

Clock Event Monitor

Local application

Mon, Mar 24, 2003

Many clock events are expected to occur with dependable periods, especially those expected to occur at 15 Hz. A new local application called `EVTM` monitors the clock event log data stream ("`EVTLOG` ") to watch for unusual differences in the occurrences of such periodic clock events. This note describes the application.

The motivation for this comes from a problem observed in the calendar GMT time support maintained by the front ends. It is based upon the access to time-of-day from `chablis.fnal.gov` and by the fact the `0x8F` clock event occurs at calendar second marks. the local application `TIME` runs in `node0616` and multicasts the results every minute, including the GMT time of the most recent `0x0C` clock event, which normally occurs on every 15 Hz cycle. In between the one-minute updates, each front end manages its own updating of that GMT time every 15 Hz cycle, using its own crystal-driven microsecond counter which it corrects it by a long-term-derived correction factor based upon the succession of `0x8F` events that it observes. What has been observed is that, when the `GMPS` trips, causing a 15 Hz cycle to be missed, the calendar time updates are observed to lag for a period of time up to one minute. The one-minute clue suggests that the resolution of the lag occurs as a result of the next multicast update message from `node0616`.

An optional data stream can be defined in any system that logs the occurrences of every clock event. The event interrupt routine writes into this data stream, which means it will not miss anything, whereas task-level logic might. The scheme used by `EVTM` is to monitor the contents of this data stream every 15 Hz cycle and look for the differences in time stamps between successive occurrences of a clock event of interest. (This is easily done via a data request that uses `listype 50` to access the new records written into the data stream by the event interrupt code.) These differences are compared against a nominal-tolerance-specified window, and any differences that fall outside this window are logged to a circular buffer inside the LA's static memory. (It could have been written to a different data stream, but this approach seemed simpler for a quick hack.) The information recorded is the anomalous difference, the absolute value of the deviation from the nominal value, and the date and time of the occurrence in the usual BCD format that is good to a half ms within a 15 Hz cycle.

For the first installation in `node0509`, the following parameters are used:

<code>ENABLE B</code>	<code>00E7</code>	Enable Bit#
<code>EVENT#</code>	<code>000C</code>	Clock event#
<code>NOM MSW</code>	<code>0001</code>	Nominal ms word
<code>NOM LSW</code>	<code>046A</code>	Nominal ls word
<code>TOL MSW</code>	<code>0000</code>	Tolerance ms word
<code>TOL LSW</code>	<code>0400</code>	Tolerance ls word

Note that the nominal value of `0x1046A` = 66666 decimal. The `0400` tolerance is about 1 ms.