

# Floating Point Raw Data

*System option*

Mon, Jun 19, 2000

Calculated results may be suitable for expressing in floating point form in order to preserve a wider dynamic range than can be provided with a 16-bit binary data word. This note discusses the changes necessary to support such data values.

The present analog data pool for each analog channel is composed of a 16-byte `ADATA` table entry that houses 16-bit values of readings, settings, nominal values, tolerances, and alarm flags. Suppose there is a parallel table called `FDATA` that also contains 16-byte entries that provide room enough for 32-bit floating point reading, setting, nominal and tolerance values. Each table is the same size, and the channel number is the index for each.

During the analog alarm scan, the alarm flags field is checked for each analog channel in all `ADATA` entries. If a (new) `FLT` flag bit is set, it means that the reading is available in the `FDATA` table in 32-bit floating point form, not in the `ADATA` table in 16-bit integer form. In such a case, the alarm check is made using the reading, nominal and tolerance values of the `FDATA` table entry.

In reporting the alarm via Classic protocol, the floating point reading value can be stored in place of the reading and setting words. Since the alarm flags word is included in the alarm message, this situation can be detected by any program that analyzes the alarm message.

There are no scale factors associated with floating point values in the `FDATA` table. These raw floating point values are always in engineering units.

To obtain 32-bit floating point readings, the client must use a new listype, one that takes a channel number ident and produces a reading from `FDATA` rather than `ADATA`. Another new listype can provide access to the setting field. A third new listype can access both the nominal and tolerance fields. A program such as the general purpose parameter page can request data from both tables, using the alarm flag bit to determine which contains the data. As another approach, the meaning of listypes 40, 41, 42, and 43 can be enhanced to detect the alarm flags bit that denotes floating point raw data and return the value directly from the appropriate `FDATA` table entry. This would give the possibility of requesting floating point data using these listypes and get results from whichever table is appropriate. The disadvantage of this approach is that 16-bit raw data would not be obtained. Perhaps both approaches can be supported.

For local applications that call `ChanFlt`, that routine may be changed to test the new alarm flags bit and return the scaled value from the 16-bit reading, or the contents of the 32-bit floating point value in `FDATA`.

The significance of the alarm flags `FLT` bit is not strictly related to alarms. One must set the bit for any channel that supports floating point raw data, independent of whether it is enabled for alarm scanning.

The `SETAFLG` routine in the `SetType` module must be changed to preserve modification of the new `FLT` alarm flags bit from change by Acnet analog alarm block settings. This bit will need to be set locally as part of the configuration of that node. To that end, it can be changed via Classic protocol.

A means of setting nominal and tolerance values via the Acnet protocol SETDAT must be provided. Doing this will mean that two internal settings must result from a single alarm block setting. One is for setting the alarm flags word, and the other is for setting the nominal and tolerance floating point values for the FDATA entry.

Performing a floating point setting to a FLT-marked channel will only result in updating the floating point setting value in the FDATA entry. A local application that monitors the change can make any required adjustments to hardware. It may be the same local application that produces the floating point raw readings. When the local application initializes, it should probably ask for the latest values of the floating point settings of its channels and do whatever it takes to “set” them.

The FLT bit in the alarm flags word is bit 12, with the mask 0x1000.

Listypes suitable for supporting access to the fields in the FDATA table are 90, 91, 92, 93, and possibly 94, for the delta setting. These, then, will be analogous to listypes 40–44. Listypes 88 and 89 are skipped to make this easy-to-remember assignment. Listype 88 was tentatively assigned to support Classic access to time-stamped data via the new CYCLE table that holds, say, the last 16 sets of 16-bit readings for each channel. Will there have to be time-stamped access to floating point raw data?