DEVELOPMENT OF A DUAL-COMPUTER SYSTEM EXECUTIVE AND APPLICATION OF THE SYSTEM TO ANALYSIS OF EXTRUDED PLASTIC SHEET

A DISSERTATION

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Peter Nevius Bartram, B.S., M.S.

* * * * *

The Ohio State University

1974

Reading Committee:
Professor Edward J. Freeh
Assnt. Prof. John T. Heibel
Assoc. Prof. R. Emerson Lynn

Approved By

Adviser
Chemical Engineering Department
ACKNOWLEDGMENTS

The writer gratefully acknowledges the support of many companies in providing materials and equipment used in this project. Notably, the Digital Equipment Corporation (Maynard, Massachusetts) donated some of the computer equipment and the RSX-Plus monitor system. Permission to reprint parts of copyright material in this work was also granted by Digital Equipment. The Beta Gauge used in Part II of this study was donated by Industrial Nucleonics (Columbus, Ohio). The NRM Corporation (Akron, Ohio) provided the extruder.

With no less gratitude, the writer thanks those who have provided for his personal financial well-being while a graduate student:
1. The people of the State of Ohio, by providing employment as a teaching assistant for eleven academic quarters and university fee waiver for thirteen. 2. The people of the United States of America, through "G. I. Bill" legislation. 3. The Shell Companies Foundation, for fellowship support for two quarters. 4. The Diamond Shamrock Company for fellowship support for one summer.

This project would not have succeeded without the help of many individuals, who are being thanked in a more personal manner.
VITA


1967 ........................ B.S., Michigan State University, East Lansing, Michigan

1967-1968 .................. Teaching Assistant, Department of Chemical Engineering, The Ohio State University, Columbus, Ohio

1969 ........................ M.S., The Ohio State University, Columbus, Ohio

1969-1970 .................. Chemical Engineering Assistant, U. S. Army Aeromedical Research Laboratory, Fort Rucker, Alabama

1970 ........................ Instructor (Part Time), Troy State University, Fort Rucker Campus, Fort Rucker, Alabama

1971-1973 .................. Teaching Associate, Department of Chemical Engineering, The Ohio State University, Columbus, Ohio

PUBLICATION


FIELDS OF STUDY

Major Field: Chemical Engineering

Studies in Hybrid Computation. Professors H. C. Hershey and E. J. Frech

Studies in the Application of Computers and Engineering Techniques in Medical Research. Major E. H. Blackstone, M.D.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>ii</td>
</tr>
<tr>
<td>VITA</td>
<td>iii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vii</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
</tbody>
</table>

## PART I. THE DUAL-PROCESSOR COMPUTER SYSTEM

### Chapter

1. THE DUAL-PROCESSOR COMPUTER HARDWARE ....................... 5
2. THE DUAL-PROCESSOR OPERATING SYSTEM .......................... 10
3. DUAL-PROCESSOR SYSTEM EVALUATION AND RECOMMENDATIONS ... 27

## PART II. APPLICATION OF THE DUAL-PROCESSOR SYSTEM
IN THE STUDY OF EXTRUDED SHEET THICKNESS

4. OBJECTIVES OF THE STUDY AND DESCRIPTION OF EQUIPMENT
   USED ................................................................................. 33
5. COMPUTATIONAL TECHNIQUES ........................................... 39
6. EXPERIMENTAL PROCEDURE ............................................ 50
7. RESULTS AND DISCUSSION ............................................. 55

## PART III. SUMMARY OF CONCLUSIONS AND RECOMMENDATIONS

8. CONCLUSIONS ............................................................... 77
9. RECOMMENDATIONS ....................................................... 79
TABLE OF CONTENTS—Continued

APPENDIX

A. SYSTEM PROGRAM DESCRIPTIONS AND START-UP PROCEDURE .......... 82
B. DUAL-PROCESSOR SYSTEM PROGRAM LISTINGS ......................... 91
C. PLASTIC SHEET THICKNESS STUDY COMPUTER PROGRAM LISTINGS ......................................................... 223
D. ADDITIONAL INFORMATION ON EQUIPMENT CHARACTERISTICS .... 261

LIST OF REFERENCES ................................................................................................................ 267
LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dual-Processor System Summary</td>
</tr>
<tr>
<td>2</td>
<td>Run Conditions</td>
</tr>
<tr>
<td>3</td>
<td>Fortran Real-Time Subroutine Calls</td>
</tr>
<tr>
<td>4</td>
<td>Calibration Constants</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dual-Processor Configuration</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Secondary Processor Priority Structure</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Example Fortran Program for Real-Time Data Acquisition</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>Schematic Representation of the Extruder and Chill Roll Assembly</td>
<td>35</td>
</tr>
<tr>
<td>5</td>
<td>Sheet Thickness Profile</td>
<td>56</td>
</tr>
<tr>
<td>6</td>
<td>PSDF for Plastic Sheet</td>
<td>59</td>
</tr>
<tr>
<td>7</td>
<td>PSDF for Plastic Sheet</td>
<td>60</td>
</tr>
<tr>
<td>8</td>
<td>Pressure PSDF, Run 2</td>
<td>65</td>
</tr>
<tr>
<td>9</td>
<td>Pressure PSDF, Run 3</td>
<td>66</td>
</tr>
<tr>
<td>10</td>
<td>Temperature PSDF, Run 2</td>
<td>67</td>
</tr>
<tr>
<td>11</td>
<td>Temperature PSDF, Run 3</td>
<td>68</td>
</tr>
<tr>
<td>12</td>
<td>PSDF Comparison</td>
<td>73</td>
</tr>
<tr>
<td>13</td>
<td>Single-Stage Filter Response</td>
<td>262</td>
</tr>
<tr>
<td>14</td>
<td>Three-Stage Filter Response</td>
<td>263</td>
</tr>
<tr>
<td>15</td>
<td>Beta Gauge Calibration</td>
<td>265</td>
</tr>
</tbody>
</table>
INTRODUCTION

If one computer can do a job adequately, then two computers should be able to do the job better. In this work, a description will be given of how this idea has been implemented by the Chemical Engineering Department of The Ohio State University. The system outlined in Table 1 and described in the first part of this work is a dual central processor configuration. With one of the processors dedicated to handling a real-time interface and some time-consuming peripherals, the other processor is more fully utilized in the execution of the user programs. The operating system for this configuration is a modification of a monitor written by the hardware vendor for a single-processor system. The modifications made are reported in Part I of this paper and program listings are given in Appendix B. Appendix A contains information needed to operate the system.

This system was designed to be as versatile as possible in handling the needs of users having minimal experience in computer programming. It is rare that assembler language programming would be required of the user; however, it is assumed he knows Fortran. Yet, a wide variety of real-time data acquisition and control functions can be performed on a priority time-shared and memory-shared basis.

Part II of this paper deals with an example of a practical application of this system. Plastic sheet produced by extrusion is not


TABLE 1
DUAL-PROCESSOR SYSTEM SUMMARY

<table>
<thead>
<tr>
<th>Purpose</th>
<th>To provide a multi-user computer facility for laboratory and process data acquisition and control and for student instruction in the use of such equipment.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
<td>2 PDP-15 computers (primary processor, secondary processor) with shared memory and interrupt links.</td>
</tr>
<tr>
<td>Peripherals</td>
<td>Disks (2), fixed head.</td>
</tr>
<tr>
<td></td>
<td>DECTape transports (3).</td>
</tr>
<tr>
<td></td>
<td>Teleprinters (3).</td>
</tr>
<tr>
<td></td>
<td>Character generator video display and keyboard.</td>
</tr>
<tr>
<td></td>
<td>Graphics video display.</td>
</tr>
<tr>
<td></td>
<td>Line printer (600 lines per minute).</td>
</tr>
<tr>
<td></td>
<td>Paper-tape reader/punch (high speed).</td>
</tr>
<tr>
<td></td>
<td>Analog-to-digital converter (64 channels).</td>
</tr>
<tr>
<td></td>
<td>Digital-to-analog converters (5).</td>
</tr>
<tr>
<td></td>
<td>Digital inputs and outputs.</td>
</tr>
<tr>
<td>Executive</td>
<td>Primary processor: RSX-Plus, modified.</td>
</tr>
<tr>
<td></td>
<td>Secondary processor: dedicated code.</td>
</tr>
<tr>
<td>Features supported</td>
<td>Multi-user real-time service.</td>
</tr>
<tr>
<td></td>
<td>Time sharing on a priority basis.</td>
</tr>
<tr>
<td></td>
<td>Repetitive scheduling of user programs.</td>
</tr>
<tr>
<td></td>
<td>Program development simultaneously with real-time usage.</td>
</tr>
<tr>
<td></td>
<td>Overlapped input/output handling.</td>
</tr>
<tr>
<td>Time frame</td>
<td>Primary processor: 60 Hz.</td>
</tr>
<tr>
<td></td>
<td>Secondary processor: 1000 Hz.</td>
</tr>
<tr>
<td>Programming languages</td>
<td>Fortran 4 with extended features.</td>
</tr>
<tr>
<td></td>
<td>Macro assembly language.</td>
</tr>
<tr>
<td>Principal advantages over a single-processor system</td>
<td>Greater throughput through reduced overhead on the primary processor.</td>
</tr>
<tr>
<td></td>
<td>Better multi-user real-time service.</td>
</tr>
<tr>
<td></td>
<td>Reduced cost/performance ratio.</td>
</tr>
</tbody>
</table>
of uniform thickness. The variations in thickness of some sheet produced by the Chemical Engineering Department have been examined using the dual-processor system. Measurement of the thickness of the sheet was by a radiation-based instrument, the output of which is automatically obtained by the computer system through the real-time interface. Monitoring of temperature and pressure in many places in the extruder was also performed by the computer as the sheet was produced. The data so obtained were analyzed using such computational techniques as the fast Fourier transform. All programming for this was done in Fortran to demonstrate the usefulness of the system to inexperienced programmers.
PART I

THE DUAL-PROCESSOR COMPUTER SYSTEM
CHAPTER 1

The Dual-Processor Computer Hardware

The dual-processor concept is an extension of the input-output processor idea implemented on most computers. Here a second computer, programmable (rather than having a hard-wired operating sequence), is used. The system described here has been described elsewhere by this writer, in a briefer, less detailed form. (Bartram, et al., 1973).


Figure 1 is a diagrammatic representation of the system configuration. The primary processor is well expanded, and processor options include: 1. the extended arithmetic element with hardware integer multiply and divide, 2. the memory protect and relocate unit, 3. the automatic priority interrupt (API) providing four priority levels for hardware generated interrupts and four software interrupt levels. The
FIGURE 1  DUAL-PROCESSOR CONFIGURATION
memory available to the primary processor consists of six pages (4092 words per page) to which it alone has access, and three pages which may be accessed by both processors through a memory multiplexer. A sixty Hertz (Hz) clock is available for timing by the software. Peripherals on the main processor are three TU55 DECtape transports, two RS09 disk platters, a 600 line per minute printer, three teletypewriters (one ASR-35 Teletype, two ASR-33 Teletypes), and a high-speed paper-tape read-punch.

The secondary processor is the same basic PDP-15 as the primary processor, but lacks the processor options and peripherals. The only processor option is API. The secondary processor has no memory, but shares three pages with the primary processor through the memory multiplexer. A faster clock, 10,000 Hz, is provided. The peripherals attached to this processor are non-standard, i.e. not supported by the Digital Equipment Corporation. Two Conrac Corporation (Covina, California) Model CVB 17 video display units are used, one driven by a Hughes Aircraft Company (Industrial Products Division, Oceanside, California) Model 639 Scan Conversion Memory for graphic-type output. The other is driven by a character generator, Model A^2T206 KSR Video Terminal Controller manufactured by Ann Arbor Terminals, Inc. (Ann Arbor, Michigan). This provides fifteen 80-character lines of output over the screen. A keyboard, also by Ann Arbor Terminals, as part of the Model A^2T206 Video Terminal Controller, is also controlled through the secondary processor as a system input device. Secondary processor programming links the keyboard input and the character generator output
functions so that to the user they are one device.

The real-time interface is the major unit under control of the secondary processor. A 64-channel multiplexed analog-to-digital converter, Raytheon Computer (Santa Anna, California) Model MADC15-04-64, provides for conversion of signals in the range -10.0 to +9.9994 volts (direct current). The converted value is a fifteen-bit twos complement word. While several modes of operation are possible for the converter, frame scan mode is used at this installation: Once initialization commands have been given, all sixty-four channels are serially converted and the values transferred to memory through the secondary processor data channel and input-output processor. Ten microseconds separate the conversion of each adjacent channel. Eight digital-to-analog converters are currently part of the real-time interface. Four are Burr-Brown Research Corporation (Tucson, Arizona) Model DAC20-10B-BTC ten-bit buffered bipolar twos complement converters; the other four are Burr-Brown Model DAC20-12B-BTC twelve-bit converters. All operate in the range -10.0 to +10.0 volts, and must be individually set by the secondary processor.

Currently one group of six digital inputs is used, these being connected to push buttons near the keyboard to function as sense switches. Three digital output groups are available. One is used with six lights as sense lights; the others are for driving relays operating process equipment or mode control for an analog computer. These discrete input and outputs are of standard M-Series logic cards by Digital Equipment Corporation.
All non-standard interfacing, for both the real-time interface and the input-output display devices on the secondary processor, was designed by Prof. J. T. Heibel of The Ohio State University Chemical Engineering Department. Logic cards and other components generally were manufactured by the Digital Equipment Corporation.
CHAPTER 2

The Dual-Processor Operating System

The software developed for the dual-processor configuration described in the previous section was designed to meet several goals:

1. It should allow for multiple users on a time-shared basis. 2. It should be versatile, allowing as much freedom as possible for the user to take advantage of available hardware. 3. It should be easy to use, requiring a minimum of special training. 4. It should be easy to modify as additional hardware is added to the system. 5. It should be as close as possible to a proven and documented system, to minimize development and check-out time so that full use of the system could be quickly realized. To meet these requirements, Digital Equipment Corporation's RSX-Plus Resource Sharing Executive system was chosen as a base. This operating system, modified as described, is used with the primary processor; the secondary processor, dedicated to handling the devices attached to it, is never programmed by the user, who need not be aware of its existence. Only a brief summary of the RSX-Plus system is given here; the reader is referred to the RSX-Plus manual (Digital Equipment Corporation, 1972) for a complete description.

RSX-Plus is a disk-based, time-shared operating system using fixed-sized memory partitions. User and monitor programs, called tasks, are
run on a priority schedule, with the priority being established by the operator when the task is scheduled, or in default, when the task is prepared. A task of given priority will be executed in preference to one of lower priority, but will be interrupted when the time for one of higher priority comes due. The exception is that once a program starts, no other task may be executed in that memory partition. The core-resident monitor requires the lowest 13000 (octal) memory cells. Additionally, some memory must go unassigned to task partitions to allow for the monitor to maintain certain tables. All other memory is broken into task partitions when the system is generated. The partitions used at this installation are given in Appendix A in the example of bringing up the system.

Some monitor functions are performed by special tasks, which are constructed and maintained in the same manner as user programs. Input-output device handlers are one such class of tasks, and are core-resident only when the device controlled is available for use. Separate partitions are usually dedicated to these tasks, though infrequently used device handlers such as those for the paper-tape reader and paper-tape punch share partitions, and only one may be used at any given time. The operator interaction with the monitor, for such purposes as task scheduling, input-output device assignments, and system information, requires a large number of tasks, usually executed from a dedicated partition. Program development tasks including a text editor, assembler, Fortran compiler, and task building routine, are another set of system tasks.
Much of the programming required for modifying the RSX-Plus executive for the dual-processor hardware was also structured into tasks. This allows for modification of the system without requiring a software reconfiguration: The revised code for the section being modified is built as a task and installed in the system as if it were a normal user task. The code required for communication with the secondary processor is fixed in core (i.e. is installed to be always core-resident, a standard monitor function used with some, but by no means all, user tasks). At system configuration time, or later if required, a program is executed which overwrites a few selected instructions of the basic monitor to provide a linkage to the modification code. The code for the secondary processor, though never executed by the primary processor, is similarly fixed in a dedicated partition in shared memory. The shared memory required for communication (flags and buffers) between the two processors appears as a standard program partition, 256 words long. A flag is set so that the partition appears to the monitor to be always occupied (and hence no user task can be loaded into that partition). The keyboard-character generator appears to the user as any other input-output device, and the portion of the handler required by the primary processor (to transfer buffers to and from the shared memory area and to communicate requests to the secondary processor) is built as any other device handler task. (The secondary processor section of the handler is combined with the rest of the secondary processor code.)

Advantages to making the modifications in this manner are many.
The operating system supplied by Digital Equipment is left intact, making easier any software corrections they may issue. The modification code is in small, neat pieces, which facilitated error tracing while the system was under development, as one did not have to worry about the operation of the large basic monitor. The excessive time and effort required to assemble the basic monitor code is avoided each time a minor modification is required. (The basic core-resident monitor has been assembled at this institution only once— to provide a source listing.) Finally, documentation is easier: A person familiar with the system as supplied has a relatively small amount of code to review to understand the modifications.

The program for the secondary processor is clock-based. The ten kiloHertz (kHz) clock provides two interrupts each millisecond. These are at level 3, the lowest hardware priority. Thus when they occur the API is disabled to lock out any higher priority interrupt. (Also, when the API is enabled and higher priority interrupts are received, the program immediately shifts them to a lower software priority. Thus should a clock interrupt occur, it will be processed without delay.) At the first of a pair of clock interrupts, the analog-to-digital converter is started, which updates a table of values in shared memory. While the analog-to-digital converter is transferring data via the data channel, digital input groups are read, and stored in shared memory, the digital output groups and digital-to-analog converters are set, using values from shared memory tables. A counter is incremented (serving as a one-millisecond clock counter), and the clock
is set to interrupt again 0.7 milliseconds after the first. The program then enables the API and performs less critical clock-based functions if required, or continues with its background function. Upon receipt of the second interrupt, the analog-to-digital converter is checked to assure that all sixty-four channels have been converted (which requires 0.642 milliseconds). If not, the secondary processor halts in a loop. With the successful completion of analog-to-digital conversion, the main processor is interrupted to give notice of the availability of fresh analog and digital input data. The clock is reset to interrupt 0.3 milliseconds after the previous clock interrupt, the API is enabled again, and control returns to the interrupted function. This process is repeated every millisecond.

The only other clock-based function presently implemented is for the character generator. The character generator operates character by character, and requires one millisecond between output commands. Thus, when output to the screen is in progress, or input from the keyboard (which is echoed on the screen), the character output commands are issued by a section of code entered through the clock interrupt handling programming. This output is performed at the lowest (software) priority, level 7. Provision has been made for other clocked functions to be added, and the level 7 priority is broken into sub-levels in the secondary processor program.

After completing any time-based operations, the secondary processor can perform non-time-critical functions, such as generating output for the scan converter for the graphic display screen. Requests for input-
output service (through the character generator and keyboard) from the primary processor are set up. The primary processor signals the need for service by interrupting the secondary processor, which determines the required function through a flag table in shared memory. The only other interrupt to which the secondary processor responds (when not involved with time-based operations) is that of the keyboard, signaling the availability of a character in the keyboard register. The secondary processor reads the character, echoes it to the character generator, and places it in an input buffer. When an input line is completed, the secondary processor packs the buffer (if required), and sets a flag in shared memory to signal to the primary processor keyboard-character generator handler of the availability of an input line.

Figure 2 shows the priority of operations by the secondary processor. When the machine is not involved in any servicing of clock or primary processor functions, it increments the accumulator. The frequency of change of state of the accumulator overflow light is an indication of the amount of idle time which could be used by the secondary processor for additional functions.

The modifications to the RSX-Plus executive for the primary processor deal with usage of the real-time interface and the other secondary processor peripherals. The graphics video-display and keyboard-character generator are used as any other peripheral device, by means of an input-output (I/O) handler task. This task processes requests passed to it through the monitor I/O queue. When service by
FIGURE 2 SECONDARY PROCESSOR PRIORITY STRUCTURE
the secondary processor is required, an interrupt is issued to the secondary processor, after a flag has been set in the shared memory table to indicate the required function. The handler then waits for a flag to be set by the secondary processor signaling completion of the request.

The most significant primary processor modification is in handling the real-time interface. As indicated previously, using tables in the shared memory the secondary processor refreshes the digital output groups and digital-to-analog converters and stores analog-to-digital converter and digital inputs every millisecond. Since most user programs cannot reference these shared memory tables (or any other part of memory outside the partition for that task), the requests must be handled by the monitor rather than directly by the user program. All monitor functions (called CAL functions) are requested by the user task by issuing a particular PDP-15 instruction (CAL) with the argument pointing to a CAL parameter block to indicate to the monitor the desired function. For the real-time interface, four new CAL functions were defined.

For real-time outputs the CAL parameter block consists of the CAL function code (32), an event variable address, the real-time output device number (digital-to-analog converters have positive numbers, digital output groups have non-positive device numbers), and the value for that device, properly coded. The event variable is a coded error indicator used for all CAL functions. Usage here corresponds to that of the rest of the system. The CAL function checks to insure that the
digital-to-analog converter value, which contains the device number, is properly coded (to prevent the wrong converter from being set by the secondary processor).

Since several user programs can be operating at the same time, it is necessary to offer a means of real-time output device protection. Safety and other considerations require that it be difficult for a control valve, for example, to be changed accidentally through a program error (such as a mis-typed output device number) of a person not intending to use that piece of equipment. Yet, a person should be able to use a given real-time output without undue hardship. The protection scheme implemented is based on a table of three-character codes, with one code for each real-time output. A new system task, LPR, lists the codes currently in force. To change the code, another new system program, PRO, is called. For example, to change the protection code of real-time output device 2 (a digital-to-analog converter) from "N-A" to "EJF", one must enter the device number, new code, and old code on the monitor teleprinter as follows:

```
MCR>PRO 2 EJF N-A
```

(The format for this is similar to the input/output device number reassignment program, REA.)

The real-time output CAL function checks the protection code for the specified output device every time a real-time output is requested. The code must be identical to the first three characters of the name of the task (program) which called for the output. This provides for the
capability of more than one task setting a device if the user so desires, by use of the same characters for the first half of his task names. Yet it is unlikely that another user would choose these same characters in his task name. The code "N-A" is special. If the protection code table entry for a particular device is "N-A", any program, regardless of task name, may set the output device.

The other three new CAL functions deal with transferring real-time inputs from the shared-memory tables to the user task. Recognizing the wide difference in data sampling rates required by various users, two methods have been provided. Real-time inputs are the analog-to-digital converters (positive real-time input device numbers), digital input groups (negative device numbers), and the fast clock counter incremented by the secondary processor every millisecond (device number zero). For the user requiring real-time inputs infrequently, (sample period measured in seconds or minutes), CAL function 31 is used. The CAL parameter block for this function contains the function number (31), an event variable address, the input device number, and a blank cell to which the monitor will transfer from shared memory the value of the input. The frequency of sampling is determined by the user by how often the CAL function is issued. Usually, this timing will be established by scheduling repetitive execution of the program at the desired frequency, and issuing the directive once per program execution.

For users requiring faster data sampling, up to every millisecond, the monitor can transfer values from the shared memory tables to a user specified buffer automatically. This requires that the user task be
fixed in core, but does not require that the program be executing, to receive the data. Thus, while no other user can share the partition, time sharing is effective. Three tables are maintained by the (modification to the) monitor, one each for ten Hertz (Hz), one hundred Hz, and one kiloHertz (kHz) sampling frequency. Each table entry consists of four words: the address in the user buffer for the next value to be transferred, the input device number, the last address of the user buffer, and the first address of the buffer. Each table may contain up to fifteen entries. Upon interrupt by the secondary processor, every millisecond, the monitor goes through the one kHz table. For each entry, the value for the specified input device is transferred from the shared memory table to the user buffer. The address is incremented, unless it is the last address of the buffer, in which case it is set to the starting address. Thus the user buffer is filled in a ring fashion. In a similar manner, the hundred Hz table is handled every tenth interrupt from the secondary processor, and the ten Hz table every hundredth interrupt. This arrangement provides for data sampling at frequencies in excess of the monitor clock, and also allows filling of user data buffers while another task is being executed. This provides for minimum overhead in the time-shared environment.

To establish an entry in one of the three tables, the user must issue another new CAL function request. The CAL parameter block is six words long; required is the CAL function code (33), an event variable address, the real-time input device number, the starting address of the buffer, the length of the buffer, and a code for the sampling frequency
desired. The CAL function checks to insure that the issuing task is fixed in core and that the buffer is wholly within the task partition, before adding the entry to the appropriate table.

For the user having multiple entries in these tables, the possibility exists of an interrupt from the secondary processor between successive issuings of this CAL directive. Therefore the fourth new CAL directive is provided to reset all addresses (first word of the table entry) for the issuing task to the starting addresses of the buffers, thus synchronizing all inputs. The CAL parameter block consists of the function code (34) and an event variable address. The event variable for this function gives a factor for determining synchronization of inputs of different sampling frequency.

To remove entries from the tables, the CAL directive for unf ixing a task from core was modified. Whenever this CAL function is requested of the monitor, all table entries the task had set are removed. The system task "...REM" (remove) has been similarly modified. Thus there is no possibility for another task to be loaded into memory while the monitor is still transferring real-time inputs to locations in that partition. However, if a task is segmented and the buffer specified is to be overlayed by another segment of the program, the user must be sure the unf ix CAL directive has been issued before calling in the next overlay, lest the transfer continue, overwriting the new program segment.

User implementation of the automatic real-time data gathering requires a buffer size—execution time optimization decision. Usually,
the user program considers each input buffer as two or three sections, setting the last word in each section to zero. The program then waits for the word to become non-zero, indicating that section of the buffer has been filled. The program then transfers to data elsewhere, usually to disk, sets the last word to zero again, and waits for the filling of the next section. Since the overall buffer is ring-filled, the process is repeated until the user program is satisfied that sufficient data have been collected. Sufficient time must be allowed to transfer the data from the buffer section before it is overwritten the next time through. Time must be allowed for the monitor function used to sense the value becoming zero. (The frequency of checking this is one second, unless the monitor is forced to check more often.) The user should be familiar with the concept of a "significant event", and WAIT FOR and MARK TIME CAL functions as described in the RSX manual (Digital Equipment Corporation, 1972). It is noted that all real-time inputs, including the clock counter, will always have at least one non-zero bit. (The clock counter is incremented by two, and is always an odd number.) There is no possibility of an input stored in the last word of a section of the buffer not being recognized as a change from the zero stored there previously by the user program. The clock counter is an effective input for checking the timing of a program. If data are lost by allowing insufficient time to transfer them from the buffer, the use of the clock as input makes this very apparent.

All of the monitor modifications for the real-time interface are designed to be used by a Fortran programmer. The definition of the
automatically filled real-time input buffer is with the DIMENSION (or more often, the COMMON) statement. The following subroutines have been added to those already provided in the system library to issue the CAL directives: 1. RTSET is used to set an entry in the automatic input request table. (CAL 33). 2. RTRST is for synchronizing the inputs requested with RTSET (CAL 34). 3. RTIN returns the values of specified input groups by issuing requests for CAL 31. These values may be raw data words or, for analog inputs, floating point numbers. 4. RTOUT issues requests for real-time outputs (CAL 32). For analog outputs, the formatted (integer) value may be used. 5. Subroutine RTIO is used when both CAL 31 and CAL 32 are required by the same task. Entries are RTINX for CAL 31 and RTOUX for CAL 32. A savings in program size results by using this combined subroutine. 6. ADCNV is used to convert analog-to-digital converter values from the raw word as stored by the automatic buffer filling (or the integer form returned by RTIN and RTINX) to floating point. Entry IADCV of this subroutine converts the raw values to scaled integers. The Fortran programmer should have little difficulty in coding the digital output words or uncoding digital inputs.

Subroutines for this have not been provided: Parameter fetching would make them much less efficient than the Fortran code to accomplish the same thing. The keyboard-character generator handler is program compatible with other I/O device handlers, and requires no special consideration.

To clarify the use of the automatic filling of user data buffers by the monitor, an example Fortran program explaining the methods
discussed above is given as Figure 3. In the example, library sub-
routine WAITFR is a call to the monitor to suspend operation of the
program until the specified variable becomes non-zero. Note that this
function is also used in conjunction with some standard input/output
functions such as creating disk files. While waiting for the specified
variable to be set to a non-zero value, execution of a lower priority
program may take place, making time-sharing viable even when one
program requires real-time data at fast sample rates.

The reader is referred to Appendix B for a more detailed descrip-
tion of the secondary processor code (MOD10F), the monitor modifications
(.PATCH), the system task to link the modifications to the original
monitor (DPP.02), the system tasks for the real-time output protection
code table (LPR.01 and PRO.01), the modification to the "remove" task
(REM.05), the keyboard-character generator handler task (KSG), and the
library routines (RTSET, RTRST, RTIO, RTIN, RTOUT, and ADCNV). The
listings of these programs contain the details of their use as well as
their organization and logic flow.
EXAMPLE PROGRAM TO OBTAIN 4233 DATA VALUES AT 100 Hz
SAMPLE FREQUENCY FOR EACH OF TWO ANALOG-TO-DIGITAL
CONVERTER CHANNELS. THIS TASK MUST BE FIXED IN CORE.

THE TWO BUFFERS INTO WHICH THE MONITOR WILL STORE THE
DATA ARE "IDA" AND "IDB".

COMMON IDA(400), IDB(400)

ANALOG-TO-DIGITAL CONVERTER CHANNELS, "ICA" AND "ICB",
ARE 32 AND 33. THE FREQUENCY CODE, "IFQ", IS 2 FOR
100 Hz.

DATA ICA/32/, ICB/33/, IFQ/2/

OPEN THE DISK FILE, TO BE NAMED "SAMPL DAT", FOR
STORING THE DATA. (SEE THE RSX MANUAL FOR FILE-
STRUCTURED BULK STORAGE USAGE.)

CALL ENTER (60, 5HSAMPL, 3HDAT, IEV)
CALL WAITF (IEV)
IF (IEV) 101, 101, 102
WRITE (63,1) IEV
1 FORMAT (44H ERROR IN OPENING DISK FILE, EVENT
1 VARIABLE: , I5)
CALL EXIT

REQUEST THE MONITOR TO AUTOMATICALLY FILL THE BUFFERS
(RTRST) AND INSURE TIME SYNCHRONIZATION (RTRST).
"IEV" IS THE EVENT VARIABLE (ERROR RETURN CODE) NAME.

102 CALL RTSET (ICA, IFQ, IDA)
CALL RTSET (ICB, IFQ, IDB, IEV)
IF (IEV) 103, 103, 104
103 WRITE (63,2) IEV
2 FORMAT (25H ERROR RETURNED BY RTSET: , I5)
CALL EXIT
104 CALL RTRST (IEV)
IF (IEV) 105, 105, 106
105 WRITE (63,3) IEV
3 FORMAT (25H ERROR RETURNED BY RTRST: , I5)
CALL EXIT

ZERO OUT THE LAST ELEMENT OF EACH HALF OF ONE OF THE
BUFFERS, AND USE THOSE ELEMENTS FOR TESTING FOR THE
PROGRESS OF THE FILLING OF THE BUFFER. AS SOON AS
ONE HALF OF EACH BUFFER IS FILLED, STORE IT ON DISK.
FOR 4233 POINTS FROM EACH ANALOG-TO-DIGITAL
CONVERTER CHANNEL, THE BUFFERS WILL BE FILLED TEN
TIMES.

FIGURE 3 EXAMPLE FORTRAN PROGRAM FOR REAL-TIME DATA ACQUISITION
106   IDA(200) = 2
107   ICY = -10
112   IDA(400) = 3
C
C WAIT FOR THE FIRST HALF TO BE FILLED.
C
CALL WAITFR (IDA(200))
C
FIRST HALF FILLED. STORE ON DISK.
C
WRITE (60) (IDA(I), IDB(I), I=1,200)
IDA(200) = 0
C
C WAIT FOR THE SECOND HALF TO BE FILLED.
C
CALL WAITFR (IDA(400))
C
SECOND HALF FILLED. STORE ON DISK.
C
WRITE (60) (IDA(I), IDB(I), I=201,400)
C
IF MORE DATA ARE NEEDED, CYCLE THROUGH AGAIN,
OTHERWISE, CLOSE THE DISK FILE WITH THE DATA,
AND EXIT.
C
ICY = ICY + 1
IF (ICY) 112, 111, 111
C
DATA COLLECTION FINISHED. CLOSE THE DISK FILE
AND EXIT.
C
111   CALL CLOSE (60, SHSAMPL, 3HDAT, IEV)
112   CALL WAITFR (IEV)
113   IF (IEV) 113, 113, 114
114   WRITE (63,4) IEV
4   FORMAT (44H ERROR IN CLOSING DISK FILE, EVENT
1   VARIABLE:, I5)
114   CALL EXIT
END

FIGURE 3--Continued
CHAPTER 3

Dual-Processor System Evaluation

and Recommendations

It is possible to obtain only a rough estimate of the savings in time resulting from use of the secondary processor. A single-processor system probably would not operate the real-time interface continuously at a one kHz rate if it were at the expense of main processor time. (Though the method used assures every user equally-spaced data.) Some functions performed by the secondary processor, such as packing and unpacking character buffers for the keyboard-character generator, could be done faster on the main processor with its extended arithmetic element. (Though the overall time is less if the main processor does not perform them at all.) Another factor is machine cycle time: When either machine references shared memory, the instruction time is longer. When both reference shared memory together, (which is whenever the main processor does, as the secondary processor code is resident in shared memory) instruction time is slowed even more. Thus without the secondary processor, the main processor could often run faster.

One indication of the effectiveness of the dual-processor system is the latency (i.e. time spent waiting for something to do) of the
secondary processor. This latency is estimated, based on observation of console lights (as discussed previously, when idle, the secondary processor changes the state of the overflow indicator), at 35 to 40 per cent. Much of this 60 to 65 per cent utilization would have to be performed by the single processor if the secondary processor were unavailable.

Ideally, the secondary processor would have 95 to 105 per cent utilization. (That is, at times non-critical functions would be noticeably delayed, insuring overall full use of the machine.)

Probably the best function performed by the primary processor to shift to the secondary processor is the handling of the teleprinters. This has not been done as the code for the teleprinters is an integral part of the monitor (not a separate task as with other peripherals), and a large programming effort would be required for both the secondary processor and the RSX-Plus monitor. A major hardware change would also be required to tie the multiple teleprinter controller to the secondary processor. Nevertheless, it is recommended that eventually this be done to improve system efficiency.

It has been observed that much of the time a program is core resident it is waiting for input or output. Some improvement in core usage could be obtained if a small output spooling program were implemented for the teleprinters.

The DECtape handler for RSX as provided by Digital Equipment is inadequate. It should be rewritten, probably with overlayed segments to conserve memory, to implement the file deletion directive and to
permit more than one tape unit to be used at one time. One problem now is that the buffer is part of the handler. It would be better if the tape buffers were in each user's partition, as with the disk file handler.

Graphic output remains the largest area of under-utilization of available resources, both hardware and software. Through the generosity of Dr. E. H. Blackstone (1972), a time-series package has been made available which includes routines for three dimensional and contour plots. These graphics routines should be modified and implemented for use with the graphics video display and the small analog plotters available which can be operated with the digital-to-analog converters.

Additional hardware would greatly enhance the system. System performance would be improved if a second interrupt line from the secondary to the primary processor were installed. This would be dedicated to the keyboard-character generator handler, and would save the overhead currently involved in waiting for shared memory flags to be set. Minor modifications in the primary and secondary processor keyboard-character generator handler code would be required.

Currently, the ten kHz clock for the secondary processor is a stand-alone unit, and should be replaced by a clock which would be an integral part of the central processor unit of the secondary machine. No change in programming would be required.

A weakness of the real-time interface is the small number of digital input and output groups. These should be expanded to provide
several general purpose input groups and several outputs driving relays. Currently the analog-to-digital converter is being used in several cases where a discrete input is more appropriate, and either a teleprinter or digital-to-analog converter is used where a few discrete outputs would serve as well. This expansion would be the best way to alleviate the shortage of analog inputs and outputs which has been occurring. Also, three additional digital-to-analog converters should be installed and dedicated to the video display unit. The present practice of having this unit share with the general purpose digital-to-analog outputs can cause problems, both from a shortage standpoint and because of the real-time output protection methods. The secondary processor refreshes these three digital-to-analog converters every millisecond as it does the other general purpose converters, but the graphics handler requires a much greater frequency, and thus also sets these converters. Only minor programming changes would be required, and the programs have been written with these changes in mind. Adding digital input and output groups would require only a few moments editing of the secondary processor code, the protection system implementation code, and the new CAL function (monitor modification) code. The changes are clearly indicated in the listings given in Appendix B.

The addition of more memory would require no changes in programming. This would greatly extend the system usefulness, as currently the user partitions are under-sized, and too few in number. Any additional peripherals must be accompanied by additional memory for
their handler tasks if they are to be used effectively. It would be a severe handicap on the system to cut down on any user partition size or for additional peripheral handlers to share I/O handler partitions with current peripherals.

It has been recommended that an industry compatible tape drive and card reader be obtained. This writer does not concur with the need for a card reader, feeling that the implementation of this peripheral, and the batch task development tasks which would make it useful, would require memory which could not be spared. Even if additional memory were made available, the tendency would be to use the system for batch computation better performed on other university computers, at the detriment of the real-time usage for which the dual-processor system was designed. However, the need for an industry compatible tape drive, or some other method of transferring data collected by real-time usage of the dual-processor system to other systems, is very real. Currently, computation such as much of that described in Part II of this work, which could be performed better on larger computers, is done on this system because of the difficulties involved in transferring large amounts of data to other machines.
PART II

APPLICATION OF THE DUAL-PROCESSOR SYSTEM IN THE STUDY
OF EXTRUDED PLASTIC SHEET THICKNESS
Extruded plastic sheet usually is not uniform in thickness. By proper adjustment of the die and cooling conditions, most cross-sheet (direction of the width) variation can be eliminated except at the edge. However, considerable machine direction (perpendicular to the width) variation may persist. It was the purpose of this study to demonstrate the dual-processor system described in Part I by analyzing the machine direction variations in sheet thickness and to attempt to relate these variations to processing conditions. Conditions examined included temperatures and pressures in the die and in various parts of the extruder.

The extruder used is described in detail by Reber (1971), Springer (1973), and Fontaine (in preparation). It is a two-and-one-half-inch, single-screw Spacemaker model by the NRM Corporation (Akron, Ohio). The die used is a 26-inch, heated Johnson sheet die. Between the die and the extruder is a viscometer and an adaptor, the adaptor being closer to the extruder.

Instrumentation of the extruder equipment is extensive. Interfaced with the secondary processor of the computer system are six
thermocouples across the die near the lip, a zero to 1000 pound per square inch (psi) pressure transducer in the die, another thermocouple in the die entrance, a thermocouple near the exit of the viscometer, two pressure transducers in the viscometer (0-1500 psi near the exit, 0-3000 psi near the entrance), a thermocouple and pressure transducer (0-3000 psi) in adaptor near the exit, a 0-5000 psi pressure transducer at the head plate (exit) of the extruder, and seven thermocouples along the barrel of the extruder. Additionally, as an electronic noise level reference, a thermocouple interfaced to the computer is left in a container of ice and water, and a 0-500 psi pressure transducer is left in the open. The thermocouples are special plastic extruder melt thermocouples manufactured by the West Instrumental Division of Gulton Industries, Inc. (Schelter Park, Illinois); the pressure transducers were made by the Dynisco Division of Microdot Inc., (Westwood, Massachusetts), models PT-422, PT-420, and TPT-432. Details of the instrumentation of the extruder are given by Fontaine (in preparation). A schematic diagram showing placement of these instruments is given as Figure 4.

The signal from the thermocouples and pressure transducers is amplified to the range of the analog-to-digital converter. Additionally, a "suppression box" is used with the thermocouples to provide a reference junction and voltage offset for the signal before amplification. The amplified signals from both the thermocouples and the pressure transducers contain 60 and 120 Hz noise. Simple R-C filters were constructed to remove this noise, and thus allow lower frequency
FIGURE 4 SCHEMATIC REPRESENTATION OF THE EXTRUDER AND CHILL ROLL ASSEMBLY

LEGEND
1 Feed Hopper
2 Barrel
3 Valve
4 Adapter
5 Viscometer
6 Die
7 Driven Rolls, Water Cooled
8 Idler Rolls
9 Driven Roll
10 Take-Up Roll

INSTRUMENTATION
B Beta Gauge
P Pressure Transducers
T Thermocouples
data sampling without fear of aliasing this noise into the frequency region of interest. For the thermocouples, single-stage filters (one resistor, one capacitor) are used. Three-stage filters are used with the pressure transducers. Bode plots showing the response for these filters are given in Appendix D.

The cooling and take-up rolls are a modification of the equipment described by Reber (1971) and by Kutscher (1970). The pass-line was changed by moving one idler roll to a higher position, inserting an additional idler roll, and between them mounting a thickness measuring device and a machine for marking the sheet. This mark is an oscillating ink line, with peak-to-peak distance about one and one-fourth inches and a frequency of 5.75 Hz. The chill roll assembly is shown schematically in Figure 4.

The thickness measuring device, called a Beta Gauge in this work, is an Industrial Nucleonics (Columbus, Ohio) Model 0-2; S-11 (geometry TLK-4) Series 400 AccuRay measuring system. This instrument, also described by Draudt (1971), is based on the adsorption of radiation by material placed between a Krypton-85 source and a detector cell. Thus it is a measure of mass per unit area. By assuming constant sheet density, the instrument readings are taken as a thickness measurement. The source window, that is, the area through which the radiation passes from the source through the sheet to the detector, is three-eighths by three inches, with the three-inch direction parallel to the machine direction of the sheet. However, it was found that regions of the window away from the middle do not significantly contribute to the
measurement, hence the effective window is smaller. (This was tested using sheet upon which a band of tape had been placed, creating a step-function increase in thickness.) The Beta Gauge is mounted on a track, with the source below the sheet pass line, and the detector above. The source and detector may be moved to any cross-sheet position either manually or by using a digital output group of the computer real-time interface. Detailed instructions for the calibration of the Beta Gauge are given in the instruction manual (Industrial Nucleonics, 1970) and by Draudt (1971). It has been found that this thickness measuring device is much more sensitive than other instruments available here for calibration, such as a hand micrometer.

Considering the random nature of radioactive decay, one would expect the thickness signal to be noisy, in spite of the electronic averaging circuit used in the Beta Gauge. It was found that noise levels of the order of one per cent of the measurement are present. However, this noise was found to be random; no predominate higher frequencies are present. Therefore, no filtering is used (other than that built into the detector and averaging circuit). As no aliasing of noise should occur, it is better to permit minor noise in the signal rather than to lose response through filtering.

The machine direction thickness at one cross sheet position can be measured during the running of the extruder at the same time that extruder temperatures and pressure are taken. To provide for a more extensive analysis of the sheet after the run is completed, the chill rolls were modified to be used to move the sheet through the Beta
Gauge. A rack in front of the apparatus holds a roll of sheet, which by-passes the driven rolls. A rubber-coated roll is mounted above the idler roll just before the Beta Gauge, and is allowed to rest on the idler, forming a pinch roll. The rubber-coated roll is driven by a small constant-speed motor, which allows the sheet to move under the roll at a fixed speed of 1.12 inches per second. The take-up roll is used to provide the force to pull the sheet; the rubber-coated pinch roll is used to insure a constant sheet speed, thus an even sample spacing of the data as taken by the computer. For different cross-sheet positions, the Beta Gauge source and detector are manually repositioned, and the sheet run again through the gauge. No use has been made in this study of the computer-controlled cross-sheet positioning of the gauge.

The polymer used in this work is an impact modified polystyrene, Dow Chemical USA (Midland, Michigan) Styron® 470. Virgin material was rarely used. Sheet produced on previous occasions has been remade into pellets and used again. Much of the polystyrene used had been used three to five times previously.
CHAPTER 5

COMPUTATIONAL TECHNIQUES

For both the extruder temperature and pressure and the sheet thickness data, the frequency of oscillation is of most interest. With the exception of the thickness measurement made at a fixed cross-sheet position while the sheet is being produced, the sample intervals for the sheet thickness and for the conditions of extruder operation are not the same. Thus, the most valuable aspect of the data is the frequency content, which may be easily compared. This may be obtained by Fourier transforming the results of standard autocovariance calculations. (See, for example, Chapter 6 of Jenkins and Watts, 1968, or Chapter 12 of Himmelblau, 1970.) A better method, one which requires fewer arithmetic operations, hence less rounding error and faster calculation time on a computer, is by use of the "fast Fourier transform". In this writer's experience, few of the many publications concerning the fast Fourier transform satisfactorily explain the method, thus it will be briefly described here.

Because of its speed in transforming time domain data to the frequency domain, the fast Fourier transform has had a remarkable impact on the analysis of cyclic data since its introduction in the mid-1960's. Many articles have been written about the method and its
use, including whole issues of journals dedicated to this subject (e.g. IEEE Transactions on Audio and Electroacoustics, June 1967 and June 1969). Numerous algorithms for computer implementation have been published. The algorithm used in this study is perhaps the most popular: the power-of-two Cooley-Tukey algorithm (Cooley, 1966).

Only slight modifications from the code submitted by Mr. Cooley to the IBM SHARE Program Library were made in this study. The most understandable derivation of this algorithm is probably that of the introductory article in the June 1967 special issue of IEEE Transactions on Audio and Electroacoustics (Cochran et al., 1967). In the presentation which follows, much has been taken from that article. Other derivations are presented in textbooks, such as Bendat and Piersol (1971) and Jenkins and Watts (1968), in program descriptions, notably that of Cooley (1966), and in journal articles, for example Cooley et al. (1969), Cooley and Tukey (1965), and Gentleman and Sande (1966). Some of these derivations are very general, and difficult to follow. An interesting algorithm, not requiring an even power of two data points, has been published by Singleton (1969).

In this study, the discrete Fourier transform will be defined by

\[ A_r = \frac{1}{N} \sum_{k=0}^{N-1} X_k \exp(-2\pi i r k / N) \]

where \( A_r \) is the \( r \)-th coefficient (complex) of the transform, \( N \) is the number of evenly-spaced data values \( X \) (real or complex), \( i \) is the square root of \(-1\), and \( \exp \) stands for "raise \( e \) to the power of", \( e \)
being the base for natural logarithms. In this study, $N$ is restricted to be an even power of two, though this is not a requirement of the definition. The algorithm derived does require this restriction, and data must be truncated, or extended by adding zero values, to meet this condition. Also, the factor $1/N$ in the definition is not uniformly accepted. Many authors omit the factor; a few compromise by using $1/\sqrt{N}$ instead.

The inverse transform is similar:

$$X_k = \sum_{r=0}^{N-1} A_r \exp\left(\frac{2\pi i r k}{N}\right) \quad k=0,1,2,\ldots,N-1.$$  

The factor $1/N$ is used here by those not including it in the transform definition. Note that because of the similarity, the same algorithm can be used to compute both the transform and its inverse.

The basic concept of the power-of-two (Cooley-Tukey) algorithm is that a longer time series can be transformed by combination of two transforms each of one-half as many data values. The definition can be written as two summations:

$$A_r = (1/N) \sum_{k=0}^{N-2} X_k \exp\left(-\frac{2\pi i r k}{N}\right) \quad k \text{ even}$$

$$+ (1/N) \sum_{k=1}^{N-1} X_k \exp\left(-\frac{2\pi i r k}{N}\right), \quad r=0,1,2,\ldots,N-1.$$  

By defining two new time series, one consisting of the even ordered values of $X$, and the other the odd, and introducing a change in
variable for the index of summation,

\[ (4) \quad k' = k/2 \quad Y_{k'} = X_k \quad k=0,2,4,\ldots,N-2 \]
\[ k'' = (k-1)/2 \quad Z_{k''} = X_k \quad k=1,3,5,\ldots,N-1 \]

the transform of \( \chi \) can be written:

\[ (5) \quad A_r = 2 \left[ \frac{1}{N/2} \sum_{k'=0}^{N/2-1} Y_{k'} \exp\left(-2\pi i r k'/(N/2)\right) \right] \]
\[ + 2 \left[ \frac{1}{N/2} \sum_{k''=0}^{N/2-1} Z_{k''} \exp\left(-2\pi i r k''/(N/2)\right) \right] \exp(-2\pi i r/N) \]

Recognizing the expressions for the transforms of the series \( Y_{k'} \) and \( Z_{k''} \), one can see how for \( r=0,1,\ldots,N/2-1 \) the \( A_r \) can be computed by combining the coefficients of the transforms of the two time series, each of \( N/2 \) values, resulting from using every other point of the original series. To extend this through the range \( r=N/2,N/2+1,\ldots,N-1 \) one must realize that the periodic property of the infinite continuous Fourier transform is also inherent in the definition of the discrete transform. Defining

\[ (6) \quad r'=r-N/2 \quad N/2 \leq r \leq N-1 \quad (0 \leq r' \leq N/2-1), \]

the expression of the exponential quantity can be written as

\[ (7) \quad \exp(-2\pi i r k/(N/2)) = \exp(-2\pi i (r'+N/2) k/(N/2)) \]
\[ = \exp(-2\pi i r' k/(N/2)) \exp(-2\pi i k). \]

Since \( k \) is always integer, \( \exp(-2\pi i k) = 1 \) for all \( k \), and thus the exponential is periodic of period \( N/2 \), as is the transform for \( N/2 \) data values. Therefore, in the expression for \( A_r \), Equation (5), the
value of $r$ in the transform for each of the series $Y$ and $Z$ can be extended for the full range of $r$ for the $X$ transform, by merely repeating values. The other exponential in Equation (5) can also be simplified:

$$\exp(-2\pi r/N) = \exp(-2\pi (r'N/2)/N)$$
$$= \exp(-2\pi r'/N)\exp(\pi i)$$
$$= -\exp(2\pi r'/N).$$

Combining these results, and defining $B$ to be the transform of the even-ordered values $Y$ of $X$ and $C$ to be the transform of the odd-ordered values $Z$ of $X$,

$$A_r = \begin{cases} 
2(B_r + \exp(-2\pi r/N)C_r) & r=0,1,\ldots,N/2-1 \\
2(B_{r-N/2} - \exp(-2\pi (r-N/2)/N)C_{r-N/2}) & r=N/2,N/2+1,\ldots,N-1 
\end{cases}$$

These results show how the transform of a time series can be formed from the transforms of series composed of every other point of the original series. In turn, each of these two series can be found by again splitting the series into two parts. The process can be continued $\log_2 N$ times, until there are $N$ time "series" of one value each. (The reason for the requirement for an even power of two data points is now obvious.) The transform of a single value is that value; thus the complete transform of the original series can be computed by $\log_2 N$ iterations of the combination procedure. This is the basis of the Cooley-Tukey fast Fourier transform algorithm. One other
feature is added. By ingenious ordering of the original series $X$, all combinations can be stored in the same computer memory cells as the original values. Thus no large amount of extra storage is required for intermediate results. Cochran et al. (1967) give a very clear explanation, complete with excellent signal-flow diagrams, for this ordering.

The fast Fourier transform subroutine used in this study, subroutine FORT (listed in Appendix C), was adapted with very little modification from the subroutine published by Cooley (1966). While Cooley's routine could handle $N$ up to 8096, the routine used here is restricted to $N$ no larger than 2048. Some of the program statements were changed to meet restrictions imposed by the PDP-15 Fortran compiler used.

In the above discussion, it has been assumed that both $X_k$ and $A_r$ are complex numbers. The usual case is for $X_k$ to be real-valued only (though $A_r$ will be complex). For this case of real-valued $X$, the real parts of the $A$ are symmetrical about $r=N/2$, and the imaginary parts anti-symmetrical, or

\begin{equation}
A_{N-r} = A^*_r,
\end{equation}

where $A^*_r$ is the complex conjugate of $A_r$. This can be shown as follows:

\begin{equation}
A_r = \frac{1}{N} \sum_{k=0}^{N-1} X_k \exp(-2\pi i r k / N)
= \frac{1}{N} \sum_{k=0}^{N-1} X_k (\cos(-2\pi r k / N) + i \sin(-2\pi r k / N))
\end{equation}
\[ A_r^* = \left( \frac{1}{N} \right) \sum_{k=0}^{N-1} X_k \left( \cos \left( \frac{2\pi rk}{N} \right) - i \sin \left( \frac{2\pi rk}{N} \right) \right) , \]

for \( X_k \) real.

\[ A_{N-r} = \left( \frac{1}{N} \right) \sum_{k=0}^{N-1} X_k \exp \left( -2\pi i(n-r)k/N \right) \]

\[ = \left( \frac{1}{N} \right) \sum_{k=0}^{N-1} X_k \exp \left( +2\pi nk/N \right) , \quad \text{since} \ \exp(-2\pi ik)=1. \]

\[ A_{N-r} = \left( \frac{1}{N} \right) \sum_{k=0}^{N-1} X_k \left( \cos \left( \frac{2\pi rk}{N} \right) + i \sin \left( \frac{2\pi rk}{N} \right) \right) \]

Applying the symmetric relations for cosine (cos) and sine (sin),

\[ A_{N-r} = \left( \frac{1}{N} \right) \sum_{k=0}^{N-1} X_k \left( \cos \left( \frac{2\pi rk}{N} \right) - i \sin \left( \frac{2\pi rk}{N} \right) \right) \]

\[ = A_r^* \]

Thus if one has an array of \( N \) real-valued data points, only \( N/2+1 \) complex values of \( A_r \) (\( A_0, A_1, \ldots, A_{N/2} \)) need be computed to give the full set of frequency coefficients of the transform. The amount of computer storage required for the transform of real data is thus the same as for the data, though the transform coefficients each require that two numbers be stored. (Note that \( A_0=A_N \) is real-valued since \( \sin(0)=\sin(-2\pi k)=0 \), as is \( A_{N/2} \) since \( \exp(-2\pi i k)=1 \).

The first transform coefficient, \( A_0 \), is the amplitude for zero frequency, or "D. C. offset". This is the average value of the time series. Usually the average value is computed and subtracted from
each element of the time series before transforming, with the result that $A_0=0$. This permits easier interpretation of the transform.

Should the time series be increasing or decreasing in time, further pre-treatment of the data is necessary. Intuitively, the need for this can be recognized by noting that the Fourier transform, in both continuous and discrete forms, is cyclic with period equal to the length of the time series being transformed. Thus a ramp function is equivalent to a saw-tooth wave. The transform thus contains large contributions of low frequency to handle the step at the break between cycles, when in fact that low frequency is not part of the true data signal. To avoid this, a least-squares linear regression line is fit to the original time series, and subtracted from it, to remove the trend. This is well described in Bendat and Piersol (1971), Chapter 9, Section 1. This reference also cautions against applying this detrending technique to series for which it is not obviously required, for fear of removing true low frequency contributions to the spectra.

Another source of misleading results can be attributed to the cyclic nature of the transform. If the period of the data sample is not an even multiple of the periods of predominant frequency contribution, then again non-existent low frequency components will be introduced into the transform because of the apparent step change between cycles of sample length. (This can be easily observed by generating a time series consisting of a single sine wave of a non-integer frequency and transforming it.) To minimize this effect, a cosine taper (or another "data window") is applied to force the two
ends of the series to zero, avoiding the artificial step change. This is most often done by multiplying the first and last ten per cent of the series by a cosine (fourth and first quadrants) of period such that the elements of the new series are zero at the ends and increase to 100 per cent of the original series ten per cent in from each end. The mathematical implications of using data windows such as this are involved, and covered in texts such as Bendat and Piersol (1971) and Jenkins and Watts (1968). One must always be aware that the use of such techniques can under some circumstances give rise to more error than they correct.

Of more interest than the coefficients of the Fourier transform is the power spectral density function (PSDF). This gives the relative contribution of each frequency in the time series. It is simply computed, being the square of the magnitude of the (complex) coefficient at each frequency of the discrete Fourier transform, multiplied by a weighting factor for the data window (such as cosine taper) if any were used. Bendat and Piersol (1971), Chapter 9, provide a good description of the method of calculation of the PSDF.

The PSDF often shows small variations resulting from random noise in the data signal. It is therefore often desired to smooth the PSDF. To insure that these variations are not true contributions to the power spectrum, it is best to average the computed PSDF for several data records for the signal. Another technique used is called "Hanning", after the originator. For each frequency, the smoothed value of the PSDF is one-half the computed value plus one-quarter of
each of the computed values for the two adjacent frequencies. As with
other techniques to improve the resulting PSDF, this writer feels that
this averaging of frequencies should be used with caution, lest
significant results be lost.

Another facet of the frequency analysis should be noted. The
Fourier transform is based on the sine function. Any sinusoidal
contribution to the signal will be transformed to give a high value
of the PSDF for the frequency of the sine wave. If the signal is not
sinusoidal, higher harmonics should be expected. Most texts on the
Fourier transform give examples of the coefficients for the trans-
form of such functions as square waves and saw-tooth waves. The same
results should be expected: Higher frequency peaks in a PSDF may be
from the Fourier decomposition of a lower frequency non-sinusoidal
signal rather than from a true variation at that frequency.

The program, named BTAl\textsuperscript{4}, which was written for the spectral
analysis of the data for this study is listed in Appendix C. This
program provides options for cosine tapering, detrending using a first
order least squares, and other features.

While comparison of the frequency content (PSDF) of time series
is useful for determining if a cyclic change in one signal is also
present in another, it offers no insight into possible correlation of
non-periodic signals. For this, the cross-covariance or cross-
correlation function is needed. The (estimate of the) cross-
correlation coefficient, which approaches ±1.0 or -1.0 if the two
series are correlated (i.e. not independent of each other), may be
calculated by use of the fast Fourier transform. (See Section 7, Chapter 9, of Bendat and Piersol, 1971.) Transforms of both series are combined, and then the inverse transform found. This method was not used in this work, but rather the direct calculation using

\[ R(t) = \left[ \frac{1}{N} \sum_{i=1}^{N} (X_i - \bar{X})(Y_i - \bar{Y}) \right]^{-1} \]

where \( R \) is the estimate of the correlation coefficient of the two time series \( X \) and \( Y \) of length \( N \) and average values \( \bar{X} \) and \( \bar{Y} \), and \( t \) the time lag separating the initial values of each series. This assumes that the two series are sampled at the same sample frequency. As this was not always the case in this study, it was often necessary to create a new series by interpolation of one of the two original series. It is preferable to interpolate original data rather than Fourier coefficients, as would have been required if the Fourier transform method had been used. Note that the calculation of \( R \) must be performed for every possible time difference \( t \) in order to determine if the two series are correlated. (The Fourier transform method computes the cross-correlation function for all lags in one step.) If the data are cyclic, a high correlation coefficient should be found for time lags at equal intervals corresponding to the period of the repeating unit of the signal.
CHAPTER 6

EXPERIMENTAL PROCEDURE

The extruder was operated three times to provide sufficient data for this study. For all runs, the extruder was at steady state: All variations in conditions were natural rather than in response to any forcing. Temperature data collected during production of the sheet were from nine of the thermocouples in the extruder equipment and the noise reference thermocouple in an ice bath. The sample interval for these ten thermocouples was 0.1 second. The six pressure transducers described in Chapter 4 were sampled at a 0.05 second interval. The thickness of the sheet produced as measured by the Beta Gauge was also measured at a 0.05 second interval. The response of the thermocouples and pressure transducers is slower than these sample rates. The filters used sufficiently attenuated high frequency noise so that with these sample intervals no higher frequency signal would be aliased.

Program BTA10 implements this data collection. Written in Fortran, this program calls the dual-processor system library subroutines to provide automatic filling of data buffers by the monitor, as described in Chapter 2. For the pressure transducers, the 100 Hz sample frequency was used, with only every fifth value being saved,
thus obtaining the desired 20 Hz sample frequency.

As a typical run lasts over twenty minutes, excessive storage would be required if the data sampled at these frequencies were to be saved over the full run. To minimize storage requirements and still provide data for finding both high and low frequency spectral contributions, program BTAl0 uses a two tier method. The 10 and 20 Hz data collected are saved for a 103 second period every ten minutes during the run. Also, throughout the run, each input is averaged over one second (10 temperature samples, 20 samples for each pressure transducer and the Beta Gauge). These averages are stored to provide data for the full run. Program BTAl0 is listed in Appendix C.

More extensive sheet thickness data were collected after the sheet was produced, as described in the discussion of the chill roll and Beta Gauge equipment in Chapter 4. Program BETA8, listed in Appendix C, implements this. The sample frequency is 10 Hz, giving a distance separation of 0.112 inches between samples. As the rate of travel of the sheet through the Beta Gauge differs from that during the run, the data sample interval is different. While comparison of power spectral density functions is unaffected by this, direct comparisons by cross-correlation calculations or by visual means requires interpolation of the data collected. To convert these data to the same time base as the extruder run-time data, the speed at which the sheet passes through the Beta Gauge must be accurately known, both during production of the sheet and for the later thickness measurements.
Of the three extruder runs made for this study, the first was preliminary. To determine if variations in sheet thickness would be detected, this run was made with a known malfunction in the chill roll assembly. A warped gear caused binding of the shaft of the first roll. This caused this driven roll to have a uniform variation in rotational speed, resulting in a varying pull on the molten sheet being extruded onto it. Thus a cyclic variation in sheet dimensions, of period equal to the period of rotation of the roll, was present.

For this run, the thermocouples used were the six near the lip of the die and the ones at the die entrance, the adapter exit, and the adapter entrance.

After examining the temperature data collected during the first run, it was decided that little information could be gained from the six thermocouples near the die lips. For the other two runs, only three thermocouples were used at the lips, one near each end and one near the middle. The adapter outlet thermocouple was not used. In their place, four thermocouples along the inner surface of the extruder barrel were used. This provided data on the early processing of the polymer as it passed through the extruder. Table 2 summarizes the conditions of each of the three runs for this study.

Calibration equations for the thermocouples and pressure transducers were provided by Fontaine (in preparation), and are given in Appendix D. The Beta Gauge was calibrated for the conditions of Run 2, but not for other runs. This calibration is also given in Appendix D. Rather than using the linearization method recommended by the
## TABLE 2

### RUN CONDITIONS

<table>
<thead>
<tr>
<th>Run</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screw speed, RPM</td>
<td>84</td>
<td>41</td>
<td>101</td>
</tr>
<tr>
<td>Approx. sheet production speed, in./sec.</td>
<td>6.6</td>
<td>5.5</td>
<td>8.1</td>
</tr>
<tr>
<td>Beta Gauge settings</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Center scale</td>
<td>3.88</td>
<td>4.75</td>
<td>3.50</td>
</tr>
<tr>
<td>Span</td>
<td>9.00</td>
<td>9.00</td>
<td>9.00</td>
</tr>
<tr>
<td>Polymer history</td>
<td>Virgin</td>
<td>4th pass</td>
<td>4th pass</td>
</tr>
<tr>
<td>Number of 103 second data bursts</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Instruments used, ordered as rank in the computer data storage file</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thermocouples</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Barrel position 1 (near feed)</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Barrel position 3</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Barrel position 5</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Barrel position 7 (near head)</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Adapter inlet</td>
<td>8</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Adapter outlet</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Die entrance</td>
<td>7</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Die lip position 1 (north edge)</td>
<td>1</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Die lip position 2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Die lip position 3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Die lip position 4</td>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Die lip position 5</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Die lip position 6</td>
<td>6</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Ice bath (reference)</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Pressure transducers</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Head</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>Adapter</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>Viscometer inlet</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Viscometer outlet</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>Die</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Open air (reference)</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Beta Gauge</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
</tbody>
</table>
manufacturer (Industrial Nucleonics, 1970), a second order polynomial regression equation was used. All calculations were performed on the raw data, as cyclic variation or cross-correlation is just as easily found without first applying calibration equations. As the calibration for all instruments used is monotonic, cross-correlation results are unaffected by the use of raw data, even for non-linear calibrations. While the numeric value of the PSDF depends upon the calibration, the frequencies at which peaks occur do not. As the relative magnitude of the peaks as compared to surrounding frequencies is of importance in this study, the use of raw data is justified. It should be noted that the calibration is linear or nearly so for the instruments in the ranges used.
CHAPTER 7

RESULTS AND DISCUSSION

In this chapter, the significant findings of the plastic sheet thickness study are presented and interpreted. The first topic is the power spectral density function (PSDF) results for the sheet. The second topic concerns cross-correlation calculations for sheet thickness with die pressure and temperature.

The third topic presented is the PSDF calculations showing screw-induced disturbances in pressure and temperature in the extruder. An illustration of the PSDF technique using electronic function generator signals is given for comparative purposes.

The sheet produced showed definite variations in thickness. Figure 5 is a plot of the thickness of a forty-five inch segment of sheet produced during the second run. For a fixed cross-sheet position, the thickness varied by more than one mil (0.001 inch), with the average thickness between six and seven mils, depending on cross-sheet position, for Run 2. Considerable cross-sheet variation was noted, as would be expected. The lips of the die are very difficult to adjust, and a completely uniform slit was not obtained. Also, the molten sheet necks down from twenty-six inches in width at the die lips to between sixteen and eighteen inches final sheet width.
Successive curves have been displaced vertically by one division. Each ordinal division represents 0.5 mils thickness.
The quality of sheet produced was generally poor. One problem was the spot wetting of the first chill roll by condensed polymer degradation fumes. Considerable effort was expended to minimize this, but it was never completely eliminated. The result of this wetting was uneven heat transfer to the roll and a tendency for the sheet to dimple as it was pulled from the first roll to the second. Run 2, at low screw speed, produced the best sheet, and therefore most of the analysis was based on thickness measurements for this run.

Figure 5 points out some interesting details. The data plotted were taken at one-quarter inch cross-sheet intervals. (The regions within two inches of the edges were not included.) As the plot was made with equal displacement of line origins, lines that are close together indicate a general increase or decrease in cross-sheet thickness. The sheet was thickest approximately one-third down from the top of the plot, and thinner near the edges. The regions at the edges, not shown in Figure 5, were thicker, but not so thick as the middle band. The thickest part of the sheet had less severe variation in thickness. The thinner die lip setting and the large neck-down of the molten sheet as it passed from the die probably caused the thickness variation to be more intense on each side of the thick band. It is to be noted that most of the ripples are continuous, though not as pronounced in the thick region. Thus the thickness variation cannot be entirely the result of the neck-down, which would give a flat profile in the middle and unrelated rippling on each side. It is suggested that the neck-down intensifies variations which were the
effects of other causes.

It has been observed with other runs not made in conjunction with this study that the sheet often seemed to sag in the middle as it necked-down. The die lip setting used in this study, which resulted in the thicker band near the middle of the sheet, tended to minimize sag, though it may not have been eliminated. As the sheet passed over the first roll, any sag present would have been stressed, perhaps causing thickness variations in the still soft sheet.

Power spectral density function (PSDF) calculations were made for the sheet thickness data. The measurement of thickness made during sheet production was noisier than the off-line measurements. Also, the off-line measurements were of shorter sample interval, providing more points for calculation. Figures 6 and 7 show plots of the PSDF for each run, based on thickness measurements made off-line near the middle of the sheet. As seen in Figure 5, this position has variations in thickness more intense than in the thick band, but not so large as near the edges. As with all PSDF plots in this work, the frequency coordinate is linear while the PSDF value coordinate is logarithmic. The important feature of any PSDF presentation is the presence of distinct peaks relative to values of surrounding frequencies. The plots of Figure 6 are averages of PSDF's computed for adjacent 2048-point segments of sheet thickness data. For Run 1, fifteen segments were used, for Run 2, thirty-two, and for Run 3, ten. The smoothness of the plot for Run 2 would be matched in Runs 1 and 3 if more segments were used in the averages.
Each ordinal division represents one power of 10.

FIGURE 6  PSDF FOR PLASTIC SHEET
Successive curves have been vertically displaced by two divisions.
Each ordinal division represents one power of 10.

Successive curves have been vertically displaced by three divisions.
Each plot of Figure 7 is the average of ten PSDF calculations, each using 2048 thickness points spaced twenty samples apart. Hanning smoothing was used for Figure 7, though not needed for Figure 6. (Figure 7 plots are not as smooth as those of Figure 6 because the same block of sheet was used for all ten PSDF calculations for each run. The averages in Figure 7 are from data slightly displaced in starting point, not separate blocks as for Figure 6.)

