NSCLSpecTcl Meeting the Needs of Preliminary Nuclear Physics Data Analysis

Ron Fox, Chase Bolen, Kanayo Orji, Jason Venema

Abstract--**The National Superconducting Cyclotron Laboratory at Michigan State University (NSCL) is a user facility performing basic reasearch in heavy ion collisions with rare isotopes. A wide variety of experiments are performed at the NSCL, each with specialized analysis needs. Providing a package that balances ad-hoc extensibility and performance was the distinct challenge facing the NSCL software support staff. We chose to implement a hybrid C/C++ application framework and TCL/Tk extension language. This paper will describe nuclear experimental physics, the needs of the community, how they are met by NSCSLSpecTcl, the structure of the software and the extension language. We will also describe several extensions to NSCLSpecTcl that have been created by the user community and others.**

I. INTRODUCTION

THIS paper will describe a Tcl/Tk[1] extension that consists of a C++ application framework that is controlled by an extended Tcl/Tk interpreter. NSCLSpecTcl is intended to be used during online and early stage offline analysis of nuclear physics event data. The structure of the paper will be as follows:

- Section II will introduce experimental nuclear physics, the National Superconducting Cyclotron Laboratory (NSCL) [2] and its research objectives. Research at the NSCL will be placed in the context of the national goals for basic research.
- Section III describes the requirements of the software. We will describe technical as well as usability goals.
- Section IV describes the approach of other packages in use in nuclear and high energy physics that embed scripting languages provides the motivation for our choice of Tcl/Tk as an embedded scripting language for NSCLSpecTcl.
- Section V will describe the structure of the NSCLSpecTcl C++ application framework, introducing the technology of application frameworks, the structure of the framework we have developed, describing how Tcl/Tk is integrated into the software as a control language.
- Section VI will describe some of the extensions that have been written for NSCLSpecTcl.
- The paper will conclude with a summary of the experiences with the system, the level of community acceptance within the NSCL and the nuclear physics community in general. The availability of the software will be described as well.

II. BACKGROUND

In this section we will describe how experimental nuclear physicists study the structure of the atomic nucleus. The National Superconducting Cyclotron Laboratory (NSCL), a National Science Foundation (NSF)[3] funded university laboratory will be introduced and its research agenda will be placed in the context of national near-term and long-range nuclear science research goals.

A. Experimental Nuclear Physics

In experimental nuclear physics, scientists study the structure of the core of the atom through particle-nucleus collisions. By scattering particles off the nucleus we can learn about the inner composition, shape of the nucleus as well as improve our understanding of the forces that govern nucleon-nucleon interactions.

Particles are scattered off the nucleus either in collider or fixed target accelerator systems. In a collider two accelerated beams of nuclei move in opposite directions. The two beams intersect, causing some of the nuclei in one beam to collide with particles in the other beam. In fixed target systems, a particle accelerator accelerates a beam of projectiles that is transported to a target consisting of the element being studied. In either case, complex detector systems surround the interaction region to capture and measure the properties of the resulting collision fragments. While higher reaction energies are possible with collider systems, these systems are generally

Ron Fox is with the National Superconducting Cyclotron Laboratory at Michigan State University, East Lansing, MI 48824-1321

Chase Bolen is currently a graduate student in the Department of Electrical Engineering at Michigan State University, East Lansing, MI 48824

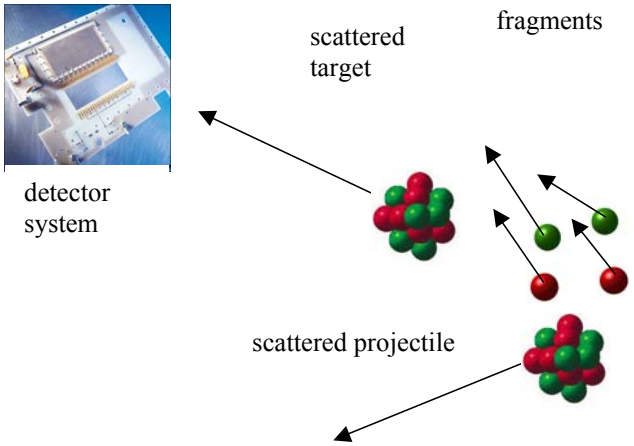
Kanayo Orji is now a Medical Student at the Medical School of Wayne State University, Detroit Michigan

Jason Venema is an employee of Boeing Corp St. Louis MO 63166

limited to the study of symmetric systems. Fixed target systems, on the other hand support the study of asymmetric systems.

Electrons and other nuclei have been used as projectiles in fixed target systems. Electrons, are not subject to the nucleon-nucleon force, but will scatter off the charge of the protons in the nucleus, and therefore make excellent probes of the spatial structure of the nucleus. Using nuclei as projectiles provides information about the interactions of the constituents of the nucleus with each other. This allows scientists to probe the energy level scheme of the nucleus, the stability of the nucleus under deformation and, if sufficient numbers of nucleons interact, the statistical properties of nuclear matter.

Figure 1 shows a schematic of an *event* in typical fixed target experiment. An event is a single collision that produces measurable results. Figure 1 shows the remnant of a projectile that has scattered at some angle to its original incident trajectory, a target remnant knocked out of the target



in the forward direction and a scattering of smaller fragments that may have come from the projectile, the target or both. A detection system around the target area will capture and detect the properties of some or all of these reaction products.

Figure 1 Schematic view of nucleus-nucleus collisions in a fixed target system.

B. The National Superconducting Cyclotron Laboratory

The National Superconducting Cyclotron Laboratory at Michigan State University (NSCL) is a National Science Foundation supported university laboratory that performs nuclear physics experiments. The NSCL has a pair of cyclotrons that accelerate beams for fixed target experiments. The cyclotrons are capable of accelerating beams as heavy as Uranium with energies of up to about 1GeV/nucleon. An electron volt (eV) is the energy gained by an electron falling through a potential energy difference of 1V. The electrons returning to your car battery have an energy of about 12eV. At the other end of the energy scale, a typical donut has the chemical energy equivalent of 10^{15} GeV.

