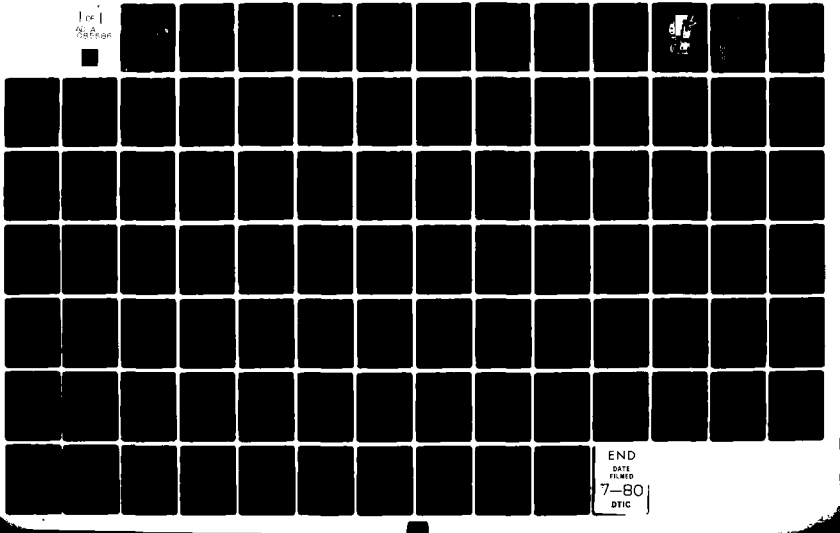
AD-A085 586 AIR FORCE WRIGHT AERONAUTICAL LABS WRIGHT-PATTERSON AFB OH F/G 14/2
OPERATION OF GENERATOR TEST FACILITY.(U)
APR 80 P G GABERDIEL
UNCLASSIFIED AFWAL-TR-80-2033 NL



END
DATE FILMED
7-80
DTIC

AFWAL-TR-80-2033

LEVEL II

OPERATION OF GENERATOR TEST FACILITY

Power Systems Branch
Aerospace Power Division

April 1980

TECHNICAL REPORT AFWAL-TR-80-2033

Final Report for Period August 1979 to February 1980

Approved for public release; distribution unlimited.

AERO PROPULSION LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Philip G. Gaberdiel
PHILIP G. GABERDIEL, 1Lt, USAF
Project Engineer

Paul R. Bertheaud
PAUL R. BERTHEAUD
Technical Area Manager

FOR THE COMMANDER

JAMES D. REAMS
Chief, Aerospace Power Division

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/POOS-2, *W-PAFB, OH 45433 to help us maintain a current mailing list".*

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

AIR FORCE/56780/2 June 1980 — 50

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFWAL-TR-80-2033	2. GOVT ACCESSION NO.: AD-A085586
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): OPERATION OF GENERATOR TEST FACILITY.	5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report. Aug [redacted] 79-Feb [redacted] 80
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Lt Philip G. Gaberdiel	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Aero Propulsion Laboratory (POOS-2) AF Wright Aeronautical Laboratories (AFSC) Wright-Patterson Air Force Base, Ohio 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Program Element 62203F, Project 3145, Task 314529, Work Unit 31452960
11. CONTROLLING OFFICE NAME AND ADDRESS: Aero Propulsion Laboratory (POO) AF Wright Aeronautical Laboratories (AFSC) Wright-Patterson Air Force Base, Ohio 45433	12. REPORT DATE: Apr [redacted] 80
	13. NUMBER OF PAGES: 90
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*: 89	15. SECURITY CLASS. *(of this report)*: Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Generator Test Facility

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This manual describes the operation of the Generator Test Facility of the Aero Propulsion Laboratory. This test facility is a computer-controlled facility for conducting performance tests of aircraft electrical generating systems.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

372662

FOREWORD

This report is an instruction manual for operating the Generator Test Facility. This test facility was built by engineers and technicians of the Power Systems Branch, Aerospace Power Division of the Aero Propulsion Laboratory, Wright-Patterson AFB, Ohio, under Project 3145, Task 314529, Work Unit 31452950. This manual was written by Lt. Philip G. Gaberdiel, AFWAL/POOS-2, during the period August 1979 through February 1980. The author submitted the report in February 1980.

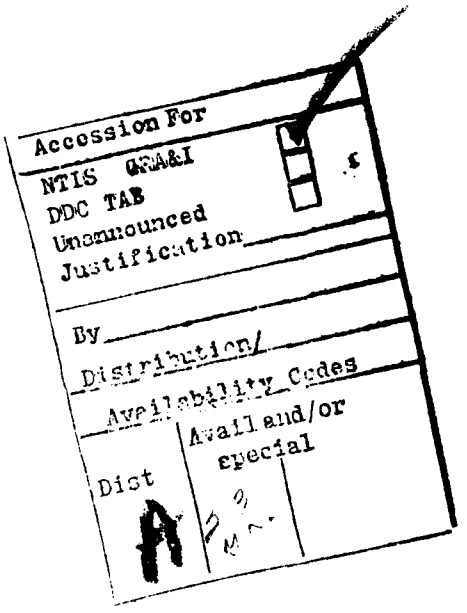


TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION I

BACKGROUND

Present and future aircraft systems require that the aircraft electrical generation system provide high quality electrical power under a wide range of operating conditions. In order to determine the performance of prototype generating systems, the test engineer must conduct a series of tests in which the operating conditions of the generator are varied to simulate aircraft operation. Tests are performed with various electrical load levels presented to the generator, including overload and fault conditions. All tests require that the generator be operated at a known speed within its operating range. Some tests require the speed of the generator to be changed at specific rates of acceleration or deceleration. Other test conditions may be varied for particular generating systems. A generator test is defined by a specific set of these generator system operating conditions.

The test engineer must examine the output of the generator during a test in order to determine its performance in response to the conditions of the test. Measurements of the raw output of the generator must usually be processed by various analysis techniques in order to portray a clear measure of the performance of the test generator.

Conventional facilities for conducting generator tests suffer from several shortcomings. Control of test conditions such as generator speed and acceleration rate and load settings is cumbersome. Techniques for measuring generator output are often relatively inaccurate and are usually too slow to accurately measure high speed transients. Synchronization of test actions, especially data acquisition, is a major problem common to all manually operated facilities. In addition, most systems require that analysis of the test data be performed at another site which greatly increases analysis turnaround time.

The Generator Test Facility of the Aero Propulsion Laboratory (Figure 1) overcomes these shortcomings in the testing of aircraft electrical generating systems.

Computer control of the test facility provides a highly flexible mechanism for performing generator tests. The inherent systematic operation of a computer provides accurate synchronization of test actions. Computer control also requires a minimum of personnel to perform generator testing.

A high speed data acquisition system (Figure 2) provides an accurate digital representation of the output of the generator under test conditions. Software resident in the system computer performs the required analysis computations on the test data. These analysis results, which exhibit the transient response of the test generator, are then available for display to the test engineer. This analysis and display scheme greatly reduces the turnaround time required for data reduction.

Therefore, the Generator Test Facility enables the test engineer to conduct testing of aircraft electrical generating systems easily and accurately and to analyze the performance exhibited by the generator in response to the test conditions.

