FERMILAB-Conf-89/130

FEREAD Front End Readout Software for the Fermilab PAN-DA Data Acquisition System *

Terry Dorries, Margaret Haire, Carmenita Moore, Ruth Pordes, Margaret Votava
Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510 U.S.A.

May 1989

* Presented at "Real Time Computer Applications in Nuclear, Particle, and Plasma Physics," Williamsburg, Virginia, May 16-19, 1989.

FEREAD
Front End Readout Software for the Fermilab PAN-DA Data Acquisition System

Terry Dorries, Margaret Haire,
Carmenita Moore, Ruth Pordes, Margaret Votava

Online and Data Acquisition Software Groups
Fermi National Accelerator Laboratory (*)
Batavia, Il 60510

The FEREAD system provides a multi-tasking framework for controlling the execution of experiment specific front end readout processes. It supports initializing the front end data acquisition hardware, queueing and processing readout activation signals, cleaning up at the end of data acquisition, and transferring configuration parameters and statistical data between a "Host" computer and the readout processes.

FEREAD is implemented as part of the PAN-DA [1] software system and is designed to run on any Motorola 68k based processor board. It has been ported to the FASTBUS General Purpose Master (GPM) interface board and the VME MVME133A processor board using the pSOS/Microtec environment.

OVERVIEW

Most fixed target experiments at Fermilab have employed data acquisition architectures in which the "intelligence" of the front end data acquisition system resided in computers (VAXs, MicroVAXs, and PDP-11s) outside of the front end electronics path. The availability of general purpose processor modules in FASTBUS and VME allows the front end intelligence to move down from these computers and into the front end electronics crates.

In order to facilitate these new architectures, the Front End Readout (FEREAD) software provides a framework within the front end electronics for controlling the execution of user supplied front end readout and monitoring processes. To achieve this, the FEREAD system offers the following functionality:

o Allow for control of data taking from a host. This includes initializing, starting, stopping and pausing the front end data acquisition system.

o Provide for multiple readout and/or monitoring programs selected on the basis of user activation triggers. These activation triggers can originate from the Host computer, external electronics (NIM, TTL, etc.), or from previously activated user processes.

o Supply a mechanism for multiple user processes to share the backplane I/O port and any auxiliary I/O ports that may be present on the board .

o Allow for the transferring of configuration parameters and statistical data between the host computer and the readout processes.

o Provide for the reporting of error and status messages to the Host computer.

A Software Requirements

The current implementation of FEREAD is built on the Fermilab enhanced pSOS/pROBE multitasking operating system kernel [5,8]. This operating system must be ported to the board before using FEREAD.

The communication between the FEREAD systems and the Host computer is accomplished using the Remote Procedure Execution [7] (RPX) software package. The RPX software package is described in detail in a paper submitted to this conference. Briefly, RPX allows subroutines to be executed on a remote CPU. This in turn allows a substantial portion of the FEREAD system to be implemented as an object library, thus reducing the design restrictions placed on the Host data acquisition system in order to interface to FEREAD. For example, if the Host wishes to start the front end data acquisition system it simply calls the appropriate FEREAD subroutine which is then executed on the readout board containing the FEREAD software.

(*)sponsored by DOE Contract No.DE-AC02-76CH03000

B Hardware Requirements

FEREAD is designed to run on any Motorola 68K based processor board residing within the front end hardware system. It requires 50 KBytes of memory for the FEREAD software, 64 KBytes for the pSOS/pROBE multitasking operating system kernel and 45 KBytes for the Remote Procedure Execution software.

The processor board running the FEREAD system must have a suitable communications path (RS232, Ethernet, etc.) to a "Host" computer, as the FEREAD programs are tightly coupled to one or more Host processes responsible for controlling and monitoring the overall data acquisition system. For simple data acquisition hardware systems it is feasible that these controlling and monitoring processes could reside on the same board as the FEREAD system.

FEREAD DESIGN

Figure 1 depicts the overall design of the FEREAD system.