A unique feature of the NSCL is the A1900 fragment separator. Primary beam from the cyclotrons interacts with a production target at the entrance of the A1900. Fragments of the projectile enter a spectrograph where the isotopes of interest are selected. These projectile fragments continue to travel with energies close to those of the primary beam. This allows exotic nuclei to be transported to experimental targets. The A1900 allows researchers at the NSCL to study the properties of highly unstable isotopes that do not occur in nature. Figure 2 and 3 show NSCL experimental area floor plan as well as as schematic of the A1900.

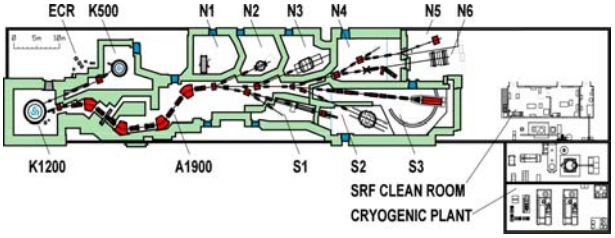


Figure 2 The NSCL experimental floor plan.

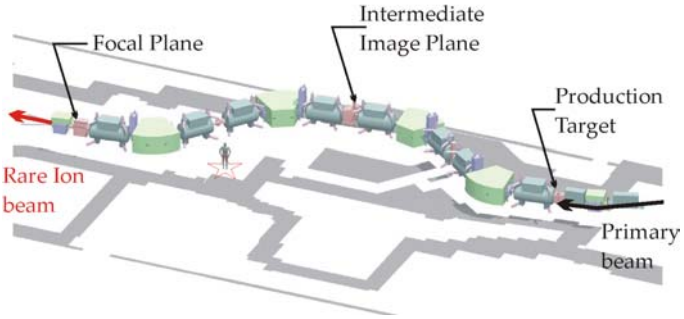


Figure 3 The A1900 mass separator

The NSCL is a user facility. Scientists from all over the world submit proposals to an independent Program Advisory Committee which awards beam time on the basis of scientific merit.

C. *The NSCL and U.S. scientific research goals.*

NSCL research is focused on interactions between heavy ions. Heavy ion physics research helps us understand the nature of the inter-nucleon strong force, and to probe the statistical properties of nuclear matter such as the nuclear equation of state and phase transitions that may occur in hot nuclear matter. Experiments with unstable isotopes from the A1900 allow us to answer questions about the properties of nuclear matter in conditions far from stability. This has direct bearing on understanding what happened in the early universe and what happens in the interiors of stars. Measuring the lifetimes, decay modes and the probabilities of each decay mode (branching ratios) contribute to an understanding of the distribution of isotopes in the universe and how this distribution came about.

In the past several years the U.S. nuclear physics community has been moving towards the development of a new experimental facility called the Rare Isotope Accelerator (RIA). The NSCL A1900 is the precursor facility to RIA. When built, RIA will extend the energies and intensities of the beams of exotic nuclei that can be produced enabling experiments with nuclei even further away from stability. In 2002, RIA was endorsed by the DOE/NSF Nuclear Science Advisory committee as the one of the second highest priorities for funding [4]. In November 2003, U.S. Energy Secretary Spencer Abraham released the Office of Science priority list showing funding for RIA tied for third place amongst the near-term priorities [5]. NSCL accelerator physicists have been a leaders in the conceptual design of the RIA facility. MSU is competing with Argonne National Laboratory to host RIA. Since the NSCL is a world leader in the production of rare isotope beams by projectile fragmentation, MSU would be the logical place to build RIA.

An architectural sketch of the ½ km long linear accelerator that will make up RIA is shown in Figure 4 below.

Figure 4: The Rare Isotope Accelerator (RIA) at MSU.

III. Requirements for Preliminary Analysis Software

This section outlines the requirements for preliminary nuclear physics data analysis software. These requirements break down in to two groups:

- Technical or functional requirements must be met for the software to be minimally useable for data analysis.
- Usability requirements enable physicists to work efficiently and effectively on their research problems.

A. *Technical requirements*

In the early stages of data analysis, physicists are trying to understand the experiment they are performing or have performed. This is done by creating various histograms from parameters in the *event data*. To create these histograms, the raw event data acquired from the digitizers attached to the detector system must be decoded to produce a set of parameters. These parameters may be as simple as the height of a voltage pulse from a solid state detector that stops a heavy fragment, or as complex as computing fragment positions vi a Gaussian fit to the charge deposited on a set of pads in a Cathode Readout Drift Chamber .

Experimenters need to:

- Histogram individual detector channels to monitor their health. Histograms can be of several types including simple single parameter histograms, histograms of parameter pairs (2-d), histograms of the bits set in a bitmask, histograms showing summaries of the data in multiple similar channels (summary Spectra) or histograms that are multiply incremented for several parameters (e.g γ-ray decay cascade histograms).
- Calculate and histogram calibrated parameters by transforming raw parameters to physically meaningful values.
- Produce "purified" histograms by defining conditions that must be met by an event for a histogram to be incremented. For example, a particle identification gate may be set around ^{10}Li fragments to gate a detector energy spectrum so that the resulting spectrum shows the allowed energy states of that neutron rich Lithium isotope.
- Produce and histogram parameters that represent correlations between raw or calculated parameters, such as fragment opening angles, total transverse and perpendicular momentum vectors etc, and histogram these as well.
- Create filtered event files that contain useful subsets of the parameters of events that meet a particular condition.

- View and graphically interact with histograms as they are being created.

B. *Usability Requirements*

Usability requirements are driven by several factors. Nuclear physicists have a wide range of computer expertise ranging from highly sophisticated to illiterate. Experiments at the NSCL are run by collaborations of scientists, some or all of whom may come from outside the NSCL and not be familiar with our data acquisition and analysis software.

Often, experiments are built by plugging together modular pieces of apparatus. For example the NSCL Segmented Germanium Array (SeGA)[6] a gamma ray spectrometer, is often run at the target chamber of a large spectrograph called the S800 [28]. Each of these systems have relatively independent software requirements. To extract physics from these experiments, however requires computing parameters that represent correlations between these two detector systems.

