DAPHNE...THE DESIGN OF A PARALLEL MULTIPROCESSOR DATA ACQUISITION SYSTEM

R. T. Daly, T. H. Moog, and L. C. Welch †
Argonne National Laboratory, Argonne, IL 60439

Abstract

The DAPHNE data acquisition system is designed to collect data at high rates from CAMAC instruments and to analyze the data on-line with multiple single-board computers using the National 32016 micoprocessor. It will be the standard data acquisition system used on all instruments at the Argonne Tandem-Linac Accelerator (ATLAS), a heavy ion accelerator. The system includes a programmable Event Handler to acquire the data from CAMAC instruments, Event Processors operating in parallel to analyze the data, an Event Distributor to supply the data to the parallel processors, and an intelligent Event Processor Interface to control the Event Processors and transfer data between the host computer and the Event Processors. Functional characteristics of each of these system components will be discussed and the design of the custom Event Distributor and intelligent Event Processor Interface will be described in detail.

Introduction

Event analysis software which is repeatedly executed for each event acquired from the CAMAC instruments in multi-parameter nuclear physics experiments, has typically been run on single computer systems with sophisticated multi-user and multi-tasking operating systems. Since each event is analyzed independently, it is possible to handle the analysis task with multiple computers operating on individual events in parallel with minimal interprocessor communication requirements; and since the analysis software is relatively simple, it can easily be run on most currently available 16- and 32-bit microcomputers without the need of an operating system. The commercial availability of single-board microcomputers with costs in the $1K to $2K price range and of IEEE standard bus systems capable of supporting multiple processors have made the design of a parallel multiprocessor system practical in nuclear and atomic physics experiments.

To use a multiprocessor system effectively, careful attention must be paid to two areas of data communication that are potential limiting factors to system performance. These areas are:

1. The distribution of data to the processors from the data source should insure that no data transfer bus becomes saturated and no processor becomes data starved.

2. The interface between the multiprocessor system and a host computer system should ensure that effective control can be applied to the multiprocessor system by the host and that data can be transferred to the host from the multiprocessors at high rates.

Each of these issues has been considered in the design of the DAPHNE data acquisition system.

System Overview

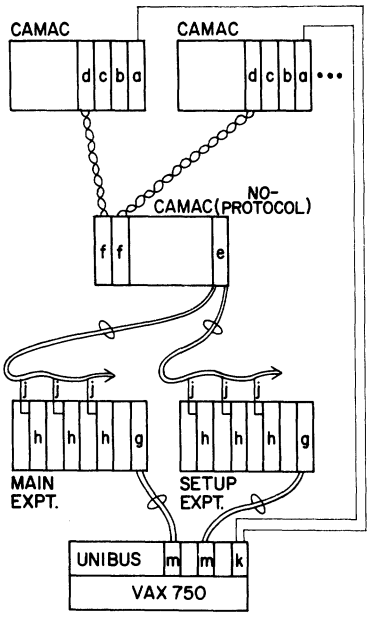
This discussion will cover the processing and movement of event data from the time the event data originates in CAMAC modules at remote experimental sites until it is eventually transferred into memory buffers on a VAX-11/750 (see Fig. 1). The applications software running on the VAX is covered in another paper at this conference.

CAMAC
EXPERIMENTAL STATION
a) RS232 CRATE CONTROLLER (KS 3989)
b) EVENT HANDLER (DAVE HENSLEY - ORNL)
c) FIFO (KS 3841)
d) MUX DRIVER (BOB DALY - ANL)

MUX
e) MUX CONTROLLER (BOB DALY - ANL)
f) MUX RECEIVER (BOB DALY - ANL)

MULTIBUS
g) INTERFACE CARD (BOB DALY - ANL)
h) SBC (NS 32016)
j) iSBX INTERFACE (BOB DALY - ANL)

VAX
k) DZII (DEC)
m) INTERFACE CARD (UMC - Z80 ASSOCIATED COMPUTER ASSOC.)



DAPHNE HARDWARE CONFIGURATION

Fig. 1. The DAPHNE system used for ATLAS experiments.

