REAL TIME UNIX TELEMETRY SUPPORT SYSTEM

REAL TIME UNIX TELEMETRY SUPPORT SYSTEM

Steven B. Crabtree and Bobby J. Feather
Principal Engineer, Loral Data Systems, P.O. Box 3041, Sarasota, Florida
Principal Engineer, Loral Data Systems, P.O. Box 3041, Sarasota, Florida

ABSTRACT

Telemetry applications today are requiring more and more computing power. The computing industry is responding to this need with more powerful machines. With these new machines the UNIX operating system is rapidly being accepted as the system of choice for the popular lowend and midrange RISC and CISC computers. The system discussed addresses the long standing question, "Can a complete UNIX system perform in a high-data-rate real-time environment?".

This paper describes the Loral Data Systems development of a Real-Time Data Transcription System (RDTS) built for Lawrence Livermore National Laboratory and TRW. This system utilizes a powerful telemetry preprocessor, internally bus-coupled to a real time UNIX host computer. An industry-standard VME-to-VME coupling provides an efficient setup, control and computational gateway for preprocessed telemetry data. This architecture illustrates a UNIX operating system to support a pseudo-real-time telemetry application.

SYSTEM DESCRIPTION

The Real-Time Data Transcription System is a single-rack real-time UNIX data processing system. It is based on the EMR 8715 Telemetry Preprocessor and the Concurrent MC6400 computer system. The system reads, formats, and archives bit-parallel data from an analog tape recorder/reproducer to the computer's digital disk units.

The RDTS system performs the following functions:

- Reads 28-bit parallel raw data from a Tape Recorder and Error Detection and Correction System (EDCS).

- Compresses and formats 28-bit data into a 16-bit integer format suitable for processing by the computer.
- Reformats incoming data, builds an image buffer, and stores the data onto the disk subsystem.
- Copies data from the disk subsystem to the tape subsystem.
- Provides a support environment for development of new DPU microcode algorithms.
- Performs preprocessor board-level and system-level diagnostics.
- Provides a standard UNIX operating system to control the preprocessor and the acquisition process.
- Provides a standard UNIX operating system for development of software to perform analysis on the stored data.
- Provides high-resolution graphics on a 19" color monitor for the analysis display development environment.

SYSTEM HARDWARE

The RDTS system consists of the following hardware as depicted in Figure 1.

- Analog Tape Recorder to reproduce the 28-track tapes on which the parallel data is recorded.
- Error detection and correction system to virtually eliminate tape record-reproduce errors.
- Preprocessor consisting of a 20 slot chassis, one Data Input Module (DIM), two Distributed Processing Units (DPU), a System Utility Module (SUM), and a display terminal console.
- VME/VME Bus Adapter to connect the preprocessor VME bus to the computer VME bus.
- Host computer system, consisting of one computer, eight SCSI disks, four 8MM cartridge tape drives, and one 19" color monitor with a four-plane graphics co-processor.

OPERATIONAL DATA FLOW

Data is input to the system's DIMs from the tape recorder/EDCS unit. The 28 tracks of raw data consist of 2 control bits, 16 bits of system data, and 10 bits of waveform data. The two control bits identify system words and waveform word combinations. The data input modules route data words to two distributed processing units for processing via an 80 MByte per second bus.

As the DPUs receive data parameters from the DIM, the buffer-building algorithm is executed on each parameter. The algorithm separates incoming data words into 16-bit system words and 10-bit waveform words, and processes each word according to its type.

Each pair of consecutive system words are averaged to form a single 16-bit word. The resultant sample is combined with the next pair of averaged words to form a 32-bit word. The final 32-bit data word, along with the address where the word is to be located in the host computer's disk data buffer, are placed back on the PCD bus and sent to the SUM.

Each pair of consecutive 10-bit waveform words are right-justified to make two 16-bit data words. The words are appended together to make a single 32-bit word for transfer via the SUM to the host's disk data buffer.

Figure 2 illustrates the incoming data format and a segment of the disk data buffer.

The SUM serves as high-speed bus bridge between the 80 MByte/second bus and the 40 MByte/second VME bus. Data samples and this point have been combined into 32-bit data words by the DPUs to maximize the efficiency of the SUM.

Once data has been reformatted and deposited into the host's disk buffer memory space and the disk buffer is full, the DPU issues a VME interrupt to the acquisition software on the host computer. The acquisition software then writes the buffer to disk.

Figure 3 shows system data flow.

Following the archiving to disk, the CPU may copy disk data to cartridge tapes by reading data into memory and transferring it to tape via SCSI bus adapters. Since there are multiple SCSI adapters controlling disks and tape units, as many as four cartridge tapes can be copied at the same time. This reduces the time required to copy large amounts of data from the high speed disks to the relatively slow tape units.