NSCL experiments run for about a week, and can produce up to 80Gbytes of raw event data that may contain as many as 2 billion events. This implies that if 1ms is required to analyze an event, 740 hours of compute time would be required to do a single pass analysis of the entire experimental data set. Event analysis is often an iterative process involving several passes over subsets of the data and over the entire data set if necessary.

These and other considerations led us to the following usability requirements:

- The software must be easy to learn to use by people at all levels of computer expertise (or lack thereof).
- The software must be quickly adaptable to specific experimental needs.
- The software must be extensible to meet needs that were not anticipated by the initial structure.
- Good histogramming performance is essential given the volume of data that is pumped through the system.
- Data acquisition system neutrality to support analysis of data taken by NSCL physicists visiting other laboratories, and to support a broader user community than the NSCL.

IV. Why embed Tcl/Tk

The desire for extensibility and a high level of adaptability pointed towards embedding a scripting language in the software. Analysis performance needs, however dictate that then inner analysis loops must run at compiled code speed. This is not a new approach in the nuclear/high energy physics community. The developers of two popular packages, the Physics Analysis Workstation (PAW) [8], and Root [9] have chosen to do this as well.

In the case of PAW, a custom interpreter called KUIP was developed and embedded. In addition, ad-hoc analysis is supported by the inclusion of a FORTRAN interpreter called COMIS. PAW is a very powerful analysis tool, well suited for late stage analysis, however the learning curve for PAW is quite steep. The KUIP command language has a rather arcane syntax as well and rather limited functionality. Paw fails to satisfy both our technical and usability requirements because of its steep learning curve and the fact that the user interface is essentially unresponsive during data analysis.

The Root developers chose to embed the C-INT[10] interpreter. C-INT is an interpreter of the C/C++ programming language. Thus scripting in Root is effectively programming in C or C++. To do this requires a thorough knowledge of the Root class library. Root has been successfully employed in High Energy Physics experimental program where the lead time for an experiment is on the order of 5 years, and each collaboration employs a software development group consisting of several people. Root's learning curve is steeper than that of PAW although, once more Root may be well suited for later analysis projects. We also have philosophical problems with C/C++ as a 'command language." Root fails our usability requirements due to its steep learning curve, and requirement that the user be a reasonably knowledgeable C++ programmer.

We chose to embed the Tcl/Tk interpreter because it was usable at several levels of complexity, depending on the needs and abilities of the user:

- To the novice user, Tcl/Tk verbs provide a simple imperative command language with a model that matches their experience with the Unix command line.
- For slightly more expert users, Tcl/Tk can be used to automate simple repetitive tasks, or script complex analysis tasks.
- Tcl/Tk allows non programmers or unskilled programmers to easily build application specific Graphical User Interfaces (GUIs) to provide simplified access to more complex, scripted functionality.
- Tcl/Tk is an extensible language that allows expert users to integrate application specific extensions to the core framework of the analysis package.
- Tcl/Tk is an existing language, with good community support, a rich set of add on packages.

V. The Structure of NSCLSpecTcl

NSCLSpecTcl [11] is an application framework that embeds and extends the base Tcl/Tk interpreter. Application frameworks provide the main flow of control of an application allowing the programmer to implement application specific

functionality by providing code that hooks into well defined extension points. In short, an application framework is a program in search of a library.

Using the application framework approach in NSCLSpecTcl frees the non-expert physicist/programmer from many design issues, and allows them to concentrate on meeting the needs of their research. This is in stark contrast to the approach taken by the developers of Root, which can be thought of as little more than a class library that delegates the bulk of these design decisions to the user.

Experience has shown that physicists absorb software documentation best via a set of sample programs, rather than through reading detailed software manuals. The base classes of the extension points for an application framework provide a relatively natural basis for these examples.

Many well known, and successful application frameworks exist. The Microsoft Foundation Classes (MFC) [12], the X11 [13] Xt [14] and Motif [15] toolkits, as well as more recent GUI class libraries such as Fox [16], Qt [17], and Gtk [18]are examples of application frameworks intended for the development of graphical user interfaces. Writing even simple applications for these frameworks requires extensive knowledge of their libraries. In the High Energy Physics community Root, mentioned in the previous section appears to the user like an application framework. This framework too requires extensive class library knowledge to perform even simple tasks.

Tcl/Tk's application programming interface (API) is another example of an application framework. This framework succeeds much better at allowing programmers to function with only a limited knowledge of the API. If you can do simple UNIX command line processing, you can write Tcl extensions.