The larger values at low frequencies shown in Figure 6 result from the nature of the discrete PSDF calculation, as explained in Chapter 5. Because no small frequency range contains significantly higher values than the surrounding area for Runs 2 and 3, it is concluded that there is no predominant frequency contribution to the observed thickness variation. For Run 1, the peak occurring at 0.0405 per inch is sufficiently greater than surrounding values to be considered probably significant. This peak is shown better in Figure 7. This frequency is for period 25.0 inches of sheet, which is close to the circumference of the first chill roll: 25.2 inches. Allowing for shrinkage as the sheet cools, the frequency of the thickness variation is the same as that of the uneven rotation of the first chill roll. Additionally, hand measurements of the sheet width showed a large variation in width (one-half inch or more). This variation repeated approximately every 24.8 inches, or frequency 0.0403 per inch. This was a difficult measurement, and the result is well within measurement error of the frequency of thickness variation found. In Figure 7, the peak at 0.081 per inch is from the nature of the Fourier transform of
non-sinusoidal functions, as explained in Chapter 5. It is unlikely that a variation in thickness actually exists at that frequency.

Except for the case of the warped gear of Run 1, no cyclic variation in sheet thickness was found. To determine if non-cyclic variations in extruder conditions influenced the sheet thickness, cross-correlation calculations with temperature and pressure data were performed for Run 2. Data records of 192 points were used, thus allowing several independent cross-correlation calculations for the 2048 value pressure and 1024 value temperature data records stored for the 103 second collection periods. The pressure used was that of the die during the third 103 second collection period of Run 2. The third collection period was used, as the most extensive thickness data had been measured for sheet produced then. Temperature data used were from the thermocouple near the middle of the die lips. Another quantity used was the pressure difference in the viscometer, an indication of the bulk flow of the molten polymer. The results were negative:

No correlation coefficient greater than 0.6 in absolute value was found for any lag time, and values in this range were not repeatable from one 192 point pressure (or temperature, or pressure difference) record to the next. It was thus concluded that the variations in temperature, pressure, or flow in the die were not of sufficient magnitude to cause measurable changes in sheet thickness. Variations in die pressure for Run 2 were of the order of one pound per square inch (psi) with the total pressure averaging about 609 psi. Calculations by Fontaine (1974) for the conditions of this run indicated that
a one psi change in pressure should result in a 0.4 per cent change in sheet thickness. The thickness data showed variations much greater: ten per cent or greater over the same time interval. The pressure difference in the viscometer varied approximately one per cent, and the die temperature, being 223.6 degrees Celsius (°C) at the end of the lip and 234.9°C near the middle of the lip, varied by approximately 0.1°C. These changes could not cause the sheet thickness variation found.

In light of the above findings, it is concluded that the post-extrusion history of the polymer could be a major cause of the thickness variation in the sheet produced by the equipment used in this study. Several factors could cause thickness variation. One is that air currents around the die could cause slight movement of the molten sheet, resulting in uneven thickness. Another is pressure instability in the die. This writer, while admitting the possibility, tends to discount this idea. If pressure instabilities large enough to cause the thickness variations observed exist anywhere in the die, the die pressure transducer should indicate some pressure difference which would correlate with the thickness. Also, the source of an instability is not clear.

Possibly the most significant source of variation is in the take-up apparatus of the chill roll assembly. In addition to the first two rolls being driven at constant speed, movement of the sheet through the chill roll assembly is facilitated by tension applied by the roll on which the sheet is wound. The tension is manually controlled by
adjusting the torque applied to this wind-up roll through a slip-clutch. As the roll of sheet becomes larger during a run, the torque must be increased to provide sufficient force to wind the sheet. Yet if too much tension is present, the sheet is pulled, stretches, and slides over the driven rolls at the beginning of the pass-line, resulting in thinner sheet. It is proposed that the slip-clutch could be causing uneven tension, resulting in variations in sheet thickness (and width) by drawing the sheet non-uniformly from the die. That variations in sheet width were noted for all runs, though these were not so severe for runs with the warped gear of Run 1 corrected, helps support this proposal. Were uneven draw being caused by gear or chain drive chatter for the driven rolls, the thickness variations would have been cyclic, as in Run 1.

As indicated above, it has been observed that sheet width varies also. Most of the variation seemed to occur on one edge of the sheet rather than equally on both edges. No explanation is offered for this, and further study might be rewarding.

It is recommended that the post-extrusion phases of the sheet production be studied more extensively to prove the true cause of the large variations in sheet thickness encountered in the use of this equipment.

The lack of demonstrated influence of extruder temperature and pressure variation on sheet thickness does not mean variations are non-existent. Figures 8 through 11 show the PSDF of pressures and temperatures for Runs 2 and 3. All plot lines are the average PSDF
Successive curves have been vertically displaced by two and one-half divisions.

Each ordinal division represents one power of 10.
Successive curves have been vertically displaced by two and one-half divisions.

Each ordinal division represents one power of 10.

**FIGURE 9 PRESSURE PSDF, RUN 3**
Each ordinal division represents one power of 10.

Successive curves have been displaced vertically by three divisions.
The top three curves have been separated vertically by four divisions, the bottom three by two divisions.
for three 103 second data records for the run. Peaks are therefore assumed to have been present throughout the run, as the data records were taken at the beginning, middle, and end of the twenty-two minute run. Cosine tapering was applied to the time series before computing the transform. Hanning smoothing was applied to the averaged PSDF's before plotting. Examination of the PSDF's of the data stored throughout the run at one-second intervals did not show any recognizable frequency contributions to the signal other than those found with the shorter interval data used for these plots.

It is difficult to draw firm conclusions from the PSDF's presented in Figures 8, 9, 10, and 11, for the variations on which they are based, while obviously cyclic, are small. For Run 2, the largest temperature variation was of the order of 0.2 to 0.3°C at the first thermocouple position in the barrel, where the average temperature was 196°C. Variation in temperature at the lips was about 0.1°C, with the average temperature 235°C. As can be seen in Figures 10 and 11, the only cyclic variation in temperature was in the barrel. (Downstream thermocouples showed results similar to the bottom lines of these plots: no significant peaks at any frequency.) Presumably there is sufficient mixing to eliminate cyclic variation in temperature.

For pressure, the variation measured by the transducer at the head was about 25 psi during Run 2; the total pressure was about 1600 psi. The next pressure transducer in the fluid stream, that in the adapter, showed only two psi variation with the total pressure averaging 932 psi. In the die, pressure variation was one psi, total
pressure 609 psi. The head pressure difference is felt large enough for meaningful interpretation. Others are questionable, though the plots show peaks for the same frequencies.

For both runs, a strong contribution to the pressure signal, as measured by the pressure transducer in the head, existed at the same frequency as the screw rotation. This is to be expected, and should be caused by the flight of the screw. Comparing both magnitude of the pressure variation and the relative strength of the peak in the PSDF plots, it is seen that this pressure pulse is quickly damped as the polymer melt moves away from the screw. For Run 3, the peak is not apparent at the next pressure transducer (in the adapter). For Run 2, it is greatly diminished by the time the melt reaches the adapter.

Assuming no flutter in the screw, this writer can offer no sound explanation for the frequency contribution at one-half the screw speed. The single-flight screw is fit into the barrel at close tolerance, and should not cause a stronger effect every other rotation. The higher frequency peaks are not true pressure oscillations, but rather the higher harmonics resulting from the Fourier decomposition of a non-sinusoidal signal, as discussed in Chapter 5. (Not shown in Figure 9 is the peak at frequency twice that of the screw, for head pressure only.) The major conclusion reached from the pressure data is that the elastic properties of the polymer melt are such that pressure variations are quickly damped out.

The temperature PSDF plots, Figures 10 and 11, are based on
calculations approaching the accuracy limits of the analog-to-digital
converter. The thermocouples in the die and the adapter outlet are
not presented as the PSDF's for these positions showed no significant
frequency peaks. Plots for these devices are similar to that present-
ed for the adapter inlet thermocouple (the lowest curve of Figures 10
and 11). The top four lines in each figure, the barrel thermocouples,
do show distinct frequency peaks. While the peaks appear more
distinct for the thermocouple nearest the beginning of the barrel,
where the polymer pellets have not melted, the variation there is less
than in the mid-range of the barrel. The peaks occur at frequencies
corresponding to one-half the screw speed, the same as the screw speed
(within screw speed determination accuracy), and at harmonics of
these. The peak corresponding to the screw speed could be caused by
viscous heating as the melt is sheared in the narrow region between
the flight of the screw and the barrel. Near the beginning, where the
pellets are still solid, the increase in temperature could result as
well from the screw conducting heat away from warmer zones in the
barrel. The screw flight is likely covered with a melt layer even in
this region (Fontaine, 1974), which would promote conduction of heat
from the screw to the thermocouple tip in the barrel wall.

As with the head pressure signal, this writer was unable to
determine a cause for the peaks appearing at twice the screw speed.
To check the possibility that these were the result of using simple
filtering of the data signal or were computational quirks of the dis-
crete Fourier transform, rather than being a true peak, an electronic
function generator was used to simulate data. A square wave and a sine wave at 1.7 Hz (setting closest possible to the screw speed) were twice used. The signal processing conditions were the same—the function generator signal was passed through a single-stage filter as was the thermocouple signal. The same sample frequency was used, and the same power of two was used for the Fourier transform. The resulting PSDF's are shown in Figure 12 with the PSDF of the first barrel thermocouple of Run 3. As shown, not only are harmonics introduced, but some lower frequency peaks occur for the filtered square wave. This writer asserts that the comparison of the square waves and the plots of Figures 8 through 11 tend to support the existence of a true low frequency peak in the data, though possibly reinforced by false contributions arising from the screw frequency peak. The low frequency peaks of the square waves are much smaller than the similar peaks for the data. In Figures 8 and 9, the peak at the frequency of the screw diminishes, but the lower frequency peak does not decrease in proportion.

One could easily suspect that for the barrel thermocouple results there may be aliasing of a higher frequency component into the region around one-half the screw frequency. With the filter attenuation (Figure 13 of Appendix D) and the Nyquist folding frequency (5 Hz) for the sample frequency used, it is possible for some medium frequency signal components to pass through the filter and be aliased to a lower frequency. Higher frequencies, such as the 60 Hz and 120 Hz known to be present before filtering, are sufficiently attenuated to cause
FIGURE 12 PSDF COMPARISON
no problem. It is here argued that the lower frequency peaks in the PSDF plots are not aliased medium frequency components because it is unlikely that such frequencies are present in the signal. If such mid-range frequencies were present from electronic noise, they would appear on all PSDF's, not just those of the barrel thermocouples. They would be of fixed (aliased) frequency, and not related to the screw speed. If there were any action in the extruder which could produce such frequency variation, the oscillations would not be strong enough to be observed with the thermocouples used. The thermocouples have a time constant of one second. The attenuation by the dynamics of the thermocouple and by the filter combined would reduce to unmeasurable all but very high amplitude temperature variations. The heat capacities and conductivities of the melt, the barrel, and the screw would prevent variations of magnitude large enough to be observable in the frequency range in question.

The four heaters in the barrel are on-off controlled, with the per cent on time during a two minute cycle being the controlled variable. Surprisingly, there was no cyclic variation found in the long term temperature data at low frequencies. The lack of any frequency contribution matching the barrel heaters cycle times leads one to the conclusion that the heat capacity of the barrel is sufficient to damp out the temperature oscillation expected.

Again, caution must be exercised in drawing conclusions from these data: The magnitude of variation is extremely small compared to the average signal. And, any conclusions drawn probably would be of
little practical value. It has been shown that for the equipment and material used in this study, cyclic variations in extruder conditions at steady state could cause no significant variation in sheet thickness in comparison with the variations present from other causes.
PART III

SUMMARY OF CONCLUSIONS AND RECOMMENDATIONS
CHAPTER 8

CONCLUSIONS

Part I of this work describes the advantages of the dual-processor computer configuration and modified RSX-Plus monitor system for multi-user real-time computer service. By dedicating a small computer to handle the high overhead functions required by some peripheral devices, more time is available on the main computer for user tasks. By operating the real-time interface at a fixed cycle time, users are assured of even-interval data from real-time inputs, regardless of the initiation or termination of other user programs with real-time input. By providing for automatic transfer of real-time input to a user specified buffer, greatly improved time-sharing is obtained. The protection scheme for real-time outputs provides adequate immunity from accidental manipulation by another user.

The conclusions reached in Part II of this work deal with the non-uniform machine direction thickness of plastic sheet. For the materials and equipment used in this study, the following conclusions have been reached:

1. The variation in sheet thickness which can be attributed to extruder processing conditions measured in this study is insignificant compared to other causes, such as post-
extrusion processing.

2. While the screw causes large variation in pressure at the end of the barrel, pressure variations are minimal at the die. The elastic properties of the polymer melt allow pressure differences at the extruder head to be quickly damped out.

3. Temperature variations present in the polymer melt in the barrel are not present in the die melt to a measurable extent.

4. Time series analysis techniques, such as the Fourier transform, are effective methods for examination of extrusion data, provided proper caution is exercised. These techniques can be implemented on medium-sized computers with resource sharing, such as the system described in Part I.
CHAPTER 9

RECOMMENDATIONS

The work presented in Part I shows the advantages of the dual-processor computer configuration, and also that there is room for expansion. The following recommendations are based on experience using the dual-processor system:

1. More digital inputs and digital outputs are needed.
2. The graphics capabilities should be better exploited.
3. An additional interrupt line should be installed from the secondary to the primary processor to reduce primary processor overhead in handling requests for keyboard input, character generator display output, and graphics display.
4. Additional equipment for the main processor would enhance the usefulness of the system. Additional core memory and an industry compatible magnetic tape unit are needed.
5. Additional programming, such as shifting the teleprinter handler to the secondary processor, revising the DECtape handler, and providing an output spooling routine, should be done to increase the effectiveness of the system.

Recommendations based on the work of Part II of this study deal
with the need for further study of the post-extrusion processing of the sheet produced. Specific recommendations are as follows:

1. A more detailed study of the variations in sheet width is needed. A correlation of width with thickness should be attempted. This study would be facilitated by construction of a computer-interfaced device for automatic width measurement.

2. Efforts to find a means of preventing the wetting of the first chill roll by condensed degradation fumes should be continued.

3. A constant tension should be applied to the sheet as it is drawn through the rolls. One possibility would be to use a pinch roll with the last driven roll.

4. Cross-sheet variations should be studied more intensively, especially the influence of the necking down of the melt as it passes from the die.

5. Flow characteristics of the polymer melt within the die and during the sheet forming phase should be investigated with a view to relating flow instabilities to sheet thickness variations.
APPENDIXES
APPENDIX A

System Program Descriptions and Start-Up Procedure

The tasks and subroutines used with the dual-processor modification for RSX-Plus are outlined below in the format of the "RSX-15 Phase 2 -- Procurement, Building and Checkout" manual (Desimone and Proteau, 1972). Also, the modified start-up procedure is given. This information is presented as an extension of the manual, and thus the section numbering is a continuation of the section numbering used there.

Table 3 follows this information, and gives a user-oriented summary of the calling sequence for the Fortran library subroutines used with the real-time interface. More detailed instructions as to their use are given as part of the program listings in Appendix B.
6.0 Dual-Processor Modifications Program Source Tape.

6.1 Dual-Processor Core-Resident Monitor Programs.

1. .PATCH SRC -- Modification of the basic executive to handle real-time CAL functions, the modified UNFIX CAL function, and the interrupt from the secondary processor. Changes in the number of real-time inputs or outputs handled by the secondary processor require small sources changes (edit) in this code, as indicated in the listing.

   Assembly Parameters: None (changes by editing)
   Task Building Options: EXM, NFP, PGR
   Name of Task: .PATCH
   Task Priority: 512 (User's option, never executed)
   Partition: PATCH(13000,1400)
   Resident Code: .PATCH
   Links & Structure: None

2. MOD1OF SRC -- Secondary Processor Program. Changes in the real-time interface require changes in this source (edit) at points clearly indicated in the listing.

   Assembly Parameters: None (changes by editing)
   Task Building Options: EXM, NFP, PGR
   Name of Task: .MOD1OF
   Task Priority: 512 (User's option, never executed)
   Partition: MOD1OF
   Resident Code: MOD1OF
   Links & Structure: None

6.2 Dual Processor MCR Sources

1. DPP.Ø2 SRC -- Executed at system start-up to fix in core tasks .PATCH and .MOD1OF, to check for the existence of partition SHARE and set its partition occupied flag, initialize the real-time input-output system, and overwrite the basic executive with links to .PATCH

   Assembly Parameters: None
   Task Building Options: EXM, NFP, PGR
   Name of Task: ...DPP
   Task Priority: 2 (Recommended for MCR)
   Partition: MCR
   Resident Code: DPP.Ø2
   Links & Structure: None
2. **PRO,Ø1 SRC** -- The reassign real-time output protection code MCR task. Changes in the number of available real-time outputs will require slight source changes (edit) in the code, at points clearly indicated in the listing.

   Assembly Parameters: None (changes by editing)
   Task Building Options: EXM, NFP, PGR
   Name of Task: ...PRO
   Task Priority: 2 (recommended for MCR)
   Partition: MCR
   Resident Code: PRO,Ø1
   Links & Structures: None

3. **LPR,Ø1 SRC** -- The list real-time output protection table MCR task. Changes in the number of available real-time outputs will require slight source changes (edit) in this code, at points clearly indicated in the listing.

   Assembly Parameters: None (changes by editing)
   Task Building Options: EXM, NFP, PGR
   Name of Task: ...LPR
   Task Priority: 2 (recommended for MCR)
   Partition: MCR
   Resident Code: LPR,Ø1
   Links & Structures: None

4. **REM,OS SRC** -- modified REMOVE MCR function task (replaces REM,Ø SRC).

   Assembly Parameters: None
   Task Building Options: EXM, NFP, PGR
   Name of Task: ...REM
   Task Priority: 2 (recommended for MCR)
   Partition: MCR
   Resident Code: REM,OS
   Links & Structures: None

6.3 **Dual-Processor I/O (Input/Output) Handler Source**

   The recommended partitions of section 2.3 for other I/O handlers do not apply. DT..., PR..., and PP... are recommended IO.1 (70000,3400), RF... for IO.2 (734Ø,24Ø0), and LF... for IO.3 (76000,1000).

1. **KSG SRC** -- Primary processor section of the handler for the keyboard, character generator, and scan converter.

   Assembly Parameters: None
Task Building Options: EXM, NFP, PGR
Name of Task: KS...
Task Priority: 1 (recommended for I/O)
Partition: IO.4(77000,10000) (User's Option)
Resident Code: KSG
Link & Structure: None

6.4 Dual-Processor System Library Sources.

1. RTIO SRC -- The Fortran-callable subroutine for real-time inputs (RTINX) and outputs (RTOUX).
   Assembly Parameters: None

2. RTIN SRC -- The Fortran-callable subroutine for real-time inputs (CAL 31).
   Assembly Parameters: None

3. RTOUT SRC -- The Fortran-callable subroutine for real-time outputs (CAL 32).
   Assembly Parameters: None

4. RTSET SRC -- The Fortran-callable subroutine for requesting automatic real-time input (CAL 33).
   Assembly Parameters: None

5. RTRST SRC -- The Fortran-callable subroutine for requesting synchronization (reset) of entries in the request table for automatic real-time input (CAL 34).
   Assembly Parameters: None

6. ADCNV SRC -- The Fortran-callable subroutine to convert analog-to-digital converter inputs to scaled integer numbers (IADCV) or floating point numbers (ADCNV).
   Assembly Parameters: None

7. GRAPH SRC -- The Fortran-callable subroutine for driving the graphic display (scan converter) section of the I/O Handler KS. Entries are SCLA (clear 'scope), PNT (point), HZ (horizontal line), VL (vertical line), and LIN (straight line in any direction).
   Assembly Parameters: None
7.0 The Dual-Processor Cold Start Tape.

This is a "Cold Start" tape containing the dual-processor tasks, an updated library with the dual-processor subroutines, and the tasks provided on the original RSX-15 Phase 2 Cold Start Tape (section 3.0). The TDV tasks were built for user mode, and all tasks were built for a 36K system, with the top 12K shared by the secondary processor. The following partition definitions are required so that these tasks can be installed:

```
PATCH(13000,14000)
MCR(15000,3000)
TDV(20000,24000)
MOD10(60000,28000)
SHARE(67400,14000)
IO.1(70000,34000)
IO.2(73400,24000)
IO.3(76000,14000)
IO.4(77000,14000)
```

Additionally the following user partitions are recommended:

```
USER1(144000,140000)
USER2(160000,160000)
USER3(66000,14000)
USER4(144000,14000)
62000 to 66000 unused (for monitor use)
```

The following devices should be defined:

```
RF
KS
LP
DT0
DT1
DT2
PR
PP
```

After installation of tasks by the configurator, the following procedure should be used:

1. Reassign LUN's 5, 10, 11, 15, 17, and 18 to RF0, and LUN 19 to DT0

2. Using FIN, bring the library (.LIBRX BIN) onto disk from the cold start tape.

3. Enter the proper time and date.
4. Call the MCR function DPP.

5. Start the secondary processor.

6. Reassign LUN's 12, 13, 16, 20, and 21 to KS0.

7. Halt the secondary processor. If any API state light is on, push the continue switch and again push stop switch to halt. Repeat if necessary.

8. Call the MCR function SAV.

9. Save the system on DECTape.

10. Restart the system (paper-tape warm start).

11. Start the secondary processor.

12. Correct the time.

Following is a sample run of the cold start procedure for the dual-processor system: The lines preceding a change from MCR to TDV or from TDV to MCR end with ALTMODE, all others end in a carriage return. The LOG command is for comments, and need not be used.

```
KSY SYSTEM CONFIGURATION

SPECIFY CODE SIZE > 36K

SPECIFY NUMBER OF TTY'S > 2

SPECIFY NUMBER OF TTY'S > 3

SPECIFY NUMBER OF CLOCK TICKS PER SECOND > 60

DEFINE PARTITIONS "NAME(FASE, SIZE)"

>FSKB(16000,1600)
>USI0(16000,2000)
>FOCI(16000,2000)
>TAH(20000,2400)
>USI1(64000,16000)
>M01(64000,16000)
>USER(00000,16000)
>SHARE(64000,16000)
>IT0 (10000,24000)
>IT1 (7000,24000)
>IT3 (7000,11000)
>IT4 (7000,11000)
>USER(10000,10000)
```

DEFINE SYSTEM COMMON BLOCKS "NAMECASE,SIZE"
>
SPECIFY DEVICE NAMES & UNIT NUMBERS (ONE PER LINE)
> DFC
> KSC
> LFC
> MTC
> H11
> H10
> H19
> H29
>
INSTALLATION OF TASKS FROM DT-6

803C NODES IN POOL

SYSTEM IS RUNNING

MCF>LOG 5,16,11,15,17,18 3K6 NONE
MCF>LOG 19 3K6 NONE
MCF>LOG 14 NONE 3K6
MCF>LOG ENTER ALTMODE AND CNVF T.
MCF>FIN *LIBRET FIN
MCF>LOG ENTER ALTMODE AND CNVF C.
MCF>FIN 16:12:46 9/5/73
MCF>IPF
*MOTIF FIXED IN CORE.
*PATCH FIXED IN CORE.
FSY-PLUS HAS BEEN MODIFIED FOR THE O.S.U.
CHEX* PNGF* FITT* EVOL PROCESSOR SYSTEM.
MCF>LOG START THE SECONDARY PROCESSOR.
MCF>FA 12,13,16,20,21 KSC T10
MCF>LOG STOP THE SECONDARY PROCESSOR.
MCF>SAV
MCF>LOG ENTER ALTMODE, THEN SAVE THE SYSTEM
MCF>LOG ON DECLARE.
FSY SAVF
DISK TO DECLARE
FSY SAVF

MCF>LOG START THE SECONDARY PROCESSOR.
MCF>FIN 16:19:30 9/5/73
MCF>LOG NOW THE SYSTEM IS REALLY RUNNING!
**TABLE 3**

FORTRAN REAL-TIME SUBROUTINE CALLS

<table>
<thead>
<tr>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL RTIN ($\phi$, IC, IV, IEV)</td>
<td>Set IV to the latest (raw) value of the real-time input number IC, with optional error code IEV. (RTINX is used if real-time output is to be called in the same program.)</td>
</tr>
<tr>
<td>CALL RTINX ($\phi$, IC, IV, IEV)</td>
<td></td>
</tr>
<tr>
<td>CALL RTIN (-1, IC, XIV, IEV)</td>
<td>Set XIV to the floating point value of analog-to-digital converter channel IC, with optional error code IEV.</td>
</tr>
<tr>
<td>CALL RTINX (-1, IC, XIV, IEV)</td>
<td></td>
</tr>
<tr>
<td>CALL RTIN (N, ICA, IVA, IEV)</td>
<td>Set the N elements of array IVA to the latest (raw) values of the N real-time inputs listed in array ICA, with optional error code IEV.</td>
</tr>
<tr>
<td>CALL RTINX (N, ICA, IVA, IEV)</td>
<td></td>
</tr>
<tr>
<td>CALL RTIN (N, ICA, XVA, IEV)</td>
<td>Set the N elements of array XVA to the floating point values of the N analog-to-digital converter channels listed in array ICA, with optional error code IEV.</td>
</tr>
<tr>
<td>CALL RTINX (N, ICA, XVA, IEV)</td>
<td></td>
</tr>
<tr>
<td>CALL RTOUT ($\phi$, IC, IV, IEV)</td>
<td>Set real-time output IC with the formatted integer value IV, with optional error code IEV. (RTOUX is used if real-time inputs using RTINX are to be requested in the same program.)</td>
</tr>
<tr>
<td>CALL RTOUX ($\phi$, IC, IV, IEV)</td>
<td></td>
</tr>
<tr>
<td>CALL RTOUT (-1, IC, XIV, IEV)</td>
<td>Set digital-to-analog converter IC to the voltage XIV, with optional error code IEV.</td>
</tr>
<tr>
<td>CALL RTOUX (-1, IC, XIV, IEV)</td>
<td></td>
</tr>
<tr>
<td>CALL RTOUT (N, ICA, IVA, IEV)</td>
<td>Set the N real-time outputs listed in array ICA with the N formatted integer values in array IVA, with optional error code IEV.</td>
</tr>
<tr>
<td>CALL RTOUX (N, ICA, IVA, IEV)</td>
<td></td>
</tr>
<tr>
<td>CALL RTOUT (N, ICA, XVA, IEV)</td>
<td>Set the N digital-to-analog converters listed in array ICA to the N voltage values contained in array XVA, with optional error code IEV.</td>
</tr>
<tr>
<td>CALL RTOUX (N, ICA, XVA, IEV)</td>
<td></td>
</tr>
</tbody>
</table>
TABLE 3--Continued

<table>
<thead>
<tr>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL RTSET (IC, IFQ, BUF, IEV)</td>
<td>Automatically fill array BUF with real-time input IC at the frequency coded by IFQ, with optional error code IEV. (IFQ is 1 for 1000 Hz, 2 for 100 Hz, 3 for 10 Hz.)</td>
</tr>
<tr>
<td>CALL RTRST (IEV)</td>
<td>Restart all automatic real-time input previously requested through RTSET to the beginning of each array. Error code IEV is mandatory.</td>
</tr>
<tr>
<td>CALL UNFIX (TSN, IEV)</td>
<td>Stop the automatic real-time input requested through RTSET, and unfix the program from core upon completion. TSN must be the task name of the program, IEV is an optional error code.</td>
</tr>
<tr>
<td>CALL ADCNV (IV, XIV)</td>
<td>Change raw analog-to-digital converter value IV to the floating point value for voltage, XIV.</td>
</tr>
<tr>
<td>CALL IADCV (IV, IX)</td>
<td>Change raw analog-to-digital converter value IV to the scaled integer value IX.</td>
</tr>
</tbody>
</table>

Note: The program listings given for these subroutines (except UNFIX) contain more detailed explanations of the functions of the subroutines and their parameters, including the meanings for values of the error return code IEV.
APPENDIX B

Dual-Processor System Program Listings

Following are assembly listings for the programs required to implement the dual-processor system: PATCH, MOD10F, DPP02, PRO01, LPR01, REM0S (parts copyright by Digital Equipment Corporation and reprinted by permission), KSG, RTIO, RTIN, RTOUT, RTSET, RTRST, ADCNV, and GRAPH. A brief description of these programs is given in Appendix A, and a complete discussion is part of the listing for each routine. (All characters to the right of a slash ("/"") are comments.)
.PATCH

.TITLE MONITOR MODIFICATIONS FOR THE DUAL PROCESSOR.
.OCT

/ THIS "TASK" CONTAINS CODE TO IMPLEMENT RSX-PLUS MODIFICATIONS
/ NECESSARY FOR EFFICIENT USE OF THE DUAL PROCESSOR CONFIGURATION.
/ THIS CODE MUST REMAIN FIXED IN CORE AT ALL TIMES. IT IS NEVER
/ SCHEDULED FOR EXECUTION AS A REGULAR TASK, BUT IS ENTERED BY
/ DIRECT REFERENCE FROM THE MAIN SECTION OF THE MONITOR. MINOR
/ CHANGES IN THE MAIN SECTION OF THE MONITOR ARE MADE BY THE MCR
/ TASK ..., OPP, WHICH OVERWRITES SELECTED INSTRUCTIONS IN THE
/ ORIGINAL DEC-SUPPLIED MONITOR.
/ THIS TASK MUST BE FIXED IN A PARTITION STARTING AT 13000 (OCTAL).
/ IN BUILDING THIS TASK, THE TKB OPTION "EXX" MUST BE SPECIFIED.
/ THIS TASK WILL BE FIXED IN CORE, OBVIOUSLY, THE TASK NAME
/ ".PATCH" MUST BE USED.
/ THIS CODE IS SUPPORTED BY P. N. BARTRAM,
/
/ THE FOLLOWING ARE THE STARTING ADDRESSES FOR THE ROUTINES
/ RESIDING HERE:
/ 13302 TABLES
/ 13340 CAL FUNCTION 31 - GET VALUE FROM SHARED CORE.
/ 13420 CAL FUNCTION 32 - PLACE VALUE IN SHARED CORE.
/ 13520 CAL FUNCTION 33 - SET INPUT REQUEST TABLE ENTRY.
/ 13640 CAL FUNCTION 34 - RESET INPUT REQUEST TABLE ENTRIES.
/ 14020 ADDITIONAL SECTION FOR CAL FUNCTION 16 - UNFIX.
/ 14140 HANDLER FOR INTERRUPTS FROM THE SECONDARY PROCESSOR.
/
/ THIS CODE MUST BE RUN IN BANK 1 MEMORY.

22281 A BANK=1
/ THE NUMBER OF ANALOG TO DIGITAL CONVERTER CHANNELS IS
/ CURRENTLY 64.
22182 A NDAC=120
/ THE NUMBER OF DIGITAL INPUT GROUPS IS CURRENTLY 1.
22221 A NOI=1
/ THE NUMBER OF DIGITAL TO ANALOG CONVERTERS IS CURRENTLY 8.
22221 A NDAC=10
/ THE NUMBER OF DIGITAL OUTPUT GROUPS IS CURRENTLY 3.
22223 A NOO=3
/ THE ADDRESS OF THE FAST CLOCK COUNTER, ALSO THE BASE ADDRESS
/ OF THE INPUT VALUE TABLE IN SHARED MEMORY, IS CURRENTLY 67677.
267677 A ADCLK=67677
MONITOR MODIFICATIONS FOR THE DUAL PROCESSOR,

/ THE BASE ADDRESS OF THE OUTPUT VALUE TABLE IN SHARED MEMORY IS
/ CURRENTLY 067437.
/ THE BASE ADDRESS OF THE PROTECT TABLE FOR OUTPUTS (CORRESPONDS
/ TO OUTPUT DEVICE NUMBER 8) IS 15 ABOVE THE FIRST TABLE POSITION.
/ (THUS THERE MAY BE A MAXIMUM OF 16 DIGITAL OUTPUT GROUPS, AND 16 DIGITAL TO ANALOG CONVERTERS.)
/ PROT=17 * PRCT
/ THE FOLLOWING TABLES ARE REQUIRED BY THE DUAL PROCESSOR HANDLERS:
/ KHE, BLOCK 100 /1 KHZ INPUT REQUEST TABLE,
/ WUNHE, BLOCK 100 /100 Hz INPUT REQUEST TABLE,
/ TENHE, BLOCK 100 /10 WE INPUT REQUEST TABLE,
/ PRCT, BLOCK 10 /PROTECT TABLE FOR OUTPUTS,
/ IF ADDITIONAL CODE IS REQUIRED, SOME OF THE CONSTANTS AND STORAGE
/ CELLS CAN BE RELOCATED BETWEEN THE CODE FOR THE VARIOUS FUNCTIONS,
/ FILLING IN SOME OF THE "HOLES". THE MONITOR IS MODIFIED (BY THE
/ MCR FUNCTION OPP - DUAL PROCESSOR PATCHES) TO EXPECT THE TABLES
/ AND SECTIONS OF CODE TO APPEAR WHERE THEY ARE, THEREFORE THE
/ STARTING ADDRESSES OF THE BLOCKS OF CODE FOLLOWING MUST NOT BE
/ ALTERED. HOPEFULLY, BY MOVING THE STORAGE CELLS AND CONSTANTS,
/ ENOUGH CONTIGUOUS MEMORY WILL BECOME AVAILABLE FOR EXPANSION
/ WITHOUT REQUIRING A PARTITION LARGER THAN 1400 (OCTAL).
/ START=KHE

00000 R ST
.PATCH

*** GET VALUE FROM INPUT TABLE CAL DIR.

.TITLE *** GET VALUE FROM INPUT TABLE CAL DIR.

/ CAL FUNCTION DIRECTIVE 31.
/ GET A WORD FROM THE TABLE IN SHARED MEMORY.
/ TABLE ENTRY 0 IS THE (5KHz) CLOCK COUNTER.
/ POSITIVE TABLE NUMBERS ARE ANALOG TO DIGITAL CONVERTER CHANNELS.
/ NEGATIVE TABLE NUMBERS ARE DIGITAL INPUT GROUPS.
/ A FOUR WORD CAL PARAMETER BLOCK IS USED:
/ 1 FUNCTION CODE (31),
/ 2 EVENT VARIABLE ADDRESS,
/ 3 TABLE NUMBER ("DEVICE NUMBER"),
/ 3 VALUE RETURNED FOR REQUESTED TABLE ENTRY,

/ IF THE DIRECTIVE IS REJECTED, THE EVENT VARIABLE, IF SPECIFIED, IS
/ SET TO -121 FOR TABLE NUMBER OUT-OF-RANGE,
/ IF THE DIRECTIVE IS ACCEPTED, THE VALUE OF THE TABLE ENTRY WILL BE
/ PLACED IN WORD 3 OF THE USER CAL PARAMETER BLOCK, AND THE EVENT
/ VARIABLE, IF SPECIFIED, WILL BE SET TO *1.
/ THE REGISTERS USED BY THIS FUNCTION ARE THE AC AND XR. CR1,
/ R2, R3, X12, X15, X14, AND L20 ARE USED BY THE CAL DISPATCH
/ AND RETURN ROUTINE.

.EJECT
**PATCH** ••• GET VALUE FROM INPUT TABLE CAL DIR.

112 03340 R 7777774 A LAW -4/ CHECK RANGE OF CB (NORMAL MODE) AND
113 03341 R 121233 R JMS* CBPRX / SET X14 TO THE 17-BIT EVENT VARIABLE ADH.
114 03342 R 724028 A PXA / GET THE ADDRESS OF THE CAL PARAMETER BLOCK
115 03343 R 341229 R TAD XADJ / ADJUST FOR THIS MEMORY BANK AND
116 03344 R 721230 A PAX / PLACE BACK IN THE XR.
117 03345 R 212222 A LAC 2,x / GET THE TABLE NUMBER.
118 03346 R 341224 R TAD LDI / IS IT TOO LOW?
119 03347 R 741177 A SPA
120 03350 R 621240 R JMP* CX101 / YES, SET E.V. TO -101 AND EXIT.
121 03351 R 341223 R TAD MLGT / NO, IS IT TOO HIGH?
122 03352 R 741178 A SMA
123 03353 R 621240 R JMP* CX101 / YES, SET E.V. TO -101 AND EXIT.
124 03354 R 341222 R TAD TBAL / NO, TABLE NUMBER IS O.K., GET
125 03355 R 241334 R DAC PNTR / VALUE FOR THIS ENTRY.
126 03356 R 221334 R LAC* FNTR
127 03357 R 052023 A DAC 3,x / PLACE VALUE IN USER CAL PARAMETER BLOCK.
128 03360 R 621236 R JMP* CXSUC / EXIT, WITH A SMILE AND E.V. = +1.
PLACE VALUE FOR OUTPUT TABLE CAL DIR.

TITLE *** PLACE VALUE FOR OUTPUT TABLE CAL DIR.
LOC START=400

CAL FUNCTION DIRECTIVE 32,
PLACE A WORD IN THE OUTPUT TABLE FOR THE SECONDARY PROCESSOR,
PROVIDING THE TABLE ENTRY HAS NOT BEEN ASSIGNED TO ANOTHER
TASK.

A FOUR WORD CAL PARAMETER BLOCK IS USED:
0 FUNCTION CODE (32),
1 EVENT VARIABLE ADDRESS,
2 TABLE ENTRY NUMBER ("DEVICE NUMBER"),
3 VALUE TO BE PLACED IN THE TABLE.

IF THE DIRECTIVE IS REJECTED, THE EVENT VARIABLE, IF SPECIFIED,
WILL BE SETToOne OF THE FOLLOWING:
-70 DIGITAL TO ANALOG CONVERTER WORD IMPROPERLY FORMATTED,
-701 TABLE NUMBER IS OUT-OF-RANGE,
-702 TABLE ENTRY IS PROTECTED AND ASSIGNED TO ANOTHER TASK,
-293 CAL WAS NOT TASK ISSUED,

THE REGISTERS USED BY THIS FUNCTION ARE THE AC AND THE XR. [R1,
R2, R3, X10, X11, X14, AND L20 ARE USED BY THE CAL DISPATCH
AND RETURN ROUTINE.]
**PLACE VALUE FOR OUTPUT TABLE CAL DIR.**

PATCH  

155  02429 R 777774 A
LAW -4 /CHECK RANGE OF CPB (NORMAL NODE) AND SET X14
156  02430 R 121233 R
LMS+ CPBRX / TO THE 17-BIT EVENT VARIABLE ADDRESS.
157  02431 R 221247 R
LAC+ X11 /HAS THIS CAL ISSUED BY A TASK?
158  02432 R 742277 A
SZA
159  02433 R 621242 R
JMP+ CX283 /NO, EXIT WITH E.V. = -203.
160  02434 R 742278 A
PXAX /YES, GET ADDRESS OF THE CAL PARAMETER BLOCK
161  02435 R 341225 R
TAD XAOJ /ADJUST FOR THIS BANK, AND PLACE BACK
162  02436 R 721270 A
PAX /IN THE XR.
163  02437 R 212270 A
LAC 2,X /GET THE TABLE ENTRY NUMBER DESIRED.
164  02438 R 341226 R
TAD LPOO /IS THE ENTRY NUMBER TOO LOW?
165  02439 R 741177 A
SPA
166  02440 R 621243 R
JMP+ CX101 /YES, SET E.V. = -101 AND EXIT.
167  02441 R 341233 R
TAD MLGTH /NO, IS IT TOO HIGH?
168  02442 R 747370 A
SHAISZA /SKIPS IF EITHER CONDITION SATISFIED.)
169  02443 R 621243 R
JMP+ CX101 /YES, SET E.V. = -101 AND EXIT.
170  02444 R 041274 R
DAC NUBH /NO, IT IS FINE, SAVE (NUMBER - # OF DAC'S).
171  02445 R 341231 R
TAD PRRT /GET THE PROTECTION CODE.
172  02446 R 741275 R
DAC TEMP
173  02447 R 221275 R
LAC+ TEMP
174  02448 R 541261 R
SAD NA /IF THIS ENTRY POSITION IS NOT PROTECTED.
175  02449 R 625635 R
JMP ENDK /GO SET VALUE IN TABLE.
176  02450 R 041279 R
DAC TEMP /IT IS PROTECTED, SAVE PROTECTION CODE.
177  02451 R 221234 R
LAC+ CURTSK /MATCH PROTECTION CODE WITH FIRST THREE
178  02452 R 732272 A
DAC +2 /LETTERS OF THE CURRENT TASK NAME.
179  02453 R 341236 R
DAC TEMP
180  02454 R 221276 R
LAC+ TEMP
181  02455 R 541275 R
SAD TEMP /ARE THEY THE SAME?
182  02456 R 741270 A
SAP
183  02457 R 621241 R
JMP+ CX102 /NO, ENTER PROHIBITED, EXIT WITH E.V. = -102.
184  02458 R 221274 R
ENDK LAC NUBH /YES, IS THIS A DAC SETTING?
185  02459 R 341227 R
TAD LDAC /SKIPS IF BOTH CONDITIONS SATISFIED.)
186  02460 R 741270 A
SPAINSAGE
187  02461 R 625641 R
JMP SET /NO, NO FORMAT CHECK REQUIRED.
188  02462 R 211275 R
DAC TEMP /YES, STORE DAC NUMBER.
189  02463 R 211733 R
DAC L40 X /GET VALUE TO BE SET,
190  02464 R 747231 A
TCA /EXTRACT THE UNCOMPLEMENTED DAC NUMBER.
191  02465 R 521262 R
AND MASK1 /ARE THE DAC NUMBERS EQUAL?
192  02466 R 541275 R
SAD TEMP /YES, ENTRY MAY BE PLACED IN THE TABLE.
193  02467 R 540241 R
JMP SET /YES, ENTRY MAY BE PLACED IN THE TABLE.
194  02468 R 777712 A
LAW -70 /NO, SET E.V. TO -70 AND EXIT.
195  02469 R 621239 R
JMP+ CXD4D
196  02470 R 201274 R
SET LAC NUBH /FINALLY READY TO PLACE THE WORD IN
197  02471 R 341232 R
TAD TBLX /THE TABLE. GET THE ADDRESS OF THE
198  02472 R 041279 R
DAC TEMP /TABLE POSITION.
199  02473 R 213032 A
LAC 3,X /GET THE VALUE TO BE SET FROM THE CAL
200  02474 R 621279 R
DAC TEMP /PARAMETER BLOCK AND SET IT IN THE TABLE.
201  02475 R 621236 R
JMP+ CXSUC /EXIT, WITH E.V. = -1 AND A GRIN.
SET ENTRY IN REAL-TIME REQUEST TABLE CAL DIR.

.TITLE SET ENTRY IN REAL-TIME REQUEST TABLE CAL DIR.
.LOC START+500

/ CAL FUNCTION DIRECTIVE 33.
/ PLACE A REQUEST FOR AUTOMATIC REAL-TIME INPUT IN THE
/ REQUEST TABLE. PERMITTED FREQUENCIES ARE 1 KHZ (CODE 1),
/ 100 HZ (CODE 2), AND 10 HZ (CODE 3). EACH FREQUENCY HAS A
/ SEPARATE TABLE.
/ EACH TABLE IS CONSTRUCTED OF UP TO FIFTEEN ENTRIES. EACH ENTRY
/ TAKING FOUR CONTIGUOUS WORDS. THE LAST "ENTRY" (SIXTEEN) OF
/ EACH TABLE MUST BE ZERO TO SIGNAL END-OF-TABLE. FOR EACH ENTRY:
/ WORD 0 ADDRESS POINTER TO WHERE IN THE USER BUFFER THE
/ INPUT VALUE IS TO BE STORED. (INITIALLY SET TO THE BEGINNING ADDRESS OF THE BUFFER.)
/ WORD 1 DEVICE NUMBER: NEGATIVE FOR DIGITAL INPUT GROUPS,
/ ZERO FOR THE CLOCK COUNTER, OR POSITIVE FOR
/ ANALOG TO DIGITAL CONVERTER CHANNELS.
/ WORD 2 LAST ADDRESS OF BUFFER.
/ WORD 3 STARTING ADDRESS OF BUFFER.
/ A SIX-WORD CAL PARAMETER BLOCK IS USED:
/ 2 FUNCTION CODE (33),
/ 1 EVENT VARIABLE ADDRESS,
/ 2 DEVICE NUMBER,
/ 3 FIRST ELEMENT ADDRESS IN USER BUFFER,
/ 4 LENGTH OF THE USER BUFFER,
/ 5 FREQUENCY CODE.
/ IF THE DIRECTIVE IS REJECTED, THE EVENT VARIABLE, IF ONE IS
/ SPECIFIED, IS SET TO ONE OF THE FOLLOWING VALUES:
/ -15 TABLE FOR SELECTED FREQUENCY IS FULL.
/ -30 BUFFER IS NOT CONTAINED WITHIN THE PARTITION
/ OF THE TASK REQUESTING THIS CAL.
/ -72 THE FREQUENCY CODE IS NOT 1, 2, OR 3.
/ -101 THE DEVICE NUMBER IS OUT-OF-RANGE.
/ -233 THE CAL WAS NOT TASK ISSUED.
/ -207 THE TASK ISSUING THE CAL IS NOT FIXED IN CORE.
/ IF THE DIRECTIVE IS ACCEPTED, THE ENTRY IS MADE IN THE PROPER TABLE.
/ AND THE EVENT VARIABLE, IF SPECIFIED, IS SET TO +1, THE VALUE OF
/ SELECTED INPUT DEVICE IS THEN PLACED IN THE USER BUFFER (RING FILLED)
/ AT THE PROPER TIME. THE TABLE ENTRY IS REMOVED WHEN THE TASK IS
/ REMOVED OR UNFIXED FROM CORE.
/ NOTE THE FOLLOWING TIMING PROBLEM: IF A LARGE NUMBER OF
/ ENTRIES APPEAR IN THESE TABLES, THE SECONDARY PROCESSOR
/ MAY TRANSFER THE NEXT VALUE FOR A LOW-NUMBERED INPUT BEFORE
/ THE INTERRUPT HANDLER HAS TIME TO TRANSFER THE CURRENT
/ VALUE TO THE USER BUFFER. LOW FREQUENCY SAMPLES ARE THE
/ LAST TO BE TRANSFERRED, THUS MOST LIKELY TO BE INVOLVED.
/ THIS MAY CAUSE A 1 MILLISECOND TIME SKEW FOR SOME 10
*** SET ENTRY IN REAL-TIME REQUEST TABLE CAL DIR.

/* hz sample frequency (and possibly some 100 hz sample
/// frequency) requests, depending on the channel number
/// and number of other requests.
/
/ the registers used by this function are the ac and xr, [acr,
/ r2, r3, x10, x11, x14, and l20 are used by the cal dispatch
/ and return routine.]

.eject
PATCH

SET ENTRY IN REAL-TIME REQUEST TABLE CAL DIR.

.PATCH ••• SET ENTRY IN REAL-TIME REQUEST TABLE CAL DIR.

PATCH 253 R 777772 A LAY -6 /CHECK THE RANGE OF THE CPB (NORMAL MODE) AND
254 R 777772 A CMS* CPBRX / SET X14 TO THE 17-BIT EVENT VARIABLE ADDRESS.
255 R 777772 A LAC* X11 /HAS THIS CAL TASK ISSUED?
256 R 777772 A SIA
257 R 777772 A JNP* CX203 /NO, EXIT WITH E.V. = -203.
258 R 777772 A LAC* CURT5K /YES, GET ACTIVE TASK LIST NODE ADDRESS
259 R 777772 A AAC -6 /FOR THIS TASK.
260 R 777772 A DAC TEMP2 /GET THE SYSTEM TASK LIST NODE ADDRESS
261 R 777772 A LAC* TEMP2 /GET THE SYSTEM TASK LIST NODE ADDRESS
262 R 777772 A AAC +4 /WORD 6 OF ATL NODE.
263 R 777772 A DAC TEMP2 /GLOBAL TEMP.
264 R 777772 A LAC* TEMP2 /GET FLAGS WORD FROM STL (WORD 4).
265 R 777772 A AND MASK3 /IS THE TASK ACTIVE AND FIXED IN CORE?
266 R 777772 A SAD MASK3
267 R 777772 A SKP
268 R 777772 A JNP* CX207 /NO, EXIT WITH E.V. = -207.
269 R 777772 A LAC* TEMP2 /YES, ASSURE THAT BIT 4 IS SET.
270 R 777772 A AND MASK4
271 R 777772 A XOR MASK5
272 R 777772 A DAC TEMP2
273 R 777772 A IS2 TEMP2 /GET THE PARTITION BLOCK DESCRIPTOR LIST
274 R 777772 A LAC* TEMP2 /ADDRESS (WORD 9 OF STL).
275 R 777772 A AAC +4 /GET THE PARTITION STARTING ADDRESS (WORD
276 R 777772 A DAC TEMP2 /4 OF PBDL) ADDRESS POINTER.
277 R 777772 A PC1 /ADJUST THE INDEX REGISTER (CPB ADDRESS)
278 R 777772 A TAD XADJ /FOR THIS MEMORY BANK.
279 R 777772 A PAX
280 R 777772 A LAC* L20 /CHECK IF THE USER MODE BIT IS SET.
281 R 777772 A RTL
282 R 777772 A C835 R 751172 A SPAICLA /IS THE TASK IN USER MODE?
283 R 777772 A LAC* CTB1AS /YES, GET THE RELOCATION FACTOR FROM MONITOR.
284 R 777772 A TAD 3, X /GET THE BUFFER STARTING ADDRESS (ADD RE-
285 R 777772 A DAC TEMP4 /LOCATION FACTOR IF USER MODE). SAVE,
286 R 777772 A TCA
287 R 777772 A C836 R 312146 A LAC* TEMP2 /IS THE STARTING ADDRESS GREATER THAN THE
288 R 777772 A TAD TEMP4 /PARTITION BASE ADDRESS? (BFR ADR - PAR ADR <= 01)
289 R 777772 A TCA
290 R 777772 A C834 R 312177 R SMASEA /PARTITION BASE ADDRESS? (BFR ADR - PAR ADR <= 01)
291 R 777772 A JNP* CX38 /NO, EXIT WITH E.V. = -30.
292 R 777772 A LAC* TEMP2 /YES, GET UPPER LIMIT OF PARTITION BY ADDING THE
293 R 777772 A IS2 TEMP2 /SIZE (WORD 5 OF PBDL) TO THE BASE ADDRESS.
294 R 777772 A DAC TEMP2
295 R 777772 A TAD TEMP2 /IS THE BUFFER END ADDRESS WITHIN THE PARTITION?
296 R 777772 A TCA
297 R 777772 A C831 R 314131 R TAD TEMP4 /BFR BASE ADR + BFR LENGTH - PAR BASE ADR - PAR
298 R 777772 A TAD 4, X /LENGTH <= 01?
299 R 777772 A TCA
300 R 777772 A SMASEA
301 R 777772 A JNP* CX38 /NO, EXIT WITH E.V. = -30.
302 R 777772 A C837 R 743127 R LAC* TEMP2
303 R 777772 A TAD 2, X /YES, CHECK THE DEVICE NUMBER NEXT,
304 R 777772 A TAD L01 /IS THE INPUT DEVICE NUMBER TOO LOW?
305 R 777772 A SPA
306 R 777772 A TCA
308 R 777772 A LAC TAD HLGT /NO, IS IT TOO HIGH?
309 R 777772 A SMASEA
310 R 777772 A JNP* CX131 /YES, EXIT WITH E.V. = -101.
311 R 777772 A LAC SMI
312 R 777772 A TAD 5, X /NO, DEVICE NUMBER IS O.K. GET THE FREQUENCY
313 R 777772 A SMASEA
314 R 777772 A JNP* CX141
315 R 777772 A LAC SMI
316 R 777772 A TAD 6, X
SET ENTRY IN REAL-TIME REQUEST TABLE CAL DIR.

/ CODE. IS IT 1 KHz?

/100 Hz?

/15 Hz?

/NO, IT IS NONE OF THESE, EXIT WITH

/E.V. -70.

/SELECTED FREQUENCY.

/GET PROPER TABLE STARTING ADDRESS FOR THE

/SAVE FOR FUTURE REFERENCE.

/INITIALIZE TABLE ENTRY COUNTER.

/IS IT ZERO?

/YES, REPLACE IT WITH THE NEW ENTRY.

/NO, LEAVE IT ALONE AND LOOK AT THE

/NEXT ENTRY.

/HAS THAT LAST ENTRY THE LAST ENTRY OF THE TABLE?

/NO, TRY THE NEXT ENTRY.

/YES, SET E.V. = -15 AND EXIT.

/FILL

/DISABLE INTERRUPTS WHILE CHANGING THE TABLE.

/PLACE THE USER BUFFER STARTING ADDRESS AS

/ADDRESS POINTER IN THE TABLE (FIRST WORD OF

/ADDRESS ENTRY).

/PLACE THE INPUT DEVICE NUMBER AS THE NEXT

/WORD OF THE TABLE ENTRY.

/TRANSFER THE BUFFER ENDING ADDRESS (START

/ADDRESS).

/AAD

/* LENGTH - 1),

/TRANSFER THE BUFFER STARTING ADDRESS.

/ALLOW INTERRUPTS NOW, TABLE IS COMPLETE.

/SET E.V. TO -1, AND EXIT WITH PRIDE.
RESET REAL-TIME REQUEST TABLE CAL DIR.

 TITLE *** RESET REAL-TIME REQUEST TABLE CAL DIR.
 LOC START=649

CAL FUNCTION DIRECTIVE 34.
RESET THE ADDRESS POINTERS FOR ALL ENTRIES FOR THIS TASK IN THE
AUTOMATIC REAL-TIME INPUT REQUEST TABLE.

A TWO-WORD CAL PARAMETER BLOCK IS USED:
FUNCTION CODE (34),
eVENT VARIABLE ADDRESS,

IF THE DIRECTIVE IS REJECTED, THE EVENT VARIABLE, IF ONE IS
SPECIFIED, IS SET TO ONE OF THE FOLLOWING VALUES:
-109 NO ENTRIES FOR THIS TASK WERE FOUND IN THE
INPUT REQUEST TABLES,
-233 THE CAL WAS NOT TASK ISSUED.

IF THE DIRECTIVE IS ACCEPTED, THE POINTERS FOR EACH ENTRY FOR
THIS TASK IS RESET TO THE START OF THE BUFFER FOR THAT ENTRY,
AND THE EVENT VARIABLE, IF ONE IS SPECIFIED, IS SET EQUAL TO
THE NUMBER OF 1 KHZ CLOCK PULSES FROM THE TIME OF RESET TO THE
TIME OF NEXT SERVICING THE 10 Hz REQUEST TABLE, (THE NEXT
PULSE = 1), THUS THE REQUESTOR USING DIFFERENT FREQUENCIES CAN
FIND THE TIME SHIFT FOR SYNCHRONIZATION. IF N IS THE VALUE OF
THE EVENT VARIABLE RETURNED, AND K IS THE INTEGER QUOTIENT N/10
(BASE 10), THE N-TH VALUE IN A 1 KHZ INPUT BUFFER, THE K+1-TH
VALUE IN A 100 Hz BUFFER, AND THE FIRST VALUE IN A 10 Hz BUFFER
WERE ALL SAMPLED ON THE SAME CLOCK PULSE. SIMILARLY, IF J IS THE
REMAINDER OF N/10 (BASE 10), THE J-TH VALUE IN A 1 KHZ INPUT
BUFFER AND THE FIRST VALUE IN A 100 Hz BUFFER WERE SAMPLED AT
THE SAME CLOCK PULSE.

ER1, R2, R3, X10, X11, X14, AND L20 ARE USED BY THE CAL DISPATCH
AND RETURN ROUTINE.
### Real-time Request Table Control

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>395</td>
<td>32640 A</td>
<td>R 777776</td>
<td>LAW -2 /Check the range of the CPB (Normal Mode)</td>
</tr>
<tr>
<td>396</td>
<td>32641 R</td>
<td>R 121233</td>
<td>JMB* CPBRX /And set X14 to the 17-bit event variable</td>
</tr>
<tr>
<td>397</td>
<td>32642 R</td>
<td>R 221247</td>
<td>X11 /Address, SZA</td>
</tr>
<tr>
<td>398</td>
<td>32643 R</td>
<td>R 742280 A</td>
<td>SEA /Is this CAL task issued?</td>
</tr>
<tr>
<td>399</td>
<td>32644 R</td>
<td>R 621242 R</td>
<td>JMB* CX223 /No, exit with E.V. = -203,</td>
</tr>
<tr>
<td>400</td>
<td>32645 R</td>
<td>R 221234 R</td>
<td>LAC* CUTF5 /Yes, get the active task list (ATL) node address for this task. Then get the system task list (STL) node address (_word 6 of PBDL).</td>
</tr>
<tr>
<td>401</td>
<td>32646 R</td>
<td>R 722756 A</td>
<td>XA /Address of the active task list.</td>
</tr>
<tr>
<td>402</td>
<td>32647 R</td>
<td>R 241372 R</td>
<td>DAC TEMP5 /Task list (STL) node address (Word 5 of PBDL).</td>
</tr>
<tr>
<td>403</td>
<td>32648 R</td>
<td>R 221322 R</td>
<td>LAC* TEMP5 /The ATL node.</td>
</tr>
<tr>
<td>404</td>
<td>32649 R</td>
<td>R 723755 A</td>
<td>AAC +5 /Get the partition block descriptor list.</td>
</tr>
<tr>
<td>405</td>
<td>32650 R</td>
<td>R 241372 R</td>
<td>DAC TEMP5 /PBDL address (Word 5 of the STL node).</td>
</tr>
<tr>
<td>406</td>
<td>32651 R</td>
<td>R 221322 R</td>
<td>LAC* TEMP5 /</td>
</tr>
<tr>
<td>407</td>
<td>32652 R</td>
<td>R 723755 A</td>
<td>AAC +4 /Get the partition base (starting) address</td>
</tr>
<tr>
<td>408</td>
<td>32653 R</td>
<td>R 241372 R</td>
<td>DAC TEMP5 /Word 4 of the PBDL.</td>
</tr>
<tr>
<td>409</td>
<td>32654 R</td>
<td>R 221322 R</td>
<td>LAC* TEMP5 /</td>
</tr>
<tr>
<td>410</td>
<td>32655 R</td>
<td>R 742331 A</td>
<td>TCA /Save the complement of this.</td>
</tr>
<tr>
<td>411</td>
<td>32656 R</td>
<td>R 243389 R</td>
<td>DAC PARBS /Get the partition length (Word 5 of PBDA).</td>
</tr>
<tr>
<td>412</td>
<td>32657 R</td>
<td>R 241372 R</td>
<td>LAC* TEMP5 /</td>
</tr>
<tr>
<td>413</td>
<td>32658 R</td>
<td>R 221322 R</td>
<td>LAC* TEMP5 /</td>
</tr>
<tr>
<td>414</td>
<td>32659 R</td>
<td>R 742331 A</td>
<td>TCA /</td>
</tr>
<tr>
<td>415</td>
<td>32660 R</td>
<td>R 341353 R</td>
<td>DAC PARBS /Form the complement of the address of the partition.</td>
</tr>
<tr>
<td>416</td>
<td>32661 R</td>
<td>R 241372 R</td>
<td>DAC PARTP /Partition top address + 1, and save it.</td>
</tr>
<tr>
<td>417</td>
<td>32662 R</td>
<td>R 221257 R</td>
<td>LAC* XTABS /Set the index reg. to reference the 1 Khz.</td>
</tr>
<tr>
<td>418</td>
<td>32663 R</td>
<td>R 721273 A</td>
<td>PAX /Request table.</td>
</tr>
<tr>
<td>419</td>
<td>32664 R</td>
<td>R 212854 R</td>
<td>LAC* LATBL /Set Auto-increment register 13 to reference the temporary storage for entries to be reset.</td>
</tr>
<tr>
<td>420</td>
<td>32665 R</td>
<td>R 761251 A</td>
<td>DAC* XI3 /Get the address pointer is below the temporary storage.</td>
</tr>
<tr>
<td>421</td>
<td>32666 R</td>
<td>R 777750 A</td>
<td>LAW -33 /Set a counter to assure the temporary storage is not overfilled.</td>
</tr>
<tr>
<td>422</td>
<td>32667 R</td>
<td>R 441343 R</td>
<td>DAC COUNT /Is not overfilled.</td>
</tr>
<tr>
<td>423</td>
<td>02674 R</td>
<td>R 217223 R</td>
<td>RST3 LAC 0.X /Get an entry from the tables.</td>
</tr>
<tr>
<td>424</td>
<td>02675 R</td>
<td>R 741223 A</td>
<td>SNA /Is this the end of a table?</td>
</tr>
<tr>
<td>425</td>
<td>02676 R</td>
<td>R 627714 R</td>
<td>JMB* RST1 /Yes, go on to the next table.</td>
</tr>
<tr>
<td>426</td>
<td>02677 R</td>
<td>R 341350 R</td>
<td>TAD PARBS /No, is this entry for this task?</td>
</tr>
<tr>
<td>427</td>
<td>02678 R</td>
<td>R 741172 A</td>
<td>SPA /</td>
</tr>
<tr>
<td>428</td>
<td>02679 R</td>
<td>R 627734 R</td>
<td>JMB* RST2 /No, the address pointer is below the base address of the partition.</td>
</tr>
<tr>
<td>429</td>
<td>02680 R</td>
<td>R 217223 R</td>
<td>LAC PARTP /</td>
</tr>
<tr>
<td>430</td>
<td>02681 R</td>
<td>R 341352 R</td>
<td>TAD /</td>
</tr>
<tr>
<td>431</td>
<td>02682 R</td>
<td>R 741253 A</td>
<td>SPA /</td>
</tr>
<tr>
<td>432</td>
<td>02683 R</td>
<td>R 627724 R</td>
<td>JMB* RST2 /No, the address pointer is above the top address of the partition.</td>
</tr>
<tr>
<td>433</td>
<td>02684 R</td>
<td>R 741253 A</td>
<td>SPA /</td>
</tr>
<tr>
<td>434</td>
<td>02685 R</td>
<td>R 627724 R</td>
<td>JMB* PXA /Top address of the partition.</td>
</tr>
<tr>
<td>435</td>
<td>02686 R</td>
<td>R 724372 A</td>
<td>DAX* 02013 /Yes, get the address (+ XAOJ) of the entry and store it temporarily.</td>
</tr>
<tr>
<td>436</td>
<td>02687 R</td>
<td>R 441343 R</td>
<td>TSE COUNT /Entry and store it temporarily.</td>
</tr>
<tr>
<td>437</td>
<td>02688 R</td>
<td>R 627724 R</td>
<td>JMB* RST2 /Go on to the next request table entry.</td>
</tr>
<tr>
<td>438</td>
<td>02689 R</td>
<td>R 777753 A</td>
<td>LAW -30 /If the temporary storage table is filled.</td>
</tr>
<tr>
<td>439</td>
<td>02690 R</td>
<td>R 607233 A</td>
<td>JMB* RST5 /Process only the first 24 request entries.</td>
</tr>
<tr>
<td>440</td>
<td>02691 R</td>
<td>R 724273 R</td>
<td>RST1 PXA /Request table end found, get the table.</td>
</tr>
<tr>
<td>441</td>
<td>02692 R</td>
<td>R 507253 R</td>
<td>TAD /Address pointer, was this the last table?</td>
</tr>
<tr>
<td>442</td>
<td>02693 R</td>
<td>R 341272 R</td>
<td>TAD /</td>
</tr>
<tr>
<td>443</td>
<td>02694 R</td>
<td>R 741270 A</td>
<td>SPA /</td>
</tr>
<tr>
<td>444</td>
<td>02695 R</td>
<td>R 627726 R</td>
<td>JMB* RST4 /Yes, process the changes.</td>
</tr>
<tr>
<td>445</td>
<td>02696 R</td>
<td>R 341473 R</td>
<td>TAD PX3X /No, increase the address pointer for the next table.</td>
</tr>
<tr>
<td>446</td>
<td>02697 R</td>
<td>R 721228 A</td>
<td>PAX /</td>
</tr>
</tbody>
</table>
**RESET REAL-TIME REQUEST TABLE CAL DIR.**

00723 R 622674 R JMP RST3 /GO TO WORK ON THE NEXT REQUEST TABLE.

00724 R 73224 A / JMP *AXR

00725 R 62674 R / JMP RST3 /READY TO LOOK AT THE NEXT ENTRY IN THE CUR-R.

00732 R 201343 R / RST4 / INCREASE THE POINTER TO IT.

00733 R 041343 R RST5 /COUNT /FIND OUT HOW MANY ENTRIES ARE TO BE CHANGED.

00734 R 221245 A LAC LTABL /CHANGES, AND X13 TO THE START OF THE

00735 R 741251 R DAC X13 /TEMPORARY STORAGE TABLE.

00736 R 723532 A IOT 55222 /DISABLE API BREAKS WHILE CHANGING THE TABLE.

00737 R 222733 A RST6 /LAC* 0013 /GET THE ADDRESS OF THE TABLE ENTRY TO BE

00738 R 741343 R PAX /CHANGED.

00739 R 211233 A LAC X13 /GET THE BUFFER STARTING ADDRESS FOR THAT

00740 R 74522 A DAC 0.X /ENTRY, AND SET THE BUFFER POINTER TO IT.

00741 R 74724 A IOT 55211 /IS THIS THE END OF THE CHANGES?

00742 R 221233 R JPH RST6 /NO, GO BACK AND DO MORE.

00743 R 741343 R LAC INHUN /YES, GET THE COUNTERS FOR SERVICING

00744 R 741343 R DAC TMPR /THE 100 Hz AND 10 Hz REQUEST TABLES.

00745 R 221233 R LAC INTEN

00746 R 221233 R IOT 55211 /ENABLE API BREAKS NOW.

00747 R 744274 A CALLICHA /GET THE NUMBER OF (1 kHz) CLOCK PULSES

00748 R 74522 A MUL 12 /FROM NOW THAT THE 10 Hz REQUEST TABLE

00749 R 53342 A 12 /WILL BE SERVICED.

00750 R 32112 R LACQ

00751 R 32112 R LACQ

00752 R 241322 R DAC TEMPS

00753 R 021012 A DAC TEMPS

00754 R 641022 A DAC TMPR

00755 R 641322 R DAC TMPR

00756 R 221233 R LAC Q

00757 R 742331 A TCA

00758 R 221233 R TAD TEMPS /EXIT, WITH THE EVENT VARIABLE SET TO

00759 R 341322 R JMP* CXDAC /THIS PULSE COUNT.

00760 R 621235 R /CXDAC /E.V. = -105.

00762 R 777673 A RST7 /NO CHANGES TO BE MADE. EXIT WITH

00763 R 621235 R JHP* CXDAC /E.V. = -105.
THIS CODE EXTENDS THE UNFIX CAL FUNCTION DIRECTIVE TO INCLUDE
A CHECK TO SEE IF A TASK FIXED IN CORE HAS ANY ENTRIES IN THE
AUTOMATIC INPUT TABLE, AND IF SO, THEY ARE REMOVED, FLAG BIT 4
OF THE SYSTEM TASK LIST (STL) NODE IS ON IF SUCH ENTRIES EXIST.

NOTE THAT THE REMOVAL OF ENTRIES FROM THE AUTOMATIC INPUT REQUEST
TABLE IS PERFORMED PRIOR TO THE CHECK FOR THE TASK BEING ACTIVE,
THUS A TASK MAY MINT AUTOMATIC DATA TRANSFER TO ITS BUFFERS BY
REQUESTING THE UNFIX DIRECTIVE FOR ITSELF, THOUGH THE TASK WILL
STILL REMAIN IN CORE UNTIL IT EXITS.

IN ADDITION TO THE REGISTERS USED BY THE REST OF THE CAL DIRECTIVE,
THIS SECTION OF CODE USES, WITHOUT SAVING OR RESTORING, X13 AND X12.
IF THE TASK ACTIVE?

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.

ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.
PATCH RESIDENT ROUTINE TO FILL USER INPUT TABLE.

TITLE RESIDENT ROUTINE TO FILL USER INPUT TABLE.