SOFTWARE TO MAKE THE HARDWARE WORK

Figure 4 depicts a high level software block diagram for the system.

User/Menu Interface

The user interface to the system is controlled by two independent processes, the DISPATCHER and the MENU PRESENTER.

The dispatcher initiates functions in response to user selections. At user login, the dispatcher executes the menu presenter, which displays the main menu to the user. The menu presenter uses a menu definition file to determine how the menu tree is to be presented. This same file identifies the function to be performed when the user selects a menu item for execution.

The menu presenter is designed as a special built-in function of the dispatcher program, and requires no arguments. This eliminates the necessity of reading the menu definition file each time a menu is presented to the user.

Other functions started by the dispatcher are script files and processing programs. Menu functions communicate with the dispatcher and menu presenter. Figure 5 illustrates the user interface.

Tape Recorder Control

The tape recorder control program provides an interface between the host computer and the analog tape recorder unit. This program may be invoked by the menu dispatcher or by command line input. The program accepts two arguments as inputs at program initiation. The first argument is the tape drive command to be executed. The second is used to establish which RS232 serial port is to be used for communications.

The program first puts the tape drive into remote status mode and determines its current status. If the status indicates everything is OK, then the command indicated by the first argument is issued to the tape drive. Any errors or conditions are reported or redirected to the dispatcher process as necessary.

Microcode Development Environment

The Microcode Development Environment (MDE) provides the end user the ability to develop DPU algorithm processing code which can be executed on any parameter processed by the preprocessor. The MDE environment provided with this system was hosted on an IBM PC. The editors, compilers, and linkers required to build executable code for the DPUs are executed on the PC. The resultant microcode file is then transferred to the MC6400 computer for inclusion and usage by the preprocessor compiler and loader.

Compile/Load Preprocessor

The compiler package prepares information to be down-loaded from a host computer system to the preprocessor. The compiler accesses information from several data sources and writes the information in machine-readable format to a file to be used by the loader package.

Information needed varies according to the hardware configuration of the unit and includes the following items:

- Supervisor module
- Location and type of each board in the card cage
- Setup information for each board
- Routes for data streams within the chassis
- Algorithm chain associated with each parameter
- Microcode that implements each algorithm
- Arguments for each parameter

After the compiler has created the load image file, the loader is executed to transfer the compiled image into individual modules.

Prepare For Data Acquisition

Preparation for acquiring data is a two-step process. The first step requires the system operator to define the size and number of data buffers to be used for recording. This process also requires the operator to identify the disk volume swap list where data is to be stored. The operator must also establish names of the data files. The second preparation step involves initiating the acquisition task. This task loads setup information into DPU memory for use by the buffer-building algorithms, and then waits for a start-acquisition command.

Perform Data Acquisition

When loaded, the preprocessor’s DPUs are turned off. To start the data acquisition and buffer building processes, DPUs are turned on via command from the host acquisition program. Once the DPUs have been started, data begins passing thru the system utility module to the host computer, and then to the selected data disk.

The current disk volume is utilized until data acquisition is terminated, the run number (internal to the data) changes, or the volume becomes full. If the run number changes, the current volume is closed and the next volume is opened. Archiving continues with the new volume. When the current volume is filled, the same volume-swapping logic is used to record data to the next device indicated by the volume-swap list.

Monitor Data Acquisition

The system operator must be able to monitor the amount of data being recorded to disk, and to get an indication of the amount of bad and questionable data from the analog tape unit. The buffer-building algorithms executing within the preprocessor DPUs maintain the necessary status information concerning buffer counts and bad data. This information is retrieved from the DPUs at every buffer-full interrupt, and made available to the user. Figure 6 illustrates the monitor data acquisition display.

Preprocessor Front Panel Monitor and Control Operation

For this system, the CPU board normally used to control the preprocessor was removed from the chassis and the host CPU serves as the controlling computer. To support this configuration, the utility program used to control and monitor the preprocessor's individual functions was ported to the CPU environment. The relocated monitor software was then executed on operator demand, under control of the operating system. Figure 7 shows the preprocessor system status monitor. The monitor allows the operator to do the following functions via interaction with the CRT front panel:

- Monitor the processing/status of boards
- Monitor the execution of a single DPU algorithm
- Perform board-level health test
- Perform selected board-level diagnostics
- Perform system (preprocessor) thruput tests
- Perform system (preprocessor) output tests
- Perform PCD bus output tests
- Turn selected boards on/off
- Setup selected board registers
- Examine/deposit VME memory
- View/modify miscellaneous configuration items
- View system (preprocessor) status messages

Copy Data From Disk to Tape

To produce multiple copies of an archived disk data file, a utility program is used to copy a disk file to multiple tape units. The two command parameters for controlling operation are the name of the disk data file to be copied and the names of the tape units where the file is to be copied. The system provides the ability to reproduce up to four copies of the named disk file while utilizing the corresponding number of tape cartridge units.

