CONF-870552--33

NEUTRAL PARTICLE BEAM DISTRIBUTED DATA ACQUISITION SYSTEM

R. T. Daly, M. R. Kraimer, and A. H. Novick*
Argonne National Laboratory
9700 S. Cass Avenue
Argonne, IL 60439
(312) 972-6966

CONF-870552--33

DE88 009943

Abstract

A distributed data acquisition system has been designed to support experiments at the Argonne Neutral Particle Beam Accelerator. The system uses a host VAXstation II/GPX computer acting as an experimenter's station linked via Ethernet with multiple MicroVAX IIs and rtVAXs dedicated to acquiring data and controlling hardware at remote sites. This paper describes the hardware design of the system, the applications support software on the host and target computers, and the real-time performance.

Introduction

The Neutral Particle Beam Test Stand (NPBTS) of Argonne National Laboratory (ANL) utilizes the 50 Mev H^- injector from the former ANL Zero Gradient Synchrotron (ZGS). The injector operates at a maximum repetition rate of 30 Hertz and a pulse width of 60 microsec.[1] The NPBTS facilities, located in the ring building previously used by the ZGS, consist of three beam lines designated as Phase 0, A, and B and the test areas or stands at the end of these beam lines. Users from government laboratories and industry will conduct experiments at the NPBTS primarily involving physics measurements, engineering measurements of components and subsystems related to a space based Neutral Particle Beam, and beam diagnostics measurements. Due to the diverse nature of the planned experiments and the differing backgrounds of the experimenters, the data acquisition system must support a wide variety of instrumentation located at the test stands. Although only one experiment may be on-line at one time, multiple experiments may be in a check-out stage; therefore the data acquisition system must support the simultaneous acquisition of data from multiple experiments.

Hardware System

The data acquisition system, NPBDAS, as shown in Figure 1 provides at each test stand and in the NPB control room a target data acquisition system (TDAS) consisting of a single board MicroVAX II computer with interfaces to a number of standardized instrumentation systems including CAMAC, IEEE 488, RS232, and VME and to the Ethernet used for host computer communications. Each TDAS is responsible for acquiring data from and controlling the instrumentation at its test stand, for buffering and preprocessing the data prior to transmission to the host, and for the transmission of the data over Ethernet to the host computer. A 500 meter coax cable provides the only data link between

