

KHz Samples on Event

Local application

Fri, Dec 16, 2005

Main Injector is interested in sampling KHz data, destined for the data pool, when a clock event occurs. This note explores how that can be done using a local application called `KHZE`.

Event-based requests

The system code support interprets event-based requests in a special way. These front ends operate at 15 Hz in synchronism with the accelerator clock. When they begin their 15 Hz cycle of activities, the clock events that have occurred during the last cycle are captured for reference during this cycle's operation. Linac front ends begin operations 3 ms after Booster reset event time, which is 1 ms after Linac beam is accelerated. This means that the Booster reset event that identifies the current cycle is captured for reference during the current cycle. If an event-based request asks for data to be returned for a given Booster reset event, then it is copied out of the current data pool and returned during this cycle.

Note that if an alternative approach were taken, that of digitizing data upon the occurrence of a clock event, it would have no interest for Linac and Booster work, as all Booster reset events occur 2 ms before beam is accelerated through the Linac.

For Booster HLRF front ends, the 15 Hz cycle of activities begins at 37 ms delay from Booster reset time, which means that the entire Booster acceleration cycle is over, so that even signals that are only valid as late as Booster extraction are accessible; all data placed into the data pool will be appropriate for the current 15 Hz cycle.

Sample and Holds

For measuring data sampled at specific times, sample-and-hold circuits are used. In this way, many signals can be captured at once for correlated analysis. The Linac has beam pulses that last only 50 μ s, so it is obvious why S&H circuits are used there. In Booster, where the acceleration cycle lasts 33 ms, S&H circuits are also used, to ensure that all waveform data is sampled simultaneously.

Main Injector beam loss

The current signals of interest to the MI group are beam loss monitors that are actually used by ECool. These signals are digitized at 1 KHz, as are analog signals in nearly all IRMs. The system samples the most recent readings from the KHz hardware buffer early in the front end's operating cycle. These signals are designed to measure electron beam loss, but because of their physical proximity to the MI ring, they pick up transients from MI, whose widths are about 5 ms. The folks from MI are interested in measuring the size of these transients. They know about the transients, because the KHz data can be plotted at full KHz resolution using the Acnet Fast Time Plot protocol. But they would like to have the data sampled at the time of a specific clock event, in this case, event 7B, to be included in the 15 Hz data pool so that it can appear normally on an Acnet parameter page, or can be data-logged.

In this case, specifying an event-based request returns data that is sampled at the start of the front end's 15 Hz operating cycle, which could be as much as 66 ms after the selected event. If they used S&H hardware, the output would be digitized within one 15 Hz cycle.

Sans S&H hardware

What if it is not convenient to install S&H hardware? Is there an alternative stopgap method that can be used? Since the data is automatically sampled at 1 KHz by hardware, and since the IRM knows about the occurrence of clock events, one can imagine sampling the

appropriate data point from the recent hardware buffer. We would only have to look back no more than a single 66 ms cycle, so the 512 ms history in the hardware buffer is more than adequate. Note that there are compromises made here. The KHz digitization is asynchronous with anything, so there is a built-in millisecond jitter. Even if the sampled waveform were perfectly repeatable, each time it is sampled, it would be good only within 1 ms. With a transient peak to be sampled that is 5 ms wide, one cannot be sure of finding the actual peak sample. But assuming that such accuracy is not necessary, one can use an LA to do the approximate sampling.

Local application logic

To do this rough sampling, a local application called `KHZE` can note the time of the last given clock event interrupt, compared with the current time, and use the difference to reach back into the KHz hardware buffer to find a sample digitized sufficiently near the event time. The results can be written to dummy channels for use on an Acnet parameter page display.

Consider the following parameter set for the LA:

ENABLE	B	Usual LA enable Bit#
SOURCE	C	First KHz digitizer Chan#
EVENT	C	Specified clock event, Chan#
DELAY	C	Optional μ s delay after clock event, Chan#
RESULT	C	First result target Chan#
NCHANS		Number of consecutive channels

Allowing for a delay channel, the user can adjust the delay to find the peak of interest, without altering the timing of the clock event.

In detail, if the specified clock event occurred within the past 15 Hz cycle, grab the time stamp for the specified clock event from the Event Times table. Subtract that from the current time, yielding a difference that is not much more than 66 ms. Using the current memory address register from the KHz hardware, adjusted to a 128-byte data set boundary, back up by the time difference, assuming 1000 μ s per data set. (More precisely, use the average delta time computed automatically in the area just before the Event Times table.)

Another approach is to reference the time stamp array for each of the 512 data sets. From the initial Source Chan# given, compute the time offset for the digitization of that channel within the 64 channels at 25 μ s per channel. Find a time stamp that, when adjusted for this offset, most nearly follows the time stamp for the specified clock event plus the optional offset. Copy out the number of channels specified starting at this point in the hardware circular buffer into the reading fields of the target series of channels. The first method may be used to find a starting point for searching the time stamp array.

Implementation

The application was written using the first approach, but the current time was taken from most recent entry in the array of time stamps for the 512 slots. The delay channel is assumed to be in units of 0.1 ms, which means its fullscale value is 3276.8. This value, multiplied by 100, is used as an offset from the microsecond time stamp of the given event. The current slot is backed up by the number of slots represented in the time of the latest slot minus the event time stamp, and the data values are copied from that slot. One can adjust the delay channel setting so that the sampling is optimal. The number of channels is at most 64.