The Vme Package at the NSCL; Large leverage From a Small Extension.

R. Fox
National Superconducting Cyclotron Laboratory
Michigan State University
East Lansing, MI 48824-1321

***Abstract*—IEEE 1014-1987 describes a backplane standard called *VME bus*. This standard is widely used in experimental Nuclear, and High Energy Physics. This paper will describe the backplane and its capabilities, describe a small Tcl extension that provides Linux user level software access to modules installed in one or more VME backplanes and the use that has been made of this extension. While the extension is quite small (~500LOC C++), users at the NSCL and elsewhere have used it to build complex device control applications with relative ease.**

I. Introduction and Outline

The IEEE P1014-1987 standard[1] defines a computer and peripheral backplane known as the VME bus. This standard is heavily used in experimental nuclear and high energy physics. It was a widely adopted as a successor to the CAMAC[2] instrumentation bus in the mid 1980's.

The remainder of this paper will be organized as follows;

1. First I will provide a brief history, and description of the VME bus and its capabilities.
2. I will then describe how the VME bus is used at the National Superconducting Cyclotron Laboratory (NSCL) at Michigan State University, and the problems I was trying to solve when I developed the Vme package.
3. Next I will describe the extension itself, and the capabilities it provides.
4. I will then describe some of the applications in daily use at the NSCL that depend on this extension.
5. I will conclude with a brief description of the implementation status, describing the environments and interfaces to which this extension has been ported.

II. An introduction to the VME bus.

A very good, brief history of the VME bus is provided at [3,4 VME stands for VersaModule Eurocard. It is a direct descendent of the VersaBus defined at Motorola during its development of the MC68000 family of microprocessors in the 1970's.

VME adds to the VersaBus the adoption of the Eurocard form factor as well as the replacement of edge connectors with more reliable DIN connectors. In addition, the VME bus is an inherently multi-master, centrally arbitrated bus.

The VME bus achieved IEEE standardization in 1987 as IEEE P1014. At that time it was specified to have a maximum address width of 32 bits and maximum data transfer width of 32 bits. A subset backplane was supported by the standard as well which limited Addressing to 24 bits and data transfer widths to 16 bits.

Since then, continued evolution of the VME bus, managed by VME International Trade Association (VITA) working groups have produced: a 64 bit version of the bus (VME64), several auxiliary bus specifications (e.g. VSB and VXI), mezzanine card interface specifications, and a switched backplane fabric intended for Advanced Telecommunications Computing Architectures (ATCA) VITA 41.

This paper will be concerned with only IEEE P1014 also known as VME Revision C. I will use the acronym VME to refer to this standard.

A. The Physical Specification of VME

A typical 6-U VME crate and backplane are shown in Figure 1. below:

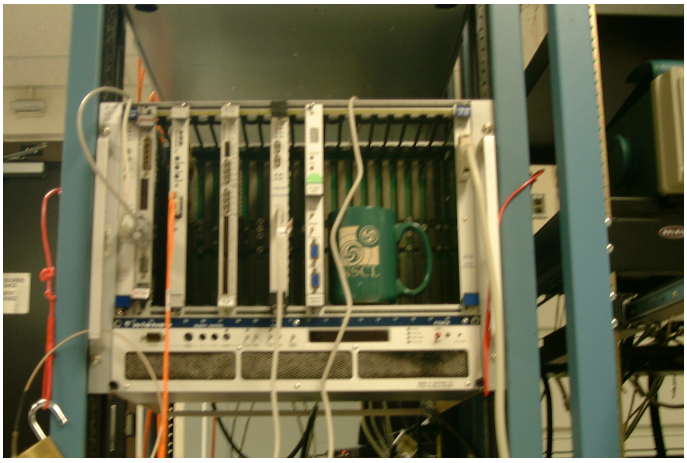

Figure 1 A typical VME crate

The VME bus has been implemented using the 6-U form factor shown in Figure 1 as well as a 9-U form factor (most commonly used in the VXI instrumentation bus extension), and a 3-U form factor (which reduces the address width to 24 bits and data transfer width to 16 bits). A 6-U card is 233.35mm high by 160mm deep providing approximately 373.4 mm^2 of board space.

Barely visible in the back of the card cage are the two 96 pin DIN connectors per slot that carry the VME bus signals. All three rows of the top connector are committed to VME bus traffic, while only the central row of the lower connector is committed. The outer pair of rows of pins in the lower connector is left uncommitted and can be used either for module I/O requirements or for auxiliary busses. A typical VME crate backplane fabrication process does not trim the leads of the DIN connectors allowing I/O connectors to be pushed on the rear of the card cage.

VME card cages are limited to at most 21 cards. The backplane is required to contain specific termination and pull up networks to support a clean open collector bus without the need for individual pull up and termination networks on the cards themselves.

Almost all signals on the backplane are bussed.

