

SSDN Introduction

Informal description

Robert Goodwin

Mon, Apr 11, 2011

The SSDN, or SubSystem Device Number, is used in requests for Acnet device-properties to specify, in front end terminology, what data the client seeks. This note is an introduction to the 8-byte SSDN structure used in front ends that support Classic protocol, whether 68K-based or PowerPC-based. More detail is found in the notes, *SSDN Nuances*, *SSDN Formats Used in IRMs*, and *RETDAT-GETS32 Data Options*.

The design of the SSDN described herein is fundamentally based upon the Classic protocol, the data request support that exists in the front ends. An underlying referencing terminology is based upon the abstract notions “listypes” and “idents,” two terms that I coined. The true significance of an ident is only known in the context of a given listype#. See the note, *Listypes and Idents*. Support for both Classic and Acnet protocols is based on these notions; satisfying a data request involves the same low level software. In each case, the request is initialized to derive an “internal ptr” from each ident, which is used in fulfilling the request to produce reply data. The idea behind this is that a data request is initialized only once, but it may have to be fulfilled many times. The internal ptrs expedite request fulfillment. A set of “ptr-type” routines create the internal ptr, depending on the listype#; a set of “read-type” routines produce the desired reply data from the internal ptrs. Within the Acnet request support are included several Acnet-specific options.

Analog reading example

As a simple example, consider a request for analog readings, perhaps the most common type of data requested. An 8-bit “listype#” (0x00) is used with a Channel# “ident” made up of two 16-bit values in the form of (node#, chan#). The node#, a value in the range 0x0500–0x07FF, is unique among front ends. Its significance is known to the DNS by the name node0xxx.fna1.gov. The chan# is in the range 0x0000–0x03FF in most nodes, but some support a range 0x0000–0x07FF. The chan# is really an index into the nonvolatile memory array of 8-word structures that house analog channel-related info, specifically reading, setting, nominal, tolerance, alarm flags, alarm count, and two additional words, so this ident format merely refers to an ADATA table entry in a given front end node. The SSDN format for this Acnet device reading property is merely

```
0001 node chan 0000
```

The Channel# ident is the 2nd and 3rd words of this layout, with the listype# 0x00 in the hi byte of the 1st word. The request specifies a length of 2 bytes and an offset of 0 to get the analog reading.

Note that there is nothing in this SSDN format that refers to the source of the analog reading. There is no reference to hardware at all, but only to a table entry in a given node. Obviously, the reading of the channel comes from *somewhere*, often a digitizer, but that is specified via other nonvolatile memory tables that are part of the front end’s configuration. It is assumed that on every 15 Hz operating cycle, the “data pool” is brought up-to-date with fresh readings that are deposited into the reading fields of the appropriate ADATA entries. Satisfying a request for an analog reading merely turns into accessing the specified ADATA entry that always includes the latest analog reading; no hardware is accessed. (Such easy access to readings assists the alarm scan, too.)

The above example is one of the simplest. Consider a few more cases. An analog setting reference looks the same, except that the listype# is 0x01. It access the setting field of an ADATA table entry. Used in an Acnet setting message, it causes a hardware setting to occur, the details of which are specified in other nonvolatile tables. But let’s concentrate on data requests here, not settings.

For an analog alarm block property, the listype# is 0x02, again with the same Channel# ident.

For the basic status property, which allows access to a collection of status bits, the same listype# 0x00 is used, because the status word occupies a reading field in an ADATA table entry. To distinguish such status bit pattern readings from numeric value readings, a special alarm flag bit is set. (This is obviously important for the alarm scanning logic.) Again, other nonvolatile tables specify how to access the hardware and even how to create the desired bit pattern.

Generic SSDN breakdown

Beyond the simple cases described above, the 4-word SSDN can be laid out in fields:

Word	Mask	Meaning
1	FF00	Listype#
1	00F0	Offset option# = 0, 1, 2, 3
1	000F	Ident size = 1 or 2
2	FFFF	Node#
3	FFFF	Chan# or other word index
4	00FF	Item size when ident size = 1, when needed

The above layout refers to cases in which the Ident size = 1, which means the ident consists of the node# following by a one-word index. If the Ident size = 2, words 3 & 4 hold the two-word index that follows the node#, and there is no item size field. (Think of a memory address, for example.)

