The NSCL Control System

J. Vincent
L. Foth, A. McGilvra, J. Priller
National Superconducting Cyclotron Laboratory
Michigan State Unviersity
East Lansing, MI 48824-1321 USA
MSUCL-1007

Reprinted from
IEEE TRANSACTIONS ON NUCLEAR SCIENCE
Vol. 43, No. 1, February 1996

The NSCL Control System

J. Vincent
L. Foth, A. McGilvra, J. Priller
National Superconducting Cyclotron Laboratory
Michigan State University
East Lansing, MI 48824-1321 USA

Abstract

This paper describes the historical development, current state, and future plans of the computer control system of the National Superconducting Cyclotron Laboratory (NSCL). A proposal entitled, "Proposal for a National Facility for Research with Heavy Ions using Coupled Superconducting Cyclotrons" was submitted to the National Science Foundation in September, 1976. The proposal was approved and the first cyclotron, the "K500", came on-line with a limited experimental program in kte 1982. The second cyclotron, the "K1200" (formerly called the K800), and another limited experimental program came on-line in early 1988. In late 1990 the entire facility came on-line along with a full experimental program. Up to now, no formal paper has been submitted describing the computer control system for this facility; this paper seeks to fill that void.

I. INTRODUCTION

The accelerator, beamline, and utility control system used at the National Superconducting Cyclotron Laboratory, collectively referred to as the NSCL Control System (NCS), consists of three distinct subsystems of computer control equipment. The subsystems include: (1) a VMEbus-based and ARCNET-interconnected network of date acquisition and distribution nodes; (2) a system of Modicon Programmable Logic Controllers (PLCs) interconnected on a Modbus Plus network; and (3) user interface consoles and servers consisting of Intel-based microcomputers and Digital Equipment Corporation VAX minicomputers and workstations. Wii the exception of interlock conditions, the system is a supervisory system, since feedback control is done with custom analog/digital circuitry or processors embedded in the equipment. The VME-based computers are running software originally developed at Fermi National Accelerator Laboratory (FNAL) for the LINAC project. The Intel-based user consoles use the Microsoft DOS version 6.21 end Microsoft Widows for Workgroups version 3.11 (WFW) or the Microsoft Widows NT Workstation or Advanced Server version 3.5 (NT) operating systems. The VAX minicomputers and workstations use the VMS version 5.5 operating system. This paper will provide a brief history of the project and an overview of each of the, major components of the system end their interconnection. The paper will conclude with the future plans es they are envisioned in May 1995 for the system.

II. HISTORICAL PERSPECTIVE

A historical perspective offers an example of how dramatic changes occurred to control system plans and designs already in progress, caused by tbe "microcomputer revolution" beginning in the middle to late 1970's and proceeding to this day. Due to this revolution, the K500 Control System (K5CS), which was intended for the entire facility and already partially implemented, was scrapped in favor of a distributed microcomputer approach.

The K5CS was designed and implemented in the late 1970's to early 1980's [1]. This system was based on centralized Digital Equipment Corporation PDP-I 1 computers, CAMAC, a custom "Auto-Poll& Bus" (AP), and a Modicon PLC (K500 PLC). This system used centralized databases and control program(s) which were the normal techniques et the time.

The K500 PLC was used to control and monitor the states of the equipment as well as to manage complex interlock and sequencing operations. The PLC was programmed based 0" a relay ladder logic metaphor, which is a natural and familiar method for the individuals responsible for accelerator equipment operation. Although tbe K500 PLC was an industry-hardened and easily programmed device for state controls, its I/O was expensive, it supplied no general data acquisition capability for analog data, it was relatively slow and it bed no user interface capability.

The K5CS supplied general data acquisition and control, not handled by the PLC, and also supplied tbe user interface controls for the entire control system, including the K500 PLC. The custom-designed AP bus was used to interconnect custom-designed device controllers and user interface modules to CAMAC crates, which were then interconnected on a CAMAC serial highway to the PDP-11 computers. The user interface controls consisted of various varieties of multiple meter modules, knob modules, multiple button modules and associated status lights. The user controls were multiplexed to the various controllable axes in a manner which was predetermined by programs running on the PDP-11 computers. The K5CS interacted with tbe K500 PLC using some of the button modules and a PLC I/O link established with a K5CS digital I/O module connected to normal PLC I/O modules. The only graphical display in the system was used to display beam current amplitude versus cyclotron radius, which was acquired with a moving beam probe. The central PDP computers, programmed in assembly language and

FORTRAN, gathered, processed, and distributed all of the data through CAMAC and the AP bus.

A. The New Direction

In the early 1980's, when the K5CS system above was just coming on-line, it became apparent that new industrial busses (such as the VMEbus and the Intel Multibus) and the associated microcomputer and data I/O modules that were becoming increasingly more available were a much more cost-effective, powerful and flexible solution for the control of equipment.