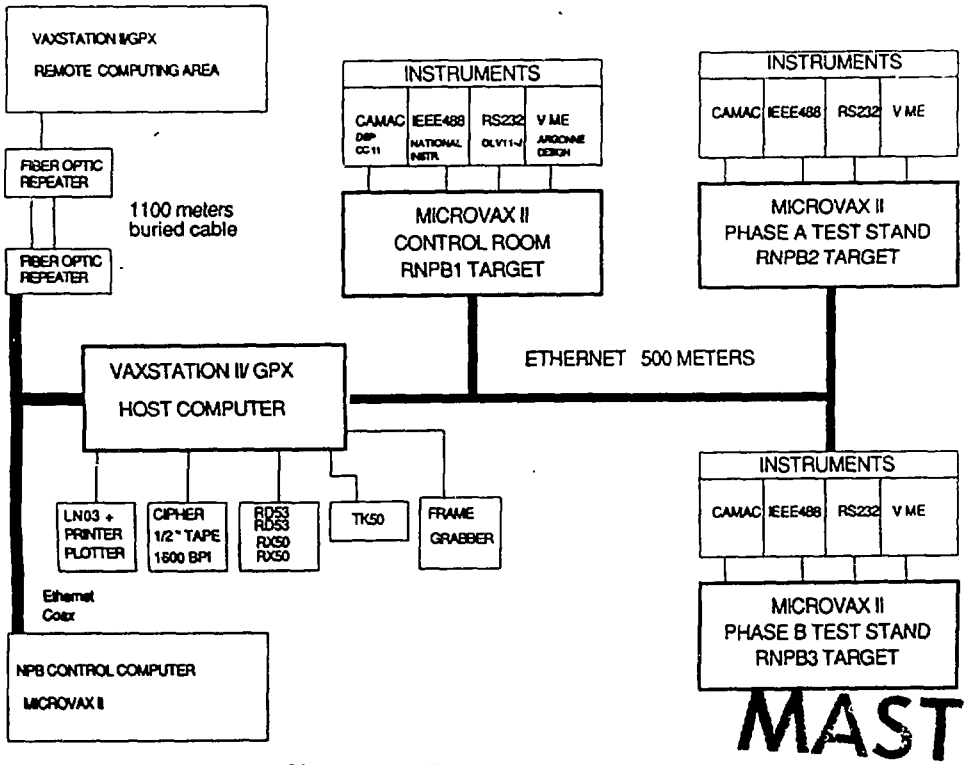


Figure 1. NPBDAS Hardware System.

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

DISTRIBUTION

MASTER

the TDAS systems and the host. The host computer system is located in the NPB control room and includes a VAXstation II/GPX color workstation, a variety of magnetic storage peripherals, a video imaging subsystem, an Ethernet interface to the TDAS systems, and a second Ethernet interface to a LAN which includes the NPB MicroVAX II Control Computer and the MicroVAX II Remote Experiment Control Computer.

Support Software

The software system developed to support the NPBDAS has the hierarchical structure depicted in Figure 2. The lowest level or Instrument Support Level is completely located in each TDAS. It is a VAXELN system which provides all the general purpose real-time functions needed to support the attached instrumentation and to communicate over a Decnet/Ethernet link with the host computer running the VMS operating system. The System Support Level on the host provides a buffer management scheme and a defined interface to the Instrument Support Level from the User levels which, in general, makes the distributed nature of the NPBDAS transparent to the User levels.

The highest level or User Support Level provides a command level interface to utilities which display data, manipulate disk data files, and analyze data stored on disk files.[2] In addition, an Experiment Definition Utility is used to define for each experiment the control functions to be executed by the TDAS, the parameters to be collected by the TDAS, the data conversions to be applied to the collected parameters at the host before storage or display, and the record structure for the data file used to store selected parameters. The experiment is initiated by a RUN command from the command level. A future paper will describe the User Support Level in more detail.

Users who wish to write their own data acquisition programs must use the System Support Level software to interact with the TDAS systems and are provided with the RS/1[3] commercial software product to analyze and display their experimental data.

Instrument Support Level

The Instrument Support Level software is run on each TDAS. The TDAS MicroVAX II computers run the VAXELN kernel and use the VAXELN Network Services to provide a real-time operating system environment for the VAXELN application programs, and to provide support for the application level logical link (program to program) communication over Ethernet to the host computer system.

Two application programs, INIT and VMS_LINK, provide all the run time support at each TDAS. A functional block diagram of the application programs is shown in Figure 3. When the TDAS is booted (either at power-up or by a host DCL command) the INIT job creates a global data area shared by all VAXELN jobs, initializes the instrumentation hardware, and dynamically creates ten jobs, each running the code contained in the single program image VMS_LINK. As each job is created it is passed a unique message identifying two logical links, a COMMAND LINK and a DATA LINK, which will be used to communicate with an application program on the host. Once established the COMMAND LINK provides a communications path over Ethernet for a host application program to pass COMMAND_LISTS to the VMS_LINK job on the target. The COMMAND_LISTS direct data acquisition and control functions at the TDAS. As each COMMAND_LIST is received, a VAXELN process is created to execute it.

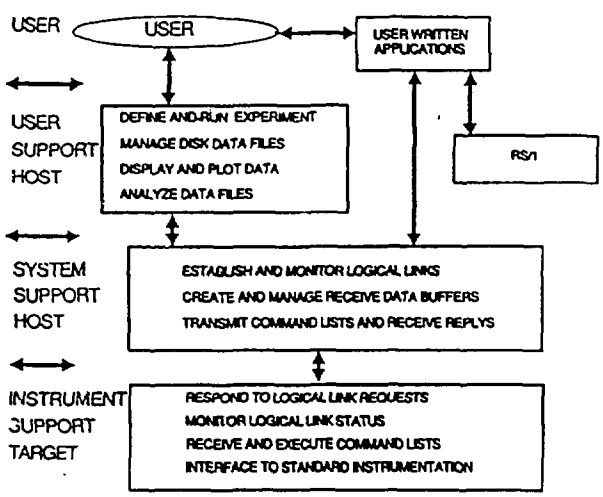


Figure 2. NPBDAS Sotware System.