LOC START*1140

UPON INTERRUPT (LINE NUMBER 30, ADDRESS 00370) FROM THE SECONDARY PROCESSOR (CLOCK SECTION OF ITS MAIN PROGRAM), CONTROL IS TRANSFERRED HERE, DIRECTLY, AT API LEVEL 0. THIS ROUTINE CHECKS TO SEE IF AN INPUT IS TO BE STORED IN A USER BUFFER, AND IF SO, TRANSFERS THE VALUE TO IT.

NOTE THE TIMING PROBLEM EXPLAINED IN THE CAL DIRECTIVE 33 CODE.

THIS ROUTINE MUST SAVE ANY REGISTERS USED, AS THE STANDARD FRONT-END INTERRUPT HANDLER SAVE ROUTINES ARE NOT USED HERE.

EJECT
RESIDENT ROUTINE TO FILL USER INPUT TABLE.

ENTRY POINT FROM API,

ENTRY POINT FOR CHECKING A TABLE AND TRANSFERS TO THE USER BUFFERS.

ENTRY POINT FROM API LEVEL 0, AND RESTORE.

ENTRY POINT FOR INSURING INDEX MODE OPERATION.

ENTRY POINT FOR MAKING TRANSFERS TO THE USER BUFFERS.

ENTRY POINT FOR GETTING TABLE ADDRESS.

ENTRY POINT FOR GETTING ENTRY FROM THE TABLE.

ENTRY POINT FOR YES, RETURN TO MAIN ROUTINE.

ENTRY POINT FOR YES, TRANSFER VALUE TO USER BUFFER, WITH SHARED MEMORY ADDRESS FOR THAT DEVICE.

ENTRY POINT FOR GETTING THE VALUE...

ENTRY POINT FOR CHECK IF THE END OF THE BUFFER IS AT HAND.

ENTRY POINT FOR IS THE ADDRESS POINTER EQUAL TO THE LAST

ENTRY POINT FOR YES, THE POINTER MUST BE RESET.

ENTRY POINT FOR NO, INCREASE THE POINTER FOR NEXT TIME.

ENTRY POINT FOR INITIALIZE FOR THE NEXT TABLE ENTRY.

ENTRY POINT FOR LOOP BACK TO HANDLE THE NEXT ENTRY.

ENTRY POINT FOR STORE IT AS THE BUFFER POINTER ADDRESS.

ENTRY POINT FOR CONTINUE.
.PATCH STORAGE AND CONSTANTS.

/TITLE STORAGE AND CONSTANTS.

/CONSTANTS BASED ON DEFINED SYMBOLS:

01222 R 070000 A TABL ADCLK=NDAC=1 /ADDRESS ONE PAST TOP MOST ENTRY, INPUT TABLE.
01223 R 777676 A MLGT =ND1-NDAC=1 /NEGATIVE OF NUMBER OF TABLE POSITIONS.
01224 R 000001 A XADJ =10000+BANK /INDEX BASE ADJUSTMENT FOR THIS MEMORY BANK.
01225 R 070000 A LND =ND1-NDAC=1 /NUMBER OF DIGITAL OUTPUT GROUPS =1.
01226 R 000002 A LDAC NDAC /NUMBER OF DIGITAL TO ANALOG CONVERTERS.
01227 R 070000 A MLGTH =ND0-NDAC=1 /NEGATIVE OF OUTPUT TABLE LENGTH.
01228 R 000327 R PROTT PROT-NDAC /TOP OF PROTECT TABLE.
01229 R 002310 A LDAC NDAC /NUMBER OF DIGITAL TO ANALOG CONVERTERS.
01230 R 777766 A MLGT =ND0-NDAC*1 /NEGATIVE OF OUTPUT TABLE LENGTH.
01231 R 000327 R PROTT PROT-NDAC /TOP OF PROTECT TABLE.
01232 R 067447 A TBLX DTCX ADJ-10330 /INDEX BASE ADJUSTMENT FOR THIS MEMORY BANK.
01233 R 033776 A CP9RX /ROUTINE TO CHECK CP9 LENGTH.
01234 R 001303 R LTA3L /ADDRESS OF TEMPORARY STORAGE TABLE (CAL 34),
01235 R 000033 R TALI /TOP OF ADC REQUEST TABLE STARTING ADDRESS,
01236 R 000103 R TABL /ADDRESS OF ADC REQUEST TABLE STARTING ADDRESS,
01237 R 000200 R TALI /TOP OF ADC REQUEST TABLE STARTING ADDRESS,
01238 R 773000 R XTAL /ADDRESS OF ADC REQUEST TABLE STARTING ADDRESS.

/ADDRESSES REFERENCED IN THE MONITOR:

01239 R 067776 A CPBRX /ROUTINE TO CHECK CPB LENGTH.
01240 R 001303 R LTA3L /ADDRESS OF TEMPORARY STORAGE TABLE (CAL 34),
01241 R 000033 R TALI /TOP OF ADC REQUEST TABLE STARTING ADDRESS,
01242 R 000103 R TABL /ADDRESS OF ADC REQUEST TABLE STARTING ADDRESS,
01243 R 000200 R TALI /TOP OF ADC REQUEST TABLE STARTING ADDRESS,
01244 R 033200 R PROTT PROT-NDAC /TOP OF PROTECT TABLE.
01245 R 030723 A CXDAC /SET E.V. TO AC VALUE AND EXIT.
01246 R 033717 A CXSUC /SET E.V. = ♦! AND EXIT.
01247 R 030652 A CX33 /SET E.V. = -30 AND EXIT.
01248 R 030656 A CX131 /SET E.V. = -101 AND EXIT.
01249 R 030563 A CX122 /SET E.V. = -102 AND EXIT.
01250 R 030670 A CX203 /SET E.V. = -203 AND EXIT.
01251 R 000700 A CX257 /SET E.V. = -207 AND EXIT.
01252 R 002473 A UN /ONE RETURN POINT IN CAL 16 MAIN PART.
01253 R 003435 A CT3IAS /RELOCATION FACTOR FOR USER MODE TASK.
01254 R 000145 A X11 /AUTO-INCREMENT REGISTER 11,
01255 R 000033 R TALI /ADDRESS OF ADC REQUEST TABLE STARTING ADDRESS,
01256 R 000103 R TABL /ADDRESS OF ADC REQUEST TABLE STARTING ADDRESS,
01257 R 000200 R TALI /ADDRESS OF ADC REQUEST TABLE STARTING ADDRESS,
01258 R 030200 /CAL ENTRY POINT.

/ADDRESSES REFERENCED IN SHARED MEMORY:

01259 R 067677 A ADCLK /FAST CLOCK COUNTER ADDRESS.

/ADDRESSES REFERENCED WITHIN THESE FUNCTIONS:

01260 R 223123 R LALI /ADDRESS OF TEMPORARY STORAGE TABLE (CAL 34),
01261 R 022273 R TALI /ADDRESS OF ADC REQUEST TABLE STARTING ADDRESS.
01262 R 030100 R TABL /ADDRESS OF ADC REQUEST TABLE StartNG ADDRESS.
01263 R 000200 R TALI /ADDRESS OF ADC REQUEST TABLE STARTING ADDRESS.
01264 R 772509 R XTAL /ADDRESS OF ADC REQUEST TABLE STARTING ADDRESS.

/OTHER CONSTANTS:

01265 R 145901 A NA /NOT ASSIGNED CODE IN THE PROTECT TABLE.
01266 R 223123 R LALI /ADDRESS OF TEMPORARY STORAGE TABLE (CAL 34),
01267 R 022273 R TALI /ADDRESS OF ADC REQUEST TABLE STARTING ADDRESS.
01268 R 000200 R TALI /ADDRESS OF ADC REQUEST TABLE STARTING ADDRESS.
01269 R 771700 R MASK1 /MASK FOR DAC NUMBER.
01270 R 777229 R MASK2 /MASK FOR TABLE BASE ADDRESS (CAL 34).
01271 R 440223 R MASK3 /MASK FOR TASK ACTIVE AND TASK FIXED FLAGS.
01272 R 577777 R MASK4 /MASK FOR TASK ACTIVE AND TASK FIXED FLAGS.
01273 R 000002 A P33K /FOR CHECKING TABLE ADDRESS (CAL 34).
01274 R 003000 A P33K /FOR CHECKING TABLE ADDRESS (CAL 34).
PATCH STORAGE AND CONSTANTS,

/ STORAGE CELLS.

/DEVICE NUMBER (CAL 32).

/TEMPORARY STORAGE (CAL 32).

/TEMPORARY STORAGE (CAL 32).

/TEMPORARY STORAGE (CAL 33).

/TEMPORARY STORAGE (CAL 16).

/TEMPORARY STORAGE (CAL 33).

/TEMPORARY STORAGE (CAL 34).

/TEMPORARY STORAGE (CAL 34).

/TEMPORARY STORAGE (CAL 33).

/TEMPORARY STORAGE (CAL 34).

/ADDRESS POINTER (CAL 31).

/ADDRESS POINTER (CAL 33).

/ADDRESS POINTER (INFIL).

/ADDRESS POINTER (INFIL).

/ADDRESS POINTER (INFIL).

/ADDRESS POINTER 1 (CAL 16).

/ADDRESS POINTER 2 (CAL 16).

/ADDRESS POINTER 2 (CAL 16).

/ADDRESS POINTER 2 (CAL 16).

/ADDRESS POINTER 2 (CAL 16).

/TABLE COUNTER (CAL 31).

/TABLE COUNTER (CAL 34).

/INDEX SAVE CELL (CAL 16).

/INDEX SAVE CELL (CAL 16).

/INDEX SAVE CELL (INFIL).

/INDEX SAVE CELL (INFIL).

/PARTITION BASE (COMPLEMENT) (CAL 16).

/PARTITION BASE (COMPLEMENT) (CAL 16).

/PARTITION TOP + 1 (COMPLEMENT) (CAL 16).

/PARTITION TOP + 1 (COMPLEMENT) (CAL 16).

/COUNTER FOR SERVICING THE 10K TABLE (INFIL).

/COUNTER FOR SERVICING THE 10K TABLE (INFIL).

/COUNTER FOR SERVICING THE 10K TABLE (INFIL).

/COUNTER FOR SERVICING THE 10K TABLE (INFIL).
MOD18F

SECONDARY PROCESSOR HARDWARE INFORMATION.

.TITLE SECONDARY PROCESSOR HARDWARE INFORMATION.

/OCT OCTAL CONSTANTS SPOKEN HERE.

/THERE ARE NO PROGRAM INTERRUPT DEVICES ATTACHED TO THE SECONDARY
/PROCESSOR.

/THE ONLY DATA CHANNEL DEVICE IS THE ANALOG TO DIGITAL CONVERTER,
/WHICH REQUIRES LOCATION 00022 FOR THE WORD COUNT AND LOCATION
/03323 FOR THE BUFFER ADDRESS. THE WORD COUNT IS INCREMENTED,
/BUT NOT USED BY THE CONVERTER, WHICH TRANSFERS 100 (64 DECIMAL)
/DATA WORDS PER CALL.

/THE FOLLOWING API BREAK LOCATIONS ARE USED, IN ADDITION TO THE
/FOLLOWING FOUR SOFTWARE BREAKS:
/03045 KEYBOARD ENTRY READY (LEVEL 3).
/02246 SERVICE REQUEST FROM PRIMARY PROCESSOR (LEVEL 0).
/02251 CLOCK OVERFLOW (LEVEL 3).

/THE FOLLOWING IOT INSTRUCTIONS ARE RECOGNIZED, IN ADDITION
/TO THOSE SUPPLIED WITH THE CENTRAL PROCESSOR AND API OPTION:
/706624 CHARACTER 'SCOPE.
/706324 KEYBOARD INPUT.
/703324 DIGITAL OUTPUT GROUP 0 ("SCOPE INDICATOR LIGHTS"),
/703352 DIGITAL INPUT GROUP -1 ("SCOPE SENSE SWITCHES"),
/703434 ANALOG TO DIGITAL CONVERTER REGISTER,
/706371 CAUSE INTERRUPT IN PRIMARY PROCESSOR AT LOCATION
/02272, API LEVEL 0.
/706224 GRAPHICS (SCOPE CONTROL).
/706224 DIGITAL OUTPUT GROUP -2 (HYBRID CONTROL),
/706444 DIGITAL OUTPUT GROUP -1 (BETA GAUGE),
/707104 DIGITAL TO ANALOG CONVERTERS.
SECON DARY PROCES SOR PROGRAM MI NG INFOR MATION.

THE PROGRAM IS WRITTEN TO BE LOADED INTO SHARED MEMORY TO BE
EXECUTED BY THE SECONDARY PROCESSOR BY HAVING THE PRIMARY
PROCESSOR USE RSX-PLUS TO INSTALL IT, AS IF IT WERE A TASK.
THE PARTITION NAME IS DEFINED TO START AT THE BOTTOM OF SHARED
MEMORY, CURRENTLY LOCATION 62622 OCTAL, IN BUILDING THE TASK,
THE "EXX" OPTION MUST BE SPECIFIED. THE PARTITION NAME IS
CURRENTLY "MOD10", REQUIRED BY
TASK ".SOP", WHICH ASSURES THAT THIS TASK IS FIXED IN CORE.

PROGRAMMING FOR THE SECONDARY PROCESSOR IS CONFINED TO THE LOWER
2048 (TOTAL) MEMORY CELLS, WITH TABLes FOR SHARED USE BEING LOCATED
IN CELLS 07432 - 27777 (CURRENTLY 67433 - 67777 AS ADDRESSED BY
THE PRIMARY PROCESSOR), ALL OTHER MEMORY ADDRESSABLE BY THE
SECONDARY PROCESSOR IS RESERVED FOR USE BY THE PRIMARY PROCESSOR.
IF MORE PROGRAM AREA IS REQUIRED, THE RSX-PLUS MONITOR FOR THE
PRIMARY PROCESSOR MUST BE RE-CONFIGURED ("COLD START") TO ALLOW
MORE ROOM IN CORE PARTITION "MOD10", IF THE SHARED TABLES
REQUIRE MORE SPACE, PARTITION "SHARE" MUST BE EXPANDED.

ALL PROGRAMMING FOR THE SECONDARY PROCESSOR IS CONTAINED IN
ONE ASSEMBLY (I. E, ONE PROGRAM). ADDITIONAL CODE IS ADDED
TO THE END OF THE PRESENT CODE AS REQUIRED, WITH NECESSARY
CHANGES MADE TO ALLOW LINKAGE, AND THE WHOLE SOURCE ASSEMBLED.
ADDITIONAL CODE MUST FOLLOW THESE CONVENTIONS:
1. NO BANK MODE PROGRAMMING IS PERMITTED, AS THE INTERRUPT
HANDLERS REQUIRE THE INDEX REGISTER, AND DO NOT
DISABLE BANK MODE ADDRESSING.
2. NO PROGRAM SEGMENT IS TO RUN ABOVE API LEVEL 4 (EXCEPT
THE CLOCK HANDLER), THUS IN RESPONSE TO A
HARDWARE LEVEL INTERRUPT, A SOFTWARE LEVEL BREAK
IS TO BE REQUESTED, AND A DEBREATHE AND RESTORE
INSTRUCTION ISSUED IMMEDIATELY.
3. AS ALL CODE IS EXECUTED AT THE SAME TIME, THE PRO-
GRAMMER MUST AVOID LABELS PREVIOUSLY USED.
4. IF CHANGES ARE OF A SIGNIFICANT NATURE, THE LAST
LETTER IN THE NAME OF THIS PROGRAM (ITS SOURCE
FILE NAME) SHOULD BE CHANGED TO THE NEXT LETTER
OF THE ALPHABET.
5. THE CAL INSTRUCTION IS CONSIDERED UNNECESSARY, AND TO
FACILITATE ERROR CHECKS, IT IS ILLEGAL.
6. PROGRAMMING WHICH IS TO RESPOND TO THE MAIN PROCESSOR
SHOULD BE DISPATCHED THROUGH THE MAIN PROCESSOR
INTERRUPT SERVICE ROUTINE SECTION OF THE SECONDARY
PROCESSOR PROGRAM. PROGRAMMING TO OPERATE AT API
LEVEL 7 SHOULD USE THE LEVEL 7 DISPATCHER, WHICH
FOLLOWS THE MAIN PROCESSOR INTERRUPT SERVICE
ROUTES. IF ANY ADDITIONAL PROGRAMMING IS TO
RUN AT MAINLINE (WITH THE GRAPHICS SCOPE AND
WAIT LOOP), A DISPATCH TABLE MUST BE CONSTRUCTED.
7. THE TABLES FOR HARDWARE INFORMATION GIVEN ABOVE, AND
FOR SOFTWARE INFORMATION GIVEN BELOW, MUST BE
THE FOLLOWING SHARED MEMORY LOCATIONS (AS RESERVED BY THE PRIMARY PROCESSOR IN PARTITION "SHARE") ARE USED BY THIS SECONDARY PROCESSOR PROGRAM:

- 07422-07437 DIGITAL OUTPUT BUFFER (REAL-TIME).
- 07442-07457 DIGITAL TO ANALOG CONVERTER BUFFER.
- 07472-07477 PLOTTING 'SCOPE BUFFER.
- 07525-07545 FLAG TABLE FOR MAIN PROCESSOR REQUESTS.
- 07527-07657 KEYBOARD-'SCOPE BUFFER.
- 07662-07677 DIGITAL INPUT BUFFER (REAL-TIME).
- 07677 CLOCK COUNTER (1 KHz).
- 07722-07777 ANALOG TO DIGITAL CONVERTER BUFFER.

NOTE THAT AT PRESENT NOT ALL RESERVED CELLS FOR THE DIGITAL INPUTS AND OUTPUTS, AND THE DIGITAL TO ANALOG CONVERTERS, ARE USED. THESE ARE TO BE USED WHEN MORE OF THOSE UNITS ARE ADDED TO THE SYSTEM.


THE STARTING ADDRESS FOR THIS PROGRAM IS 02200.
TITLE INITIALIZATION SECTION.

INITIALIZATION SECTION, EXECUTED ONLY WHEN THE PROGRAM IS FIRST LOADED OR DURING A SYSTEM RESTART.


LOC 22021* START /CAL ADDRESS TRAP.

LOC 22031* START /CAL ADDRESS TRAP.

LOC 22032* START /SOFTWARE INTERRUPT.

LOC 22043* START /SOFTWARE INTERRUPT.

LOC 22046* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22047* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22050* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22055* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22060* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22065* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22070* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22075* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22080* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22085* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22090* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22095* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22100* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22105* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22110* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22115* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22120* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22125* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22130* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22135* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22140* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22145* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22150* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22155* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22160* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22165* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22170* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22175* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22180* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22185* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22190* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22195* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22200* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22205* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22210* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22215* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22220* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22225* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22230* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22235* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22240* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22245* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22250* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22255* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22260* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22265* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22270* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22275* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22280* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22285* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22290* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22295* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22300* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22305* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22310* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22315* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22320* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22325* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22330* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22335* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22340* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22345* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22350* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22355* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22360* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22365* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22370* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22375* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22380* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22385* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22390* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22395* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22400* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22405* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22410* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22415* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22420* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22425* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22430* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22435* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22440* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22445* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22450* API INTERRUPT FROM THE PRIMARY PROCESSOR.

LOC 22455* API INTERRUPT FROM THE CONTROL UNIT.

LOC 22460* API INTERRUPT FROM THE ADVANCED INTRP.

LOC 22465* API INTERRUPT FROM THE PRIMARY PROCESSOR.
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>153</td>
<td>02214 R 202220 A</td>
<td>CALH LAC 00020 /IF CAL, GET CAL ADDRESS.</td>
</tr>
<tr>
<td>154</td>
<td>02215 R 742240 A</td>
<td>PI HLT /STOP FOR OPERATOR EXAMINATION.</td>
</tr>
<tr>
<td>155</td>
<td>02216 R 74C220 A</td>
<td>DDON IAC /DO NOTHING LOOP - BLINK LINK</td>
</tr>
<tr>
<td>156</td>
<td>02217 R 602216 R</td>
<td>JMP -.1 /AND COUNT AC LIGHTS.</td>
</tr>
<tr>
<td>157</td>
<td>/</td>
<td>/CONSTANTS.</td>
</tr>
<tr>
<td>158</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>159</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>160</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>161</td>
<td>02222 R 102224 R</td>
<td>CKAD1 JMS CLK1 /INITIAL CLOCK OVERFLOW BREAK INSTRUCTION.</td>
</tr>
<tr>
<td>162</td>
<td>02221 R 007514 A</td>
<td>DSPCM 07504 /ADDRESS OF THE DISPATCH TABLE. -1.</td>
</tr>
<tr>
<td>163</td>
<td>02221 R 007514 A</td>
<td>EMSP 07514 /TOP ADDRESS OF THE DISPATCH TABLE.</td>
</tr>
<tr>
<td>164</td>
<td>02223 R 402220 A</td>
<td>SGNB 400220 /SIGN BIT.</td>
</tr>
</tbody>
</table>
CLOCK INTERRUPT SERVICE ROUTINE.

.TITLE CLOCK INTERRUPT SERVICE ROUTINE.

// THIS SECTION HANDLES ALL FUNCTIONS BASED ON REGULAR
// CLOCK INTERVALS:
// - RESETTING THE CLOCK OVERFLOW REGISTER,
// - INITIALIZATION OF THE ANALOG-TO-DIGITAL CONVERTER,
// - LOADING ALL DIGITAL-TO-ANALOG CONVERTERS,
// - READING ALL DIGITAL INPUT GROUPS,
// - LOADING ALL DIGITAL OUTPUT GROUPS,
// - UPDATING THE CLOCK COUNTER,

// THIS SECTION IS SUPPORTED BY P. N. BARTRAM,
// LATEST REVISION: 2 FEBRUARY 1973,

// PRESENTLY, ONE DIGITAL INPUT GROUP,
// PRESENTLY, THREE DIGITAL OUTPUT GROUPS.

// CLOCK INTERRUPT HANDLER, FIRST SECTION.

// DISABLE API WHILE SERVICING THE CLOCK.
// TURN OFF THE CLOCK TO RESET.
// ON THE NEXT CLOCK INTERRUPT (0.7 MILLISECONDS
// FROM NOW), GO TO THE OTHER HANDLER SECTION,
// RESET THE CLOCK REGISTER TO OVERFLOW IN 0.7
// FROM NOW), GO TO THE OTHER HANDLER SECTION,
// ENABLE THE CLOCK INTERRUPT AGAIN.

// SERVICE THE ANALOG-TO-DIGITAL CONVERTER.

// GET BUFFER ADDRESS - 1.
// STORE IN BUFFER ADDR. REG.
// INITIALIZE BUFFER WORD COUNT.
// SET THE FRAME SCAN BIT IN THE
// A/D REGISTER.
// START THE A/D START PULSE,
// END THE START PULSE,

// SERVICE THE DIGITAL OUTPUT GROUPS.

// SAVE THE INDEX REGISTER,
// SAVE THE LIMIT REGISTER,
// CLEAR THE LIMIT REGISTER,
// INITIALIZE THE DIGITAL OUTPUT POINTER
// RELATIVE TO THE DAC TABLE, IN THE XR.

// GET THE DIGITAL OUTPUT WORD.
MOD10F

CLOCK INTERRUPT SERVICE ROUTINE.

02257 R 42342 R XCT 10D00+NOO,X /SET THE DIGITAL OUTPUTS.
02260 R 725201 A AXS +1 /IF THERE ARE MORE DIGITAL OUTPUT GROUPS.
02261 R 602256 R JMP -3 /LOOP BACK.

/ SERVICE THE DIGITAL-TO-ANALOG CONVERTERS.

02262 R 222347 R LAC EIGHT /PUT THE NUMBER OF D/A CONVERTERS
02264 R 217440 A PAL / IN THE LIMIT REGISTER.
02265 R 727124 A IOT 7104 /CORE TABLE, AND OUTPUT IT.
02266 R 725221 A AXS +1 /INDEX INITIALLY ZERO FROM PREVIOUS SECTION.
02267 R 622264 R JMP +3 /LOOP THROUGH ALL CHANNELS.

/ READ THE DIGITAL INPUT GROUPS.

02270 R 736220 A CLR /CLEAR THE LIMIT REGISTER.
02271 R 777777 A LAW CND1 /SET THE INDEX REGISTER FOR REFERENCING THE
02272 R 721220 A PAX /CLOCK AND DIGITAL INPUT GROUPS.
02273 R 412344 R XCT 10D1+ND1,X /EXECUTE THE IOT FOR DIGITAL INPUTS.
02274 R 744013 A RCL /MAKE SURE BIT 17 IS "ALWAYS ON" BY SHIFTING
02275 R 742230 A IACIN IAC /ONE PLACE LEFT, AND SETTING BIT 17.
02276 R 657677 A DAC TABLA,X /STORE THE VALUES INPUT.
02277 R 725221 A AXS +1 /IF THERE ARE MORE TO READ, LOOP BACK.
02278 R 607275 R JMP -3 /LOOP THROUGH ALL CHANNELS.
02279 R 705521 A IOT 5521 /ENABLE THE API AGAIN.
02280 R 222354 R LAC DSAVL /RESTORE THE LIMIT REGISTER.
02283 R 722220 A PAL DSAVB /RESTORE THE INDEX REGISTER.
02284 R 2C2353 R LAC PAX /CLEAR THE LIMIT REGISTER.
02285 R 725223 A FIL /READ THE DIGITAL INPUT GROUPS.
02286 R 2C2357 R LAC /INCREMENT THE CLOCK COUNTER BY TWO TO ASSURE
02287 R 727677 A LAC TABLA /BIT 17 IS ALWAYS ON.
02288 R 477677 A DAC X /MAKE SURE BIT 17 IS "ALWAYS ON".
02289 R 203352 R LAC BTNIN /REQUEST A PASS THROUGH THE LEVEL 7 DISPATCHER
02290 R 705524 A ISA /TO SEE IF ANYTHING IS WAITING FOR THE CLOCK.
02291 R 222355 R LAC CLISV /RESTORE THE A/C.
02292 R 703344 A DSR /REQUEST A PASS THROUGH THE LEVEL 7 DISPATCHER.
02293 R 622224 R JMP* CLK1 /EXIT.
02295 R 742343 A CLK2 XX /CLOCK INTERRUPT HANDLER, SECOND SECTION.
02296 R 705522 A IOT 5522 /DISABLE THE API WHILE SERVICING THE CLOCK.
02297 R 843355 R DAC CLISV /SAVE THE A/C.
02298 R 720224 A IOT 0204 /TURN OFF THE CLOCK TO RESET IT.
02299 R 777775 A LAN -3 /RESET THE CLOCK TO INTERRUPT IN 0.3
02300 R 842027 A DAC 02007 /MILLISECONDS.
02301 R 2222220 R LAC CKAD1 /ON THE NEXT CLOCK INTERRUPT, GO TO THE
02302 R 543351 A DAC 02007 /OTHER SECTION OF THE HANDLER.
02303 R 722244 A IOT 0244 /ENABLE THE CLOCK INTERRUPT AGAIN.
02304 R 722222 A LAC 02002 /SEE IF THE I/O PROCESSOR HAS FILLED THE
02305 R 742370 A GEF /THE A/D BUFFER. (IT HAS TO BE IN THE TIME.)
02306 R 623327 R JMP -2 /IF NOT, HANG THE SYSTEM UNTIL IT WAS.
02307 R 723721 A IOT 3701 /INTERRUPT THE MAIN PROCESSOR.
CLOCK INTERRUPT SERVICE ROUTINE.

00333 R 705521 A IOT 5521 /ENABLE THE API AGAIN.
00334 R 205555 R LAC CLISV /RESTORE THE AC.
00335 R 703344 A DBR /DEBREAK AND RESTORE.
00336 R 620316 R JMP* CLK2

/* CONSTANT TABLE AND STORAGE CELLS,*/
/* NOTE: FOR THE IOT TABLES, THE MOST NEGATIVE GROUP NUMBER COMES
/* FIRST (GIVING IT THE LOWEST ADDRESS IN THE TABLE), I. E. ORDER
/* GROUPS ..., -3, -2, -1, 0. (THERE IS NO GROUP 0 FOR THE INPUTS.)
/* /
/* IOT TABLE FOR THE DIGITAL OUTPUT GROUPS,
/* /
00337 R 706024 A IOTD0 IOT 6024 /DIGITAL OUTPUT GROUP -2, HYBRID CONTROL.
00340 R 706044 A IOT 6044 /DIGITAL OUTPUT GROUP -1, BETA GAUGE.
00341 R 702844 A IOT 3844 /DIGITAL OUTPUT GROUP 0, LIGHTS.
00342 R 740040 A MLT /NO MORE DIGITAL OUTPUT GROUPS.

/* IOT TABLE FOR THE DIGITAL INPUT GROUPS,
/* /
00343 R 703052 A IOTDI IOT 3252 /DIGITAL INPUT GROUP -1.
00344 R 740040 A MLT /NO MORE DIGITAL INPUT GROUPS.

/*CKAD2 JM5 CLK2/SECOND SECTION CLOCK INTERRUPT HANDLER,*/

00345 R 120316 R CKAD2 JM5 CLK2

/*TABL 67677 /A/D BUFFER, ADDRESS ONE BELOW START.*/
00346 R 227677 A TABL 67677 /FIRST ADDRESS OF THE DAC TABLE.
00347 R 227677 A TABL=67677 /CLOCK COUNTER ADDRESS - BETWEEN DIG. IN & ADC.

00347 R 207212 A EIGHT 22200 /EIGHT, OR OCTAL TEN (CONSTANT).
00348 R 222200 A FSBIT 222200 /FOR A/D REGISTER.
00349 R 622200 A FSBIT 622200 /FOR A/D REGISTER.
00350 R 422420 A BTNIN 42400 /AC FOR ISA FOR LEVEL 7.
00351 R 222000 A CSABL 0 /SAVE CELL FOR THE INDEX REG.
00352 R 222000 A CSAYL 0 /SAVE CELL FOR THE LIMIT REG.
00353 R 228000 A CLISV 0 /SAVE CELL FOR THE AC.
MOD10F MAIN PROCESSOR INTERRUPT SERVICE ROUTINE.

TITLE MAIN PROCESSOR INTERRUPT SERVICE ROUTINE,

SECTION INTERRUPT HANDLER AND DISPATCHER FOR INTERRUPTS CAUSED

BY THE MAIN PROCESSOR.

THIS SECTION SUPPORTED BY P. N. BARTRAM.


INTERRUPT HANDLER AND DISPATCHER FOR INTERRUPTS CAUSED

BY THE MAIN PROCESSOR.


ENTRY POINT FROM API BREAK 46.

SERVICE THE INTERRUPT FROM THE

MAIN PROCESSOR AT SOFTWARE

LEVEL 4.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

SAVE XI1 AUTO-INCREMENT REG.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPT. SAVE AC.

ENTRY FOR SERVICING THE MAIN

PROCESSOR INTERRUPTIONS, AS INITIATED

BELOW.

REMAINING ROUTINES ARE CALLED THROUGH

SOFTWARE API INTERRUPTS, AS INITIATED

BELOW.
MOD10F MAIN PROCESSOR INTERRUPT SERVICE ROUTINE.

/ 2044 R 147516 A SCP10 D2M 07516 /*ZERO FLAG FOR CHARACTER
358 2042 R 204450 R LAC SFTWR /* 'SCOPE AND GO TO HANDLER
359 2043 R 042443 R DAC LEVLT+*/ /* AT PRIORITY 7 (SUB 4),
360 2044 R 204352 R LAC STIRN /* CAUSE A SOFTWARE BREAK AT
361 2045 R 765624 A ISA 07624 /* LEVEL 7, AND EXIT.
362 2046 R 603371 R JMP 147513 /*KEY10 D2M 07513 /*ZERO FLAG FOR KEYBOARD
363 2047 R 147514 A SCP13+ /*LEVEL 7 (SUB 4) PRIORITY.
364 2048 R 042451 R DAC LEVL7 */ /* PRIORITY LEVEL.
365 2049 R 602371 R JMP MORE /* DISPATCHER FOR LEVEL 7 SOFTWARE INTERRUPTS.
366 2050 R 147512 A HEI8L DZM 07512 /*ZERO FLAG FOR GRAPHICS
367 2051 R 042452 R LAC MALIN /* 'SCOPE AND GO TO
368 2052 R 042216 R DAC PI+1 /*HANDLER AT MAINLINE
369 2053 R 625371 R JMP MORE /* PRIORITY LEVEL.
370 2054 R 047457 R OAC AC7 /*SAVE THE AC REGISTER,
371 2055 R 742070 A NOP /*SUBLEVEL 2.
372 2056 R 742016 A NOP /*SUBLEVEL 3.
373 2057 R 742020 A NOP /*SUBLEVEL 4.
374 2058 R 742032 A NOP /*SUBLEVEL 5.
375 2059 R 742036 A NOP /*SUBLEVEL 6.
376 2060 R 742030 A NOP /*SUBLEVEL 7.
377 2061 R 742024 A NOP /*SUBLEVEL 8.
378 2062 R 742018 A NOP /*SUBLEVEL 9.
379 2063 R 742012 A NOP /*SUBLEVEL 10.
380 2064 R 742008 A NOP /*SUBLEVEL 11.
381 2065 R 742004 A NOP /*SUBLEVEL 12.
382 2066 R 742000 A NOP /*SUBLEVEL 13.
383 2067 R 220457 R LAC AC7 /*RESTORE THE AC REGISTER.
384 2068 R 101354 R MALIN /* 'SCOPE AND GO TO
385 2069 R 403402 R XCTIN XCTB /*BASE OF DISPATCH XCT INS.
386 2070 R 0436 R 742048 A LEVL7 XX /*ENTRY POINT.
387 2071 R 0437 R 042457 R DAC AC7 /*SAVE THE AC REGISTER.
388 2072 R 0440 R 742232 A NOP /*SUBLEVEL 1.
389 2073 R 0441 R 742230 A NOP /*SUBLEVEL 2.
390 2074 R 0442 R 742232 A NOP /*SUBLEVEL 3.
391 2075 R 0443 R 742230 A NOP /*SUBLEVEL 4.
392 2076 R 0444 R 742232 A NOP /*SUBLEVEL 5.
393 2077 R 0445 R 222457 R LAC AC7 /*RESTORE THE AC REGISTER.
394 2078 R 0446 R 703344 A DBR /*EXIT ON RETURN.
395 2079 R 0447 R 622436 R JMP* LEVL7 /*CONSTANT TABLE AND STORAGE CELLS.
396 2080 R 0453 R 101311 R SFTWR /*SOFTWARE LEVEL 7 ADDRESS TRANSFER INSTRUCTIONS.
397 2081 R 0451 R 102534 R SFTWR /*HANDLERS INSERT ADDITIONAL TRANSFER INSTRUCTIONS.
398 2082 R 0452 R 101354 R MALIN /*FOR GRAPHICS 'SCOPE.
399 2083 R 0453 R 403402 R XCTIN XCT /*BASE OF DISPATCH XCT INS.
MAIN PROCESSOR INTERRUPT SERVICE ROUTINE.

<table>
<thead>
<tr>
<th>Line</th>
<th>Address</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>420</td>
<td>00454</td>
<td>R 404200 A</td>
<td>BITSX 404200 /AC FOR ISA FOR LEVEL 4.</td>
</tr>
<tr>
<td>421</td>
<td>00455</td>
<td>R 000000 A</td>
<td>SAVE</td>
</tr>
<tr>
<td>422</td>
<td>00456</td>
<td>R 000000 A</td>
<td>SAVAC</td>
</tr>
<tr>
<td>423</td>
<td>00457</td>
<td>R 000000 A</td>
<td>AC7</td>
</tr>
<tr>
<td>424</td>
<td>00460</td>
<td>R 000000 A</td>
<td>SAV11</td>
</tr>
</tbody>
</table>
**KEYBOARD-'SCOPE HANDLER:**

1. **TITLE:** KEYBOARD-'SCOPE HANDLER.
2. **KEYBOARD-'SCOPE ROUTINES - SECONDARY PROCESSOR SECTION.**
3. **SUPPORTED BY P. N. BARTHA.**
4. **LATEST REVISION: 4 JUNE 1973.**

**FOLLOWING IS THE SHARED CORE USED BY THIS HANDLER:**

- Trigger Cause Flag
- I/O Handler Busy Flag
- Mode Code
- Trigger
- **(BASE 6) BUFFER (HEADER)**
- Guard Word for Output Buffer
- Request Input from Keyboard
- Request Output to 'Scope

**KEYBOARD INTERRUPT HANDLER:**

- This section checks for control C, control T, control L characters at current priority.
- And if not found, and if input is expected, debugs to software level 7 for input.

**NOTE:** Under certain conditions the control C or control T may be lost before accepted by the primary processor.

**PROCEDURE:**

- **ENTRY POINT,** DAC KBSAV / SAVE THE AC
- **READ KEYBOARD,** IOT 3012
- **SAVE, DROP TO SOFTWARE LEVEL 5,** DAC TEMP2
- **TO FINISH THIS PART OF THE HANDLER,** LAC BIT2V
- **SOFTWARE LEVEL 5 ENTRY TO COMPLETE,** DAC LEV17*4
- **THE INITIAL PART OF THE HANDLER,** DAC TEMP2
- **GET THE CHARACTER READ FROM THE KEYBOARD,** DAC CNT1C
- **IF CONTROL T, SET UP FOR,** DAC CNT15
- **TOV CALL (MAIN PROCESSOR),** DAC CNT1C
- **IF CONTROL C, SET UP FOR,** DAC CNT1C
- **MCR CALL (MAIN PROCESSOR),** DAC CNT1C
- **CHECK IF FORM FEED,** DAC CNT1C
- **CLEAR SCREEN - CONTROL L,** DAC CNT1C
- **IF NONE OF THESE, CHECK IF,** DAC CNT1C
- **INPUT EXPECTED,** DAC CNT1C
- **IF NOT, EXIT,** DAC CNT1C
- **SET UP TO,** DAC CNT1C
- **HANDLE AT LEVEL 7 (SUB 3),** DAC CNT1C
MOD10F  KEYBOARD-SCOPE HANDLER.

00551 R 705524 A  CNOUT    ISA     SAVE5   /RESTORE THE AC.
00552 R 201320 A  CNOUT    LAC     /RETURN TO INTERRUPTED
00553 R 723344 A  DSR      LEVL5   /PROGRAM.
00554 R 428472 R  SPC10    LAC     TGFLG   /IF THE MAIN PROCESSOR HAN'T
00555 R 207527 A  SPC10    SEA     /HANDLED THE LAST FLAG, ANOTHER
00556 R 742200 A  JMP      CNOUT   /WILL NOT BE SENT: IGNORE *T, TG,"
00557 R 507512 R  JMP      TGFLG   /MAIN PROCESSOR KEYBOARD-SCOPE HANDLER
00558 R 201312 R  JMP      TGFLG   /MAY BE TRIGGERED. GIVE THE CAUSE
00559 R 247532 A  JMP      TG     /AS WELL.
00560 R 625512 R  JMP      CNOUT   
00561 R 207530 A  CNTK     LAC     BUSY   /CLEAR SCREEN DESIRED, IS
00562 R 740020 A  CNTK     SEA     /I/O IN PROGRESS? IF SO,
00563 R 603512 R  JMP      CNOUT   /IGNORE, ELSE, 
00564 R 762214 A  LAH     14     /CLEAR SCREEN AND RESET
00565 R 723204 A  IOT     3004   /LINE COUNT.
00566 R 777761 A  LAH     -17   
00567 R 613390 R  DAC     PENTR   
00568 R 607512 R  JMP      CNOUT   

/  THE REMAINDER OF THIS HANDLER IS EXECUTED
/  AT API LEVEL 7 (LOWEST SOFTWARE LEVEL)
/  AS SUBROUTINES CALLED FROM THE LEVEL 7
/  DISPATCH.
/  WHEN REQUEST FOR INPUT IS RECEIVED,
/  INITIALIZE EVERYTHING.
00599 R 740040 A  KBTEN     XX     /ENTRY POINT.
00600 R 201334 R  LAC     N0PIN   /RESET DISPATCH TABLE
00601 R 428443 R  DAC     LEVL7+5  /FOR LEVEL 7.
00602 R 201327 R  LAC     R0RTN   /RETURN TO READ HANDLER
00603 R 243211 R  DAC     S0CTN   /IF SCOPE LINE FILLS FIRST.
00604 R 257330 A  GLAIAC   /INITIALIZE EVENT.
00605 R 241316 A  DAC     R0EV    /VARIABLE.
00606 R 213233 R  LAC     BFST2   /INITIALIZE BUFFER POINTER.
00607 R 243236 R  DAC     ROUPF   /LEAVING ROOM FOR HEADER,
00608 R 443313 R  ISE     R0FLG   /RAISE INPUT ACCEPT FLAG.
00609 R 101271 R  JMS     WA1T   /OUTPUT AN "*" TO INDICATE INPUT
00610 R 762252 A  LAH     52     /FROM THE KEYBOARD IS EXPECTED.
00611 R 702040 A  IOT     3004   /BACKSPACE TO HAVE INPUT LINE START
00612 R 101271 R  JMS     WA1T   /BY OVERWRITING THE "*" JUST
00613 R 762210 A  LAH     15     /PUT ON THE SCREEN.
00614 R 723204 A  IOT     3004   
00615 R 625534 R  JMP      KBTEN   
00616 /  SECTION TO ACCEPT CHARACTER INPUT FROM THE KEYBOARD.
00617 /  
00618 00555 R 742040 A  KEYIN   XX     /ENTRY POINT.
00619 00556 R 201334 R  LAC     NOPIN   /RESET DISPATCH TABLE
00620 00557 R 248442 R  DAC     LEVL7+4  /FOR LEVEL 7.
MOO10F KEYBOARD-'SCAPE HANDLER.

02546 R 201312 R  LAC TEMP2 /GET CHARACTER BACK.
02551 L AC ROFLG /ALLOW NO MORE UNTIL FINISHED.)
02552 / IF CARRIAGE RETURN OR ALTMODE FOUND,
02553 / THE INPUT WILL BE TERMINATED, (FOR
02554 / CARRIAGE RETURN, THE WRITE ROUTINE,
02555 / AS ENTERED THROUGH ECHO HERE, WILL
02556 / TERMINATE THE INPUT, AND RETURN TO
02557 / LOCATION ENDRD.)
02558 / AND MASK8 /CHECK IF SPECIAL
02560 L AC SWA / CHARACTER.
02565 R 741220 A  SNP SPECL / DELETE (RO) KEY.
02566 R 501307 R AND MASK5 /CHECK IF DELETE.
02567 R 601244 A  JAC LAC* / IF ALTMODE INPUT, TREAT AS A
02568 R 221326 R  LAC* ROBF / CARRIAGE BACK.
02569 R 541334 R SAD MASK5 / IF DELETE (RO) KEY.
02570 R 402644 R JAC DELET / CHECK IF DELETE.
02571 R 541343 R SAD ALT / IF ALTMODE INPUT, TREAT AS A
02572 R 541334 R LAC ENDRD / TERMINATOR, BUT NO NEW LINE.
02573 R 101133 R  ECHO JAC OUTP / ECHO CHARACTER.
02574 R 201326 R LAC ROBUF /TEST IF 80TH CHARACTER READ.
02575 R 541324 R SAD BFEND / IF 80TH, END READING.
02576 R 601550 R JAC ENDRD / IF 80TH, END READING.
02577 R 442336 R JAC DELET / OTHERWISE, INCREMENT POINTER
02578 R 443133 R RDEXT LAC KEYIN / AND WAIT FOR NEXT
02579 R 625555 R / JAC* KEYIN / CHARACTER.
02580 / DELET LAC ROBF /DELETE (RUB OUT) HANDLER.
02581 R 221326 R SAD BFS2 / IGNORE IF FIRST
02582 R 541323 R SAD BFEXT / CHARACTER.
02583 R 541323 R SAD AAC -1 / DECREMENT BUFFER
02584 R 221326 R DAC ROBF / POINTER.
02585 R 241336 R DAC RDBUF / POINT.
02586 R 742212 A  LAW 10 / ECHO BACKSPACE.
02587 R 131173 R  JMS CUTRT / WAIT FOR NEXT CHAR.
02588 R 602552 R / JAC RDEXT
02589 / SPEC LAC ROBUF /GET THE CHARACTER AGAIN.
02590 R 221326 R SAD CR / IF CARRIAGE RETURN,
02591 R 541336 R SAD ECHO / END INPUT.
02592 R 402575 R  JNP ECHO / IF TAB, HANDLE AS
02593 R 541341 R SAD TAB / REGULAR CHARACTER.
02594 R 541341 R SAD COTHU / IF LINE DELETE (CTRL U).
02595 R 22522 R 623555 R JNP ROCTU / RESET EVERYTHING.
02596 R 761337 A  LAW 137 / ALL OTHERS ECHOED
02597 R 625575 R JNP ECHO / AS UNDERSCORE.
02598 / SPECT LAC ROBUF / FOR LINE DELETE, OUTPUT
02599 R 221326 R SAD LAW 15 / A CARRIAGE, FOLLOWED BY A
02600 R 541336 R SAD IOT 3004 / LINE OF BLANK SPACES...
02601 R 201312 R  RAC DLAY
/RE-INITIALIZE ALL COUNTERS,
/RE-INITIALIZE CHARACTER
/COUNTER AND TAB COUNTER
/UP POINTER POSITION.
/CARRIAGE RETURN MEANS
/WILL RETURN TO ENDRD.
/CHECK MODE.
/PACK CHARACTERS IF MODE 3,
/GET
/COMPLETE HEADER LATER.)
/INITIALIZE XCT INSTRUCTION
/HEADER WORD 2, GET
/BFST1 /INITIALIZE PACK POINTER TO
/HEADER WORD 2 ADDRESS,
/INITIALIZE THE DATA POINTER,
/INITIALIZE XCT INSTRUCTION
/USED TO DISPATCH TO THE
/PROPER PART OF PACKING PROCESS,
/INITIALIZE DOUBLE-WORD COUNT.
/INITIALIZE STOP FLAG,
/INCREMENT DISPATCHER (XCT OPRO),
/GET NEXT CHARACTER,
/IF CR, ADJUST EVENT
/VARIABLE.
/CHECK IF LAST CHARACTER
/UP POINTER.
/CARRIAGE RETURN MEANS
/IF MODE 2, GET
/INPUT,
/NUMBER OF CHARACTERS
(Main processor will form header.)
/SET EVENT VARIABLE, AND EXIT,
/CARRIAGE RETURN AS END OF BUFFER FLAG,
/IF CR, ADJUST EVENT
/CHECK IF LAST CHARACTER
/UP POINTER.
/CARRIAGE RETURN MEANS
/WILL RETURN TO ENDRD.
/CHECK MODE.
/PACK CHARACTERS IF MODE 3,
MOD10F

KEYBOARD-SCOPE HANDLER.

638 62712 R 402020 A  NXCH1  XCT  /GO TO PROPER CODE FOR
639 62713 R 602254 R  JMP  PACK1  / THE POSITION OF THIS
640 62714 R 622732 R  JMP  PACK2  / CHARACTER IN THE
641 62715 R 602733 R  JMP  PACK3  / DOUBLE-WORD.
642 62716 R 602753 R  JMP  PACK4
643 62717 R 602762 R  JMP  PACK5
644
645 62721 R 201300 R  EVSET  LAC  TWO  / EVENT VARIABLE = 2.
646 62721 R 241316 R  DAC  RDEV  / IF CR INPUT ON MODE 2.
647 62722 R 221336 R  LAC  CR  / RESTORE CR TO AC.
648 62723 R 602712 R  JMP  NXCH1
649
650 651 62724 R 746220 A  PACK1  CLRIRTR  / MOVE 8 PLACES AROUND TO
652 62725 R 742020 A  RTR  / POSITION IN BITS 0-6 OF FIRST
653 62726 R 742020 A  RTR  / HALF OF DOUBLE-WORD, OTHER BITS
654 62727 R 742020 A  RTR  / BLANK.
655 62731 R 602733 R  JMP  NXCHR
656
657 658 62732 R 746210 A  PACK2  CLIRTR  / MOVE 4 PLACES TO LEFT TO
659 62733 R 742210 A  RTL  / POSITION IN BITS 7-13 OF FIRST
660 62734 R 261335 R  XOR*  ROBUF  / HALF OF DOUBLE-WORD, THEN
661 62735 R 261335 R  DAC*  ROBUF  / COMBINE WITH FIRST CHARACTER.
662 62736 R 602713 R  JMP
663
664 665 62737 R 744420 A  PACK3  RCR  / SPLIT THIRD CHARACTER -
666 62740 R 742220 A  RTR  / FIRST 4 BITS TO BITS 14-17 OF
667 62741 R 241311 R  DAC  TEMP1  / HALF OF DOUBLE-WORD.
668 62742 R 521325 R  AND  MASK6
669 62743 R 261325 R  XCR*  ROBUF  / COMBINE WITH REST OF FIRST
670 62744 R 261325 R  DAC*  ROBUF  / HALF, COMPLETING IT.
671 62745 R 441325 R  ISZ  ROBUF
672 62746 R 221311 R  LAC  TEMP1  / OTHER THREE BITS OF CHARACTER
673 62747 R 742230 A  RAR  / GO IN BITS 0-2 OF SECOND HALF
674 62752 R 261325 R  DAC*  ROBUF
675 62752 R 602723 R  JMP  NXCHR
676
677 678 62753 R 744212 A  PACK4  CLIRTR  / MOVE 8 PLACES TO LEFT TO
679 62754 R 742210 A  RTL  / POSITION IN BITS 3-9 OF SECOND
680 62755 R 742210 A  RTL  / HALF OF DOUBLE-WORD.
681 62756 R 742210 A  RTL
682 62757 R 261325 R  XOR*  ROBUF  / COMBINE WITH THIRD CHARACTER
683 62760 R 261325 R  DAC*  ROBUF  / BITS.
684 62761 R 602723 R  JMP  NXCHR
685
686 687 62762 R 744112 A  PACK5  RCL  / MOVE 1 PLACE TO LEFT TO POSITION
688 62763 R 261325 R  XOR*  ROBUF  / IN BITS 10-16, COMBINE WITH
689 62764 R 261325 R  DAC*  ROBUF  / REST OF SECOND HALF OF DOUBLE-
690 62765 R 441325 R  ISZ  ROBUF  / WORD, NOW COMPLETE.
691 62766 R 441315 R  /  ISZ  WDCNT  / INCREMENT DOUBLE-WORD COUNT.
LAC PAGPT /RESET XCT INSTRUCTION
LAC NXCH1 /OPERAND FOR FIRST CHARACTER,
LAC ENDCP /CHECK IF END FLAG SET.
JMP NXCH1+1 /IF SET, PACKING FINISHED,
LAC WOCDT /FORM TRUE WORD COUNT.
DAC BiFSTP /WORD OF BUFFER.
LAC RDEV /PUT EVENT VARIABLE IN TRIGGER
DAC TOFLG /AND TRIGGER CAUSE FLAG.
JMP* KEIN /EXIT
STEND LAC RDBUF1 /MODIFICATION FOR PACKING LAST
DAC -1 /CHAR, PREVENT POINTER FROM INCREASING.
DAC RDBUF /PUT NONE-ZERO CODE IN STOP PLAG.
CLA /IF NOT FIRST TIME, USE NULL FOR
DAC RDEV /FOR CHARACTER,
DAC /THE BUFFER HEADER LENGTH SPECIFICATION
DAC /IS IGNORED - OUTPUT PROCESSES UNTIL
DAC /A CARRIAGE RETURN IS FOUND ON THE LINE
DAC /IS COMPLETED. CODES FOR LINE FEED
DAC /IMPLIED AUTOMATICALLY AT THE END OF A LINE),
DAC /CURSOR ADDRESS, AND CURSOR MODE (USE CLEAR
DAC /SCREEN INSTEAD! ARE TREATED AS ILLEGAL CHARACTERS.
DAC /IMAGE ASCII ASSUMED.
JMP UNPAC /UNPACKING ROUTINE,
DAC* ROBUF /IMAGE ASCII ASSUMED,
JMS OUTPT /OUTPUT CHARACTER, CHECKING
ISZ ROBUF /FOR END OF OUTPUT.
JMP .-3 /GO BACK FOR NEXT CHAR.
UNPAC LAC UNPPT /ROUTINE FOR UNPACKING
DAC UPXCT /AND OUTPUTTING 5/7 ASCII,
LAC* UPXCT /XCT INSTRUCTION OPERAND SET
UPXCT XCT /ACCORDING TO THE CHARACTER
JMP UNPK1 /POSITION IN THE DOUBLE-
JMP UNPK3 /WORD,
JMP UNPK4
JMP UNPK5
MOD10F
KEYBOARD-SCOPE HANDLER.

744  P1230 R 211326 R  UNPK1  LAC*  ROBUF  /FIRST CHAR. OF DOUBLEWORD.
745  P1037 R 742518 A  RTL  /MOVE BITS 8-15 TO LOW
746  P1247 R 742518 A  RTL  /ORDER POSITION.
747  P1141 R 742518 A  RTL
748  P1242 R 742518 A  RTL
749  P1243 R 121113 R  JMS  OUTPUT /OUTPUT CHARACTER.
750  P1244 R 621227 R  JMP  UNPK1*2 /GO GET NEXT CHARACTER.
751  P1245 R 221426 R  UNPK2  LAC*  ROBUF  /SECOND CHAR. OF DBL-WD.
752  P1146 R 742228 A  RTR  /MOVE BITS 7-13 TO LOW
753  P1247 R 742228 A  RTR  /ORDER POSITION.
754  P1152 R 101113 R  JMS  OUTPUT /OUTPUT CHARACTER.
755  P1151 R 621227 R  JMP  UNPK1*2 /GET ANOTHER.
756  P1252 R 221326 R  UNPK3  LAC*  ROBUF  /THIRD CHAR. OF DBL-WD.
757  P1153 R 521375 R  RCL  AND  MASK6  /MOVE FIRST FOUR BITS
758  P1154 R 742412 A  RTL  /TO POSITION 11-14.
759  P1155 R 742113 A  RTL
760  P1156 R 742113 A  DAC  TEMP1
761  P1257 R 441336 R  ISZ  ROBUF  /MOVE THE REMAINING THREE
762  P1151 R 921374 R  LAC*  ROBUF  AND  MASK7  /BITS TO THE LOW ORDER
763  P1152 R 742113 A  CLL/RTL  /POSITION.
764  P1153 R 742113 A  RTL
765  P1154 R 742113 A  XOR  TEMP1  /COMBINE THE TWO PARTS
766  P1155 R 121113 R  JMS  OUTPUT /OF THE CHAR. AND
767  P1156 R 742113 A  RTR
768  P1256 R 601227 R  JMP  UNPK1*2 /OUTPUT IT,
769  P1157 R 221326 R  UNPK4  LAC*  ROBUF  /FOURTH CHAR. OF DBL-WD.
770  P1158 R 742223 A  RTR  /MOVE BITS 3-9 TO THE
771  P1159 R 742223 A  RTR  /LOW ORDER POSITION,
772  P1257 R 601227 R  JMP  UNPK1*2 /OUTPUT CHARACTER,
773  P1158 R 221326 R  UNPK5  LAC*  ROBUF  /FIFTH CHAR. OF DBL-WD.
774  P1159 R 742223 A  RTR  /MOVE TO LOW ORDER POSITION.
775  P1258 R 101113 R  JMS  OUTPUT /OUTPUT CHARACTER,
776  P1259 R 441336 R  ISZ  ROBUF  /SET POINTER FOR NEXT WORD.
777  P1160 R 601027 R  JMP  UNPK1
778  P1161 R 601027 R  JMP  UNPK1
779  P1162 R 441336 R  ISZ  ROBUF
780  P1163 R 601225 R  JMP  UNPK1
781  P1164 R 742424 A  OUTPUT /ROUTINE TO TEST CHARACTER
782  P1165 R 742424 A  AND  MASKS5  /AND OUTPUT IT,
783  P1166 R 742424 A  DAC  TEMP1  /CHECK IF SPECIAL CHARACTER,
784  P1167 R 742424 A  DAC  TEMP1  AND  MASK8
785  P1168 R 741228 A  SWA
786  P1169 R 742228 A  JAC  TEMP1  /REGULAR CHARACTER,
787  P1161 R 201313 R  LAC  TEMP1  /IF ALTMODE, TREAT AS A LINE TERMINATOR,
788  P1162 R 541343 R  SAD  ALT  /IF ALTMODE, TREAT AS A LINE TERMINATOR,
789  P1163 R 621155 R  JMP  STPOT  /BUT NO NEW LINE,
790  P1164 R 541343 R  SAD  UDSOR  /*UNDERSCORE, TREAT AS
791  P1165 R 621123 R  JMP  OUTPUT /ILLEGAL CHARACTER.
792  P1166 R 121127 R  JMS  WAITF  /OTHERWISE, OUTPUT THE CHARACTER
793  P1167 R 201313 R  LAC  TEMP1  /TO THE 'SCOPE,
794  P1168 R 201313 R  LAC  TEMP1  /TO THE 'SCOPE,
795  P1169 R 3024 A  JST  COUNT  /INCREMENT CHARACTER COUNT,
MOD10F KEYBOARD-'SCOPE HANDLER.

777 01122 R 741020 A  ENDLN  / AND END LINE IF NEEDED.
778 01123 R 601153 R  JCMP  /INCREMENT TAB COUNT.
779 01124 R 441351 R  ISE  /RESETTING IF NEEDED.
780 01125 R 741220 A  SKP  / AS A LINE TERMINATOR.
781 01126 R 612127 R  JMP  /TSET.
782 01127 R 611183 R  JMP*  /EXIT.
783 01132 R 201311 R  JMP  /SIMULATE TABS.
784 01133 R 511341 A  JMP  /SPCH LAC TEMPI /SPECIAL CHARACTER HANDLER.
785 01134 R 612146 A  JMP  /RESETTING IF NEEDED.
786 01135 R 531351 A  JMP  /CARRY BACK.
787 01136 R 612156 R  JMP  /CLEAR SCREEN AND RESET.
788 01137 R 541357 A  JMP  /CLEAR SCREEN AND RESET.
789 01138 R 612152 R  JMP  /CLEAR SCREEN AND RESET.
790 01139 R 541342 R  JMP  /CLEAR SCREEN AND RESET.
791 01140 R 621123 R  JMP  /CLEAR SCREEN AND RESET.
792 01141 R 542116 A  JMP  /CLEAR SCREEN AND RESET.
793 01142 R 621122 R  JMP  /CLEAR SCREEN AND RESET.
794 01143 R 541340 R  JMP  /CLEAR SCREEN AND RESET.
795 01144 R 621124 R  JMP  /CLEAR SCREEN AND RESET.
796 01145 R 621123 R  JMP*  /OTHERS ARE ILLEGAL.
797 01146 R 121271 R  JMP  /CLEAR SCREEN AND RESET.
798 01147 R 762142 R  JMP  /CLEAR SCREEN AND RESET.
799 01150 R 733324 R  JMP  /CLEAR SCREEN AND RESET.
800 01151 R 441342 R  JMP  /CLEAR SCREEN AND RESET.
801 01152 R 621114 R  JMP  /CLEAR SCREEN AND RESET.
802 01153 R 101271 R  JMP  /CLEAR SCREEN AND RESET.
803 01154 R 762015 A  JMP  /CLEAR SCREEN AND RESET.
804 01155 R 752224 A  JMP  /CLEAR SCREEN AND RESET.
805 01156 R 441359 R  JMP  /CLEAR SCREEN AND RESET.
806 01157 R 741200 A  JMP  /CLEAR SCREEN AND RESET.
807 01158 R 121127 R  JMP  /CLEAR SCREEN AND RESET.
808 01159 R 777725 A  JMP  /CLEAR SCREEN AND RESET.
809 01160 R 441352 R  JMP  /CLEAR SCREEN AND RESET.
810 01161 R 777725 A  JMP  /CLEAR SCREEN AND RESET.
811 01162 R 741352 R  JMP  /CLEAR SCREEN AND RESET.
812 01163 R 612114 R  JMP  /CLEAR SCREEN AND RESET.
813 01164 R 621123 R  JMP*  /CLEAR SCREEN AND RESET.
814 01165 R 201111 R  JMP  /CLEAR SCREEN AND RESET.
815 01166 R 541307 R  JMP  /CLEAR SCREEN AND RESET.
816 01167 R 621554 R  JMP  /CLEAR SCREEN AND RESET.
817 01168 R 441352 R  JMP  /CLEAR SCREEN AND RESET.
818 01169 R 612127 R  JMP  /CLEAR SCREEN AND RESET.
819 01170 R 441351 A  JMP  /CLEAR SCREEN AND RESET.
820 01171 R 621111 R  JMP*  /CLEAR SCREEN AND RESET.
821 01172 R 741242 A  JMP  /CLEAR SCREEN AND RESET.
822 01173 R 777725 A  JMP  /CLEAR SCREEN AND RESET.
823 01174 R 241350 R  JMP  /CLEAR SCREEN AND RESET.
824 01175 R 511341 A  JMP  /CLEAR SCREEN AND RESET.
825 01176 R 612127 R  JMP  /CLEAR SCREEN AND RESET.
826 01177 R 621111 R  JMP*  /CLEAR SCREEN AND RESET.
827 01178 R 741242 A  JMP  /CLEAR SCREEN AND RESET.
828 01179 R 777725 A  JMP  /CLEAR SCREEN AND RESET.
829 01180 R 241350 R  JMP  /CLEAR SCREEN AND RESET.
830 01181 R 511341 A  JMP  /CLEAR SCREEN AND RESET.
831 01182 R 612127 R  JMP  /CLEAR SCREEN AND RESET.
832 01183 R 777725 A  JMP  /CLEAR SCREEN AND RESET.
833 01184 R 441351 A  JMP  /CLEAR SCREEN AND RESET.
834 01185 R 741200 A  JMP  /CLEAR SCREEN AND RESET.
835 01186 R 121127 R  JMP  /CLEAR SCREEN AND RESET.
836 01187 R 777725 A  JMP  /CLEAR SCREEN AND RESET.
837 01188 R 241352 R  JMP  /CLEAR SCREEN AND RESET.
838 01189 R 777725 A  JMP  /CLEAR SCREEN AND RESET.
839 01190 R 441351 A  JMP  /CLEAR SCREEN AND RESET.
840 01191 R 741200 A  JMP  /CLEAR SCREEN AND RESET.
841 01192 R 121127 R  JMP  /CLEAR SCREEN AND RESET.
842 01193 R 777725 A  JMP  /CLEAR SCREEN AND RESET.
843 01194 R 441351 A  JMP  /CLEAR SCREEN AND RESET.
844 01195 R 741200 A  JMP  /CLEAR SCREEN AND RESET.
845 01196 R 121127 R  JMP  /CLEAR SCREEN AND RESET.
846 01197 R 777725 A  JMP  /CLEAR SCREEN AND RESET.
847 01198 R 441351 A  JMP  /CLEAR SCREEN AND RESET.
848 01199 R 777725 A  JMP  /CLEAR SCREEN AND RESET.
849 01110 R 241350 R  JMP  /CLEAR SCREEN AND RESET.
MCO10F KEYBOARD-SCOPE HANDLER.

IF SIXTH SENSE SWITCH FROM RIGHT IS ON, THE USER WANTS A DELAY, WAIT UNTIL THE SWITCH IS OFF, CHECKING EVERY MILLISECOND.

CONTINUE, OUTPUT A LINE

FEED.

RETURN TO MAIN PROGRAM.

SIMULATE TAB BY SPACES.

MAY HAVE REACHED END OF THE LINE.

KEEP SENDING OUT SPACES UNTIL TAB COUNT IS RIGHT, THEN RESET COUNTER AND EXIT.

OUTPUT A BACKSPACE.

OUTPUT CHARACTER, BUT DO NOT INCREMENT COUNTERS.

OUTPUT CHARACTER, BUT

CURSOR UP ONE LINE ROUTINE.

SCOPE INITIALIZATION ROUTINE.
MOD10F KEYBOARD-SCOPE HANDLER.

923 01253 R 01271 R INIT JMS WAITF /IS THE 5TH PUSHTOPTY FROM THE
924 01251 R 703252 A IOT 3052 /IS THE 5TH PUSHBOTTOM FROM THE
925 01252 R 501351 R AND SXTN / RIGHT ON
926 01253 R 74272 A SIA
927 01254 R 601263 R JMP NOFF /YES, TREAT THE FORM FEED AS A CAR, RTN.
928 01255 R 763014 A LAW 14 /NO, BLANK SCREEN AND MOVE THE CURSOR TO
929 01256 R 733024 A IOT 3054 /THE TOP.
930 01257 R 101271 R JMS WAITF /FORCE ADDITIONAL DELAY.
931 01260 R 121271 R JMS WAITF /HANDLE THE NEW FRAME INITIALIZATION.
932 01261 R 121172 R JMS FRAME /HANDLE THE NEW FRAME INITIALIZATION.
933 01252 R 621123 R JHP OUTPT
934 01265 R 76772 A NOFF LAW 12 /ISSUE A LINE FEED IN PLACE OF THE
935 01264 R 733024 A IOT 3054 /NEW FRAME (FORM FEED).
936 01265 R 441350 R ISE PONT /CHECK IF END OF FRAME.
937 01266 R 621173 R JMP OUTPT /NO, RETURN.
938 01267 R 121172 R JMS FRAME /YES, INITIALIZE FOR IT BEFORE
939 01270 R 621173 R JHP OUTPT / RETURNING.
940 01272 / SUBROUTINE TO DELAY UNTIL AFTER THE NEXT CLOCK PULSE.
941 01273 / CONSTANTS AND STORAGE CELLS,
942 01271 R 74234 A WAITF XX /ENTRY POINT.
943 01272 R 211331 R LAC SFIT /AFTER NEXT CLOCK PULSE.
944 01273 R 24724 R DAC LEVEL7 +4 /RETURN TO CALLING ROUTINE.
945 01274 R 672444 R JMS LEVEL7 +6 /EXIT TO LEVEL 7 DISPATCHER.
946 01275 R 291334 R WATR LAC NOPIN /LEVEL 7 DISPATCHER.
947 01276 R 247442 R DAC LEVEL7 +4 /CONTROL HERE AFTER CLOCK
948 01277 R 621271 R JMP WAITF / PULSE, RETURN TO CALLER.
949
950 / SHARED CORE BUFFER ADDRESSES.
PAGE 22

MOD10F

KEYBOARD - SCOPE HANDLER.

956

957 007527 A TGFLG 07527 /TRIGGER CAUSE FLAG.

958 027530 A BUSY 07530 /BUSY FLAG.

959 037531 A MODE 07531 /MODE INDICATOR.

960 037532 A TS 07532 /TRIGGER.

961 037533 A BFFT 07533 /HEADER WORD 1.

962 01321 R 027534 A BFFT 07534 /HEADER WORD 2.

963 01322 R 027535 A BFFT 07535 /START OF DATA.

964 01324 R 037536 A SPEND 07536 /LAST DATA WORD, BUT

965 2 EXTRA USED FOR GUARDS.

966

967 01325 R 027537 A REGU 0 /POINTERS FOR THE

968 01326 R 027538 A REGU 0 /BUFFER.

969

970 01327 R 027539 A RREGU .DSA ENDRO-START

971 01330 R 037540 A SFTKB JMS KEYIN /FOR LEVEL 7 (SUB 3) DISPATCH.

972 01331 R 037541 A SFTK JMP WAIT /FOR LEVEL 7 (SUB 3) DISPATCH.

973 01332 R 037542 A PACPT XCT MXCHI+1 /FOR PACKING DISPATCHER.

974 01333 R 037543 A UNPPT XCT UXPCT /FOR UNPACKING DISPATCHER.

975 01334 R 037544 A NOPIN NOP /FOR CLEARING LEVEL 7 DISPATCH TABLE.

976

977

978

979

980 01335 R 037545 A LF 12 /LINE FEED.

981 01336 R 037546 A CR 15 /CARRIAGE RETURN.

982 01337 R 037547 A CLR 14 /CLEAR SCREEN & HOME CURSOR.

983 01338 R 037548 A BKSP = EIGHT /BACKSPACE (12).

984 01339 R 037549 A NUP 16 CURSOR ONE LINE UP.

985 01340 R 037550 A TAB 11 /TAB.