Offset options

The Acnet request protocol can specify a length and an offset. In Acnet, the offset field is considered a byte offset into the data array specified by the SSDN. In some cases, this byte offset is supported in these front ends, but in many cases, it is not. Offset options 0 and 3 ask for such byte offset support, when it exists. Option 0 is used when accessing a waveform array of data points. An example would be an analog channel that is known to the front end (via another nonvolatile memory table) to have not only a 2-byte reading, but an entire waveform as well. If a request specifies a suitably long length, the request accesses part of the waveform, and the offset allows access to an arbitrary segment of that waveform. (Acnet does not support a waveform property with its own special SSDN. The reading property SSDN must be used.)

Offset options 1 and 2 are used to specify how to modify the ident in the SSDN. This was originally designed to allow for a single Acnet device's SSDN to access any table entry, in order to facilitate some kinds of system management. In the case of option 1, the given offset field is used to modify (by addition) the ident in the SSDN. For a simple (but probably not useful) example, consider being able to access any ADATA raw reading value. One could use the following SSDN:

```
0011 node 0000 0000
```

The user would then specify any offset desired, and the request support in the front end would add that offset value to the base channel# 0x0000 before interpreting the SSDN. An Acnet client cannot modify the SSDN; it is taken directly from the database and placed in the request message. This scheme offers a way around that restriction for specific applications.

Offset option 2 is like option 1, except that the offset field is shifted left 8 bits before being added to the ident, and only 2-word idents may apply. In RETDAT, the offset field is only 16 bits, so this was a way to use a single SSDN to access up to 16 MB of memory, say.

Offset option 3 is new. Its intent is to help allow Acnet parameter page access to bracketed array

elements. The parameter page uses an offset value that is a multiple of the fundamental data size for the device. One type of array device supported is an array made up of a series of consecutive channel readings. With option 3, and the item size byte set to 2, a request could allow some multiple-of-2 offset value to select an element of such an array. Option 3 support divides the offset by the item size before adding the result to the ident in the SSDN.

Consecutive channels

Asking for analog channel data by specifying a large length, assuming a nonzero item size, usually 2, results in a request for consecutive channel readings.

Data averaging

There is no way to specify that data should be averaged by what is in an SSDN, but automatic averaging of 15 Hz data is provided for slower periodic requests. The problem addressed here is to average beam pulse data preferentially, since if one averaged readings from all cycles, the beam cycle readings would be diluted by the readings from non-beam cycles. Each front end knows about beam cycles via a fixed beam status Bit #0x009F, which is zero for beam pulses, or one for non-beam pulses. For a typical 1 Hz periodic request, for example, the average of all beam cycle readings is computed. If there were no beam cycles, then the average of all cycles is returned. The original reason for interest in average values is to use all the data available. As all data is read on every cycle, it is readily accessible. Note that only periodic requests get averaging support.

Cycle data

It is possible to access 15 Hz data at slower rates. (Some early client computers did not want to accept 15 Hz replies.) The front end keeps readings of the last 32 cycles for each channel. If a periodic request at 1 Hz is made for 32 bytes, the reply data will be 15 readings from consecutive cycles, followed by a cycle# "time stamp" for the last cycle. The cycle# is derived from the cycle counter that is included in the Acnet multicast events message; hence, it can be used to assist in correlating such readings from multiple front ends. With GETS32 now in common use, it is probably better to simply ask for 15 Hz replies, if that is what is desired.

IRM/HRM digitizers

Special support is added for access to digitizer data, whether the 1KHz IRM digitizers or the 10KHz HRM digitizers. It makes use of the hardware circular buffers that hold the last 0.5 sec (IRM) or 1.6 sec (HRM) digitized samples. The listype# is 0x52 for this case, and the ident uses 2 words after the node#: the chan#, and a clock event#. The SSDN looks like this:

```
5212 node chan 0000
```

The offset can specify a user-chosen clock event used as a base for the sampled data time stamps. The data returned includes an 8-byte header, followed by (time, data) pairs of words. The header includes a 32-bit base time, relative to the last clock event occurrence. The time words are offsets from that base time. All time values are in units of 10 μ s. The sample rate is indirectly implied by the specified period of the request and the data length requested. The idea is to fill the reply buffer with fairly evenly spaced samples to bring the user up-to-date.

More...

For more on such related matters, browse the notes on the web starting at the Fermilab Accelerator Controls Department web page:

```
Integrated Engineering > Software > Local System > System RETDAT
```