BatteryMonitor Software Overview

Rights & Permission

This document was created by Wayne Getchell of Sagacitic Solutions, amateur radio callsign VE3CZO. It is meant for reference use. 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.

Please read the BatteryMonitor Circuit Description document before tackling this one as hardware blocks are talked about in this document without providing any descriptive detail.

BatteryMonitor measures, calculates and then displays six values on its 16 character two line LCD display; source voltage, load current, load power, accumulated battery use in either Amp hours or Watt hours, elapsed time, and percent remaining battery capacity.
BatteryMonitor's central concept is the use of an interrupt driven timing loop to execute 10,000 measurement cycles per hour or about 2.8 cycles per second accumulating battery use information.
During each 360ms cycle the source voltage and load current are measured. From these measurements load power is calculated. Additionally calculations for accumulated battery use, in Amp hours and Watt hours are performed. Next battery capacity is estimated based on a stored battery characterization table that plots the battery's terminal voltage against percent remaining capacity. Lastly during each measurement cycle the clock is incremented keeping track of the total elapsed time.
BatteryMonitor's accuracy depends on the accuracy of the battery voltage and load current measurements, the accuracy of the calculations that produce voltage, current and battery use readings and the accuracy of the timing loop in achieving 10,000 measurement cycles per hour as this is used to accumulate Amp hour and Watt hour battery use parameters.

Notes about

1. ***Memory Use***
   BatteryMonitor memory use is detailed in the BatteryMonitor Memory Map document.
   All of the 20X2 EEPROM is used for battery data storage.
   All 128 bytes of scratchpad memory is used as a lookup table for the battery terminal voltage versus capacity data.
   Special Function Registers (SFR) locations $3c to $4d are used to hold data to be written to the display as well as locations used by the 'LoadBatteryCap' subroutine.
2. ***Symbol Reuse***
   The PICAXE 20X2 provides 55 Special Function Registers (SFR) for symbols used either as single 8 bit registers (b0 to b54), 16 byte word registers (w0-w27) or a combination of both. Some of these registers are reused.
   Registers b2, called var1; b3 called var2, and b4 called var3 are used as generic variable registers and are leased by many subroutines. These register values are thus valid only within the subroutine in which they are used, the register is considered reusable so is generally NOT valid across subroutines.
   Likewise word registers w10, varw10; w11, varw11; and w12, varw12, are leased by subroutines, and again the register contents are generally NOT valid across subroutines. One exception is a situation where a variable is used specifically to hand off data from one subroutine to another. This allows the receiving subroutine to be built so that it can handle a 'generic' variable input and therefore can be used by many routines. Varw10 for example is used as an input to the Decimal and RoundIt subroutines. So the routine calling Decimal or RoundIt must place data to be processed in varw10.
   Registers b2,b3,b4,b5,b6,b7,b8,b20,b21,b22,b23 are reused by the Interrupt subroutine to write the display.

Program Flow

Start

The program starts by telling the compiler which PICAXE processor is being used, then the internal EEPROM is programmed, and following that symbols are defined.

a. A' #picaxe 20X2' directive is issued to let the compiler know that this program should be compiled for a 20X2
b. The 20X2 internal EEPROM is programmed with battery information For further information on adding or changing battery look-up table information see the BatteryMonitor users guide 'Changing or Adding Battery Data' section.

Power-up initialization

The Initialize program configures BatteryMonitor from a cold power up to get the unit ready for its first measurements. Initialization performs the following tasks:

c. Set the processor to run at 4MHz. Four megahertz minimizes processor power consumption and is the slowest speed capable of performing all calculations during the 360ms measuring timing loop.
d. Configure processor output switches to initial values setting the following outputs high
    i. shdn – keep the low drop out regulator on. This regulator supplies all internal blocks including the processor.
    ii. wpEEPROM – write protect the EEPROM
    iii. BVD – switch on the source voltage monitor
    iv. backlight2 – turn on the display backlight. Initialize it to a medium illumination level
    v. GainLo – set the current to voltage converter gain to low
e. Pause to allow the power supply time to stabilize before initializing the I2C display.
f. Configure the I2C display.
g. Display the start-up splash screen.
h. Write special characters to the display.
i. Configure the processor analog to digital converter inputs. This section also provides code to accommodate a bug in the compiler version C.0.
j. Load configuration data from the external EEPROM.
k. Set the backlight level to that previously stored in EEPROM.
l. Check the values of Tick (loop timing) and Sf (current to voltage converter scale factor) loaded from EEPROM. If they are outside predefined limits use default values.
m. LoadBatteryCap section creates a percent remaining battery capacity vs. battery terminal voltage lookup table using scratchpad memory. In this table the scratchpad address is the battery voltage and the register content is the capacity that will be displayed as a two digit number = xx%. The table is based on data stored in EEPROM. This table uses all 128 bytes of the scratchpad memory. To create the table first all readings for the selected battery are read from the EEPROM and assembled in SFR locations 65-76. Next 53 is added at SFR location 64 as the battery minimum and 179 in location 77 as the battery maximum. Next begin populating the scratchpad using a loop with the addresses driven by the battery voltage along with lookup function to put battery capacities into scratchpad addresses. Now using any voltage between 53 and 179 as a 'get' address will return the remaining battery capacity as a percentage. Because the 20X2 only has a 128byte scratchpad, entries from 128 to 179 will appear in the scratchpad as 0 to 51. Address 52 is equal to 0. Addressing registers greater than 127 will overflow the address counter & but will return the correct battery capacity.
n. Pause to allow time for the splash screen to be read.
o. Check if the unit is waking from hibernation. All data has been restored from external EEPROM to the appropriate registers so there is little to do if the user wishes to resume from hibernation. If the HibernationState value is 0 then the

