```
#rem
BatteryMonitor Source Code

This source code was created by Wayne Getchell of Sagacitic Solutions, amateur radio callsign VE3CZO.  It is meant to be used only with BatteryMonitor hardware.
Do not copy or publish it without permission and citation. If asked, permission is
easily obtained for not for profit use.  Send your request to Wayne at getch@sagacitic.com. If you want to use any part or all of it for profit let's talk.

HARDWARE
BatteryMonitor main board versions 0.4 and 1.0

SOFTWARE REVISION HISTORY
  V0.0a Initial coding
  V0.1a Added Settimer loop for more consistent time & time tick cal routine
  V0.1c Added peak current and minimum voltage capture
  V0.2a Added watt hours
  V0.3a Current scalingfactor + Sf set routine
  V0.3e Added batteries 3cell SLA 3cell LiPo clean up autorange when scaling
  V0.3f Improves transient response via two GainRange changes per cycle &
        Faster response C15=2.2uf from 10. C18 added to insure wake on power-up
        Sw1 & Sw2 human i/f corrections & timing tweaks.
  V0.3g Optimzed transient response positioning 2nd AutoGainRange + bug fixes
  V0.3h Changed current display resolution keeping x.xxA from end of range 1
        to 9.96A.  Note in range 2 readings increment in 40mA steps.
  V1.0a Initial release menu tidyup ILpeak & Vmin speeded up only one meas/cycle
        Improved LoadPower reading accuracy at low currents
  V1.0b Improved AutoGainRange, improve Watthour calculation & display resoultion
  V1.0c Improved LoadPower calculation & display resolution.  Introduce RoundIt subroutine
  V1.0d Improve Ah & Wh routine adding x.xx Ah/Wh.  Increase code space by eliminating some
        subroutines that are only called once. Fixed AutoGainRange bug that caused
        wrong range on restart
  V1.0e Modified Hibernate so user doesn't have to push a button to resume from
        hibernation at startup, happens automatically. Removed NextGainRange variable
        Roll over Ah Wh if over 9,999 and time if over 99hrs.ILraw reassigned fromW8
        which was reused to w9 exclusive
  V1.0f Gain400 changed to Gain160 for 10mA/bit 1to10A range. Needs R7,R8 value change
  V1.0g Added WaitForSwitch timeout; improved clock minute calc accuracy & tidyup

PROGRAM DOWNLOAD PROCEDURE
  There is no reset function on the 20X2 & this unit employs a soft power switch
  To download a program to the 20X2 if the unit is off and connected to a power
  source skip steps 1 and 2
    1) Connect BatteryMonitor to a power source
    2) If the unit turns on then turn it off by momentarily pressing Sw2
    3) Start the download process on the PC
    4) When the 'Connecting to hardware...' window appears push S2,and keep it
       held down until programming is complete
#endrem

Start:
#picaxe 20x2      'directive informs compiler to use 20X2 hardware
'#########################EEPROM SECTION##############################################
'The 20X2 internal EEPROM is used to store battery voltage vs capacity data.
'Up to 8 different battery data sets can be stored in the 256bytes of EEPROM.
```

```
'Each entry uses 32bytes.
'
'BatteryMonitor uses a look-up table to turn Vsource into % remaining battery capacity
'During initialization scratchpad memory locations are populated with the lookup
'table as outlined in the LoadBatteryCap subroutine.
'The % capacity can then be found by coding   get vbat,var1.
'
'The storage format uses a user friendly header of 15 bytes so that the
'battery can can be easily identified by the user. All 15 bytes must be programmed.
'Use ASCII char " " if there is no need to use all characters.
'The next bytes are used to store the battery voltage vs capacity thresholds in
'three digit format where xxx = xx.xV as found through characterization.
'Voltage entries are from low to high and represent 0,2,5,10,20,30,50,60,70,80,90%
'battery capacity.
'If not all battery entries are used the first two bytes of the first unused
'entry must contain $ff,$ff to act as a marker for the menu program.
'20X2 memory limits the battery voltage range to 128bytes, 5.3V to 17.9V
'           2%  5%  10% 20% 30% 40% 50% 60% 70% 80% 90% 99%
'Battery0
EEPROM $00,("3 Cell LeadAcid")
EEPROM $10,(053,054,055,056,057,058,059,060,061,062,063,064)
'Battery1
EEPROM $20,("6 Cell LeadAcid")
EEPROM $30,(109,110,112,114,116,118,120,121,123,124,125,127)
'Battery2
EEPROM $40,("10 Cell NiMH   ")
EEPROM $50,(105,109,113,116,120,123,124,125,126,127,128,129)
'Battery3
EEPROM $60,("11 Cell NiMH   ")
EEPROM $70,(116,120,125,128,133,135,137,138,139,140,141,142)
'Battery4
EEPROM $80,("3 Cell LiPo    ")
EEPROM $90,(095,104,107,110,113,114,116,117,118,120,121,122)
'Battery5
EEPROM $a0,("4 Cell LiPo    ")
EEPROM $b0,(127,138,144,147,150,152,154,156,158,159,161,162)
'Battery6
EEPROM $c0,($ff,$ff)'marker so menu cycles back to first entry
EEPROM $d0,(0)
'Battery7
EEPROM $e0,(0)
EEPROM $f0,(0)
'####End EEPROM Data Load####

'############################## DEFINE VARIABLES ######################
variables:'used by BatteryMonitor
   symbol BVD=c.4        'switched voltage divider on the Source
   symbol Gain160=b.2    'Current to voltage amplifer 160x gain setting
   symbol Gain40=b.3     'Current to voltega amplifier 40x gain setting
   symbol Backlight1=c.5'display backlight low current input=off high=on
   symbol Backlight2=c.7'display backlight high current input=off high=on
   symbol wpEEPROM=b.6   'Write Protect 24AA02 lo=write enable
   symbol shdn=c.0       'regulator shutdown low=shutdown     '
   symbol Sw1=pinc.6     'menu switch 1 to 20X2 leg 4 -display line1
   symbol Sw2=pinb.1     'menu switch 2 to 20X2 leg 10 -display line2

   symbol GainRange=b0   'tracks U1 gain 0=1600, 1=400, 2=40
   symbol var1=b1        'generic variable leased by subroutines
   symbol var2=b2        'generic variable leased by subroutines
   symbol var3=b3        'generic variable leased by subroutines
   symbol unit=b4        'used to parse numbers for display units
   symbol ten=b5         'used to parse numbers for display tens
   symbol hundred=b6     'used to parse numbers for display hundreds
   symbol thousand=b7    'used to parse numbers for display thousands
   symbol tenthousand=b8'used to parse number for display
```

```
121      symbol Backlight=b9  'Backlight level
122      symbol Vsource=b10   'stores battery voltage
123      symbol VsourceMin=b11'captures minimum battery voltage
124      symbol Battery=b12   'stores the battery entry number
125      symbol BattPointer=b13'used to select battery REUSED as SetBattLO
126      symbol sbattlo=b13   'stored battery entry low value
127      symbol sbatthi=b14   'stored battery entry hi value
128      symbol InnerLoopLo=b15'used to write battery level info to scratchpad
129      symbol InnerLoopHi=b16'used to write battery level info to scratchpad
