

Fast Control Interface Update for DCT Upgrade

Version 0.0

Su Dong

The original trigger fast control (FC) protocols are still followed for the the DCT trigger upgarde, but with some additional FC commands and some modifications of implementation for the new DCT boards. For all old DCT boards TSF, BLT, PTD and the GLT, they all used a single FPGA (ORCA OR2C10) with identical firmware for the core FC logic. The same FC firmware is adopted with some minor modifications to the core FC code for the new ZPD and TSF, but this part of logic generally lives in a small part of a much larger Xilinx FPGA for the new boards. The “Op-control” section which interprets the FC command for board specific communications have changed signifiantly from the old DCT implementation and also very different between TSF and ZPD. The updated FC command documentation and VHDL simulation results can be found at [1] and [2]. The changes to the fast control commands:

- One major addition is the variable length block read and block write commands 12_{16} and 13_{16} . The old FC commands for fixed length block R/W commands 15_{16} and 16_{16} are not supported for the new TSF,ZPD boards, while the new commands $12,13$ will not work on any old DCT or GLT boards because the small ORCA FPGA hosting the FC is full for even considering an update.
- Additional CSR registers or bit fields in CSR are added for the control of the additional diagnostic memories in the new boards.
- CSR1 board ID fields extended to 3 bits for the more types of boards, stealing one of the spare control bits under Run-mode (no known application ever used or using it).

There is a major change of DAQ logic implementation for the both the new TSF and ZPD compared to the old DCT boards. This only involve a minor change in the core FC logic, but the “op-control” logic to assemble the DAQ data has completely changed. The previous DAQ logic for the old DCT boards typically following the steps:

- 1) Upon L1A, a frontend buffer is frozen and copying and formatting starts to reorganize/sparsify the raw frontend buffer data and builds the actual DAQ buffer data.

- 2) Upon ReadEvent command arrival, the FC core logic actually delays the ReadEvent command for $\sim 80\mu s$ to allow the DAQ data formatting in step 1) to complete. This is actually only needed for the TSF, which has variable sized event data. The reason this is fixed for a rather long time is that the ROM readout have to get synchronized Dlink header back from all TSFs in the same crate so that this delay has to be able to preserve the longest time spent by any single TSF for formatting its data. As soon as the DAQ data copying is complete internally the frozen frontend buffer can be released and starting to be refreshed with current data.
- 3) The delayed ReadEvent released from FC to the DAQ buffers and the DAQ buffers actively drives its data onto the DLINK path and FC and OP-control watches passively for a last-data flag to sense end of transfer and freeing the DAQ buffer for next event.

While the new DAQ logic adopted a different scheme:

- 1) The internal DAQ logic is actually continuously formatting the rawdata into the DAQ buffer. With some appropriate delay adjustment, when the L1A for the same event arrives (typically quite a few μs later), it will freeze the DAQ buffer exactly to the window one wants the read out for so that no additional copying/formatting triggered by the L1A are needed.
- 2) Upon ReadEvent command arrival, the readout can immediately commence with no waiting. Also, instead of passively receiving data from DAQ buffer, the OP-control actually actively control the readout to pull the data from a known memory address of the current DAQ buffer using the standard block-read protocol. At the end of the readout, OP-control frees the buffer just read to start refreshing that DAQ buffer with current data.

Comparing the two schemes, one can see that a major problem with the old scheme was that the long ReadEvent hold-off of $\sim 80\mu s$ significantly increases the total DAQ readout time and can have deadtime consequences. The new scheme not only avoided the hold-off to shorten the total readout time, the fact that OP-control takes active control of the readout using existing block-read protocol simplified the logic within the DAQ buffers and it only needs to behave like a passive ordinary memory. The new scheme also allow more debugging options as the DAQ memories are radable like an ordinary memory.

The current GLT DAQ scheme is in fact the same as the new scheme, but because it is using the identical FC FPGA as DCT, it is also unnecessarily held for $\sim 80\mu s$ before data coming back on the DLINK. Since the GLT has a rather long fixed data length of 583 bytes, the total readout time of $160\mu s$ has observable deadtime effect at a few percent once trigger rates reaches $\sim 4\text{Khz}$. A modification to the old FC logic will be attempted for the GLT and BLT to remove the FC hold-off and the deadtime influence.

There are two caveats from the new DAQ logic which still require some attention:

- While the old scheme has the disadvantage of the hold-off, it at least guarantees that the frontend raw buffer is fully refreshed with current data by the time the actual DLINK readout is complete. For the new readout scheme, there is a potential loop hole at the very rare occasion when all 4 buffers are filled with L1A and the system is in 'full' condition while trying to complete the readout of the first buffer. If another L1A arrives immediately (within $1\mu s$ or so) after the first buffer readout is complete, the system will try to free the first buffer again to store the new event. There might not be enough time after the release of this buffer to actually refresh its data with the new events so that one might get staled old data for the new event. Although this should be a very rare occurrence, it should be sealed solidly. There are two different approaches to address this:
 - ROM DAQ readout can force some additional readout of dummy words after the real data to effectively introduce a delay before allowing a 5th trigger to come in, while the board DAQ can use the dummy readout time to guarantee the DAQ buffer refreshing. This should only need a few μs of delay.
 - The ROM DAQ logic can also try to follow the DAQ readout block-read address pointer so that to refresh the memory region just read with new data immediately after the readout pointer moved passed the address. This can be done for the new TSF and ZPD, but it is not clear if it is realistic to do this for GLT and BLT.
- In the new scheme of DAQ readout, since the OP-control logic uses the normal R/W commands to perform DAQ data fetching, it will alter the current block address and memory address internally. For some diagnostic test FC commands which assume block-address and memory address being set previously, their operation after some DAQ operations will go to wrong addresses. Similarly the CSR3/4 current address read will not behave correctly after DAQ operations. This should not be a problem as most diagnostic tests will be very unlikely to be sandwiched by DAQ operations. In any case, ensuring the setting of block-address and address before any procedure is also recommendable.

References

- [1] The new updated fast control command protocol:
<http://www.slac.stanford.edu/BFROOT/www/Detector/Trigger/upgrade/docs/interface/fc-commands.ps>
- [2] The VHDL simulation of the updated FC protocol:
<http://www.slac.stanford.edu/BFROOT/www/Detector/Trigger/upgrade/docs/interface/fc/fc-simulation.html>