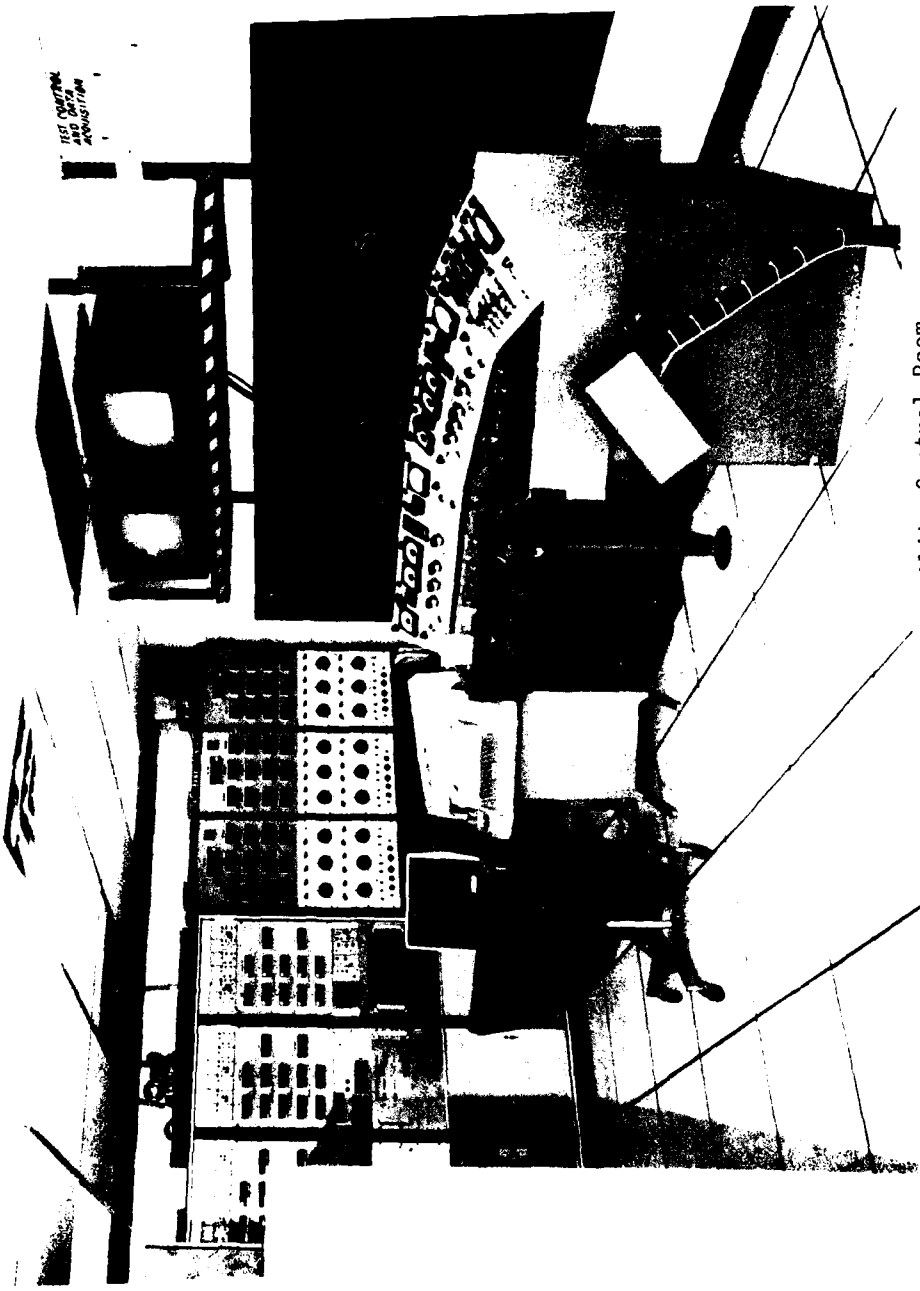


Figure 1. Generator Test Facility Control Room

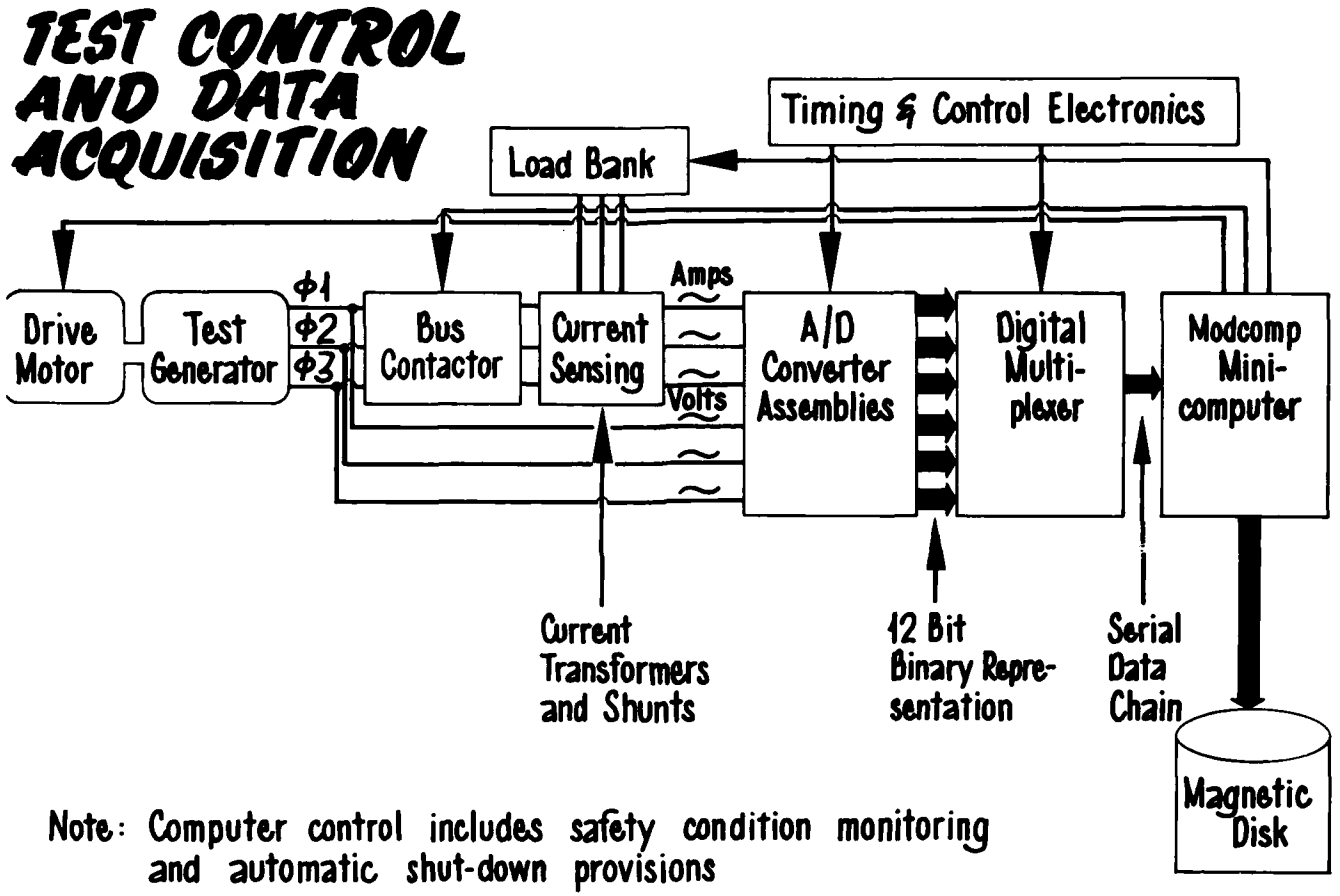


Figure 2. Test Control and Data Acquisition

SECTION II

INTRODUCTION

This manual describes the operation of the computer-controlled test capability of the Generator Test Facility. First discussed is the mechanism for creating the generator test sequences which will be used to direct operation of the generator system during a test. Next the manual explains the mechanisms, both software and hardware, which perform the test actions of a generator test sequence. And finally, the manual discusses operator actions required during computer-controlled testing.

The appendices include copies of all test control software required by the computer. Also included in the appendices are discussions about two special subsystems of the Generator Test Facility. These are the safety condition monitoring system and the drive stand controller.

This manual serves three major functions. First, it is a user's manual for the test engineer using the Generator Test Facility. Second, it can be used as a troubleshooting manual for the engineer or technician performing maintenance. Third, it serves as a training manual for personnel unfamiliar with the test facility. As such, the manual is written in general terms but with sufficient detail to serve the maintenance manual function.

SECTION III

CREATING A GENERATOR TEST SEQUENCE

A generator test sequence consists of a sequence of commands which control the test facility and the generator system during execution of a test. The commands available to the user are:

TM,	which controls relative timing of test actions;
DS,	which controls drive stand speed and acceleration rate;
LB,	which controls load bank settings;
AC,	which controls data acquisition; and
RC,	which controls various contact closure type functions.

By composing the appropriate sequence of these test commands, the test engineer can specify a particular generator test to be run.

The Fortran routine, CR8SYS, enables the user to compose and catalog generator test sequences. A listing of the code for CR8SYS is given in Appendix A. This section of the manual will give a description of each of the commands available for use in a test sequence. Also given is an example of the creation of a generator test sequence.

The creation of a test sequence is performed at the Tektronix CRT terminal. The routine CR8SYS, prompts the user to give the responses required to create the sequence. Each time the routine expects a new test sequence instruction, it will display the message, "ENTER COMMAND," on the CRT screen. The user will respond with a command and the parameters associated with that command. The user signals the end of the test sequence by entering the command "ST." The routine will then display the newly created test sequence on the ISC color display terminal. The user must then catalog the sequence so that it will be available for execution. Execution of a test sequence is explained in the next section of this manual. The following are the commands available for use in generator test sequences.

TM-Time

This command controls relative timing of events during the test sequence. It actually represents a delay until the time specified in its associated parameter. To specify a time delay, enter (via the Tektronix keyboard)

TM (time in seconds)*

The time value is a positive floating point number and must be greater than any previous time value used since the beginning of the test sequence.

As with all other commands requiring parameters, the TM command can be entered in two steps. If the user enters only "TM," the routine will prompt the user with the message "TIME (SEC)." The user then must enter the time in seconds of the delay required. This two-step entry is useful if the user is not familiar with the format of the parameters for the particular command chosen.

DS-Drive Stand Speed and Acceleration Rate

This command is used to select drive stand speed and acceleration/deceleration rate. The speed must be a positive integer between 0 and 10,000 rpm. The acceleration rate is a positive integer representing the rate in 100's of rpm per second at which the drive stand will accelerate from the previously requested speed (zero if no previous DS command exists in this sequence) to the speed requested by the current DS command. Therefore if the previous speed is greater than the requested speed, the rate actually represents deceleration.

The format of the command is:

DS (drive stand#) (speed) (rate)**

All parameters must be positive integers and are entered in free format with blanks or commas separating each one.

Again, if the user enters only "DS," the routine will respond with the prompt message, "DS#, SPEED, RATE (3I)." The user must then enter the appropriate parameters.

* All command entries must be completed by pressing the RETURN key.

** 3 is the minimum rate accepted by the routine, and 0 or 10 directs the drive stands to accelerate at the maximum rate of 1000 rpm/sec.

LB-Load Bank Settings

This command is used to select the setting of the load banks. Load banks numbered 1, 2, and 3 can be controlled. A load setting, both resistive (KW) and reactive (KVAR), is selected for each of three phases. The total load requested must not exceed the maximum available from the individual load bank. This would be 80 KW per phase and 60 KVAR per phase on load banks 1 or 2, and 40 KW per phase and 30 KVAR per phase on load bank 3.

The format of the command is:

LB (load bank#) (ϕ1KW) (ϕ2KW) (ϕ3KW) (ϕ1KVAR) (ϕ2KVAR) (ϕ3KVAR)

All parameters must be positive integers entered in free format.

If the user enters only "LB," the routine issues the prompt message, "LB#, 3*KW, 3*KVAR (7I)." The user must then enter the appropriate parameters.

AC-Acquire Data

This command causes data to be acquired from the data acquisition system for the amount of time specified in the parameter and stored on a dedicated magnetic disk file (file 30). Time spans of up to 25 seconds of data can be stored, but the analysis time required for this amount of data makes it practical to store time spans of 1 to 5 seconds of data.

The format of the command is:

AC (duration in seconds)

The duration value is a positive floating point number. If the user enters only "AC," the routine prompts with the message, "DURATION OF ACQUISITION (SEC)." Note that this command just initiates data acquisition and then proceeds to the next command. It does not wait for data acquisition to be completed before proceeding.

RC-Relay Contact Closure

This command is used to control opening or closing of contacts which are part of the input/output interface subsystem of the Modcomp minicomputer. These contacts can be used to control external relays used for such generator test functions as closing a bus contactor, applying generator field excitation, and the like. The contacts can accommodate loads of up to 10 VA with a maximum voltage of 100 V.

The user can control up to 13 contacts with this command. The desired state of the contact is specified as "0" or "1." A "0" opens the contact, and a "1" closes it.

The format of the command is:

RC (contact number) (state)

The parameters are positive integers entered in free format. If the user enters only "RC," the routine will prompt with the message, "CONTACT#, 1/0 (2I)." The user must then enter the required parameters.

There are also several commands to the routine CR8SYS which are used to maintain and catalog test sequences. These commands are explained below.

ST-Stop Sequence

This command signals the end of the test sequence being created. When the user responds with "ST," the routine displays the newly created test sequence on the ISC color display terminal. If correct, the sequence must be catalogued so that it will be available to be executed. If incorrect, the sequence must be reentered.

CA-Catalog Sequence

This command causes a test sequence to be stored on a library file of the magnetic disk (file 31). The test sequences are stored in a format such that the RD command can access them. There is storage available for sequences between 0 and 599 inclusive.

The format for this command is "CA." The routine will display the test sequence being stored on the ISC terminal and ask for "TEST SEQUENCE NUMBER, I5." The user enters the number at which he wishes the sequence to be catalogued. Any sequence previously catalogued at the same number will be overwritten. The sequence number is used for recalling the test sequence with the RD command.

RD-Read Test Sequence

This command allows the user to recall a test sequence from the magnetic disk (file 31). The sequence number requested must have been previously catalogued using the CA command.

The format of this command is:

RD (test sequence number)

If the user enters only "RD," the routine will prompt with the message, "ENTER TEST SEQUENCE NUMBER."

After the sequence number is entered, the routine will display the selected sequence on the ISC terminal. If the test sequence displayed has no instructions, this means that no sequence has been catalogued at the requested number.

SA-Read Same Sequence

This command causes the most recent sequence read or created to be displayed. The user chooses this command by typing "SA."

Following is a demonstration of the creation and cataloguing of a generator test sequence. All prompt messages produced by the routine and all responses required by the user are given.

This particular test sequence performs application of full load to a 150 KVA generator at base speed. The details of how these functions are accomplished during execution are given in the next section of this manual.