986 01341 R 037551 A VRTAB 13 /VERTICAL TAB.

987 01342 R 037552 UDS - MASKS / Rubin Gut - UNDERSCORE = 177.

988 01343 R 037553 A ALT 175 /PROPER ALT MODE.

989 01344 R 037554 A CTRU 25 /CONTROL U.

990 01345 R 037555 A CNTCTR 174 /CONTROL C (COMPLEMENT).

991 01346 R 037556 A CNTCTR 193 /CONTROL T (COMPLEMENT).

992 01347 R 037557 A CNTCRK 163 /CONTROL L (COMPLEMENT).

993

994

995 01352 R 037558 A PCNTR -17 /COUNTER FOR LINES/PAGE FOR SCOPE OUTPUT.

996 01353 R 037559 A TFLG 0 /COUNTER FOR TAB SIMULATION.

997 01354 R 037560 A COUNT 2 /COUNTER FOR CHAR./LINE.

998 01355 R 037561 A CLAY 0 /DELAY COUNTER FOR SCOPE OUTPUT.
TITLE GRAPHICS MODULE.

SCPE PLOT ROUTINES, SUPPORTED BY J. T. HEIBEL.
REVISED 7 FEBRUARY 1973 BY P. N. BARTRAM.

NOTE: SINCE THIS CODE OPERATES AT MAINLINE PRIORITY, AND
ONLY THE DO-NOTHING LOOP "DONO" ALSO OPERATES AT THIS LEVEL,
THE LIMIT AND INDEX REGISTERS ARE NOT SAVED UPON ENTRY TO
THIS ROUTINE. RATHER, IT IS THE RESPONSIBILITY OF HIGHER
PRIORITY PROGRAMMING TO SAVE THESE REGISTERS FOR THIS ROUTINE
IF THEY ARE USED.

DISPATCH TO ONE OF FIVE ROUTINES.

DISPATCH TO ROUTINE BASED ON BIT
ON IN THE DISPATCH FLAG WORD.

SCPLT XX ENTRY.
DAC PLSAV SAVE THE AC FOR "DONO".

dispatch to one of five routines.

DISPATCH TO ROUTINE BASED ON BIT
ON IN THE DISPATCH FLAG WORD.

SCPLT XX ENTRY.
DAC PLSAV SAVE THE AC FOR "DONO".
<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Address</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1052</td>
<td>01411 R 725222 A</td>
<td>AXS</td>
<td>200</td>
</tr>
<tr>
<td>1053</td>
<td>01412 R 601426 R</td>
<td>JMP</td>
<td>WONE</td>
</tr>
<tr>
<td>1054</td>
<td>01413 R 601371 R</td>
<td>JMP</td>
<td>XT</td>
</tr>
<tr>
<td>1055</td>
<td>01414 R 207474 A</td>
<td>VL</td>
<td>LAC</td>
</tr>
<tr>
<td>1056</td>
<td>01415 R 241560 R</td>
<td>DAC</td>
<td>INT</td>
</tr>
<tr>
<td>1057</td>
<td>01416 R 227473 A</td>
<td>LAC</td>
<td>Y</td>
</tr>
<tr>
<td>1058</td>
<td>01417 R 241562 R</td>
<td>DAC</td>
<td>XOUT</td>
</tr>
<tr>
<td>1059</td>
<td>01418 R 207471 A</td>
<td>LAC</td>
<td>X1</td>
</tr>
<tr>
<td>1060</td>
<td>01419 R 721222 A</td>
<td>PAX</td>
<td></td>
</tr>
<tr>
<td>1061</td>
<td>01420 R 207472 A</td>
<td>LAC</td>
<td>X2</td>
</tr>
<tr>
<td>1062</td>
<td>01421 R 722225 A</td>
<td>PAL</td>
<td></td>
</tr>
<tr>
<td>1063</td>
<td>01422 R 724233 A</td>
<td>TWOH</td>
<td>PXA</td>
</tr>
<tr>
<td>1064</td>
<td>01423 R 724554 R</td>
<td>DAC</td>
<td>YOUT</td>
</tr>
<tr>
<td>1065</td>
<td>01424 R 601432 R</td>
<td>JMS</td>
<td>PN</td>
</tr>
<tr>
<td>1066</td>
<td>01425 R 725223 A</td>
<td>AXS</td>
<td>200</td>
</tr>
<tr>
<td>1067</td>
<td>01426 R 601432 R</td>
<td>JMP</td>
<td>TWOH</td>
</tr>
<tr>
<td>1068</td>
<td>01427 R 725223 A</td>
<td>JMS</td>
<td>XT</td>
</tr>
<tr>
<td>1069</td>
<td>01431 R 601371 R</td>
<td>LOADH=706224</td>
<td>/LOAD SCOPE 10T.</td>
</tr>
<tr>
<td>1070</td>
<td>01432 R 725224 A</td>
<td>DACLO=707124</td>
<td></td>
</tr>
<tr>
<td>1071</td>
<td>01433 R 725224 A</td>
<td>PN</td>
<td>XX</td>
</tr>
<tr>
<td>1072</td>
<td>01434 R 725124 A</td>
<td>LAC</td>
<td>(100200</td>
</tr>
<tr>
<td>1073</td>
<td>01435 R 725124 A</td>
<td>LOADH</td>
<td></td>
</tr>
<tr>
<td>1074</td>
<td>01436 R 725124 A</td>
<td>LAC</td>
<td>YOUT</td>
</tr>
<tr>
<td>1075</td>
<td>01437 R 725124 A</td>
<td>TCA</td>
<td>(777760</td>
</tr>
<tr>
<td>1076</td>
<td>01438 R 725124 A</td>
<td>AND</td>
<td></td>
</tr>
<tr>
<td>1077</td>
<td>01439 R 725124 A</td>
<td>XOR</td>
<td>MASK6</td>
</tr>
<tr>
<td>1078</td>
<td>01440 R 725124 A</td>
<td>DAC</td>
<td>TABLE</td>
</tr>
<tr>
<td>1079</td>
<td>01441 R 725124 A</td>
<td>DACLO</td>
<td>/STORE DAC SETTING IN GENERAL SERVICE TABLE.</td>
</tr>
<tr>
<td>1080</td>
<td>01442 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1081</td>
<td>01443 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1082</td>
<td>01444 R 725124 A</td>
<td>DAC</td>
<td>TABLE+1</td>
</tr>
<tr>
<td>1083</td>
<td>01445 R 725124 A</td>
<td>DACLO</td>
<td>/STORE DAC SETTING IN GENERAL SERVICE TABLE.</td>
</tr>
<tr>
<td>1084</td>
<td>01446 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1085</td>
<td>01447 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1086</td>
<td>01448 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1087</td>
<td>01449 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1088</td>
<td>01450 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1089</td>
<td>01451 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1090</td>
<td>01452 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1091</td>
<td>01453 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1092</td>
<td>01454 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1093</td>
<td>01455 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1094</td>
<td>01456 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1095</td>
<td>01457 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1096</td>
<td>01458 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1097</td>
<td>01459 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1098</td>
<td>01460 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1099</td>
<td>01461 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td>01462 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td>01463 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1102</td>
<td>01464 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1103</td>
<td>01465 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1104</td>
<td>01466 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1105</td>
<td>01467 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1106</td>
<td>01468 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1107</td>
<td>01469 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1108</td>
<td>01470 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1109</td>
<td>01471 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>01472 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td>01473 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1112</td>
<td>01474 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1113</td>
<td>01475 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1114</td>
<td>01476 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1115</td>
<td>01477 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1116</td>
<td>01478 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1117</td>
<td>01479 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1118</td>
<td>01480 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1119</td>
<td>01481 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1120</td>
<td>01482 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1121</td>
<td>01483 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1122</td>
<td>01484 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
<tr>
<td>1123</td>
<td>01485 R 725124 A</td>
<td>DACLO</td>
<td></td>
</tr>
<tr>
<td>1124</td>
<td>01486 R 725124 A</td>
<td>DAC</td>
<td></td>
</tr>
</tbody>
</table>

Note: The table above contains machine code instructions with addresses. Each instruction is followed by notes in parentheses, indicating specific values or actions.
<table>
<thead>
<tr>
<th>PAGE</th>
<th>MOD</th>
<th>GRAPHICS MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1175</td>
<td>01466</td>
<td>R 207471 A</td>
</tr>
<tr>
<td>1176</td>
<td>01467</td>
<td>R 241562 R</td>
</tr>
<tr>
<td>1177</td>
<td>01470</td>
<td>R 207472 A</td>
</tr>
<tr>
<td>1178</td>
<td>01471</td>
<td>R 241564 R</td>
</tr>
<tr>
<td>1179</td>
<td>01472</td>
<td>R 207473 A</td>
</tr>
<tr>
<td>1180</td>
<td>01473</td>
<td>R 041568 R</td>
</tr>
<tr>
<td>1181</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>1182</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>1183</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>1184</td>
<td>01474</td>
<td>R 121432 R</td>
</tr>
<tr>
<td>1185</td>
<td>01475</td>
<td>R 621371 R</td>
</tr>
<tr>
<td>1186</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>1187</td>
<td>01476</td>
<td>R 222352 R</td>
</tr>
<tr>
<td>1188</td>
<td>01477</td>
<td>R 726724 A</td>
</tr>
<tr>
<td>1189</td>
<td>01522</td>
<td>R 201370 R</td>
</tr>
<tr>
<td>1190</td>
<td>01521</td>
<td>R 741585 R</td>
</tr>
<tr>
<td>1191</td>
<td>01522</td>
<td>R 441555 R</td>
</tr>
<tr>
<td>1192</td>
<td>01523</td>
<td>R 621452 R</td>
</tr>
<tr>
<td>1193</td>
<td>01524</td>
<td>R 752372 A</td>
</tr>
<tr>
<td>1194</td>
<td>01525</td>
<td>R 726724 A</td>
</tr>
<tr>
<td>1195</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>1196</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>1197</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>1198</td>
<td>01526</td>
<td>R 222222 R</td>
</tr>
<tr>
<td>1199</td>
<td>01527</td>
<td>R 726724 A</td>
</tr>
<tr>
<td>1200</td>
<td>01532</td>
<td>R 221370 R</td>
</tr>
<tr>
<td>1201</td>
<td>01510</td>
<td>R 441555 R</td>
</tr>
<tr>
<td>1202</td>
<td>01512</td>
<td>R 041560 R</td>
</tr>
<tr>
<td>1203</td>
<td>01513</td>
<td>R 752372 A</td>
</tr>
<tr>
<td>1204</td>
<td>01514</td>
<td>R 621371 R</td>
</tr>
<tr>
<td>1205</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>1206</td>
<td>01515</td>
<td>R 601371 R</td>
</tr>
<tr>
<td>1207</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>1208</td>
<td>01516</td>
<td>R 207475 A</td>
</tr>
<tr>
<td>1209</td>
<td>01517</td>
<td>R 207476 A</td>
</tr>
<tr>
<td>1210</td>
<td>01522</td>
<td>R 241561 R</td>
</tr>
<tr>
<td>1211</td>
<td>01521</td>
<td>R 207472 A</td>
</tr>
<tr>
<td>1212</td>
<td>01522</td>
<td>R 241563 R</td>
</tr>
<tr>
<td>1213</td>
<td>01523</td>
<td>R 227473 A</td>
</tr>
<tr>
<td>1214</td>
<td>01524</td>
<td>R 241556 R</td>
</tr>
<tr>
<td>1215</td>
<td>01525</td>
<td>R 241566 R</td>
</tr>
<tr>
<td>1216</td>
<td>01526</td>
<td>R 207474 A</td>
</tr>
<tr>
<td>1217</td>
<td>01527</td>
<td>R 702070 A</td>
</tr>
<tr>
<td>1218</td>
<td>01530</td>
<td>R 227476 A</td>
</tr>
<tr>
<td>1219</td>
<td>01532</td>
<td>R 601371 R</td>
</tr>
<tr>
<td>1220</td>
<td>01533</td>
<td>R 752372 A</td>
</tr>
<tr>
<td>1221</td>
<td>01534</td>
<td>R 726724 A</td>
</tr>
<tr>
<td>1222</td>
<td>01535</td>
<td>R 702070 A</td>
</tr>
<tr>
<td>1223</td>
<td>01536</td>
<td>R 772070 A</td>
</tr>
<tr>
<td>1224</td>
<td>01537</td>
<td>R 621371 R</td>
</tr>
<tr>
<td>1225</td>
<td>01538</td>
<td>R 207475 A</td>
</tr>
<tr>
<td>1226</td>
<td>01539</td>
<td>R 227474 A</td>
</tr>
<tr>
<td>1227</td>
<td>01540</td>
<td>R 441556 R</td>
</tr>
<tr>
<td>1228</td>
<td>01541</td>
<td>R 041560 R</td>
</tr>
<tr>
<td>1229</td>
<td>01542</td>
<td>R 601371 R</td>
</tr>
<tr>
<td>1230</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>1231</td>
<td>01543</td>
<td>R 241564 R</td>
</tr>
</tbody>
</table>

Notes:
- **PNT**: LAC X1
- **DAC**: XOUT
- **DAC**: X2
- **DAC**: YOUT
- **DAC**: Y
- **DAC**: INT
- **JMP**: PN
- **JMP**: XT
- **SCLR**: LAC FS81T
MOD1RF

GRAPHS MODULE.

PAGE 26

1158 01543 R 341557 R TAD DELYH
1159 01544 R 041563 R DAC YIT
1160 /
1161 01545 R 101432 R JMS PN
1162 /
1163 01546 R 725291 A AXS 1
1164 01547 R 621555 R JMP GO
1165 01550 R 621371 R JMP XT
1166 /
1167 01551 R 022273 A PLSAV B
1168 .LTORG / AC SAVE CELL.

1169 01552 R 022220 A =L
01553 R 777762 A =L
01554 R 147773 A =L

1169 .END SIZE=01565 NO ERROR LINES
EDIT #2

MCR FUNCTION: DUAL PROCESSOR PATCHES 07 MAR 73 P. BARTRAM

TASK NAME: "...DPP". TASK MODIFIES THE RSX PLUS MONITOR TO RECOGNIZE ADDITIONAL CAL DIRECTIVES 31, 32, 33, & 34, AND TO FIX IN CORE THE ROUTINES IMPLEMENTING THESE DIRECTIVES AND THE ROUTINES FOR THE SECONDARY PROCESSOR.

THE SYNTAX OF THE COMMAND INPUT LINE IS

"DPP"$<CHARACTER><CR><AM>

<CHARACTER> = <LETTER>/<DIGIT>
<CR> = CARRIAGE RETURN
<AM> = ALTMODE
$ "ANY NUMBER OF, INCLUDING ZERO"

THE RESIDENT MCR READS THE FIRST THREE CHARACTERS "DPP", FLUSHES THE NEXT CHARACTERS UNTIL A BREAK OR TERMINATOR CHARACTER IS FOUND. "...DPP" FLUSHES ANY REMAINING CHARACTERS, SAVING THE TERMINATOR ONLY. NO DATA ARE TAKEN FROM THE INPUT LINE.

"...DPP" HONORS THE TERMINATOR CONVENTION OF OTHER MCR TASKS.
*** MCR FUNCTION "DUAL PROCESSOR PATCHES" (DPP). ***

TITLE MCR FUNCTION "DUAL PROCESSOR PATCHES" (DPP).

OCT

/ DUAL PROCESSOR PATCHES (DPP).

/ WHICHEVER IN "TERM".

/ WHICH SAVE IN "TERM".

/ THE TERMINATOR,

/ THROUGH TO THE TERMINATOR,

/ WHICH SAVE IN "TERM".

/ THE "TASK" MOD12 WHICH CONTAINS THE SECONDARY PROCESSOR

/ PROGRAMS MUST BE BUILT TO RUN IN THE LOWER PART OF SHARED

/ MEMORY, AND A PARTITION DEDICATED THERE FOR ITS USE. THE

/ "TASK" PATCH, WHICH CONTAINS ADDITIONAL MONITOR ROUTINES,

/ MUST BE BUILT TO RUN IN THE LOWER 32K MEMORY, PREFERABLY

/ NEAR THE MONITOR. A PARTITION MUST BE DEDICATED FOR IT ALSO.

/ A THIRD PARTITION, NAMED "SHARE" MUST BE PROVIDED SUCH THAT

/ ITS TOP ADDRESS IS 7777 (OCTAL) GREATER THAN THE START OF

/ SHARED MEMORY.

/ CURRENTLY, THE PROGRAMS FOR THE SECONDARY

/ PROCESSOR IN SHARED CORE.

/ FIX THE ADDITIONAL MONITOR ROUTINES IN CORE,

/ CHECK TO INSURE THAT PARTITION "SHARE" IS PROPERLY DEFINED.

/ IF SO, INSURE THAT NO TASK IS EVER INSTALLED THERE. ("SHARE"

/ IS USED FOR DUAL PROCESSOR COMMUNICATION FLAGS AND BUFFERS.)

/ SEARCH THE PARTITION BLOCK DESCRIPTOR LIST

/ DEQUE FOR PARTITION "SHARE", AND GET THE

/ NODE ADDRESS.

/ "SHARE" IS NOT IN THE SYSTEM.

/ IT IS IN THE SYSTEM, GET THE PARTITION

/ BASE ADDRESS (WORD 4 OF NODE),

/ CURRENTLY, SHARED MEMORY STARTS AT 62000 (OCTAL).

/ CURRENTLY, SHARED MEMORY STARTS AT 62000 (OCTAL).

/ BASE ADDRESS (WORD 4 OF NODE),

/ CHECK TO INSURE THAT PARTITION "SHARE" IS PROPERLY DEFINED.

/ IF SO, INSURE THAT NO TASK IS EVER INSTALLED THERE. ("SHARE"

/ IS USED FOR DUAL PROCESSOR COMMUNICATION FLAGS AND BUFFERS.)

/ SEARCH THE PARTITION BLOCK DESCRIPTOR LIST

/ DEQUE FOR PARTITION "SHARE", AND GET THE

/ NODE ADDRESS.

/ "SHARE" IS NOT IN THE SYSTEM.

/ IT IS IN THE SYSTEM, GET THE PARTITION

/ BASE ADDRESS (WORD 4 OF NODE),

/ CURRENTLY, SHARED MEMORY STARTS AT 62000 (OCTAL).

/ CURRENTLY, SHARED MEMORY STARTS AT 62000 (OCTAL).

/ BASE ADDRESS (WORD 4 OF NODE),
PAGE 3

DPP.02

*** MCR FUNCTION "DUAL PROCESSOR PATCHES" (DPP).***

00332 R 4437717 R ISZ TEMP //ADD THE PARTITION LENGTH (WORD 5 OF THE
3027314 R 36e7717 R TAD TEMP //NODE.
302732 R 5457715 R SAD TPSHR //IF THE BASE + LENGTH IS NOT EQUAL TO 10000
822533 R 665363 R JMP +1 //GREATER THAN THE ASE OF SHARED MEMORY,
812534 R 222564 R LAC AOD51 //THE PARTITION IS IMPROPERLY DEFINED.
822535 R 662144 R JMP KOD5MR //PRINT DIAGNOSTIC AND EXIT.
302736 R 222717 R LAC TEMP //THE PARTITION IS ACCEPTED. SET THE
842737 R 723233 A AAD +3 //OCCUPIED FLAG (BIT 0, WORD 10 OF THE
852242 R 2242717 R DAC TEMP //NODE).
862241 R 222714 R DAC SG517
872242 R 826717 R DAC TEMP

/ INSERT THE CODE PATCHES INTO THE MONITOR.

02243 R 222346 R DAC RTINT //INSERT THE JUMP ADDRESS FOR THE INTERRUPT FROM
02244 R 222345 R DAC ADD41 //THE SECONDARY PROCESSOR CLOCK PROGRAM.
03245 R 222332 R DAC CAL31 //INSERT JUMP TO CAL DIRECTIVE 31 INTO
04246 R 222321 R DAC LC637 //THE CAL DISPATCH TABLE.
052047 R 222326 R LAC CAL32 //INSERT JUMP TO CAL DIRECTIVE 32 INTO
052250 R 222325 R DAC LC642 //THE CAL DISPATCH TABLE.
052251 R 222332 R DAC CAL33 //INSERT JUMP TO CAL DIRECTIVE 33 INTO
052252 R 222331 R DAC LC641 //THE CAL DISPATCH TABLE.
052253 R 222335 R DAC CAL34 //INSERT JUMP TO CAL DIRECTIVE 34 INTO
062254 R 222335 R DAC LC642 //THE CAL DISPATCH TABLE.
072755 R 222334 R DAC ADD31 //INSERT JUMP ADDRESS FOR CAL DIRECTIVE
072255 R 222333 R DAC LC623 //31 INTO BANK A.
072256 R 222333 R DAC ADD32 //INSERT JUMP ADDRESS FOR CAL DIRECTIVE
072257 R 222332 R DAC LC621 //32 INTO BANK C.
072250 R 222327 R DAC ADD33 //INSERT JUMP ADDRESS FOR CAL DIRECTIVE
072251 R 222332 R DAC LC532 //33 INTO BANK A.
072252 R 222331 R DAC ADD34 //INSERT JUMP ADDRESS FOR CAL DIRECTIVE
072253 R 222331 R DAC LC323 //34 INTO BANK C.
072254 R 222334 R DAC LC244 //INSERT CHANGES IN CAL DIRECTIVE 16 -
072255 R 222341 R DAC L2464 //UNFIX - TO JUMP TO THE ADDITIONAL
072256 R 222342 R DAC L2465 //CAL DIRECTIVE.
072257 R 222343 R DAC LC2465 //CODE IN PATCH.

/ ZERO OUT THE THREE REAL-TIME INPUT REQUEST TABLES,

02271 R 222347 R LAC TABLS //ADDRESS THE TABLES VIA THE INDEX REGISTER.
02272 R 721228 A PAC
02273 R 721328 A AAC +300
02274 R 722222 A PAL
02275 R 152230 A DEM 0.X
02276 R 722231 A AXS +1
02277 R 662275 R JMP +2
022123 R 722243 A AAC +40 //PUT "N-A" FOR ALL ENTRIES IN THE OUTPUT
02121 R 722222 A PAL //PROTECT TABLE.
02122 R 222675 R LAC NA
02123 R 250003 A DAC 0.X
02124 R 722221 A AXS +1
02125 R 662213 R JMP +2

/
E X T 1 M C R D I S P A T C H E R 1 H O N I T O R, C L E A R I N G M C R F L A G.
00142 R 200302 R C A L E X I T /T H E N E X I T, 
*** MCR FUNCTION "DUAL PROCESSOR PATCHES" (DPP),

ENTRY POINT.

COMPLETE THE CAL PARAMETER BLOCK.

ISSUE THE FIX CAL DIRECTIVE.

GET THE NAME OF THE "TASK" JUST FIXED. CONVERT TO IMAGE ASCII, AND PLACE IN A PRINT BUFFER.

FIRST CHARACTER OBTAINED, NOW GET THE SECOND.

THE THIRD CHARACTER.

THE FOURTH.

THE FIFTH.

THE LAST CHARACTER OF THE NAME.

SET UP THE CAL PARAMETER BLOCK FOR PRINTING THE "TASK" NAME.

BUFFER ADDRESS TRANSFERRED ABOVE, NOW SEND OVER THE DATA MODE.

ISSUE THE PRINT CAL DIRECTIVE.

CHECK THE FIX DIRECTIVE RETURNED EVENT VARIABLE. IF POSITIVE, FIX IS O.K.

OTHERWISE, DETERMINE THE SOURCE OF ERROR AND PRINT DIAGNOSTIC.

ERROR! PARTITION OCCUPIED, PRINT DIAGNOSTIC.

ERROR: BUT NOT ONE THAT SHOULD BE POSSIBLE.

WITH THE FIX DIRECTIVE, PRINT DIAGNOSTIC.

ERROR: NO AVAILABLE NODES, PRINT DIAGNOSTIC.

ERROR: PARTITION OCCUPIED, PRINT DIAGNOSTIC.
*** MCR FUNCTION "DUAL PROCESSOR PATCHES" (DPP).***

236 00240 R 622252 R JMP PRINT
237 00241 R 622240 R ER207 LAC AD207 /ERROR: ALREADY FIXED IN CORE (NOT A FATAL
238 00242 R 622252 R JMP PRINT /ERROR, LIKE THE OTHERS). PRINT DIAGNOSTIC,
239 00243 R 622241 R ER24 LAC AD24 /ERROR: TASK DISABLED. PRINT DIAGNOSTIC, THOUGH
240 00244 R 622252 R JMP PRINT /IT WOULD BE UNREASONABLE FOR THIS ERROR TO OCCUR,
241 00245 R 622242 R ER202 LAC AD202 /ERROR: TASK ACTIVE. PRINT DIAGNOSTIC, THOUGH
242 00246 R 622252 R JMP PRINT /THE "TASK" IS NEVER SCHEDULED, THUS SHOULD
243 00247 R 622243 R ER201 LAC AD201 /NEVER BE ACTIVE.
244 00248 R 622252 R JMP PRINT /ERROR: TASK NOT INSTALLED IN THE SYSTEM,
245 00249 R 622252 R FIXOK LAC AD50 /PRINT SUCCESSFUL FIX MESSAGE,
246 00250 R 622252 R ER252 R 242377 R PRINT DAC TPCPB=4 /PRINT CAL PARAMETER BLOCK COMPLETE,
247 00253 R 622271 R LAC T40 /PUT MODE 2 CODE IN CAL PARAMETER BLOCK.
248 00254 R 622256 R DAC TPCPB=3 /DAS or TPCPB=4
249 00255 R 622373 R CAL TPCPB /ISSUE THE CAL PRINT DIRECTIVE,
250 00256 R 622311 R CAL WTPRT /CAL PRINT DIRECTIVE.
251 00257 R 622322 R DAC EVFIX /IF UNSUCCESSFUL, EXIT TO THE MCR DISPATCHER.
252 00258 R 622372 R LAC EVFIX /ALREADY FIXED "ERROR" CONSIDERED SUCCESSFUL.
253 00259 R 622153 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
254 00260 R 622153 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
255 00261 R 622153 R CAL R3MCR /REQUEST MCR DISPATCHER TASK.
256 00262 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
257 00263 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
258 00264 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
259 00265 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
260 00266 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
261 00267 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
262 00268 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
263 00269 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
264 00270 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
265 00271 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
266 00272 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
267 00273 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
268 00274 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
269 00275 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
270 00276 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
271 00277 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
272 00278 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
273 00279 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
274 00280 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
275 00281 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
276 00282 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
277 00283 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
278 00284 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
279 00285 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
280 00286 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
281 00287 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
282 00288 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
283 00289 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
284 00290 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
285 00291 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
286 00292 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
287 00293 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
288 00294 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
289 00295 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
290 00296 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
291 00297 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
292 00298 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
293 00299 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
294 00300 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
295 00301 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
296 00302 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
297 00303 R 622260 R JMP TSKFX /SUCCESSFUL, RETURN TO FINISH THIS PROGRAM.
DPP.02 MCR FUNCTION "DUAL PROCESSOR PATCHES" (DPP).

/ WAIT FOR COMPLETION OF PRINT-OUT CPB.
00311 R 000020 A WPRINT 20 / WAIT FOR EVENT VARIABLE TPEV TO
00312 R 003112 R .DSA TPEV / TO BE SET CPB.

/ REQUEST MCR CPB.
00313 R 003001 A RQMCR 1 / REQUEST EXECUTION OF TASK ... MCR CPB.
00314 R 003002 A RQMCR 2 / (NO EVENT VARIABLE SPECIFIED.)
00315 R 00316 A SIXBT "...MCR"

00316 R 003012 A t / (USE DEFAULT PRIORITY.)

/ EXIT CPB.
00320 R 003010 A EXIT / EXIT CPB.

/ CONSTANTS USED FOR MODIFYING THE RSX PLUS MONITOR.
/ UNUSED MEMORY LOCATIONS 00320-00324 WILL BE USED AS ADDRESS
/ POINTERS FOR INDIRECT JUMPS FOR THE NEW CAL DIRECTIVES

/ LOCATED OUTSIDE BANK 0,

00321 R 003067 A LC637 .DSA 00327 / LOCATION 00637 TO BE CHANGED TO AN
00322 R 003070 A CAL31 JMP 00322 / INDIRECT) JUMP TO THE CAL DIRECTIVE
00323 R 003072 A LC322 .DSA 00322 / 31 - OBTAIN A WORD FROM SHARED CORE.
00324 R 003134 A AD31 .DSA GETSC

00325 R 003040 A LC640 .DSA 00324 / LOCATION 00640 TO BE CHANGED TO AN
00326 R 003042 A CAL32 JMP 00324 / INDIRECT) JUMP TO THE CAL DIRECTIVE
00327 R 003044 A LC324 .DSA 00324 / 32 - PLACE A WORD INTO SHARED CORE.
00328 R 003134 A AD32 .DSA PUTSC

00329 R 003046 A LC642 .DSA 00326 / LOCATION 00642 TO BE CHANGED TO AN
00330 R 003048 A CAL33 JMP 00326 / INDIRECT) JUMP TO THE CAL DIRECTIVE
00331 R 003050 A LC326 .DSA 00326 / 33 - PLACE AN ENTRY INTO THE AUTO-FILL
00332 R 003134 A AD33 .DSA ADTEN / REAL-TIME INPUT REQUEST TABLE.

00333 R 003052 A LC644 .DSA 00328 / LOCATION 00644 TO BE CHANGED TO AN
00334 R 003054 A CAL34 JMP 00328 / INDIRECT) JUMP TO THE CAL DIRECTIVE
00335 R 003056 A LC328 .DSA 00328 / 34 - RESET AUTO-FILL REAL-TIME INPUT
00336 R 003134 A AD34 .DSA RTST / TABLE ENTRIES,

00337 R 003058 A LC646 .DSA 00330 / LOCATION 00646 TO BE CHANGED TO AN
00338 R 003060 A CAL35 JMP 00330 / INDIRECT) JUMP TO THE CAL DIRECTIVE
00339 R 003062 A LC330 .DSA 00330 / 35 - REKennotto THE LEAD-THROUGH TABLE,
00340 R 003134 A AD35 .DSA RST / TABLE ENTRIES,

00341 R 003064 A LC648 .DSA 00332 / LOCATION 00648 TO BE CHANGED TO AN
00342 R 003066 A CAL36 JMP 00332 / INDIRECT) JUMP TO THE CAL DIRECTIVE
00343 R 003068 A LC332 .DSA 00332 / 36 - REKennotto THE LEAD-THROUGH TABLE,
00344 R 003134 A AD36 .DSA RST / TABLE ENTRIES,

00345 R 003070 A LC650 .DSA 00334 / LOCATION 00650 TO BE CHANGED TO AN ADDRESS
00346 R 003072 A CAL37 JMP 00334 / INDIRECT) JUMP TO THE CAL DIRECTIVE
00347 R 003074 A LC334 .DSA 00334 / 37 - INSERT AN ENTRY INTO THE AUTO-FILL
00348 R 003134 A AD37 .DSA ADTEN / REAL-TIME INPUT REQUEST TABLE,

00349 R 003076 A LC654 .DSA 00336 / LOCATION 00654 TO BE CHANGED TO AN
00350 R 003078 A CAL38 JMP 00336 / INDIRECT) JUMP TO THE CAL DIRECTIVE
00351 R 003080 A LC336 .DSA 00336 / 38 - REKennotto THE LEAD-THROUGH TABLE,
00352 R 003134 A AD38 .DSA RST / TABLE ENTRIES,

00353 R 003082 A LC658 .DSA 00338 / LOCATION 00658 TO BE CHANGED TO AN ADDRESS
00354 R 003084 A CAL39 JMP 00338 / INDIRECT) JUMP TO THE CAL DIRECTIVE
00355 R 003086 A LC338 .DSA 00338 / 39 - INSERT AN ENTRY INTO THE AUTO-FILL
00356 R 003134 A AD39 .DSA ADTEN / REAL-TIME INPUT REQUEST TABLE,

00357 R 003088 A LC660 .DSA 00340 / LOCATION 00660 TO BE CHANGED TO AN ADDRESS
00358 R 003090 A CAL30 JMP 00340 / INDIRECT) JUMP TO THE CAL DIRECTIVE
00359 R 003092 A LC340 .DSA 00340 / 40 - INSERT AN ENTRY INTO THE AUTO-FILL
00360 R 003134 A AD30 .DSA ADTEN / REAL-TIME INPUT REQUEST TABLE,

00361 R 003094 A LC664 .DSA 00342 / LOCATION 00664 TO BE CHANGED TO AN ADDRESS
00362 R 003096 A CAL31 JMP 00342 / INDIRECT) JUMP TO THE CAL DIRECTIVE
00363 R 003098 A LC342 .DSA 00342 / 41 - INSERT AN ENTRY INTO THE AUTO-FILL
00364 R 003134 A AD31 .DSA ADTEN / REAL-TIME INPUT REQUEST TABLE,

00365 R 003100 A LC666 .DSA 00344 / LOCATION 00666 TO BE CHANGED TO AN ADDRESS
00366 R 003102 A CAL32 JMP 00344 / INDIRECT) JUMP TO THE CAL DIRECTIVE
00367 R 003104 A LC344 .DSA 00344 / 42 - INSERT AN ENTRY INTO THE AUTO-FILL
00368 R 003134 A AD32 .DSA ADTEN / REAL-TIME INPUT REQUEST TABLE,

/ ADDRESSES FOR THE NEW CAL DIRECTIVE ENTRY POINTS,
00346 R 003140 A PRINT 1000 / THE INTERRUPT HANDLER FOR THE SECONDARY PROC.
00347 R 003142 A PRINT 1410 / THE INTERRUPT HANDLER FOR THE SECONDARY PROC.
00348 R 003144 A PRINT 1410 / THE INTERRUPT HANDLER FOR THE SECONDARY PROC.

/ ADDRESSES FOR THE NEW CAL DIRECTIVE ENTRY POINTS,
00350 R 003142 A PUTSC+PTCH+400 / CAL 32.
00351 R 003144 A ADTEN+PTCH+500 / CAL 33.
PAGE 8

DPP.02

*** MCR FUNCTION "DUAL PROCESSOR PATCHES" (DPP).

013640 A RTRST=PTCH=640 /CAL 34.
013642 / ADDRESS FOR THE NEW SYSTEM TABLES.
013644 00347 R 003000 A TAPS .DSA -BANK*1000=PTCH /ADJUSTED FOR USE WITH INDEX REG.
013645 / NAMES OF "TASKS" TO BE FIXED IN CORE.
013647 00350 R 561517 A MOD10 "SIXBT", "MOD10".
013648 02352 R 562271 A PATCH "SIXBT", "PATCH".
013649 / NAME OF PARTITION FOR DUAL PROCESSOR COMMUNICATION FLAGS AND BUFFERS.
013650 00353 R 240310 A SHARE .DSA +1.
013651 / OUTPUT MESSAGE BUFFERS.
013652 00357 R 222355 R ATBSF .DSA TSKBF /BUFFER FOR THE "TASK NAME", MODE 2.
013653 022362 R 222353 A TSKBF 025283.
013654 00365 R 222353 A ?
013655 020363 R 742243 A XX /SIX CHARACTER NAME TO BE FILLED IN
013656 022363 R 742243 A XX /PRIOR TO REQUESTING PRINT-OUT.
013657 022364 R 742243 A XX
013658 022365 R 742243 A XX
013659 022366 R 742243 A XX
013660 022367 R 742243 A XX
013661 022372 R 002240 A 020340 /SPACE.
013662 022371 R 022175 A 022175 /TERMINATOR.
013663 / OUTPUT MESSAGE BUFFERS.
013664 022372 R 022373 R ATBSF .DSA TPOK /PRINT-OUT IF FIX SUCCESSFUL, MODE 3.
013665 022373 R 022322 A TPOK 025322.
013666 022374 R 022229 A 0
013667 022375 R 422233 A .ASCII "FIXED IN CORE."
013668 022376 R 422212 A .ASCII "FIXED IN CORE."
013669 022377 R 222231 A .ASCII "FIXED IN CORE."
013670 02402 R 622226 A .ASCII "FIXED IN CORE."
013671 02401 R 726450 A .ASCII "FIXED IN CORE."
013672 02402 R 522232 A .ASCII "FIXED IN CORE."
013673 / OUTPUT MESSAGE BUFFERS.
013674 02403 R 022429 R ATBSF .DSA TP201 /PRINT-OUT IF "TASK" NOT IN SYSTEM, MODE 3.
013675 02404 R 022201 A TP201 027002.
013676 02405 R 022222 A 0
013677 02406 R 422272 A .ASCII "NOT FIXED - NOT IN SYSTEM."
013678 02407 R 422212 A .ASCII "NOT FIXED - NOT IN SYSTEM."
013679 02408 R 444610 A .ASCII "NOT FIXED - NOT IN SYSTEM."
013680 02409 R 522210 A .ASCII "NOT FIXED - NOT IN SYSTEM."
013681 02410 R 522211 A .ASCII "NOT FIXED - NOT IN SYSTEM."
013682 02412 R 422214 A .ASCII "NOT FIXED - NOT IN SYSTEM."
013683 02412 R 542120 A .ASCII "NOT FIXED - NOT IN SYSTEM."
013684 02412 R 622226 A .ASCII "NOT FIXED - NOT IN SYSTEM."
013685 02416 R 544672 A .ASCII "NOT FIXED - NOT IN SYSTEM."
013686 02417 R 442632 A .ASCII "NOT FIXED - NOT IN SYSTEM."
013687 02420 R 273020 A .ASCII "NOT FIXED - NOT IN SYSTEM."
DPP.02 MCR FUNCTION "DUAL PROCESSOR PATCHES" (DPP),

02421 R 002220 A
02422 R 002423 R AD222 .DSA TP222 /PRINT-OUT IF "TASK ACTIVE" ERROR, MODE 3.
02423 R 027220 A TP222 007222
02424 R 002222 A 0
02425 R 472372 A .ASCII "NOT FIXED - ACTIVE (FUBAR),"<15>
02426 R 422214 A
02427 R 446610 A
02428 R 542100 A
02429 R 265213 A
02430 R 141652 A
02431 R 446550 A
02432 R 522120 A
02433 R 446530 A
02434 R 247644 A
02435 R 265341 A
02436 R 522232 A

02441 R 222442 R AD244 .DSA TP244 /PRINT-OUT IF "TASK DISABLED" ERROR, MODE 3.
02442 R 213322 A TP244 010222
02443 R 022442 R 0
02444 R 472372 A . ASCII "NOT FIXED - DISABLED (FUBAR),"<15>
02445 R 422214 A
02446 R 446610 A
02447 R 542100 A
02448 R 265213 A
02449 R 446646 A
02450 R 255251 A
02451 R 444513 A
02452 R 222128 A
02453 R 652624 A
02454 R 426445 A
02455 R 127232 A

02461 R 204461 R AD267 .DSA TP267 /PRINT-OUT IF "TASK" ALREADY FIXED, MODE 3.
02462 R 025722 A TP267 005002
02463 R 202722 A 0
02464 R 432233 A .ASCII "FIXED PREVIOUSLY,"<15>
02465 R 432610 A
02466 R 224122 A
02467 R 446513 A
02468 R 551633 A
02469 R 56341 A
02470 R 522320 A

02473 R 022447 R AD218 .DSA TP218 /PRINT-OUT IF PARTITION OCCUPIED, MODE 3.
02474 R 212722 A TP218 002802
02475 R 027220 A 0
02476 R 472372 A .ASCII "NOT FIXED - PARTITION OCCUPIED,"<15>
02477 R 446610 A
02478 R 542102 A
02479 R 446410 A
02480 R 446410 A
*** MCR FUNCTION "DUAL PROCESSOR PATCHES" (DPP).

/ AD777 DSA TP777 PRINT-OUT IF NOT ENOUGH NODES, MODE 3.
/ TP777 007002
/ ASCII "NOT FIXED - NODE POOL EMPTY."<15>

/ ADOTH DSA TPOTH PRINT-OUT IF ERROR NOT DETERMINED, MODE 3.
/ TPOTH 005002
/ ASCII "NOT FIXED, SNAFU!"<15>

/ ADSH1 DSA NOSH1
/ NOSH1 011002
/ ASCII "PARTITION SHARE IS IMPROPERLY DEFINED."<15>
PAGE II
dpp;02

*** MGR FUNCTION "DUAL PROCESSOR PATCHES" (DPP).***

```
00564 R 454533 A
00565 R 422130 A
00566 R 644634 A
00567 R 426120 A
00570 R 608482 A

00614 R 227615 R
00615 R 012722 A
00616 R 012030 A
00617 R 512473 A
00620 R 226643 A
00621 R 425332 A
00622 R 321220 A
00623 R 425444 A
00624 R 241212 A
00625 R 426344 A
00626 R 046536 A
00627 R 4222332 A
00630 R 644612 A
00631 R 421612 A
00632 R 647644 A
00633 R 222511 A
00634 R 242552 A
00635 R 475352 A
00636 R 357372 A
00637 R 272322 A
00640 R 222222 A

00641 R 025442 R
00642 R 012202 A
00643 R 022222 A
00644 R 416310 A
00645 R 546534 A
```

ASCII "PARTITION SHARE HAS NOT BEEN DEFINED."<15>

ASCII "CHEM. ENGR. DEPT. DUAL PROCESSOR SYSTEM."<15>
MONITOR ENTRY POINTS FOR SYSTEM SUBROUTINES USED.

/ OTHER MONITOR ADDRESS POINTERS.

/ OTHER CONSTANTS.

/ STORAGE CELLS.
MCR FUNCTION "DUAL PROCESSOR PATCHES" (DPP).

END DPP

THAT'S ALL FRIENDS!
MCR FUNCTION: PROTECT REAL TIME OUTPUT  03 FEB 73  P. BARTRAM

TASK NAME "...PRO" TO REASSIGN A PROTECTION CODE TO A REAL TIME OUTPUT DEVICE.

THE FIRST LINE OF THE COMMAND INPUT FOR THE MCR FUNCTION IS READ BY THE RESIDENT MCR TASK ("...MCR"). FOR THE "PROTECT" FUNCTION, THERE IS ONLY ONE LINE OF COMMAND INPUT, AND ITS SYNTAX IS AS FOLLOWS:

SYNTAX: 
"PRO"<BREAK CHAR> <$<DEVN><BREAK CHAR>><NTAS><SPACE><OTAS><CR OR ALT>

<DEVN> = NON-BREAK CHARACTER, BREAK CHAR> = " " OR "", (<BREAK CHAR> JUST PRIOR TO<br>
<NTAS> MUST BE " "),
<DEVN> = REAL TIME OUTPUT DEVICE NUMBER:
NUMBER > 0 - DIGITAL TO ANALOG CONVERTER,
NUMBER <= 0 - DIGITAL OUTPUT GROUP.
<SPACE> = " ",
<NTAS> = FIRST THREE LETTERS OF THE NEW TASK NAME, USED FOR THE PROTECT CODE,
<OTAS> = OLD PROTECT CODE - THREE CHARACTERS,
<CR> = CARRIAGE RETURN.
<AM> = ALTMODE.
<NULL> = THE EMPTY SET (FOR < CR OR ALT>, USE " ") IF MORE OF THE COMMAND LINE IS TO FOLLOW OTHERWISE, USE NOTHING; <CR> OR <AM> WOULD FOLLOW INSTEAD.
$ = "ANY NUMBER OF, INCLUDING ZERO" (THOUGH A SYNTAX ERROR WILL RESULT FROM OMITTING NECESSARY PARTS OF THE COMMAND LINE WHICH MAY BE REPEATED IF DESIRED).

THE RESIDENT MCR READS A LINE, FETCHES THE FIRST THREE CHARACTERS TO FORM THE MCR FUNCTION TASK NAME ("...PRO"), FLUSHES CHARACTERS THROUGH TO THE FIRST BREAK CHARACTER, REQUESTS "...PRO", AND EXITS. THE TASK "...PRO" PROCESSES THE REMAINDER OF THE LINE, AND IF THE RESULT IS VALID, CHANGES THE PROTECT CODE FOR THE SPECIFIED REAL TIME OUTPUT DEVICE. ONLY THOSE TASKS WHOSE FIRST THREE LETTERS OF THEIR TASK NAMES ARE IDENTICAL (SIXST) TO THE PROTECT CODE MAY OUTPUT THROUGH THAT DEVICE. ALL TASKS MAY OUTPUT THROUGH DEVICES WITH THE PROTECT CODE "N-A".

THIS MCR FUNCTION FOLLOWS THE STANDARD MCR FUNCTION TERMINATION CONVENTION.

THE NUMBER OF DIGITAL TO ANALOG CONVERTERS IS CURRENTLY 8.

THE NUMBER OF DIGITAL OUTPUT GROUPS IS CURRENTLY 3.
THE BASE ADDRESS (DEVICE NUMBER 0) OF THE PROTECT TABLE IS CURRENTLY 13317.
**PRO.01**  ***MCR FUNCTION PROTECT REAL TIME OUTPUT.***

`TITLE ***MCR FUNCTION PROTECT REAL TIME OUTPUT.`

```
70080 R 750001 A PRO
70300 R 750002 A CLC
70600 R 750003 A DGT
70900 R 750004 A DAC
71200 R 750005 A DAC
71500 R 750006 A DAC
71800 R 750007 A DAC
72100 R 750008 A DAC
72400 R 750009 A DAC
72700 R 750010 A DAC
73000 R 750011 A DAC
73300 R 750012 A DAC
73600 R 750013 A DAC
73900 R 750014 A DAC
74200 R 750015 A DAC
74500 R 750016 A DAC
74800 R 750017 A DAC
75100 R 750018 A DAC
75400 R 750019 A DAC
75700 R 750020 A DAC
76000 R 750021 A DAC
76300 R 750022 A DAC
76600 R 750023 A DAC
76900 R 750024 A DAC
77200 R 750025 A DAC
77500 R 750026 A DAC
77800 R 750027 A DAC
78100 R 750028 A DAC
78400 R 750029 A DAC
78700 R 750030 A DAC
79000 R 750031 A DAC
79300 R 750032 A DAC
79600 R 750033 A DAC
79900 R 750034 A DAC
80200 R 750035 A DAC
80500 R 750036 A DAC
80800 R 750037 A DAC
81100 R 750038 A DAC
81400 R 750039 A DAC
81700 R 750040 A DAC
82000 R 750041 A DAC
82300 R 750042 A DAC
82600 R 750043 A DAC
82900 R 750044 A DAC
83200 R 750045 A DAC
83500 R 750046 A DAC
83800 R 750047 A DAC
84100 R 750048 A DAC
84400 R 750049 A DAC
84700 R 750050 A DAC
85000 R 750051 A DAC
85300 R 750052 A DAC
85600 R 750053 A DAC
85900 R 750054 A DAC
86200 R 750055 A DAC
86500 R 750056 A DAC
86800 R 750057 A DAC
87100 R 750058 A DAC
87400 R 750059 A DAC
87700 R 750060 A DAC
88000 R 750061 A DAC
88300 R 750062 A DAC
88600 R 750063 A DAC
88900 R 750064 A DAC
89200 R 750065 A DAC
89500 R 750066 A DAC
89800 R 750067 A DAC
90100 R 750068 A DAC
90400 R 750069 A DAC
90700 R 750070 A DAC
91000 R 750071 A DAC
91300 R 750072 A DAC
91600 R 750073 A DAC
91900 R 750074 A DAC
92200 R 750075 A DAC
92500 R 750076 A DAC
92800 R 750077 A DAC
93100 R 750078 A DAC
93400 R 750079 A DAC
93700 R 750080 A DAC
94000 R 750081 A DAC
94300 R 750082 A DAC
94600 R 750083 A DAC
94900 R 750084 A DAC
95200 R 750085 A DAC
95500 R 750086 A DAC
95800 R 750087 A DAC
96100 R 750088 A DAC
96400 R 750089 A DAC
96700 R 750090 A DAC
97000 R 750091 A DAC
97300 R 750092 A DAC
97600 R 750093 A DAC
97900 R 750094 A DAC
98200 R 750095 A DAC
98500 R 750096 A DAC
98800 R 750097 A DAC
99100 R 750098 A DAC
99400 R 750099 A DAC
99700 R 750100 A DAC
```

- **Title**: MCR Function Protect Real Time Output.
- The code provides instructions for initializing and processing digit counters, reading device numbers, and handling various input cases.
- It includes checks for validity of input, handling of positive and negative numbers, and error handling for out-of-range values.
- The code is designed to protect the real-time output by ensuring the correct processing of input data.
112 0057 222226 R  PRO.31 MCR FUNCTION PROTECT REAL TIME OUTPUT.
113 0060 722222 A  /NO; GET THE DEVICE NUMBER.
114 0061 722222 A  /XX = "JMP DGT3" IF DEV. NO. > 0; "TCA" IF < 0.
115 0062 722222 A  /STORE COMPLEMENTED DEVICE NUMBER.
116 0063 722222 A  /IS THIS NEGATIVE DEVICE NUMBER IN RANGE?
117 0064 622222 A  /YES, CONTINUE.
118 0065 222222 A  DGT5  /NO, RANGE ERROR, EXIT WITH DIAGNOSTIC.
119 0066 222222 A  CAL  /MESSAGE, CALLING THE MCR DISPATCHER
120 0067 222222 A  CAL  /UPON EXIT.
121 222222 A  CAL  /EXIT.
122 0071 722222 A  DGT3  /POSITIVE DEVICE NUMBER. IS IT IN RANGE?
123 0072 722222 A  DGT5  /NO, RANGE ERROR, EXIT WITH DIAGNOSTIC.
124 0073 222222 A  DGT4  /YES, CONTINUE, THE LAST CHARACTER WAS A SPACE
125 0074 222222 A  DGT4  /OR A COMMA. WAS IT A COMMA?
126 0075 622222 A  JMP  /YES, INITIALIZE FOR ANOTHER NUMBER, AND GET IT.
127 0076 222222 A  JMS  /NO, IT IS A SPACE, THE NEXT THREE CHARACTERS
128 0077 222222 A  JMP  /SHOULD BE THE NEW CODE, GET IT.
129 0078 122222 A  JMS  /GET THE NEXT CHARACTER.
130 0079 222222 A  DGT4  /IS IT A SPACE? (IT SHOULD BE,
131 0080 222222 A  DGT4  /OR A COMMA, WAS IT A COMMA?
132 0081 222222 A  DGT4  /YES. INITIALIZE FOR ANOTHER NUM8ER, AND GET IT,
133 0082 222222 A  JMS  /REPLACE THE OLO PROTECT CODE BY THE NEW
134 0083 222222 A  DGT4  /YES, NEXT THREE CHARACTERS ARE THE OLD CODE.
135 0084 222222 A  DGT4  /YES, NEXT THREE CHARACTERS ARE THE OL CODE.
136 0085 722222 A  DGT5  /GET IT;
137 0086 722222 A  DGT5  /GET THE TABLE ADDRESS BY ADDING THE BASE
138 0087 222222 A  JMS  /ADDRESS TO THE DEVICE NUMBER (LAST ENTERED
139 0088 222222 A  JMS  /HANDED FIRST).
140 0089 222222 A  JMS  /GET THE OLD CODE. IS IT THE SAME AS THAT?
141 0090 222222 A  JMS  /YES, CONTINUE.
142 0091 222222 A  DGT4  /NO, EXIT TO THE MCR DISPATCHER, GIVING
143 0092 222222 A  DGT4  /THE DIAGNOSTIC FOR AN INCORRECT OLD
144 0093 222222 A  DGT4  /PROTECT CODE.
145 0094 222222 A  DGT4  /MORE DEVICE NUMBERS FOLLOW, RESET THE
146 0095 222222 A  DGT4  /DIGIT COUNTER AND UP THE POINTER FOR THE
147 0096 222222 A  DGT4  /THE BEGINNING TO GET THE NEXT NUMBER.
148 0097 222222 A  DGT4  /REPLACE THE OLD PROTECT CODE BY THE NEW
149 0098 222222 A  DGT4  /TASK NAME PROTECT CODE.
150 0099 222222 A  DGT4  /DECREASE THE DEVICE NUMBER POINTER.
151 0100 222222 A  DGT4  /HAS THIS THE LAST DEVICE NUMBER?
152 0101 222222 A  DGT4  /YES, FINISHED WITH THIS REASSIGNMENT.
153 0102 222222 A  DGT4  /NO, THERE ARE MORE DEVICES FOR THIS SAME
154 0103 222222 A  DGT4  /REASSIGNMENT, DECREASE THE DEVICE NUMBER
155 0104 222222 A  DGT4  /POINTER AND LOOP BACK FOR THE REST.
156 0105 222222 A  DGT4  /YES, GET THE NEXT COMMAND LINE CHARACTER.
157 0106 222222 A  DGT4  /IS IT A CARRIAGE RETURN?
158 0107 222222 A  DGT4  /YES, EXIT TO THE MCR DISPATCHER.
00148 R 547214 R SAD ALT /NO, IS IT AN ALTMODE?
00149 R 542213 R JMP NOMCR /YES, EXIT WITHOUT CALLING THE MCR DISPATCHER.
00150 R 602208 R SAD SLASH /NO, IS IT A SLASH?
00151 R 602200 R JMP PRO /YES, START OVER AGAIN.
00152 R 602201 R JMP SYNER /NO, NONE OF THESE, EXIT WITH SYNTAX DIAGNOSTIC.
00153 R 742217 R DAC TEMPI /NO, LEGAL INPUT, STORE TEMPORARILY.
00154 R 122225 R JMP* FAC /YES, START OVER AGAIN.
00155 R 633030 R JMP PRO /YES, SYNTAX ERROR. EXIT WITH SYNTAX DIAGNOSTIC.
00156 R 633341 R JMP SYNER /YES, SYNTAX ERROR. EXIT WITH DIAGNOSTIC.
00157 R 543214 R SAD ALT /NO, IS IT AN ALTMODE?
00158 R 602213 R SAD CR /IS IT A CARRIAGE RETURN?
00159 R 602214 R SAD ALT /NO, IS IT AN ALTMODE?
00160 R 602215 R SAD CR /IS IT A CARRIAGE RETURN?
### MCR Function Protect Real Time Output

```
218   00212 R 000040 A BLANK 45 /* ",*/
219   00211 R 000054 A COMMA 54 /*","*/
220   00212 R 000055 A WSIGN 55 /* ",*/
221   00213 R 000057 A SLASH 37 /* ",*/
222   00214 R 000179 A ALT 179 /*ALTMODE*/
223   / STORAGE.
224   / CAL PARAMETER BLOCKS.
225   /
226   00215 R 000030 A DGT 0 /*DIGIT COUNTER FOR DEVICE NUMBERS*/
227   00216 R 000070 A NUMBR 0 /*ADDRESS POINTER FOR THE DEVICE NUMBER ARRAY*/
228   00217 R 000070 A TEMPP 0 /*TEMPORARY STORAGE*/
229   00220 R 000070 A TEMPP 0 /*TEMPORARY STORAGE*/
230   00221 R 000070 A EV 0 /*EVENT VARIABLE FOR CAL DIRECTIVES*/
231   00222 R 000070 A NEWCD 0 /*NEW PROTECT CODE READ IN*/
232   00223 R 000070 A OLDCD 0 /*OLD PROTECT CODE AS READ IN*/
233   00224 R 0 A NUMBK «BLOCK 40 /*DEVICE NUMBERS READ IN*/
234   /
235   /
236   00226 R 000070 A HFT 20 /*WAIT FOR CPB*/
237   00226 R 000070 A R.0SA EV
238   /
239   00226 R 000070 A SYNTX 2700 /*WRITE SYNTAX ERROR DIAGNOSTIC CPB*/
240   00227 R 000070 A SYNTX 2700 /*WRITE SYNTAX ERROR DIAGNOSTIC CPB*/
241   00228 R 000070 A E 2 /*EVENT BLOCK*/
242   00229 R 000070 A E 2 /*EVENT BLOCK*/
243   00230 R 000070 A MSYNT /*OUTPUT MESSAGES*/
244   00231 R 000070 A MSYNT /*OUTPUT MESSAGES*/
245   00232 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
246   00233 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
247   00234 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
248   00235 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
249   00236 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
250   00237 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
251   00238 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
252   00239 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
253   00240 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
254   00241 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
255   00242 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
256   00243 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
257   00244 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
258   00245 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
259   00246 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
260   00247 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
261   00248 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
262   00249 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
263   00250 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
264   00251 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
265   00252 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
266   00253 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
267   00254 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
268   00255 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
269   00256 R 000070 A BADCD 2700 /*WRITE INCORRECT PROTECT CODE DIAGNOSTIC CPB*/
```

PAGE 7  PRO.01  *** HCR FUNCTION PROTECT REAL TIME OUTPUT,

*»• M C f i  function  pr o t e c t  r e a l  t i m e  output,  

270  271  272  273  00316 R 726644 A
00317 R 543352 A
00320 R 442660 A
00321 R 202132 A
00322 R 251256 A
00323 R 511341 A
00324 R 502220 A

/  RANGE  007802  /OUTPUT MESSAGE FOR RANGE ERRORS,

. ASCII "PRO-DEVICE NO. OUT-OF-RANGE."<15>

274  275  276  277  00333 R 064202 A
00334 R 202000 A
00335 R 502451 A
00336 R 726612 A
00337 R 426551 A
00338 R 141612 A
00339 R 203391 A
00340 R 727172 A
00341 R 475552 A
00342 R 426336 A
00343 R 425372 A
00344 R 245344 A
00345 R 43125 A
00346 R 620422 A

/  HBCD  011002  /OUTPUT MESSAGE FOR INCORRECT FORMER

. ASCII "PRO-TASK PROTECT CODE IS INCORRECT."<15>

278  279  00365 R 20365 A

SIZE=20365  NO ERROR LINES
MCR FUNCTION -- LIST PROTECT TABLE (LPR)  03 FEB 73  P. BARTRAM

TASK NAME "...LPR" TO LIST THE ENTRIES IN THE TABLE FOR PROTECTION
OF DIGITAL TO ANALOG CONVERTERS AND DIGITAL OUTPUT GROUPS.

THE FIRST LINE OF THE COMMAND INPUT FOR ANY MCR FUNCTION IS READ BY
THE RESIDENT MCR TASK ("...MCR"), FOR THE LIST PROTECT TABLE --
"LPR" -- FUNCTION, THERE IS ONLY ONE LINE OF COMMAND INPUT, AND
ITS SYNTAX IS AS FOLLOWS:

SYNTAX = "LPR"{<NBC>{<CR>/<AM>}}

<NBC> = NON-BREAK CHARACTER
<CR> = CARRIAGE RETURN
<AM> = ALTMODE
5 -- "ANY NUMBER OF, INCLUDING ZERO"

THE RESIDENT MCR READS A LINE; FETCHES THE FIRST THREE CHARACTERS
TO FORM THE MCR FUNCTION TASK NAME ("...LPR"), FLUSHES CHARACTERS
THROUGH THE FIRST BREAK CHARACTER (" " OR ";"), CARRIAGE
RETURN, OR ALTMODE; REQUESTS "...LPR" AND EXITS. THE TASK
"...LPR" PROCESSES THE REMAINDER OF THE LINE (IF ANY REMAINS);
AND LISTS THE PROTECTION TABLE ON LUN 3, TABLE ENTRY "N-A"
INDICATES NO PROTECTION CODE HAS BEEN ASSIGNED; OTHER ENTRIES
CONTAIN THE FIRST THREE LETTERS OF THE TASK NAMES WHICH MAY
SET OUTPUTS THROUGH THAT DEVICE.

THIS MCR FUNCTION Follows THE TERMINATION CONVENTION OF ALL OTHER
MCR FUNCTIONS.
*** MCR FUNCTION 'LPR' -- LIST PROTECT TABLE.

** CURRENTLY, THERE ARE EIGHT DIGITAL TO ANALOG CONVERTERS.
** CURRENTLY, THERE ARE THREE DIGITAL OUTPUT GROUPS.
** THE BASE ADDRESS OF THE PROTECT TABLE (ENTRY FOR DEVICE 0, THE
** HIGHEST NUMBERED DIGITAL OUTPUT GROUP) IS 13317.

LPR CAL ATTY /* ATTACH LUN 3 FOR OUTPUT.
LPR CAL HEADR /* WRITE THE HEADER FOR THE TABLE LISTING.
LPR CAL WFR /* WAIT FOR COMPLETION.
LPR CAL HDAC /* WRITE THE HEADER FOR THE DIGITAL TO ANALOG
LPR CAL WFR /* CONVERTER SECTION, WAIT FOR COMPLETION.
LPR CAL DVHDR /* WRITE THE COLUMN LABELS.
LPR CAL WFR
LPR CAL MCRRI /* HAS *C BEEN ENTERED THROUGH THE MCR TTY?
LPR CAL MCRRI /* HAS *C BEEN ENTERED THROUGH THE MCR TTY?
LPR CAL MCRRI /* HAS *C BEEN ENTERED THROUGH THE MCR TTY?

LPR CAL ATTY /* ATTACH LUN 3 FOR OUTPUT.
LPR CAL HEADR /* WRITE THE HEADER FOR THE TABLE LISTING.
LPR CAL WFR /* WAIT FOR COMPLETION.
LPR CAL HDAC /* WRITE THE HEADER FOR THE DIGITAL TO ANALOG
LPR CAL WFR /* CONVERTER SECTION, WAIT FOR COMPLETION.
LPR CAL DVHDR /* WRITE THE COLUMN LABELS.
LPR CAL WFR
LPR CAL MCRRI /* HAS *C BEEN ENTERED THROUGH THE MCR TTY?

LPR CAL ATTY /* ATTACH LUN 3 FOR OUTPUT.
LPR CAL HEADR /* WRITE THE HEADER FOR THE TABLE LISTING.
LPR CAL WFR /* WAIT FOR COMPLETION.
LPR CAL HDAC /* WRITE THE HEADER FOR THE DIGITAL TO ANALOG
LPR CAL WFR /* CONVERTER SECTION, WAIT FOR COMPLETION.
LPR CAL DVHDR /* WRITE THE COLUMN LABELS.
LPR CAL WFR
LPR CAL MCRRI /* HAS *C BEEN ENTERED THROUGH THE MCR TTY?

LPR CAL ATTY /* ATTACH LUN 3 FOR OUTPUT.
LPR CAL HEADR /* WRITE THE HEADER FOR THE TABLE LISTING.
LPR CAL WFR /* WAIT FOR COMPLETION.
LPR CAL HDAC /* WRITE THE HEADER FOR THE DIGITAL TO ANALOG
LPR CAL WFR /* CONVERTER SECTION, WAIT FOR COMPLETION.
LPR CAL DVHDR /* WRITE THE COLUMN LABELS.
LPR CAL WFR
LPR CAL MCRRI /* HAS *C BEEN ENTERED THROUGH THE MCR TTY?
LPR:01

*** MCR FUNCTION 'LPR' -- LIST PROTECT TABLE.

00051 R 760240 A LAW 40 /NO, SET THE TENS PLACE OF THE DEVICE NUMBER
00052 R 040245 R DAC NUMBR+1 / TO A BLANK.
00053 R 762260 A LAW 40 /HANDLE UNIT ZERO AS A SPECIAL CASE.
00054 R 042246 R DAC NUMBR+2 /SET THE DEVICE NUMBER TO ZERO.
00055 R 222352 R LAC* LPROT /GET THE PROTECT CODE FOR DEVICE ZERO.
00056 R 00356 R JMS DECO /PUT THE CHARACTERS IN THE BUFFER.
00057 R 222229 R CAL LINE /PRINT THE LINE FOR DEVICE ZERO.
00058 R 022172 R CAL WFR
00059 R 222363 R LAC* MCRRI /HAS 'C BEEN ENTERED THROUGH THE MCR TTY?
00060 R 741103 A SPA /NO, SET THE MINUS SIGN FOR THE DEVICE NUMBER.
00061 R 222352 R LAC* /GET THE PROTECT CODE FOR DEVICE ZERO.
00062 R 222363 R LAC* MCRRI /HAS 'C BEEN ENTERED THROUGH THE MCR TTY?
00063 R 220365 R L00P2 LAC PNTR2 /GET THE NEXT ADDRESS.
00064 R 742123 R JMS DECO /ENTRY FOR CONVERTING SIXBIT TO IMAGE ASCII.
00065 R 620153 R JMP ABORT /YES, EXIT WITH MESSAGE TO THE MCR DISPATCHER.
00066 R 760255 A LAW 55 /NO, SET THE MINUS SIGN FOR THE DEVICE NUMBER.
00067 R 242244 R OAC NUMBR*1 /SET THE TENS PLACE TO ONE.
00068 R 220365 R L00P2 LAC PNTR2 /GET THE NEXT ADDRESS.
00069 R 742033 A I  AC /ENTRY FOR CONVERTING SIXBIT TO IMAGE ASCII.
00070 R 652223 A LMO /PLACE THE SIXBIT CODE IN THE MO TO BE SHIFTED
00071 R 641606 A EACILLS 6 /CHARACTER BY CHARACTER INTO THE AC.
00072 R 723743 A AAC -43 /IS THE FIRST CHARACTER A LETTER?
00073 R 741123 R SPA
00074 R 742240 A LAW 40 /ENTRY FOR CONVERTING SIXBIT TO IMAGE ASCII.
00075 R 652033 A LMO /PLACE THE SIXBIT CODE IN THE MO TO BE SHIFTED
00076 R 641626 A EACILLS 6 /SHIFT THE SECOND CHARACTER INTO THE AC.
00077 R 723740 A AAC -40 /LETTER?
00078 R 741123 R SPA /ENTRY FOR CONVERTING SIXBIT TO IMAGE ASCII.
00079 R 652223 A LMO /PLACE THE SIXBIT CODE IN THE MO TO BE SHIFTED
00080 R 641606 A EACILLS 6 /RESTORE THE 4B SUBTRACTED EARLIER.
00081 R 242253 R DAC TASKN /STORE THE IMAGE ASCII CHARACTER IN THE OUTPUT LINE.
00082 R 641626 A EACILLS 6 /ENTRY FOR CONVERTING SIXBIT TO IMAGE ASCII.
00083 R 723740 A AAC -40 /LETTER?
00084 R 741123 R SPA /ENTRY FOR CONVERTING SIXBIT TO IMAGE ASCII.
00085 R 652223 A LMO /PLACE THE SIXBIT CODE IN THE MO TO BE SHIFTED
00086 R 641606 A EACILLS 6 /CHARACTER BY CHARACTER INTO THE AC.
00087 R 723743 A AAC -43 /IS THE FIRST CHARACTER A LETTER?
00088 R 741123 R SPA
00089 R 742240 A LAW 40 /ENTRY FOR CONVERTING SIXBIT TO IMAGE ASCII.
00090 R 652033 A LMO /PLACE THE SIXBIT CODE IN THE MO TO BE SHIFTED
00091 R 641626 A EACILLS 6 /RESTORE THE 4B SUBTRACTED EARLIER.
00092 R 723740 A AAC -40 /LETTER?
00093 R 741123 R SPA /ENTRY FOR CONVERTING SIXBIT TO IMAGE ASCII.
00094 R 652223 A LMO /PLACE THE SIXBIT CODE IN THE MO TO BE SHIFTED
00095 R 641606 A EACILLS 6 /SHIFT THE SECOND CHARACTER INTO THE AC.
00096 R 723740 A AAC -40 /LETTER?
00097 R 741123 R SPA /ENTRY FOR CONVERTING SIXBIT TO IMAGE ASCII.
00098 R 652223 A LMO /PLACE THE SIXBIT CODE IN THE MO TO BE SHIFTED
00099 R 641606 A EACILLS 6 /CHARACTER BY CHARACTER INTO THE AC.
00100 R 723743 A AAC -43 /IS THE FIRST CHARACTER A LETTER?
00101 R 741123 R SPA
00102 R 742240 A LAW 40 /ENTRY FOR CONVERTING SIXBIT TO IMAGE ASCII.
00103 R 652033 A LMO /PLACE THE SIXBIT CODE IN THE MO TO BE SHIFTED
00104 R 641626 A EACILLS 6 /RESTORE THE 4B SUBTRACTED EARLIER.
00105 R 723740 A AAC -40 /LETTER?
00106 R 741123 R SPA /ENTRY FOR CONVERTING SIXBIT TO IMAGE ASCII.
00107 R 652223 A LMO /PLACE THE SIXBIT CODE IN THE MO TO BE SHIFTED
00108 R 641606 A EACILLS 6 /SHIFT THE SECOND CHARACTER INTO THE AC.
00109 R 723740 A AAC -40 /LETTER?
00110 R 741123 R SPA /ENTRY FOR CONVERTING SIXBIT TO IMAGE ASCII.
00111 R 652223 A LMO /PLACE THE SIXBIT CODE IN THE MO TO BE SHIFTED
00112 R 641606 A EACILLS 6 /CHARACTER BY CHARACTER INTO THE AC.
00113 R 723743 A AAC -43 /IS THE FIRST CHARACTER A LETTER?
00114 R 741123 R SPA
00115 R 742240 A LAW 40 /ENTRY FOR CONVERTING SIXBIT TO IMAGE ASCII.
00116 R 652033 A LMO /PLACE THE SIXBIT CODE IN THE MO TO BE SHIFTED
00117 R 641626 A EACILLS 6 /RESTORE THE 4B SUBTRACTED EARLIER.
00118 R 723740 A AAC -40 /LETTER?
00119 R 741123 R SPA /ENTRY FOR CONVERTING SIXBIT TO IMAGE ASCII.
00120 R 652223 A LMO /PLACE THE SIXBIT CODE IN THE MO TO BE SHIFTED
00121 R 641606 A EACILLS 6 /SHIFT THE SECOND CHARACTER INTO THE AC.
00122 R 723740 A AAC -40 /LETTER?
00123 R 741123 R SPA /ENTRY FOR CONVERTING SIXBIT TO IMAGE ASCII.
00124 R 652223 A LMO /PLACE THE SIXBIT CODE IN THE MO TO BE SHIFTED
00125 R 641606 A EACILLS 6 /CHARACTER BY CHARACTER INTO THE AC.
**MCR FUNCTION 'LPR' -- LIST PROTECT TABLE.**

00133 R 723100 A
AAC 100 /YES, ADD BIT 11.

00134 R 723840 A
AAC 48 /

00135 R 042054 R
DAC TASKN+1 /STORE THE OUTPUT LINE.

00136 R 641060 A
EAC:LLS 5 /GET THE THIRD CHARACTER INTO THE AC.

00137 R 723740 A
AAC -40 /LETTER?

00140 R 723100 A
AAC 100 /YES, ADD BIT 11.

00141 R 723170 A
AAC 40 /

00142 R 042055 R
DAC TASKN+2 /STORE THE OUTPUT LINE.

00144 R 620120 R
JMP DECQ /RETURN TO THE MAIN PROGRAM.

00145 R 123357 R
/ END JMS* FAC /GET THE LAST CHARACTER OF THE COMMAND LINE.

00146 R 542341 R
SAD CR /IS IT A CARRIAGE RETURN?

00147 R 602155 R
JMP END1 /YES, EXIT CALLING THE MCR DISPATCHER.

00148 R 602156 R
SAD ALT /NO, IS IT AN ALTMODE?

00149 R 602157 R
JMP END2 /YES, EXIT WITHOUT CALLING THE MCR DISPATCHER.

00150 R 620149 R
JMP END /NO, NEITHER. IT MUST NOT BE THE LAST CHAR.

00153 R 002322 R
/ ABORT CAL CNTLC /C HAS BEEN ENTERED ON THE MCR TTY. PRINT

00154 R 003172 R /
CAL HFR /MESSAGE THAT THIS FUNCTION HAS BEEN KILLED.

00155 R 003173 R
/ END1 CAL ROMCR /CARRIAGE RETURN. REQUEST TASK ".MCR"

00156 R 003174 R
CAL DTY /DISPATCHER, DETACH LUN 3, AND EXIT.

00157 R 003175 R
CAL EXIT /

00158 R 167366 R
END2 D2* MCRRI /ALTMODE. CLEAR THE MCR INHIBIT FLAG,

00159 R 022167 R
CAL DTY /DETACH LUN 3, AND EXIT.

00160 R 022168 R
CAL EXIT /

00161 R 002172 R
/ CAL PARAMETER BLOCKS,

00162 R 022169 R
/ CAL 10 /EXIT CPB.

00163 R 022170 R
EXIT 10 /EXIT CPB.

00164 R 022402 A
ATTY 2400 /ATTACH LUN 3 CPB.

00165 R 022403 A
0

00166 R 022404 A
3

00167 R 022502 A
DTTY 2500 /DETACH LUN 3 CPB.

00168 R 022503 A
0

00169 R 022504 A
3

00170 R 022505 A
0

00171 R 022506 A
3

00172 R 022522 A
WFR 20 /WAIT FOR CPB.

00173 R 003363 R
/ DSA EV

00174 R 022001 A
ROMCR 1 /REQUEST ".MCR" TASK CPB.

00175 R 022002 A
0

00176 R 022003 A
0

00177 R 152322 R
.6XBT ".MCR"

00202 R 022020 A
0

00203 R 022021 A
3

00204 R 022022 A
2

00201 R 022702 A
MEADR 2700 /PRINT CPB FOR THE FIRST LINE.
*** MCR FUNCTION 'LPR' -- LIST PROTECT TABLE.

PRINT CPB FOR THE DIGITAL TO ANALOG CONVERTER
/ SECTION HEADER.
PRINT CPB FOR LISTING COLUMN HEADINGS.
PRINT CPB FOR THE DIGITAL OUTPUT GROUPS.
/ SECTION HEADER.
PRINT CPB FOR A LINE OF THE TABLE.
PRINT ABORT MESSAGE CPB.
/ OUTPUT BUFFERS.
OUTPUT LINE: DEVICE NUMBER AND PROTECT CODE
/ SET BY THE PROGRAM.
FIRST LINE: TABLE HEADER.
PAGE 6

LPR:01

--- MCR FUNCTION 'LPR' -- LIST PROTECT TABLE.