Each of the initial six instruments at the ATLAS accelerator will have its own CAMAC data acquisition system and will be located at distances from 30 to 300 feet from the central computer room. In the computer room the parallel multiprocessing systems and the VAX-11/750 with its disk and tape peripherals will be located. Two of the instruments, a primary and a secondary instrument, can be actively taking data at one time. It is expected that the primary instrument will be operating on-line while the other instrument is operating in a setup mode. This limitation is only operational and not imposed by the system design.

At each remote CAMAC experimental site, the data is read from CAMAC modules and packed into a CAMAC FIFO by an Event Handler. The Event Handlers are built at Argonne and are copies of the unit designed at Oak Ridge National Laboratory. A low speed asynchronous serial link from the VAX to an RS232 Crate Controller at each instrument is used to download programs into the Event Handler and to issue CAMAC commands to the Event Handler which enable, pause, or halt data acquisition. Each Event Handler and CAMAC FIFO module combination produces a single stream of raw event data which is optically coupled to the input of a Driver module. The Driver module transmits the data over RS422 differential lines to the Event Distributor in the central computer room.

At the Event Distributor, the data stream is differentially received from each remote site by Receiver modules. One of the data streams is selected by the Event Distributor to be sent to a primary Event Procesor system, while a second data stream is sent to a similarly configured secondary Event Processor system. The Event Distributor segments the data stream into individual buffers containing integral

numbers of events and transfers these buffers into iSBX Multimodule boards. Each iSBX Multimodule board is implemented with a 265 x 16 FIFO and is interfaced to the iSBX I/O bus located on each Event Processor.

The Event Processors are National 32016 Single-board Multibus computers. Each processor is configured with 128K bytes of dual-ported RAM, 32K bytes of user PROM, an interrupt controller, and a floating point co-processor. Each Event Processor obtains raw event data to be analyzed from its associated iSBX FIFO buffer. Buffers of both raw event data and fully formed VAX increment instructions with increment addresses formed by analyzing the raw data are stored in the National single-board computers.

The buffers are transferred from the Event Processors to a receiver process on the VAX by a DMA interface consisting of single intelligent Unibus Interface and a Multibus Interface located in each Multibus chassis. The single intelligent Unibus Interface can control the DMA transfer of data from as many as seven Multibus Interfaces. Both the primary and secondary channels have an intelligent Unibus Interface on the VAX.

Event Handler (EH)

The EH consists of two CAMAC modules, a programmable sequencer and a CAMAC auxiliary controller. The sequencer contains a 2K by 24-bit memory which can be programmed to pack event data into an associated CAMAC FIFO with all the variations of fixed and variable event formats needed by the ATLAS instruments. The program code is generated by a cross-assembler on the VAX and downloaded through the RS232 CAMAC link. Also, the EH has sufficient front panel inputs and outputs to control the CAMAC data acquisition instruments at the experimental sites without host processor intervention during on-line operation.

Program execution is enabled by a command from the VAX over the RS232 link. An event read is initiated by a trigger pulse at the front panel of the sequencer. The trigger can be sensed by the EH and the first CAMAC read cycle initiated within 1-2 μsecs of the leading edge of the trigger pulse for the simplest event types. Successive CAMAC read and FIFO write cycles can occur at a maximum rate of 2.0 μsec/cycle. Once started, the EH runs without intervention by any processor until the VAX issues a command to pause or halt its operation, a mode of operation which simplifies the DAPHNE design considerably. During data acquisition neither the Host computer nor the Event Processors need be aware that CAMAC is the data source and no protocol is required of the Event Distributor to move the data to the Event Procesors. The data is delivered to the Event Distributor by the EH as one continuous stream, which the Event Distributor can throttle by simply not accepting the currently available CAMAC FIFO output word.

Five units are being constructed at Argonne according the specifications supplied by ORNL. At present, one unit is finished and the checkout went smoothly. The unit is quite easy to program by someone familiar with CAMAC in general and with a specific knowledge of the operation of the CAMAC modules being used. Our goal is to provide a user interface which eliminates the need for specific knowledge of a module's operation.