StartNewReadings subroutine is called and program execution continues. If the HibernationState value is 1 then the unit is returning from hibernation and the display is written with 'Resuming from Hibernation'. The program then pauses so that the user can read the screen then program execution continues.

p. As the processor's GainLo switch was set high define the GainRange as 2 to be consistent.
q. Put the current to voltage converter in the correct measuring range in preparation for starting measurements. This also insures that the first current measured in the measuring loop for display and ILmax are valid. Put the device in the correct measuring range by doing the following:
    i. Execute the AutoGainRange subroutine
    ii. Pause to allow the reading to settle before executing a second AutoGainRange in case the current to voltage converter needs to change two ranges. The gain range reading will settle in well under 100ms
    iii. Execute the AutoGainRange a second time.
    iv. Pause to allow the output of the current to voltage converter to settle if the range changed a second time.

The initialization sequence is now complete and the measuring cycle begins.

Measuring

The Measure routine is BatteryMonitor's core function. Some housekeeping is done before starting the do loop that executes all BatteryMonitor measurements and calculations. The measuring cycle timing is key to BatteryMonitor's ability to accurately collect Amp hour or Watt hour use data. BatteryMonitor is configured to execute 10,000 measurement cycles per hour, so that each measurement cycle takes precisely 360 milliseconds. Each cycle is made up of two segments. The first is an interrupt driven timing loop that is defined by using the Timer and Settimer commands configured so that the loop time is defined by the value of Tick. This time accounts for approximately 91.2% of the total cycle time. The remaining time is used by the interrupt service routine named 'Interrupt'. That routine writes the display, resets the timing counters and rearms the interrupt before returning control back to the measuring do loop. The amount of time spent in the interrupt service routine is well defined as there are no conditional statements in the section.

This measuring section briefly describes all measurement activities. The next section describes each measurement and calculation subroutine in more detail.

a. Define the Timer and preload Settimer with the value Tick so that an interrupt will occur every 360ms. As noted earlier the 360ms period is made up of the timing loop plus the time servicing the interrupt routine.
b. Arm the interrupts using the SetInitFlags command
c. Clear the display
d. Loop executing the following activities. At the same time check for any activity on switches Sw1 and Sw2.
    i. Execute the SourceVoltage subroutine to measure as save the source voltage,
    ii. Execute the LoadCurrent subroutine to measure and save the load current.
    iii. Execute the GainAutoRange subroutine. This routine measures the raw A-D reading from the output of the current to voltage converter amplifier block to assess whether the measurement range needs to change. The range will be changed if the measurement is outside range limits. Each instance of the AutoGainRange subroutine can increment or decrement the gain range through one of the unit's three ranges. This command is executed twice in each measurement loop in case the range needs to be changed twice. This command is located in time, at about half way through a measurement cycle so it affords maximum settling time before the next GainAutoRange subroutine is

executed. The second GainAutoRange subroutine is placed at the end of the measurement cycle before the wait statement that is used to trigger the Settimer interrupt.

iv. Check for a new source minimum voltage measurement. If a new minimum was measured store it.
v. Check for a new load current maximum. If new maximum was just measured store it.
vi. Calculate and save load power.
vii. Update and store the elapsed time.
viii. Estimate the remaining battery capacity using the BattCapLeft subroutine and save the result.
ix. Execute the AmpWattHours subroutine to calculate both the accumulated Amp hours and Watt hours. Format both readings and save the active unit for display.
x. Increment the clock counter.
xi. Execute a second GainAutoRange this measurement cycle. Check the current to voltage converter range a second time in this loop in case the measurement range needs to change twice. Change the range if the current measurement is outside the range limits. This command is located in time at about half way through a measurement cycle so it affords maximum settling time before the next load current measurement.
xii. Pause longer than the interrupt interval to make sure an interrupt occurs during the pause command. This completes one measurement cycle.
xiii. When the interrupt occurs, execute the Interrupt subroutine. The Interrupt subroutine activities:
    1. Write data to the display
    2. Set Timer and Settimer values for the next loop
    3. Re-enable the interrupt flags
    4. Return to the measurement loop at the next command (top of the loop).

e. If either Sw1 or Sw2 is pushed break out of the measuring loop and do the following:
    i. Disable the loop timer and timer interrupt.
    ii. If Sw1 is pushed check for a Hibernation request by determining if the switch is being held closed. If the switch is held closed for more than about two seconds go to the GoHibernate subroutine otherwise the switch was pressed only momentarily so proceed to the Utilities menu.
    iii. If Sw2 is pushed check to determine if it was pushed and held for more than about two seconds. If it was only pushed momentarily then return to the measuring loop. If it was held for more than about two seconds then power the unit off. While in the process of powering off, check to determine if the HibernateState flag is true. If true then execute the GoHibernate routine to update and save Amp hour, Watt hour, and time readings before executing the PowerOff subroutine.