130      symbol HibernateState=b17'Flag to determine if device is returning from
     hibernation
131      symbol ILraw=w9      'store raw load current A-D in sys var coparitor
     results
132      symbol varw10=w10    'Word generic variable leased by subroutines
133      symbol varw11=w11    'Word generic variable leased by subroutines
134      symbol varw12=w12    'Word generic variable leased by subroutines
135      symbol WaitForSwitchTimeout=b26'Counter defines WaitForSwitch timeout
     interval
136      'b27 not used reserved for offset and sign if needed
137      symbol MainMenuItem=b28'keeps track of main menu current location
138      symbol BattCap=b29   'stores % battery capacity
139      symbol AhWh=b30      'Flag slelects either Ah or Wh for display 1=Wh
140      symbol Hours=b31     'keeps track of hours
141      symbol clock=w16     'keeps track of elapsed time in 100ms increments up to
     36000
142      symbol Tick=w17      'used to define length of timing element
143      symbol nAh=w18       'accumulates battery energy use basic unit
144      symbol mAh=w19       'accumulates battery energy use
145      symbol Ah=w20        'accumulates battery energy use
146      symbol uWh=w21       'accumulates battery power basic unit
147      symbol mWh=w22       'accumulates battery power use in mWh
148      symbol Wh=w23        'accumulates battery power use in Watt hours
149      symbol Watts=w24     'stores calculated value for power being used
150      symbol Sf=w25        'Scaling factor used to correct IL reading
151      symbol IL=w26        'stores load current
152      symbol ILmax=w27     'maximum current captured during session
153
154   Initialize:
155      setfreq m4 'optimize power by choosing lowest frequency that will execute
     10k samples/sec
156      'Note at 4MHz the pause statements are 2x command eg. a 5ms command is
     actually 10ms long.
157      high shdn,wpEEPROM,BVD,Backlight2,Gain40 'keep unit powered &
158      'configure outputs, start with lowest GainRange Av=40 (arbitrary decision)
159      pause 100'time for supply to settle before display initializes
160      hi2csetup i2cmaster,$7C,i2cfast_8,i2cbyte  'configure I2C display
161      hi2cout ($00,$39,$14,$70,$5e,$6b,$0c,$06,$01) 'initialize display
162      gosub SplashScreen
163      'write up and down arrow special chars to the display CGRAM
164      hi2cout ($00,$38,$40)'set instruction table to low to write characters
165      hi2cout
     ($40,$04,$0e,$15,$04,$04,$04,$04,$00,$04,$04,$04,$04,$15,$0e,$04,$00)'write
     chars
166      hi2cout ($00,$39) 'set instruction table back to hi
167      adcsetup=%0100000011000000'removes digital input from ADC converters used
168      pokesfr $22,4  'for C.0 fw - issue after adcsetup to fix V+ reference bug
169      '##### Load Configuraiton Data from External EEPROM ########
170      '                                          clock   nAH     mAh     Ah      uWh
           mWh     Wh
171      hi2cin
     [$a0],$00,(HibernateState,Hours,b32,b33,b36,b37,b38,b39,b40,b41,b42,b43,b44,b45,
     b46,b47)
172      '                                  ILmax   Tick     Sf
173      hi2cin [$a0],$10,(AhWh,VsourceMin,b54,b55,b34,b35,b50,b51,Battery,Backlight)
174      gosub SetBacklightLevel
175      if Tick<54300 or Tick>65534 then'outside +/-10% of nominal so use a default
     value.
176         Tick=60400
177      endif
178      if Sf<900 or Sf>1100 then 'if stored scaling factor is >10% error then use
```

```
    default
179         Sf=1000
180      endif
181      gosub LoadBatteryCap
182      pause 1000 'leave some time to read the spash screen
183      if HibernateState=0 then
184            gosub StartNewReadings 'clear Ah Wh Time data
185         else 'hibernationState=1 so device is resuming from Hibernation
186            hi2cout[$7c],($80,$80,$40," Resuming from  ")
187            hi2cout($80,$c0,$40," Hibernation    ")
188            pause 2000
189      endif
190      GainRange=2 'as defined by Gain160 Gain40 settings
191      gosub gainAutoRange'prevents incorrect peak capture on initial cycle
192      pause 80'wait a time close to half a measuring cycle before proceeding
193      gosub gainAutorange'second autorange first reading if 2 range changes
    needed.
194      pause 80'wait a time close to half a measuring cycle before proceeding
195
196   '##########  INITIALIZATION COMPLETE....START MEASURING
    ##########################
197
198    Measure: 'main measuring routine
199      hi2cout [$7C],($80,$01)'clear the display
200      ContinueMeasuring:
201      Timer=$ffff 'preloads timer to overflow in 1 settimer period
202      settimer Tick 'starts timer with period=Tick total loop time target=360ms
203      SetIntFlags %10000000, %10000000 'arm settimer interrupts
204      do while Sw1=0 and Sw2=0 'main measuring loop
205         gosub SourceVoltage
206         gosub LoadCurrent
207         gosub GainAutoRange 'Isense range placed here for max settling time
208         if Vsource<VsourceMin then'records min voltage if gain range hasn't just
    changed
209            VsourceMin=Vsource
210         endif
211         if ILmax<IL then 'save maximum peak current if gain range hasn't changed
212            ILmax=IL
213         endif
214         gosub LoadPower
215         gosub AmpWattHours
216         gosub ElapsedTime
217         gosub BattCapLeft
218         inc clock
219         gosub GainAutoRange'autorange again in case 2 range changes needed this
    cycle
220         'high Backlight1,Backlight2 'used only for development. Provides a loop
    timing test
221         'if loop time is sufficient backlight will blink regularly each cycle
222         pause 1000'pause longer than the Settimer period interrupt will occur
    here
223      loop
224      settimer OFF'get rid of potential interrupts
225      setintflags OFF 'disarm interrupts
226      if Sw1=1 then 'check switches Sw1<2sec=Utilites >2sec Hibernate
    Sw2>2sec=PwrOff
227            for var1=0 to 9
228               pause 70
229               if Sw1=0 then goto Utilities
230            next var1
231            goto GoHibernate
232         else 'Sw2 must be 1
233            for var1=0 to 9
234               pause 70
235                  if Sw2=0 then goto ContinueMeasuring
236            next var1
237            if HibernateState=1 then
238               goto GoHibernate
239            else
240               goto PwrOff
```

```
241             endif
242          endif
243       end
244
245       GainAutoRange:'finds best range for load current measurements
246          'current measurements are divided into 3 ranges
247          'range 0 Gain40=0 Gain160=0 Av=1600 1mA/bit range limit 0-999 0-999mA
248          'range 1 Gain40=0 Gain160=1 Av=160 10mA/bit range 100-999 1.00-9.99A