CONCLUSION

Design objectives which were met in this system include using a VME to VME adapter to set up and control the preprocessor. This close coupling of the preprocessor to a generic VME host architecture eliminates the need for a CPU card in the preprocessor chassis. Also, the DPUs are executing a new algorithm which averages incoming data, reformats, and outputs converted data directly into the CPU memory. Next, the configuration supports inputting, processing, and recording data to disk at a sustained rate of more than 1.0 Mbyte/second, utilizing a commercial off-the-shelf UNIX operating system, AT&T System V Release 2, Berkeley 4.2 BSD compatible. Finally, the disk and cartridge tape drives perform well in this SCSI bus configuration.

ACKNOWLEDGEMENTS

We would like to thank the Loral Data Systems software and hardware development team; Larry, Gwen, Dan S., Dan C., Chuck, Mark, and James for their persistent efforts in making this project a success.

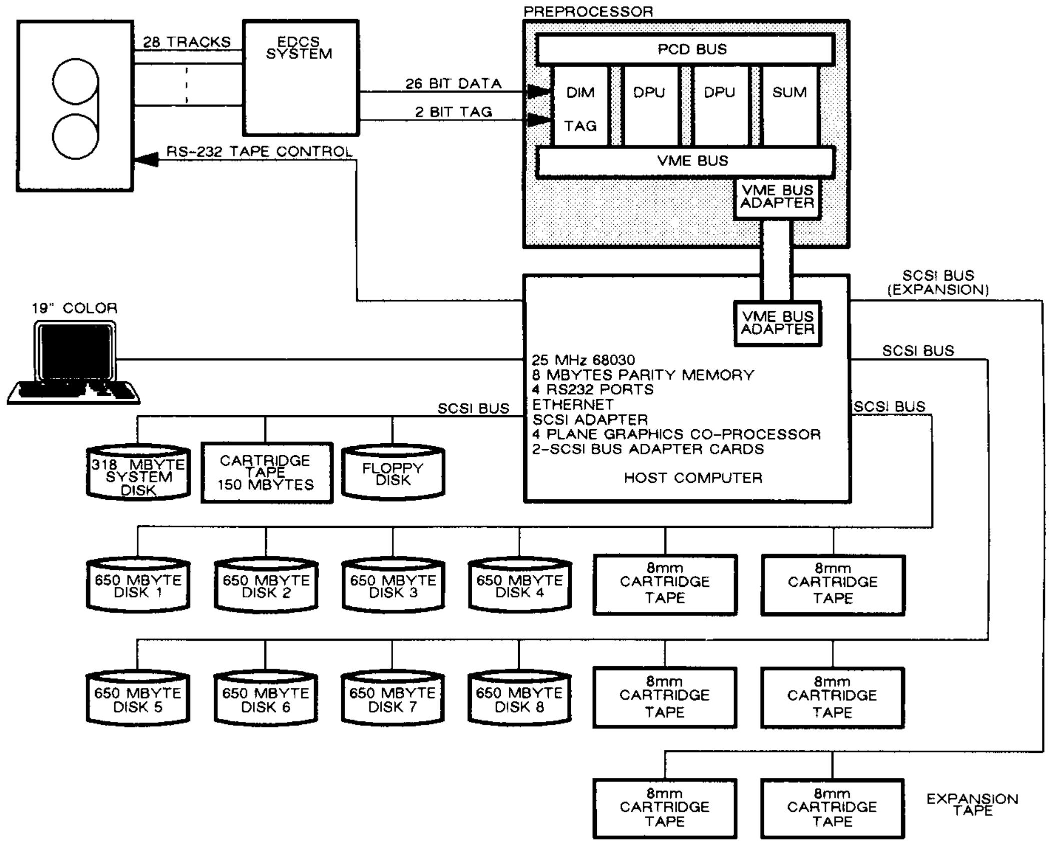


Figure 1. Real-Time Data Transcription System

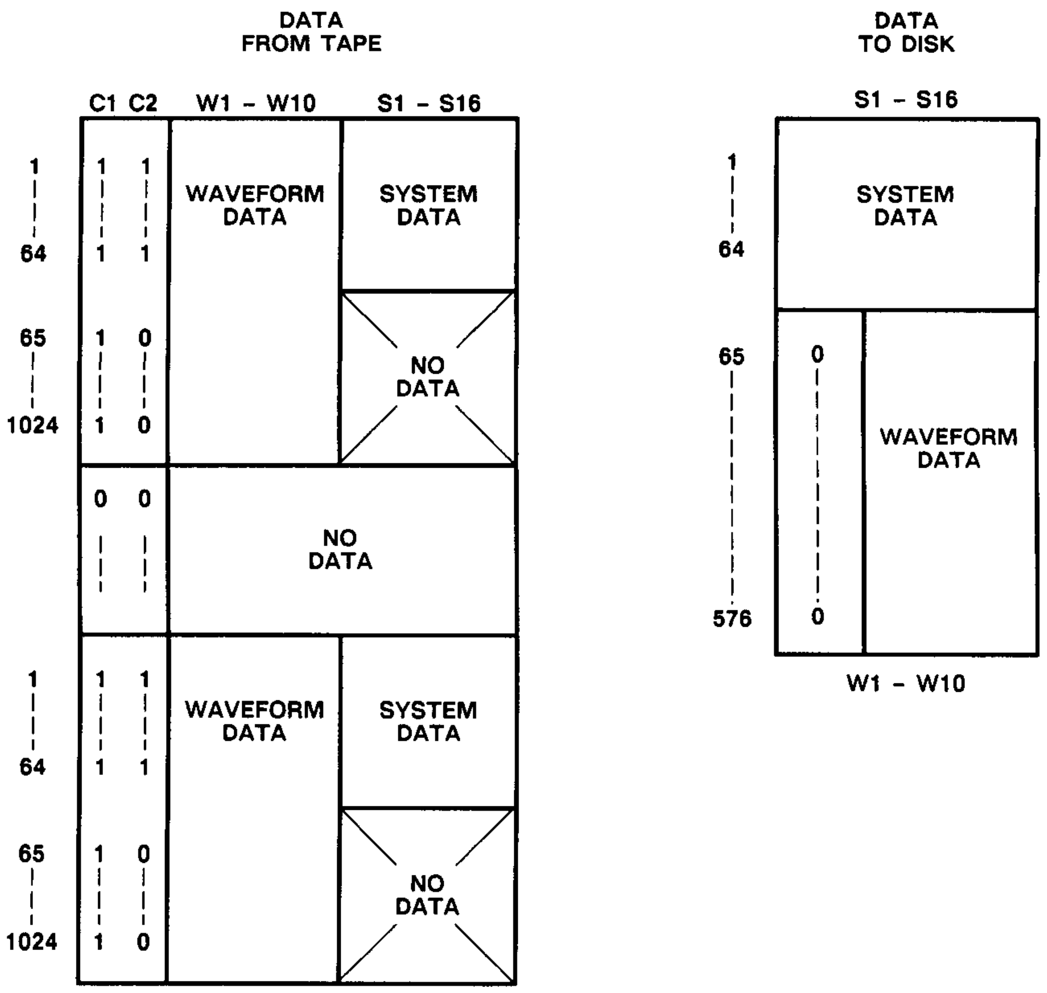


Figure 2. Input / Output Data Format

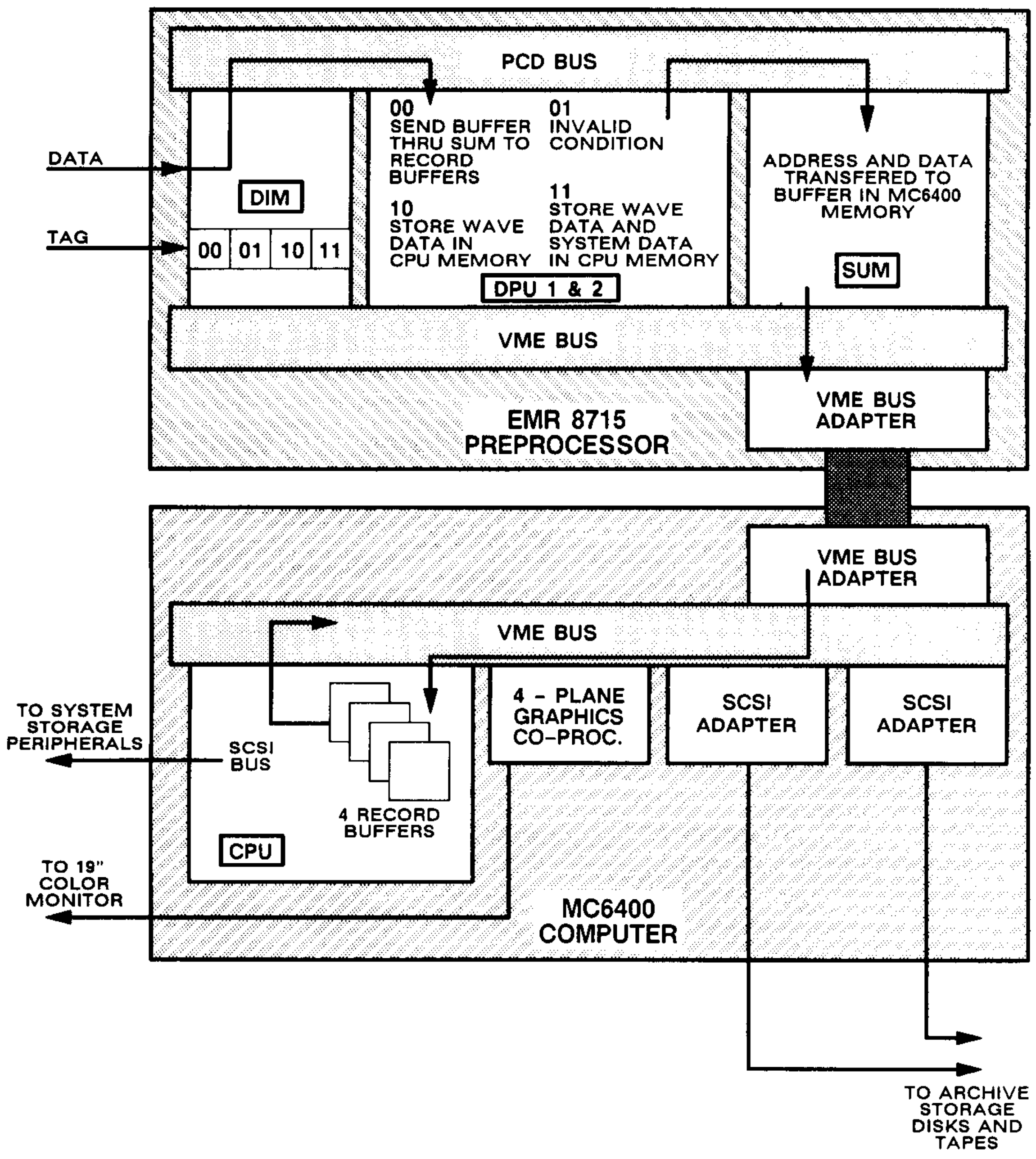

PCD BUS
DATA
TAG
DIM
00
01
10
11
00 SEND BUFFER THRU SUM TO RECORD BUFFERS
01 INVALID CONDITION
10 STORE WAVE DATA IN CPU MEMORY
11 STORE WAVE DATA AND SYSTEM DATA IN CPU MEMORY
DPU 1 & 2
ADDRESS AND DATA TRANSFERED TO BUFFER IN MC6400 MEMORY
SUM
VME BUS
VME BUS ADAPTER
EMR 8715 PREPROCESSOR
VME BUS ADAPTER
VME BUS
TO SYSTEM STORAGE PERIPHERALS
SCSI BUS
4 RECORD BUFFERS
CPU
4 - PLANE GRAPHICS CO-PROC.
SCSI ADAPTER
SCSI ADAPTER
TO 19" COLOR MONITOR
MC6400 COMPUTER
TO ARCHIVE STORAGE DISKS AND TAPES