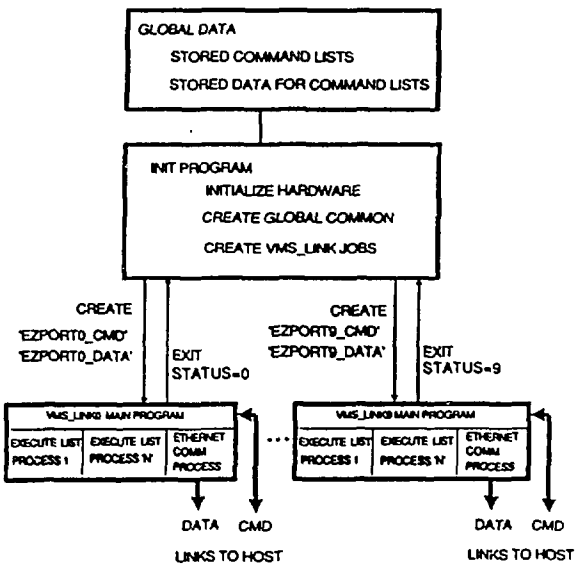


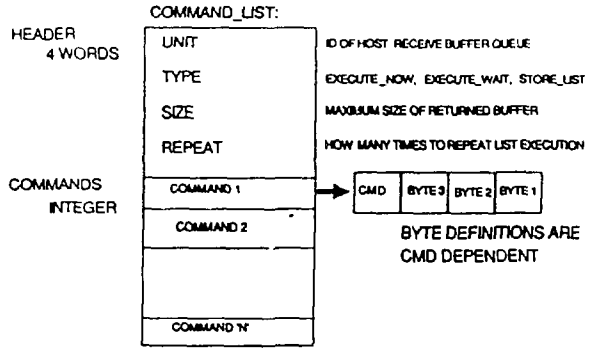
Figure 3. TDAS Application Programs.

Any data collected by the TDAS during the execution of the COMMAND_LIST will be returned to the host over the DATA LINK by a separate COMM process in the VMS LINK job. Thus, real-time data acquisition can proceed concurrently with data transmission to the host.

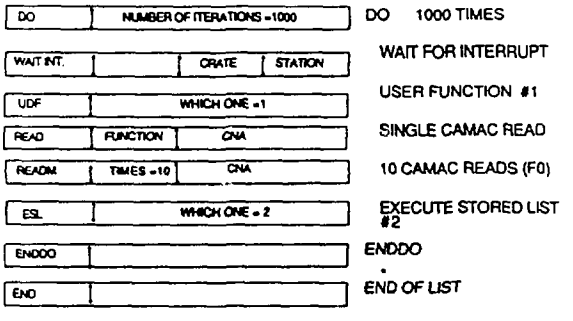
With ten VMS_LINK jobs running on each TDAS, up to ten application programs may be concurrently directing data acquisition at each TDAS via COMMAND_LISTS. In the NPBTS environment, this feature allows one experiment to be collecting data on-line at a TDAS while multiple experiments are collecting data in a check-out or background mode using instruments interfaced to the same TDAS. Also, it is typical for a single experiment to establish links to more than one target, i.e., a link to TDAS at Phase B beam line and a TDAS in the control room, in order to collect data from instruments interfaced to more than one TDAS.

The structure of a COMMAND_LIST is shown in Figure 4. The COMMAND_LIST consists of a four word header and a variable number of integer commands. The header words are:

1. UNIT. The identifier of a queue maintained by the System Support Level software which is created by an application program on the host prior to sending a COMMAND_LIST to a TDAS, and to which any buffers returned from the TDAS as a result of executing the commands in the COMMAND_LIST will be queued. The application program retrieves buffers from the UNIT queue using the System Support Level software.



COMMAND_LIST STRUCTURE



EXAMPLE COMMANDS IN COMMAND_LIST

Figure 4. COMMAND_LIST Structure.

