NSLS CONTROL SYSTEM INTERFACE TO PARKER 6K MOTION CONTROLLER SYSTEM

S. Ramamoorthy[1], J. Dabrowski[1], C. Nielson[1], W. Rambo[1], E. Zitvogel[1], J. Kulesza[2]
[1]*BNL, New York, USA,* [2]*ADC Inc, New York, USA*

ABSTRACT

A 4-axes mini-gap in-vacuum undulator scheduled to be installed for one of the beam lines at the National Synchrotron Light Source facility (NSLS) will be controlled by a Parker 6K motion controller system. The controller is equipped with an Ethernet port for communication with other host systems. The undulator will be delivered with a 6K-based stand-alone control system by Advanced Design Consulting Inc., N.Y., USA. The Parker 6K system will be interfaced to the NSLS control system via Ethernet. This paper describes the integrated system and the Ethernet driver interface between the Parker 6K controller and the NSLS front-end system.

INTRODUCTION

The NSLS facility at Brookhaven National Laboratory was the first dedicated second-generation light source and it has been in operation for more than 20 years. The X-ray ring of the NSLS, presently operating at 2.8 GeV, provides photons to over 50 user beam-lines. Most of them are built off dipole magnets while a smaller fraction operates from various insertion devices (IDs) installed in the straight sections of the ring. The IDs (undulators and wigglers) are generally superior to bending magnets as they can be tailored to cover a higher photon energy range, to achieve higher brightness and flux, or deliver radiation with special properties, for example, variable polarization. A wiggler in the straight section for the beam line X25 has served as a source of a very successful mixed-use high brightness beam-line program for over 15 years. A recent decision to dedicate this beam-line exclusively for the needs of macromolecular crystallography has triggered the upgrade of the existing wiggler to an in-vacuum mini-gap undulator (MGU). The proposed MGU is expected to provide higher brightness in the 11-14 keV photon energy range conventionally used for the protein crystallography research as well as to provide nearly continuous energy tunabilty from 2 to 20 keV. Additional improvements in beam-line design and optics will also result in a factor of 3 higher energy resolution. The X25 MGU is being built by Advanced Design Consulting (ADC), New York, USA and will be installed by the end of 2005. It will be delivered with a stand-alone control system consisting of a Parker 6K controller and a touch panel display. Principle control parameters include the undulator gap (to vary the emitted photon energy), as well as the elevation of the entire device (to center it with respect to the electron beam orbit). This unit will be interfaced to the NSLS control system via Ethernet. This paper describes the architecture of the integrated control system and the Ethernet driver module.

PARKER 6K-BASED CONTROL SYSTEM BY ADC

Hardware

The control system is based on Parker 6K controller. This is a dedicated, motion controller capable of controlling up to 8 axes of servo or stepper motors. All 6K series controllers feature extensive I/O facilities and include fast trigger inputs as well as analog/digital outputs and inputs. Each axis is provided with shutdown and enable outputs, and inputs for home and end-of-travel limits, drive fault indication, an incremental encoder etc. For applications requiring additional IOs, the controller can be serially linked to as many as eight EMV32 I/O expansion modules (Brick IOs). The controller is equipped with two RS232 ports. The second can be configured as RS485 if necessary. A built-in 10Base-T Ethernet interface facilitates easy integration with local area networks.

The stand-alone control system [1] that has been designed for the MGU control uses 4 stepper motors. Each motor has a rotary encoder and TR linear absolute encoders for feedback. The linear encoders will use an SSI to DIDO interface module and the TR Electronics PU-30. This module can

multiplex 2 encoders to one set of digital inputs to the 6K using a TTL select input. The linear encoders can be programmed for 0.1micron increment. These have self-contained electronics with SSI or RS485 interface. In addition, a Keyence LS5000 laser scanner with a resolution of 50nm and accuracy of 2 micron will be used to measure the gap directly using ports in the vacuum chamber. The 4 motors are configured as 2 master/slave pairs, left/right upper and left/ right lower.

A large number of IOs are needed to support 4 linear encoders, Keyence data transfer and special limit requirements. ADC has designed a custom surface mount card to interface between the ID limits and kill switches and the 6K and the NSLS system.

For local controls and monitoring of the MGU parameters, a CTC (Control Technology Corporation) full color touch-screen, connected to the second RS232 port of the controller, is used.