To create and store the test sequence, the following steps are required:

1. User types on the Tektronix keyboard

   JOB

   EXE CR8 LM

2. The display and responses illustrated in Figure 3 are via the Tektronix CRT terminal. Prompt messages issued by the routine are in all caps. Responses entered by the user are in brackets. A two-part entry of the AC command is included. Note: All responses by the user must be completed by striking the RETURN key.

3. At this point, the routine will produce on the ISC screen a display of the test sequence just created. Figure 4 illustrates that display.

4. The user must now catalog this sequence for later use. The following is the series of commands needed to do so.

```
ENTER COMMAND
[CA]
```

The routine will again display the sequence on the ISC screen and ask:

```
TEST SEQUENCE NUMBER, I5
[10]
ENTER COMMAND
$$ (This exits the CR8SYS routine.)
```

The newly created sequence is now catalogued as sequence number 10.

This section has demonstrated how to create and maintain generator test sequences using the routine CR8SYS. This section may be used as a user's manual and as an instructional guide for new personnel. Any modification to the sequence creation function of the routine CR8SYS will require a detailed examination of the Fortran code contained in Appendix A. The following section of this manual discusses the actual execution of a generator test sequence.

```
[JOB]
[EXE CR8 LM]
 ENTER COMMAND
[DS 3 4600 10]
 ENTER COMMAND
[TM 20.]
 ENTER COMMAND
[RC 5 1]
 ENTER COMMAND
[LB 2 50 50 50 0 0 0]
 ENTER COMMAND
[TM 24.8]
 ENTER COMMAND
[AC]
 DURATION OF ACQUISITION (SEC)
[1.5]
 ENTER COMMAND
[TM 25.]
 ENTER COMMAND
[RC 3 1]
 ENTER COMMAND
[TM 25.5]
 ENTER COMMAND
[RC 3 0]
 ENTER COMMAND
[TM 27.]
 ENTER COMMAND
[RC 4 1]
 ENTER COMMAND
[TM 27.5]
 ENTER COMMAND
[RC 4 0]
 ENTER COMMAND
[TM 30.]
 ENTER COMMAND
[RC 5 0]
 ENTER COMMAND
[LB 2 0 0 0 0 0 0]
 ENTER COMMAND
[DS 3 0 5]
 ENTER COMMAND
[ST]
 ENTER COMMAND
[CA]
 TEST SEQUENCE NUMBER, I5
[10]
 ENTER COMMAND
[$$]
```

Figure 3 - Test Sequence Creation Commands

TIME	DS	SPD	RATE	LB	KW			KVAR			DATA	RC
	3	4600	10									
20.0												5 1
				2	50	50	50	0	0	0		
24.8											1.50	
25.0												3 1
25.5												3 0
27.0												4 1
27.5												4 0
30.0												5 0
				2	0	0	0	0	0	0		
	3	0	5									

Figure 4 - Test Sequence Display

SECTION IV

EXECUTING A GENERATOR TEST SEQUENCE

The Fortran routine RUNSYS directs the test facility during execution of a generator test sequence. A copy of the code for RUNSYS is included in Appendix B. The routine first allows the user to access a test sequence which was previously created and catalogued using the routine CR8SYS. The commands required to initiate execution of the routine RUNSYS are as follows: (via the Tektronix terminal).

```
[JOB]
[EXE RUN LM]
ENTER COMMAND
[RD 10]
```

The routine will display the requested test sequence (in this example, sequence 10) on the ISC terminal and ask,

READY TO RUN THIS SEQUENCE? (Y OR N)

Section V will discuss actions required by the operator during execution of a test sequence.

The minicomputer directs execution of a generator test by causing appropriate actions to occur at specified times. These actions will be referred to as test control functions. Test control functions include: control of load bank settings, control of drive speed and acceleration rate, control of test data acquisition, and control of a set of relay contacts. A generator test is defined by a timed sequence for performing the test control functions required to accomplish the test. This timed sequence, referred to as a generator test sequence, is created using the test sequence creation function of the routine CR8SYS. The procedure for creating test sequences is given in Section III of this manual.

In order to cause the test defined in the test sequence to occur, a scheme must be devised which translates each command in the sequence

into an electrical output which performs the test function specified by the command. Generally this scheme is as follows: the Fortran routine RUNSYS steps through each command in the test sequence. For those functions which can be performed solely by software, RUNSYS calls the routine which implements the function. For those functions which are performed by software directing hardware interfaces, RUNSYS calls the routine which initiates action to perform the function. In some cases, several levels of control routines must be executed before the electrical signal is output to perform the function.

The test control functions used in generator testing are:

(1) Timing control

(2) Drive stand control

(3) Load bank control

(4) Data acquisition control

(5) Relay contact control

Following is a description of each function and the means by which this function is performed:

1. Timing control

In order to maintain proper sequencing of test actions during a generator test, the relative timing of the test control functions must be controlled. As explained in Section III, the TM command is used in the test sequence to specify this timing.

During test sequence execution, RUNSYS uses the subroutine TWAIT to provide timing control. A copy of TWAIT can be found in the code of RUNSYS in Appendix B.

At the beginning of execution of a test sequence, TWAIT is called to obtain a beginning time, TBEGIN, from the minicomputer internal timer. As TM commands are encountered during a test sequence,

TWAIT is called to provide a delay until the time specified in the command. By inserting TM commands just prior to other test control commands in a test sequence, the operator can specify the relative times at which events of the test occur.

2. Drive stand control

During generator testing, it is necessary to control the drive stand, which provides rotation to the generator, at a known speed. Also, some tests require the generator to be accelerated or decelerated to a new speed at a known rate. The drive stand control function provides this control.

The DS command is used in the test sequence to specify that one of the three drive stands be accelerated/decelerated to a specified speed at a specified rate. Actual control of the drive stand is accomplished by providing a DC voltage to the drive stand power supply. A voltage of 0-10V DC input to the power supply produces a speed of 0-10,000 rpm on the drive stand output pad. To control this speed, then, a DS command must be translated into the appropriate DC voltage. Control of acceleration rate is accomplished by outputting this DC voltage in small steps at an appropriate rate. For example, to accelerate from 1,000 rpm to 2,000 rpm at 2,000 rpm/second, a command to increase the speed by 20 rpm is issued every 10 msec for 0.5 seconds.

At this point, it is instructive to discuss the form in which certain control routines exist in the minicomputer. The routine RUNSYS is what is called a batch task. During execution of a test sequence, RUNSYS is executed by the minicomputer central processing unit (CPU) from its central memory. Certain test control functions, such as load bank control and relay contact control, can be performed by RUNSYS directly. However, other functions, namely, drive stand control and data acquisition control, are special in nature. These functions are iterative and must be performed with precise timing. For these reasons, the routines

which implement these functions exist as separate tasks. These tasks, named DSC for drive stand control and DAC for data acquisition control, share operating time in the minicomputer with the routine RUNSYS. It is useful to think of the DSC and DAC tasks as executing all the time. It should be clear that whenever a generator test is not being executed, the DSC task must output signals which keep the drive stands at zero speed and the DAC task must refrain from acquiring data.

The routine RUNSYS, when action is required from either of these tasks, collects the parameters from the appropriate test command, makes this data available to the task, and then signals the task to act on this data.

Explanation of how RUNSYS and the DSC task relate is now presented. A copy of the code for the DSC task is included in Appendix C. Discussion of the DAC task will be presented later in the appropriate section.

When RUNSYS encounters a DS command, it stores the drive stand number, requested speed, and requested rate in a buffer. It then sets a flag (DSC flag) which informs the DSC task that a valid command is available.

The DSC task has been executing a loop of instructions which output a null command to keep the drive stands at zero speed. On each iteration of this loop, the DSC flag is checked to see if a new command is available. When the DSC flag is found to be set, the DSC task first resets the flag so that it will know when a new command has been issued. Then it collects the command parameters from the buffer, scales and packs the command, and outputs the command to the drive stand controller. The DSC task continues to output this command until RUNSYS issues a new command.

As indicated above, the DSC task supplies its output to the drive stand controller. The DSC task, in a sense, interprets drive stand commands and passes these interpretations along to the controller.

The drive stand controller is a microprocessor based interface which implements drive stand commands. A digital-to-analog converter produces the DC voltage required by the drive stand power supply. The controller also performs the stepping increase/decrease described earlier to achieve acceleration rate control. A full description of the drive stand controller is given in Appendix E. By employing the controller to provide actual control of the drive stand, the DSC task must only initiate the action, thus leaving CPU time for RUNSYS and the DAC task.

The drive stand controller and the DSC task operate in a handshaking mode. If either device fails to respond correctly, the controller is designed to bring the drive stands to zero speed. Thus, if communication between the mini-computer and the drive stand controller is broken or if the minicomputer malfunctions, the drive stands are shut down.

3. Load bank control

The minicomputer controls the load bank settings *during* generator test execution by controlling a set of contacts in its input/output interface subsystem (I/OIS). There are 15 contacts for each phase of each load bank. Each contact, when closed, introduces a particular resistive or reactive load into the load circuit of the generator. Resistive loads up to 80 KW/phase and inductive loads up 60 KVAR/phase are available from loads banks #1 and #2. Loads up to 40 KW/phase and 30 KVAR/phase are available from load bank #3.

The computer control of the load bank settings parallels the manual control available via switch settings located on the load bank control panel. Therefore during execution of a generator test, these manual controls must be set to zero. Refer to the circuit drawings of the load bank control panel for physical location of these relays.

During the creation of a test sequence, the test operator specified load bank settings with the LB command. During execution of the

test sequence, the subroutine SETLB implements this control. SETLB examines the LB command parameters, sets up a control word which will close the appropriate relays to achieve the requested load settings, and then outputs this control word to the relay control system of the minicomputer. A listing of SETLB is included in the code of the routine RUNSYS in Appendix B.

4. Data acquisition control

The data acquired during a generator test consists of signals proportional to the 3-phase voltages and the 3-phase currents of the generator output. Data acquisition electronics convert these 6 analog signals to 12-bit digital representations. These binary words are multiplexed into data blocks which serve as input to the minicomputer through its 4805 data terminal. Each data block consists of 10 binary words: a null word, a timing word, 6 data words, and 2 data integrity words (parity and checksum).

In the creation of a test sequence, the user specifies the beginning time and duration of the data acquisition with the TM and AC commands. For instance, if the user were interested in a load application occurring 25 seconds into the test, he would specify at TM 24.8 seconds an AC command of duration 1.5 seconds.

During test execution when the routine RUNSYS encounters an AC command, it calls the subroutine ACQUIRE. Refer to the listing of RUNSYS in Appendix B for a copy of the code of ACQUIRE. The subroutine ACQUIRE first determines how many magnetic disk tracks will be required to store data for the duration requested. It then prepares buffer areas in central memory in which the data will be stored prior to being written to the disk. ACQUIRE also prepares data structures known as user file tables (UFT's) which are required in order to write to the disk. It then loads pointers to this information in a task communication area and resumes the data acquisition task (DAC) which has been in a hold state.