Main Measurement Cycle Subroutines

SourceVoltage

This routine reads the source voltage from the 6:1 switched input voltage divider. It rounds the result to one decimal place, and then invokes the Decimal subroutine to format the reading for the display. The routine then saves the result in the Special Function Register (SFR) for use during the Interrupt routine when the display is written with stored SFR data.

LoadCurrent

This routine reads the current to voltage converter output then scales the raw reading by the scale factor Sf. The scaled reading is then manipulated for display depending on the GainRange and value as either xxxmA, x.xxA or xx.xA. The LoadCurrent routine executes the following functions:

1. Read the A-D converter output of the current to voltage converter and store the reading as ILraw.
2. Scale the reading. Multiply ILraw * Sf. Sf, the scaling factor, is determined by calibration. It accounts for any errors in the value of the sense resistors, the absolute value of the voltage reference, and any error in the current to voltage converter based on the gain in GainRange0 (the highest gain). The scaling routine takes any ILraw between 0 and 999 and multiplies it by any scale factor between 0 and 1.999. Valid scale factors are limited to the 0.900 to 1.100 range (accommodating a +/- 10% total error).
3. Determine IL from ILraw and the GainRange. In GainRange0 the current IL is 1*ILraw or 1mA per bit. In GainRange1 its 10*ILraw or 10mA per bit. And in GainRange2 its 40*ILraw or 40mA per bit.
4. IL is then saved in varw10 and rounded using the RoundIt subroutine. The result then sent to the Decimal routine where it's turned into ASCII characters for direct use with the display.
5. The display is formatted depending on the value of IL. If IL<1000 then display xxxm(A). If IL is greater than 1000 but less than 10,000 then display x.xx(A). And lastly if IL is greater than or equal to 10,000 then display xx.x(A).

GainAutoRange

This routine selects one of three gain ranges for the current sense to voltage converter amplifier to insure load current is accurately measured.

GainRange0 is invoked when PICAXE outputs GainLo=0 and GainMed=0. This is a high gain range having a gain of 1600 and produces a 1mA/bit output that is used with the 0-999mA range. GainRange1 is invoked when GainLo=0 and GainMed=1. This medium gain range has a gain of 160 producing an output of 10mA/bit and is used for the 1.00 to 9.99A range. GainRange2 is invoked when GainLo=1 and GainMed=0. This low gain range has a gain of 40 producing an output of 40mA per bit and is used for the 10.0 to 40.0A range.

The routine reads the output of the current to voltage converter block as ILraw, a straightforward A to D count. From this value and the value of the current GainRange the routine determines whether to keep the gain the same, or invoke the next higher or lower gain. Each pass through this routine can either increment or decrement one gain range, or keep it the same. As there are three gain ranges this routine is invoked twice so that any gain range may be used each measurement cycle. Two gain range changes per measurement cycle enable measurements that change from minimum (0mA) to maximum (40A). In addition the routine changes the GainRange variable as appropriate.

LoadPower

This routine multiplies load current, IL, by the source voltage Vsource to get load power. Load current values range from 0 to 40,920 (40.9A) and depending on the scaling factor could be as much as ten percent higher or 45,012 (45.0A). The source voltage is stored as a three digit number and can range from 53 (5.3V) to 246 (24.6V). To prevent the V*I product from overflowing the processor's 65,535 limit while retaining as much resolution and accuracy as possible, IL is divided into five ranges and each range assumes that the battery voltage can be as high as its 246 maximum.

In the first range IL is less than 267. The maximum power in this range is then 266*246 or 65,436 which is displayed as 6.54 watts. The product is rounded by the RoundIt subroutine. The value is converted to characters that can be directly displayed by the Decimal routine and saved to SFR in the format x.xx watts for use in the display write section of the Interrupt subroutine.

