Submitted at the IEEE 1993 Particle Accelerator Conference
May 17-20, 1993, Washington, DC

BNL-49141

NSLS Control Monitor and its Upgrade*

Susila Ramamoorthy and J.D.Smith
National Synchrotron Light Source, Brookhaven National Laboratory
Upton, New York 11973

Abstract

The NSLS Control Monitor is a real-time operating system designed for the microprocessor subsystems that control the machine hardware in the NSLS facility. Its major functions are to control the hardware in response to the commands from the host computers, monitor hardware status and report errors to the alarm handler. The software originally developed for the Multibus micros has been upgraded to run on the VME-based systems. The upgraded monitor provides ethernet communication with the new system and serial link with the old system. The dual link is the key feature for a smooth and nondisruptive transition at all levels of the control system. This paper describes the functions of the various modules of the monitor and future plans.

1. Introduction

The microprocessor subsystems controlling the machine hardware in the National Synchrotron Light Source facility are driven by a realtime operating system referred to as NSLS Control Monitor. As in any control system, the monitor performs hardware control, data acquisition and closed loop algorithms in real-time. It provides a standard interface to the control commands and data requests from the high level application programs on the host computers. The functional blocks of the upgraded monitor software, the current status and future plans are reported.

2. Upgrade Goals

The control system designed in 1978 was upgraded to meet the increasing demands on data acquisition rates and CPU power and to provide better diagnostics. The original system used Multibus-1 with 8 bit Intel 8080/85 CPU single board computers. A realtime multitasking monitor developed in-house is used for low level equipment control. The communication with host computers uses a serial link at 19.2 kbaud. The serial link, 16 bit address space, assembly language programming and absence of floating point coprocessors were the major limitations. The architecture, hardware and software components have been changed. The micros and the host workstations have been connected by ethernet in a distributed network.[Ref. 1,2]. As part of the upgrade, the Multibus micros have been replaced by VME based 32-bit computers with an ethernet controller. The software has been upgraded to be compatible with the hardware changes. A goal of the upgrade is to make the transition smooth with no impact on machine operations. This was realized by modifying the communication software which provides simultaneous access to both the new and old systems.

3. Hardware Configuration

The minimum hardware required to run the control monitor on a microprocessor system are a VME-based 68020 CPU with ethernet controller, 1 megabyte battery backed-up ram and the General Purpose Light Source (GPLS) board which has timers, serial ports, bus-interrupter module, video display generator, diagnostic LEDs and software selectable switches. Further hardware requirements are dependent on the equipment to be controlled. The hardware I/O interfaces include analog and digital cards in the VME crate, bus extenders, GPIB and RS-232/422 and Camac interfaces.

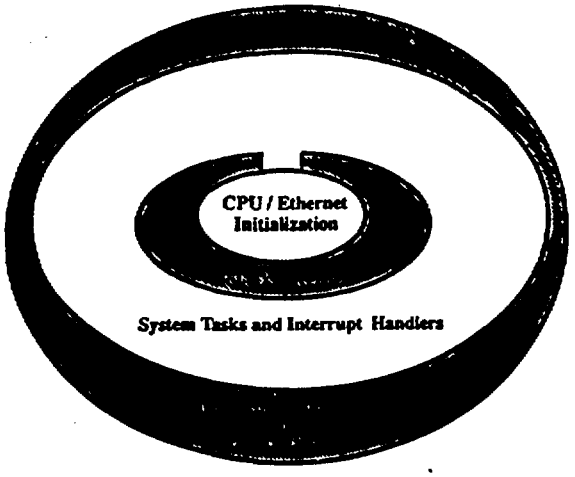

Fig. 1. Monitor Software Layers

4. Software Overview

The software consists of a set of system and application tasks and interrupt handlers. It provides an easy environment for developing application-specific modules in the micro. The Control Monitor is organized in logical layers as shown in Fig 1. The real-time kernel initializes the CPU and the Ethernet hardware thereby making the monitor software CPU/Ethernet hardware independent. The monitor system software initializes a few peripheral chips on the CPU for its own purpose. The system tasks are responsible for the management of the system hardware (GPLS board

*Work performed under the auspices of the U.S.Department of Energy.

peripherals). This isolates the system hardware and the real-time kernel from the micro application tasks. The application tasks use device drivers and system services to control and monitor the hardware. The interface between the system and the application software has been defined in such a way that any modification and upgrade at the system level will not require rewriting of the micro application tasks.