249          'range 2 Gain40=1 Gain160=0 Av=40 40mA/bit range 250-1023 10.0A-40,9A
250          'may have to use this loop twice to find the right gain
251          readadc10 6,ILraw
252          if GainRange=0 and ILraw>999 then'reduce gain as current>1A
253                high Gain160 'change gain to 400
254                low Gain40
255                GainRange=1
256             elseif GainRange=1 and ILraw>999 then'reduce gain as current >10A
257                low Gain160 'change gain to Av=40
258                high Gain40
259                GainRange=2
260             elseif GainRange=2 and ILraw<250 then'increase gain as current <10A
261                high Gain160 'change gain to Av=160
262                low Gain40
263                GainRange=1
264             elseif GainRange=1 and ILraw<100 then'increase gain as current <1A
265                low Gain160 'change gain to Av=1600
266                low Gain40
267                GainRange=0
268          endif
269       return
270
271       SourceVoltage: 'measures & displays Vsource as xx.xV
272          'Leases
273          'var1 stores dig1 of initial multiply for rounding
274          'varw10 'stores voltage for ADC and presentation
275          readadc10 7,varw10
276          varw10=varw10*24 'with a 6:1 voltage divider and 4.096Vref each A/D bit is
          24mV (4.096/1024*6)
277          'now round up
278          var1=varw10 dig 1 'store voltage reading second least significant bit in
          var1
279          Vsource=varw10/100   'Vsource is now in form xxx
280          if var1>4 then
281                Vsource=Vsource+1 'round up Vsource if Var1=>5
282          endif
283          varw10=Vsource
284          gosub Decimal'returns data for display
285          poke 60,hundred,ten,unit 'put battery voltage in displayable format is SFR
          locations
286       return
287
288       LoadCurrent: 'Load current autoranged and displayed as xxxmA, x.xxA or xx.xA
289       'Leases
290       'Var1 used in calculations to flag rounding
291       'Var2 flags Sf>=1
292       'Varw11 stores intermediate scaling calculation number
293       'Vwrw12 stores intermediate scaling calculation number
294          readadc10 6,ILraw 'take a new reading
295          'now scale the reading
296          if Sf>=1000 then'scale factor is greater than 1
297                var2=1
298                varw12=Sf-1000'varw12 now holds modified Sf
299             else
300                var2=0
301                varw12=Sf
302          endif
303          varw11= varw12 dig 0 * ILraw/20
304          varw11= varw12 dig 1 * ILraw/2 + varw11
305          varw11= varw12 dig 2 * ILraw * 5 + varw11
306          var1=varw11 dig 1
307          varw12=varw11//50
```

```
308       varw11=varw11/50
309       if varw12>=25 then 'round the result to three digits
310          inc varw11
311       endif
312       if var2=1 then 'scale factor is greater than 1
313          ILraw=ILraw+varw11
314          else
315          ILraw=varw11
316       endif
317     'end scaling factor calculation
318     'now calculate IL from scaled ILraw.  Don't round here as watts calculations
319     'need unrounded numbers for best accuracy.
320       if GainRange=0 then
321             IL=ILRaw
322          elseif GainRange=1 then
323             IL=ILraw*10
324          elseif GainRange=2 then
325             IL=ILraw*40
326       endif
327       varw10=IL
328       gosub RoundIt
329       gosub Decimal 'now prepare to display the results
330       if IL<1000 and GainRange=0 then
331             poke 71,hundred,ten,unit,"m"
332          elseif IL<10000 and GainRange<2 then
333             poke 71,thousand,".",hundred,ten
334          else
335             poke 71,tenthousand,thousand,".",hundred
336       endif
337    return
338
339    LoadPower: 'calculates and displays load power in x.xx W,xx.xW, or xxxxW
340       'Leases
341       'varw10 -takes reading to display for parsing  & ASCII conversion
342       'varw12 -stores IL/10 to prevent mult. overflow
343       'var1 - used for rounding
344       'rem Vsource max=246 ILmax=40920+10%=45012
345       if IL<267 then 'max=6.051W
346             varw10=IL*Vsource
347             gosub RoundIt
348             gosub Decimal
349             poke 63,tenthousand,".",thousand,hundred 'display x.xxW
350          elseif IL >=267 and IL<1331 then 'max=32.7W
351             varw10=IL/5*Vsource/2
352             gosub RoundIt
353             gosub Decimal
354             if Varw10<10000 then
355                   poke 63,thousand,".",hundred,ten'display x.xxW to 9.99W
356                else
357                   poke 63,tenthousand,thousand,".",hundred'display xx.xW to32.7W
358             endif
359          elseif IL>=1331 and IL<=26600 then 'max=130W to 5321 and 654W
360             if IL<5321 then
361                   varw10=IL/20*Vsource/5
362                else
363                   varw10=IL/100*Vsource
364             endif
365             gosub RoundIt
366             gosub Decimal
367             if Varw10<10000 then
368                   poke 63,thousand,hundred,".",ten
369                else
370                   poke 63," ",tenthousand,thousand,hundred
371             endif
372          else if IL>26600 then'max=999W
373             varw10=IL/200*Vsource/5
374             gosub RoundIt
375             gosub Decimal
376             if Varw10<10000 then
377                   poke 63," ",thousand,hundred,ten
```

```
378               else
379               poke 63,tenthousand,thousand,hundred,ten
380            endif
381      endif
382   return
383
384   ElapsedTime:'calculates and displays elapsed time in HH:mm format
385      'clock is stored in tick increments up to 10000 then Hours is incremented.
386   'Leases
387   'varw10 - clock minutes and total time calculations
388      if clock>10000 then
389         inc hours
390         if hours >99 then'reset hours to zero if >99
391            hours=0
392         endif
393         clock=clock-10000
394      endif
395      varw10=clock*6/1000 'find minutes
396      varw10=hours*100+varw10 'add hours to the number displayed
397      gosub Decimal
398      if varw10<1000 then
399            thousand=" " 'blank out tens of hours rather than cram display
400      endif
401      poke 67,thousand,hundred,ten,unit
402   return
403
404   BattCapLeft: 'shows approximate remaining battery capacity as xx%
405      if Vsource>179 then 'limit Vsource to stored number limits
406            Vsource=179
407         elseif Vsource<52 then
408            Vsource=52
409      endif
410      get Vsource,Varw10
411      gosub Decimal
412      poke 80,ten,unit
413   return
414
415   #rem
416   'Use this code to display Amplifier range in place of battery capacity
417   'but don't forget to remark out BattCapLeft subroutine.
