BPM Waveform Averaging

Local application BPMR

Robert Goodwin

Fri, Feb 20, 2009

Linac Beam Position Monitors are interfaced via 4-channel digitizers that are triggered just before the Linac beam pulse and digitize at 10 MHz. Beam positions are calculated by averaging the data points over the beam pulse, or at least the first part of the beam pulse. Elliott McCrory wrote a local application called BPMQ to perform this calculation. A special feature is that it only averages data points that come from the BPM hardware during the beam pulse. To do this, it uses a "beam present" signal such as a beam intensity signal coming from the BPM, and it requires a minimal signal to indicate that beam is actually there.

A new version of this BPMQ program for BPMs is called BPMR, whose name refers to its ability to access the digitizer data remotely for testing purposes. Running as a local application in a test node, it makes a data request to the target node for waveform data to be delivered on every beam cycle. Each beam cycle is indicated by Bit 0x009F being 0, so the beam status is also acquired remotely. But there is a *caveat*. When testing this way, the LA processes the BPM data on the cycle *after* the one on which the digitizations took place. After program testing, if the very same LA is run in the node that connects to the digitizers, the LA processing will take place on the actual beam cycle, and its results will therefore be correlated with data from any other Linac front end.

To explain why remotely accessed data is processed by an LA on the next cycle, note that LAs are invoked during the Update task processing, at a time following the updating of the local data pool according to the instructions in the Data Access Table (RDATA). Even if a reply to the data request for remote digitizer data arrives before the Update task is finished, it is not accepted until after the Update task completes. All this is done to create the appearance externally that the local data pool changes instantly; another node cannot "see" a data pool that is only partially updated.

Parameters layout

Field	*Size*	*Meaning*
ENABLE B	2	Usual local application enable Bit#
SIGMA B	2	Set this Bit to enable sigma option
FIRST C	2	Index to first data point Chan#
NUM SAM C	2	Number of data points Chan#
BASEINCR	4	Base address of digitizer memory plus increment
BEAM C	2	Beam present Chan#
NUMGOOD C	2	#points scanned with beam present Chan#
AVERAGE C	2	Average results Chan#, followed by optional sigmas
TARGNODE	2	Target node#, to facilitate testing

The FIRST and NUM SAM parameters allow a user to easily change the portion of the digitized waveforms for which the averages are calculated. The program monitors the readings of these channels, and if a change occurs, it adapts to the new portions right away. Note that changing any of the other parameters may require restarting (disable, re-enable) the LA.

The BASEINCR parameter is set to the base address of the digitizer memory, plus the increment between successive channel waveform memories. For example, if the base address of the digitizer memory is 0x47000000, and the increment is 0x00040000, this parameter should be 0x47040000. By assuming that the increment is a power-of-two, the program can get both base and increment.

The optional BEAM parameter can be interpreted in several ways. If it is nonzero, there is a beam

present waveform. If it is in the range 0x0001–0x07FF, it is assumed to be an actual (local) channel number for which the "analog control field" in its ADESC entry holds the waveform base address. If it is negative, between 0x8000 and 0x8003, it indicates a digitizer index in the 4-channel digitizer of the beam present waveform. For example, if the second waveform is the beam present signal, then this parameter would be 0x8001. This scheme helps to optimize access to digitizer waveform data when the beam present signal is one of the four digitizer channels.

The NUMGOOD parameter is used as a diagnostic to show the number of data points in the scan of points indicated by the FIRST and NUM SAM parameters for which the beam is present.

The AVERAGE is the first results Chan#. The 4 averages are deposited beginning here, and the 4 sigmas, assuming sigma calculations are enabled, follow.

The TARGNODE parameter indicates the target node for acquiring both the waveform data and the beam status. As stated above, this parameter facilitates remote testing, allowing one to operate the program in a test node. Once testing is complete, it can safely be installed in the target node, with TARGNODE set to the local node#, making all the data requests local, so that average processing will be done on data digitized during the current cycle.

Pedestal determination

The signals digitized are expected to be triggered at a time before the start of the Linac beam pulse. To determine a reasonable pedestal offset for the beam signal, BPMR averages the first few points to derive an offset to be subtracted from the beam present data points. The beam present signal does not change much when beam is present, so it is important to have a good value for this offset. Sampling a few data points before the beam pulse is one way to do this. The threshold used for detection of beam presence is small (0x0020), so we need an accurate value. The number of early data points currently averaged for this purpose is 8, starting with the third point. At 10 MHz, we need 10 points, or 1 μs, before the beam signal shows beam.

Program logic flow

During initialization, BPMR examines the parameters and makes data requests for three kinds of data. One is for the waveform data to be averaged. A second is for the early beam data points whose average will determine a suitable beam pedestal offset. The third is for beam status.

During cyclic executions, the LA checks the beam status value, and if it indicates beam, it reads in the reply data points for the other two requests. It computes the averages of the early data points to determine an offset for the beam signal. Then it examines the signal data, calculating averages of all points corresponding to minimal beam. If the SIGMA bit is set, it also computes the related sigmas. It installs the results of both averages and sigmas into the result channels.

If the channel readings for the FIRST or NUM SAM parameter has changed, it sends a new waveform data request that reflects those new values. (In practice, such changes are seldom made.)

