Data Acquisition System for Light-Ion Irradiation Creep Experiment

P. L. Hendrick
T. J. Whitaker

July 1979

Prepared for the U.S. Department of Energy under Contract EY-76-C-06-1830

Pacific Northwest Laboratory
Operated for the U.S. Department of Energy by Battelle Memorial Institute

Battelle
NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

The views, opinions and conclusions contained in this report are those of the contractor and do not necessarily represent those of the United States Government or the United States Department of Energy.

PACIFIC NORTHWEST LABORATORY
 operated by
   BATTELLE
 for the
UNIVERSITY OF WASHINGTON
Under Contract EY-76-C-06-1830

Printed in the United States of America
Available from
National Technical Information Service
United States Department of Commerce
5285 Port Royal Road
Springfield, Virginia 22151
Price: Printed Copy $_____; Microfiche $3.00

<table>
<thead>
<tr>
<th>Pages</th>
<th>Selling Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>001-025</td>
<td>$4.00</td>
</tr>
<tr>
<td>026-050</td>
<td>$4.50</td>
</tr>
<tr>
<td>051-075</td>
<td>$5.25</td>
</tr>
<tr>
<td>076-100</td>
<td>$6.00</td>
</tr>
<tr>
<td>101-125</td>
<td>$6.50</td>
</tr>
<tr>
<td>126-150</td>
<td>$7.25</td>
</tr>
<tr>
<td>151-175</td>
<td>$8.00</td>
</tr>
<tr>
<td>176-200</td>
<td>$9.00</td>
</tr>
<tr>
<td>201-225</td>
<td>$9.25</td>
</tr>
<tr>
<td>226-250</td>
<td>$9.50</td>
</tr>
<tr>
<td>251-275</td>
<td>$10.75</td>
</tr>
<tr>
<td>276-300</td>
<td>$11.00</td>
</tr>
</tbody>
</table>
DATA ACQUISITION SYSTEM FOR
LIGHT-ION IRRADIATION CREEP EXPERIMENT

P. L. Hendrick
T. J. Whitaker

July 1979

Prepared for the
U.S. Department of Energy
under Contract EY-76-C-06-1830

Pacific Northwest Laboratory
Operated for the U.S. Department of Energy
by Battelle Memorial Institute
ACKNOWLEDGEMENTS

The authors acknowledge the support provided by the Division of Basic Energy Sciences, U.S. Department of Energy. Additionally, the authors acknowledge hardware support provided by Jan Goolsbey and Jerry Jackman of the Craft and Operation Services Department and comments received from Conny Pavloff and Charles Henager, Jr. during the software development period. Finally, the editorial review provided by Judy Hooper is gratefully acknowledged.
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>iii</td>
</tr>
<tr>
<td>SUMMARY</td>
<td>1</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>3</td>
</tr>
<tr>
<td>OBJECTIVES OF SOFTWARE</td>
<td>5</td>
</tr>
<tr>
<td>PROGRAM DESCRIPTION</td>
<td>7</td>
</tr>
<tr>
<td>DAC04 (DATA ACQUISITION, REVISION 04) DESCRIPTION</td>
<td>7</td>
</tr>
<tr>
<td>GRAFTR (GRAPH TRANSIENT DATA) DESCRIPTION</td>
<td>11</td>
</tr>
<tr>
<td>GRAFAV (GRAPH AVERAGED DATA) DESCRIPTION</td>
<td>13</td>
</tr>
<tr>
<td>TYPAV (TYPE AVERAGED DATA) DESCRIPTION</td>
<td>15</td>
</tr>
<tr>
<td>TYPLOG (TYPE LOG) DESCRIPTION</td>
<td>16</td>
</tr>
<tr>
<td>RETXT (READ TEST DATA) DESCRIPTION</td>
<td>17</td>
</tr>
<tr>
<td>REINT3 (IREAD INITIALIZATION DATA, REVISION 3) DESCRIPTION</td>
<td>17</td>
</tr>
<tr>
<td>RESET (RESET GRAPHICS DISPLAY) DESCRIPTION</td>
<td>19</td>
</tr>
<tr>
<td>GETGAI (GET GAIN)</td>
<td>19</td>
</tr>
<tr>
<td>TIMDAT (GET TIME AND DATE) DESCRIPTION</td>
<td>20</td>
</tr>
<tr>
<td>DECDAT (DECODE DATE)</td>
<td>20</td>
</tr>
<tr>
<td>ULIB (UTILITY LIBRARY) DESCRIPTION</td>
<td>20</td>
</tr>
<tr>
<td>ULIB1 (UTILITY LIBRARY, REVISION 1) DESCRIPTION</td>
<td>23</td>
</tr>
<tr>
<td>PROGRAM OPERATION</td>
<td>24</td>
</tr>
<tr>
<td>DAC04 OPERATION</td>
<td>28</td>
</tr>
<tr>
<td>GRAFTR OPERATION</td>
<td>37</td>
</tr>
<tr>
<td>GRAFAV OPERATION</td>
<td>39</td>
</tr>
<tr>
<td>TYPAV OPERATION</td>
<td>40</td>
</tr>
<tr>
<td>TYPLOG OPERATION</td>
<td>44</td>
</tr>
<tr>
<td>RETXT OPERATION</td>
<td>45</td>
</tr>
<tr>
<td>REINT3 OPERATION</td>
<td>45</td>
</tr>
<tr>
<td>RESET OPERATION</td>
<td>46</td>
</tr>
<tr>
<td>APPENDIX A (COMPUTER HARDWARE DESCRIPTION)</td>
<td>A-1</td>
</tr>
<tr>
<td>APPENDIX B (COMPUTER SOURCE CODE LISTINGS)</td>
<td>B-1</td>
</tr>
<tr>
<td>APPENDIX C (PROCEDURE FOR MAKING DELETED FILES PERMANENT)</td>
<td>C-1</td>
</tr>
<tr>
<td>APPENDIX D (HARDWARE ADDRESS AND VECTOR LOCATIONS)</td>
<td>D-1</td>
</tr>
<tr>
<td>APPENDIX E (SPECIAL TIMING CONSIDERATIONS FOR DAC04)</td>
<td>E-1</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>REF-1</td>
</tr>
</tbody>
</table>
FIGURES

1 Example of Graph Produced by GRAFTR  . . . . . 12
2 Example of Graph Produced by GRAFAV  . . . . . 15

TABLES

1 Summary of Software Programs  . . . . . . . . . 8
2 Summary of Parameters in REINT3  . . . . . . . 18
3 Subroutines Contained in ULIB  . . . . . . . . 21
4 Generally Useful Programs  . . . . . . . . . 25
5 MACRO and FORTRAN Development Programs  . . . . 26
6 Summary of Disk Space Utilization  . . . . . . . 31
7 Gain Code Interpretation  . . . . . . . . . 31
8 DAC04 Input Codes  . . . . . . . . . . . 36
9 Data Table Produced by DAC04  . . . . . . . . 37
10 Disk Block Data Produced by TYPAV  . . . . . . . 41
11 Two Channel Data Produced by TYPAV  . . . . . . . 43
12 Initialization Data Produced by REINT3  . . . . . . 46
DATA ACQUISITION SYSTEM FOR
LIGHT-ION IRRADIATION CREEP EXPERIMENT

1. SUMMARY

Software has been developed for a PDP11V/03-based data acquisition system to support the Light-Ion Irradiation Creep Experiment installed at the University of Washington Tandem Van de Graaff Accelerator. The software consists of a real-time data acquisition and storage program, DAC04, written in assembly language. This program provides for the acquisition of up to 30 channels at 100 Hz, data averaging before storage on disk, alarming, data table display, and automatic disk switching. All analog data are acquired via an amplifier-per-channel type analog-to-digital converter subsystem having a resolution of 14 bits, a maximum throughput of 20 kHz, and an overall system accuracy of ±0.01%. These specifications are considered essential for the long-term measurement of irradiation creep strains and temperatures during the light-ion bombardment of irradiation creep specimens.

The software package developed also contains a collection of FORTRAN programs designed to monitor a test while in progress. These programs use the foreground/background feature of the RT-11 operating system. Under this system, the real-time program runs in foreground and has a high priority. Whenever the foreground job is suspended, a program operating in background is free to run until the foreground program requires service. In this manner, the user may monitor the test without interrupting the periodic acquisition of data.

The background programs provide a variety of services. The program, GRAFTR, allows transient data (i.e., prior to averaging) to be graphed at the graphics terminal. This program can be used as a diagnostic tool to assure, for example, that each channel is free of noise. The program, GRAFAV, allows averaged data to be read from disk and displayed graphically at the terminal. The user determines the time scale of the graphed data through his choice of the clock interrupt period and number of words averaged. The program, TYPAV, reads averaged data from disk and displays it at the terminal in tabular form.
Other programs allow text messages to be written to disk, read from disk, and allow access to DAC04 initialization data.

Although the software was designed to provide a stand-alone package, which could be operated by anyone familiar with RT-11, it is expected that the described programs will continue to evolve as actual operating experience is acquired. To this end, software has been developed in as straightforward a manner as possible to facilitate future modifications and/or enhancements. For example, extensive use has been made of DEC-supplied subroutines that permit simple modification and provide a large degree of flexibility.
This document describes software developed to support the Light-Ion Irradiation Creep Experiment presently installed at the University of Washington Tandem Van de Graaff Accelerator. This software was developed on and for use with a Digital Equipment Corporation (DEC) PDP11V/03 computer equipped with a Neff Model 620/100 analog-to-digital converter (ADC). The computer is configured with 32K words of memory, 16 channels of digital input, 32 channels of digital output, 4 channels of digital-to-analog output, 24 channels of analog-to-digital input, a dual floppy disk, a DEC VT-55 graphics terminal with hardcopy, 4 channels of serial interface, and a real time programmable clock.

The PDP11V/03 computer was purchased with the DEC RT-11 operating system and documentation. The RT-11 operating system provides a dual-job operating environment in which high-priority jobs are run in foreground and low-priority jobs are run in background. In the current application, the real-time acquisition of data is accomplished in foreground. Graphic display and other lower priority functions are performed in background. In this way, the real-time data acquisition is always given priority and background jobs are only serviced while the foreground job is idle (suspended). The real-time programming is written in MACRO (DEC's assembly language software) to speed execution. However, most background programming is written in FORTRAN. BASIC is also included but has not yet been utilized. In several instances, the FORTRAN programs call MACRO subroutines (e.g., in cases where bit manipulation is required).

We assume in this report that the user is familiar with the use of RT-11. For example, the PIP routine provides the capability of copying files from one disk to another and the EDIT routine allows one to modify and/or create code. These operations and others are described in a series of four volumes supplied with RT-11.

This report describes the objectives of the software and then the various software programs. This is followed by a discussion of program operations. Appendices include a description of the hardware, listings of the source codes, the procedure for making detailed files permanent, hardware address and vector locations, and special timing consideration for the real time program, DAC04.
3. OBJECTIVES OF SOFTWARE

The general objectives of the software were specified at the onset of development as follows:

- Provide the real-time acquisition of up to 24 analog input signals at a rate such that the sample period is small compared to the specimen time constant ($\tau \sim 200$ msec at 200°C).
- Average the data prior to storage on disk to more efficiently use disk space and increase the period between writes to disk.
- Store channel number, programmable gain, date and time in addition to data on each disk block to facilitate post-test analysis.
- Provide an auto-ranging option (via programmable gain).
- Provide the capability to display in graphical form both transient (i.e., prior to averaging) and averaged data.
- Provide the capability to restart the real-time data acquisition program within 1 min if a computer crash occurs.
- Provide the capability of automatically switching disks when a disk is full.
- Provide alarm capability should selected channels exceed preset levels.

Based upon comments and evaluation during software development, several enhancements have been added:

- The foreground program has the option of providing an automatically updated table of current data in millivolts, also showing fixed and programmable gain.
- The foreground program automatically writes a text block that allows the user to record comments relative to a particular experiment.
- A background program allows the user to write text data to disk during the test if he wishes to record comments concerning the test conduct.
• The foreground program automatically records on a disk block appropriate initialization data that may be helpful during post-test data analysis and which enables an automatic restart of DAC04 if desired.

• Background programs allow the user to read text data or initialization data.

• The user may alter, on-line, the number of data words averaged.

Other enhancements are conceptually feasible, but will not be implemented until after the computer is installed at the University and the software is evaluated. These potential enhancements include the following:

• Control specimen temperature with one of four digital-to-analog output channels. Implementation of this enhancement must await an evaluation of the computer's reliability (e.g., a system crash could introduce a serious temperature excursion). Also, the computer overhead (i.e., the frequency with which the computer samples data might have to be reduced to accommodate the temperature control function) must be considered.

• Control on/off devices such as the beam stop. An evaluation of the computer's reliability and the effect on system overhead needs to be made prior to implementation.

• Sense the status of devices such as beam stop or pump operation. This type of enhancement can readily be incorporated when the need exists.

• Transfer data to the PDP11/60 and/or VAX installed in an adjacent room at the University of Washington's Nuclear Physics Laboratory. The hardware to accomplish this task is available through serial interfaces and the programming can be performed as the need arises. The primary use of this communication would most likely occur during post-test analysis (e.g., use of disk space, plotter, line printer, or graphics terminal).

Other enhancements are expected to be required once the system is installed and operating experience is acquired. Fortunately, the system is versatile and most needs can be met provided that full consideration is given to the trade-offs involved.
4. PROGRAM DESCRIPTION

The real-time data acquisition program was run in foreground and written entirely in MACRO. All other programs were run in background and, for the most part, written in FORTRAN. A summary of these programs is given in Table 1.

4.1 DAC04 (Data ACquisition, Revision 04) DESCRIPTION

DAC04 performs all real-time functions and is run in foreground. The program is basically driven on an interrupt basis from an internal programmable clock operating at a programmed frequency of 100 kHz. The clock interrupts the program at an interval entered from the keyboard. It then branches to a clock service interrupt routine which scans the desired ADC channels and stores the data in memory before returning to the section of code from which it was interrupted. In this manner, many other peripheral functions can be performed in either background or foreground without impairing the flow of analog input data at a periodic frequency. Once DAC04 is started and initialized, it executes a portion of the foreground code until it is clock interrupted, at which point it finishes the current instruction and branches to the interrupt service routine. Once the clock interrupt has been serviced, the computer resumes execution of the main foreground program. This process can be repeated many times until eventually the foreground program reaches a TWAIT (timed wait) instruction, which suspends further foreground operation although the clock interrupts continue to occur. When foreground is suspended in this manner, background code is permitted to run for the suspension period (i.e., between clock interrupts). Once the suspension period (set at 1.67 sec) is over, the foreground program continues at the instruction following the TWAIT. In this manner a system of priorities and timing is established between the clock interrupt routine, foreground code, and background code.

The DAC04 program is divided into five major sections as follows:

1. Initialization Section
2. Main Foreground Section
3. Clock Interrupt Service Routine
4. Completion Routines
5. Message and Data Storage Sections
<table>
<thead>
<tr>
<th>Name</th>
<th>Current Version</th>
<th>Source Language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAC04</td>
<td>19-Jul-79</td>
<td>MACRO(a)</td>
<td>A real-time data acquisition program interrupted at a user-selected frequency to scan up to 24 channels of analog-to-digital input data. Other services performed include alarms and auto ranging, etc.</td>
</tr>
<tr>
<td>GRAFTR</td>
<td>19-Jul-79</td>
<td>FORTRAN(b)</td>
<td>A program which is capable of accessing transient data buffers stored by the foreground program and displaying the data graphically.</td>
</tr>
<tr>
<td>GRAFAV</td>
<td>19-Jul-79</td>
<td>FORTRAN(b)</td>
<td>A program which accesses averaged data on disk and displays it graphically.</td>
</tr>
<tr>
<td>TYPAV</td>
<td>19-Jul-79</td>
<td>FORTRAN(b)</td>
<td>A program which accesses averaged data on disk and displays it in tabular form at the terminal.</td>
</tr>
<tr>
<td>TYPLOG</td>
<td>14-Jun-79</td>
<td>FORTRAN(b)</td>
<td>A program designed to write and store text data on disk for retrieval at a later date.</td>
</tr>
<tr>
<td>RETXT</td>
<td>14-Jun-79</td>
<td>FORTRAN(b)</td>
<td>A program designed to read text data from disk.</td>
</tr>
<tr>
<td>REINT3</td>
<td>19-Jul-79</td>
<td>FORTRAN(b)</td>
<td>A program designed specifically to read initialization data associated with DAC04.</td>
</tr>
<tr>
<td>RESET</td>
<td>14-Sep-79</td>
<td>FORTRAN(b)</td>
<td>A program which resets graphic display.</td>
</tr>
<tr>
<td>GETGAI</td>
<td>10-May-79</td>
<td>MACRO(b)</td>
<td>Subroutine designed to return the gain, given the gain code. Called by GRAFAV and TYPAV.</td>
</tr>
<tr>
<td>TIMDAT</td>
<td>19-Jul-79</td>
<td>MACRO(b)</td>
<td>A FORTRAN callable routine which prints the current time and date.</td>
</tr>
<tr>
<td>DECDAT</td>
<td>15-Jun-79</td>
<td>MACRO(b)</td>
<td>A program designed to decode and display the date at the terminal.</td>
</tr>
<tr>
<td>ULIB</td>
<td>16-Mar-78</td>
<td>MACRO(a,b)</td>
<td>Set of library routines used extensively in DAC04 and to a lesser extent in background programs.</td>
</tr>
<tr>
<td>ULIB1</td>
<td>15-Jun-79</td>
<td>MACRO(b)</td>
<td>A library consisting of ULIB plus GETGAI, TIMDAT, and DECDAT.</td>
</tr>
</tbody>
</table>

(a) Run in foreground  
(b) Run in background
The initialization section contains the code that queries the user and accepts the initialization data that the code requires to run. This section also writes the initialization data block (DATA.INT) and the text data block (DATA.TXT) to disk. The user enters from the keyboard the following parameters:

- auto restart option
- clock interrupt period in msec (typically 10)
- data file extension (typically A)
- text data (up to 510 characters)
- number of channels/scan (maximum of 30)
- channel/gain code for each channel
- fixed gains
- alarm set points
- auto ranging option
- number of data words to be averaged.

The initialization section is also responsible for printing various error messages, performing calculations and displaying useful information relative to program startup at the console terminal. In the event that the auto restart option is selected, the code reads the required data from a previously created initialization file.

The main foreground section is that MACRO code immediately following the initialization section. Tasks performed within the foreground main section tend to be repetitive, but at a much slower rate than the clock interrupts. For example, the writing of data to disk is perhaps the most important function performed in this section. The functions are performed in the order given below:

1. Test to see if a memory buffer is full and ready to be written to disk. If full, perform the following, otherwise go to Step 2.
   a. Compute averaged data.
   b. Adjust programmable gains as required if auto gain option is selected.
   c. Get date and time of day.
   d. Write a block (256 words) of data to disk.
   e. Check digital input and messages from terminal.
f. Print data table at console if request is pending.
g. Alter number of data words averaged if request pending.
h. Check for data outside alarm limits.
i. Go to Step 4.

2. Check for message from background program (i.e., request for transient data). If message not received, post request and go to Step 4.

3. If data have been received, send transient data to background (GRAFTR).

4. TWAIT (timed wait): Suspend foreground for specified period and return to Step 1.

Thus, the foreground program executes in a repetitious fashion with a period equal to the execution time of the code plus the TWAIT period (1.67 sec). This period must be short compared to the period between disk writes so that all data gets stored on disk. The foreground main section also switches disks and increments the file name extension as each disk fills.

The clock interrupt suspends the currently executing code and causes the program to branch to the clock interrupt service routine. This occurs at a frequency specified by the user. Within the service routine, analog-to-digital conversion is initiated and the resulting data are read (approximately 50 μsec later) for each of the desired channels and at the desired programmable gain level. The ADC is operated in the random mode, which means that channels can be sampled randomly, as opposed to operation in sequential mode in which channels are sampled in ascending order only. Although somewhat greater overhead is incurred in the random mode, significantly greater versatility is achieved. Since the interrupt routine is entered at a typical frequency of 100 Hz (i.e., a period of 10 msec), it is imperative that the code within the routine be as short as possible to permit foreground and background ample opportunity to execute. It is for this reason that the alarm limits are compared with averaged data following each write to disk as opposed to within the clock interrupt routine itself.

Completion routines are those sections of code entered following certain input/output (I/O) operations. In DAC04 a completion routine is entered once the foreground receives data from background. A flag is set to signal the main section that the background program awaits transient data. A second completion
routine is entered following each write to disk. In this case a flag is cleared. If, however, a write to disk is requested before this flag clears from the previous write operation, the following message is printed: "Backed up Write to Disk Requests." This is a serious error message and usually signifies that the code is initialized improperly (e.g., the period between disk writes is too short).

The last general section of code includes messages and data storage. Organizationally, it is simpler to segregate these sections to the end. Since foreground programs are loaded high in memory, one advantage of having data storage at the end of the code is that its location is not altered if modifications are made to another section. This is helpful if one wishes to verify the value of data in a particular memory location [e.g., by the online debugging technique (ODT)].