B. Multi-Master Bus arbitration

The VME bus is an inherently multi-master bus. A bus arbiter, installed in the left most slot; slot 1 determines which of the simultaneously competing masters gains control of the bus. VME supports two forms of arbitration (Priority and Round Robin), as well as two bus release strategies, Release When Done (RWD), and Release On Request (ROR). While only one arbitration strategy can be used (usually jumper selected on the arbiter), the release strategies can be mixed to optimize the expected access patterns of each master.

A master attempts to gain control of the bus by asserting one of four Bus Request lines (BR0 – BR3). In priority arbitration mode, BR3 is considered the highest bus request level and BR0, the lowest, in Round Robin Arbitration, the arbiter polls the four BR lines cyclically. Assuming the bus is free, the arbiter signals which master has gained access by replying to a bus request with by asserting the corresponding Bus Grant signal (BG0 – BG3). This signal is daisy chained through the modules so that ties are broken by proximity to slot 1.

The master that was granted control of the bus indicates the bus is busy by asserting BBSY (Bus Busy), and can then perform data transfer cycles. Once the master has completed a data transfer cycle, it can either release the bus (by de-asserting BBSY), or hold on to the bus until the arbiter asserts BCLR (Bus Clear) indicating another bus request (higher priority in priority arbitration) has been asserted. The first option is called Release when Done (RWD), the second Release on Request (ROR). Clearly if a bus master will perform several transactions in a short time, it can amortize the arbitration overhead by using ROR. On the other hand, if a master is not likely to do bursts of data transfers, *other* masters will have smaller bus access latencies if the master releases the bus when done.

C. The Data Transfer bus

The VME bus supports a wide range of transfers and addressing. 31 address lines and 5 *address modifiers* define both a transfer address and address space. Two data strobes (DS0*, DS1*) and an LWORD* signal define transfer width and offset information.

The address modifier lines select an address space and a transfer type. The predefined address spaces have different address widths and are given in Table I below:

Table 1 VME address spaces

Address Space Name	Significant bits of Address
Short IO	16
Standard	24
Extended	32

Most recent boards support either Extended or Short IO space. The Standard address space is a relic of the 3-U form factor VME cards.

A typical data transfer cycle, is shown in Figure 2 below. This figure includes the initial arbitration for bus master. Minor division of the time scale are 100ns.

Once the master has asserted BBSY (shown as BUSY*) it proceeds with an address cycle. The address (0x00500000) is presented on the address lines, and an address strobe (AS*) is asserted when the address bits have stabilized.

Note that in Figure 2, BBSY* is held true indicating that this master is using release on request bus release. It is holding the bus busy and owned in anticipation of another bus transaction in the near future.

