

Distribute Spark Counts

Exploring options

Wed, Jul 9, 2003

Booster RF systems measure spark counts in the RCC station, even though these counts relate to each individual station. This note explores how the RCC node can share this information with the individual station nodes, with the chosen solution described.

Requests from stations

Suppose each individual station requests the appropriate spark count from the RCC. This might also be supported via the RFBS local application that does everything else. There should be two parameters added, one to hold the RCC node# and the other to hold the channel number in the RCC that houses the spark count needed. There is a question of how often the data should be returned. If it were at 1 Hz, it should be enough for the purpose of seeing a count advance very soon after a spark occurred. It would mean that the RCC node would be delivering 18 replies per second 24/7. We don't yet have support for requests for which replies are only sent when data changes.

Delivery to stations

Suppose we turn it around and have the RCC send the spark counts to the other stations as a kind of setting. In this case, the RCC could send the update whenever there is a spark, since it knows when any of the counts changed. Perhaps it should send them also every minute, say, just to be sure a node that has just reset has the correct counts.

What kind of setting would be sent? One possibility is 18 separate setting messages, each of which targets a specific channel number in the station to which it sends the setting. This sounds a bit like the requests-from-stations scenario above, except that these settings only need to be sent very infrequently, unless there is a spark count. In fact, only the node whose spark count just changed would need to be updated via a new setting. If desired, this updating could be quite prompt, even delivered within the same cycle of the spark.

Suppose the setting were to a memory area that is generally available in all nodes of the project, plus any node that listens to alarms by enabling reception of the project multicast address. Then only a single setting would be made that would carry a kind of mini-pool of data including the spark counts of all stations. A data access table entry in each node would select out its own spark count for assignment to a given analog channel, which would likely be the same in all nodes. Again, this memory setting would not have to be made very often, since the sender knows when any node accumulates a spark count. Still, it would be useful to send an update occasionally, in case a node has just booted.

To make the spark counts locally viewable, the local application that supports the 4-line local console would need to include the new spark count channel in its page repertoire.

Implementation

The chosen solution is to target a dummy analog channel in each individual station for which the RCC counts a spark in the local application RFBS. The Data Access Table in each station includes a new entry that copies that value to the reading of a new spark count channel. The local application (HLRF) that supports the 4-line local display allows displaying the reading of the spark count channel, which provides the local display of that station's spark count. The dummy channels are named SDxxSS, and the spark count channels are named SDxxSC, where xx is the station number in the range 01–18. The reason for using the copy logic from a dummy channel was to insure that the spark count channel that is viewable is not settable.