

# Log the Linear BLM Signals

Wed, Jan 17, 2007

The linear BLM values from the new digitizers designed by Craig Drennan must be accessible via the FTPMAN Snapshot protocol. But the integrated values will be 32 bits long. Either we must make changes to allow delivery of 4-byte values in reply to such requests, or we must convert the 32-bit values into 16-bit values. To do the latter, consider converting the linear 32-bit values into 16-bit log values. One will lose some precision, of course, but it may be quite adequate for plotting.

The formula used to convert the log values into engineering units is arbitrary, but for the sake of this discussion, assume it is chosen to match that used for the older log amplifier digitized values.

$$R = 10^{(2.5*v - 3.343)}$$

where R is radiation in engineering units, and v is the raw log amplifier output voltage.

Assume that the new linear values are directly proportional to radiation.

$$R' = k*v'$$

where R' is the radiation result, and v' is the linear voltage reading. Note that this v', since it is the sum of successive raw voltage readings, may be considerably larger than 10 volts.

Compute a value for v such that R' = R.

$$k*v' = 10^{(2.5*v - 3.343)}$$

$$\log_{10}(k*v') = 2.5*v - 3.343$$

$$v = (\log_{10}(k*v') + 3.343)/2.5$$

$$\log_{10}(k*v') = \log_{10}(k) + \log_{10}(v')$$

$$\log_{10}(v') = \log_2(v') * \log_{10}(2)$$

As an integer value, what we have for v' is  $n = 32768*v'/10$ , so that  $v' = n/3276.8$ .

$$\log_2(v') = \log_2(n) - \log_2(3276.8)$$

We can easily compute  $2048*\log_2(n)$  from the floating point exponent field and a table lookup.

Putting it all together, we have

$$v = (((\log_2(n) - \log_2(3276.8)) * \log_{10}(2) + \log_{10}(k) + 3.343) / 2.5)$$

$$v_{\text{raw}} = (((\log_2(n) - c_1) * c_2 + c_3) * c_4)$$

where

$$k = 0.0108558 * 3276.8 \quad 35.57229$$

$$c_1 = \log_2(3276.8) \quad 11.67807$$

$$c_2 = \log_{10}(2) \quad 0.30103$$

$$c_3 = \log_{10}(k) + 3.343 \quad 4.894112$$

$$c_4 = 3276.8 / 2.5 \quad 1310.72$$

$$v_{\text{raw}} = \text{Integer representation of voltage } v.$$

We can simplify it even further to:

$$v_{\text{raw}} = \log_2(n) * c_5 + c_6$$

where

$$\begin{aligned} c_5 &= c_2 * c_4 && 394.566 \\ c_6 &= (c_3 - c_1 * c_2) * c_4 && 1807.04 \end{aligned}$$

Note that this could be done using integer arithmetic. What we have originally, when examining the value of  $n$  is  $2048 * \log_2(n)$ . Rewrite the above formula for  $v_{\text{raw}}$  as:

$$\begin{aligned} v_{\text{raw}} * 65536 &= (2048 * \log_2(n)) * (32 * c_5) + c_6 * 65536 \\ v_{\text{raw}} * 65536 &= (2048 * \log_2(n)) * 12626 + 118426154 \end{aligned}$$

This may be done as 32-bit integers, with  $v_{\text{raw}}$  obtained from right shifting the result by 16 bits.

### How to get $2048 * \log_2(n)$

Convert the 32-bit integer  $n$  into single precision floating point. Then examine the 8-bit biased exponent field of this 32-bit value, and remove the bias of 127. This gives the 5-bit characteristic of the result. Take the most significant 11 bits of the 23-bit mantissa, not counting the hidden “1” bit, and index into a array of 2048 elements to get a similar 11-bit value that obeys the log curve.

The values in the table elements are  
 $2048 * \log_2(1.0 + i/2048)$

where  $i$  ranges from 0 to 2047. Note that the  $\log_2$  ranges here from 0 to nearly 1, so the final result, when converted to an integer, ranges from 0 to 2047. Take the characteristic times 2048 and add the 11-bit value from the table to get the final value of  $2048 * \log_2(n)$ .

A few examples follow:

$n$	$float(n)$	$exp$	$ch'ristic$	$2048 * \log_2(n)$	$v$	$R$
16	41800000	131	4	2000	1.033	0.174
1024	44800000	137	10	5000	1.755	11.12
16384	46800000	141	14	7000	2.237	177.9
1048576	49800000	147	20	A000	2.960	11383
33554432	4C000000	152	25	C800	4.361	364260

The last example is larger than anything mathematically possible with a 500-point summation.

The precision of the result is about 3 decimal digits, which should be enough for plotting.

### Computing $2048 * \log_2(n)$ for the PowerPC

The PowerPC instruction set includes `cntlzw`, which yields the number of leading “0” bits in a 32-bit integer. It turns out that this is considerably faster than the method described above that starts with converting a 32-bit integer into a single precision `float`. Converting the integer to a `double` on the PowerPC, for which double precision is fundamental, resulted in a total time—for building the 32-bit sums and computing the required logs—of 0.35 ms for 2K points. Replacing the floating point operations with the `cntlzw` approach gave a total time of 0.20 ms. These times were achieved on a PowerPC running at 233 MHz.