A successful example of the adaptation of this equipment to accelerator control was identified in the FNAL LINAC project [2]. In late 1983, the decision was made to investigate a control system based on an ARCNET-interconnected LAN of VME crates initially using software developed at FNAL [3,4]. The system would make use of VME-based equipment that had been developed at FNAL, including ARCNET LAN boards, a crate utility board, and rudimentary local control consoles. Each local console featured a 5-inch diagonal text mode CRT, keyboard, shaft encoder knob, and local buttons, all of which connected directly to the VME crate through the crate utility board. The system was slated to eventually include a MicroVAX-based main console, and to continue using a Modicon PLC state and interlock controller.

B. Device Interface

The initial method of connecting devices to the control system consisted of a custom-designed embedded controller card based on the Motorola 68701 microcontroller [5]. A serial connection allowed multiple embedded controllers to be daisy-chained on a single serial line to a custom VME-based host controller. The host card contained the processing power to independently maintain the serial communications and data transactions to the embedded controllers. The VMEbus CPU could read and set the embedded controller data via RAM in the VMEbus address space shared with the host controller. In addition to the custom embedded controllers, commercial VME-based D/A, A/D, and digital I/O cards were tested and proposed as viable options.

Between 1984 and 1986, the custom serial embedded controllers, host controller, and direct D/A, A/D and digital I/O VME-based cards were implemented in small scale trials to gain familiarity with the system and to identify potential problems. The trial cases all used the VME local consoles as the operator interface. The initial installations suffered from signal-to-noise problems with the direct connections, and general reliability and maintainability problems with the custom serial connections. However, the VME-based hardware and software posed no great technical problems, and seemed well suited to the task.

Changes were made to the system to address the problems encountered in the trials. Using custom amplifiers to scale the input and output signals to a full +/- 10 Volts at the source, properly managing the system ground, properly pairing the input signals through twisted pair wiring, properly shielding the pairs, and connecting to the VME through differential inputs eliminated the poor signal-to-noise ratio. The 68701 embedded controller system was changed to use an RS-485 industry-standard serial connection and a new "NSCL Serial Protocol", which used readable ASCII text and allowed the use of standard serial protocol analyzers to aid in repair and debugging. The change to RS-485 also allowed the use of a commercially available RS-422 serial board, eliminating the need for the custom host controller board. The success of this installation became a basis for justifying the use of this system for the entire facility [6,7,8].

C. Operator Console Development

Initially, the VME local consoles were used as the control system user interface consoles. A decision to use a VAX cluster with a Digital Equipment Corporation MicroVAX as the server and diskless GPX workstations as the main operator console computer(s) was arrived at in 1987. Serious development on the VAX-based consoles began in 1987, with the first console deployed in 1988 [9,10,11]. A Qbus-to-VMEbus data link was purchased to incorporate knobs, meters, switches, and buttons into the design. In addition, a Qbus ARCNET board was purchased to gather the data from the VME stations. The hardware and software needed to tie this all together into a console was developed over a 6 month period, with the first console installed in 1988.

Although the VAX consoles did accomplish the task they were intended for, the computers and auxiliary equipment were expensive and the software tended to be sluggish. In the meantime, the capabilities and usage of MS-DOS/MS-Windows-based personal computers were rapidly expanding. It became apparent that the most responsive and cost-effective solution for a main console might well be a personal computer running Microsoft Windows [12]. Development of a trial console began in 1992 and was successfully implemented that year. Both the hardware and software were significantly less expensive and much more responsive. The software developed for the project also took a new direction, using the relatively new Object-oriented technology. Application development using the new hardware, development tools, and object methodology significantly reduced the time and cost of the development cycle and is the technique which is pursued at NSCL today.

D. K5CS Integration

As a final note, the remainder of the K5CS was integrated into the NCS through a CAMAC-to-VMEbus translator module, computer programs on the VME crate, and programs on a VAX over a four month period in 1991 [13]. The project minimized the amount of equipment coupled into the original K5CS controllers, and eliminated the AP-bus-based operator console.

III. CURRENT SYSTEM

A. Overall System View

This section will describe each of the major NCS subsystems. Also included is a section on the custom-designed NSCL embedded controllers, which include the previous generation 68701 and current generation Model 400 controllers.

Figure 1 is an overall system view, and Figure 2 details a typical device installation, that of a power supply driving a superconducting magnet.