4.2 GRAFTR (GRAph TRansient Data) DESCRIPTION

The GRAFTR program is written in FORTRAN and run in background. DAC04 must be running concurrently. The code is capable of receiving data from DAC04 in order to display transient data graphically that is stored by DAC04. Since transient data are not written to disk and since DAC04 writes over each buffer when full, one can graph only the most recent transient data. For each of the first 10 channels scanned, 512 points are stored. This, therefore, requires 5K memory storage by DAC04. This quantity of data then allows each graph to be filled (i.e., 512 points/graph).

Although the method of obtaining the data is different, GRAFTR is in most other respects similar to GRAFAV, discussed next. The only other significant difference relates to the method of acquiring and decoding the programmable gain. In GRAFAV the gain codes are read from the appropriate disk block in a byte format and then decoded by a MACRO subroutine called GETGAI. In GRAFTR, however, DAC04 returns the data for each graph desired and an octal gain code digit which FORTRAN is capable of converting to the appropriate gain. Due to the close coupling between DAC04 and GRAFTR, modifications to the communication portion of either code should be performed with care.
The user input to GRAFTR is similar to that of GRAFAV. The user is asked to input the clock interrupt period. This input permits the program to label the horizontal axis in seconds. If this feature is not desired, simply enter a return to override it. Next the user enters scan number, span, offset, and fixed gain. As with GRAFAV, the user is able to label the X and Y axis and insert a title. In addition, a summary table of programmable and fixed gain is displayed. The current data and time of day is displayed. A "*" is displayed when a graph is completed. As with other programs, this implies the statement "Type 1 to Continue." A simple return will terminate the program. A typical example of the graph produced is shown in Figure 1.

Several error messages may be displayed. The statements "Error Receiving M Buffer," "Error Receiving N Buffer," and "Error Receiving Gain Codes" mean that an error was encountered during an IRCVD or IRCVDW call. See page 4-53 of the Advanced Programmers Guide for an explanation. The statement "Foreground Job Not Running" occurs when the background program attempts to transmit a data request to the foreground program which is not received. The message, "Gain Code Outside of Range 1-7?", indicates that the foreground sent the background
an incorrect gain code which may signify a logic problem (e.g., as a result of code modification). Finally, if the user attempts to enter a scan number less than 0 or greater than 10, the code loops back to the start without an explicit error message.

4.3 **GRAFAV (GRAph AVeraged Data) DESCRIPTION**

GRAFAV is a FORTRAN program run in background. This program is capable of accessing binary data (written by DAC04) from disk and displaying it graphically at the graphics terminal. The program can be run with or without DAC04 in operation. If DAC04 is running concurrently, the user has the choice of accessing an active (temporary) file or a closed (permanent) file. This feature allows the user to view current data or data written previously.

The program queries the user about the data desired. The user must specify:
- file name
- starting block number
- number of channels/scan
- scan numbers for graph 0 and 1
- span
- offset
- fixed gain.

As an example, the data desired might be located on file, DX0:DATA.00C. (Note: DXl is assigned as the default disk drive so DATA.00C is equivalent to DX1: DATA.00C.) If the user desires to start at the beginning of the file, he would then enter zero for the starting block number.

The number of channels/scan must coincide with that used when DAC04 generated the data. The program REINT3 may be run to locate this value in the initialization data. It will probably be useful for the user to copy the initialization data produced by REINT3 for future reference.

Next the user must enter the scan numbers for each graph desired. Scan numbers refer to the order in which channels are sampled. For example, the user can arrange to have channel 2 sampled as the 30th channel scanned. Therefore, the scan number is 30 and the channel number is 2. Scan numbers are
strictly a function of the software; whereas, channel numbers are hard-wired as part of the hardware.

Finally, the span, offset, and fixed gain must be entered for each of two graphs where the span and offset are in millivolts. Span is always the value in millivolts from the bottom to top of the graph; whereas, offset is the value in millivolts at the lower edge of the graph. The program is capable of reading and correcting for programmable gain, but the user must supply the fixed gain. Fixed gain can also be acquired by running REINT3.

Once the prerequisite data are entered, the code reads the disk and draws two graphs. Each point on the graph represents an averaged data value converted to millivolts. Since the graphics hardware has the capability of displaying 512 points along the horizontal axis, the time span depends upon the period over which each data point was averaged. The X axis is divided into vertical grids at 50-point intervals to allow a time base to be identified. For example, if a clock interrupt period of 10 msec is chosen with 1000 points averaged, each averaged point represents 10 sec and the distance between each grid (50 points) represents 500 sec or 8.33 min.

Once the two graphs are drawn, Graph 1 is identified with short vertical markers along its length. The graph is identified with the current date and time, the data starting time and date; and starting block number. The data start time and date corresponds to the time and date recorded in the first block requested. In addition, a summary table is printed showing the fixed and programmable gain for each channel selected.

Finally, the user is queried and given the opportunity to enter a X-axis label, Y-axis label, and title. A sample graph is shown in Figure 2. The horizontal grids divide the graph into four equal parts vertically. Thus, if a span of 20,000 were selected with an offset of -10,000, there would be 5 volts between horizontal grids with the zero axis at the center (i.e., -10 volts to +10 volts).

Various error codes are reported by the program. For example, the code might print "IREAD ERROR CODE = -2". The IREAD refers to a programmed request listed alphabetically in Chapter 4 of the Advanced Programmer's Guide.¹
In this case, the value, -2, means that a hardware error occurred (see page 4-57). Other error messages involve the ICHCPY (page 4-35), and the LOOKUP (page 4-92) requests.

4.4 **TYPAV (TYP Average Data) DESCRIPTION**

The program TYPAV is written in FORTRAN and designed to run in background. It is designed to facilitate the display of averaged data in a tabular format at the terminal. This program, as with GRAFAV, accesses either active (temporary) or inactive (permanent) files. If a permanent file is desired, the file specification must be entered in the correct format (e.g., DX1:DATA.00A). Next the starting block number and number of channels/scan must be entered. Next the message, "Enter 1 to Type First Block Specified", appears. If desired, the code will display all the data in the first block. On the first line the date and time will appear in octal. On the next one to three lines (12 values/line) the octal channel/gain status codes are displayed in the same format entered into DAC04. In fact, they will be equivalent if the auto gain option was not selected. Below the status codes will appear the data values formatted for 12 values/line. These values are simply the digital representations of the numbers stored on disk and, therefore, have not been converted to millivolts or corrected for gain.
The second half of TYPAV enables any two channels to be displayed in millivolts. This gives a direct comparison in tabular format with the data displayed by GRAFAV. For purposes of displaying these data, the scan numbers and fixed gains must be entered. TYPAV decodes the octal channel/gain status codes for the scans selected by calling the MACRO subroutine GEGAI. The data are formatted in six columns with the data word number and block number appearing in the first and last column, respectively. The value of the first scan selected appears with its programmable gain in columns 2 and 3, respectively. The value of the second scan selected and its programmable gain, likewise, appear in columns 4 and 5, respectively. Error codes and their explanation are identical to those used in GRAFAV (see Section 4.3).

4.5 **TYPLOG (TYPE LOG) DESCRIPTION**

TYPLOG is a short background program written in FORTRAN. The program allows the user to periodically, at will, record text data on disk for purposes of noting important occurrences (e.g., tank sparks, beam drifting, equipment failures, etc). The program allows the user to name the file (maximum of six characters plus extension). Some type of name sequencing (e.g., LOG$1, LOG$2, etc) is recommended since naming a file with a name previously given to a file on the same disk will delete the previous file. The format of the code is such that six lines must be entered. For example, if four lines of text information were entered, the remaining two lines could be entered by entering an additional two returns. A maximum of 80 characters may be entered per line (limitation of VT55 terminal).

Several error messages can be generated by TYPLOG. Both IENTER and IWRITW error codes can be reported. The user should consult page 4-42 and 4-80 of the *Advanced Programmer's Guide*, respectively, for an explanation of these error codes.\(^1\) In the case of IENTER, an error code equal to -1 signifies that the channel is already in use and a specific error message is displayed. This error should not normally occur since the program first identifies an unused channel through the programmed request IGETC. As with several of these programs, the code will loop back if a file name specification is in an incorrect format. This error is detected by an error code not equal to zero for the ICSI call (page 4-36 of *Advanced Programmer's Guide*).\(^1\)
4.6 RETXT (REad TeXT Data) DESCRIPTION

RETXT is a short background program written in FORTRAN. The program is designed to read ASCII text data files generated by DAC04 (i.e., DATA.TXT) and TYPLOG. Since both of these text files are one block in length, the code reads only block 0 of the file specified. It could, of course, be easily modified to read text data files of any length.

As with TYPLOG and other programs, the programs loop back if an incorrectly formatted file-name specification is entered. The error message, "Not Enough Que Space", is printed should the code not have enough input/output channels to read the disk. See IQSET (page 4-52 of Advanced Programmer's Guide) for a detailed explanation. LOOKUP and IREAD error codes are reported. LOOKUP error code equal to -2 is specifically reported as, "Cannot Find File Specified on Device" (see Advanced Programmer's Guide for IREAD and LOOKUP error codes). The message "Fatal Read" is reported if an IWAIT error code is generated following an IREAD (see page 4-80 of the Advanced Programmer's Guide for an explanation of possible IWAIT errors).

4.7 REINT3 (REad INITialization Data, Revision 3) DESCRIPTION

REINT3 is a FORTRAN program run in background. The program is specifically designed to read the initialization data file (DATA.INT) generated by the current version of DAC04. The data are displayed in a format which would be meaningless for any other type of data. The user need only enter the file name, DATA.INT, after which the disk is read and initialization data are displayed. The first and second line display the date and time of day when DAC04 was initialized. The next four lines list parameters of potential interest. Each parameter is displayed following the symbol used in DAC04. The meaning of each parameter is given in Table 2.

Following the table of values, REINT3 types 24 fixed gain values. Each value is written below the channel number (i.e., 0 through 23). Next, REINT3 types 10 pair of alarm values. Each pair is given below the scan number (i.e., 0 through 9). The high limit is given before the low limit. Next REINT3 types the channel/gain status codes (CHNSTA) in octal in the same format.
TABLE 2. Summary of Parameters in REINT3

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Display</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBCH</td>
<td>digital</td>
<td>The NUMBER of Channels/scan (max of 30); user selectable</td>
</tr>
<tr>
<td>TRNDAT</td>
<td>digital</td>
<td>The number of Transient Data channels (set to 10)</td>
</tr>
<tr>
<td>NUMAVG</td>
<td>digital</td>
<td>The NUMBER of data values over which Average is made (min = 200; max = 32,000)</td>
</tr>
<tr>
<td>REPAVG</td>
<td>digital</td>
<td>The number of Repeated Averaged data scans per disk block</td>
</tr>
<tr>
<td>AVGSIZ</td>
<td>digital</td>
<td>The Averaged data Size in words per disk block (NUMBCH x REPAVG)</td>
</tr>
<tr>
<td>TIM</td>
<td>octal</td>
<td>High and low order Time (double precision) value of suspension period in tics (TWAIT)</td>
</tr>
<tr>
<td>AUTOGN</td>
<td>digital</td>
<td>Auto Gain option. If value = 1 auto gain is selected, otherwise fixed programmable gain</td>
</tr>
<tr>
<td>CLOCK</td>
<td>digital</td>
<td>CLOCK interrupt period in msec</td>
</tr>
<tr>
<td>DRCSR</td>
<td>octal</td>
<td>DRV-11 (ADC interface) Control Status Register address</td>
</tr>
<tr>
<td>OUT</td>
<td>octal</td>
<td>DRV-11 Output register (for sending data to ADC) address</td>
</tr>
<tr>
<td>INP</td>
<td>octal</td>
<td>DRV-11 Input register (for receiving data from ADC) address</td>
</tr>
<tr>
<td>CSR</td>
<td>octal</td>
<td>Programmable Clock Control Status Register address</td>
</tr>
<tr>
<td>BPR</td>
<td>octal</td>
<td>Programmable Clock Buffer/Preset Register address</td>
</tr>
<tr>
<td>DGOUT1</td>
<td>octal</td>
<td>Digital Output No. 1 address</td>
</tr>
<tr>
<td>DGOUT2</td>
<td>octal</td>
<td>Digital Output No. 2 address</td>
</tr>
<tr>
<td>DIGINP</td>
<td>octal</td>
<td>Digital Input address</td>
</tr>
<tr>
<td>DATE</td>
<td>octal</td>
<td>DATE code when DAC04 was initialized</td>
</tr>
<tr>
<td>TOD</td>
<td>octal</td>
<td>Time Of Day in tics past midnight when DAC04 was Initialized (high-order part)</td>
</tr>
<tr>
<td>TOD1</td>
<td>octal</td>
<td>Time Of Day in tics past midnight when DAC04 was Initialized (low-order part)</td>
</tr>
</tbody>
</table>
as entered in DAC04. The program fills in unused codes with 0 up to the maximum of 30. Since 0 is a legitimate gain code (channel 0 with a gain of 1), the user should use the value of NUMBCH to determine how many codes are meaningful.

Possible error messages are similar to previous programs described. Error codes are reported for LOOKUP and IREAD programmed requests (see pages 4-92 and 4-56, respectively, of Advanced Programmer's Guide). The LOOKUP error code of -2 is specifically reported as "Cannot Find File Specified on Device." A "Fatal Read" message is displayed if the IWAIT generates an error code following an IREAD call. (See page 4-80 of Advanced Programmer's Guide). Also, a "Not Enough Que Space" message is displayed if insufficient input/output channels are available to read the disk (page 4-52).

4.8 **RESET (RESET Graphics Display) DESCRIPTION**

RESET is a short FORTRAN program run in background. RESET is designed to clear the graphics display. No error messages are reported by RESET.

4.9 **GETGAI (GET GAIN) DESCRIPTION**

GETGAI is a short MACRO subroutine callable from FORTRAN. The FORTRAN program supplies GETGAI with a channel/gain status code and GETGAI returns the gain. This operation is efficiently performed in MACRO since bit manipulation of the channel/gain status code is required. The following format as used in GRAFAV is shown below:

```
Call GETGAI [CHNSTA(ICH0+1),IPRGN0]
```

where: CHNSTA is the array containing the channel/gain code  
ICH0 is the scan number for graph 0  
IPRGN0 is the gain value returned for graph 0.

No error messages are generated by GETGAI.
4.10 TIMDAT (Get TIMe and DAtE) DESCRIPTION

TIMDAT is a MACRO subroutine callable by FORTRAN which in turn calls the TIMOUT MACRO subroutine in ULIB (Utility LIBrary). TIMOUT displays the current time of day and date at the terminal. This chain is necessary since subroutines in ULIB are only callable from MACRO (with the exception of FILINP). No error messages are generated by TIMDAT.

4.11 DECDAT (DECode DAte) DESCRIPTION

The program DECDAT is a short MACRO routine callable from FORTRAN. The code will output an ASCII date string to the terminal given a date code (see page 2-47 of Advance Programmer's Guide for information on date code). The FORTRAN calling program uses the following format:

CALL DECDAT (IDATE)

where IDATE contains the date code. The ASCII string output to the terminal is in the following format: 6/15/79. This function is most efficiently performed in MACRO since bit manipulation of the date code is required. DECDAT calls the subroutine DIGOUT described in the next section. GRAFAV, TYPAV, and REINT3 make use of DECDAT by displaying the date stored in the first disk block specified. This subroutine is to be distinguished from the program TIMDAT, which outputs the current date and time stored by the RT-11 monitor. No error messages are generated by DECDAT.

4.12 ULIB (Utility LIBrary) DESCRIPTION

ULIB is a collection of MACRO callable subroutines compiled in a library which can be linked to a user's program. DAC04 makes extensive use of several of these programs. As described in previous sections, the MACRO program, TIMDAT, and DECDAT use ULIB.

In general, the subroutines contained within ULIB are called by the MACRO statement:

JSR R5, SUBROUTINE

Where SUBROUTINE is the name of the program. Arguments are passed on general
register R0. The terminal bell is activated to signal that the program awaits data. Overflows or illegal characters in any of the input generate error messages. All programs, except DFIX and DFLOAT, use single precision integers and/or real numbers. A brief description of each subroutine is listed in Table 3.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OINP(a)</td>
<td>Octal INPUT. An octal number (0-177777) is brought in from the terminal and stored in R0 in binary.</td>
</tr>
<tr>
<td>OOUT(a)</td>
<td>Octal OUTPUT. A binary number, stored in R0 is output to the terminal in octal. Leading zeroes are included.</td>
</tr>
<tr>
<td>DIGIN(a)</td>
<td>DIGit al INPUT. A digital integer (single precision) is brought in from the terminal and stored in R0 in binary form.</td>
</tr>
<tr>
<td>DIGOUT(a,b)</td>
<td>DIGit al OUTPUT. A binary number in R0 is output to the terminal in digital form. Leading zeroes are ignored.</td>
</tr>
<tr>
<td>FLOAT(a)</td>
<td>FLOATing point. An integer in R0 is converted to floating point format and returned in R0. The integer in R0 is lost.</td>
</tr>
<tr>
<td>FIX</td>
<td>A real number, the address of which is stored in R0 is converted to integer form after rounding off. Positive or negative overflows are flagged and 77777 (octal) is returned. The resulting integer is stored in R0.</td>
</tr>
<tr>
<td>SQRT</td>
<td>SQUARE Root. The square root of a real number, the address of which is stored in R0, is calculated and the address of the result is returned in R0. If the number pointed to by R0 is negative, a NSQRT message is typed and the program is terminated.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>TIMOUT(a,c)</strong></td>
<td><strong>TIME OUTput.</strong> The current time of day and date are output at the terminal. This program uses the line clock.</td>
</tr>
<tr>
<td><strong>LN</strong></td>
<td><strong>Natural Log.</strong> The natural log of a real number, the address of which is located in R0, is calculated and the address of the answer is returned in R0.</td>
</tr>
<tr>
<td><strong>FLTIN</strong></td>
<td><strong>FLOATing point INput.</strong> A real number is input from the terminal and the address of the result is stored in R0. The format of the input is:</td>
</tr>
<tr>
<td></td>
<td>$F = NNN.NNNNE \pm NN$ where $</td>
</tr>
<tr>
<td><strong>FLTOUT(a)</strong></td>
<td><strong>FLOATing point OUTput.</strong> A real number, the address of which is in R0, is output to the terminal in E12.5 format.</td>
</tr>
<tr>
<td><strong>DFIX</strong></td>
<td><strong>Double precision FIXed point.</strong> A single precision floating point number, the address of which is in R0 is converted (after rounding off) to a double precision integer. The address of the result is passed to the calling program in R0.</td>
</tr>
<tr>
<td><strong>DFLOAT(a)</strong></td>
<td><strong>Double precision FLOATing point.</strong> A double precision integer (address in R0) is converted to a single precision floating point number (address returned in R0).</td>
</tr>
</tbody>
</table>
### TABLE 3. Subroutines Contained in ULIB (contd)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILINP</td>
<td>FILE INPut. Input a filename through terminal. FORTRAN callable. Use JSR PC,FILINP from MACRO.</td>
</tr>
</tbody>
</table>

(a) Used by DAC04  
(b) Used by DECDAT  
(c) Used by TIMDAT

#### 4.13 ULIB1 (Utility LIBrary Revision 1) DESCRIPTION

ULIB1 is a library consisting of the following object programs:

- ULIB (library of MACRO callable subroutines)
- GETGAI (FORTRAN callable subroutine)
- TIMDAT (FORTRAN callable subroutine)
- DECDAT (FORTRAN callable subroutine)

Its creation was necessitated by a limit imposed by RT-11 on the number of object files which could be linked at once. For example, to link GRAFAV, all of the following object codes would be required:

- GRAFAV
- FORLIB
- PLOT55
- ULIB
- GETGAI
- TIMDAT
- DECDAT

where ULIB1 should be substituted for the last four codes listed. As new subroutines are developed, new libraries may be created (e.g., ULIB2, etc).
5. PROGRAM OPERATION

The operation of the programs previously discussed is intended to be self-explanatory in most instances provided that the user is generally familiar with the RT-11 operating system. The input to DAC04 is, of course, critical to its proper operation and, therefore, one or more thermal creep tests should be conducted to evaluate the choice of input. These tests may also form the basis for some enhancements and/or modifications prior to the first irradiation creep test. The thermal creep tests will also yield valuable data relative to the reliability of the computer and its peripherals. As discussed earlier, the value of several anticipated enhancements depends largely upon the computer's reliability (e.g., freedom from system crashes).

Prior to operating DAC04 the foreground/background monitor must be booted in. Several monitor commands have been included as part of STARTF.COM, which are executed automatically after the system is booted. They could likewise be executed from the keyboard. The commands are:

- ASSIGN DX1:DK  makes disk drive 1 the default disk
- SET TT:SCOPE   modifies operation of delete key
- SET USR NOSWAP makes USR resident in memory

These commands are described fully within the RT-11 documentation. (See Chapter 4 of System User's Guide.) In addition to booting in the system, the user must identify which programs should be available and on which disk drive. It is anticipated that early operating experience will largely dictate the requirements, depending primarily upon the types of operations that will be performed in background during creep testing. (See Chapter 2 of System Generation Manual for general recommendation.) An outline of the programs that may be resident on disk is given in Tables 4 and 5.

