July 15, 1999
H.J. Frisch

Thoughts for SVT Code Meeting

1 Functionality- What we need to do

I've tried to list what functionality we'll need in SVT online code. Apologies to all those who have thought more deeply about some of these issues: I thought I'd just try it cold to see if anything had been missed. Thanks to Ray and Luciano for discussions.

1.1 Standalone Board Tests

1.1.1 Board Debugging (Teststands in B0 and at home)

Common Window program available at all institutions? (Is CDFVME too inflexible for this?)

1.1.2 Board Production Testing (at home)

Proprietary- everybody does what they need to.

1.1.3 Board Book-keeping

Each institution does their own? Or do we use CDF Hardware db? Access (read/write) from B0?

1.2 B0 Tests (no store in- no run)

1.2.1 Single Board Checkout

Easy interface to any board to run a frozen suite of tests.

1.2.2 SVT Board level slice

Run data from an Ispy to an Ospy from board A to board B Compare hardware output to simulation output

1.2.3 SVT Crate level slice

Run data from an Ispy to an Ospy from HF crate to TF Compare hardware output to simulation output

1.2.4 SVT Full system slice

Run data from an Ispy to an Ospy from HF crate to TF Compare hardware output to simulation output

1.2.5 SVX/SVT/L2 test

Book SVX and trigger; take data to trash. Compare hardware output to simulation output Controlled from RC.

1.2.6 Glink test

Is there anything other than the SVX/SVT/L2 test above? (FIB to VRB and HF test?) Can we checkout the parts of the Glink system without booking whole SVX?

1.3 Run Control B0 Code:

- Cold Start (download everything)
- Stage 0 Calibration: finding the beam before data taking; db access
- Hot Channel masking/unmasking/reporting/recording/fetching
- Warm Start
- Error recovery
- Error reporting
- Error diagnostics (run internally in RC- no operator intervention)
- Error procedures (e.g. replace a certain board, call SVT pager, etc.)
- End of Run (?)
- Start up SVTSIM (interface to db's)
- Close SVTSIM (interface output to db's?)
- RdB Interface: Defining what is in SVT partition.
- Partition Control: Lock out other VME users when SVT is in use.

1.4 Monitoring

1.4.1 Hardware level monitoring

- SVT System Integrity: Check SVT input to SVT output with simulation for some fraction of events
- SVT System Performance: Average tracks/wedge, clusters/track, etc.;
- SVT System Errors:Tabulating all error conditions: e.g, number of times Hold has come on per board, Lost locks, etc.
- SVX Data: Hot channels, dead channels: Check SVX data by looking for uniformity SVT wedge-to-wedge, channel-to-channel, SVT tracks vs phi

- SVT Pt/L3 Pt,d0,phi comparisons for J/psi, Upsilons, Z's(ee,mumu), W's(enu,munu)
- SVT efficiency vs L3 tracking (and maybe offline if available)

1.4.2 Higher Level (almost Physmon, but not) Monitoring

- Z mass in Z to mumu with SVT tracks
- E/p in Wenu with SVT tracks
- J/Ψ d_0 distribution
- J/Ψ mass
- Upsilon d_0 distribution
- Upsilon mass
- B-lifetime
- Z to bbbar (?)
- W to tau nu (?)

1.4.3 Accelerator Beam Monitoring

- Beam pos at z=0, slopes
- Beam envelope
- Measuring β
- Ability to give feedback to MCR before run starts for beam tuning.
- History plots of beam pos, slopes vs time in store, also vs store.

2 Other Issues

2.1 Partitioning

Xin has proposed that it be possible to partition SVT for debugging. We struggled with this issue for the Trigger System when we were designing it years ago, and decided it was simplest if the Trigger was not partionable. We worried about it, but it worked out fine: one could work on separate crates independent of Run Control, but if one wanted to be part of Run Control, it was basically all or nothing. I would suggest the same for SVT. (The 'basically' means that one could be doing stand-alone board tests in an SVT crate that was not defined as part of the partition as long as it didn't have any way to mess up the data in the partition, but this is unlikely and not all so useful.).

2.2 Locking Access

Since Run Control doesn't deal directly with the VME and the crates will be read out by the SVT workstation, we have a problem of resource management. We have to make it so that when the SVT is booked in a partition only the processes dealing with Run Control have access to the VME registers in the SVT crates. Is one way to do this is to have Run Control control the access codes to the processors (?) and to spawn the corresponding process(es).

2.3 Unix side vs Cside: Questions on Band Width Numbers

In the Rome proposal the question is raised how much is done in the crate processor and how much up in the Unix workstation. I think there is an advantage to working in the Unix world inside the offline framework as then the monitoring software is ensured to be exactly the same as any offline code that performs the same function, and the extra layer of compiling and downloading is avoided. Interfaces to other things (databases, for example) should be simpler, I would think. The one possible exception is monitoring at the channel level, particularly for the raw data: this probably belongs in the processor. My guess is that the default should be that anything else resides up in the workstation in an offline environment, but before I can tell this makes any sense I'd like to know the numbers. What is the bandwidth into the workstation vs. into the processor? What are the present limitations, and what is the ultimate limit? How hard is it to get past the present limitations? What is the required bandwidth to do the monitoring we have specified?

2.4 Offline Code

I didn't think about offline code. We should try to make any code that looks at data in the output stream (after L3, for example) run out of the offline. Hopefully this time we won't have separate offline libraries on B0 and FNAL, and so we can run directly out of the offline repositories. Much of this code is also used in the online: briefly thinking about what we need that overlaps with the online (as opposed to analysis code only):

- standard bank(object) unpacking routines
- standard dump routines
- display routines
- alignment package (aligns and makes geom db constants)
- geometry db read/write/display
- calibration db read/write/display