Java Driver Implementation for the Ethernet Flash ADC
John McKisson, Thomas Jefferson National Accelerator Facility

- Multi-platform library for Ethernet Flash ADC (EFADC) and Ethernet Trigger Supervisor (ETS)
- Easy to use API
  - Simple DAQ system running in ~15 lines of code
  - Usable by 3rd party DAQ software (ie Kmax) for larger systems
  - Coincident trigger setup
- OO abstraction of the ETS/EFADC connectivity tree
- Event aggregation
Java Driver Implementation for the Ethernet Flash ADC

John McAllister

Abstract:

An EPADC is a data acquisition module designed to provide multifunctional fast ADC data collection with a synchronous timing mechanism. Each EPADC is built from seven separate units to within 4 ns. Each EPADC and provides 16 simultaneous flash ADC converter channels with 8-bit resolution and multiple EPADCs can be managed by an Ethernet Trigger Supervisor (ETS). It provides a single network connection in a tree hierarchy. The trigger mechanism provides hierarchical trigger capability, with each module being capable of generating triggers. Each trigger section can be configured with varying length integration windows, individual thresholds, and can provide coincident trigger conditions for use in applications requiring coincidence such as PET.

User Interface for Applications:

We use Akara (http://www.akaraweb.com) as a graphical environment to interface with the EPADC. Triggers are written in Java which allows the use of standard graphics libraries such as Java2D. A Command-line in the Java standard as the interface to an EPADC is used and is then able to run image and event generation algorithms with the provided data.

The EPADC Driver:

The software for the EPADC is managed by a local version of EPADC driver software, and control is implemented over a physical layer of 1000BASE-T Ethernet using UDP protocol. The driver software is capable of handling a connection to a single EPADC unit or multiple through an ETS. The driver software is implemented in Java making use of the Java library Zipper interface (http://www.zipper_), and has the package which provides a higher level protocol driver network stack.

Object-Oriented Design:

The software provides an object-oriented driver model after the hardware's network interface connectivity, and is able to determine and provide the proper driver configuration depending on the hardware's physical configuration.