf      Vrec,W              ;get p in W
0083   1878              M           btfsc     Vrec,0
0084   07C8              M            addwf    Vsqr+1,F
0085   0CC8              M           rrf       Vsqr+1,F
0086   0CC7              M           rrf       Vsqr,F
0087   18F8              M           btfsc     Vrec,1
0088   07C8              M            addwf    Vsqr+1,F
0089   0CC8              M           rrf       Vsqr+1,F
008A   0CC7              M           rrf       Vsqr,F
008B   1978              M           btfsc     Vrec,2
008C   07C8              M            addwf    Vsqr+1,F
008D   0CC8              M           rrf       Vsqr+1,F
008E   0CC7              M           rrf       Vsqr,F
008F   19F8              M           btfsc     Vrec,3
0090   07C8              M            addwf    Vsqr+1,F
0091   0CC8              M           rrf       Vsqr+1,F
0092   0CC7              M           rrf       Vsqr,F
0093   1A78              M           btfsc     Vrec,4
0094   07C8              M            addwf    Vsqr+1,F
0095   0CC8              M           rrf       Vsqr+1,F
0096   0CC7              M           rrf       Vsqr,F
0097   1AF8              M           btfsc     Vrec,5
0098   07C8              M            addwf    Vsqr+1,F
0099   0CC8              M           rrf       Vsqr+1,F
009A   0CC7              M           rrf       Vsqr,F
009B   1B78              M           btfsc     Vrec,6
009C   07C8              M            addwf    Vsqr+1,F
009D   0CC8              M           rrf       Vsqr+1,F
009E   0CC7              M           rrf       Vsqr,F
009F   1BF8              M           btfsc     Vrec,7
00A0   07C8              M            addwf    Vsqr+1,F
00A1   0CC8              M           rrf       Vsqr+1,F
00A2   0CC7              M           rrf       Vsqr,F    ; no carry out
                         M ; p*(p+1)*2^9       (W still contains p)
00A3   0CC8              M           rrf       Vsqr+1,F  ;align q+1 at bit 9, produces carry
00A4   0CC9              M           rrf       Vsqr+2,F  ;save carry bit
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

00A5   1879               M           btfsc     Vrec+1,0  ;if p+1<0> bit is set, then:
00A6   07C8               M            addwf    Vsqr+1,F  ;add p*2, produces carry
00A7   0CC8               M           rrf       Vsqr+1,F  ;save carry bit
00A8   0CC9               M           rrf       Vsqr+2,F  ;save bit in q+2
00A9   18F9               M           btfsc     Vrec+1,1  ;if p+1<1> bit is set, then:
00AA   07C8               M            addwf    Vsqr+1,F  ;add p*4, produces carry
00AB   0DC9               M           rlf       Vsqr+2,F  ;carry into q+2
00AC   0DC8               M           rlf       Vsqr+1,F  ;restore q+1
00AD   0DC9               M           rlf       Vsqr+2,F  ;
00AE   0DC8               M           rlf       Vsqr+1,F  ;restore q+1
00AF   0DC9               M           rlf       Vsqr+2,F  ;shift q+2 in place
                          M ;+p+1^2*2^16
00B0   0879               M           movf      Vrec+1,W
00B1   1879               M           btfsc     Vrec+1,0
00B2   07C9               M            addwf    Vsqr+2,F  ;no carry
00B3   0D79               M           rlf       Vrec+1,W  ;no carry out
00B4   18F9               M           btfsc     Vrec+1,1
00B5   07C9               M            addwf    Vsqr+2,F
00B6   0847            00490          movf      Vsqr,W              ;get low byte to add
00B7   07A0            00491          addwf     SumVsqr,F ;add low byte to sum
00B8   0848            00492          movf      Vsqr+1,W  ;get middle byte to add
00B9   1803            00493          skpnc                         ;if carry from low byte, then:
00BA   0F48            00494           incfsz   Vsqr+1,W  ;increment byte to add, if no overflow, then:
00BB   07A1            00495           addwf    SumVsqr+1,F         ;add second byte to sum
00BC   0849            00496          movf      Vsqr+2,W  ;get third byte to add
00BD   1803            00497          skpnc                         ;if carry from second byte
00BE   0F49            00498           incfsz   Vsqr+2,W  ;increment byte to add
00BF   07A2            00499           addwf    SumVsqr+2,F         ;add third byte to sum
00C0   1803            00500          skpnc                         ;if carry from third byte, then:
00C1   0AA3            00501           incf     SumVsqr+3,F         ;handle Carry into fourth byte
00C2   2831            00502          goto      AD_wait             ;done, next interrupt should already be set
                       00503
                       00504 ;read Isrc and sum to SumIsrc
00C3   149F            00505 ADIsrc   bsf       ADCON0,GO ;start A/D conversion for Ref sample
00C4   159F            00506          bsf       ADCON0,CHS1         ;select AN2 (Vsrc) to allow acquisition time
00C5   00F6            00507          movwf     Isrc                ;save lower byte of Isrc
00C6   07BE            00508          addwf     SumIsrc,F ;sum Source A/D for averaging DC current offset
00C7   081E            00509          movf      ADRESH,W  ;get upper 2 bits of Isrc
00C8   00F7            00510          movwf     Isrc+1              ;and save it
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

00C9   1803           00511           skpnc                           ;if carry from low byte add, then:
00CA   0F77           00512            incfsz   Isrc+1,W  ;include carry bit to add
00CB   07BF           00513            addwf    SumIsrc+1,F            ;add Isrc high byte to sum
00CC   1803           00514           skpnc                           ;if carry, then:
00CD   0AC0           00515            incf     SumIsrc+2,F            ;increment upper byte of sum
                      00516 ;calculate voltage across shunt (Isrc-Ref) and save back in Isrc
00CE   0872           00517           movf     Ref,W                  ;get Ref low byte
00CF   02F6           00518           subwf    Isrc,F                 ;calculate offset from Ref low byte
00D0   0873           00519           movf     Ref+1,W                ;get Ref high byte
00D1   1C03           00520           skpc                            ;if borrow, then:
00D2   0A73           00521            incf     Ref+1,W                ;add borrow
00D3   02F7           00522           subwf    Isrc+1,F  ;calculate offset from Ref high byte
                      00523 ;Check for negative value and convert to positive
00D4   0876           00524           movf     Isrc,W
00D5   00FA           00525           movwf    Irec
00D6   0877           00526           movf     Isrc+1,W
00D7   00FB           00527           movwf    Irec+1
00D8   1803           00528           skpnc                           ;if positive Isrc (no borrow from sub), then:
00D9   28DF           00529            goto     Iabs                   ;skip conversion to positive value
00DA   09FA           00530           comf     Irec,F                 ;complement low byte
00DB   09FB           00531           comf     Irec+1,F  ;complement high byte
00DC   0AFA           00532           incf     Irec,F                 ;make low byte 2's complement
00DD   1903           00533           skpnz                           ;if zero, then:
00DE   0AFB           00534            incf     Irec+1,F  ;add carry into high byte
                      00535 ;add result to SumIrec
00DF   087A           00536 Iabs      movf     Irec,W                 ;get low byte to add
00E0   07C4           00537           addwf    SumIrec,F ;add low byte
00E1   087B           00538           movf     Irec+1,W  ;get high byte to add
00E2   1803           00539           skpnc                           ;if carry from add low byte, then:
00E3   0F7B           00540            incfsz   Irec+1,W  ;increase byte to add, if not 0, then:
00E4   07C5           00541            addwf    SumIrec+1,F            ;add high byte
00E5   1803           00542           skpnc                           ;if carry, then:
00E6   0AC6           00543            incf     SumIrec+2,F            ;increment 3rd byte
                      00544 ;Square Irec and add to SumIsqr
                      00545           Sqr10    Irec,Isqr ;square Irec and place result in Isqr
                          M ;q+1:q= p<7:0>^2
00E7   1003               M           clrc
00E8   087A               M           movf     Irec,W                 ;get p in W
00E9   187A               M           btfsc    Irec,0
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

00EA   07CB               M            addwf    Isqr+1,F
00EB   0CCB               M           rrf       Isqr+1,F
00EC   0CCA               M           rrf       Isqr,F
00ED   18FA               M           btfsc     Irec,1
00EE   07CB               M            addwf    Isqr+1,F
00EF   0CCB               M           rrf       Isqr+1,F
00F0   0CCA               M           rrf       Isqr,F
00F1   197A               M           btfsc     Irec,2
00F2   07CB               M            addwf    Isqr+1,F
00F3   0CCB               M           rrf       Isqr+1,F
00F4   0CCA               M           rrf       Isqr,F
00F5   19FA               M           btfsc     Irec,3
00F6   07CB               M            addwf    Isqr+1,F
00F7   0CCB               M           rrf       Isqr+1,F
00F8   0CCA               M           rrf       Isqr,F
00F9   1A7A               M           btfsc     Irec,4
00FA   07CB               M            addwf    Isqr+1,F
00FB   0CCB               M           rrf       Isqr+1,F
00FC   0CCA               M           rrf       Isqr,F
00FD   1AFA               M           btfsc     Irec,5
00FE   07CB               M            addwf    Isqr+1,F
00FF   0CCB               M           rrf       Isqr+1,F
0100   0CCA               M           rrf       Isqr,F
0101   1B7A               M           btfsc     Irec,6
0102   07CB               M            addwf    Isqr+1,F
0103   0CCB               M           rrf       Isqr+1,F
0104   0CCA               M           rrf       Isqr,F
0105   1BFA               M           btfsc     Irec,7
0106   07CB               M            addwf    Isqr+1,F
0107   0CCB               M           rrf       Isqr+1,F
0108   0CCA               M           rrf       Isqr,F    ; no carry out
                          M ; p*(p+1)*2^9       (W still contains p)
0109   0CCB               M           rrf       Isqr+1,F  ;align q+1 at bit 9, produces carry
010A   0CCC               M           rrf       Isqr+2,F  ;save carry bit
010B   187B               M           btfsc     Irec+1,0  ;if p+1<0> bit is set, then:
010C   07CB               M            addwf    Isqr+1,F  ;add p*2, produces carry
010D   0CCB               M           rrf       Isqr+1,F  ;save carry bit
010E   0CCC               M           rrf       Isqr+2,F  ;save bit in q+2
010F   18FB               M           btfsc     Irec+1,1  ;if p+1<1> bit is set, then:
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0110   07CB               M            addwf    Isqr+1,F  ;add p*4, produces carry
0111   0DCC               M           rlf       Isqr+2,F  ;carry into q+2
0112   0DCB               M           rlf       Isqr+1,F  ;restore q+1
0113   0DCC               M           rlf       Isqr+2,F  ;
0114   0DCB               M           rlf       Isqr+1,F  ;restore q+1
0115   0DCC               M           rlf       Isqr+2,F  ;shift q+2 in place
                          M ;+p+1^2*2^16
0116   087B               M           movf      Irec+1,W
0117   187B               M           btfsc     Irec+1,0
0118   07CC               M            addwf    Isqr+2,F  ;no carry
0119   0D7B               M           rlf       Irec+1,W  ;no carry out
011A   18FB               M           btfsc     Irec+1,1
011B   07CC               M            addwf    Isqr+2,F
011C   084A            00546          movf      Isqr,W              ;get low byte to add
011D   07A4            00547          addwf     SumIsqr,F ;add byte to sum
011E   084B            00548          movf      Isqr+1,W  ;get middle byte to add
011F   1803            00549          skpnc                         ;if carry from low byte, then:
0120   0F4B            00550           incfsz   Isqr+1,W  ;increment byte to add, if no overflow , then:
0121   07A5            00551           addwf    SumIsqr+1,F         ;add middle 8 bits to sum
0122   084C            00552          movf      Isqr+2,W  ;get high byte to add
0123   1803            00553          skpnc                         ;if carry from middle byte
0124   0F4C            00554           incfsz   Isqr+2,W
0125   07A6            00555           addwf    SumIsqr+2,F         ;add middle 8 bits to sum
0126   1803            00556          skpnc                         ;if carry from high byte, then:
0127   0AA7            00557           incf     SumIsqr+3,F         ;handle Carry into bits 24 to 31
                       00558 ;Calculate power, Watt= (Vsrc-Ref)*(Isrc-Ref) (20 bits result)
                       00559          Mul_10    Vrec,Irec,Watt      ;multiply Vrec with Irec, 20 bit result
0128   1003               M           clrc                          ;make sure C=0
                          M ;p1:p0= x0*y0
0129   087A               M           movf      Irec,W    ;get y in W
012A   1878               M           btfsc     Vrec,0    ;if x<0> is 1, then:
012B   07B1               M            addwf    Watt+1,F  ;add y to p1
012C   0CB1               M           rrf       Watt+1,F  ;shift p1 right, bit 0 into C
012D   0CB0               M           rrf       Watt,F    ;shift p0 right, including carry
012E   18F8               M           btfsc     Vrec,1    ;etc.
012F   07B1               M            addwf    Watt+1,F
0130   0CB1               M           rrf       Watt+1,F
0131   0CB0               M           rrf       Watt,F
0132   1978               M           btfsc     Vrec,2
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0133   07B1               M            addwf    Watt+1,F
0134   0CB1               M           rrf       Watt+1,F
0135   0CB0               M           rrf       Watt,F
0136   19F8               M           btfsc     Vrec,3
0137   07B1               M            addwf    Watt+1,F
0138   0CB1               M           rrf       Watt+1,F
0139   0CB0               M           rrf       Watt,F
013A   1A78               M           btfsc     Vrec,4
013B   07B1               M            addwf    Watt+1,F
013C   0CB1               M           rrf       Watt+1,F
013D   0CB0               M           rrf       Watt,F
013E   1AF8               M           btfsc     Vrec,5
013F   07B1               M            addwf    Watt+1,F
0140   0CB1               M           rrf       Watt+1,F
0141   0CB0               M           rrf       Watt,F
0142   1B78               M           btfsc     Vrec,6
0143   07B1               M            addwf    Watt+1,F
0144   0CB1               M           rrf       Watt+1,F
0145   0CB0               M           rrf       Watt,F
0146   1BF8               M           btfsc     Vrec,7
0147   07B1               M            addwf    Watt+1,F
0148   0CB1               M           rrf       Watt+1,F
0149   0CB0               M           rrf       Watt,F    ;C=0, p1:p0 holds product of x0 and y0, p2=0
                          M ;p2:p1=+ x[9:8]*y   ;y is still in W
014A   1879               M           btfsc     Vrec+1,0
014B   07B1               M            addwf    Watt+1,F  ;produces new bit 16 into C
014C   0CB1               M           rrf       Watt+1,F  ;get bit 16 into bit 15, bit 8 into C
014D   0CB2               M           rrf       Watt+2,F  ;get bit 8 into bit 23, C=0
014E   18F9               M           btfsc     Vrec+1,1
014F   07B1               M            addwf    Watt+1,F  ;produces new bit 17 into C
0150   0DB2               M           rlf       Watt+2,F  ;move C (bit 17) into bit 16, bit 8 into C
0151   0DB1               M           rlf       Watt+1,F  ;move C (bit 8) into bit 8, bit 16 into C
0152   0DB2               M           rlf       Watt+2,F  ;move C (bit 16) into 16, C=0
                          M ;p2:p1=+x*y[9:8]
0153   01B3               M           clrf      Watt+3
0154   0878               M           movf      Vrec,W
0155   187B               M           btfsc     Irec+1,0
0156   07B1               M            addwf    Watt+1,F  ;produces new bit 16 into C
0157   0CB1               M           rrf       Watt+1,F  ;get bit 16 into bit 15, bit 8 into C
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0158   0CB3              M           rrf       Watt+3,F  ;get bit 8 into bit 23, C=0, p+3= ?000 0000
0159   18FB              M           btfsc     Irec+1,1
015A   07B1              M            addwf    Watt+1,F  ;produces new bit 17 into C
015B   0DB3              M           rlf       Watt+3,F  ;move C (bit 17) into bit 16, bit 8 into C, p+3= 0000 000?
015C   0DB1              M           rlf       Watt+1,F  ;move C (bit 8) into bit 8, bit 16 into C
015D   0D33              M           rlf       Watt+3,W  ;move C (bit 16) into 16, C=0, p+3= 0000 000?
015E   07B2              M           addwf     Watt+2,F
                         M ;p2=+x[9:8]*y[9:8]
015F   087B              M           movf      Irec+1,W  ;get high 2 bits in W
0160   1879              M           btfsc     Vrec+1,0  ;if bit 8 is 1, then:
0161   07B2              M            addwf    Watt+2,F  ;add high 2 bits into high byte, C=0
0162   0D7B              M           rlf       Irec+1,W  ;get high 2 bits *2
0163   18F9              M           btfsc     Vrec+1,1  ;if bit 9 is 1, then:
0164   07B2              M            addwf    Watt+2,F  ;add high 2 bits *2 into high byte
0165   0875           00560          movf      Vsrc+1,W  ;get value of upper signed Vsrc byte
0166   0677           00561          xorwf     Isrc+1,W  ;exclusive or with upper signed Isrc byte
0167   3980           00562          andlw     0x80                ;get product sign
0168   1D03           00563          skpz                          ;if negative (positive * negative), then:
0169   2977           00564           goto     Wneg                ;go subtract product from sum
                      00565 ;add absolute sample result to SumWatt
016A   0830           00566          movf      Watt,W              ;get low byte to add
016B   07A8           00567          addwf     SumWatt,F ;add low byte to sum
016C   0831           00568          movf      Watt+1,W  ;get middle byte to add
016D   1803           00569          skpnc                         ;if carry from low byte, then:
016E   0F31           00570           incfsz   Watt+1,W  ;increment byte to add, if no overflow , then:
016F   07A9           00571           addwf    SumWatt+1,F         ;add middle 8 bits to sum
0170   0832           00572          movf      Watt+2,W  ;get high 4 bits to add
0171   1803           00573          skpnc                         ;if carry from high byte
0172   0F32           00574           incfsz   Watt+2,W  ;include carry, if not zero, then:
0173   07AA           00575           addwf    SumWatt+2,F         ;add high byte to sum
0174   1803           00576          skpnc
0175   0AAB           00577           incf     SumWatt+3,F         ;add carry
0176   2983           00578          goto      chk_end             ;done
                      00579
                      00580 ;subtract absolute sample result from SumWatt
0177   0830           00581 Wneg     movf      Watt,W              ;get low byte to subtract
0178   02A8           00582          subwf     SumWatt,F ;subtract from sum
0179   0831           00583          movf      Watt+1,W  ;get middle byte to subtract
017A   1C03           00584          skpc                          ;if borrow from low byte, then:
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

