


Battery Management for 12V System

[caravan_bat_mon.pdf](#)

This module is part of the original MMBasic library. It is reproduced here with kind permission of Hugh Buckle and Geoff Graham. Be aware it may reference functionality which has changed or is deprecated in the latest versions of MMBasic.

' Battery Management Program for Caravan 12V System ' Copyright Doug Pankhurst May 2013 '
Written for MMBasic V4.3a, copyright Geoff Graham 2012 ' '
' File Version: 2.04 May 2013 '

This is an updated (Mark II ) version of a program first submitted in May 2012. Lots of comments and it only JUST fits into a CMM running in 8 colour mode - I have intentionally made it verbose and not optimised to allow easy reading.

This program monitors a 12V Battery System, monitoring the battery voltage, the solar panel voltage and the amount of charge supplied into and out of the battery via current shunt ICs. The charge/discharge state of the battery is given by comparing current in versus current out.

It also estimates time to 30% capacity based on battery voltage and current discharge rate and provides a warning.

Real time data is displayed on VGA display as well as historical data recorded to a log file on the SD card for later analysis.

It utilises a range of subroutines and defined functions based on software and ideas from Doug Pankhurst, Geoff Graham, Tom Pankhurst, and also from programs provided in the MMBASIC Software library, authors unknown. This code is free for anyone to use or modify as long as they acknowledge it's source and credit the contributors above.

Physical configuration: Uses a CG Colourmax 2 as the hardware platform and an Arduino blank shield populated with voltage dividers for battery and solar voltages and Allegro ACS758 current sensors for input (charging) current and load current. Battery charge/discharge current is calculated by subtracting load current from charging current.

The 11.8V to 14.4V from the caravan battery system (depending on whether charging or driving load) is fed into an LM317 variable voltage regulator to provide regulated 8.5V to the CGCMM.

WARNING: The Allegro current sensors have VERY fragile leads so they need to be physically stabilised. The units I used were unidirectional and had an offset around 600mV and a 60mV/amp capacity.

A general synopsis of the program:

- Disable interrupts;
- Define and initialise variables and constants;
- Initialise display with static data; initialize 1 minute arrays to allow averaging;
- Enter main loop; wait for interrupt (which sets NewData flag) then handle 2 second, 1 minute, 1 hour and end of day (6AM) occurrences;

- Start main loop, waiting for (set by interrupt routine);
 - Cycle through 2 second, 1 minute, 1 hour and End-of-Day processing with initial data and display initial readings
1. 2 second routine processes all instantaneous readings from interrupt routines - battery voltage, solar array voltage, input current from both solar controller and/or 240VAC battery charger
 2. 1 minute routine averages 30 two second readings, then pushes the averages into an array, also writes out running graph
 3. 1 hour routine takes array data and writes out to files in .CSV format
 4. EOD (at 6AM) closes log file and creates a new log file;
- Interrupt provides battery volts, solar volts, input current and load current.

Global Variables: These are here just for explanatory purposes - they are defined in the program proper. The initialised values are to simplify averaging and also represent typical values.

BVolts : Battery Voltage read from Pin A0. Read as 1/6th of real voltage via 6:1 resistive divider.

BVArray(29) : Battery Volts 30 element array - 1 each 2 secs.

BatCharge : Battery state of charge as a percentage where 12.75V or higher is 100%, down to 12.05V, being 35% - danger level.

AverageBV : Average battery voltage over last 60 seconds.

BVAArray(59) : 1 hours worth of minute average battery volts.

DeltaBV : Battery voltage variation over 60 seconds. The difference between the first and the last used to calculate the "Time to 30% discharge point".

Time2Discharge : Time to 30% discharge.

SVolts : Solar Array voltage from Pin A1. Read as 1/10th of real voltage by 10:1 resistive divider.

SVArray(29) : Solar Volts 30 element array - 1 each 2 secs.

AverageSV : Average Solar Panel volts over last 60 seconds.

SVAArray(59) : 1 hours worth of minute average solar volts.

InputCurrent : Charging current into battery/load read from Pin A2 2.5V reading = 50mA across the shunt, which = 50A through shunt. This can be both battery charging current and load current.

ICArray(29) : Input Current 30 element array.

AverageIC : Average input (generator) current over last 60 seconds.