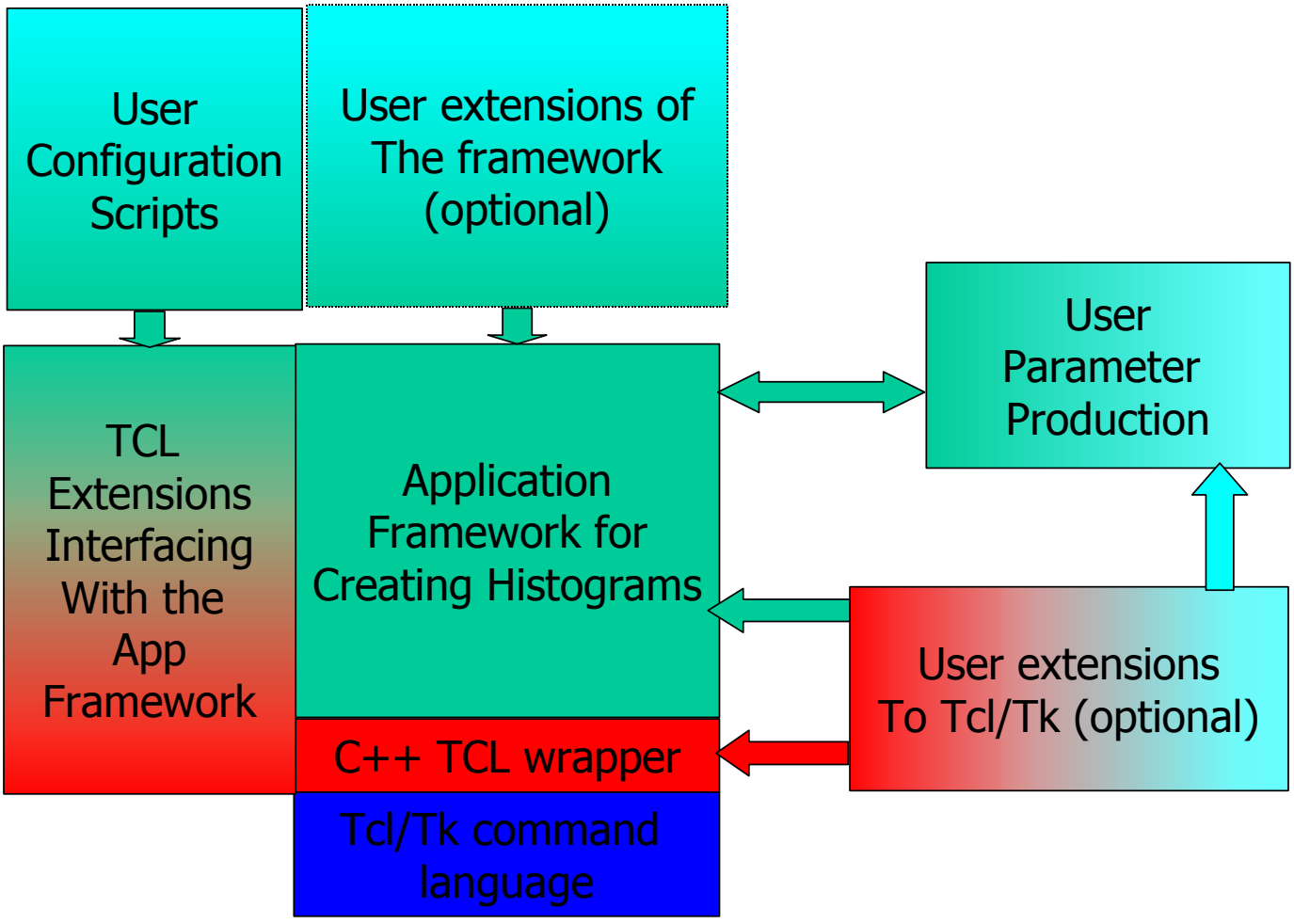


Figure 5 Block structure of the NSCLSpecTcl application framework.

Our goal in designing the NSCLSpecTcl's framework was to be closer to the Tcl/Tk model than the Root model. In fact, NSCLSpecTcl uses the Tk application framework to provide the program control flow, and external event dispatch.

A. *High Level Structure of NSCLSpecTcl*

Figure 5 is a diagram of NSCLSpecTcl's major subsystems. It also shows the extension points for the framework.

The core of the system is a library for efficiently taking parameter n-tuples and incrementing (possibly conditionally) histograms. This is the block labeled "Application Framework for Creating Histograms". An n-tuple is High Energy/Nuclear Physics jargon for a one-dimensional array of parameters, where some indices may not always refer to valid values.

The Framework interacts with the Tcl/Tk interpreter through a C++ encapsulation of the Tcl API. The histogramming data flow is registered with Tk as an event processor that is called by the Tk application framework when the SpecTcl event data source becomes readable.

The Histogramming kernel is configured via a set of Tcl language extensions. These extensions are implemented in terms of the C++ Tcl API wrappers. The extensions provide commands that define, manipulate and introspect the definitions of parameters, spectra, gates and other entities needed to analyze data.

To use SpecTcl the user must provide code that transforms a raw event into its n-tuple representation. In NSCLSpecTcl, an n-tuple is represented as a simple wrapper around the Standard Template Library's [23] vector class called CEvent. CEvents contain *valid-value* objects. A valid value is a floating point number that knows if it has been assigned a value since the last time it was invalidated. To the experimentalist, CEvent is an array. CEvent objects automatically expand as needed to accommodate the largest referenced index. The allocation of CEvent objects, and their dynamic resizing presented performance problems in the first versions of NSCLSpecTcl.