418   BattCapLeft:
419   varw10=GainRange
420   gosub Decimal
421   poke 80,"0",unit
422   return
423   #endrem
424
425   AmpWattHours:'calculates and displays battery use in Amp hours or Watt hours.
426      'displayed as xxxm, x.xx, xx.x, or xxx Ah or Wh
427      'Calculations are based on collecting 10k samples per hour (360ms per
      sample or
428      'about 2.8 samples per second)
429      '
430      '#########################  BEGIN AMP HOUR CALCULATION
      ########################
431      'rem IL then ranges between 0 and 40,960.  Could be as high as 45012 if
      Ilraw correction -10%
432      'Each measuring cycle accumulates amp hours as IL value + nAh *10^-7Ah
433      'Reading interval is 10^-4 (10k samples per hour).
434      'Unit reading is lowest current
      reading*ReadingInterval=10^-3A*10^-4hr=10^-7Ah
435      'Smallest reading is 1mA in 1/10000hr = 10^-7Ah so every 10,000=1mAh.
436      'and 1000mAh=1Ah
437      nAh=IL+nAh 'accumulate battery use
438      if nAh>=10000 then 'if over 1mAh
439         var1=nAh dig 4 'large IL can accumulate more than 10k every measuring
      cycle so find out.
440         varw11=var1*10000
441         mAh=mAh+var1
442         nAh=nAh-varw11 'adjust nAh accumulator to account for mAh
```

```
443       endif
444       if mAh >= 1000 then 'accumulate Amp hours and adjust the mAh register
445             Ah=Ah+1
446             If Ah>9999 then 'roll over Ah if value gets to 10k.
447                Ah=0
448             endif
449             mAh=mAh-1000
450       endif
451       '########################  BEGIN WATT HOUR CALCULATION
      ########################
452       'IL is saved in mA and can range from 0 to 40,920 +10% = 45,012
453       'Vsource ranges from 52 (5.2V) to 246 (24.6V)
454       'Reading interval is 10^-4 (10k samples per hour).
455       'Unit incrment is IL*Vsource*ReadingInterval=10^-3A*10^-1V*10^-4hr=10^-8 Wh
456       'Smallest reading is 1mA*0.1V in 1/10000hr = 10^-8Ah so every 100,000=1mWh..
457       'Multiplying Vsource with Il to get 10^-4Wh (base unit
      0.1V*10^-3A*10^-4h=10^-8Wh)
458       'can easily overflow the processor's 65535 limit.  So this calculation is
      broken into four
459       'ranges to prevent overflow while minimizing rounding errors. .
460       'Both base watt hour accumulation unit named uWh and the current IL are
      divided
461       'by a factor in each range that optimizes the resolution while preventing
      the product
462       'from overflowing.
463       'rem max Vsource is 246
464       'rem symbol uWh is actually 10^-8 mWh  NOT 10^-6 and the symbol mWh is
      10^-4 Wh NOT 10^-3 Wh
465          if IL<226 then 'first range for small values of IL.  In this range IL is
      not divided.
466          uWh=IL*Vsource+uWh'uWh max =IL*Vsource+10000=225*246+10000=65360
467          var1=uWh dig 4 'large Vsource & IL within range can accumulate more than
      10k uWh every cycle.
468          varw11=var1*10000
469          mWh=mWh+var1
470          uWh=uWh-varw11
471          elseif IL>=226 and IL<=1290 then 'second range Wh and IL are divided by 5
472             uWh=uWh/5   'change uWh to a value suitable for this range (overflow
      vs accuracy)
473             uWh=IL/5*Vsource+uWh
474             mWh=uWh/2000+mWh 'as IL uWh were divided by 5 only need to * by 2000
      to =10k
475             uWh=uWh//2000*5'before leaving determine remainder uWh and
      re-normalize
476          elseif IL>=1292 and IL<=5286 then 'third range Wh and IL are divided by
      20
477             uWh=uWh/20 'change uWh to a value suitagle for this range (overflow
      vs accuracy)
478             uWh=IL/20*Vsource+uWh '/20 uWh max
      =IL/20*Vsource+uWh=4396/20*246+10000/20=65444
479             mWh=uWh/500+mWh 'calculate mWh
480             uWh=uWh//500*20 'calculate residual
481          elseif IL>=5287 and IL<=26599 then 'fourth range Wh & Il are divided by
      100
482             uWh=uWh/100
483             uWh=IL/100*Vsource+uWh '/100 uWh max=IL/100*Vsource+uWh=65290
484             mWh=uWh/100+mWh
485             uWh=uWh//100*100
486          elseif IL>26599 then 'fifth range for very large currents.
487             uWh=uWh/200
488             uWh=IL/200*Vsource+uWh '/200 uWh max
      =IL/200*Vsource+uWh=45012/200*246*9999/200+1=55175
489             mWh=uWh/50+mWh
490             uWh=uWh//50*200 'residual
491       endif
492       if mWh>=10000 then 'if 1000 real mWh (10k in this calculation) accumulated
      increment Wh & decrement mWh
493          Wh=Wh+1
494          if Wh>9999 then 'roll over Wh if value gets to 10k
495             Wh=0
```

```
496         endif
497         mWh=mWh-10000
498      endif
499      'Now display either Ah or Wh
500      if AhWh=0 then 'display Ah
501               if Ah=0 then 'under 1Ah so display in mAh
502                  varw10=mAh
503                  gosub Decimal
504                  poke 75,hundred,ten,unit,"m","A"
505               else if Ah<100 then
506                  var1=mAh dig 2*10 'parse hundreds of an Ah (don't bother to
    round up)
507                  var2=mAh dig 1
508                  varw10=Ah*100+var1+var2 'creates Ah up to 3 digits for display
    parsing
509                  gosub Decimal
510                  if varw10<1000 then 'Ah is less than 10
511                        poke 75,hundred,".",ten,unit,"A"
512                     else
513                        poke 75,thousand,hundred,".",ten,"A"
514                  endif
515               else
516                  varw10=Ah
517                  gosub Decimal
518                  if varw10<1000 then
519                        poke 75," ",hundred,ten,unit,"A"'supress leading zero
520                     else
521                        poke 75,thousand,hundred,ten,unit,"A"'max display is
    6553Ah
522                  endif
523               endif
524      endif
525      if AhWh=1 then'display Wh
526            'Now display mWh or Wh
527            'For display remember Wh to three or four digits if >1k
528            if Wh=0 then
529                  varw10=mWh
530                  gosub Decimal
531                  poke 75,thousand,hundred,ten,"m","W"
532               elseif Wh<100 then 'count is over 1Wh but less than 100
533                  var1=mWh dig 3*10'parse for tenths of a Watt hour
534                  var2=mWh dig 2'parse for 100'th of a Watt hour
535                  varw10=Wh*100+var1+var2'creates Wh with .1Wh resolution for
    display
536                  gosub Decimal
537                  if varw10<1000 then
538                        poke 75,hundred,".",ten,unit,"W"
539                     else
540                        poke 75,thousand,hundred,".",ten,"W"
541                  endif
542               else 'Wh>100
543                  varw10=Wh
544                  gosub Decimal
545                  if varw10<1000 then 'Wh is less than 1000 so blank leading 0
546                        poke 75," ",hundred,ten,unit,"W"
547                     else 'Wh is over 1000 so display leading digit.