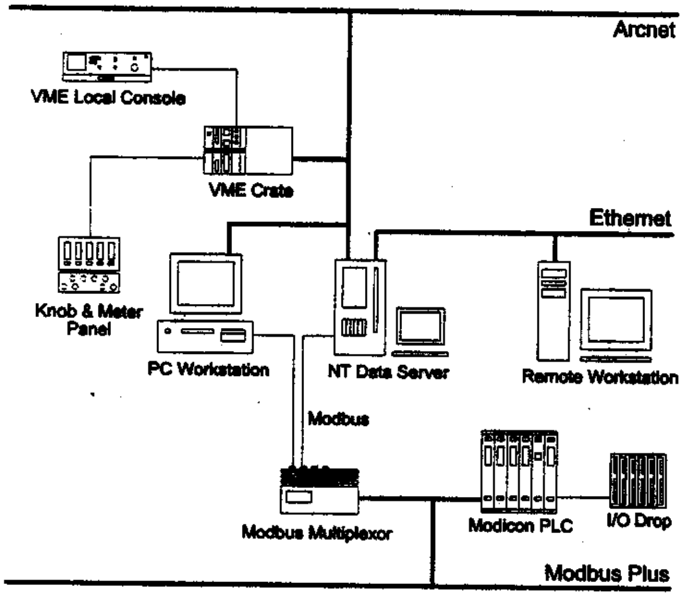

Figure 1. Overall system view

B. VME System

The VME-based portion of the NSCL Control System currently consists of 23 VME stations connected to each other via ARCNET. Fourteen device control VME stations are placed near the equipment they are interfaced to, another six assist the PC consoles, and three are test/development stations. Each station consists of four system boards and numerous I/O boards. The four system boards are: (1) a Motorola 68010 CPU running the Software Components Group's pSOS software; (2) a crate utility board (handling lights, switches, timers and I/O to the local console); (3) an ARCNET network interface; and (4) a battery-backed static RAM board (housing the station's database). I/O boards include analog, digital and serial interfaces to control equipment.

Each station contains all of its code in local memory and maintains its own database of information about the equipment it is interfaced to. Because of this, each station can boot and function stand-alone in case of network failures. The ARCNET is a 2.5 Mbps token-bus peer network. New stations are added to the network with essentially no disruption to existing nodes.

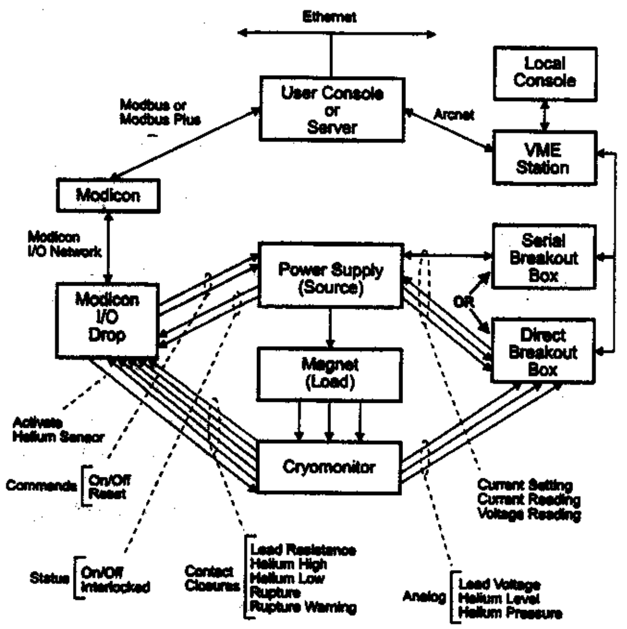

Figure 2. Interface for a typical power supply

B.1. Device I/O

All of the VME station I/O is memory-mapped. The system contains a mix of direct-connect analog and digital I/O, and low-speed serial communications. Short, twisted-pair, shielded ribbon cables carry the direct-connect analog and digital signals from the VME stations to distribution panels called "breakout boxes". Screw terminals on the barrier strips of the breakout boxes provide convenient test points, allowing easy troubleshooting of problems.

Serial communications are used to control many devices where the environment is very noisy, where devices are sparsely distributed geographically, or where some local intelligence near the device is desired. There are two different RS-422/485, 9600-baud, character-oriented protocols supported. One is the Opto-22 Optomux protocol, which interfaces to the K1200 trim coil power supplies. The other, developed at NSCL, communicates with hardware containing embedded Motorola 68701, 68008, or 68332 microprocessors.

The serial I/O (SIO) card runs a multiple-protocol application developed on top of a pSOS kernel. This multiple-protocol application makes it possible to configure which serial lines run which protocols "on the fly". Code for each of the supported protocols resides in EPROM on the SIO card. Configuration tables which specify what protocol runs on which line and what types of devices are on each line, are loaded into the VME station's battery-backed memory. When the SIO board is reset, it copies these configuration tables to its own memory, initializes each port for the correct protocol, and manages all of the communications work, passing data back and forth to the VME station processor via the SIO board's dual-ported RAM. To add devices or change protocols, one merely loads new configuration tables and resets the SIO card.

The current inventory of devices at NSCL consists of 886 analog inputs, 308 analog outputs, 234 digital I/O bytes, and 236 serial I/O devices.