Figure 3. System Data Flow

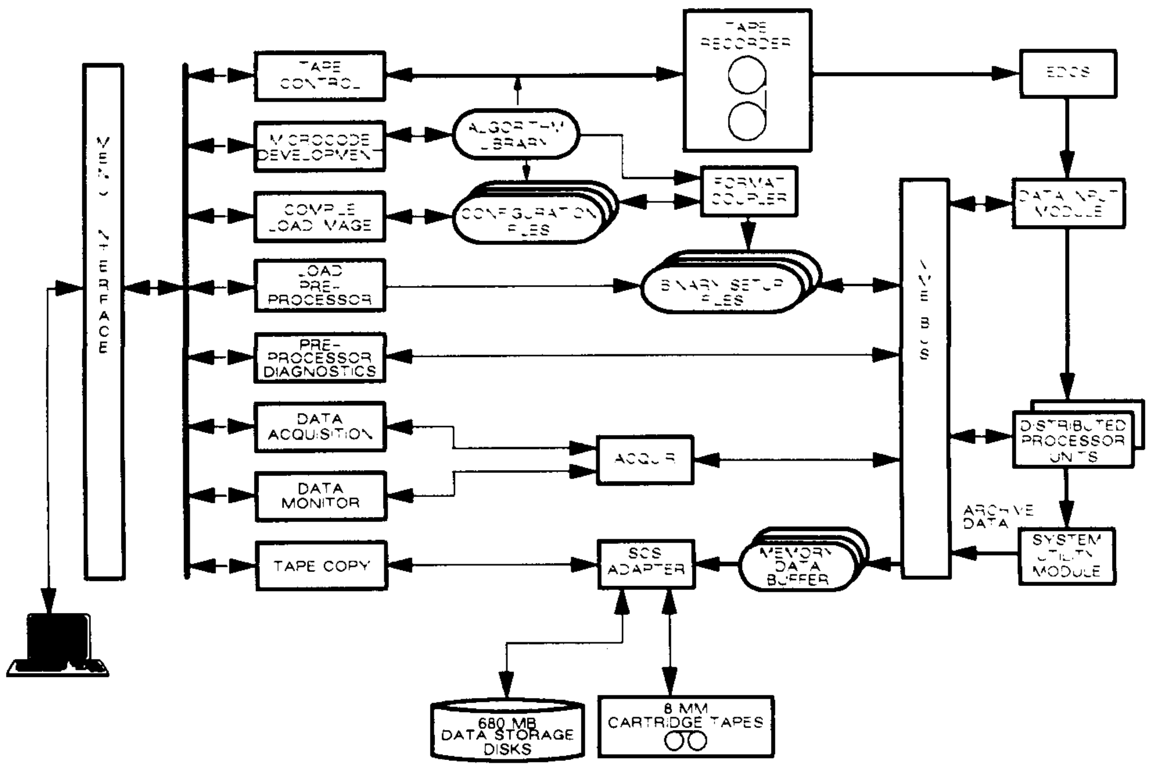


