Analog Control Addresses

Mapping 24 bits to 32 bits

Wed, May 10, 2000

The analog control field in the analog descriptor record is only 4 bytes in length. The first byte specifies the analog control type number, and the meaning of the other 3 bytes depends upon that type number. In several cases, the meaning of the other 3 bytes is a memory address, and for these cases, a routine called `MAPTO32` was invoked to convert the 24 bits into a 32-bit address. This note describes a new scheme for doing this that offers more flexibility for the future.

The logic of the `MAPTO32` routine is simple, as it has been used until now. If the most significant nibble is `0xF`, then the upper 8 bits of the 32-bit address is `0xFF`; otherwise it is `0x00`. This scheme was sufficient for reaching any address in the first 15MB range, plus the upper 1MB of memory space, which includes both VME and IndustryPack I/O space.

For the PowerPC version of the system, this solution is no longer adequate. The first 15MB of memory is almost entirely unknown, as it is used by VxWorks and for dynamic memory allocation. The upper 1MB (`0xFFFxxxxx`) has no particular use. The 2MB of non-volatile memory space is based at `0x18000000`. The Digital PMC board is based at `0x19000000`. The Altera part, including timers and other registers, is based at `0x19100000`. VME I/O space is based at `0xFBFF0000`. The mapping of 1MB of "low memory" is based at `0x00E00000`. VME A32 memory space is in the range `0x10000000—0x17FFFFFF`. A new scheme should be able to reach all such addresses.

The new scheme uses the first 8 bits of the 24-bits as an index into a table of 16-bit entries, each of which specifies the upper 16 bits of the resultant 32-bit address. Thus we need a table large enough for 256 words, or 512 bytes total. The contents of this table would be established as part of system configuration for a node. There will likely be a desire to make all nodes use the same table, so that the mapping is the same across different front end nodes.

A new system table is needed to support the above scheme. Since the D0 protocol is no longer needed, as the D0 detector will be using EPICS control system nodes, one of the three tables used to support it may be used. Suppose we use table #23, which was called CADIB and had 64 entries of 32 bytes each, for a total of 2K bytes. Change this table specification to 256 entries of 2 bytes each, and call it MAP32. Listype #59 can be used to access the entries in this table. This will allow saving and restoring the contents of the table.

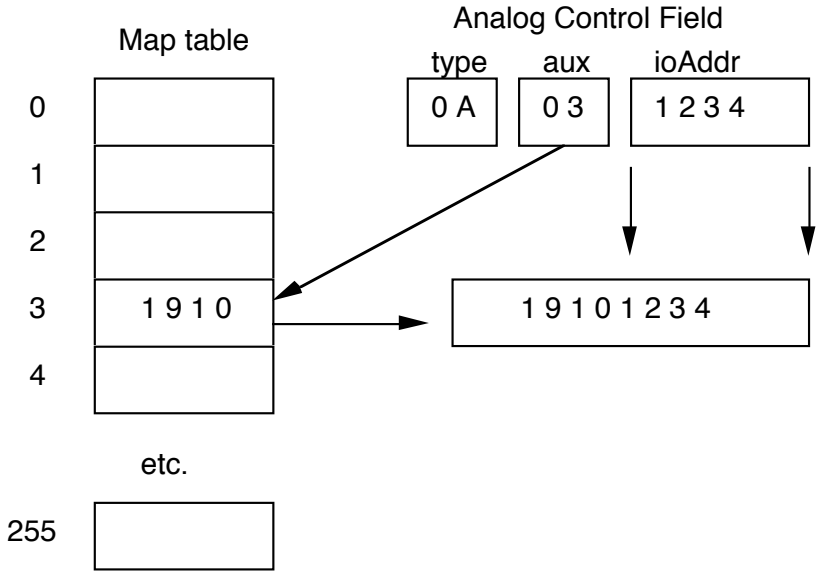
Besides `MAPTO32`, the new scheme can be used to support the use of the analog control field to contain a waveform address. This is used by the Macintosh Parameter Page in its support for Memory Plots. One would specify any 32-bit address by using `0x00` as the first byte, meaning that no analog control is supported, then using a 24-bit form of the address that is convertible via `MAPTO32`.

The Parameter Page program needs the support for `MAPTO32` to derive the memory address to be used in a memory data request. A new listype can be supported that allows access to memory words using `MAPTO32` to derive the address. The ident is a 3-byte memory address (a 4-byte address with the upper byte `0x00`). The pointer type routine invokes `MAPTO32` to derive the 32-bit address that is used for the internal pointer. This does not deliver the 32-bit address to the user, but it would deliver the memory data.

In `MAPTO32`, bits 23–16 are used as the index into the 256-entry table. If the word at that table

entry is zero, then return the address unmodified, else concatenate the nonzero word with bits 15–0 of the address to form the true 32-bit address.

The above scheme provides for up to 256 areas of 64K bytes in size to be referenced via 3 bytes of an analog control field entry. It is expected to be enough. A schematic example to illustrate the table lookup described above is as follows:



The analog control type `0x0A` specifies a simple memory word access, in which the other 3 bytes specify the target address. Using the second byte as a table index, the word from the table specifies the upper 16 bits of the target address. The lower 16 bits is taken directly from the "IOAddr" part of the field.