B.2. System Software

Each VME station acquires data from its field devices and monitors them every 1/15 second. During each cycle, the analog and binary data is read into the station's database and checked for out-of-tolerance values. Any new alarm messages resulting from the scan are broadcast to all ARCNET nodes. If other VME stations or workstation consoles request data, the requested readings are prepared and sent out via ARCNET. In addition, the VME station responds to setting requests and performs multi-step control algorithms implemented as state machines. Once all these functions have been completed, an application program executes to service the local console (if one is present) and update its video displays.

The VME database contains device names, analog readings, settings and calibration constants, and digital control and reading characteristics. All of this information is organized in tables accessed through a standard mechanism. Because of the table-driven nature of the software, device calibrations can be changed and new devices of known types added "on the fly". Readings and settings are handled by device drivers, and each device has a code informing the software of which drivers to use.

A local station or control console workstation can examine data elements of any other station by requesting information via a "list". This list specifies which channels of data (or entries in a table) the requester wants, the repetition desired (once only or every 1/15 to 17 seconds) and the order of the data. Lists allow one to request many different types of information for a whole set of channels at the same time. Any station can have a number of requested data lists and a number of lists it is returning data for outstanding at the same time. Cancellation of one list does not effect any of the others.

Figure 3 shows the basic message flow for a Host Console making both a setting and a data request to a VME station. The messages sent by the Host Console are received by the VME station's ARCNET Interrupt Service Routine (ISR). The ISR places the messages into the Message Queue. The Network Task examines the messages in this queue. Setting requests are applied immediately to both the device and the database's copy of the most recent setting for that device. For a data request, the Network Task compiles a set of pointers to the requested data items. Each 15 Hz interrupt triggers a number of actions. First, the reading values of each device are updated in the station's database. Next, any required multi-step algorithms are executed. The Update Task then uses the set of pointers compiled by the Network Task to locate the data requested by the Host Console. These data items are placed in an Answer Buffer in the Output Queue. The NetXmit routine is called to transmit the answers to the Host Console via ARCNET. When the Host no longer is interested in those data items, it cancels the data request list.

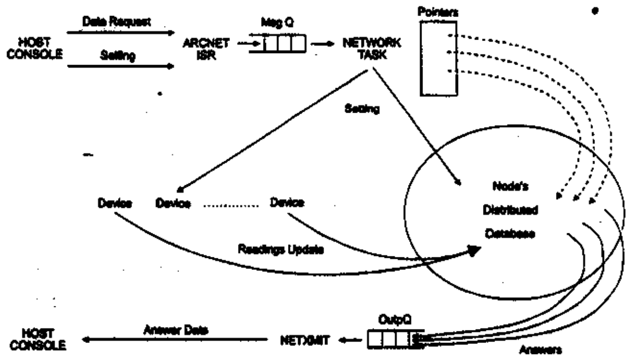

Figure 3. Message flow between host and VME

Each VME station runs identical system software, with the exception of some of the device drivers. This exception is due to the fact that stations have been installed over a number of years, with new device drivers added as new types of devices went into place. Eventually all stations may include all of the new drivers, to make station administration and documentation easier, but this is not required.

Over the years, a number of features have been added to the VME system. They include: the SIO system; support for PC console knobs and meters (to be described later); the ability to handle 32-bit readings and settings; interprocessor communication routines to support the 40/20 main magnet regulation processor; a mechanism to allow board-specific initialization for I/O boards; reading and setting drivers for I/O boards; and state machine routines for devices requiring multi-step algorithms.

The VME-based device interface stations have proven highly reliable and expandable. Due to the battery-backed memory, the stations come back on-line immediately after power failures without human intervention. All in all, these stations are well suited to NSCL's control interfacing needs.

C. Model 400 Embedded Controller

The NSCL Model 400 Controller is a single-board computer designed for embedded control applications. It has two 16-bit analog inputs, one 16-bit analog output, 16 digital I/O points, and an incremental optical encoder interface. The Model 400 communicates with the NSCL control system via an RS-485 serial port.

The Model 400 contains two Motorola microprocessors running at 8 MHz, a 68332 and a 68HC711. The 68332 is the main processor, it carries out communication with the control system, controls the digital I/O and the encoder interface, and runs the application code specific to a piece of equipment. The 68332 has access to 32K bytes of on-board static RAM for data and up to 64K bytes of on-board EPROM for program space.

The 68HC711 runs the analog subsystem. The analog subsystem consists of three 16-bit analog-to-digital converters (ADC) and one 16-bit digital-to-analog converter (DAC). Two of the ADC's are used for inputs and one monitors the DAC output for calibration purposes. The two processors communicate via a Serial Peripheral Interface (SPI) link. The SPI is a three-wire master/slave serial link.

The main motivation for designing the Model 400 was to replace the laboratory's 68701 controller in demanding control applications. The 68701 has only 2K bytes for program space and 128 bytes for data. The analog subsystem of the 68701 consists of the two inputs and one output; however, these were only 14-bits. In addition, the ADC's were implemented using a DAC and software. This implementation made the conversion slow and somewhat less accurate than 14-bits. The Model 400 gives much higher performance and allows for more complex applications.