For the next range IL is greater than or equal to 267 and less than 1331. In this range IL is divided by 5 then multiplied with the source voltage. So maximum power in this range is 1330 divided by five times 246 or 65436. To get the value in watts the product is divided again by two to yield a maximum of 32,718 or 32.718 watts. The result is rounded by the RoundIt subroutine then converted to a format that can be directly displayed by the Decimal subroutine. The value saved in SFR depends on the value of Varw10. If it's under 10,000 that is under ten watts then it's saved as x.xx watts so the display shows 10mW resolution up to ten watts. Otherwise it's saved as xx.x watts displaying 100mW resolution for values up to 32.7 watts.
The next range includes values of IL from 1331 to 26,600 and it is split into two sub-ranges. If IL is less than 5321 then power is calculated by first dividing IL by 20, then multiplying it by the source voltage and then dividing the product by five. In this range the maximum power is 5320/20*246/5 or 13,087 which equals 131 watts when rounded. If IL is between 5321 and 26600 then IL is divided by 100 then multiplied by the source voltage. In this case the maximum is 26600/100*246 or 65,346 which equals 635 watts when rounded. As before the product is rounded by the RoundIt subroutine then formatted for the display by the Decimal subroutine. The display is formatted depending on the value of warw10. If it's under 10,000 then the display format is xx.x Watts, otherwise its xxx Watts with the leading zero blanked. Again the results are saved in the SFR for use with the display write section of the Interrupt subroutine.
In the final range IL is greater than 26600. Here IL is divided by 200 multiplied by the source voltage then divided by 5. In this range the maximum power is 45021/200*246/5 or 11,070 or 1,107 Watts. As before the product is rounded by the RoundIt subroutine then converted for display by the Decimal subroutine. The product is then stored in SFR in two formats. If varw10 is less than 10,000 then the number is stored as xxx watts with the leading zero blanked. If varw10 is equal to or over 10,000 then the number is stored as xxxx Watts.

ElapsedTime

This routine calculates then displays the elapsed time in a HH:mm format that can display up to 99 hours and 59 seconds. The elapsed time clock runs only while the unit is measuring. Elapsed time is saved if the unit is placed into hibernation. It can be zeroed using the menu item "Clear Readings' or on returning from hibernation if the user chooses not to resume from hibernation.
Accumulating elapsed time is based on the fact that BatteryMonitor will perform 10,000 measurements per hour, and the clock in incremented once each measuring cycle. If the Clock register is greater than 10,000 then the hours register is incremented and the clock counter is decremented by 10,000. If hours is greater than 99 then the hours register is set to 0. The clock register is multiplied by 6 then divided by 1000 to produce minutes. The total time is calculated by multiplying hours by 100 and adding minutes to produce a result in the form xxxx where the thousand and hundred represent hours while the tens and units represent minutes. The result is then sent to the Decimal routine for conversion to ASCII format that can be directly displayed. The result from Decimal is then saved in SFR for use by the display writing section in the Interrupt routine.

BattCapLeft

This routine displays an estimate of the remaining battery capacity in the format xx%. The value is based on the measured source voltage and a lookup table stored in the scratchpad memory that associates the terminal voltage value with remaining battery capacity. The scratchpad memory is populated with numbers representing remaining battery capacity in a 0,2,5 10,20,30,40,50,60,70,80,90,99% sequence. The percentages are based on data the user can derive from batteries. Battery table data is stored in the 20X2 EEPROM and then translated to percent capacity populating the scratchpad memory using the LoadBattCap subroutine that is invoked at initialization. Note that 99% is used rather than 100% because of display space limitations. Only 128 bytes of scratchpad memory are available, so this limits the voltage range. Values between 52 and 179 representing 5.2 volts to 17.9 volts are stored. Values above 128 (12.8V) wrap

around the scratchpad so that 129 is 0 and 179 is 52. This routine reads a scratchpad location using the Vsource value. The contents of that location are a two digit percentage number. The routine limits the Vsource numbers to between 52 and 179. Numbers below 52 are treated as 00% and those above 179 are 99%. The capacity value is then sent to the Decimal subroutine to be translated to ASCII data that can be used directly by the display. This data is then stored in SFR for use by the display writing section in the Interrupt routine.

AmpWattHours

This routine calculates accumulated battery use in both Amp hours and Watt hours then displays the result in either Amp hours or Watt hours depending on user preference as stored in the AhWh variable (set by using the BM utilities menu). Note that while only one value is displayed, both Amp hours and Watt hours are both continuously calculated. During each measuring cycle battery use accumulation in Amp hours is calculated first followed by Watt hours and then the value is written to the SFR for display during the Interrupt subroutine. In order to optimize accuracy and resolution both Amp hour and Watt hour calculations are broken down into ranges.

1. ***Amp Hour Calculation***

   Amp hours are calculated first. Remember that the reading interval is 10^-4 hrs. The unit accumulation is in milliamps so each cycle accumulates IL*time. The smallest accumulation is 1ma in 10^-4 hrs or 10^-7 Amp hours. This unit is called nAh. So every 10,000nAh is equal to 1mAh and 1000 mA hours equals 1Ah. Variables nAh, mAh and Ah store the accumulated battery use. All values are zeroed on initialization unless resuming from hibernation. The code accumulates nAh by adding the load current in mA to the value of previously accumulated nAh. If nA hours is over 10,000 then the most significant digit is parsed and stored as var1, as at very large currents up to 45,012 nAh could accumulate each measuring cycle. If nAh was over 10,000 the mA hour register is updated adding var1 to the accumulated total and the nAh register decremented appropriately. If the mAh register is 1000 or more the Ah register is incremented and the mAh register is decremented by 1000. If the Ah register is greater than 9,999 then its reset to zero.