The DAC task performs the actual input and storage of generator test data. A copy of the code for the DAC task is included as Appendix D. In order to input data and write this data to the disk at a high rate, the DAC task employs a circular buffering technique. Nine temporary buffers in central memory are circularly filled with data incoming from the 4805. These are the buffers prepared by ACQUIRE. The transfer of data from the 4805 to these buffers is controlled by the direct memory processor (DMP) of the minicomputer. The central processor of the minicomputer is then directed by the DAC task to circularly output these buffers to the disk. A lockout scheme prevents overwriting a buffer before it has been written to disk or writing a buffer to disk before it is full. This lockout scheme reports to the user any attempted errors of this sort.

Nine buffers allow the data acquisition system to operate at approximately 9,200 hertz. This provides accurate sampling of up to 4,600 hertz input. A higher sampling rate or a change in the data word format will require additional buffering or a different buffering technique.

5. Relay contact control

The input/output interface subsystem (I/OIS) of the minicomputer contains relay contacts controllable by software. The relay contact control function enables the user to specify the state (on or off) of any of 13 of these contacts. Past test sequences have used these contacts to control such test actions as CSD disconnects, generator excitation voltage, etc.

The RC command is used in test sequence creation to specify the state of a particular relay. During test execution when RUNSYS encounters an RC command, it calls the subroutine SETRC. The code for SETRC can be found in the listing of RUNSYS in Appendix B. SETRC sets up a control word to properly set the requested contact and outputs it to the I/OIS.

This concludes the discussion of the test control functions available for generator testing. The next section of this manual discusses operator actions required during execution of a generator test.

test sequence, the subroutine SETLB implements this control. SETLB examines the LB command parameters, sets up a control word which will close the appropriate relays to achieve the requested load settings, and then outputs this control word to the relay control system of the minicomputer. A listing of SETLB is included in the code of the routine RUNSYS in Appendix B.

4. Data acquisition control

The data acquired during a generator test consists of signals proportional to the 3-phase voltages and the 3-phase currents of the generator output. Data acquisition electronics convert these 6 analog signals to 12-bit digital representations. These binary words are multiplexed into data blocks which serve as input to the minicomputer through its 4805 data terminal. Each data block consists of 10 binary words: a null word, a timing word, 6 data words, and 2 data integrity words (parity and checksum).

In the creation of a test sequence, the user specifies the beginning time and duration of the data acquisition with the TM and AC commands. For instance, if the user were interested in a load application occurring 25 seconds into the test, he would specify at TM 24.8 seconds an AC command of duration 1.5 seconds.

During test execution when the routine RUNSYS encounters an AC command, it calls the subroutine ACQUIRE. Refer to the listing of RUNSYS in Appendix B for a copy of the code of ACQUIRE. The subroutine ACQUIRE first determines how many magnetic disk tracks will be required to store data for the duration requested. It then prepares buffer areas in central memory in which the data will be stored prior to being written to the disk. ACQUIRE also prepares data structures known as user file tables (UFT's) which are required in order to write to the disk. It then loads pointers to this information in a task communication area and resumes the data acquisition task (DAC) which has been in a hold state.

SECTION V

TEST OPERATOR ACTIONS

Computer-controlled testing with the Generator Test Facility requires very little action by the test operator. Basically the test operator first ensures that the test facility and generator system are safe for operation; then the operator initiates the RUNSYS routine which will direct execution of the test. As detailed in Section IV, the steps required to initiate execution of RUNSYS are as follows: (via the Tektronix terminal)

[JOB]
[EXE RUN LM]
ENTER COMMAND
[RD n], where n is the test sequence number

The chosen sequence is displayed on the ISC terminal and the routine asks,

READY TO RUN THIS SEQUENCE? (Y OR N)

To initiate test execution, the user then strikes the "Y" and RETURN keys on the Tektronix keyboard. The routine RUNSYS then directs the test control actions as described in the previous section.

Actions required by the test operator during test sequence execution are of two basic categories. These are normal operation and abnormal operation of the test execution.

(1) Normal operation

Under normal operation, the operator performs the steps described above to initiate the test. The operator should continue to visually monitor the generator system during the test. If any unsafe condition arises, the operator should take actions to shut down the test.

In addition, the operator should note whether test actions (contact closures, load application and removal, etc.) occur at the expected time into the test. If these phenomena do not occur at approximately

the correct time, the test operator should take test shutdown actions since this is an indication that computer control of the test is not functioning properly. In order to monitor timing of the test, the test facility has a digital sequence timer. This timer should be started when execution of a test sequence is initiated.

Thus, under normal operation, the test operator should: (1) check the displayed test sequence for correctness, (2) visually inspect the generator and test area for safe operating conditions, (3) simultaneously start the digital timer and enter a "Y" (followed by RETURN) on the Tektronix keyboard, and then, (4) continue visual monitoring of the generator setup while checking that test phenomena occur at the proper time.

(2) Abnormal operation

The list of possible malfunctions which could occur to cause abnormal operation of a generator test is almost limitless. The following discussion will address several situations in which the generator test should be shut down. This shutdown action is required to prevent damage to the generator under test and possibly to the test facility or personnel.

Several types of malfunctions likely to occur during a generator test are automatically monitored by the minicomputer. This automatic monitoring is achieved in the following manner. First, a transducer develops a voltage signal proportional to the condition being monitored. In some cases, this voltage signal must be rectified, amplified, filtered, etc. to make it acceptable as input to the minicomputer. The conditional signal is then input to the mini-computer via its wide range analog input system. Then software routines check this data to determine if any of the signals are out of limit. The out-of-limit bounds are preselected for the particular condition being monitored and the particular test sequence. For example, a phase current of 600 amps would be an overcurrent condition under rated load testing of a generator, whereas the same current might be expected during fault testing.

If an out-of-limit condition is sensed, the minicomputer takes shutdown action automatically. A full discussion of the safety condition monitoring system is given in Appendix F. Also discussed in Appendix F is a safety monitoring routine available during manual operation of the test facility.

In some instances, the generator test should be shut down even if the safety monitoring system is not activated. These instances occur when the test operator observes any condition which he considers unsafe. For instance, suppose a test sequence specifies that the drive stand be accelerated to 4,600 rpm. During execution of the sequence, the operator observes the drive stand speed meter on the drive stand control console exceeding this speed. This indicates that computer control of the drive stands is not functioning properly. The operator should manually decelerate the drive. This is done by switching the control mode switch located on the drive stand control console to MANUAL and turning the DC motor lever to the READY position momentarily.

Another situation requiring manual shutdown is as follows. A test sequence specifies that a contact which is used to excite the test generator is to be closed 20 seconds after test initiation. During execution of the sequence, excitation does not occur. This situation in itself poses no danger, but it is evidence that the automatic sequencing of the test is not operating properly and more severe failures may occur. At the very least, without generator excitation, the data acquired during the test would be useless. Therefore, the operator should manually shut down the test.

Following is a list of the actions the test operator can take in shutting down a generator test.

1. In the event of a catastropic failure in which power should be removed from the drive stand as quickly as possible, test operator should press the red MASTER EMERGENCY TRIP button on the center drive stand control console

2. In the event the drive stand is required to coast to a halt and assuming the drive motor electronics are functioning properly,	test operator should switch control mode switch to MANUAL and turn the DC MOTOR lever to the READY position.
3. To remove all electrical load from the generator,	test operator should set the master load switch on the load bank control console to OFF.

This concludes the discussion of operator actions during a computer-controlled generator test. The following section of this manual discusses actions required at the conclusion of a generator test sequence.

SECTION VI

CONCLUSION OF A GENERATOR TEST SEQUENCE

Every generator test sequence must conclude by restoring the computer-controlled functions of the test facility to their original states. For instance, all the load contacts in the load bank should be opened, the drive stands should be at zero speed, and all external relay contacts should be opened. The user must ensure that commands to perform these actions are included in the generator test sequence.

Thus, normal execution of a test sequence will reset all test functions. If the test is abnormally halted, either by the safety monitoring system or manually, the RUNSYS routine must be allowed to complete execution of the test sequence in order that all resetting is accomplished.

When the test has completed, the RUNSYS routine again issues the message, "ENTER COMMAND." At this point, the test data which was acquired during the test is stored on a dedicated file of the magnetic disk. By entering the following commands, the user initiates the data analysis and display routines stored in the minicomputer.

```
[EXE]
ENTER PROGRAM NAME; (A3)
[ANL]
```

These routines compute generator performance parameters and display the results to the test engineer via the Tektronix terminal. Details of the analysis procedures and use of the display routines are the subject of a separate instruction manual.

APPENDIX A
CR8SYS

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  09:30:56        PAGE   1

  1         PROGRAM CRSYS
  2   C
  3   C  THIS ROUTINE ENABLES THE USER TO MAINTAIN GENERATOR
  4   C  TEST SEQUENCES.
  5   C  MAINTENANCE INCLUDES INITIAL CREATION, CATALOGUING, AND READING
  6   C  ALREADY CREATED SEQUENCES.
  7   C
  8   C  FILE/DEVICE ASSIGNMENTS REQUIRED FOR THIS ROUTINE:
  9   C
 10   C     8 = TKO (TEKTRONIX OUTPUT)  PROMPT MESSAGES TO USER
 11   C     7 = TKI (TEKTRONIX INPUT)   USER RESPONSES
 12   C     6 = A02 (ISC OUTPUT)        SEQUENCE DISPLAY
 13   C    31 = TSL (DISK PARTITION)    TEST SEQUENCE "LIBRARY"
 14   C
 15   C
 16   C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
 17   C  *  *  *  CATALOGUED ON LM 12,21,79   *  *  *  *  *  *  *  *
 18   C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
 19   C
 20   C  EXTERNAL SUBROUTINES REQUIRED:
 21   C
 22   C   NAME       LOCATION
 23   C
 24   C  2. READ        LB
 25   C  3. WRITE       LB
 26   C
 27   C
 28         INTEGER COMMAND(512),R(2),UFT(6),INUM,LINE(72)
 29         EQUIVALENCE (T,R)
 30   C  NOTE:  R IS AN INTEGER ARRAY WHICH IS EQUIVALENT TO THE REAL
 31   C  VARIABLE T.
 32   C
 33   C  INITIALIZE MAXIMUM NUMBER OF COMMANDS TO BE ALLOWED IN A TEST
 34   C  SEQUENCE
 35         DATA MAXCOM/512/
 36   C  SET UP UFT FOR WRITING/READING SEQUENCE TO DISK
 37   C  NOTE:  #BFE0 = @31 = TSL
 38         DATA UFT/0,ZBFE0,ZA400,3*0/
 39   C
 40   C  DESCRIPTION OF TEST SET-UP COMMANDS
 41   C
 42   C    TM - TIME DELAY
 43   C    DS - DRIVE STAND CONTROL
 44   C    LB - LOAD BANK CONTROL
 45   C    RC - RELAY CONTACT CONTROL
 46   C    AC - DATA ACQUISITION CONTROL
 47   C    ST - SPECIFIES END OF TEST SEQUENCE
 48   C
 49   C  MAINTENANCE COMMANDS:
 50   C
 51   C    CA - CATALOG SEQUENCE
 52   C    PD - READ SEQUENCE
 53   C    SA - REREAD SEQUENCE
 54   C
 55   C
 56   C  INITIALIZE POINTER INTO COMMAND BUFFER
```