017B   0F31           00585            incfsz   Watt+1,W  ;include borrow, if no overflow, then:
017C   02A9           00586            subwf    SumWatt+1,F         ;subtract middle byte from sum
017D   0832           00587           movf      Watt+2,W  ;get high 4 bits to add
017E   1C03           00588           skpc                         ;if borrow from high byte subtract
017F   0F32           00589            incfsz   Watt+2,W  ;include borrow to subtract
0180   02AA           00590            subwf    SumWatt+2,F         ;subtract high byte
0181   1C03           00591           skpc                         ;if borrow, then:
0182   03AB           00592            decf     SumWatt+3,F         ;propagate borrow
0183   0855           00593 chk_end   movf      Cycles,W  ;check remaining cycles
0184   1D03           00594           skpz                         ;if Cycles=0, then:
0185   282A           00595            goto     T2_wait             ;go and wait for next Timer 2 interrupt
0186   1112           00596           bcf       T2CON,TMR2ON        ;all done, stop timer 2       to signal completion
0187   109F           00597           bcf       ADCON0,GO ;abort A/D conversion for Ref sample
0188   130C           00598           bcf       PIR1,ADIF ;clear interrupt
                      00599 ;restore interrupted status
0189   0E71           00600 int_ret   swapf     int_S,W             ;get saved swappped STATUS register normal
018A   0083           00601           movwf     STATUS              ;restore interrupted STATUS register
018B   0EF0           00602           swapf     int_W,F             ;swap saved W (status unchanged)
018C   0E70           00603           swapf     int_W,W             ;restore interrupted W value normal
018D   0009           00604           retfie                        ;return to main program
                      00605 ;_______End of interrupt routines_____________________________________________
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00606           page
                     00607 ;_____________________________________________________________________________
                     00608 ;Subroutine to translate low hex nibble in W for 7 segment display.
                     00609 ;Returns segment value in W (0=a, 1=b, 2=c, 3=d, 4=e, 5=f, 6=g, 7=dp, active high)
                     00610 ;execution time: 7 µsec
                     00611 ; a    _     _  _     _  _  _  _  _  _     _        _  _  _                    _
                     00612 ;fgb | |   | _| _||_||_ |_   ||_||_||_||_ |   _  _||_ |_ |  |_||_   |  ||   _ |_||_ | |   |_| _
                     00613 ;edc |_|   ||_  _|  | _||_|  ||_| _|| ||_||_ |_ |_||_ |  |_|| || |  | _||_ |_||  |_ |_||_| _|
                     00614 ; dp  0  1  2  3  4  5  6  7  8  9  A  b  C  c  d  E  F  G  H  h  I  J  L  o  P  t  U  u  y  -
                     00615 ;hex: 3F 06 5B 4F 66 6D 7D 07 7F 6F 77 7C 39 58 5E 79 71 3D 76 74 06 06 36    73            40
018E   390F          00616 bcd7seg   andlw     0Fh       ;convert single BCD digit to 7 segment
018F   0782          00617           addwf     PCL,F
0190   343F          00618           retlw     3Fh       ;0
0191   3406          00619           retlw     06h       ;1
0192   345B          00620           retlw     5Bh       ;2
0193   344F          00621           retlw     4Fh       ;3
0194   3466          00622           retlw     66h       ;4
0195   346D          00623           retlw     6Dh       ;5
0196   347D          00624           retlw     7Dh       ;6
0197   3407          00625           retlw     07h       ;7
0198   347F          00626           retlw     7Fh       ;8
0199   346F          00627           retlw     6Fh       ;9
019A   3477          00628           retlw     77h       ;A (A)
019B   347C          00629           retlw     7Ch       ;B (b)
019C   3439          00630           retlw     39h       ;C (C)
019D   345E          00631           retlw     5Eh       ;D (d)
019E   3479          00632           retlw     79h       ;E (E)
019F   3471          00633           retlw     71h       ;F (F)
                     00634 ;_____________________________________________________________________________
01A0   0857          00635 fun2led   movf      sw_stat,W ;convert function switch code to led
01A1   390F          00636           andlw     0Fh       ;isolate push buttons
01A2   0782          00637           addwf     PCL,F
01A3   3410          00638           retlw     1<<4      ;0= Watt
01A4   3420          00639           retlw     1<<5      ;1= VA
01A5   3440          00640           retlw     1<<6      ;2= Freq
01A6   29B3          00641           goto      fun_Kf    ;3= Kf
01A7   3401          00642           retlw     1<<0      ;4= inc Va
01A8   3402          00643           retlw     1<<1      ;5= inc Vrms
01A9   3404          00644           retlw     1<<2      ;6= inc Ia
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

01AA   3408           00645           retlw     1<<3      ;7= inc Irms
01AB   3401           00646           retlw     1<<0      ;8= Va
01AC   3402           00647           retlw     1<<1      ;9= Vrms
01AD   3404           00648           retlw     1<<2      ;10= Ia
01AE   3408           00649           retlw     1<<3      ;11= Irms
01AF   3401           00650           retlw     1<<0      ;12= dec Va
01B0   3402           00651           retlw     1<<1      ;13= dec Vrms
01B1   3404           00652           retlw     1<<2      ;14= dec Ia
01B2   3408           00653           retlw     1<<3      ;15= dec Irms
                      00654
01B3   1858           00655 fun_Kf    btfsc     prefun,KfI          ;if previous function was I
01B4   3488           00656            retlw    1<<7|1<<3 ;show Irms and Kf indicator
01B5   3482           00657           retlw     1<<7|1<<1 ;show Vrms and Kf indicator
                      00658 ;_____________________________________________________________________________
01B6   0E57           00659 WsclLed   swapf     sw_stat,W ;convert Vscale and Iscale into Wscale
01B7   390F           00660           andlw     0Fh                 ;isolate Scale bits
01B8   0782           00661           addwf     PCL,F               ;Jump into table
01B9   3410           00662           retlw     1<<milli  ;5V & 100mA= 500 mW/mVA
01BA   3402           00663           retlw     1<<Scale2 ;50V & 100mA= 5.00 W/VA
01BB   3401           00664           retlw     1<<Scale1 ;500V & 100mA= 50.0 W/VA
01BC   3400           00665           retlw     0                   ;unused
01BD   3402           00666           retlw     1<<Scale2 ;5V & 1.00A= 5.00 W/VA
01BE   3401           00667           retlw     1<<Scale1 ;50V & 1.00A= 50.0 W/VA
01BF   3400           00668           retlw     0                   ;500V & 1.00A= 500 W/VA
01C0   3402           00669           retlw     1<<Scale2 ;unused
01C1   3401           00670           retlw     1<<Scale1 ;5V & 10.0A= 50.0 W/VA
01C2   3400           00671           retlw     0                   ;50V & 10.0A= 500 W/VA
01C3   3422           00672           retlw     1<<Scale2|1<<kilo   ;500V & 10.0= 5.00 kW/VA
01C4   3401           00673           retlw     1<<Scale1 ;unused
01C5   3400           00674           retlw     0                   ;5V & 100A= 500 W/VA
01C6   3422           00675           retlw     1<<Scale2|1<<kilo   ;50V & 100A= 5.00 kW
01C7   3421           00676           retlw     1<<Scale1|1<<kilo   ;500V & 100A= 50.0 kW
01C8   3420           00677           retlw     1<<kilo             ;unused
                      00678 ;_____________________________________________________________________________
                      00679 ;Macro to square-root 32 bit integer with.rounding to nearest root. by Ton Daas
                      00680 ; input: x:4, output: Root:2
                      00681 Sqrt_32   macro     x
                      00682           movf      x,W
                      00683           movwf     Square
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00684           movf      x+1,W
                     00685           movwf     Square+1
                     00686           movf      x+2,W
                     00687           movwf     Square+2
                     00688           movf      x+3,W
                     00689           movwf     Square+3
                     00690           call      Sqroot
                     00691           endm
                     00692 ;_____________________________________________________________________________
                     00693 ;Subroutine called by macro. Input: Square:4, Output: Root:2, Local: Counter, Remain,
                     00694 ; 45 (+21 for rounding) instructions, 629 (+21 for rounding) steps max.
01C9   01B4          00695 Sqroot    clrf      Root                ;clear initial root
01CA   01B5          00696           clrf      Root+1
01CB   01D2          00697           clrf      Remain              ;clear initial remainder
01CC   01D3          00698           clrf      Remain+1
01CD   3010          00699           movlw     16                  ;set number of bits to process
01CE   00D4          00700           movwf     Counter             ;set bit counter
                     00701 ;Test for next bit of Root
01CF   3040          00702 sqrt_lp   movlw     0x40                ;load test mask
01D0   022F          00703           subwf     Square+3,W          ;compare mask with msb of Square
01D1   0834          00704           movf      Root,W              ;get low byte of Root so far
01D2   1C03          00705           skpc                          ;if borrow, then:
01D3   0F34          00706            incfsz   Root,W              ;add borrow bit
01D4   0252          00707            subwf    Remain,W  ;compare Root low so far with remainder low so far
01D5   0835          00708           movf      Root+1,W  ;get high byte of Root so far
01D6   1C03          00709           skpc                          ;if borrow, then:
01D7   0F35          00710            incfsz   Root+1,W  ;add borrow bit
01D8   0253          00711            subwf    Remain+1,W          ;compare Root high so far with remainder high so far
01D9   1C03          00712           skpc                          ;if borrow (C=0), then:
01DA   29E5          00713            goto     sqrt_rl             ;skip subtraction and append root with 0
                     00714 ;Subtract root so far from Sumsqr and Rmdr
01DB   3040          00715           movlw     0x40                ;load test mask again
01DC   02AF          00716           subwf     Square+3,F          ;subtract mask from msb Square
01DD   0834          00717           movf      Root,W              ;get low byte of root so far
01DE   1C03          00718           skpc                          ;if borrow, then:
01DF   0F34          00719            incfsz   Root,W              ;add borrow bit
01E0   02D2          00720            subwf    Remain,F  ;subtract root low so far from remainder low so far
01E1   0835          00721           movf      Root+1,W  ;get high byte of root so far
01E2   1C03          00722           skpc                          ;if borrow, then:
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