Event Distributor (ED)

The ED (see Fig. 2a) consists of a single CAMAC crate in the central computer room with Receiver modules in the normal stations, an ED controller in the control stations which is linked to multiple distributed iSBX FIFO buffers over a 25-pair twisted cable, and a Driver module located in each remote experimental site which is optically coupled to the CAMAC FIFO. The CAMAC crate serves as a convenient housing for the ED. No CAMAC protocol is used by the ED. One iSBX FIFO buffer is located on each Event Processor and is interfaced to the processor according to the Multibus and IEEE iSBX I/O Bus standard.

The Receivers differentially receive the data stream from the Driver and gate the data stream onto the CAMAC Dataway when selected by the switches on the ED controller. In addition to selecting a data stream, the ED Controller is responsible for inserting an event parameter count into the data stream and for finding an empty iSBX FIFO buffer on one of the Event Processors to receive the data stream whle insuring that an event is not split between iSBX FIFOs. To accomplish this, the PAL sequencer in the ED controller (see Fig. 2b) executes the following loop continuously:

1. Gate the address counter in the ED controller onto the data lines, strobe the address control line, and sense the EMPTY status returned from the iSBX FIFO which has matched the address on the data lines with its own internal address.

2. If not EMPTY, then increment the address counter and repeat (1), else go to (3).

3. Clear the iSBX FIFO MOD 128 register and DONE bit.

4. Sense the DAVAIL (data available) line from Receiver and when true, strobe CAMAC Dataway data into the input register on the ED Controller, send DACCPT (data accepted) to the Receiver, gate the register onto the data lines and strobe the data control line loading the iSBX FIFO.

5. For each word transferred to the iSBX FIFO, increment the EVENT counter in the ED controller and when the end-of-event (EOE) bit from the Receiver becomes true, gate the EVENT counter contents onto the data lines, and strobe the data control line loading the iSBX FIFO. The EOE bit is a unique bit in the data which the EH sets to indicate that an event read has completed. Finally, clear the EVENT counter before continuing.

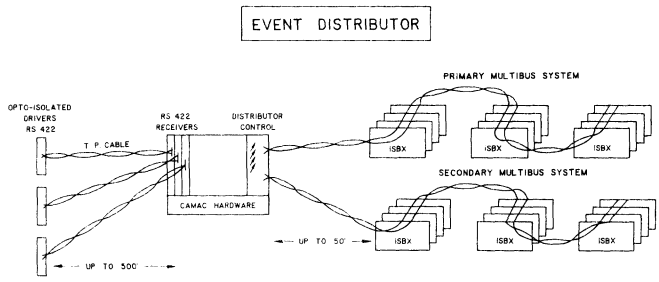
6. Continue transferring data (5) until the 1/2 FULL signal from iSBX FIFO becomes true. For each data word transferred thereafter, increment the MOD 128 counter in the ED controller.

7. The next time the EOE becomes true and after transferring the Event counter according to (5), gate the MOD 128 counter contents onto the data lines and strobe the counter line which loads the iSBX MOD 128 register (with the DONE bit set true).

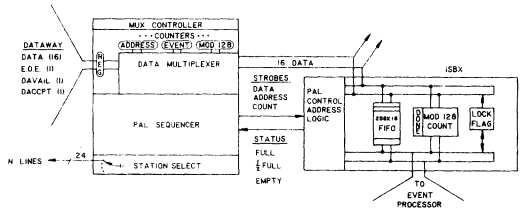
8. Search for the next empty iSBX FIFO (1).

The Event Processor can read the 1/2 FULL status of the iSBX FIFO, the MOD 128 register, the DONE bit, and the iSBX FIFO data. After sensing the 1/2 FULL bit true the Event Processor will start moving iSBX FIFO data into an internal buffer until the 1/2 FULL bit is false and the DONE bit is true. At this time the MOD 128 register will indicate the number of words remaining in the iSBX FIFO. This method allows for efficient transfer of iSBX FIFO data into a Event

Processor buffer using block move Event Processor instructions.