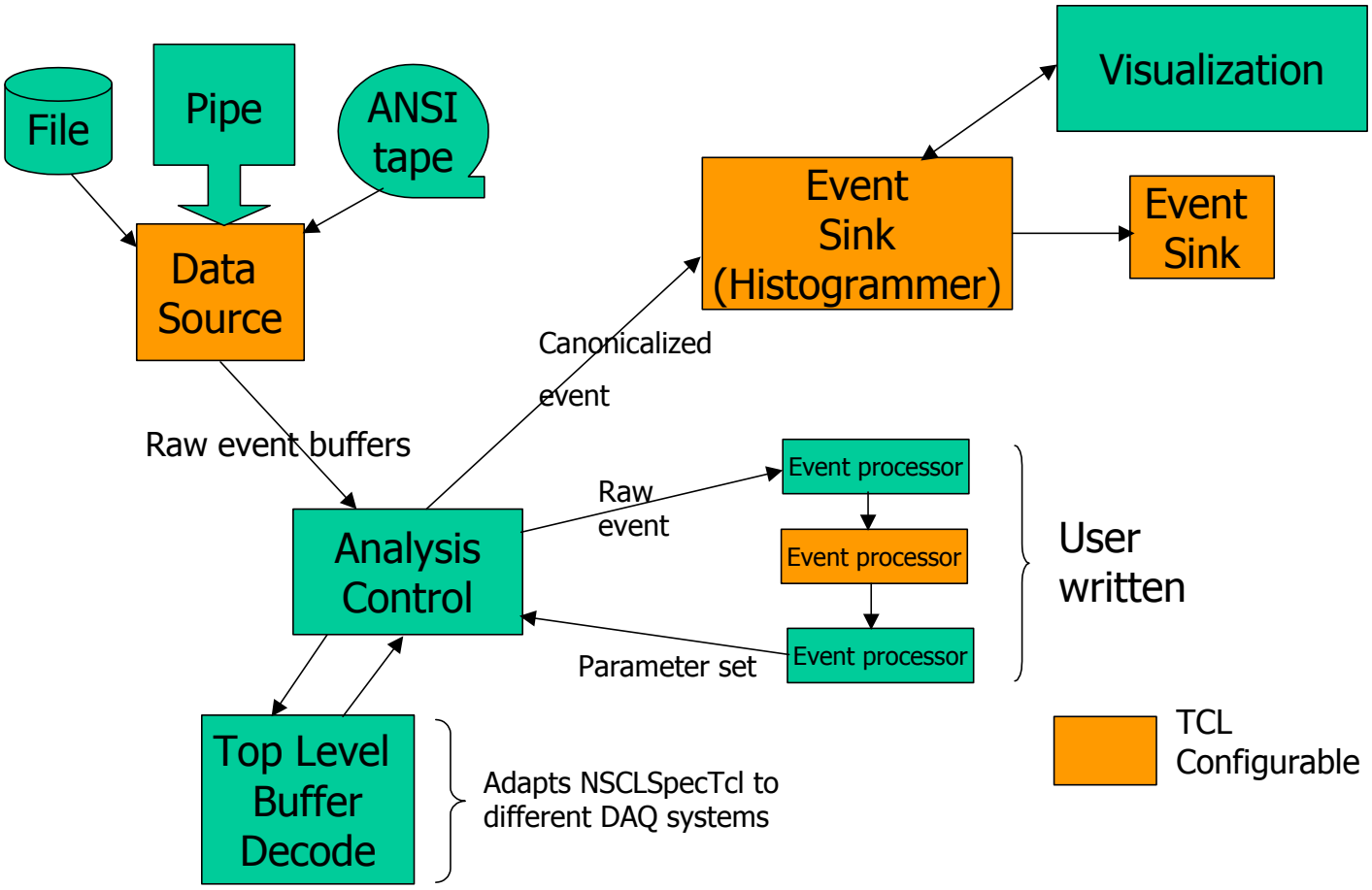


Figure 6 Event processing flow

Therefore the framework re-circulates a set of CEvent objects

which eventually will expand to hold the worst case size needed by the application.

The user can extend NSCL SpecTcl in three other ways as shown in Figure 6:

1. User written scripts configure the core framework and can provide application specific GUIs.
2. Users can extend the framework. Several subsystems of the NSCLSpecTcl application framework are built with extensibility in mind. For example, the Segmented Germanium Array collaboration at the NSCL (SeGA) have extended the spectrum I/O subsystem to support transparent export of histograms to the Gelifit gamma ray spectrum fitting component of Radware [19].
3. Users can make use of the C++ Tcl classes to extend the command language understood by NSCLSpecTcl. These extensions are often hybrids extending NSCLSpecTcl framework to add functionality that is controlled and monitored via new Tcl commands or steered via Tcl variables.

B. Event Processing Model of NSCLSpecTcl

Figure 6 shows NSCLSpecTcl's event processing model. Tcl extensions configure a source of event data. At present, data sources can be disk files, ANSI labeled tapes (the nuclear physics community has a large amount of older data on ANSI labelled tapes), or the standard output of a program attached to a pipe. The pipe data source is how NSCLSpecTcl connects to an online data acquisition system. The pipe data source is one mechanism that maintains NSCLSpecTcl's data acquisition system neutrality.

The event data source is processed as a Tk event handler. This allows the user interface to remain fully live while a data set is being processed. While this feature is useful for offline analysis of large data sets, it is crucial for online data analysis.

Buffers from the data source are routed to the analysis control subsystem. The Analysis control subsystem interacts with a Top Level Buffer Decoding module (TLBD). The TLBD module is an extensible and replaceable NSCLSpecTcl module. This module is the second mechanism used to maintain data acquisition system neutrality within NSCLSpecTcl. The TLBD module makes callbacks to well defined entries in the Analysis Control subsystem to dispatch control to the appropriate handler software. In general, files of event data meta-data describing the run as well as raw event data. In this paper, we will only be concerned with what happens to the event data.

Replacement TLBD modules have been written. These modules have supported transparent analysis of data produced by TUNL/IUCF Xsys [24] as well as data taken from the NIRS HIMAC in Chiba Japan [25].

NSCLSpecTcl cannot analyze raw events. It only knows how to analyze decoded event data: n-tuples. The user of NSCLSpecTcl must provide at least one *event processor*. Event processors are user written classes fill an n-tuple from a raw event. The user can provide a many-staged logical pipeline that performs this transformation. Each stage of the pipeline has access to the raw event, and to partial n-tuple created by previous pipeline stages.

This pipelined organization serves three functions:

1. If raw event data are appropriately formatted, it is possible to use separate stages of the pipeline to unpack the data from relatively independent detector subsystems.
2. The event pipeline isolates the production of computed parameters from the format of the raw data. For example, a separate stage can calibrate the data from a detector, producing calibrated parameters, by referencing the previously unpacked data in the n-tuple.
3. When detector systems are combined in an experiment, computed parameters that span detector subsystem boundaries can be computed without any knowledge of the format of the raw data produced by the subsystems.

At the discretion of the programmer, event processors may be controlled via Tcl command extensions or steered via Tcl variables. For example, calibrated parameters require determining the parameterization of a calibration function. This parameterization is not known at the beginning of the experiment. The calibration event processor can be written so that it can be dynamically enabled or disabled via a Tcl variable treated as a boolean flag or by Tcl command extensions. The calibration function parameters could be Tcl variables.

N-tuples created by the event processing pipeline are passed to another logical pipeline of event sinks. Two types of event sinks have been written:

1. The histogrammer, which produces the set of histograms defined by the Tcl commands used to configure it. The histogrammer makes the histograms it creates available to the visualization component, a separate Motif/X11 program called Xamine [26].
2. Event filters. Event filters allow the creation of output event files that receive a specified subset of the parameters in the event only when a condition evaluated on the event is true. Filter output files are written in a simple self describing form. An pre-written event processor that unpacks filtered event files to n-tuples is provided.