01E3   0F35           00723            incfsz   Root+1,W  ;add borrow bit
01E4   02D3           00724            subwf    Remain+1,F          ;subtract root high so far from remainder (C=1)
01E5   0DB4           00725 sqrt_rl    rlf      Root,F              ;append new root bit
01E6   0DB5           00726            rlf      Root+1,F  ;and propagate previous bits, C=0
01E7   0DAC           00727            rlf      Square,F  ;shift Square up 1 bit
01E8   0DAD           00728            rlf      Square+1,F          ;and second byte,
01E9   0DAE           00729            rlf      Square+2,F          ;and third byte
01EA   0DAF           00730            rlf      Square+3,F          ;and fourth byte, shift msb into Remainder
01EB   0DD2           00731            rlf      Remain,F  ;append remainder
01EC   0DD3           00732            rlf      Remain+1,F          ;shift up other bits, C=0
01ED   0DAC           00733            rlf      Square,F  ;shift Square up 2nd bit
01EE   0DAD           00734            rlf      Square+1,F          ;and second byte
01EF   0DAE           00735            rlf      Square+2,F          ;and third byte
01F0   0DAF           00736            rlf      Square+3,F          ;and fourth byte, shift into Remainder
01F1   0DD2           00737            rlf      Remain,F  ;append remainder second bit
01F2   0DD3           00738            rlf      Remain+1,F          ;shift high byte up, C=0 (except for last itteration)
01F3   0BD4           00739            decfsz   Counter,F ;if not all 16 bits done, then:
01F4   29CF           00740             goto    sqrt_lp             ;repeat procedure
                      00741 ;Round result to nearest integer
01F5   1803           00742            skpnc                        ;if Carry shifted out of Remain, then:
01F6   2A03           00743             goto    sqrt_ru             ;skip test as remainder > root
01F7   3040           00744            movlw    0x40                ;load test mask
01F8   022F           00745            subwf    Square+3,W          ;compare mask with msb of Square fraction
01F9   0834           00746            movf     Root,W              ;get low byte of Root so far
01FA   1C03           00747            skpc                         ;if borrow, then:
01FB   0F34           00748             incfsz  Root,W              ;add borrow bit
01FC   0252           00749             subwf   Remain,W  ;compare Root low so far with remainder low so far
01FD   0835           00750            movf     Root+1,W  ;get high byte of Root so far
01FE   1C03           00751            skpc                         ;if borrow, then:
01FF   0F35           00752             incfsz  Root+1,W  ;add borrow bit
0200   0253           00753             subwf   Remain+1,W          ;compare Root high so far with remainder high so far
0201   1C03           00754            skpc                         ;if borrow, then:
0202   0008           00755             return                      ;done, keep rounded down
0203   0AB4           00756 sqrt_ru    incf     Root,F              ;round up
0204   1903           00757            skpnz                        ;if overflow, then:
0205   0FB5           00758             incfsz  Root+1,F  ;propagate carry, if result is not 0, then:
0206   0008           00759             return                      ;done, if result 0, then:
0207   03B4           00760            decf     Root,F              ;undo rounding up
0208   03B5           00761            decf     Root+1,F  ;and its high byte
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0209   0008           00762           return
                      00763 ;_____________________________________________________________________
                      00764 ; subroutine to Multiply 24 bit by 8 bit unsigned integers. by Ton Daas
                      00765 ; Input: Product:3, W, Output: Product:4 (32 bits), Local: Counter
                      00766 ; 14 instructions, 245 steps
020A   01D4           00767 Mul24x8   clrf      Counter
020B   1654           00768           bsf       Counter,4 ;set Counter to 16
020C   15D4           00769           bsf       Counter,3 ;set Counter to 24
020D   01B7           00770           clrf      Product+3
020E   1003           00771 Mul24lp   clrc
020F   1834           00772           btfsc     Product,0
0210   07B7           00773            addwf    Product+3,F
0211   0CB7           00774           rrf       Product+3,F
0212   0CB6           00775           rrf       Product+2,F
0213   0CB5           00776           rrf       Product+1,F
0214   0CB4           00777           rrf       Product,F
0215   0BD4           00778           decfsz    Counter,F
0216   2A0E           00779            goto     Mul24lp
0217   0008           00780           return
                      00781 ;_____________________________________________________________________
                      00782 ; subroutine to Multiply 16 bit by 16 bit unsigned integers. by Ton Daas
                      00783 ; Input: Divisor:2, Product:2, local: Counter, Output: Product:4 (32 bits)
                      00784 ; 20 instructions, 181 to 261 steps
0218   3010           00785 Mul_16    movlw     16
0219   00D4           00786           movwf     Counter
021A   01B6           00787           clrf      Product+2
021B   01B7           00788           clrf      Product+3
021C   1003           00789 M16_lp    clrc
021D   1C34           00790           btfss     Product,0
021E   2A25           00791            goto     M16_sr
021F   0850           00792           movf      Divisor,W
0220   07B6           00793           addwf     Product+2,F
0221   0851           00794           movf      Divisor+1,W
0222   1803           00795           skpnc
0223   0F51           00796            incfsz   Divisor+1,W
0224   07B7           00797            addwf    Product+3,F
0225   0CB7           00798 M16_sr    rrf       Product+3,F         ;shift down, get C back in
0226   0CB6           00799           rrf       Product+2,F
0227   0CB5           00800           rrf       Product+1,F
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0228   0CB4           00801           rrf       Product,F
0229   0BD4           00802           decfsz    Counter,F
022A   2A1C           00803            goto     M16_lp
022B   0008           00804           return
                      00805 ;_____________________________________________________________________
                      00806 ; subroutine to Multiply 8 bit by 16 bit unsigned integers. by Ton Daas
                      00807 ; Input: W and Product:2, Output: Product:3 (24 bits), Local: Counter
                      00808 ; 12 instructions, 16*9+3+2= 149 steps (including return)
022C   01D4           00809 Mul16x8   clrf      Counter
022D   1654           00810           bsf       Counter,4 ;set Counter to 16
022E   01B6           00811           clrf      Product+2
022F   1003           00812 Mul16lp   clrc
0230   1834           00813           btfsc     Product,0
0231   07B6           00814            addwf    Product+2,F
0232   0CB6           00815           rrf       Product+2,F         ;shift down, get C in bit 23
0233   0CB5           00816           rrf       Product+1,F         ;shift down
0234   0CB4           00817           rrf       Product,F ;shift down
0235   0BD4           00818           decfsz    Counter,F
0236   2A2F           00819            goto     Mul16lp
0237   0008           00820           return
                      00821 ;_____________________________________________________________________
                      00822 ; Routine to divide 32 bits by 16 bits unsigned integers with rounding
                      00823 ; Dividend in Dividen (32 bits), Divisor in Divisor (16 bits),
                      00824 ; Quotient replaces Dividen (32 bits), Remainder in Remain (16 bits)
                      00825 ; WARNING! If divisor= 0, then Quotient= 0xFFFFFFFF
                      00826 ; 42 instructions, 483 to 642 cycles (incl. return)
0238   3021           00827 dv32_16   movlw     33
0239   00D4           00828           movwf     Counter             ;set bitcount
023A   01D3           00829           clrf      Remain+1  ;clear remainder
023B   01D2           00830           clrf      Remain
023C   1003           00831           clrc
023D   0DB4           00832 d32loop   rlf       Dividen,F ;shift left dividend and Quotient
023E   0DB5           00833           rlf       Dividen+1,F         ;
023F   0DB6           00834           rlf       Dividen+2,F         ;shift msb into C
0240   0DB7           00835           rlf       Dividen+3,F
0241   0DD2           00836           rlf       Remain,F  ; and then into remainder
0242   0DD3           00837           rlf       Remain+1,F
0243   1803           00838           skpnc                         ;if remainder has overflow, then:
0244   2A4D           00839            goto     d32sub              ;skip test as remainder > Divisor
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0245   0850           00840           movf      Divisor,W
0246   0252           00841           subwf     Remain,W  ;compare Divisor with Remainder low byte
0247   0851           00842           movf      Divisor+1,W
0248   1C03           00843           skpc                          ;if borrow, then:
0249   0F51           00844            incfsz   Divisor+1,W         ;include borrow bit, if 0, skip subtract and maintain C
024A   0253           00845            subwf    Remain+1,W
024B   1C03           00846           skpc                          ;if borrow,then:
024C   2A54           00847            goto     d32save             ;skip subtraction, shift in 0 on next shift
024D   0850           00848 d32sub    movf      Divisor,W
024E   02D2           00849           subwf     Remain,F  ;actually perform subtract
024F   0851           00850           movf      Divisor+1,W
0250   1C03           00851           skpc
0251   0F51           00852            incfsz   Divisor+1,W
0252   02D3           00853            subwf    Remain+1,F
0253   1403           00854           setc                          ;shift in 1 on next shift
0254   0BD4           00855 d32save   decfsz    Counter,F
0255   2A3D           00856            goto     d32loop
                      00857 ;round to nearest integer
0256   1C03           00858           skpc                          ;if remainder < divisor, then:
0257   0008           00859            return                       ;keep rounded down result
0258   0FB4           00860           incfsz    Dividen,F ;round up, if no carry, then:
0259   0008           00861            return                       ;done, else:
025A   0FB5           00862           incfsz    Dividen+1,F         ;include middle byte, if no carry, then:
025B   0008           00863            return                       ;done, else:
025C   0FB6           00864           incfsz    Dividen+2,F         ;also include high byte, if no overflow, then;
025D   0008           00865            return                       ;done, else:
025E   0FB7           00866           incfsz    Dividen+3,F
025F   0008           00867            return
0260   03B4           00868           decf      Dividen,F ;undo rounding, since this is largest integer
0261   03B5           00869           decf      Dividen+1,F
0262   03B6           00870           decf      Dividen+2,F
0263   03B7           00871           decf      Dividen+3,F
0264   0008           00872           return
                      00873 ;_____________________________________________________________________________
                      00874 ; Routine to divide 24 bits by 16 bits unsigned integers with rounding
                      00875 ; Dividend in Dividen (24 bits), Divisor in Divisor (16 bits),
                      00876 ; Quotient replaces Dividen (24 bits), Remainder in Remain (16 bits)
                      00877 ; WARNING! If divisor= 0, then Quotient= 0xFFFFFF
                      00878 ; 42 instructions, 483 to 642 cycles (incl. return)
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0265   3019           00879 dv24_16   movlw     25
0266   00D4           00880           movwf     Counter             ;set bitcount
0267   01D2           00881           clrf      Remain              ;clear remainder
0268   01D3           00882           clrf      Remain+1
0269   1003           00883           clrc
026A   0DB4           00884 d24loop   rlf       Dividen,F ;shift left dividend
026B   0DB5           00885           rlf       Dividen+1,F         ; and Quotient
026C   0DB6           00886           rlf       Dividen+2,F         ;shift msb into C
026D   0DD2           00887           rlf       Remain,F  ; and then into remainder
026E   0DD3           00888           rlf       Remain+1,F
026F   1803           00889           skpnc                         ;if remainder has overflow, then:
0270   2A79           00890            goto     d24sub              ;ignore test as remainder > Divisor
0271   0850           00891           movf      Divisor,W
0272   0252           00892           subwf     Remain,W  ;compare Divisor with Remainder low byte
0273   0851           00893           movf      Divisor+1,W
0274   1C03           00894           skpc                          ;if borrow, then:
0275   0F51           00895            incfsz   Divisor+1,W         ;include borrow, if 0, skip subtract and maintain C
0276   0253           00896            subwf    Remain+1,W
0277   1C03           00897           skpc                          ;if borrow,then:
0278   2A80           00898            goto     d24save             ;skip subtraction, shift in 0 on next shift
0279   0850           00899 d24sub    movf      Divisor,W
027A   02D2           00900           subwf     Remain,F  ;actually perform subtract
027B   0851           00901           movf      Divisor+1,W
027C   1C03           00902           skpc
027D   0F51           00903            incfsz   Divisor+1,W
027E   02D3           00904            subwf    Remain+1,F
027F   1403           00905           setc                          ;shift in 1 on next shift
0280   0BD4           00906 d24save   decfsz    Counter,F
0281   2A6A           00907            goto     d24loop
                      00908 ;round to nearest integer
0282   1C03           00909           skpc                          ;if remainder < divisor, then:
0283   0008           00910            return                       ;keep result rounded down
0284   0FB4           00911           incfsz    Dividen,F ;round up, if no overflow, then:
0285   0008           00912            return                       ;done, else:
0286   0FB5           00913           incfsz    Dividen+1,F         ;include middle byte, if no overflow, then:
0287   0008           00914            return                       ;done, else:
0288   0FB6           00915           incfsz    Dividen+2,F         ;also include high byte, if no overflow, then;
0289   0008           00916            return                       ;done, else:
028A   03B4           00917           decf      Dividen,F ;undo roundup, since this is largest integer
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

