VMEbus FAQs— Frequently Asked Questions for the Beginning VMEbus User

John Rynearson, Technical Director, VITA

Questions: What are address modifier codes? Are they an option on VME? What purpose do they serve? How does a master generate these codes? Must all slaves decode these lines?

Address modifier (AM) codes have turned out to be one of the best ideas conceived by the original developers of the VMEbus. AM codes, as they are called, have allowed the VMEbus to evolve with new technology and to meet new application requirements.

Physically, the AM lines are implemented via six signal lines labeled AM0 through AM5. These lines provide address modifier codes on each and every address cycle. The address modifier codes give the VMEbus its ability to handle different addressing widths — 16 bits, 24 bits, 32 bits, 40 bits, and 64 bits; to handle block transfers; and to provide processor state information on a cycle by cycle basis. Thus a single VMEbus system can be configured with boards of varying capabilities that all work together correctly and dynamically on a cycle to cycle basis. Per the VMEbus specification, master boards must generate address modifier codes during each address cycle and slave boards must decode address modifier codes to determine whether or not to respond.

In the case of handling different addressing widths, two slave boards, one A24 and the other A16, may decode the same address value on address lines a1 through a15. When a master puts out an address for the A24 board, what keeps the A16 board from responding also? The address modifier codes for A24 space are different from the ones for A16 space. So even if address decoding on address lines a1 through a15 is identical for both boards, the A16 board will not respond to A24 address modifier codes.

Besides handling different addressing widths, address modifier codes specify block transfer operations. Block transfers maximize bus bandwidth by transferring multiple bytes of data per single address cycle. Address modifier codes are used to specify A24, A32, and A64 block transfer operations.

Address modifier codes can be used to control slave board response based on specific processor states. Multitasking microprocessors run in one of two distinct modes - supervisory state or user state. In supervisory state all instructions can be executed. In user state the execution of certain instructions, such as halt and reset, are not allowed. Separate address modifier codes are defined for processor state and user state and can be used to control slave board access. For example, in most real-time operating systems access to peripheral devices takes place through kernel system calls that execute in the microprocessor's supervisory state. Application tasks, on the other hand, execute in the microprocessor's user state. Thus address modifier codes can be used as a hardware means of controlling software access to critical I/O boards. The slave boards in a system can be set up to respond only to supervisory address modifier codes. An application task running in user mode that becomes corrupted cannot inadvertently access the slave board and cause unintended or possible dangerous actions.

Address modifier codes can be used to differentiate between program access and data access. Many microprocessors provide signal lines that indicate whether they are doing an instruction fetch or a data access. Slave boards can be set up so that they respond only to program accesses or data accesses. For example, a system could be configured with a shared global memory board for data only storage. Since the board will only respond to data mode address modifier codes, program mode accesses are prohibited. Critical data can be protected from unintended accesses thus enhancing system integrity.

Out of 64 possible address modifier codes, 16 have been set aside as "user defined". Application developers are free to use these address modifier codes to meet specific requirements.

When the VMEbus specification was first released only 14 address modifier codes were defined in addition to the 16 user defined codes. In the new VME64 specification 24 additional codes have been added. Because there are only a total of 64 AM codes defined for VME, there was concern during the development of VME64x (extensions to VME64) that we would run out of AM codes during the development of the 2eVME protocol. This protocol will allow faster data transfer by using both edges of the data strobes rather than the current one transition protocol. To resolve this problem, a method of supplying a primary address modifier code and then a secondary code during the address phase of the 2eVME protocol was developed. The use of a secondary address modifier code will allow new protocols to be added as required while insuring that interoperability with previously designed boards is maintained.

AM codes have served the VMEbus well. They allow the system designer the ability to configure a system with maximum flexibility to handle a wide range of address and data widths. 64-bit data transfers can be used to achieve optimum performance while 8-bit data transfers can be a cost effective alternative when a simple peripheral card is used. Both capabilities can exist in the same VMEbus system at the same time and accesses can take place on a dynamic basis. This capability is present because of AM codes.

* * *