ICAArray(59) : array with 1 hours worth of minute average input currents.

LoadCurrent : Load current of caravan - can be supplied from the batteries and the charging systems (solar, AC Mains or generator). Pin A3.

LCArray(29) : Load Current 60 element array.

AverageLC : Average load current over last 60 seconds.

LCAArray(59) : array to hold 1 hours worth of average load currents.

BatCurrent : A calculated value; if positive, = a charging current to the battery; if negative, = a discharge current from the battery.

AverageBC : Average charge/discharge current over last min calculated by AverageLC-AverageIC.

InputCurrentAH : Input current as an amp hour calculated value - this is an accumulating value over 24 hours.

PreviousICAH : yesterdays Input Current AHs.

LoadCurrentAH : Load current as an amp hour calculated value - this is an accumulating value over 24 hours.

PreviousLCAH : yesterdays Load Current AHs.

BatCurrentAH : Bat charge (+value) or discharge (-value) as an amp hour calculated value - accumulating value over 24 hours.

PreviousBCAH : yesterdays Bat Current AHs.

LogArray\$: String to use for plugging into log array

NewDataFlag : Two second flag - set every 2 seconds by the interrupt routine when new voltage and current values are collected.

MinFlag : Minute flag - set once every min at 00 seconds by sec interrupt process and cleared by min processing code.

HourFlag : Hour Flag - set on the hour to enable writing the 60 average readings in the log array to be written to file

EODFlag : End of Day Flag - set at 6AM to enable daily log file to be closed and new log file created by LogFileCreate subroutine

CVBM204.BAS:

```
.....  
' Battery Management Program for Caravan 12V System  
' Copyright Doug Pankhurst May 2012
```

```
' Written for MMBasic V4.3a, copyright Geoff Graham 2012 '
' .....
' File Version:      2.04 May 2013
'
' For a detailed explanation of the program, it's
' construction and rationale, please read the
' CVBMREAD.TXT file.

' Main Entry to Program

EntryPoint:  SetTick 0,0      ' Disable timer interrupt

' Global Variables - initialised to default values
BVolts = 12.8      ' Battery Voltage read from Pin A0. Read as
' 1/6th of real voltage via 6:1 resistive divider

Dim BVArray(29)    ' Battary Volts 30 element array - 1 each 2 secs

BatCharge = 100    ' Battery state of charge as a percentage where
' 12.75V or higher is 100%, down to 12.05V being 35% - danger level

AverageBV=12.8     ' Average battery voltage over last 60 seconds

Dim BVAArry(59)    ' 1 hours worth of minute average battery volts

DeltaBV = 0.1      ' Battery voltage variation over 60 seconds. The
' difference between the first and the last used to
' calculate the "Time to 30% discharge point"

Time2Discharge = 20 ' Time to 30% discharge

SVolts = 20        ' Solar Array voltage from Pin A1. Read as 1/10th
' of real voltage by 10:1 resistive divider

Dim SVArray(29)    ' Solar Volts 30 element array - 1 each 2 secs

AverageSV = 21     ' Average Solar Panel volts over last 60 seconds

Dim SVAArray(59)   ' 1 hours worth of minute average solar volts

InputCurrent = 18   ' Charging current into battery/load read from Pin A2
' 2.5V reading = 50mA across the shunt, which equals 50A through the shunt.
' This can be both battery charging current and load current.

Dim ICArry(29)     ' Input Current 30 element array

AverageIC = 18     ' Average input (generator) current over last 60 seconds

Dim ICAArray(59)   ' array with 1 hours worth of minute average input currents
```

```

LoadCurrent = 12 ' Load current of caravan - can be supplied from the
' batteries and the charging systems (solar, AC Mains or generator). Pin A3

Dim LCArray(29) ' Load Current 60 element array

AverageLC = 12 ' Average load current over last 60 seconds

Dim LCAArray(59) ' array to hold 1 hours worth of average load currents

BatCurrent = 6 ' A calculated value; if positive, = a charging current to
' the battery; if negative, equals a discharge current from the battery.

AverageBC = 6 ' Average charge/discharge current over last min
' calculated by AverageLC-AverageIC

InputCurrentAH = 0 ' Input current as an amp hour calculated value -
' this is an accumulating value over 24 hours

PreviousICAH = 0 ' yesterdays Input Current AHs

LoadCurrentAH = 0 ' Load current as an amp hour calculated value -
' this is an accumulating value over 24 hours

PreviousLCAH = 0 ' yesterdays Load Current AHs

BatCurrentAH = 0 ' Bat charge (pos value) or discharge (neg value)
' as an amp hour calculated value - accumulating value over 24 hours

PreviousBCAH = 5 ' yesterdays Bat Current AHs

LogArray$ = "" ' String to use for plugging into log array

NewDataFlag = 0 ' Two second flag - set every 2 seconds by the
' interrupt routine when new voltage and current values are collected.

MinFlag = 0 ' Minute flag - set once every min at 00 seconds
' by sec interrupt process and cleared by min processing code.

HourFlag = 0 ' Hour Flag - set on the hour to enable
' writing the 60 average readings in the log array to be writtent to file

EODFlag = 0 ' End of Day Flag - set at 6AM to enable daily log file
' to be closed and new log file created by LogFileCreate subroutine

.....
' Carry out initialisation routines here - enable interrupt last

Entry:
Mode 3,7 ' 8 colour mode white on black
Colour White,Black
Cls

