# SRM Status Alarms

*Problem analysis*

Tue, Jan 29, 2002

For some months, occasional alarm messages that signify missing responses from SRM data collection have occurred. This note describes how the cause was determined.

***Introduction***

Linac front-ends obtain a large portion of their controls data via Smart Rack Monitors, or SRMs, each of which connects to the actual hardware interfaces and acts as a front-end's front-end. One or more SRMs are connected via arcnet (modified token-passing standard) network to each Linac front-end, which resides on a vxWorks-based MVME-2401 PowerPC CPU board. Early in every Linac 15 Hz cycle, about 3 ms after the `0x0C` clock event, the front-end sends a special request message to all of its connected SRMs, the meaning of which is "read all your controls data and return it in a single arcnet reply frame." The SRMs do not know about accelerator 15 Hz timing; they key their cyclic activities to this request.

During the next 10–12 ms, the front-end waits for each SRM to deliver its reply, mapping the data received into the front-end's analog and digital data pool. A deadline is established in case one or more SRM replies is not received. If the deadline is exceeded, a corresponding software status bit is not set, which is monitored by the alarm scan. One Acnet device in each front-end is mapped to all such SRM status bits, so that missing replies produce an Acnet alarm message. It has been observed on occasion that such alarm messages do occur, and it is often the case that this happens at some time during an Acnet Big Save operation, in which all devices in the Acnet database are queried via one-shot requests and logged. This operation may take about 30 minutes for the entire accelerator, although only about 30 seconds elapse for collecting all Linac devices.

***Possible solutions***

We have sought to understand the cause of such SRM status errors. Is it because of a hardware or software problem in the SRM or in the arcnet network that ties them together? Is the deadline used for detecting missing replies too optimistic? Since that deadline is expressed as a time after the 15 Hz interrupt, can it be that some task activity delays the start of normal 15 Hz processing enough so that the request message is sent too late to give the SRM a chance to return its reply before the deadline? How does the Big Save operation affect the likelihood of such alarms?

The SRM reply deadline in several of the stations was set to 16 ms. Since the normal return of the latest SRM reply is observed to be about 12 ms, it was thought that this 16 ms deadline would be sufficient. If it were not, we would certainly want to know why.

***Meet the 4.5 ms anomaly***

It has been observed in recent months that some mysterious high-priority task activity occurs in the vxWorks-based systems. The initial indication of this came from routine timing diagnostics that each front-end supports for the execution time of each entry in the Data Access Table, which houses a list of instructions that are interpreted to update the data pool during each 15 Hz cycle. Many of these entries execute in much less than 100 us; an empty entry "executes" in 1 us, which represents the overhead of instruction interpretation and sequencing. Not only is each entry's execution time updated every cycle, but a worst case execution time for each entry is also maintained. It was observed that after a few hours and days, more and more of these short entries exhibited maximum execution times of 4.5 ms. This implied that something was occurring during this time that took time away from the expected activities in

that front-end. The system software is written so that no task-switching takes place during Data Access Table processing, so that a partially-complete data pool can never be seen. (This is part of what guarantees the ability to collect 15 Hz correlated data from the 15 Hz Linac.) The conclusion was that either a high priority task or a long-winded interrupt routine was to blame.

A local application called `DATM` was written to monitor the Data Access Table entry execution times, and if any was observed to be unusually long, a one-shot data request for recent task activity diagnostics was issued, the reply to which was logged. From the results, it was seen that a high priority task was the culprit. Its vxWorks task priority, where low numbers mean high priority, was 50. (System task priorities range from 110–140.) A simple check of the task listing, using the "`i`" command at a vxWorks prompt, showed that the vxWorks `tNetTask` is the only one associated with priority 50. Since `tNetTask` typically executes in 40 us, an elapsed time of 4.5 ms stands out like a sore thumb.

Another local application called `TSKM` was written to investigate this further. It monitored task activity diagnostics continuously to identify occasions when the priority 50 task execution time was measured to be about 4.5 ms, logging task activity occurring immediately before and after that time. The results were limited to occasions when another task was preempted by this one, since the `tNetTask`'s measured elapsed time cannot be relied upon when it starts at a time when the system is truly idle. The results showed that it was most often the `Update` task (including Data Access Table processing) that was preempted by a priority 50 task. Furthermore, it was clear that these occasions were not at all random in time, but appeared to occur every 5 seconds. But the 5 seconds did not appear to be synchronous with any known activity. It was not synchronous with the `0x02` clock event, which happens to occur every 5 seconds, nor was it synchronous with accelerator 15 Hz timing, which is itself synchronous with the line frequency.

If it is the `tNetTask` that is causing these 4.5 ms preemptions, then it would likely be associated with some kind of network activity, although nothing was recorded in the system's internal network diagnostics on this. To eliminate the possibility of some multicast-driven messages causing it, sensitivity to all multicast IP addresses was removed from the test node. The effect was still observed. A network Sniffer was attached to monitor all network activity occurring at the test node. But no network activity was observed to be occurring at a 5 second period, all the while that the `TSKM` local application was registering the same anomalous 5 second task activity. Investigation of the cause is ongoing. For now, we must assume its occurrence is a given.