243
00261 R 000000 A
00262 R 202450 A
00263 R 540630 A
00264 R 202511 A
00265 R 146612 A
00266 R 202372 A
00267 R 592240 A
00268 R 232511 A
00271 R 475646 A
00272 R 516233 A
00273 R 747232 A
00274 R 491332 A
00275 R 431332 A

244
00276 R 072202 A
00277 R 202250 A
00278 R 202111 A
00279 R 143522 A
00280 R 322331 A
00281 R 475250 A
00282 R 147232 A
00283 R 146270 A
00284 R 722726 A
00285 R 476332 A
00286 R 642644 A
00287 R 52132 A
00288 R 251432 A

245
00289 R 072202 A
00290 R 202250 A
00291 R 202111 A
00292 R 143522 A
00293 R 322331 A
00294 R 475250 A
00295 R 147232 A
00296 R 146270 A
00297 R 722726 A
00298 R 476332 A
00299 R 642644 A
00300 R 52132 A
00301 R 251432 A

246
00302 R 072202 A
00303 R 202250 A
00304 R 202111 A
00305 R 143522 A
00306 R 322331 A
00307 R 475250 A
00308 R 147232 A
00309 R 146270 A
00310 R 722726 A
00311 R 476332 A
00312 R 642644 A
00313 R 52132 A
00314 R 251432 A

247

--- DAC 007002 --- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DAC 007002 --- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.

--- ASCII "DIGITAL TO ANALOG CONVERTERS"<15> ---

--- DIGITAL TO ANALOG CONVERTER HEADER LINE.
*** MCR FUNCTION 'LPR' -- LIST PROTECT TABLE.

00342 R 522031 A
00343 R 420236 A
00344 R 526512 A
00345 R 052658 A
00346 R 220172 A
00347 R 247652 A
00350 R 522451 A
00351 R 522078 A

/ CONSTANTS AND STORAGE.

/ CLEAR AC INSTRUCTION -- EAE UNIT.

/ EAC=641220
00352 R 013317 A LPROT .DSA PROT / TABLE ADDRESS FOR DEVICE 0.
00353 R 013330 A DAFIN .DSA PROT-NDAC-1 / ADDRESS ONE ABOVE TOP OF TABLE.
00354 R 013314 A DFIN .DSA PROT-NDO / ADDRESS ONE BELOW BOTTOM OF TABLE.
00355 R 013331 A DAC10 .DSA PROT-12 / ADDRESS OF DAC UNIT NO. 10 IN TABLE.
00356 R 013325 A D010 .DSA PROT-12 / ADDRESS OF DIGITAL OUTPUT GROUP #10.
00357 R 027174 A FAC 02174 / MONITOR SUBROUTINE ENTRY1 CHAR. FETCH.
00358 R 027171 A MCR1 06171 / MCR INHIBIT FLAG IN THE MONITOR.
00359 R 027015 A CR 020170 / CARRIAGE RETURN (IMAGE ASCII).
00360 R 027175 A ALT 020173 / ALTMODE (IMAGE ASCII).
00361 R 027200 A EV 0 / EVENT VARIABLE.
00362 R 027200 A PNTR1 0 / ADDRESS POINTER STORAGE.
00363 R 027200 A PNTR2 0 / ADDRESS POINTER STORAGE.
00364 R 052203 A END LPR

SIZE=00366 NO ERROR LINES
/* EDIT #10 */
/* COPYRIGHT 1970, 1971 DIGITAL EQUIPMENT CORP., MAYNARD, MASS. */
/* MCR FUNCTION -- REMOVE 26 SEP 71 R. MCLEAN */
/* MODIFIED FOR THE OHIO STATE CHEMICAL ENGINEERING */
/* DEPARTMENT DUAL PROCESSOR 13 NOV 72 P. BARTRAM */
/* TASK NAME "...REM" TO REMOVE A TASK FROM THE SYSTEM TASK LIST. */
/* THE FIRST LINE OF THE COMMAND INPUT FOR ANY MCR */
/* FUNCTION IS READ BY THE RESIDENT MCR TASK (...MCR). */
/* FOR THE "REMOVE" FUNCTION, THERE IS ONLY ONE LINE OF */
/* COMMANDINPUT, AND IT'S SYNTAX IS AS FOLLOWS: */
/* SYNTAX = 'REM'$.<NBC><BREAK CHARACTER><TASK NAME> */
/* (CR)/(AM) */
/* <BREAK CHAR> = " /", "." */
/* <TASK NAME> = 1-6 ALPHANUMERIC CHARACTERS */
/* <CR> = CAR RTN */
/* <AM> = ALTMODE */
/* <NBC> = NON BREAK CHARACTER */
/* $ = " ANY NUMBER OF "INCLUDING ZERO " */
/* THE RESIDENT MCR READS A LINE, FETCHES THE */
/* FIRST THREE CHARACTERS TO FORM THE MCR FUNCTION TASK */
/* NAME (...REM), FLUSHES CHARACTERS THRU THE FIRST */
/* BREAK CHARACTER, REQUESTS "...REM", AND EXITS */
/* THE TASK "...REM" PROCESSES THE REMAINDER OF THE LINE */
/* AND IF THE REQUEST IS VALID, ISSUES AN APPROPRIATE "REMOVE" */
/* DIRECTIVE. */
/* IF THE COMMAND INPUT LINE IS TERMINATED BY A CAR RTN, */
/* THE RESIDENT MCR TASK IS REQUESTED, AND THE FUNCTION TASK */
/* EXITS. */
/* IF THE COMMAND INPUT LINE IS TERMINATED BY AN ALTMODE, THE */
/* FUNCTION TASK EXITS WITHOUT REQUESTING "...MCR", A "C TYPEIN */
/* IS THEN NECESSARY TO RE-ESTABLISH MCR DIALOGUE."
**TITLE**: MCR FUNCTION 'REMOVE'

```
00171 A MCRRI=171
00174 A FAC=174
00123 A SNAM=123
70521 A END=70521
70522 A INH=70522
00243 A POOL=243
02117 A NADD=107
02112 A NDEL=112
02111 A R1=111
02122 A R2=122
02242 A STKL=242
02253 A X1=19
02263 A P,Fw=10
00877 A REM LAW -7 /SET UP TO FETCH TASK NAME AND STORE
00823 R 042362 R DAC CNT /SIX CHARACTERS (ZERO RIGHT FILL)
00822 R 022462 R LAC (REMN=1)
00823 R 052423 R DAC* (X10)
01244 R 122424 R REMN1 JMS* (FAC) /FETCH A CHARACTER
01225 R 542425 R SAD (054) /IS IT A COMMA?
01226 R 602426 R JHP ERR1 /YES -- ERROR IN SYNTAX
01227 R 542426 R SAD (046) /NO -- BLANK?
01212 R 622423 R JHP ERR1 /YES -- ERROR IN SYNTAX
01211 R 542427 R SAD (015) /NO -- CAR RTN?
01212 R 622423 R JHP ENDCRA /YES -- END OF REQUEST
01213 R 542430 R SAD (175) /NO -- ALTMODE?
01214 R 622423 R JHP ENDCRA /YES -- END OF REQUEST
01215 R 652310 R DAC* X10 /NO -- STORE CHARACTER
01216 R 443326 R ISE CNT /LAST CHARACTER OF TASK NAME?
01217 R 602324 R JHP REMN1 /NO -- GET NEXT CHARACTER
01220 R 202361 R ERR1 LAC (MES2) /GET MESSAGE ADDRESS SYNTAX ERROR
01221 R 742333 R DAC TYPEP=4 /PUT IN TYPE REQUEST
01222 R 602352 R JHP ERRTY /REQUEST MCR AND RETURN
01223 R 042355 R ENDGRA DAC SVBKCH /SAVE CAR RTN OR ALTMODE
01224 R 16236 A REMN2 OZH* X10 /FILL REMAINING CHARACTERS WITH ZERO
01225 R 442356 R ISE CNT
01226 R 602224 R JMP REMN2
01227 R 202264 R LAC REMN=4 /FORM FIRST HALF OF TASK NAME
01232 R 642556 A LRS 6
01231 R 202253 R LAC REMN=3
01232 R 642556 A LRS 6
01233 R 252252 R LAC REMN=2
01234 R 742250 A SMA /IS THIS A NULL NAME?
01235 R 602250 R JMP ERR1 /YES EXIT WITH ERROR
01236 R 642254 A LLS 14
01237 R 042252 R DAC REMN=2 /STORE FIRST HALF OF WORD IN REMN
01240 R 202257 R DAC REMN=7 /FORM SECOND HALF OF TASK NAME
01241 R 642556 A LRS 6
```
```assembly
PAGE 3  REM 03S  *** MCR FUNCTION 'REMOVE'

93   02342 R 28288 R  LAC  REMNAM=6
94   02043 R 64056 A  LRS  6
95   02244 R 20228 R  LAC  REMNAM=5
96   02545 R 64251 A  LLS  1
97   02446 R 24223 R  DAC  REMNAM=3
98   02347 R 20243 R  LAC  (REMNAM=2) / PICK UP POINTER TO TASK NAME
99   02047 R 64243 R  DAC* (R2) / SAVE IT IN R2 FOR SNAM
100  02541 R 20243 R  LAC (STKL) / PICK UP POINTER TO SYSTEM TASK LIST HEAD
101  02052 R 64245 R  DAC* (R1) / SAVE IT ALSO
102  02353 R 12246 R  JMS* (SNAM) / LOOK FOR PROGRAM IN SYSTEM TASK LIST
103  02054 R 60316 R  JMP  ENTYA / TASK NOT IN SYSTEM ERROR
104  02255 R 24211 R  DAC  STLNOD / SAVE STKL NODE ADDRESS
105  02056 R 72327 A  AAC  4 / ADD 4 TO PICK UP FLAGS
106  02257 R 24227 R  DAC  FLAGS / SAVE POINTER TO FLAGS
107  02258 R 74238 A  IAC
108  02259 R 04221 R  DAC  PARBA / PARTITION BLOCK ADDRESS
109  02260 R 74232 A  IAC
110  02261 R 04236 R  DAC  CTN / SAVE DISK ADDRESS POINTER
111  02264 R 20247 R  LAC  (40002) / RAISE TO LEVEL 6 TO PREVENT EXIT
112  00265 R 70557 A  ISA / DURING CANCEL
113  02266 R 70552 A  JINH / INHIBIT INTERRUPTS WHILE MODIFYING FLAGS
114  00267 R 22027 R  LAC  FLAGS / PICK UP FLAGS
115
116 // DELETE ON EXIT NOT IMPLEMENTED SO EXIT WITH ERROR
117 // IF TASK ACTIVE
118 //
119 //
120 00070 R 74118 A  SPA
121 00071 R 60214 R  JMP  ACTIVE / MASK OFF DISARM AND REMOVE ON EXIT BITS
122 00272 R 50244 A  AND (577777) / YES SET REMOVE ON EXIT FLAGS
123 00273 R 24241 R  XOR (290000) / RAISE TO LEVEL 6 TO PREVENT EXIT
124 00274 R 70552 A  LEBF / ENABLE INTERRUPTS
125 00275 R 02227 R  DAC* FLAGS / RESTORE FLAGS
126 00276 R 50244 A  AND (040000) / MASK OFF FIXED IN CORE BITS
127 00277 R 74128 A  SNA / IS IT FIXED IN CORE?
128 00070 R 60213 R  JMP  NOTFIX / NO DON'T FREE PARTITION
129 00100 R 62213 R  LAC* NEXT FIVE LINES ARE MODIFICATIONS TO ORIGINAL.
130 00101 R 22227 R  LAC* FLAGS / GET THE FLAGS BACK.
131 00102 R 50244 A  AND (020000) / ARE THERE ENTRIES IN THE
132 00103 R 74220 A  SEA / AUTOMATIC INPUT TABLE?
133 00104 R 63307 R  JMP  CLEAN / YES, REMOVE THEM.
134 00105 R 22210 R  UNCLN LAC* PARBA / NO, CONTINUE.
135 00106 R 22210 R  LAC* PARBA / PICK UP THE PARTITION BLOCK POINTER
136 00107 R 23219 A  AAC P.FW / MOVE POINTER TO THE PARTITION BUSY FLAG
137 00108 R 64219 A  DAC PARBA / PICK UP THE FLAGS WORD
138 00109 R 24244 R  XOR (400200) / FREE PARTITION
139 00110 R 60319 R  DAC* PARBA
140 00111 R 03220 R  NOTFIX CAL  REMNAM / ISSUE CANCEL DIRECTIVE
```
REM.OS  *** MCR FUNCTION 'REMOVE' ***

00114 R 222211 R LAC STLNOD /NO -- DELETE NODE FROM STKL
00115 R 062435 R DAC (R1) /DELETE NODE
00116 R 120445 R JMS* (DEL) /DELETE NODE
00117 R 223227 R LAC* FLAGS /IS TASK BRFY
00120 R 723364 A DBK

00121 R 223236 R LAC* CNT /PICK UP DISK ADDRESS
00122 R 060435 R OAC® (Rl) /DELETE NOOE
00125 R 223236 R LAC* CNT /PICK UP DISK ADDRESS AGAIN
00126 R 737237 A AAC -10 /SUBTRACT 10 TO FIND BEGINNING OF STORAGE
00127 R 242234 R DAC CTNLV /SAVE IT IN CPB FOR DEALLOCATE
00128 R 222216 R CAL GETSIB /PICK UP THE SIZE OF THE STORAGE
00129 R 022235 R CAL WAITEV /CHECK TO SEE IF OK
00130 R 203234 R DAC CNT /PICK UP DISK ADDRESS AGAIN
00133 R 203234 R LAC EV /CHECK TO SEE IF OK
00134 R 622195 R JMP DSKEIV /NOT OK DISK ERROR
00136 R 222235 R CAL DEALOC /DEALLOCATE DISK SPACE
00137 R 223234 R LAC EV /CHECK TO SEE IF OK
00138 R 741172 A SPA
00139 R 622195 R JMP DSKEIV /NOT OK DISK ERROR
00142 R 223235 R LAC STLNOD /ADD NOOE TO EMPTY POOL
00143 R 664333 R DAC® (R2) /ADD NODE TO EMPTY POOL
00144 R 203242 R DAC® (POOL)
00145 R 203242 R DAC® (R1) /ADD NODE TO EMPTY POOL
00146 R 122491 R JMS (WADD) /ADD NODE TO EMPTY POOL
00147 R 602165 R JMP EXT11 /EXIT
00149 R 224242 R ERRTYA LAC® (MES3) /PRINT TASK NOT IN SYSTEM ERROR
00150 R 224233 R DAC TYPDCP®+4 /MAKE TYPE CPB REQUEST
00151 R 223227 R ERRTY CAL TYPDCP® /WAIT FOR ERROR MESSAGE TO BE PRINTED
00152 R 022235 R CAL WAITEV /WAIT FOR DEALLOCATE TO COMPLETE
00153 R 022235 R CAL WAITEV /WAIT FOR DEALLOCATE TO COMPLETE
00154 R 602167 R JMP EXT2 /EXIT
00155 R 222453 R DSKEIV LAC® (MES4)
00156 R 042333 R DAC TYPDCP®+4 /PUT MESSAGE IN BUFFER POINTER
00157 R 602162 R JMP ERRTY /AND TYPE MESSAGE

00160 R 200211 R LAC STLNOD /ADD NOOE TO EMPTY POOL
00161 R 062435 R DAC (R1) /DELETE NODE
00162 R 203324 A DBK /DELETE BREAK FROM LEVEL 6
00163 R 203454 R LAC (MES1) /DELETE NODE
00164 R 203324 A DBK /DELETE BREAK FROM LEVEL 6
00165 R 222305 R JMS* (DEL) /DELETE NODE
00166 R 542427 R SAD (15) /DELETE BREAK FROM LEVEL 6
00167 R 023173 R EXT CAL RECMCR /REQUEST MCR TASK
00168 R 942435 R SAD (175) /DELETE NODE
00169 R 023173 R EXT CAL RECMCR /REQUEST MCR TASK
00170 R 942435 R SAD (175) /DELETE NODE
00171 R 162455 R DEM® (MCRRRI) /CLEAR +C SWITCH
00172 R 00423 R  
/ CAL (10) /RETURN
/REOMCR 1 /CALL MCR DIRECTIVE
00173 R 02226 A  
REOMCR 1
00174 R 02226 A  
0
00175 R 565656 A  
"SIXBYT "...
00176 R 19322 A  
"SIXBYT "MCR"
00177 R 02225 A  
0
00220 R 02225 A  
REMNAM 4 /CANCEL DIRECTIVE
02201 R 02223 A  
EV /EVENT VARIABLE ADDRESS
02202 R 02223 A  
0 /TASK NAME (FIRST HALF)
02203 R 02223 A  
0 /TASK NAME (SECOND HALF)
02204 R 02223 A  
FLAGS 0
02205 R 02223 A  
PASA 0
02211 R 02220 A  
STLNO 0
02212 R 02160 A  
DEALOC 1600 /DEALLOCATE CPB
02213 R 02234 A  
EV /EVENT VARIABLE ADDRESS
02214 R 02201 A  
1 /LOGICAL UNIT NUMBER
02215 R 02222 R  
CNTRLT /CONTROL TABLE ADDRESS
02216 R 02223 A  
GETSI2 3000 /GET A WORD FROM THE DISK
02217 R 02234 A  
EV /EVENT VARIABLE ADDRESS
02218 R 02201 A  
1 /LUN NUMBER
02219 R 02222 R  
CNTRLU /CONTROL TABLE ADDRESS
02222 R 02222 R  
CNTRLT 0 /NUMBER OF WORDS TO BE DELETED
02223 R 02222 R  
CNTRLU 0 /UNIT NUMBER
02224 R 02220 A  
CNTRLA 0 /DISK ADDRESS
02225 R 02222 R  
CNTO CNTRLT /CORE ADDRESS FOR GET
02226 R 02201 A  
4 /WORD COUNT FOR GET
02227 R 02270 A  
TYPCPB 2700
02230 R 02223 A  
EV
02231 R 02203 A  
3
02232 R 02202 A  
2
02233 R 02201 A  
MES3
02234 R 02203 A  
EV 0
02235 R 02220 A  
WAITEV 20 /WAIT FOR
02236 R 02234 R  
EV /EVENT VARIABLE ADDRESS
02237 R 02502 A  
MES1 MES2-MES1/2*1000+2
02240 R 02200 A  
0
02241 R 512131 A  
.ASCII "REM-TASK ACTIVE"<15>
02242 R 526550 A  
02243 R 426471 A  
02244 R 326720 A  
02245 R 416511 A  
02246 R 153212 A  
02247 R 064220 A  
02250 R 02202 A  
/
REM OSM CR U N C T I O N ' R E M O V E ' 

245 0251 R 004229 A MES 2 MES 3-MES 2/2*1000+2
246 0252 R 005200 A 1
247 0253 R 512131 A ,ASCII "REM-SYNTAX ERR" <15>
248 0254 R 526646 A
249 0255 R 546352 A
250 0256 R 446449 A
251 0257 R 202132 A 0
252 0258 R 251232 A
253 0259 R 526652 A
254 0260 R 446471 A
255 0261 R 322234 A
256 0262 R 476574 A
257 0263 R 244634 A
258 0264 R 202473 A
259 0265 R 511550 A
260 0266 R 425521 A
261 0267 R 512260 A
262 0268 R 512441 A
263 0269 R 502200 A
264 /
265 /
266 /
267 03325 R 007229 A SVBKCH 0
268 0336 R 005200 A CNT 0
269 /********** ALL FOLLOWING CODE ADDED AS MODIFICATION.**
270 03207 R 202110 R CLEAN LAC* PARBA /GET THE PARTITION BLOCK ADDRESS.
271 03310 R 723324 A AAC +4 /GET THE ADDRESS OF THE PARTITION BASE ADDR.
272 03311 R 344216 R DAC PARB# /GET THE ADDRESS OF THE PARTITION LENGTH.
273 03312 R 723321 A AAC +1 /GET ADDRESS OF THE PARTITION LENGTH.
274 03313 R 424417 R DAC PART# /GET THE PARTITION BASE ADDRESS.
275 03314 R 224416 R LAC* PARB /STORE THE COMPLEMENT.
276 03315 R 742231 A TCA /COMPLEMENT.
277 03316 R 242146 R DAC PARB /GET THE COMPLEMENT OF THE PARTITION TOP
278 03317 R 222417 R LAC* PART /ADDRESS + 1 BY ADDING COMPLEMENTS OF BASE
279 03318 R 742231 A TCA /AND LENGTH.
280 03319 R 342416 R TAD PARB /CHECK FOR ENTRIES IN THE 1 KHZ TABLE.
281 03320 R 342417 R DAC PART /CHECK FOR ENTRIES IN THE 100 Hz TABLE.
282 03321 R 202132 A LAC TABL 1 /CHECK FOR ENTRIES IN THE 1 KHZ TABLE.
283 03322 R 202132 A LAC TABL 2 /CHECK FOR ENTRIES IN THE 100 Hz TABLE.
284 03323 R 202132 A LAC TABL 3 /CHECK FOR ENTRIES IN THE 10 Hz TABLE.
02331 R 608105 REM*** MCR FUNCTION 'REMOVE' *RETURN TO ORIGINAL PROGRAM.*
279 02332 R 615200 JMP UNCLN /BASE ADDRESS FOR 1 KHZ TABLE.
281 02333 R 615200 TABL1 /BASE ADDRESS FOR 100 HZ TABLE.
282 02334 R 615200 TABL2 /BASE ADDRESS FOR 10 HZ TABLE.
283 02335 R 742420 CHECK XX /ENTRY FOR CHECK & REMOVE TABLE ENTRY ROUTINE.
284 02336 R 642420 DAC PNTR1# /POINTER 1 HAS THE TABLE STARTING ADDRESS.
285 02337 R 725274 A AAC +4 /POINTER 2 HAS THE NEXT ENTRY ADDRESS FOR
286 02338 R 627421 DAC PNTR2# /THE TABLE.
287 02339 R 725222 INH /DISABLE API IN CASE TABLE IS TO BE CHANGED.
288 02340 R 040420 OAC PNTR1# /IS THE TABLE ENTRY ZERO?
289 02341 R 741200 DAC PNTR2# /POINTER 1 HAS THE TABLE STARTING ADDRESS!
290 02342 R 047421 DAC PNTR1# /POINTER 2 HAS THE NEXT ENTRY ADDRESS FOR
291 02343 R 741100 SPA /THE TABLE.
292 02344 R 705522 A AAC +4 /DISABLE API IN CASE TABLE IS TO BE CHANGED.
293 02345 R 602404 R OAC PNTR1# /IS THE TABLE ENTRY ZERO?
294 02346 R 540420 R SAD PNTR2# /POINTER 1 HAS THE TABLE STARTING ADDRESS.
295 02347 R 623394 A AAC +4 /POINTER 2 HAS THE NEXT ENTRY ADDRESS FOR
296 02348 R 227421 R CHK1 LAC PNTR1# /YES, STOP THE SEARCH.
297 02349 R 227421 R CHK1 LAC PNTR1# /NO, IS USER BUFFER ABOVE PARTITION BASE?
298 02350 R 741123 A SPA /USER ADR - PAR, BASE > 0 ?)
299 02351 R 340416 R TAD PAR8 /DISABLE API IN CASE TABLE IS TO BE CHANGED.
300 02352 R 741200 A SAC /POINTER 1 HAS THE TABLE STARTING ADDRESS!
301 02353 R 741200 A SAC /POINTER 2 HAS THE NEXT ENTRY ADDRESS FOR
302 02354 R 725222 A AAC +4 /THE TABLE.
303 02355 R 723204 A DAC PNTR1# /NO, USER TABLE IS NOT IN THIS PARTITION.
304 02356 R 642420 A DAC PNTR1# /YES, IS USER BUFFER BELOW PARTITION TOP?
305 02357 R 224240 A LAC PNTR1# /NO, USER TABLE IS NOT IN THIS PARTITION.
306 02358 R 340416 R TAD PAR8 /DISABLE API IN CASE TABLE IS TO BE CHANGED.
307 02359 R 741123 A SAC /POINTER 1 HAS THE TABLE STARTING ADDRESS!
308 02360 R 340416 R TAD PAR8 /POINTER 2 HAS THE NEXT ENTRY ADDRESS FOR
309 02361 R 725222 A AAC +4 /YES, THE USER BUFFER IS IN THIS PARTITION.
310 02362 R 723204 A DAC PNTR1# /NO, USER BUFFER IS OUTSIDE THIS
311 02363 R 642420 A DAC PNTR1# /PARTITION; DO NOT OVERWRITE THIS ENTRY.
312 02364 R 224240 A LAC PNTR1# /ARE THE TWO POINTERS THE SAME?
313 02365 R 224240 A LAC PNTR1# /YES, NO NEED TO OVERWRITE THE LAST ENTRY.
314 02366 R 723204 A DAC +4 /NO, BRING DOWN A LATER ENTRY,
315 02367 R 723204 A DAC +4 /ARE THE TWO POINTERS THE SAME?
316 02368 R 224240 A LAC PNTR1# /YES, NO NEED TO OVERWRITE THE LAST ENTRY.
317 02369 R 723204 A DAC PNTR1# /NO, BRING DOWN A LATER ENTRY,
318 02370 R 723204 A DAC PNTR1# /ARE THE TWO POINTERS THE SAME?
319 02371 R 224240 A LAC PNTR1# /YES, NO NEED TO OVERWRITE THE LAST ENTRY.
320 02372 R 723204 A DAC PNTR1# /NO, BRING DOWN A LATER ENTRY,
321 02373 R 723204 A DAC PNTR1# /ARE THE TWO POINTERS THE SAME?
322 02374 R 224240 A LAC PNTR1# /YES, NO NEED TO OVERWRITE THE LAST ENTRY.
323 02375 R 723204 A DAC PNTR1# /NO, BRING DOWN A LATER ENTRY,
324 02376 R 723204 A DAC PNTR1# /ARE THE TWO POINTERS THE SAME?
325 02377 R 224240 A LAC PNTR1# /YES, NO NEED TO OVERWRITE THE LAST ENTRY.
326 02378 R 723204 A DAC PNTR1# /NO, BRING DOWN A LATER ENTRY,
327 02379 R 723204 A DAC PNTR1# /ARE THE TWO POINTERS THE SAME?
328 02380 R 224240 A LAC PNTR1# /YES, NO NEED TO OVERWRITE THE LAST ENTRY.
329 02381 R 723204 A DAC PNTR1# /NO, BRING DOWN A LATER ENTRY,
330 02382 R 723204 A DAC PNTR1# /ARE THE TWO POINTERS THE SAME?
331 02383 R 224240 A LAC PNTR1# /YES, NO NEED TO OVERWRITE THE LAST ENTRY.


**Title**: 'Scope Handler Routines, Information.

- Keyboard-'Scope Handler for RSX Phase II.
- Control C and Control T are implemented.
- Other special input keys are not, on input.
- Control U and Rub-Out are implemented, and
- Carriage return on mode 2 input gives
- Rise to a returned event variable of *2
  - Instead of +2.

**The Following** is the shared core used by this handler:

- (6)7527 Secondary processor request flag.
- (6)7530 Secondary processor handler busy flag.
- (6)7531 Mode code.
- (6)7532 Trigger.
- (6)7533-(6)7656 124 (base b) buffer (header
  - 80 char, - room to insert CR, LF
- (6)7557 Guard word for output buffer.
- (6)7513 Flag for input.
- (6)7514 Flag for output.

**As an expedience to save memory, the main processor section**

- of the plotter 'Scope handler is also contained within
- this handler, special I/O function code 10000 is used,
- with the following CAL parameter block:

- Word 0: function code 10000.
- Word 1: event variable address.
- Word 2: logical unit number.
- Word 3: sub-function code for the plotter.
- Word 4: address of the parameter table for this function.

**The sub-function code is passed to the secondary processor,**

- which decodes it as follows:

- Code 0: function SCLR performed.
- Bit 17 on: function PNT performed.
- Bit 16 on: function HZ performed.
- Bit 15 on: function VL performed.
- Bit 14 on: function LIN performed.
- Only one function is performed per call; if more than one

**Bit is on, preference is in the order listed, no error**

- Is returned for an illegal code; the secondary processor
- Will ignore the request, the event variable, if specified.
- Will always be either zero (arguments not yet transferred)
- To shared memory), +1 (arguments transferred, though not
- Necessarily acted upon by the secondary processor), or -22
- (The secondary processor has not yet finished the last

**Graphics, if the event variable is -22, the request**

- Should be issued again.

- Shared memory used by this section of the handler is location

- (6)7512 for the dispatch code and (6)7470 = (6)7477 for

- The sub-function code, arguments, and busy flag.


**This program is supported by P. N. Bartram,**
KSG INITIALIZATION.

TITLE INITIALIZATION.

/ OCT
/OCTAL CONSTANTS SPOKEN HERE.
/BIAS FOR SHARED CORE ADDRESSES.

START LAC PDVL /SCAN PHYSICAL DEVICE LIST
DAC R1 /FOR NODE FOR THIS DEVICE.
DAC LHNAM
JMS SNAM /IF NODE FOUND, EXIT.
CAL TEN
DAC PVNA /IF NODE FOUND, SAVE PDVL NODE ADR.
AAC 10 /AND TRIGGER EVENT VARIABLE ADR.
DAC PVTA
DAC WFTCPB*1 /SET TRIGGER EVENT VARIABLE
DAC PVTA /IN PHYSICAL DEVICE NODE.
LAC MARK-1 /GET STORAGE ADR IN THIS
AAC 13 /AND DETERMINE "XR-ADJ".
DAC XADJ
LAC BUSY /INITIALIZE SCOPE SCREEN.
DAC X12 /SET UP TO WRITE A FORM
LAC MDE /FEED (MODE 3 USED).
DAC 12
DAC* 12
LAC HEAO /(GIVE HEADER.),
DAC* 12
LAC MDE 3 /MODE FOR ABOVE.
LHNAM .OSA HNAM /ADDRESS OF DEVICE NAME.
LHNAM .SIXBT "KSG0000" /HANDLER TASK NAME KS....

/ MAIN TASK STARTS HERE, PART OF ABOVE
/ IS OVERWRITTEN BY STORAGE.
/ WAIT FOR TRIGGER, CAL PARAMETER BLOCK.
WFTCPB 20 /CODE 20 IS FOR WAIT FOR.
DAC PVTA
DAC* 12
DAC WFTGR /WAIT FOR TRIGGER.
DAC WFTGR /WAIT FOR TRIGGER.
DAC* 12
LAC HEAO /FIRST PROCESSOR TO
DAC WFTGR /INITIALIZE 'SCOPE.
DAC WFTGR /INITIALIZE 'SCOPE.
DAC WFTGR /INITIALIZE 'SCOPE.
DAC WFTGR /INITIALIZE 'SCOPE.
DAC WFTGR /INITIALIZE 'SCOPE.
DAC WFTGR /INITIALIZE 'SCOPE.
DAC WFTGR /INITIALIZE 'SCOPE.
DAC WFTGR /INITIALIZE 'SCOPE.
DAC WFTGR /INITIALIZE 'SCOPE.
DAC WFTGR /INITIALIZE 'SCOPE.
FUNCTION DISPATCHER.

TITLE FUNCTION DISPATCHER,

/ TASK HAS BEEN TRIGGERED.

00047 R 222547 R PQ LAC* TGFLG / DID THE SECONDARY PROCESSOR SET THE TRIGGER?

00750 R 741200 A SNA

03051 R 602205 R JMP P01 / NO, THE I/O QUEUE MUST HAVE SET IT.

03052 R 162547 R DEM* TGFLG / YES, ZERO THE FLAG,

03053 R 547533 R SAD ONE / INPUT LINE IS READY, EVENT VARIABLE IS

03054 R 622722 R JMP REEND / TO BE +1,

03055 R 547534 R SAD TWO / INPUT LINE IS READY, EVENT VARIABLE IS

03056 R 622722 R JMP REEND / TO BE +2,

03057 R 547535 R SAD CNTRT / CONTROL T WAS KEYED IN, REQUEST THE

03058 R 622733 R JMP TDV / TOV DISPATCHER, IF APPROPRIATE,

03059 R 547536 R SAD CNTRC / CONTROL C WAS KEYED IN, REQUEST THE

03060 R 622733 R JMP MCRC / MCR DISPATCHER, IF APPROPRIATE,

03061 R 222744 R PQ1 LAC WTFLG / IS THERE A SEMI-PROCESSED REQUEST IN

03062 R 742220 A SZA

03063 R 637524 R JMP WAII / YES, DO NOT DEQUEUE ANOTHER REQUEST,

03064 R 222744 R LAC REAO / NO, IS THERE AN UNSATISFIED READ

03065 R 742220 A SEA

03066 R 630125 R JMP PQA30 / YES, CHECK IF THIS REQUEST IF FOR ABORT,

03067 R 747270 A SEA / REQUEST?

03068 R 637346 R JMP WFTGR / IF NONE FOUND, WAIT FOR TRIGGER.

03069 R 767566 R OAC* R1

03070 R 127576 R JMS* DORQ / R1, R2, R4, R5, XR, & AC ALTERED,

03071 R 573537 R AND MASK2

03072 R 543557 R SAD A8CFC / IF ABORT REQ, ABORT TASK I/O,

03073 R 677133 R JMP ABORT

03074 R 540560 R SAD ATCFC / IF ATTACH REQ, ATTACH TO A TASK,

03075 R 637564 R SAD DETACH / IF DETACH REQ, DETACH FROM A TASK,

03076 R 547563 R SAD WRFC / IF WRITE REQ, GO TO IT.

03077 R 547562 R SAD HICFC / IF HANDLER INFORMATION REQUESTED,

03078 R 630125 R SAD MCF C / IF HANDLER INFORMATION REQUESTED,

03079 R 602175 R JMP HINF / GIVE IT,

03080 R 540564 R SAD DEFC / IF EXIT REQUESTED (DEASSIGNED), DETACH

03081 R 620177 R JMP DEX / AND EXIT,

03082 R 547564 R SAD PLTFC / IF PLOTTING 'SCOPE REQUESTED, GO TO

03083 R 602172 R JMP SCLPT / ITS HANDLER SECTION,

03084 R 777777 A ILFUNC LAW = 6 / OTHERWISE RETURN UNIMPLEMENTED

03085 R 622426 R JMP SEV / FUNCTION EV CODE,

03086 R 222552 R PQA80 LAC* TG / READ IS IN PROGRESS, IS THE NEXT

03087 R 162552 R DEM* TG / REQUEST FOR AN ABORT?
PACE « KSG FUNCTION DISPATCHER.

157  00127 R 742018 A  RTL
158  00130 R 742100 A  SMA
159  00131 R 600346 R  JMP  WFTGR /NO. WAIT UNTIL THE READ IS COMPLETE.
160  00132 R 600372 R  JMP  P02 /YES. DEQUEUE THIS REQUEST.
ABORT FUNCTION.

TITLE ABORT FUNCTION.

/ ABORT ALL I/O INITIATED BY THE INDICATED TASK.

/ ABORT IS AN ILLEGAL FUNCTION

/ GET THE STL NODE ADDRESS, AND CHECK IF IT IS THE SAME AS THE TASK WAITING

/ FOR INPUT, IF ANY.

/ GET PHYSICAL DEVICE NODE ADR,

/ GET REQUEST NODE ADR,

/ THEN EMPTY THE QUEUE OF ALL I/O REQUESTS

/ MADE BY THE TASK BEING ABORTED, (R1,

/ R2,R3,R5,R8,X10,X11,X12,RX,AC ALTERED,)

/ PROGRAM ABORTED THAT REQUESTED INPUT,

/ CLEAR THE READ IN PROGRESS FLAG, AND

/ TRANSFER THE REQUEST NODE ADDRESS TO

/ FOR THE ABORTED TASK, AND RETURN THE READ

/ REQUEST NODE TO THE POOL,

/ CONTINUE WITH THE ABORT FUNCTION,
ATTACH, DETACH, MINF, AND EXIT FUNCTIONS.

TITLE ATTACH, DETACH, MINF, AND EXIT FUNCTIONS.

/ ATTACH TO A TASK.
00157 R 200001 R ATTACH LAC PDVNA /ATTACH LUN & DEVICE.
00160 R 207556 R DAC x R1 /IF ALREADY ATTACHED, EV TO -20 OR -203.
00161 R 207557 R DAC x R2
00162 R 207557 R R2
00163 R 127560 R JMS x ALAD
00164 R 200240 R JMP SEV /IF NOT PREVIOUSLY ATTACHED, EV TO +1.
00165 R 200240 R

/ DETACH FROM A TASK.
00166 R 200001 R DETACH LAC PDVNA /DETACH LUN & DEVICE.
00167 R 207556 R DAC x R1
00170 R 207557 R DAC x R2
00171 R 207557 R R2
00172 R 127560 R JMS x DLAO /(R3,R4,R5,X10,X11,XR,6AC ALTERED.)
00173 R 600226 R JMP SEV /IF LUN NOT DETACHED, EV TO -24 OR -203.
00174 R 600225 R

/ RETURN HANDLER INFORMATION IN EVENT VARIABLE.
00175 R 200543 R MINF LAC CHINF
00176 R 200542 R JMP SEV

/ EXIT REQUEST (FROM TASK "...REA").
00177 R 207634 R
00200 R 207566 R DAC x R1
00201 R 207567 R DAC x R2
00202 R 127560 R JMS x NADD
00203 R 400202 R ISE PDVTA /CLEAR ADDING INHIBIT FLAG IN THE
00204 R 500522 A IOT 5522 /POVL NODE (INHIBIT INTERRUPTS
00205 R 500522 A PDVTA /WHILE GOING SO),
00206 R 500521 A IOT 5521
00207 R 500521 A CAL TEN /EXIT.

PAGE 6  KSG
KSC INPUT (READ) FUNCTION,

TITLE INPUT (READ) FUNCTION,

READ ONE LINE FROM 'SCOPE KEYBOARD. ONE LINE IS UP TO 83 CHARACTERS, REGARDLESS OF BUFFER SIZE SPECIFIED BY THE REQUESTING PROGRAM, IF THAT REQUESTED IS SMALLER, THE LINE INPUT WILL BE TRUNCATED WHEN TRANSFERRED TO THE REQUESTOR'S BUFFER.

ONLY MODES 2 AND 3 ARE AVAILABLE.

READ LAC 11,X /GET REQUESTOR'S BUFFER SIZE.
DAC ROSIE
LAC 2,X /GET THE REQUESTOR'S STL NODE ADDRESS.
DAC READP /AND USE IT AS THE READ IN PROGRESS FLAG.
LAC RN /SAVE THE REQUEST NODE ADDRESS IN CASE
DAC RNRD /AN ABORT REQUEST IS PROCESSED.
JMS SETUP /ASSURE THE SECONDARY PROCESSOR IS READY.
DAC RNRED /GET MODE, AND GET REQ. BUFFER ADDRESS.
DAC ROSTR /DEPOSIT ADDR, MODE STORED BY SETUP.
LAC RO10 /INTERCEPT THE SECONDARY PROCESSOR TO
JMS CJ10 /INITIATE READ.
JMP WFTGR /WAIT FOR THE LINE TO BE FINISHED.

TRIGGER DISPATCHER SENDS CONTROL HERE IF LINE IS READY.

READ LAC 11,X /GET REQUESTOR'S BUFFER SIZE.
DAC ROSIE
LAC 2,X /GET THE REQUESTOR'S STL NODE ADDRESS.
DAC READP /AND USE IT AS THE READ IN PROGRESS FLAG.
LAC RN /SAVE THE REQUEST NODE ADDRESS IN CASE
DAC RNRD /AN ABORT REQUEST IS PROCESSED.
JMS SETUP /ASSURE THE SECONDARY PROCESSOR IS READY.
DAC RNRED /GET MODE, AND GET REQ. BUFFER ADDRESS.
DAC ROSTR /DEPOSIT ADDR, MODE STORED BY SETUP.
LAC RO10 /INTERCEPT THE SECONDARY PROCESSOR TO
JMS CJ10 /INITIATE READ.
JMP WFTGR /WAIT FOR THE LINE TO BE FINISHED.

ONLY MODES 2 AND 3 ARE AVAILABLE.

READ LAC 11,X /GET REQUESTOR'S BUFFER SIZE.
DAC ROSIE
LAC 2,X /GET THE REQUESTOR'S STL NODE ADDRESS.
DAC READP /AND USE IT AS THE READ IN PROGRESS FLAG.
LAC RN /SAVE THE REQUEST NODE ADDRESS IN CASE
DAC RNRD /AN ABORT REQUEST IS PROCESSED.
JMS SETUP /ASSURE THE SECONDARY PROCESSOR IS READY.
DAC RNRED /GET MODE, AND GET REQ. BUFFER ADDRESS.
DAC ROSTR /DEPOSIT ADDR, MODE STORED BY SETUP.
LAC RO10 /INTERCEPT THE SECONDARY PROCESSOR TO
JMS CJ10 /INITIATE READ.
JMP WFTGR /WAIT FOR THE LINE TO BE FINISHED.

TRIGGER DISPATCHER SENDS CONTROL HERE IF LINE IS READY.

READ LAC 11,X /GET REQUESTOR'S BUFFER SIZE.
DAC ROSIE
LAC 2,X /GET THE REQUESTOR'S STL NODE ADDRESS.
DAC READP /AND USE IT AS THE READ IN PROGRESS FLAG.
LAC RN /SAVE THE REQUEST NODE ADDRESS IN CASE
DAC RNRD /AN ABORT REQUEST IS PROCESSED.
JMS SETUP /ASSURE THE SECONDARY PROCESSOR IS READY.
DAC RNRED /GET MODE, AND GET REQ. BUFFER ADDRESS.
DAC ROSTR /DEPOSIT ADDR, MODE STORED BY SETUP.
LAC RO10 /INTERCEPT THE SECONDARY PROCESSOR TO
JMS CJ10 /INITIATE READ.
JMP WFTGR /WAIT FOR THE LINE TO BE FINISHED.

TRIGGER DISPATCHER SENDS CONTROL HERE IF LINE IS READY.

READ LAC 11,X /GET REQUESTOR'S BUFFER SIZE.
DAC ROSIE
LAC 2,X /GET THE REQUESTOR'S STL NODE ADDRESS.
DAC READP /AND USE IT AS THE READ IN PROGRESS FLAG.
LAC RN /SAVE THE REQUEST NODE ADDRESS IN CASE
DAC RNRD /AN ABORT REQUEST IS PROCESSED.
JMS SETUP /ASSURE THE SECONDARY PROCESSOR IS READY.
DAC RNRED /GET MODE, AND GET REQ. BUFFER ADDRESS.
DAC ROSTR /DEPOSIT ADDR, MODE STORED BY SETUP.
LAC RO10 /INTERCEPT THE SECONDARY PROCESSOR TO
JMS CJ10 /INITIATE READ.
JMP WFTGR /WAIT FOR THE LINE TO BE FINISHED.

TRIGGER DISPATCHER SENDS CONTROL HERE IF LINE IS READY.

READ LAC 11,X /GET REQUESTOR'S BUFFER SIZE.
DAC ROSIE
LAC 2,X /GET THE REQUESTOR'S STL NODE ADDRESS.
DAC READP /AND USE IT AS THE READ IN PROGRESS FLAG.
LAC RN /SAVE THE REQUEST NODE ADDRESS IN CASE
DAC RNRD /AN ABORT REQUEST IS PROCESSED.
JMS SETUP /ASSURE THE SECONDARY PROCESSOR IS READY.
DAC RNRED /GET MODE, AND GET REQ. BUFFER ADDRESS.
DAC ROSTR /DEPOSIT ADDR, MODE STORED BY SETUP.
LAC RO10 /INTERCEPT THE SECONDARY PROCESSOR TO
JMS CJ10 /INITIATE READ.
JMP WFTGR /WAIT FOR THE LINE TO BE FINISHED.

TRIGGER DISPATCHER SENDS CONTROL HERE IF LINE IS READY.

READ LAC 11,X /GET REQUESTOR'S BUFFER SIZE.
DAC ROSIE
LAC 2,X /GET THE REQUESTOR'S STL NODE ADDRESS.
DAC READP /AND USE IT AS THE READ IN PROGRESS FLAG.
LAC RN /SAVE THE REQUEST NODE ADDRESS IN CASE
DAC RNRD /AN ABORT REQUEST IS PROCESSED.
JMS SETUP /ASSURE THE SECONDARY PROCESSOR IS READY.
DAC RNRED /GET MODE, AND GET REQ. BUFFER ADDRESS.
DAC ROSTR /DEPOSIT ADDR, MODE STORED BY SETUP.
LAC RO10 /INTERCEPT THE SECONDARY PROCESSOR TO
JMS CJ10 /INITIATE READ.
JMP WFTGR /WAIT FOR THE LINE TO BE FINISHED.

TRIGGER DISPATCHER SENDS CONTROL HERE IF LINE IS READY.

READ LAC 11,X /GET REQUESTOR'S BUFFER SIZE.
DAC ROSIE
LAC 2,X /GET THE REQUESTOR'S STL NODE ADDRESS.
DAC READP /AND USE IT AS THE READ IN PROGRESS FLAG.
LAC RN /SAVE THE REQUEST NODE ADDRESS IN CASE
DAC RNRD /AN ABORT REQUEST IS PROCESSED.
JMS SETUP /ASSURE THE SECONDARY PROCESSOR IS READY.
DAC RNRED /GET MODE, AND GET REQ. BUFFER ADDRESS.
DAC ROSTR /DEPOSIT ADDR, MODE STORED BY SETUP.
LAC RO10 /INTERCEPT THE SECONDARY PROCESSOR TO
JMS CJ10 /INITIATE READ.
JMP WFTGR /WAIT FOR THE LINE TO BE FINISHED.

TRIGGER DISPATCHER SENDS CONTROL HERE IF LINE IS READY.

READ LAC 11,X /GET REQUESTOR'S BUFFER SIZE.
DAC ROSIE
LAC 2,X /GET THE REQUESTOR'S STL NODE ADDRESS.
DAC READP /AND USE IT AS THE READ IN PROGRESS FLAG.
LAC RN /SAVE THE REQUEST NODE ADDRESS IN CASE
DAC RNRD /AN ABORT REQUEST IS PROCESSED.
JMS SETUP /ASSURE THE SECONDARY PROCESSOR IS READY.
DAC RNRED /GET MODE, AND GET REQ. BUFFER ADDRESS.
DAC ROSTR /DEPOSIT ADDR, MODE STORED BY SETUP.
LAC RO10 /INTERCEPT THE SECONDARY PROCESSOR TO
JMS CJ10 /INITIATE READ.
JMP WFTGR /WAIT FOR THE LINE TO BE FINISHED.

TRIGGER DISPATCHER SENDS CONTROL HERE IF LINE IS READY.

READ LAC 11,X /GET REQUESTOR'S BUFFER SIZE.
DAC ROSIE
LAC 2,X /GET THE REQUESTOR'S STL NODE ADDRESS.
DAC READP /AND USE IT AS THE READ IN PROGRESS FLAG.
LAC RN /SAVE THE REQUEST NODE ADDRESS IN CASE
DAC RNRD /AN ABORT REQUEST IS PROCESSED.
JMS SETUP /ASSURE THE SECONDARY PROCESSOR IS READY.
DAC RNRED /GET MODE, AND GET REQ. BUFFER ADDRESS.
DAC ROSTR /DEPOSIT ADDR, MODE STORED BY SETUP.
LAC RO10 /INTERCEPT THE SECONDARY PROCESSOR TO
JMS CJ10 /INITIATE READ.
JMP WFTGR /WAIT FOR THE LINE TO BE FINISHED.

TRIGGER DISPATCHER SENDS CONTROL HERE IF LINE IS READY.
INPUT (READ) FUNCTION,

284 00257 R 040004 R DAC TEMP
285 00258 R 220512 A LAC* 12 /TRANSFER REST OF BUFFER.
286 00259 R 060013 A DAC* 13 /REMEMBER X12, X13 AUTOINCREMENT.
287 00260 R 440004 R IS INT
288 00261 R 600260 R JMP -3
289 00262 R 160550 R DEM* BUSY /ZERO READ ACCEPT FLAG.
290 00263 R 140545 R DEM READP /ZERO THE READ IN PROGRESS FLAG.

THE FOLLOWING CODE IS SIMILAR TO THAT OF SUBROUTINE SEVRN, BUT
THE READ NODE MUST BE USED, AND AN ABO REQUEST MAY HAVE BEEN
DEQUEUED WHILE WAITING FOR THE INPUT READY FLAG FROM THE
SECONDARY PROCESSOR. THE CODE IS REPEATED HERE TO SAVE TIME.
AS THERE IS ROOM IN THIS PARTITION FOR A LITTLE REDUNDANCY,
SHOULD CHANGES BE REQUIRED THAT REQUIRE MORE SPACE, THIS SHOULD
BE RECODED TO USE A REVISED SEVRN.

297 00266 R 200012 R LAC RNRED /GET THE NODE ADDRESS FOR THE READ REQUEST,
298 00267 R 340233 R TAD XADJ / WHICH IS BEING COMPLETED,
299 00268 R 721500 A PAX /READ NODE ADDRESS;
300 00269 R 210025 A LAC 6.X /GET THE REQUESTR'S EVENT VARIABLE ADDRESS,
301 00270 R 741200 A SNA / IF ONE WAS SPECIFIED.
302 00271 R 623323 R JMP RNOST /SET THE EVENT VARIABLE.
303 00272 R 340573 R TAD XADJ /AND DECLARING A SIGNIFICANT EVENT,
304 00273 R 721703 R PAX /RETURN HERE.
305 00274 R 731000 A PLA /EVENT VARIABLE WAS SAVED IN THE LIMIT REG,
306 00275 R 052000 R DAC 0.X /DECLARIE THE /0 READ REQUEST COMPLETED BY
307 00276 R 252012 A RNOST LAC RNRED /DECLARE THE /0 READ REQUEST COMPLETED BY
308 00277 R 020006 R DAC 8.X /DECLARIE THE /0 READ REQUEST COMPLETED BY
309 00278 R 722000 A PLA /EVENT VARIABLE WAS SAVED IN THE LIMIT REG,
310 00279 R 622012 A PAX /RETURN HERE.
311 00280 R 705004 A LAC SEEN /RETURN THE READ REQUEST NODE ADDRESS
312 00281 R 705004 A LAC SEEN /RETURN THE READ REQUEST NODE ADDRESS
313 00282 R 705004 A LAC SEEN /RETURN THE READ REQUEST NODE ADDRESS
314 00283 R 200012 R LAC POO /RETURN THE READ REQUEST NODE ADDRESS
315 00284 R 340233 R TAD XADJ / WHICH IS BEING COMPLETED,
316 00285 R 061500 A DAC* 11 /READ NODE ADDRESS;
317 00286 R 122500 A JNS* ADD /READ NODE ADDRESS;
318 00287 R 050000 A OAC 0.X /GET THE NEXT /0 REQUEST, IF ANY,
319 00288 R 160550 R ABRED DEM* BUSY /READ WAS ABORTED, CLEAR HANDLER
320 00289 R 020012 R JP PO /AND GET THE NEXT REQUEST.

321 00290 R 220542 R LAC LUT /CHECK IF LUN 2 IS
322 00291 R 742333 A IAC /ASSIGNED TO THIS DEVICE
323 00292 R 042029 R DAC TEMP /BY COMPARING PHYSICAL
324 00293 R 221500 A LAC TEMP /DEVICE NODE ADDRESSES,
325 00294 R 542341 R SAD POWN
326 00295 R 751200 A SKP*CLA
327 00296 R 602207 R JMP -3 /EXIT IF NOT.
328 00297 R 542341 R SAD MCRAL /THIS IS THE MCR DEVICE
329 00298 R 603232 R JMP MCRCL /IF INHIBIT CODE IS
330 00299 R 777777 A LAW -1 /NOT ZERO, SET IT TO
INPUT (READ) FUNCTION

334 00325 R 065567 R  
DAC* MCCR 2 / -1.
335 00326 R 667847 R  
JMP PO
336 00327 R 465687 R MCRCL ISZ* MCR 2 / IF INHIBIT CODE IS
337 00328 R 065232 R CAL MRCAL / ZERO, SET IT TO -1
338 00331 R 667847 R  
JMP PO / AND CALL MCR DISPATCHER.
339  
340 / CAL PARAMETER BLOCK FOR REQUESTING MCR.
341  
342 00332 R 065232 A  
MCCAL 1 /REQUEST CODE.
343 00333 R 062200 A 0 /NO EVENT VARIABLE.
344 00334 R 656856 A ,SIXBT "...MCR" /TASK NAME.
345 00335 R 152322 A  
346 00336 R 002009 A 0 /DEFAULT PRIORITY,
347  
348 / CONTROL T INPUT.
349  
350 00337 R 222676 R  
TDV LAC* LUT /CHECK IF LUN 12 IS
351 00342 R 34573 R  
TAD Y1S / ASSIGNED TO THIS
352 00343 R 243295 R  
DAC TEMP2 / DEVICE BY COMPARING
353 00342 R 222676 R  
LAC* TEMP2 / PHYSICAL DEVICE
354 00343 R 542681 R  
SAD PDMNA / NODE ADDRESSES.
355 00344 R 057346 R  
CAL TDCL /IF IT IS, REQUEST TDV.
356 00345 R 602242 R  
JMP PO /OTHERWISE, IGNORE.
357  
358 / CAL PARAMETER BLOCK FOR REQUESTING TDV DISPATCHER.
359  
360 00346 R 062200 A  
TDCL 1 /REQUEST CODE.
361 00347 R 062200 A 0 /NO EVENT VARIABLE.
362 00348 R 240426 A ,SIXBT "TDV..." /TASK NAME.
363 00349 R 565656 A  
364 00350 R 003030 A 0 /USE DEFAULT PRIORITY.
.TITLE CHARACTER OUTPUT (WRITE) FUNCTION.

/ PRINT LINE.
/ IF DATA IS NOT OF MODE 2 (5/7 ASCII) OR
/ MODE 3 (IMAGE ASCII), RETURN EV = -7.
/

00353 R 100462 R WRITE JMS SETUP
/ INSURE THAT THE SECONDARY PROCESSOR IS READY,
/ CHECK MODE, AND GET BUFFER ADDRESS IN AC.
00354 R 205572 R DAC X12 /X12 = BUFFER STARTINGADR = 1.
00355 R 205552 R LAC TO /SCAPE ADDR = 1 TO X13.
00356 R 205573 R DAC X13
00357 R 227012 A LAC 12 /MOVE FIRST HEADER LINE AND
00358 R 226213 A DAC 13 /GET LENGTH IN WORDS.
00359 R 525543 R AND MASK3
00360 R 762571 R DAC R4
00361 R 723776 A AAC -2 /IF LESS THAN TWO WORDS, SET
00362 R 742170 A SMA /USER EV TO -16.
00363 R 627372 R JMP
00364 R 160550 R DEM BUSY /THE SECONDARY PROCESSOR IS NOT NEEDED
00365 R 777762 A LAW -16 /AFTER ALL.
00366 R 203552 R LAC TG /SCOPE BFR AOR - 1 TO X13.
00367 R 620422 R JMP ERS30 /IF NOT OK, EV TO -30,
00368 R 767013 A DAC R4 /IF OK, SET UP TEMPI AS WORD
00369 f 747231 A TCA / COUNT FOR MOVE (ADO 1
00370 R 747Z3Z A I  AC / FOR REST OF HEADER).
00371 R 747074 R OAC TEMPI /CHECK THAT BUFFER WILL
00372 R 723121 A AAC 121 /BE TOO LARGE, ADJUSTING
00373 R 740170 A SMA /SIZE IF NECESSARY.
00374 R 687426 R JMP
00375 R 207074 R LAC TEMPI /VERIFY LINE SIZE (NORMAL MODE),
00376 R 205224 R LAC TEMPI
00377 R 205570 R DAC R3 /R2 & R4 ARE SET UP.
00378 R 223652 R JMS VAJX /(R3,R5,XA &AC ALTERED.)
00379 R 223422 R DAC R4 /IF NOT OK, EV TO -30.
00380 R 22376 R 222571 R LAC R4
00381 R 747031 A TCA /COUNT FOR MOVE (ADO 1
00382 R 747223 A IAC /FOR REST OF HEADER),
00383 R 249204 R OAC TEMP1 /CHECK THAT BUFFER WILL
00384 R 723121 A AAC 121 /BE TOO LARGE, ADJUSTING
00385 R 740170 A SMA /SIZE IF NECESSARY.
00386 R 742170 A SMA /USER EV TO -16.
00387 R 742170 A SMA
00388 R 607047 R JMP
00389 R 207554 R LAC TEMPI /LINE HAS BEEN MOVED AND IS READY TO
00390 R 222012 A DAC 12 /MOVE REMAINDER OF LINE TO
00391 R 222013 A DAC 13 /SCAPE BUFFER. REMEMBERING
00392 R 222013 A DAC 13 /IS THERE AT THE END.
00393 R 757030 A LAC TEMPI /SCOPE BUFFER, REMEMBERING
00394 R 741030 A CLK SEVRN /BE PRINTED FROM THE INTERNAL BUFFER,
00395 R 607047 R JMP /INDICATE TO THE REQUESTOR THAT HIS LINE
00396 R 607047 R JMP /HAS BEEN PRINTED BY SETTING HIS EV TO -1.
00397 R 207554 R LAC WRAT10 /INFORM THE SECONDARY PROCESSOR TO
00398 R 107555 R JMS CLL10 /WRITE FROM THE BUFFER,
00399 R 607047 R JMP /PICK UP NEXT REQUEST.
00400 R 227571 R LAC GUARD /MAKE SURE THAT THE GUARD
00401 R 741030 A CL #1AC /LINE HAS BEEN MOVED AND IS READY TO
00402 R 100430 R JMS SEVRN /BE PRINTED FROM THE INTERNAL BUFFER,
00403 R 207554 R LAC TEMPI /MOVE REMAINDER OF LINE TO
00404 R 222012 A DAC 12 /SCOPE BUFFER. REMEMBERING
00405 R 222013 A DAC 13 /IS THERE AT THE END.
00406 R 757030 A LAC TEMPI /SCOPE BUFFER, REMEMBERING
00407 R 741030 A CLK SEVRN /BE PRINTED FROM THE INTERNAL BUFFER,
00408 R 607047 R JMP /INDICATE TO THE REQUESTOR THAT HIS LINE
00409 R 607047 R JMP /HAS BEEN PRINTED BY SETTING HIS EV TO -1.
00410 R 207554 R LAC WRAT10 /INFORM THE SECONDARY PROCESSOR TO
00411 R 107555 R JMS CLL10 /WRITE FROM THE BUFFER,
00412 R 607047 R JMP /PICK UP NEXT REQUEST.
00413 R 777762 A LAW -30 /EV TO -32.
00414 R 167550 A DAC BUSY /SECONDARY PROCESSOR WILL NOT BE NEEDED.
CHARACTER OUTPUT (WRITE) FUNCTION.

/ TERMINATION OF COMPLETED REQUESTS.

/SET THE EVENT VARIABLE TO *1.

/SET EV, DECLARE SIG. EV., RETURN

/ REQ. NODE TO POOL, PICK UP NEXT REQ.
.TITLE SUBROUTINES

SEVRN - SUBROUTINE TO SET THE REQUESTOR'S EVENT VARIABLE TO THE QUANTITY IN AC, DECLARE A SIGNIFICANT EVENT, DECREMENT I/O TRANSFERS PENDING COUNT (NORMAL MODE), AND RETURN REQUEST NODE TO THE POOL.

ENTRY POINT.

