1553 Control for Co-processors

Keeping control of the interface

Feb 2, 1989

In the Loma Linda VME system, co-processor boards are used to provide waveform generation to drive ramped power supplies. The interface to the supplies is via 1553. Access to 1553 cannot be shared, as sending a new command will cancel one which is in progress. One could make use of semaphores to support resource ownership, but the ramp generation is extremely real-time, with up to 4 power supplies driven simultaneously at a rate of 720 Hz. There is very little time that the 1553 interface is not busy.

The ramp co-processor routinely plays out the ramp and reads back the data from the four power supplies at 720 Hz. But digital control needs to be handled somehow. We must be able to turn power supplies on or off, for example. The request to do this will come from the main VME cpu either from its own current application—especially the parameter page—or from a network setting request. The hardware connections are there to allow the VME cpu to talk to the 1553 controller, but it dare not in order to keep from affecting the ramp adversely. We must get the co-processor to do it when it has a brief period of time available to sandwich it in with its other I/O.

The natural way to let the co-processor know about a digital control request is via a message queue in shared memory. By using a queue, several control messages can be awaiting co-processor service. (At the 720 Hz rate, there may not be need for a queue to hold a large number of messages.) There is a system table which contains pointers to co-processor queues. One of these is a co-processor command queue—a separate one for each co-processor.

We can send a message via a co-processor's command queue requesting that a 1553 control action be taken. The co-processor monitors the command queue when it has time available and processes requests it finds there. In the case of the ramp co-processor, the 1553 control must be handled by its interrupt code, which is used to drive the ramp I/O. So, the ramp task level processing may need to use another queuing mechanism to pass such requests to the interrupt code. Or, it may simply use a buffer for the purpose—a one element queue, if you will.

Returning to the VME main cpu, how shall it know to send a message to the co-processor to handle the 1553 control? (The case of supporting digital control is not simple, as has been covered in a separate note called "Digital Control Pulse Delays.") This note describes how to handle this at a low enough level so the complexities of the higher levels remain unaffected.

Let the various device tables in the VME system be built as if the VME cpu itself will control the 1553 interface. But, at the last moment, when the 1553 transaction is about to be processed, let the code realize that the co-processor must be sent a request message instead to do the 1553 I/O.

Whatever processing goes on from the point of a user pressing the keyboard interrupt key to initiate an off command, for example, ultimately results in a word of digital control data being sent to the 1553 interface. At that point, the routine called `OUTW1553` is invoked. The arguments passed to this routine are a pointer to the command block in 1553 memory where the command word is stored, the data word to be output, and a try count in case of errors. The command block pointer is enough to identify the 1553 controller being accessed. (Each 1553 VME board houses two controllers.) Normally, the job to be done is to copy the data word into the command block and alert the 1553 interface chip to process the command. But, if this command block is one which should be handled by a co-processor, then we build a short message instead and send it to the command queue for the proper co-processor.

How shall we determine whether a given command block should be handled by another cpu? One plan is to key on the address of the command block itself. In the case of the ramp co-processors, each uses a 1553 controller whose base address is `0x00Ex0000`, where "`x`" is the co-processor number. (Each 1553 controller uses 64K of address space.) This scheme is straightforward if the main cpu assumes knowledge of this formulation. One disadvantage it might have is that the ownership of a given 1553 controller is not program controlled. It depends upon the setting of the address switches on the 1553 board. (One could still get program control by changing the formulation dynamically.)

Another way to detect whether the 1553 control should be passed on is by examining the content of the command block itself. The hi byte of the first word is used to specify an optional offset to a "diagnostics block" for making a record of errors and usage counts. If the byte is zero, no diagnostics are recorded; if it is positive, it is taken as an offset from the start of the command block to an 8-byte area used for placing the diagnostics information. If the byte were negative, it could signify that a co-processor is to do the control of the 1553 controller which houses the command block. Specifically, if the first byte were in the range `0xC0`–`0xCF`, it could mean that the corresponding co-processor in the range 0–15 should be passed a message via its command queue. A possible disadvantage of this scheme would be that every 1553 command block in that controller's memory would have to be identified with the proper byte value in order to insure that the main VME cpu would not try to drive that 1553 controller itself.

A third approach is to keep a table of 1553 controller ownership. The table could be indexed by 1553 controller number, which ranges from 0–15 and is obtained from a command block memory address by isolating bits 19–16 of the address. In the version of the system software which uses 1553 interrupts, there is a table of 1553 queue pointers. An entry is placed into this table the first time 1553 I/O is done to a particular controller. One could add an additional field to the entries in this non-volatile table which would declare co-processor ownership for a given controller.

Alternatively, one could place a special code word in the controller's memory that could be interpreted as a declaration of ownership. Let a word near the end of a controller's 64K block of memory space be used for this code. The last two words (at offset `0xFFFC` from the base address) are actually the controller's register block. If we back up two more words before the register block, using the offset `0xFFF8`, we could use the word at that offset for the purpose. To make it definite, the word at the offset of `0xFFF8` will be the code word. If its value is in the range `0xC000`–`0xC00F`, it denotes corresponding co-processor ownership. If the VME system encounters values in this range when about to do 1553 I/O, it should not do the 1553 I/O. If it is sending a word to the 1553, it can instead build a small message to pass to the co-processor via its command queue.

After consideration of the above choices, the last one was chosen. The Data Access Table entries which drive 1553 data acquisition are ignored when the code showing co-processor ownership is encountered. When a word of data is to be sent to a 1553 RT—to set a D/A, for example—a message is built and sent to the co-processor's command queue. When the 1553 Test Page is being run, the code word in memory will have to be altered in order to get it to run. The 1553 driver that is called by that test page will not send any 1553 commands if it finds that the code word exhibits 1553 controller co-processor ownership.