The software of the Model 400 consists of an assembly language program running on the 68HC711 to control the analog subsystem and a C program running on the 68332. The 68HC711 code does not change from one application to another. The 68332 code is designed to be customized for a specific application. It consists of a rudimentary operating system and an Application Program Interface (API).

The operating system handles the RS-485 and SPI interrupt service routines and controls the I/O hardware directly. The API is a set of functions that the user application can call to access and control the analog and digital I/O, get data from the control system, and send data to the control system.

D. Modicon

The equipment state controls at NSCL are now handled by an interconnected set of eight Modicon PLCs (MPLCs) and sixteen associated I/O drops. A total of 5468 I/O points are currently in use.

The Modicon processors are linked to each other and to the user consoles and servers (via a Modbus multiplexor) with Modbus Plus. Modbus Plus is electrically a 1 Mbps, twisted-pair, balanced network similar to RS-485 in its electrical properties. Both data communications and logic operations can be engaged among all network peers. Eleven Modbus Plus Multiplexors, each supplying four RS-232 Modbus serial ports, are distributed about the building.

The RS-232 connections are configured as 9600 or 19.2 Kbps with either ASCII or a proprietary RTU data format. The communications are based on the Modicon Modbus protocol. The error checking includes parity, and either uses Longitudinal Redundancy Checksum (LRC) when running the ASCII data format, or the Cyclic Redundancy Checksum (CRC) when using the RTU data format. The protocol based on ASCII data encoding is easier for network troubleshooting, but is less compact and therefore less efficient than the protocol based on RTU data encoding.

In addition to PCs and workstation user-interface consoles, a number of industrial consoles including IDT PanelMate and METRA brands are used. The industrial consoles were the only consoles available for user control initially; however, more advanced control software either written by NSCL or available commercially for the PC workstations is now available and installed. As the industrial consoles fail, they are being replaced by PC workstations.

E. Console Computers and Servers

E.1. Hardware

The primary control system consoles at NSCL are Intel-based "IBM Compatible" PCs, using either Microsoft Windows for Workgroups or Microsoft Windows NT as their operating system. The control system PCs interact with both of the primary control subsystems at NSCL, the VME subsystem and the Modicon/PLC subsystem. Beyond a standard PC, the only additional hardware required for a control console PC is an inexpensive ARCNET card.

The VME interface is performed via ARCNET to a VME crate acting as a data server. A data server simplifies the PC's ARCNET-request software considerably, as it does not have to break up data requests or reassemble responses that span multiple VME crates. A VME station also drives the knob and meter hardware panels associated with some control console PCs. This provides considerably better responsiveness than in the previous VAX workstations, where the workstations were responsible for servicing the knobs and meters themselves. The PC can send commands at any time to change how the knobs and meters function, to change what channels are assigned, and to deassign knobs and meters. Each knob and each meter can be individually captured by different applications on the PC (or even by applications on different PCs) for maximum flexibility.

The Modicon interface is performed via a serial line linking the PC to a RS-232 port on a Modbus multiplexor. This serial line interface is comparatively slow (it typically runs at 19.2 Kbps, compared to ARCNET's 2.5 Mbps). The nature of Modicon data is such that it changes slowly, and so a response time on the order of a few seconds is not considered problematical.

E.2. Software

Initial development of the modern NSCL control system software was done for the Microsoft Windows and Windows for Workgroups 3.11 operating environment, using Borland's Pascal for Windows. Native Windows NT 32-bit versions of the ARCNET and serial/Modbus communications drivers and libraries have also been developed for NT machines being used as either control consoles or data servers. This 32-bit NT code was written in C with portability in mind, with all machine-specific code relegated to a single module, so that the primary code modules would compile without modification on VMS, UNIX or any other platform possessing an ANSI standard C compiler. "Thunk" layers, which allow the 16-bit Windows code to call the 32-bit NT drivers, were also written for the NT operating system, allowing control system applications developed for Microsoft Windows to run unmodified on NT-based control consoles.