The programs listed in the first two categories of Table 4 can easily be fit onto a single disk to facilitate booting the system and execution of DAC04. Disk drive 0 is recommended for this purpose since data are initially written to drive 1.
## TABLE 4. Generally Useful Programs

RT-11 Operating System:

<table>
<thead>
<tr>
<th>Name</th>
<th>Blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DXMNFB.SYS</td>
<td>74</td>
<td>Foreground/background monitor</td>
</tr>
<tr>
<td>SWAP.SYS</td>
<td>24</td>
<td>External monitor swap blocks</td>
</tr>
<tr>
<td>STARTF.COM</td>
<td>1</td>
<td>Command file</td>
</tr>
</tbody>
</table>

To Run DAC914:

<table>
<thead>
<tr>
<th>Name</th>
<th>Blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAC914.REL</td>
<td>41</td>
<td>Real-time data acquisition program</td>
</tr>
</tbody>
</table>

Very Useful:

<table>
<thead>
<tr>
<th>Name</th>
<th>Blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIR.SAV</td>
<td>17</td>
<td>Lists files on disk</td>
</tr>
<tr>
<td>DUMP.SAV</td>
<td>7</td>
<td>Shows binary data on disk</td>
</tr>
</tbody>
</table>

Useful:

<table>
<thead>
<tr>
<th>Name</th>
<th>Blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIP.SAV</td>
<td>16</td>
<td>Transfer and copy files</td>
</tr>
<tr>
<td>DUP.SAV</td>
<td>21</td>
<td>Misc. disk file operations</td>
</tr>
</tbody>
</table>

Background Data Trend Analysis:

<table>
<thead>
<tr>
<th>Name</th>
<th>Blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRAFAV.SAV</td>
<td>32</td>
<td>Graph averaged data</td>
</tr>
<tr>
<td>GRAFTR.SAV</td>
<td>31</td>
<td>Graph transient data</td>
</tr>
<tr>
<td>TYPAV.SAV</td>
<td>34</td>
<td>Tabulate averaged data</td>
</tr>
<tr>
<td>REINT3.SAV</td>
<td>22</td>
<td>Read initialization data</td>
</tr>
<tr>
<td>RETXT.SAV</td>
<td>18</td>
<td>Read text data</td>
</tr>
<tr>
<td>TYPLOG.SAV</td>
<td>18</td>
<td>Write text data to disk</td>
</tr>
<tr>
<td>RESET.SAV</td>
<td>10</td>
<td>Clear graphic display</td>
</tr>
</tbody>
</table>
### TABLE 5. MACRO and FORTRAN Development Programs

<table>
<thead>
<tr>
<th>Name</th>
<th>Blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACRO Program Development</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDIT.SAV</td>
<td>19</td>
<td>Modify source code</td>
</tr>
<tr>
<td>MACRO.SAV</td>
<td>45</td>
<td>Assemble MACRO code</td>
</tr>
<tr>
<td>LINK.SAV</td>
<td>29</td>
<td>Link object codes</td>
</tr>
<tr>
<td>SYSMAC.SML</td>
<td>37</td>
<td>Default MACRO library</td>
</tr>
<tr>
<td>ULIB.OBJ</td>
<td>12</td>
<td>User library</td>
</tr>
<tr>
<td></td>
<td>142</td>
<td></td>
</tr>
<tr>
<td>FORTRAN Program Development:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDIT.SAV</td>
<td>19</td>
<td>Modify source code</td>
</tr>
<tr>
<td>FORTRA.SAV</td>
<td>181</td>
<td>Compile FORTRAN code</td>
</tr>
<tr>
<td>LINK.SAV</td>
<td>29</td>
<td>Link object codes</td>
</tr>
<tr>
<td>SYSLIB.OBJ</td>
<td>42</td>
<td>Default FORTRAN library</td>
</tr>
<tr>
<td>FORLIB.OBJ</td>
<td>151</td>
<td>FORTRAN library</td>
</tr>
<tr>
<td>PLOT55.OBJ</td>
<td>3</td>
<td>Graphic subroutines</td>
</tr>
<tr>
<td>ULIB1.OBJ</td>
<td>13</td>
<td>User library</td>
</tr>
<tr>
<td></td>
<td>438</td>
<td></td>
</tr>
</tbody>
</table>

As can be seen there are many programs with large space requirements. Since each disk has an effective storage space of 484 blocks (assuming two blocks for the directory) and since space must be reserved for data storage, compromises must be made. It is suggested initially that a foreground/background system disk be inserted into drive 0 and used to boot the system up and run DAC04. A disk with the background trend analysis programs would be inserted into drive 1 with data storage on the remaining blocks. It would also be desirable to include DIR and DUMP and possibly PIP and DUP if the need evolves. The inclusion of either the MACRO or FORTRAN development files for use during tests is not recommended initially since:
1. The space consumed is large and reduces the data storage capacity of the disk.

2. Program development in background increases the probability of a system crash or foreground program termination. The probability of these occurrences, however, may be small for the experienced user who is careful to avoid incorrect input (e.g., fatal monitor errors).

It would be desirable in a long test that requires more than two disks for the user to prepare two sets of disks in advance. The first set would be properly initialized disks for drive 0. The second set would have the necessary background files and be squeezed to achieve a contiguous free space for data storage (see Chapter 4 of System User's Guide for a description of all interactive commands including the SQUEEZE command) for use in drive 1.(2)

As part of the initialization process it might be useful to develop an identification system. In this manner up to 12 characters can be stored in the directory as a permanent record, in addition to the user's name. (See pages 4-77 of System User's Guide for VOLUMEID Option.)(2) As an example:

```
.DIR[CR](a,b)
DIR-F-Illegal directory

.INIT/SEG:2/VOLUMEID[CR]
Device?DX1:
DX1:/Init are you sure?[CR]

Valid?TEST 24-3-35[CR]
Owner Name?J.DOE[CR]

.DIR/VOL[CR]

Volume ID: Test 24-3-35
Owner: J. Doe
0 Files, 0 Blocks
484 Free Blocks
```

(a) [CR] indicates a carriage return entered by striking the return key
(b) Underlined characters are entered by user; all others are generated by computer.
5.1 **DAC04 OPERATION**

Once the foreground/background monitor is booted up, DAC04 can be run. The first step is to set the current date and time as follows:

- `.DATE 12-May-79[CR]` Enter date
- `.DATE[CR]` Check date
- `12-May-79`
- `.TIME 8:15[CR]` Enter time
- `.TIME[CR]` Check time
- `08:15:09`

Next, assuming DAC04.REL is resident on drive 0 and a data disk is loaded into drive 1, type:

- `.FRUN DX:DAC04[CR]`

**F >**

HAS DATE AND TIME BEEN PREVIOUSLY ENTERED? IF NOT RESTART.
ENTER CTRL/F FIRST
TYPE 1 FOR AUTO RESTART
CTRL/F[CR](a)
ENTER CLOCK INTERRUPT PERIOD (10 MS TYPICAL; 320 MS MAX)

The response F> on the first line indicates that a message from the foreground follows. The second line reminds the user to enter the date and time if not done previously. The third line reminds the user that a CTRL/F (simultaneously push CTRL and F keys) must be struck before proceeding to inform the computer that any characters that follow are intended for the foreground job. The fourth line gives the user the option of an automatic restart. If selected, the code reads the initialization file, DATA.INT, in lieu of accepting user input. As shown a carriage return was struck in lieu of the 1 needed for the auto restart option. The last line asks the user to enter the clock interrupt period. There are a number of considerations which will undoubtedly influence the choice of this parameter. Ten milliseconds is recommended initially. Therefore, the user entry continues as follows:

- `10[CR]`

(a) CTRL/F is entered by simultaneously striking the CTRL and F key
ENTER DATA FILE EXTENSION (TYPICALLY = "A")

As indicated, the user is asked to enter the data file extension. Since the code is set up to increment the extension automatically, the letter "A" is recommended. However, if the code is restarted (either by manual input or automatic restart), the user should enter the next unused extension so that the data file extensions will be continuous. Therefore:

A[CR]

293 = NUMBER OF FREE BLOCKS ON DISK

ENTER TEXT: LIMIT = 510 CHARACTERS; TERMINATE WITH @ SYMBOL

THIS IS A TEST OF DAC04 [CR]

to demonstrate its [CR]

use @[CR]

ENTER NUMBER OF CHANNELS PER SCAN; MAX=30

The user is given the free disk space to judge if the disk has been properly initialized. The "@" symbol informs the program that the text string is terminated. The user should not type more than 80 characters per line and limit the message to 510 characters. After the message is received, the program requires the number of channels/scan. This will, of course, usually correspond to the number of hardware channels wired up unless the user wishes to sample one or more channels more than once. This may be appropriate under one of the following conditions:

1. The auto gain option is not desired and so a single hardware channel is scanned several times with different programmable gain settings. This use might be appropriate for transient analysis since programmable gain is only checked at the frequency of writing to disk.

2. A time correlation is critical. For example, the temperature is required at the instant the strain is to be measured in order to correct for thermal expansion. Therefore, the same thermocouple might be read just prior to and just after the strain reading.
3. Statistical analysis may show that more readings would significantly improve accuracy. Therefore, critical readings might be read several times during each scan.

In addition to the above considerations, the user might consider the efficiency with which disk space is used. Each disk block is only filled in increments of complete scans; therefore, blank space can exist, the amount of which depends upon the number of channels/scan. The blank space is calculated by recognizing that there are 256 words per disk block. The first three words are used by the date (1 word) and time (2 words). The channel/gain status codes then use \( X/2 \) words, where \( X \) equals the number of channels/scan. If \( X \) is odd, the remainder is rounded up to the next whole number of words. Thus, both 29 and 30 channels/scan use 15 words to store the channel/gain status codes. The remaining space is then available for an integral number of complete scans. In the case of 29 channels/scan we have:

\[
256 - 3 - 15 = 238
\]

words available in which eight complete scans of 29 words each will fit, taking a total of 232 words and leaving 6 unused blanks. This turns out to be considerably more efficient than 30 scans/channels, which only permits seven scans per disk block and results in 28 blank spaces. Some cases have been summarized in Table 6. The entry continues as follows:

21

ENTER CHANNEL/GAIN CODE FOR NUMBER OF CHANNELS SELECTED

STATUS=XXY; WHERE XX=OCTAL CHANNEL NO. Y=OCTAL GAIN CODE

The user must, therefore, enter in this example 24 codes, each of which designates a hardware channel and gain code. Gain codes are interpreted as shown in Table 7.
TABLE 6. Summary of Disk Space Utilization

<table>
<thead>
<tr>
<th>Channels/Scan</th>
<th>Data Space Available</th>
<th>Scans/Block</th>
<th>Data Words/Block</th>
<th>Unused Space, Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>248</td>
<td>24</td>
<td>240</td>
<td>8</td>
</tr>
<tr>
<td>20</td>
<td>243</td>
<td>12</td>
<td>240</td>
<td>3</td>
</tr>
<tr>
<td>21</td>
<td>242</td>
<td>11</td>
<td>231</td>
<td>11</td>
</tr>
<tr>
<td>22</td>
<td>242</td>
<td>11</td>
<td>242</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>241</td>
<td>10</td>
<td>230</td>
<td>11</td>
</tr>
<tr>
<td>24</td>
<td>241</td>
<td>10</td>
<td>240</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>240</td>
<td>9</td>
<td>225</td>
<td>15</td>
</tr>
<tr>
<td>26</td>
<td>240</td>
<td>9</td>
<td>234</td>
<td>6</td>
</tr>
<tr>
<td>27</td>
<td>239</td>
<td>8</td>
<td>216</td>
<td>23</td>
</tr>
<tr>
<td>28</td>
<td>239</td>
<td>8</td>
<td>224</td>
<td>15</td>
</tr>
<tr>
<td>29</td>
<td>238</td>
<td>8</td>
<td>232</td>
<td>6</td>
</tr>
<tr>
<td>30</td>
<td>238</td>
<td>7</td>
<td>210</td>
<td>28</td>
</tr>
</tbody>
</table>

TABLE 7. Gain Code Interpretation

<table>
<thead>
<tr>
<th>Gain Code</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>32</td>
</tr>
</tbody>
</table>
The octal channel number is simply the channel number expressed in octal where the first channel is number 0 and the last channel is number 27_8 (23_{10}).

As a typical entry:

```
0[CR]
60[CR]
140[CR]
150[CR]
220[CR]
230[CR]
240[CR]
160[CR]
170[CR]
270[CR]
10[CR]
20[CR]
30[CR]
40[CR]
50[CR]
100[CR]
200[CR]
210[CR]
250[CR]
260[CR]
0[CR]
```

ENTER 24 ADC FIXED GAIN VALUES

The order of hardware channels is immaterial and gain codes are selected at will. In this example, channel 0 was sampled twice at a gain of one each time. In this example, the programmable gain of each channel was set to 1 since auto gain is to be selected.

Next, the fixed gains are entered for each of the 24 hardware channels as follows:
ENTER 20 ALARM VALUES (HI/LO) FOR FIRST 10 CHANNELS SCANNED

Fixed gain values may be verified by sliding out each amplifier board approximately 4 in. and reading the gain value on each plug-in gain module (a special key hangs on the inside of ADC door for withdrawing boards). The values given above correspond to the installed configuration at delivery.

The amplifier boards are of two types: high level (Model 620060) and low level (Model 620050). High-level boards use fixed gains in the range of 1-10 and low-level boards use fixed gains in the range of 10-1000 (see page 9-10 under General Description in the NEFF Manual for further details)(4)
Also, the installed filter cutoff is 100 Hz on all 24 channels as delivered. This value can be changed with a suitable plug-in resister module as was the case with the fixed gain values. Next, the alarm values are entered in millivolts, where the first value is the upper limit and the second value for each channel is the lower limit as follows:

9500
-9500
-1000
-3000
25
15
30
10
10
5
10
5
10
5
15
10
0
10
2
0
-2

TYPE 1 FOR AUTO GAIN

The 9500/-9500 entry could serve to alarm if a channel went full scale. An 11000/-11000 entry would effectively remove an alarm from service since the maximum output the ADC can generate is 10000/-10000. Also, negative signals can be treated as evidenced by the entry, -1000/-3000. Only integers in the range of ±32000 are legal.

Experience will largely govern the choice of the auto gain option. It will depend largely on whether the user or computer (as programmed) is judged to
make the better choice of programmable gain setting. As programmed, the computer changes gain settings so as to maintain the amplified analog value in the range of 3.5 to 7.0 volts or -3.5 to -7.0 volts. These values were selected as a compromise between good resolution and freedom from saturation during transients. The auto gain option is selected as follows:

1[CR]

ENTER NUMBER OF DATA POINTS TO BE AVERAGED: 200-32000

Next, the number of averages made is entered as follows:

600[CR]

6.00000E 01 = SEC PER DISK WRITE
291 = NUMBER OF FREE BLOCKS ON DISK
7/19/79  14:35:23

Here, 600 was calculated to give 1 min between disk writes or 6 sec per averaged word. For comparison, an entry of 6000 would have given 1 min averaged data values which would result in 10 min per disk write and allow a test period of 80 hr on a blank disk with 484 blocks. Next, the free space is indicated, which is two blocks less than the previous value since DAC04 wrote a DATA.INT and DATA.TX file, each having one block. This, therefore, completes the entry. The user should observe that the ADC data and channel lights are flashing as an indication of data conversion. Bit 1 will alarm on Digital Output Panel 1 to indicate that drive 1 is in operation (bit 0 indicates drive 0). Just after the first write to disk, the data will be checked against the alarm limits input. Alarms are displayed on bits 0 to 9 on Digital Output Panel 2 where bit 0 indicates scan 0, etc.

Certain codes are available to communicate with DAC04 from the keyboard and the digital input panel. Of course CTRL/F must first be struck if previous communications were with the background. The codes shown in Table 8 are available.

An example of the data table is shown in Table 9. The number (000201) below the table is the octal representation of the 16-bit digital input word. Bit 7 is set since the table is desired. The value of the word in bits 0-5 is 1(one); therefore, the number of words averaged has not been altered from the initialized value.
TABLE 8. DAC04 Input Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7 Set&lt;sup&gt;(a)&lt;/sup&gt;</td>
<td>Requests a table of scan number, channel number, fixed gain (FX), programmable gain (PR). Data is displayed in millivolts and automatically updated for each channel scanned. The heading of the table displays the current time and date along with the disk status. The last disk block written is given (note that the number of blocks written is one greater since the first block is zero) along with the number of remaining blocks. Therefore, the sum of both numbers plus one gives the total space available for data storage.</td>
</tr>
<tr>
<td>Bit 7 Cleared&lt;sup&gt;(a)&lt;/sup&gt;</td>
<td>Cancels table requests.</td>
</tr>
<tr>
<td>@@@</td>
<td>Terminates program after closing current file (i.e., making file permanent and updating directory accordingly).</td>
</tr>
<tr>
<td>CTRL/C&lt;sup&gt;(b)&lt;/sup&gt;</td>
<td>Terminates program without closing current file (i.e., file name does not appear on directory). This code is part of RT-11 monitor.</td>
</tr>
<tr>
<td>BREAK</td>
<td>Electrically disabled within terminal. Otherwise would halt computer (Enable/Halt switch on computer provides the same function). The risk of accidentally striking this key seemed to outweigh its convenience.</td>
</tr>
<tr>
<td>Bit 6 Set&lt;sup&gt;(a)&lt;/sup&gt;</td>
<td>Causes number of words averaged to be divided by word in bits 0-5.</td>
</tr>
<tr>
<td>Bit 6 Cleared&lt;sup&gt;(a)&lt;/sup&gt;</td>
<td>Causes number of words averaged to be multiplied by word in bits 0-5.</td>
</tr>
<tr>
<td>Bits 0-5&lt;sup&gt;(a)&lt;/sup&gt;</td>
<td>Value of word set determines factor by which number of words averaged is either increased or decreased. Switch up indicates one, zero, otherwise.</td>
</tr>
</tbody>
</table>

<sup>(a)</sup> Refers to digital input. Bits set with switch up, cleared otherwise.  
<sup>(b)</sup> CTRL/C is entered by simultaneously striking the CTRL and C keys.
<table>
<thead>
<tr>
<th>Scan</th>
<th>Channel</th>
<th>X Data (MV)</th>
<th>Scan</th>
<th>Channel</th>
<th>X Data (MV)</th>
<th>Scan</th>
<th>Channel</th>
<th>X Data (MV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1/16</td>
<td>1</td>
<td>6</td>
<td>2/2</td>
<td>9</td>
<td>15</td>
<td>2/16</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>200/32</td>
<td>3</td>
<td>13</td>
<td>200/4</td>
<td>10</td>
<td>1</td>
<td>1/32</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>500/4</td>
<td>5</td>
<td>19</td>
<td>500/8</td>
<td>12</td>
<td>3</td>
<td>1/1</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>500/4</td>
<td>7</td>
<td>14</td>
<td>200/32</td>
<td>14</td>
<td>5</td>
<td>2/1</td>
</tr>
<tr>
<td>9</td>
<td>15</td>
<td>200/16</td>
<td>9</td>
<td>23</td>
<td>500/4</td>
<td>16</td>
<td>16</td>
<td>500/4</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1/32</td>
<td>11</td>
<td>2</td>
<td>1/1</td>
<td>18</td>
<td>21</td>
<td>500/8</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>1/1</td>
<td>13</td>
<td>4</td>
<td>2/2</td>
<td>20</td>
<td>0</td>
<td>1/16</td>
</tr>
</tbody>
</table>

5.2 GRAFTR OPERATION

GRAFTR operation is similar in most respects to that of GRAFAV. Of course, DAC04 must be running concurrently since it supplies the data for graphic display. Assuming GRAFTR.SAV is resident on drive 1 and DAC04 is operating, simply type:

CTRL/B
.RU GRAFTR[CR]

ENTER CLOCK INTERRUPT PERIOD (MS)

The clock interrupt period is only required to enable the program to label the X-axis in seconds. Enter a [CR] if this option is not desired. The entry continues as follows:

10[CR]

ENTER CHANNEL NO., SPAN (MV), OFFSET (MV), AND FIXED GAIN FOR GRAPH 0

0[CR]
10000[CR]
0[CR]
1[CR]

ENTER CHANNEL NO., SPAN (MV), OFFSET (MV), AND FIXED GAIN FOR GRAPH 1

3[CR]
TYPE: 1 TO DELETE GRIDS

where the desired scan numbers were entered. All spans and offsets are in millivolts.

