

Floating Point 15 Hz Samples

Addition to Cycle table support

Mon, Dec 13, 2004

The recent note, *Recent 15 Hz Samples*, describes the support for access to 2-byte analog channel readings sampled from the 15 Hz data pool. This note describes an addition that provides similar access to recent floating point channel readings that can come either from scaled `ADATA` readings or from `FDATA` raw floating point readings.

The former scheme results in reply data that is an array of 2-byte values, beginning with the most recent sample from `ADATA`, followed by such samples from ever-earlier cycles, and ending with the cycle number of the most recent reading. The support described here returns reply data that is an array of 4-byte floating point values, where the first one is the most recent sample from `FDATA` in engineering units, followed by ever-earlier samples from `FDATA`, and ending with the cycle number, expressed as a floating point unsigned value. For channels in which `FDATA` entries are not valid, namely those for which the `FLT` flag bit is *not* set in the analog alarm flags word, the returned floating point values are scaled values of the `ADATA` samples.

In practice, very few raw floating point channels are in use. Each is indicated by setting the `FLT` bit in the alarm flags word in the `ADATA` entry for that channel. Consider using the `cycle` table entries for the raw floating point channel readings, with the advantage of not having to find space for a new `FCycle` table that is only sparsely occupied. We then have to accept keeping only 16 floating point values rather than 32 integer values. To make it palatable, we cache the `FLT` bits of the analog alarm flag fields, keeping them in a new bit map that uses 1 bit per channel. With 2K channels, we require a 256-byte bit map. The inner copy loop checks a bit for every channel, to avoid overwriting the previously captured floating point values with unused integer values. The former bit map used for the integer case keeps one bit for every 16 channels, in order to optimize the copying action to used channels (those with a name in at least one of the 16).

When fulfilling a request, the `CYCLEXB` value, an even number of bytes offset in the range 0–62, or 0–0x3E, is used in a slightly different way for caching the `FDATA` values. If we take this value, shift it left by 1 bit, and `AND` it with 0x3C, we have an offset in the range 0–0x3C that is suitable for storing 4-byte values. Obviously, this modified index “wraps” twice as often as that used in the integer case, about 1 Hz rather than 2 Hz. Note that this assumes a 64-byte size of a `Cycle` table entry.

The bit map keeps the `FLT` bits organized as 16-bit words. If the original bit map has a one, meaning that 16 channels should be copied, the new bit map word can easily be checked for having any of its 16 bits set; if any is set, code like the following skips the copy when appropriate.

```
CYCLEL      LSR    #1,D7
            IF# CC THEN.S    ;copy only if cached FLT bit = 0
            MOVE (A0), (A1)
            ENDIF#
            ADD   D1,A0
            ADD   #CYCLENB,A1
            DBRA D0,CYCLEL
```

If desired, the above loop can be replaced with the simpler loop if the bit map word of 16 bits is known to be zero before entering the 16-word copy loop. The result of all this is to avoid overwriting the `Cycle` table entries for raw floating point channels.

Although one might try to combine the integer and floating point cases into the same copy loop, it may be easier to perform a separate scan through the new bit map to do the copy of the `FDATA` values.

It is a good idea to try to keep up-to-date the bit map that denotes channels for which the `FLT` bit is set. If we update one 16-bit word of the `FLT` bit map every 15 Hz cycle, by monitoring the `FLT` bits for 16 consecutive channels, it takes about 4 seconds to update all 1K channels, or 8 seconds to update 2K channels. Maybe this is good enough. To be safe, it is probably a good idea to clear the `Cycle` table entry of a channel for which the `FLT` bit has just been found to be set. This ensures that the contents of that entry do not contain invalid floating point values, since until the time that the `FLT` bit is set, the entry would have been filled by integer values.

Armed with the array of 64 words of 16-channel bit maps, assuming a 1K channel node, we have some help during the copy operation that is performed for each raw floating point channel each cycle. We scan through each bit map word looking for a nonzero bit map, then scan through that group of 16 channels looking for a bit to be set to cause the current `FDATA` reading value to be copied into the `Cycle` table entry using the modified index offset described above.

Call the new bit map that is organized into 16-bit words the `FLTMAP`. It needs space for an array of 128 such words, in order to cover the 2K channel case. (The earlier bit map, located in the Name Table header, was called `NTBMAP`. It was arranged as bytes, each of whose bits represent groups of 16 channels, for which at least one is actually in use; it only requires 16 bytes for the 2K case.) One place that is available is the 256 bytes of low memory based at `0x2E00`.

Request support

Listype 89 is used to support floating point access to `Cycle` data, either scaled values taken from `ADATA`, or raw floating point values taken from `FDATA`. (Listype 88 is used to access 2-byte integer readings from the `ADATA` table.) The read-type# is 33, and the ptr-type# is 49. The internal ptr format consists of 4 long words:

- ptr to `Cycle` table entry
- ptr to `ADESC` table entry
- ptr to `FLTMAP` word
- original channel ident

The read-type routine uses this structure to generate the reply data efficiently. The low 4 bits of the channel ident are needed to sample the correct bit of the `FLTMAP` word that represents the cached `FLT` bit for 16 consecutive channels. If this bit is 0, integer `Cycle` table values are converted to engineering units using the scale factors from the `ADESC` entry. If the bit is 1, raw floating point values are copied from the `Cycle` table entry.