

# Booster BLMs

## *Local Application BLMS*

Fri, Jul 14, 2000

Booster Beam Loss Monitors are supported by a local application called BLMS that accumulates dose measurements from each BLM according to the Booster reset type of the cycle on which the measurements are made. This note describes how it is done.

### *BLMS dose accumulations*

Each 15 Hz cycle, a determination is made of which Booster cycle reset occurred at the start of the cycle. The function `HaveEvt` is called for each possible Booster reset event number until one is found whose bit is set in the clock events bit array maintained by the system. The valid Booster reset events are the following:

11, 12, 13, 14, 15, 16, 17, 19, 1C, 1D

Separate accumulations are made for each reset type and for each BLM. With accommodations for up to 16 BLMs in one node, using two 8-channel Swift digitizer modules, there are 160 separate accumulations maintained using the above list of 10 valid reset events. These accumulations are placed into the data pool covering a range of 160 consecutive analog channels—out of a typical total of 1024 allocated for each IRM node. If only 12 BLMs are used, say, there are gaps of unused channels in the range of 160. These gaps are reserved for additional sums in the case that more BLMs are added. If additional Booster reset events are added, the total list will be longer than 160 channels, 16 channels longer for each additional reset event.

The sums are measured over a selected period of time and are updated multiple times during this period. For example, a period of 60 minutes may be used, so that the sums represent data accumulated over the last hour. These hourly sums may be updated every 5 minutes, so that 12 sets of partial sums must be maintained in order to be able to update the long term sums that often. Currently, a maximum of 16 updates are allowed per long term period. (This limit can easily be increased if needed.)

The long term sums are cleared when the BLMS local application starts up. During the first long term period, then, the sums might be observed to increase for each partial update. Once the first long term period is over, the sums should follow the relative beam losses that the BLMs measure, updated after every partial period. (If the ramping behavior during the first long term period is undesirable, a modified scheme can be used without this effect.)

The behavior of the long term sums will be such that an unusual spike of beam loss will influence the long term sum to the same extent for the entire long term period, after which its influence is lost. In addition, one may have to wait for as long as a partial sum period before the impact of a spike is seen on the long term sum.

Another scheme of maintaining a continuously-updated running sum may be used instead, if desired, in which a spike would have a declining influence in the long term sum over time; after the long term period, its influence would have dropped to  $1/e$  (0.368) of its initial influence. Whichever method is used, for tuning purposes, one would presumably use the instantaneous beam loss measurements captured on a selected reset event of interest rather than any long term summation.

Independent of which long term summation scheme is used, comparing beam loss between different Booster resets must take into account how many of each reset event occurred during the long term period. If the long term period is too short (as in the example shown used for

testing), an error of one reset event in the long term period may be significant. (The long term period is based upon time, not, say, upon supercycles.)

Because of the wide range of potential beam loss occurring during one Booster acceleration cycle, the BLM hardware signal is a log of the integrated beam loss measurement during the 33 ms Booster acceleration cycle. Because a log amplifier is used, an offset is applied on purpose. (It's hard to take the log of zero.) In dealing with these readings, it is desirable to remove the offset. Since the BLM data comes into the IRM as a waveform, and recognizing that the waveform is an integrated signal, the log value is converted to a loss (in units of rads/sec) at the end of the acceleration cycle and at the start of the cycle before beam is injected, and the difference of these two losses provides the values that are summed. The conversion is done via table lookup, since all BLMs obey the same conversion formula:

$$\text{rads/sec} = 0.00721196 * \text{Exp}(1.0057772 * \text{volts})$$

Separate sums are accumulated for each BLM on each 15 Hz cycle, according to the reset event number of that cycle.