2. ***Watt Hour Calculation***

   Watt hours are a bit more complicated to mange because the load current must first be multiplied with the source voltage and then watt hours accumulated each measuring cycle. There are 10,000 measurements per hour so each measurement occurs in 10^-4 hours. The source voltage, Vsource, can range from 53 (5.3V) to 246 (24.6V). IL can range from 0 to 45,012 mA. The base watt hour unit is then 1mA times 0.1V times 10^-4hrs or 10^-8 Watt hours. Throughout this calculation the unit uWh refers to 1*10^-8 Watt hours. mWh unfortunately is not 10^-3 for convenience during this calculation it's 10^-4 or 0.1mWhr. So it takes 10,000 mWhs to make 1Wh. As with the load current in order to optimize accuracy the Watt hour calculation is performed in five load current ranges arranged so that IL*Vsource+accumulated uWh is never more than 65,535. The additional value of accumulated uWh means the limits for Watt hour calculation are a bit different than those used for the Watts calculation. The first range covers the values of IL up to 225. The maximum product is IL*Vsource+uWh or 225*246+9999 or 65,349. Digit four is parsed to variable one as large values for either IL or Vsource will cause results that are over 10,000. Var1 now contains the number of mWh, so that is added to the mWh count and the value of var1 multiplied by 10,000 is subtracted from the uWh value.

   In the second range IL is between 226 and 1290. To handle these values the uWh value is first de-normalized to the value for this range by dividing by five. uWh are then calculated by dividing IL by 5 multiplying by Vsource and adding the existing value of uWh (now actually uWh/5). The maximum possible value in

this range is then IL/5*Vsource+uWh/5 or 1,290/5*246+(9,999/5) or 65,467. The number of additional mWh is calculated by taking the uWh value and dividing it by 2,000 (not 10,000 as the number has already been de-normalized by dividing by 5). The result is then added to the existing value for mWh. The remaining uWh are calculated by taking the uWh value and finding the remainder (uWh//2000) then multiplying the result by 5 to normalize the value for the next measuring cycle.

Range three covers IL values between 1292 and 5286. In this range the load current is divided by 20. So the maximum value is 5,286/20*246+(9,999/20) or 65,516. As with the last range IL is divided by 20 then multiplied by Vsource then the de-normalized value of uWh is added to produce the total. The number of additional mWh is calculated by taking the uWh value and dividing it by 500 then adding the result to the existing value for mWh. The remaining uWh value is calculated by taking the uWh number and finding the remainder in this case uWh//500. The result is then multiplied by 20 to normalize the value for the next measuring cycle.

Range four covers IL values between 5287 and 26,599. In this range the load current is divided by 100. So the maximum value is 26,599/100*246+(9,999/100) or 65,289. As with the last range IL is divided by 100 then multiplied by Vsource then the de-normalized value of uWh is added to produce the total. The number of additional mWh is calculated by taking the uWh value and dividing it by 100 then adding the result to the existing value for mWh. The remaining uWh value is calculated by taking the uWh number and finding the remainder in this case uWh//100. The result is then multiplied by 100 to normalize the value for the next measuring cycle.

Range five covers IL values above 26599. In this range the load current is divided by 200. So the maximum value is 45,012/200*246+(9,999/200) or 55,399. As with the last range IL is divided by 200 then multiplied by Vsource then the de-normalized value of uWh is added to produce the total. The number of additional mWh is calculated by taking the uWh value and dividing it by 50 then adding the result to the existing value for mWh. The remaining uWh value is calculated by taking the uWh number and finding the remainder in this case uWh//50. The result is then multiplied by 200 to normalize the value for the next measuring cycle.

During each measuring cycle the resulting value of the mWh register is checked to determine if it's over 10,000. If it is the Wh register is incremented and the mWh register is decremented by 10,000. If the Wh register is greater than 9,999 then its reset to zero.

3. AhWh Display Formatting

The display will show either Amp hours or Watt hours depending on the state of the AhWh variable that can be set using the BatteryMonitor utilities. If AhWh is zero then Amp hours are displayed if AhWh is 1 then Watt hours is displayed.

a. Displaying Amp hours

Check if Ah register is zero, if so display mAh. Set varw10 to equal the mAh register then call the Decimal subroutine to convert the numbers to ASCII data that can be displayed directly. This data is then stored in SFR in the format xxxmA(h) for use by the display writing section in the Interrupt routine. In all cases the hours, h is part of the display write portion of the Interrupt subroutine so isn't included. If the Ah register is less than 100mAh then the mA register is parsed to determine the hundreds and tens of mAh value. The hundreds value is stored in var1 then multiplied by 10. The tens value is stored in var2. Next the Ah register is multiplied by 100 and var1 and var2 are added get a result that includes tenths and hundredths of an Ah. The value is stored in varw10. The result is processed though the Decimal subroutine to convert the result to ASCII data that can be directly displayed. The data

