

ECR and BCR Counters

Background

Both the Event Counter Reset (ECR) and Bunch Counter Reset (BCR) are now expected to be counted and included in the event sent up the S-link from the ROD to the ROS. The exact details of how the BCR counter will be represented in the data are still to be decided, but this is a minor issue that has only a slight bearing on the following discussion.

In ATLAS running, ECRs and BCRs are received by the TIMs from the Central Trigger Logic (CTL) and then propagated to the RODs via the TTC1 and TTC2 backplane lines respectively [1]. The ROD creates the appropriate commands to transmit to the front ends. In SCT an ECR generates a Soft Reset in the ABCD [2] chips, a BCR generates a Bunch Counter Reset. (I'm not sure what happens in detail for the Pixels, but I believe that it is analogous)

Since both the TIM and RODs see the ECRs and BCRs, either (or both) could in principle keep count of them. Ultimately, the values of the counters are added to the event by the RODs, so that if the RODs don't actually count the resets, they still have to know the values of the counters.

Thus far, our planning has assumed that we would count the resets in the RODs. In the report from the TIM FDR, we were asked to consider counting the resets in the TIM instead and propagate the values to the RODs. This note is an attempt to look at the pros and cons.

The issues

If the RODs are responsible for counting ECRs and BCRs, then the advantages that I can think of are:

- ◆ The counters are where they are needed for their contents to be added to the event.
- ◆ There is less data transmission from TIMs to RODs. For the ECR counter, where the value will change on a periodicity of $\sim 1-160$ seconds, and hence will be the same for $\sim 10^5-10^7$ L1As, this will avoid resending the same value many times. For the BCR, this is 32 bits of data that has to be transmitted every event (the BCR will change roughly every 10 L1As at 100kHz).

I did initially think that there was also the benefit that Run Control need only set up the RODs in this case. However, it is likely that we will wish to change the TIM configuration at the start and end of runs (e.g. we might wish to move the TIM in and out of global mode, or reset its counters).

On the other hand, if we use the TIM to count ECRs and BCRs and then transmit this information to the RODs, the advantages are:

- ◆ The counters are being incremented in one location rather than sixteen, with the attendant reduced risk of loss of synchronisation due to miscounting. However, if we

get out of synchronisation, this is likely to be due to a ROD missing a ECR or BCR, in which case we have a problem anyway, as the front ends will be out of synchronisation.

The belt and braces approach would be to count the ECRs and BCRs on both the TIM and the RODs, transmit the values from the TIM to the RODs on each event and have the ROD check that its own and the TIM's values agree. This would protect against loss of synchronisation, but would cause up to 42 (8 ECR counter + 32 BCR counter + 2 control) extra bits to be transmitted every event. Also both the TIM and RODs would have to reset the counters at the same time (and the algorithm to be used to decide when to reset is not necessarily simple), and the ROD would have to do extra processing for every event.

The other consideration is how to transmit this information (ie. which TTC line)? Logically the ECR counter could be added to the L1ID, as the "extended L1ID" which appears in the event consists of 24-bit L1ID and 8-bit ECR counter. However, the Serial ID is already the longest bit stream to be sent from TIM to RODs – do we want to make it even longer (there may be implications for ensuring that the L1ID and BCID are known to the RODs early enough)? The BCR counter is less clear – partly because it isn't yet decided where the counter will appear in the event. It is also potentially up to 32 bits long. We **do** have a spare TTC channel (TTC7) which we could use to send these two counters – this might be the least disruptive approach? In fact TTC6 has no purpose at present for SCT either – it was set aside for "Front End Reset" but there are only two resets required for the front ends, covered by ECR and BCR (do Pixels use TTC6?), so we could consider using two channels. However, 42 bits of data only require just over 1 μ sec to transmit, and the transmission should be able to start soon after L1A propagation (How long after L1A would the TIM be ready to transmit these numbers?). Hence it probably isn't necessary to use two channels.

. I'm told that TIM-2 will not be able to handle the 32-bit BCR properly, but I had never envisaged that the existing generations of TIMs would be modified to provide this functionality – the requirement is aimed at ATLAS operation mainly.

Conclusions

It seems to me that the best solution is to count ECRs and BCRs on both the RODs and TIM and use TTC7 to transmit the data from the TIM to the RODs. The RODs would compare the TIM and local counters and flag up any discrepancies. In the case where TIM-2 is in use, ROD should count ECRs and BCRs and not try to compare with the TIM-supplied values – this should not be too difficult to implement in the ROD code, and I would expect there to be a flag to tell the ROD whether or not to expect the TTC7 signals.

References

- [1] SCT/Pixel TIM-ROD Interface Specification
(http://www.hep.ucl.ac.uk/~jbl/SCT/TIM_interface_ROD.html)
- [2] ABCD3T Chip Specification
(http://chipinfo.web.cern.ch/chipinfo/docs/abcd3t_spec_v1.2.pdf)

John Hill
2 August 2004