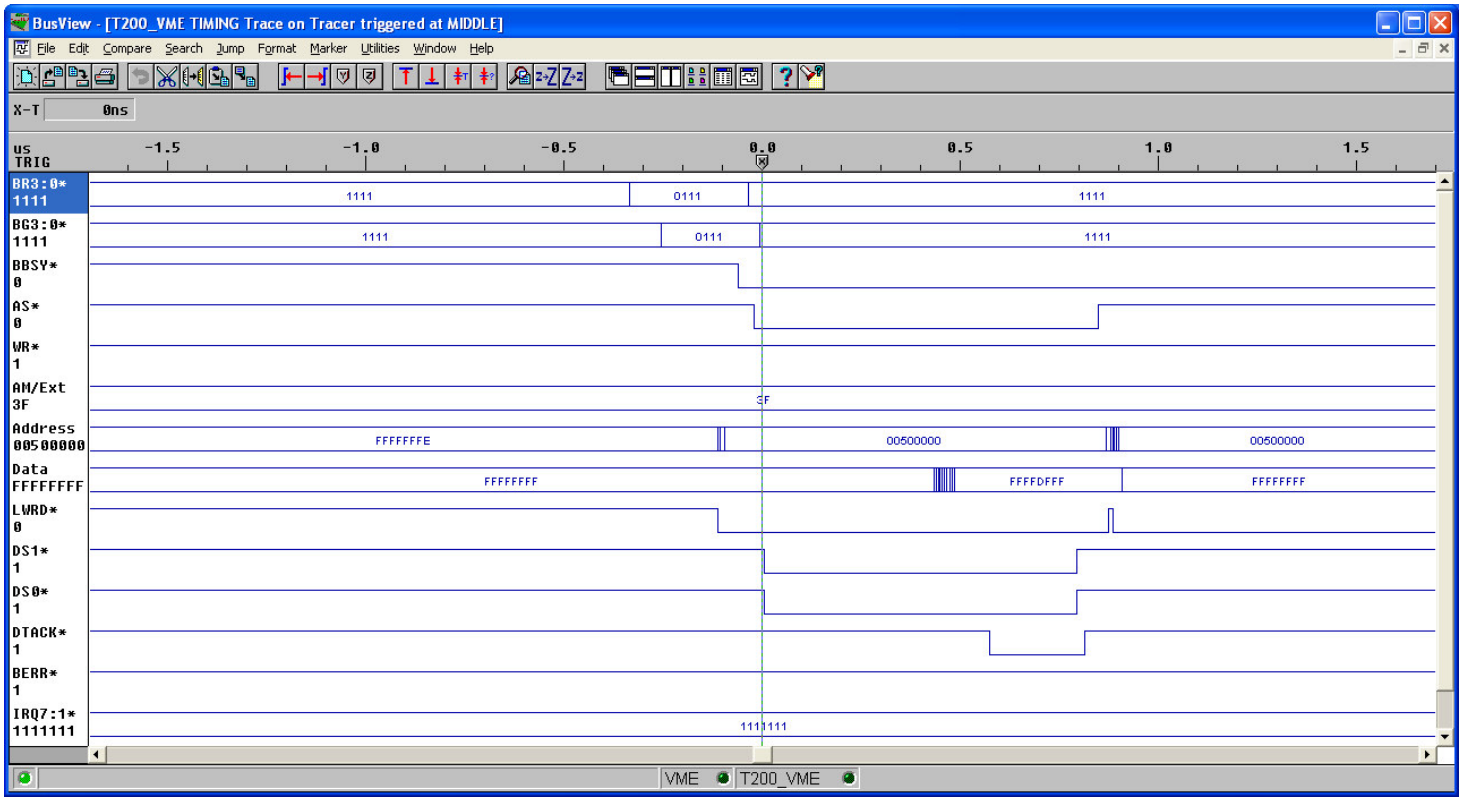

Figure 2 A VME bus read from 0x05000000

At the same time LWORD is asserted indicating that this is a long word transfer. Following LWORD both data strobes are asserted by the Master indicating it is ready to accept data. The target board, places data on the bus and, when that stabilizes asserts DTACK (Data Transfer ACKnowledge). When the master has accepted the data, it releases the data strobes, the slave releases DTACK and the cycle completes when the master releases the address strobe.

If this were a write cycle, the DS0*, DS1* data strobes are also used to indicate that the master has stabilized the data being written to the slave on the data bus. The DTACK line in that case is used by the slave to indicate that it has received the data and that the master can float both the data and the data strobes.

D. Interrupts and interrupt handling

The VME bus supports prioritized, vectored interrupts. Seven interrupt priority levels each support up to 2^{32} vectors (there are D8, D16, and D32 vectors). In practice, most interrupters only use D8 vectors, providing 256 status/ID codes for each interrupt priority level.

An interrupt cycle takes place between an interrupter and an interrupt handler. A backplane can have more than one interrupt handler. The interrupter signals an interrupt by asserting one of the interrupt request lines (IRQ1-IRQ7). The interrupt handler for that interrupt priority level first arbitrates for the bus as for any normal data transfer operation. Once the interrupt handler has gained control of the bus, it asserts the interrupt acknowledge line (IACK), and, using AS*, strobes the interrupt priority level it is responding to on the address

bus lines A1-A3. It then initiates data strobes appropriate to the width of the interrupt status/ID it is prepared to handle.

The interrupter responds by using DTACK to strobe the status/ID on the bus, where it is read by the master, which then removes IACK. Note that the interrupt acknowledge cycle is essentially identical to a VME bus read cycle. The address, however is the interrupt priority level, and IACK is asserted.

IACK is a daisy chain line, like the bus grant lines. Therefore, if more than one interrupter is active simultaneously on the same interrupt priority level, the one closes to the interrupt handler is serviced. Note that the daisy chain additionally requires that interrupt handlers be installed in the bus to the left of interrupters that they handle (each module receives IACK-in from the slot on its left and sends IACK-out to the slot on its right).

III. The VME bus at the NSCL

The VME bus has been in use at the NSCL since 1985[5]. First used in data taking applications, the NSCL accelerator controls group next adopted first the FNAL VME bus LINAC control system [6], and later the VME based EPICS[7] control system.

Use of the VME bus for experimental device controls is, however, comparatively recent. With the completion of the NSCL coupled cyclotron upgrade, and corresponding data acquisition system upgrade[8], commercial VME based electronics that met the needs of the NSCL experimental control had become increasingly available. Of particular interest to NSCL users were and are the following devices:

- VME based detector Bias supplies.
- VME based constant fraction discriminators
- VME based CAENnet controllers (CAENnet is a proprietary serial device control network that is used primarily to control CAEN's series of shaping amplifiers, and card-cage based multi-channel detector bias supplies. Do not confuse CAENnet with CANbus.
- VME based re-programmable FPGA modules.
- VME based gate and delay generators.

These devices are normally integrated into what experimental nuclear and particle physicists call the experiment *slow control system*. The name comes from the fact that the timing of interactions with these devices is on the order of human interaction timings (1/30'th of sec latency), as contrasted with the timings required of interactions with devices that contribute to the experimental data flow (microseconds).

