INTRODUCTION

This document outlines the software requirements for the 1999 beam test at SLAC. The requirements are based on the following principles:

1. As with the other efforts on the prototype tower, the aim is to carry forward as much of the work as possible beyond the beam test, through development and into flight. As such, there should be minimal reliance on existing End Station A (ESA)-specific DAQ software, except perhaps for the acquisition of beamline-related information. GLAST-specific software tools should be portable beyond the beam test.

2. Existing software tools, especially GLASTSIM, should be integrated into the beam test software. In particular, the model of the prototype tower using GLASTSIM should be implemented as fully as possible to take advantage of reconstruction algorithms, one-event displays, etc., and to allow a straightforward comparison of the simulation with the data.

3. As much information as possible must be available online in real time, or close to real time, to monitor and diagnose the experiment.

Items 2 and 3 are driven by our desire to do things sensibly, but also by a very practical and over-riding constraint: the development program requires that we must produce detailed results very soon after the beam test (when everyone is exhausted).

In the following sections, the various tasks will be described. This will certainly be incomplete, and it will be necessary to iterate the design and functionality. This document should serve only to help us get organized and get the work started to define the architecture. After we agree on the requirements and a basic architectural framework (very soon!), we will be able to generate a detailed task list.

Run Control

The Run Control (RC) presents a window that will serve as the main control of the experiment, and will give the operators information about the status of the DAQ (possibly in a separate window). It is the main client process that communicates with the GLASTDAQ process in the tower. Functions include:

1. Start a run. The run number will automatically be incremented (with the ability to overwrite with some safeguards). The time/date stamp will be provided to the DAQ from the run control to be bound into the run header (see DAQ Functions below), along with other run information that includes run type (photon, electron), readout configuration, primary beam energy, B0 magnet setting, initial collimator settings, scan table settings, operator names, and operator comments. The DAQ will perform initializations, and allow triggers.

2. Pause the run. Restart the run.
3. End a run.
4. DAQ setup menu page, including reinitialization, trigger configuration, readout configuration (which components to include in readout in case we need to remove one) setup and polling of beamline information sources, etc. Probably much of this needs password protection to make sure operators don’t screw up the configuration by mistake.
5. Slow control menu for remote control and readback of HVs, scan table positions, comparator threshold sets, etc. A list should be compiled from the components. There will be setup pages for each of the components as required. (Proposal: THERE WILL NOT BE INDIVIDUAL EQUIPMENT COMPUTERS FOR EACH COMPONENT IN THIS BEAM TEST. The required tasks for each component will be accomplished by their respective TEM board processors, and the run control will serve as the message passer to those control and setup processes.)
6. Control menu for data dump, including starting a new data tape with auto-incrementing tape ID, flushing local cache, etc.

The information presented during a run will include
1. Number of triggers taken in the run.
2. Run time start.
4. All status/error messages (different text color for each category) passed back from the GLASTDAQ process running in the tower, in a scrolling box (with a BIG buffer to check the history).

GLASTDAQ

The GLASTDAQ process is the main DAQ process running in the tower CPU. It takes its setup and control orders from the RC, and passes information about its status to the RC (see RC functions above). It administers the gathering of data from all sources, packetizing of data, and shipping of data out to the Data Handler on the local workstation (see Data Handler section below). Packet types include
1. Run header. Added to the data stream at run start, this includes all the relevant run control and configuration information.
2. End of Run packet. Added to the data stream at the end of a run, this includes all the relevant run control information, number of triggers, etc.
3. Pause and restart run packets give the time and status of run pauses and restarts. These are very small packets that let us know later why event timestamp sequences have large gaps.
4. Event packet. All the information from all processes relevant for each event. This includes all the raw data from the instrument and the beamline components, as well as any level 2 and level 3 trigger and reconstruction information, daytime stamp, trigger number, etc. (NOTE, our goal is to have as much of level 2 and level 3 algorithms running real time as possible.)
5. Housekeeping packets. Passed periodically in the data stream, and at the beginning and end of runs.

An important task is to define the content of these packets.
In addition to gathering data, GLASTDAQ will keep watchdog timers and will send RC warning messages when, e.g., no event interrupts come from the beam line clock, a component isn’t ready or gives incorrect status messages, etc.

**Data Handler**

This is the process that runs on a local workstation to receive the data from the tower. It is the equivalent of the ground-link level 0 processing station. For the beam test, the data handler will

1. Receive the data, do any necessary additional formatting, and place it in shared memory to be analyzed by the Online Monitoring tasks (see next section). **The final format of the data needs definition.**
2. Hand the data to the Data Dump (SLAC tape silos?) for storage.
3. Report its status to the RC or to GLASTDAQ (which will pass on status messages to RC).
4. Take its setup and control orders from either RC or GLASTDAQ.

**Online Monitoring**

The online monitoring tasks spy on the data in shared memory on the workstation. There are two basic categories of tasks:

1. Single event display. This shows individual events as they come from the tower to shared memory. The rate of display is selected, including the ability to freeze an event. It would be helpful if selections could be made on the display (e.g., only show events with a minimum calorimeter energy or with a minimum activity in the tracker). The GLASTSIM display tools should be used. Raw data (tracker and ACD hits, cal energy deposits) and reconstructed quantities (tracks) should be displayed. In addition, if possible, the photon tagging hodoscope and Pb Glass should be implemented.
2. Histograms, strip charts, etc. Pre-defined histograms should be automatically cleared at run start and on command, and then filled during the run from shared memory. It would be most helpful if the data were stored in a tuple of some existing package (PAW, IDL, Hippodraw -- TBD) that allowed definition of histograms (including cuts and other criteria) and their display (e.g., log, linear) properties by the operators during data taking.

**Data dump**

This process administers the buffering of data on a local disk and the dump to the tape silos (or whatever we choose for mass storage). It reports its status to, and is controlled by, the RC or the GLASTDAQ process. This should be as transparent as possible, with minimal operator interference.