```
100   IPNT=1
C  CHECK IF COMMAND BUFFER HAS BEEN EXCEEDED
110   IF(IPNT.LE.MAXCOM-8)GO TO 115
C  OUTPUT ERROR MESSAGE
      WRITE(8,1)
1     FORMAT(' COMMAND BUFFER EXCEEDED')
C  OUTPUT SEQUENCE AS IT STANDS
      GO TO 190
C  ISSUE PROMPT MESSAGE TO USER
115   WRITE(8,2)
2     FORMAT(' ENTER COMMAND')
C  INPUT COMMAND AND PARAMETERS
      READ(7,3,END=9999)ICOM,LINE
3     FORMAT(A2,72A1)
C  IS THIS A MAINTENANCE COMMAND?
      IF(ICOM.EQ.'SA')GO TO 190
C  REQUEST TO RE-CATALOG A SEQUENCE?
      IF(ICOM.EQ.'CA'.AND.IPNT.EQ.1)GO TO 190
C  IF NOT A MAINTENANCE COMMAND, ENTER INTO COMMAND BUFFER
      COMMAND(IPNT)=ICOM
C  DETERMINE WHICH COMMAND THIS IS AND GO TO ITS HANDLING ROUTINE
      IF(ICOM.EQ.'DS')GO TO 120
      IF(ICOM.EQ.'TM')GO TO 130
      IF(ICOM.EQ.'LB')GO TO 140
      IF(ICOM.EQ.'AC')GO TO 150
      IF(ICOM.EQ.'RC')GO TO 160
      IF(ICOM.EQ.'RD')GO TO 170
      IF(ICOM.EQ.'CA')GO TO 180
      IF(ICOM.EQ.'ST')GO TO 190
C  IF INVALID COMMAND, GO BACK AND GET NEW ONE
      WRITE(8,3000)
3000  FORMAT(' INVALID COMMAND')
      GO TO 110
C  ROUTINE TO INTERPRET DS COMMAND
C  IFORM COLLECTS INTEGER PARAMETERS FROM LINE AND STORES THEM IN
C  COMMAND BUFFER.  IERR INDICATES WHETHER PARAMETERS WERE VALID
C  INTEGERS.
120   CALL INTFORM(3,LINE,72,COMMAND(IPNT+1),IERR)
C  CHECK IF PARAMETERS WERE ENTERED
      IF(COMMAND(IPNT+1).EQ.0)GO TO 125
C  CHECK FOR INVALID PARAMETER
      IF(IERR.NE.0)GO TO 125
C  IF PARAMETERS WERE O.K., ADJUST POINTER
C  INTO COMMAND BUFFER
121   IPNT=IPNT+4
C  GET NEXT COMMAND
      GO TO 110
C  PROMPT USER TO ENTER PARAMETERS
125   WRITE(8,4)
4     FORMAT(' DS#,SPEED,RATE;(3I)')
C  INPUT FREE FORMAT PARAMETERS AND STORE IN COMMAND BUFFER
      CALL READI(7,COMMAND(IPNT+1))
C  ADJUST POINTER AND RETURN
      GO TO 121
C  ROUTINE TO INTERPRET TM COMMAND
C  COLLECT FLOATING POINT PARAMETER AND STORE IN T.  IERR RETURNS
```

```
113  C  ERROR INDICATION.
114  130   CALL FPTFORM(LINE,72,T,IERR)
115  C  WAS PARAMETER ENTERED?
116        IF(T.EQ.0.)GO TO 135
117  C  WAS PARAMETER VALID?
118        IF(IERR.NE.0)GO TO 135
119  C  STORE FLOATING POINT PARAMETER AS TWO INTEGERS IN COMMAND BUFFER
120  132   COMMAND(IPNT+1)=R(1)
121        COMMAND(IPNT+2)=R(2)
122  C  INCREMENT POINTER INTO COMMAND BUFFER
123        IPNT=IPNT+3
124  C  GET NEXT COMMAND
125        GO TO 110
126  C  PROMPT USER TO ENTER PARAMETER
127  135   WRITE(8,5)
128  5     FORMAT(' TIME(SEC)')
129  C  INPUT PARAMETER
130        CALL READF(1,T)
131  C  STORE IN COMMAND BUFFER
132        GO TO 132
133  C  ROUTINE TO INTERPRET LB COMMAND
134  C  COLLECT INTEGER PARAMETERS AND STORE IN COMMAND BUFFER
135  140   CALL INTFORM(7,LINE,72,COMMAND(IPNT+1),IERR)
136  C  CHECK FOR ERROR OR NO PARAMETERS
137        IF(IERR.EQ.0.AND.COMMAND(IPNT+1).NE.0)GO TO 145
138  C  PROMPT USER TO ENTER PARAMETERS
139        WRITE(8,6)
140  6     FORMAT(' LB#,3*KW,3*KVAR;(7I)')
141  C  COLLECT PARAMETERS AND STORE IN COMMAND BUFFER
142        CALL READI(7,COMMAND(IPNT+1))
143  C  INCREMENT POINTER
144  145   IPNT=IPNT+8
145  C  GET NEXT COMMAND
146        GO TO 110
147  C  ROUTINE TO INTERPRET AC COMMAND
148  C  COLLECT FLOATING POINT PARAMETER IN T
149  150   CALL FPTFORM(LINE,72,T,IERR)
150  C  CHECK FOR ERROR OR NO PARAMETER
151        IF(IERR.EQ.0.AND.T.NE.0.)GO TO 155
152  C  PROMPT USER TO ENTER PARAMETERS
153        WRITE(8,7)
154  7     FORMAT(' DURATION OF ACQUISITON(SEC)')
155  C  INPUT PARAMETER
156        CALL READF(1,T)
157  C  STORE PARAMETER AS INTEGERS IN COMMAND BUFFER
158  155   COMMAND(IPNT+1)=R(1)
159        COMMAND(IPNT+2)=R(2)
160  C  INCREMENT POINTER INTO COMMAND BUFFER
161        IPNT=IPNT+3
162  C  GET NEXT COMMAND
163        GO TO 110
164  C  ROUTINE TO INTERPRET RC COMMAND
165  C  COLLECT INTEGER PARAMETERS AND STORE IN COMMAND BUFFER
166  160   CALL INTFORM(2,LINE,72,COMMAND(IPNT+1),IERR)
167  C  CHECK FOR ERROR OR NO PARAMETERS
168        IF(IERR.EQ.0.AND.COMMAND(IPNT+1).NE.0)GO TO 165
```

```
C  PROMPT USER TO ENTER PARAMETERS
      WRITE(9,8)
8     FORMAT(' CONTACT#,1/0;(2I)')
C  COLLECT PARAMETERS AND STORE IN COMMAND BUFFER
      CALL READI(2,COMMAND(IPNT+1))
C  INCREMENT POINTER
165   IPNT=IPNT+3
C  GET NEXT COMMAND
      GO TO 110
C  ROUTINE TO INTERPRET RD COMMAND
C  INPUT SEQUENCE NUMBER
170   CALL INTFORM(1,LINE,72,INUM,IERR)
C  CHECK FOR ERROR OR NO RESPONSE
      IF(IERR.EQ.0.AND.INUM.NE.0)GO TO 175
C  PROMPT USER TO ENTER SEQUENCE NUMBER
172   WRITE(9,9)
9     FORMAT(' ENTER TEST SEQUENCE NUMBER')
C  COLLECT SEQUENCE NUMBER
      CALL READI(1,INUM)
C  CHECK FOR VALID SEQUENCE NUMBER
175   IF(INUM.LT.0.OR.INUM.GT.599)GO TO 172
C  SET RANDOM ADDRESS POINTER IN UFT
C  NOTE:  EACH SEQUENCE OCCUPIES 4 SECTORS
      UFT(4)=4*INUM
C  READ SEQUENCE AND STORE IN COMMAND BUFFER
      DO 177 I=1,4
177   CALL READ(COMMAND((I-1)*128+1),256,UFT)
C  OUTPUT SEQUENCE IDENTIFIER
      WRITE(6,10)INUM
10    FORMAT(//' TEST SEQUENCE NUMBER',I5)
C  OUTPUT SEQUENCE
      GO TO 190
C  ROUTINE TO HANDLE CA COMMAND
C  ENSURE THAT SEQUENCE ENDS WITH STOP COMMAND
180   COMMAND(IPNT)='ST'
C  OUTPUT SEQUENCE FOR USER TO VIEW
C  RESET POINTER
190   IPNT=1
C  OUTPUT HEADINGS
      WRITE(6,11)
11    FORMAT('   TIME',3X,'DS',2X,'SPD',2X,'RATE',3X,'LB',6X,'KW',
     19X,'KVAR',8X,'DATA',7X,'RC'//)
C  OUTPUT COMMANDS, ONE AT A TIME
200   NCMD=COMMAND(IPNT)
      IF(NCMD.EQ.'DS')GO TO 201
      IF(NCMD.EQ.'TM')GO TO 202
      IF(NCMD.EQ.'LB')GO TO 203
      IF(NCMD.EQ.'AC')GO TO 204
      IF(NCMD.EQ.'RC')GO TO 205
      IF(NCMD.EQ.'ST')GO TO 206
C  IF NONE OF THESE, ASSUME ST
      GO TO 206
C  OUTPUT DS COMMAND PARAMETERS
201   WRITE(6,12)(COMMAND(IPNT+I),I=1,3)
12    FORMAT(10X,I1,2I6)
C  UPDATE POINTER
```

```
225         IPNT=IPNT+4
226   C  GET NEXT COMMAND
227         GO TO 200
228   C  OUTPUT TM COMMAND PARAMETER
229   C  COLLECT REAL PARAMETER INTO INTEGER ARRAY EQUIVALENCED TO
230   C  REAL VARIABLE
231   202   R(1)=COMMAND(IPNT+1)
232         R(2)=COMMAND(IPNT+2)
233   C  OUTPUT PARAMETER
234         WRITE(6,13)T
235   13    FORMAT(1X,F6.1)
236   C  UPDATE POINTER
237         IPNT=IPNT+3
238   C  GET NEXT COMMAND
239         GO TO 200
240   C  OUTPUT LB COMMAND PARAMETERS
241   203   WRITE(6,14)(COMMAND(IPNT+I),I=1,7)
242   14    FORMAT(25X,I2,1X,6I4)
243   C  UPDATE POINTER
244         IPNT=IPNT+8
245   C  GET NEXT COMMAND
246         GO TO 200
247   C  OUTPUT AC COMMAND PARAMETER
248   C  COLLECT REAL PARAMETER INTO INTEGER ARRAY EQUIVALENCED TO
249   C  REAL VARIABLE
250   204   R(1)=COMMAND(IPNT+1)
251         R(2)=COMMAND(IPNT+2)
252   C  OUTPUT PARAMETER
253         WRITE(6,15)T
254   15    FORMAT(55X,F6.2)
255   C  UPDATE POINTER
256         IPNT=IPNT+3
257   C  GET NEXT COMMAND
258         GO TO 200
259   C  OUTPUT RC COMMAND PARAMETERS
260   205   WRITE(6,16)(COMMAND(IPNT+I),I=1,2)
261   16    FORMAT(65X,2I3)
262   C  UPDATE POINTER
263         IPNT=IPNT+3
264   C  GET NEXT COMMAND
265         GO TO 200
266   C  CHECK TO SEE IF THIS SEQUENCE IS TO BE CATALOGUED
267   206   IF(ICOM.NE.'CA')GO TO 100
268   C  PROMPT USER TO ENTER SEQUENCE NUMBER TO CATALOG UNDER
269   210   WRITE(8,17)
270   17    FORMAT(' ENTER TEST SEQUENCE NUMBER')
271   C  INPUT SEQUENCE NUMBER
272         CALL READI(1,I)
273   C  CHECK VALIDITY OF SEQUENCE NUMBER
274         IF(I.LT.0.OR.I.GT.599)GO TO 210
275   C  SET RANDOM ADDRESS POINTER IN UFT
276         UFT(4)=4*I
277   C  OUTPUT SEQUENCE TO DISK AS 4 SECTORS
278         DO 220 I=1,4
279   220   CALL WRITE(COMMAND((I-1)*128+1),256,UFT)
280   C  GET NEXT COMMAND
```