is then stored in SFR for use by the display writing section of the interrupt routine in one of two formats. If warw10 is less than 10,000 then the display format is x.xxA(h) and if over 10,000 the display format is xx.xA(h). If the Ah register is greater than 100 then the Ah number saved in warw10, and the Decimal subroutine is called to convert the data to ASCII format that can be used directly by the display. If varw10 is less than 1,000 then the data is written to the SFR to be read by the display portion of the Interrupt routine in the format xxxA(h) with the leading zero blanked. If varw10 is over 10,000 then the result is stored as xxxxA

b. Displaying Watt hours

If the Wh register is zero then contents of the mWh register is used to display the battery use in the form xxxmW (in all cases the hours, h is part of the display write portion of the Interrupt subroutine so isn't included). Varw10 is loaded with the contents of the mW register, the Decimal subroutine is called to format the contents for display and the data is saved in SFR memory for use in the display write portion of the Interrupt subroutine. If the Wh register isn't zero then the display is formatted depending on its value. If it's under 100 then a value for Wh that includes hundredths and or tenths of a Watt hour is calculated and the display format is either x.xxW(h) or xx.xW(h). To do this the mW register is parsed for the value of the thousands digit which is then multiplied by 10 and saved in var1. The value of the hundreds digit is saved in var2. The value of the Wh register is multiplied by 100 and var1 and var2 added to the product to yield a result that includes tenths and hundredths of a watt hour. The result is sent to the Decimal subroutine for conversion to ASCII format that can be directly displayed. If the result is less than 10,000 then the display is stored in SFR as x.xxW(h). If the result is greater than 10,000 then the result is saved in SFR in the format xx.xW(h) This SFR data will be used by the display write portion of the Interrupt subroutine. If the Wh register is over 100 then the warw10 is set equal to the value of the Wh register and the value is sent to the Decimal routine for conversion to ASCII format that can be used directly by the display. If the Wh value is less than 1000 then SFR is written to produce the format xxxW with the leading zero suppressed. If it's over 1000 the SFR is written to produce the format xxxxW.

Interrupt

The interrupt routine is invoked each time the timer function overflows and sets the timer interrupt flag. Settimer command controls the duration and its value is preloaded so that each measuring cycle lasts 360ms. When an interrupt occurs program execution vectors to this routine, the routine is then executed and control returns back to the main program resuming at the next instruction. By design the interrupt will occur each measuring cycle at the bottom of the cycle during the last wait command. The Interrupt routine begins by taking data that was stored by each subroutine in the last measuring cycle and placing that data into system variable registers. The contents of the system variable registers are then written to the display. Values for the timer and settimer registers are written followed by the SetInitFlags registers which re-enable the interrupt. The program then returns to the measurement loop at the next command at the top of the loop.

LoadBatteryCap

The LoadBatteryCap subroutine creates a percent remaining battery capacity vs. battery terminal voltage lookup table using scratchpad memory. In this table the scratchpad address is the battery voltage (source voltage) and the register content is the capacity that will be displayed as a two digit number xx%. Valid battery voltages range from 52 (5.2V) to 179 (17.9V). The table is based on data stored for any one of eight batteries in EEPROM. This lookup table uses all 128 bytes of the scratchpad memory. Scratchpad

address 52 is equal to 0. Because the 20X2 only has a 128byte memory entries from 128 to 179 will appear in the scratchpad as 0 to 51. Addressing registers greater than 127 will overflow the address counter & but it will return the correct battery capacity. To create the table data for the selected battery is read from the EEPROM and assembled in SFR locations 65-76. Next 53 is added at SFR location 64 as the battery minimum and 179 in location 77 as the battery maximum. Then the process then begins populating the scratchpad using a for next loop to and a lookup function that puts battery capacities into scratchpad registers. Scratchpad addresses are based on the data assembled in SFR addresses 64 to 77 and register content by the value of the lookup function. Now using any voltage between 53 and 179 as a 'get' address will return the remaining battery capacity as a percentage.

Utilities Menu Items

BatteryMonitor has eight utility menu items that are presented to the user in a continuous loop. Line one of the display is used a header to let the user know that they are in the utilities menu and line two presents the user with one of eight menu options in the order listed below. When in the utilities menu, pressing Sw1 scrolls through the menu items from items a to h then repeats by cycling back to a. Pressing Sw2 selects the menu item, the appropriate subroutine is executed the program exits the utilities menu and returns to the measurement system.

a. MinAndPeak
b. ClearReadings
c. ChooseAhWh
d. SetBacklight
e. SelectBattery
f. GoHibernate
g. PwrOff
h. Calibration

MinAndPeak

The measuring system continuously records the minimum source voltage and peak load current. This routine allows the user to view this data and optionally clear the readings. The routine begins by clearing the display then writes the first line with Clr Vmin & Ipk? It then recalls the VsourceMin and ILmax register values and converts them to ASCII for display using the Decimal subroutine. The ILmax value is then evaluated to determine which of three display formats to use. If ILmax is greater than 10,000 then, xx.xAp is displayed, if ILmax is less than 1000 then xxxmAp is displayed, otherwise x.xxAp is displayed. The minimum battery voltage and peak current are then written to the display. A leading left arrow and 'N' is also written to line 1 and a leading left arrow and 'Y' is written to the display's second line. The' WaitForSwitch' subroutine is then called and the unit waits for the user to press either Sw1 or Sw2. If Sw1 is pressed the minimum voltage and peak current registers are reset. If Sw2 is pressed the minimum voltage and peak current registers are left unaltered. The subroutine then ends and program execution returns to the measuring subsystem.