```

```
' Display all static data
Option USB Off      ' only to VGA
DisplayStatic
BarGraph
' Create Log File on program entry and then at start of new day - 6AM
CreateLogFile

' Set up all initial values to default for orderly entry.
PreviousICAH = 50
PreviousLCAH = 45
PreviousBCAH = 5
OldABV = 13
LFAPointer = 0

' Fill the arrays
For Ax=0 To 29
    BVArray(Ax) = 12.8
    SVArray(Ax) = 19
    ICAArray(Ax) = 18
    LCArray(Ax) = 12
Next Ax

' Wait until 00 seconds to start for clean entry.
Do While Val(Mid$(Time$,7,2)) <> 58
Loop

' Enable timer tick interrupts to start data collection at 00 seconds
SetTick 2000,GetData  ' Every other second, get battery voltage and current
' data through the GetData interrupt routine.

' end of initialisation routines
.....

' Main program loop

Main:
Do While NewDataFlag = 0  ' wait until interrupt provides fresh data
Loop

.....

' 2 second processing code
' Write out time and date to display
Print @(1,1)Time$ + "    " + Date$

' Push most recent values into arrays for minute averaging
For Ax = 29 To 1 Step -1
    BVArray(Ax) = BVArray(Ax-1)
    SVArray(Ax) = SVArray(Ax-1)
    ICAArray(Ax) = ICAArray(Ax-1)
    LCArray(Ax) = LCArray(Ax-1)
Next Ax
```

```
BVArray(0) = BVolts
SVArray(0) = SVolts
ICArray(0) = InputCurrent
LCArray(0) = LoadCurrent
' Charge/Discharge Current is InputCurrent-LoadCurrent - if positive
' it is a charging current into battery, if negative, it is a
' discharging current out of battery
BatCurrent = InputCurrent-LoadCurrent

' Convert both input, output & battery instantaneous currents into Amp Hours
InputCurrentAH = InputCurrentAH + (InputCurrent/1800) ' use 2 sec interrupt
LoadCurrentAH = LoadCurrentAH + (LoadCurrent/1800)    ' rate for calculation
BatCurrentAH = InputCurrentAH-LoadCurrentAH

' Note: All AH accumulating totals are plugged into last day values
' then reset to zero at 6AM - now update terminal display with all values
Font #2
Colour 2
Print @(372,25)Format$(BatCharge,"%3.0f")+""
Font #1
Colour 7
Print @(123,45)Format$(BVolts,"%5.2fV")
Print @(123,56)Format$(SVolts,"%5.2fV")
Print @(123,67)Format$(InputCurrent,"%5.2fA")
Print @(123,78)Format$(LoadCurrent,"%5.2fA")
Print @(312,45)Format$(AverageBV,"%5.2fV")
Print @(312,56)Format$(AverageSV,"%5.2fV")
Font #2
Colour 7
Print @(420,184)Format$(AverageBV,"%4.1f")
Colour 3
Print @(420,232)Format$(AverageSV,"%4.1f")
Colour 7
Font #1
Print @(376,149)Format$(Time2Discharge,"%4.0f min")
Print @(312,67)Format$(AverageIC,"%5.1fA")
Print @(312,78)Format$(AverageLC,"%5.1fA")
Colour 6
Font #2
Print @(420,285)Format$(AverageIC,"%4.1f")
Colour 5
Print @(420,340)Format$(AverageLC,"%4.1f")
Colour 7
Font #1
Print @(117,127)Format$(InputCurrentAH,"%5.1fahrs")
Print @(117,138)Format$(LoadCurrentAH,"%5.1fahrs")
If (Sgn(BatCurrentAH) = +1) Then
    Colour 2
    Print @(117,149)Format$(BatCurrentAH,"%+5.1fahrs")
Else
    Colour 4
```

```
Print @(117,149)Format$(BatCurrentAH,"%+5.1fahrs")
EndIf
Colour 7
Print @(312,127)Format$(PreviousICAH,"%5.1fahrs")
Print @(312,138)Format$(PreviousLCAH,"%5.1fahrs")
If (Sgn(PreviousBCAH) = +1) Then
    Colour 2
    Print @(312,149)Format$(PreviousBCAH,"%+5.1fahrs")
Else
    Colour 4
    Print @(312,149)Format$(PreviousBCAH,"%+5.1fahrs")
EndIf
Colour 7
If (Sgn(AverageBC) = +1) Then
    Colour 2
    Print @(312,89)Format$(AverageBC,"%+5.1fA")
    Font #2
    Print @(418,410)Format$(AverageBC,"%+4.1f")
    Font #1
Else
    Colour 4
    Print @(312,89)Format$(AverageBC,"%+5.1fA")
    Font #2
    Print @(418,410)Format$(AverageBC,"%+4.1f")
    Font #1
EndIf
If (Sgn(BatCurrent) = +1) Then
    Colour 2
    Print @(123,89)Format$(BatCurrent,"%+5.2fA")
Else
    Colour 4
    Print @(123,89)Format$(BatCurrent,"%+5.2fA")
EndIf
Colour 7
NewDataFlag = 0      ' clear new data flag ready for next interrupt
'
' End of two second processing code, now check for minute, hour or end of
' day flags and process accordingly
' .....
' Check for minute flag, calculate the average of the last 60 seconds of
' readings, plug them into the log file array at the log file array pointer
If MinFlag = 1 Then
    ' calculate average for last minute
    OldABV = AverageBV
    OldASV = AverageSV
    AverageBV = 0
    AverageSV = 0
    AverageIC = 0
    AverageLC = 0
    For Ax = 0 To 29    'don't forget only sampled every 2 secs
```



```

AverageBV = AverageBV + BVArray(Ax)
AverageSV = AverageSV + SVArray(Ax)
AverageIC = AverageIC + ICArray(Ax)
AverageLC = AverageLC + LCArray(Ax)
Next Ax
AverageBV = AverageBV/30
AverageSV = AverageSV/30
AverageIC = AverageIC/30
AverageLC = AverageLC/30
AverageBC = AverageIC - AverageLC
' Calculate battery change as percentage and fill charge state graph
' now calculate charge and paint capacity graph
' 12.1V equates to 30% charge
BatCharge = Int((AverageBV - 11.75) * 100)
If BatCharge < 30 Then BatCharge = 30
If BatCharge > 100 Then BatCharge = 100
FillBarGraph(BatCharge)

' Calculate time to 30% discharge of battery
DeltaBV = OldABV - AverageBV
If Sgn(DeltaBV) = +1 Then
    DVRate = BVArray(0)          ' set initial values for rate of
    Time2Discharge = 0           ' discharge and time to discharge
    Do While DVRate > 12.1        ' 12.1 volts equates to 30% discharge
        DVRate = DVRate - DeltaBV ' loop through calculating the amount
                                   ' of time to get the current voltage
                                   ' down to the 30% discharge point
    Time2Discharge = Time2Discharge + 1 ' inc minutes to discharge count
Loop
Else
    Print @(376,149)"Charging "   ' if current voltage is greater than
EndIf                             ' previous minute
BVArray(LFAPointer) = AverageBV ' update average arrays
SVArray(LFAPointer) = AverageSV
ICArray(LFAPointer) = AverageIC
LCArray(LFAPointer) = AverageLC
LFAPointer = LFAPointer + 1      ' LFA pointer reset by eod routine

' Now paint out running graph
' - first work out the current minute hour combo in minutes
rg_mins = (((Val(Mid$(Time$,1,2)) Mod 6)*60)+(Val(Mid$(Time$,4,2)))
rg_mins = rg_mins + 30           ' offset to start of graph
rg_abv = 412 - Int((AverageBV-8) * 10)
If rg_abv > 412 Then rg_abv = 412
rg_asv = 412 - Int((AverageSV-8) * 10)
If rg_asv > 412 Then rg_asv = 412
If rg_asv < 163 Then rg_asv = 163
rg_aic = 412 - Int(AverageIC * 5)
rg_alc = 412 - Int(AverageLC * 5)
If (Sgn(AverageBC) = +1) Then
    rg_abc = 412 - Int(AverageBC * 5)