### *BLMS parameters*

```

E LOCAL APPS      07/11/00 1434
NODE<06C3>  NTRY< 5>/64  H<0508>
NAME=BLMS  CNTR=95  DT= 0  MS
TITL"BOOSTER BEAM LOSS MONTRS"
SVAR=000417F0      07/10/00 1545
ENABLE  B<00B0>*BLMS ENABLE
INIT INX <0001>
FINL INX <01B8>
BLM     C<0200>  BLMTS0 0      vlts
#BLMS   <0002>
PART #CY <0100>
#PARTS  <0006>
SUMS    C<0210>  0
          <0000>
          <0000>

```

Following the enable bit parameter that all local application instances use, the two indexes specify the zero-based word index values that cover the Booster acceleration cycle. If the Swift digitizer operates at 12.5 KHz to capture the waveforms, beginning at the reset event time that occurs 2 ms before BMIN, the initial value might be near 0, and the final index might be near 440, or 0x01B8. This corresponds to a time of 35.2 ms after the reset event time, or 33.2 ms after Booster beam injection. The intent is to sample the value measured just after accelerated Booster beam is extracted.

The analog channel number associated with the first BLM is determined by the configuration data in the CINFO system table. This is how BLMS determines where the waveform data is to be found in memory space. The BLMs used in a node are assumed to be assigned consecutive channel numbers. If a second 8-channel Swift digitizer module is used, its channels should follow the first eight, allowing for up to 16 channels. The parameter specifying the number of BLMs determines how many channels will be processed, beginning at the initial channel. For the case shown above, only two BLM signals were actually connected in this test node.

The number of 15 Hz cycles specifies the length of the partial period. The number of these

periods (“parts”) determines the long term period. In the test node case shown above, the partial period was about 17 seconds, and the long term period was about 100 seconds. To establish a partial period of 5 minutes and a long term period of one hour, use `PART #CY <1194>` and `#PARTS <000C>`. (These hex values correspond to 4500 cycles and 12 partial periods.)

The sums are exhibited as analog channel readings. The initial channel for these output values is specified. The number of channels used is 16 times the number of clock events used. As described above, this is 160 channels for 10 clock events. The order of these channels is the number of BLMs in use for the first clock event followed by the BLMs in use for the second clock event, etc, with gaps to make room for those BLMs not in use. The sums for the first BLM will therefore be found at channels 0200, 0210, 0220, etc, in the above example, corresponding to clock events 11, 12, 13, etc. The sums for the second BLM will be at channels 0201, 0211, 0221, etc.

### *Floating point sums and alarms*

The sums are computed in floating point, of course. Because of the wide range of possible beam losses, it is impractical to convert these into 16-bit values as has been done for all IRM channel readings heretofore. Additional support has therefore been added to the IRM system software for floating point raw data. An analog channel may be configured to have floating point values of integer values. If it is designated a floating point channel, its data is housed in a parallel table to the usual `ADATA` (Analog Data) table. The parallel table has space to hold floating point values for reading, setting, nominal and tolerance values. The floating point values are assumed to be in engineering units; there is no “raw” version that exists. These data are “born” in engineering units; they have no other form. Suitable Acnet primary and common transforms must be used to reflect this.

Along with adding a new table that houses floating point data values, the alarm scanning task was modified to support them as well. If a channel is designated floating point, then alarm checking is done using floating point comparisons.

Because of the nature of beam loss sums, alarm checking will naturally specify minimum and maximum values rather than the usual nominal and tolerance values. The minimum value is likely to be zero, and the maximum value will be a value of the long term sum above which an alarm message is desired. As soon as the loss improves, and the sum drops below that maximum threshold, one would wish the alarm system to indicate the losses are “good.” As a result, the new minimum and maximum alarm checking logic does not exhibit the “hysteresis” behavior that it does for the nominal and tolerance cases, in which a channel in the “bad” alarm state does not regain “good” status until the reading falls within half the tolerance amount. The new min/max logic can also be used for integer channels.

The designation of a channel as floating point is done via bit 12 of the alarm flags word field in the `ADATA` table entry. The designation of min/max alarm checking logic is done by setting bit 5 of the same alarm flags word. The Acnet analog alarm block handling in both `RETDAT` and `SETDAT` will be modified to deal with these new options. Another note deals with these details.