

Booster BLM Support

Front-end software

Tue, Sep 4, 2001

Introduction

The latest generation of Beam Loss Monitor hardware is installed in the Booster to improve Booster operational diagnostics. With new requirements placed upon the Booster to operate in multiple modes to serve the MiniBooNE, NuMI, as well as the CDF and D0 colliding beam detectors, this hardware and software are needed to assess the health of accelerator operation, which may have to be restricted if beam losses are too high. It is hoped that the new BLM system can help diagnose beam loss and find a way to reduce it.

This note focuses on the software support provided in the Internet Rack Monitor (IRM) front-ends to make accessible beam loss data from the Booster BLMs. Some of the required support was added to the basic IRM system software anticipating that it may also be used for other purposes. The rest is supported by a local application called `BLMS` that runs as an extension to the basic system software.

Hardware configuration

The BLM hardware signals connect to the IRM via a chassis that includes up to 12 signals, each of which is the log amplifier output of a beam loss integrator that is reset before the start of each Booster acceleration cycle. Each of these signals is fed to a Swift digitizer module installed within the IRM. One digitizer module supports 8 channels, so that one or two modules are installed in each IRM. With a total of 75 BLM signals originating from various positions around the Booster ring, eight IRM front-ends are used. Each IRM is a full-fledged Acnet-compatible front-end that connects to ethernet. The Swift digitizers operate at a 12.5 KHz digitize rate, because this $80\mu\text{s}$ period is compatible with the BLM signal rise-time characteristics.

Synchronization

Since the Booster acceleration cycle is about 33.5 ms from beam injection to extraction, the measured Swift digitizer waveforms cover 40 ms, using 500 points. The digitizers are configured to measure each waveform on every 15 Hz accelerator cycle, whether beam is present or not. Clock event `0x0c`, which occurs about 2 ms before beam injection, is used to trigger the start of digitization. The IRMs that support the BLMs begin their 15 Hz operation at 40 ms, which is shortly after Booster beam extraction time, so that all waveforms just measured on that cycle are immediately available to the CPU in the digitizer waveform memory.

Software components

No software is required to collect the waveform data, since the Swift digitizer hardware delivers it into memory every 15 Hz cycle. But if it is needed by a user, it must be captured promptly for delivery over the network. If waveform information is to be collected by a console requester at 15 Hz, the receiver software must be able to keep up. Although current Acnet consoles cannot reliably collect 15 Hz data correlated across front-ends, a scheme was designed to deliver such data even while operating within the existing limitations.

The "time-stamped" scheme described here was implemented in the `RETDAT` support of IRM system software. Using a single Acnet device name for an individual BLM, one can obtain the raw 2-byte readings from the usual data pool, which results in the value measured just after extraction, or one can obtain any selected portion of the raw waveform. In addition, one has the option of obtaining this data in the form of a "time-stamped" data structure, where two cycles of data are delivered at 7.5 Hz. The form of data returned depends upon the request

parameters, including the period, offset, and length. (The ability to get these time-stamped 15 Hz data structures can be used for access to any data, not only BLM waveform data.) Key to making such a request is requesting that data be returned at 7.5 Hz, and specifying a suitable length for the reply data structure.

The essence of the scheme is to deliver on every other 15 Hz cycle a data structure that includes the requested data captured from two successive cycles. The receiving console application software operates at a nominal 15 Hz rate, although it is asynchronous with accelerator operation. This software checks for the arrival of new data at every nominal 15 Hz opportunity, recognizing that there will be fresh data about every other time. To correlate measurements of waveform data across multiple front-end sources, the data is delivered to the console in a structure that includes a globally-available 15 Hz cycle counter. The application uses the cycle counter value to match the response data from multiple IRMs. In this way, it can work with data from any or all BLMs that was measured on a single 15 Hz cycle. To assist in finding the cycle of interest, a software device `B:BREVNT`, whose value is the current Booster reset clock event, should also be requested via the same "time-stamped" scheme. This can permit selecting out from the stream of 15 Hz beam loss waveform data those that were measured on `0x17` event cycles, for example.