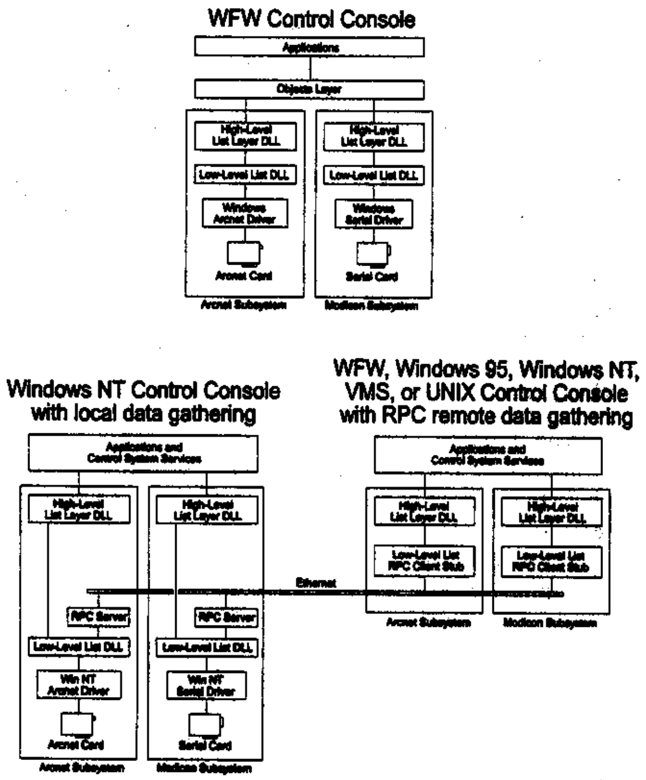
WFW Control Console
Applications
Objects Layer
High-Level List Layer DLL
Low-Level List DLL
Windows Arcnet Driver
Arcnet Card
Windows Serial Driver
Serial Card
Arcnet Subsystem
Modicon Subsystem
Windows NT Control Console with local data gathering
WFW, Windows 95, Windows NT, VMS, or UNIX Control Console with RPC remote data gathering
Applications and Control System Services
RPC Server
Win NT Arcnet Driver
Win NT Serial Driver
Low-Level List RPC Client Stub
Ethernet

Figure 4. Software layers of the control consoles

The software structure of both the ARCNET and Modbus serial interfaces for Windows and Windows NT are essentially the same (see figure 4). Above the hardware itself (either the ARCNET board or serial port) are the Windows or Windows NT device drivers. Serial drivers are provided with the operating system, and the ARCNET drivers were developed at NSCL. Above the driver layer is the low-level list layer, which is responsible for creating and maintaining both repetitive and one-shot device request lists, and notifying clients when their requests complete. Above this low-level list layer is a higher level list layer, concerned with parsing the incoming lists into device and channel-specific data types, scaling incoming data, and providing data-type-specific request lists for applications.

E.3. RPC vs. direct communications

The NT machines contain a Distributed Computing Environment (DCE) compliant RPC server process for each subsystem. These run on an NT data server, just above the low-level list layer. Client machines calling the data server have special versions of their own low-level list layers, which rather than calling the device driver layer, funnel outgoing requests to and incoming data from RPC client modules that communicate over Ethernet with the NT data server's RPC servers processes. This process is entirely transparent to applications programs.

E.4. Object API

Above the highest level layers of both the ARCNET and Modbus communications libraries in the 16-bit Windows code is the Object Layer code. Written in Borland Pascal's object-oriented ObjectWindows, this layer consists of several modules that provide objects for many important control system items, including automatically-repeating request lists for all device-specific data types, interface objects for controlling knobs and meters, and re-useable screen elements for making device assignments, settings, and displaying returned data.

E.5. Application programs

A number of applications have thus far been developed for the NSCL Windows-based control system. These are described briefly below. In general, all applications allow the user to select which VME channels to use, and allow multiple configurations to be saved and restored. Multiple instances of each application can usually be run simultaneously as well.

DDE Server. The Dynamic Data Exchange (DDE) Server process is capable of responding to VME and Modicon data requests made to it by DDE-aware commercial Windows applications, such as spreadsheets, graphing programs and databases. This allows users to imbed live control system data into a spreadsheet and perform calculations with them. Such applications may also perform device settings, allowing a spreadsheet or other program to calculate optimal settings for a given beam and then set beamline devices accordingly. Another possibility would be capturing the device settings used in a particular beamline setup to a database, to either generate a report or re-tune to that beam at a later time.

Chart Recorder. This application allows any eight VME analog channels to be simultaneously monitored, with their readings or settings graphed in a scrolling, chart-recorder-like fashion. The time scale for the graph is adjustable, and each channel has separate and configurable scaling and offset parameters. Pen colors for the eight channels are also configurable.

Knob/Meter Application. The knob/meter application provides an interface to the VME hardware knobs and meters that accompany PC control system workstations. It provides an interface where any desired VME channels can be controlled, VME-channel knob and meter assignments can be made and broken, and where device readings and reading-to-setting differences can be viewed. Several types of knob ganging and banking assignments are provided, as well as user-defined "knob functions" comprised of multiple device channels and scaling factors. Hardware meter ranges can be changed, and device settings can be saved and restored.

PanelMate Emulator. This application was developed in response to the increasing cost of maintaining the proprietary Modicon PLC monitoring and control systems at NSCL. The screen layout of the program mimics the look of the existing hardware, with a simple mouse-based interface replacing the membrane-key interface of the commercial product. Template configuration is done via menus and dialog boxes.