028B   03B5           00918           decf      Dividen+1,F
028C   03B6           00919           decf      Dividen+2,F
028D   0008           00920           return
                      00921 ;_____________________________________________________________________________
                      00922 ; Routine to divide 16 bits by 8 bits unsigned integers and round result.
                      00923 ; Dividend in Dividen (16 bits), Divisor in Divisor,
                      00924 ; Quotient replaces Dividend (16 bits), Remainder in Remain
                      00925 ; WARNING! If divisor= 0, then Quotient= 0xFFFF
                      00926 ; 27 instructions, 210 to 265 steps (incl. return)
028E   3011           00927 div16_8   movlw     17
028F   00D4           00928           movwf     Counter             ;set bitcount
0290   01D2           00929           clrf      Remain
0291   1003           00930           clrc
0292   0DB4           00931 d16loop   rlf       Dividen,F ;shift left dividend bit out and Quotient bit in
0293   0DB5           00932           rlf       Dividen+1,F         ;and high byte
0294   0DD2           00933           rlf       Remain,F  ; and then into remainder
0295   1803           00934           skpnc                         ;if remainder has overflow, then:
0296   2A9B           00935            goto     d16sub              ;skip test as remainder > Divisor
0297   0850           00936           movf      Divisor,W ;get divisor
0298   0252           00937           subwf     Remain,W  ;compare Divisor with Remainder
0299   1C03           00938           skpc                          ;if borrow,then:
029A   2A9E           00939            goto     d16save             ;skip subtraction, leave C=0 for next shift
029B   0850           00940 d16sub    movf      Divisor,W ;get divisor
029C   02D2           00941           subwf     Remain,F  ;subtract from remainder
029D   1403           00942           setc                          ;set C=1 for next shift
029E   0BD4           00943 d16save   decfsz    Counter,F
029F   2A92           00944            goto     d16loop
                      00945 ;round to nearest integer
02A0   1C03           00946           skpc                          ;if no binairy fraction, then:
02A1   0008           00947            return                       ;done
02A2   0FB4           00948           incfsz    Dividen,F ;round up, if no carry, then:
02A3   0008           00949            return                       ;done, else:
02A4   0FB5           00950           incfsz    Dividen+1,F         ;including high byte, if no overflow, then:
02A5   0008           00951            return                       ;done, else:
02A6   03B4           00952           decf      Dividen,F ;undo round up
02A7   03B5           00953           decf      Dividen+1,F
02A8   0008           00954           return
                      00955 ;_____________________________________________________________________________
                      00956 ;Subroutine to convert 16 bit binary to 5 unpacked BCD bytes
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00957 ; From Microchip PICDEM 2 PLUS DEMO BOARD. Uses number in NumH,NumL,
                     00958 ; Returns BCD in TenK,OneK,Huns,Tens,Ones. C=0 if succesfull
                     00959 ; Subroutine Requires 50 instructions and returns in about 160 cycles.
02A9   0E35          00960 b16_bcd   swapf     NumH,W    ;W  = A2*16+A3
02AA   38F0          00961           iorlw     0xF0      ;= andlw 0x0F ;W = A3         +addlw 0xF0 ;W = A3-16
02AB   00DC          00962           movwf     OneK      ;B3 = A3-16
02AC   07DC          00963           addwf     OneK,F    ;B3 = 2*(A3-16) = 2A3 - 32
02AD   3EE2          00964           addlw     .226      ;W  = A3-16 - 30 = A3-46
02AE   00DD          00965           movwf     Huns      ;B2 = A3-46
02AF   3E32          00966           addlw     .50       ;W  = A3-46 + 50 = A3+4
02B0   00DF          00967           movwf     Ones      ;B0 = A3+4
02B1   0835          00968           movf      NumH,W    ;W  = A3*16+A2
02B2   390F          00969           andlw     0x0F      ;W  = A2
02B3   07DD          00970           addwf     Huns,F    ;B2 = A3-46 + A2 = A3+A2-46
02B4   07DD          00971           addwf     Huns,F    ;B2 = A3+A2-46  + A2 = A3+2A2-46
02B5   07DF          00972           addwf     Ones,F    ;B0 = A3+4 + A2 = A3+A2+4
02B6   3EE9          00973           addlw     .233      ;W  = A2 - 23
02B7   00DE          00974           movwf     Tens      ;B1 = A2-23
02B8   07DE          00975           addwf     Tens,F    ;B1 = 2*(A2-23)
02B9   07DE          00976           addwf     Tens,F    ;B1 = 3*(A2-23) = 3A2-69 (Doh! thanks NG)
02BA   0E34          00977           swapf     NumL,W    ;W  = A0*16+A1
02BB   390F          00978           andlw     0x0F      ;W  = A1
02BC   07DE          00979           addwf     Tens,F    ;B1 = 3A2-69 + A1 = 3A2+A1-69 range -69...-9
02BD   07DF          00980           addwf     Ones,F    ;B0 = A3+A2+4 + A1 = A3+A2+A1+4 and Carry = 0 (thanks NG)
02BE   0DDE          00981           rlf       Tens,F    ;B1 = 2*(3A2+A1-69) + C = 6A2+2A1-138 and
                     00982                               ;Carry is now 1 as tens register had to be negitive
02BF   0DDF          00983           rlf       Ones,F    ;B0 = 2*(A3+A2+A1+4) + C = 2A3+2A2+2A1+9
                     00984                               ; (+9 not +8 due to the carry from prev line, Thanks NG)
02C0   09DF          00985           comf      Ones,F    ;B0 = ~(2A3+2A2+2A1+9) = -2A3-2A2-2A1-10
                     00986                               ; (one's complement plus 1 is two's complement. Thanks SD)
                     00987 ;;Ones    = -1 * Ones - 1
                     00988 ;;        = - 2 * (A3 + A2 + A1) - 9 - 1
                     00989 ;;        = - 2 * (A3 + A2 + A1) - 10
02C1   0DDF          00990           rlf     Ones,F      ;B0 = 2*(-2A3-2A2-2A1-10) = -4A3-4A2-4A1-20
02C2   0834          00991           movf      NumL,W    ;W  = A1*16+A0
02C3   390F          00992           andlw     0x0F      ;W  = A0
02C4   07DF          00993           addwf     Ones,F    ;B0 = -4A3-4A2-4A1-20 + A0 = A0-4(A3+A2+A1)-20
                     00994                               ; range -215...-5 Carry=0
02C5   0DDC          00995           rlf       OneK,F    ;B3 = 2*(2A3 - 32) = 4A3 - 64
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

02C6   3007           00996           movlw     0x07      ;W  = 7
02C7   00DB           00997           movwf     TenK      ;B4 = 7
                      00998 ;B0 = A0-4(A3+A2+A1)-20       ;-5...-200
                      00999 ;B1 = 6A2+2A1-138   ;-18...-138
                      01000 ;B2 = A3+2A2-46               ;-1...-46
                      01001 ;B3 = 4A3-64                  ;-4...-64
                      01002 ;B4 = 7                       ;7
                      01003 ; At this point, the original number is equal to TenK*10000+OneK*1000+Huns*100+Tens*10+Ones
                      01004 ; if those entities are regarded as two's compliment binary.
                      01005 ; To be precise, all of them are negative except TenK. Now the number needs to be normalized,
                      01006 ; but this can all be done with simple byte arithmetic.
02C8   300A           01007           movlw     .10       ;w  = 10
02C9   03DE           01008 Lb1       decf      Tens,F    ; B1 -= 1
02CA   07DF           01009           addwf     Ones,F    ; B0 += 10
02CB   1C03           01010           skpc
02CC   2AC9           01011            goto     Lb1       ; while B0 < 0
02CD   03DD           01012 Lb2       decf      Huns,F    ; B2 -= 1
02CE   07DE           01013           addwf     Tens,F    ; B1 += 10
02CF   1C03           01014           skpc
02D0   2ACD           01015            goto     Lb2       ; while B1 < 0
02D1   03DC           01016 Lb3       decf      OneK,F    ; B3 -= 1
02D2   07DD           01017           addwf     Huns,F    ; B2 += 10
02D3   1C03           01018           skpc
02D4   2AD1           01019            goto     Lb3       ; while B2 < 0
02D5   03D2           01020 Lb4       decf      Remain,F ;TenK      ; B4 -= 1
02D6   07DC           01021           addwf     OneK,F    ; B3 += 10
02D7   1C03           01022           skpc
02D8   2AD5           01023            goto     Lb4       ; while B3 < 0
02D9   025B           01024           subwf     TenK,W
02DA   0008           01025           return              ;C=1 if value too large
                      01026 ;_____________________________________________________________________________
                      01027 ;Read byte at address in W from EEPROM in W
02DB   1683           01028 reeprom   bsf       STATUS,RP0          ;select bank 1
02DC   009B           01029           movwf     EEADR               ;set address to read
02DD   1003           01030           clrc
02DE   141C           01031           bsf       EECON1,RD ;perform EEPROM read
02DF   181C           01032           btfsc     EECON1,RD ;if EEPROM Data not available yet, then:
02E0   2ADF           01033            goto     $-1                 ;wait for it
02E1   081A           01034           movf      EEDAT,W             ;read byte in W
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

02E2   1283           01035           bcf       STATUS,RP0          ;reselect bank 0
02E3   0008           01036           return                        ;if byte =0, tnen Z=1
                      01037
                      01038 ;Write byte in INDF to EEPROM at address in W
02E4   1683           01039 weeprom   bsf       STATUS,RP0          ;select data bank 1
02E5   009B           01040           movwf     EEADR               ;set Data Memory Address to write
02E6   0800           01041           movf      INDF,W              ;get data byte for EEPROM
02E7   009A           01042           movwf     EEDAT               ;set Data Memory Value to write
02E8   189C           01043           btfsc     EECON1,WR ;check if previous write complete, if not, then:
02E9   2AE8           01044            goto     $-1                 ;wait for complete
02EA   138B           01045           bcf       INTCON,GIE          ;Disable all interrupts
02EB   1B8B           01046           btfsc     INTCON,GIE
02EC   2AEA           01047            goto     $-2                 ;repeat as disable may fail in some cases
02ED   151C           01048           bsf       EECON1,WREN         ;Enable writes
02EE   3055           01049           MOVLW     0x55
02EF   009D           01050           MOVWF     EECON2              ;Write 55h
02F0   30AA           01051           MOVLW     0xAA
02F1   009D           01052           MOVWF     EECON2              ;Write AAh
02F2   149C           01053           bsf       EECON1,WR ;Set WR bit to begin write
02F3   111C           01054           bcf       EECON1,WREN         ;Disable writes
02F4   178B           01055           bsf       INTCON,GIE          ;Enable INTs.
02F5   1283           01056           bcf       STATUS,RP0
02F6   0008           01057           return
                      01058 ;_____________________________________________________________________________
                      01059 ; Subroutine to write command/data to TM1637 by Ton Daas
                      01060 ; Entrypoint for command is: TM_cmd and for data write is: TM_wrt.
                      01061 ; On entry TM_dat has command or byte to send.
                      01062 ; Assume CLK and DIO are both set as input and TM_CLK and TM_DIO are both high.
                      01063 ; Command byte:     0100tfr0  Data      t: 0=normal, 1=test mode,
                      01064 ;                                           f: 0=auto increment, 1=no increment,
                      01065 ;                                           r: 0=write, 1=read
                      01066 ;                   1100aaaa  Address   aaaa=Character address (0= GRID1 > 5= GRID6)
                      01067 ;                                           6 or 7 are ignored
                      01068 ;                   1000dwww  Display and control d: 0=off, 1=on
                      01069 ;                                           w: duty cycle       000=1/16, 001=1/8, 010=1/4
                      01070 ;                                           011=5/8, 100=11/16, 101=3/4, 110=13/16, 111=7/8
                      01071 ; TM_cmd 4 words, duration 4+98+7= 102 µs, assuming processor is 4 MHz
02F7   30F3           01072 TM_cmd    movlw     ~(1<<TM_CLK|1<<TM_DIO)
02F8   0587           01073           andwf     PORTC,F             ;  |        |       ensure CLK and DIO latch are low
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