SAVE EV VALUE,
REQUEST NODE ACR,
ADJUST FOR MEMORY BLOCK,
SET EV.
DECLARE I/O REQUEST COMPLETED.
(DECLEMENT TRANSFERS PENDING COUNT),
DECLARE SIGNIFICANT EVENT.
(AFTER SERVICING THE SOFTWARE INTERRUPT JUST DECLARED),
CONTROL RETURNS HERE.
RETURN REQUEST NODE TO POOL = R2
IS STILL SET.
EXIT SEVRN ROUTINE,
SUBROUTINE ENTRY.
RAISE PROPER FLAG IN THE SECONDARY PROCESSOR DISPATCH TABLE.
INTERRUPT SECONDARY PROCESSOR.
RETURN.
SUBROUTINE ENTRY,
RAISE PROPER FLAG IN THE SECONDARY PROCESSOR DISPATCH TABLE.
INTERRUPT SECONDARY PROCESSOR.
RETURN.
SUBROUTINE ENTRY.
IF KS BUSY, WAIT.
SUBROUTINE ENTRY.
IF NOT, DECLARE IT BUSY.
GET THE MODE INDICATOR.
IF NOT MODE 2 OR 3, SET REQUESTOR'S
<table>
<thead>
<tr>
<th>LINE</th>
<th>ADDRESS</th>
<th>INSTRUCTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>475</td>
<td>00472 R 540534 R</td>
<td>SAD TWO / EVENT VARIABLE TO -7 AND EXIT.</td>
</tr>
<tr>
<td>476</td>
<td>00473 R 603477 R</td>
<td>JMP PLSET /SECONDARY PROCESSOR NOT NEEDED.</td>
</tr>
<tr>
<td>477</td>
<td>00474 R 777777 A</td>
<td>LAW -7 /VERIFY &amp; ADJUST (NORMAL MODE) THE</td>
</tr>
<tr>
<td>478</td>
<td>00475 R 165555 R</td>
<td>D2H* BUSY / ADDRESS OF THE BEGINNING OF THE</td>
</tr>
<tr>
<td>479</td>
<td>00476 R 660626 R</td>
<td>JMP SEV /SECONDARY PROCESSOR NOT NEEDED.</td>
</tr>
<tr>
<td>480</td>
<td>00477 R 222320 R</td>
<td>PLSET LAC R2 / LINE (HEADER ADDRESS).</td>
</tr>
<tr>
<td>481</td>
<td>00478 R 662067 R</td>
<td>DAC* R2 / ADDRESS OF THE BEGINNING OF THE</td>
</tr>
<tr>
<td>482</td>
<td>00479 R 217171 A</td>
<td>LAC 10.X / SECONDARY PROCESSOR NOT NEEDED.</td>
</tr>
<tr>
<td>483</td>
<td>00480 R 763580 R</td>
<td>DAC* R3 /SAVE UN-ADJUSTED BASE FOR SECOND</td>
</tr>
<tr>
<td>484</td>
<td>00481 R 234374 R</td>
<td>DAC TEMP1 / JMS.)</td>
</tr>
<tr>
<td>485</td>
<td>00482 R 752329 A</td>
<td>CLAIAC /</td>
</tr>
<tr>
<td>486</td>
<td>00483 R 232571 R</td>
<td>DAC* R4 /</td>
</tr>
<tr>
<td>487</td>
<td>00484 R 127672 R</td>
<td>JMS* VAJX / (R3,R4,XR,&amp;AC ALTERED,.)</td>
</tr>
<tr>
<td>488</td>
<td>00485 R 699422 R</td>
<td>JMP ERR30 /IP B04 ADR NOT OK, EV = -30.</td>
</tr>
<tr>
<td>489</td>
<td>00486 R 777777 A</td>
<td>LAW -1 /AC HAS BUFFER STARTING ADDRESS -1.</td>
</tr>
<tr>
<td>490</td>
<td>00487 R 367570 R</td>
<td>TAD* R3 /RETURN.</td>
</tr>
<tr>
<td>491</td>
<td>00488 R 622445 R</td>
<td>JMP* SETUP /RETURN.</td>
</tr>
<tr>
<td>492</td>
<td>00489 R 724200 A</td>
<td>WAIT PXA /IF BUSY, WAIT 1/6 SEC.</td>
</tr>
<tr>
<td>493</td>
<td>00490 R 442418 B</td>
<td>DAC CKSVX /SAVE &amp; RESTORE INDEX,</td>
</tr>
<tr>
<td>494</td>
<td>00491 R 447546 R</td>
<td>ISZ WTFLG /INDICATE WAIT STATUS.</td>
</tr>
<tr>
<td>495</td>
<td>00492 R 447546 R</td>
<td>CAL MARK /</td>
</tr>
<tr>
<td>496</td>
<td>00493 R 302526 R</td>
<td>CAL WBSY /</td>
</tr>
<tr>
<td>497</td>
<td>00494 R 220531 R</td>
<td>JHP PG /SEE IF SECONDARY PROCESSOR SENT A FLAG.</td>
</tr>
<tr>
<td>498</td>
<td>00495 R 220531 R</td>
<td>/</td>
</tr>
<tr>
<td>499</td>
<td>00496 R 602424 R</td>
<td>WAIT1 LAC CKSVX /RETURN HERE AFTER CLEARING ANY FLAGS</td>
</tr>
<tr>
<td>500</td>
<td>00497 R 208010 R</td>
<td>RETURN HERE AFTER CLEARING ANY FLAGS</td>
</tr>
<tr>
<td>501</td>
<td>00498 R 208010 R</td>
<td>LAC CKSVX /RETURN HERE AFTER CLEARING ANY FLAGS</td>
</tr>
<tr>
<td>502</td>
<td>00499 R 208170 A</td>
<td>PAX /SET BY THE SECONDARY PROCESSOR.</td>
</tr>
<tr>
<td>503</td>
<td>00500 R 142546 R</td>
<td>WM WTFLG /CLEAR THE WAIT STATUS FLAG IN CASE THE</td>
</tr>
<tr>
<td>504</td>
<td>00501 R 602463 R</td>
<td>JHP SETUP+1 /THE BUSY FLAG IS DOWN, TRY AGAIN.</td>
</tr>
<tr>
<td>505</td>
<td>00502 R 222513 A</td>
<td>MARK 13 /CPB TO MARK TIME</td>
</tr>
<tr>
<td>506</td>
<td>00503 R 222513 A</td>
<td>EVH 13 /FOR 1/6 SEC.</td>
</tr>
<tr>
<td>507</td>
<td>00504 R 222513 A</td>
<td>EVH 12 /FOR 1/6 SEC.</td>
</tr>
<tr>
<td>508</td>
<td>00505 R 222513 A</td>
<td>EVH 1</td>
</tr>
<tr>
<td>509</td>
<td>00506 R 222513 A</td>
<td>WTBSY 20 /CPB FOR WAITING</td>
</tr>
<tr>
<td>510</td>
<td>00507 R 222513 A</td>
<td>EVH 20 /FOR THE MARK TIME.</td>
</tr>
</tbody>
</table>
TITLE CONSTANTS AND STORAGE.

/ CONSTANTS AND STORAGE CELLS.

00533 R 000201 A ONE 1 /ONE.
00534 R 000202 A TWO 2 /TWO.
00535 R 000200 A TEN 10 /EIGHT.

20536 R 272200 A MASK1 272080 /FOR DETERMINING XR-ADJ.
00537 R 000777 A MASK2 202777 /FOR GETTING CAL FUNCTION CODE.
00542 R 000776 A MASK3 000776 /FOR GETTING BUFFER LENGTH.

20540 R RN=START /ADDRESS OF RED NODE PICKED FROM QUEUE.
20541 R PDVNA=START+1 /POVL NODE ADDRESS STORAGE.
20542 R PDVTA=START+2 /ADDRESS OF TRIGGER EVENT VARIABLE ADDRESS.
20543 R XADJ=START+3 /XR ADJUST CONSTANT FOR MEMORY BLOCK.
20544 R TEMPL=START+4 /TEMP STORAGE.
20545 R TEMP2=START+5 /MORE TEMP STORAGE.
20546 R RDST=START+6 /READ BUFFER SIZE.
20547 R ROST=START+7 /ADDR OF START OF USER BUFFER.
02210 R CKVX=START+10 /SAVE CELL FOR WAIT ROUTINE.
02211 R EVM=START+11 /EVENT VARIABLE FOR BUSY CHECK WAIT.
02212 R NRED=START+12 /REQUEST NODE FOR THE LAST READ REQUEST.

00541 R 064175 A GUARD 064175 /GUARD WORD FOR END OF BUFFER.
00542 R 401280 A SECNS 401280 /ASC FOR ISA FOR SIGNIFICANT EVENT.
02220 R CHINF 302277 /MINF CODE = DEVICE # 63. READ & WRITE.
02221 R STBF 140220 /FOR SIGNALING BUFFER TOO SMALL.
02222 R VPDP 0 /IF READ IN PROGRESS, STL NODE ADR, ELSE 0.

00546 R 082220 A WTFLG 0 /REQUEST IN WAIT STATUS FLAG.

/ SHARED CORE BUFFER ADDRESSES.

00547 R 087527 A TGFLG 087527+BIAS /SECONDARY PROCESSOR REQUEST FLAG.
00548 R 087530 A BUSY 087530+BIAS /SECONDARY PROCESSOR BUSY FLAG.
00549 R 087531 A MODE 087531+BIAS /MODE INDICATOR.
00550 R 087532 A TG 087532+BIAS /TRIGGER.
00551 R 087533 A RO10 087533+BIAS /SECONDARY PROCESSOR TO READ.
00552 R 087534 A WAT10 087534+BIAS /SECONDARY PROCESSOR TO WRITE.

/ SCOPE CHARACTERS.

00555 R 000174 A CNTRC 174 /CONTROL C (COMPLEMENT).
00556 R 000153 A CNTRT 153 /CONTROL T (COMPLEMENT).

/ I/O FUNCTION CODES.

00557 R 000017 A ABCFC 017 /ABORT CAL FUNCTION CODE.
00558 R 000024 A ATFC 024 /ATTACH CAL FUNCTION CODE.
00559 R 000225 A DTFCFC 025 /DETACH CAL FUNCTION CODE.
00560 R 000266 A RDFCFC 026 /READ CAL FUNCTION CODE.
00563 R 000227 A WRCFC 027 /WRITE CAL FUNCTION CODE.
00564 R 000036 A HICFC 036 /HANDLER INFORMATION CAL FUNC. CODE.
<table>
<thead>
<tr>
<th>Page</th>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>564</td>
<td>00565 R 000120 A</td>
<td>PLTF 100 /SPECIAL CODE FOR PLOTTER 'SCOPE USE.</td>
</tr>
<tr>
<td>565</td>
<td>02537 R</td>
<td>DECFC=MASK2 /DETACH AND EXIT CAL FUCN CODE = 777.</td>
</tr>
<tr>
<td>566</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>567</td>
<td>/</td>
<td>ABSOLUTE ADDRESS LOCATIONS IN RESIDENT MONITOR.</td>
</tr>
<tr>
<td>568</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>569</td>
<td>00566 R 000121 A</td>
<td>R1 101 /RE-ENTRANT REGISTER ONE.</td>
</tr>
<tr>
<td>570</td>
<td>02567 R 000122 A</td>
<td>R2 122 /RE-ENTRANT REGISTER TWO.</td>
</tr>
<tr>
<td>571</td>
<td>02570 R 000123 A</td>
<td>R3 103 /RE-ENTRANT REGISTER THREE.</td>
</tr>
<tr>
<td>572</td>
<td>02571 R 000124 A</td>
<td>R4 104 /RE-ENTRANT REGISTER FOUR.</td>
</tr>
<tr>
<td>573</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>574</td>
<td>00572 R 000212 A</td>
<td>X12 12 /AUTO INCREMENT REGISTER 12.</td>
</tr>
<tr>
<td>575</td>
<td>02573 R 000213 A</td>
<td>X13 13 /AUTO INCREMENT REGISTER 13.</td>
</tr>
<tr>
<td>576</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>577</td>
<td>02574 R 000252 A</td>
<td>PDVL 252 /LISTHEAD FOR PHYSICAL DEVICE LIST.</td>
</tr>
<tr>
<td>578</td>
<td>02575 R 000212 A</td>
<td>SNAM 123 /ENTRY POINT FOR NAME SCAN ROUTINE.</td>
</tr>
<tr>
<td>579</td>
<td>02576 R 000337 A</td>
<td>DCRD 337 /DETACH-QUEUE REQUEST ENTRY POINT.</td>
</tr>
<tr>
<td>580</td>
<td>02577 R 000361 A</td>
<td>DMTQ 361 /DETACH &amp; EMTPY QUEUE ENTRY POINT.</td>
</tr>
<tr>
<td>581</td>
<td>02601 R 000325 A</td>
<td>ALAD 325 /ATTACH LUN &amp; DEVICE ENTRY POINT.</td>
</tr>
<tr>
<td>582</td>
<td>02601 R 000332 A</td>
<td>DLAD 332 /DETACH LUN &amp; DEVICE ENTRY POINT.</td>
</tr>
<tr>
<td>583</td>
<td>02622 R 027342 A</td>
<td>VAJX 342 /VERIFY &amp; ADJUST ROUTINE ENTRY POINT.</td>
</tr>
<tr>
<td>584</td>
<td>02623 R 000345 A</td>
<td>IDCC 345 /DECLARE I/O REQ COMPLETE ENTRY Pt.</td>
</tr>
<tr>
<td>585</td>
<td>02624 R 000248 A</td>
<td>POOL 240 /LISTHEAD FOR POOL OF EMPTY NODES.</td>
</tr>
<tr>
<td>586</td>
<td>02625 R 000127 A</td>
<td>NADD 127 /NODE ADDITION ROUTINE ENTRY POINT.</td>
</tr>
<tr>
<td>587</td>
<td>02606 R 020142 A</td>
<td>LUT 142 /LISTHEAD ADR FOR LUN TABLE.</td>
</tr>
<tr>
<td>588</td>
<td>02607 R 000171 A</td>
<td>MCRRI 171 /MCR FLAG.</td>
</tr>
</tbody>
</table>
PLOTTING 'SCOPE HANDLER SECTION.

.TITLE PLOTTING 'SCOPE HANDLER SECTION.

509 00613 R 226658 R / SEPLT  LAC*  PLBSY  / CHECK IF SECONDARY PROCESSOR
510 02611 R 744222 A  SNA  / IS BUSY.
511 02612 R 622643 R  JMP  PLERR  / YES, EXIT WITH EV = -22.
512 02613 R 162658 R  DEM*  PLBSY  / NO. SET THE BUSY FLAG.
513 02614 R 212657 A  LAC  7,X  / GET THE SUB-FUNCTION CODE, AND
514 02615 R 252664 R  DAC*  PLFLG  / PLACE IN SHARED MEMORY.
515 02616 R 742620 A  SNA  / SCLA HAS CODE 256, AND NO ARGUMENTS
516 02617 R 622636 R  JMP  SCLA  / TO PASS.
517 02618 R 542635 R  BAD  TEN  / LIN, CODE 0, HAS SIX ARGUMENTS.
518 02619 R 522641 R  JMP  LIN  / ALL OTHERS WILL PASS 4 ARGUMENTS.
519 02620 R 777774 A  LAW  -4  / LIN1
520 02621 R 042624 A  DAC*  PLCTNT  / GET THE ADDRESS, CORRECTED FOR
521 02622 R 222651 R  LAC  PLADR  / NORMAL MODE TASKS, OF THE
522 02623 R 212662 R  DAC*  SETUP  / ARGUMENT TABLE.
523 02624 R 622677 R  JMP  PLSET  / ARGUMENT TABLE.
524 02625 R 222666 R  PLRTN  DAC*  X13  / PUT THE ADDRESS, -1, IN X13.
525 02626 R 222693 R  LAC*  PLFLG  / SET UP X12 AS THE POINTER FOR THE
526 02627 R 222637 R  DAC*  X12  / SHARED CORE TABLE.
527 02628 R 222633 A  LAC*  13  / TRANSFER THE ARGUMENTS TO SHARED
528 02629 R 222639 A  DAC*  12  / MEMORY.
529 02630 R 222646 R  LAC  PLFLG  / PUT THE ADDRESS, -1, IN X13.
530 02631 R 262752 R  DAC*  X13  / PUT THE ADDRESS, -1, IN X13.
531 02632 R 226731 A  LAC*  13  / TRANSFER THE ARGUMENTS TO SHARED
532 02633 R 226737 A  DAC*  12  / MEMORY.
533 02634 R 226744 A  ISZ  PLCTNT  / ISSUE AN INTERRUPT TO THE SECONDARY
534 02635 R 622632 R  JMP  , -3  / SUB-FUNCTION LIN HAS 2 MORE ARGUMENTS
535 02636 R 226747 R  SCLA  LAC  PLINT  / ISSUE AN INTERRUPT TO THE SECONDARY
536 02637 R 102655 A  JMS  CLL10  / PROCESSOR.
537 02638 R 622645 R  JMP  SP1  / EXIT WITH EVENT VARIABLE SET TO +1.
538 02639 R 102655 A  LIN  LAW  -6  / SUB-FUNCTION LIN HAS 2 MORE ARGUMENTS
539 02640 R 226737 A  JMS  LIN1  / THAN THE OTHERS.
540 02641 R 777772 A  LAW  -6  / THAN THE OTHERS.
541 02642 R 602623 R  JMP  LIN1  / THE REQUEST NOE TO THE POOL.
542 02643 R 777756 A  PLERR  LAW  -22  / THE REQUEST NOE TO THE POOL.
543 02644 R 102643 R  JMS  SEVRN  / THE REQUEST NOE TO THE POOL.
544 02645 R 602647 R  JMP  PQ  / AND GET THE NEXT REQUEST.
545 02646 R 622645 A  PLFLG  27474+BIAS  / ADDITIONAL STORAGE AND CONSTANTS FOR THE PLOTTER 'SCOPE.
546 02647 R 622645 A  PLCTNT  START+14  / ADDITIONAL STORAGE AND CONSTANTS FOR THE PLOTTER 'SCOPE.
547 02648 R 742612 A  PLINT  0+12+BIAS  / ADDITIONAL STORAGE AND CONSTANTS FOR THE PLOTTER 'SCOPE.
548 02649 R 622645 A  PLBSY  07477+BIAS  / ADDITIONAL STORAGE AND CONSTANTS FOR THE PLOTTER 'SCOPE.
549 02650 R 00652 R  PLADR  0+12+BIAS  / ADDITIONAL STORAGE AND CONSTANTS FOR THE PLOTTER 'SCOPE.
550 02651 R 00652 R  PLADR  0+12+BIAS  / ADDITIONAL STORAGE AND CONSTANTS FOR THE PLOTTER 'SCOPE.
551 02652 R 00652 R  END  SIZE*00652  / ADDITIONAL STORAGE AND CONSTANTS FOR THE PLOTTER 'SCOPE.
REAL TIME INPUT-OUTPUT SUBROUTINE.

.TITLE REAL TIME INPUT-OUTPUT SUBROUTINE,

SUBROUTINE TO HANDLE THE REAL TIME INTERFACE BY PLACING VALUES TO
OUTPUT BY THE SECONDARY PROCESSOR INTO SHARED MEMORY AND RETRIEVING
FROM SHARED MEMORY VALUES INPUT BY THE SECONDARY PROCESSOR.

THIS SUBROUTINE HAS TWO ENTRY POINTS: RTINX FOR INPUT AND RTOUX FOR
OUTPUT. USERS NOT REQUIRING BOTH INPUT AND OUTPUT FUNCTIONS IN
THEIR TASK WOULD BE BETTER SERVED BY USING SUBROUTINE RTIN FOR INPUT
OR RTOUX FOR OUTPUT. THIS SUBROUTINE, RTIO, ELIMINATES MUCH REDUNDANT
CODE FOR THOSE REQUIRING BOTH INPUT AND OUTPUT THROUGH THE REAL TIME
INTERFACE, BUT IS LARGER THAN EITHER OF THE SINGLE FUNCTION SUB-
ROUTINES. (THIS SUBROUTINE REQUIRES MUCH LESS THAN THE SUM OF THE
TWO INDIVIDUAL SUBROUTINES BY ELIMINATION OF REDUNDANT CODE, AT THE
EXPENSE OF FEW ADDITIONAL INSTRUCTIONS.)

FOR INPUT, THE FOLLOWING CONVENTION IS USED FOR DEVICE NUMBERS:
POSITIVE DEVICE NUMBERS ARE THE ANALOG TO DIGITAL CONVERTER
CHANNELS.
DEVICE NUMBER ZERO IS THE 1 KHZ CLOCK COUNTER.
NEGATIVE DEVICE NUMBERS ARE DIGITAL INPUT GROUPS.

FOR OUTPUT, THE FOLLOWING CONVENTION IS USED FOR DEVICE NUMBERS:
POSITIVE DEVICE NUMBERS ARE THE DIGITAL TO ANALOG CONVERTERS.
NON-POSITIVE DEVICE NUMBERS (INCLUDING ZERO) ARE DIGITAL
OUTPUT GROUPS.

FOR INPUT, THE FORTRAN CALLING SEQUENCE IS
CALL RTINX (N, ICHAN, IARRY FOR ARRAY), IEV)
WITH THE PARAMETERS DEFINED AS FOLLOWS:
N N>3 NUMBER OF INPUTS REQUESTED.
N=0 ONE INPUT REQUESTED, IN FIXED POINT FORM.
N=-1 ONE ANALOG TO DIGITAL CONVERTER INPUT REQUESTED,
CONVERTED TO FLOATING POINT FORMAT (NOT VALID
FOR NON-POSITIVE DEVICE NUMBERS).
ICHAN N>0 ARRAY CONTAINING THE DEVICE NUMBERS FOR WHICH
INPUT IS DESIRED. ICHAN MUST BE INTEGER MODE
AND DIMENSIONED N OR GREATER.
IARRY, N>0 ARRAY INTO WHICH THE INPUT VALUES ARE TO BE STORRED.
VALUE OF THE SINGLE DEVICE NUMBER REQUESTED.
THIS ARRAY MUST BE INTEGER IF ANY DEVICE NUMBER
IN ICHAN IS NON-POSITIVE, IF THE ARRAY IS OF
FLOATING POINT MODE, THE ANALOG TO DIGITAL CON-
VERTER VALUES ARE CHANGED TO FLOATING POINT NUM-
BERS. IF THE ARRAY IS OF FIXED POINT MODE, THE
RAW INPUT DATA IS STORED. (SUBROUTINE ADCNV MAY
BE CALLED TO TRANSFORM RAW ANALOG TO DIGITAL CON-
VERTER DATA TO NORMAL FLOATING POINT VALUES,
THOUGH IT IS MORE EFFICIENT TO HANDLE THAT WITH
THIS SUBROUTINE WHEN TIME ALLOWS.) STORAGE IN
THE ARRAY IS IN THE SAME ORDER AS THE DEVICE
NUMBERS ARE SPECIFIED IN ICHAN.
REAL TIME INPUT-OUTPUT SUBROUTINE.

N=0  FIXED POINT VARIABLE (NOT ARRAY) WHICH IS TO BE
     SET TO THE RAW DATA INPUT WORD.
N=-1 FLOATING POINT VARIABLE (NOT ARRAY) WHICH IS TO
     BE SET TO THE FLOATING POINT REPRESENTATION OF
     THE ANALOG TO DIGITAL CONVERTER VALUE INPUT.
     (THE DEVICE NUMBER MUST BE POSITIVE.)
IEV NOT REQUIRED, BUT STRONGLY RECOMMENDED. IEV IS AN INTEGER
     VARIABLE USED TO RETURN ERROR CONDITIONS TO THE
     CALLING PROGRAM. THE POSSIBLE CODES AND MEANINGS
     ARE GIVEN BELOW.

FOR OUTPUT, THE FORTRAN CALLING SEQUENCE IS SIMILAR;
CALL RTOUX (N, ICHAN, IARRY OR ARRAY), IEV
WITH THE PARAMETERS DEFINED:
N N>0 NUMBER OF OUTPUTS REQUESTED.
N>0 ONE OUTPUT, FIXED POINT STORAGE AND OF PROPER
    FORMAT FOR THE DEVICE SELECTED.
N=-1 ONE DIGITAL TO ANALOG CONVERTER OUTPUT, TO BE
    FORMED FROM A FLOATING POINT VALUE.
ICHAN N>2 ARRAY CONTAINING THE DEVICE NUMBERS. ICHAN MUST
    BE INTEGER MODE AND DIMENSIONED N OR GREATER.
N>2, -1 INTEGER VALUE OF THE SINGLE DEVICE NUMBER REQUESTED.
IARRY, N>2 ARRAY CONTAINING THE VALUES TO BE OUTPUT ON THE
    DEVICE SPECIFIED BY THE CORRESPONDING ELEMENT
    OF ARRAY ICHAN. IARRY (ARRAY) MUST BE DIMEN-
    SIONED N OR GREATER. IF THE ARRAY IS FLOATING
    POINT MODE, THE VALUES ARE CHANGED TO THE PRO-
    PER FORMAT FOR OUTPUT TO THE DIGITAL TO ANALOG
    CONVERTERS, AND NO NON-POSITIVE DEVICE NUMBERS
    ARE PERMITTED. IF THE ARRAY IS FIXED POINT
    MODE, THE VALUES ARE ASSUMED TO BE FORMATTED
    PROPERLY FOR THE DEVICE SELECTED. DATA FOR A DIGITAL TO ANALOG
    CONVERTER WHICH REQUIRES THE MOST SIGNIFICANT BIT (AFTER THE
    SIGN BIT) BE EQUIVALENT TO 5 VOLTS OUTPUT, THE
    FORTRAN USER WISHING TO AVOID FLOATING POINT
    ARITHMETIC MAY USE THE FOLLOWING PROCEDURE:
(1) FORM AN INTEGER EQUAL TO THE DESIRED OUT-
    PUT DIVIDED BY 4.8828125 MILLIVOLTS (5/2^10),
(2) MULTIPLY BY 64 TO POSITION THE VALUE IN
    THE HIGH ORDER BIT POSITIONS AND ASSURE THAT
    NONE OF THE BITS 12-17 ARE ON, (3) ADD: 16
    MINUS THE DEVICE NUMBER. OBVIOUSLY, THE
    USER WITH FLOATING POINT VALUES WILL FIND IT
    MUCH MORE EFFICIENT TO SPECIFY ARRAY TO BE
    FLOATING POINT AND HAVE THIS SUBROUTINE PER-
    FORM THE OUTPUT FORMATTING.
N=0 INTEGER VALUE OF THE SINGLE OUTPUT REQUESTED,
N=-1 FLOATING POINT VALUE OF THE SINGLE DIGITAL TO
     ANALOG CONVERTER OUTPUT REQUESTED. THE DEVICE
     NUMBER MUST BE POSITIVE.
IEV NOT REQUIRED, BUT STRONGLY RECOMMENDED. IF AN ERROR
REAL TIME INPUT-OUTPUT SUBROUTINE.

CONDITION IS FOUND; IEV (INTEGER) IS SET TO ONE OF THE CODES LISTED BELOW.

FOR BOTH SUBROUTINE CALLS, THE FOLLOWING ERROR CODES ARE RETURNED THROUGH THE PARAMETER IEV. (IF AN ERROR CONDITION IS FOUND, NO FURTHER PROCESSING OF INPUT/OUTPUT REQUESTS IS ATTEMPTED; CONTROL RETURNS TO THE CALLING PROGRAM.) THESE CODES ARE CONSISTENT, WHERE POSSIBLE, WITH THE RSX STANDARD RETURNED EVENT VARIABLE CODES.

-1 SUCCESSFUL COMPLETION OF ALL REQUESTS,

-2 NC-1,

-3 IARRY (ARRAY) DIMENSIONED LESS THAN N, (APPLIES ONLY FOR N>2.)

-4 ICHAN DIMENSIONED LESS THAN N, OR ICHAN IS NOT A FIXED POINT ARRAY. (APPLIES ONLY FOR N>0.)

-7 IARRY (ARRAY) IS NEITHER FIXED POINT NOR FLOATING POINT SINGLE PRECISION, (CHECKED ONLY IF N>0.)

-56 (-70 OCTAL) DIGITAL TO ANALOG CONVERTER OUTPUT WORD FOUND IMPROPERLY FORMATTED (RTOUX, WITHOUT FLOATING POINT CONVERSION, ONLY).

-65 (-101 OCTAL) DEVICE NUMBER FROM ICHAN IS OUT OF RANGE, OR FLOATING POINT CONVERSION IS REQUESTED WITH A NON-POSITIVE DEVICE NUMBER.

-66 (-102 OCTAL) THE REQUESTED OUTPUT DEVICE IS PROTECTED FOR USE BY ANOTHER TASK, (APPLIES ONLY TO RTOUX.)

THIS SUBROUTINE MAY BE CALLED BY MACRO PROGRAMS (REGISTERS USED HERE ARE SAVED AND RESTORED), THOUGH THE MACRO PROGRAMMER WOULD PROBABLY BE BETTER SERVED BY INSERTING IN HIS PROGRAM SIMILAR CODE TO ISSUE THE APPROPRIATE CAL DIRECTIVE FOR HIS SPECIAL PURPOSES. MUCH OF THE CODE IN THIS SUBROUTINE IS FOR PARAMETER FETCHING AND ERROR CHECKS.

THIS SUBROUTINE IS SUPPORTED BY P. N. BARTRAM.

RTIO
REAL TIME INPUT-OUTPUT SUBROUTINE.

GLOBAL RTINX,RTOUX

00000 R 740040 A RTOUX XX ENTRY POINT FOR REQUESTING VALUES TO BE OUTPUT.
00001 R 040441 R DAC ACSAV /SAVE THE AC.
00002 R 230437 R LAC NOPIS /USE OUTPUT FUNCTIONS LATER.
00003 R 252222 R DAC RTOUX /PLACE THE RETURN ADDRESS IN THE OTHER ENTRY
00004 R 040207 R DAC RTINX / POINT.
00005 R 680114 R JMP FETC /START FETCHING PARAMETERS, AND SAVE THE MO.

00006 R 740040 A RTINX XX ENTRY POINT FOR REQUESTING INPUT.
00007 R 040441 R DAC ACSAV /SAVE THE AC.
00008 R 230440 R LAC SKPIS /USE INPUT FUNCTIONS LATER.
00009 R 040137 R DAC INSTR
00010 R 200027 R LAC RTINX /GET THE RETURN ADDRESS.
00011 R 040207 R DAC RTINX / PLACE THE RETURN ADDRESS IN THE OTHER ENTRY
00012 R 040443 R FETC DAC TEMPI /SAVE TEMPORARILY.
00013 R 520433 R AND MASK6 /MASK OUT THE STATUS BITS, LEAVING JUST THE ADDRESS
00014 R 040444 R DAC TEMP2 / (WITHIN THE BANK - 13 BIT ADDRESS),
00015 R 040433 R DAC MQ SAV
00016 R 220443 R LAC* TEMPI /GET THE MAIN PROGRAM INSTRUCTION WHICH JUMPS
00017 R 520430 R AND MASK6 / AROUND THE PARAMETER LIST, REMOVE THE OP. CODE,
00018 R 741300 A SPA /AND THEN SUBTRACT FROM THE RETURN ADDRESS TO GET
00019 R 340444 R TAO TEMP2 / THE TWO'S COMPLEMENT OF THE NUMBER OF PARAMETERS,
00020 R 240443 R DAC TEMP3 /MOVE THE ADDRESS POINTER TO THE FIRST PARAMETER,
00021 R 220443 R LAC* TEMP3 /N, AND FETCH IT.
00022 R 741133 A SPA / (CHECK FOR INDIRECT ADDRESSING.,)
00023 R 043127 R JMS NORT /YES, IS THE ARRAY LARGE ENOUGH? (LENGTH - N)
00024 R 220443 R LAC* TEMP3 /IS N > 0 ?
00025 R 040445 R DAC TEMP3 /NO, CHECK FOR SINGLE VALUE INPUT/OUTPUT.
00026 R 222443 R LAC* TEMP3 /YES, SAVE ITS COMPLEMENT.
00027 R 040445 R DAC TEMP3 /YES, MOVE TO THE NEXT PARAMETER, ICHAN.
00028 R 440443 R IS2 TEMP1 /MOVE TO THE NEXT PARAMETER OF THE ARRAY DESCRIPTOR
00029 R 222443 R LAC* TEMP3 /BLOCK, AND SAVE FOR A WHILE.
00030 R 720437 R AAC =3 /FORM THE ADDRESS OF WORD 0 OF THE ARRAY DESCRIPT.
00031 R 040444 R DAC TEMP4 /AND TEMP3 /TOR BLOCK, AND SAVE.
00032 R 222443 R LAC* TEMP3 /GET THE CONTENTS OF WORD 0: THE ADDRESS OF THE
00033 R 440443 R DAC ICHAN / FIRST ELEMENT OF THE ARRAY (MODE CODE SHOULD BE 00.)
00035 R 592333 A SPA /AND MASK6 /EXTRACT THE LENGTH.
00036 R 040445 R DAC TEMP3 /IS THE ARRAY MODE FIXED POINT? (MODE CODE 02?)
00037 R 741137 A SPA /YES.
00038 R 043127 R JMS NORT /NO, EXIT WITH ERROR CODE -4.
00039 R 342443 A TAO N /YES, IS THE ARRAY LARGE ENOUGH? (LENGTH = N
00040 R 741130 A SPA /A >= 0 ?
00041 R 040445 R DAC TEMP3 /YES, MOVE ON TO THE NEXT PARAMETER.
00042 R 222443 R LAC* TEMP3 /GET THE ADDRESS OF WORD 0 OF THE ARRAY DESCRIPT.
00043 R 440443 R DAC TEMP3 /TOR BLOCK FOR IARRAY (ARRAY), SAVE FOR A MOMENT.
RTIO REAL TIME INPUT-OUTPUT ROUTINE.

00062 R 723775 A AAC /GET THE ADDRESS OF WORD 1 OF THE ARRAY DESCRIP-
00063 R 840446 R DAC TEMP4 / FOR BLOCK, AND SAVE.
00064 R 224445 R LAT TEMP3 /GET THE FIRST ELEMENT ADDRESS FROM WORD 4, MASKING
00065 R 502424 R AND MASK1 / OUT NON-ADDRESS BITS.
00066 R 24451 R D AC I ARRAY
00067 R 22446 R LAT TEMP3 /GET THE MODE AND LENGTH, REMOVE THE MODE.
00068 R 55233 R AND MASK6 / AND COMPARE WITH THE ORIGINAL WORD.
00069 R 82443 R LAT TEMP4 / MODE IS 0E - INTEGER, HANDLE AS SUCH.
00070 R 72446 R LAT TEMP2 / COMPARE WITH THE FLOATING POINT MODE
00071 R 840446 R LAT TEMP4 / BITS ADDED.
00072 R 22443 R LAT TEMP2
00073 R 723775 A AAC /IS THE ARRAY LARGE ENOUGH? (NO. ELEMENTS - N >= 0?)
00074 R 840446 R LAT TEMP4 /NEITHER. EXIT WITH ERROR CODE -3.
00075 R 723884 A AAC /IS AN EVENT VARIABLE (ERROR RETURN VARIABLE) SPECIFIED? (ARE THERE OTHER THAN 3 PARAMETERS?)
00076 R 741283 A SPA /IS THE ARRAY LARGE ENOUGH?
00077 R 608364 R JMP ERR7 /NO. EXIT WITH ERROR CODE -3.
00078 R 741283 A SPA /NO. EXIT WITH ERROR CODE -3.
00079 R 741283 A SPA /(LENGTH - N >= 0?)
00080 R 741283 A SPA /SET PROPER CODE FOR LATER DATA TRANSFER.
00081 R 448434 R ISH TEMP1 /POSITION THE ADDRESS POINTERS FOR THE NEXT TRANSFER TO/FROM SHARED MEMORY.
00082 R 220450 R LAT ICHAN /GET THE DEVICE NUMBER.
00083 R 840420 R LAT CPB /GET THE (COMPLEMENT OF THE) NUMBER OF PARAMETERS.
00084 R 741280 A SPA /IS AN EVENT VARIABLE (ERROR RETURN VARIABLE) SPECIFIED? (ARE THERE OTHER THAN 3 PARAMETERS?)
00085 R 741280 A SPA /NO. EXIT WITH ERROR CODE -3.
00086 R 741280 A SPA /SET PROPER CODE FOR LATER DATA TRANSFER.
00087 R 448450 R LOOP /GET THE ADDRESS POINTERS FOR THE NEXT TRANSFER TO/FROM SHARED MEMORY.
00088 R 741280 A SPA /IS THE ARRAY LARGE ENOUGH?
00089 R 608364 R JMP ERR3 /NO. EXIT WITH ERROR CODE -3.
00090 R 688373 R JMP ERR3 / (NO. ELEMENTS - N >= 0?)
00091 R 741280 A SPA /NO. EXIT WITH ERROR CODE -3.
00092 R 741280 A SPA /YES, SET THE PROPER CODE FOR LATER DATA TRANSFER.
00093 R 448450 R LOOP /ENTER THE TRANSFER LOOP.
00094 R 448450 R LOOP /PUT IT IN THE CAL PARAMETER BLOCK.
00095 R 72446 R D AC CPB /IF OUTPUT, ISSUE THE CAL FUNCTION DIRECTIVE.
00096 R 72446 R D AC CPB /IS IT POSITIVE? (WAS THE CAL SUCCESSFULLY COMPLETED?)
00097 R 448450 R LOOP /IF OUTPUT REQUESTED, GO GET THE WORD FOR OUTPUT.
00098 R 448450 R LOOP /IS THE EVENT VARIABLE.
00099 R 741280 A SPA /IS IT POSITIVE? (WAS THE CAL SUCCESSFULLY COMPLETED?)
00100 R 741280 A SPA /NO, THE RETURN ERROR CODE HAS BEEN SET, EXIT.
00101 R 448450 R LOOP /YES, EXIT THE TRANSFER LOOP.
00102 R 682112 R JMP SETF /JUMP ARC UNO CODE FOR INTEGER VALUES.
00103 R 848170 R LAT NOEV /GET THE EVENT VARIABLE ADDRESS IN THE CAL PARAMETER BLOCK, AND SET THE CAL FUNCTION CODE.
00104 R 741280 A SPA /CREATE AN EVENT VARIABLE ADDRESS IF NONE GIVEN.
00105 R 840450 R LAT CPB-1 /PLACE THE EVENT VARIABLE ADDRESS IN THE CAL.
00106 R 72446 R D AC CPB /PLACE THE EVENT VARIABLE ADDRESS IN THE CAL.
00107 R 448450 R LOOP /ENTER THE TRANSFER LOOP.
00108 R 448450 R LOOP /PUT IT IN THE CAL PARAMETER BLOCK.
00109 R 72446 R D AC CPB /IF OUTPUT, ISSUE THE CAL FUNCTION DIRECTIVE.
00110 R 72446 R D AC CPB /IS IT POSITIVE? (WAS THE CAL SUCCESSFULLY COMPLETED?)
00111 R 72446 R D AC CPB /NO, THE RETURN ERROR CODE HAS BEEN SET, EXIT.
00112 R 72446 R D AC CPB /YES, EXIT THE TRANSFER LOOP.
00113 R 682112 R JMP SETF /JUMP ARC UNO CODE FOR INTEGER VALUES.
00114 R 741280 A SPA /CREATE AN EVENT VARIABLE ADDRESS IF NONE GIVEN.
00115 R 448450 R LOOP /ENTER THE TRANSFER LOOP.
00116 R 448450 R LOOP /PUT IT IN THE CAL PARAMETER BLOCK.
00117 R 72446 R D AC CPB /IF OUTPUT, ISSUE THE CAL FUNCTION DIRECTIVE.
00118 R 72446 R D AC CPB /IS IT POSITIVE? (WAS THE CAL SUCCESSFULLY COMPLETED?)
00119 R 72446 R D AC CPB /NO, THE RETURN ERROR CODE HAS BEEN SET, EXIT.
00120 R 72446 R D AC CPB /YES, EXIT THE TRANSFER LOOP.
REAL TIME INPUT-OUTPUT SUBROUTINE.

00147 R 200420 R LAC CPB-2 /*INPUT REQUEST, GET THE DEVICE NUMBER.
00150 R 744360 A SHA5EA /*IS THE DATA WORD FROM THE ANALOG TO DIGITAL CONVERTER?
00151 R 485016 R JMP ADC /*YES, GO TO A SEPARATE HANDL.
00152 R 427130 R XCT FXFLT /*NO, HAS FLOATING CONVERSION BEEN REQUESTED?
00153 R 742400 A SKP /*YES, EXIT WITH ERROR CODE -101.
00154 R 403532 R JNP ER101 /*AND, GET THE DATA WORD.
00155 R 742401 R RCR /*REMOVE THE "ALWAYS ON" BIT.
00156 R 744391 A TSEND DAG- MARRY /*DEPOSIT IN THE USER STORAGE SPECIFIED.
00157 R 424447 R JMP IS2 /*ARE THERE MORE TRANSFERS REQUESTED?
00158 R 631133 R JMP LOOP /*YES, GO DO THEM.
00159 R 204242 R RSTR LAD MOSAV /*NO, ALL FINISHED, RESTORE THE MQ REGISTER.
00160 R 655220 A LMD /*RESTORE THE AC.
00161 R 285441 R LAC ACSAY /*RESTORE THE AC.
00162 R 627227 R JNP* RTINX /*RETURN TO THE CALLING PROGRAM.

00166 R 227421 R ADC LAC CPB-3 /*ANALOG TO DIGITAL CONVERTER DATA TO BE INPUT.
00167 R 524244 R AND MASK1 /*GET THE DATA WORD, REMOVING THE STATUS BITS.
00168 R 747040 A FXFLT XX /*(SKP IF FLOATING CONVERSION, NOP IF INTEGER.)
00169 R 605159 R JMP TSEND /*INTEGER, NO CONVERSION REQUIRED.
00170 R 742410 A RTL /*FLOATING POINT CONVERSION REQUIRED, FORM THE.
00171 R 742410 A RMO /*COMPLEMENT FORM BY SHIFTING THE SIGN (BIT 3)
00172 R 647534 A LRS 4 /*TO THE LINK, THEN SHIFTING BACK, EXTENDING
00173 R 647534 A LMS /*THE SIGN THROUGH THE HIGH ORDER BIT POSITIONS,
00174 R 647534 A LAC /*IF THE VALUE IS ZERO, HANDLE AS A SPECIAL CASE.
00175 R 741420 A SNA /*IF THE VALUE IS NEGATIVE, COMPLEMENT IT,
00176 R 607225 R JMP ZEO /*MULTIPLY BY TEN (BY 5 HERE, LATER SHIFT ONE
00177 R 744420 A SEL /*PLACE EXTRA).
00178 R 744420 A TCAICLL /*PUT THE PRODUCT IN THE AC (NEVER MORE THAN 17
00179 R 653172 R MUL /*BITS), AND CLEAR THE MQ REGISTER.
00180 R 202222 R 202222 S /*LEFT JUSTIFY 640422 = NORM WITH 18 BIT SHIFT
00181 R 641292 R LACO /*COUNT, SAVE THE NORMALIZED VALUE.
00182 R 652222 R LMS /*MAKE IT POSITIVE.
00183 R 645131 A LAC /*PUT THE SIGN COUNT IN THE AC.
00184 R 653131 A TCA /*MAKE IT POSITIVE.
00185 R 732431 R AND MASK7 /*FORM THE FLOATING POINT EXPONENT.
00186 R 732762 A AAD -16 /*RETAI THE LOW ORDER HALF WORD AS THE EXPONENT.
00187 R 732762 A AND MASK5 /*AND TRANSFER IT TO USER STORAGE.
00188 R 746391 R DAG- MARRY /*STO THE MANTESSA LATER.
00189 R 424351 R ISZ MARRY /*MOVE THE MANTESSA (LESS SIGN IN MQ) 1 LEFT.
00190 R 202421 R LAC CPB-3 /*GET THE SIGN OF THE INPUT WORD INTO THE LINK.
00191 R 742410 A RTL /*GET THE ABSOLUTE VALUE OF THE MANTESSA.
00192 R 742410 A LACO /*FOR THE SIGN & MAGNITUDE MANTESSA.
00193 R 742410 A RAC /*STORE THE MANTESSA AND CONTINUE.
00194 R 621557 R JNP TSEND /*IF THE VALUE IS ZERO, SET TO ZERO BOTH
00195 R 621557 R 060451 R N0 /*THE EXPONENT ANO THE MANTESSA.
00196 R 621557 R 440451 R ISZ MARRY /*AND, GET THE DEVICE NUMBER.
00197 R 621557 R 602160 R JMP TSEND+1 /*STORE THE MANTESSA AND CONTINUE.
00198 R 621557 R 060451 R ZER /*IF THE VALUE IS ZERO, SET TO ZERO BOTH
00199 R 621557 R 440491 R ISZ MARRY /*THE EXPONENT ANO THE MANTESSA.
00200 R 621557 R 602160 R
REAL TIME INPUT-OUTPUT SUBROUTINE.

00231 R 220451 R OUTPT /OUTPUT REQUESTED, GET THE WORD TO BE OUTPUT.
00232 R 400170 R XCT /SKIP IF FLOATING CONVERSION REQUIRED, NOP IF FIXED.
00233 R 600276 R JMP /FLOATING CONVERSION REQUIRED, SAVE THE EXPONENT.
00234 R 640245 R ISE /FLOATING CONVERSION REQUIRED, SAVE THE MANTISSA.
00235 R 440251 R LAC* IARRY /GET THE WORD TO BE OUTPUT.
00236 R 220245 R ROL
00237 R 74013 A RCL /IS THE NUMBER NEGATIVE?
00238 R 741400 A ZSL
00239 R 600247 R JMP NEGA /YES, MAKE SPECIAL ARRANGEMENTS.
00240 R 747020 A R /NO, RETURN THE WORD TO THE NORMAL BIT POSITION.
00241 R 040445 R OAC Temp3 /FLOATING CONVERSION REQUIRED, SAVE THE EXPONENT.
00242 R 220451 R LAC* IARRY
00243 R 744013 A RCL /IS THE NUMBER NEGATIVE?
00244 R 741400 A ZSL
00245 R 600247 R JMP NEGA /YES, MAKE SPECIAL ARRANGEMENTS.
00246 R 747020 A R /NO, RETURN THE WORD TO THE NORMAL BIT POSITION.
00247 R 040446 R OAC Temp4 /SAVE THE MAGNITUDE.
00248 R 200435 R LAC NOPIS /PLACE NOP IN PLACE OF COMPLEMENT INSTRUCTION
00249 R 040314 R DAC COMP /LATER SINCE THE VALUE IS POSITIVE.
00250 R 607254 R JMP EXPON /JUMP AROUND THE CODE FOR NEGATIVE VALUES,
00251 R 744033 A nega R /VALUE IS NEGATIVE, RETURN THE ABSOLUTE VALUE
00252 R 742030 A SWHA /THE NUMBER IS NEGATIVE. RETURN IT TO THE LOW ORDER
00253 R 723773 A AAC -5 /POSITION, IF THE EXPONENT IS GREATER THAN -4,
00254 R 740100 A SMA /THE SATURATION VALUE IS USED.
00255 R 630332 R JMP SATR /YES, USE THE SATURATION VALUE INSTEAD.
00256 R 740001 A SMA /FOR IN-RANGE VALUES, GET THE POSITIVE SHIFT COUNT.
00257 R 240436 R SFTCT XOR RTSHF /FORM THE OP, CODE.
00258 R 047304 R DAC SHIFT /THE OP, CODE.
00259 R 200446 R LAC TEMP4 /GET THE ABSOLUTE VALUE OF THE MANTISSA BACK.
00260 R 742010 A r t l /POSITION FOR DIGITAL TO ANALOG CONVERTERS,
00261 R 742013 A r a l /IS THE VALUE TOO LARGE?
00262 R 600332 R JMP SATR /YES, USE THE SATURATION VALUE INSTEAD.
00263 R 740100 A SMA /COMPLEMENT THE VALUE IF IT WERE ORIGINALLY NEGATIVE.
00264 R 641002 A LACQ /IS THE NUMBER TOO LARGE?
00265 R 742013 A r a l /COMPLEMENT THE VALUE IF IT WERE ORIGINALLY NEGATIVE.
00266 R 640503 A LACQ /IS THE NUMBER TOO LARGE?
00267 R 741100 A SPA /COMPLEMENT THE VALUE IF IT WERE ORIGINALLY NEGATIVE.
00268 R 602323 R AND Mask3 /REMOVE EXTRA BITS.
RTIO REAL TIME INPUT-OUTPUT SUBROUTINE.

00316 R 040421 R CHAN DAC CPB*3 /STORE THE VALUE TEMPORARILY - DEVICE NUMBER NEEDED.
02317 R 200420 R LAC DAC CPB*2 /GET THE DEVICE NUMBER.
00320 R 741300 A SPA'SNA /IS THE DEVICE NUMBER FOR A DIGITAL TO ANALOG CONV.?
00321 R 602362 R JMP ER1B1 /NO, EXIT WITH ERROR CODE -101.
00322 R 723777 A AAC -1 /YES, THE DEVICE NUMBER -1 = DIGITAL TO ANALOG
02323 R 740100 A CM A / CONVERTER NUMBER. COMPLEMENT FOR THE OUTPUT
00324 R 520426 R ANO MASK4 / WORD, EXTRACT THE 4 BIT COMPLEMENT, AND MERGE
00325 R 242421 R XOR CP3*3 / WITH THE VALUE TO BE CONVERTED.
00326 R 042421 R FXOUT DAC CP9*3 /PUT THE OUTPUT WORD IN THE CAL PARAMETER BLOCK.
02327 R 622141 R JMP CALL /ISSUE THE CAL DIRECTIVE AND CONTINUE.
00328 R 142421 R ZERO OHM CPB*3 /IF THE NUMBER TO BE CONVERTED IS SMALLER THAN THE
02329 R 632 31/ R JMP CWAN / L.S.B. OF THE CONVERTER, USE ZERO INSTEAD.
00330 R 200433 R SATR LAC SAT /IF THE NUMBER IS TOO LARGE, USE THE SATURATION
02333 R 600314 R JMP COMP / VALUE INSTEAD.

00334 R 742120 A SINGL SNA /SINGLE INPUT OR OUTPUT REQUESTED, IS IT FIXED POINT?
02335 R 602343 R JMP SFIX /YES, FIXED POINT, HANDLE SEPARATELY.
00336 R 742230 A IAC /NO, IS IT FLOATING POINT?
02337 R 523372 R JMP ERR2 /NO, IT IS NEITHER. EXIT WITH ERROR CODE -2
00338 R 740032 A IC/ SINGLE INPUT OR OUTPUT REQUESTED. IS IS FIXED POINT?
02339 R 620343 R JMP SFIX /YES, FIXED POINT, HANDLE SEPARATELY.
00340 R 740230 A SNA /NO, IS IT FLOATING POINT?
02341 R 620437 R JMP SP/ SINGLE INPUT OR OUTPUT REQUESTED. IS IS FIXED POINT?
00342 R 740100 A SPA /NO, EXIT WITH ERROR CODE -7.
02343 R 230437 R JMP SFIX /YES, FLOATING POINT. PLACE PROPER CODE FOR LATER,
00344 R 242170 A DAC FXF/ FIXED POINT, PLACE PROPER CODE FOR LATER.
02345 R 777777 A LAW -1 /SET THE TRANSFER LOOP COUNTER FOR ONE TRANSFER
00346 R 442447 R DAC N / (INPUT OR OUTPUT).
02347 R 442443 R ISP TEM/ SET THE REST OF THE PARAMETER ADDRESSES.
00348 R 722443 R LACP TEM/ GET THE DEVICE NUMBER ADDRESS.
02349 R 741202 A SPA /CHECK FOR INDIRECT ADDRESSING.
00350 R 220443 R LAC TEM/ /GET THE ADDRESS FOR THE INPUT STORAGE OR FOR
02351 R 741102 A SPA /THE LOCATION OF THE OUTPUT WORD.
02352 R 742220 A SKP /CHECK FOR INDIRECT ADDRESSING.
02353 R 742437 R DAC FXFL/ GET THE ADDRESS FOR THE INPUT STORAGE OR FOR
00354 R 442443 R ISP TEM/ THE LOCATION OF THE OUTPUT WORD.
02355 R 722443 R LACP TEM/ /CHECK FOR INDIRECT ADDRESSING.
00356 R 741102 A SPA /GET ITS ADDRESS.
02357 R 742220 A SKP /CONTINUE WITH THE PROGRAM: EVENT VARIABLE NEXT.
00358 R 777777 A LAW -101 /RETURN ERROR CODE -101 IN THE EVENT VARIABLE.
02359 R 602373 R JMP ERRX /RETURN ERROR CODE -7.
02361 R 602373 R JMP ERRX /RETURN ERROR CODE -3.
00362 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02363 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00364 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02365 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00366 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02367 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00368 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02369 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00370 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02371 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00372 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02373 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00374 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02375 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00376 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02377 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00378 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02379 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00380 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02381 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00382 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02383 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00384 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02385 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00386 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02387 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00388 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02389 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.

00390 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02391 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00392 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02393 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00394 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02395 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00396 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02397 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00398 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02399 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00400 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02401 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00402 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02403 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00404 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02405 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00406 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02407 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00408 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
02409 R 602373 R JMP ERRX /STORE ERROR CODE TEMPORARILY.
00410 R 777777 A ER/ STORE ERROR CODE TEMPORARILY.
REAL TIME INPUT-OUTPUT SUBROUTINE.

PACE 9 RTIO

02401 R 72324 A AAC *+4
02402 R 22443 A DAC TEMP1
02403 R 22443 A LAC TEMP1

/(CHECK FOR INDIRECT ADDRESSING.)

02404 R 74129 A SPA NRDCT
02405 R 10412 R JMS NRDCT
02406 R 242 A DAC TEMP1

/TRANSFER THE ERROR CODE TO THE USER EVENT VARIABLE,

02407 R 64125 A LAC TEMP1
02408 R 62443 A DAC TEMP1
02409 R 62443 A LAC TEMP1
02410 R 62443 A JMP RSTR

/RESTORE REGISTERS AND EXIT.

02411 R NRDCT XX

/SUBROUTINE TO REMOVE AN EXTRA LEVEL,

02412 R 745248 A DAC TEMP3
02413 R 22445 A LAC TEMP3

/ OF INDIRECT PARAMETER ADDRESSING,

02414 R 22445 A JHP NRDCT

/ IF FOUND,

02415 R 62442 A CPB .BLOCK 4

/CAL PARAMETER BLOCK. ALL ENTRIES SET BY PROGRAM.

02416 R A NOEVA ,DSA +1

/EVENT VARIABLE ADDRESS AND LOCATION USED IF NO
02417 R 22435 R 022225 A * E EVENT VARIABLE IN THE PARAMETER LIST.

02418 R 02424 R 077777 A MASK1 077777

/ADDRESS MASK,

02419 R 02425 R 777777 A MASKS 777777

/DIGITAL TO ANALOG CONVERTER VALUE MASK,

02420 R 02426 R 022227 A MASK4 022227

/UNIT NUMBER MASK FOR DIGITAL TO ANALOG CONVERTER,

02421 R 02427 R 222227 A MASKS 222227

/EXPERTMENT MASK (FLOATING POINT NUMBERS),

02422 R 02432 R 017777 A MASK6 017777

/BANK MODE ADDRESS MASK,

02423 R 02431 R 022227 A MASK7 022227

/SHIFT COUNT MASK FOR ADC FLOATING CONVERSION,

02424 R 02432 R 022227 A FLTMD 022227

/FLOATING POINT MODE CODE IN WORD 1, DESC, BLOCK.

02425 R 02433 R 377777 A SAT 377777

/SATURATED (ABSOLUTE) VALUE FOR DIGITAL TO ANALOG CONV,

02426 R 02434 R 222231 A C31 31

/CAL FUNCTION CODE FOR INPUT (ADD 1 FOR OUTPUT),

02427 R 02435 R 742221 A NEG CMA

/COMPLEMENT CODE USED FOR CONVERTING NEGATIVE NUMBERS,

02428 R 02436 R 642558 A RTSHIP LRS O

/SHIFT INSTRUCTION, COUNT INSERTED LATER,

02429 R 02437 R 740220 A NCP IS NOP

/USED FOR OUTPUT BRANCHES AND FOR FIXED POINT BRANCHES,

02430 R 02449 R 741958 A SKPS ISK

/USED FOR INPUT BRANCHES AND FOR FLOATING PT. BRANCHES,

02431 R 02441 R 022220 A ACSAV 0

/SAVE CELL FOR THE AC,

02432 R 02442 R 002220 A MOSAV 0

/SAVE CELL FOR THE NO REGISTER,

02433 R 02443 R 002220 A TEMPS TEMP1

/TEMPORARY STORAGE - PARAMETER LIST ADDRESSES,

02434 R 02444 R 002220 A TEMPS TEMP2

/TEMPORARY STORAGE - NUMER OF PARAMETERS,

02435 R 02445 R 002220 A TEMPS TEMP3

/TEMPORARY STORAGE - PARAMETER ADDRESSES, OTHER,

02436 R 02446 R 002220 A TEMPS TEMP4

/TEMPORARY STORAGE - PARAMETER ADDRESSES, OTHER,

02437 R 02447 R 002220 A N 0

/NUMBER OF I/O REQUEST, COMPLEMENT,

02438 R 02450 R 002220 A ICHAN 0

/DEVICE NUMBER ADDRESS,

02439 R 02451 R 002220 A IARRY O

/DATA VALUE ADDRESS,

02440 R 002220 A .END

SIZ£=02452 NO ERROR LINES
REAL TIME INPUT SUBROUTINE.

.TITLE REAL TIME INPUT SUBROUTINE.

SUBROUTINE TO HANDLE THE REAL TIME INTERFACE BY RETRIEVING FROM
SHARED MEMORY VALUES INPUT BY THE SECONDARY PROCESSOR.
FOR INPUT, THE FOLLOWING CONVENTION IS USED FOR DEVICE NUMBERS:
POSITIVE DEVICE NUMBERS ARE THE ANALOG TO DIGITAL CONVERTER
CHANNELS,
DEVICE NUMBER ZERO IS THE 1 KHZ CLOCK COUNTER,
NEGATIVE DEVICE NUMBERS ARE DIGITAL INPUT GROUPS,
THE FORTRAN CALLING SEQUENCE IS
CALL RTIN (N, ICHAN, IARRY (OR ARRAY), IEV)
WITH THE PARAMETERS DEFINED AS FOLLOWS:
N > 0 NUMBER OF INPUTS REQUESTED,
N=2 ONE INPUT REQUESTED, IN FIXED POINT FORM,
N=-1 ONE ANALOG TO DIGITAL CONVERTER INPUT REQUESTED,
CONVERTED TO FLOATING POINT FORMAT (NOT VALID
FOR NON-POSITIVE DEVICE NUMBERS),
ICHAN > 0 ARRARY CONTAINING THE DEVICE NUMBERS FOR WHICH
INPUT IS DESIRED, ICHAN MUST BE INTEGER MODE
AND DIMENSIONED N OR GREATER,
IARRAY, N > 0 ARRARY INTO WHICH THE INPUT VALUES ARE TO BE STORED,
THIS ARRAY MUST BE INTEGER IF ANY DEVICE NUMBER
IN ICHAN IS NON-POSITIVE, IF THE ARRAY IS OF
FLOATING POINT MODE, THE ANALOG TO DIGITAL CON-
VERTER VALUES ARE CHANGED TO FLOATING POINT NUM-
BERS, IF THE ARRAY IS OF FIXED POINT MODE,
RAW INPUT DATA IS STORED, (SUBROUTINE ADCNV MAY
CALLED TO TRANSFORM RAW ANALOG TO DIGITAL CON-
VERTER DATA TO NORMAL FLOATING POINT VALUES,
THOUGH IT IS MORE EFFICIENT TO HANDLE THAT WITH
THIS SUBROUTINE WHEN TIME ALLOWS,) STORAGE IN
THE ARRAY IS IN THE SAME ORDER AS THE DEVICE
NUMBERS ARE SPECIFIED IN ICHAN,
N=0 FIXED POINT VARIABLE (NOT ARRAY) WHICH IS TO BE
SET TO THE RAW DATA INPUT WORD,
N=-1 FLOATING POINT VARIABLE (NOT ARRAY) WHICH IS TO
BE SET TO THE FLOATING POINT REPRESENTATION OF
THE ANALOG TO DIGITAL CONVERTER VALUE INPUT,
(I'RE THE DEVICE NUMBER MUST BE POSITIVE,) IEV NOT REQUIRED,
BUT STRONGLY RECOMMENDED, IEV IS AN INTEGER
VARIABLE USED TO RETURN ERROR CONDITIONS TO THE
CALLING PROGRAM, THE POSSIBLE CODES AND MEANINGS
ARE GIVEN BELOW,

THE FOLLOWING ERROR CODES ARE RETURNED THROUGH THE PARAMETER
IEV, IF SPECIFIED. (IF AN ERROR CONDITION IS FOUND, NO
FURTHER PROCESSING OF INPUT REQUESTS IS ATTEMPTED) CONTROL
RETURNS TO THE CALLING PROGRAM.) THESE CODES ARE CONSIDERED,
WHERE POSSIBLE, WITH THE RSX STANDARD RETURNED EVENT VARIABLE CODES.
REAL TIME INPUT SUBROUTINE.

-1 SUCCESSFUL COMPLETION OF ALL REQUESTS.
-2 NK<1
-3 IARRY (ARRAY) DIMENSIONED LESS THAN N, (APPLIES ONLY FOR N>0.)
-4 ICHAN DIMENSIONED LESS THAN N, OR ICHAN IS NOT A FIXED POINT ARRAY, (APPLIES ONLY FOR N>0)
-5 IARRY (ARRAY) IS NEITHER FIXED POINT NOR FLOATING POINT SINGLE PRECISION, (CHECKED ONLY IF N>0.)
-65 (-101 OCTAL) DEVICE NUMBER FROM ICHAN IS OUT OF RANGE, OR FLOATING POINT CONVERSION IS REQUESTED WITH A NON-POSITIVE DEVICE NUMBER.

This subroutine is supported by P. N. Bartram.

ENTRY POINT FOR REQUESTING INPUT, SAVE THE AC, SAVE THE MQ REGISTER, GET THE RETURN ADDRESS, SAVE FOR INDIRECT ADDRESSING, MASK OUT THE STATUS BITS, STORE TEMPORARILY, GET THE MAIN PROGRAM INSTRUCTION WHICH JUMPS AROUND THE PARAMETER LIST, REMOVE THE OP. CODE, THEN SUBTRACT FROM THE RETURN ADDRESS TO GET THE TWO'S COMPLEMENT OF THE NUMBER OF PARAMETERS, MOVE THE ADDRESS POINTER TO THE FIRST PARAMETER, (CHECK FOR INDIRECT ADDRESSING,) IS N > 0 ?, (NO, CHECK FOR SINGLE VALUE INPUT, YES, SAVE ITS COMPLEMENT), MOVE TO THE NEXT PARAMETER, ICHAN, (YES, SAVE ITS COMPLEMENT), GET THE ADDRESS OF WORD 4 OF THE ARRAY DESCRIPTOR BLOCK, AND SAVE FOR A WHILE, FORM THE ADDRESS OF WORD 1 OF THE ARRAY DESCRIPTOR BLOCK, AND SAVE, GET THE CONTENTS OF WORD 4 OF THE ARRAY (MODE CODE SHOULD BE 00), GET THE CONTENTS OF WORD 11 THE LENGTH (ALSO MODE), IS THE ARRAY MODE FIXED POINT? (MODE CODE 02?), GET THE ADDRESS OF WORD 4 OF THE ARRAY DESCRIPTOR BLOCK FOR IARY (ARRAY), SAVE FOR A MOMENT, GET THE ADDRESS OF WORD 1 OF THE ARRAY DESCRIPTOR BLOCK, TOR BLOCK, AND SAVE, GET THE FIRST ELEMENT ADDRESS FROM WORD 4, MASKING OUT NON-ADDRESS BITS, MODE IS 00 - INTEGER, HANDLE AS SUCH,
RTIN REAL TIME INPUT SUBROUTINE.

133 08063 R 568314 R  SAD* TEMP4 / BITS ADDED.
134 08054 R 741400 A  SKP
135 08065 R 602408 R  JMP ERR2 /NEITHER MODE. EXIT WITH ERROR CODE -7.
136 08066 R 283392 R  JMP ERR1 /NO, REMOVE THE MODE CODE AGAIN.
137 08067 R 744222 A  RCR /DIVIDE LENGTH BY 2 TO GET THE NUMBER OF ELEMENTS.
138 08070 R 342315 R  ADD N /IS THE ARRAY LARGE ENOUGH? (NO, ELEMENTS = N >= 0?)
139 08071 R 741410 A  SPA
140 08072 R 602444 R  JMP ERR3 /NO, EXIT WITH ERROR CODE -3.
141 08073 R 223306 R  LAC SKP15 /YES, SET THE PROPER CODE FOR LATER DATA TRANSFER.
142 08074 R 627110 A  JMP SETF /JUMP AROUND CODE FOR INTEGER VALUES.
143 08075 R 342315 R  INTAR TAD N /INTEGER ARRAY. IS THE ARRAY LARGE ENOUGH?
144 08076 R 741410 A  SPA / (LENGTH - N >= 0 ?)
145 08077 R 602444 R  JMP ERR3 /NO, EXIT WITH ERROR CODE -3.
146 08078 R 223306 R  LAC NOEV15 /YES, SET PROPER CODE FOR LATER DATA TRANSFER.
147 08079 R 242417 R  SETF DAC FXFLT /GET THE (COMPLEMENT OF THE) NUMBER OF PARAMETERS.
148 08080 R 223212 R  LAC TEMP2 /IS AN EVENT VARIABLE (ERROR RETURN VARIABLE)?
149 08083 R 723324 A  AAC *4 /IS AN EVENT VARIABLE SPECIFIED? (ARE THERE OTHER THAN 3 PARAMETERS?)
150 08084 R 741420 A  SPA /SPECIFIED? (ARE THERE OTHER THAN 3 PARAMETERS?)
151 08085 R 622113 R  JMP NOEV /NO EVENT VARIABLE GIVEN.
152 08086 R 442311 R  ISZ TEMPI /YES, GET ITS ADDRESS.
153 08087 R 223312 R  LAC* TEMPI /YES. SET THE PROPER CODE FOR LATER DATA TRANSFER.
154 08088 R 741410 A  SPA
155 08089 R 102265 R  JMS NORDCT
156 08090 R 741420 A  SKP
157 08091 R 202776 R  NOEV LAC NOEV15 /CREATE AN EVENT VARIABLE ADDRESS IF NONE GIVEN.
158 08092 R 202773 R  DAC CPB+1 /PLACE THE EVENT VARIABLE ADDRESS IN THE CAL.
159 08093 R 622100 R  JMP-*3 /ENTER THE TRANSFER LOOP.
160 08094 R 442316 R  LOOP ISZ ICHAN /POSITION THE ADDRESS POINTERS FOR THE NEXT
161 08095 R 442311 R  ICHAN /TRANSFER FROM SHARED MEMORY.
162 08096 R 223316 R  LAC* ICHAN /GET THE DEVICE NUMBER.
163 08097 R 442316 R  DAC CPB+2 /PUT IT IN THE CAL PARAMETER BLOCK.
164 08098 R 622172 R  CAL CPB /ISSUE THE CAL FUNCTION DIRECTIVE,
165 08099 R 222273 R  LACCPB+1 /GET THE EVENT VARIABLE.
166 08100 R 741430 A  SPAISNA /IS IT POSITIVE? (HAS THE CAL SUCCESSFULLY COMPLETED?)
167 08101 R 622141 R  JMP RSTR /NO, THE RETURN ERROR CODE HAS BEEN SET, EXIT.
168 08102 R 222274 R  LAC CPB+2 /INPUT REQUEST, GET THE DEVICE NUMBER.
169 08103 R 741430 A  SHAISZA /IS THE DATA WORD FROM THE ANALOG TO DIGITAL CONVERTER?
170 08104 R 622145 R  JMP ADC /YES, GO TO A SEPARATE HANDLER.
171 08105 R 442317 R  XCT FXFLT /YES, HAS FLOATING CONVERSION BEEN REQUESTED?
172 08106 R 202773 R  SKP
174 08108 R 223275 R  LAC CPB+3 /NO, GET THE DATA WORD.
175 08109 R 744223 A  RCR /REMOVE THE "ALWAYS ON" BIT.
176 08110 R 223117 R  TSEND ICHR DEPOT IN THE USER STORAGE SPECIFIED.
177 08111 R 442315 R  TSEND ICHR N /ARE THERE MORE TRANSFERS REQUESTED?
178 08112 R 622126 R  JMP LOOP /YES, GO DO THEM.
179 08113 R 223210 R  RSTR LAC MOSAV /NO, ALL FINISHED. RESTORE THE HG REGISTER.
180 08114 R 622220 A  LAC
181 08115 R 223207 R  LAC ACSAV /RESTORE THE AC.
182 08116 R 741430 R  JHP* RTIN /RETURN TO THE CALLING PROGRAM.
183 08117 R 202775 R  ADC CPB+3 /ANALOG TO DIGITAL CONVERTER DATA TO BE INPUT.
184 08118 R 582332 R  AND MASK1 /GET THE DATA WORD, REMOVING THE STATUS BITS.
REAL TIME INPUT SUBROUTINE.

FXFLT XX / (SKIP IF FLOATING CONVERSION, NOP IF INTEGER,)

JMP TSEND / INTEGER, NO CONVERSION REQUIRED.

RTL / FLOATING POINT CONVERSION REQUIRED. FORM THE

/ COMPLEMENT FORM BY SHIFTING THE SIGN (BIT 3)

/ TO THE LINK, THEN SHIFTING BACK, EXTENDING

/ THE SIGN THROUGH THE HIGH ORDER BIT POSITIONS.

/ IF THE VALUE IS ZERO, HANDLE AS A SPECIAL CASE.

SNR ZRO / IF THE VALUE IS NEGATIVE, COMPLEMENT IT.

JMP ZRO / MULTIPLY BY TEN (BY 5 HERE, LATER SHIF TED ONE

SAL / PLACE EXTRA).

TCA / LACO / PUT THE PRODUCT IN THE AC (NEVER MORE THAN 17

CLO / BITS), AND CLEAR THE NO REGISTER.

EAE 0422 / LEFT JUSTIFY 640422 = NORM WITH 18 BIT SHIFT

LACO / COUNT, SAVE THE NORMALIZED VALUE.

/ PUT THE SHIF T COUNT IN THE AC.

TCA / MAKE IT POSITIVE.

AND MASK7 / FORM THE FLOATING POINT EXPONENT.

A AC -45 / AND MASK5 /

DAC* LARRY / AND TRANSFER IT TO USER STORAGE.

ISZ LARRY / PREPARE TO STORE THE MANTESSA LATER.

LAC CPB+3 / MOVE THE MANTESSA (LESS SIGN IN MDI 1 LEFT.

RTL / GET THE SIGN OF THE INPUT WORD INTO THE LINK.

LACO / GET THE ABSOLUTE VALUE OF THE MANTESSA.

