A BRIEF DESCRIPTION OF THE GSE SOFTWARE

The GSE software that I am writing falls into two main Catagories. The The SEP GSE software which deals with the telemetry at the packet level addressed to any part of the SEP intrument suite. The LET GSE software that deals with extracting, displaying, and logging the LET data from the packets.

The SEP GSE software will perform the following five things in real-time:

1. Maintain an interfaces to the data source
   a. IMPACT IDPU simulator (Caltech),
   b. IMPACT GSE (UCB),
   c. Ground System or IMPACT GSE (APL).
2. Log all packets containing any SEP data to disk,
3. Passes thru a TCP/IP socket all packets containing any SEP data to all the individual telescope GSEs present,

3. Maintains page displays of any data specific to SEP central,
4. Maintains a hex display of packets sent to the individual telescope GSEs.

It was originally plan that the SEP GSE software would be responsible for commanding the SEP suite of instruments while SEP was under Caltech's control, and after delivery to UCB, the SEP GSE software would have no role in commanding. Since then, there have been discussions and e-mails about a remote command capability. In which the SEP GSE will forward commands from the individual telescopes. This issue was never definitely answered, so the role that the SEP GSE will perform in commanding remains unclear.

All SEP GSE software will be written in Java.

The LET GSE software will perform the following items in real time:

1. Maintain a TCP/IP socket connnection to the SEP GSE,
2. Log all packets containing LET data to disk,
3. Extract standard (to be determined) LET data structures for housekeeping, rates, events, command table ... from the packets,

4. Log to individual disk files the standard structures extracted in the step above,

5. Maintain windows containing page displays of any standard LET data structure requested by the user,
6. Maintain windows with time series plots of user-selected fields from the standard (one field per window) page displays,
7. Maintain a scrolling window for LET command echoes,
8. Log LET command echoes to disk.

   The LET GSE software will also provide a playback capability that replaces the TCP/IP socket data source with a disk file and turns off all disk logging.

All of the above software will be written in Java.

The LET GSE software will provide an offline IDL interface to the LET data. At the conclusion of each run, the LET data will be added to calendar-based archive data system. All that is meant by a calendar-based is that the files have names that identify the date on which they were generated and the directory structure which contains the files reflects this calendar-based organization. For example, archive/2003/2003.03/2003.03.10/let.2003.03.10.hskp would contain the standard housekeeping data structures for March 10, 2003. By arranging the data files in this manner, it

makes it possible to write IDL routines that return all the data structures of a given type that occur between a user-specified start time and a user specified end time.

In addition, the LET GSE software will provide a simpler off-line IDL interface for data from the current run. In this interface, the user will call a routine that reads one of the standard log files from the current run and returns all the structures in that file.

The purpose of these IDL interfaces is to provide additional display and analysis capabilities beyond that which can be provided in the real-time Java code. It is far quicker and cost effective to do any offline analysis in IDL than Java. Since the data needs to be archived, it makes sense to put in a system which is accessable to all the IDL users at SRL.

One part of the GSE software which I have refained from discussing is the software that will be needed for the accelerator runs. I suspect that the requirements will not be generated anytime in the near future, and I have little idea of what the software requirements will be. I am also uncertain what type of computer hardware will be necessary.

A BRIEF DESCRIPTION OF GENERIC GSE SOFTWARE

The design that I am planning on using for the real-time GSE software has only a few parts that are dependent of the format of the data.

The remaining part of the real-time software are independent of the format and can be written before the data format is defined. This set of format-independent parts are what I mean by the generic GSE software.

The design makes heavy use of the Java observable and observer interfaces. The main objects in this design are described below.

There is an observable object which acts as a packet listener. Once the packet listener detects that a packet has arrived, it logs the packet to the disk, notifies its observers that a new packet has arrived, and returns to listening for the next packet.

The observers of the packet listener are the filtering objects. Each filtering object takes the packets which the packet listener provides and tries to complete one or more copies of the data structure it was designed to build. Everytime it completes one of its data structure, it logs it to the disk and notifies its observers. It then tries returns to the task of trying to build the next data structure. When it exhausts the data in the packet, it ceases to operate until the packet listener informs it that a new packet has arrived.

The observers of the filtering objects are the objects responsible for building the page display models. A page display models is little more than a collection of objects which contain a string to be displayed, the color in which that string will be displayed, the x and y coordinates of the location in the window in which the string will be displayed, and if the string represents a quantity for which can be used in a time series plot the object contains the numerical value represented by the string. Whenever the objects that build the page models are notified that a new structure is available, they build a new page display model from the information in the structure and notify their observers.

The observers of the objects that build the page display models are are the page display viewers and time series viewers of that data structure. A page display viewer has the responsibility of writing the text from the page display model in the window at the proper locationis and the time series viewers have the responsibility for updating the time series plots.

The only places that I need knowledge of the format of the telemetry are in the filtering objects and the page display model builders. All the classes that generate the other objects can be written without needing to wait for the format description to be completed. In particular, all the GUI code can be written before the format is known.

STATUS OF GSE

Last fall, I began the task of developing the generic GSE software. I worked on this task until November. The result was an almost complete complete design of the generic pieces of the real-time GSE display code. Since I have only a design and no code, I put the status of this task at about 30% complete.

Starting in November and running to February, I put aside this task to work on the software for detector development system. In December, while working on the detector development system, I realized that the detector development system had similar needs as far as archiving data. Since I wanted to based the archive system on time and wanted to use the same time system for all the STEREO work, I went and wrote the IDL code for the time routines that are necessary to move data in and out of the archives. (This is not actually part of the generic GSE effort, but it is part of the code for the GSE data archive which will implemented in the future. The detector development system will have a need for a data archive before the GSE system.)

In March and April, the amount of time spent on the GSE effort and the amount of progress has been minor. In March, a small amount of time was spent on deciding which data packets the SEP GSE would pass to the HET and SIT GSE. In April, the issue of remote commanding arose. The resulting confusion has left the commanding role which must be implemented in the GSE software unclear. I have no plans on working on any commanding software until this is clarified.

Having finished the SIS anisotropy code (minus document) for ACE, I will be increasing my efforts to get the generic GSE software finished, I am especially interested in getting the GUI code finished as soon as possible.

GSE HARDWARE

No GSE hardware has yet been specified . The GSE will require one computer per spacecraft. At the moment, I favored a Pentium laptop over a desktop to ease any future transportation problems. However, I am not planning on making a decision until the software requirements are complete.

Also, I am not sure of the hardware needs during the accelerator runs.