```
      GO TO 100
9999  STOP
      END
C
C
C  THE FOLLOWING ROUTINES ARE USED DURING TEST SEQUENCE CREATION
C  TO INPUT FREE-FORMAT REAL AND INTEGER PARAMETERS FOR
C  GENERATOR TEST COMMANDS.
C
C
      SUBROUTINE READI(N,IARRAY)
C
C  ROUTINE INPUTS ASCII STRING IN FREE FORMAT AND RETURNS N
C  EQUIVALENT INTEGER VALUES IN IARRAY
C
      INTEGER IARRAY(1),BUF(74),BUFSIZ
      DATA BUFSIZ/74/
C  INPUT STRING OF ASCII CHARACTERS
100   READ(7,1000)BUF
1000  FORMAT(74A1)
C  CALL ROUTINE TO FORMAT AND CONVERT TO INTEGER
      CALL INTFORM(N,BUF,BUFSIZE,IARRAY,IERR)
C  CHECK FOR ERROR IN PARAMETERS
      IF(IERR.EQ.0)RETURN
C  ECHO INVALID CHARACTER TO USER
      WRITE(8,2000)BUF(IERR)
2000  FORMAT(' INVALID CHARACTER -',A1,5X,'RE-ENTER LINE')
C  GET NEW STRING
      GO TO 100
      END
C
C
      SUBROUTINE INTFORM(NPAR,STRING,LTH,INT,IERR)
C
C  ROUTINE CONVERTS ASCII STRING (STRING) OF LENGTH, LTH, TO
C  NPAR EQUIVALENT INTEGERS STORED IN ARRAY INT.  IERR IS SET
C  IF ANY CHARACTER IN STRING IS NOT A VALID DIGIT.
C
      INTEGER INT(1),STRING(1)
C  INITIALIZE ERROR FLAG
      IERR=0
C  INITIALIZE POINTER INTO STRING
      IPNT=1
C  CONVERT EACH INTEGER
      DO 100 I=1,NPAR
C  INITIALIZE VALUE
      INT(I)=0
C  CHECK IF POINTER EXCEEDS STRING LENGTH
10    IF(IPNT.GT.LTH)GO TO 100
C  CHECK FOR DELIMITER
20    IF(STRING(IPNT).NE.' '.AND.STRING(IPNT).NE.',')GO TO 30
C  INCREMENT POINTER
      IPNT=IPNT+1
C  GET NEXT CHARACTER
      GO TO 10
C  CHECK FOR VALID DIGIT
```

```
337  30     IF(STRING(IPNT).LE.'9'.AND.STRING(IPNT).GE.'0')GO TO 40
338  C   SET ERROR FLAG AND RETURN
339         IERR=IPNT
340         RETURN
341  C   CONVERT TO INTEGER AND ADD TO ACCUMULATION OF VALUE
342  40     INTVAL=INTVAL*10 + (STRING(IPNT)-'0')/256
343  C   INCREMENT POINTER
344         IPNT=IPNT+1
345  C   CHECK FOR END OF STRING OR DELIMITER
346         IF(IPNT.LE.LTH.AND.STRING(IPNT).NE.' '.AND.
347        1STRING(IPNT).NE.',')GO TO 30
348  100    CONTINUE
349         RETURN
350         END
351  C
352  C
353         SUBROUTINE READF(N,ARRAY)
354  C
355  C   ROUTINE INPUTS ASCII STRING IN FREE-FORMAT AND RETURNS N
356  C   EQUIVALENT FLOATING POINT VALUES IN ARRAY.
357  C   NOTE:  IN ALL CURRENT CALLS TO READF, ONLY 1 VALUE IS
358  C   INPUT. (I.E., N=1)
359  C
360         INTEGER BUF(74),BUFSIZ
361         REAL ARRAY(1)
362         DATA BUFSIZ/74/
363  C   INPUT STRING OF ASCII CHARACTERS
364  100    READ(7,1000)BUF
365  1000   FORMAT(74A1)
366  C   CALL ROUTINE TO FORMAT AND CONVERT TO FLOATING POINT
367         CALL FPTFORM(BUF,BUFSIZ,ARRAY,IERR)
368  C   CHECK FOR ERRORS
369         IF(IERR.EQ.0)RETURN
370  C   ECHO INVALID CHARACTER TO USER
371         WRITE(8,2000)BUF(IERR)
372  2000   FORMAT(' INVALID CHARACTER -',A1,5X,'RE-ENTER LINE')
373  C   GET NEW STRING
374         GO TO 100
375         END
376  C
377  C
378         SUBROUTINE FPTFORM(STRING,LTH,FPT,IERR)
379  C
380  C   ROUTINE CONVERTS ASCII STRING (STRING) OF LENGTH, LTH, TO
381  C   EQUIVALENT FLOATING POINT VARIABLE, FPT.
382  C   IERR IS SET IF ANY CHARACTER IN STRING IS NOT A VALID DIGIT
383  C   OR DECIMAL POINT.
384  C
385         INTEGER STRING(1),PFLAG,PAMT
386  C   INITIALIZE ERROR FLAG
387         IERR=0
388  C   INITIALIZE POINTER INTO STRING
389         IPNT=1
390  C   INITIALIZE RETURN VALUE
391         FPT=0.
392  C   INITIALIZE DECIMAL POINT INDICATOR
```