(a)



(b)

Fig. 2. A block diagram showing the Event Distributor layout (a) and the Event Distributor Controller linked to a iSBX FIFO (b).

The iSBX FIFOs are scanned at a 5 MHz rate. The movement of data by the ED into the iSBX FIFO overlaps the shifting out of the CAMAC FIFO data. The Event Processor reads the iSBX FIFO data at 1.3 sec per word, and it is possible for the Event Processor to be reading the iSBX FIFO while the ED is loading the same iSBX FIFO. There is no direct interaction or protocol overhead necessary to move the data from the EH to the Event Processor; the transfer is synchronized in hardware by the data available (DAVAIL) and data accepted (DACCPT) lines connecting the ED and EH through the Driver and Receiver modules. We feel that the ED operations are sufficiently fast and overlapped so that the dead time contribution of the ED to the overall data acquisition system will be negligible.

Event Processor (EP)

Five factors influenced the choice of the EP.

1. Availability--the single-board computers comprising the EPs must be available in quantity off a local supplier's shelf.

2. Power--the processor must run the typical event analysis algorithms on 16 bit data with at least the speed of a VAX-11/750.

3. Features--the processor must support a floating point coprocessor and must have features which support multiprocessor operation such as dual ported memory and bus interlocked instructions.

4. Support--cross development support must be available on VAX/VMS to assemble, compile, and debug (on the target processor) software developed for the microprocessor.

5. Price--since many of the single-board computers will be used, the price of the single-board computers must be considered.

Two single-board microcomputers fit all of our criteria, the INTEL 286/10 and the National 32016. Both microcomputers are implemented on Multibus boards and support the iSBX I/O Bus. A benchmark was run on both microcomputers to compare their speeds in performing event analysis type mathematical and logical operations. The results are shown in Fig. 3. Since the code was necessarily different for each microprocessor, the small difference in performance was not considered significant. The National 32016 was chosen since it has more cross-development tools available on the VAX running VMS and has an instruction set similar to the VAX/VMS instruction set.

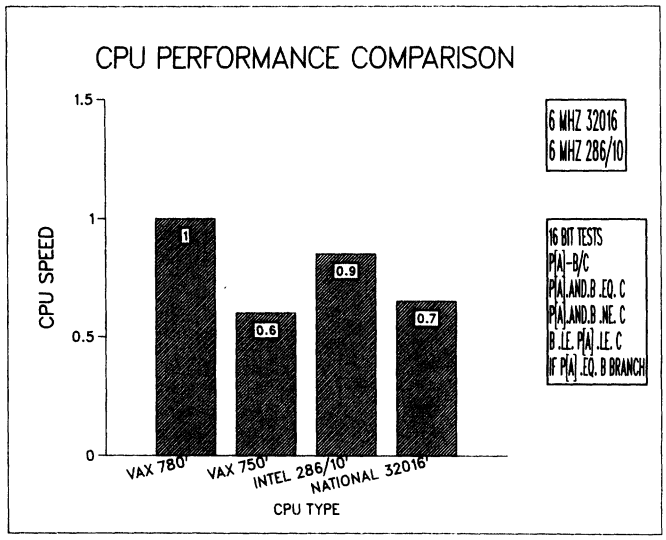


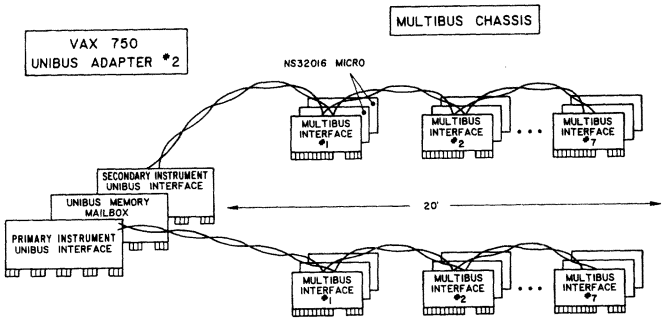
Fig. 3. Shows a CPU performance comparison of the two microcomputers evaluated as Event Processors, the host VAX-11/750, and the VAX-11/780 for reference.