Users of these devices need a simple way to build control software that is experiment specific. An extension that allows Tcl to talk with the VME bus coupled with pure Tcl module support packages, and Tk mega widget control panel elements seemed like a natural fit to this problem because:

- NSCL physicists are already familiar with Tcl/Tk[9].
- The timing requirements of these devices are quite loose.
- Each experiment or experimental setup has experiment specific requirements and "desirements" for control applications that are best realized by the experimental groups themselves rather than software developers that are not interacting on a daily basis with their devices.

IV. A description of the extension.

In this section I will describe the low level Tcl extension that was written to provide access to the VME bus. First I will describe the VME facilities I wanted to support, and second, the syntax and semantics of the extension. Finally some simple examples will be given.

A. *VME Facilities covered by the extension*

While many of the modules used in control applications can be configured to be interrupt requesters, this facility is not necessary for use in slow controls applications. Furthermore, exposing the asynchronous nature of interrupts to physicist programmers would open a can of worms that is similar to but worse than the issues faced by programmers of threaded applications.

The extension therefore only supports the data transfer protocols of the VME (arbitration is implemented in the hardware of bus masters which is not exposed to the software). As section II describes, each address on the VME bus is accompanied by an address modifier. The address modifier serves to qualify the address, by selecting an address space. Section II described the standard address modifiers defined by VME Revision C; Short IO (also called A16), Standard (also called A24), and Extended (also called A32).

Extensions propagated by the European Nuclear Physics Laboratory (CERN), also provide support for geographical (slot number) addressing. This is done by providing an additional connector on the backplane that provides, among other things, an encoding of the slot number. Geographical addressing requires this special backplane and, of course, modules that support it. Since NSCL physicists were

migrating to VME from CAMAC, an instrumentation bus that is fundamentally geographically addressed, where possible, the NSCL has purchased modules and backplanes that support this CERN extension. As a side note, the VME64 standard defines a mechanism for geographical addressing, however it is implemented differently than the CERN extensions. Geographical addressing coupled with defined manufacturer and model registers provide support for auto configuration of VME systems.

Each VME module generally responds to one or more contiguous blocks of addresses in one or more address spaces. When geographic addressing is implemented, it is implemented as a base addresses recognized by the module are a set of computable functions of the slot number.

These considerations led to the following informal specification:

- The extension supports the creation of blocks of address space within an address modifier.
- Once created, the extension supports 8, 16, or 32 bit reads and writes to arbitrary addresses within an block address space.
- Attempts to access locations outside the block result in an error.
- The extension supports the destruction of existing address blocks.
- The extension supports introspection of existing address blocks.

An address block is also called a *map* since it mimics a mapping between process address space and VME bus address space.

B. Syntax and semantics of the extension

The extension is implemented as a dynamically loadable package named **Vme**. The extension consists of a single command ensemble. The command is an *generating* command. That is it generates other commands. The ensemble base command is **vme** and the syntax summary for this command is shown below:

```
package require Vme

vme create mapname -device  modifier  \
                        ?-crate cratenum?  \
                         base size
vme list
vme delete mapname
```

Example 2 Syntax of the vme ensemble commands.

The **create** subcommand creates an address map. the **–device** switch is required and its parameter specifies a mnemonic for the address modifier to be used when accessing the VME addresses represented by this map. Allowed values for the *modifier* switch value are; **shortio**, **A16**, **standard**, **A24**, **extended**, **A32**, and **geo**. The *base* parameter is the VME base address of the map, and the *size* parameter the size in bytes of the block. *mapname* is a name given to the address map. A new Tcl command *mapname* will be created by this command if successful. The optional **–crate** switch allows the map to be created in any VME crate connected to the host system (larger experiments at the NSCL may require as many as 5 VME crates).

The **list** subcommand lists the vme maps that have been created. The information is returned as a list of two element sub lists. Each element is a list containing the name of the map and its base address.

The **delete** command releases all resources associated with a map and destroys the command that is used to access it.

Once a map is created it too is a simple ensemble command:

The **get** command performs a VME bus read from the *offset* relative to the start of the map. One of the switches **–b**, **-w**, **-l**

```
mapname  get -b|-w|-l offset
mapname  set -b|-w|-l offset value
```

Example 1 Accessing address maps.

is required and is the width of the transfer (8, 16, or 32 bits). If the offset is negative or greater than or equal to the *size* parameter of the map, the command fails with an appropriate error result.

The **set** command performs a VME bus write to *offset* writing *value*. The meaning of the **–b**, **-w**, **-l** switches are the same as for the **get** subcommand.

The entire extension is implemented in approximately 500 lines
of C++ code, in three implementation files implementing two classes and one unbound function.

C. Examples

1) Testing a memory cell with a rotating bit pattern

```
package require Vme

vme create mem -device standard 0x500000
0x4

set mask 0x80000000
while {$mask} {
    mem set -l 0 $mask
    set is [mem get -l 0]
    set is [format 0x%x $is]
    if {$mask != $is} {
       error "Should be $mask Was $is"
    }
    set mask [expr $mask >> 1]
}
```

Example 2 Testing a single memory cell.