02F9   1683          01074           bsf       STATUS,RP0          ;  |        |       select bank 1
02FA   1187          01075           bcf       TRISC,TM_DIO        ;  |      -<-       force DIO output low (start bit)
                     01076 ;TM_wrt 13 words, duration 3+11*8+7= 98 µs
02FB   3008          01077 TM_wrt    movlw     8                   ;  |      |         load bit count
02FC   00FD          01078           movwf     TM_cnt              ;  |      |         set bit count in TM_cnt
02FD   1683          01079           bsf       STATUS,RP0          ;  |      |         ensure bank 1 is selected
02FE   1107          01080 TM_wlp    bcf       TRISC,TM_CLK        ;-<-     | ?       lower CLK, keep other latches low
02FF   0CFC          01081           rrf       TM_dat,F  ;|        | ?      get LSB into C
0300   1803          01082           skpnc                         ;|       | ?
0301   1587          01083            bsf      TRISC,TM_DIO        ;|       ->-       allow DIO high (by pull-up)
0302   1C03          01084           skpc                          ;|       | |      if C=0, then:
0303   1187          01085            bcf      TRISC,TM_DIO        ;|       -<-       force DIO output low
0304   0000          01086           nop                           ;|       | |      delay
0305   1507          01087           bsf       TRISC,TM_CLK        ;->-     | |      allow CLK high (bit n)
0306   0BFD          01088           decfsz    TM_cnt,F  ;  |      | |      if not yet bit 7, then:
0307   2AFE          01089            goto     TM_wlp              ;  |      | |      repeat
                     01090 ;TM_ack 6 words, duration 7 µs
0308   1107          01091 TM_ack    bcf       TRISC,TM_CLK        ;-<-     | |      lower CLK last bit
0309   1587          01092           bsf       TRISC,TM_DIO        ;|       ->-       allow DIO high (by pull-up)
030A   0000          01093           nop                           ;|         |      delay
030B   1507          01094           bsf       TRISC,TM_CLK        ;->-       |      end byte transfer
030C   1283          01095           bcf       STATUS,RP0          ;  |       |      reselect bank 0
030D   0008          01096           return                        ;done
                     01097 ;_____________________________________________________________________________
                     01098 ; Subroutine to read byte from TM1637. Returns key scan code in TM_dat.
                     01099 ; Entrypoint is: TM_rdk
                     01100 ; Assume CLK is initialized as output and high, and DIO as input.
                     01101 ; TM_rdk 14 words, duration 4+10*8+9= 93 µs, assuming processor is 4 MHz
030E   1683          01102 TM_rdk    bsf       STATUS,RP0          ;  |      select bank 1
030F   01FC          01103           clrf      TM_dat              ;  |      initialize all bits 0 in result
0310   1403          01104           setc                          ;  |      set terminal marker
0311   0CFC          01105           rrf       TM_dat,F  ;  |      shift marker in collect byte, clear carry
0312   1107          01106 TM_rlp    bcf       TRISC,TM_CLK        ;-<-     lower CLK, keep DIO latch low
0313   1283          01107           bcf       STATUS,RP0          ;|       reselect bank 0
0314   1987          01108           btfsc     PORTC,TM_DIO        ;|       check data input; if high, then:
0315   1403          01109            setc                         ;|       set carry
0316   1683          01110           bsf       STATUS,RP0          ;|       select bank 1
0317   0CFC          01111           rrf       TM_dat,F  ;|       shift marker and rotate data bit in
0318   1507          01112           bsf       TRISC,TM_CLK        ;->-     allow CLK to raise
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0319   1C03           01113           skpc                            ;  |      if not last bit received, then:
031A   2B12           01114            goto     TM_rlp                ;  |      repeat bit read
031B   2B08           01115           goto      TM_ack                ;  |      append with acknowledge sequence
                      01116 ;_____________________________________________________________________________
                      01117 ; Subroutine to terminate transmission with TM1637.
                      01118 ; Entrypoint is: TM_stp
                      01119 ; TM_stp 9 words, duration 10 µs, assuming processor is 4 MHz
031C   1683           01120 TM_stp    bsf       STATUS,RP0            ;  |      |         select bank 1
031D   1107           01121           bcf       TRISC,TM_CLK          ;-<-     |         force CLK low
031E   1187           01122           bcf       TRISC,TM_DIO          ;|       |         force DIO low
031F   0000           01123           nop                             ;|       |
0320   1507           01124           bsf       TRISC,TM_CLK          ;->-     |         allow CLK high
0321   0000           01125           nop                             ;  |     |
0322   1587           01126           bsf       TRISC,TM_DIO          ;  |     ->-       allow DIO go high (by pull-up)
0323   1283           01127           bcf       STATUS,RP0            ;  |       |       reselect bank 0
0324   0008           01128           return                          ;done
                      01129 ;_____________________________________________________________________________
                      01130 ;subroutine to read mode switch status
0325   3042           01131 read_sw   movlw     42h                   ;read key data command
0326   00FC           01132           movwf     TM_dat                ;load in TM_dat
0327   22F7           01133           call      TM_cmd    ;161µs      ;send read data command
0328   230E           01134           call      TM_rdk    ; 79µs      ;call read data routine
0329   231C           01135           call      TM_stp    ; 14µs      ;terminate data routine
032A   097C           01136           comf      TM_dat,W  ;get scan code complement in W
032B   1903           01137           skpnz                           ;if no key is pressed, then:
032C   2B2F           01138            goto     sw_wait               ;do not change selection
032D   3EF8           01139           addlw     -8                    ;convert scancodes to below 16
032E   00D7           01140           movwf     sw_stat               ;save new mode request
032F   0181           01141 sw_wait   clrf      TMR0                  ;reset Timer 0
0330   110B           01142           bcf       INTCON,T0IF           ;clear any pending Timer 0 interrupt
0331   1D0B           01143           btfss     INTCON,T0IF           ;if no Timer 0 overflow, then:
0332   2B31           01144            goto     $-1                   ;wait for it, delay ±1 ms
0333   0008           01145           return                          ;done
                      01146 ;_____________________________________________________________________________
                      01147 ;subroutine to delay 10 µs
0334   3004           01148 wait_10   movlw     4
0335   3EFF           01149           addlw     -1
0336   1D03           01150           skpz
0337   2B35           01151            goto     $-2
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0338   0008           01152          return
                      01153 ;_____________________________________________________________________________
                      01154 ;Subroutine to get Voltage scale switches on C2IN+ (0= 0V, 1= 1.2 V, 2= 2,4 V)
                      01155 ; and apply scaling (divide by 2560 ±5%)
                      01156 ;input: Dividen:24, Output:Dividen:16
0339   30CF           01157 Vsclget   movlw     0xCF                ;mask to protect other bits
033A   05D7           01158          andwf     sw_stat,F ;remove old Vscale bits
033B   302A           01159          movlw     2|1<<C2INV|1<<CIS ;comparator mode 2, C2 inverted, C2IN+ (ANS4/RC0) on C2-
033C   0099           01160          movwf     CMCON0              ;turn on comparator with VRef on C2+ and Vscale on C2-
033D   1683           01161          bsf       STATUS,RP0          ;select register bank 1
033E   30A3           01162          movlw     0xA3                ;Voltage reference for 3/24*Vdd = 0,625 V
033F   0099           01163          movwf     VRCON               ;enable Voltage reference to detect scale 0 (0V)
0340   1283           01164          bcf       STATUS,RP0          ;reselect register bank 0
0341   2334           01165          call      wait_10             ;delay to allow Comparator and Vref to settle
0342   1F99           01166          btfss     CMCON0,C2OUT        ;if Comparator 2 Output= 0 (Vscale < 0.625 V), then:
0343   2B4D           01167           goto     Vsclok              ;leave Voltage scale= 0
0344   1683           01168          bsf       STATUS,RP0          ;select register bank 1
0345   30A9           01169          movlw     0xA9                ;Voltage reference for 9/24*Vdd = 1,875 V
0346   0099           01170          movwf     VRCON               ;set Voltage reference to detect scale 1 (2.4 V)
0347   1283           01171          bcf       STATUS,RP0          ;reselect register bank 0
0348   2334           01172          call      wait_10             ;delay to allow Vref to settle
0349   1F99           01173          btfss     CMCON0,C2OUT        ;if C2OUT=0 (Vscale < 2,4 V). then:
034A   1657           01174           bsf      sw_stat,Vscale1     ;set Voltage scale= 1
034B   1B99           01175          btfsc     CMCON0,C2OUT        ;if C2OUT=1 (Vscale > 2.4 V), then:
034C   16D7           01176           bsf      sw_stat,Vscale2     ;set Voltage scale= 2
034D   1683           01177 Vsclok    bsf       STATUS,RP0          ;select register bank 1
034E   0199           01178          clrf      VRCON               ;disable Voltage reference
034F   1283           01179          bcf       STATUS,RP0          ;reselect register bank 0
0350   0199           01180          clrf      CMCON0              ;disable Comparators
                      01181 ;get scale adjustment value
0351   304D           01182          movlw     Vsc0cal             ;get eeprom address for Voltage scale 0 calibration
0352   1A57           01183          btfsc     sw_stat,Vscale1     ;if scale is 1, then:
0353   304E           01184           movlw    Vsc1cal             ;get eeprom address for Voltage scale 1 calibration
0354   1AD7           01185          btfsc     sw_stat,Vscale2     ;if scale is 2, then:
0355   304F           01186           movlw    Vsc2cal             ;get eeprom address for Voltage scale 2 calibration
0356   22DB           01187          call      reeprom             ;get current calibration value for selected scale
0357   00D0           01188          movwf     Divisor             ;save calibration value
0358   1D57           01189          btfss     sw_stat,adjust      ;if not in adjust mode, then:
0359   2B6F           01190           goto     skpVadj             ;skip adjustment
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     01191 ;adjust calibration value
035A   1DD7          01192           btfss     sw_stat,adj_dec     ;if inc request, then:
035B   2B61          01193            goto     decVadj
035C   3A7F          01194           xorlw     0x7F                ;check for max postive
035D   1903          01195           skpnz                         ;if max, then:
035E   2B6D          01196            goto     endVadj             ;quit adj
035F   0AD0          01197           incf      Divisor,F ;increment calibration value
0360   2B65          01198           goto      savVadj
                     01199
0361   3A80          01200 decVadj   xorlw     0x80                ;check for min negative
0362   1903          01201           skpnz                         ;if min, then:
0363   2B6D          01202            goto     endVadj             ;quit adj
0364   03D0          01203           decf      Divisor,F
0365   3050          01204 savVadj   movlw     Divisor             ;get pointer of new scale calibration data
0366   0084          01205           movwf     FSR                 ;set pointer to data to write
0367   304D          01206           movlw     Vsc0cal             ;assume scale 0
0368   1A57          01207           btfsc     sw_stat,Vscale1     ;if scale is 1, then:
0369   304E          01208            movlw    Vsc1cal             ;replace with eeprom address for scale 1 adjust
036A   1AD7          01209           btfsc     sw_stat,Vscale2     ;if scale is 2, then:
036B   304F          01210            movlw    Vsc2cal             ;replace with eeprom address for scale 2 adjust
036C   22E4          01211           call      weeprom             ;write new adjustment value for selected scale
036D   1157          01212 endVadj   bcf       sw_stat,adjust      ;acknowledge adjust
036E   15D7          01213           bsf       sw_stat,adj_dec     ;also set adjust direction bit for normal display
                     01214 ;Apply calibration adjustment and Divide by 2560 ±5%
036F   300A          01215 skpVadj   movlw     high 2560 ;get Divisor high byte
0370   00D1          01216           movwf     Divisor+1
0371   1BD0          01217           btfsc     Divisor,7 ;if fine adjustment value is negative, then:
0372   03D1          01218            decf     Divisor+1,F         ;propagate sign
0373   2A65          01219           goto      dv24_16             ;continue with divide routtine
                     01220 ;_____________________________________________________________________________
                     01221 ;Subroutine to get Current scale switches on C2IN- (0= 0V, 1= 1.2V, 2= 2,53V, 3= 5V)
                     01222 ; and apply scaling (divide by 2560 ±5%)
                     01223 ;input: Dividen:24, Output:Dividen:16   scale detections 3/24, 9/24, 15/24
0374   303F          01224 Isclget   movlw     0x3F                ;isolate bit mask
0375   05D7          01225           andwf     sw_stat,F ;remove old Iscale bits
0376   3022          01226           movlw     2|1<<C2INV          ;comparator mode 2, C2 inverted, C2IN- (ANS5/RC1) on C2-
0377   0099          01227           movwf     CMCON0              ;turn on comp. with Iscale (AN5) on C2- and C2+ on VRef
0378   1683          01228           bsf       STATUS,RP0          ;select register bank 1
0379   30A3          01229           movlw     0xA3                ;Voltage reference for 3/24*Vdd = 0,625 V
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

037A   0099           01230           movwf     VRCON               ;enable Voltage reference to detect scale 0 (0V)
037B   1283           01231           bcf       STATUS,RP0          ;reselect register bank 0
037C   2334           01232           call      wait_10             ;delay to allow Comparator and Vref to settle
037D   1F99           01233           btfss     CMCON0,C2OUT        ;if Comparator 2 Output= 0 (Vscale <0.625 V), then:
037E   2B8E           01234            goto     Isclok              ;leave scale status=0
037F   1683           01235           bsf       STATUS,RP0          ;select register bank 1
0380   30A9           01236           movlw     0xA9                ;Voltage reference on for 9/24Vdd = 1,875 V
0381   0099           01237           movwf     VRCON               ;set Voltage reference to detect scale 1
0382   1283           01238           bcf       STATUS,RP0          ;reselect register bank 0
0383   2334           01239           call      wait_10             ;set delay to allow Vref to settle
0384   1F99           01240           btfss     CMCON0,C2OUT        ;if Comparator 2 output=0 (Vscale < 1.875 V), then:
0385   2B8D           01241            goto     Iscl13              ;continue set scale 1
0386   17D7           01242 Isclhi    bsf       sw_stat,Iscale2     ;set scale= 2
0387   1683           01243           bsf       STATUS,RP0          ;select register bank 1
0388   30AF           01244           movlw     0xAF                ;Voltage reference on for 15/24Vdd = 3.125 V
0389   0099           01245           movwf     VRCON               ;set Voltage reference to detect scale 2 or 3
038A   1283           01246           bcf       STATUS,RP0          ;reselect register bank 0
038B   2334           01247           call      wait_10             ;set delay to allow Vref to settle
038C   1B99           01248           btfsc     CMCON0,C2OUT        ;if C2OUT=0 (Vscale > 3.125 V), then:
038D   1757           01249 Iscl13     bsf      sw_stat,Iscale1     ;set scale= 1 or 3
038E   1683           01250 Isclok    bsf       STATUS,RP0          ;select register bank 1
038F   0199           01251           clrf      VRCON               ;disable Voltage reference
0390   1283           01252           bcf       STATUS,RP0          ;reselect register bank 0
0391   0199           01253           clrf      CMCON0              ;disable Comparators
0392   3050           01254           movlw     Isc0cal             ;get eeprom address for scale 0 calibration
0393   1B57           01255           btfsc     sw_stat,Iscale1     ;if scale1=1, then:
0394   3E01           01256            addlw    1                   ;increment eeprom address to scale 1 (assume seq.)
0395   1BD7           01257           btfsc     sw_stat,Iscale2     ;if scale2=1, then:
0396   3E02           01258            addlw    2                   ;increment eeprom address to scale 2 or 3 (assume seq.)
0397   22DB           01259 Isclcal   call      reeprom             ;get current adjustment value for selected scale
0398   00D0           01260           movwf     Divisor             ;save it
0399   1D57           01261           btfss     sw_stat,adjust      ;if not in adjust mode, then:
039A   2BB0           01262            goto     skpIadj             ;skip adjustment
                      01263 ;adjust calibration value
039B   1DD7           01264           btfss     sw_stat,adj_dec     ;if inc request, then:
039C   2BA2           01265            goto     decIadj
039D   3A7F           01266           xorlw     0x7F                ;check for max postive
039E   1903           01267           skpnz                         ;if max, then:
039F   2BAE           01268            goto     endIadj             ;quit adj
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

03A0   0AD0           01269           incf      Divisor,F ;increment calibration value
03A1   2BA6           01270           goto      savIadj
                      01271
03A2   3A80           01272 decIadj   xorlw     0x80                ;check for min negative
03A3   1903           01273           skpnz                         ;if min, then:
03A4   2BAE           01274            goto     endIadj             ;quit adj
03A5   03D0           01275           decf      Divisor,F
03A6   3050           01276 savIadj   movlw     Divisor             ;get pointer of new scale calibration data
03A7   0084           01277           movwf     FSR                 ;set pointer to data to write
03A8   3050           01278           movlw     Isc0cal             ;assume scale 0
03A9   1B57           01279           btfsc     sw_stat,Iscale1     ;if scale1=1, then:
03AA   3E01           01280            addlw    1                   ;increment eeprom address to scale 1 (assume seq.)
03AB   1BD7           01281           btfsc     sw_stat,Iscale2     ;if scale2=1, then:
03AC   3E02           01282            addlw    2                   ;increment eeprom address to scale 2 or 3 (assume seq.)
03AD   22E4           01283           call      weeprom             ;write new adjustment value for selected scale
03AE   1157           01284 endIadj   bcf       sw_stat,adjust      ;acknowledge adjust
03AF   15D7           01285           bsf       sw_stat,adj_dec     ;also set adjust direction bit, to alow normal display
                      01286 ;Apply calibration adjustment and Divide by 2560 ±5%
03B0   300A           01287 skpIadj   movlw     high 2560 ;get Divisor high byte
03B1   00D1           01288           movwf     Divisor+1
03B2   1BD0           01289           btfsc     Divisor,7 ;if fine adjustment value is negative, then:
03B3   03D1           01290            decf     Divisor+1,F         ;propagate sign
03B4   2A38           01291           goto      dv32_16             ;continue with divide routine
                      01292 ;_____________________________________________________________________________
                      01293 ;Set up special function registers
03B5   1683           01294 init      bsf       STATUS,RP0          ;select register page 1
03B6   160F           01295           bsf       OSCCON,IRCF0        ;set internal osc to 8 MHz
03B7   30D2           01296           movlw     0xD2                ;enable Timer 0 to run from internal clock at rate 1:8
03B8   1281           01297           bcf       OPTION_REG,5        ;set Timer 0 options
03B9   303F           01298           movlw     0x3F                ;all A ports inputs
03BA   0085           01299           movwf     TRISA               ;set all of Port A input
03BB   302F           01300           movlw     0x2F                ;Port RC4 output, others input
03BC   0087           01301           movwf     TRISC               ;set Port C input/output
03BD   0199           01302           clrf      VRCON               ;disable Voltage reference
03BE   3050           01303           movlw     0x50                ;TAD= Fosc/16
03BF   009F           01304           movwf     ADCON1              ;config A/D clock
03C0   3037           01305           movlw     0x37                ;enable AN0, AN1, AN2, AN4 and AN5
03C1   0091           01306           movwf     ANSEL               ;configure ports for analog use
03C2   3063           01307           movlw     99
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