Event Processor Interface (EPI)

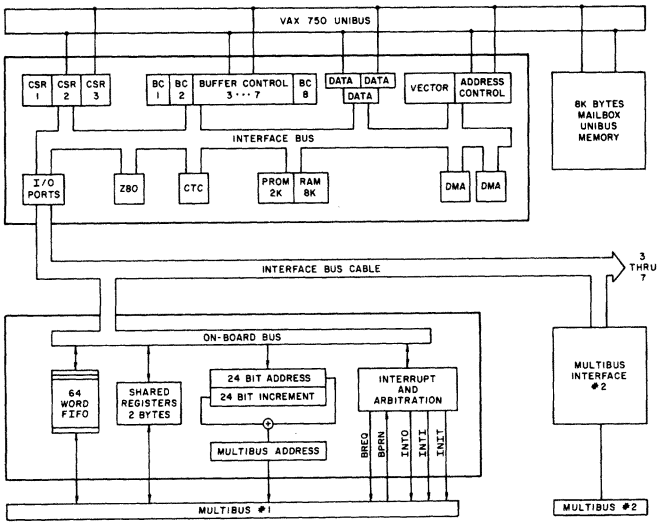
A separate EPI exists for the primary and secondary instruments (see Fig. 4a). Each EPI consists of a single intelligent DMA Unibus Interface board linked to multiple Multibus Interface boards over two 50 conductor flat cables. One Multibus Interface is required for each Multibus chassis and up to seven chassis can be linked to one Unibus Interface. In addition, both EPIs share an 8K byte Unibus memory which serves as a mailbox between VAX/VMS driver code and the EPIs.

The Unibus Interface (see Fig. 4b) contains a Z80 microcomputer, a 2K byte firmware monitor, an 8K byte RAM Z80 program memory, two DMA controller chips, and 16 bytes of shared registers which can be read and written by both the Z80 and the VAX. The DMA chips are programmed by the Z80 and are used to move blocks of data between the EPs and the Unibus. Each VMS channel supported by the EPI uses one of the shared registers as a CSR register while the other shared registers are used for special purposes by the VMS I/O channels.

Each Multibus Interface contains a 64-word 10 MHz FIFO which buffers Multibus data transfers, two bytes of shared registers which can be read and written by both the Z80 and the EPs in a Multibus chassis, a 24-bit Multibus address register which is incremented between transfers by the contents of a 24-bit increment register, and the controls which allow the Z80 to initialize the Multibus and generate two Multibus interrupts. All the registers on each Multibus Interface are accessed by the Z80 as I/O ports, and all Multibus control and data transfer is initiated by command from the Z80. During data transfer operations, Multibus data is transferred between the Multibus and the 10 MHz FIFO at 1 μsec/word, and between the FIFO and the Unibus under control of the DMA controller chip at the rate of 2.5 μsec/word.



(a)



(b)

Fig. 4. A block diagram of the Event Processor Interface layout (a) and the intelligent Unibus Interface linked to a Multibus Interface (b).

To the VMS driver the EPI appears as three separate controllers or channels. These channels are: 1) Utility Data Channel, 2) Command Message Channel and 3) Event Data Channel.

For all channels the EPI retrieves information necessary to complete a QIO operation from a page of Unibus memory assigned to that channel (the channel mailbox). The VMS driver fills in the information in the channel mailbox from the QIO parameters. To initiate execution of the QIO by the EPI, the device driver need only set a bit (GO bit) in the channel's CSR. Upon completion of the QIO operation, the EPI returns a status longword to the channel mailbox, indicating the status of the data transfer operation, and interrupts the driver.

Utility Data Channel: This channel performs general purpose DMA transfers between physical Multibus memory and VMS process buffers. It is used to download EP programs, to supply EPs with raw event data during tape replay, and to monitor the operation of the EPs during on-line operation.

