

Time-stamped Data

New listype for 15Hz data

Tue, May 23, 2000

In order to provide time-stamped access for RETDAT-based data requests, a scheme must be developed that includes time values along with the data values. This note describes a scheme for returning the most recent 15Hz data as stored in a channel-indexed table. The number of points saved depends upon the size of each element in that system table. A Data Access Table entry is used to copy readings from the data pool into this table. One must use a table so that the reply data can be updated in a single time. This makes it possible for GetDat calls to work, in which only a single call is made when the user wants the reply data copied into his buffer. (One may argue that GetDat does not need to work properly for this listype.)

4-byte ident format

node
chan

Internal pointer format

chan
#cycles

The high byte of the #cycles word may be an offset into the ADATA table entry, so that the same read routine could service other listypes that want to provide time-stamped access to fields other than the reading. Doing this would imply copying into another table, which may not be likely. To capture setting values, one may rather want to copy settings into readings for some range of channels. It is likely that only readings would need to be accessed in this way. Since digital status that Acnet reads is maintained as combined binary status words that are kept as channels, access to digital status is covered, too.

Reply data format example (12 bytes needed for 4 points returned)

#points=4
time (of first data value)
data
data
data
data

The "read routine" knows the next index, which is the advanced index of the present data, that will be used on the next cycle for copying data pool readings into the new system table. It copies the number of points into the buffer as given by the second word of the internal pointer. The last data point in the buffer is always the most recent one.

Since an immediate reply is given to a new data request, the second reply to a periodic request may include data that was already returned. This does not matter, since the time stamps are correct in both cases.

To filter based upon clock events, one must include in such a request a device that gives the time-stamped event#. If more than one device is requested in this way, all devices will use the same initial time stamp, but that's life.

To ask for 15Hz data from a Vax console, ask for 8 bytes of data to be returned at 7.5Hz. This will give two 15Hz data points with an associated initial time stamp. The second time stamp is,

of course, one more than the first. If one makes a 1Hz request for 15Hz time-stamped data, request 34 bytes from each device. This gives room for the 4-byte header and 15 data values.

A 6-byte request will return one time-stamped data point, the most recent one, in the generic format.

A 4-byte request can also return the most recent time-stamped data point, in the form of (time, data), the most recent one, using an abbreviated format.

A 2-byte request can return the latest time value, or cycle counter. To ask for the latest data value, simply use listype 0.

This scheme is not for requesting data measured at rates slower than one cycle. The usual approach will work for that.

Note that one must change the SSDN to use the new listype to get time-stamped data. This may mean that users will devise special names for this purpose. Of course, the application code to process such data is quite different. But the old names cannot work, if they are also to service the parameter page in the usual way.

Many uses of SSDN for analog channel

A given Acnet device can have only one SSDN that relates to the reading property. But there are many variations on what kind of data is available for a given device.

```
00f1 node chan 00sz
```

A typical example might be

```
0011 06D3 0119 0002
```

The `f` nibble is the offset flag nibble. If it is 1, it allows a requester to access the reading of a different channel (sum of channel number + offset) via the offset word. (This feature was included in order to allow a single Acnet device to serve as a base for access to many similar data fields, especially analog descriptors. In cases where this feature is used, the channel number used would likely be 0000, so that the offset word can mean the channel number for which data is sought.)

If the offset flag nibble is 0, the interpretation of the offset word is a byte offset, which is not now supported. But it could be used to access elements of a waveform, assuming that one exists that is related to the given channel number. The `CINFO` table can be searched to determine whether a waveform exists.

Independent of the offset flag nibble value, if the `sz` byte is nonzero, one can access successive channel readings when the #bytes requested is more than 2. As an example, this feature is used to access HLRF trip logs, in which the trip logs are updated by the local application by writing to the reading words of consecutive channels. This scheme has an advantage in that the trip counts are maintained in nonvolatile memory, so that the trip log record is not lost if the IRM is reset.

New methods of access

It is attractive to consider finding a way to access multiple values related to a single channel. One is `RETDAT` access to an analog channel waveform. Although it may be measured

with completely independent hardware from that normally used to update the data pool, the hardware signal being measured is the same. It would be convenient to be able to use the same Acnet device name, and therefore the same SSDN.

It may be useful to allow access to individual points of a waveform. This would be needed by Booster BLMs in order to access readings of the entire suite of loss monitor waveforms at the same time--something like a software sample-and-hold facility.

Another type of related data for an analog channel would be the most recent 15Hz readings. This may be needed to provide reliable 15Hz time-stamped data for an Acnet console application. Again, it would be convenient to use the same device name.

The question is how to allow the RETDAT server support to correctly guess the actual meaning of the user's request. The variations under control of the application user of RETDAT, perhaps via the DIO library, are the reply period (Frequency Time Descriptor), and the offset and length words.