Figure 4. Software Block Diagram

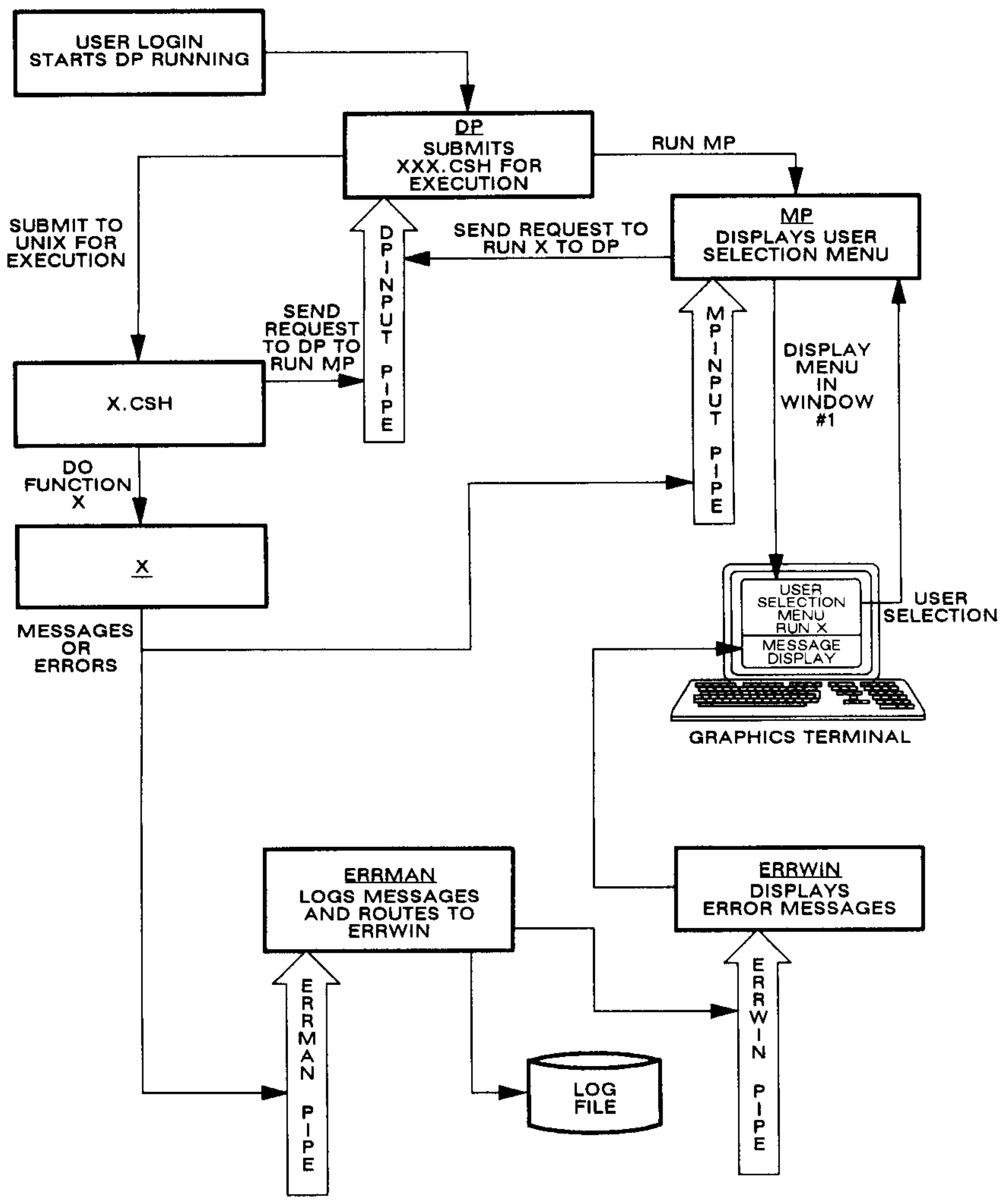
