Programming Model Rules (1)

Murrough Landon – 17 Sep 1999

We have evolved (from experience!) a set of rules and conventions which the programming models of all modules produced by our engineers should adhere to. Most of pretty obvious.

The aim is to aid the software control of the module - especially software development.

No "write only" registers

- This includes memories, FIFOs, etc.
- The idea is that all information about the state of the module must be readable by computer. This may not always be appropriate for highly volatile data, but in many cases the decision whether to trust the data should be left to the software.

Separation of status/control/reset

- Dont mix read only and read/write bits in single registers.
- Status registers are read only, control and other registers are read/write.
- Some registers are used for dataless actions, eg resets. The latter are the only "write only" exceptions to the above rule.

Programming Model Rules (2)

Read-only bits

- Attempts to write to read-only bits must leave the existing values unchanged!

Unimplemented bits

- Actions on unimplemented bits should have a defined behaviour: eg ignored on write, zero on read.

Power up status

- On power up, all registers should be set to zero - unless otherwise stated in the (full and complete!) documentation.

Volatile information

- Data integrity cannot be guaranteed if the computer tries to read or write data which the module can also update at the same time. However the Status and Control registers should always be accessible to determine the status of the module.

Address space

- When the address space occupied by the module (including unimplemented addresses) is addressed it will always respond with a handshake to avoid a bus error.

Programming Model Rules (3)

Module ID Register

- Every module must have a unique ID reported by a (read only) Module ID register which should be at the same offset (preferably $0) in every module in the system.
- The value encodes the module type, revision and serial number of each module. The encoding should be same for every type of module in the system.
- Normally daughtercards should also have a unique ID.
- This allows the configuration to be verified, module changes to be tracked automatically. If the Module ID is at offset $0 it is also easy to scan the VME address space to determine the configuration.

Status Register

- Each module should have one or more Status registers.

Control Register

- Each module will usually have one or more Control registers.

Comments on TTCvi

Comments on TTCvi Programming Model

- FIFOs are “write only”. No way to read back even persistent (repetitive) mode data. Preferable if FIFO contents, read and write pointers were available in read only memory and registers.
- 24 bit event counter in two D16 registers. Attempt to read with D32 gives bus error.

Other Comments on TTCvi

- Automatic clock generation if no input LHC clock may be dangerous (loose connection?). Ditto for orbit signal.
- No way for software to know whether internal or external clock is being used.
- Default (power up) state should be: no external clock means no output clock or data. Internal clock should be explicitly selected.
- Extra control and status bits for this should be provided.