```
393         PFLAG=0
394   C  PAMT KEEPS TRACK OF HOW MANY DIGITS OCCUR AFTER DECIMAL POINT
395         PAMT=0
396   C  CHECK IF ALL OF STRING HAS BEEN EXAMINED
397   100   IF(IPNT.GT.LTH)RETURN
398   C  CHECK FOR DELIMITER
399         IF(STRING(IPNT).NE.' '.AND.STRING(IPNT).NE.',')GO TO 110
400   C  INCREMENT POINTER
421         IPNT=IPNT+1
402   C  CHECK NEXT CHARACTER
403         GO TO 100
404   C  CHECK FOR DECIMAL POINT
405   110   IF(STRING(IPNT).NE.'.')GO TO 120
406   C  SET INDICATOR
407         PFLAG=1
408   C  GET NEXT CHARACTER
409         GO TO 140
410   C  CHECK FOR VALID DIGIT
411   120   IF(STRING(IPNT).LE.'9'.AND.STRING(IPNT).GE.'0')GO TO 130
412   C  SET ERROR FLAG AND RETURN
413         IERR=IPNT
414         RETURN
415   C  CHANGE ASCII DIGIT TO FLOATING POINT AND ADD TO ACCUMULATION
416   130   FPT=FPT*10. + FLOAT((STRING(IPNT)-'0')/256)
417   C  UPDATE NUMBER OF DIGITS AFTER DECIMAL POINT
418         PAMT=PAMT+PFLAG
419   C  INCREMENT POINTER
420   140   IPNT=IPNT+1
421   C  CHECK FOR END OF STRING OR DELIMITER
422         IF(IPNT.GT.LTH.OR.STRING(IPNT).EQ.' '.OR.STRING(IPNT).EQ.
423        1',')GO TO 150
424   C  CHECK NEXT CHARACTER
425         GO TO 110
426   C  ADJUST FPT TO ACCOUNT FOR DECIMAL POINT
427   150   FPT=FPT*10.**(-PAMT)
428   C  RETURN TO MAIN ROUTINE
429         RETURN
430         END
```

APPENDIX B
RUNSYS

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  09:30:20        PAGE   1

      1         PROGRAM RUNSYS
      2   C
      3   C   THIS ROUTINE PERFORMS EXECUTION OF A SEQUENCE SELECTED
      4   C   BY THE USER.  EXECUTION OF A TEST SEQUENCE IS
      5   C   ACCOMPLISHED BY SEQUENTIALLY INVOKING THE ROUTINE OR TASK
      6   C   REQUIRED TO PERFORM EACH COMMAND IN THE SEQUENCE.
      7   C
      8   C   FILE/DEVICE ASSIGNMENTS REQUIRED FOR THIS ROUTINE:
      9   C
     10   C      8 = TKO (TEKTRONIX OUTPUT)  PROMPT MESSAGES TO USER
     11   C      7 = TKI (TEKTRONIX INPUT)   USER RESPONSES
     12   C      6 = AO2 (ISC OUTPUT)        SEQUENCE DISPLAY
     13   C     31 = TSL (DISK PARTITION)    TEST SEQUENCE "LIBRARY"
     14   C
     15   C
     16   C   *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
     17   C   *  *  *  CATALOGUED ON LM 12,26,79   *  *  *  *  *  *  *  *
     18   C   *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
     19   C
     20   C   EXTERNAL SUBROUTINES REQUIRED:
     21   C
     22   C    NAME        LOCATION
     23   C
     24   C   1. TIMER        UL
     25   C   2. READ         LB
     26   C   3. WRITE        LB
     27   C   4. INITSAF      UL
     28   C   5. TSKDEL       UL
     29   C   6. SETBF        UL
     30   C   7. SETBT        UL
     31   C   8. GETDATA      UL
     32   C
     33   C
     34         INTEGER COMMAND(512),R(2),UFT(6),PNAME(2),TNUM,LINE(72)
     35         EQUIVALENCE (T,R)
     36   C   NOTE:  R IS AN INTEGER ARRAY WHICH IS EQUIVALENT TO THE REAL
     37   C   VARIABLE T.
     38   C
     39   C   INITIALIZE MAXIMUM NUMBER OF COMMANDS TO BE ALLOWED IN A TEST
     40   C   SEQUENCE
     41         DATA MAXCOM/512/
     42   C   SET UP UFT FOR WRITING/READING SEQUENCE TO DISK
     43   C   NOTE:  #BFE0 = @31 = TSL
     44         DATA UFT/0,ZBFE0,ZA400,3*0/
     45   C
     46   C   DESCRIPTION OF TEST COMMANDS
     47   C
     48   C     TM - TIME DELAY
     49   C     DS - DRIVE STAND CONTROL
     50   C     LB - LOAD BANK CONTROL
     51   C     RC - RELAY CONTACT CONTROL
     52   C     AC - DATA ACQUISITION CONTROL
     53   C     SE - SPECIFIES END OF TEST SEQUENCE
     54   C
     55   C   MAINTENANCE COMMANDS:
     56   C
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  09:30:20        PAGE   2


  57  C     RD - READ SEQUENCE
  58  C     SA - REREAD SEQUENCE
  59  C     EX - EXECUTE OVERLAY FROM LM FILE
  60  C
  61  C
  62  C    *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
  63  C    *   *   *   SELECTION OF TEST SEQUENCE   *   *   *   *   *   *
  64  C    *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
  65  C
  66  C  INITIALIZE POINTER INTO COMMAND BUFFER
  67  100    IPNT=1
  68  C  ISSUE PROMPT MESSAGE TO USER
  69  115    WRITE(8,2)
  70  2      FORMAT(' ENTER COMMAND')
  71  C  INPUT COMMAND AND PARAMETERS
  72         READ(7,3,END=9999)ICOM,LINE
  73  3      FORMAT(A2,72A1)
  74  C  IS THIS A MAINTENANCE COMMAND?
  75         IF(ICOM.EQ.'SA')GO TO 190
  76  C  IF NOT A MAINTENANCE COMMAND, ENTER INTO COMMAND BUFFER
  77         COMMAND(IPNT)=ICOM
  78  C  DETERMINE WHICH COMMAND THIS IS AND GO TO ITS HANDLING ROUTINE
  79         IF(ICOM.EQ.'RD')GO TO 170
  80         IF(ICOM.EQ.'EX')GO TO 300
  81  C  IF INVALID COMMAND, GO BACK AND GET NEW ONE
  82         WRITE(8,2000)
  83  2000   FORMAT(' INVALID COMMAND')
  84         GO TO 115
  85  C  ROUTINE TO INTERPRET RD COMMAND
  86  C  INPUT SEQUENCE NUMBER
  87  170    CALL INTFORM(1,LINE,72,TNUM,IERR)
  88  C  CHECK FOR ERROR OR NO RESPONSE
  89         IF(IERR.EQ.0.AND.TNUM.NE.0)GO TO 175
  90  C  PROMPT USER TO ENTER SEQUENCE NUMBER
  91  172    WRITE(8,9)
  92  9      FORMAT(' ENTER TEST SEQUENCE NUMBER')
  93  C  COLLECT SEQUENCE NUMBER
  94         CALL READI(1,TNUM)
  95  C  CHECK FOR VALID SEQUENCE NUMBER
  96  175    IF(TNUM.LT.0.OR.TNUM.GT.899)GO TO 172
  97  C  SET RANDOM ADDRESS POINTER IN UFT
  98  C  NOTE:  EACH SEQUENCE OCCUPIES 4 SECTORS
  99         UFT(4)=4*TNUM
 100  C  READ SEQUENCE AND STORE IN COMMAND BUFFER
 101         DO 177 I=1,4
 102  177    CALL READ(COMMAND((I-1)*128+1),256,UFT)
 103  C  OUTPUT SEQUENCE IDENTIFIER
 104         WRITE(6,10)TNUM
 105  10     FORMAT(//' TEST SEQUENCE NUMBER',I5)
 106  C  OUTPUT SEQUENCE FOR USER TO VIEW
 107  C  RESET POINTER
 108  190    IPNT=1
 109  C  OUTPUT HEADINGS
 110         WRITE(6,11)
 111  11     FORMAT('   TIME',3X,'DS',2X,'SPD',2X,'RATE',3X,'LB',6X,'KW',
 112        19X,'KVAR',8X,'DATA',2X,'RC'//)
```

```
C  OUTPUT COMMANDS, ONE AT A TIME
200    NCMD=COMMAND(IPNT)
       IF(NCMD.EQ.'DS')GO TO 201
       IF(NCMD.EQ.'TM')GO TO 202
       IF(NCMD.EQ.'LB')GO TO 203
       IF(NCMD.EQ.'AC')GO TO 204
       IF(NCMD.EQ.'RC')GO TO 205
       IF(NCMD.EQ.'ST')GO TO 900
C  IF NONE OF THESE, ASSUME ST
       GO TO 900
C  OUTPUT DS COMMAND PARAMETERS
201    WRITE(6,12)(COMMAND(IPNT+I),I=1,3)
12     FORMAT(10X,I1,2I6)
C  UPDATE POINTER
       IPNT=IPNT+4
C  GET NEXT COMMAND
       GO TO 200
C  OUTPUT TM COMMAND PARAMETER
C  COLLECT REAL PARAMETER INTO INTEGER ARRAY EQUIVALENCED TO
C  REAL VARIABLE
202    R(1)=COMMAND(IPNT+1)
       R(2)=COMMAND(IPNT+2)
C  OUTPUT PARAMETER
       WRITE(6,13)T
13     FORMAT(1X,F6.1)
C  UPDATE POINTER
       IPNT=IPNT+3
C  GET NEXT COMMAND
       GO TO 200
C  OUTPUT LB COMMAND PARAMETERS
203    WRITE(6,14)(COMMAND(IPNT+I),I=1,7)
14     FORMAT(25X,I2,1X,6I4)
C  UPDATE POINTER
       IPNT=IPNT+8
C  GET NEXT COMMAND
       GO TO 200
C  OUTPUT AC COMMAND PARAMETER
C  COLLECT REAL PARAMETER INTO INTEGER ARRAY EQUIVALENCED TO
C  REAL VARIABLE
204    R(1)=COMMAND(IPNT+1)
       R(2)=COMMAND(IPNT+2)
C  OUTPUT PARAMETER
       WRITE(6,15)T
15     FORMAT(55X,F6.2)
C  UPDATE POINTER
       IPNT=IPNT+3
C  GET NEXT COMMAND
       GO TO 200
C  OUTPUT RC COMMAND PARAMETERS
205    WRITE(6,16)(COMMAND(IPNT+I),I=1,2)
16     FORMAT(65X,2I3)
C  UPDATE POINTER
       IPNT=IPNT+3
C  GET NEXT COMMAND
       GO TO 200
C  ROUTINE TO LOAD OVERLAY FROM LM FILE.
```

```
169  C  NOTE:  LOADING OVERLAY TRANSFERS CONTROL FROM RUNSYS TO THE
170  C  ROUTINE OVERLAYED.
171  C  ONLY USE OF THIS FEATURE SO FAR HAS BEEN TO LOAD ANALYSIS
172  C  ROUTINE AT THE COMPLETION OF A GENERATOR TEST SEQUENCE.
173  C
174  C  PROMPT USER TO ENTER OVERLAY NAME
175  300   WRITE(8,18)
176  18    FORMAT(' ENTER PROGRAM NAME;(A3)')
177  C  INPUT NAME
178        READ(7,19)PNAME
179  19    FORMAT(2A2)
180  C  ROUTINE WHICH FOLLOWS PERFORMS OVERLAY OF REQUESTED ROUTINE
181        INLINE
182          LDM,2  PNAME
183          LDM,3  PNAME+1    GET ROUTINE NAME
184          REX,#37           CONVERT TO CAN CODE
185          DFC  $+1
186          STM,3  PNAM       STORE IN PNAM
187          REX,#43           GET INFO ON RUNSYS
188          DFC  0
189          TRR,2,3           REG 2=START ADDR OF RUNSYS
190          REX,#2B           LOAD CHAIN OVERLAY
191          DFC  2,@LM
192  PNAM    RES  1        STORAGE FOR PROGRAM NAME
193          DFC  0
194        FINI
195  C  NOTE:  AT THIS POINT DURING EXECUTION, CONTROL HAS PASSED TO
196  C  THE OVERLAY.
197  C
198  C
199  C  ASK USER IF THIS SEQUENCE IS TO BE EXECUTED
200  900   WRITE(8,20)
201  20    FORMAT('0READY TO RUN THIS SEQUENCE?(Y OR N)')
202  C  INPUT RESPONSE
203        READ(7,21)ICH
204  21    FORMAT(A2)
205  C  CHECK RESPONSE; IF NOT Y, GET NEXT COMMAND
206        IF(ICH.NE.'Y')GO TO 100
207  C
208  C   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
209  C    BEGINNING OF TEST SEQUENCE EXECUTION
210  C   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
211  C