548                        poke 75,thousand,hundred,ten,unit,"W"
549                  endif
550            endif
551      endif
552   return
553
554   LoadBatteryCap:'loads a lookup table of battery capacity into scratchpad memory
555      'Vsource reading (3 digits) is used as the address, battery capacity in %
    is the data
556      'So to get battery capacity code -  get vbat,var1 where var1 now holds batt
    cap.
557      'capacity is displayed as a two digit number = xx%.
558      'Uses all 128 bytes of scratchpad memory.
559
```

```
560      'First all readings for the selected battery are assembled in FSR locations
    65-76
561      'add 053 as the first entry and 179 as the last to the FSR list. This
    represents
562      '5.3V minimum and 17.9V maximum, the range limited by 128 bytes of
    available memory
563      BattPointer=2*Battery*$10+$10'set pointer to start of battery entry numeric
    data
564      for var1=65 to 76
565         read BattPointer,var2
566         poke var1,var2
567         inc BattPointer 'get next entry
568      next
569      poke 64,53  'store minimum battery voltage reading
570      poke 77,179 'store max battery voltage reading
571      'now associate the battery readings to a lookup table that will return
    battery capacity
572      'To do this populate the scratchpad so that when an address is used that
    equals
573      'Vsource the entry returns the battery capacity as defined by the key
    percentage
574      'markers in the list stored in the FSR.
575      'with entries such that using any voltage between min and max as a 'get'
    address
576      'will return the percentage as prescribed by the battery entry data
577      'because the 20X2 only has 128byte scratchpad  entries from 128 to 179
578      'will appear in scratchpad 0 to 51 52=0
579      'But addressing >128 loop the counter & return the correct battery
    capacity.
580      InnerLoopLo=64
581      InnerLoopHi=65
582      for var1=0 to 12
583         lookup var1,(0,2,5,10,20,30,40,50,60,70,80,90,99),BattCap'now contains
    %cap
584         peek InnerLoopLo,sbattlo
585         peek InnerLoopHi,sbatthi
586         for var3=sbattlo to sbatthi
587            put var3,BattCap
588         next
589         inc InnerLoopLo
590         inc InnerLoopHi
591      next
592   return
593
594     '###################### UTILITIES SECTION ################################
595   Utilities:
596     'displays main title on the display line 1 and menu items on line 2
597     'pressing S1 increments menu choice, S2 selects
598     'While in utilities menus time stands still, Ah & Wh don't accumulate.
599      'Leases
600      'None
601      MainMenuItem=0 'keeps track of menu items; init to 0
602      MainMenuInnerMenuLoop:
603      if MainMenuItem>7 then
604         MainMenuItem=0
605      endif
606      gosub ClearDisplay
607      hi2cout [$7c],($80,$80,$40,$01,"Scroll BM Utils")'write MainMenu header
608      hi2cout ($80,$c0,$40,$7f)'set display for start for most menu items to save
    code
609      on MainMenuItem gosub
    MenuMinAndPeak,MenuClearReadings,MenuChooseAhWh,MenuSetBacklight,MenuSelectBatte
    ry,MenuGoHibernate,MenuPwrOff,MenuCalibrate
610      pause 400 'wait a bit so display can be viewed
611      gosub WaitForSwitch
612      if Sw1=1 then
613            pause 100 'bit of time to scroll menu if button held down
614            inc MainMenuItem
615         elseif Sw2=1 then
616            pause 400 'get finger off button
```

```
617            gosub ClearDisplay
618            on MainMenuItem gosub
       MinAndPeak,ClearReadings,ChooseAhWh,SetBacklight,SelectBattery,GoHibernate,PwrOf
       f,Calibrate
619            pause 400 'get finger off button
620            goto EndMainMenuLoop 'returned from subroutine so go back to measuring
621         else
622            goto EndMainMenuLoop'Neither switch was pushed and WaitForSwitch
       timed out so resume measuring
623      endif
624      goto MainMenuInnerMenuLoop
625      EndMainMenuLoop:
626      goto Measure
627
628   MenuMinAndPeak: 'Restarts display
629      hi2cout ($40,"Show Vmin & ILp")
630   return
631   MenuClearReadings: 'Restarts display
632      hi2cout ($40,"Clear Readings")
633   return
634   MenuChooseAhWh: 'Selects either Amp hours or Watt hours for display
635      hi2cout ($40,"Choose Ah or Wh")
636   return
637   MenuSetBacklight: 'sets backlight level
638      hi2cout ($40,"Set Backlight")
639   return
640   MenuSelectBattery:'Battery Type selection submenu
641      hi2cout ($40,"Select Battery")
642   return
643   MenuGoHibernate: 'Hibernation subroutine
644      hi2cout ($40,"Hibernate")
645   return
646   MenuPwrOff:
647      hi2cout ($40,"Power Off")
648   return
649   MenuCalibrate: 'Calibration submenu
650      hi2cout ($40,"Calibrate")
651   return
652
653   MinAndPeak: 'Displays minimum voltage and peak current
654      hi2cout ($80,$80,$40,$7f,"Y Clr Vmin & Ipk?")
655      varw10=VsourceMin
656      gosub Decimal'returns data for display
657      hi2cout($80,$c0,$40,$7f,"N ",hundred,ten,".",unit,"Vm")
658      varw10=ILmax
659      gosub Decimal
660         if ILmax>10000 then
661               hi2cout($80,$ca,$40,tenthousand,thousand,".",hundred,"Ap")
662            elseif ILmax<1000 then
663               hi2cout($80,$ca,$40,hundred,ten,unit,"mAp")
664            else
665               hi2cout($80,$ca,$40,thousand,".",hundred,ten,"Ap")
666         endif
667      gosub WaitForSwitch
668      if Sw1=1 then 'reset peak values
669         VsourceMin=255
670         ILmax=0
671      endif
672   return
673
674   ClearReadings:'clears all stored readings resetting to zero & resets
       hibernation state
675      hi2cout ($80,$80,$40,$7f,"Y  Clear")
676      hi2cout ($80,$c0,$40,$7f,"N  Readings?")