```

```
Else
    rg_abc = 412 + Int(AverageBC * 5)
EndIf
For yrg = 163 To 412 Step 1
    If rg_abc = yrg Then
        If (Sgn(AverageBC) = +1) Then
            Pixel(rg_mins,yrg) = 2
        Else
            Pixel(rg_mins,yrg) = 4
        EndIf
    Else
        Pixel(rg_mins,yrg) = 0
    EndIf
    If rg_abv = yrg Then
        Pixel(rg_mins,yrg) = 7
    EndIf
    If rg_asv = yrg Then
        Pixel(rg_mins,yrg) = 3
    EndIf
    If rg_aic = yrg Then
        Pixel(rg_mins,yrg) = 6
    EndIf
    If rg_alc = yrg Then
        Pixel(rg_mins,yrg) = 5
    EndIf
    Line(rg_mins+1,yrg)-(rg_mins+3,yrg),0
Next yrg

MinFlag = 0    ' reset the min flag
EndIf
' End of minute code
.....

' Check for hour flag - if true then append log array out to file
' (we want to limit log file writes to 24 per day to keep
' SD card writes as low as possible).

If HourFlag = 1 Then    ' set each hour in interrupt routine
    LogArray$ = ""
    For x=0 To 59 Step 1 ' get the average values from each average array
        LogArray$ = LogArray$+Format$(BVAArray(x),"%+6.2fV")+Chr$(44)
        LogArray$ = LogArray$+Format$(SVAArray(x),"%+6.2fV")+Chr$(44)
        LogArray$ = LogArray$+Format$(ICAArray(x),"%+5.1fA")+Chr$(44)
        LogArray$ = LogArray$+Format$(LCAArray(x),"%+5.1fA")
        Print #1,LogArray$
        LogArray$ = ""
    Next x
    LFAPointer = 0    ' reset array pointer
    HourFlag = 0      ' clear hour flag until next hour
EndIf
' End of hour code
```

```

' Check for end of day - if true then close log file and create new file
' for new day starting at 6AM
If EODFlag = 1 Then
    Close #1          ' close out last log file, save amp hour data
    PreviousICAH = InputCurrentAH
    InputCurrentAH = 0
    PreviousLCAH = LoadCurrentAH
    LoadCurrentAH = 0
    PreviousBCAH = BatCurrentAH
    BatCurrentAH = 0
    CreateLogFile    ' create new log file with todays date
    EODFlag = 0      ' clear end of day flag until next 6AM
EndIf
' End of day code
' ..

GoTo Main          ' back to Main ready for next interrupt

' End of main program loop
' ..

' Interrupt Routine Handling

GetData:           ' Timer Interrupt handler
                  ' Arduino analogue input pins
    SetPin a0,1    ' pin 35 - battery voltage - 0 to 2.5V = 0 to 15V
    SetPin a1,1    ' pin 36 - solar volts - 0 to 2.5V = 0 to 25V
    SetPin a2,1    ' pin 37 - input current shunt - 0 to 2.5V = 0 to 50A
    SetPin a3,1    ' pin 38 - load current shunt - 0 to 2.5V = 0 to 50A

' Get the battery voltage
ic=0
BVolts = 0
GetPin35 = 0
For ic = 1 To 20
    GetPin35 = Pin(a0)
    BVolts = BVolts + GetPin35
Next ic
BVolts = BVolts/20    ' Average out Battery voltage
BVolts = BVolts*6     ' 15 battery volts = 2.5 MM volts

' Get the Solar Array voltage
ic=0
SVolts=0
GetPin36 = 0
For ic = 1 To 20
    GetPin36 = Pin(a1)
    SVolts = SVolts + GetPin36
Next ic
SVolts = SVolts/20    ' same as for BVolts
SVolts = SVolts*10    ' 25 solar volts = 2.5 MM volts

