

Clock Event-based Requests

Front end interpretation

Thu, Oct 19, 2006

Data requests supported by the front ends can be periodic or clock event-based. Periodic requests are supported by periods that are multiples of 15 Hz, or 10 Hz, cycles. Clock event-based requests are also supported in units of cycles. The interpretation of such requests is the subject of this note.

Early in each operating cycle, which is triggered by a clock event-derived timer interrupt, each front end updates its data pool with fresh readings. Note that this update does not depend upon the nature of any active data requests; all data is refreshed. Once the data pool is updated, it builds and delivers reply messages for all active requests for which replies are due on the current cycle. Assuming a 15 Hz environment, then, a 1 Hz request would build and return replies every 15 cycles, beginning when the original request was received.

The timing of the operating cycle is usually delayed from the 0x0C clock event, which occurs on the accelerator timeline every 15 Hz cycle at the Booster reset clock event time. For Linac front ends, the delay chosen is 3 ms, or 1 ms after "Booster BMIN time," when the 50 μ s Linac beam pulse is accelerated and injected into the Booster. For Booster front ends, the delay used is 40 ms, which is a few ms after the Booster beam has been accelerated over 33 ms of time and extracted from the Booster. The idea is that the front end should commence its cyclic activity "after all the shooting is over," when all data measured is ready to be digitized.

Now consider the clock event-based request. The front end treats such requests the same way, building and delivering replies after the data pool has been updated and the selected clock event is "true." While clock events can occur at any time, most clock events of interest to the Linac and Booster are Booster reset events, which are 0x11–0x17, 0x19, 0x1C, 0x1D. For each 15 Hz cycle, one of these ten events will appear. In addition, a 0x10 event will occur for any Booster beam cycle, which is any of the Booster reset events except 0x11 or 0x12.

What does it mean for a clock event to be "true," as referenced above? In each front end, clock decoder hardware causes an interrupt for each clock event seen by the hardware. The interrupt routine records a μ s counter "time stamp" for each such clock event into the Event Times table, with entries indexed by event number, along with the elapsed time since the last occurrence of the same event. It also sets a bit in a 256-bit map to mark the occurrence of that event. Note that the hardware uses a FIFO, so that each event interrupt may process multiple events when they occur closely spaced in time. Early in each operating cycle, the software copies the above bit map, at the same time clearing it to prepare for marking subsequent events. The bit map copy is then referenced by software during the operating cycle whenever it needs to test whether a given clock event is "true."

It would be possible for a front end to interpret a clock event-based request literally; *i.e.*, when the clock event occurs, digitize the data of interest. But what would that mean for Linac and Booster, when the clock event specified is, say, 0x1D, the MiniBooNE Booster reset event? At any Booster reset event time, nothing is going on in either Linac or Booster. No Linac beam will occur for another 2 ms. If one wanted a reading of the RF gap envelope in Booster, that signal will not be available for tens of ms. The Booster reset event announces something to occur in the near future. For DC power supplies, or vacuum readings, it will not matter, but for anything relating to beam or RF, it matters, the data can only be collected later, not at reset event time. Because of this, the front ends interpret event-based requests as referring to data that will be collected during the following cycle that is announced by the reset event. This is the data sampled from the data pool that will be updated during that cycle.

The new GETS32 request protocol allows specifying a delay after a specified event. Given the above description, this delay could be interpreted in units of cycles, even though the delay is specified in ms. The data pool from which the readings would be taken is only updated every cycle. Quantizing the delay in cycles may be too limited for some cases.

Consider the case of the 1KHz digitizer used in IRM front ends. This digitizer automatically digitizes 64 channels and places the readings into a 64K-byte circular buffer each ms. The buffer allows for 512 sets of 64-channel readings, or about a half second of history. We could interpret a data request for readings from this buffer at an event plus delay as sampling the digitizer reading for the specified channel from the circular buffer. The code would only have to look back less than one cycle in time. But if we do this, what will happen for Booster reset event-based requests? With a zero delay, we would not get the result we are used to. We could interpret a request specifying a nonzero delay as sampling from the circular buffer, but with a zero delay as sampling from the data pool. Another refinement can allow sampling from the circular buffer even for a zero delay if the specified event is not one that occurs very near Booster reset event time.