2. TYPE. Specifies to the TDAS how to manage the commands in the COMMAND_LIST. For example, the TYPE specifies whether to store the commands in the global data area or to execute the commands; and, if the commands are to be executed whether to execute them before or after a reply is sent to the host acknowledging receipt of the COMMAND_LIST, i.e., synchronous or asynchronous execution. At the host control is returned to the program sending the COMMAND_LIST only after a reply is received.

3. SIZE. Specifies the maximum SIZE of the data buffer that the host can receive from the TDAS.
4. REPEAT. Directs the TDAS to execute the commands in the COMMAND_LIST a REPEAT number of times. A value of 0 means forever. Each time the end of the commands in the COMMAND_LIST is reached any data collected will be transmitted to the host. Thus, this word when used in COMMAND_LISTs which collect data and indicates how many buffers of data will be returned.

There are presently over twenty-five commands available within the VMS_LINK program. Since each command is implemented as a Pascal procedure, new commands are easily added. The commands fit into four categories:

1. Hardware specific - Examples are commands to control CAMAC modules in single transfer and block transfer modes and commands to read and write strings to an RS232 controlled device.

2. Structural - Examples are the Do loop command which executes a group of commands repeatedly, and the command Execute Stored List which executes command list previously stored in the global data region.

3. Control - Examples are the command to Wait For Interrupt which waits for an interrupt from a specified CAMAC module, and the command to Wait for a fixed period of time.

4. Special Purpose - Examples are the command to execute a User Defined Function to handle a unique device, and the command Time which inserts the VAXELN system time into the returned data buffer.

An example of the commands contained in a typical COMMAND_LIST is shown in Figure 4. Together with the comments to the right of the commands, the list is self-explanatory.

TDAS Performance

The maximum rate at which a COMMAND LIST, which contains only a single CAMAC write, can be sent from the host to a TDAS and a reply received at the host is 70 COMMAND LISTs per second (see Figure 5). Also listed are time measurements for reading from a CAMAC module using three available methods: consecutive single CAMAC read commands, a single command which includes a number indicating how many reads are to be made from a single CAMAC address, and optimized reads from CAMAC which can be programmed into a User Defined Function. With consecutive commands it takes 50 microsecs per read. With a single command it takes 15 microsecs per read. The maximum rate at which a CAMAC read can be executed from Pascal is 4.7 microsec. This type of CAMAC read could be used in the procedures invoked by the User Defined Function

command.

The interrupt response time, which is the time from a CAMAC interrupt until the first instruction in the interrupt routine is executed, is in 98% of the interrupts between 23 and 26 microsecs but can extend out to 75 microsecs. The time from when an interrupt occurs until a VAXELN process waiting for the interrupt can proceed, i.e., a Wait for Interrupt command is completed, is in 98% of interrupts between 265 and 275 microsecs but can extend out to 2000 microsecs.

The last box in Figure 5 lists the amount of time spent in the VAXELN Network Services procedure SEND for various sizes of buffers sent to the host. If this procedure were put in-line in the process executing the COMMAND LIST, the process would be blocked for the amounts of time shown. In the TDAS system, a separate COMM process is used for sending data to the host. This process runs concurrently with process executing the COMMAND LIST, thus reducing the blocking effect of the VAXELN SEND procedure.

Shown in Figure 6 is a observation of the activity on Ethernet for large messages sent from the TDAS to the host. Over a circuit or logical link, VAXELN Network Services automatically breaks up large messages into multiple segments which are sent over Ethernet one at a time. Each segment contains a maximum of 1460 bytes of user data and takes 1.4 millisec to transmit over the Ethernet. An intersegment gap of .3 millisec occurs between each segment. In addition, after every eight segments sent from a TDAS to the host, a 9 millisec acknowledge

EXECUTE_LIST <--->REPLY CYCLE		70 CYCLES/SEC
CONSECUTIVE CAMAC READ CMDS		50 USEC/READ
MULTIPLE CAMAC READS IN ONE CMD		15 USEC/READ
MINIMUM CAMAC READ TIME IN 'UDF'		4.7 USEC/READ
INTERRUPT RESPONSE TIME	98%	23 ->26 USEC
	2%	26->75 USEC
SIGNAL PROCESS FROM INTR.	98%	265->275 USEC
	2%	275->2000 USEC

TIME SPENT IN NETWORK SERVICES MILLISEC	BUFFER SIZE BYTES
4.5	2K
7.0	4K
12.5	8K
18.5	16K
23.5	64K

Figure 5. TDAS Performance.