03C3   0092           01308           movwf     PR2                 ;set Timer 2 period time to 200 µs
03C4   168C           01309           bsf       PIE1,CCP1IE         ;enable CCP interrupts
03C5   140C           01310           bsf       PIE1,TMR1IE         ;enable Timer 1 interrupt
03C6   148C           01311           bsf       PIE1,TMR2IE         ;enable Timer 2 interrupt
03C7   170C           01312           bsf       PIE1,ADIE ;enable A/D conversion interrupt
03C8   1283           01313           bcf       STATUS,RP0          ;reselect register page 0
03C9   3080           01314           movlw     0x80                ;right justified, Voltage ref=Vdd, AN0, converter Off
03CA   009F           01315           movwf     ADCON0              ;initialize ADCON0
03CB   3030           01316           movlw     0x30                ;prescale 1:8, use internal clock
03CC   0090           01317           movwf     T1CON               ;set Timer 1, timer off
03CD   0192           01318           clrf      T2CON               ;set Timer 2 prescaler 1:1, postscaler 1:1, Timer off
03CE   1412           01319           bsf       T2CON,0             ;set Timer 2 prescale to 1:4
03CF   018C           01320           clrf      PIR1                ;clear any periferal interrupt
03D0   170B           01321           bsf       INTCON,PEIE         ;enable peripheral interrupts
03D1   178B           01322           bsf       INTCON,GIE          ;enable global interrupts
03D2   3001           01323           movlw     HIGH bcd7seg        ;get high address of lookup tables
03D3   008A           01324           movwf     PCLATH              ;set PC counter high address latch
03D4   01D5           01325           clrf      Cycles              ;clear
03D5   3009           01326           movlw     Vrms                ;default display Vrms
03D6   00D7           01327           movwf     sw_stat             ;preselect Vrms display
                      01328 ;Wait for frequency measurement finished, meanwhile check button request
03D7   2325           01329 F_wait    call      read_sw
03D8   1810           01330           btfsc     T1CON,TMR1ON        ;if Timer 1 is running, then:
03D9   2BD7           01331           goto      F_wait              ;wait for it to finish
03DA   0878           01332           movf      LastCCP,W
03DB   00CE           01333           movwf     CCPtime             ;copy result
03DC   0879           01334           movf      LastCCP+1,W
03DD   00CF           01335           movwf     CCPtime+1
03DE   0855           01336           movf      Cycles,W
03DF   00CD           01337           movwf     CCP_cnt
                      01338 ;Initialize and Start A/D conversion for Ref and Vsrc
03E0   141F           01339           bsf       ADCON0,ADON         ;turn on A/D
03E1   01B8           01340           clrf      SumRef              ;clear sum reference voltage registers
03E2   01B9           01341           clrf      SumRef+1
03E3   01BA           01342           clrf      SumRef+2
03E4   01BB           01343           clrf      SumVsrc             ;clear sum source voltage registers
03E5   01BC           01344           clrf      SumVsrc+1
03E6   01BD           01345           clrf      SumVsrc+2
03E7   01BE           01346           clrf      SumIsrc
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

03E8   01BF           01347           clrf      SumIsrc+1
03E9   01C0           01348           clrf      SumIsrc+2
03EA   01C1           01349           clrf      SumVrec             ;clear sum source rectified voltage
03EB   01C2           01350           clrf      SumVrec+1
03EC   01C3           01351           clrf      SumVrec+2
03ED   01C4           01352           clrf      SumIrec             ;clear sum source rectified current
03EE   01C5           01353           clrf      SumIrec+1
03EF   01C6           01354           clrf      SumIrec+2
03F0   01A0           01355           clrf      SumVsqr             ;clear sum source squared registers
03F1   01A1           01356           clrf      SumVsqr+1
03F2   01A2           01357           clrf      SumVsqr+2
03F3   01A3           01358           clrf      SumVsqr+3
03F4   01A4           01359           clrf      SumIsqr
03F5   01A5           01360           clrf      SumIsqr+1
03F6   01A6           01361           clrf      SumIsqr+2
03F7   01A7           01362           clrf      SumIsqr+3
03F8   01A8           01363           clrf      SumWatt
03F9   01A9           01364           clrf      SumWatt+1
03FA   01AA           01365           clrf      SumWatt+2
03FB   01AB           01366           clrf      SumWatt+3
03FC   01DA           01367           clrf      OV_flag
03FD   3064           01368           movlw     100
03FE   00D6           01369           movwf     Samples             ;set sample count
03FF   3019           01370           movlw     25
0400   00D5           01371           movwf     Cycles              ;set cycle count
0401   0191           01372           clrf      TMR2
0402   130C           01373           bcf       PIR1,ADIF ;clear any spurious AD interrupt
0403   108C           01374           bcf       PIR1,TMR2IF         ;clear any spurious Timer 2 interrupt
0404   1512           01375           bsf       T2CON,TMR2ON        ;start Timer 2
                      01376 ;Wait for Timer 2 to stop, meanwhile check switch change
0405   2325           01377 V_wait    call      read_sw
0406   1912           01378           btfsc     T2CON,TMR2ON        ;if Timer 2 is still running, then:
0407   2C05           01379            goto     V_wait              ;loop on switches until Timer 2 stops
0408   101F           01380           bcf       ADCON0,ADON         ;turn off A/D
                      01381 ;initialize for frequency measurement and start it
0409   01F4           01382           clrf      FrstCCP             ;clear First time capture
040A   01F5           01383           clrf      FrstCCP+1
040B   01F8           01384           clrf      LastCCP
040C   01F9           01385           clrf      LastCCP+1
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

040D   01D5           01386           clrf      Cycles              ;initialize Cycle counter
040E   018E           01387           clrf      TMR1L               ;clear Timer 1 register low byte
040F   018F           01388           clrf      TMR1H               ;clear Timer 1 register high byte
0410   128C           01389           bcf       PIR1,CCP1IF         ;clear spurious CCP interrupts
0411   1410           01390           bsf       T1CON,TMR1ON        ;start Timer 1 to run for 16*32768 µs
0412   3005           01391           movlw     0x05                ;set capture mode, every rising edge
0413   0095           01392           movwf     CCP1CON             ;set CCP mode and pin configuration
                      01393 ;get source and convert to scale with 3 decimal digits; Sum/2500*5/1024*200= Sum/2560
0414   085A           01394           movf      OV_flag,W ;check for over-voltage
0415   1D03           01395           skpz                          ;if either condition
0416   2D68           01396            goto     Disp_ov             ;handle condition
0417   0857           01397           movf      sw_stat,W
0418   390F           01398           andlw     0x0F                ;isolate function request buttons
0419   3A00           01399           xorlw     P                   ;check for P= true power
041A   1903           01400           skpnz                         ;if P requested, then:
041B   2CF9           01401            goto     calc_P
041C   3A01           01402           xorlw     S^P                 ;check for S= apparent power
041D   1903           01403           skpnz               ;        if S requested, then:
041E   2D17           01404            goto     calc_S
041F   3A03           01405           xorlw     Freq^S              ;check for F= frequency
0420   1903           01406           skpnz
0421   2D89           01407            goto     calc_F              ;do Frequency calculations
0422   3A01           01408           xorlw     Kf^Freq             ;check for Kf= form factor
0423   1903           01409           skpnz
0424   2C70           01410            goto     cal_Kf              ;do Kf calculations
0425   0857           01411           movf      sw_stat,W
0426   3903           01412           andlw     0x03                ;isolate function request buttons, ignore adjust bits
0427   3A00           01413           xorlw     Va&3h               ;check Va= DC averige voltage
0428   1903           01414           skpnz                         ;if Va, then:
0429   2C41           01415            goto     calc_Va             ;calculate averige voltage
042A   3A01           01416           xorlw     (Vrms^Va)&3h        ;check Vrms= DC/AC effective voltage
042B   1903           01417           skpnz
042C   2C62           01418            goto     calVrms             ;calculate effective votage
042D   3A03           01419           xorlw     (Ia^Vrms)&3h        ;check for Ia= DC averige current
042E   1903           01420           skpnz
042F   2CC2           01421            goto     calc_Ia             ;calculate averige current
0430   3A01           01422           xorlw     (Irms^Ia)&3h        ;check for Irms= DC/AC effective current
0431   1903           01423           skpnz
0432   2CE0           01424            goto     calIrms             ;calculate effective current
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     01425 ; Ref Calculate SumRef/2500*5/1024*100= SumRef/5120 (never executed)
0433   0838          01426 cal_ref   movf      SumRef,W
0434   00B4          01427           movwf     Dividen
0435   0839          01428           movf      SumRef+1,W
0436   00B5          01429           movwf     Dividen+1
0437   083A          01430           movf      SumRef+2,W
0438   00B6          01431           movwf     Dividen+2
0439   01D0          01432           clrf      Divisor             ;set Divisor low byte
043A   3014          01433           movlw     high 5120 ;get Divisor high byte
043B   00D1          01434           movwf     Divisor+1
043C   2265          01435           call      dv24_16
043D   01DA          01436           clrf      Sign                ;Ref is always positive
043E   01D9          01437           clrf      Scale
043F   14D9          01438           bsf       Scale,Scale2        ;Ref always has 2 decimals
0440   2D35          01439           goto      Display
                     01440
                     01441 ; Calculate (SumVsrc-SumRef)/2560
0441   1058          01442 calc_Va   bcf       prefun,KfI          ;set indicator for Kf of Voltage
0442   0838          01443           movf      SumRef,W  ;get 2560*mean of Ref low byte
0443   023B          01444           subwf     SumVsrc,W ;calculate offset from Ref low byte
0444   00B4          01445           movwf     Dividen             ;save as relative value low byte
0445   0839          01446           movf      SumRef+1,W          ;get 2560*mean of Ref middle byte
0446   1C03          01447           skpc                          ;if borrow, then:
0447   0F39          01448            incfsz   SumRef+1,W          ;increment for borrow
0448   023C          01449            subwf    SumVsrc+1,W         ;calculate offset from Ref middle byte
0449   00B5          01450           movwf     Dividen+1 ;save as relative value high byte
044A   083A          01451           movf      SumRef+2,W          ;get 2560*mean of Ref high byte
044B   1C03          01452           skpc                          ;if borrow, then:
044C   0A3A          01453            incf     SumRef+2,W          ;take care of borrow
044D   023D          01454           subwf     SumVsrc+2,W         ;result will reflect sign
044E   00B6          01455           movwf     Dividen+2 ;save for display
044F   00DA          01456           movwf     Sign
0450   1FDA          01457           btfss     Sign,7              ;if result is positive, then:
0451   2C5A          01458            goto     skpVcom             ;skip complement result
0452   09B4          01459           comf      Dividen,F ;make value positive for display (9BFF » 6400, FFFF » 0000)
0453   09B5          01460           comf      Dividen+1,F
0454   09B6          01461           comf      Dividen+2,F         ;assumes 2.5V is halfway between 1FFh and 200h
0455   0AB4          01462           incf      Dividen,F
0456   1903          01463           skpnz                         ;if low byte changed to 0, then:
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0457   0AB5           01464            incf     Dividen+1,F         ;add carry to middle byte
0458   1903           01465           skpnz
0459   0AB6           01466            incf     Dividen+2,F         ;and into high byte
045A   2339           01467 skpVcom   call     Vsclget             ;get Vscale and adjust if required, result in Dividen
045B   3002           01468           movlw    1<<Scale2 ;assume Vscale0= Display scale 2 (two decimal places)
045C   1A57           01469           btfsc    sw_stat,Vscale1     ;if Vscale1 is set, then:
045D   3001           01470            movlw   1<<Scale1 ;set scale 1 (one decimal place)
045E   1AD7           01471           btfsc    sw_stat,Vscale2     ;if Vscale2 is set, then:
045F   0103           01472            clrw                        ;set scale 0 (no decimal places)
0460   00D9           01473           movwf    Scale               ;set required scale
0461   2D35           01474           goto     Display             ;display result in Dividen as Va
                      01475
                      01476 ;we also have sum of 50 RMS voltage samples as digital values (C800h max.)
0462   1058           01477 calVrms   bcf      prefun,KfI          ;set indicator for Kf of Voltage
                      01478           Sqrt_32  SumVsqr             ;(2500 * Vsqr^2)^0.5 = 50*RMS in Product:2 (C800 max)
0463   0820               M           movf     SumVsqr,W
0464   00AC               M           movwf    Square
0465   0821               M           movf     SumVsqr+1,W
0466   00AD               M           movwf    Square+1
0467   0822               M           movf     SumVsqr+2,W
0468   00AE               M           movwf    Square+2
0469   0823               M           movf     SumVsqr+3,W
046A   00AF               M           movwf    Square+3
046B   21C9               M           call     Sqroot
046C   01DA           01479           clrf     Sign                ;Square is always positive
                      01480 ;adapt Vrms to scale = (50*Vrms) /50*5/1024*200= (50*RMS)*50/2560
046D   3032           01481           movlw    50
046E   222C           01482           call     Mul16x8             ;multiply root with 50, result in product:3 (=Dividen)
046F   2C5A           01483           goto     skpVcom
                      01484
                      01485 ;we also have sum of 2500 rectified voltage samples relative to Ref in SumVrec
                      01486 ;get value and convert to scale; SumVrec/2560
0470   1858           01487 cal_Kf    btfsc    prefun,KfI
0471   2C9A           01488            goto    cal_KfI
0472   0841           01489           movf     SumVrec,W
0473   00B4           01490           movwf    Dividen
0474   0842           01491           movf     SumVrec+1,W
0475   00B5           01492           movwf    Dividen+1
0476   0843           01493           movf     SumVrec+2,W
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0477   00B6           01494           movwf     Dividen+2
                      01495 ;now we can also calculate the form factor= Vrms/Var.
                      01496 ;Dividen= 2500*Var as digitized value. Root= 50*Vrms as digitized value.
                      01497 ; Kf= 100*50*Root/Dividen = (8*Root)/(Dividen/625)
                      01498 ;divide SumVrec (in Dividen) by 625 and place result to Divisor (max:1000h)
0478   3071           01499           movlw     low 625
0479   00D0           01500           movwf     Divisor
047A   3002           01501           movlw     high 625
047B   00D1           01502           movwf     Divisor+1
047C   2265           01503           call      dv24_16
047D   0834           01504           movf      Dividen,W
047E   00D0           01505           movwf     Divisor
047F   0835           01506           movf      Dividen+1,W
0480   00D1           01507           movwf     Divisor+1
                      01508 ;multiply Root by 8 and place result in Dividend
                      01509           Sqrt_32   SumVsqr             ;(2500 * Vsqr^2)^0.5 = 50*RMS in Dividen (C800 max)
0481   0820               M           movf      SumVsqr,W
0482   00AC               M           movwf     Square
0483   0821               M           movf      SumVsqr+1,W
0484   00AD               M           movwf     Square+1
0485   0822               M           movf      SumVsqr+2,W
0486   00AE               M           movwf     Square+2
0487   0823               M           movf      SumVsqr+3,W
0488   00AF               M           movwf     Square+3
0489   21C9               M           call      Sqroot
048A   01B6           01510           clrf      Dividen+2
048B   1003           01511           clrc
048C   0DB4           01512           rlf       Dividen,F           ;=2*Root
048D   0DB5           01513           rlf       Dividen+1,F
048E   0DB6           01514           rlf       Dividen+2,F         ;shift in Carry, leave with C=0
048F   0DB4           01515           rlf       Dividen,F ;=4*Root
0490   0DB5           01516           rlf       Dividen+1,F
0491   0DB6           01517           rlf       Dividen+2,F         ; leave with C=0
0492   0DB4           01518           rlf       Dividen,F ;=8*Root
0493   0DB5           01519           rlf       Dividen+1,F
0494   0DB6           01520           rlf       Dividen+2,F
                      01521 ;divide (Root*8) by (SumVrec/625)
0495   2265           01522           call      dv24_16             ;=100*Kf in Quotient expecting byte result (=Product) C=0
0496   01DA           01523           clrf      Sign                ;value is always positive
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0497   01D9           01524           clrf      Scale
0498   14D9           01525           bsf       Scale,Scale2        ;Kf always has 2 decimals
0499   2D35           01526           goto      Display
                      01527
                      01528 ;calculate Kf of current
049A   0844           01529 cal_KfI   movf      SumIrec,W
049B   00B4           01530           movwf     Dividen
049C   0845           01531           movf      SumIrec+1,W
049D   00B5           01532           movwf     Dividen+1
049E   0846           01533           movf      SumIrec+2,W
049F   00B6           01534           movwf     Dividen+2
                      01535 ;divide SumIrec (in Dividen) by 625 and place result to Divisor (max:1000h)
04A0   3071           01536           movlw     low 625
04A1   00D0           01537           movwf     Divisor
04A2   3002           01538           movlw     high 625
04A3   00D1           01539           movwf     Divisor+1
04A4   2265           01540           call      dv24_16
04A5   0834           01541           movf      Dividen,W
04A6   00D0           01542           movwf     Divisor
04A7   0835           01543           movf      Dividen+1,W
04A8   00D1           01544           movwf     Divisor+1
                      01545 ;multiply Root by 8 and place result in Dividend
                      01546           Sqrt_32   SumIsqr             ;(2500 * Isqr^2)^0.5 = 50*RMS in Dividen (C800 max)
04A9   0824               M           movf      SumIsqr,W
04AA   00AC               M           movwf     Square
04AB   0825               M           movf      SumIsqr+1,W
04AC   00AD               M           movwf     Square+1
04AD   0826               M           movf      SumIsqr+2,W
04AE   00AE               M           movwf     Square+2
04AF   0827               M           movf      SumIsqr+3,W
04B0   00AF               M           movwf     Square+3
04B1   21C9               M           call      Sqroot
04B2   01B6           01547           clrf      Dividen+2
04B3   1003           01548           clrc
04B4   0DB4           01549           rlf       Dividen,F           ;=2*Root
04B5   0DB5           01550           rlf       Dividen+1,F
04B6   0DB6           01551           rlf       Dividen+2,F         ;shift in Carry, leave with C=0
04B7   0DB4           01552           rlf       Dividen,F ;=4*Root
04B8   0DB5           01553           rlf       Dividen+1,F
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