677      pause 400 'finger off button
678      gosub WaitForSwitch
679      if Sw1=1 then 'begin the process of clearing all readings
680            pause 400 'finger off button
681            if HibernateState=1 then 'only wirte the EEPROM if necessary
682               HibernateState=0
```

```
683               low wpEEPROM 'enable EEPROM write
684               hi2cout[$a0],$00,(HibernateState)'and save int in EEPROM
685               pause 10'EEPROM write cycle time
686               high wpEEPROM
687            endif
688            gosub StartNewReadings
689         elseif Sw2=1 then
690            pause 400 'finger off button
691      endif
692   return
693
694   ChooseAhWh:'Menu item selects display format Amp hours or Watt hours
695      hi2cout ($80,$80,$40,$7f,"Ah  Sel Display")
696      hi2cout ($80,$c0,$40,$7f,"Wh  Format")
697      gosub WaitForSwitch
698      if Sw1=1 then
699            AhWh=0
700         elseif Sw2=1 then
701            AhWh=1
702      endif
703      low wpEEPROM 'enable EEPROM write
704      hi2cout[$a0],$10,(AhWh) 'and save in EEPROM
705      pause 200'EEPROM write cycle time & finger off button
706      high wpEEPROM
707   return
708
709   SetBacklight:'sets backlight to one of four levels: off,low,med,hi and saves
      setting
710      hi2cout ($80,$80,$40,$01," Next BL Level")'set up display heading for this
      section
711      hi2cout ($80,$c0,$40,$7f," Backlight")
712
713      if Backlight=0 then
714            hi2cout ($80,$cc,$40,"Off")
715         else if Backlight=1 then
716            hi2cout ($80,$cc,$40,"Low")
717         else if Backlight=2 then
718            hi2cout ($80,$cc,$40,"Med")
719         else 'Backlight must be 3
720            hi2cout ($80,$cc,$40,"Hi ")
721      endif
722      gosub SetBacklightLevel
723      gosub WaitForSwitch
724      if Sw1=1 then
725         pause 400 'finger off switch
726         inc Backlight
727         if Backlight>3 then
728            Backlight=0
729         endif
730         goto SetBacklight
731      elseif Sw2=1 then
732         pause 400'finger off switch
733         low wpEEPROM 'save backlight setting in EEPROM
734         hi2cout[$a0],$19,(Backlight)
735         pause 10'wait for write to complete
736         high wpEEPROM
737      else 'WaitForSwitch timed out so reset the backlight level & continue
      measuring
738         hi2cin [$a0],$19,(Backlight)
739         gosub SetBacklightLevel
740      endif
741      return
742
743   SetBacklightLevel:
744      if Backlight=0 then 'backlight is off
745            low Backlight1,Backlight2
746         elseif Backlight=1 then'backlight is on low
747            input Backlight2
748            high Backlight1
749         elseif Backlight=2 then 'backlight is on medium
```

```
750            input Backlight1
751            high Backlight2
752         elseif Backlight=3 then 'backlight is on high
753            high Backlight1,backlight2
754      endif
755   return
756
757   SelectBattery: 'start with battery being used then cycle through battery
      choices
758      hi2cout ($80,$80,$40,$01," Next Battery")
759      BattPointer=2*Battery*$10'set pointer to start of battery entry numeric data
760      for var1=$c1 to $cf
761         read BattPointer,var2
762         hi2cout ($80,$c0,$40,$7f)
763         hi2cout ($80,var1,$40,var2)
764         inc BattPointer 'get next entry
765      next
766      gosub WaitForSwitch
767      if Sw1=1 then
768            pause 400 'get finger off button
769            inc Battery
770            if Battery=8 then'only 8 batteries in the list
771               Battery=0
772            endif
773            BattPointer=2*Battery*$10
774            read BattPointer,Var2
775            inc BattPointer
776            read BattPointer,Var3
777            if Var2=$ff and Var3=$ff then'beginning of any empty entries uses
      $ff,$ff as tag
778               Battery=0
779            endif
780            goto SelectBattery
781         elseif Sw2=1 then
782            pause 400 'get finger off button
783            low wpEEPROM 'write enable EEPROM
784            hi2cout[$a0],$18,(Battery)
785            pause 10 'time for write
786            high wpEEPROM'write protect EEPROM
787            gosub LoadBatteryCap
788         else 'WaitForSwitch timed out so get original battery info from EEPROM
789            hi2cin[$a0],$18,(Battery) 'get the stored battery info
790      endif
791   return
792
793   GoHibernate:'FlagHibernation save current data in EEPROM and go to sleep
794      HibernateState=1
795      low wpEEPROM'enable EEPROM write
796      'External EEPROM write requirements - 8byte max writes must start on a
797      'writepage boundary, $00,$08,$10,$18 etc.
798      '                                                Time    nAh     mAh
799      hi2cout [$a0],$00,(HibernateState,Hours,b32,b33,b36,b37,b38,b39)
800      pause 10 'write time
801      '               Ah     uWh     mWh     Wh
802      hi2cout $08,(b40,b41,b42,b43,b44,b45,b46,b47)
803      pause 10 'write time
804      '                        ILmax
805      hi2cout $11,(VsourceMin,b54,b55)
806      pause 10 'write time
807      high wpEEPROM'disable EEPROM write
808   'now proceed to PwrOff NB PwrOff must start after GoHibernate
809
810   PwrOff:
811      gosub ClearDisplay
812      if HibernateState=1 then
813            hi2cout($80,$82,$40,"Hibernating")
814         else
815            hi2cout($80,$86,$40,"Bye")
816      endif
817      hi2cout[$7c],($80,$c6,$40,"Bye")
```

```
818       pause 1000'wait a bit for the button to be released or unit will repower
819       low shdn
820       pause 2000 'pause 4 seconds to make sure user finger is off the pushbutton
821       high shdn'just in case it didn't shut down mustn't stay low or Sw1
      inoperative
822   goto Initialize
823
824   Calibrate:'store value for ScaleFactor; calculate and store Tick
825   #rem
826   Scale Factor
827   The scale factor corrects current measurements accounting for errors in Vref
      voltage,
828   shunt resistor value, and current to voltage converter gain based on the gain
      of
829   GainRange0. Note that this routine does not calculate the scaling factor, it
      only
830   enables the user to enter and save a calculated scale factor.
831   To accurately calculate the scale factor follow these steps:
832   Apply a load current to the battery monitor that is close to the upper end of
833   GainRange0. The current should be in the range of 930 to 990mA with an known
834   naccuracy of +/-.25%.  A vriable power supply around 12 to 15 volts and a
      stable
835   power resistive load of about 15 ohms works well to produce the desired
      current.
836   Insure BatteryMonitor remains in GainRange0. If it autoranges to a higher
      range,
837   reduce the current to the upper range of GainRange 0.
838   The load current can be measured using a current meter or a precision shunt
      resistor
839   in series with BatteryMonitor and the resistive load. If a resistive shunt is
      used
840   use one in the range of 1.0 or 0.1 Ohms with a tolerance of no greater than
841   0.25%.  The voltage can then be measured across the shunt resistor with an
      accurate
842   voltmeter and the load current calculated.
843   Note the current reading on the display.  Calculate the scale factor by
      dividing
844   the measured load current by the reading on the display.