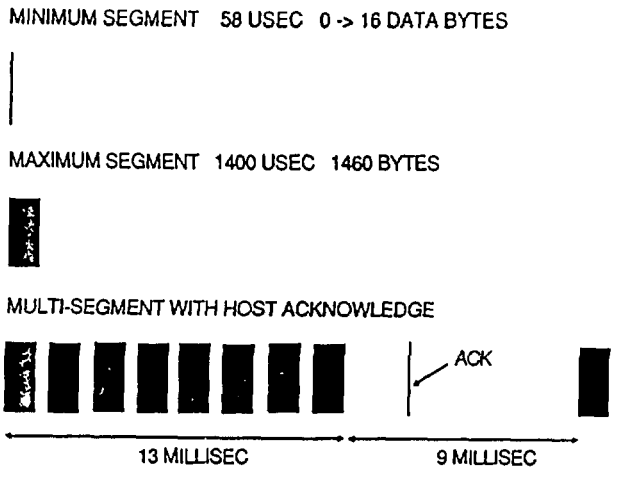


Figure 6. Ethernet Performance.

cycle occurs before VAXELN starts sending the next segment. Therefore, it takes about 22 millisec to transmit 8 segments of user data for an effective bit transmission rate of 4.3 megabits per second. Since Ethernet has a 10 megabit per second bandwidth, the maximum bandwidth utilization which can occur while sending large messages from the TDAS to the host is 43%. In Data Transfers over Ethernet at the data-link level bandwidth utilizations as high as 70% have been reported.[4]

In actual tests using the NPBDAS data acquisition software, we have been able to send 64 Kbyte messages every 165 millisec (every 5 beam spills) from a TDAS to a host for a rate of 384 Kbytes per second. The tests did not try to send data at the maximum rate possible.

System Support Level Software

A library of functions is provided by the System Support Level to interface the User levels to the Instrument Support Level for data acquisition and to implement a buffer management scheme at the host for data buffers returned from the TDASs. There are only seven functions supplied in the System Support Level software. An outline of an application program (see Figure 7) using all the functions, will be used to explain the System Support Level software. Each numbered line in the program outline is explained below.

1. Parameter definitions used in System Support function calls.

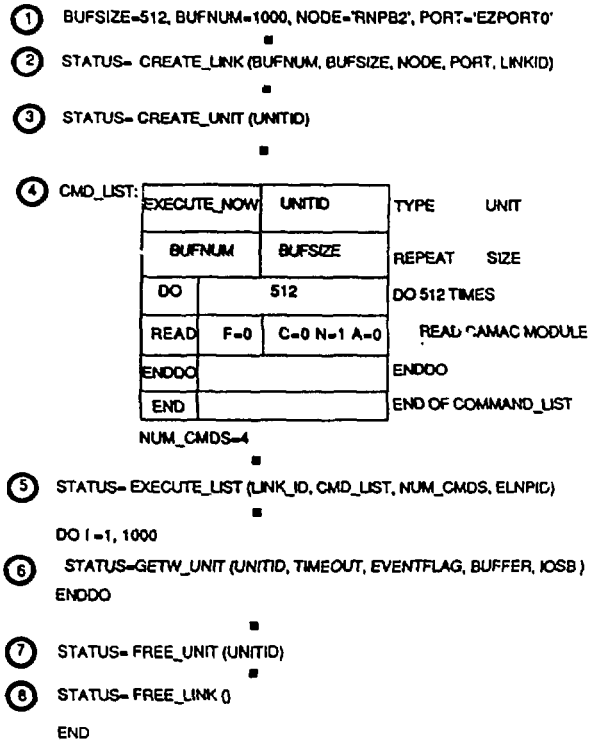


Figure 7. Program Outline Using System Support Level Software.