***SRM status again***

Since there is a mysterious 4.5 ms time-out occurring at any and all times through the 15 Hz cycle, it may play a part in the occurrence of SRM status errors. In those nodes that used a 16 ms deadline time, when the last SRM reply arrives at 12 ms, the 4.5 ms problem implies that 16 ms is too optimistic. So the Data Access Table contents in all such nodes were modified to use a 24 ms deadline. The extra 8 ms should easily cover this problem. But what does this have to do with Big Save operations? The answer is that it doesn't, and occasional SRM alarms still occurred. So there may be more than one cause to the SRM status errors.

A third local application was written to watch for SRM status alarms, then capture recent task activity. It captured the last 87 entries in the task execution log, along with the most recent network diagnostics, monitoring several front-ends simultaneously.

The results showed a surprising effect. The `ACReq` task, which supports Acnet protocols such as

RETDAT and SETDAT, was sometimes observed to execute for 16 ms, and such occasions appeared to occur around the time of Big Save operations. Now a 16 ms execution time for any system task, except for the Update task while it awaits SRM replies, is shocking! A check of the RETDAT log in Linac node0600, which as the designated Linac data server receives nearly all Acnet RETDAT requests, showed no request was received at this time. But the network diagnostics in the affected front-end did show something was received. That only meant it was not a RETDAT message. ACReq is in the path for handling all Acnet protocols. The most likely after RETDAT and SETDAT is FTPMAN, which is used for collecting data for supporting "Fast Time" plots or Snapshot plots.

One FTPMAN message type is used prior to requesting plot data in order to determine what kinds of plots can be supported for a set of devices. The set size is normally limited to 5 devices. One can imagine this to be a simple message type to support. But although it is simple, it has a problem in the PowerPC implementation.

***CINFO system table***

The CINFO table houses information about analog channels that is used only by a small subset of channels, so that it is not included in the channel-indexed array of records that is the ADESC system table. For each CINFO entry, the channel number to which it relates is included in the entry, so that a search is needed to find a match on given channel.

So why should this matter? How much of a search can it be, and how significant can it be, given that the PowerPC is a 233 MHz CPU? The reason is the speed of access to nonvolatile memory, where most system tables, including CINFO, reside. Every access to nonvolatile memory requires about 1 $\mu$s, which represents more than 200 wait states, to put it another way. For this CPU, it is very slow. What's more, accesses to this memory cannot be cached. The memory resides on a PMC board plugged into the PCI bus on the CPU board. Because of this slow access time, we recently rewrote the Alarm task logic so that it uses the least number of accesses to this memory during its scan, with the result that a complete alarm scan in the worst case front-end is now performed in less than 1 ms.

The CINFO table was made large in the PowerPC systems, since they were new installations, and we weren't sure what needs the future would hold for them. Each CINFO table was installed with room for 512 eight-byte entries. Actually, entries in this table can occupy one or more of these entries, but 8 bytes represents the granularity. A search of this table is done based on an entry type number and a channel number. (The code was not written in a way that recognizes the slow access time of nonvolatile memory.) The search logic scans through each entry looking for a match on the type of the entry and on the given channel number. To get to the next entry, it must take into account the size of the current entry, which is a field in the entry. So, it might require 3 accesses per entry. With 512 entries to search, for a channel that has no entry, it might take 1.5 ms for one search. But one search may not be enough. Using different type numbers, additional searches may be performed.

After adding a diagnostic to the TSKM local application to keep track of the worst-case task execution times for each task, it was found that a query of the type above for one channel, for which no CINFO entry existed, required about 4 ms. If one asked for 4 devices, none of which had a table entry, one might expect that 16 ms would be needed. This is probably what was happening to cause such long ACReq task execution times. As to why such requests could occur for devices that have no plot capability and therefore have no entry in the CINFO table, it is just something that is routinely done during a Big Save, even though no plotting of such data may be contemplated.

As to why such unusually long task execution times can cause SRM status errors, one must bear in mind that such requests can arrive at any time during the 15 Hz cycle, including times near the end of the cycle. If `ACReq` is busy, it can delay the normally prompt start of the `Update` task, so that the request for SRM data over arcnet is delayed, and the arrival time of the ensuing replies can extend beyond the deadline.

***Solutions***

Part of the solution should be to rewrite the logic that determines what plot capabilities a channel has so that it is optimized for the least number of accesses to nonvolatile memory. But what can be done immediately is to reduce the size of the CINFO table. The vast majority of front-ends have no entries installed in that table. By modifying those nodes so that the CINFO table size is small, the inefficiencies that exist in the current searching algorithm will not be so disastrous.

As soon as this was understood, the CINFO tables were reduced greatly. Because it was the simplest and quickest thing to do, late on a Friday afternoon, those front-ends with empty CINFO tables were set up with only two entries, rather than the default 512 entries. Those few that had entries installed had their declared sizes much reduced. A simple test was made to measure the time of a request for a single device in a node that has two entries, with the result that only 200 us execution time was needed in the `ACReq` task. In the worst case node, with 96 entries in its CINFO table, such a request for 4 devices might take 3 ms. But with a rewrite of the logic, this can be greatly reduced.

It is hoped that this remedy will nearly eliminate SRM status alarms, except for cases in which something is really broken or the SRM is down. If more mysterious SRM status errors occur, however, additional tests will be made to find the cause.

We need to keep in mind the limitations brought about by slow access to nonvolatile memory. There may be other similar inefficient processing yet to be discovered, in which the logic was not written to anticipate such slow accesses.

The achievement of reliable 15 Hz performance in a front-end is not free. It requires careful monitoring of task execution times. Several timing diagnostics are included in Linac-style front-ends that can help verify consistent 15 Hz operation.