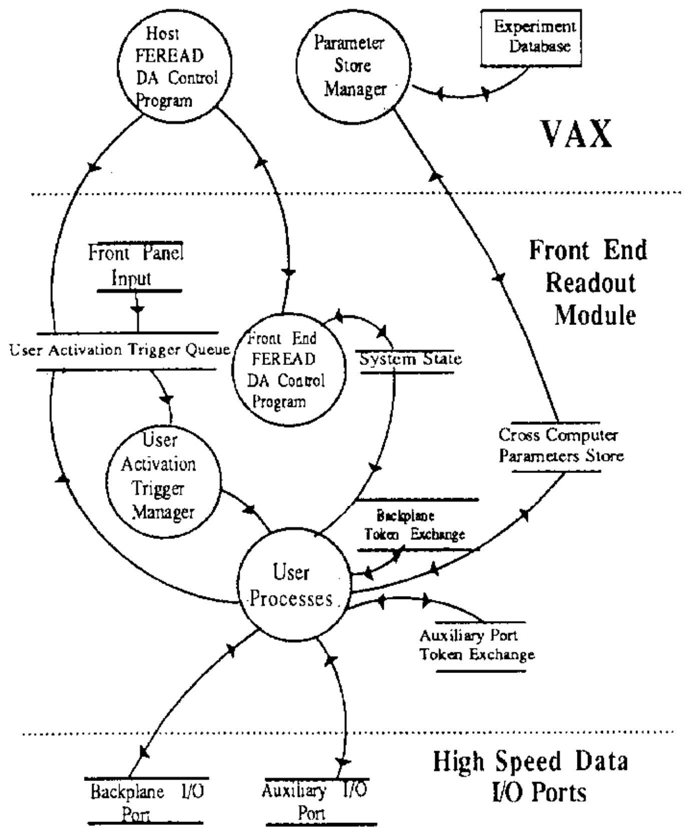
Host FEREAD DA Control Program
Parameter Store Manager
Experiment Database
VAX
Front Panel Input
Front End Readout Module
User Activation Trigger Queue
Front End FEREAD DA Control Program
System State
User Activation Trigger Manager
Cross Computer Parameters Store
Backplane Token Exchange
User Processes
Auxiliary Port Token Exchange
High Speed Data I/O Ports
Backplane I/O Port
Auxiliary I/O Port

Figure 1: FEREAD System Design Overview

FEREAD is split into three components: the Host computer, the front end readout module, and the high speed DATA I/O ports available to the readout module. For the GPM implementation of FEREAD [2,3,4] these components consist of a VAX or MicroVAX acting as the Host computer, the GPM 68K processor board acting as the readout module, and FASTBUS supplying the high speed Data I/O ports.

The FEREAD software system is divided into five subsystems: user readout and monitoring processes, data acquisition control, user activation trigger management, parameter store management, and high speed data I/O port management.

A User Processes Subsystem

It is expected that the kernel readout and monitoring processes will be written by the users to match the specific requirements of their experiments. The FEREAD system has a well defined structure in which these user processes are to execute and includes a set of user callable subroutines to allow the user processes to interact with the FEREAD system.

User processes are divided into classes according to when they are allowed to execute. The classes currently supported are:

o Class I - Process is to be downloaded to the readout module but never executed.

o Class II - Process is to execute when its User Activation Trigger is encountered by the User Activation Trigger Manager.

o Class III - Process is to be executed each time the Initialize User Processes DA control signal is received from the Host (see DA control subsystem).

o Class IV - Process is to be executed each time the Stop User Processes DA control signal is received from the Host (see DA control subsystem).

Class I user processes can be dynamically changed to Class II, III, or IV processes. This allows diagnostic programs to be built into the FEREAD system and inserted into the data acquisition processes as needed.

Class III and IV user processes are used to perform pre-DA and post-DA functions. For example, a Class III user process could be used to initialize a bank of TDCs or ADCs.

Class II user processes are used to perform the main readout functions. Each Class II user process has associated with it a User Activation Trigger. Once created, these processes execute any required initialization code and enter a sleep state until their trigger is processed by the User Activation Trigger Manager, at which time they begin their readout operation. After completing their readout task they again enter a sleep state until their next trigger is processed.

A.1 Example User Subsystem Implemented On GPM

The first implementation of the FEREAD system has been done on the FASTBUS GPM. This includes an example user process for reading out data from LeCroy 1892 memory modules and then pushing the data into VME ACP [10,11] nodes. Data may be further manipulated in the ACP nodes before being logged to tape. This example process includes code for initializing the board specific hardware of the GPM. While a particular readout process may be unique to a given implementation of FEREAD, the framework within which the process runs remains the same for every FEREAD system.

B Data Acquisition Control Subsystem

The front end data acquisition control subsystem consists of two FEREAD DA control programs, one that resides on the Host computer and one that resides on the front end readout module. These control programs are implemented using a set of RPX subroutines. The Host control program receives control commands from the user and calls the appropriate subroutine which is executed on the readout module. Once the control operation has completed, the overall status of the operation is returned to the Host. The current state of the front end data acquisition system is communicated to other FEREAD subsystems through a System State control word (see Figure 1).

The following control subroutines are currently supported within FEREAD.