The last message above refers to the optional grids superimposed upon the graph. Preference will dictate the selection. The entry continues as:

TYPE: "SPACE, TITLE OF GRAPH"

where the user enters a space by striking the space bar before entering an optional title. A [CR] will delete the title option. Entry continues as:

 TEST GRAPH OF TRANSIENT DATA[CR]

TYPE: SPACE, LABEL FOR X-AXIS"

TIME[CR]

TYPE: LABEL FOR Y-AXIS"

AMPLITUDE IN MV[CR]

* 

where a space is not required before entering the Y-axis label due to the method of programming. The * implies, "Type 1 to continue." A [CR] will terminate GRAFTR. Vertical bars resembling error bars denote graph 1. A "+" symbol following the graph 1 label serves as a reminder. Current date and time are displayed. Should a 1 be struck the code continues as follows:

1[CR]

TYPE 1 TO OVERRIDE GRAPHIC RESET

The user can type a 1 to preserve the graphic display while entering input for the next graph. From this point on the input is the same as given previously. A typical example of the graph produced appeared previously in Figure 1.
5.3 GRAFAV OPERATION

The operation of all background programs is largely self-explanatory. Assuming, GRAFAV.SAV is resident on drive 1, simply type:

CTRL/B
.RU GRAFAV[CR]

ENTER 1 TO GRAPH ACTIVE DATA FILE

The CTRL/B is necessary if DAC04 is running concurrently in foreground and if foreground had been receiving messages from the keyboard. The user need not specify the device since DX1 is assumed to be the default device. The program responds by asking if an active data file is desired (i.e., the data being generated concurrently by DAC04). Input proceeds as follows for a permanent file:

[CR]

ENTER PERMANENT FILE NAME

*DATA.00A

ENTER STARTING BLOCK NUMBER

0[CR]

ENTER NUMBER OF CHANNELS/SCAN

24[CR]

16 24 1 51

ENTER SCAN NUMBERS FOR GRAPHS 0 AND 1

0[CR]

3[CR]

ENTER SPAN, OFFSET, AND FIXED GAIN FOR GRAPH 0

20000[CR]

-10000[CR]

1[CR]

ENTER SPAN, OFFSET, AND FIXED GAIN FOR GRAPH 1
20000[CR]  
-10000[CR]  
1[CR]  
TYPE 1 TO DELETE GRIDS  
1[CR]  
TYPE: "SPACE, TITLE OF GRAPH"  
TEST GRAPH OF DATA.ØA[CR]  
TYPE: SPACE, LABEL FOR X-AXIS"  
TIME[CR]  
TYPE: "LABEL FOR Y-AXIS"  
AMPLITUDE IN MV[CR]  
*  
The number of channels/scan corresponds to the initialization value used for DACØ4. The scan numbers refer to the order in which the channels were scanned as opposed to the hardware channel number. Both scan and offset are in millivolts. Once the graph is displayed the "*" implies: "Type 1 to Continue." For example:  
1[CR]  
ENTER 1 TO GRAPH ACTIVE DATA FILE  
A typical example of the graph produced appeared previously in Figure 2.  

5.4 TYPAV OPERATION  
Assuming TYPAV.SAV is resident on drive 1 it may be run by simply typing:  
CTRL/B  
.RU TYPAV[CR]  
TYPE 1 TO ACCESS ACTIVE FILE  
[CR]  
ENTER PERMANENT FILE NAME; DEFAULT EXT = ØØB
DATA

ENTER STARTING BLOCK NUMBER

20

ENTER NUMBER OF CHANNELS/SCAN

21

ENTER 1 TO TYPE FIRST BLOCK SPECIFIED

1

where the data output appears as shown in Table 10.

ENTER 1 TO TYPE 2 CHANNELS IN MILLIVOLTS

It is only possible to access an active data file if DAC04 is running concurrently. If an active file is not desired, simply strike a [CR] and enter the file name. If the file is on drive 0, it must be specified (e.g., DX:DATA.00B). If the extension is not specified, 00A is assumed. Therefore, the entry, DATA, implies DX1:DATA.00A.

TABLE 10. Disk Block Data Produced by TYPHAV

<table>
<thead>
<tr>
<th>15647</th>
<th>107</th>
<th>12427</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>60</td>
<td>140</td>
</tr>
<tr>
<td>5</td>
<td>43</td>
<td>50</td>
</tr>
<tr>
<td>30</td>
<td>50</td>
<td>200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>200</th>
<th>160</th>
<th>174</th>
<th>275</th>
<th>17</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3955</td>
<td>-3584</td>
<td>2182</td>
<td>4613</td>
<td>3287</td>
<td>3450</td>
</tr>
<tr>
<td>5191</td>
<td>3480</td>
<td>7008</td>
<td>-8192</td>
<td>3464</td>
<td>3401</td>
</tr>
<tr>
<td>4610</td>
<td>3290</td>
<td>3103</td>
<td>3450</td>
<td>3367</td>
<td>3113</td>
</tr>
<tr>
<td>-8192</td>
<td>3465</td>
<td>3492</td>
<td>3360</td>
<td>-8192</td>
<td>-3843</td>
</tr>
<tr>
<td>3456</td>
<td>3375</td>
<td>-3917</td>
<td>-4816</td>
<td>-2489</td>
<td>6194</td>
</tr>
<tr>
<td>3369</td>
<td>-8192</td>
<td>-3840</td>
<td>-3833</td>
<td>-3584</td>
<td>3184</td>
</tr>
<tr>
<td>-4815</td>
<td>-2542</td>
<td>6194</td>
<td>5192</td>
<td>3479</td>
<td>7005</td>
</tr>
<tr>
<td>-3831</td>
<td>-3583</td>
<td>3182</td>
<td>4606</td>
<td>3287</td>
<td>3103</td>
</tr>
<tr>
<td>5191</td>
<td>3479</td>
<td>7005</td>
<td>-8192</td>
<td>3464</td>
<td>3401</td>
</tr>
<tr>
<td>4601</td>
<td>3289</td>
<td>3102</td>
<td>3451</td>
<td>3375</td>
<td>3108</td>
</tr>
<tr>
<td>-8192</td>
<td>3466</td>
<td>3492</td>
<td>3360</td>
<td>-8192</td>
<td>-3835</td>
</tr>
<tr>
<td>3451</td>
<td>3366</td>
<td>3089</td>
<td>4827</td>
<td>2535</td>
<td>6195</td>
</tr>
<tr>
<td>3366</td>
<td>-8192</td>
<td>3833</td>
<td>-3844</td>
<td>-3584</td>
<td>3183</td>
</tr>
<tr>
<td>-4822</td>
<td>-2523</td>
<td>6195</td>
<td>5191</td>
<td>3490</td>
<td>7004</td>
</tr>
<tr>
<td>-3947</td>
<td>-3583</td>
<td>3182</td>
<td>4620</td>
<td>3285</td>
<td>3102</td>
</tr>
<tr>
<td>5191</td>
<td>3460</td>
<td>7004</td>
<td>-8192</td>
<td>3463</td>
<td>3398</td>
</tr>
<tr>
<td>4622</td>
<td>3299</td>
<td>3103</td>
<td>3449</td>
<td>3379</td>
<td>3110</td>
</tr>
<tr>
<td>-8192</td>
<td>3464</td>
<td>3490</td>
<td>3358</td>
<td>-8192</td>
<td>-3855</td>
</tr>
<tr>
<td>3466</td>
<td>3366</td>
<td>3184</td>
<td>-4846</td>
<td>-2448</td>
<td>6194</td>
</tr>
<tr>
<td>3360</td>
<td>-8192</td>
<td>-3849</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
As before, the number of channels/scan must agree with the value used in DAC04. A 1 is entered if the entire first block specified is desired in binary (i.e., the decimal equivalent of the binary number stored on disk). The first line gives the date (1 word) and time (2 words), respectively. The second and third lines are the channel/gain status codes in octal. (Note: gain code can change from initial value if auto gain option is selected in DAC04.) The remaining lines give the data. The values 8191 and -8192 represent +10 volts and -10 volts, respectively. With 21 channels/scan there is eleven blank spaces per disk block, which explains the zeroes on the last two lines. The same kind of data without any formatting could have been obtained by typing:

DUMP/TERM DATA.00A[CR]

without using TYPAV. The major advantage of using TYPAV to read disk blocks is the formatting of each channel/gain status word into bytes for ease of understanding. To continue with the entry:

1[CR]
ENTER TWO CHANNELS
0[CR]
5[CR]
ENTER FIXED GAIN FOR FIRST CHANNEL SPECIFIED
1[CR]
ENTER FIXED GAIN FOR SECOND CHANNEL SPECIFIED
500[CR]

where Table 11 shows a typical output.

TYPE 1 TO CONTINUE

The date code and start time is reported from the first block specified. Both data in millivolts and programmable gain (PRGN) are given for each channel selected. In addition, each data value is numbered consecutively and the disk block from which it was read is displayed. The headings DATA0, PRGN0, DATA1, and PRGN1 correspond to the symbols used in both GRAFAV and TYPAV. It is worth noting that the value, -294.113 millivolts corresponds to the value -3855 in the
<table>
<thead>
<tr>
<th>NO.</th>
<th>DATA0</th>
<th>PRO01</th>
<th>DATA1</th>
<th>PRO11</th>
<th>BLOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-294.113</td>
<td>16.</td>
<td>7.573</td>
<td>1.</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>-293.274</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>-292.969</td>
<td>16.</td>
<td>7.581</td>
<td>1.</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>-292.435</td>
<td>16.</td>
<td>7.579</td>
<td>1.</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>-292.282</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>-292.587</td>
<td>16.</td>
<td>7.573</td>
<td>1.</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>-292.435</td>
<td>16.</td>
<td>7.581</td>
<td>1.</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>-293.427</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>-293.563</td>
<td>16.</td>
<td>7.573</td>
<td>1.</td>
<td>20</td>
</tr>
<tr>
<td>10</td>
<td>-294.037</td>
<td>16.</td>
<td>7.573</td>
<td>1.</td>
<td>20</td>
</tr>
<tr>
<td>11</td>
<td>-294.342</td>
<td>16.</td>
<td>7.581</td>
<td>1.</td>
<td>21</td>
</tr>
<tr>
<td>12</td>
<td>-295.029</td>
<td>16.</td>
<td>7.579</td>
<td>1.</td>
<td>21</td>
</tr>
<tr>
<td>13</td>
<td>-294.647</td>
<td>16.</td>
<td>7.578</td>
<td>1.</td>
<td>21</td>
</tr>
<tr>
<td>14</td>
<td>-295.105</td>
<td>16.</td>
<td>7.573</td>
<td>1.</td>
<td>21</td>
</tr>
<tr>
<td>15</td>
<td>-294.876</td>
<td>16.</td>
<td>7.578</td>
<td>1.</td>
<td>21</td>
</tr>
<tr>
<td>16</td>
<td>-294.571</td>
<td>16.</td>
<td>7.573</td>
<td>1.</td>
<td>21</td>
</tr>
<tr>
<td>17</td>
<td>-294.800</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>21</td>
</tr>
<tr>
<td>18</td>
<td>-295.105</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>21</td>
</tr>
<tr>
<td>19</td>
<td>-294.495</td>
<td>16.</td>
<td>7.573</td>
<td>1.</td>
<td>21</td>
</tr>
<tr>
<td>20</td>
<td>-294.571</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>22</td>
</tr>
<tr>
<td>21</td>
<td>-294.571</td>
<td>16.</td>
<td>7.573</td>
<td>1.</td>
<td>22</td>
</tr>
<tr>
<td>22</td>
<td>-294.113</td>
<td>16.</td>
<td>7.578</td>
<td>1.</td>
<td>22</td>
</tr>
<tr>
<td>23</td>
<td>-294.266</td>
<td>16.</td>
<td>7.581</td>
<td>1.</td>
<td>22</td>
</tr>
<tr>
<td>24</td>
<td>-294.418</td>
<td>16.</td>
<td>7.571</td>
<td>1.</td>
<td>22</td>
</tr>
<tr>
<td>25</td>
<td>-294.113</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>22</td>
</tr>
<tr>
<td>26</td>
<td>-294.113</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>22</td>
</tr>
<tr>
<td>27</td>
<td>-293.579</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>22</td>
</tr>
<tr>
<td>28</td>
<td>-293.350</td>
<td>16.</td>
<td>7.585</td>
<td>1.</td>
<td>23</td>
</tr>
<tr>
<td>29</td>
<td>-292.358</td>
<td>16.</td>
<td>7.581</td>
<td>1.</td>
<td>23</td>
</tr>
<tr>
<td>30</td>
<td>-293.198</td>
<td>16.</td>
<td>7.573</td>
<td>1.</td>
<td>23</td>
</tr>
<tr>
<td>31</td>
<td>-293.405</td>
<td>16.</td>
<td>7.578</td>
<td>1.</td>
<td>23</td>
</tr>
<tr>
<td>32</td>
<td>-293.503</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>23</td>
</tr>
<tr>
<td>33</td>
<td>-293.198</td>
<td>16.</td>
<td>7.571</td>
<td>1.</td>
<td>23</td>
</tr>
<tr>
<td>34</td>
<td>-292.892</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>23</td>
</tr>
<tr>
<td>35</td>
<td>-292.587</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>23</td>
</tr>
<tr>
<td>36</td>
<td>-292.282</td>
<td>16.</td>
<td>7.571</td>
<td>1.</td>
<td>23</td>
</tr>
<tr>
<td>37</td>
<td>-291.824</td>
<td>16.</td>
<td>7.578</td>
<td>1.</td>
<td>23</td>
</tr>
<tr>
<td>38</td>
<td>-291.901</td>
<td>16.</td>
<td>7.578</td>
<td>1.</td>
<td>23</td>
</tr>
<tr>
<td>39</td>
<td>-291.749</td>
<td>16.</td>
<td>7.578</td>
<td>1.</td>
<td>24</td>
</tr>
<tr>
<td>40</td>
<td>-291.519</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>24</td>
</tr>
<tr>
<td>41</td>
<td>-291.519</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>24</td>
</tr>
<tr>
<td>42</td>
<td>-291.901</td>
<td>16.</td>
<td>7.578</td>
<td>1.</td>
<td>24</td>
</tr>
<tr>
<td>43</td>
<td>-291.901</td>
<td>16.</td>
<td>7.578</td>
<td>1.</td>
<td>24</td>
</tr>
<tr>
<td>44</td>
<td>-291.977</td>
<td>16.</td>
<td>7.578</td>
<td>1.</td>
<td>24</td>
</tr>
<tr>
<td>45</td>
<td>-292.969</td>
<td>16.</td>
<td>7.581</td>
<td>1.</td>
<td>24</td>
</tr>
<tr>
<td>46</td>
<td>-293.809</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>24</td>
</tr>
<tr>
<td>47</td>
<td>-293.029</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>24</td>
</tr>
<tr>
<td>48</td>
<td>-294.724</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>24</td>
</tr>
<tr>
<td>49</td>
<td>-294.418</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>24</td>
</tr>
<tr>
<td>50</td>
<td>-294.495</td>
<td>16.</td>
<td>7.571</td>
<td>1.</td>
<td>25</td>
</tr>
<tr>
<td>51</td>
<td>-293.864</td>
<td>16.</td>
<td>7.581</td>
<td>1.</td>
<td>25</td>
</tr>
<tr>
<td>52</td>
<td>-293.579</td>
<td>16.</td>
<td>7.573</td>
<td>1.</td>
<td>25</td>
</tr>
<tr>
<td>53</td>
<td>-293.045</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>25</td>
</tr>
<tr>
<td>54</td>
<td>-292.372</td>
<td>16.</td>
<td>7.576</td>
<td>1.</td>
<td>25</td>
</tr>
</tbody>
</table>
disk block data above [i.e., -3855 x 10000/(8192x16)]. Likewise, the value, 7.573 corresponds to the 3102 (sixth entry from left in first data line) in the disk block data above. (Note that scan 0 and 5 were selected, which corresponds to the first and sixth data entry.) As is customary, the final message allows the routine to be continued without restarting it.

5.5 TYPLOG OPERATION

The program TYPLOG is a short FORTRAN program designed to run in background. A sample entry is given below assuming TYPLOG.SAV is resident on drive 1.

CTRL/B
.RU TYPLOG [CR]
Enter file name
*LOG01

Enter 6 lines of text data

A TANK SPARK OCCURRED AT [CR]
1653 WITH A RESULTING TEMPERATURE [CR]
TRANSIENT TO 325 C....[CR]
RUN 24-3-5/14-JUN-79...[CR]
[CR]
[CR]
Type 1 to type more data
[CR]
Stop--

Device DX1 is implied and .TXT is the default extension (i.e., if not specified). Six lines of text data must be entered where the appropriate number of carriage returns may be used to complete the balance as shown above. Again, as shown, the user is given the option of entering more text data. Ordinarily the file name, DATA, should not be used unless it is intended to delete DATA.TXT created during initialization of DAC04. It is recommended that text data be written to the nonactive disk (i.e., the disk not in current use by
DAC04). Should the user attempt to create a text file on a properly squeezed active disk, the error message: "-2 = IENTER ERROR CODE," will be displayed, which means that the program was unable to find space to create a file. This happens because DAC04 reserves the largest contiguous block of space on the active disk for itself.

5.6 RETXT OPERATION

The program RETXT is a short FORTRAN program run in background. It is a general purpose routine designed to read ASCII text data. It is coded, however, to read only block 0 of the specified file. Nevertheless, the code could be generalized in the future to read text data of any length. A typical example is given below:

CTRL/B
. RU RETXT[CR]
ENTER FILE NAME
*LOG01[CR]
A TANK SPARK OCCURRED AT
1653 WITH A RESULTING TEMPERATURE
TRANSIENT TO 325 C....
RUN 24-3-5/14-JUN-79...
TYPE 1 TO READ MORE TEXT DATA
where the extension is assumed to be .TXT unless otherwise specified.

5.7 REINT3 OPERATION

REINT3 is a FORTRAN program designed specifically to read and display the initialization data file, DATA.INT, written by DAC04. An example of its operation is given below:

CTRL/B
. RU REINT3[CR]
ENTER FILE NAME: DEFAULT EXT = INT
*DATA[CR]
Table 12 shows the typical output table which immediately follows the last entry above. (See Table 2 for a description of each entry in the table.) Since DAC04 creates the file DATA.INT by simply writing a 256-word section of memory to disk, the user should exercise caution in modifying the data storage section in DAC04 between the variables SPAVAL and CHNSTA. Any changes in this section of MACRO code will likely create formatting problems in REINT3 and prevent the automatic restart feature of DAC04 from functioning properly.

5.8 RESET OPERATION

Reset is a very short program which uses the PLOT55 routine to clear the graphic display. Simply type:

CTRL/B
.RU RESET[CR]
STOP--

The program automatically terminates and control returns to the RT-ll monitor.

**TABLE 12. Initialization Data Produced by REINT3**

<table>
<thead>
<tr>
<th>START DATE=7/24/79</th>
<th>TIME OF DAY=14:00:34</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBCH 21 TRNDAT</td>
<td>10 NUMAVG 600 REPAVG</td>
</tr>
<tr>
<td>TIM 0</td>
<td>144 AUTOGN 1 CLOCK</td>
</tr>
<tr>
<td>OUT 167772 INP 167774 CSR 170420 BPR 170422 DIGOUT1 177550</td>
<td></td>
</tr>
<tr>
<td>BEOUT2 177552 DIGINP 177602 DATE 17407 TOD 56 TOD1 26170</td>
<td></td>
</tr>
<tr>
<td>FIXED GAIN VALUES:</td>
<td></td>
</tr>
<tr>
<td>1 2 3 4 5 6 7 8 9 10 11</td>
<td></td>
</tr>
<tr>
<td>1 2 3 4 5 6 7 8 9 10 11</td>
<td></td>
</tr>
<tr>
<td>12 13 14 15 16 17 18 19 20 21 22 23</td>
<td></td>
</tr>
<tr>
<td>200 200 200 200 500 500 500 500 500 500 500 500</td>
<td></td>
</tr>
<tr>
<td>ALARM SET POINTS (HIGH/LOW):</td>
<td></td>
</tr>
<tr>
<td>0 1 9500 -9500 -1000 -3000 2 25 15 30 10 10 5</td>
<td></td>
</tr>
<tr>
<td>5 10 5 10 5 10 0 10 2 0 -2</td>
<td></td>
</tr>
<tr>
<td>CHNSTA(OCIAL):</td>
<td></td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14</td>
<td></td>
</tr>
<tr>
<td>0 60 140 150 220 230 240 160 170 270 10 20 30 40 50</td>
<td></td>
</tr>
<tr>
<td>15 16 17 18 19 20 21 22 23 24 25 26 27 28 29</td>
<td></td>
</tr>
<tr>
<td>100 200 210 250 260 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>TYPE 1 TO READ MORE INITIALIZATION DATA</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX A

COMPUTER HARDWARE DESCRIPTION
APPENDIX A

COMPUTER HARDWARE DESCRIPTION

A.1 COMPUTER CONFIGURATION

The computer was originally purchased in a stand-alone configuration from Digital Equipment Corporation (DEC) through an OEM supplier, First Computer Corporation. It was purchased as a package, designated PDP11V/03, in a self-contained, roll-around cabinet. However, several changes were stipulated in the purchase order that caused the package delivered to be somewhat nonstandard. The system as delivered included the components shown in Table A.1.