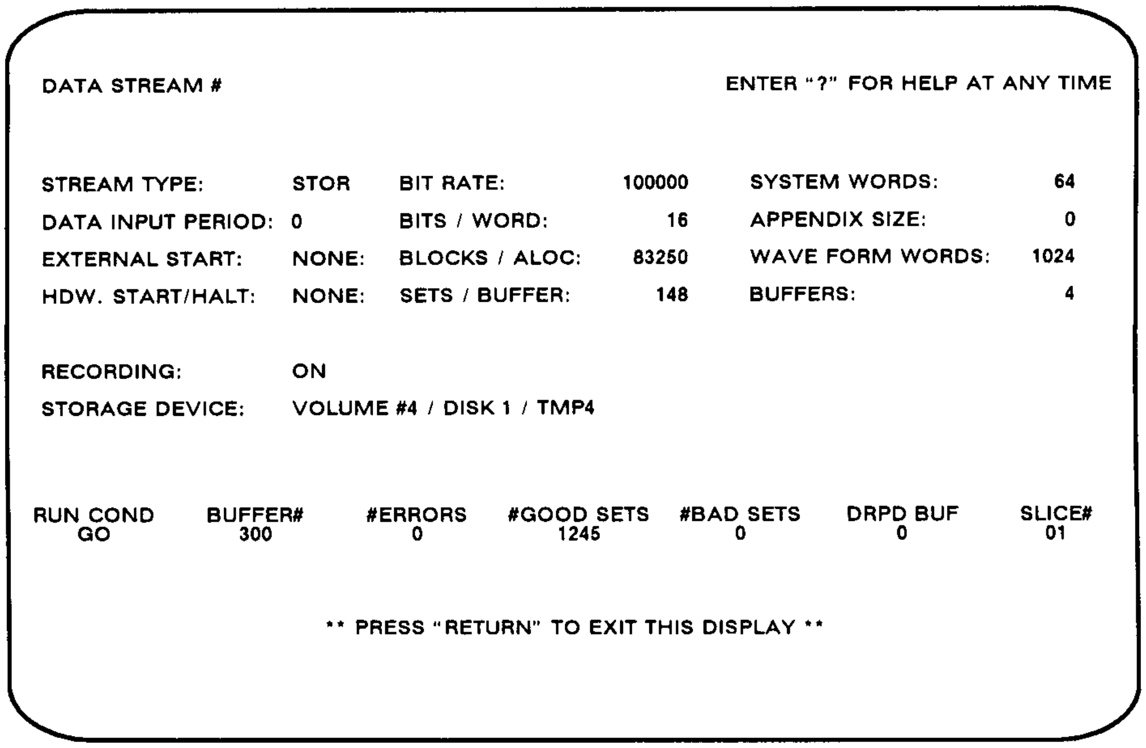
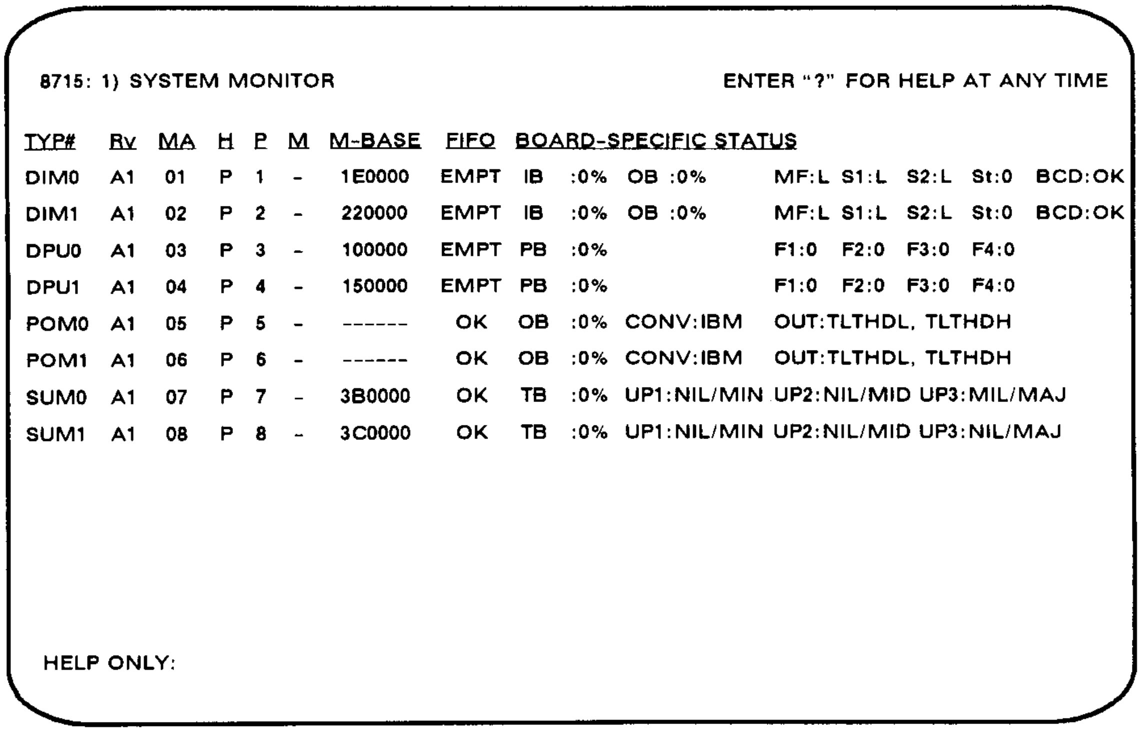