Probe Application. This program allows users to control the beam probes of the K500 Cyclotron, and after the installation of Model 400 boards (described in an earlier section), the beam probes of the K1200 Cyclotron as well. Graphs of several user-selectable beam-current meters can be made simultaneously, with the results of the most recent passes overlaid upon earlier ones to aid in tuning.

IV. FUTURE PLANS

A. VME System

Future goals for the VME system include upgrading and simplifying the system, and making it easier to maintain and expand. Planned upgrades include upgrading the processors from a Motorola 68010 to a 68020 with 68881 floating-point co-processor, and increasing the amount of battery-backed memory available for database tables and specialized device control algorithms. We are also considering using Motorola PowerPC processors in lieu of 68000-series processors. We recently upgraded our development tools to a PC-based package of Microtec Research's assembler, C/C++ compiler, linker and debugger. This allows us to migrate towards C and C++, and to make use of newer, more powerful CPUs that were not supported by our previous tools. We will investigate upgrading our core kernel from pSOS to VxWorks or VRTX, or even higher-end operating systems, to pick up added functionality.

With the development of more powerful console machines, we plan to simplify the VME code by off-loading some functions to the consoles and servers. One job the NT servers will take over is the data server function. No longer will the VME stations need to assemble answer fragments for the consoles. We may even remove the requirement that VME stations be able to request data from each other. This would do away with all of the code for issuing data requests and assembling answers. The VME stations would focus on communicating with devices and responding to data requests from the consoles. All local applications would be phased out and replaced by applications on the consoles. This would also free us from having to maintain the original local consoles which, due to the unavailability of many components, we can no longer manufacture.

With these simplifications to the system design, it will be much easier to re-write the entire system in C and C++. The goal is to encapsulate in C++ objects the heart of the system, the concept of requesting data via "lists", and the table-driven database and device drivers. Reducing the complexity of the system will make it easier to maintain and expand, as will re-writing it in C and C++ (it is currently 100% assembly language).

We are considering a plan in which we would use an NT machine equipped with a PCI-to-VMEbus interface and C/C++ compiler as our development platform. After developing and testing code on the NT machine, we would then port it to a VME-based CPU board running our chosen kernel. Doing the development in this way supplies three advantages. First, the NT development environment is faster and better integrated then the VME development tools. Second, more of the programming staff could participate in the programming by using a more common environment. Third, we would end up with another form of control station, usable in conjunction with a VME crate at the locations that need user consoles and special features, which are more easily developed on a general purpose platform. This new PC-VME hybrid station could supply unique data access services, for example.

B. Modicon

The proposed improvements to the existing Modicon infrastructure include replacing the last remnants of the obsolete hardware of the K500 PLC and possible upgrades and redistribution of existing equipment. The removal of the obsolete hardware will be done as a part of the Coupled Cyclotron Proposal (CCP), if it is funded. The upgrades to existing equipment will be performed if suitable new equipment is developed by Modicon to better match our needs. In particular, we would like more powerful processors with more memory, designed to work in a VMEbus environment. This would allow the control system to access PLC data both through the ARCNET system and via Modbus. In connection with such an upgrade, we would also rearrange some of our current equipment groupings. For example, the K500 and K1200 cyclotrons would each have a processor that controlled all of the PLC processes of the cyclotron, whereas currently these processes are distributed among two processors.

C. Console Computers and Servers

C.1. Hardware

The next hardware upgrade in the NSCL PC control consoles will most likely be a move to dual-Pentium motherboards. This goes hand-in-hand with the move to Windows NT (detailed in the next section), which can take advantage of multiple processors automatically. Dual-processor boards have been becoming less expensive throughout the year, as interest in them for file severs and engineering workstations increases. We would most likely use the boards initially in a single processor configuration, and purchase the additional processor as prices decrease and demand for more horsepower in the control consoles increases.

Another hardware upgrade would be to replace the serial line connection between the NT data servers and the Modbus system with a Modbus Plus network board, which would substantially increase the Modicon data bandwidth. A third-party NT device driver for Modicon's SA85 Modbus Plus card has already been obtained, and preliminary invesitgation shows that it will integrate easily with our existing software.

C.2. Software

The current direction in control system software improvements will likely be moving the main control consoles from Windows for Workgroups to Windows NT. We had at one time considered the intermediate step of Windows 95, but it seems likely this platform will not be as robust as NT, being more closely related to WFW and its VXD driver technology. We are already familiar with Windows NT driver development, and Windows NT would give us the benefit of multi-processor support, true multitasking, and more sophisticated interprocess communication. Recent news from Digital Equipment Corporation mentions the future merge of OpenVMS with Windows NT [14,15], which would allow us to run our control system applications on the numerous VAX workstations in the lab. In order to move the current control programs to native 32-bit NT applications (instead of continuing to use the 16-bit applications), the Borland Pascal object code would need to be either ported to C++, or converted to Borland's Pascal-like Delphi application builder.