<table>
<thead>
<tr>
<th>Model No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KD11-HA</td>
<td>CPU</td>
</tr>
<tr>
<td>MSV11-DD</td>
<td>16K words memory</td>
</tr>
<tr>
<td>BA11-ME</td>
<td>Expander box</td>
</tr>
<tr>
<td>H984</td>
<td>Roll-around cabinet</td>
</tr>
<tr>
<td>BCV1B-06</td>
<td>Expansion cable with interface modules</td>
</tr>
<tr>
<td>KWV11-A</td>
<td>Programmable real-time clock</td>
</tr>
<tr>
<td>REV11-A</td>
<td>Bootstrap module</td>
</tr>
<tr>
<td>RXV11-BA</td>
<td>Dual floppy disk and interface</td>
</tr>
<tr>
<td>DRV11</td>
<td>Parallel line interface</td>
</tr>
<tr>
<td>VT55-FA</td>
<td>Graphics terminal with hardcopy</td>
</tr>
<tr>
<td>DLV11</td>
<td>Serial interface</td>
</tr>
<tr>
<td>KEV11</td>
<td>EIS/FIS Micron Chip</td>
</tr>
</tbody>
</table>

Peripherals were subsequently interfaced to the computer in order to configure it as a self-contained Data Acquisition System. These peripherals are listed in Table A.2.
TABLE A.2. Additional Components Added to PDP11V/03 Computer

<table>
<thead>
<tr>
<th>Model No.</th>
<th>Mfg</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>620/100</td>
<td>NEFF</td>
<td>24-channel analog-to-digital converter</td>
</tr>
<tr>
<td>PM-SV32A</td>
<td>Plessey</td>
<td>16 K words memory</td>
</tr>
<tr>
<td>1412DA-4</td>
<td>ADAC</td>
<td>4-channel digital-to-analog converter</td>
</tr>
<tr>
<td>1632HCO</td>
<td>ADAC</td>
<td>32-bit digital output interface</td>
</tr>
<tr>
<td>1616CCI</td>
<td>ADAC</td>
<td>16-bit digital input interface</td>
</tr>
<tr>
<td>DLV11-J</td>
<td>DEC</td>
<td>4-channel serial interface</td>
</tr>
<tr>
<td>BN-52A-7F</td>
<td>DEC</td>
<td>20 mA to EIA conversion kit</td>
</tr>
<tr>
<td>BC 21B-05</td>
<td>DEC</td>
<td>Cable</td>
</tr>
<tr>
<td>BC 01V-125</td>
<td>DEC</td>
<td>Cable</td>
</tr>
<tr>
<td>BC 20N-05</td>
<td>DEC</td>
<td>Cable</td>
</tr>
</tbody>
</table>

These additional components permitted the acquisition of analog and digital input, output of analog and digital signals, expansion of memory to the maximum allowable (i.e., 29K words), and the net addition of three serial interface channels for communication with existing peripherals and/or computers. The analog-to-digital converter can be expanded up to 64 analog input channels within the present A/D chassis or to a total of 256 channels by the addition of three satellite chassis (i.e., 64 channels/chassis). Thus, considerable flexibility is obtained at a relatively low incremental cost. Likewise, additional peripherals can be added to the computer by using the necessary interface cards. The principal limitation is the number of available slots within the computer backplane. The backplane is designed to accept circuit boards having either two or four edge connectors. A board with two edge connectors is called a double-width board and measures approximately 132 by 228 mm (5.2 by 9 in.). Likewise, a board with four edge connectors is called a quad-width board and measures approximately 267 by 228 mm (10.5 by 9 in.). Figure A.1 shows the schematic backplane as delivered to the University of Washington. As indicated, the programmable clock is the only board requiring a full quad-width slot. Also shown in Figure A.1 are the two double-width slots used in connecting the main backplane box to the expansion backplane and the two spare double-width slots available for future expansion. Three of the four serial interface channels are also available (i.e., the first channel is used as the VT55 terminal interface).

A-2
FIGURE A.1. Schematic Backplane Configuration
A.2 ANALOG-TO-DIGITAL CONVERTER SUBSYSTEM

The analog-to-digital (A/D) converter subsystem was procured from Neff Instrument Corporation. A summary of characteristics and specification is given in Table A.3.

<table>
<thead>
<tr>
<th>Model no.</th>
<th>620/100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum no. of channels</td>
<td>256</td>
</tr>
<tr>
<td>Low-level differential amplifiers</td>
<td>12</td>
</tr>
<tr>
<td>High-level differential amplifiers</td>
<td>12</td>
</tr>
<tr>
<td>Fixed gain range (low-level)</td>
<td>10-1000</td>
</tr>
<tr>
<td>Fixed gain range (high-level)</td>
<td>1-10</td>
</tr>
<tr>
<td>Options</td>
<td>8-channel output buffer, control panel</td>
</tr>
<tr>
<td>Programmable gains</td>
<td>1, 2, 4, 8, 16, 32</td>
</tr>
<tr>
<td>Modes</td>
<td>Random/sequential</td>
</tr>
<tr>
<td></td>
<td>remote/local</td>
</tr>
<tr>
<td>Interface</td>
<td>Programmed I/O</td>
</tr>
<tr>
<td></td>
<td>(DEC/DRV11)</td>
</tr>
<tr>
<td>ADC resolution (including sign)</td>
<td>14 bits</td>
</tr>
<tr>
<td>Throughput rate</td>
<td>20 kHz</td>
</tr>
<tr>
<td>Full-scale input</td>
<td>±10V</td>
</tr>
<tr>
<td>Overall system accuracy</td>
<td>±0.01%</td>
</tr>
</tbody>
</table>

Detailed specifications can be found in the Neff Operation and Maintenance Manual. The A/D Converter is of the amplifier-per-channel design with the inherent flexibility of changing fixed gain and filter cutoff for each channel. The system is delivered to the University of Washington with the settings indicated in Table A.4.

Both fixed gain and filter cutoff can be altered by changing plug-in modules on the appropriate printed circuit board. Spares have been procured and include twelve 10-Hz filters, two x5-gain modules, and two x100-gain modules plus an extender card kit for maintenance.
### TABLE A.4. Summary of Fixed Gains and Filter Cutoff Frequencies

<table>
<thead>
<tr>
<th>Channel</th>
<th>Type (a)</th>
<th>Card No.</th>
<th>Fixed Gain</th>
<th>Filter Cutoff, Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>H/L</td>
<td>0</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>H/L</td>
<td>0</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>H/L</td>
<td>0</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>H/L</td>
<td>0</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>H/L</td>
<td>1</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>H/L</td>
<td>1</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>H/L</td>
<td>1</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>7</td>
<td>H/L</td>
<td>1</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>8</td>
<td>H/L</td>
<td>2</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>9</td>
<td>H/L</td>
<td>2</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>H/L</td>
<td>2</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>11</td>
<td>H/L</td>
<td>2</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>12</td>
<td>L/L</td>
<td>3</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>13</td>
<td>L/L</td>
<td>3</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>14</td>
<td>L/L</td>
<td>3</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>15</td>
<td>L/L</td>
<td>3</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>16</td>
<td>L/L</td>
<td>4</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>17</td>
<td>L/L</td>
<td>4</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>18</td>
<td>L/L</td>
<td>4</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>19</td>
<td>L/L</td>
<td>4</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>20</td>
<td>L/L</td>
<td>5</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>21</td>
<td>L/L</td>
<td>5</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>22</td>
<td>L/L</td>
<td>5</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>23</td>
<td>L/L</td>
<td>5</td>
<td>500</td>
<td>100</td>
</tr>
</tbody>
</table>

(a) H/L = high level; L/L = low level

There are three locations on the I/O page of memory that are used by the DRV11 parallel interface for communications and control of the A/D converter. They are listed in Table A.5, as follows:
TABLE A.5. Analog-to-Digital Converter Addresses

<table>
<thead>
<tr>
<th>Address</th>
<th>Symbol</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>167770</td>
<td>DRCSR</td>
<td>DRVll Control Status Register where:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bit 0 = CSR0 (sequential/random)(a)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bit 1 = CSR1 (not used)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bit 5 = INT ENB B (not used)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bit 6 = INT ENB A (not used)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bit 7 = REQ A (ready/busy)(a)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bit 15 = REQ B (remote/local)(a)</td>
</tr>
<tr>
<td>167772</td>
<td>OUT</td>
<td>Output Data Register (or ADC input)</td>
</tr>
<tr>
<td>167774</td>
<td>INP</td>
<td>Input Data Register (or ADC output)</td>
</tr>
</tbody>
</table>

(a) The convention for describing the function of digital lines is as follows: high state/low state. For example, a high state on the sequential/random line selects sequential mode; a low state selects random mode. As with all TTL logic, an open line is held in the high state. Therefore, with the sequential/random line open, the system operates in sequential mode.(4)

As mentioned, the Neff analog-to-digital converter is interfaced to the computer via a DRVll general purpose 16-bit parallel interface. A schematic of the interface connection is given in Figure A.2.
Both the Neff manual\(^{(4)}\) and the DEC Memories and Peripherals Handbook\(^{(5)}\) should be consulted for more details. Since the computer is a 16-bit machine and the analog-to-digital converter is 14 bits, bit 13 (i.e., the sign bit) of the ADC data word has been connected to bits 13, 14, and 15 of the DRV11 data input word. This particular connection causes negative numbers generated by the analog-to-digital converter to be correctly interpreted by the computer in 2's complement format without the necessity of performing any special software bit manipulation.

A.3 DIGITAL OUTPUTS

An ADAC model 1632HCO 32-channel digital output card was procured to interface to the computer. These outputs, in principle, could be used to actuate the beam stop, helium circulating pump, alarms, etc. Initially, the interface has been used to operate two 16-channel alarm panels, each consisting of 16 LEDs and one alarm speaker. Each panel is equipped with a test switch to verify proper operation of the LEDs. At present, lights 0 and 1 of digital output panel No. 1 are used to signify operation of disk drive 0 and 1, respectively. This feature applies only to DAC04 operation, however. Also digital channels 0 through 9 of digital output panel No. 2 have been programmed to alarm when scans 0 through 9, respectively, reach either their high- or low-level alarm set point. The audio alarm is designed to actuate on any change in status of the LEDs (i.e., when a light is either turned on or off). A description of the digital output card is given in the ADAC manual.\(^{(6)}\)

A.4 DIGITAL INPUTS

An ADAC model 1616CCI 16-channel digital input card was procured to interface with the computer. In principle, it could be used to check the status of the beam stop, helium circulating pump, switch positions, etc. Initially, the interface has been connected to a 16-channel switch panel which allows the user to input and/or change parameters (e.g., during real-time data acquisition). The switch panel is mounted on a 48-cm (19-in.) relay chassis which also contains the analog input terminations for the ADC, the digital-to-analog output
terminations, and a series of eight banana plug jacks for the ADC output buffers. A description of the digital input card is given in the ADAC manual. Table 8 describes the present use of bits 0-7 of the digital input interface.

A.5 DIGITAL-TO-ANALOG CONVERTER

An ADAC model 1412DA-A four-channel digital-to-analog converter card was procured. The card is capable of either current or voltage proportioning control on each channel. The interface is capable of replacing up to four analog temperature controllers if desired. There is an incentive to replace the specimen temperature controller due to its response characteristics. The ultimate decision will probably be based upon the reliability of the computer and compromises associated with the additional CPU overhead incurred (i.e., the temperature of the specimen would most likely have to be controlled inside the clock interrupt service routine in order to improve the response behavior). Details on the operation of the digital-to-analog converter can be found in the manual supplied by ADAC Corporation.
APPENDIX B

COMPUTER SOURCE CODE LISTINGS
APPENDIX B
COMPUTER SOURCE CODE LISTINGS

B.1. DAC04 MAC LISTING

.TYPD DAC04.MAC

.MC ALL .EXIT, .PRINT, .TWAIT, .PROTECT, .DEVICE, .SDAT, .RCVDC
.MCALL .ENTER, .CLOSE, .WRITE, .QSET, .INTEN, .TTYIN, .WRITW
.MCALL .TTINR, .TTOUT, .STIN, .DATE, .TTYOUT, .READ, .LOOKUP
.GLOBAL .DIGIN, .DIGOUT, .FTOUT, .DFLOAT, .TIMOUT, .DOUT, .FLOAT

JSW=44
ERRBYT=52
.TITLE DAC04

; THIS PROGRAM IS CAPABLE OF SAMPLING UP TO 30 TIMES, 24 ADC CHANNELS UPON
; INTERRUPT FROM A PROGRAMmABLE CLOCK. CHANNEL NUMBER AND GAIN IS
; ENTERED FROM THE KEYBOARD. TWO BUFFERS ARE FILLED AND MAY BE
; READ FROM A BACKGROUND PROGRAM; A TRANSIENT BUFFER AND AN
; AVERAGED BUFFER. THE AVERAGED BUFFER IS WRITTEN TO DISK.
; AUTOMATIC GAIN CHANGING IS PERFORMED IF DESIRED. TIME OF DAY
; AND DATE IS STORED AS FIRST THREE WORDS ON EACH DISK BLOCK. A
; SEPARATE TEXT BUFFER AND INITIALIZATION FILE IS WRITTEN AUTOMATICALLY.
; THIS VERSION DISPLAYS DATA/GAIN SUMMARY ON SCREEN UPON REQUEST...
; THIS VERSION ADDS TABLE OF FIXED GAINS FOR STORAGE IN DATA.INT
; THIS VERSION ALLOWS OUT OF LIMIT ALARMS ON FIRST 10 CHANNELS
; THIS VERSION PROVIDES FOR AUTO RESTART USING DISK RESTART DATA
; BY P.L. HENDRICK, PACIFIC NORTHWEST LABORATORIES, VERSION OF 19-JUL-79

DAC03: TST @RCR

RDADC; REMOTE

.PRINT #MSG4

PRINT .EXIT

.REMOTE: .PRINT #MSG12

.DEVICE @AREA, @SAVE

.QSET @Q1, @6

.MOV @AREA, -(SP)

.MOV @440, R5

.PROTECT(SP), R5

.BCC NONERR

.BCC .PRINT #MSG7

.EXIT

.NONERR: MOV @CLKINT, (R5)+

.MOV @440, (R5)

.PRINT #MSG28

.JSR R5, DIGIN

.CHIP $1, R0

.BNE SKIP15

.JMP .RESTART

.SKIP15: .PRINT #MSG16

.JSR R5, DIGIN

.CHIP $320, R0

.BMI SKIP15

.MOV R0, CLOCK

.MUL $100, R0

.MOV R1, R0

.COM R0

.MOV R0, CLOCK1

;PUT ADC IN REMOTE

;DATE & TIME REMINDER

;ENTER CTRL/F

;ALLOCATE MORE QUE ELEMENTS

;PROTECT CLOCK INTERRUPT VECTORS

;PROTECT ERROR

;ESTABLISH INTERRUPT VECTORS

;RESTART OPTION SELECT

;ENTER CLOCK PERIOD(MS)

;STORE CLOCK IN MS

;SET CLOCK INTERRUPT PERIOD

DATE 19-77-79

CTRL/F
.PRINT  *MSG22  :READ IN DATA FILE EXTENSION
.TTYIN
SUB    #101, R0
ADD    R0, DX0B
ADD    R0, DX1B
MOV    RAVBUF1, AVPTR1
MOV    RAVBUF0, AVPTR0
MOV    #CHN1, ADDRESS
MOV    #512, R2
MOV    #TEXT, R1
.EXIT   #AREA, #1, #DX1C, #1
JSR    R5, DIGOUT
.PINT   *MSG3
BCC    TEXIT
.PINT   *MSG10
.EXIT

TEXINP:  .PRINT   *MSG1
.TTYIN
CMPB   R0, #100
BNE    SKIP2
MOV    R0, (R1)+
SDB    R2, TEXTIN
SKIP2:  .TTYIN
CMPB   R0, #12
BNE    SKIP2
MOV    R0, (R1)

SKIP3:  .WRITE
.PINT   *AREA, #1, #TEXT, #256, #0  :STORE TEXT DATA ON DISK #1
BCC    CLOSE
.PINT   *MSG11
.EXIT
CLOSE:  .CLOSE #1
.BCC    WXTIME
.PINT   *MSG10
.EXIT
WXTIME:  .PRINT     *MSG2
.JSR    R5, DIGIN
MOV    R0, NUMBCH
CMP    #30, R0
BMI    WXTIME
ASL    R0
SDB    R0, NOCHX4
MOV    NUMBCH, SPACE
BNE    SPACE, #1
BCC    SKIP#1
SDB    SPACE

SKIP#1:  ADD    #CHNSTA, SPACE
.PINT   *MSG5
.PINT   *MSG19
MOV    NUMBCH, R4
MOV    #CHCDD0, R1
MOV    #CHCDD1, R2
MOV    #CHNSTA, R3
RDCONS: JSR    R5, OINP
MOV    R0, (R1)+
MOV    R0, (R2)+
MOV    R0, (R3)+
SDB    R4, RDCONS
MOV    #FXGN, R2
MOV    #24, R1
.PINT   *MSG26
.FIXED:  JSR    R5, DIGIN
MOV    R0, (R2)+
SDB    R1, FIXED
.PINT   *MSG27
MOV    @ALARM2, STORE
MOV    @ALARMS, R2

B-2
ALARML: JSR R5, DIGIN
MOV R0, BSTORE
ADD $02, BSTORE
MUL $8192, R0
DIV $10000, R0
MOV R0, NUMBER
MOV #CHNSTA, R4
MOV CNTER, R5
ASR R5
ADD R5, R4
MOV (R4), R5
MOV R5, R4
BIC $177400, R4
ASR R4
ASR R4
ASR R5
ADD $FXGN, R4
MOV (R4), R1
MUL NUMBER, R1
MOV R1, (R2)+
INC CNTER
SUB R3, ALARM1
.PRINT @MSG14
JSR R5, DIGIN
MOV R0, AUTOGN

SKIP6: .PRINT @MSG6
JSR R5, DIGIN
MOV R0, NUMAVG
MOV R0, NUMAV1
CMP $32000, R0
BMI SKIP16
MOV NUMAV1, AVGCNT
MOV $512, R5
MOV @AVBUF0, R2
MOV @AVBUF1, R3

CLEAR1: CLR (R2)+
CLR (R3)+
SUB R5, CLEAR1
MOV NUMBCH, R2
ASR R2
BCC NOINC
INC R2

NOINC: MOV $253, R1
SUB R2, R1
CLR R0
DIV NUMBCH, R0
MOV R0, REP AVG
MOV R0, R1
MUL NUMBCH, R1
MOV R1, AVGCHZ
MOV REP AVG, AVBFCT
MOV CLOCK, R0
MUL REP AVG, R0
TST R0
BNE SKIP13
MOV R1, R0
MUL NUMAV1, R0
BIV $10000, R0
MOV R0, WRPER1
MOV @WRPER, R0
JSR R5, DFLOAT
JSR R5, FLTOUT
.PRINT @MSG17
JMP SKIP14

SKIP13: .PRINT @MSG23

 :READ IN ALARM SET POINTS

 :READ IN AUTO GAIN OPTION

 :READ IN NUMBER OF DATA POINTS AVERAGED

 :CLEAR AVERAGE BUFFERS

 :CALCULATE TIMES TO AVERAGE

 :R2=WAVES FOR CHANNEL/GAIN STATUS

 :FIND FREE SPACE FOR DATA

 :FIND TIMES TO LOAD AVERAGED DATA

 :FIND ACTUAL WORDS OF BINARY DATA

 :DISPLAY SEC/DISK WRITE
SKIP14: .DATE
    MOV R0, DATE
    TST R0
    BNE NODATE
    .PRINT @MSG20
    ; NO DATE ENTERED
    .EXIT

NODATE: .GETIM #AREA, #TOD
    .ENTER #AREA, #1, #DX1E, #1-1
    ; ESTABLISH I/O CHANNEL FOR DISK #1
    BCC SKIP8
    .PRINT @MSG10
    ; BAD ENTER
    .EXIT

SKIP8: .WRITE #AREA, #1, #SPAVAL, #256, #0
    ; STORE INITIALIZATION DATA
    BCC CLOSE1
    .PRINT @MSG11
    ; BAD WRITE
    .EXIT

CLOSE1: .CLOSE #1
    BIS $2, #DGOUT1
    .PRINT @MSG11
    ; CLOSE INITIALIZATION FILE
    .EXIT

RESTART: .LOOKUP @AREA, #1, #DX1E
    BCC READW
    .PRINT @MSG29
    .EXIT

READW: .READW @AREA, #1, #SPAVAL, #256, #0
    .PRINT @MSG22
    ; READ IN DATA FILE EXTENSION
    SUB $101, R0
    ADD R0, #DX0B
    ADD R0, #DX1B
    .CLOSE #1
    .EXIT

CONTINUE: .ENTER @AREA, #1, #DX1, #0-1
    MOV R0, #SPAVAL
    INC #DX0B
    INC #DX1B
    .JSR R5, #DIGOUT
    ; SHOW FREE DISK BLOCKS
    .PRINT @MSG3
    BCC SKIP9
    .PRINT @MSG10
    ; BAD ENTER
    .EXIT

:***********END OF INITIALIZATION:**********

:***********START OF DATA ACQUISITION:**********

SKIP9: .JSR R5, #TIMOUT
    ; DISPLAY CURRENT TIME AND DATE
    MOV #CLOCK1, #BP#R
    ; LOAD CLOCK INTERRUPT COUNT
    MTS
    MOV #123, #OCR
    JMP WRTC#K