2) Reading the firmware of a CAEN V785 ADC.

CAENV 785 ADCs had several initial firmware issues. The first firmware revision level that actually works is 8.09. This sample test program reads and outputs the firmware revision

```
package require Vme
vme create adc -device geo \
               [expr 10 << 19] 0x2000
set fwreg [adc get -w 0x1000]
set major [expr ($fwreg >> 8) & 0xff]
set minor [expr $fwreg & 0xff]

puts [format "Firmware revision is %d.%02d" $major $minor]

vme delete adc
```

Example 2 Printing the firmware revision of a CAEN V785

level (offset 0x1000 from the module base address) of a CAEN V785 ADC that is inserted in slot 10. The CAEN V785 ADC is capable of geographical addressing following the CERN extension. When geographically addressed, its base address is the slot number shifted left by 19 bits. The firmware register is 16 bits wide. The most significant 8 bits are the major revision, the least significant 8 bits are the minor revision level.

V. Applications Sampler

This section presents a sampler of applications that have been written by myself , physicists at the NSCL and elsewhere.

For the purposes of this section an application is defined to be any "useful" software that has been written on top of the Vme Package.

A. Module Support Libraries

Several module support libraries have been written on top of the simple model provided by the Vme package. These packages are pure Tcl "device drivers" for specific hardware in common use. These include:

- CAMAC interface support modeled on the IEEE ESONE CAMAC functions (IEEE 785) for both the CES 8210 parallel branch highway driver and the Wiener VC/CC32 board set.
- Support for CAENnet messaging with the CAEN V262 VME CAENnet controller.
- Support for the CAEN N568 shaping amplifier (a CAENnet device).
- Support for the CAEN V812 constant fraction discriminator.
- Support for the iSeg dual channel detector Bias supply.

B. Detector Bias supply controls for the NSCL SEETF

The NSCL Single Event Effect Test Facility (SEETF) is a beam line and experimental area that was funded by NASA. NASA and their collaborators use this facility to test the radiation hardening of electronics scheduled fly in satellites. A defocused heavy ion beam of hopefully uniform spatial intensity irradiates electronics installed in the SEETF.

A segmented plastic scintillator is used to continuously monitor both the temporal and spatial uniformity of the heavy ion beam. The gain of plastic scintillators is determined by

the high voltage bias applied to their photomultiplier tubes. The desired gain is a function of the incoming beam particle species and the beam energy. A Parallel Plate Avalanche Counter (PPAC) tracks incoming beam particles prior to the electronics interaction station. It's gain is a function of both the gas pressure in the detector and the HV bias applied to the detector.
In practice, the gas pressure is held fixed while the HV bias is modified to change the gain.

The operating agreement between the NSCL and NASA calls for NSCL support staff to determine the appropriate HV settings for these detectors prior to turning the beam over to the NASA group. Since the NASA scientists will need to control the HV bias supplies, and respond to power supply interlocks, and since the same beam is used for more than one experiment, the NSCL provides two control panels. The first, shown below is used by the NSCL support staff to set the required values of the detector bias supplies several instances of this control panel are shown simultaneously allowing all 5 gains to be adjusted .

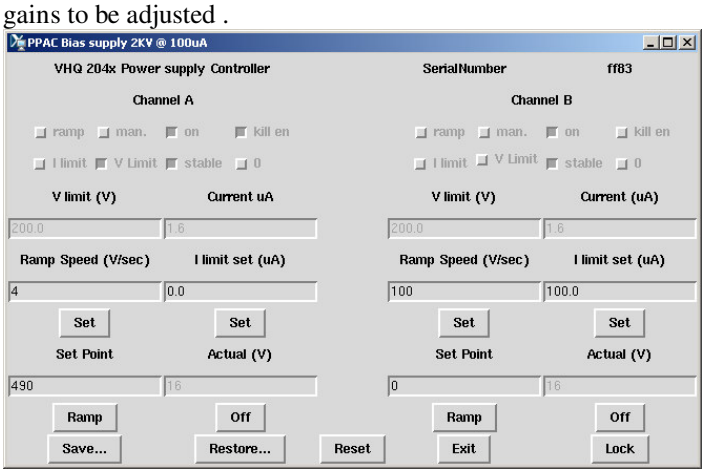

Figure 3 SEETF iSeg bias supply controller (expert panel).

This panel is usable for general purpose control applications making use of the iSeg VHQ dual channel detector bias supply .
The experimenters are presented with the simplified control panel shown below.