04B9   0DB6           01554           rlf       Dividen+2,F         ; leave with C=0
04BA   0DB4           01555           rlf       Dividen,F ;=8*Root
04BB   0DB5           01556           rlf       Dividen+1,F
04BC   0DB6           01557           rlf       Dividen+2,F
                      01558 ;divide (Root*8) by (SumIrec/625)
04BD   2265           01559           call      dv24_16             ;=100*Kf in Quotient expecting byte result (=Product) C=0
04BE   01DA           01560           clrf      Sign                ;value is always positive
04BF   01D9           01561           clrf      Scale
04C0   14D9           01562           bsf       Scale,Scale2        ;Kf always has 2 decimals
04C1   2D35           01563           goto      Display
                      01564
                      01565 ; Calculate (SumIsrc-SumRef)*2/2560
04C2   1458           01566 calc_Ia   bsf       prefun,KfI          ;set indicator for Kf of Corrent
04C3   0838           01567           movf      SumRef,W  ;get 2560*mean of Ref low byte
04C4   023E           01568           subwf     SumIsrc,W ;calculate offset from Ref low byte
04C5   00B4           01569           movwf     Dividen             ;save as relative value low byte
04C6   0839           01570           movf      SumRef+1,W          ;get 2560*mean of Ref middle byte
04C7   1C03           01571           skpc                          ;if borrow, then:
04C8   0F39           01572            incfsz   SumRef+1,W          ;increment for borrow
04C9   023F           01573            subwf    SumIsrc+1,W         ;calculate offset from Ref middle byte
04CA   00B5           01574           movwf     Dividen+1 ;save as relative value high byte
04CB   083A           01575           movf      SumRef+2,W          ;get 2560*mean of Ref high byte
04CC   1C03           01576           skpc                          ;if borrow, then:
04CD   0A3A           01577            incf     SumRef+2,W          ;take care of borrow
04CE   0240           01578           subwf     SumIsrc+2,W         ;result will reflect sign
04CF   00B6           01579           movwf     Dividen+2 ;save for display
04D0   00DA           01580           movwf     Sign
04D1   1FDA           01581           btfss     Sign,7              ;if result is positive, then:
04D2   2CDB           01582            goto     skpIcom             ;skip complement result
04D3   09B4           01583           comf      Dividen,F ;make value positive for display (9BFF » 6400, FFFF » 0000)
04D4   09B5           01584           comf      Dividen+1,F
04D5   09B6           01585           comf      Dividen+2,F         ;assumes 2.5V is halfway between at 200h
04D6   0AB4           01586           incf      Dividen,F
04D7   1903           01587           skpnz                         ;if low byte changed to 0, then:
04D8   0AB5           01588            incf     Dividen+1,F         ;add carry, to middle byte
04D9   1903           01589           skpnz
04DA   0AB6           01590            incf     Dividen+2,F         ;and into high byte
04DB   1003           01591 skpIcom   clrc
04DC   0DB4           01592           rlf       Dividen,F
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

04DD   0DB5           01593           rlf       Dividen+1,F
04DE   0DB6           01594           rlf       Dividen+2,F
04DF   2CED           01595           goto      setIscl
                      01596
                      01597 ; Calculate SumIsqr^0.5/50
04E0   1458           01598 calIrms   bsf       prefun,KfI          ;set indicator for Kf of Corrent
                      01599           Sqrt_32   SumIsqr             ;(2500 * Vsqr^2)^0.5 = 50*RMS in Product:2 (C800 max)
04E1   0824               M           movf      SumIsqr,W
04E2   00AC               M           movwf     Square
04E3   0825               M           movf      SumIsqr+1,W
04E4   00AD               M           movwf     Square+1
04E5   0826               M           movf      SumIsqr+2,W
04E6   00AE               M           movwf     Square+2
04E7   0827               M           movf      SumIsqr+3,W
04E8   00AF               M           movwf     Square+3
04E9   21C9               M           call      Sqroot
04EA   01DA           01600           clrf      Sign                ;Square is always positive
                      01601 ;adapt Irms to scale = (50*RMS) /50*5/1024*400 = (50*RMS)*100/2560
04EB   3064           01602           movlw     100
04EC   222C           01603           call      Mul16x8             ;multiply root by 100, result in product (=Dividen)
04ED   01B7           01604 setIscl   clrf      Dividen+3
04EE   2374           01605           call      Isclget
04EF   3011           01606           movlw     1<<Scale1|1<<milli  ;assume Iscale0 =Display scale 0 (±99.9 mA)
04F0   1B57           01607           btfsc     sw_stat,Iscale1     ;if Iscale1 is set, then:
04F1   3010           01608            movlw    1<<milli  ;set for Scale 1 (±999 mA)
04F2   1FD7           01609           btfss     sw_stat,Iscale2     ;if Iscale2 is not set, then:
04F3   2CF7           01610            goto     use_scl
04F4   3002           01611           movlw     1<<Scale2 ;set for Scale 2 (±9.99 A)
04F5   1B57           01612           btfsc     sw_stat,Iscale1     ;if Iscale1 is also set, then:
04F6   3001           01613            movlw    1<<Scale1 ;set for Scale 1 (±99.9 A)
04F7   00D9           01614 use_scl   movwf     Scale
04F8   2D35           01615           goto      Display             ;display result in Dividen as Ia
                      01616
                      01617 ;calculate real power (W*2500/16)=SumWatt:4 signed; SumWatt*5/2560^2= mW display
04F9   0828           01618 calc_P    movf      SumWatt,W
04FA   00B4           01619           movwf     Dividen
04FB   0829           01620           movf      SumWatt+1,W
04FC   00B5           01621           movwf     Dividen+1
04FD   082A           01622           movf      SumWatt+2,W
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

04FE   00B6           01623           movwf     Dividen+2
04FF   082B           01624           movf      SumWatt+3,W
0500   00B7           01625           movwf     Dividen+3
0501   01DA           01626           clrf      Sign                ;assume value is positive
0502   1FAB           01627           btfss     SumWatt+3,7         ;if value is positive
0503   2D10           01628            goto     Pabs                ;skip conversion to absolute
0504   09DA           01629           comf      Sign,F              ;make Sign negative
0505   09B4           01630           comf      Dividen,F
0506   09B5           01631           comf      Dividen+1,F
0507   09B6           01632           comf      Dividen+2,F
0508   09B7           01633           comf      Dividen+3,F
0509   0AB4           01634           incf      Dividen,F
050A   1903           01635           skpnz
050B   0AB5           01636            incf     Dividen+1,F
050C   1903           01637           skpnz
050D   0AB6           01638            incf     Dividen+2,F
050E   1903           01639           skpnz
050F   0AB7           01640            incf     Dividen+3,F
0510   2374           01641 Pabs      call      Isclget             ;input 32 bits, result 24 bits
0511   3005           01642           movlw     5
0512   220A           01643           call      Mul24x8             ;32 bit product, 24 bits relevant
0513   2339           01644           call      Vsclget
0514   21B6           01645           call      WsclLed             ;get Scale settings for milli, kilo and dp leds
0515   00D9           01646           movwf     Scale               ;set in Scale register
0516   2D35           01647           goto      Display
                      01648
                      01649 ;calculate apparent power (50*Vrms)/2560 * (50*Irms)*5/2560
                      01650 calc_S    Sqrt_32   SumVsqr             ;(2500 * Vrec^2)^0.5 = 50*Vrms in Product:2 (6400h max)
0517   0820               M           movf      SumVsqr,W
0518   00AC               M           movwf     Square
0519   0821               M           movf      SumVsqr+1,W
051A   00AD               M           movwf     Square+1
051B   0822               M           movf      SumVsqr+2,W
051C   00AE               M           movwf     Square+2
051D   0823               M           movf      SumVsqr+3,W
051E   00AF               M           movwf     Square+3
051F   21C9               M           call      Sqroot
0520   0834           01651           movf      Product,W ;get result
0521   00D0           01652           movwf     Divisor             ;and move to Divisor
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0522   0835           01653           movf      Product+1,W         ;and its high byte
0523   00D1           01654           movwf     Divisor+1
                      01655           Sqrt_32   SumIsqr             ;(2500 * Irec^2)^0.5 = 50*Irms in Dividen:2 (6400h max)
0524   0824               M           movf      SumIsqr,W
0525   00AC               M           movwf     Square
0526   0825               M           movf      SumIsqr+1,W
0527   00AD               M           movwf     Square+1
0528   0826               M           movf      SumIsqr+2,W
0529   00AE               M           movwf     Square+2
052A   0827               M           movf      SumIsqr+3,W
052B   00AF               M           movwf     Square+3
052C   21C9               M           call      Sqroot
052D   2218           01656           call      Mul_16              ;Multiply (50*Irms) with Divisor, result in Product:4
052E   2374           01657           call      Isclget             ;get Iscale and apply Iscaling, result in Dividen
052F   3005           01658           movlw     5                   ;apply hardware scale factor
0530   220A           01659           call      Mul24x8             ;multiply with W, result in Product:3 (138800h max)
0531   2339           01660           call      Vsclget             ;get Vscale and apply Vscaling, result in Dividen
0532   21B6           01661           call      WsclLed             ;get Scale settings for milli, kilo leds and dp
0533   00D9           01662           movwf     Scale               ;set in Scale register
0534   01DA           01663           clrf      Sign                ;result is always positive
                      01664 ;Convert to BCD and Display result (takes ±1200 µs)
0535   22A9           01665 Display   call      b16_bcd   ;100µs    ;convert NumH:NumL to BCD Ones, Tens and Huns
0536   3040           01666           movlw     40h                 ;Write data to display register auto increment
0537   00FC           01667           movwf     TM_dat
0538   22F7           01668           call      TM_cmd    ;161µs    ;send data command 1
0539   231C           01669           call      TM_stp    ; 14µs    ;terminate transmission
053A   30C0           01670           movlw     0C0h                ;set data address command 2
053B   00FC           01671           movwf     TM_dat
053C   22F7           01672           call      TM_cmd    ;161µs    ;send address command 2
053D   0103           01673           clrw                          ;blank sign
053E   1BDA           01674           btfsc     Sign,7              ;if value is negative, then:
053F   3040           01675            movlw    0x40                ;load 7-segment value for '-'
0540   00FC           01676           movwf     TM_dat
0541   22FB           01677           call      TM_wrt    ;148µs    ;append data byte transmission Digit 1
0542   085D           01678           movf      Huns,W              ;get second BCD digit in low W
0543   1903           01679           skpnz                         ;if 0, then:
0544   18D9           01680            btfsc    Scale,Scale2        ;if not 0 or scale 2, then:
0545   218E           01681            call     bcd7seg   ;  9µs    ;convert to segment value
0546   00DD           01682           movwf     Huns                ;flag for no leading blanks
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0547   18D9           01683           btfsc     Scale,Scale2        ;if scale 2, then:
0548   3880           01684            iorlw    0x80                ;include decimal point
0549   00FC           01685           movwf     TM_dat
054A   22FB           01686           call      TM_wrt    ;148µs    ;append data byte transmission Digit 2
054B   085E           01687           movf      Tens,W              ;get third BCD digit in low W
054C   04DD           01688           iorwf     Huns,F              ;check for blank leading 0
054D   1903           01689           skpnz
054E   1859           01690            btfsc    Scale,Scale1        ;if scale=1, then: force leading 0
054F   218E           01691            call     bcd7seg   ;  9µs    ;convert to segment value
0550   1859           01692           btfsc     Scale,Scale1        ;if scale= 50.0, then:
0551   3880           01693            iorlw    0x80                ;include decimal point
0552   00FC           01694           movwf     TM_dat
0553   22FB           01695           call      TM_wrt    ;148µs    ;append data byte transmission Digit 3
0554   085F           01696           movf      Ones,W              ;get fourth BCD digit in low W
0555   218E           01697           call      bcd7seg   ;  9µs    ;convert to segment value
0556   00FC           01698           movwf     TM_dat
0557   22FB           01699           call      TM_wrt    ;148µs    ;append data byte transmission Digit 4
0558   21A0           01700           call      fun2led   ; 10µs    ;convert to led display
0559   00FC           01701           movwf     TM_dat
055A   22FB           01702           call      TM_wrt    ;148µs    ;append data byte transmission Function LEDs
055B   0103           01703           clrw                          ;assume no modifiers
055C   1A59           01704           btfsc     Scale,milli         ; 10µs    ;get scale modifiers
055D   3002           01705            movlw    1<<Seg_B  ;set SB (milli) on
055E   1AD9           01706           btfsc     Scale,kilo          ; 10µs    ;get scale modifiers
055F   3040           01707            movlw    1<<Seg_G  ;set SG (kilo) on
0560   00FC           01708           movwf     TM_dat
0561   22FB           01709           call      TM_wrt    ;148µs    ;append data byte transmission Scale milli, kilo
0562   231C           01710           call      TM_stp    ; 14µs    ;terminate transmission
0563   3088           01711           movlw     88h                 ;set Display on command with 1/16 brightness
0564   00FC           01712           movwf     TM_dat              ;turn on display with low brightness
0565   22F7           01713           call      TM_cmd    ;161µs    ;send display command
0566   231C           01714           call      TM_stp    ; 14µs    ;terminate tranmission
0567   2BD7           01715           goto      F_wait
                      01716
                      01717 ;Display over-voltage condition
0568   3009           01718 Disp_ov   movlw     Vrms                ;get function for Vrms display
0569   00D7           01719           movwf     sw_stat             ;force display of related function
056A   3040           01720           movlw     40h                 ;Write data to display register auto increment
056B   00FC           01721           movwf     TM_dat
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