SENDAT: CLR RECFL6
    CLR RECSTA
    MOV #CHN1, #BUF#R1
    MOV #CHN1, #BUF#R2
    ADD #EQ2, #BUF#R1
    ADD #EQ3, #BUF#R2
    TST AVGSM
    BST OVER
    MOV #CHC#D0, #GAIN
    BR OVER1

OVER: MOV #CHC#D1, #GAIN

OVER1: CLR R0
    MOV #EQ2, #R1
    DIV #1024, #R0
    MOV R0, #R2
    CLR R0
    MOV #EQ3, #R1
    DIV #1024, #R0
    MOV R0, #R3
    ADD #GAIN, #R2
    ADD #GAIN, #R3
    MOVB (R2), #R1

B-4
MOV B (R3), R4
BIC $177770, R1
BIC $177770, R4
MOV R1, GAIN3
MOV R4, GAIN4
SEND: .SDAT #AREA1, BUFFER1, #512.
#SDAT #AREA2, BUFFER2, #512.
#SDAT #AREA3, GAIN3, #2
BR WAI
RECEI: .ST TST RECFLG
BR SENDAT
TST RECSTA
BR WAI
INC RECSTA
.RECVD #AREA, #REQUEST, #4, #RECCRT:
#RECEIVE DATA
TST WRTFLG
BR RECEIVE
WRITE: CLR WRTFLG
TST AVGSW
BR OTHBUF
MOV #AVBUFO, BUFLOC
MOV #CHCDD0, R4
BR AVOID
OTHBUF: MOV #AVBUFK, BUFLOC
MOV #CHCDD1, R4
AVOID: MOV AVGSIZ, R5
MOV BUFLOC, R2
MOV SPACE, R3
MOV #DATE, DISPTR
AVERAGE: MOV (R2) +, R1
MOV (R2) +, R0
DIV NUMAV1, R0
MOV R0, (R3) +
SOB R5, AVERAGE
MOV R4, R0
MOV NUMBCH, R3
MOV #CHNSTA, R5
END
COPY: MOV B (R0) +, (R3) +
SOD R3, COPY
TST AUTOGN
BEQ MOGNCK
MOV NUMBCH, R5
MOV SPACE, R0
GAINCK: TST (R0)
BPL POSITY
CMP $2867, (R0)
:B -3.5 VOLTS
BMI INCGN
CMP $5734, (R0)
:B -7.0 VOLTS
BMI END
BR REDGN
POSITY: CMP $2867, (R0)
:B +3.5 VOLTS
BPL INCGN
CMP $5734, (R0)
:B +7.0 VOLTS
BPL END
REDGN: MOV B (R4), EXAM
BIC $177770, EXAM
CMP EXAM, #2
BLE END
DEC B (R4)
BR END
INC B (R4)
END ADD #2, R0

;DECREASE GAIN BY X2
;INCREASE GAIN BY X2
INC \( R^4 \)
NOENCK: SOB \( R^5 \), GAINCK
MOV \$512, R5
MOV BUFLOC, R2
CLEAR: CLR \( (R2)+ \)
SOB R5, CLEAR
BR WRTERR
NORBDY: .PRINT #MSG13
WRTST: IST WRTSTA
BGT NOTRBDY
.DATE
MOV R0, DATE
.GTIM @AREA, $TOD
INC WRTSTA
TST DISKSW
BNE SKIP5
SKIPS: .WRITE @AREA, #0, DISPTR, #256, #WRTCRT, BLKNUM
BCE SKIP4
JMP WRTERR
SKIP5: .WRITE @AREA, #1, DISPTR, #256, #WRTCRT, BLKNUM
BCE SKIP4
JMP WRTERR
SKIP4: .PRINT @MSG15
TSTB @ERRBYT
BED SWDISK
.PRT @MSG11
.EXIT
SWDISK: .CLOSE DISKSW
.Exit
TST BLKNUM
BNE SKIP12
DEC DX0B
DEC DX1B
SKIP12: INC DISCNT
CMP #10, DISCNT
BGT SKIP20
JMP FULLUP
SKIP20: CLR BLKNUM
TST DISKSW
BGT DRIVE1
INC DISKSW
BIC #1, @DGOUT1
BIS #2, @DGOUT1
.ENTER @AREA, #1, @DX1, #1
MOV R0, SPAVAL
JSR R5, DIGOUT
.PRT @MSG3
BCE SKIP11
INC DX0B
INC DX1B
JMP SKIP5
SKIP11: .PRINT @MSG10
.EXIT
DRIVE1: CLR DISKSW
BIC #2, @DGOUT1
BIS #1, @DGOUT1
.ENTER @AREA, #0, @DX0, #1
MOV R0, SPAVAL
JSR R5, DIGOUT
.PRT @MSG3
BCE SKIP10
INC DX0B
INC DX1B
JMP SKIP6
SKIP10: .PRINT @MSG10
.JMP WAI
.EXIT
WRTERR: INC BLKNUM
JUS .10133-

C1..R
ro...s£: .TTINR ; RECEI
V£ REQUEST
TO CLDSE DISt::
CHANNEl...
B2 g:::!P21
• TTOUTR
OtPB
BHE SKIP21
INC Ri
QIP
R1. .... 3
BNE CLSE
EXIT: .CLOSE DISKSW
.EXIT

SKIP21: MOV
NUMAV5,R1
MOV
@DIGINPUT,R2
BIT
$100,R2
BES
MULT
BIC
$177700,R2
TST
R2
BEO
TABCH
CLR
R0
DIV
R2,R0
MOV
R0,NUMAV1
BR
TABCH
MULT: BIC
$177700,R2
TST
R2
BEO
TABCH
MUL
R2,R1
MOV
R1,NUMAV1
TABCH: TSTB
@DIGINPUT
BMI
SKIP22
JMP
TABLE1
SKIP22: MOV
$33,R0
.TTYOUT
MOV
$110,R0
.TTYOUT
JSR
R5,TIMOUT
MOV
BLKNUM,R0
DEC
R0
JSR
R5,DIGOUT
MOV
SPAVAL,SLEFT
SUB
BLKNUM,SLEFT
MOV
SLEFT,R0
JSR
R5,DIGOUT
.PRINT
#MSG26
.PRINT
#MSG24
CLR
COUNT
MOV
#Chnst,A,R2
MOV
SPACE,DATPTR
MOV
NUMBCH,R3
TABLE: MOV
COUNT,R0
INC
COUNT
JSR
R5,DIGOUT
MOV
$11,R0
.TTYOUT
MOV
(R2),R4
BIC
$177400,R4
ASR
R4
ASR
R4
ASR
R4
MOV
R4,HDC
MOV
R4,R0
JSR
R5,DIGOUT
MOV
$11,R0
.TTYOUT
MOV
#FXGN,R1
ASL
HDC

:ENTER SPECIAL MODE FOR TERMINAL INPUT

:RECEIVE REQUEST TO CLOSE DISK CHANNEL

:ECHO CHARACTER ON CONSOLE

:HAVE 3 & SYMBOLS BEEN RECEIVED?

:CLOSE LAST FILE/TERMINATE RUN

:DISPLAY SOFTWARE CHANNEL NUMBER

:DISPLAY HARDWARE CHANNEL NUMBER

B-7
ADD       HDC1,R1
MOV       (R1),R0
MOV       R0,FIXGN
JSR       R5,DIGOUT
ADD       #10,R0
TTYOUT
MOV       #57,R0
TTYOUT
MOVB      (R2)+,R4
BICB      #370,R4
MOV       R4,CODE
CMPB      #2,R4
BLT       NEXT30
MOV       #1,GN
JMP       FIN

NEXT30:  CMPB      #3,R4
BNE       NEXT31
MOV       #2,GN
JMP       FIN

NEXT31:  CMPB      #4,R4
BNE       NEXT32
MOV       #4,GN
JMP       FIN

NEXT32:  CMPB      #5,R4
BNE       NEXT33
MOV       #8,GN
JMP       FIN

NEXT33:  CMPB      #6,R4
BNE       NEXT34
MOV       #16,GN
JMP       FIN

NEXT34:  MOV       #32,GN
FIN:      MOV       GN,R0
JSR       R5,DIGOUT
MOV       #11,R0
TTYOUT
MOV       @DATPTR,R0
MUL       #10000,R0
DIV       #8192,R0
JSR       R5,FLOAT
MOV       (R0),DIV3
MOV       (R0),DIV4
MOV       FIXGN,R0
JSR       R5,FLOAT
MOV       (R0)+,DIV1
MOV       (R0),DIV2
MOV       #DIV1,R0
FDIV      R0
MOV       GN,R0
JSR       R5,FLOAT
MOV       (R0)+,DIV1
MOV       (R0),DIV2
MOV       #DIV1,R0
FDIV      R0
JSR       R5,FLTOUT
ADD       #2,DATPTR
MOV       #11,R0
TTYOUT
DEC       R3
BEQ      TABLE4
JMP      TABLE

TABLE4:  MOV       @DIGIMP,R0
JSR       R5,OUT

TABLE1:  MOV       #1,R4
MOV       #10,R1
MOV       @ALARMS,R2
MOV       SPACE,R3
ALARM:
 MOV @CHNSTA, STATUS
 INC STATUS
 BICB #370, TEST
 CMPB #2, TEST
 BLT NXT30
 MOV $1, GAN
 JMP FIN1
 NXT30:
 CMPB #3, TEST
 BNE NXT31
 MOV #2, GAN
 JMP FIN1
 NXT31:
 CMPB #4, TEST
 BNE NXT32
 MOV #4, GAN
 JMP FIN1
 NXT32:
 CMPB #5, TEST
 BNE NXT33
 MOV #8, GAN
 JMP FIN1
 NXT33:
 CMPB #6, TEST
 BNE NXT34
 MOV #16, GAN
 JMP FIN1
 NXT34:
 MOV $32, GAN
 F1N1:
 MOV (R2)+, R5
 MUL GAN, R5
 CMP R5, (R3) 
 BGT NOAL
 BIS R4, @DDOUT2
 ADD #2, R2
 ADD #2, R3
 BR NOAL1
 NOAL:
 MOV (R2)+, R5
 MUL GAN, R5
 CMP R5, (R3)+
 BNE NOAL2
 BIS R4, @DDOUT2
 BR NOAL1
 NOAL2:
 BIC R4, @DDOUT2
 :CHECK FOR LOWER ALARM LIMIT
 NOAL1:
 ASL R4
 ADD $2, COUNTR
 DEC R1
 BRD OVER10
 JMP ALARM
 OVER10:
 JMP WAI
 FULLUP: .PRINT .MSG21
 .CLOSE DISKS
 .EXIT

;**********COMPLETION ROUTINES~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
RECEIV: IN BSTFLG ;RECEIVE DATA FROM BG COMPLETION ROUTINE
RTS PC
WRITCR: CLR WRTSTA ;WRITE TO DISK COMPLETION ROUTINE
SKP19: RTS PC
;**********CLOCK INTERRUPT ROUTINE~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
CLKINT: MOV $123, @CSR ;RESET CLOCK TO ENABLE NEXT INTERRUPT
 IMTEN 7
 MOV R0, -(SP)
 MOV R1, -(SP)
 MOV R2, -(SP)
 MOV R3, -(SP)
 MOV ADDRESS, R3
 MOV @1, TRNDAT ;SET NUMBER OF TRANSIENT DATA CHANNELS STORED
 MOV NUMCH, CHCNT
 ADD $1, TIMERO ;STORE COUNTER
 ADD CARRY BIT TO HIGH ORDER WORD

B-9
TST AVGSW ; SELECT PROPER AVERAGE BUFFER TO FILL
BGT AVBFCH
MOV AVPTR0, R2
MOV @CHCD0, R1
BR NEXTCH
AVBFCH: MOV AVPTR1, R2
MOV @CHCD1, R1
NEXTCH: MOVB (R1)+, @OUT ; INITIATE ADC CONVERSION
DEC CHCNT
BMI CHKBUF
DONE: TSTB @ADDR ; CHECK FOR READY BIT
BPL DONE
ADD @INP, (R2)+ ; STORE DATA IN AVERAGE BUFFER
ADC (R2) ; ADD CARRY BIT TO HIGH ORDER WORD
TST @INP ; CHECK SIGN OF DATA WORD
SXT R0, R2 ; EXTEND SIGN
ADD R0, (R2)+ ; ADD SIGN EXTENSION TO HIGH ORDER WORD
DEC TRNDAT
BMI NEXTCH
MOV @INP, (R3) ; STORE DATA IN TRANSIENT BUFFER
ADD $1024, R3
BR NEXTCH
CHKBUF: DEC BUFCNT
BPL RESTOR
MOV $512, BUFCNT
MOV @CHN1, ADDR
BR CHKAVG
BUFR1: MOV @CHN1, ADDR
INC BUFSW
BR CHKAVG
RESTOR: ADD $2, ADDR
CHKAVG: DEC AVGCNT
BGT RETURN
MOV NUMAV1, AVGCNT
TST AVGSW
BGT CHECK2
CHECK1: DEC AVBFCT
BEC WRSIG2
ADD NOCHX4, AVPTR0
RETURN: MOV (SP)+, R3 ; RESTORE REGISTERS PRIOR TO RETURN
MOV (SP)+, R2
MOV (SP)+, R1
MOV (SP)+, R0
RTS PC
WRSIG2: INC WRFLG ; SET WRITE FLAG
INC AVGSW
MOV @AVBUF0, AVPTR0
MOV REPAVG, AVBFCT
BR RETURN
CHECK2: DEC AVBFCT
BEC WRSIG1
ADD NOCHX4, AVPTR1
BR RETURN
WRSIG1: INC WRFLG ; SET WRITE FLAG
CLR AVGSW
MOV @AVBUF1, AVPTR1
MOV REPAVG, AVBFCT
BR RETURN
ERR1: HALT
MSG0: .ASCIZ "ENTER CTRL/F FIRST"
MSG1: .ASCIZ "ENTER TEXT; LIMIT=510 CHARACTERS; TERMINATE WITH @ SYMBOL/
MSG2: .ASCIZ "ENTER NUMBER OF CHANNELS PER SCAN; MAX=30/
MSG3: .ASCIZ "NUMBER OF FREE BLOCKS ON DISK/
MSG4: .ASCIZ "PUT ADC IN REMOTE OPERATION AND RESTART PROGRAM/
MSG5: .ASCIZ "ENTER CHANNEL/GAIN CODE FOR NUMBER OF CHANNELS SELECTED/
MSG6: .ASCIZ "ENTER NUMBER OF DATA POINTS TO BE AVERAGED; 200-32000/
MSG7: .ASCIZ "VECTORS ALREADY IN USE/
MSG8: .ASCIZ /DISK CHANNEL CLOSED/
MSG9: .ASCIZ /BAD ENTER/
MSG10: .ASCIZ /BAD WRITE/
MSG11: .ASCIZ /HAS DATE AND TIME BEEN PREVIOUSLY ENTERED? IF NOT RESTART./
MSG12: .ASCIZ /BACKED UP WRITE TO DISK REQUESTS/
MSG13: .ASCIZ /TYPE 1 FOR AUTO GAIN/
MSG14: .ASCIZ /SWITCHING DISKS; INSERT BLANK/
MSG15: .ASCIZ /ENTER CLOCK INTERRUPT PERIOD (10 MS TYPICAL; 320 MS MAX) /
MSG16: .ASCIZ /SEC PER DISK WRITE/
MSG17: .ASCIZ /STATUS=XX: WHERE XX=OCTAL SCAN NO. & Y=OCTAL GAIN VALUE/
MSG18: .ASCIZ /??????????DATE HASN'T BEEN ENTERED????????????????/
MSG19: .ASCIZ /10 FILES HAVE BEEN WRITTEN: PROGRAM EXITED/
MSG20: .ASCIZ /ENTER DATA FILE EXTENSION (TYPICALLY = 'A') /
MSG21: .ASCIZ /TIME BETWEEN DISK WRITES GREATER THAN 32000 SEC?/
MSG22: .ASCIZ /SCAN# CHAN# FX PR DATA(MV) /
MSG23: .ASCIZ /SCAN# CHAN# FX PR DATA(MV)/
MSG24: .ASCIZ /CURRENT AND REMAINING DISK BLOCKS/
MSG25: .ASCIZ /ENTER 24 ADC FIXED GAIN VALUES/
MSG26: .ASCIZ /ENTER 20 ALARM VALUES(HI/LO) FOR FIRST 10 CHANNELS SCANNED/
MSG27: .ASCIZ /TYPE 1 FOR AUTO RESTART/
MSG28: .ASCIZ /BAD LOOKUP/

DON: .BLKW 7*6.
AVBUF: .BLKW 512.
AVBUF1: .BLKW 512.
TEXT: .BLKB 512.
AREA1: .BLKW 8.
AREA2: .BLKW 8.
AREA3: .BLKW 8.
AREA: .BLKW 8.
DX: .RADS 0 /DX0DATA /
DX1: .RADS 0 /DX1DATA /
DX2: .RADS 0 /DX2DATA /
DX3: .RADS 0 /DX3DATA /
DX4: .RADS 0 /DX4DATA /
DX5: .RADS 0 /DX5DATA /
DX6: .RADS 0 /
SPAV: .WORD 0
SLEFT: .WORD 0
NUMPR: .WORD 0
CODE: .WORD 0
HCH: .WORD 0
DIV1: .WORD 0
DIV2: .WORD 0
DIV3: .WORD 0
DIV4: .WORD 0
FIXEN: .WORD 0
NUMBER: .WORD 0
ORDER: .WORD 0
TEST: .WORD 0
CODE1: .WORD 0
GAIN: .WORD 0
STATUS: .WORD 0
COUNTER: .WORD 0
STORE: .WORD 0
COUNT: .WORD 0
GMC: .WORD 0
TABFLG: .WORD 0
WRAP: .WORD 0
WRRAP: .WORD 0
DISCH: .WORD 0
DISKS: .WORD 1
ALRMD: .BLKW 20.
A1: .WORD 0
BLGNUM: .WORD 0
EXAM: .WORD 0
DISSPTR: .WORD 0
SPACE: .WORD 0
AVBFCT: .WORD 0
GAIN: .WORD 0
GAIN1: .BLKW 1
GAIN2: .BLKW 1
CNDT: .WORD 0
BUFFM: .WORD 0
BUFFR1: .WORD 0
BUFFER2: .WORD 0
AMPTR1: .WORD 0
AMPTR2: .WORD 0
REOSTA: .WORD 0
REOSFLG: .WORD 0
BUFFLOC: .WORD 0
WRITEFLG: .WORD 0
AVAGNT: .WORD 0
AVRG: .WORD 0
AVTG: .WORD 0
NOCNEXT: .WORD 0
ABSG: .WORD 0
REQUEST: .BLKW 1
REQ1: .BLKW 1
REQ2: .BLKW 1
REQ3: .BLKW 1
REQ4: .BLKW 1
REQ5: .BLKW 1
REQ6: .WORD 0
THER0: .WORD 0
THER1: .WORD 0
NUMAV1: .WORD 0
SAME: .WORD 170420
CMCROM: .BLKW 30
CMCRO1: .BLKW 30
CLOCK1: .WORD 0
MUSCHR: .WORD 0
THERMAT: .WORD 10
NUMAV6: .WORD 0
REPAV6: .WORD 0
AVGTSZ: .WORD 0
TIM: .WORD 0
AUTOCH: .WORD 0
CLOCK: .WORD 0
INCSR: .WORD 167770
OUT: .WORD 167772
IMP: .WORD 167774
CSR: .WORD 170420
BFR: .WORD 170422
BASE1: .WORD 177550
BASE2: .WORD 177552
BIGIMP: .WORD 177602
FIXMA: .BLKW 24
ALARM2: .BLKW 20
DATE: .WORD 0
TOD: .WORD 0
TOD1: .WORD 0
CHASTA: .BLKW 30
AVGSM: .BLKW 256
CHIN1: .BLKW 512
CHIN2: .BLKW 512
CHIN3: .BLKW 512
CHIN4: .BLKW 512
B.2 GRAFTR.FOR LISTING