Another change being planned is to have as many control consoles as possible make use of the RPC data servers, rather than gathering their data themselves. This would give us the opportunity to overlap common requests from the various control consoles, reducing the traffic on the ARCNET and Modbus networks. It would also allow us to eliminate the data-server code from the VME crates, freeing their resources for device communication. Several data-server PCs could be in operation at any one time, with the remote control consoles using the least-loaded server for new requests, and moving their requests to other servers automatically should one go down.

A new control application is being planned, to replace the VME local console terminals attached to many of the VME crates in the control system. These currently provide a text-based interface to the VME system, and are still used for debugging, device setup, and simple control purposes. The VME Local Console Replacement application would run on Windows and Windows NT control PCs, and provide the same low-level-access functionality with a substantially improved user interface.

V. FINAL REMARKS

The NSCL Control System is believed to represent a modern architecture for accelerator control systems. The system architecture has evolved from totally centralized control, to totally distributed control, and is now progressing back to a middle ground of fundamentally distributed control, with some central control when it can optimize access and management without degrading performance unnecessarily. We have chosen to base it on common industrial products that appear to be gaining dominance. Our projections of future trends and market forces have played as significant a role as technical capability in our architectural decision processes. This approach seems to be yielding increasing rewards, as our functionality is being driven up even as our costs are simultaneously and radically driven down by external market forces.

The standard VMEbus, Modicon PLC, and Microsoft Windows NT operating system forms a powerful triad of control components. We have chosen Microsoft Windows NT as our console computers and data servers because we believe that NT and Microsoft Windows 95 will evolve into a common product over the next 5 years, and will capture an even more significant fraction of the entire operating system market than they now have. Industry literature seems to indicate that large computer companies have drawn similar conclusions [14,15]. With this growth, development tools and commercial applications will also tend to evolve rapidly. In addition, since NT is available on a multitude of platforms, our hardware choices will continue to expand, furthering system flexibility.

Since the accelerator community is small with respect to the global computer market, it should be obvious that our choices as a whole do not have any impact on the computer market; business needs are the dominant influence. Unless we should stumble upon something the industry wants, it should be clear that the industry is not going to bend in the slightest to meet our needs. If "acceptable" power is available from systems that benefit from rapid development and competition due to market forces out of our control, then it would be unwise not to exploit these capabilities, particularly during periods of decreased funding such as we are now experiencing. We believe we have found such a system without sacrificing any significant capability.

We would like to evolve into a system where the operations staff and other users or facilitators of the NSCL can choose or create their preferred control interface applications and management processes essentially independent of programming concerns. As commercial industrial control programs become more available, affordable, and functional, we will incorporate them into this system, provided funds and manpower are made available. We will strive to create the controls infrastructure that will make this vision possible and affordable in terms of both money and manpower.

VI. REFERENCES

[1] R. Au, R. Fox, B. Jeltema, and B. Johnson, "Status of Control System Software Development," *NSCL Annual Report*, 1982-1983, pp 90-92.

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

[3] L. Foth and A. VanderMolen, "Phase II Control System Console Level Control," *NSCL Annual Report*, 1983-1984, pp 251-256.

[4] R. Au, D. Blue, L. Foth, R. Fox, H. Hanawa, J. H. Jenkins, E. Kashy, G. Siedelberg, and A. VanderMolen, "Phase 2 Control System I: Overview," *NSCL Annual Report*, 1983-1984, pp 249-250.

[5] J. H. Jenkins, " Phase II Control System: Low Level Network," *NSCL Annual Report*, 1983-1984, pp 257-260.

[6] L. Foth, " Phase II Control System," *NSCL Annual Report*, 1988, pp 127-131.

[7] G. Humenik and D. Scott, "Programmable Logic Controller Use in the Phase II Control System," *NSCL Annual Report*, 1988, pp 147-148.

[8] A. McGilvra, P. Koblas, G. Zheng, J. Vincent, and W. Nurnberger, "NSCL Phase II Beamline Electronics, " *NSCL Annual Report*, 1988, pp 158-159.

[9] J. Priller, " Main Console and Software, " *NSCL Annual Report*, 1988, pp 132-134.

[10] G. Humenik, "Phase II Discrete Control System, " *NSCL Annual Report*, 1989, pp 131-132.

[11] L. Foth and J. Priller, "Current Status and Plans for the NSCL Control System Software," *NSCL Annual Report*, 1989, pp 133.

[12] J. Vincent, "A Control System Based on Microsoft Windows," *NSCL Annual Report*, 1991, pp 194-195.

[13] J. Vincent, L. Foth, and J. Priller, " Integrating the K500 Control System into the NSCL Control System, " *NSCL Annual Report*, 1991, pp 196-201.

[14] N. Weinberg, "NT, OpenVMS to blend," *Computerworld*, vol. 29. no. 19, 1995, pp 1, 127.

[15] A. Patrizio, "DEC plans OpenVMS transition to NT," *PC Week*, vol. 12 no. 17, 1995, pp 8.