Data request-based access

The data is collected via data requests made for memory data words. Whether the target node is local or not, the slow data accesses will be made by the code that builds the reply data to satisfy the requests. The program itself does not have to worry about bus errors or any unusual aspects to the hardware waveform memory considerations. It simply process the reply data words.

To optimize the timing, the waveform data requested is limited to only those data words to be scanned to produce the average data.

Post-implementation

The new BPMR was tested in node0590, targeting real BPM waveform data in node061E. The calculated average and sigma results match those calculated by BPMQ in node061E. The execution times are much faster than before, where BPMQ, using its direct access to the digitizer memory, takes 2.1 ms without sigma, or 2.8 ms with sigmas included. The times of the new BPMR, when operating remotely for testing, are only 80 μs and 170 μs. When running in node061E, it will be much longer, due to accessing the slow memory for the data points, but it should be < 1 ms. The difference is due to a larger number of accesses to slow memory made by BPMQ.

Checking the specs of the Joerger digitizer used for these signals, there were a few surprises. The digitizer is 12 bits, covering a range of ±2 volts, where –2 volts is 0x0000 and +2 volts is 0x07FF. But there is also a status bit included in the bit 12 position that, when set, means no overflow. (When it is zero, it means overflow.) This means that the data word always has bit 12 set, since we are not planning to cause any overflows. Looking at the beam present signal before the beam pulse, we normally see readings very near 0x180A, meaning there is no overflow, and the 12-bit data is 0x080A, which should correspond to near 0 volts. When the beam present signal indicates beam, the readings are about 0x17CC, implying a drop of 0x003E. If we assume that the full range of 0x07FF represents 4 volts, then this drop is about 60 mv. The actual threshold used by the program to qualify beam presence is 0x0020, or about 30 mv.

The beam signal is small, which is why it is important to have an accurate value of the pedestal. In BPMQ, a constant pedestal value of 0x17F8 is used. This is 0x0012 less than the beam signal currently reads just before the beam pulse. This causes BPMQ to sometimes see beam signal data point values whose drops are considered too small to qualify as beam. (An average drop of 0x002C might for some digitized points be less than the 0x0020 threshold. This is why the diagnostic count of "good" beam data points is often somewhat short of the expected 160.)

The scaling used for converting the beam signal to volts is fs=32 and off=–6. These two values will convert a raw value of 0x1800 into 0 volts. A raw value of 0x1FFF will convert to +2 volts, and a raw value of 0x1000 will yield –2 volts. So, our scale factors compensate for the non-overflow status bit being accepted as part of the data value.

At first, the sigmas did not match very well with those produced by BPMQ. But after changing the calculation so it subtracts the beam offset value (near 0x180A) when accumulating the sum of squares and also when subtracting the square of the mean, the sigmas matched much better. (The logic in BPMQ does the same thing.) The difference may be one of computational accuracy. Before, using the full-size values, with single precision floating point, the calculation of the variance may involve subtracting two very large nearly equal numbers, resulting in a large loss of significance. A final change computes the sigma as the average squared deviation from the mean. This requires a separate pass over the data points after computing the mean. But it eliminates the problem of subtracting nearly equal large numbers.

Static memory

During testing, here is an example part of the static memory block used by BPMR.

```
0590:00B507C0   18C0 0000 0000 0007
    :00B507C8   0590 344B 0014 2484          node#, beamCnt, cycleCnt(4)
    :00B507D0   0073 00A0 00A0 180C          first, numSam, numGood, beam pedestal
    :00B507D8   1812 17CD 17FF 17E3          averages for each digitizer channel
    :00B507E0   0010 0003 0020 0016          sigmas
    :00B507E8   0073 00A0 0000 0004          firstPrev, numSamPrev,, #waveforms
    :00B507F0   0000 0000 0000 0000
    :00B507F8   0000 0000 0000 0000
```

```
:00B50800   00A0 323D 000F 0BCB          AverageResults for each digitizer:
:00B50808   0000 1812 4390 2F33
:00B50810   00A0 323D 000E E042          numGood, activeCnt, sum(4)
:00B50818   0000 17CD 4117 6666          average(4), variance(4)
:00B50820   00A0 323D 000E FF9D
:00B50828   0000 17FF 4482 8E33
:00B50830   00A0 323D 000E EE68
:00B50838   0000 17E3 4402 F000

:00B50840   01E9 01D9 01E1 0000          listNums for wave, beam, offs
:00B50848   0000 0000 0000 0000          status for wave, beam, offs
:00B50850   4700 0000 4704 0000          Base addresses of digitizer memories
:00B50858   4708 0000 470C 0000
:00B50860   0000 0000 0000 0000
:00B50868   1FFF 180C 180B 180D          early 10 points of beam signal
:00B50870   180C 180C 180B 180C          pedestal is (rounded) average of last 8
:00B50878   180C 180C 0000 0000

:00B50880   003C 003D 003A 0043          beam present deltas
:00B50888   003D 003E 003B 003D
:00B50890   0040 003F 003F 003D
:00B50898   003D 003A 0041 0040
:00B508A0   0039 0040 0040 003F
:00B508A8   0041 0042 0039 003A
:00B508B0   003C 0042 003B 0040
:00B508B8   003E 0043 003E 003C
```

In this example, the number of good samples (beam above threshold) is the full 160. This listing only shows the first part of the static memory block. The waveform data is found beginning at offset `0x4C0` from the start of the block.