```

```
' Get the input current as a voltage
InputCurrent = 0
GetPin37 = 0
For ic = 1 To 20
    GetPin37 = Pin(a2) - 0.5817
    InputCurrent = InputCurrent + GetPin37
Next ic
InputCurrent = InputCurrent/20 ' Average out input current
InputCurrent = Abs(InputCurrent*16.667) ' Correct for opamp
amplification
' current by 16.667 now gives a value 50mV = 1A

' Get the load current as a voltage
LoadCurrent = 0
GetPin38 = 0
For ic = 1 To 20
    GetPin38 = Pin(a3) - 0.5803
    LoadCurrent = LoadCurrent + GetPin38
Next ic
LoadCurrent = LoadCurrent/20 ' Average out load current
LoadCurrent = Abs(LoadCurrent*16.667) ' Correct for opamp amplification
' current by 16.667 now gives a value 50mV = 1A

' Get the current time, extract the minutes component
If Mid$(Time$,7,2) = "00" Then
    MinFlag = 1
EndIf

' Get the current time, test for hour AND minute turnover
If Mid$(Time$,7,2) = "00" And Mid$(Time$,4,2) = "00" Then
    HourFlag = 1
EndIf

' Get the current time, test for 0600:00
If MinFlag And HourFlag And Mid$(Time$,1,2)="06" Then
    EODFlag = 1
EndIf

NewDataFlag = 1 ' set to indicate fresh data

IReturn ' Returns with the following values:-
' BVolts = 0 to 15 equalling battery volts
' SVolts = 0 to 25 equalling solar Array volts
' InputCurrent=0 to 50 equalling 0 to 50Amps
' LoadCurrent=0 to 50 equalling 0 to 50Amps
'End of Interrupt code
.....

' Defined Subroutines
.....

' Set up terminal display with static data (field names etc.)
```