In addition to the software modules, the monitor has a device database in memory which is accessed by both the system and micro application tasks. The software views the various hardware signals (analog and digital) of the equipment as a set of logical devices. The database is a collection of data structures representing the logical devices. A standard device Format is used. This contains a number of fields for various information about a device (readback, setpoint, tolerance, limits for setpoint and alarms, digital command and states, and error status). The format also supports calibration and four data arrays. The devices can be a simple analog/digital input/output type or a composite type consisting of more than one simple type. A physical device with various types of signals can be represented by a group of simple logical devices or by a composite logical device. The device need not necessarily represent hardware. There can be soft devices that may be used by control algorithms. The device format has a configuration mask to indicate applicable fields and commands for a device. The database also includes all the pertinent hardware information such as its type, address etc. All the devices are treated in a similar fashion as far as updating or retrieving commands from the database. The host computers refer to a logical device by a descriptive name. The Device Data Record library (DDR) at the host computer translates the name to a physical address. (For serial link this is a serial line ID and device number. For ethernet link it is node name and device number). Thus the high level software controls and acquires data from the machine hardware using names and a set of standard commands without any knowledge of its location or the internals of the hardware. The micro application software and the device driver modules take care of all the details.

5. Software Components.

Fig 2 illustrates the software components, their interaction and the data flow.

5.1 Real-time Kernel

The processor runs RTUX, a real-time operating system developed by Emerge Systems Inc. in Florida. This is a fast, memory resident real-time kernel with the interrupt latency time in range (10-12 μsec) for 20 MHz CPU. The task switching time is approximately 40 to 50 μsec. The kernel supports multitask control, memory management, interrupt handling utilities, event handling, message queuing, intertask communications. The RTUX-ethernet clerk (Network package) provides the standard socket level abstraction. Software development is carried out in a standard Motorola UNIX development system using the C language. The program can be downloaded into the target system and debugged using the RTUX tool ANALYZ. The rombuild utility generates an executable image that can be burned into proms.

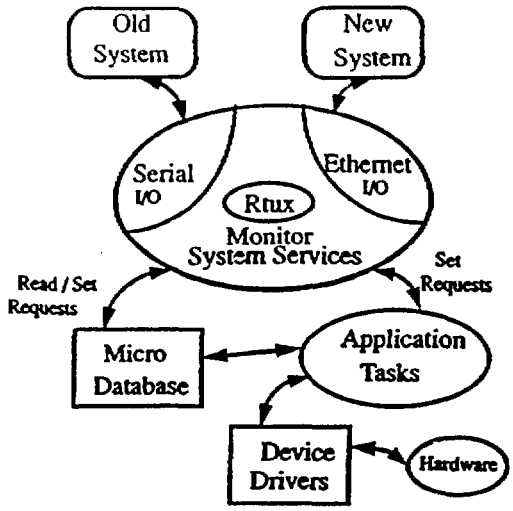

Fig. 2, Software Components.

5.2 Monitor System software.

This module is standard for all the micros and consists of multiple system tasks and interrupt handlers. It provides system timing (1 millisec resolution) and uses RTUX primitives to synchronize and coordinate the activities of the various tasks. The important features of the system module are described below:

A. Initialization module.

The monitor initializes the necessary timing, display and serial port hardware, sets up interrupt handlers for timing, video, console and serial IO functions. After initialization, the application tasks are spawned.

B. Communication management

This is the most important activity of the system tasks. It receives operator messages in a standard format for all micros and updates the micro database fields. Since the high level applications can use serial or ethernet link during the conversion period it is imperative that the micros should accept requests via any link.

The serial server uses interrupt handlers for message I/O and provides handshake on every message. The message can have up to 64 bytes and contains only one request or reply per device.

