Charlie Strohman
4/5/05

DISCLAIMER: These ideas represent a significantly different way of using the instrumentation and may require a larger programming effort than what we have resources for.

There are probably operational requirements that I do not understand that (like 50 Hz data rates) that will invalidate these ideas.

A proposed system for management and coordination of CESR DSP-based instrumentation.

Provide *management tools* to allow a collection of *devices* to be shared by multiple *applications*.

The attached diagram shows a pictorial representation of the various items discussed in this proposal.

Devices:

- BPM (MKI)
- BPM (MKII)
- BSM
- FLM
- JIMA Box
- Timing System

Applications:

- Orbit
- Luminosity
- Injection
- Tune
- Hardware Test
- What else?

Management Tools:

- High-level subroutine library (VMS/Unix)
- DSP subroutine library
- Hardware programmer's manual
- Compilation scripts
- Configuration files
- Shared data structures
- Timing system interface

A typical sequence of events:

1. An application is started. It reads a file containing a list of required devices. It also reads and stores configuration data related to the specific devices.
2. The application calls the process manager. The application provides identifying information (PID and node). The process manager returns an ID byte that is unique to the instance of the application.
3. Each time the application is ready to use the devices, it calls the allocation manager. The application provides its ID byte and a list of required devices. The allocation manager attempts to allocate all of the asked for devices. Assuming that it is successful, it returns the ID bytes of the previous owners of each device. More thought is required to deal with unavailable devices.
4. The application now owns the devices. The application examines the ID bytes of the previous owners. If the bytes match the application's byte, the application will know that no other applications used the devices since it last used them. It can assume that they are configured correctly. If the ID byte is different, then the application needs to examine the configuration status table and see who last changed any of the configuration structures that are relevant to this application. If any are changed, the application needs to configure the device. When any configuration structures are changed, the ID of the application making the change is stored in the configuration status table.
5. The devices are now configured correctly. The application is ready to issues commands. It can use Ethernet or Xbus. If acquisition needs to be synchronized by the timing system, the application makes a call to the timing manager and provides its ID byte. The ID byte will be sent over the timing cable. Each device has been configured to look for that specific ID byte. More thought is required on how to deal with the situation where more than 2 hardware triggers are required and they are switched with a multiplexer.
6. After acquisition is complete, each application manages its own data handling. It determines what data is read from the devices. It can write to a file, send something to a display, put something in the MPM, perform calculations, or do whatever is required.
7. When the application has completed collecting its data, it calls the allocation manager and de-allocates the devices. They are now available to be allocated by another application or re-allocated by the same application