Booster Beam Charge

BCHG local application

Mon, Nov 24, 2003

To assist Booster monitoring, it is desired to compute the energy lost due to the measured loss of beam charge by analysis of the beam charge waveform. This note discusses the method used in developing the local application `BCHG` for this purpose.

It was initially thought that this new feature could be supported by adding logic to the `BLMS` local application, which processes the Booster Beam Loss Monitor waveforms, plus the beam charge waveform. But it soon became clear that keeping the new logic separate would be easier to organize and manage.

The idea is to compute a weighted sum of the beam charge lost each millisecond, where the weights reflect the heavier cost of beam lost at higher energies. Peter Kasper developed a table of weight values, in units of joules per 10^{12} protons, for each millisecond during the 33 ms Booster acceleration cycle. Multiplying by lost beam charge in units of 10^{12} protons, we get units of joules. To get a kind of efficiency, we divide by the initial beam charge signal. The formula for the `result` is the quotient of two sums:

```
energy = sum over i of the product (q[i] - q[i+1]) * w[i]
initCharge = q[0]

energySum = energySum + energy
qSum = qSum + initCharge
```

At the end of the update interval,

```
result = energySum/qSum
```

where `q[i]` is the beam charge waveform sampled value scaled to units of 10^{12} protons, `w[i]` is the weight at that energy, and `i` ranges from 1 to 33 milliseconds. Note that `q[i]` decreases as `i` increases, since beam charge can be lost during acceleration.

Insure that `q[0]` is sampled after beam injection. Also, insure that `q[33]` is sampled before beam extraction, lest it be interpreted as a major beam loss. The weight values range from 64 to 1281 as the beam energy ranges from 400 Mev to 8 Gev.

Parameters used in this LA as installed in `node06C6`:

Prompt	*Value*	*Meaning*
`ENABLE`	`00B2`	Enable Bit# for LA
`BCHG C`	`0206`	Beam charge Chan#
`INIT INX`	`0019`	Initial index in zero-based 12.5 KHz waveform array
`RESULT C`	`0350`	Initial result Chan# for clock event 11, others to follow
`PERIOD`	`001E`	Update period in seconds

Even for beam reset events, there are often times when no beam is present. There is an internal parameter `chgThresh`, in units of 10^{12}, for which an initial beam charge value less than the threshold produces a zero result value. This avoids dividing by zero or even noise.

The sums are to be computed separately for each Booster reset clock event. To facilitate doing this calculation, the following 32-byte context record is used, one for each clock event:

```
EvtRecType=
     RECORD
     iCharge: Integer; { initial raw beam charge this cycle }
     fCharge: Integer; { final raw beam charge this cycle }
     nCyc: Integer;    { number of cycles accumulated in sum }
     nLow: Integer;    { number of cycles without beam above threshold }

     energy: Real;     { energy lost this cycle }
     rChg: Real;       { final charge this cycle included in sum }

     qSum: Real;       { sum of initial beam charge each cycle }
     energySum: Real;  { sum of energy lost each cycle }

     date: DateType;   { date of last event }
     END;
```

In order to keep the record useful for diagnostic purposes, only the fields `nCyc`, `nLow`, `qSum` and `energySum` are zeroed at the start of each update period. One can therefore view the last initial and final beam charge raw values, plus the converted final charge as summed. The time shows the time of the last occurrence of the given event. Times are in the usual BCD format, making it easy to read in hexadecimal. The final byte is a binary value of half milliseconds within the indicated cycle during which time the LA executed.

The weight values are precomputed and stored in a data file called `DATABCHG`, making it easy to access when the LA is initialized.

Each 15 Hz cycle, when the present reset clock event has been determined, sample the `iCharge` value and compute the corresponding `rChg` value scaled in 10^{12} units. If `rChg` is below the threshold, increment `nLow` only; otherwise, increment `nCyc`, perform summation over acceleration cycle of charge lost each millisecond times the weight factor, giving the result `energy`. Accumulate the `energy` value into `energySum`. Also accumulate the initial charge into `qSum`. Whether the threshold was reached or not, set the `date` field to the current time-of-day. (The time associated with event `10`, however, is the time of the last beam event with beam charge above the threshold.)

At the end of the update period, build array of results for each reset event by dividing `energySum` by `qSum`. (If `nCyc` is zero, set result to zero.) Then update these floating point results into successive result channels. Use `SetReads` to target the reading fields in the `ADATA` table, using the full scale to produce the 16-bit result. Then reset `nLow`, `nCyc`, `qSum` and `energySum` to prepare for the next update period.

Just as for the case of the BLM support, the clock events in use are as follows:

`11, 12, 13, 14, 15, 16, 17, 19, 1C, 1D, 10`

Event `10` refers to any beam event, which is all of the previous events except `11` and `12`. For each beam event, a second context record is also managed using the same data. In this way, any beam event gets a separate result as well as applying to this composite result.

Initially, the `chgThresh` parameter is set, rather arbitrarily, to 0.1 E12.