For ethernet, the message headers and device packet formats have been carefully designed for future enhancements. UDP protocol with message size 1024 bytes per packet is used. Multiple read requests or commands or replies are packed in one message, resulting in low network traffic. The communication model supports both server and client roles. The server receives commands from host computers or any micro in the network. The client software pro-

vides micro to micro communication facility. Both the application and system tasks use the client services to acquire data from or to control any device on a different node. Handshake is provided for all messages. Integrity of the messages is checked and duplicate and out of sequence messages are identified and appropriate actions taken. Valid messages are disassembled into individual packets and passed to the command decoder. The decoder returns a reply which may be the requested data or acknowledgement. All the replies are assembled into a single message in the same order as received and returned to Ethernet IO module.

C. Command Decoder

The device commands are divided into two types, READ and SET. The READ requests are handled by the monitor system task. It builds the requested reply packets using the current values from the device database which is updated at rates of more than 10 Hz. There are more than 20 set commands for setpoints, limits, arrays and digital state control. Most of the set commands are handled by the system. The monitor uses the configuration mask before updating the Database fields. A setpoint command for an ON/OFF type is automatically discarded. All the necessary checks such as limits check, lock/unlock status check etc. are carried out before updating the command field. The application tasks are notified only for a few commands (setpoint, digital state control, device reset and new array) for further low level control.

D. Error And Alarm Reporting

The monitor provides services for alarm check, tolerance check and error reporting to the micro application tasks. The device error messages are sent asynchronously to an Error processor system on the network using micro to micro communication. Error flags are latched in the device records to prevent error storms. A new Set or Reset or device Error Clear command will unlatch the error. If the device is no longer in error, an ERROR RESET message is sent to the Error Processor. During the transition, a switch was provided to select the error reporting either through the serial or ethernet link. The system has the option to inhibit error reporting on an individual device basis or on a micro basis by selecting the error disable switch.

E. Diagnostic statistics

The monitor provides a display for the system statistics such as number of devices in error, number of spurious bus interrupts, transmission errors, message traffic and the time the system has been up etc.

F. Display management

Cable TV compatible ASCII display is a new feature added to the upgraded micros. There are 4 switch selectable hardware display pages. One hardware page can multiplex up to 8 software selectable pages . The display page can be controlled either by a push-button panel connected to the micro or by a remote command. Both system and application tasks generate displays for diagnostics and for continuous monitoring of device parameters and status without imposing any load on the network.

5.3 Micro Application Tasks

This software is responsible for all the control and monitoring of the low level hardware. It sets up the application-specific device database and initializes it either with default parameters or from the battery backed up ram. The device control may be as simple as setting a bit in a bit/io card or setting a DAC or may involve control of a group of devices synchronously or sequentially, or may require a complicated software algorithm. The application tasks use the appropriate device drivers to accomplish the commands. The tasks also monitor the hardware signals at 10 Hz or more and update the database and generate application-specific video displays. The application tasks can use periodic timer interrupts provided by the system clock to trigger data acquision. They use the system services for getting interrupts from an external source or from a slave on the VME bus.

6. Conclusions

The initial goals of the micro upgrade have been completed. All the 70 Multibus micros have been converted to 50 VME-based systems. New features such as video displays and optional recovery from battery backed up ram on start-up have been extremely useful from the operations and diagnostic points of view. The dual communication allowed us to carry out the workstation and the micro upgrade activities with practically no impact on the machine operations. During this period, the prime objective has been conversion to VME micros with ethernet link. Future plans include more device types which will support multivalued setpoint or readback devices and automatic device configuration module from a file image representing the device parameters and hardware information. A Device Name server to furnish various information such as the micro location, record number, calibration constants etc. is being planned. The monitor will provide a real time Name server to micro application tasks to access device information. An optional watchdog timer and a facility to save crucial parameters when AC failures occur, have been tested for future additions. Plans to boot the software from the rom image on a battery backed up ram is being tested.

7. Acknowledgement

The authors express their gratitude to S.Krinsky and J.Keane for their support and encouragement during this period. We thank S.Kramer and N.Fewell for helpful suggestions and W.Rambo and his group for the technical support.

8. References

[1] J.Smith *et al.*,"NSLS Control System Upgrade Status,"*These Proceedings, 1993.*

[2] Y.N.Tang *et al.*, "The High Level Programmer and User Interface,"*These Proceedings, 1993.*

DISCLAIMER