User written event sinks are supported. At this time none have been written. One use for a user written event source would be

to produce an output event file in a format other than the NSCL event file format.

C. *The Extension Language*

NSCLSpecTcl implements a set of about 25 commands and scripts that extend the functionality of Tcl/Tk. These commands are intended to configure and control the application framework. Several Tcl global variables are also maintained to expose the state of analysis and provide some analysis statistics. A minimal sample GUI is provided as well.

Commands have been added to Tcl/Tk to:

- Establish the event data source and control the analysis from it.
- Associate names with n-tuple slots and define the properties of the parameters in these slots. A **pseudo** command allows a new parameter to be created from existing parameters by executing a Tcl procedure.
- Define, clear and access spectra, and make them known to Xamine. Spectra can also be written to and read from disk files or any Tcl file descriptor created with the Tcl **open** command.
- Define event conditions and apply them to spectra. These conditions can also be set by directly clicking points on spectra displayed by Xamine.
- Define and activate filters.
- Save and restore the state of the analysis.

All commands that produce objects also allow complete introspection of these objects. This introspection is necessary to suport many features required by GUI front ends. Example 1 shows a sample analysis session for a very simple experiment. In this experiment a pair of silicon detectors each produce an energy (de and e), and timing information that help to determine whether or not events in the detector are beam related (t).

Using parameter introspection, spectra are created for all raw parameters. A 2-d spectrum (pid) is created with de on the X axis and e on the Y axis. This produces a crude particle ID spectrum for charged particles (each particle species produces a hyperbolic hill in the 2-d spectrum). The axis specification for the spectrum command specifies the low and high limits of the parameter and the number of channels used to bin that interval. The ADC and TDC channels are assumed to be 12 bit digitizers.

We show the creation of a gate on the time spectrum to select beam related particles, and assume that the user clicked in a contour gate around the ^{10}Li peak in the 2-d spectrum. We then create an energy spectrum for the beam related ^{10}Li particles.

```
#   Define the parameters:
parameter de 0                  # name n-tuple slot number
parameter e   1
parameter t   2

#   define 1-d spectra doing it this way is overkill but
#   It demonstrates introspection. Large detector systems
#  (1000's of params) do this.
foreach parame [parameter –list] {;   #Iterate over params.
  set name [lindex $parname 0];       # Extract param name.
#             spname type   parname   axis specification
  spectrum $name 1       $name    {{0 4095 4096}}
}
#   Make the 2-d spectrum:

spectrum pid 2 {de e} {{0 4095 1024} {0 4095 1024}}

#   name    type    param    low   hi    (slice gate).
gate Beam    s     {t          {287 305}}

#.... user clicks in the 10Li gate:

gate BeamRelated10Li * {Beam 10Li}; # And gate.
spectrum 10LiEnergy 1  e {{0 1023 1024}}
apply BeamRelated10Li 10LiEnergy

sbind –all;                     # bind all spectra to Xamine.

clear -all
attach –file run1234.evt
start
#   .... Run is analyzed....
swrite –format ascii run1234LiEnergy.spec 10LiEnergy
```

Example 1: Sample NSCLSpecTcl extension commands in action

Finally we start analyzing from the event file run1234.evt and, when analysis is complete we write the gated energy spectrum. Note that an analysis state variable *RunState* can be used to automate end of analysis actions via either **trace** or **vwait** (depending on whether or not the analysis is batch or interactive). For example, preceding the swrite command in Example 1 with a "**vwait** RunState" would have blocked script execution until the run had been analyzed and then proceded with the **swrite** command. Experimenters use this idiom to do offline batch analysis.

VI. Extensions to NSCLSpecTcl

This section will describe three specific, significant extensions to NSCLSpecTcl. Two of these projects were undertaken by experimental groups at the NSCL. The last was performed by one of the authors as part of a consulting project.

A. *The MoNA Graphical User Interface*

An important research topic at the NSCL are the properties of neutron rich nuclei. These nuclei, when excited will often decay to more stable nuclei by shedding their neutron excesses. The Modular Neutron Array (MoNA) [20] a detector system that came online at the NSCL in August 2004 to study the properties of neutron rich nuclei.

The detector consists of 9 layers of 16 blocks of scintillator. Neutrons that enter a scintillator block will produce light. The light output is amplified by photomultipliers attached to either end of each brick. Position along the brick can be determined by a light division algorithm, while the brick position itself determines position in the axis perpendicular to the long axis of the bricks. Time of flight relative to an upstream start detector provides an accurate measurement of neutron energy as well.
The efficiency of this detector is high enough that neutrons can be tracked through the detector to ensure that they originated at the target.
MoNA is shown in figure 7. The photomultiplier tubes are the cylindrical devices at the left and right side of the detector.

Figure 7 The Modular Neutron Array (MoNA).

MoNA was built collaboratively by undergraduate students from several colleges and universities. In addition to providing information about the properties of neutron rich nuclei, MoNA provides undergraduate students valuable experience in experimental nuclear physics.

MoNA produces a large number of spectra of both raw and computed parameters. William Peters, a graduate student with the NSCL part of the MoNA collaboration produced the control panel shown in figure 8. This panel allows spectra to be incrementally created as they become meaningful. The GUI can also control the data source and start and stop analysis. A status frame displays the progress of the current analysis case.