A somewhat simpler approach for capturing correlated data across multiple front-ends can work if the data to be captured is to be measured on clock event cycles that occur only infrequently, say, at 1 Hz or less. It is also necessary that the console software have some means of knowing ahead of time that the chosen event is coming up soon (but not too soon). It then issues a one-shot request for all such data, specifying the clock event of interest. When all the data is returned, it will be correlated, which is to say, measured on the same 15 Hz cycle. The prediction requirement is not too difficult to achieve if the one-shot requests are made repeatedly following successful reception of the reply data from the previous set of requests. The arrival of the reply data from all front ends is a golden opportunity to issue the next set of one-shot requests. One could imagine a variation on this approach by issuing clock event requests that are not one-shot. But one gets into the danger of confusing which reply is appropriate to compare with reply data from another front-end, since the console program, operating asynchronously with accelerator operation, cannot depend upon all the data arriving at once, even if the front-ends transmit their replies simultaneously.

Another way to view waveform data on Acnet consoles is via the `FTPMAN` protocol, in which a snapshot request is used. Since the Swift digitizers used for measuring the BLM data are used in only a single mode (clock event = `0x0C`, delay = 0, digitize rate = 12.5 KHz) on every 15 Hz cycle, the user cannot be allowed to change that mode. Consequently, a snapshot request for such data returns the same data that resides in the waveform memory. Also, since the snapshot client cannot operate at 15 Hz, this is not a suitable method for obtaining BLM waveform data from successive 15 Hz cycles. Only the "time-stamped" scheme described above can do that reliably, using the `RETDAT` protocol.

Accessing 500 points of waveform data from 75 BLM signals at 15 Hz may seem to be a large amount of data to collect, amounting to 75000 bytes. There will of course be a limitation on how many such requests can be simultaneously supported. Experience will determine those limits. Internal IRM timing diagnostics can be used to help assess the front-end CPU load.

Examining typical BLM waveform signals, one sees that at the time the digitization begins, which is the same time as the Booster reset clock event, the integrators are not yet reset. It takes about 1.0–1.5 ms for the integrator reset to complete. We are therefore currently assuming a 1.5 ms delay before the start of the waveform signal to be analyzed for a given

cycle. (One would not want to mix data left over from the effects of beam losses occurring on the previous cycle.) But the waveform buffer begins at clock event reset time, so that its response at integrator reset time may be viewed.

BLMS support

The BLMS local application provides two key services. The first provides a continuous summation of recent total beam loss for each BLM and for each reset clock event. Although the parameters are adjustable, current parameters specify that the beam loss sums cover the most recent 100 seconds of time, updated 6 times (about every 17 seconds) within that period, which is approximately consistent with the averaging characteristics of "chipmunk" radiation monitors.

The logic for this summation is to sample the waveform at two points, perform the exponential to derive the integrated loss in rads/second, and report the difference as a floating point result. The two points are BLMS parameters, and they are chosen to sample the loss after integrator reset but before beam injection and just after beam extraction. Sums of these differences are accumulated over the 100 second period. To access these data values, there is an Acnet device defined for each reset clock event for each BLM. A separate analog channel is defined in the front end for each BLM. The channel allocation logic allows for up to 16 BLMs and 12 clock events. The first 10 clock events are 0x11–0x17, 0x19, 0x1C, 0x1D. The eleventh one represents events 0x10+0x12, which shows loss for any power cycle. (A Booster power cycle event is any Booster reset event besides 0x11.) If 12 BLM signals are connected to one IRM, then $12 \times 11 = 132$ devices are required to access the complete set of recent total beam loss sums available in that node. (Although it would be possible to define fewer Acnet devices to access these data via array devices, alarm support could not be offered for each individual BLM/event.)