056C   22F7           01722           call      TM_cmd    ;161µs    ;send data command 1
056D   231C           01723           call      TM_stp    ; 14µs    ;terminate transmission
056E   30C0           01724           movlw     0C0h                ;set data address command 2
056F   00FC           01725           movwf     TM_dat
0570   22F7           01726           call      TM_cmd    ;161µs    ;send address command 2
0571   3076           01727           movlw     0x76                ;"H"
0572   00FC           01728           movwf     TM_dat
0573   22FB           01729           call      TM_wrt    ;148µs    ;append data byte transmission
0574   3006           01730           movlw     0x06                ;"I"
0575   00FC           01731           movwf     TM_dat
0576   22FB           01732           call      TM_wrt    ;148µs    ;append data byte transmission
0577   303D           01733           movlw     0x3D                ;"G"
0578   00FC           01734           movwf     TM_dat
0579   22FB           01735           call      TM_wrt    ;148µs    ;append data byte transmission
057A   3076           01736           movlw     0x76                ;"H"
057B   00FC           01737           movwf     TM_dat
057C   22FB           01738           call      TM_wrt    ;148µs    ;append data byte transmission
057D   21A0           01739           call      fun2led   ; 10µs    ;convert to led display
057E   00FC           01740           movwf     TM_dat
057F   22FB           01741           call      TM_wrt    ;148µs    ;append data byte transmission
0580   3040           01742           movlw     1<<Seg_G  ;set SG (kilo LED) on
0581   00FC           01743           movwf     TM_dat
0582   22FB           01744           call      TM_wrt    ;148µs    ;append data byte transmission Scale milli, kilo
0583   231C           01745           call      TM_stp    ; 14µs    ;terminate transmission
0584   3088           01746           movlw     88h                 ;set Display on command with 1/16 brightness
0585   00FC           01747           movwf     TM_dat              ;turn on display with low brightness
0586   22F7           01748           call      TM_cmd    ;161µs    ;send display command
0587   231C           01749           call      TM_stp    ; 14µs    ;terminate tranmission
0588   2BD7           01750           goto      F_wait
                      01751
                      01752 ;We have time in CCPtime for count in Cycles.
                      01753 ;Frequency= 250000/(CCPtime/Cycles)
0589   01D9           01754 calc_F    clrf      Scale
                      01755 ;         bsf       Scale,Scale3        ;set 3 decimals
058A   01B4           01756           clrf      NumL                ;clear result in case Cycles=0
058B   01B5           01757           clrf      NumH
058C   084E           01758           movf      CCPtime,W
058D   044F           01759           iorwf     CCPtime+1,W
058E   1903           01760           skpnz                         ;if 0, then:
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

058F   2DEF           01761            goto     Disp_Hz             ;Display 0
                      01762 ;check for < 100 Hz and calculate frequency for range 03.81 to 99.99 Hz
0590   084D           01763           movf      CCP_cnt,W
0591   1903           01764           skpnz                         ;if CCP_cnt reached maximum count, then:
0592   2DBC           01765            goto     chkFscK             ;bypass low frequency calculation
0593   00D0           01766           movwf     Divisor
0594   3EE5           01767           addlw     -27
0595   1803           01768           skpnc                         ;if borrow, then:
0596   2DAC           01769            goto     Fsc1
0597   084E           01770           movf      CCPtime,W ;get
0598   00B4           01771           movwf     Dividen
0599   084F           01772           movf      CCPtime+1,W
059A   00B5           01773           movwf     Dividen+1
059B   228E           01774           call      div16_8
059C   0834           01775           movf      Dividen,W ;get quotient
059D   00D0           01776           movwf     Divisor             ;move to divisor
059E   0835           01777           movf      Dividen+1,W
059F   00D1           01778           movwf     Divisor+1
05A0   3040           01779           movlw     25000000%256
05A1   00B4           01780           movwf     Dividen
05A2   3078           01781           movlw     (25000000/256)%256
05A3   00B5           01782           movwf     Dividen+1
05A4   307D           01783           movlw     (25000000/65536)%256
05A5   00B6           01784           movwf     Dividen+2
05A6   3001           01785           movlw     25000000/16777216
05A7   00B7           01786           movwf     Dividen+3
05A8   2238           01787           call      dv32_16
05A9   01D9           01788           clrf      Scale
05AA   14D9           01789           bsf       Scale,Scale2                  ;set decimal places 2
05AB   2DEF           01790           goto      Disp_Hz
                      01791
                      01792 ;Calculate frequency for range 100.0 to 999.9 Hz
05AC   30A0           01793 Fsc1      movlw     2500000%256
05AD   00B4           01794           movwf     Product
05AE   3025           01795           movlw     (2500000/256)%256
05AF   00B5           01796           movwf     Product+1
05B0   3026           01797           movlw     2500000/65536
05B1   00B6           01798           movwf     Product+2
05B2   084D           01799           movf      CCP_cnt,W
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

05B3   220A           01800           call      Mul24x8
05B4   084E           01801           movf      CCPtime,W
05B5   00D0           01802           movwf     Divisor
05B6   084F           01803           movf      CCPtime+1,W
05B7   00D1           01804           movwf     Divisor+1
05B8   2238           01805           call      dv32_16
05B9   01D9           01806           clrf      Scale
05BA   1459           01807           bsf       Scale,Scale1                  ;set decimal places 1
05BB   2DEF           01808           goto      Disp_Hz
                      01809
                      01810 ;check for < 10 kHz and calculate for range 1.000 to 9.999 kHz
05BC   084E           01811 chkFscK   movf      CCPtime,W ;get time
05BD   00D0           01812           movwf     Divisor             ;move to divisor
05BE   084F           01813           movf      CCPtime+1,W         ;and its high byte
05BF   00D1           01814           movwf     Divisor+1
05C0   3001           01815           movlw     low 6401
05C1   024E           01816           subwf     CCPtime,W
05C2   3019           01817           movlw     high 6401
05C3   1C03           01818           skpc
05C4   301A           01819            movlw    1+high 6401
05C5   024F           01820           subwf     CCPtime+1,W
05C6   1C03           01821           skpc                          ;if time is less then;
05C7   2DD4           01822            goto     Fsc10K
05C8   01B4           01823           clrf      Dividen
05C9   3090           01824           movlw     250000%256
05CA   00B5           01825           movwf     Dividen+1
05CB   30D0           01826           movlw     (250000/256)%256
05CC   00B6           01827           movwf     Dividen+2
05CD   3003           01828           movlw     (250000/65536)%256
05CE   00B7           01829           movwf     Dividen+3
05CF   2238           01830           call      dv32_16
05D0   01D9           01831           clrf      Scale
05D1   1559           01832           bsf       Scale,Scale3
05D2   16D9           01833           bsf       Scale,kilo
05D3   2DEF           01834           goto      Disp_Hz
                      01835
                      01836 ;Check fot < 100 kHz and calculate for range 10.00 to 99.99 kHz
05D4   3081           01837 Fsc10K    movlw     low 641
05D5   024E           01838           subwf     CCPtime,W
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

05D6   3002           01839           movlw     high 641
05D7   1C03           01840           skpc
05D8   3003           01841            movlw    1+high 641
05D9   024F           01842           subwf     CCPtime+1,W
05DA   1C03           01843           skpc                          ;if time is less then;
05DB   2DE6           01844            goto     Fsc100K
05DC   01B4           01845           clrf      Dividen
05DD   30A8           01846           movlw     25000%256
05DE   00B5           01847           movwf     Dividen+1
05DF   3061           01848           movlw     25000/256
05E0   00B6           01849           movwf     Dividen+2
05E1   2265           01850           call      dv24_16
05E2   01D9           01851           clrf      Scale
05E3   14D9           01852           bsf       Scale,Scale2
05E4   16D9           01853           bsf       Scale,kilo
05E5   2DEF           01854           goto      Disp_Hz
                      01855
                      01856 ;Calculate for range 100.0 to 999.9 kHz
05E6   01B4           01857 Fsc100K   clrf      Dividen
05E7   30C4           01858           movlw     2500%256
05E8   00B5           01859           movwf     Dividen+1
05E9   3009           01860           movlw     2500/256
05EA   00B6           01861           movwf     Dividen+2
05EB   2265           01862           call      dv24_16
05EC   01D9           01863           clrf      Scale
05ED   1459           01864           bsf       Scale,Scale1
05EE   16D9           01865           bsf       Scale,kilo
                      01866 ;10*Frequency is now in NumH:NumL (=Dividen)
05EF   22A9           01867 Disp_Hz   call      b16_bcd   ;160µs    ;convert NumH:NumL to BCD Ones, Tens, Huns and OneK
05F0   3040           01868           movlw     40h                 ;Write data to display register auto increment
05F1   00FC           01869           movwf     TM_dat
05F2   22F7           01870           call      TM_cmd    ;161µs    ;send data command 1
05F3   231C           01871           call      TM_stp    ; 14µs    ;terminate transmission
05F4   30C0           01872           movlw     0C0h                ;set data address command 2
05F5   00FC           01873           movwf     TM_dat
05F6   22F7           01874           call      TM_cmd    ;161µs    ;send address command 2
05F7   085C           01875           movf      OneK,W
05F8   1903           01876           skpnz                         ;if digit is 0, then:
05F9   1959           01877            btfsc    Scale,Scale3        ;if not 0 or scale 3, then:
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

05FA   218E           01878           call     bcd7seg
05FB   1959           01879          btfsc     Scale,Scale3
05FC   3880           01880           iorlw    80h
05FD   00FC           01881          movwf     TM_dat
05FE   22FB           01882          call      TM_wrt    ;148µs    ;append data byte transmission
                      01883
05FF   085D           01884          movf      Huns,W              ;get second BCD digit in low W
0600   04DC           01885          iorwf     OneK,F
0601   1903           01886          skpnz
0602   18D9           01887           btfsc    Scale,Scale2
0603   218E           01888           call     bcd7seg   ;  9µs    ;convert to segment value
0604   18D9           01889          btfsc     Scale,Scale2
0605   3880           01890           iorlw    80h
0606   00FC           01891          movwf     TM_dat
0607   22FB           01892          call      TM_wrt    ;148µs    ;append data byte transmission
                      01893
0608   085E           01894          movf      Tens,W              ;get third BCD digit in low W
0609   04DC           01895          iorwf     OneK,F
060A   1903           01896          skpnz
060B   1859           01897           btfsc    Scale,Scale1
060C   218E           01898           call     bcd7seg   ;  9µs    ;convert to segment value
060D   1859           01899          btfsc     Scale,Scale1
060E   3880           01900           iorlw    80h
060F   00FC           01901          movwf     TM_dat
0610   22FB           01902          call      TM_wrt    ;148µs    ;append data byte transmission
0611   085F           01903          movf      Ones,W              ;get fourth BCD digit in low W
0612   218E           01904          call      bcd7seg   ;  9µs    ;convert to segment value
0613   00FC           01905          movwf     TM_dat
0614   22FB           01906          call      TM_wrt    ;148µs    ;append data byte transmission
0615   21A0           01907          call      fun2led             ;get value for mode leds
0616   00FC           01908          movwf     TM_dat
0617   22FB           01909          call      TM_wrt    ;148µs    ;append data byte transmission
0618   0103           01910          clrw
0619   1AD9           01911          btfsc     Scale,kilo
061A   3040           01912           movlw    40h       ;         display k- indicator
061B   00FC           01913          movwf     TM_dat
061C   22FB           01914          call      TM_wrt
061D   231C           01915          call      TM_stp    ; 14µs    ;terminate transmission
061E   3088           01916          movlw     88h                 ;set Display on command with 1/16 brightness
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

061F   00FC           01917          movwf     TM_dat              ;turn on display with low brightness
0620   22F7           01918          call      TM_cmd    ;161µs    ;send display command
0621   231C           01919          call      TM_stp    ; 14µs    ;terminate tranmission
0622   2BD7           01920          goto      F_wait              ;loop on Frequency display, else:
                      01921          end


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

0000 : XXX-XXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0100 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0140 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0180 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
01C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0200 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0240 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0280 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
02C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0300 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0340 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0380 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
03C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0400 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0440 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0480 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
04C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0500 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0540 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0580 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
```

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

```
05C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0600 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXX------------- ----------------
2000 : XXXX---X-------- ---------------- ---------------- ----------------
2100 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
2140 : XXXXXXXXXXXXXXXX XXXX------------ ---------------- ----------------
```

All other memory blocks unused.

Program Memory Words Used: 1570
Program Memory Words Free: 478

Errors   :   0
Warnings :   0 reported,   0 suppressed
Messages :   0 reported,   46 suppressed