USER LOGIN STARTS DP RUNNING
DP SUBMITS XXX.CSH FOR EXECUTION
RUN MP
MP DISPLAYS USER SELECTION MENU
SUBMIT TO UNIX FOR EXECUTION
SEND REQUEST TO RUN X TO DP
DP INPUT PIPE
SEND REQUEST TO DP TO RUN MP
X.CSH
MP INPUT PIPE
DISPLAY MENU IN WINDOW #1
DO FUNCTION X
X
USER SELECTION MENU RUN X
MESSAGE DISPLAY
USER SELECTION
MESSAGES OR ERRORS
GRAPHICS TERMINAL
ERRMAN LOGS MESSAGES AND ROUTES TO ERRWIN
ERRWIN DISPLAYS ERROR MESSAGES
ERRMAN PIPE
LOG FILE
ERRWIN PIPE

Figure 5. User Interface Data Flow



```
DATA STREAM #                                              ENTER "?" FOR HELP AT ANY TIME


STREAM TYPE:          STOR    BIT RATE:         100000   SYSTEM WORDS:          64
DATA INPUT PERIOD:    0       BITS / WORD:          16   APPENDIX SIZE:          0
EXTERNAL START:       NONE:   BLOCKS / ALOC:     83250   WAVE FORM WORDS:     1024
HDW. START/HALT:      NONE:   SETS / BUFFER:       148   BUFFERS:                4

RECORDING:            ON
STORAGE DEVICE:       VOLUME #4 / DISK 1 / TMP4


RUN COND     BUFFER#     #ERRORS     #GOOD SETS    #BAD SETS    DRPD BUF     SLICE#
   GO          300          0           1245           0            0           01


                       ** PRESS "RETURN" TO EXIT THIS DISPLAY **
```

Figure 6. Monitor Data Acquisition Menu



```
8715: 1) SYSTEM MONITOR                                 ENTER "?" FOR HELP AT ANY TIME

TYP#  Rv  MA  H  P  M  M-BASE   FIFO  BOARD-SPECIFIC STATUS
DIM0  A1  01  P  1  -  1E0000   EMPT  IB  :0%  OB :0%       MF:L  S1:L  S2:L  St:0  BCD:OK
DIM1  A1  02  P  2  -  220000   EMPT  IB  :0%  OB :0%       MF:L  S1:L  S2:L  St:0  BCD:OK
DPU0  A1  03  P  3  -  100000   EMPT  PB  :0%               F1:0  F2:0  F3:0  F4:0
DPU1  A1  04  P  4  -  150000   EMPT  PB  :0%               F1:0  F2:0  F3:0  F4:0
POM0  A1  05  P  5  -  ------    OK   OB  :0%  CONV:IBM     OUT:TLTHDL, TLTHDH
POM1  A1  06  P  6  -  ------    OK   OB  :0%  CONV:IBM     OUT:TLTHDL, TLTHDH
SUM0  A1  07  P  7  -  3B0000    OK   TB  :0%  UP1:NIL/MIN UP2:NIL/MID UP3:MIL/MAJ
SUM1  A1  08  P  8  -  3C0000    OK   TB  :0%  UP1:NIL/MIN UP2:NIL/MID UP3:NIL/MAJ





 HELP ONLY:
```

Figure 7. System Monitor Display Page