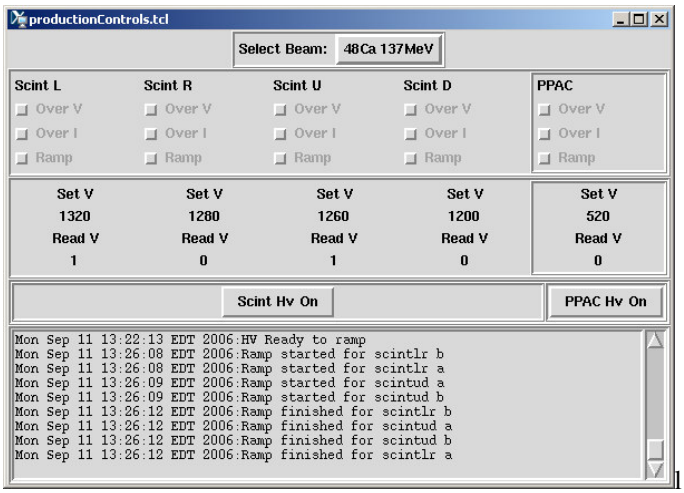

Figure 4 The See user bias supply control panel.

In the control panel in figure 4, the user simply selects the stored settings from a previously run beam species and energy. These settings are loaded into the device set points and the user can ramp the supplies up or down using the buttons on the panel. The log window at the bottom of the panel provides log information about the status of bias ramps.

C. CAEN V812 CFD control panel

Figure 5 below shows a control panel for a single CAEN V812 control panel. The piece of the control panel located below the menu bar is a snit[10] based mega widget that provides control for a CAEN V812 constant fraction discriminator:

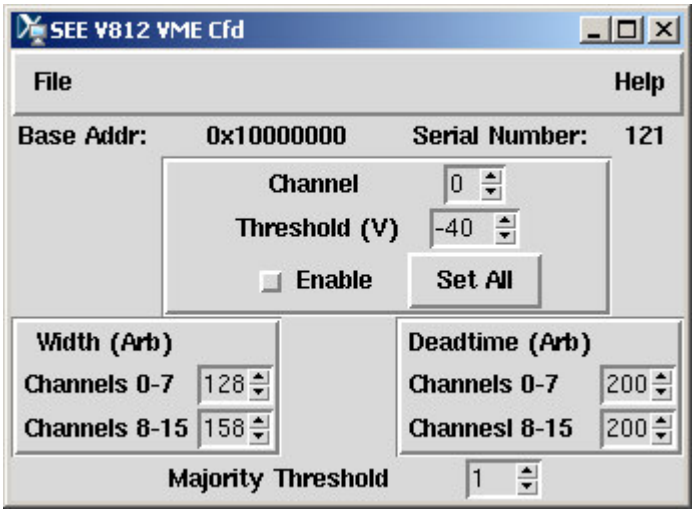

Figure 5 CAEN V812 mega widget.

The control panel is provided as a canned application, with the mega widget available for use as a user interface element in other applications.

D. Mega widget control for CAEN N568 shaping amplifier.

Figure 6 below shows a snit mega widget that can control a CAEN n568 shaping amplifier. This mega widget an be embedded into a user written control panel. An application that provides generic support for controlling this device has also been written.

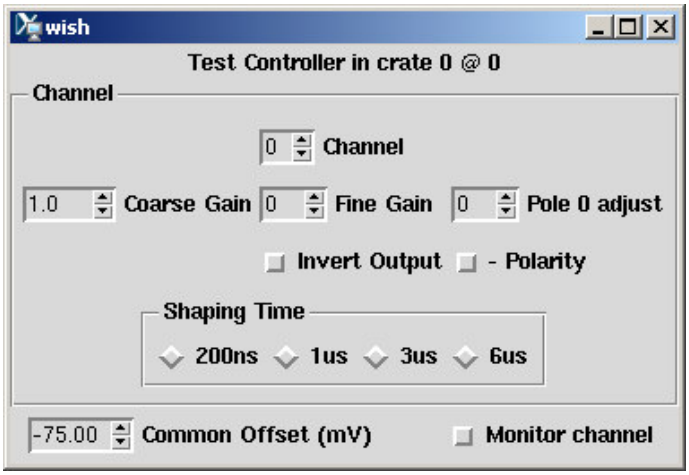

Figure 6 CAEN N568 mega widget

E. iSeg VHQ bias supply meg awidget

A snit mega widget that can control an iSeg dual channel detector bias supply is shown in figure 7:

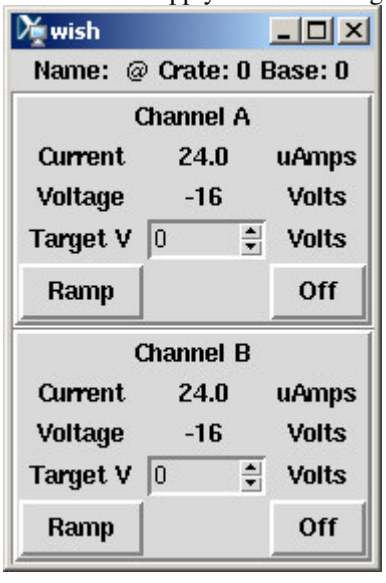

Figure 7 vhq mega widget

This compressed version of the VHQ panel of the SEETF provides a mega widget that can be embedded into a user's control application.