RAR / FOR THE SIGN & MAGNITUDE MANT ISSA.

JMP TSEND / STORE THE MANTESSA AND CONTINUE.

ISZ IARRY / IF THE VALUE IS ZERO, SET TO ZERO BOTH

ISZ IARRY / THE EXPONENT AND THE MANTESSA.

JMP TSEND+1 / SINGLE INPUT OR OUTPUT REQUESTED, IS IS FIXED POINT?

JMP SFIX / YES, FIXED POINT, HANDLE SEPARATELY.

/ NO, IS IT FLOATINGPOINT?

LAC / / NO, IT IS NEITHER. EXIT WITH ERROR CODE -2.

NAC / YES, FLOATING POINT. PLACE PROPER CODE FOR LATER.

DAC DXFLT / FIXED POINT, PLACE PROPER CODE FOR LATER.

DAC / SET THE TRANSFER LOOP COUNTER FOR ONE TRANSFER.

DAC / SET THE REST OF THE PARAMETER ADDRESSES.

DAC / GET THE DEVICE NUMBER ADDRESS.

DAC / (CHECK FOR INDIRECT ADDRESSING.)

DAC / GET THE ADDRESS FOR THE INPUT STORAGE WORD,
REAL TIME INPUT SUBROUTINE.

(CHECK FOR INDIRECT ADDRESSING.)

(RETURN ERROR CODE -101 IN THE EVENT VARIABLE.)

(RETURN ERROR CODE -7.)

(RETURN ERROR CODE -4.)

(RETURN ERROR CODE -3.)

(RETURN ERROR CODE -2.)

(STORE ERROR CODE TEMPORARILY.)

(HAS A USER EVENT VARIABLE (ERROR CODE VARIABLE)

(SPECIFIED IN THE CALL STATEMENT PARAMETER LIST?)

(YES, GET ITS ADDRESS.)

_TRANSFER THE ERROR CODE TO THE USER EVENT VARIABLE.

(RESTORE REGISTERS AND EXIT.)

(SUBROUTINE TO REMOVE AN EXTRA LEVEL

(IF PRESENT),

(USED FOR OUTPUT BRANCHES AND FOR FIXED POINT BRANCHES,

(USED FOR INPUT BRANCHES AND FOR FLOATING PT, BRANCHES,

(SAVE CELL FOR THE AC,

(SAVE CELL FOR THE MO REGISTER,

(TEMPORARY STORAGE - PARAMETER LIST ADDRESSES,
RTIN

REAL TIME INPUT SUBROUTINE.

292 00312 R 002000 A TEMP2 0
293 00313 R 002000 A TEMP3 0
294 00314 R 002000 A TEMP4 0
295 00315 R 002000 A N 0
296 00316 R 002000 A ICHAN 0
297 00317 R 002000 A IARRY 0
298
299 00000 A .END
   SIZE=00320  NO ERROR LINES
REAL TIME OUTPUT SUBROUTINE.

.TITLE REAL TIME OUTPUT SUBROUTINE.

SUBROUTINE TO HANDLE THE REAL TIME INTERFACE BY PLACING VALUES TO
OUTPUT BY THE SECONDARY PROCESSOR INTO SHARED MEMORY.

THIS SUBROUTINE HANDLES OUTPUT ONLY. USERS WITH BOTH REAL
TIME INPUT AND REAL TIME OUTPUT WOULD BE BETTER SERVED BY
USING THE SUBROUTINE RTIO (ENTRY POINT RTOUX FOR OUTPUT),
WHICH ELIMINATES MUCH REDUNDANT CODING NECESSARY IF BOTH
RTOUT AND RTIN ARE USED. THE CALLS TO SUBROUTINE RTIO ARE
THE SAME AS TO RTOUT AND RTIN EXCEPT FOR THE NAME (RTOUX
IN PLACE OF RTOUT, RTINX IN PLACE OF RTIN).

FOR OUTPUT, THE FOLLOWING CONVENTION IS USED FOR DEVICE NUMBERS:
POSITIVE DEVICE NUMBERS ARE THE DIGITAL TO ANALOG CONVERTERS,
NON-POSITIVE DEVICE NUMBERS (INCLUDING ZERO) ARE DIGITAL
OUTPUT GROUPS.

FOR OUTPUT, THE FORTRAN CALLING SEQUENCE IS AS FOLLOWS:
CALL RTOUT (N, ICHAN, IARRY (OR ARRAY), IEV)
WITH THE PARAMETERS DEFINED:
N>3 NUMBER OF OUTPUTS REQUESTED,
N<3 ONE OUTPUT, FIXED POINT STORAGE AND OF PROPER
FORMAT FOR THE DEVICE SELECTED,
N=1 ONE DIGITAL TO ANALOG CONVERTER OUTPUT, TO BE
FORMED FROM A FLOATING POINT VALUE.
ICHAN>8 ARRAY CONTAINING THE DEVICE NUMBERS. ICHAN MUST
BE INTEGER MODE AND DIMENSIONED N OR GREATER.
N>0, -1 INTEGER VALUE OF THE SINGLE DEVICE NUMBER REQUESTED.
IARRY, N>0 ARRAY CONTAINING THE VALUES TO BE OUTPUT ON THE
DEVICE SPECIFIED BY THE CORRESPONDING ELEMENT
OF ARRAY ICHAN. IARRY (ARRAY) MUST BE DIMEN-
SIONED N OR GREATER, IF THE ARRAY IS FLOATING
POINT MODE, THE VALUES ARE CHANGED TO THE PRO-
PER FORMAT FOR OUTPUT TO THE DIGITAL TO ANALOG
CONVERTERS, AND NO NON-POSITIVE DEVICE NUMBERS
ARE PERMITTED. IF THE ARRAY IS FIXED POINT
MODE, THE VALUES ARE ASSUMED TO BE FORMATTED
PROPERLY FOR THE DEVICE SELECTED, TO FORMAT
DATA FOR A DIGITAL TO ANALOG CONVERTER, WHICH
REQUIRES THE MOST SIGNIFICANT BIT (AFTER THE
SIGN BIT) BE EQUIVALENT TO 5 VOLTS OUTPUT. THE
FORTRAN USER WISHING TO AVOID FLOATING POINT
ARITHMETIC MAY USE THE FOLLOWING PROCEDURE:
(1) FORM AN INTEGER EQUAL TO THE DESIRED OUT-
PUT DIVIDED BY 4,8628125 MILLIVOLTS (5/2**13),
(2) MULTIPLY BY 64 TO POSITION THE VALUE IN
THE HIGH ORDER BIT POSITIONS AND ASSURE THAT
NONE OF THE BITS 12-17 ARE ON. 131 ADD! 16
MINUS THE DEVICE NUMBER. OBVIOUSLY, THE
USER WITH FLOATING POINT VALUES WILL FIND IT
MUCH MORE EFFICIENT TO SPECIFY ARRAY TO BE
FLOATING POINT AND HAVE THIS SUBROUTINE PER-
FORM THE OUTPUT FORMATING,

\[ N=0 \] INTEGER VALUE OF THE SINGLE OUTPUT REQUESTED,
PROPERLY FORMATTED.

\[ N=1 \] FLOATING POINT VALUE OF THE SINGLE DIGITAL TO
ANALOG CONVERTER OUTPUT REQUESTED. THE DEVICE
NUMBER MUST BE POSITIVE.

IEV NOT REQUIRED, BUT STRONGLY RECOMMENDED. IF AN ERROR
CONDITION IS FOUND, IEV (INTEGER) IS SET TO
ONE OF THE CODES LISTED BELOW.

THE FOLLOWING ERROR CONDITION CODES ARE RETURNED THROUGH THE
PARAMETER IEV, IF SPECIFIED, IF AN ERROR CONDITION IS FOUND.
NO FURTHER PROCESSING OF OUTPUT REQUESTS IS ATTEMPTED; CONTROL
RETURNS TO THE CALLING PROGRAM. THESE CODES ARE CONSISTANT.
WHERE POSSIBLE, WITH THE RSX STANDARD RETURNED EVENT VARIABLE CODES.
SUCCESSFUL COMPLETION OF ALL REQUESTS,
N<1, (APPLIES ONLY
FOR N=0.)
ICHAN DIMENSIONED LESS THAN N, OR ICHAN IS NOT A FIXED
POINT ARRAY. (APPLIES ONLY FOR N>0.)
IARRAY (ARRAY) IS NEITHER FIXED POINT NOR FLOATING POINT
SINGLE PRECISION, (CHECKED ONLY IF N>0.)
-56 (-70 OCTAL) DIGITAL TO ANALOG CONVERTER OUTPUT WORD
FOUND IMPROPERLY FORMATTED (ONLY IF FLOATING POINT
CONVERSION WAS NOT SPECIFIED),
-65 (-101 OCTAL) DEVICE NUMBER FROM ICHAN IS OUT OF RANGE,
OR FLOATING POINT CONVERSION IS REQUESTED WITH A NON-
NEGATIVE DEVICE NUMBER.
-66 (-102 OCTAL) THE REQUESTED OUTPUT DEVICE IS PROTECTED
FOR USE BY ANOTHER TASK.

THIS SUBROUTINE MAY BE CALLED BY MACRO PROGRAMS (REGISTERS USED
HERE ARE SAVED AND RESTORED), THOUGH THE MACRO PROGRAMMER WOULD
PROBABLY BE BETTER SERVED BY IMBEDDING IN HIS PROGRAM SIMILAR
CODE TO ISSUE THE APPROPRIATE CAL DIRECTIVE FOR HIS SPECIAL
PURPOSES. MUCH OF THE CODE IN THIS SUBROUTINE IS FOR PARAMETER
FETCHING AND ERROR CHECKS.

THIS SUBROUTINE IS SUPPORTED BY P. N. BARTRAM.
EJECT
ENTRY POINT FOR REQUESTING VALUES TO BE OUTPUT.

SAVE THE AC.

SAVE THE MQ REGISTER.

GET THE RETURN ADDRESS.

SAVE TEMPORARILY,

MASK OUT THE STATUS BITS, LEAVING JUST THE ADDRESS (WITHIN THE BANK - 13 BIT ADDRESS).

GET THE MAIN PROGRAM INSTRUCTION WHICH JUMPS AROUND THE PARAMETER LIST, REMOVE THE OP. CODE,

THEN SUBTRACT FROM THE RETURN ADDRESS TO GET THE TWO'S COMPLEMENT OF THE NUMBER OF PARAMETERS.

MOVE THE ADDRESS POINTER TO THE FIRST PARAMETER,

LAC* TEMP1 / N, AND FETCH IT.

IS N > 0 ?

IS N > 0 ?

CHECK FOR INDIRECT ADDRESSING.

YES, SAVE ITS COMPLEMENT.

NO, CHECK FOR SINGLE VALUE INPUT/OUTPUT.

YES, SAVE ITS COMPLEMENT.

NO, EXIT WITH ERROR CODE -4.

YES, IS THE ARRAY LARGE ENOUGH? (LENGTH - N > 0 ?)

YES, EXIT WITH ERROR CODE -4.

YES, MOVE ON TO THE NEXT PARAMETER.

/YES. MOVE ON TO THE NEXT PARAMETER.

/YES.

GET THE ADDRESS OF WORD 4 OF THE ARRAY DESCRIPTOR BLOCK, AND SAVE FOR A WHILE.

GET THE ADDRESS OF WORD 1 OF THE ARRAY DESCRIPTOR BLOCK, AND SAVE.

GET THE CONTENTS OF WORD 4: THE ADDRESS OF THE FIRST ELEMENT OF THE ARRAY (MODE CODE SHOULD BE 00.)

GET THE CONTENTS OF WORD 1: THE LENGTH (ALSO MODE).

IS THE ARRAY MODE FIXED POINT? (MODE CODE 00?)

MODE IS 00 - INTEGER, HANDLE AS SUCH.

COMPARE WITH THE FLOATING MODE SITS
RTOUT REAL TIME OUTPUT SUBROUTINE.

153 0066 R 962845 R SAQ TEMP4 / ADDED,

154 0066 R 741260 A SPK

155 0066 R 622265 R JMP

156 0066 R 523331 R AND

157 0066 R 744222 A RCR

158 0066 R 743246 R TEMP

159 0066 R 741170 A SPA

160 0066 R 52727 R JMP

161 0066 R 622271 R JMP

162 0066 R 223356 A LAC

163 0066 R 74346 R JMP

164 0066 R 741120 A SPA

165 0066 R 5277 R JMP

166 0066 R 222337 R LAC

167 0066 R 241273 R SETF

168 0066 R 207234 R LAC

169 0066 R 72304 A AAC

170 0066 R 741200 A SNA

171 0066 R 62213 R JMP

172 0066 R 442342 R ISE

173 0066 R 222342 R LAC

174 0066 R 741120 A SPA

175 0066 R 10733 R JMS

176 0066 R 741200 A EXP

177 0066 R 223333 R NOAA

178 0066 R 622122 R DAC

179 0066 R 62120 R JNP

180 0066 R 442347 R LOOP

181 0066 R 442350 R IS2

182 0066 R 222347 R LAC

183 0066 R 242351 R DAC

184 0066 R 222352 R LAC

185 0066 R 742243 A FXLT

186 0066 R 742244 A DAC

187 0066 R 443533 R ISE

188 0066 R 742353 R ISE

189 0066 R 222350 R LAC

190 0066 R 744210 A RCL

191 0066 R 741420 A SEL

192 0066 R 62137 R JMP

193 0066 R 742223 A RAR

194 0066 R 622349 R DAC

195 0066 R 222336 R LAC

196 0066 R 740224 A DAC

197 0066 R 740224 A JMP

198 0066 R 523332 R NOAA

199 0066 R 744022 A RCR

200 0066 R 222353 R DAC

201 0066 R 742344 A DAC

202 0066 R 742344 A DAC

203 0066 R 503330 R AND

204 0066 R 742330 A SWHA

205 0066 R 742322 A SMA

RTOUT REAL TIME OUTPUT SUBROUTINE.
REAL TIME OUTPUT SUBROUTINE.

276 22192 A 724332 R JMP PEXP /EXPONENT IS POSITIVE, HANDLE SEPARATELY.
277 22151 R 742233 A SMHA /EXPONENT IS NEGATIVE, RETURN IT TO THE LOW ORDER HALF OF THE WORD, COMPLEMENT, AND SAVE.
223 33154 R 343344 R DAC TEMP3
222 33153 R 537333 R AND MASK5
221 33152 R 743331 A SWHA /EXPONENT IS NEGATIVE, RETURN IT TO THE LOW ORDER HALF OF THE WORD, COMPLEMENT, AND SAVE.
220 33151 R 632231 R JMP EERO /EXPONENT IS GREATER THAN +4, USE SATURATION.
219 33150 R 723373 A AAC -5 /POSITION, IF THE EXPONENT IS GREATER THAN +4,
218 723773 A AAC -10 /IF IT IS LESS THAN -8, 2ER0 IS OUTPUT (NUMBER WOULD BE LESS THAN THE PRECISION OF THE DIGITAL TO ANALOG CONVERTER).
217 723773 A AAC -1 /YES, THE DEVICE NUMBER -1 = DIGITAL TO ANALOG CONVERTER NUMBER, COMPLEMENT FOR THE OUTPUT.
216 723773 A AAC /THE MANTESSA TO A NUMBER SCALED WITH THE BINARY POINT BETWEEN BITS 4 AND 5,
215 723773 A AAC /THE SATEURATION VALUE IS USED.
214 723773 A AAC /THE SATURATION VALUE IS USED.
213 723773 A AAC /THE CP.code.
212 723773 A AAC /THE SATURATION VALUE IS USED.
211 723773 A AAC /THE CP.code.
210 723773 A AAC /THE SATURATION VALUE IS USED.
REAL TIME OUTPUT SUBROUTINE.

* VALUE INSTEAD.

* SINGLE OUTPUT REQUESTED, IS IT FIXED POINT?

* YES, FIXED POINT. HANDLE SEPARATELY.

* NO, IS IT FLOATING POINT?

* FIXED POINT. PLACE PROPER CODE FOR LATER.

* NO, IT IS NEITHER. EXIT WITH ERROR CODE -2.

* SET THE TRANSFER LOOP COUNTER FOR ONE TRANSFER.

* SET THE REST OF THE PARAMETER ADDRESSES, CHECK FOR INDIRECT ADDRESSING.)

* GET THE ADDRESS FOR THE LOCATION OF THE

* CONTINUE WITH THE PROGRAM EVENT VARIABLE NEXT.

* RETURN ERROR CODE -101 IN THE USER EVENT VARIABLE.

* RETURN ERROR CODE -7.

* RETURN ERROR CODE -4.

* RETURN ERROR CODE -3.

* RETURN ERROR CODE -2.

* STORE ERROR CODE TEMPORARILY.

* WAS A USER EVENT VARIABLE (ERROR CODE VARIABLE)

* SPECIFIED IN THE CALL STATEMENT PARAMETER LIST?

* NO, THE USER CAN'T BE TOLD WHAT HE DID WRONG, EXIT.

* YES, GET ITS ADDRESS.

* (CHECK FOR INDIRECT ADDRESSING.)

* (CHECK FOR INDIRECT ADDRESSING.)

* TRANSFER THE ERROR CODE TO THE USER EVENT VARIABLE.

* RESTORE REGISTERS AND EXIT,

* SUBROUTINE TO REMOVE AN EXTRA LEVEL

* OF INDIRECT PARAMETER ADDRESSING,
REAL TIME OUTPUT SUBROUTINE

0316 R 623313 R / JMP* NORCT
0317 R 000032 A CPB 32 / CAL PARAMETER BLOCK, ALL OTHER ENTRIES SET BY
0318 R 00320 A / BLOCK 3 / THE PROGRAM,
0319 R 00323 R 022324 A NOEVA 0SA */1 / EVENT VARIABLE ADDRESS AND LOCATION USED IF NO
0320 R 00324 R 022200 A 0 / EVENT VARIABLE IN THE PARAMETER LIST.
0321 R 00325 R 077777 A MASK1 077777 / ADDRESS MASK,
0322 R 00326 R 077777 A MASK3 077777 / DIGITAL TO ANALOG CONVERTER VALUE MASK.
0323 R 00327 R 022217 A MASK4 022217 / UNIT NUMBER MASK FOR DIGITAL TO ANALOG CONVERTER.
0324 R 00328 R 022277 A MASK5 022277 / EXPONENT MASK (FLOATING POINT NUMBERS).
0325 R 00329 R 017777 A MASK6 017777 / BANC MODE ADDRESS MASK.
0326 R 00330 R 022222 A FLTMD 022222 / FLOATING POINT MODE CODE IN WORD 1, OESCH. BLOCK,
0327 R 00331 R 022277 A SAT 022277 / SATURATED (ABSOLUTE) VALUE FOR DIGITAL TO ANALOG CONV,
0328 R 00332 R 022220 A NEG CMA / COMPLEMENT CODE USED FOR CONVERTING NEGATIVE NUMBERS.
0329 R 00333 R 042200 A RTSHF LRS 0 / SHIFT INSTRUCTION, COUNT INSERTED LATER.
0330 R 00334 R 042270 A FLOAT NOP / FOR FLOATING CONVERSION BRANCHES.
0331 R 00335 R 052216 R FIX JMP FXOUT / FOR FIXED POINT BRANCHES.
0332 /
0333 R 00340 R 022222 A ACSAV 0 / SAVE CELL FOR THE AC.
0334 R 00341 R 022222 A MOSAV 0 / SAVE CELL FOR THE MD REGISTER.
0335 R 00342 R 022222 A TEMP1 0 / TEMPORARY STORAGE - PARAMETER LIST ADDRESSES.
0336 R 00343 R 022222 A TEMP2 0 / TEMPORARY STORAGE - NUMBER OF PARAMETERS.
0337 R 00344 R 022222 A TEMP3 0 / TEMPORARY STORAGE - PARAMETER ADDRESSES, OTHER.
0338 R 00345 R 022222 A TEMP4 0 / TEMPORARY STORAGE - PARAMETER ADDRESSES, OTHER.
0339 R 00346 R 022222 A N 0 / NUMBER OF OUTPUT REQUESTS, COMPLEMENT.
0340 R 00347 R 022200 A ICHAN 0 / DEVICE NUMBER ADDRESS.
0341 R 00348 R 032200 A IARRY 0 / DATA VALUE ADDRESS.
0342 /
0343 R 032270 A / END
SIZE=3035% NO ERROR LINES
SUBROUTINE TO SET UP CPB FOR CAL DIR, 33.

.TITLE SUBROUTINE TO SET UP CPB FOR CAL DIR, 33.

/ THIS SUBROUTINE TRANSFERS THE ARGUMENTS FROM THE PARAMETER LIST TO A CAL PARAMETER BLOCK (CPB), THEN ISSUES THE CAL DIRECTIVE 33 TO PLACE AN ENTRY IN THE TABLE FOR AUTOMATIC FILLING OF THE USER BUFFER (A DEFINED ARRAY IN THE CALLING PROGRAM). THIS ROUTINE MAY BE CALLED BY A FORTRAN OR BY A MACRO PROGRAM, THOUGH THE MACRO PROGRAMMER IS MUCH BETTER SERVED BY SETTING UP THE CPB AND ISSUING THE CAL DIRECTIVE HIMSELF.

/ FORTRAN CALLING SEQUENCE:
/ CALL RTSET (N, IFREQ, IARRY, IEV)
/ ALL ARGUMENTS ARE ASSUMED TO BE INTEGER; IEV IS OPTIONAL, THOUGH STRONGLY RECOMMENDED TO ALLOW THE USER TO KNOW WHEN THE CALL WAS NOT SUCCESSFULLY COMPLETED. PARAMETERS:
/ N DEVICE NUMBER; N>2 - ANALOG TO DIGITAL CONVERTER CHANNEL NUMBER,
/ N=2 - CLOCK COUNTER,
/ N<2 - DIGITAL INPUT GROUP NUMBER.
/ IFREQ FREQUENCY CODE FOR AUTOMATIC PLACING OF VALUES:
/ 1 - 1 KHZ RATE,
/ 2 - 100 KHz RATE,
/ 3 - 10 Hz RATE.
/ IARRY AN ARRAY IN THE CALLING PROGRAM INTO WHICH THE VALUE OF THE SPECIFIED DEVICE WILL BE PLACED AT THE RATE OF ONE VALUE PER INTERVAL SPECIFIED BY IFREQ. (CIRCULAR FILLING OF BUFFER.)
/ IEV EVENT VARIABLE FOR RETURNING ERROR CONDITIONS:
/ -203 CAL NOT TASK ISSUED (SHOULD NOT OCCUR),
/ -201 DEVICE NUMBER, N, IS NOT IN RANGE.
/ -70 FREQUENCY CODE, IFREQ, IS NOT 1, 2, OR 3.
/ -30 USER BUFFER, IARRY, IS NOT CONTAINED WITHIN THIS PARTITION,
/ -15 TABLE FOR SELECTED FREQUENCY IS FULL.

/ THE CAL PARAMETER BLOCK USED IS
/ 0 33 /CAL FUNCTION CODE.
/ 1 IEV /EVENT VARIABLE ADDRESS.
/ 2 IN3 /VALUE OF DEVICE NUMBER, N.
/ 3 IARRY(1) /ADDRESS OF FIRST ELEMENT OF THE USER BUFFER IARRY.
/ 4 LENGTH /VALUE OF THE LENGTH OF THE BUFFER FOUND FROM DESCRIPTOR BLOCK SET /UP BY THE FORTRAN DECLARATION "DIMENSION IARRY(LENGTH)".
/ 5 IFREQ /VALUE OF THE FREQUENCY CODE, IFREQ.

/ THIS SUBROUTINE IS SUPPORTED BY P. N. BARTRAM.

T H E  C A L  P A R A M E T E R  B L O C K  U S E D  I S
/ 0 33 /C A L  F U N C T I O N  C O D E .
/ 1 IEV /E V E N T  V A R I A B L E  A D D R E S S .
/ 3 IARR Y (1 ) /A D D R E S S  O F  F I R S T  E L E M E N T  O F  T H E U S E R  B U F F E R  I A R R Y ,
/ 4 L E N G T H /V A L U E  O F  T H E  L E N G T H  O F  T H E  B U F F E R
/ F O U N D  F R O M  D E S C R I P T O R  B L O C K  S E T
/ /U P  B Y  T H E  F O R T R A N  D E C L A R A T I O N
/ "D I M E N S I O N  I A R R Y ( L E N G T H ) ."
RTSET

SUBROUTINE TO SET UP CPB FOR CAL DIR, 33.


.EJECT
RTSET

SUBROUTINE TO SET UP CPB FOR CAL DIR, 33.

.globl rtset

entry point.

save the ac.

get the return address.

mask out the status bits, leaving the address.

get the main program instruction which jumps

around the parameter list, and remove the

op. code, then subtract from the return

address to get the complement of the number

of parameters.

move the address pointer to the first parameter.

fetch the address of n, the first parameter.

(check for indirect addressing.)

get the value of n, and transfer to the cpb.

get the value of ifreq, and place in the cpb.

get the address of the third parameter.

get the address of the word 4 of the descriptor

block for array iarray (third parameter).

get the value of word 4, extract the address

of the first element of the user buffer

and store in the cpb.

get the address, then value of word 1 of

the array descriptor block.

is an event variable. iev. included in the

calling sequence?

issue the cal directive. returned event

variable is in the calling program, if

one given in the parameter list.

restore the ac.
SUBROUTINE TO SET UP CPB FOR CAL DIR. 33.

JMP* RTSET /RETURN TO THE CALLING PROGRAM.

SUBROUTINE TO REMOVE AN EXTRA LEVEL OF INDIRECT PARAMETER ADDRESSING.

NORCT XX

DAC TEMP3

LAC* TEMP3

JHP* NORCT

CAL PARAMETER BLOCK FOR THE CAL DIRECTIVE 33 ISSUED.

CPB33 33 /CAL DIRECTIVE.

XX /EVENT VARIABLE ADDRESS, IF ONE SPECIFIED.

XX /DEVICE NUMBER, N.

XX /ADDRESS OF FIRST ELEMENT OF BUFFER, IARRY(1).

XX /LENGTH OF THE BUFFER IARRY.

XX /FREQUENCY CODE, IFREQ.

MASK1 077777 /MASK TO EXTRACT ADDRESSES.

MASK2 017777 /MASK TO EXTRACT THE ARRAY LENGTH.

ACS1V 0 /SAVE CELL FOR THE AC.

TEMP1 0 /PARAMETER LIST ADDRESS POINTER.

TEMP2 0 /NUMBER OF PARAMETERS, COMPLEMENTED.

TEMP3 0 /ADDRESS OF PARAMETER TO BE FETCHED.

END

SIZE=0122 NO ERROR LINES
SUBROUTINE TO ISSUE CAL DIRECTIVE 34.

.TITLE SUBROUTINE TO ISSUE CAL DIRECTIVE 34.

/* THIS SUBROUTINE TRANSFERS THE ARGUMENT FROM THE PARAMETER LIST OF
THE CALL STATEMENT TO A CAL PARAMETER BLOCK (CPB), THEN ISSUES THE
CALL DIRECTIVE 34 TO RESET THE ADDRESS POINTER TO THE BEGINNING OF
THE ARRAY FOR EACH ENTRY IN THE REAL-TIME INPUT REQUEST TABLE USED
BY THIS TASK. THIS ROUTINE MAY BE CALLED BY A FORTRAN OR BY A
MACRO PROGRAM, THOUGH THE MACRO PROGRAMMER IS MUCH BETTER SERVED
BY SETTING UP THE CPB AND ISSUING THE CAL INSTRUCTION HIMSELF,

FORTRAN CALLING SEQUENCE:
CALL RTRST (IEV)

THE ARGUMENT, IEV (INTEGER), IS MANDATORY FOR EXECUTION OF THIS
SUBROUTINE. POSSIBLE VALUES RETURNED TO THE CALLING PROGRAM AS
IEV ARE
-233 CAL NOT TASK ISSUED (SHOULD NEVER OCCUR IN FORTRAN),
-105 NO ENTRIES IN THE REAL-TIME REQUEST TABLE WERE FOUND
FOR THIS TASK,
+N DIRECTIVE SUCCESSFULLY COMPLETED, N IS THE NUMBER
OF (1 KHZ) CLOCK PULSES FROM THE TIME OF
COMPLETION THAT AN ENTRY IN THE 10 HZ REQUEST
TABLE WILL BE SERVICED, (N/10 FOR THE NEXT CLOCK
PULSE.) THIS IF K IS THE REMAINDER FOR N/10
(BASE 10), THEN THE N-TH VALUE IN A 1 KHZ ARRAY,
THE K-TH VALUE IN A 100 HZ ARRAY, AND THE FIRST
VALUE IN A 10 HZ ARRAY WERE STORED ON THE SAME
CLOCK PULSE.

THIS SUBROUTINE IS SUPPORTED BY P. N. BARTMAN.


.EJECT
SUBROUTINE TO ISSUE CAL DIRECTIVE 34.

00000 R 740040 A RTRST              /ENTRY POINT.
00001 R 242226 R DAD ACSAV           /SAVE THE AC.
00002 R 227228 R LAC RTRST           /GET THE RETURN ADDRESS.
00003 R 242225 R DAD TEMP1           /GET THE JUMP INSTRUCTION BEFORE THE PARAMETER
00004 R 227225 R LAC* TEMP1          /LIST, EXTRACT THE OP. CODE.
00005 R 507330 R AND *MASK1         /IS IT REALLY A JUMP INSTRUCTION?
00006 R 547237 R SAD JMPIN          /NO, EXIT DOING NOTHING.
00007 R 741222 A SKP                /YES, GET THE ADDRESS OF THE PARAMETER.
00008 R 622217 R JMP RST1           /GET THE PARAMETER -- THE ADDRESS OF IEV.
00009 R 442225 R ISP TEMP1          /(CHECK FOR INDIRECT ADDRESSING.)
00010 R 227225 R LAC* TEMP1
00011 R 741133 A SPA                /ISSUE CAL DIRECTIVE 34. IEV IS RETURNED
00012 R 12221 R JMS NDRCT
00013 R 247332 R DAD CPB+1          /STORE THE ADDRESS IN THE CPB.
00014 R 227231 R CAL CPB            /ISSUE CAL DIRECTIVE 34. IEV IS RETURNED
00015 R 222226 R RST1 LAC ACSAV     /DIRECTLY TO THE CALLING PROGRAM.
00016 R 622222 R JMP* RTRST         /RESTORE THE AC AND RETURN TO THE CALLING PROG.
00017 R 742040 A NDRCT XX
00018 R 042225 R DAD TEMP1
00019 R 227225 R LAC* TEMP1
00020 R 62221 R JMP* NDRCT
00021 R 742042 A NDRCT XX
00022 R 042225 R DAD TEMP1
00023 R 227225 R LAC* TEMP1
00024 R 62221 R JMP* NDRCT
00025 R 02220 A TEMP1 0              /TEMPORARY STORAGE.
00026 R 02220 A ACSAV 0              /SAVE CELL FOR THE AC.
00027 R 622030 A JMP* JMP         /JUMP INSTRUCTION.
00028 R 762220 A MASK1 76000        /OP. CODE MASK (BANK MODE).
00029 R 207334 A CPB 34             /CAL PARAMETER BLOCK FOR CAL DIRECTIVE 34.
00030 R 742240 A XX                  /EVENT VARIABLE ADDRESS IS SET BY THIS ROUTINE.
00031 R 02223 A .END
00032 SIZE=02233 NO ERROR LINES

/ SUBROUTINE TO REMOVE AN EXTRA LEVEL OF INDIRECT PARAMETER ADDRESSING.

/ CONSTANTS AND STORAGE.

/ TEMPORARY STORAGE.

/ SAVE CELL FOR THE AC.

/ JUMP INSTRUCTION.

/ OP. CODE MASK (BANK MODE).

/ CAL PARAMETER BLOCK FOR CAL DIRECTIVE 34.

/ EVENT VARIABLE ADDRESS IS SET BY THIS ROUTINE.

/ THAT'S ALL THERE IS TO IT!
ANALOG TO DIGITAL CONVERTER DATA CONVERSION.

*TITLE ANALOG TO DIGITAL CONVERTER DATA CONVERSION.

SUBROUTINE TO CONVERT THE RAW DATA WORD FROM THE ANALOG TO DIGITAL CONVERTER. RETURNED IS A FLOATING POINT NUMBER BETWEEN -10.000 AND +9.9994 CORRESPONDING TO THE 15 BIT RAW DATA WORD.

THIS SUBROUTINE IS DESIGNED FOR USE WITH DATA GATHERED QUICKLY AND STORED BEFORE CONVERSION TO A USEFUL FORM. USERS WITH TIME AND STORAGE (FLOATING POINT VALUES REQUIRE TWO WORDS EACH; RAW DATA JUST ONE WORD EACH) WILL ACHIEVE BETTER OVERALL EFFICIENCY BY USING THE CONVERSION OPTION IN THE DATA GATHERING SUBROUTINE RTIN COR RTINXJ.

AN ALTERNATE ENTRY POINT, IADCV, MAY BE USED BY THOSE WISHING TO HANDLE ANALOG TO DIGITAL CONVERTER VALUES AS SCALED INTEGERS. THE VALUE OF EACH UNIT IS 0.0226103515625 VOLTS.

FORTRAN CALLING SEQUENCE:
   CALL ADCNV (IVAL, VALUE)
   CALL IADCV (IVAL, IVSCL)

IVAL RAW DATA WORD, INTEGER MODE.
VALUE VALUE OF THE DATA AS RETURNED BY THIS SUBROUTINE, FLOATING POINT MODE.
IVSCL SCALED INTEGER VALUE OF THE DATA AS RETURNED BY THIS SUBROUTINE.

NO CHECKS ARE PERFORMED BY THIS SUBROUTINE FOR THE CORRECTNESS OF THE USER CALLING SEQUENCE. THE RESULT IS UNPREDICTABLE, THOUGH PROBABLY FATAL, IF AN INCORRECT PARAMETER MODE OR TOO FEW PARAMETERS ARE USED.

THIS SUBROUTINE MAY BE CALLED BY MACRO ASSEMBLER PROGRAMS, THOUGH THE ASSEMBLER PROGRAMMER WOULD PROBABLY BE BETTER SERVED BY INCLUDING SIMILAR CODE WITHIN HIS PROGRAM TO AVOID THE PARAMETER FETCHING. (REGISTERS USED HERE ARE SAVED AND RESTORED.)

THIS SUBROUTINE IS SUPPORTED BY P. N. BARTRAM.

ADCNV ANALOG TO DIGITAL CONVERTER DATA CONVERSION.

ADATA ADCNV/ENTRY POINT.

GLOBL ADCNV, IADC1

OCT

ENTRY POINT.

SAVE THE ALU.

POST FLOATING CONVERSION FLAG.

SAVE THE MQ REGISTER.

GET THE ADDRESS OF THE RAW DATA WORD.

SAVE THE AC, THE AC.

POST FLOATING CONVERSION FLAG.

GET THE ADDRESS OF THE STORAGE CELL WHERE THE CONVERTED VALUE IS TO BE STORED.

CHECK FOR INDIRECTLY ADDRESSED PARAMETER.

GET THE ADDRESS OF THE RAH DATA WORD.

GET THE RAH DATA WORD.

THE VALUE IS TO BE STORED.

CHECK FOR INDIRECTLY ADDRESSED PARAMETER, IF PRESENT.

PARAMETER, IF PRESENT.

GET THE STORAGE CELL WHERE THE CONVERTED VALUE IS TO BE STORED.

GET THE RAH DATA WORD.

IS THE VALUE ZERO?

YES, HANDLE AS A SPECIAL CASE.

IS THE VALUE NEGATIVE?

YES, COMPLEMENT IT AND CLEAR THE LINK.

MULTIPLY BY 5 (SHIFT LATER FOR OVERALL MULTIPLICATION BY 10). THE PRODUCT WILL NOT EXCEED 17 BITS.

GET THE PRODUCT INTO THE AC, CLEAR THE STATUS BITS.

IS THE VALUE ZERO?

YES, HANDLE AS A SPECIAL CASE.

IS THE VALUE NEGATIVE?

YES, COMPLEMENT IT AND CLEAR THE LINK.

MULTIPLY BY 5 (SHIFT LATER FOR OVERALL MULTIPLICATION BY 10). THE PRODUCT WILL NOT EXCEED 17 BITS.

GET THE PRODUCT INTO THE AC, CLEAR THE STATUS BITS.

IS THE VALUE ZERO?

YES, HANDLE AS A SPECIAL CASE.

IS THE VALUE NEGATIVE?

YES, COMPLEMENT IT AND CLEAR THE LINK.

MULTIPLY BY 5 (SHIFT LATER FOR OVERALL MULTIPLICATION BY 10). THE PRODUCT WILL NOT EXCEED 17 BITS.

GET THE PRODUCT INTO THE AC, CLEAR THE STATUS BITS.

IS THE VALUE ZERO?

YES, HANDLE AS A SPECIAL CASE.

IS THE VALUE NEGATIVE?

YES, COMPLEMENT IT AND CLEAR THE LINK.

MULTIPLY BY 5 (SHIFT LATER FOR OVERALL MULTIPLICATION BY 10). THE PRODUCT WILL NOT EXCEED 17 BITS.

GET THE PRODUCT INTO THE AC, CLEAR THE STATUS BITS.

IS THE VALUE ZERO?

YES, HANDLE AS A SPECIAL CASE.

IS THE VALUE NEGATIVE?

YES, COMPLEMENT IT AND CLEAR THE LINK.

MULTIPLY BY 5 (SHIFT LATER FOR OVERALL MULTIPLICATION BY 10). THE PRODUCT WILL NOT EXCEED 17 BITS.

GET THE PRODUCT INTO THE AC, CLEAR THE STATUS BITS.

IS THE VALUE ZERO?

YES, HANDLE AS A SPECIAL CASE.

IS THE VALUE NEGATIVE?

YES, COMPLEMENT IT AND CLEAR THE LINK.
AOCNV ANALOG TO DIGITAL CONVERTER DATA CONVERSION.

LHQ /RESTORE THE AC AND THE MO REGISTERS.

JMP* AOCNV /RETURN TO THE CALLING PROGRAM.

ZERO DEM* VALUE /SPECIAL CASE - VALUE IS ZERO, ZERO

ISE VALUE / BOTH THE EXPONENT AND THE MANTISSA.

DEN* VALUE

JMP RSTR /RESTORE REGISTERS AND RETURN.

/ ALTERNATE ENTRY FOR NO FLOATING CONVERSION.

IADCV XX /ENTRY POINT.

DAC ACSAV /SAVE THE AC.

LAC IADCV /RETURN THROUGH AOCNV.

DAC AOCNV /RETURN THROUGH AOCNV.

LAC IADCN /POST FLAG FOR NO FLOATING CONVERSION.

JMP IADCI

/ SUBROUTINE TO REMOVE AN EXTRA LEVEL OF INDIRECT PARAMETER ADDRESSING.

IADCN XX

DAC TEMPN

LAC* TEMPN

JMP* IADCI

/ CONSTANTS AND STORAGE CELLS.

00077 A 00077 A /FOR EXTRACTING THE POSITIVE SHIFT COUNT.

00077 A 00077 A /FOR EXTRACTING THE EXPONENT.

00077 A 00077 A /INSTRUCTION FOR XCTIN FOR FLOATING.

IADCN JHP RSTR-1 /INSTRUCTION FOR XCTIN FOR INTEGER.

ACSAV A /SAVE CELL FOR THE AC.

MSAV A /SAVE CELL FOR THE MO REGISTER.

VALUE A /ADDRESS FOR THE CONVERTED VALUE.

IVAL A /ADDRESS OF THE RAW DATA VALUE.

TEMP A /TEMPORARY STORAGE.

TEMP A /MORE TEMPORARY STORAGE.

NO ERROR LINES

PAGE 3
TITLE GRAPHICS 'SCOPE HANDLER SUBROUTINE.

SUBROUTINE TO CALL THE GRAPHICS 'SCOPE (KS0) HANDLER.

THIS IS A FORTRAN CALLABLE SUBROUTINE; NO REGISTERS ARE SAVED.
THE MACRO ASSEMBLER PROGRAMMER WOULD BE BETTER SERVED BY
IMBEDDING SIMILAR CODE WITHIN HIS PROGRAM.

THE CALLING SEQUENCES ARE AS FOLLOWS:

FOR THE CLEAR 'SCOPE COMMAND -
CALL SCL (LUN, IEV)

FOR THE POINT COMMAND -
CALL PNT (LUN, IEV, IX1, IY1, INTNS, MDE)

FOR THE HORIZONTAL LINE COMMAND -
CALL HZ (LUN, IEV, IX1, IX2, IY1, INTNS)

FOR THE VERTICAL LINE COMMAND -
CALL VZ (LUN, IEV, IX1, IY1, IY2, INTNS)

FOR THE LINE COMMAND -
CALL LIN (LUN, IEV, IX1, IY1, IDELX, IDELY, IS, INTNS)

WHERE
LUN = LOGICAL UNIT NUMBER, WHICH MUST BE ASSIGNED TO KS0.
IEV = EVENT VARIABLE. VALUE RETURNED = -1 IF THE REQUEST
IS ATTEMPTED; NEGATIVE IF THE I/O PROCESSOR COULD
NOT HONOR THE REQUEST.
INTNS = INTENSITY, 0<INTNS<7, 7 BRIGHTEST.
IX1 = 6220 = DISTANCE TO THE RIGHT OR LEFT OF CENTER FOR THE
BEGINNING OF THE LINE (-10 < DISTANCE < +10).
IY1 = 6220 = DISTANCE ABOVE OR BELOW CENTER FOR THE BEGINNING
OF THE LINE (-10 < DISTANCE < +10).
IX2 = 6220 = DISTANCE TO THE RIGHT OR LEFT OF CENTER FOR THE
END OF THE LINE.
IY2 = 6220 = DISTANCE ABOVE OR BELOW CENTER FOR THE END OF THE
LINE.
IS = 100 = LINEAR DISTANCE OF THE LINE (USING THE SAME VERTICAL
AND HORIZONTAL RANGE FOR SCALING AS THE DISTANCE USED IN
COMPUTING IX1, IY1).
IDELX = (IX2-IX1)/IS
IDELY = (IY2-IY1)/IS
MDE = MODE - NOT YET IMPLEMENTED. VALUE IGNORED.

THIS SUBPROGRAM WRITTEN BY P. N. BARTRAM AND SUPPORTED BY J. T. MEIBEL.

.EJECT
GRAPHICS 'SCOPE HANDLE SUBROUTINE.

GLOBL SCLA,PNT,HZ,VL,LIN,DAA

ENTRY FOR CALL TO SCLA.
ENTRY FOR CALL TO PNT.
ENTRY FOR CALL TO HZ.
ENTRY FOR CALL TO VL.
ENTRY FOR CALL TO LIN.
ENTRY FOR CALL TO CAL.
PARAMETER FETCHING ROUTINE.
FETCH THE PARAMETER ADDRESSES.
GET THE LUN, AND PLACE IN THE CAL PARAMETER BLOCK.
PLACE THE EVENT VARIABLE ADDRESS IN THE CAL PARAMETER BLOCK.
GRAPHICS 'SCOPE HANDLER SUBROUTINE.

/ALSO, IN THE WAITFOR CAL PARAMETER BLOCK.
CHECK IF PARAMETERS ARE TO BE TRANSFERRED
FROM THE CALLING PROGRAM TO THE TABLE
/INITIALIZE THE LIMIT REGISTER (# PARAMETERS),
/INITIALIZE THE INDEX REGISTER,
/FORM THE LAC* INSTRUCTION,
/GET THE PARAMETER,
/STORE IN THE CAL TABLE,
/UPDATE THE LAC* INSTRUCTION FOR THE NEXT
PARAMETER, CHECK FOR COMPLETION, UPDATE
THE INDEX, AND LOOP BACK FOR THE NEXT ONE,
ISSUE THE I/O DIRECTIVE,
WAIT FOR IT TO BE DE-QUEUED,
/WAS THE GRAPHICS BUSY (EVENT VARIABLE = -22)?
/NO, RETURN,
/YES, MARK TIME FOR 1/2 SECOND, AND TRY
AGAIN.
/CAL PARAMETER BLOCK FOR THE GRAPHICS 'SCOPE
/I/O HANDLER, PARAMETERS SUPPLIED,
PARAMETER BLOCK FOR THE WAITFOR.
PARAMETER BLOCK FOR THE MARK TIME.
/CAL PARAMETER BLOCK FOR THE WAITFOR FOR MARK,
/CAL PARAMETER BLOCK FOR THE WAITFOR FOR MARK.
/LAC* FOR TRANSFERRING PARAMETER VALUES,
/STORAGE FOR NUMBER OF PARAMETERS.
APPENDIX C

Plastic Sheet Thickness Study

Computer Program Listings

The following pages contain the source listings for the most important of the programs for the collection and analysis of the polymer sheet data. Other programs dealing with rearrangement of data in computer mass storage files, PSDF averaging, and other routine functions are not given.

Program BETA8, which is segmented with overlayed subroutines BETA8A and BETA8B, is for obtaining the sheet thickness as previously produced sheet is passed through the Beta Gauge at a fixed speed. It is an example of the use of the automatic filling of user buffers with real-time input by the modified RSX-Plus monitor. Real-time output usage is also illustrated.

Program BTAl$\Phi$, segmented with overlayed subroutines BTAl$\Phi$A and BTAl$\Phi$B, is for the on-line collection of extruder data, and provides another example of automatic real-time input and the setting of real-time outputs.

Program BTAl$^4$ is the major computational program, containing routines for the fast Fourier transform (RFORT and FORT) and computation of the power spectral density function. This program is organized into seven overlays, as explained in the listing, with subroutines BTAl$^4$A,
BTA1^B, BTA1^C, RFT, FRT, BTA1^D, BTA1^E, BTA1^F, PLOT, BTA1^G, and COSTP.
TASK BETA8 FOR OBTAINING MACHINE DIRECTION SHEET THICKNESS DATA
AT A FIXED CROSS-SHEET POSITION, PLASTIC SHEET THICKNESS PROJECT,
THE SHEET IS TO BE PASSING UNDER THE BETA GAUGE AT CONSTANT
VELOCITY,

THIS TASK MUST BE FIXED IN CORE,

LUN 61 MUST BE ASSIGNED TO A WRITE-ENABLED DECTAPE, OR DISK,
63 TO AN OPERATOR INTERACTIVE DEVICE, SUCH AS THE KEYBOARD-
'SCOPE OR THE MCR TELEPRINTER.

NORMAL TERMINATION OF THIS TASK IS BY SETTING SENSE SWITCH 6
(TO THE RIGHT, UNDER THE MIDDLE 'SCOPE) ON.

AFTER INITIALIZATION, THERE IS A 15 SECOND PAUSE, FOLLOWED BY
SENSE LIGHTS 3 AND 5 (NUMBERED FROM LEFT) BEING LIT. THE
RUBBER ROLLER SHOULD BE STARTED BY THE OPERATOR WHEN THE LIGHT
APPEARS, INDICATING THE START OF DATA TAKING.

DATA ARE STORED ON DECTAPE OR DISK IN AN OPERATOR SPECIFIED FILE,
IN BLOCKS OF 32 WORDS. THERE IS NO HEADER BLOCK, AND NO TERMINAL
BLOCK. IN READING THE TAPE, USE THE END= OPTION WITH THE READ
STATEMENT.

THIS TASK CONSISTS OF THE RESIDENT DRIVER (THIS CODE) AND TWO SUB-
ROUTINES WHICH OVERLAY EACH OTHER. THE FIRST SUBROUTINE, BETA8A,
IS FOR INITIALIZATION, TERMINATION, AND CONTAINS ALL CHARACTER
(FORMATTED) INPUT AND OUTPUT. BETA8A IS CALLED BY THE DRIVER
AND BY THE OTHER SUBROUTINE, SUBROUTINE BETA8B, THE SECOND
OVERLAY, IS CALLED BY BETA8A, AND CONTAINS CODE FOR DATA
COLLECTION AND TRANSFER TO DECTAPE.

COMMON /PARAM/ IERR1, IERR2, IERR3, FILE
IERR1 = 0
CALL BETA8A
END
SUBROUTINE BETA8A

C SUBROUTINE BETA8A, FIRST OF TWO OVERLAYED SUBROUTINES FOR TASK
C TASK BETA8. THIS SUBROUTINE HANDLES ALL CHARACTER (FORMATTED)
INPUT AND OUTPUT REQUIRED FOR INITIALIZATION AND ERROR
DIAGNOSTICS. SUBROUTINE BETA8B IS CALLED BY THIS SUBROUTINE FOR
COLLECTION OF DATA FROM THE BETA GAUGE. THIS SUBROUTINE IS
CALLED BY THE DRIVER FOR INITIALIZATION AND BY SUBROUTINE
BETA8B FOR TERMINATION, WHETHER IN RESPONSE TO AN ERROR OR
BY OPERATOR REQUEST.

C COMMON /PARAM/ IERR1, IERR2, IERR3, FILE
IF (IERR1) 101,102,103

C CALL IS FOR INITIALIZATION,
C
102 WRITE (63,1)
  1 FORMAT (' BETA GAUGE DATA COLLECTION. ENTER FILE
  1 NAME, A5,' )
  READ (63,2) FILE
  2 FORMAT (A5)
  CALL ENTER (61, FILE, 3HPNB, IEV)
  CALL WAITFR (IEV)
  IF (IEV) 111, 111, 112
  WRITE (63,3) FILE, IEV
  111 FORMAT (' *** TAPE FILE "", A5, 'PNB" CANNOT BE OPENED,
  1 EVENT VARIABLE", I5)
  CALL UNFIX (5HBETA8)
  GO TO 199
  112 CALL BETA8B

C ERROR DETECTED BY SUBROUTINE BETA8B.
C
103 IF (IERR1-3) 121,121,101
  101 WRITE (63,4) IERR1
  4 FORMAT (' *** ERROR, UNIDENTIFIED, IERR1', I5)
  GO TO 198
  121 GO TO (131,132,133), IERR1
  131 WRITE (63,5) IERR2
  5 FORMAT (' *** ERROR RETURNED BY RTSET, EVENT VARIABLE", I5)
  GO TO 199
  132 WRITE (63,7) FILE
  7 FORMAT (' *** WRITE ERROR OCCURED, TAPE FILE "", A5, "PNB",')
  GO TO 197

C NORMAL TERMINATION, BY OPERATOR REQUEST,
C
133 WRITE (63,8)
  8 FORMAT (' NORMAL OPERATOR TERMINATION REQUESTED,' )
  197 CALL CLOSE (61, FILE, 3HPNB, IEV)
  CALL WAITFR (IEV)
  IF (IEV) 141,141,199
  141 WRITE (63,9) FILE, IEV
  9 FORMAT (' *** ERROR IN CLOSING TAPE FILE "", A5, 'PNB",
  1 EVENT VARIABLE", I5)
  199 CALL EXIT
C
END
SUBROUTINE BETA8B

SUBROUTINE BETA8B, SECOND OF TWO OVERLAYERED SUBROUTINES FOR TASK BETA8. THIS SUBROUTINE IS CALLED BY SUBROUTINE BETA8A FOR COLLECTING BETA GAUGE DATA AND STORING THEM ON DECTAPE. THIS SUBROUTINE EXITS TO SUBROUTINE BETA8A UPON DETECTION OF AN ERROR CONDITION OR OPERATOR TERMINATION REQUEST.

DIMENSION IDT(512), IWR(32), IT(2)
COMMON /PARAM/ IERR1, IERR2, IERR3, FILE
DATA IT/15,2/

C INITIALIZE REAL-TIME INPUT FROM THE BETA GAUGE AT 10 Hz SAMPLING RATE. ADC CHANNEL 11 IS ASSUMED TO BE THE THICKNESS SIGNAL.

CALL MARK (IT,IEV)
CALL WAITFR (IEV)
CALL RTOUX (0,0,10)
CALL RTSET (11,3,IDT,IERR2)
IF (IERR2) 101,101,102
101 IERR1 = 1
GO TO 103
102 DO 202 I=32,512,32
202 IDT(I) = 0
121 JK = 0
121 IK = JK + 32
CALL WAITFR (IDT(IK))
DO 201 JK = JK + 1
201 CONTINUE
IDT(JK) = 0
WRITE (61,ERR=301) IWR
CALL RTINX (0,-1,IBT)
IF (IBT-IBT/2*2) 113,114,113
113 IERR1 = 3
114 IF (IJK=512) 121,122,122

C OPERATOR TERMINATION.

C 103 CALL UNFIX (5HBETA8)
C CALL RTOUX (0,0,0)
C CALL BETA8A

C WRITE ERROR DETECTED.

C 301 IERR1 = 2
GO TO 103
END
PROGRAM BTA10: MAIN DRIVER FOR TASK BTA10. THIS TASK CONSISTS
OF THIS DRIVER AND TWO OVERLAYED SUBROUTINES, BTA10A AND BTA10B.
SUBROUTINE BTA10A IS CALLED BY THIS DRIVER; BTA10B IS CALLED
BY BTA10A.

THIS TASK MUST BE FIXED IN CORE.

THIS TASK MUST BE NAMED "BTA10".

THIS TASK IS FOR COLLECTION OF RUN-TIME EXTRUDER DATA FOR THE
SHEET THICKNESS PROJECT. THE COLLECTION IS HANDLED BY SUBROUTINE
BTA10A. ERROR MESSAGES AND OTHER FORMATTED OUTPUT IS PERFORMED
BY SUBROUTINE BTA10B. ANALOG TO DIGITAL CONVERTER CHANNELS
USED MUST BE DEFINED IN CALLS TO LIBRARY ROUTINE RTSET IN
SUBROUTINE BTA10A.

THE DATA CONSISTS OF 10 CHANNELS SAMPLED AT 10 HERTZ (THERMO-
COUPLES) AND 7 SAMPLED AT 20 HZ (PRESSURES AND THICKNESS).
THE DATA ARE AVERAGED OVER ONE SECOND AND STORED ON DECTAPE
IN FILE "AVRG" PNB. EVERY 10 MINUTES, THE UNAVERAGED DATA
IS ALSO STORED, FOR 103 SECONDS, IN DISK FILES "BRST1 PNB",
"BRST2 PNB", "BRST3 PNB", "BRST4 PNB", AND "BRST5 PNB",
EACH FAST BURST DATA FILE CONSISTS OF 1030 SAMPLES OF EACH
10HZ CHANNEL AND 2060 OF EACH 20HZ CHANNEL.

LOGICAL UNIT NUMBER 61 MUST BE ASSIGNED TO THE RF DISK HANDLER,
62 TO A DECTAPE UNIT, AND
63 TO A PRINTING DEVICE, SUCH AS A TELEPRINTER.
A DECTAPE WITH A NEW DIRECTORY MUST BE USED ON LUN 62.

THE RUN PROCEEDS UNTIL PUSHBUTTON 6, TO THE RIGHT UNDER THE CHAR-
ACTER 'SCOPE', IS SET. THIS SWITCH IS NOT CHECKED WHILE A BURST
OF FAST DATA IS BEING STORED, OR DURING THE FILE MANIPULATION
THEREAFTER, THIS SHOULD BE LEFT SET SO THAT THE PROGRAM WILL BE
TERMINATED AFTER THE BURST HAS BEEN STORED.

THE DATA STRUCTURE FOR THE TAPE STORAGE OF THE AVERAGES IS IN
BLOCKS OF 17, EACH BLOCK CONSISTING OF THE AVERAGE OF EACH OF
THE 17 A/D CHANNELS FOR ONE SECOND. THE FIRST AND THE LAST
BLOCKS CONSIST OF 10 ZEROS, FOLLOWED BY THE DATE, TIME AND AN-
OTHER ZERO. FOR THE FAST BURST DATA, THE FIRST BLOCK OF EACH
FILE IS SEVEN WORDS LONG, CONSISTING OF A SYNCHRONIZATION KEY,
THE DATE, AND THE TIME APPROXIMATELY ONE SECOND BEFORE THE START
OF THE BURST. THIS IS FOLLOWED BY 103 BLOCKS OF DATA, EACH
BLOCK CONSISTING OF ONE SECOND OF DATA FOR EACH A/D CHANNEL,
GROUPED BY CHANNELS. THE LAST BLOCK IS ALSO SEVEN WORDS LONG,
REPEATING THE SYNCHRONIZATION KEY AND DATE, AND CONTAINING THE
TIME ABOUT ONE SECOND AFTER THE END OF THE BURST.

COMMON /PARAM/ IER(22)

CALL BTA10A

END
SUBROUTINE BTA10A

SUBROUTINE TO COLLECT DATA FROM THE EXTRUDER AND STORE IT ON TAPE
AND DISK. THIS IS THE FIRST OF TWO OVERLAYERED SUBROUTINES
FOR TASK BTA10, AND CONTAINS CODE FOR INITIALIZATION, DATA
COLLECTION AND STORAGE, AND RUN TERMINATION. THIS SUBROUTINE
CALLS SUBROUTINE BTA10B, WHICH CONTAINS THE TERMINAL PRINT
CODE, FOR BOTH NORMAL AND ERROR TERMINATION.

A/D CHANNEL NUMBERS MUST BE ASSIGNED IN THE CALLS TO RTSET GIVEN
BELOW. THE CHANNEL NUMBER IS THE FIRST PARAMETER. THE SAMPLE
RATE IS INDICATED BY THE SECOND PARAMETER: 3 IF FOR 10 SAMPLES
PER SECOND; 2 IS FOR 20 SAMPLES PER SECOND (ACTUALLY, 100, BUT
ONLY EVERY FIFTH VALUE IS USED IN THIS PROGRAM).

THE AVERAGES DATA IS CONTINUOUSLY STORED ON DECTAPE, LUN 62,
IN FILE "AVRG PNB", BURST DATA IS STORED ON DISK, LUN 61,
IN FILES "BRST1 PNB", "BRST2 PNB", ..., "BRST5 PNB".

CODES FOR FILE MANIPULATION WITH RESPECT TO COLLECTION OF
FAST DATA BURSTS ARE

ISTR - INCREMENTED BY ONE EVERY TWO SECONDS
ISTR>0 WRITE FAST DATA BURST TO DISK,
ISTR=0 STORE LAST OF DATA FOR THIS BURST AND
WRITE TERMINATION BLOCK,
ISTR<0 NO FAST DATA IS TO BE STORED,
ISTR=248 RESET ISTR TO -51, AND WRITE INITIAL
BLOCK FOR FAST DATA BURST.

IFSEN -
IFSEN=0 FAST DATA BURSTS MAY BE STORED IF REQUIRED,
IFSEN=1 NO MORE FAST DATA BURSTS MAY BE STORED
ERROR DETECTED,
IFSEN=-1,-2 FILE MANIPULATION MUST BE COMPLETED
BEFORE MORE FAST DATA CAN BE STORED,

DIMENSION ITA(10), IZR(10), IPA(7), ITM(2), BUR(5)
DIMENSION IT1(20), IT2(20), IT3(20), IT4(20), IT5(20), IT6(20),
1 IT7(20), IT8(20), IT9(20), IT10(20), IP1(200), IP2(200),
2 IP3(200), IP4(200), IP5(200), IP6(200), IP7(200)
COMMON /DATA/ ITA(20,10), IZR(200,7)
COMMON /PARAM/ IERR(9), INOT(6), IDT(6), IBF
EQUIVALENCE (IT1,IT(1,1)), (IT2,IT(1,2)), (IT3,IT(1,3)),
1 (IT4,IT(1,4)), (IT5,IT(1,5)), (IT6,IT(1,6)), (IT7,IT(1,7)),
2 (IT8,IT(1,8)), (IT9,IT(1,9)), (IT10,IT(1,10)), (IP1,
3 IP(1,1)), (IP2,IP(1,2)), (IP3,IP(1,3)), (IP4,IP(1,4)),
4 (IP5,IP(1,5)), (IP6,IP(1,6)), (IP7,IP(1,7))

DATA ISTR /-52/, ITM/30,1/, IFSEN/0/
DATA BUR /5HBRST1, 5HBRST2, 5HBRST3, 5HBRST4, 5HBRST5/
DATA IZR /100/0/

INITIALIZE SOME OF THE COMMON VARIABLES,

IERR(5) = 0
IERR(6) = 0
IERR(7) = 0
IBF = 1

OPEN THE TAPE FILE FOR THE AVERAGED DATA, AND THE DISK FILE
FOR THE FIRST BURST DATA.
CALL ENTER (62, 5HAVRG* , 3HPNB, IERR(1))
CALL ENTER (61, BUR(1), 3HPNB, IERR(2))
CALL WAITFR (IERR(2))
IF (IERR(2)) 403, 403, 404
403 IFSN = 1
IBF = 0
404 CALL WAITFR (IERR(1))
IF (IERR(1)) 401, 401, 402
401 CALL BTA10B
C
C WRITE THE HEADER (AVERAGED DATA FILE).
C
402 CALL DATE (INLDT)
WRITE (62, ERR=503) IZR, INLDT, IFSN
C
C SET THE ENTRIES FOR AUTOMATIC FILLING OF THE DATA ARRAYS,
C
CALL RTSET (33, 3, IT1)
CALL RTSET (35, 3, IT2)
CALL RTSET (37, 3, IT3)
CALL RTSET (39, 3, IT4)
CALL RTSET (38, 3, IT5)
CALL RTSET (34, 3, IT6)
CALL RTSET (36, 3, IT7)
CALL RTSET (44, 3, IT8)
CALL RTSET (85, 3, IT9)
CALL RTSET (21, 3, IT10, IERR(3))
IF (IERR(3)) 405, 405, 406
405 IBF = 0
CALL BTA10B
406 CALL RTSET (50, 2, IP1)
CALL RTSET (49, 2, IP2)
CALL RTSET (48, 2, IP3)
CALL RTSET (51, 2, IP4)
CALL RTSET (52, 2, IP5)
CALL RTSET (53, 2, IP6)
CALL RTSET (11, 2, IP7, IERR(4))
IF (IERR(4)) 405, 405, 406
C
C GET THE UPDATED TIME.
C
408 CALL DATE (INLDT)
C
C SYNCHRONIZE THE DATA INPUT, AND CALCULATE THE OFFSET FOR
C THE TWO DATA RATES,
C
CALL RTRST (ILEAD)
ILEAD = ILEAD/10 + 1
C
ILEAD NOW IS THE ELEMENT IN THE IP DATA ARRAYS WHICH IS TRANS-
C FERED AT THE SAME TIME AS THE FIRST ELEMENT OF EACH IT ARRAY.
C
ISTRA = ILEAD - 5
IF (ISTRA) 411, 411, 412
411 ISTRA = ILEAD
412 ISTA5 = ISTRA + 5
ISTRB = ISTRA + 100
ISTB5 = ISTA5 + 100
ISTPA = ISTRA + 95
ISTPR = ISTRB + 95
C
ISTRA, ISTA5, ISTRB, ISTB5, ISTPA, AND ISTPB ARE ARRAY INDECES
C FOR GETTING IP DATA CORRESPONDING TO THE SAME SAMPLES AS IT
C DATA. THESE ARE REQUIRED BY THE CODE WRITING TO MASS STORAGE,
C
C WRITE THE SYNCHRONIZATION FACTOR AND THE TIME ON DISK AS THE
C HEADER FOR THE FIRST FILE OF FAST BURST DATA.
C WRITE (61, ERR=501) IHEAD, INLT
C SET PROPER WORDS TO ZERO FOR INITIALIZING DATA COLLECTION.
C
C ITI(10) = 0
C ITI(20) = 0
C
C CALL RTOUT (0, 0, 6)
C
C CHECK IF DATA ARE READY;
C DATA SHOULD BE IN THE FIRST HALF OF THE BUFFER, POSITION THE SIGN
C RIT, AND SUM; EACH "AVERAGE" IS THE TRUE AVERAGE TIMES FIVE,
C
ISTR = ISTR + 1
IF (ITI(10)) 107, 108, 107
CALL MARK (ITM, IEV)
CALL WAITFR (IEV)
GO TO 102
DO 201 I=1,10
CALL IADCVF (IT(I,1), IT(I,1))
ITA(I) = IT(I,1)/2
DO 201 J=2,10
CALL IADCVF (IT(J,1), IT(J,1))
201
ITA(I) = ITA(I) + IT(J,1)/2
DO 202 I=1,7
CALL IADCVF (IP(ISTR, I), IP(ISTR, I))
IPA(I) = IP(ISTR, I)/4
DO 202 J=ISTR+1, ISTPA-5, 5
CALL IADCVF (IP(J, I), IP(J, I))
202
IPA(I) = IPA(I) + IP(J, I)/4

IF (ISTR) 111, 111, 112
IF (ITI(20)) 112, 114, 112
114 WRITE (61, ERR=192) ((IT(I, J), I=1,10), (IP(I, J),
1 I=ISTR, ISTPA, 5), J=1,7)
C CHECK FOR DATA MISSED (TIME TO WRITE MAY HAVE BEEN TOO LONG),
C THEN, IF O.K., WRITE THE DATA TO DISK.
C
116 IERR(7) = (-1)
GO TO 199
115 WRITE (62, ERR=196) ITA, IPA
ITI(10) = 0
C NOW WORKING ON THE SECOND HALF OF THE BUFFER.
C
126 ITI(20) = 127, 128, 127
128 CALL MARK (ITM, IEV)
CALL WAITFR (IEV)
GO TO 117
127 DO 211 I=1,10
CALL IADCVF (IT(I,1), IT(I,1))
ITA(I) = IT(I,1)/2
DO 211 J=12,20
CALL IADCVF (IT(J,1), IT(J,1))
211
ITA(I) = ITA(I) + IT(J,1)/2
DO 212 I = 1,7
CALL IADCVF (IP(ISTR, I), IP(ISTR, I))
IPA(I) = IP(ISTR, I)/4
DO 212 J = ISTB5, ISTPB, 5
CALL IADCV (IP(J, I), IP(J, I))
212 IPA(I) = IPA(I) + IP(J, I) / 4

C CHECK IF FAST BURST IS IN PROGRESS. IF SO, WRITE IT TO DISK.
C IF THE LAST OF THIS BURST HAS JUST BEEN WRITTEN, WRITE THE
C TERMINATION BLOCK.
C
IF (ISTR) 121, 124, 122
124 IF (IFSEN) 122, 125, 122
125 CALL DATE (IDT)
WRITE (61, ERR=198) ILEAD, IDT
CALL RTOUX (0, 0, 4)
GO TO 122
121 IF (IFSEN) 122, 123, 122
123 WRITE (61, ERR=198)((IT(I, J), I=11, 20), J=1, 7)

C CHECK FOR BAD TIMING. THEN WRITE THE DATA IF O.K.
C
122 IF (IT1(10)) 116, 126, 116
126 WRITE (62, ERR=196) ITA, IPA
IT1(20) = 0

C CHECK IF FILE MANIPULATION REQUIRED FOR FAST BURST DATA.
C
162 CALL CLOSE (61, BUR(IBF), 3HPNB, IERR(8))
IFSEN = (-1)
GO TO 101
163 IF (IFSEN=1) 164, 165, 134
165 IF (IERR(8)) 166, 101, 167
161 IBF = IBF - 1
166 IFSEN = 1
GO TO 101
167 IF (IBF<0) 168, 166, 166
168 IBF = IBF + 1
IFSEN = (-2)
CALL ENTER (61, BUR(IBF), 3HPNB, IERR(2))
GO TO 101
164 IF (IERR(2)) 161, 101, 169
169 IFSEN = 0
GO TO 101

C CHECK FOR END-OF-RUN. TERMINATE IF END OF RUN.
C
134 CALL RTINX (0, -1, IERR(7))
IERR(7) = IERR(7) - IERR(7)/2*2
IF (IERR(7)) 598, 151, 598
598 CALL DATE (IDT)
WRITE (62, ERR=511) IZR, IDT, ILEAD
GO TO 199

C CHECK IF NEXT TIME IS START OF FAST DATA BURST.
C
151 IF (ISTR=248) 101, 101, 153
153 ISTR = (-51)
CALL DATE (IDT)
WRITE (61, ERR=188) ILEAD, IDT
CALL RTOUX (0, 0, 6)
GO TO 102

C SET UP ERROR CONDITION CODE FOR BAD WRITE ATTEMPTS.
C
C FAST BURST ATTEMPT,
192  IERR(6) = (-1)
    IFSEN = 1
    GO TO 117
198  IERR(6) = (-1)
    IFSEN = 1
    GO TO 122
188  IERR(6) = (-2)
    IFSEN = 1
    GO TO 102
501  IERR(6) = (-2)
    IFSEN = 1
    GO TO 420

C FATAL ERROR: BAD WRITE OF AVERAGE DATA.
C
511  IERR(5) = (-3)
    GO TO 199
503  IERR(5) = (-2)
    GO TO 199
196  IERR(5) = (-1)
199  CALL RTOUX (0, 0, 0)
C END OF DATA TAKING; STOP AUTOMATIC DATA INPUT.
C
CALL UNFIX (SHBTA10)
C CLOSE THE DATA FILES.
C
496  CALL CLOSE (62, SHAVRG, 3HPNB, IERR(9))
    IF (IFSEN) 498,497,498
497  CALL CLOSE (61, BUR(IBF), 3HPNB, IERR(8))
    IBF = IBF - 1
    CALL WAITFR (IERR(8))
498  CALL WAITFR (IERR(9))
    CALL BTA10B
C
END
SUBROUTINE BTA108

C THIS SUBROUTINE IS THE SECOND OVERLAY OF TASK "BTA10", WHICH
C HANDLES DATA COLLECTION FOR THE EXTRUDER IN THE SHEET THICK-
C NESS PROJECT. THIS SUBROUTINE IS CALLED ONLY TO TERMINATE
C THE DATA COLLECTION, EITHER BECAUSE OF AN ERROR OR THROUGH
C OPERATOR REQUEST.
C
C THE OPERATOR SHOULD TRANSFER THE FAST BURST DATA FROM DISK TO
C DECTAPE AS SOON AS PRACTICAL.
C
C LUN 63 SHOULD BE ASSIGNED TO AN OPERATOR INTERACTIVE DEVICE, SUCH
C AS THE KEYBOARD-'SCOPE OR A TELEPRINTER.
C
COMMON /PARAM/ IERR(9), IDSTR(6), IDT(6), IBF
C
WRITE (63,1) IDSTR, IDT
1 FORMAT ('0EXTRUDER SHEET THICKNESS PROJECT, RUN-TIME DATA
1 COLLECTION PROGRAM/', 'RUN STARTED!', 4X, I2, '!', I2,
2 '!', I2, ' AT ', I2, '!', I2, '!', I2, '!', I2, 'RUN TERMINATED!
3 ', I2, '!', I2, '!', I2, ' AT ', I2, '!', I2, '!', I2, '!
C
C CHECK IF NORMAL TERMINATION.
C
IF (IERR(7)) 101, 102, 103
101 WRITE (63,2) 2 FORMAT ('0** PROGRAM TERMINATED - BAD TIMING, POSSIBLE
1 LOSS OF DATA.')
GO TO 104
102 WRITE (63,3) 3 FORMAT ('0** PROGRAM WAS NOT OPERATOR TERMINATED, FATAL
1 ERROR FOUND.')
GO TO 104
103 WRITE (63,18) 18 FORMAT ('0NORMAL OPERATOR TERMINATION.')
104 WRITE (63,20) 20 FORMAT ('0 NUMBER OF FAST DATA BURSTS!', I5)
C
C CHECK IF THE DISK FILE FOR THE AVERAGES WAS OPENED.
C
IF (IERR(1)) 111, 111, 112
111 WRITE (63,4) IERR(1) 4 FORMAT ('0** PROGRAM ABORTED - TAPE FILE "AVRG-
1 PNB" COULD NOT BE OPENED, EVENT VARIABLE!', I5)
GO TO 199
C
C CHECK IF THE DISK FILE FOR THE FAST BURST DATA WAS OPENED.
C
112 IF (IERR(2)) 113, 113, 114
113 WRITE (63,5) IERR(2) 5 FORMAT ('0** ERROR IN OPENING A DISK FILE FOR FAST
1 BURST DATA, EVENT VARIABLE!', I5)
C
C DISK FILES WERE OPENED, CHECK IF THERE WERE ANY DATA
C STORED IN THEM.
C
114 IF (IERR(3)) 121, 121, 122
121 WRITE (63,6) IERR(3) 6 FORMAT ('0** PROGRAM ABORTED - RTSET ERROR, EVENT
1 VARIABLE!', I5)
122 IF (IERR(4)) 124, 124, 125
124 WRITE (63,7)
7 FORMAT ('*** PROGRAM ABORTED - RTRST ERROR, EVENT
1 VARIABLE:', 15)
GO TO 199
C
C CHECK FOR ERRORS IN WRITING AVERAGED DATA TO DECTAPE.
C
125 IF (IERR(5)) 131, 132, 133
131 IEV = IERR(5) + 2
IF (IEV) 136, 134, 135
134 WRITE (63,8)
8 FORMAT ('*** PROGRAM ABORTED - INITIAL WRITE TO TAPE
1 FILE "AVRG* PNB" UNSUCCESSFUL,')
GO TO 199
135 WRITE (63,9)
9 FORMAT ('*** PROGRAM ABORTED - ERROR IN WRITING TO
1 TAPE FILE "AVRG* PNB",')
GO TO 132
136 IF (IEV+1) 133,137,133
137 WRITE (63,19)
19 FORMAT (' ERROR IN WRITING TERMINAL BLOCK TO
1 TAPE FILE "AVRG* PNB",')
GO TO 132
133 WRITE (63,10) IERR(5)
10 FORMAT (' UNIDENTIFIED ERROR - TAPE WRITE ERROR
1 CODE:', 15)
C
C CHECK FOR ERRORS IN WRITING THE FAST BURST DATA TO DISK.
C
132 IF (IERR(6)) 141, 154, 145
141 IEV = IERR(6) + 2
IF (IEV) 145, 142, 143
142 WRITE (63,11)
11 FORMAT ('*** ERROR DETECTED IN THE INITIAL WRITE TO A
1 FAST BURST DATA DISK FILE,')
GO TO 154
143 WRITE (63,12)
12 FORMAT ('*** FAST BURST DATA TRUNCATED - ERROR IN
1 WRITING TO DISK DETECTED,')
GO TO 154
145 WRITE (63,13) IERR(6)
13 FORMAT ('*** UNIDENTIFIED ERROR - DISK WRITE ERROR
1 CODE:', 15)
C
C CHECK FOR ERRORS IN CLOSING THE DISK FILES.
C
154 IF (IERR(9)) 161, 161, 162
161 WRITE (63,16) IERR(9)
16 FORMAT ('*** ERROR IN CLOSING TAPE FILE "AVRG* PNB"
1 DETECTED, EVENT VARIABLE:', 15)
162 IF (IERR(2)) 199, 199, 164
164 IF (IERR(8)) 165, 165, 199
165 WRITE (63,17) IERR(8)
17 FORMAT ('*** ERROR IN CLOSING A DISK FILE FOR FAST BURST DATA
1 DETECTED, EVENT VARIABLE:', 15)
C
199 CALL EXIT
END
TASK BTA14, GENERAL FREQUENCY ANALYSIS TASK. THIS PROGRAM
TAKES DATA FROM DECTAPE (LUN 62), PERFORMS A FAST FOURIER
TRANSFORM, THEN COMPUTES THE POWER SPECTRAL DENSITY FUNCTION,
PRINT-OUTS AND PLOTS ARE AT THE OPTION OF THE OPERATOR.
LOGICAL UNIT (LUN) 63 MUST BE ASSIGNED TO AN OPERATOR INTER-
ACTIVE DEVICE SUCH AS A TELEPRINTER, AND LUN 64 TO THE LINE
PRINTER. AN ADDITIONAL BULK STORAGE DEVICE MAY BE OPERATOR
SPECIFIED FOR SAVING THE POWER SPECTRAL DENSITY FUNCTION
(REQUIRED IF THIS IS TO BE PLOTTED) OR FOR SAVING THE FOURIER
COEFFICIENTS,
THIS TASK IS TERMINATED IF A BLANK (CARRIAGE RETURN ONLY)
FILE NAME IS ENTERED WHEN THE REQUEST FOR THE DATA FILE
NAME IS MADE,
THIS TASK CONTAINS MANY OVERLAYS IN ADDITION TO THIS MAIN
CODE, WHICH SERVES ONLY TO CALL THE OVERLAYS,
LINK1=BTA14A READS THE DATA FROM TAPE (ASSUMING INTERGER
NUMBERS WRITTEN IN BLOCKS OF 32), AND CONVERTS THEM TO
FLOATING POINT,
LINK2=BTA14B PRINTS THE DATA, DETRENDS IT, AND
PRINTS THE DETRENDED DATA AS REQUIRED.
LINK3=BTA14C,RFORT,FORT PERFORMS THE FAST FOURIER
TRANSFORM, AND INVERSE IF REQUIRED.
LINK4=BTA14D PRINTS THE FOURIER COEFFICIENTS IF REQUIRED.
LINK5=BTA14E COMPUTES THE POWER SPECTRAL DENSITY FUNCTION
(PSDF), AND STORES IT ON TAPE OR DISK, IF REQUIRED.
LINK6=BTA14F, PLOT PLOTS THE PSDF IF REQUIRED,
LINK7=BTA14G, COSINE TAPERS THE DATA (AFTER DETRENDS) AND PRINTS THE TAPERED DATA IF REQUIRED. ALSO,
AFTER THE INVERSE TRANSFORM HAS BEEN CALLED TO RE-COMPUTE
THE ORIGINAL DATA, THE COSINE TAPER AND DETRENDING ARE
REMOVED.