Software

All 6K series products are supplied with a Motion Planner code development package that runs on a PC and is specifically designed to develop complex motion control programs in 6K language which is a collection of highly specialized function calls. The language resembles BASIC but yet supports a broad range of special function commands related to motion control. Software is developed in a text editor and downloaded to the unit for execution. The Parker 6K allows up to 10 tasks to run concurrently. These can be prioritized for best response. 6K programs can declare integer, real and binary variables (up to 255 each). An external system, depending on the type of communication method, can access all or a subset of these variables for reading and writing.

X25MGU_6K software [1] uses 2 tasks: motion task and update task. Following a power cycle/reset, the startup program will set the system parameters and assign default values to the relevant MGU related parameters and then start the two tasks. Update task will run 5 instructions before the execution is turned over to the motion task, which will run 1 instruction, thus giving the update task higher priority. The motion task will check for a move command from an external host and execute it provided the values are within the range and there are no fault conditions that would inhibit the move. The primary move commands are (1) to set the gap between the upper and lower magnet arrays to the commanded value and (2) to change the gap placement relative to the electron beam. The update task will continuously read the encoder positions, monitor the following Error and update the variables. It will also measure the gap using the Keyence on demand.

The data transactions across a host system and the 6K controller and proper hand-shake procedures take place via a set of variables assigned by the 6K software in its memory. For example, the sequence for producing a motion is the following. The host first sets a value in the proper variable and then the command bit corresponding to the desired move. The 6k will set an in-process bit, start the move and reset the bit on completion. If an error condition is detected prior to or during the move, it will branch to an Error routine and set an error bit. The program branches back to the MAIN loop (without a Reset) only after the host system clears the error bit.

INTEGRATION OF 6K CONTROLLER TO THE NSLS CONTROL SYSTEM

A distributed control system with a two level architecture is used for controlling the NSLS machine hardware. The operations rely heavily on a set of high-level programs that run on workstations constituting the upper level of the control system. They communicate with the lower or equipment level (consisting of mostly VME-based microprocessor systems and a few PC-based systems, referred to as Micros) via Ethernet. The programs access the various analog and/or digital hardware parameters and software variables by easily identifiable names. The real-time software (NSLS Control Monitor) [2], which is the kernel for all micros, hides all the complexity and diversity of the hardware controlled by each micro and presents a standard interface to the high-level programs. A standard set of commands (READ/SET) has been defined for access by high-level programs. The application modules in the micro interpret the commands and operate on the hardware.

In order to control the new MGU with the standard NSLS programs and also to exercise the operator-supervised gap control by beam-line users, the stand-alone Parker 6K system has to be interfaced to an NSLS front-end micro. Once integrated, all the software utilities (control and monitoring programs, Alarm handler, History (archiver), gap displays on local CATV etc.) will be available for the MGU operations.

Hardware Interface

The VME system uses a Motorola MVME 1603 Power-PC based single board computer in addition to other necessary IO boards for running the NSLS real-time control monitor. The Parker 6K Ethernet interface is connected to the same control network as the VME system. The RS-232 port 1 on the 6 K controller is connected to a serial IO port in the VME system to pass some special commands to Parker 6K. Figure 1 is a block diagram of the integrated system. Figure 2 is an example of the touch panel display generated by ADC Inc.