B. *Structure on top of the n-tuple: TreeParam*

The model of a flat n-tuple is difficult to use for large

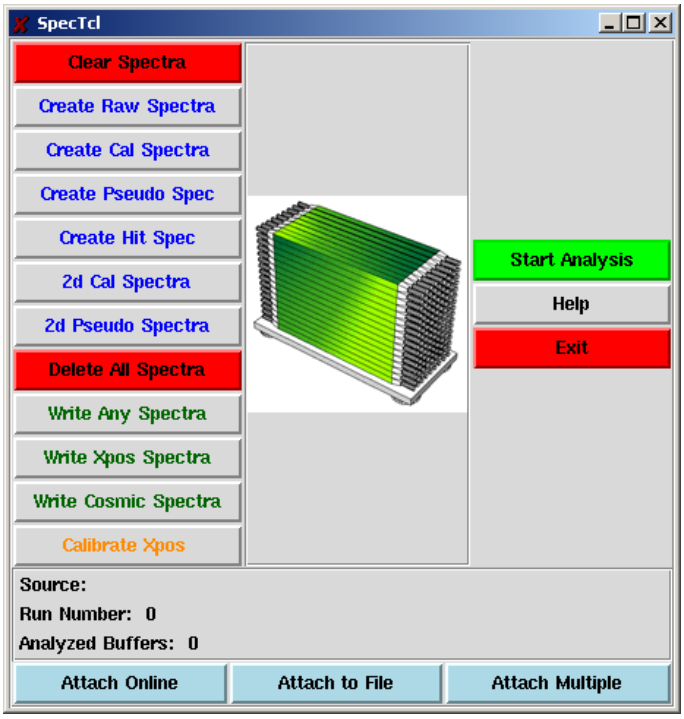


Figure 8 The MoNA GUI

hierarchically organized detector systems. Daniel Bazin [21], the NSCL S800 device physicist, has written an extension to the NSCLSpecTcl framework that supports superimposing a hierarchical structure on top of the NSCLSpecTcl n-tuple. This extension is called *treeparam*.

Treeparam is used by most large detector system collaborations at the NSCL. In addition to class library support for this structured overlay, Dr. Bazin has extended Tcl to allow treeparam parameters to be arbitrarily scaled and for this scaling to be introspected. He has used this support to build a Treeparam GUI that supports, among other things, creating new spectra by navigating the parameter hierarchy, setting and inspecting parameter scale factors, and maintaining a similar structured set of parameters obund to Tcl variable. This GUI is shown in figure 8.

C. *Scripting the Raw event to n-tuple unpack.*

Experiments at the NSCL and other nuclear physics facilities use a wide variety of nuclear electronics. Optimum data rates are achieved by acquiring data in an event format that is very close to what the hardware produces. Unfortunately, this

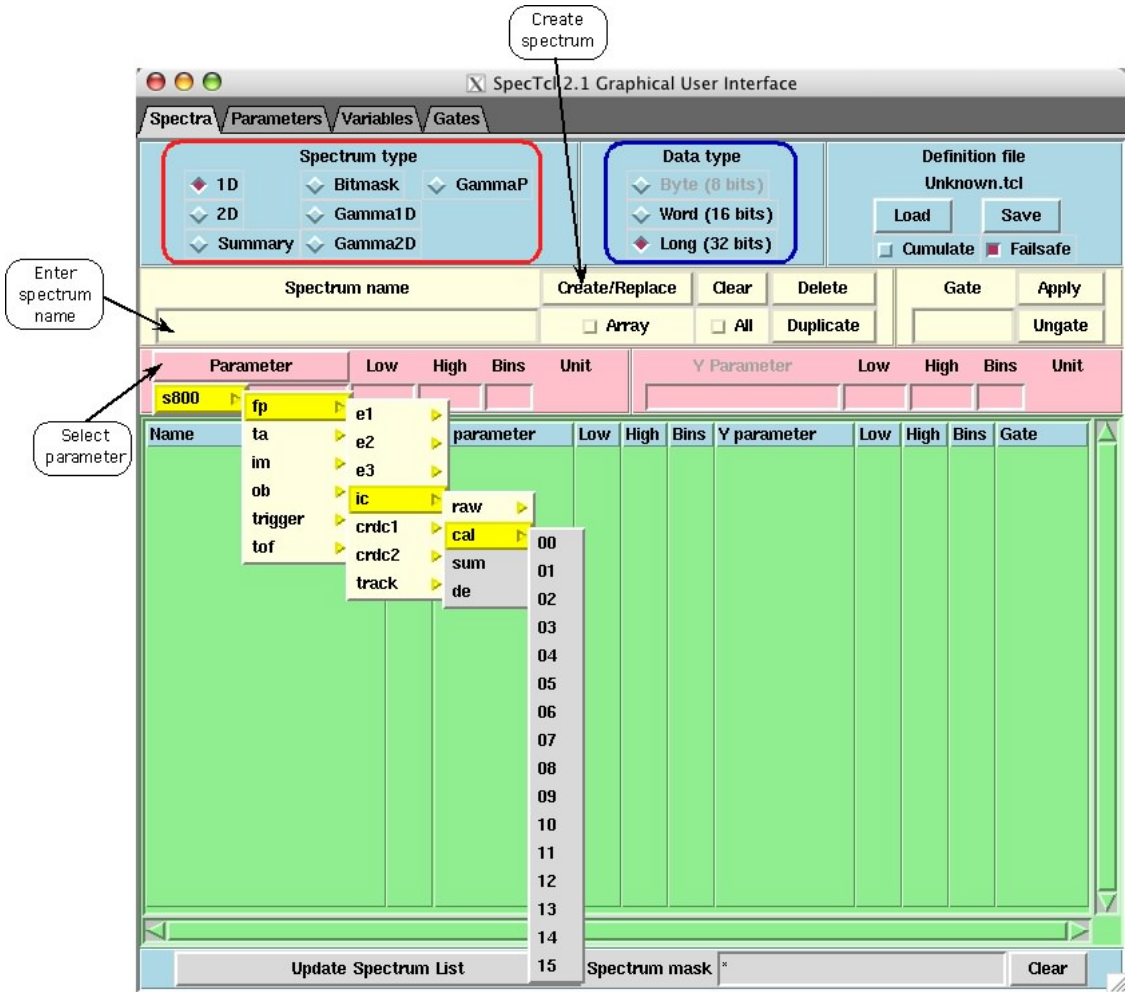


Figure 8 The treeparam GUI

format is not always self describing. If it were, users would not have to write software to unpack their raw data into an n-tuple, NSCLSpecTcl could do it for them.

