## Formulas in microcontroller code using calibration data

### Battery temperature from microcontroller built-in temperature sensor

// Reading register "ADCW" takes care of how to read ADCL and ADCH.

wADC = ADCW;

// The offset of 324.31 seems to be reasonably accuratem at least down to -9C

// calibration results

// actual temp at chip reported temp raw wADC

// 20.4C 21.1C+-0.5

// 0C 0.5

// 60C 54C

// -23C

dblBattTempDegC = ((double)wADC - 324.31) / 1.22; // ours

### Battery voltage from ADC0 raw count

sensorValue0 = analogRead(sensorPin0);

dblBattVoltageV=((double)sensorValue0 \* 15.0)/1023.0; // full scale is 15.0 V when 1:3 divider is used

### Battery current from ADC1 raw count and battery temperature

const double cbdlZeroCurrVoffset=2.469; // in Volts, at 20°C

// 0.0V = raw count 0 makes -107A, 5.0V = raw count 1023 makes +106A

// if -1 is returned, then it is ADC error, i.e. overflow or underflow

sensorValue1 = analogRead(sensorPin1);

// perform temperature compensation for hall DC current sensor

sensorValue1=**iHallSensorTempCompensate**(sensorValue1, dblBatteryTempDegC); //

// now subtract Voffset, 2.465V -> 505 in raw units

dblBattCurrentA=(((double)sensorValue1 \* 5.0)/1023.0 - cbdlZeroCurrVoffset)\*43.67 ; // 0.0V is -100A, 2.478V is 0A (505 raw ADC counts @ 20C), 5.0 V is +100A

// function performs temperature compensation for Hall DC current sensor to bring to +20̊̊C

// in raw ADC counts

int **iHallSensorTempCompensate**(int iInputSensorValue, double dblTemperatureDegC)

{

int iTempCompensatedCount=0;

iTempCompensatedCount=(int)((double)iInputSensorValue - 0.09125 \* (dblTemperatureDegC-20.0));

return iTempCompensatedCount;

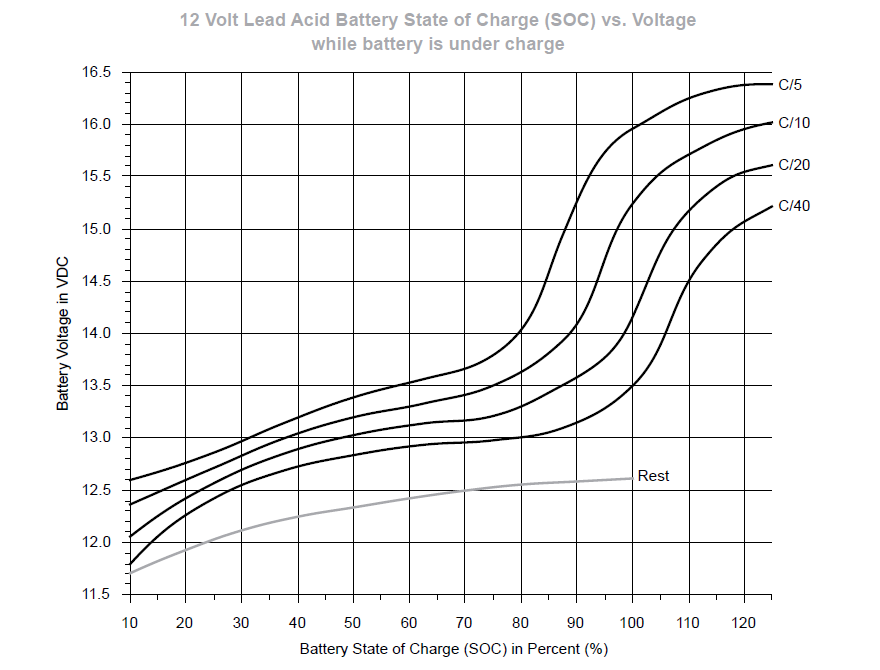
}

## State of Charge Tables

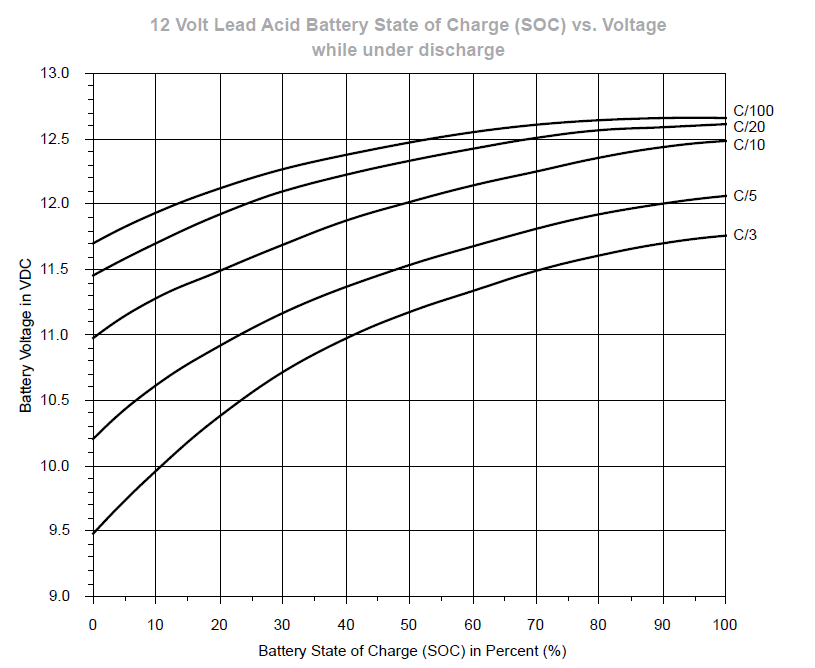
Lead Acid battery State of Charge (SoC) tables are taken from this document lead\_acid\_battery\_charging\_graphs.pdf based on the article “Lead-Acid

Battery State of Charge vs. Voltage” by Richard Perez.

### SoC under charge



### SoC under discharge



Combined curves for under discharge, at rest, and under charge