F. The S800 Trigger GUI (D. Bazin)

The S800 spectrograph at the NSCL is the apparatus used by a bit more than ½ of the experiments run at this laboratory. The trigger system for the spectrograph data acquisition system is based on a LeCroy ULM 2367. The ULM 2367 is a CAMAC module that contains a CAMAC programmable FPGA. The trigger firmware has several dynamically configurable parameters, and these are controlled by a GUI written by Daniel Bazin, shown below[11].

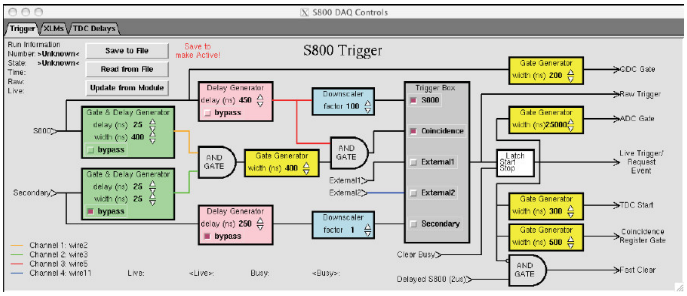

Figure 8 The S800 Trigger GUI.

This GUI presents the experimenter with a direct manipulation interface on top of a block diagram of the trigger logic.

G. MoNA triggers system (W. Peters and T. Baumann)

The XLM-72 is VME based FPGA module is the trigger used for the **Mo**dular **N**eutron **A**rray (MoNA) trigger subsystem[12]. The MoNA trigger system is a multilevel trigger using three FPGA modules. As with the S800, the trigger has several dynamically adjustable parameters. The control panels for this trigger system are shown below:

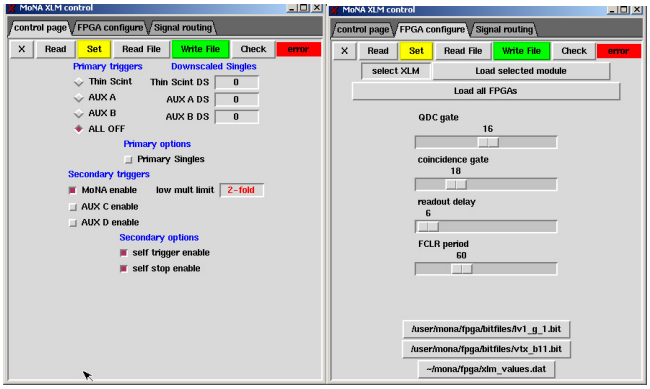

Figure 8 MoNA Trigger controls.

H. VME Pulse generator control (K. Minamisono)

The Beta end station detector package is used for a wide variety of experiments involving nuclei that decay through the emission of beta or positron particles. The magnetic moments of unstable nuclei are studied using nuclear magnetic resonance techniques. When studying quadrupole moments, it is necessary to control the NMR sweeps by using several frequencies and duty cycles[13]. This is accomplished via a VME programmable function generator that is essentially a lookup memory, sequencer and DAC. The control panel for the function generator, written by K. Minamisono is shown below:

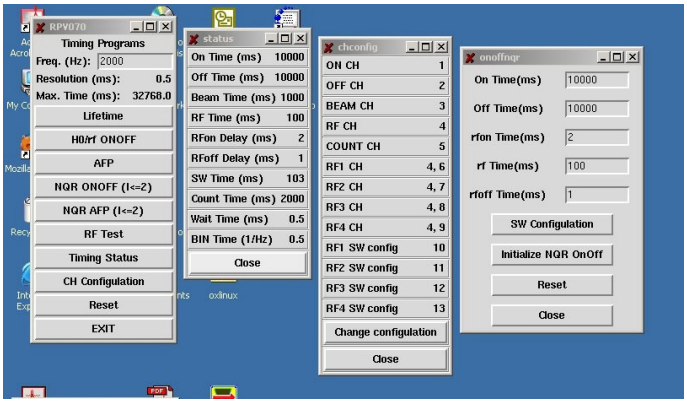

Figure 9 Control of the Beta NQR system via function generators

I. Implementations

The NSCL primarily uses commodity Linux server systems in experimental data acquisition systems. This is in marked contrast to many other laboratories that use embedded VME processor boards. Our reasons for this choice is documented elsewhere [8].

Our choice of commodity 'boxed' Linux server systems also provides us with an additional choice. Which PC/VME interface product to use to connect these systems to their VME subsystems. At this time, the NSCL uses the SBS model 620(-3) PIC/VME bus bridge. This product includes an on-board DMA engine, and mapping registers that allow PCI address space to be mapped directly to VME address/address modifier spaces. The interface connects to a VME bus master card via a fiber optic cable which provides both electrical and spatial isolation from the radiation vaults (where high magnetic and radiation fields can be found during some runs).