Implementing support for these total loss summations resulted in adding generic system support for single precision floating point raw data. To that end, a parallel table to the usual ADATA (analog table) was created called FDATA. Indexed by channel number, its entries include space for a reading, setting, nominal and tolerance values, all in 4-byte floating point. Alarm support was added for such floating point channels. With the alarm support was also added generic support for min/max limits, as such would certainly be needed for monitoring these recent total beam loss accumulations. Two additional alarm flag field bits were assigned to denote these two new alarm options. In any case, whether a channel is actively included in the alarm scan or not, the floating point flag bit should be set in any channel that has floating point raw data. Of course, such channels are defined in Acnet as having 4-byte values, rather than the usual 2-byte values. (Analog channels in IRM systems do not have 4-byte integer values.) The Primary units of such channels are the appropriate floating point format, and the Common units are the identity formula ($x' = x$). For such data, there is no raw A/D volts value; there is only the floating point value delivered by the front end.

The second service provided by the BLMS local application is to accumulate coarsely-sampled differential losses throughout the Booster acceleration cycle. To that end, each waveform is sampled every millisecond, and differences of the losses between successive milliseconds are accumulated into double precision floating point sums. Since the BLM signals are integrator outputs, this provides sums of the losses occurring during each millisecond, for each BLM and for each reset clock event. These accumulations are intended to be data-logged over the long term. Interpretation of such data must be done by computing differences of archived data between two times of interest; the individual absolute summation values carry no useful information. Because double precision values are 8 bytes in length, using 1 bit for a sign, 11

bits for a binary exponent, and 52 bits for a mantissa, there is no need to reset these sums. They are maintained in nonvolatile memory in the IRMs, so they even survive a power failure. If the front end dies, the accumulations continue to count beam losses as soon as it is back in operation. This means that any time interval can be used for querying the archived data, without being concerned about when the loss accumulations may have been reset or "rolled over." The only exceptions could be the failure of the nonvolatile memory hardware or inadvertent destruction of that area of nonvolatile memory by errant software. (This memory is nonvolatile, but it is still writable.)

BLM data summary

Reviewing the possibilities that exist for accessing the BLM data from the front ends, there are schemes appropriate for both short term and long term study. The utmost in real-time access for the short term is provided by the "time-stamped" waveform support that returns waveform data captured from two successive 15 Hz cycles on every other 15 Hz cycle. Although it requires a special application to do the necessary data-caching gymnastics, it makes possible reliable access to correlated 15 Hz data across multiple front ends. For certain types of beam studies, this may be a useful tool. It provides access to the full spectrum of BLM data that is acquired by the IRM hardware.

Another short-term view would be access to the differential losses that are accumulated into the loss summations for each millisecond during the acceleration cycle. One can imagine a 3D presentation of such differential losses for all BLMs and reset events that would show where and when the losses are greatest. This could be very useful for tuning, where millisecond timing precision is good enough.

A third not-quite-so-short-term view is provided by the recent total loss accumulations over the previous 100 seconds, say, separated out for each BLM and each reset event. This can be used for tuning, keeping in mind that any unusual spike in loss will be around for no more than the summation period. Perhaps it's like tuning with "chipmunks." Another use for this could be as a beam loss alarm vehicle. If the recent average losses exceed a specified limit, an alarm message can result. With the current parameters in force, such a message remains for about 17 seconds.

The main long-term level of support is the differential millisecond beam loss accumulations that should be data-logged. As a result, it will likely be accessed off-line, but it can be used to compare beam losses over any period of time covered by the data-logger archive. It can serve to provide a long view of Booster beam operation efficiency.

Operational performance

Data logging of millisecond data can be done currently for all BLMs in less than 500 milliseconds. It is done via separate requests for the 3840-byte millisecond data of each of the 75 BLMs, for a total of 288000 bytes. During this time, about 10% of the front-end CPU time is required to serve up the data. In the absence of such requests, these front-ends are typically busy about 10% of the time, so the request sequence approximately doubles this load.

When a BLM front-end is serving up 7.5 Hz data for 12 BLM waveforms, it spends about 7 ms collecting the waveform data each cycle, plus 10 ms every other cycle and queuing two sets of waveforms to the network. The network spends 20 ms of its 10 MHz bandwidth to transport the data to the requesting node.

The front-end spends 3 ms each cycle (12 BLMs) performing various loss data summations.