.C     TYPE GRAFTR.FOR
.C     TITLE=GRAFTR
.C     THIS PROGRAM DISPLAYS TRANSIENT DATA STORED IN FOREGROUND
.C     PROGRAM. ANY 2 OF 10 TRANSIENT CHANNELS MAY BE DISPLAYED
.C     AT ONCE. VERSION OF 7-19-79 BY PETER L. HENDRICK, PNL....
.C     INTNC=24, M(513), N(513), GRID, REQUEST(4), CLKPER, IGAINC(3)
.C     REAL*4 XAXIS(10), TITLE(10), SCALE(10)
.C     LOGICAL*1 IDSET(5)
.C     GO TO 4G0
.C     TYPE 465
.C     FORMAT(X,'TYPE 1 TO OVERRIDE GRAPHIC RESET')
.C     ACCEPT 25, IRESET
.C     IF (IRESET .LE. 1) GO TO 4G0
.C     CALL PLOTT55(2,0,1)
.C     25
.C     TYPE 7G5
.C     FORMAT(X,'ENTER CLOCK INTERRUPT PERIOD, (MS)')
.C     ACCEPT 25, CLKPER
.C     TYPE 385
.C     FORMAT(X,'ENTER SCAN NO., SPAN(MV), OFFSET(MV), AND')
.C     TYPE 5G5
.C     FORMAT(' FIXED GAIN FOR GRAPH 0')
.C     ACCEPT 3G, ICHG, IFSG, IAOFFG, IFXGN
.C     IF (ICHG .GT. 10 .OR. ICHG .LT. 0) GO TO 4G0
.C     TYPE 5G5
.C     FORMAT(X,'ENTER SCAN NO., SPAN(MV), OFFSET(MV), AND')
.C     TYPE 5G5
.C     FORMAT(' FIXED GAIN FOR GRAPH 1')
.C     ACCEPT 3G, ICH1, IFS1, IAOFF1, IFXGN1
.C     IF (ICH1 .GT. 10 .OR. ICH1 .LT. 1) GO TO 715
.C     30
.C     FORMAT(X,/,1B,/,1B,/,1B)
.C     REQUEST(2)=(ICHG)*1024
.C     REQUEST(3)=(ICH1)*1024
.C     FSO=IFSO
.C     FSI=IFS1
.C     AOFF=IAOFF0
.C     AOFF=IAOFF1
.C     FXGN=IFXGN0
.C     FXGN1=IFXGN1
.C     IF (ISDATW(REQUEST,4),NE. 0) STOP 'FOREGROUND JOB NOT RUNNING'
.C     IF (IRCDV(N,512),NE. 0) STOP 'ERROR RECEIVING M BUFFER'
.C     IF (IRCDV(N,512),NE. 0) STOP 'ERROR RECEIVING N BUFFER'
.C     IF (IRCDW(IGAINC,2),NE. 0) STOP 'ERROR RECEIVING GAIN CODES'
.C     IF (IGAINC(2),GT. 2) GO TO 705
.C     IGAINC(2)=1
.C     GO TO 799
.C     705
.C     IF (IGAINC(2),NE. 3) GO TO 710
.C     IGAINC(2)=2
GO TO 799
IF (IGAINC(2) .NE. 4) GO TO 716
IGAINC(2)=4
GO TO 799
716  IF (IGAINC(2) .NE. 5) GO TO 720
IGAINC(2)=8
GO TO 799
720  IF (IGAINC(2) .NE. 6) GO TO 725
IGAINC(2)=16
GO TO 799
725  IF (IGAINC(2) .NE. 7) GO TO 999
IGAINC(2)=32
GO TO 799
799  IF (IGAINC(3) .GT. 2) GO TO 730
IGAINC(3)=1
GO TO 798
730  IF (IGAINC(3) .NE. 3) GO TO 735
IGAINC(3)=2
GO TO 798
735  IF (IGAINC(3) .NE. 4) GO TO 740
IGAINC(3)=4
GO TO 798
740  IF (IGAINC(3) .NE. 5) GO TO 745
IGAINC(3)=6
GO TO 799
745  IF (IGAINC(3) .NE. 6) GO TO 750
IGAINC(3)=16
GO TO 798
750  IF (IGAINC(3) .NE. 7) GO TO 999
IGAINC(3)=32
798  DO 225 , J=1,512
M(J)=(236./FS1)*((10000./(8192.*FXGN1*IGAINC(2))))*N(1+J)-AOFF2
M(J)=(236./FS1)*((10000./(8192.*FXGN1*IGAINC(3))))*N(1+J)-AOFF1
CONTINUE
225  CALL PLOT55(2,1+2+4+32+64+128+256,)
CALL PLOT55(7,0,0)
CALL PLOT55(1,0,0)
CALL PLOT55(3,-512,0)
CALL PLOT55(3,-512,M)
DO 185 , I=1,15 !PLACE VERTICAL MARKERS ON GRAPH 0
CALL PLOT55(6,1#32,1)
CALL PLOT55(4,1,0) !PLACE HORIZONTAL AXIS AT X=0 AND 235
185  TYPE 190,
190  FORMAT(X,'TYPE:1 TO DELETE GRIDS')
ACCEPT 210,GRID
210  FORMAT(I4)
IF (GRID .NE. 1) GO TO 805
CALL PLOT55(2,0,32+64)
GO TO 220
805  DO 190 , I=1,236,59 !PUT HORIZONTAL GRIDS IN 1/4 INCREMENTS
190  CALL PLOT55(4,1,1-1)
DO 200 , I=1,512,50 !PUT VERTICAL GRIDS IN 50 POINT INCREMENTS
200  CALL PLOT55(5,1,1)
CONTINUE
220  CALL PLOT55(4,1,235)
CALL PLOT55(5,0,1) !PLACE VERTICAL AXIS AT X=0 AND 511
CALL PLOT55(5,511,1)
TYPE 170,
ACCEPT 125, (TITLE(I),I=1,10)
170  FORMAT(X,'TYPE:"SPACE,TITLE OF GRAPH")
TYPE 120,
ACCEPT 125, (XAXIS(I),I=1,10)
120  FORMAT(X,'TYPE:"SPACE, LABEL FOR X-AXIS")
TYPE 140,
140  FORMAT(X,'TYPE:"LABEL FOR Y-AXIS")
ACCEPT 150, (YAR(I),I=1,20)
150  FORMAT(20A1)
CALL PLOT55(9, 0, 0)
CALL PLOT55(10, 0, 0, 0)
CALL PLOT55(9, 0, 0, 1)
TYPE 60, ! LABEL CHANNEL NO., FULL SCALE, AND OFFSET
TYPE 65, ICH, ICH1
TYPE 70, IFS0, IFS1
TYPE 80, IAOFF0, IAOFF1
TYPE 545, IFXGM0, IFXGN1
TYPE 3005, IGAINC(2), IGAINC(3)
60 FORMAT(16X, 'GRAPH 0', GRAPH 1(+))
65 FORMAT(3X, 'SCAN NO.', I10, I11)
70 FORMAT(3X, 'FULL SCALE', I10, I11)
80 FORMAT(3X, 'OFFSET', I10, I11)
545 FORMAT(3X, 'FIXED GAIN', I10, I11)
3005 FORMAT(3X, 'PROG. GAIN', I10, I11)
CALL PLOT55(9, 10, 21)
TYPE 125, (TITLE(I), I=1, 10)
775 CALL PLOT55(9, 55, 21)! LABEL X-AXIS
TYPE 125, (XAXIS(I), I=1, 10)
125 FORMAT(X, 19A4)
760 DO 160, I=1, 10
CALL PLOT55(9, 0, I-1)! LABEL Y-AXIS
CALL PLOT55(12, 1, IAR(I))
160 CONTINUE
CALL PLOT55(9, 55, 2)
CALL TIMDAT
CALL PLOT55(9, 2, 23)
IF (CLKPER .EQ. 0) GO TO 285
SCALE(I)=50.#CLKPER/1000.
TYPE 765, (I*SCALE(I), I=1, 9)
765 FORMAT(3F8.2)
785 CALL PLOT55(9, 2, 12)
937 TYPE 20,
20 FORMAT(5X, 'I')
ACCEPT 25, IFLG
IF (IFLAG .EQ. 1) GO TO 375
CALL PLOT55(2, 512, 0)
CALL PLOT55(2, 0, 1)
CALL PLOT55(9, 2, 22)
195 STOP
399 TYPE 998,
998 FORMAT(X, 'GAIN CODE OUTSIDE OF RANGE 1-7?')
GO TO 997
END

B.3. GRAFAV.FOR LISTING

C TYPE GRAFAV.FOR
C TITLE=GRAFAV(GRAPH AVERAGED DATA)
C THIS BACKGROUND PROGRAM READS BINARY DATA FROM DISK
C AND DISPLAYS IT GRAPHICALLY. IT WILL PLOT ANY TWO
C CHANNELS REQUESTED. BOTH PERMANENT AND TEMPORARY
C FILES MAY BE ACCESSED.
C PETER L. HENDRICK; VERSION OF 19-JUL-79
C LOGICAL*1 IAR(24), STRNG(8), CHNSTA(24)
C DIMENSION IDATA(512), IRDATA(256), IDATA1(512)
C INTEGER*2 GRID, NAME(4), SPEC(39), RCODE, ITIME(2)
C EQUIVALENCE (NAME(1), SPEC(16))
C EQUIVALENCE (IRDATA(4), CHNSTA(1))
C REAL*4 XAXIS(10), TITLE(10), EXT(2)
C DATA EXT/6R00A000, 6R00A00A/
IF(IDSET(5).NE.0) STOP 'NOT ENOUGH QUEUE SPACE'
ICHAN=IDETC() 'ALLOCATE CHANNEL ICHAN'
CALL PURGE(ICHAN)
TYPE 3005.

3005 FORMAT(X,'ENTER 1 TO GRAPH ACTIVE DATA FILE')
ACCEPT 125, IIN
IF(IIN.NE.1) GO TO 400
IF(ICHCPY(ICHAN,1).EQ.0) GO TO 370
IF(ICHCPY(ICHAN,0).EQ.0) GO TO 370
TYPE 3010.

3010 FORMAT(X,'NO ACTIVE FILE?')

400 TYPE 390

400 FORMAT(X,'ENTER PERMANENT FILE NAME')
IF(ICSI(SPEC,EXT,0).NE.0) GO TO 400
IF(LOOKUP(ICHAN,NNAME).GE.0) GO TO 370
IF(LOOKUP(ICHAN,NNAME).NE.-2) GO TO 1125
TYPE 105

105 FORMAT(X,'ENTER STARTING BLOCK NUMBER')
ACCEPT 110,IIBLI

110 FORMAT(16)
TYPE 120.

120 FORMAT(X,'ENTER NUMBER OF CHANNELS/SCAN')
ACCEPT 125,ICNLS

125 FORMAT(15)
ISTART=(ICNLS+1)/2+4
ILEFT=257-ISTART
ITIMES=ILEFT/ICNLS
IBLCKS=512/ITIMES
TYPE 1250. ISTART, ILEFT, IBLCKS

1250 FORMAT(315)

20 FORMAT(X,'ENTER SCAN NUMBERS FOR GRAPHS 0 AND 1')
ACCEPT 4015,ICH0,ICH1

4015 FORMAT(16,/,16)
TYPE 50.

50 FORMAT(X,'ENTER SPAN,OFFSET,AND FIXED GAIN FOR GRAPH 0')
ACCEPT 25,IFS0,IAOFF0,IFXGN0
TYPE 55.

55 FORMAT(X,'ENTER SPAN,OFFSET,AND FIXED GAIN FOR GRAPH 1')
ACCEPT 25,IFS1,IAOFF1,IFXGN1
FS0=IFS0
FS1=IFS1
AOFFSET=IAOFFSET
AOFF1=IAOFF1
FXGNO=IFXGN0
FXGM1=IFXGN1

25 FORMAT(17,/,17,/,17)
CALL PLOT55(2,1,2+4,32+64,128+256.)
DO 915,M=1,512
IDATA(M)=0

915 IDATA(M)=0
IELK=IIBLI
RCODE=IREAD(256,IRDATA,IBLK,ICHAN)
IF(RCODE.LT.0) GO TO 1090
IF(IWAIT(ICHAN).NE.0) GO TO 1030
IDATE=IRDATA(1)
ITIME(1)=IRDATA(2)
ITIME(2)=IRDATA(3)
CALL TIMASC(ITIME,STRING)
CALL GETGAI(CHNSTA(ICH0+1),IPRGN0)
CALL GETGAI(CHNSTA(ICH1+1),IPRGN1)
DO 245,K=1,IBLCKS
RCODE=IREAD(256,IRDATA,IBLK,ICHAN)

245 IBLK=IBLK+1
IF(RCODE.EQ.-1) GO TO 1000
IF(RCODE.LT.0) GO TO 1050

1020 IF(IWAIT(ICHAN).NE.0) GO TO 1030
CALL GETGAI(CHNSTA(ICH1+1),IPRGN1)
CALL GETGAI(CHNSTA(ICH1),IPRGN1)
DO 115,J=1,ITIMES
IDATA0(J+ITIMES*(K-1))=((36.4*FS0)/(10000.1892*N*IPRGN0))
CALL IDATA(1+ICH1+ICHLS*(J-1))=IDATA0(J+ITIMES*(K-1))-(OFF0)
CALL IDATA1(J+ITIMES*(K-1))=(36.4*FS1)/(10000.1892*N*IPRGN1))
X
call IDATA(1+ICH1+ICHLS*(J-1))=IDATA1(J+ITIMES*(K-1))-(OFF1)
115 CONTINUE
245 CONTINUE
910 CALL PLOT55(7,0,0)
CALL PLOT55(1,0,0)
CALL PLOT55(3,-510,IDATA0)
CALL PLOT55(1,1,1)
CALL PLOT55(3,-510,IDATA1)
125 DO 205,1=1,512,32; PLACE VERTICAL MARKERS ON GRAPH 1
205 CALL PLOT55(6,1,1)
CALL PLOT55(4,1,0); PLACE HORIZONTAL AXIS AT Y=0 AND 235
TYPE 190,
ACCEPT 210,GRID
TYPE 230,GRID
230 FORMAT(X,1)
IF (GRID .EQ. 1) GO TO 320
DO 190, I=1,236,59; PLACE HORIZONTAL GRIDS
190 CALL PLOT55(4,1,I-1)
DO 200, I=1,501,50; PLACE VERTICAL GRIDS
200 CALL PLOT55(5,1-1,1)
190 FORMAT(X,'TYPE 1 TO DELETE GRIDS')
210 FORMAT(14)
320 CONTINUE
CALL PLOT55(4,1,235)
CALL PLOT55(5,1,0); PLACE VERTICAL AXIS AT X=0 AND 511
CALL PLOT55(5,511,1)
TYPE170,
ACCEPT 225,(TITLE(I),I=1,10)
170 FORMAT(X,'TYPE: "SPACE,TITLE OF GRAPH")
TYPE 220,
ACCEPT 225,(XAXIS(I),I=1,10)
220 FORMAT(X,'TYPE: "SPACE, LABEL FOR X-AXIS")
TYPE 140,
ACCEPT 150,(YAR(I),I=1,20)
CALL PLOT55(9,0,0)
CALL PLOT55(10,0,1); ERASE ALL ALPHANUMERIC FROM SCREEN
CALL PLOT55(9,0,1)
TYPE 60; LABEL CHANNEL NO., FULL SCALE, AND OFFSET
TYPE 65,ICH1,ICH1
TYPE 70,IF50,IF51
TYPE 80,IAOFF0,IAOFF1
TYPE 930,IFXGN0,IFXGN1
TYPE 4010,IPRGN0,IPRGN1
65 FORMAT(16X,'GRAPH 0 GRAPH 1(+)')
65 FORMAT(3X,'SCAN NO. ',II0,I11)
70 FORMAT(3X,'FULL SCALE',II0,I11)
80 FORMAT(3X,'OFFSET',II0,I11)
930 FORMAT(3X,'FIXED GAIN',II0,I11)
4010 FORMAT(3X,'PROG. GAIN',II0,I11)
CALL PLOT55(9,55,1)
CALL TIMDAT
CALL PLOT55(9,55,1)
TYPE 5075,IBK1
5075 FORMAT(X,'START BLOCK=',I4)
CALL PLOT55(9,55,2)
TYPE 4075,
4075 FORMAT(X,'START DATE=')
CALL PLOT55(9,66,3)
CALL DECDAT(IDATE)
CALL PLOT55(9,55,3)
TYPE 925,(STRING(I),I=1,8)
B.4. Typav.for Listing

.TYP A
.TITLE=Typav(type averaged data)
C THIS BACKGROUND PROGRAM READS BINARY DATA FROM DISK
C AND DISPLAYS IT IN TABULAR FORM AT THE CONSOLE.
C IT WILL PLOT ANY TWO CHANNELS REQUESTED. BOTH
C PERMANENT AND TEMPORARY FILES MAY BE ACCESSED.
C Peter L. Hendrick: version of 19-Jul-79
C logical. String(b), Spec(b), Spec(3)
C Dimension data(512), irdata(256), data(512)
C Integer*2 name(4), spec(39), rcode, time(2), spec(30)
C equivalence (name(1), spec(16))
C Real*4 ext(2)
C Data ext/[gr00000a, gr00000a]
C if(idset(5).ne.0) stop 'not enough queue space'
C ichan=igetc() 'allocate channel ichan
C 365 call purge(ichan)
C type 4005.
C 4005 format(x,'type 1 to access active file')
C accept 110. if file
C if (ifile .ne. 1) go to 400
C if (ichcmpy(ichan,1).eq.0) go to 370
C if (ichcmpy(ichan,2).eq.0) go to 370
C type 4010.
C 4010 format(x,'no active file?')
C 400 type 390
C 390 format(x,'enter permanent file name; default ext=00a')
C if(icsi(spec,ext,.ne.0).ne.0) go to 400
IF(LOOKUP(ICHAN, NAME), GE, 0) GO TO 370
IF(LOOKUP(ICHAN, NAME), NE, -2) GO TO 1125

TYPE 105
105 FORMAT(X, 'ENTER STARTING BLOCK NUMBER')
ACCEPT 110, IBLKI

110 FORMAT(I6)
RCODE=IREAD(256, IRDATA, IBLKI, ICHAN)
IF(RCODE .LT. 0) GO TO 1050
IF(IWAIT(ICHAN), NE, 0) GO TO 1030
ITIME(1)=IRDATA(2)
ITIME(2)=IRDATA(3)
CALL TIMASC(ITIME, STRNG) , 'GET TIME OF DAY AND CONVERT TO ASCII'
TYPE 120
120 FORMAT(X, 'ENTER NUMBER OF CHANNELS/SCAN')
ACCEPT 125, ICNLs
125 FORMAT(I5)
ISTART=(ICNLS+1)/2+4
IREPAV=(257-ISTART)/ICNLs
IBLCKS=512/IREPAV
TYPE 2000,
2000 FORMAT(X, 'ENTER 1 TO TYPE FIRST BLOCK SPECIFIED')
ACCEPT 2005, IREQS
2005 FORMAT(I6)
IF(IREQS .NE. 1) GO TO 3005
TYPE 2015, IRDATA(1), IRDATA(2), IRDATA(3)
2015 FORMAT(I6)
BD 4015, J=1, ICNLs
SPECC(J)=SPECB(J)
IF(SPECC(J), GE, 0) GO TO 4015
SPECC(J)=SPECC(J)+256
4015 CONTINUE
TYPE 2025, (SPECC(J), J=1, ICNLs)
2025 FORMAT(X, 1206)
TYPE 2040,
2040 FORMAT(X)
TYPE 2030, (IRDATA(J), J=ISTART, 256)
2030 FORMAT(X, 1216)
3005 TYPE 2050
2050 FORMAT(X, 'ENTER 1 TO TYPE 2 CHANNELS IN MILLIVOLTS')
ACCEPT 25, IREQS
IF(IREQS .NE. 1) GO TO 1000
TYPE 20,
20 FORMAT(X, 'ENTER TWO CHANNELS')
ACCEPT 9005, ICH0, ICH1
905 FORMAT(I6, /, 16)
TYPE 50,
50 FORMAT(X, 'ENTER FIXED GAIN FOR FIRST CHANNEL SPECIFIED')
ACCEPT 25, IFXGN0
TYPE 55,
55 FORMAT(X, 'ENTER FIXED GAIN FOR SECOND CHANNEL SPECIFIED')
ACCEPT 25, IFXGM1
FXGM0=IFXGN0
FXGM1=IFXGM1
PRGM0=IPRGN0
PRGM1=IPRGM1
25 FORMAT(I7)
BD 915, M=1, 512
DATA0(M)=0
915 DATA1(M)=0
IBLK=IBLKI
TYPE 4075,
4075 FORMAT(*, X, 'START DATE=')
CALL DCEDAT(IRDATA(1))
TYPE 925, (STRNG1, I=1, 8)
925 FORMAT(X, 'START TIME=' BA1)
TYPE 4025,
4025 FORMAT(4X, 'NO.', 4X, 'DATA0', 4X, 'PRGN0', 5X, 'DATA1', 4X, 'PRGN1', 4X, 'PRGM1')
B.5 TYPLOG.FOR LISTING

.TYP TYPLOG.FOR
C TITLE=TYPLOG(TYPE A LOG DATA FILE ON DISK)
C THIS PROGRAM ALLOWS THE USER TO ENTER TEXT DATA
C FROM THE KEYBOARD AND STORE IT ON DISK FOR FUTURE REFERENCE
C PETER L. HENDRICK; VERSION OF 23-MAR-79
C LOGICAL*1 ERR,STRG(560)
C INTEGER*2 NAME(4),SPEC(39),RCODE
C EQUIVALENCE (NAME(1),SPEC(16))
C REAL*4 EXT(2)
C DATA EXT/GRXTXTXT,GRXTXTXT/
C IF(IDSET(5),NE.0) STOP 'NOT ENOUGH QUEUE SPACE'
C ICHAN=IGETC() 'ALLOCATE CHANNEL ICHAN'
C GO TO 400
365 CALL PURGE(ICHAN)
400 TYPE 390
390 FORMAT(X,'ENTER FILE NAME'/)
C IF(ISIS(SPEC,EXT,,3),NE.0) GO TO 400
C IENTCD=ENTER(ICHAN,NAME,1)
IF (IENTCD .EQ.-1) GO TO 2025
IF (IENTCD .LT. 0) GO TO 2015
TYPE 2005.

