

IRM Support for 1553

Necessary changes made

Aug 20, 1997

A few changes in the IRM system code were made to allow access to 1553 Remote Terminals, or RT's.

The addresses used for the 1553 controller 64K memory in D0 were \$002yxxxx, where y was the 1553 controller# in the range 0–F. For an IRM, we cannot use the same addresses, of course, since they refer to part of the 4MB on-board dynamic memory. IRMs are configured to access VMEbus for addresses from \$00800000 and up. But the 1553 controller board only decodes 24 bits of addressing; it has no P2 connector interface. This means that its address should be in the range \$00800000–\$00F00000. The system software doesn't have direct knowledge of the base address; it only knows about it via Data Access Table entries and other data base fields. We decided to use \$00800000 as the base address for controller 0.

Using a base address of \$00800000 means that accesses to the controller's memory would be cached. But we cannot allow this, since the contents of that memory are driven by the 1553 controller's DMA access as it performs I/O to the 1553 serial bus; the cache logic circuitry cannot "snoop" on such changes. The data caching scheme supported in the IRM system code is very simple, using only Transparent Translation registers for the purpose. But they only cache in units of 16MB, so that the choice of caching the first 16MB of memory space *includes* the chosen area for 1553 controllers. Without implementing the more complex support for memory management, we must either not use the data cache at all, or we must disable use of the data cache for systems that use 1553. The latter option was chosen. A change was made to the `Enq1553` and `Exec1553` modules so that before an access is about to be made to a 1553 controller's memory, the data cache is disabled. The disable remains in effect until the system is reset; there is no way to indicate that 1553 access is no longer to be used.

If a system is brought up that will use 1553, but nothing in the Data Access Table would require access to 1553, the system will come up with data caching enabled. If one tries to view controller memory, it won't be reliable, and it may even produce bus errors, as described in the next paragraph. But in any real system that uses 1553, there will always be DAT entries. The other logic that will result in setting up to address 1553 is during Settings Restore following reset. Any channel or byte that accesses 1553 data will cause a write to an RT, and that will initialize the system to use 1553, including disabling the data cache.

An additional reason why data caching cannot be used with the 1553 board is that the board only supports 16-bit data accesses, but four longword access are used when filling a cache line. The 162bug-based environment parameters can be altered to select D16 access for cache filling, but we don't want to use the data cache anyway.

An additional change was needed to support 1553 for IRMs. That is because the 1553 controller board resides on the VMEbus, so that we need to get the interrupt to be recognized that appears on a VMEbus IRQ line. We use IRQ5 for this purpose, and the CPU interrupt level chosen is level 5. In the `InzQueue` routine that is invoked by both `Enq1553` and `Exec1553`, is a call to `IPLInit` to enable the use of the 1553 interrupt.

New routines added to the system code are `IPLInit` and `DCache`, as follows:

```
PROCEDURE DCache(flag: Integer);
```

Enable/disable data cache if CPU is a MC68040. If `flag = 0`, disable data cache, else enable data cache.

```
FUNCTION IPLInit(irq, lvl: Integer): Integer;
```

Connect VMEbus IRQ line to CPU interrupt level. Argument `irq` must be in range 1–7. Argument `lvl` must be in range 0–7. To disconnect an interrupt, use `lvl = 0`. If the CPU board is not a 162 board, no action is taken. Normal return status is zero. Nonzero error return codes are as follows:

- 1 CPU board not 162.
- 2 `lvl` outside range 0–7
- 3 `irq` outside range 1–7

For convenience, all registers are preserved (for both routines) in the assembly language version of the system code.