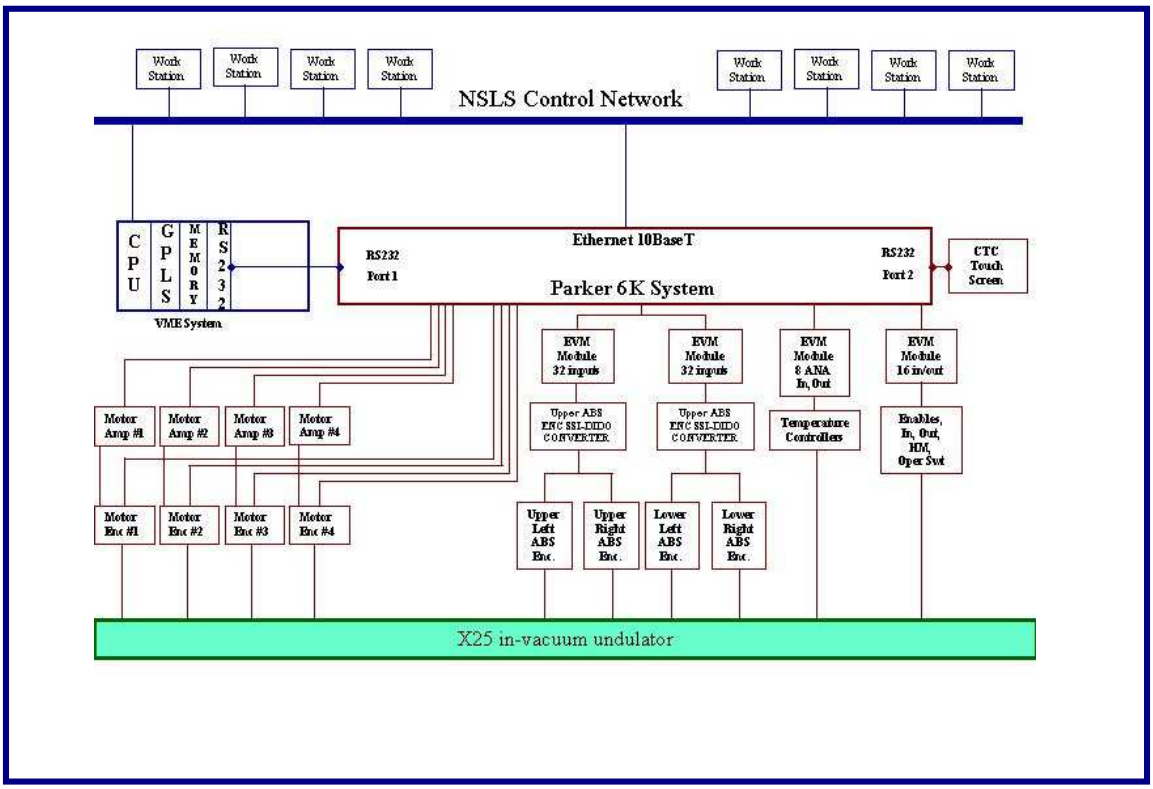

Figure 1: Integrated ADC Parker System with the NSLS Control System

Figure 2: Touch Panel Display generated by ADC

Software Interface

The MGU application module of the control monitor uses the utility functions, provided by the Ethernet driver developed at NSLS, to send the control commands to and retrieve various parameters associated with the MGU, from the 6K and updates the corresponding device records. If fault conditions on the motor, cable etc. are detected, error messages will be sent to the Error processor to alert the operators. The software also monitors a heart beat counter to determine whether the 6K program is running or not. If the 6K program quits due to some unexpected reasons, the NSLS software will issue a "RESET" command to the 6K via the serial port.

ETHERNET DRIVER

The driver module that is loaded into the VME system is comprised of client tasks and utility functions. Since the data exchange across the VME and 6K systems take place via the variables in the 6K memory, the VME software carefully handles the variables to achieve the desired results. The utility functions in the driver module hide all these details and provide an easy interface to the VME applications for sending the control commands and retrieving encoder data and other status information. Future modification in the 6K software/variables will require changes only in the driver and not in the application code. To understand the factors that contributed to the choice of the message format for communication, a brief description of the 6K Server is provided in this section.

Parker 6K Server

The resident Ethernet server of the 6K system is based on the standard TCP/IP and UDP protocol. Four ports (5001 to 5004) are available for communication. Except for the port 5003 which is a UDP port, the other three are TCP/IP ports and will be in the listen mode when the 6K is powered up/reset. The UDP port (5003) will be available when the port 5002 connects. Every port has a different purpose and format for message transfer to and from a client. Only one client is supported on each port at a time.

The port 5001 is used to set/read a subset of variables (12 integers, 12 real and 8 binary) by a client. A variable mask in the message packet indicates to the 6K which variables are modified by the client during this transaction. The message size sent to the port 5001 is always 192 bytes. The server ignores the data for unmodified variables in the message packet. If the client sets the bits for the 'status update and expanded data set' in the action mask field of the message packet, the server will return a reply with 384 bytes.

The port 5002 uses ASCII as its message format. A client can send any ASCII command listed in the 6K Command Reference manual and receive a response in ASCII. The message frame is the same as the one used by the RS232 port. When this port is connected, the RS232 port on the 6K will be automatically disabled. The second RS232/RS485 will still be active.

The UDP port 5003 can be configured to enable automatic status update at a specified interval. The data format is binary.

The port 5004 is used to configure a watchdog timer in the 6K Server. When enabled, it will check the integrity of the Ethernet connection.

Except for port 5002, the data bytes in the message packet are in the binary format. The description of the fields and the data formats necessary for packing and unpacking the messages are described in detail in the Parker document on Ethernet [3].