ClearReadings

This display is cleared then written with options to Restart measurements (Sw1) or return to the measurement system (Sw2) without clearing readings. The WaitForSwitch subroutine is then called to do just that, wait for a user to press one of the two switches. If Sw1 is pressed then the process of clearing data begins. If the HibernateState variable is 1 it is set to 0 and the new value stored in EEPROM. Otherwise if the HibernateState was 0 the EEPROM write is bypassed. The RestartReadings subroutine is then called to clear the clock, Ah and Wh registers. The program then returns to the measuring subsystem. If Sw2 is pressed then the program pauses to let the user get their finger off the button and the program returns to the measuring system.

ChooseAhWh

This routine sets the AhWh variable and saves it in external EEPROM.
This display is cleared then written with options to set the display format to Ah, Amp hours located by (Sw1), or Wh, Watt hours located by (Sw2). The 'WaitForSwitch' subroutine is then called and the unit waits for the user to press either Sw1 or Sw2. If Sw1 is pushed to select Amp hours the AhWh variable is set to 0. If Sw2 is pushed selecting Watt hours the AhWh variable is set to 1. The value is then written to external EEPROM.

SetBacklight

Set Backlight selects one of four display backlight levels, off, low, medium, high. This value is saved to external EEPROM.
Line one of the display is written with the down arrow then 'Next BL Level' and line 2 with a left arrow then Backlight followed by the level as contained in the backlight register. Next the SetBacklightLevel subroutine is called to set the backlight level. The WaitForSwitch subroutine is then called. If Sw1 is pressed the Backlight value is incremented and the new value of backlight is displayed and backlight drives set accordingly. If Sw2 is pushed the present value of Backlight is saved to external EEPROM then the program returns to the measuring system.

SelectBattery

The 20X2 EEPROM can store up to eight different battery characteristic data sets. This data provides battery terminal voltage versus percent remaining capacity information. This routine enables the user to select one of the batteries. It then saves the selected battery number (0-7) in the Battery register as well as external EEPROM. The routine then calls the LoadBatteryCap subroutine to populate the scratchpad memory with a battery terminal voltage versus percent capacity look-up table. At the start of this subroutine line one of the display is written with the down arrow and Next battery. The Battery pointer is then determined from the Battery register content and is used to write the second line which consists of a left arrow symbol followed by the battery description section as stored in the EEPROM. The Waif for switch subroutine is called. If Sw1 is pushed then the next battery is to be selected. To do this the Battery register is incremented. If Battery is greater than 7 then its set to zero as there are only seven batteries available. Next the value of the BatteryPointer determined and the first two characters from the battery description read. If they are both $ff then the Battery register value is zeroed as the $ff is used as a flag to indicate the last battery data set. Program execution is then vectored to the start of SelectBattery so that the next battery info can be displayed. If Sw2 is pushed then the current battery is selected. The value of Battery is saved in external EEPROM after which the LoadBatteryCap subroutine is executed to load the new battery look-up table values into the scratchpad memory. The program then returns to the measuring system.

GoHibernate

This routine sets the HibernateState register to one. It then saves the following registers to external EEPROM: HibernateState, Hours, Time, nAh, mAh, Ah, uWh mWh and Wh. The routine then calls the PwrOff subroutine which will turn the unit off.

PwrOff

This routine turns the unit off. If HibernateState is one then line one of the display is written with Hibernating, otherwise Bye and the second display line is written with Bye. A two second pause is started to allow the user to stop pressing the push button. The shutdown pin shdn is then set low and a four second pause executed to ensure the unit is powered down completely. As a failsafe in case for some reason the unit isn't powered down (a fault or the user may have kept their finger on the pushbutton, the shutdown pin is commanded high so that Sw1 remains operative.

Calibration

At start this subroutine provides the user with two choices written to the display, Sf Set or Time-Tick Cal. The scale factor, Sf, modifies load current measurements correcting for errors in reference voltage, shunt resistor value and current to voltage converter gain based on the gain of GainRange0. Note that this routine does not calculate the scale factor; it only enables the user to enter and save a calculated scale factor. The Time-Tick cal routine adjusts the measuring cycle to more precisely set it to 10,000 measurements per hour. This makes the battery use (Amp hours or Watt hours) measurements more precise. For information how to use the ScaleFactorSet or Time-Tick routines refer to the user manual.
After the initial display the subroutine calls the WaitForSwitch subroutine. If Sw1 is pressed the program executes the ScaleFactorSet routine otherwise it begins the Time-Tick cal section.