B.1 Initialize User Programs

This control subroutine causes the front end DA control program to initialize its internal data structures and to sequentially create the Class II and III user processes as defined in the table of user processes (see section on user processes). Once created, each user process is allowed to execute any initialization code required by the process. The user process is free to use any input and/or output ports available on the board. Once a user process completes its initialization, it signals Completion of Initialization back to the front end DA control program. The Class II user processes then enter a sleep state waiting for the User Activation Trigger Manager to send their activation signal.

B.2 Enable Data Taking

The Enable Data Taking control subroutine places the FEREAD system in the Running state, thus enabling the User Activation Trigger Manager to begin processing user activation signals.

B.3 Pause Data Taking

Calling this control subroutine causes the front end DA control program to set the System State to a Pause Pending state. It is up to the user processes to periodically monitor the System State to determine if a change in state is pending (the FEREAD system supplies user callable routines to facilitate this System State monitoring). Once all user processes have completed or paused, or the pause timeout has occurred, the front end DA control program disables further user activation signals and places the System in the Paused state.

All user processes are prevented from running while the system is in a Paused state. The FEREAD system locks out the user processes by executing a high priority "idle" loop until the Host calls the next control subroutine.

B.4 Resume Data Taking

This control subroutine causes the front end DA control program to enable user activation signals, set the FEREAD System System to the Running state and activate all user processes waiting at the pause queue.

B.5 Stop Data Taking

The Stop Data Taking control subroutine causes the front end DA control program to set the System State to Stop Pending and wait for any active user processes to signal that they are done. Once all active user processes have signaled their completion, or a timeout condition has occurred, the front end DA control program disables further user activation signals, deletes all user processes, cleans up the system tables and sets the system to a Stopped state. It then searches the table of user processes (see section on user processes) and sequentially creates and executes any Class IV user processes.

B.6 Abort Data Taking

The Abort control subroutine causes an immediate Stop to occur - active user processes are given no notification that a Stop state is pending nor are the Class IV user processes allowed to execute.

C User Activation Trigger Management Subsystem

User Activation Triggers are 32-bit unsigned integers sent to the FEREAD system informing it that a particular user process, or set of user processes, are to begin execution. These activation triggers can be sent from the Host computer, from the front panel input port (assuming the readout module contains such a port), or from running processes. User processes wait at Activation Trigger Exchanges[5,8], in a blocked state (i.e. not absorbing any CPU cycles), until informed that their trigger has arrived. Once informed, they perform the readout or monitoring functions associated with the trigger and reattach to their Activation Trigger Exchange (Figure 2).

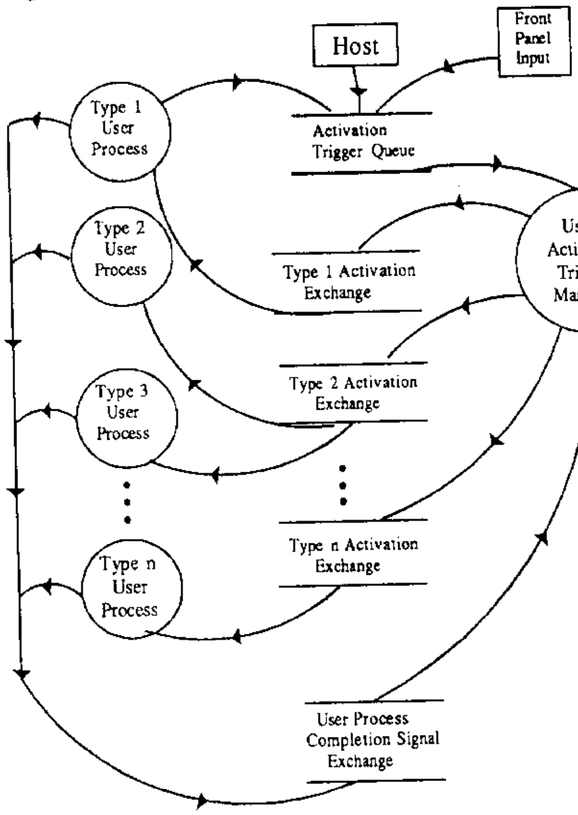
Host
Front Panel Input
Activation Trigger Queue
Type 1 User Process
Type 2 User Process
Type 3 User Process
Type n User Process
Type 1 Activation Exchange
Type 2 Activation Exchange
Type n Activation Exchange
User Activation Trigger Manager
User Process Completion Signal Exchange

Figure 2: Activation Trigger Subsystem

The User Activation Triggers are sequentially processed by the User Activation Trigger Manager. The Trigger Manager is responsible for locating the Activation Trigger Exchange corresponding to the trigger and informing the user processes waiting at the exchange that their trigger has arrived. Activation Triggers arriving while a particular trigger is being processed are queued until all user processes associated with the current trigger have signaled the Trigger Manager that they have completed. User Activation Trigger sources can choose to have their triggers placed at either the top or the bottom of the queue.

