

Memory-mapped I/O for D0

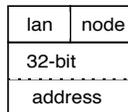
Selective access for DAQ boards

Nov 12, 1991

The online data acquisition used in D0 is based upon VMEbus access to the physics circuit boards. These boards often consist of a complex structure of memory-mapped registers. Some of these registers are normally not addressed at all. Some are read-only. But for downloading purposes, and for reading back to insure that the setting "worked," it is desirable to be able to transfer data to/from the board as if it were accessible contiguously. This note describes a means for the VME Local Stations to handle a variety of specially-configured memory-mapped boards in this pseudo-contiguous fashion.

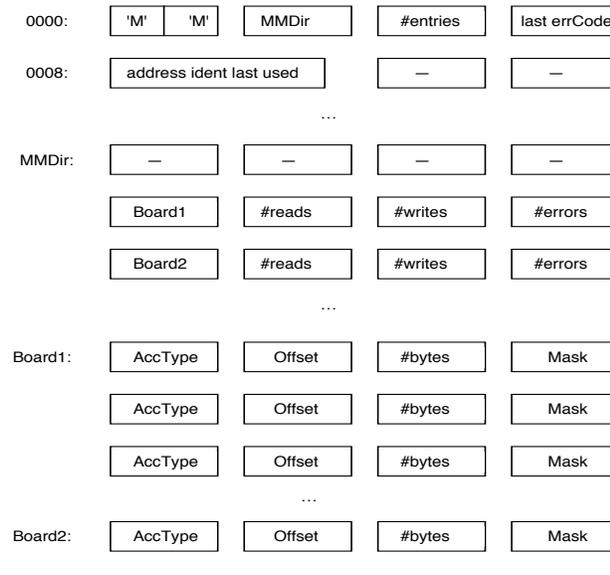
There is a table in the VME systems that holds the memory map information for a number of boards. Listype #46 is used to allow access to boards whose memory maps are described in this table (#16 called *MMAPS* internally). (Listype #47 is used to access the *MMAPS* table itself for downloading purposes.)

The ident used with Listype #46 is a memory address which is the base address of the board. It is expected to have several least-significant zero bits as used on the VMEbus. For this document, let's assume that the board always resides on a 64-byte boundary, at least. This will allow 6 bits that can be used to denote the board type. (It will be seen that this does not limit the accessibility beginning at areas of the board which are offset from the base address.) This allows a single listype to be used to access several different boards—up to 63. And it means that the D0 database can contain the address portion of the ident in 4 bytes. The ident format is as follows:



Again, the low order few bits of the address hold the board type number. The actual base address of the board is assumed to end in just as many zero bits.

MMAPS Table Memory Layout



In the `MMAPS` table format, the first word is a key to check for a valid table. The second word specifies the offset to the first directory entry indexed by board type#. (Board0 is not used in order to check for the presence of the board type in the low order bits of the ident address.) The third word gives the number of board types which are supported. The fourth word is the last nonzero error code produced. (A list of these error codes can be found at the end of this document.) The next longword is the last address ident used in read or write access using the `MMAPS` table.

From the board type, the directory entry is found consisting of an offset (from the start of the table) to the array of command entries that describe the memory map for that board plus some diagnostic counters, including counts of successful read and write accesses and a count of errors. The number of commands to describe each board can be variable; the directory entry only tells where the first command begins.

Each command may consist of two basic forms, each consisting of four words of information. The normal form includes an access type code to describe how the memory is accessed. The only access type initially supported is `type=1`, denoting a sequence of memory words. The second word is an offset value to be added to the "current" target address before processing this command. The third word gives the number of data bytes to be processed for that command. The fourth word allows for a mask that identifies the read/write bits in each word processed for purposes of verifying a successful setting. At the conclusion of the processing of a command, the current address points to the next location beyond the block, allowing for the loop processing described next.

The second form of command is the loop command. The first word of a loop command is `$D0D0`. (Read this as "D-zero DO" loop—small joke.) The second word gives the number of subsequent commands that comprise the body of the loop. The third word specifies the loop count. Loops can be nested.

As an example of a series of commands to access a hypothetical board which has 16 groups of 64-byte blocks, in which it is desired to access only the 3rd–5th and the 25th–31st words. The loop command form would be used for the 16 groups, and the two sets of registers would be specified by one command for each. A third command would serve to skip to the end of each group. The following specification could be used:

D 0 D 0	0 0 0 3	0 0 1 0	–
0 0 0 1	0 0 0 4	0 0 0 6	F F F F
0 0 0 1	0 0 2 6	0 0 0 E	F F F F
0 0 0 1	0 0 0 2	0 0 0 0	F F F F

Note that, as a crude check, the sum of the non-loop command offsets and #bytes words should add up to the length of the block in bytes. For this example, one would specify that the number of bytes needed to access the board was $(6+14)*16= 320$.

As each command is processed, a check is made that the #bytes remaining to be processed as specified by the user is large enough to cover the block described by the command. If it is not, then the number of bytes of data to be processed in the block is

reduced to match the number of bytes remaining. All this means is that the bytes requested can end anywhere within the command processing. In addition, if a bus error is encountered during access to any word of memory, further processing is terminated at the listype level. This can happen if the board is not plugged into the crate, for example, or if the vertical interconnect is unplugged.

The scheme described here makes it possible to accommodate the memory maps of several D0 physics boards. Note that different memory layouts can be handled for the same physics board. The VME system doesn't really know about boards; it only knows about memory maps given by entries in the MMAPS table. The table contents can be entered manually, or they can be downloaded by a host utility program. The format of the table is designed for efficient processing by the VME system software. The inner loop for processing the command which specifies a block of memory words is as small as it is for normal memory access.

Diagnostic error codes in MMAPS table header or a board's directory entry:

- 2: Invalid MMAPS table entry size (odd or < 8 bytes)
- 3: Invalid board# (> #entries)
- 4: Invalid board directory entry (offset to command list)
- 5: End of MMAPS table reached during command processing
- 6: Invalid command type code
- 7: Invalid loop command parameters or invalid nested loop
- 8: Verify failure after entire setting
- \$4000: Bus error occurred during access to target memory

Error codes returned to CDAQ are 42–48 for the above codes. In addition are these relevant error codes:

- 40: MMAPS table doesn't exist in VME station
- 41: MMAPS table is invalid (key ≠ 'MM')
- 4: Bus error occurred during read or write access
- 7: Invalid #bytes
- 52: Invalid ident length