2. The CREATE_LINK function creates a buffer pool of BUFNUM buffers each with a size of BUFSIZE and locks the buffer pool into memory. These buffers are managed by the System Support software and are used to store data buffers received from a TDAS. In addition two logical links, a DATA LINK (EZPORTO_DATA) for receiving data buffers from a TDAS and a COMMAND LINK (EZPORTO_CMD) for transmitting COMMAND LISTS to and receiving replies from a TDAS, are established. The links are established with node RNPB2, which in the NPBDAS is the TDAS associated with Test Stand A. An identifier LINKID is returned to the caller and will be used in subsequent calls to refer to the two logical links just established.

3. The CREATE_UNIT function creates a System Support Level maintained queue for data buffers returned from a TDAS. The identifier of the queue is returned to the caller as UNITID. The UNITID identifier will be used later in the header of a COMMAND LIST. Any data buffers returned from a TDAS as a result of executing the COMMAND LIST will be queued to UNITID by the System Support Level.

4. A COMMAND LIST (pictorially shown in Figure 6) which reads 512 words from a CAMAC module at crate 0, station 1, subaddress 0 with a function code of 0.

5. The EXECUTE_LIST function transmits the COMMAND LIST over the EZPORTO_CMD link to node RNPB2 as identified by the LINKID parameter. Since the COMMAND LIST header TYPE is EXECUTE_NOW, the TDAS at RNPB2 will reply to the EXECUTE_LIST function call prior to executing the commands in the COMMAND LIST. Thus, control will return to the caller while the COMMAND LIST is being executed at the TDAS, and both systems will be operating in parallel. If the header TYPE had been EXECUTE_WAIT, control would only be returned to the caller after the TDAS had executed the COMMAND LIST. The ELNPID parameter is returned to the caller and identifies the process created at the TDAS to execute the list. Any data buffers returned from the TDAS at RNPB2 node as a result of executing the commands in this COMMAND LIST will be queued to the UNITID queue identified in the COMMAND LIST header word, UNIT. The header REPEAT parameter is equal to 1000 so the list will be executed 1000 times returning to the host 1000 buffers each containing 512 words. Had the header REPEAT parameter been 0, the commands in the COMMAND LIST would be repeatedly executed until explicitly stopped by another EXECUTE_LIST function call with a header TYPE of EXECUTE_KILL and a single command containing the ELNPID identifier.

6. The GETW_UNIT function retrieves a data buffer from the queue identified by UNITID. If a buffer is queued to UNITID or no buffer is received within a TIMEOUT period, control will return to the caller. The IOSB four word block contains the completion status and, if a buffer was received, the length of the buffer. A pointer to the buffer is returned in BUFFER. Another form of the call, GET_UNIT, is available. In the alternate form, control is immediately returned to the caller and an event flag, EVENTFLAG, is set whenever a TIMEOUT occurs or a buffer is queued to UNITID. The GETW_UNIT function is called 1000 times from within a do loop to retrieve all 1000 buffers sent over link EZPORTO_DATA from node RNPB2.

7. The FREE_UNIT function call returns to the System Support Level buffer pool, all buffers previously retrieved from the UNITID queue by calls to GETW_UNIT. In this example, 1000 buffers will be returned to the buffer pool.

8. The FREE_LINK function disconnects all logical links in an orderly manner.

Conclusion

System implementation from conception to installation at the NPBTS took approximately 10 months with a total software effort of 10 man months and a hardware effort of 2 man months. The use of high level languages throughout the software implementation, the use of an excellent VAXELN debugger to debug software running at the TDASs, and the use of VMS Decnet and VAXELN Network Services were key factors in bringing the system into operation quickly. Much of the User Support Level software was easily ported from an existing RSX11-M system since it was written in Fortran.

In the immediate future we plan to add significantly to the commands available in the COMMAND LIST as new experiments come on-line and to add histogramming functionality at the Instrument Support Level. At the User Support Level our efforts will be focused on making the system more user friendly.

References

[1] Neutral Particle Beam Test Stand, Users Handbook, Document No. G 0558-0004-SA-00, July, 1986.

[2] NPDAS--Neutral Particle Beam Data Acquisition System, R. T. Daly and M. R. Kraimer, Unpublished information, 1987.

[3] RS/1 is a trademark of Bolt Beranek and Newman, Inc.

[4] "Performance Characteristics of Ethernet," ACM Signmetrics, 1985. Timothy A. Gonsalves, Stanford University, Computing System Lab., Dept. of E.E.