845   Enter this number in the Sf routine, The Sf routine will initially display the
846   present value for the scale factor. The reading takes the form xxxx where the
      most
847   significant digit is either zero or one. For example 1.056 enters as 1056,
      0.983
848   as 0983. To enter a new value push Sw1 next to the up arrow to increase the
      displayed
849   reading or Sw2 next to the down arrow to decrease the reading.
850   The unit accepts any value within a 10% limit.(0900 to 1100).
851   Because of code space limitations you can enter and store a number outside
852   the limit range but on restart the unit will default to a Sf=1.000
853
854   Tick
855   In order to accumulate Ah & Wh accurately the unit must collect 10,000 samples
856   per hour 3600/10000=0.360 seconds per cycle. Cycle time is controlled
857   by a Timer/Settimer interrupt loop. All BatteryMonitor measurements
858   and calculations are placed within this timed loop. Two activities, servicing
      the
859   loop interrupt and writing the display are outside the timing loop.
860   But these activities use a fixed time per cycle as the code is straight line,
861   no subroutines or branching. At a 4MHz clock these activities use about 8% of
862   the total period so the interrupt driven timer that controls the loop time is
      adjusted,
863   reduced by 8%, to account for the external tasks.
864   Timer is set to ffff so it will overflow in one major tick and Setimer is
      preloaded so
865   that it overflows and creates an interrupt in a fixed period.
866   The processor clock is trimmed to within 1% frequency during manufacture so is
      quite
867   accurate without calibration tweaking.  But the cycle can be made even more
      accurate
868   by adjusting the Settimer preload value by a number of 'ticks'.to account for
```

```
    any
869 frequency errors. At 4MHz each 'tick' changes the cycle time by 64us* 1+%time
    outside
870 the loop or 70us.
871 'Each hour can therefore be adjusted by +/-700mS.  So the best accuracy that
    can
872 'be expected is about 2-3 seconds in three hours.
873 'Tick will run with a default value if a calibration hasn't been performed.
874 'A stop watch is required to calibrate the time.
875 'A reliable power source such as a battery is required to power BatteryMonitor
876 'A load isn't required for this procedure.
877 'Invoke the utilities menu by powering the unit and pressing S1
878 'Navigate to the 'Clear Readings' menu item and select it.
879 'Press S1 to Clear readings and at the same time start the stopwatch
880 'Come back about three hours later
881 'Watch the time display.  Just as the minutes readout changes simultaneously
882 'push Sw1 and stop the stopwatch.  When Sw1 is pushed the clock stops
    incrementing
883 'and the user enters the Utilities menu.  In the Utilities menu navigate to
    the
884 'Calibrate menu then push Sw2 to enter the Time-Tick Cal submenu.  From here
885 'Enter the time from the stopwatch, first hours, minutes, then seconds.
886 'Press Sw2 to calculate then save the new value for Tick.
887 'Also note this value for future reference.
888 'It may be necessary to perform this cal procedure a couple times until Tick
889 'no longer changes by more than a few counts.
890 #endrem
891 'Leases
892 'Var1 stores hours
893 'var2 stores seconds
894 'var3 negative flag
895 'var10 stores difference between internal and external seconds & displays no.
896 'var11 stores internal elapsed time in seconds
897 'var12 stores external seconds elapsed time in seconds
898    hi2cout ($80,$80,$40,$7f," Sf Set")
899    hi2cout ($80,$c0,$40,$7f," Time-Tick Cal")
900    gosub WaitForswitch
901    if Sw1=0 and Sw2=0 then 'WaitForSwitch timed out so go back to measuring
902       goto EndCal
903    elseif Sw1=1 then
904       goto ScaleFactorSet
905    endif
906 'Note The WaitForSwitchTimeout is NOT active when in either the Sf Set or
    Time-Tick Cal routines
907    var1=3 'temporary storage for hours
908    var2=0 'temporary storage for minutes
909    varw11=0 'temporary storage for seconds
910    pause 400'get finger off button
911    GetHours:
912    varw10=var1
913    gosub Decimal
914    gosub ClearDisplay
915    hi2cout ($80,$80,$40,$00," Chg Hrs")
916    hi2cout ($80,$c0,$40,$7f," ",unit)
917    gosub WaitForSwitch
918    if Sw1=1 then
919       pause 200 'finger off button
920       inc var1
921       if var1>8 then 'arbitrary maximum time interval -8hrs 59 min 59 sec.
922          var1=0
923       endif
924       elseif Sw2=1 then
925          pause 400 'get finger off switch
926          goto GetMinutes
927    endif
928    goto GetHours
929    GetMinutes:
930    varw10=var2
931    gosub Decimal
932    hi2cout ($80,$80,$40,$00," Chg Min")
```

```
933       hi2cout ($80,$c0,$40,$7f," ",ten,unit)
934       gosub WaitForSwitch
935       if Sw1=1 then
936          pause 100 'increment display time control
937             inc var2
938             if var2>59 then
939                var2=0
940             endif
941          elseif Sw2=1 then
942             pause 400 'get finger off switch
943             goto GetSeconds
944       endif
945       goto GetMinutes
946       GetSeconds:
947       varw10=varw11
948       gosub Decimal
949       hi2cout ($80,$80,$40,$00," Chg Sec")
950       hi2cout ($80,$c0,$40,$7f," ",ten,unit)
951       gosub WaitForSwitch
952       if Sw1=1 then
953             pause 100 'increment display time control
954             inc varw11
955             if varw11>59 then
956                varw11=0
957             endif
958          elseif Sw2=1 then
959             pause 400 'get finger off switch
960             goto AskIfTimeCorrect
961       endif
962       goto GetSeconds
963       AskIfTimeCorrect:
964       hi2cout ($80,$80,$40,$7f,"Y Time Ok?")
965       hi2cout ($80,$c0,$40,$7f,"N  ")
966       varw10=var1 'show hours
967       gosub Decimal
968       hi2cout ($40,unit,":")
969       varw10=var2 'show minutes
970       gosub Decimal
971       hi2cout($40,ten,unit,":")
972       varw10=varw11'show seconds
973       gosub Decimal
974       hi2cout($40,ten,unit)
975       gosub WaitForSwitch
976       if Sw2=1 then
977             pause 400 'get finger off switch
978             goto GetHours 'Time entry was wrong so try again
979       endif
980       varw12=var1*3600'stopwatch hours to seconds
981       varw12=var2*60+varw11+varw12'recorded stopwatch time in seconds
982       varw11=Clock*3'now calculate internal time in seconds as there are
983       varw11=Clock*6/10+varw11'10000 samples per hour sec=clock*.36 so
984       var1=varw11 dig 0 'multiply clock by 3 then add remainder 6/10 then
985       varw11=varw11/10'divide by 10 to get value *.36 then round
986       if var1>4 then
987          inc varw11'now have Clock value in seconds so have to.
988       endif          'calculate total time in seconds in next line
989       varw11=Hours*3600+varw11'calculates total internal clock time in seconds
990       varw10=varw12-varw11'seconds diff between Stopwatch and internal clock
991       'positive var10 indicates Stopwatch is faster than internal clock so
992       'Tick should be increased.