Time-Tick Cal

This section begins by getting the user to enter the elapsed time from the stop watch in hours, minutes, and seconds. The program limits the maximum stop watch time entered to 8:59:59. Once seconds are entered the routine will then display the time and ask if it's correct using Sw1 for 'Y' and Sw2 for 'N'. If Sw2 is pushed then the program goes back to the GetHours routine. If Sw1 is pushed the routine begins to calculate a new value for Tick. To do this both the internal time and the stopwatch time are converted to seconds. The stopwatch time is then subtracted from the internal time. If the difference is greater than 32,768 the number is negative so the internal time is subtracted from the stopwatch time and a flag is set to indicate the negative value. Each tick shifts the internal clock by 0.64 seconds per hour so the next task calculates the number of whole ticks to change. This is done by calculating the difference between the two clocks in seconds per hour then multiplying by 1.563 (1/0.64). To do this while optimizing error correcting range and accuracy, the difference in seconds between the two clocks is multiplied by 15.63. The result is then divided by stopwatch time in seconds divided by 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% so errors as large as 10% should not occur. The result in number of ticks is then added or subtracted from the current Tick register value depending the value of var3, the negative flag. Line one of the display is then written with 'Y Save New Tick?' and both the old and new values of Tick are displayed on line two preceded by an 'N'. If Sw1 is pressed then the new tick value is saved in EEPROM and the program returns to the measuring routine. If Sw2 is pressed the old value of Tick is kept and the program returns to the measuring routine.

ScaleFactorSet

The temporary register Varw10 is loaded with the current value of Sf. It's then converted to ASCII using the Decimal subroutine then displayed with a leading down arrow on line two along with a leading up arrow and Chg SF? on line 1. The WaitForSwitch subroutine is then called. If Sw1 was pressed then the scale factor is incremented, if Sw2 was pressed the scale factor is decremented. If both switches are pressed then the scale factor value is saved to external EEPROM and the program returns to the measuring routine.

Other Subroutines

WaitForSwitch

This routine is designed to wait for a pushbutton switch closure. Sw1 and Sw2 are normally open push button switches. This routine contains a timeout timer that is used to exit the utilities menu if neither of the two pushbutton switches is pushed

within about 20 seconds. The battery use (Ah,Wh), and session time doesn't change when the utilities menu is invoked. So the WatiForSwitchTimeout is intend to limit time spent in the utilities menu if Sw1 is mistakenly pushed or if the user is distracted while the device is in the utilities menu. The WaitForSwitchTimeout registry is cleared. The routine then loops while Sw1 and Sw2 is zero (open) and WaitForSwitchTimout is under 200. A 100ms wait (2X50ms) is used both as part of the timeout period as well as a switch debounce. The WaitForSwitchTimeout register is incremented and the loop repeats.

StartNewReadings

This routine clears the following registers; clock, Hours, nAh, mAh, Ah, uWh, mWh, Wh, VsourceMin, and ILmax.

SetBacklightLevel

This subroutine sets the display backlight to one of four levels by controlling the Backlight 1 and Backlight 2 pins (C.5 and C.7). The display backlight level is defined by the Backlight register to be one of four levels, off(0), low(1), medium(2), or high(3).

RoundIt

RoundIt is used to round up measurement results or calculations. The routine is used primarily to prepare data for display. The routine assumes three digits will be displayed in the form xxx, xx.x or x.xx. A number to be rounded is provided to this routine using the generic variable varw10 and the rounded up number is returned using varw10.

If varw10 is under 1000, then all three digits will be displayed so there is really nothing to round. In this case the rounding routine is bypassed. If the result is over 1000 but less than 10000 then the least significant digit, digit 0, is parsed and if its value is over 4 the number is rounded up by increasing varw10 by 10. If the result is over 10,000 then the second digit in the number, digit 1, is parsed and if its value is over 4 then the number is rounded up by increasing varw10 by 100.

Decimal

Decimal takes a value stored in varw10 and parses the unit, ten, hundred, thousand, and tenthousand values turning them into ASCII characters that can be used directly with the LCD display.

EEPROM

Battery Data Storage

Data for up to eight different batteries is stored in the PICAXE 20X2 EEPROM. A comment line is used to indicate the battery number, zero through seven. This is followed by two lines of data. The first line contains fifteen characters between the quotes that describe the battery. This information is used by the 'SelectBattery' utility menu to describe the battery to the user. The second line contains twelve voltage entries representing 2,5,10.20,30,40,50,60,70,80,90, and 99% battery capacity. Each entry is three digits representing the battery terminal voltage so xxx represents xx.xV at the respective battery capacity. If the terminal voltage is under 10.0 volts the leading zero must be populated. For example 5.6 volts would appear as 056. The data starts with 2% battery capacity and ends with 99% battery.

The following lines show a battery data entry.

```
'Battery0
EEPROM $00,("6 Cell LeadAcid")
EEPROM $10,(105,109,111,113,115,118,119,121,122,123,125,126)
```

If all eight battery entries aren't used then the first two bytes of the first unused entry must contain $ff to act as an end of data marker as shown in the following example

```
'Battery6
EEPROM $c0,($ff,$ff) 'marker so menu cycles back to first entry
EEPROM $d0,(0)
```