' - uses the PRINT @ to print at defined positions

Sub DisplayStatic

Local xrg,yrg,volts,amps

Cls

Print @(150,1)"CARAVAN POWER MONITOR V2.04"

Line(150,11)-(312,11)

Print @(1,25)"Instantaneous Readings"

Print @(198,25)"Minute Average Readings"

Line(1,36)-(152,36)

Line(198,36)-(346,36)

Print @(376,1)"Battery"

Print @(1,45)"Battery Volts ="

Print @(198,45)"Battery Volts ="

Print @(379,12)"Charge"

Print @(1,56)"Solar Array Volts ="

Print @(198,56)"Solar Array Volts ="

Print @(1,67)"Input Current ="

Print @(198,67)"Input Current ="

Print @(1,78)"Load Current ="

Print @(198,78)"Load Current ="

Print @(1,89)"Bat Chg/Dis Curr ="

Print @(198,89)"Bat Chg/Dis Curr ="

Print @(1,108)"Daily Accumulating Readings"

Print @(198,108)"Previous Days Total Readings"

Line(1,120)-(170,120)

Line(198,120)-(362,120)

Print @(1,127)"Input Charge ="

Print @(198,127)"Input Charge ="

Print @(384,127)"Time to"

Print @(1,138)"Load Discharge ="

Print @(198,138)"Load Discharge ="

Print @(400,138)"30%"

Print @(1,149)"Battery Chg/DisChg="

Print @(198,149)"Battery Chg/DisChg="

Line(0,162)-(479,431),1,B

Colour 7,0

Print @(424,164)"Battery"

Print @(430,174)"Volts"

Colour 3,0

Print @(430,210)"Solar"

Print @(430,220)"Volts"

Colour 6,0

Print @(430,260)"Input"

Print @(433,270)"Amps"

Colour 5,0

Print @(433,315)"Load"

Print @(433,325)"Amps"

Colour 2,0

Print @(427,375)"Battery"