COMMON /PARAM/ M,N,ICS,FLE,FRQ,UNT,PLX,FLS,LUN,ISD,
1 INV,IERR,ITR,BC,B1
COMMON /DATA/ X(2050)

105 CALL BTA14A
CALL BTA14B
IF (ICS) 111, 112, 111
111 CALL BTA146 (0)
112 CALL BTA14C (-1)
101 CALL BTA14D
IF (ISD) 101, 102, 101
103 CALL BTA14F
102 IF (INV) 104, 105, 104
104 CALL BTA14G (+1)
GO TO 105
END
SUBROUTINE BTA14A, THE FIRST OVERLAY OF TASK BTA14. THIS SUBROUTINE OBTAINS THE DATA FROM TAPE AND CONVERTS IT TO FLOATING POINT.

SUBROUTINE BTA14A

DIMENSION IDA(32)
COMMON /PARAM/ M,N,ICS,FLE,FRO,UNT,IPX,FLS,LUN,ISD,
1 INV,IERR,ITR,B0,B1
COMMON /DATA/ X(2050)
DATA YES/3HYES/, EXT/1H /

GET INITIAL INFORMATION ABOUT DATA TO BE ANALYSED.

100 WRITE (63,1)
1 FORMAT ('ENTER FILE NAME, A5,' )
READ (63,2) FLE
2 FORMAT (A5)
C EXIT IF NO FILE NAME GIVEN.
C IF (FLE ',EQ . EXT) CALL EXIT
C OPEN THE TAPE FILE WHILE GETTING THE REST OF THE INITIAL INFORMATION.
C CALL SEEK (62, FLE, 3HPNB, IEV)
WRITE (63,3)
3 FORMAT ('ENTER POWER OF TWO FOR NUMBER OF DATA VALUES TO BE
1 USED, 12,' )
READ (63,4) M
4 FORMAT (12)
IF (M) 101, 101, 102
101 WRITE (63,5) M
5 FORMAT ('VALUE OUT OF RANGE!', I5)
GO TO 104
102 IF (M-11) 103, 103, 101
103 N = 2**M
107 WRITE (63,6)
6 FORMAT ('ENTER NUMBER OF VALUES TO BE SKIPPED AT THE
1 BEGINNING OF THE FILE, I5,' )
READ (63,7) ISK
7 FORMAT (15)
IF (ISK), 105, 106, 106
105 WRITE (63,5) ISK
GO TO 107
106 WRITE (63,8)
8 FORMAT ('ENTER NUMBER OF VALUES SKIPPED BETWEEN VALUES
1 USED, 12,' )
READ (63,4) ISP
9 FORMAT (15)
IF (ISP), 108, 109, 109
108 WRITE (63,5) ISP
GO TO 106
109 ISP = ISP + 1
111 WRITE (63,9)
9 FORMAT ('ENTER # OF 0'S TO FILL AT END, I5")
READ (63,7) N2R
112 NR = N - N2R
CHECK IF THE TAPE FILE WAS SUCCESSFULLY OPENED.

CALL WAITFR (IEV)
IF (IEV) 115, 115, 116
WRITE (63, 10) FLE, IEV
10 FORMAT (' *** TAPE FILE"', A5, ' PNB" COULDN'T BE OPENED,' /
     1 5X, 'EVENT VARIABLE1', I5)
GO TO 100

COMPUTE THE BASE FREQUENCY FOR THE RESULTS.

WRITE (63, 22)
22 FORMAT (' ENTER TIME OR DISTANCE SPACING BETWEEN POINTS /
     1 USED, F10.4,')
READ (63, 23) FRQ
23 FORMAT (F10.4)
FRQ = 1.0/(FRQ*FLOAT(N))
WRITE (63, 24)
24 FORMAT (' ENTER UNIT OF MEASURE, A4,')
READ (63, 2) UNT

READ IN THE DATA, CONVERTING TO FLOATING FORMAT, AND STORING IN THE COMMON BLOCK "DATA".

INX = 0
ISK = ISK + 33
ISK = ISK - 32
READ (62, END=151, ERR=152) IDA
123 ISM = ISK - 32
IF (ISM) 122, 122, 123
122 INX = INX + 1
CALL ADCNV (IDA(ISK), X(INX))
IF (INX-NR) 125, 126, 126
125 ISK = ISK + ISP
GO TO 124
126 INX = INX + 1
DO 201 ISM = INX, N
201 X(ISM) = 0.0

CLOSE THE TAPE FILE, AND GO TO SUBROUTINE BTA14B TO CONTINUE.

CALL CLOSE (62, FLE, 3HPNB, LUN)
RETURN

ERROR CONDITIONS FOUND IN READING THE TAPE.

WRITE (63, 20) FLE
20 FORMAT (' *** UNEXPECTED END-OF-FILE REACHED, TAPE FILE "'
     1 A5, ' PNB"'
153 CALL CLOSE (62, FLE, 3HPNB, IEV)
CALL WAITFR (IEV)
GO TO 100
152 WRITE (63, 21) FLE
21 FORMAT (' *** ERROR IN READING TAPE FILE "', A5, ' PNB",')
GO TO 153
END
SUBROUTINE BTA14B, THE SECOND OVERLAY OF TASK BTA14. THIS
SUBROUTINE IS A CONTINUATION OF SUBROUTINE BTA14A, IF
REQUIRED, THE DATA IS PRINTED, DE-TRENDED, AND PRINTED AFTER
DE-TRENDING. ALSO, IT IS DETERMINED IF SUBROUTINE BTA14G,
FOR COSINE TAPERING, IS TO BE CALLED.

SUBROUTINE BTA14B

COMMON /PARAM/ M, N, ICS, FLE, FRQ, UNT, IPL, PMX, FLS, LUN, ISD,
1 INV, IERR, ITR, B0, B1
COMMON /DATA/ X(2050)
DATA YES/3HYES/

WRITE (63,11)
11 FORMAT (' PRINT DATA?')
READ (63,12) QRY
12 FORMAT (A3)
IF (QRY .NE. YES) GO TO 150
WRITE (64,13) FLE, N
13 FORMAT (' TASK BTA14, DATA FILE '', A5, ' PNB'', 10X,
1 ' ORIGINAL DATA, ', 14, ' POINTS.')
WRITE (64,14)
14 FORMAT ('X', 'N', 9X, 'X(N)', 9X, 'X(N+1)', 9X, 'X(N+2)',
1 9X, 'X(N+3)', 9X, 'X(N+4)', 9X, 'X(N+5)', 9X, 'X(N+6)',
2 9X, 'X(N+7)')
DO 301 INX = 1, N, 8
ISM = INX -1
ISK = INX + 7
WRITE (64,15) ISM, (X(ISP), ISP = INX, ISK)
301 CONTINUE

Determine if de-trending is desired.

WRITE (63,21)
21 FORMAT (' SUBTRACT THE MEAN?')
ITR = -1
READ (63,12) QRY
IF (QRY .NE. YES) GO TO 151
ITR = 0
GO TO 152
22 FORMAT (' SUBTRACT 1ST ORDER TREND?')
READ (63,12) QRY
IF (QRY .NE. YES) GO TO 131
ITR = 1

Compute the sample mean, and print.

AVG = 0.0
DO 211 INX = 1, N
211 AVG = AVG + X(INX)
AN = N
AVG = AVG/AN
WRITE (64,23) N, FLE, AVG
23 FORMAT (' THE SAMPLE MEAN FOR ', 15, ' DATA POINTS FROM FILE'
1 '', A5, ' PNB'' IS', 1PE11.4)
IF (ITR) 131, 153, 154
B0 = AVG
DO 213 ISM = 1, N
   X(ISM) = X(ISM) - B0
GO TO 159
C
C COMPUTE THE LINEAR LEAST-SQUARES FIT, AND SUBTRACT THE LINE, C IF REQUIRED.
C
154 AVGN = 0, 0
   DO 212 INX = 1, N
   212 AVGN = AVGN + X(INX)*FLOAT(INX)
   ORY = (AN-1, 0)*AN
   B0 = (FLOAT(4*N + 2) * AVG - 6, 0*AVGN) / ORY
   B1 = (12, 0*AVGN - FLOAT(6*N + 6) * AVG) / (ORY*AN^2)
   WRITE (64, 25) B0, B1
   FORMAT (' THE DATA HAVE BEEN DE-TRENDED BY SUBTRACTING: 
   1, 1PE12.4, , * ', E12.4, , ' K, K = 1, 2, ..., N.')
   DO 216 J = 1, N
216 X(J) = X(J) - B0 - B1*FLOAT(J)
C
C PRINT THE ADJUSTED DATA, IF REQUIRED.
C
159 WRITE (63, 26)
   26 FORMAT (' PRINT DE-TRENDED DATA?')
   READ (63, 12) ORY
   IF (ORY .NE. YES) GO TO 131
   WRITE (64, 27) FLE
27 FORMAT (' DATA FROM FILE '', A5, ', PNB'' AFTER DE-TRENDING!')
   WRITE (64, 14)
   DO 217 INX = 1, N, 8
   217 ISM = INX - 1
   ISK = INX * 7
   WRITE (64, 15) ISM, (X(ISP), ISP = INX, ISK)
   CONTINUE
C
131 WRITE (63, 16)
   16 FORMAT (' COSINE TAPER (10%) THE DATA?')
   READ (63, 12) ORY
   ICS = 0
   IF (ORY .NE. YES) GO TO 132
   ICS = 1
C
C CHECK TO INSURE THE TAPE FILE OPENED BY SUBROUTINE BTA14A WAS C CLOSED, THEN CONTINUE WITH SUBROUTINE BTA14C.
C
132 CALL WAITFR (LUN)
   IF (LUN) 141, 141, 142
   141 WRITE (63, 19) FLE, LUN
19 FORMAT (' *** ERROR IN CLOSING TAPE FILE '', A5, ', PNB'' / 
1 5X, 'EVENT VARIABLE1', IS)
   142 RETURN
END
SUBROUTINE BTA14C, THE THIRD OVERLAY OF TASK BTA14. THIS
SUBROUTINE COMPUTES THE FOURIER COEFFICIENTS FOR THE REAL-
VALUED DATA, AND STORES THEM IN THE COMMON ARRAY ORIGINALLY
USED FOR THE DATA. ALTERNATELY, THE FOURIER SERIES (VALUES
SHOULD EQUAL THE ORIGINAL DATA) MAY BE COMPUTED FROM THE
COEFFICIENTS. IFS = -1 FOR THE COEFFICIENTS; = +1 FOR THE
SERIES.

THIS OVERLAY REQUIRES SUBROUTINES RFORT AND FORT TO COMPUTE
THE SERIES AND COEFFICIENTS. THESE SUBROUTINES ARE THE
FAST FOURIER TRANSFORM ROUTINES FOR REAL VALUED DATA.
LINK3=BTA14C,RFORT,FORT

SUBROUTINE BTA14C (IFS)

DIMENSION S(512)
COMMON /PARAM/ M,N,ICS,FLE,FRO,UNT, IPL,PMX,FLS,LUN,ISD,
1 INV,IERR,ITR,B0,B1
COMMON /DATA/ X(2050)

CALL RFORT (X,M,S,IFS,IERR)

RETURN
END
SUBROUTINE RFOURT (A,M,S,IFS,IFERR)

ONE-DIMENSIONAL REAL FINITE FOURIER TRANSFORM.

FOURIER TRANSFORM SUBROUTINE FOR REAL DATA, TAKEN FROM
IBM SHARE LIBRARY.

THIS PROGRAM USES THE SUBROUTINE RDOT TO COMPUTE COMPLEX
FOURIER TRANSFORMS OF REAL DATA.

THE FOURIER SERIES IS

\[ X(j) = \sum_{k=0}^{N} C(k) \exp(2\pi i j k/N) \]

WHERE i = \sqrt{-1} AND WHERE C(k) IS COMPLEX.

SINCE X(j) IS REAL, C(k) = \text{CONJ}(C(N-k)), THEREFORE ONLY
C(k), k=0,1,...,N/2 ARE COMPUTED AND/OR USED.

ARGUMENTS -
A IS INITIALLY THE INPUT ARRAY, X, WHEN COMPUTING A FOURIER
TRANSFORM AND C WHEN COMPUTING A FOURIER SERIES, A IS RE-
PLACED BY THE OUTPUT ARRAY, C, ON THE FORMER CASE, X ON THE
LATTER. THE X VECTOR CONTAINS THE REAL DATA X(0), X(1),
..., X(N-1), THE C VECTOR CONTAINS THE COMPLEX FOURIER
AMPLITUDES C(0), C(1), ..., C(N/2). THE COMPLEX VECTOR C
IS STORED ACCORDING TO THE NORMAL FORTRAN IV CONVENTION FOR
STORING COMPLEX NUMBERS, I.E., REAL PARTS IN ALTERNATE CELLS
STARTING WITH THE FIRST, IMAGINARY PARTS IN ALTERNATE CELLS
STARTING WITH THE SECOND, TO ADHERE TO FORTRAN RULES, X(0),
X(1), ..., ARE REFERRED TO AS X(1), X(2), ..., RESPECTIVELY
IN THE PROGRAMS. ALSO, C(0), C(1), ..., ARE REFERRED TO AS
C(1), C(2), ..., RESPECTIVELY, IF C IS DESIGNATED AS COMPLEX
IN A TYPE STATEMENT.

M GIVES N=2**M

THE ARGUMENTS S, IFS, AND IFERR ARE THE SAME AS IN SUB-
ROUTINE RDOT AND THE USER IS REFERRED TO COMMENT LINES
IN RDOT FOR THEIR EXPLANATION.

DIMENSION STATEMENTS - THE DIMENSIONS OF ARRAYS A AND S SHOULD
BE N+2 AND N/4, RESPECTIVELY FOR THE LARGEST N USED, FOR
EXAMPLE, IF THE LARGEST M IS 11, THEN N=2048 AND ONE SHOULD
HAVE THE DIMENSION STATEMENT -
DIMENSION A(2050), S(512)

IF ONE WISHES TO SPECIFY A TO BE COMPLEX BY A TYPE STATEMENT,
ONE SHOULD GIVE IT A DIMENSION OF N/2+1, FOR THE LARGEST N.

DIMENSION A(1), S(1)
IFERRS=0
N=2**M
NV2=N/2
NV4M1=N/4-1
MM1=M-1
IF(IABS(IFS)-1) 10,10,5
IF(MP-MM1) 6,20,20
IFERRS=1
C COMPUTE SINE TABLE
C
10 NP=N
MP=M
CALL FORT (A, M, S, 0, IFERR1)
IFERRS=IFERRS+IFERR1
20 KD=NP/N
KT=KD
NPV4=NP/4
IF(IFS)30, 50, 60
C
C COMPUTE FOURIER TRANSFORM.
C
30 CALL FORT (A, MM1, S, -2, IFERR2)
IFERRS=IFERRS+IFERR2
DO 40 K=1, NV4M1
J=NV2-K
A1R=A(2*K+1)+A(2*J+1)
A1I=A(2*K+2)-A(2*J+2)
A2R=A(2*K+1)+A(2*J+2)
A2I=A(2*K+1)-A(2*J+1)
KKT=NPV4-KT
AWR=A2R*S(KK)+A2I*S(KT)
AWI=A2I*S(KK)-A2R*S(KT)
A(2*K+1)=(A1R+AWR)/4,

A(2*K+2)=(A1I+AWI)/4,

A(2*J+1)=(A1R-AWR)/4,

A(2*J+2)=(A1I+AWI)/4,

40 KT=KT+KD
T=A1
A(1)=(T+A(2))/2,
A(N+1)=(T-A(2))/2,
A(2)=0,
A(N+2)=0,
A(NV2+1)=0.5*A(NV2+1)
A(NV2+2)=(-0.5*A(NV2+2))
50 IFERR=IFERRS
RETURN
C
C COMPUTE FOURIER SERIES.
C
60 DO 80 K=1, NV4M1
J=NV2-K
A1R=A(2*K+1)+A(2*J+1)
A1I=A(2*K+2)-A(2*J+2)
AWR=A(2*K+1)-A(2*J+1)
AWI=A(2*K+2)+A(2*J+2)
KKT=NPV4-KT
A2R=AWR*S(KK)+AWI*S(KT)
A2I=AWR*S(KK)-AWI*S(KT)
A(2*K+1)=A1R-A2I
A(2*K+2)=A1I+A2R
A(2*J+1)=A1R+A2I
A(2*J+2)=A2R-A1I
80 KT=KT+KD
T=A1
A(1)=T*A(N+1)
A(2)=T*A(N+1)
A(NV2+1)=2*A(NV2+1)
A(NV2+2)=(-2.*A(NV2+2))
CALL FORT (A, MM1, S, 2, IFERR2)
IFERRS=IFERRS+IFERR2
GO TO 50
END
SUBROUTINE FORT (A,M,S,IFS,IFERR)

FORT, ONE-DIMENSIONAL FINITE COMPLEX FOURIER TRANSFORM,
FOURIER TRANSFORMING SUBROUTINE, TAKEN FROM IBM SHARE LIBRARY,
DOES EITHER FOURIER SYNTHESIS, I.E., COMPUTES COMPLEX FOURIER
SERIES GIVEN A VECTOR OF N COMPLEX FOURIER AMPLITUDES, OR,
GIVEN A VECTOR OF COMPLEX DATA X, DOES FOURIER ANALYSIS,
COMPUTING AMPLITUDES.

A IS A COMPLEX VECTOR OF LENGTH N = 2**M COMPLEX NUMBERS OR
2*N REAL NUMBERS. A IS TO BE SET BY THE USER,
M IS AN INTEGER 0<M<11, SET BY THE USER,
S IS A VECTOR S(J) = SIN(2*PI*J/NP), J = 1, 2, ..., NP/4-1,
COMPUTED BY THE PROGRAM.
IFS IS A PARAMETER TO BE SET BY THE USER AS FOLLOWS-
IFS = 0 TO SET NP = 2**M AND SET UP SINE TABLE,
IFS = 1 TO SET N = NP = 2**M, SET UP SINE TABLE, AND DO
FOURIER SYNTHESIS, REPLACING THE VECTOR A BY
X(J) = SUM OVER K = 0, N-1 OF A(K)*EXP(2*PI*I/N)*EX(J*K),
J = 0, N-1, WHERE I = SQRT(-1)
THE X'S ARE STORED WITH RECJx(J) IN CELL 2*J+1 AND IMCx(J)
IN CELL 2*J+2 FOR J = 0, 1, 2, ..., N-1.
THE A'S ARE STORED IN THE SAME MANNER,
IFS = -1 TO SET N = NP = 2**M, SET UP SINE TABLE, AND DO
FOURIER ANALYSIS, TAKING THE INPUT VECTOR A AS X AND REPLACING
IT BY THE A SATISFYING THE ABOVE FOURIER SERIES.
IFS = +2 TO DO FOURIER SYNTHESIS ONLY, WITH A PRE-COMPUTED S,
IFS = -2 TO DO FOURIER ANALYSIS ONLY, WITH A PRE-COMPUTED S,
IFERR IS SET BY THE PROGRAM TO-
= 0 IF NO ERROR DETECTED,
= 1 IF M IS OUT OF RANGE, OR, WHEN IFS = +2 OR -2, THE
PRE-COMPUTED S TABLE IS NOT LARGE ENOUGH,
= -1 WHEN IFS = +1 OR -1, MEANS ONE IS RECOMPUTING S TABLE
UNNECESSARILY.

NOTE: IT MAY BE NECESSARY TO IMPOSE A LOWER MAXIMUM BOUND ON
M, DEPENDING ON THE CORE AVAILABLE FOR STORAGE, ALSO, IT MAY
POSSIBLE TO INCORPORATE A HIGHER VALUE BY EXTENDING THE K
ARRAY AND ADDING MORE "DO 30 ..." LOOPS,

DIMENSION A(1),S(1),K(12)
EQUIVALENCE (K(11),K1), (K(10),K2), (K(9),K3), (K(8),K4),
(1,K(7),K5), (K(6),K6), (K(5),K7), (K(4),K8), (K(3),K9),
(2,K(2),K10), (K(1),K11,N2)
IF (M) 2, 2, 3
IF (M-11) 5, 5, 2
2 IFERR=1
1 RETURN
5 IFERR=0
N=2**M
IF (IARS(IFS)-1) 200,200,10
C WE ARE DOING TRANSFORM ONLY. SEE IF PRE-COMPUTED S TABLE
IS SUFFICIENTLY LARGE.
10 IF (N-NP) 20, 20, 12
12 IFERR = 1
GO TO 200
C SCRAMBLE A, BY SANDE'S METHOD.
20     K(1) = 2*N
       DO 22 L = 2,M
22     K(L) = K(L-1)/2
       DO 24 L = M,10
24     K(L+1) = 2
C  NOTE THE EQUIVALENCE OF KL AND K(L-1)
C  BINARY SORT -
   IJ = 2
   DO 30 J1 = 2,K1,2
      J1 = 2
530    IF (J1-K1) 331,331,431
C  DO 30 J2 = J1,K2,K1
331     J2 = J1
332    IF (J2-K2) 332,332,432
332     DO 30 J3 = J2,K3,K2
332     DO 30 J4 = J3,K4,K3
332     DO 30 J5 = J4,K5,K4
332     DO 30 J6 = J5,K6,K5
332     DO 30 J7 = J6,K7,K6
332     DO 30 J8 = J7,K8,K7
332     DO 30 J9 = J8,K9,K8
332     DO 30 J10 = J9,K10,K9
331     IF (IJ-J1) 28,30,30
28     T = A(IJ-1)
28     A(IJ-1) = A(IJ-1)
28     A(IJ) = T
28     A(IJ+1) = T
   J2 = J2 + K1
   GO TO 531
432     J1 = J1 + 2
   GO TO 530
431    IF (IJS) 32,2,36
C  DOING FOURIER ANALYSIS, SO DIVIDE BY N AND CONJUGATE.
32    FN = N
32     DO 34 I = 1,N
34    A(2*I-1) = A(2*I-1)/FN
34    A(2*I) = (-A(2*I))/FN
C  SPECIAL CASE - L=1
36     DO 40 I = 1,N,2
36     T = A(2*I-1)
36    A(2*I-1) = T*A(2*I+1)
36    A(2*I+1) = T*A(2*I+1)
36    T = A(2*I)
36    A(2*I) = T*A(2*I+2)
36    A(2*I+2) = T*A(2*I+2)
   IF (M-1) 2,1,50
C  SET FOR L = 2
50     LEXP1 = 2
C  LEXP1 = 2*8*(L-1)
50     LEXP = 8
50     LEXP = 2*8*(L+1)
50     NPL = 2*8*MT
50     NPL = NPL + 2*8*(-L)
60     DO 130 L = 2,M
C  SPECIAL CASE - J=0
   DO 80 I = 2,N2,LEXP
51     I1 = I + LEXP1
51     I2 = I + LEXP1
51     I3 = I + LEXP1
51     T = A(I-1)
51    A(I-1) = T*A(I+1)
51    A(I+1) = T*A(I+1)
   A(I+1) = T*A(I+1)
   A(I+1) = T*A(I+1)
80     I = I + LEXP1
130    CONTINUE
T=A(1)  
A(1)=T+A(12)  
A(12)=T-A(12)  
T=(-A(13))  
TI=A(13-1)  
A(13-1)=A(I1-1)-T  
A(13)=A(I1)-TI  
A(I1)=A(I1)+TI  
IF(L-2)=120,120,90  
90 KLAST=N2-LEXP  
JJ=NPL  
DO 110 J=4,LEXP1,2  
NPJJ = NT - JJ  
UR = S(NPJJ)  
UI = S(JJ)  
ILAST = J* KLAST  
DO 100 I = J,ILAST,LEXP  
I1 = I + LEXP1  
I2=I1+LEXP1  
I3=I2+LEXP1  
T=A(12-1)*UR-A(12)*UI  
TI=A(I2-1)*UI-A(12)*UR  
A(I2-1)=A(I-1)-T  
A(I2)=A(I)+TI  
A(I-1)=A(I)+TI  
A(I)=A(I)+TI  
T=(-A(13-1)*UI-A(13)*UR)  
TI=A(13-1)*UR-A(13)*UI  
A(13-1)=A(I1-1)-T  
A(13)=A(I1)-TI  
A(I1-1)=A(I1-1)+T  
A(I1)=A(I1)+TI  
100 C END OF I LOOP.  
110 JJ=JJ+NPL  
C END OF J LOOP.  
120 LEXP1=2*LEXP1  
LEXP=2*LEXP  
130 NPL=NPL/2  
C END OF L LOOP.  
140 IF (IFS) 145,2,1  
C DOING FOURIER ANALYSIS, REPLACE A BY CONJUGATE.  
145 DO 150 I=1,N  
150 A(2*I)=(-A(2*I))  
C 160 GO TO 1  
RETURN  
C MAKE TABLE OF S(J)=SIN(2*PI*J/NP), J=1,2,...,NT-1, NT=NP/4  
200 NP=N  
MP=M  
NT=N/4  
MT=M+2  
IF(MT) 205,205,205  
205 THETA=0.78539816  
C THETA = PI/2**(L+1) FOR L=1  
210 JSTEP=NT  
C JSTEP = 2**(MT-L+1) FOR L=1  
215 JDIF=NT/2  
C JDIF = 2**(MT-L) FOR L=1  
S(JDIF)=SIN(THETA)  
IF (MT-2) 220,220,220  
220 DO 220 L = 2,NT  
THETA = THETA/2.  
JSTEP2=JSTEP  
JSTEP=JDIF  
JDIF=JDIF/2
S(JDIFF) = SIN(THETA)
JC1 = NT - JDIF
S(JC1) = COS(THETA)
JLAST = NT - JSTEP2
IF(JLAST = JSTEP) 250, 230, 230

DO 240 J = JSTEP, JLAST, JSTEP
JC = NT - J
JD = J + JDIF
240 S(JD) = S(J) * S(JC1) + S(JDIFF) * S(JC)
CONTINUE
260 IF(IFS) 20, 1, 20
END
SUBROUTINE BTA140, THE FOURTH OVERLAY OF TASK BTA14. THIS
SUBROUTINE PRINTS THE FOURIER COEFFICIENTS IF REQUESTED,
Saves them in an operator specified disk or tape file
IF REQUESTED, AND CONTINUES THE OPERATOR QUESTIONING.

SUBROUTINE BTA140

DIMENSION XX(14)
COMMON /PARAM/ M,N,ICS,FLE,FRO,UNT,IPX,PMX,FLS,LUN,ISD,
1 INV,IERR,ITR,B0,B1
COMMON /DATA/ X(2050)
DATA YES /3HYES/ DATA XX /14*0,0/

CHECK TO INSURE THAT THE FOURIER COEFFICIENTS WERE COMPUTED.

INV = 0
ISD = 0
IF (IERR) 111, 112, 111
WRITE (63,6) IERR
6 FORMAT ('*** ERROR IN CALLING THE FAST FOURIER ROUTINE,'/
1 5X, 'IERR!', 15)
GO TO 132

CHECK IF FOURIER COEFFICIENTS ARE TO BE SAVED, AND IF SO,
OPEN THE FILE FOR THEM.

WRITE (63,11)
11 FORMAT ('SAVE THE FOURIER COEFFICIENTS?')
READ (63,2) QRY
ISV = 0
IF (QRY .NE. YES) GO TO 114
ISV = 1
WRITE (63,12)
12 FORMAT ('ENTER LUN, 12, AND FILE NAME, A5,')
READ (63,13) LU, FSV
13 FORMAT (12, A5)
CALL ENTER (LU, FSV, 3HPNB, IEV)

PRINT THE FOURIER COEFFICIENTS, IF REQUIRED.

WRITE (63,3)
3 FORMAT ('ARE THE FOURIER COEFFICIENTS TO BE PRINTED?')
READ (63,2) QRY
IF (QRY .NE. YES) GO TO 113
WRITE (64,4) FLE, FRO, UNT
4 FORMAT ('FOURIER COEFFICIENTS FOR DATA FROM FILE ", A5,
1 'PNB", HARMONICS OF FREQUENCY', 1PE11.4, 'PER ', A4 /
2 4X, 'K', 10X, 'A(K)', 20X, 'A(K+1)', 19X, 'A(K+2)',
3 19X, 'A(K+3)', 19X, 'A(K+4)'
N2 = N + 2
K = 0
ISR = 1
IST = 10
101 WRITE (64,5) K, (X(I), I=ISR,IST)
5 FORMAT (1H , I4, 5(1PE12.3, ' + ', E10.3, 'I'))
IST = IST + 10
ISR = ISR + 10
K = K + 5
IF (IST-N2) 101, 102, 102
WRITE (64,5) K, (X(I), I=ISR,N2)
C
C IF REQUESTED, WRITE THE FOURIER COEFFICIENTS TO TAPE OR DISK,
C
113 IF (IEV) 116, 115, 116
116 CALL WAITFR (IEV)
IF (IEV) 117, 117, 118
117 WRITE (63,14) FSV, LU, IEV
14 FORMAT ('*** FILE "', A5, ' PNB" ON LUN ', I3, ' CANNOT BE OPENED; 1 COEFFICIENTS WILL NOT BE SAVED,' / 5X, 'EVENT VARIABLE!'
2 15)
ISV = 0
GO TO 115
118 K = 0
ISR = 1
IST = 16
119 IF (IST-N) 121, 122, 122
122 WRITE (LU, ERR=401) (X(I), I=ISR,IST)
ISR = ISR * 16
IST = IST * 16
GO TO 119
121 IST = ISR + 1
WRITE (LU, ERR=401) X(ISR), X(IST), XX
124 CALL CLOSE (LU, FSV, 3HPNB, IEV)
C
C SEE IF SUBROUTINE BTA14E, COMPUTING THE POWER SPECTRAL DENSITY
C FUNCTION (PSDF), IS TO BE CALLED.
C
115 WRITE (63,1)
1 FORMAT ('IS THE PSDF TO BE COMPUTED?')
READ (63,2) QRY
2 FORMAT (A3)
IF (QRY ,EQ, YES) ISD = 1
C
C SEE IF THE INVERSE TRANSFORM IS TO BE USED TO RECOMPUTE THE
C ORIGINAL DATA.
C
7 WRITE (63,7)
7 FORMAT ('PERFORM INVERSE TRANSFORM?')
READ (63,2) QRY
IF (QRY ,EQ, YES) INV = 1
C
C CHECK FOR THE PROPER CLOSING OF THE STORAGE FILE, IF OPENED.
C
131 CALL WAITFR (IEV)
IF (IEV) 133, 133, 132
133 WRITE (63,15) FSV, LU, IEV
15 FORMAT ('*** FILE "', A5, ' PNB" ON LUN ', I5, ' FOR SAVING 1 THE FOURIER COEFFICIENTS WAS NOT CLOSED,' / 5X, 'EVENT 2 VARIABLE!';, I5)
132 RETURN
C
C ERROR DETECTED IN WRITING TO TAPE OR DISK.
C
401 WRITE (63,16) FSV, LU
16 FORMAT ('*** WRITE ERROR IN SAVING THE FOURIER COEFFICIENTS 1 IN FILE "', A5, ' PNB" ON LUN ', I3, '.')
GO TO 124
C
END
SUBROUTINE BTA14E, THE FIFTH OVERLAY OF TASK BTA14. THIS SUBROUTINE COMPUTES, PRINTS, AND SAVES ON TAPE OR DISK THE POWER SPECTRAL DENSITY FUNCTION (PSDF), ACCORDING TO OPERATOR RESPONSE. THIS SUBROUTINE ASSUMES THE FOURIER COEFFICIENTS HAVE BEEN PLACED IN COMMON ARRAY X BY SUBROUTINES RFORT AND FORT, WHICH THEREFORE MUST HAVE BEEN CALLED PREVIOUSLY. IF THE PSDF IS SAVED, IT IS WRITTEN IN BLOCKS OF 16 FLOATING POINT NUMBERS, WITH 0.0 TO FILL THE LAST BLOCK AFTER THE LAST OF THE N/2+1 VALUES OF THE FUNCTION.

SUBROUTINE BTA14E

DIMENSION PSD(16)
COMMON /PARAM/ M,N,ICS,FLE,FRQ,UNT,1PL,PHX,FLS,LUN,ISD,
1 INV,IERR,ITR,B0,B1
COMMON /DATA/ X(2050)
DATA YES/3HYES/, IPS/0/, IPR/0/

OBTAIN THE OPERATOR'S DESIRES CONCERNING THE PSDF,

PMX = 0.0
IPL = 0
WRITE (63,12)
12 FORMAT (' IS THE PSDF TO BE PLOTTED?')
READ (63,2) ORY
IF (ORY .NE. YES) GO TO 101
IPL = 1
GO TO 102
101 WRITE (63,1)
1 FORMAT (' IS THE PSDF TO BE SAVED?')
READ (63,2) ORY
2 FORMAT (A3)
IF (ORY .NE. YES) GO TO 111
102 IPS = 1
WRITE (63,3)
3 FORMAT (' ENTER LUN AND SAVE FILE NAME, 12,1X,A5,')
READ (63,4) LUN, FLS
4 FORMAT (12,1X,A5)
OPEN THE SAVE FILE IF REQUIRED, THEN PROCEED WITH THE OPERATOR QUESTIONS,

CALL ENTER (LUN, FLS, 3HPNB, IEV)
WRITE (63,5)
5 FORMAT (' IS THE PSDF TO BE PRINTED?')
READ (63,2) ORY
IF (ORY .NE. YES) GO TO 112
PRINT HEADER IF REQUIRED,

WRITE (64,6) FLE, FRG, UNT
FORMAT ('POWER SPECTRAL DENSITY FUNCTION ESTIMATE, DATA FILE "", A5, ' PN'B, HARMONICS OF FREQUENCY',1PE11.4,
2 'PER', A4)
WRITE (64,7)
7 FORMAT (4X,'K',1DX,'G(K)',1DX,'G(K+1)',9X,'G(K+2)',
1 9X,'G(K+3)',9X,'G(K+4)',9X,'G(K+5)',9X,'G(K+6)',
2 9X,'G(K+7)')
IPR = 1
IF (IPS) 113, 114, 113
112 IF (IPS) 113, 115, 113
C CHECK FOR THE SAVE FILE OPENING, IF REQUIRED.
C 113 CALL WAITFR (IEV)
IF (IEV) 116, 116, 114
116 IPS = 0
IPL = 0
WRITE (63,8) FLS, LUN, IEV
8 FORMAT ( 'PSD CANNOT BE SAVED (OR PLOTTED). FILE "', A5, 1 'PNB" ON LUN' ', I3, ' NOT OPENED.', / 5X, 'EVENT VARIABLE:', 2 I5)
C COMPUTE THE PSDF AND PRINT OR STORE, AS REQUIRED.
C 114 K = 0
J = 0
N2 = N+2
C THE FACTOR FOR THE PSDF IS DIFFERENT IF COSINE TAPERING HAS BEEN USED ON THE DATA. ADJUST IF NEEDED, THEN CONTINUE WITH THE CALCULATION.
C FTR = 2.0/FRO
IF (ICS) 161, 162, 161
FTR = FTR/0.875
161 DO 201 I = 1, N2, 2
 I1 = I+1
 J = J+1
 PSD(J) = (X(I)*X(I1) + X(I1)*X(I1)) * FTR
IF (IPL) 119, 117, 119
119 IF (PMX - PSD(J)) 118, 117, 117
118 PMX = PSD(J)
117 IF (J-8) 121,122,121
122 IF (IPR) 123,201,123
123 WRITE (64,9) K, (PSD(K1), K1=1,8)
9 FORMAT (1H 14, 8(1PE15.3))
 K = K+8
GO TO 201
121 IF (J-16) 201, 124, 124
124 J = 0
IF (IPR) 126, 125, 126
126 WRITE (64,9) K, (PSD(K1), K1 = 9,16)
 K = K+8
125 IF (IPS) 127, 201, 127
127 WRITE (LUN,ERR=301) PSD
201 CONTINUE
IF (IPR) 131, 132, 131
131 JS = 1
IF (J-8) 133, 133, 134
134 JS = 9
133 WRITE (64,9) K, (PSD(K1), K1 = JS,J)
132 IF (IPS) 141, 115, 141
141 JS = J+1
DO 202 K1 = JS,16
202 PSD(K1) = 0.0
WRITE (LUN,ERR=302) PSD
C FINISHED WITH THE PSDF CALCULATION. CLOSE THE SAVE FILE IF THE FUNCTION IS TO BE SAVED.
C 152 CALL CLOSE (LUN, FLS, 3HPNB, IEV)
CALL WAITFR (IEV)
IF (IEV) 151, 151, 115
115 RETURN
C 151 WRITE (63,10) FLS, LUN, IEV
10 FORMAT ('*** FILE "", A5, ' PNB" ON LUN', I3, ' COULD
1 NOT BE CLOSED,' / 5X, 'EVENT VARIABLE:', I5)
GO TO 115
C C WRITE ERRORS DETECTED, GIVE DIAGNOSTIC MESSAGE.
C 301 WRITE (63,11) LUN, FLS
11 FORMAT ('*** WRITE ERROR DETECTED, LUN', I3, ',', FILE ''
1 A5, ' PNB"."
CALL CLOSE (LUN, FLS, 3HPNB, IEV)
CALL WAITFR (IEV)
IPS = 0
GO TO 201
302 WRITE (63,11) LUN, FLS
GO TO 152
END
SUBROUTINE BTA14F, SIXTH OVERLAY FOR TASK BTA14. THIS
SUBROUTINE PLots THE POWER SPECTRAL DENSITY FUNCTION
(PSDF). THE PLOT IS LOG PSDF VS. EITHER SAMPLE NUMBER
OR LOG SAMPLE NUMBER. THE RANGE FOR THE PSDF IS UP TO
SIX DECADES (VALUES MORE THAN SIX ORDERS OF MAGNITUDE
BELOW THE MAXIMUM VALUE ARE PLOTTED AS BEING SIX DECADES
RELOW). THE PLOT IS UP TO 3 INCHES HIGH AND IS 8 INCHES
LONG. THE D.C. OFFSET IS NOT PLOTTED, AND THE OPERATOR MAY
SPECIFY THE NUMBER OF HARMONICS USED.

THIS SUBROUTINE EXPECTS THE PSDF TO BE ON A MASS STORAGE
DEVICE (WITH THE LOGICAL UNIT NUMBER PASSED THROUGH COMMON
AS VARIABLE LUN AND THE FILE NAME AS VARIABLE FLS).

THIS SUBROUTINE REQUIRES SUBROUTINE PLOT TO BE IN THE SAME
OVERLAY:
LINK6=BTA14F, PLOT

SUBROUTINE BTA14F

DIMENSION PSD(16)
COMMON /PARAM/ M, N, ICS, FLE, FREQ, UNT, IPL, PMX, FLS, LUN, ISO,
1 INV, IERR, ITR, B0, B1
DATA YES/3HYES/

POSITION THE PEN INITIALLY,
CALL PLOT (0.0, 0.0, 0.0, 1)

OPEN THE MASS STORAGE FILE CONTAINING THE PSDF VALUES,
CALL SEEK (LUN, FLS, 3HPNLB, IEV)

OBTAIN THE DETAILS OF THE PLOT FROM THE OPERATOR,
WRITE (63,1) PMX
1 FORMAT (' FOR THE PSDF PLOT, THE PEAK VALUE IS ', 1PE11.4
1 / ' IS THIS TO BE USED?')
READ (63,2) QRY
2 FORMAT (A3)
IF (QRY 'EQ.' YES) GO TO 101
WRITE (63,3)
3 FORMAT (' ENTER VALUE TO REPLACE MAX, FOR SCALING, E12.4, ')
READ (63,4) SMX
4 FORMAT (E12.4)
GO TO 102
101 SMX = PMX
102 SMX = ALOG10(SMX)
WRITE (63,5)
5 FORMAT (' ENTER NUMBER OF POINTS TO BE PLOTTED, 14, ')
N2 = N/2
READ (63,6) NP
6 FORMAT (14)
IF (NP - N2) 103, 103, 104
104 NP = N2
103 SNP = NP
WRITE (63,7)
7 FORMAT (' IS THIS A SEMI-LOG PLOT?')
READ (63,2) QRY
LOG = 0
IF (Q.R.Y .EQ. YES) GO TO 105
LOG = 1
SNP = ALOG10(SNP)
105 SNP = 8.0/SNP
C
C CHECK TO INSURE THE FILE IS OPEN.
C
CALL WAITFR (IEV)
IF (IEV) 111, 111, 112
111 WRITE (63,8) FLS, LUN, IEV
8 FORMAT (' *** FILE "", A5, ' PNB" ON LUN', I3, ' COULD
1 NOT BE OPENED.' / 5X, 'EVENT VARIABLE:' ', I5, 5X, 'NO
2 PLOT.' )
GO TO 132

C READ THE PSDF FILE (TAPE OR DISK) AND PLOT THE VALUES.
C
I = 0
112 READ (LUN,END=301,ERR=302) PSD
IF (I) 114, 114, 115
114 J = 2
GO TO 116
115 J = 1
116 I = I + 1
Y = (ALOG10(PSD(J)) - SMX)*0.5 + 3.0
IF (Y) 121, 122, 122
121 Y = 0.0
122 IF (LOG) 123, 124, 123
124 X = FLOAT(I) * SNP
GO TO 125
123 X = ALOG10(FLOAT(I)) * SNP
125 CALL PLOT (X, Y, 0)
IF (I - NP) 127, 128, 128
127 IF (J - 16) 113, 113, 113
126 J = J + 1
GO TO 116
128 CALL PLOT (X, Y, 1)
129 CALL CLOSE (LUN, FLS, 3HPNB, IEV)
CALL WAITFR (IEV)
IF (IEV) 131, 131, 132
131 WRITE (63,16) FLS, LUN, IEV
16 FORMAT (' ** ERROR IN CLOSING FILE "", A5, ' PNB" ON
1 LUN', I3, ', ' / 5X, 'EVENT VARIABLE:' ', I5)
132 RETURN

C PRINT ERRORS IN READING THE PSDF DATA FILE.
C
301 WRITE (63,17) FLS, LUN
17 FORMAT (' *** REACHED UNEXPECTED END OF FILE "", A5,
1 ' PNB" ON LUN', I3, ', ' PLOT ABORTED.' )
GO TO 128
302 WRITE (63,18) FLS, LUN
18 FORMAT (' ** ERROR IN READING FILE "", A5, ' PNB" ON
1 LUN', I3, ', ' PLOT ABORTED.' )
GO TO 128
END
SUBROUTINE PLOT TO CONTROL THE SMALL FLAT-BED PLOTTER

NORMALLY USED WITH THE ANALOG COMPUTERS. THIS SUBROUTINE
ASSUMES THAT DAC LINES 1 AND 2 ARE CONNECTED TO THE X AND
Y PLOT SERVOS, RESPECTIVELY, AND THAT DAC LINE 4 CONTROLS
THE PEN POSITION. THESE DAC LINES SHOULD BE ASSIGNED TO
THE TASK CALLING THIS SUBROUTINE.

SUBROUTINE PLOT (X,Y,IPEN,IEV)
DIMENSION P(3), IOUT(3), IT(2), ITZ(2)
DATA ITZ/20,1/, IT/3,1/, IOUT/5,6,4/, P/0,0,0,0,0,0/

C PEN CODE: IF IPEN IS 0, PEN IS TO BE DOWN, OTHERWISE, PEN
IS TO BE UP.

IF (IPEN) 11,2,11
P(3)=0.0
GO TO 1

C PEN DOWN - BREAK LINE INTO SEGMENTS, (IF PEN UP, DO
FULL INTERVAL AT ONCE)

2 P(3) = 9',99
XDIF=X-P(1)
XDIF=ABS(XDIF)
YDIF=Y-P(2)
YDIF=ABS(YDIF)

C BASE NUMBER OF SEGMENTS ON THE SHORTEST DISTANCE:
C EACH SEGMENT = 1/20 TH OF AN INCH ALONG THE SHORTEST
C DIRECTION.
DINT=AXDIF
IF (AYDIF-AXDIF) 3,4,4
DINT=AYDIF
4 DINT=DINT/20;
N=DINT
TRINT=N+1
XINT=XDIF/TRINT
YINT=YDIF/TRINT
IF (N) 5,5,9
DO 5 J=1,N
P(1)=P(1)+XINT
P(2)=P(2)+YINT
CALL RTOUT (3,IOUT,P,IEV)
IF (IEV) 7,7,6
5 CONTINUE

7 RETURN
6 CALL MARK (IT,IEVM)
CALL WAITFR(IEVM)

C CONTINUE
1 P(1)=X
P(2)=Y
CALL RTOUT (3,IOUT,P,IEV)
IF (IEV) 7,7,8
8 CALL MARK (ITZ,IEVM)
CALL WAITFR(IEVM)
RETURN
END
SUBROUTINE BTA14G, THE SEVENTH OVERLAY OF TASK BTA14, PART ONE OF

THIS SUBROUTINE, FOR IGO = 3, COSINE TAPERS THE DATA, AND IF
REQUESTED, PRINT THE TAPERED DATA. PART TWO, FOR IGO = 1,
REMOVES THE TAPER FROM THE RE-COMPUTED DATA AS RETURNED BY
PREVIOUS CALLS TO RFT AND FORT TO INVERT THE TRANSFORM.

THIS OVERLAY MUST ALSO CONTAIN SUBROUTINE COSTP, FOR COSINE
TAPERING OF DATA ARRAYS, WHICH IS CALLED BY THIS SUBROUTINE,
LINK7=BTA14G,COSTP

SUBROUTINE BTA14G (IGO)

COMMON /PARAM/ M,N,ICS,FLE,FRQ,UNT,IPL,PMX,FLS,LUN,ISO,
  1 INV,IERR,ITR,B0,B1
COMMON /DATA/ X(2050)
DATA YES/3HYES/

IF (IGO) 101, 100, 101

PART ONE, COSINE TAPER THE DATA.

10 FORMAT (A3)
12 FORMAT (4X, 'N', 9X, 'X(N)', 9X, 'X(N+1)', 9X, 'X(N+2)',
  1 9X, 'X(N+3)', 9X, 'X(N+4)', 9X, 'X(N+5)', 9X, 'X(N+6)',
  2 9X, 'X(N+7)')
15 FORMAT (1H, 14, 8(1PE15.3))

100 R = 0.1
CALL COSTP (X, N, R, ISK, 0)
WRITE (63,17)
17 FORMAT ('PRINT TAPERED DATA?')
READ (63,12) QRY
IF (QRY .NE. YES) GO TO 132
WRITE (64, 18) FLE, N
18 FORMAT ('0 DATA FROM FILE A5, ' PNB' WITH A 10% COSINE
  1 TAPER,' , 15, ' POINTS TOTAL.')
WRITE (64,14)
DO 302 INX = 1,N,8
ISM = INX - 1
ISK = INX + 7
WRITE (64,15) ISM, (X(ISP), ISP = INX,ISK)
302 CONTINUE
132 RETURN

PART TWO, CALLED AFTER INVERTING THE FOURIER TRANSFORM, TO PRINT
THE RE-COMPUTED DATA.

101 IF (IERR) 111, 112, 111
111 WRITE (63,1) IERR
1 FORMAT ('*** ERROR IN INVERTING THE FOURIER TRANSFORM.' /
  1 5X, 'IERR:', 13)
112 IF (ICS) 103, 104, 103
103 R = 0.1
CALL COSTP (X, N, R, I, *1)
C
PUT THE TREND BACK IN IF REMOVED.

104 IF (ITR) 121,122,123
123 DO 202 ISP = 1,N
202 X(ISP) = X(ISP) + B0 + B1*FLOAT(ISP)
GO TO 121
122 DD 203 ISP = 1,N
203 X(ISP) = X(ISP) + B0
C PRINT THE RE-COMPUTED DATA,
C
121 WRITE (64,2) FILE
2 FORMAT ('RECOMPUTED DATA FROM FILE ', A5, ' PNB,')
WRITE (64,14)
DO 201 INX = 1,N
ISM = INX - 1
ISK = INX + 7
WRITE (64,15) ISM, (X(ISP), ISP=INX,ISK)
201 CONTINUE
RETURN
END
COSINE TAPER (DATA WINDOW) ROUTINE

THIS SUBROUTINE IS A MINOR MODIFICATION OF ONE WRITTEN
BY E. H. BLACKSTONE, M.D., U.S. ARMY AEROMEDICAL
RESEARCH LABORATORY, FORT RUCKER, NOW OF THE MEDICAL
CENTER, UNIVERSITY OF ALABAMA IN BIRMINGHAM.

----------------------------------------

SUBROUTINE COSTP

PURPOSE
TO APPLY COSINE TAPER TO BOTH ENDS OF A DATA SERIES
IN ORDER TO MINIMIZE THE SmeARING IN THE FREQUENCY
DOMAIN OF NON-DISCRETE FREQUENCIES PRESENT IN A TIME
ORDERED DATA SERIES, ALSO CALLED "DATA WINDOWING."
THE TAPER IN A CIRCULAR DIGITAL FOURIER TRANSFORM
BRINGS THE END AND BEGINNING OF THE DATA SMOOTHLY
TOGETHER TO PREVENT THIS SMEARING TO A DEGREE.
OPTIONALLY, THIS TAPERING MAY BE UNDONE.

USAGE
CALL COSTP(Y,NS,R,IERR,INV)

DESCRIPTION OF PARAMETERS

INPUT
Y - INPUT DATA ARRAY TO BE WINDOWED
NS - NUMBER OF DATA POINTS IN EACH SERIES
R - PROPORTION OF EACH END OF THE DATA SERIES
    TO BE TAPERED, VALUES SHOULD BE GREATER
    THAN 0.05 AND LESS THAN 0.5. THE LITER-
   ATURE SUGGESTS 0.1 AS A REASONABLE TAPER,
    THE MORE "SIGNAL-LIKE" THE WIGGLES, THE
    MORE TAPERING THAT CAN BE DONE.
INV - TAPER IF 0; UNTAPER IF NON-ZERO.

OUTPUT
Y - TAPERED SERIES (IN PLACE WINDOWING)
R - INPUT R WILL BE REPLACED BY THE ACTUAL
    NUMBER OF DATA POINTS EQUAL TO THE
    INITIAL PROPORTION. R=R/NS

ERROR
IERR - NS NOT GIVEN, R TOO SMALL OR TOO LARGE
0 - NO ERROR
1 - ERROR

REMARKS
1. AS CLEARLY EXPLAINED BY BINGHAM, ET AL., THE
   DISCRETE FOURIER TRANSFORM PERFECTLY TRANSFORMS
   ONLY THOSE DISCRETE FREQUENCIES OMEGA(J), ANY
   FREQUENCY NOT A J IS SmeARED OVER ALL FREQUENCIES
   WITH A DECAY CENTERED ABOUT THE NON-DISCRETE
   FREQUENCY "OMEGA" OF 1./ABS(Omega-Omega(K)) AS K
   RECEDES FROM OMEGA. SIGNS ALTERNATE, THIS DECAY
   REPRESENTS "LEAKAGE" AND IS OFTEN UNACCEPTABLE.
   THIS DATA WINDOW (COSINE TAPER) REDUCES LEAKAGE
   TO ABOUT 1./ABS((OMEGA - OMEGA(K))**3),
2. If an R less than 0.05 is given, a default value of 0.1 will be substituted.

Subroutines and function subprograms required

COSINE (COS), FLOAT

Method

Ref. Bingham, C., Godfrey, M. D., and Tukey, J. W.,
"Modern Techniques of Power Spectrum Estimation."
IEEE Trans. on Audio and Electroacoustics, Vol. AU-15,

\[ Y(T) = Y(T) \cdot \text{WINDOW}(T) \]

WHERE

\[ \text{WINDOW}(T) = 0.5 \cdot (1 - \cos(\pi (T-1)/2)/R) \]

For \( T = 1, \ldots, R \)

\[ \text{WINDOW}(T) = 1 \]

For \( T = (R+1), \ldots, (NS-R) \)

\[ \text{WINDOW}(T) = 0.5 \cdot (1 - \cos(\pi (NS-T+1)/2)/R) \]

For \( T = (NS-R+1), \ldots, NS \)

\[ PI = 3.1415927 \ldots \]

\( Y \) = vector of data points

This gives a weighting function which in the frequency domain looks like

\[ (1/N) \cdot \text{sum over all } T \text{ of the squared transform of } \]

the window weights, which in this case = \( N/(S/4) \cdot R \)

Subroutine COSTP(Y, NS, R, IERR, INV)

DIMENSION Y(1)

EQUIVALENCE (T, WINDOW)

IERR = 0

Check to see if data window is within the prescribed limits as set
forth above. If not, error flag is set.

IF (NS) 1010,1010,1011
1011 IF (R-0.05) 1012,1013,1013
1012 R = 0.1
IERR = 1
GO TO 1014
1013 IF (R-0.5) 1014,1010,1010

Set up appropriate constants to minimize calculations within the
DO loop.

1014 R = FLOAT(NS) + 0.01
IR = R
IF (IR) 1010,1010,1020
1020 C1 = 3.141593/R
NSP1 = NS + 1
DO 30 i=1,IR
T = I
TMHAF = T-0.5

Taper weight formed

C1A = C1*TMHAF
WINDOW = 0.5-0.5*COS(C1A)
IF(INV) 12,11,12
12 WINDOW = 1./WINDOW

Apply taper to both ends of data
C

11 Y(I) = WINDOW*Y(I)
   J = NSP1-I
30 Y(J) = WINDOW*Y(J)
100 RETURN
1010 IERR = 1
   GO TO 100
   END
APPENDIX D

Additional Information on Equipment Characteristics

On the next several pages, some of the details of the equipment used in the polymer sheet thickness study are given. Figure 13 is the frequency response (Bode) diagram for the single-stage R-C filters used with the thermocouples in this study. Figure 14 is the Bode diagram for the three-stage R-C filters used with the pressure transducers. Table 4 gives the calibration equation constants for the thermocouples and pressure transducers, as provided by Fontaine (in preparation). The calibration curve for the Beta Gauge for the sheet thickness range of Run 2 is given in Figure 15.
Figure 13 Single-Stage Filter Response
FIGURE 14 THREE-STAGE FILTER RESPONSE
TABLE 4

CALIBRATION CONSTANTS

\[ T \text{ or } P = a + b \cdot V + c \cdot V^2, \]

where \( T \) is temperature in degrees Celsius, \( P \) is pressure in pounds per square inch, and \( V \) is the analog-to-digital converter (channel ADC) voltage in scaled integer form (as returned by subroutine IADCV).

<table>
<thead>
<tr>
<th>ADC</th>
<th>Position</th>
<th>a</th>
<th>b</th>
<th>c \cdot 10^7</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>Barrel position 1 (near feed)</td>
<td>124.389</td>
<td>0.0106747</td>
<td>0.0</td>
</tr>
<tr>
<td>35</td>
<td>Barrel position 3</td>
<td>123.980</td>
<td>0.0104913</td>
<td>0.0</td>
</tr>
<tr>
<td>37</td>
<td>Barrel position 5</td>
<td>122.159</td>
<td>0.0107254</td>
<td>0.0</td>
</tr>
<tr>
<td>39</td>
<td>Barrel position 7 (head)</td>
<td>123.464</td>
<td>0.0106302</td>
<td>0.0</td>
</tr>
<tr>
<td>38</td>
<td>Adapter inlet</td>
<td>121.562</td>
<td>0.0109017</td>
<td>0.0</td>
</tr>
<tr>
<td>40</td>
<td>Adapter outlet</td>
<td>123.042</td>
<td>0.0110383</td>
<td>0.0</td>
</tr>
<tr>
<td>34</td>
<td>Die entrance</td>
<td>125.014</td>
<td>0.0107122</td>
<td>0.0</td>
</tr>
<tr>
<td>36</td>
<td>Die lip position 1 (north)</td>
<td>125.501</td>
<td>0.0101326</td>
<td>1.47647</td>
</tr>
<tr>
<td>41</td>
<td>Die lip position 2</td>
<td>123.818</td>
<td>0.0108311</td>
<td>0.0</td>
</tr>
<tr>
<td>43</td>
<td>Die lip position 3</td>
<td>120.869</td>
<td>0.0118279</td>
<td>-0.848289</td>
</tr>
<tr>
<td>44</td>
<td>Die lip position 4</td>
<td>127.086</td>
<td>0.00949632</td>
<td>1.85308</td>
</tr>
<tr>
<td>45</td>
<td>Die lip position 5</td>
<td>125.807</td>
<td>0.0108208</td>
<td>0.0</td>
</tr>
<tr>
<td>5</td>
<td>Die lip position 6 (south)</td>
<td>126.707</td>
<td>0.0179739</td>
<td>0.0</td>
</tr>
</tbody>
</table>

THERMOCOUPLES

<table>
<thead>
<tr>
<th>Position</th>
<th>Temperature</th>
<th>a</th>
<th>b</th>
<th>c \cdot 10^7</th>
</tr>
</thead>
<tbody>
<tr>
<td>34 Die entrance</td>
<td>125.014</td>
<td>0.0107122</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>36 Die lip position 1 (north)</td>
<td>125.501</td>
<td>0.0101326</td>
<td>1.47647</td>
<td></td>
</tr>
<tr>
<td>41 Die lip position 2</td>
<td>123.818</td>
<td>0.0108311</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>43 Die lip position 3</td>
<td>120.869</td>
<td>0.0118279</td>
<td>-0.848289</td>
<td></td>
</tr>
<tr>
<td>44 Die lip position 4</td>
<td>127.086</td>
<td>0.00949632</td>
<td>1.85308</td>
<td></td>
</tr>
<tr>
<td>45 Die lip position 5</td>
<td>125.807</td>
<td>0.0108208</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>5 Die lip position 6 (south)</td>
<td>126.707</td>
<td>0.0179739</td>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>

PRESSURE TRANSDUCERS

<table>
<thead>
<tr>
<th>Position</th>
<th>Temperature</th>
<th>a</th>
<th>b</th>
<th>c \cdot 10^7</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 Head</td>
<td>-159.357</td>
<td>0.323803</td>
<td>34.4063</td>
<td></td>
</tr>
<tr>
<td>49 Adapter</td>
<td>-22.9341</td>
<td>0.186817</td>
<td>3.08678</td>
<td></td>
</tr>
<tr>
<td>53 Viscometer inlet</td>
<td>-136.201</td>
<td>0.254273</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>48 Viscometer outlet</td>
<td>14.2382</td>
<td>0.0977191</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>51 Die</td>
<td>16.8120</td>
<td>0.0611813</td>
<td>1.70698</td>
<td></td>
</tr>
</tbody>
</table>
THICKNESS

CENTER SCALE: 4.75
SPAN: 9.00

\[ M = 6.48363 - 0.149937 \cdot V + 0.007567614 \cdot V^2 \]

FIGURE 15  BETA GAUGE CALIBRATION
LIST OF REFERENCES
LIST OF REFERENCES


Blackstone, E. H. 1972 (Department of Surgery, The Medical Center, University of Alabama in Birmingham.) Private communication.


267