D Parameter Store Management Subsystem

This FEREAD subsystem provides transferring of configuration parameters and statistical data between the Host computer and front end readout controller. When the FEREAD system is generated space is created for a set of cross computer parameter stores. The number of stores, their lengths, and the name of a global symbol pointing to the starting memory location of each store are specified by the user in an ASCII text file residing on the Host computer.

These stores may in principal be written to and read from at any time by the Host computer and user processes. However, in order to insure the integrity of these stores, a process may temporarily lock read and write access to a particular parameter store.

The FEREAD parameter store management subsystem includes a set of subroutines for reading from and writing to these stores, as well as subroutines for locking parameter store read and write access. These subroutines are available to processes executing on the Host using the RPX software package.

E High Speed Data I/O Port Management Subsystem

Although the FEREAD system supports multiple user processes to be activated upon receiving a single user activation signal, the data input and output ports are single user resources (i.e. it may be essential that a user process perform several uninterrupted operations through the port). The FEREAD I/O Port management subsystem supplies a locking mechanism allowing a particular user private access to one or both ports.

The FEREAD system also supplies a set of subroutines and associated configuration parameter stores for using the data I/O ports. These routines are hardware specific and may require tailoring by the users to match their experiment specific requirements.

The GPM implementation of FEREAD includes a set of output port subroutines for pushing data through a FASTBUS to VME interface. These output port subroutines allow users to incorporate their FASTBUS data into the PAN-DA VME buffer management system.

CONCLUSIONS

FEREAD provides the framework for incorporating front end readout routines into data acquisitions (eg. VAXONLINE and PAN-DA). This frees experimenters to concentrate their software efforts on the specific front end readout requirements of their particular experiment. FEREAD is highly modularized, with many of its subsystems completely hardware independent. This allows it to be easily ported to a variety of front end readout modules.

The first implementation of FEREAD is for the FASTBUS GPM. The GPM FEREAD system includes a set of user processes for reading out LeCroy FASTBUS 1892 memories. It also includes output routines for pushing this data into VME using the PAN-DA VME buffer management scheme. Throughput for this system has been measured at just over 4 MBytes per second.

FEREAD will be ported to the FASTBUS Smart Crate Controller (FSCC) [12] which is currently under development at Fermilab. It is predicted that the throughput for this implementation of FEREAD will be in the 20-30 MByte per second range.

REFERENCES

1. See accompanying paper: "The PAN-DA Data Acquisition System", Don Petravick, et al.

2. T. Dorries, et al, "Front End Readout for PAN-DA", Fermilab Computing Department Note DS-181.

3. T. Dorries, et al, "FEREAD Output Port Control for GPM", Fermilab Computing Department Note DS-186.

4. Struck Catalog Module 500, General Purpose Master.

5. "pSOS-68K Real-time Multi-tasking Operating System Kernel", Software Components Group, Inc. 1985.

6. David Berg, et al., "SYS68K - Compound Product for pSOS/68K Development", Fermilab Computing Department Note PN-387.

7. See accompanying paper: "Remote Procedure Execution Software for Distributed Systems", Eileen Berman, et al.

8. See accompanying paper: "A Real Time Integrated Environment for Motorola 680x0 Based VME and FASTBUS Modules", David Berg et al.

9. See accompanying paper: "Software for FASTBUS and Motorola 68K Based Readout Controllers for Data Acquisition", Ruth Pordes, et al.

10. H. Areti et al, "The ACP Multiprocessor System at Fermilab", Proceedings of the XXIII International Conference on High Energy Physics, Berkeley, California, July 1986. ACP, "ACP 68020 CPU Module User's Manual", Feburary, 1986. ACP, "VMSbus Resource Manager", March 1986. ACP, "VME Crate Hardware", March 1986. ACP, "Branch Bus Specifications", March 1986. ACP, "Qbus Branch Bus Controller", March 1986. ACP, "Branch Bus to VMEbus Interface", October 1985.

11. T. Nash et al, "The ACP Multiprocessor System at Fermilab", presented at the XXIII International Conference on High Energy Physics, Berkeley, California, July 16-23 1986. 12. S .Hansen et al., "The Fermilab Smart Crate Controller", IEEE Transactions on Nuclear Science, Vol. NS-34, No. 4, August 1987.

12. S. Hansen et al., "The Fermilab Smart Crate Controller", IEEE Transactions on Nuclear Science, Vol. NS-34, No. 4, August 1987.