The author, in consulting projects for other laboratories and users, however, has had the opportunity to use other PC/VME interfaces. Each of these interfaces has its own set of strengths and weaknesses, and it is beyond the scope of this paper to detail them at this point. I will provide a list of the interfaces to which the extension has been ported. Since the NSCL data acquisition system makes use of a VME interface abstraction layer, and since the Vme extension makes use of this abstraction layer,

The Wiener PCIDA/VC32 module is a PCI/VME interface that couples a PCI card to a VME bus master via a SCSI-2 cable (the cable does not run the SCSI protocols however). The module requires interactions with a device driver for each VME bus data transfer or block transfer. Under contract to Wiener Plein-Baus in support of a project done for Rensselaer Polytechnic Institute's Department of Mechanical, Aerospace & Nuclear Engineering, support for this interface was written. This support was completed in 2004, and the isolation of the extension from the VME interface's API via the VME abstraction layer made possible the use of the Vme extension under this system. Control panels specific to this project were in fact built on top of Vme.

The Jtec VM-USB is a VME bus master with a USB interface. The VM-USB is based on a Xilinx Spartan-III FPGA and provides the capability for USB directed single shot VME operations as well as the capability for autonomous data taking by associating lists of VME operations with specific trigger conditions. Support for single shot operations was completed in late 2005 through an implementation of the VME abstraction layer for this device. Once this was done, the Vme package automatically worked. This work was done under contract to Wiener Plein-Baus which now uses the Vme package to produce control panels for demonstrators of their nuclear electronics. A project for Lawrence Livermore National Laboratories is in progress which will make use of the list mode capabilities of this interface to do high performance data acquisition in support of a neutron imaging system.

Other VME interfaces the author is aware of are:

- The SIS 3100 PCI/VME fiber optic interface. No effort has been made to support this interface at this time.
- The SIS 3150 USB VME interface. Support code has been written under contract to SIS Gmbh and completed in mid 2006. This contract, however was to support their windows APIs under Linux. No funding was provided to implement the VME Abstraction layer on top of that API, however SIS has indicated it might be interested in that as the Linux customer base for this device grows. A Tcl extension specific to this module was written both for

test and demonstration purposes, however this extension is not compatible with the Vme extension.

- The CAEN V1718/V2718 modules are USB and fiber optic controlled bus masters. The fiber version interfaces to the PC via a PCI fiber driver card. The card has a site for an FPGA daughterboard which is intended to contain user written application specific firmware to provide application specific intelligence at the VME side of this interface, this daughterboard has not yet been produced. CAEN has expressed interest in supporting this board under the NSCL data acquisition system (and hence the Vme extension), however while they have initiated negotiations to support this development effort, they have never followed through to a contract.
- National Instruments has a series of VME bus interfaces ranging from PCI controlled to GPIB controlled. At this time no effort has been made to make these devices usable under the NSCL data acquisition system or the Vme package.

VI. References

[1] Standard online at http://tesla.desy.de/doocs/hardware/pci/pci21.pd

[2] IEEE Std 583-1975 IEE Standard Modular Instrumentation and Digital Interface System (CAMAC)

[3] http://www.vita.com/jubilee/earlyyears.html

[4] http://en.wikipedia.org/wiki/VMEbus

[5] See, for example A. Vander Molen, R. Au, R. Fox, M. Maier, and M. Robertson "Status of the NSCL 4π data acquisition system" *IEEE Trans. Nucl Sci.*NS 36 No. 5 1559-1561 (1985).

[6] R.W. Goodwin and M.F. Shea "Modern Control Techniques for Accelerators" *Tenth International Conference on Cyclotrongs and their Applications*, IEEE Catalog No. 84CH1996-3 pp 547-558, 1984

[7] L.R. Dalesio, M.R. Kraimer, A.J. Kozubal EPICS Architecture available from the EPICS website: http://www.aps.anl.gov/epics/EpicsDocumentation/EpicsGeneral/EPICS_Architecture.ps

[8] R. Fox, E. Kasten, K. Orji, C. Bolen, C.Maurice, J. Venema "Real-Time Results without Real-Time systems" IEEE Trans Nucl. Sci. NS 51, NO. 3, June 2004 pp 571-575

[9] R. Fox, C. Bolen, K. Orji, J. Venema "SpecTcl a Nuclear Physics Data Analysis Package" Tcl 2004 (New Orleans) http://www.tcl.tk/community/tcl2004/Papers/RonFox/fox.pdf

[10] W. Duquette is the author of Snit which is now part of tcllib. For one description of the snit syntax, see http://tcllib.sourceforge.net/doc/snit.html

[11] D. Bazin *S800 Spectrograph Service Level Description* online at http://www.nscl.msu.edu/tech/devices/s800/sld.pdf#search=%22s800%20trigger%22

[12] T. Baumann et al. *FPGA-based trigger logic for the Modular Neutron Array (MoNA)* Presented at American Physical Society, April Meeting, 2004, May 1-4, 2004, Denver, Colorado April 2004

[13] Minamisono et al. *Nuclear Magnetic Moment of the ^{57}Cu Ground State* Phys. Rev. Lett. **96**, 102501 (2006)