True RMS Voltage Calculation with C

Xiao Xu **11/14/12** **Design Team 7**

Abstract

The ability to acquire fast and accurate True RMS values is extremely beneficial to applications in various industries such as medical, industrial and even businesses. The Texas Instrument's MSP430 microcontroller is going to be used to calculate the true RMS voltage of an AC voltage signal. The methods of acquisition and flash memory storage will be covered.

Objectives

The purpose of this application note is to explain the definition and methods of true RMS mathematically and how to implement this method into the MSP430 microcontroller using C. This application note will first start with a short explanation of RMS, then go into how to correctly use and interface the MSP430 microcontroller to a windows computer, and lastly finish with an explanation and implementation of the RMS calculation C code.

True RMS Calculation

RMS voltage is defined to be a DC equivalent of an AC voltage. As an example, if a 20V DC source is using up 20W of power, then a 20Vrms sinusoidal or non sinusoidal AC voltage either will dissipate the same power across the same resistor. The mathematical meaning of true RMS is below:

$$\mathrm{Vrms} = \sqrt{\frac{\sum_{n=1}^{w}(\mathrm{V_n})^2}{\mathrm{w}}}$$

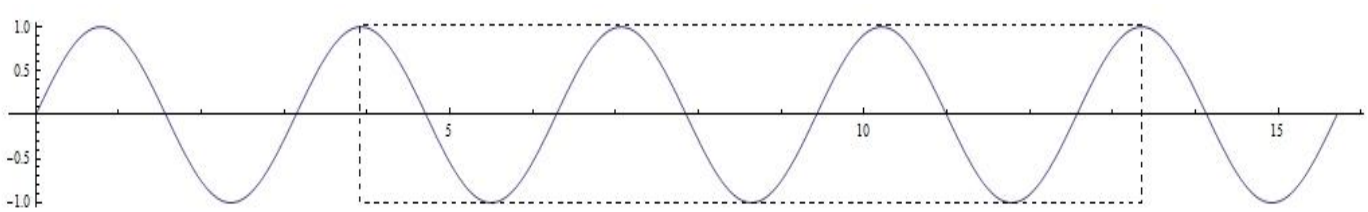
Where Vn is the instantaneous voltage and w is the window size.

For example, the true RMS voltage of a window size of 80 is would be the square root of each of instantaneous voltage of the 80 voltages squared divided by 80.

If under the rare circumstance that the AC voltage is a pure sinusoid, then the Vrms = $\sqrt{2}$ of the instantaneous voltage.

True RMS Signal Acquisition

To measure the true RMS of a signal, we need to first define w, the RMS window. The window size will determine the size of the samples. Below is a



You want to make the window size large enough to collect enough samples, but not too large as the result will be a lot less responsive. A good guideline is to use a window size that's large enough to include at least 2 or more cycles.

True RMS Calculation utilizing C code

The trick here to define a window size, and have it shifts accordingly in time. If the required window of values is 80, we're going to need to create an array of 80 and continuously shift the last value out while taking the new input voltage and putting it in the first value of array. This way the window is always moving with time with a speed dependent on the clock of the microcontroller. The information flash memory is also needed to store the instantaneous voltage values so the calculations can be made. Below is a sample code of taking the input voltage and storing it in the flash memory and calculating the output RMS.

Sample Code and Flash Memory storage

```
int newrmsx=0,newrmsy=0,newrmsz=0;


   newrmsx=adcvaluex;
    newrmsx=newrmsx/4;
    adcvaluey = analogRead( INPUT_1 );
    newrmsy=adcvaluey;
    newrmsy=newrmsy/4;
    adcvaluez = analogRead( INPUT_2 );
    newrmsz=adcvaluez;
    newrmsz=newrmsz/4;


count=0;
    write_SegC(newrmsx, count);
    count=count+1;
    write_SegC(newrmsy, count);
    count=count+1;
    write_SegC(newrmsz, count);
    char *Flash_ptr;
    Flash_ptr = (char *) 0x1000;
    unsigned long xtot=0,ytot=0,ztot=0,xnew=0,ynew=0,znew=0;
    for(i=0;i<80;i++){
      xnew=(*Flash_ptr);
      xtot=xtot+(xnew*xnew);
      Flash_ptr=Flash_ptr+1;
      ynew=(*Flash_ptr);
      ytot=ytot+(ynew*ynew);
      Flash_ptr=Flash_ptr+1;
```

```
    znew=(*Flash_ptr);
    ztot=ztot+(znew*znew);
    Flash_ptr=Flash_ptr+1;
  }

    xtot=sqrt(xtot/80)*0x04;
    ytot=sqrt(ytot/80)*0x04;
    ztot=sqrt(ztot/80)*0x04;
```

Because the flash information ram has 256 values, and X, Y, and Z arrays both contain 80 value arrays, we have just enough space for all three arrays. In the sample code above we're storing the first X, Y, Z bits in tot the first 3 spaces in the information flash and then incrementing 3 appointed pointers to the next 3 flash memory address. The resulting flash memory map would be X1, Y1,Z1, X2,Y2,Z2,X3,Y3,Z3 etc.