A subset of the nuclear electronics do produce data with enough internal structure to permit a mechanical transformation from raw event to n-tuple. An example of these devices are the 32 channel VME bus digitizers produced by Construzuioni Apparecchiature Elettroniche Nucleari spa. (CAEN). The model 785 peak sensing ADC, models 792 and 864 charge integrating ADCs (QDCs), and 775 time interval digitizer (TDC) [27], all produce compatible self-describing output.

One of the authors, under contract to Lawrence Livermore National Laboratory to provide a turnkey, but flexible data acquisition and analysis system for γ-ray spectroscopy wrote extensions to the SpecTcl framework and Tcl command set to support mechanically unpacking experiments based solely on these modules [22]. The concept of the extension is that the user defines modules of various types. These modules are configured in much the same way Tk widgets are configured. Modules can be organized in a hierarchy of *packets*. A packet is a block of event data that is preceded by a size and an identifier. Modules are added to packets or the top level event (which is itself a packet). The parameters associated with each module can be described when configuring the module. The Readout software of the data acquisition systems that use this scheme source the same script, and use it to configure the online readout of the hardware, ensuring consistency between the structure of the event data and the configuration of the software that unpacks it.

Example 2 shows a fragment of a configuration script chosen to give the flavor of this extension. In this example, two packets are created named *first* and *second*. The first packet is given an identifying word with value 0x1000 and the second an identifying word with the value 0x2000. The fragment shows the creation and configuration of a CAEN V785 adc, as well as the syntax to insert it an other modules into the packets, and then the packets into the top level event definition.

The module command is extensible. Additional module types can be easily added to the system, as long as their associated unpackers are capable of recognizing and unpacking their data. The extension has since been used in turnkey systems delivered to research groups at the Lawrence Berkeley National Laboratory 88" cyclotron, Rensellear Polytechnic University, Triangle University National Laboratory, University of New Hampshire, and within the NSCL, used in the single event effect test beamline, as well as in the MoNA and Sweeper magnet data acquisition systems.

```
module first     packet id 0x1000
module second packet id 0x2000

module adc1 caenv785 slot 5 crate 0
adc1 config parameters {ring1_0 ring1_1 ring1_2...
                              ... ring1_31}
...
#   Define and configure modules adc2, adc3 and adc4
...
first      add adc1 adc2
second add adc3 adc4

unpack add first second
```

Example 2 Sample unpacking definition script

The NSCL Sweeper magnet collaboration has extended the module command to encompass special purpose gate array driven acquisition modules as well as, as the the CAEN V1290 multihit Time digitizer.

VII. Experience, Status and Availability

NSCLSpecTcl is the laboratory standard online and early stage data analysis package at the NSCL. NSCLSpecTcl has been used in all of the experiments run the NSCL for online data analysis since the commissioning of the A1900 in 2001. NSCLSpecTcl is also used by most experiments for early offline analysis.

The MSU board of trustees released NSCLSpecTcl to the GPL. The software is currently in use at about 20 other institutions in the U.S. and Europe. Wiener Plein-Baus recommends NSCLSpecTcl, the NSCL data acquisition system and the scripted unpacking extension of NSCLSpecTcl to its U.S. customers of CAEN electronics. The software is freely available for download and installation from SourceForge at: http://www.sourceforge.net/projects/nsclspectcl.

VIII. References

[1] J.K. Ousterhout TCL and the Tk Toolkit Addison-Wesley 1994
[2] http://www.nscl.msu.edu
[3] http://www.nsf.gov
[4] 2002 NSAC Long-Range Plan for Nuclear Science http://www.phy.anl.gov/atlas/NSAC_transmittal_letter.htm
[5] DOE Publication: Facilities for the Future of Science A Twenty-Year Outlook http://www.science.doe.gov/bes/FFS_10NOV03.pdf
[6] T. Glasmacher et al. http://groups.nscl.msu.edu/gamma/research/gedets_description/home.html
[7] D. Bazin et al. http://groups.nscl.msu.edu/s800
[8] H. Jonstad *Physics Analysis Workstation* IEEE Trans. Nucl. Sci. V36 No. 5 1989 pg. 1568 See as well R. Brun et al. http://paw.web.cern.ch/paw
[9] R. Brun et al. http://root.cern.ch
[10] M. Goto http://root.cern.ch/root/Cint.html
[11] R. Fox et al. *The SpecTcl analysis system* in Proc. of the IEEE Real-Time Conference, Montreal, QC, Canada, March 2004.
[12] (unattributed) Programming with MFC Microsoft Press 1995
[13] A. Nye Xlib Reference ManualO'Reilly & Assoc 1990
[14] D. Flanagan X Toolkit Intrinsics Reference Manual O'Reilly & Assoc. 192
[15] P.M. Ferguson and D. Brennan Motif Reference Manual O' Reilly & Assoc. 1993
[16] The Fox Toolkit homepage: http://www.fox-toolkit.com
[17] Qt homepage: http://www.trolltech.com
[18] Gtk homepage: http://www.gtk.org
[19] D. Radford et al. http://radware.phy.ornl.gov
[20] T. Baumann et al. http://groups.nscl.msu.edu/mona
[21] D. Bazin http://docs.nscl.msu.edu/daq/appnotes/TreeParameter.html
[22] R. Fox http://docs.nscl.msu.edu/daq/scripted/ScriptedReference.pdf
[23] D. Mussser, G.J. Derge, A. Saini STL Tutorial and Reference Guide, Second Edition Addison Wesley 2001.
[24] C.R. Gould, N.R. Roberson *Vax 11/780 Data Acquisition Facility at Triangle Universities Nuclear Laboratory* IEEE Trans. Nucl Sci. Vol 30 No. 5 Oct. 1983 pg. 3758
[25] http://www.nirs.go.jp
[26] R. Fox, A. Vander Molen *The Xamine Online/Offline Display Program* IEEE Trans. Nucl. Sci Vol 43 No 1 pg 55.
[27] See e.g. Technical Information Manual Mod. V785 32 channel Peak Sensing Converter CAEN 2000
[28] See http://groups.nscl.msu.edu/s800