Command Message Channel: This channel transfers 16 bytes of command information and up to 112 bytes of message data to any one or all (broadcast) of the EPs. The VMS device driver places the command and message data in the channel mailbox. The EPI transfers the data from the mailbox to fixed locations in the target EP's local memory reserved for commands and messages. The EPI then generates an Interrupt Level 0 on the Multibus. Only the EP with a command in its buffer will respond. At the completion of the command the EPI retrieves a 4-byte command response from the EP and returns it to the channel mailbox before starting the normal channel completion routine. The command response is returned to the VMS process as the second longword of the QIO status block. The commands are used to halt or pause data acquisition by the EPs, to cause the EPs to clear a section of local memory or to fill a portion of local memory with the message data.

Event Data Channel: This channel reads the event-related data from the EP buffers and manages the distribution of the data to as many as eight buffers in a VMS receiver process. This channel is designed with high throughput and low VMS software overhead as primary objectives. Operation on this channel is started by a QIO and stopped by a QIO, but successive transfers of event data buffers from the EPs are handled by direct communication between the VMS receiver process on the VAX and the Z80 on the EPI. This direct communication eliminates the VMS QIO processing overhead normally encountered in transferring each buffer of data. Of the sixteen bytes of shared registers on the Unibus Interface, eight are used for buffer management. These registers are labeled BC1 through BC8 in Fig. 4b. When an EP has a buffer of data to be sent to the VAX receiver process, it sets up a control block in its local memory indicating that a buffer of event data is available and the size and address of the buffer. The EP then signals the EPI, and the EPI reads and interprets the control block. Knowing from the information contained in the control block that a buffer of event data is to be transferred, the EPI scans the BCx registers looking for a Buffer Free status. The contents of the buffer BCx indicate the buffer's status. The possible status values indicated by the bits in the BCx are:

Bit 0	Clear when buffer free to EPI, set when buffer in use
Bit 1	Set when buffer filled by EPI

Bit 2 Set when buffer ready for VAX receiver process

Bit 3 Set when buffer being processed by VAX receiver process

Bit 7 Set when buffer not used

After finding a free buffer, the EPI reads the Unibus address assigned by the VMS driver to the associated VMS receiver process buffer from the channel mailbox and transfers the data to the receiver process. The EPI then sets bit 1 in the BCx register, fills in the Buffer Completion Register with the BCx number, informs the VMS driver that an event buffer was just transferred through the contents of the I/O status longword in the mailbox, and interrupts the driver. The Buffer Completion Register is one of the 16 shared byte registers on the intelligent Unibus Interface. The VMS driver reads the I/O status longword from the mailbox and the Buffer Completion Register, and then sets an Event Flag associated with a particular buffer in the VMS receiver process. Finally, the VMS driver sets bit 2 in the BCx register. When activated, the receiver process sets bit 3 in the BCx register when starting to process the buffer and clears the BCx register when finished processing the buffer. Thus, the receiver process is able to free the buffer for further use by the EPI by directly clearing the BCx register. This method eliminates all QIO processing overhead in the VAX associated with starting a buffer transfer operation.

Status and Conclusion

The DAPHNE system will begin to take data and be debugged as a complete hardware/software system in June 1985. At that time extensive performance checking and tuning to achieve maximum throughput will be done. The system will be ready for the full operation of the ATLAS accelerator instruments in September 1985.

References

[1] D. C. Hensley, "The Event Handler II, a Fast, Programmable, CAMAC-Coupled Data Acquisition Interface," IEEE Trans. Nucl. Sci., vol. NS-26 (4), 4454 (1979).

[2] L. C. Welch, "DAPHNE...A Parallel Multiprocessor Data Acquisition System for Nuclear Physics," IEEE Trans. Nucl. Sci., vol. NS-32 (1) 238 (1985).

[3] R. T. Daly, J. R. Haumann, et al, "Data Acquisition and Control System for the IPNS Time-of-Flight Neutron Scattering Instruments," IEEE Trans. Nucl. Sci., vol. NS-26 (4) 4554 (1979).

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