NSLS Client Module

The decision to use port 5001 was made after some preliminary tests were carried out on ports 5001 and 5002. Even though this port allows access to a subset of variables, the response packet provides information on the status of the limit switches, Brick IOs, motor axes, system errors, drive faults and stall condition etc. in one message packet. Since the 6K controller utilizes a 68340 processor, the multi-byte field (integer, short, real) are transmitted in Big Endian order which is the same as the Motorola VME systems. Hence the data transfer from the VME memory to Ethernet message packets

does not require re-ordering of the bytes while packing and unpacking the messages. Since the parameters that need to be controlled and read can easily be handled with the variables made visible in the 5001 port message packet, this port has been chosen for communication. Use of port 5002 requires binary to ASCII conversion for transmission to the 6K. Also the response has to be parsed first to extract the data string, which then has to be converted to the binary format. To acquire all the relevant data, a number of messages have to be sent. While it is desirable to use both ports 5001 and 5002 (5001 for data transfer and 5002 for special commands like RESET or to access variables outside the subset range) the tests indicate the use of port 5002 kills the connection to 5001 when a variable is set or even when connection to the port 5002 is gracefully closed. Unlike port 5002, use of 5001 does not disable the RS232 port 1. The VME system takes advantage of this feature and uses the serial port for sending special commands like "RESET" via RS232 port. The client task opens a TCP/IP connection to the 6K Server. If the connection is successful, the client will send a READ command (i.e. an expanded status update request) to the server at 2 Hz interval. Set messages (Move gap, Home etc.) always take precedence over Read messages. The client uses the "select call" with 5 seconds time-out before reading the socket. If the 6K is turned off or if the physical link is lost, the task closes the connection and retries again.

In addition, a second TCP/IP client task is started using the port 5004 to configure the watchdog timer of the server. The client sends a configuration message specifying the time-out interval in seconds. This is essential since the 6K Server does not close the socket automatically when a client connection breaks abruptly (for example, when the VME system loses power or reset). To establish the connection to the 6K Server again, the 6K has to be reset or cycled through the power. When the watchdog timer is configured, the 6K will shut down all the sockets that are alive after the specified timer interval.

Utility Functions

Init Function

The application should first invoke this function with the following parameters: A pointer for client ID and the IP address of the 6K system. The function first checks the validity of the parameters. A successful **init** call will spawn a client task and a watchdog timer task with the same priority as the application module. A NULL will be returned with a client ID number. An error code will be returned if there is a problem.

Read Function

The application module uses the **Read** function to retrieve the encoder data and other relevant status information associated with the MGU control. The application module calls this function with the client ID and a pointer to the MGU data structure. If there is no response from the server or if the heart beat counter does not change in the reply message, an error code will be returned to the application module. Otherwise, the Read function extracts the encoder data, status of limit switches etc. from the appropriate variables in the reply message and passes them back to the application module in the MGU data variable. The function returns either a NULL or an error code if the data in the reply is invalid. The application module should always check the return code before updating the data fields within the device record. There is no need to call this function faster than 2 Hz.

Write Function

When a high-level program sends a command to perform a specific move, the application module calls the write function with the parameters specifying the value and an action code (the action code indicates whether the requested move is a gap or elevation or home or any other move). If invalid parameters are detected, the function will return an error code to the application module. The function will set the appropriate variables and command bits for the required action in the message packet and post the information to the client task. A returned code to the calling routine is either a NULL if the command has been accepted, or an error code. The NULL code is not an indication that the move has

been completed successfully. The READ function will monitor the 'motion in-process bit' set by 6K software for the completion of the move and then clear the in-process field in the structure.

CONCLUSIONS

The Ethernet adapter on the Parker 6K controller facilitates a seamless integration between the 6K unit and the NSLS control system. The NSLS client driver module and the utility functions will provide an easy software interface to the 6K system for the VME application modules.

ACKNOWLEDGEMENT

Financial support comes principally from the Offices of Biological and Environmental Research and of Basic Energy Sciences of the US Department of Energy. The authors wish to acknowledge the support provided by U.S. DOE Contract No: DE-AC02-98CH10886. The authors express their thanks to L. Berman for providing information on the characteristics of the proposed X25 MGU from research point of view, R. Michta for his help in porting the client program to a PC platform for test purposes and B. Podobedov for helpful suggestions Thanks are also due to R. Biscardi and J. Skaritka for their support.

REFERENCES

[1] J. Kulesza, "UHVAC specification Manual for BNL UHVAC-X25 undulator", ADC Inc, September 2005
[2] S. Ramamoorthy etal., "NSLS Control Monitor Upgrade", Proc IEEE, PAC (1993), 1849.
[3] Parker Automation 6K Ethernet Driver Specification Revision 1.3 Manual.