2005 FORMAT(X,’ENTER 6 LINES OF TEXT DATA’)
        DO 3020 L=1,6
3020       ACCEPT 3000, (STRING(J*80*(L-1)),J=1,80)
3000       FORMAT(B80A1)

CONTINUE
STRING(481)=0
WRCODE=IWRITE(256,STRING,0,0)
        IF (WRCODE .LT. 0) GO TO 2045
CALL CLOSEC(0)

1050       TYPE 335,
135       FORMAT(X,’TYPE 1 TO TYPE MORE DATA’)
        ACCEPT 125, IREQ
125       FORMAT(15)
        IF (IREQ .EQ. 1) GO TO 365
STOP
2015       TYPE 2020, IENTCD
2020       FORMAT(X,16,’=ENTER ERROR CODE’)
        , GO TO 1000
2025       TYPE 2030,
2030       FORMAT(X,’CHANNEL ALREADY IN USE’)
        GO TO 1000
2045       TYPE 2050, WRCODE
2050       FORMAT(X,16,’=WRITE ERROR CODE’)
        GO TO 1000
END

B.6 RETXT.FOR LISTING

**TYPE’ RETXT; FOR
C   TITLE=RETXT(READ TEXT DATA)
C   THIS BACKGROUND PROGRAM READS ASCII TEXT DATA
C   FROM DISK
C   PETER L. HENDRICK: VERSION OF 28-MAR-79
INTEGER*2 NAME(4),SPEC(39),RCODE
EQUIVALENCE (NAME(1),SPEC(16))
REAL*4 EXT(2)
LOGICAL*1 ITXT(512)
DATA EXT/6RTXTXT,6RTXTXT/
   IF (ISET(5),NE. 0) STOP ’NOT ENOUGH QUEUE SPACE’
   ICHAN=ICETC( ) !ALLOCATE CHANNEL ICHAN
   CALL PURGE(ICHAN)
365       TYPE 390
390       FORMAT(X,’ENTER FILE NAME’/
   IF (ICSI(SPEC,EXT,,0),NE. 0) GO TO 400
   IF (LOOKUP(ICHAN,NAM),NE. 0) GO TO 370
   IF (LOOKUP(ICHAN,NAM),EQ.-2) GO TO 1135
   IF (LOOKUP(ICHAN,NAM),LT. 0) GO TO 1125
370       IBLK=0
RCODE=IREAD (256,ITXT,IBLK,ICHAN)
   IF (RCODE.LT. 0) GO TO 1000
   IF (WAIT (ICHAN),NE. 0) GO TO 1030
   CALL PRINT (ITXT)
1000      TYPE 335
335       FORMAT(X,’TYPE 1 TO READ MORE TEXT DATA’)
        ACCEPT 125, IREQ
125       FORMAT(15)
        IF (IREQ .EQ. 1) GO TO 365
STOP
B.7 REINT3.FOR LISTING

**TYPE REINT3.FOR**

**TITLE=REINT3(READ INITIALIZATION DATA, REV 3)**

THIS BACKGROUND PROGRAM READS BINARY INITIALIZATION TEXT DATA FROM DISK USING IN CONJUNCTION WITH DAC4.

PETER L. HENDRICK: VERSION OF 18-JUL-78

LOGICAL*1 ISTAT(31), STRING(8)

INTEGER*2 NAME(4), SPEC(39), RCODE, INITIAL(256), ISTAT(30), SUMMY(256)

INTEGER*2 ITIME(2)

EQUIVALENCE (SUMMY, INITIAL(115))

EQUIVALENCE (TIME, INITIAL(177))

EQUIVALENCE (NAME(1), SPEC(16))

EQUIVALENCE (ISTAT(1), INITIAL(179))

REAL*4 EXT(2)

DATA EXT/6RINTHINT .6RINTHINT/

IF(IGET(E).NE.0) STOP 'NOT ENOUGH QUEUE SPACE'

ICHAN=IGET(E) ! ALLOCATE CHANNEL ICHAN

CALL PURGE(ICHAN)

CALL DECDAT(SUMMY(62))

CALL TIMASC(ITIME, STRING)

**TYPE 390**

**FORMAT(X, 'ENTER FILE NAME; DEFAULT EXT=INT'/)**

IF(ICSJ(SPEC.EXT,..,0).NE.0) GO TO 400

IF(LOOKUP(ICHAN, NAME).GE.0) GO TO 370

IF(LOOKUP(ICHAN, NAME).EQ.-2) GO TO 1135

IF(LOOKUP(ICHAN, NAME).LT.0) GO TO 1125

IBLK=0

RCODE=IREAD(256, INITIAL, IBLK, ICHAN)

IF(RCODE .LT. 0) GO TO 1090

IF(IWAIT(ICHAN).NE.0) GO TO 1030

DO 1011 J=1,30

ISTAT(J)=ISTAT(J)

IF(IWAIT(ICHAN).NE.0) GO TO 1011

ISTAT(J)=ISTAT(J)+256

CONTINUE

**TYPE 4075.**

**FORMAT(X, 'START DATE=')**

**CALL DECDAT(SUMMY(62))**

**CALL TIMASC(ITIME, STRING)**

**TYPE 3025, (STRING(J), J=1,8)**

**FORMAT(X, 'TIME OF DAY:', BA1)**

**TYPE 935, (SUMMY(J), J=1,5)**

**FORMAT(X, 'NUMBC', I7, 1X, 'TRMDAT', I7, 1X, 'NAMAG', I7, 1X, 'REPAVG', I7, 1X, 'AVGSIZ', I7)**

**TYPE 940, (SUMMY(J), J=6,19)**


**TYPE 945, (SUMMY(J), J=11,15)**


B-22
X 1X, 'DGOUTI', 07)
TYPE 950, SUMMY(16), SUMMY(17), SUMMY(62), SUMMY(63), SUMMY(64)
FORMAT(X, 'DGOUT2', 07, IX, 'DIGINP', 07, IX, 'DATE ', 07, IX, 'TOD ', 07,
X 1X, 'TODI ', 07)
TYPE 3020.
3020 FORMAT(X, 'FIXED GAIN VALUES: ')
TYPE 3010.
3010 FORMAT(6X, '0', 5X, '1', 5X, '2', 5X, '3', 5X, '4', 5X, '5', 5X, '6', 5X, '7',
X 5X, '8', 5X, '9', 4X, '10', 4X, '11')
TYPE 3005, (SUMMY(J), J=10, 29)
3005 FORMAT(X, 12I6)
TYPE 3015.
TYPE 3005, (SUMMY(J), J=30, 41)
TYPE 4005.
4005 FORMAT(X, 'ALARM SET POINTS (HIGH/LOW): ')
TYPE 4010.
4010 FORMAT(3X, '0', 13X, '1', 13X, '2', 13X, '3', 13X, '4')
TYPE 4015, (SUMMY(J), J=42, 51)
4015 FORMAT(X, '1917')
TYPE 4015, (SUMMY(J), J=52, 61)
TYPE 965.
965 FORMAT(X, 'CHANSTA(OCTAL): ')
TYPE 965.
TYPE 960, (ISTATU(J), J=1, 15)
960 FORMAT(X, 1505)
TYPE 960.
960 FORMAT(X)
TYPE 970.
TYPE 960, (ISTATU(J), J=16, 30)
1000 TYPE 335
335 FORMAT(X, 'TYPE 1 TO READ MORE INITIALIZATION DATA')
ACCEPT 125, IREQ
125 FORMAT(I15)
IF (IREQ .EQ. 1) GO TO 365
STOP
1125 TYPE 1130, LOOKUP(ICHAN, NAME)
1130 FORMAT(X, 'LOOKUP ERROR CODE=', I3)
GO TO 1080
1135 TYPE 1140.
1140 FORMAT(X, 'CANNOT FIND FILE SPECIFIED DN DEVICE')
GO TO 1080
1080 TYPE 1095, RCODE
1095 FORMAT(X, 'IREAD ERROR CODE=', I3)
GO TO 1080
1080 STOP 'FATAL READ'
END

B.8 RESET.FOR LISTING

CALL PLOT55 (2, 512, 0)
STOP
END

B-23
B.9 GETGAI.MAC LISTING

TYPE GETGAI.MAC
; THIS MACRO SUBROUTINE RETURNS THE GAIN TO A FORTRAN BACKGROUND
; PROGRAM GIVEN THE CHANNEL STATUS CODE FOR THE DESIRED CHANNEL
; VERSION OF 29-MAR-79 BY P.L. HENDRICK
.GLOBL GETGAI
.TITLE GETGAI
GETGAI: MOV R5,R1
TST (R1)+
MOVB @(R1)+,R3 ;STORE CHANNEL STATUS CODE
INC (R1)
MOV (R1),GMPTR ;STORE ADDRESS TO PUT GAIN
BCB $370,R3
CMPB $2,R3 ;CHECK FOR GAIN=1
BLT NEXT1
MOV $1,GAIN
JMP FINISH

NEXT1: CMPB $3,R3 ;CHECK FOR GAIN=2
BNE NEXT2
MOV $2,GAIN
JMP FINISH

NEXT2: CMPB $4,R3 ;CHECK FOR GAIN=4
BNE NEXT3
MOV $4,GAIN
JMP FINISH

NEXT3: CMPB $5,R3 ;CHECK FOR GAIN=8
BNE NEXT5
MOV $8,GAIN
JMP FINISH

NEXT5: CMPB $6,R3 ;CHECK FOR GAIN=16
BNE NEXT6
MOV $16,GAIN
JMP FINISH

NEXT6: MOV $32,GAIN ;GAIN=32
FINISH: MOV GAIN,GMPTR
RTS PC

CNTR: .WORD 0
GMPTR: .WORD 0
GAIN: .WORD 0
.END GETGAI

B.10 TIMDAT.MAC LISTING

TYPE TIMDAT.MAC
; TIMDAT (GET TIME AND DATE AND DISPLAY)
; THIS PROGRAM USES MACRO PROGRAM TIMOUT WRITTEN BY
; TOM WHITAKER TO DISPLAY TIME AND DATE AS A
; FORTRAN CALLABLE ROUTINE
; WRITTEN BY P.L. HENDRICK, VERSION OF 15-MAY-79
.GLOBL TIMDAT,TIMOUT
.TITLE TIMDAT
TIMDAT: JSR R5,TIMOUT
RTS PC
.END TIMDAT
DECDAT.MAC LISTING

.TYP£ DECDAT.MAC
.THEE PROGRAM ACCEPTS A DATE CODE AND OUTPUTS AN
.ASCII DATE STRING TO THE TERMINAL.
;VERSION OF 14-JUN-79 BY T. WHITAKER
;DESIGNED TO BE FORTRAN CALLABLE

.MCALL .TTYOUT
.IOGLB TIMITOUT, DIGOUT, DECDAT
.TITLE DECDAT

.DECDAT: IST
(R5)+
MOV @(R5), R0
MOV R0, DATE
ASH $-12, R0
.JSR R5, DIGOUT
.JSR R5, DDEL
MOV DATE, R0
BIC @176037, R0
ASH $-5, R0
.JSR R5, DIGOUT
.JSR R5, DDEL
MOV DATE, R0
BIC @177740, R0
ADD @72, R0
.JSR R5, DIGOUT
MOV @15, R0
.TTYOUT
MOV @12, R0
.TTYOUT
RTS PC
.DDEL: MOV @19, R0
.TTYOUT
MOV @57, R0
.TTYOUT
RTS R5

DATE: .WORD 0
.END DECDAT
APPENDIX C

PROCEDURE FOR MAKING DELETED FILES PERMANENT
APPENDIX C
PROCEDURE FOR MAKING DELETED FILES PERMANENT

In certain cases (e.g., a system crash) the user may need to make a file permanent that was previously deleted. For example, should the system crash while DAC04 is operating, the current data file will not be made permanent (i.e., the directory will not list the file name). Since the data will remain intact until that area of the disk is rewritten, it is possible to modify the directory to cause the data file to become permanent. The following procedure will make a deleted file permanent:

1. Reboot the system with system disk in drive 0 and deleted file in drive 1.
2. Obtain a directory of the deleted file. For example:

   .DIR/DEL/BLO[CR]
   DATA.00A  480  14

   where, in this example, the deleted file, DATA.00A, started at absolute block 14 and had 480 blocks originally allocated to it.
3. Convert the decimal numbers above (i.e., starting block number and temporary length) to octal.
4. Call DUP:

   .R DUP[CR]
*DX1:DATA.00A=/C:16:740 [CR]
   where 168 = 1410 and 7408 = 48010
5. Run DIR to confirm results. Data file DATA.00A, should now be listed as a permanent file.
APPENDIX D

HARDWARE ADDRESS AND VECTOR LOCATIONS
APPENDIX D
HARDWARE ADDRESS AND VECTOR LOCATIONS

All hardware interfaces are addressed in a manner identical to memory in the PDP11V/03. Generally the address space between 28K and 32K is reserved for I/O addresses, thus leaving 0-28K available for memory. Some I/O addresses have been altered, however, to allow an effective memory space of 29K. The current addresses are summarized in Table D.1. If the addresses for the ROM bootstrap and DRV11 parallel interfaces were relocated to a higher address, an effective memory of 30K could be realized. In the case of the DRV11 parallel interface this is easily accomplished. However, a method has not been discovered to relocate the ROM bootstrap addresses.

Many of the interfaces listed in Table D.1 can be operated on an interrupt basis if desired. In the present application, only the real-time programmable clock utilizes the interrupt feature. The clock interrupt vector is located at octal 440. Most installed interface cards have DIP switches, which allow either the interface address or interrupt vector addresses to be altered within a given range.
## TABLE D.1. Summary of I/O Addresses

<table>
<thead>
<tr>
<th>Address (octal bytes)</th>
<th>Interface Model</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>17777</td>
<td>*********</td>
<td>32K word limit</td>
</tr>
<tr>
<td>177602 (DATA)</td>
<td>ADAC</td>
<td>Digital input</td>
</tr>
<tr>
<td>177600 (CSR)</td>
<td>1616CCI</td>
<td>Interface</td>
</tr>
<tr>
<td>177566 (Spare)</td>
<td></td>
<td>Four-channel</td>
</tr>
<tr>
<td>177564 (Spare)</td>
<td>DEC</td>
<td>Serial</td>
</tr>
<tr>
<td>177562 (Spare)</td>
<td>DLV11-J</td>
<td>Interface</td>
</tr>
<tr>
<td>177560 (Terminal)</td>
<td></td>
<td>Card</td>
</tr>
<tr>
<td>177552 (No. 2)</td>
<td>ADAC</td>
<td>Digital output</td>
</tr>
<tr>
<td>177550 (No. 1)</td>
<td>1632HCO</td>
<td>Interface card</td>
</tr>
<tr>
<td>177172</td>
<td>DEC</td>
<td>Floppy disk</td>
</tr>
<tr>
<td>177170</td>
<td>RXVII</td>
<td>Interface</td>
</tr>
<tr>
<td>176756 (Ch 4)</td>
<td>ADAC</td>
<td>Four Channel</td>
</tr>
<tr>
<td>176754 (Ch 3)</td>
<td></td>
<td>Channel</td>
</tr>
<tr>
<td>176752 (Ch 2)</td>
<td>1412DA-4</td>
<td>Digital-to- Analog</td>
</tr>
<tr>
<td>176750 (Ch 1)</td>
<td></td>
<td>Converter</td>
</tr>
<tr>
<td>173777</td>
<td>*********</td>
<td>31K words</td>
</tr>
<tr>
<td>173776</td>
<td>DEC</td>
<td>ROM</td>
</tr>
<tr>
<td>.</td>
<td>REVll</td>
<td>Bootstrap</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>173000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>170422 (BPR)</td>
<td>DEC</td>
<td>Real-time</td>
</tr>
<tr>
<td>170420 (CPR)</td>
<td>KWV11-A</td>
<td>Programmable clock</td>
</tr>
<tr>
<td>167777</td>
<td>*********</td>
<td>30K words</td>
</tr>
<tr>
<td>167774 (INP)</td>
<td>DEC</td>
<td>Sixteen-bit</td>
</tr>
<tr>
<td>167772 (OUT)</td>
<td>DRV11</td>
<td>Parallel interface to ADC</td>
</tr>
<tr>
<td>167770 (DRCSR)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>165776</td>
<td>DEC</td>
<td>ROM</td>
</tr>
<tr>
<td>.</td>
<td>REVll</td>
<td>Bootstrap</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>165000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>163777</td>
<td>*********</td>
<td>29K words</td>
</tr>
</tbody>
</table>
APPENDIX E

SPECIAL TIMING CONSIDERATIONS FOR DAC04
APPENDIX E
SPECIAL TIMING CONSIDERATIONS FOR DAC04

The number of channels scanned, the number of data values averaged and the clock interrupt period together dictate the overhead on the computer (i.e., CPU). The user must insure that all clock interrupts are serviced, all writes to disk are performed, and that ample time remains to perform desired jobs in background.

To permit sufficient time for disk writes, it is recommended that the period between disk writes be set to 1 min or longer. This period, T, can be calculated as follows:

\[ T = tAS \]

where:
- \( T \) = disk write period in seconds
- \( t \) = clock interrupt period in seconds
- \( A \) = number of words averaged
- \( S \) = number of complete scans per disk block as calculated in Section 5.1 and shown in Table 6.

For example, a clock interrupt period of 10 msec, with 600 averages taken and 24 channels scanned, would give a 1-min disk write interval. In special cases, periods as short as 15 sec have been utilized. However, all background operations anticipated should be tested under these conditions before committing to a test.

A second criterion on the operation of DAC04 relates to the CPU overhead. It is evident that the more channels scanned the longer the program spends in the clock interrupt service routine which, therefore, increases the computer overhead. Therefore, for each particular number of channels scanned there is a clock interrupt period below which DAC04 will not allow background programs to run. Figure E.1 shows test data obtained during development for a disk write period of 20 sec. Based upon these data, a clock interrupt period equal to or greater than 10 msec is recommended. In special cases the graph may serve as a guide for the selection of a shorter period; however, the proper operation of DAC04 should be verified before committing to a test.
FIGURE E.1 DAC04 Operating Regimes
REFERENCES


<table>
<thead>
<tr>
<th>No. of Copies</th>
<th>No. of Copies</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OFFSITE</strong></td>
<td><strong>ONSITE</strong></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DOE Chicago Patent Group</td>
<td>DOE Richland Operations Office</td>
</tr>
<tr>
<td>9800 South Cass Avenue</td>
<td>H. E. Ransom</td>
</tr>
<tr>
<td>Argonne, IL 60439</td>
<td></td>
</tr>
<tr>
<td>A. A. Churm</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>39</td>
</tr>
<tr>
<td>DOE Division of Basic Energy Science</td>
<td>Pacific Northwest Laboratory</td>
</tr>
<tr>
<td>Washington, D.C. 20545</td>
<td></td>
</tr>
<tr>
<td>D. K. Stevens</td>
<td>E. R. Bradley</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
</tr>
<tr>
<td>DOE Technical Information Center</td>
<td></td>
</tr>
<tr>
<td></td>
<td>J. L. Brimhall</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>University of Washington</td>
<td></td>
</tr>
<tr>
<td>Seattle, WA 98195</td>
<td>L. A. Charlot</td>
</tr>
<tr>
<td>D. H. Polonis</td>
<td>E. L. Courtright</td>
</tr>
<tr>
<td>R. G. Stang</td>
<td>C. R. Hann</td>
</tr>
<tr>
<td>W. G. Weitkamp</td>
<td>C. H. Henager, Jr. (2)</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>T. C. Reilley</td>
<td>P. L. Hendrick (15)</td>
</tr>
<tr>
<td>Oak Ridge National Laboratory</td>
<td>J. L. Hooper</td>
</tr>
<tr>
<td>P.O. Box X</td>
<td>R. H. Jones</td>
</tr>
<tr>
<td>Oak Ridge, TN 37830</td>
<td>H. E. Kissinger</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>F. V. Nolfi</td>
<td>R. P. Marshall</td>
</tr>
<tr>
<td>Argonne National Laboratory</td>
<td>R. D. Nelson</td>
</tr>
<tr>
<td>9700 South Cass Ave.</td>
<td>N. J. Olson</td>
</tr>
<tr>
<td>Argonne, IL 60439</td>
<td>C. Pavloff</td>
</tr>
<tr>
<td></td>
<td>E. P. Simonen</td>
</tr>
<tr>
<td></td>
<td>T. J. Whitaker (2)</td>
</tr>
<tr>
<td></td>
<td>Technical Information (5)</td>
</tr>
<tr>
<td></td>
<td>Publishing Coordination (2)</td>
</tr>
<tr>
<td></td>
<td>Goo</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Hanford Engineering Development Laboratory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>J. L. Straalsund</td>
</tr>
</tbody>
</table>