Print @(436,385)"Amps"

```
Print @(430,395)"In/"
Colour 4,0
Print @(449,395)"Out"
Colour 7,0
volts=22
amps = 48
For yrg =167 To 387 Step 10
  Print @(2,yrg)Format$(volts,"%2.0fV")
  Print @(400,yrg)Format$(amps,"%2.0fA")
  amps=amps-2
  yrg=yrg+10
  Print @(400,yrg)Format$(amps,"%2.0fA")
  volts=volts-1
  amps=amps-2
Next yrg
Line(29,163)-(29,413)
Line(393,163)-(393,413)
For yrg = 163 To 413 Step 10 'minor ticks
  Pixel(28,yrg)=7
  Pixel(394,yrg)=7
Next yrg
For yrg = 173 To 413 Step 20 'major ticks
  Pixel(27,yrg)=7
  Pixel(395,yrg)=7
Next yrg
Line(30,414)-(390,414) 'now for the bottom
For xrg = 30 To 390 Step 5
  Pixel(xrg,415)=7
Next xrg
For xrg = 30 To 390 Step 15
  Line(xrg,415)-(xrg,416),7
Next xrg
For xrg = 30 To 390 Step 30
  Line(xrg,415)-(xrg,417),7
Next xrg
For xrg = 30 To 390 Step 60
  Line(xrg,415)-(xrg,418),7
Next xrg
Print @(100,419)"6 Hour Graphing period - starts 6AM"
End Sub
.....

' Bar Graph - Draw Static
' top left x=432, y=0, bottom right x=479, y=162
' This equates to 2 steps per percentage for the range of 35% to 100%
Sub BarGraph
  Local xtl,ytl,xbr,ybr,percent,xpos,ypos,y

  xtl=432:ytl=0:xbr=xtl+47:ybr=ytl+162
  ' First draw the box and values
  Line(xtl,ytl)-(xbr,ybr),1,B
```

```
percent=100
Print @(xtl+3,ytl+2)"Battery"
Print @(xtl+13,ytl+13)100
Print @(xtl+40,ytl+13)%"
percent=percent-5
For y=ytl+22 To 142 Step 10
    Print @(xtl+19,y)percent
    Print @(xtl+40,y)%"
    percent = percent-5
Next y
End Sub
' . . . . .

Sub CreateLogFile
    LogFileName$="B:"+Mid$(Date$,9,2)+Mid$(Date$,4,2)+Left$(Date$,2)+".CSV"
    Open LogFileName$ For Output As #1
    Print #1, "Battery Voltage";Chr$(44);
    Print #1, "Solar Array Voltage";Chr$(44);
    Print #1, "Input Current";Chr$(44);
    Print #1, "Load Current"
End Sub
' . . . . .

' Bar Graph - Fill
' top left x=432, y=0, bottom right x=479, y=162
' To fill, start at x=434 thru 443 and y=150 back up to y=12
' This gives 138 steps, equates to 1.97 steps per % for 30% to 100%
' Enter with actual battery charge as a percent

Sub FillBarGraph(BatCharge)
Local xtl,ytl,bat_chg,xpos,ypos,yper,xright,xleft,ybot,ytop
bat_chg=BatCharge
xtl=432:ytl=0:xbr=xtl+47:ybr=ytl+162
' Now fill the bar graph
If bat_chg < 31 Then
    bat_chg=30
    Colour 4,7
    Print @(xtl + 4,ytl + 151,2)"DANGER"
Else
    Colour 2,7
    Print @(xtl + 4,ytl + 151,2)"  OK  "
EndIf
Colour 7,0
If bat_chg > 100 Then bat_chg=100
ytop=12
yper=Int(((100 - bat_chg) * 1.97) + 12)
ybot=150
xleft=437
xright=446
For ypos=ybot To ytop Step -1
    If yper < ypos Then
        For xpos=xleft To xright Step 1
            Pixel(xpos,ypos)=2
```

```
Next xpos
Else
  For xpos=xleft To xright Step 1
    Pixel(xpos,ypos)=0
  Next xpos
EndIf
Next ypos
End Sub
' End program.
.....
```

From:
<https://fruitoftheshed.com/wiki/> - **FotS**

Permanent link:
https://fruitoftheshed.com/wiki/doku.php?id=mmbasic_original:battery_management_for_12v_system

Last update: **2024/01/19 09:39**

