

# Nonvolatile Tables

*Heavy usage candidates for optimization*

Fri, Jul 19, 2002

## ADATA

Normally not searched, but one DAT type copies readings into a spare field for the entire table. This job requires 3 ms for a system with 2K channels allocated, but it is not enabled in all nodes, so it may not be important. If it were, the copies could be stored into volatile memory. Alarm scan only examines active entries.

## ADESC

Not searched. Name table lookup already uses a hashing algorithm.

## BALRM

Not searched. Alarm scan only examines active entries.

## BDESC

Not searched.

## RDATA

Processed every 15 Hz cycle. It may be useful to create a copy of this table at reset time in volatile memory. But since this table is normally modified via the Memory Dump Page, one has to take care to be able to notice changes and update the volatile memory copy. See note on this.

## BBYTE

Updated every 15 Hz cycle. There is probably not much to be done about this without checking each time whether the updated value is a change. But it may be difficult to track down all updating accesses. (The current time for the entry in the RDATA table that updates this table is about 1 ms.)

## PAGEP

Seldom referenced, except when calling up a new page application, or rewriting the Index Page list of titles.

## PAGEM

Referenced by page applications, but not normally very often. The Auto-page feature may check all related fields every minute.

## LISTP

This is used when obtaining a list number for use with data request messages. It also is used for looking up where a request block is located given a list number. There is no reason why it must be searched. Still, it may be easiest to move this table into dynamic memory, as its contents need not be preserved across system resets.

## CODES

Searching this table for LA execution addresses has already been dealt with for the case of working from an LATBL entry, so that the first CODES entry examined almost always matches.

CDATA

Seldom accessed. Alarm scanning only examines active entries.

BADDR

All entries read every 15 Hz cycle. An update may take more than 1 ms for a 2K-channel node. A copy could be kept in volatile memory, but care would have to be taken to learn about updates, as they are normally done via the Memory Dump page.

OUTPQ

This table is already in volatile memory.

PRNTQ

This table is already in volatile memory.

LATBL

This table is scanned every 15 Hz cycle. Each LA may have to access the parameters stored there. Timing is kept there for each call. Also, a counter is incremented.

CPROQ

No longer in active use.

MMAPS

No longer in active use.

Q1553

Used by 1553 interrupt processing. Not used much at all in Linac. This table could easily be kept in volatile memory, as no memory of its contents need persist across system resets.

DSTRM

Used at initialization when data stream queues are initialized. Searched for a match against an 8-character name, although not very often.

SERIQ

Already in volatile memory.

TRUNK

Entire table updated by LOOPAAUX every 12 hours. New table lookup algorithm eliminates almost all accesses to this table.

ADEVX

This table not yet used at all.

CSTAT

This table scanned, by the RDATA entries that direct it, to build combined binary status words that are placed into ADATA reading fields.

CINFO

Already have scheme for dealing with this one by maintaining a volatile memory copy.

FDATA

Raw floating point channels reading, setting, nominal and tolerance values.

IPNAT

Table lookup algorithm eliminates need to search this table.

IPARP

Table lookup algorithm eliminates need to search this table. It is still searched to find a vacant entry when adding a new one. It is also scanned by `TimIPARP`, which performs timeout logic to ultimately free entries.

MAP32

Only needed to access when performing memory-mapped settings.

TRING

Used for sundry network-related info. Arcnet support finds its own variables there. Some of this should be maintained across resets, but not all, by any means.

DLOAD

Not used in PowerPC system. although its dos file system uses 1.5 MB for storing files.

Half-millisecond timing

At 15 Hz interrupt time, a copy of the TBL could be stored in some new global. The `TimeCycl` routine could read the TBL and make use of this value, convert it to microseconds, then divide by 500 to get half millisecond units. This is useful when time within a cycle needs to be stored within a byte.