

BLM Event Counts

Local application

Tue, Nov 23, 2004

There is a need to keep ongoing counts of the last 100 seconds of Booster reset events. This can be used to assist in analyzing the 100 second BLM accumulations including subtracting background effects. This note describes a local application that can do this. This is separate from the extensive LA called BLMS that is used to accumulate the loss monitor signals, in which each of 8 different front end nodes makes such calculations over a 100 second window of time that it determines independently. This new LA would do similar logic also independently. It is thought that independent windows of 100 second intervals do not matter much.

Imagine an array of clock events versus what was called partial sums in BLMS. If there are 10 clock events, including the composite event 0x10, and if there are 6 intervals of 17 seconds in each 100 second interval, then we need an array that is 6 x 11. There should also be a partial index number that advances through the range 0-5 every 17 seconds, or actually, every 250 cycles at 15 Hz.

Each 15 Hz cycle, determine the prevailing Booster reset event number by making calls to the HAVEEvt function, and set the event index. Also note whether it is a beam cycle for which event 0x10 applies. Increment the appropriate counter using the event index and the partial index. Also increment the counter used by 0x10, if that applies.

Every 250 cycles, after performing the needed incrementing, tally the sums into the result counts, one per clock event. Then advance the partial index, and clear the partial counts for all events at that index to prepare for new counting on the following cycles.

The result counts are then updated about every 17 seconds, and they reflect the event counts over the last 100 second interval.

Choose some names. The 2-dimensional array is eCounts[11, 6]. (Sequential words are indexed by event index.) The total results are called tCounts[11]. The partial index is called pInx, and the event index is called eInx. The flag for event 0x10 is beamEvent. the counter for 250 cycles is called pCount.

Assume as parameters we have PCYCLES, NPARTS, COUNTS C, where PCYCLES is the number of cycles to make a partial period of 17-ish seconds, NPARTS is the number of partial intervals, which is 6, and COUNTS is the initial channel number for the resulting event counts. The name of the LA is ECNT, and the description is "INTERVAL EVENT COUNTS".

Implementation

The above ideas were incorporated in a new LA called ECNT. It runs with typical execution times of 30 μ s each cycle, but with 110 μ s needed every 17 seconds, when the counts are tallied and the 11 result channel readings updated.