993       if varw10>32768 then'if number is greater than half it's likely negative
      so...
994             varw10=varw11-varw12'simpler than two's compliment
995             var3=1'set flag indicates internal clock is faster than stopwatch
996                   'so Tick delay should be shortened.
997       endif
998       'each Tick shifts the internal clock by 0.64 seconds per hour so calculate
999       'number of whole ticks to change by multiplying sec./hr *1.563
1000      'to do this while optimizing error correcting range and resolution
1001      'multiply seconds error *15.63 then divide by stopwatch time in seconds
```

```
    '/36 to get decimal hours *10.
    'This will correct for clock errors in excess of 10% without overflow
    'Note PICAXE devices with internal clocks are trimmed to 1%.
    varw12=varw12/360'total no of hours *10 to optimize resolution
    varw11=varw10*63/100'remainder for 15 63
    varw10=varw10*15+varw11/varw12 'delta sec*15+remainder/timedecimalHrs*10
    if var3=1 then
          varw12=Tick-varw10
       else
          varw12=Tick+varw10
    endif
    varw10=Tick
    gosub Decimal
    hi2cout[$7c],($80,$80,$40,$7f,"Y Save New Tick?")
    hi2cout($80,$c0,$40,$7f,"N ",tenthousand,thousand,hundred,ten,unit)
    varw10=varw12
    gosub Decimal
    hi2cout($40," ",$7e," ",Tenthousand,thousand,hundred,ten,unit)
    pause 600 'get finger off switch
    gosub WaitForSwitch
    if Sw1=1 then
       Tick=varw12
       low wpEEPROM'enable EEPROM write
       hi2cout[$a0],$14,(b34,b35) 'save Tick
       pause 10
       high wpEEPROM'disable EEPROM write
       varw10=Tick
       gosub Decimal
       gosub ClearDisplay
       hi2cout[$7c],($80,$83,$40,"Tick=",tenthousand,thousand,hundred,ten,unit)
       pause 2000 'some time to read display
    endif
EndCal:
return

ScaleFactorSet:
    varw10=Sf
    gosub Decimal
    gosub ClearDisplay
    hi2cout ($80,$80,$40,$00," Chg Sf")
    hi2cout ($80,$c0,$40,$01," ",thousand,hundred,ten,unit)
    gosub WaitForSwitch
    pause 100   'wait a bit in case both switches are pressed
    if Sw1=1 and Sw2=1 then
          goto WriteSfEEPROM
       else if Sw1=1 then
          inc Sf
       else if Sw2=1 then
          dec Sf
    endif
    goto ScaleFactorSet
    WriteSFEEPROM:
       low wpEEPROM'enable EEPROM write
       hi2cout[$a0],$16,(b50,b51) 'save Sf
       pause 10
       high wpEEPROM'disable EEPROM write
       gosub ClearDisplay
       hi2cout[$7c],($80,$84,$40,"Sf=",thousand,hundred,ten,unit)
       pause 2000 'get finger off button
EndScaleFactorSet:
return

StartNewReadings:'zeros clock and Ah Wh data to start new readings
    clock=0
    Hours=0
    nAh=0
    mAh=0
    Ah=0
    uWh=0
    mWh=0
```

```
1072     Wh=0
1073     VsourceMin=255
1074     ILmax=0
1075  return
1076
1077  RoundIt: 'used to round many calculations
1078  'This code rounds numbers for presentation to the display.  It assumes that
     three
1079  'digits will be displayed.  If the number is less than 10000 then the units
     digit
1080  'is rounded, If over 10,000 the the tens digit is rounded.
1081  'Rounding up occurs if the digit is 5 or more.
1082  'Leases
1083  'var1 - rounding flag
1084  'var10- input to the routine for rounding, output rounded result also is varw10
1085        if varw10<1000 then 'nothing to round so leave
1086           goto EndRoundIt
1087        elseif varw10<10000 then
1088           var1=varw10 dig 0'round up units to tens for small values
1089           if var1>4 then
1090              varw10=varw10+10
1091           endif
1092        else 'variable is over 10,000
1093           var1=varw10 dig 1'round up tens to hundreds
1094           if var1>4 then
1095              varw10=varw10+100
1096           endif
1097     endif
1098  EndRoundIt:
1099  return
1100
1101  Decimal: 'parses contents of varw10 to digits that can be used directly by the
     display
1102    'Leases
1103    'varw10 used as value to parse
1104     unit=varw10 dig 0 + "0" 'format for display by taking each digit to be
     displayed
1105     ten=varw10 dig 1 + "0"  'and placing it in a variable for use with the
     hi2cout command
1106     hundred=varw10 dig 2 + "0"
1107     thousand=varw10 dig 3 + "0"
1108     tenthousand=varw10 dig 4 + "0"
1109  return
1110
1111  SplashScreen:'Creates information displayed to the user at power up.
1112     hi2cout($40,"BatteryMon V1.0g")'cursor is at home pos so write first line
1113     hi2cout($80,$c0,$40,"Source",$7e,$7e,$7e,$7e,$7e,$7e,"Load")'cursor to
     start of 2nd line & write
1114  return
1115
1116  ClearDisplay:'Erases all displayed data so that the screen is blank.
1117     hi2cout [$7C],($80,$01) 'clear display
1118     pause 1
1119  return
1120
1121  WaitForSwitch:'Waits for one of two push buttons to be closed and does a cheap
     100ms debounce.
1122     WaitForSwitchTimeout=0 'reset WaitForSwitch timeout register
1123     do while Sw1=0 and Sw2=0 and WaitForSwitchTimeout<200
1124     pause 50'really cheap debounce & part of WaitForSwitchTimeout
1125     inc WaitForSwitchTimeout' Total time out =50*2*200=20sec.
1126     loop
1127  return
1128
1129  Interrupt:'special subroutine is entered each time Tick overflows from $ffff.
1130     WriteDisplay:'2line x 16 char display updated here. MUST be done OUTSIDE the
1131                  'Timer Settime loop or code is corrupted by I2C output routine!
1132     'low Backlight1,Backlight2'a test of loop timing
1133     peek 60,b1,b2,b3,b4,b5,b6,b7,b20,b21,b22,b23
1134     hi2cout($80,$80,$40,b1,b2,".",b3,"V ",b4,b5,b6,b7,"W",b20,b21,":",b22,b23)
```

```
1135      peek 71,b1,b2,b3,b4,b5,b6,b7,b8,b20,b21,b22
1136      hi2cout($80,$c0,$40,b1,b2,b3,b4,"A ",b5,b6,b7,b8,b20,"h ",b21,b22,"%")
1137      Timer=$ffff'preload timer to flag at next tick
1138      Settimer Tick
1139      SetIntFlags %10000000,%10000000    'Set Timer 0 to interrupt
1140   return
```