

# CINFO Search

*New scan logic*

Mon, Feb 4, 2002

The current search logic used for the function `CINFOEntry`, used extensively by the `LOOPFTPM` local application (supporting `FTPMAN` protocols) was not written to anticipate that memory access would be slow. But the `CINFO` system table is housed in nonvolatile memory, and the PowerPC front-ends experience slow, non-cacheable, access to that memory. Each access requires about 1 us, which is not so fast for a 233 MHz cpu.

The code that searches the table is not optimized for slow memory access. Each entry in the table can be of variable length, and each includes a length field, a type field and a channel number field. The `LOOPFTPM` code calls the search routine several times searching for a match on both type and channel number. For each search, three accesses are made to fields in each entry. The `CINFO` system table was defined as having a maximum of 512 eight-byte entries. When a request is received that asks, "what plotting support exists for the following 4 channels," we have found that 16 ms can be used up in answering this question. Out of a 66 ms cycle, this is a very serious perturbation. It has in fact been observed to produce alarm messages because of interference with other system code.

As soon as this inefficiency was understood, we vastly reduced the declared size of the `CINFO` table, since most nodes had no entries in it at all. But we need to consider how this logic might be rewritten to perform more efficiently in the future.

If the logic were kept the same, but the memory were faster to access, there would not be a problem. To that end, one could imagine copying the `CINFO` table contents into volatile memory upon initialization of `LOOPFTPM`, and perhaps periodically after that. But we may have to modify the contents of `CINFO` on-line, and one would not like to have to wait for a reboot before testing with the new contents.

Another approach might require that entries in that table be contiguous, without "holes," so that a search could stop as soon as a null entry is reached. This might make modifications of table entries slightly more tedious, but it is probably workable.

Included in an improved search algorithm can be a single long word (32-bit) access to the length, type, and channel number fields, rather than 3 separate accesses. Given the layout, this can work; those three fields comprise the first 4 bytes of an entry, and the size of each entry must be a multiple of 8 bytes. (Only lengths of 8 and 16 bytes have been used so far.)

Another variation of the algorithm can be that a single search is made for a channel number, and the first match "wins," returning the matching entry. It would make more difficult searching for the second occurrence of a channel number, say, one that uses a different type. Perhaps this scheme could return multiple matches for a given channel number. There is one case in which this will not quite work. For an entry that specifies that 1 KHz digitization is supported, only the first channel number (0x0100) is present, rather than all 64 channels being installed as separate entries. If such a search is to be made after the usual multi-search, a second attempt must be made using the base channel number.