212  C  INITIATE SAFETY MONITORING PERFORMED IN DSC TASK.
213  C  NOTE:  ITEMP IS STORAGE LOCATION USED BY ROUTINE INITSAF TO
214  C  SAVE SYSTEM SI POINTER
215        ITEMP=0
216        CALL INITSAF(ITEMP)
217  C  INITIALIZE POINTER INTO COMMAND BUFFER
218        IPNT=1
219  C  OBTAIN SYSTEM CLOCK READING AT START OF TEST
220        CALL TWAIT(0.)
221  C  CHECK IF POINTER EXCEEDS BUFFER
222  1000  IF(IPNT.GE.MAXCOM)GO TO 100
223  C  EXAMINE EACH COMMAND OF SEQUENCE AND JUMP TO INSTRUCTIONS
224  C  WHICH WILL INITIATE ACTION REQUIRED BY THAT COMMAND.
```

```
225         NCMD=COMMAND(IPNT)
226         IF(NCMD.EQ.'DS')GO TO 1100
227         IF(NCMD.EQ.'TM')GO TO 1200
228         IF(NCMD.EQ.'LB')GO TO 1300
229         IF(NCMD.EQ.'AC')GO TO 1400
230         IF(NCMD.EQ.'RC')GO TO 1500
231   C  IF NONE OF THESE COMMANDS, TEST SEQUENCE IS THROUGH
232   C  TERMINATE INPUT OF SAFETY MONITORING PARAMETERS AND RELOAD
233   C  ORIGINAL SI POINTER
234         INLINE
235           LDI,3  #4400      TERMINATE COMMAND
236           OCB,3,2           ISSUE COMMAND TO OUTPUT DEVICE
237           OCB,3,3           ISSUE COMMAND TO INPUT DEVICE
238           LDM,10  ITEMP     GET SI POINTER
239           STM,10  #00D3     STORE IN SYSTEM SI TRAP
240           LDI,2  @SBF       SBF FLAG
241           ZRR,3
242           REX,#4E           ZERO FLAG
243         FINI
244         GO TO 100
245   C  ROUTINE WHICH PASSES DS PARAMETERS TO SUBROUTINE WHICH WILL
246   C  INITIATE DRIVE STAND CONTROL
247   1100  CALL SETDS(COMMAND(IPNT+1))
248   C  INCREMENT POINTER INTO COMMAND BUFFER
249         IPNT=IPNT+4
250   C  GET NEXT COMMAND
251         GO TO 1000
252   C  CALL ROUTINE TO PROVIDE REQUESTED DELAY
253   1200  CALL TWAIT(COMMAND(IPNT+1))
254   C  INCREMENT POINTER
255         IPNT=IPNT+3
256   C  GET NEXT COMMAND
257         GO TO 1000
258   C  PASS LB PARAMETERS TO SUBROUTINE WHICH CONTROLS LOAD BANK
259   C  SETTINGS
260   1300  CALL SETLB(COMMAND(IPNT+1),COMMAND(IPNT+2),COMMAND(IPNT+5)
261        1,IERR)
262   C  INCREMENT POINTER
263         IPNT=IPNT+8
264   C  GET NEXT COMMAND
265         GO TO 1000
266   C  CALL ROUTINE TO INITIATE DATA ACQUISITION
267   1400  CALL ACQUIRE(COMMAND(IPNT+1))
268   C  INCREMENT POINTER
269         IPNT=IPNT+3
270   C  GET NEXT COMMAND
271         GO TO 1000
272   C  CALL ROUTINE WHICH CONTROLS RELAY CONTACTS
273   1500  CALL SETRC(COMMAND(IPNT+1),IERR)
274   C  INCREMENT POINTER
275         IPNT=IPNT+3
276   C  GET NEXT COMMAND
277         GO TO 1000
278   C
279   C   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
280   C    END OF TEST SEQUENCE EXECUTION
```

```
281  C   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
282  C
283  9999  STOP
284        END
285  C
286  C
287  C  FOLLOWING ROUTINES INITIATE ACTION REQUIRED BY INDIVIDUAL
288  C  COMMANDS IN TEST SEQUENCE.
289  C
290  C
291        SUBROUTINE SETDS(COMMAND)
292  C
293  C  THIS ROUTINE MAKES PARAMETERS OF DS COMMAND AVAILABLE TO
294  C  DRIVE STAND CONTROL TASK, DSC.
295  C
296        INTEGER COMMAND(1),COM(3)
297  C  GET PARAMETERS IN ARRAY ACCESSIBLE TO INLINE CODE
298        DO 10 I=1,3
299  10    COM(I)=COMMAND(I)
300  C  PLACE POINTER TO PARAMETERS IN TASK COMMUNICATION AREA
301        INLINE
302          LDI,2  @DSC      POINTER IS NAMED DSC
303          LDI,3  COM
304          REX,#4E          STORE POINTER
305        FINI
306        RETURN
307        END
308  C
309  C
310        SUBROUTINE TWAIT(T)
311  C
312  C  ROUTINE DELAYS EXECUTION OF RUNSYS (I.E., OPERATION OF
313  C  GENERATOR TEST) UNTIL SPECIFIED TIME, T.  THIS DELAY PROVIDES
314  C  TIMING CONTROL OF TEST SEQUENCE.
315  C
316        DOUBLE PRECISION TBEGIN,TNOW
317  C  IF REQUESTED TIME EQUALS ZERO, SAVE CURRENT SYSTEM CLOCK READING
318        IF(T.NE.0.)GO TO 1
319        CALL TIMER(TBEGIN)
320        RETURN
321  1     CONTINUE
322  C  GET SYSTEM CLOCK READING
323        CALL TIMER(TNOW)
324  C  IS DELAY POSITIVE?
325        IF((DBLE(T)+TBEGIN-.025D0).GT.TNOW)GO TO 2
326  C  NOTIFY USER OF TIMING ERROR
327        WRITE(6,100)T
328  100   FORMAT(' TIME ERROR',E15.5)
329        RETURN
330  C  CONVERT TIME IN SECONDS TO MINUTES AND TICS
331  2     MIN=IFIX(SNGL((TBEGIN+DBLE(T))/60.D0))
332        ITIC=IFIX(SNGL(TBEGIN+DBLE(T)-60.D0*DBLE(FLOAT(MIN)))*200.)
333  C  CALL SYSTEM ROUTINE TO DELAY UNTIL REQUESTED TIME
334        CALL .SKDEL(MIN,ITIC,.TRUE.,.FALSE.)
335        RETURN
336        END
```

```
337  C
338  C
339        SUBROUTINE SETLB(LB,KW,KVAR,IERR)
340  C
341  C  ROUTINE SETS LOAD BANK SWITCHES BY CONTROLLING A GROUP OF CONTACTS
342  C  IN I/OIS SYSTEM.
343  C
344  C  LB=1,2,OR 3 (DESIGNATES LOAD BANK NUMBER)
345  C  KW IS INTEGER ARRAY WITH RESISTIVE LOAD VALUES FOR PHASE 1,2, AND 3.
346  C  MAXIMUM OF 40 KW PER LOAD BANK.
347  C  KVAR IS ARRAY WITH REACTIVE LOAD VALUES.  MAXIMUM OF 30 KVAR PER
348  C  LOAD BANK.
349  C  IERR=0 IF NO ERRORS IN INPUT PARAMETERS; =1 IF ERRORS.
350  C
351        INTEGER KW(3),KVAR(3),FACTOR,S12(3),S21(3),S20(3),S13(3)
352        INTEGER OUTCOD(3),SCAN(3),TAB21(8),TAB12(6),TAB13(6),TAB20(6)
353  C
354  C  TAB12 THROUGH TAB20 ARE ARRAYS OF BIT CONFIGURATIONS WHICH WHEN
355  C  OUTPUT TO A GROUP OF CONTACTS OF THE I/OIS, SET UP THE SAME LOADS
356  C  AS THE SWITCH SETTINGS AVAILABLE ON THE LOAD BANK CONTROL CONSOLE.
357  C  THE NUMBERS 12,21,13,20 CORRESPOND TO THE SWITCH MARKINGS FOR
358  C  PHASE #1 OF LOAD BANK #1 USED IN SCHEMATIC DRAWING OF CONTROL
359  C  CONSOLE.
360  C
361  C  THIS ROUTINE, THEN, COMBINES THE REQUIRED BIT PATTERN TO ACHIEVE
362  C  THE LOAD SETTING REQUESTED AND OUTPUTS THIS CONFIGURATION TO THE
363  C  GROUP OF CONTACTS WHICH CONTROLS THE LOAD BANK BEING USED.
364  C
365  C
366        DATA TAB12/0,Z8000,Z4000,ZC000,Z6000,ZE000/
367        DATA TAB21/0,Z1000,Z0800,Z1800,Z0C00,Z1C00,Z0E00,Z1E00/
368        DATA TAB13/0,Z0100,Z0080,Z0180,Z00C0,Z01C0/
369        DATA TAB20/0,Z0020,Z0010,Z0030,Z0018,Z0038/
370  C  INITIALIZE ERROR FLAG
371        IERR=0
372  C  CHECK FOR VALID LOAD BANK #
373        IF(LB.LT.1.OR.LB.GT.3)GO TO 1000
374  C  FACTOR ACCOUNTS FOR PARALLELED LOAD BANKS
375        FACTOR=2
376        IF(LB.EQ.3)FACTOR=1
377  C  CHECK VALIDITY OF KW AND KVAR SETTINGS
378        DO 10 I=1,3
379        IF(KW(I).LT.0.OR.KW(I)/FACTOR.GT.40)GO TO 1000
380  10    IF(KVAR(I).LT.0.OR.KVAR(I)/FACTOR.GT.30)GO TO 1000
381  C  COMPUTE SWITCH SETTINGS
382        DO 20 I=1,3
383  C  FOR HIGHEST KW SETTING, SELECT SWITCH CONFIGURATION
384        IF(KW(I)/FACTOR.NE.40)GO TO 11
385        S12(I)=6
386        S21(I)=8
387        GO TO 12
388  C  SELECT LESSER KW SWITCH SETTING
389  11    S12(I)=MOD(KW(I)/FACTOR,5) + 1
390  C  SELECT GREATER KW SETTING
391        S21(I)=KW(I)/FACTOR/5 + 1
392  C  FOR HIGHEST KVAR SETTING, SELECT SWITCH SETTING
```

```
393  12     IF(KVAR(I)/FACTOR.NE.30)GO TO 13
394         S13(I)=6
395         S20(I)=6
396         GO TO 20
397  C   SELECT LESSER KVAR SETTING
398  13     S13(I)=MOD(KVAR(I)/FACTOR,5) + 1
399  C   SELECT GREATER KVAR SETTING
400         S20(I)=KVAR(I)/FACTOR/5 + 1
401  20     CONTINUE
402  C   SET UP WORDS TO BE OUTPUT TO I/OIS
403         DO 30 I=1,3
404  30     OUTCOD(I)=TAB12(S12(I))+TAB13(S13(I))+TAB20(S20(I))+
405        1TAB21(S21(I))
406  C   SET UP SCAN TABLE TO SERVE AS ADDRESS CONTROL FOR I/OIS
407         DO 40 I=1,3
408  40     SCAN(I)=4Z4010+(LB-1)*3+I-1
409  C   OUTPUT CONTROL WORDS TO CONTACTS THROUGH I/OIS
410         INLINE
411           LDI,2  UFT
412           FEX,1
413           DFC  OUTCOD,6      OUTPUT 3 WORDS
414           BRU OUT
415           DFC  SCAN,3
416  UFT      DFC 0,0D00,#8400,0,0,0
417  OUT      NOP
418         FINI
419         RETURN
420  C   PRINT ERROR MESSAGE
421  1000   IERR=1
422         WRITE(6,1)
423  1      FORMAT(' ***ERROR - SUBROUTINE SETLB',//,
424        1' ILLEGAL SETTING REQUESTED')
425         RETURN
426         END
427  C
428  C
429         SUBROUTINE SETRC(COM,IERR)
430  C
431  C   ROUTINE SETS CONTACTS IN I/OIS SYSTEM TO DESIRED STATE (1 OR 0)
432  C   COM(1) HAS NUMBER OF CONTACT, COM(2) HAS DESIRED STATE
433  C   IERR=0 IF NO ERRORS IN INPUT PARAMETERS
434  C
435         INTEGER COM(1),WORDOUT,SCAN
436         DATA WORDOUT/0/,SCAN/4Z4019/
437  C   INITIALIZE ERROR FLAG
438         IERR=0
439  C   CHECK VALIDITY OF PARAMETERS
440         IF(COM(1).LT.1.OR.COM(1).GT.13.OR.COM(2).LT.0.OR.COM(2).GT.1)
441        1GO TO 1000
442  C   CONTACT NUMBERING AND ACTUAL ADDRESS ARE NOT EQUIVALENT.
443  C   CONTACTS NUMBERED 1-8 ARE CONTROLLED BY BITS 6-13 OF CONTROL
444  C   WORD.  CONTACTS 9-12 ARE CONTROLLED BY BITS 1-4.
445  C   THEREFORE, A SEPARATE ROUTINE IS REQUIRED TO DEVELOP THE CONTROL
446  C   WORD FOR EACH GROUP OF CONTACTS.
447  C
448  C   SELECT APPROPRIATE ROUTINE BASED ON CONTACT NUMBER
```

```
      IF(COM(1).GT.8)GO TO 2
C  ROUTINE FOR CONTACTS 1-8
C  REQUEST TO OPEN CONTACT?
      IF(COM(2).EQ.0)GO TO 1
C  SET BIT IN CONTROL WORD TO CLOSE CONTACT
      CALL SETBF(WORDOUT,COM(1)+5)
      GO TO 10
C  SET BIT IN CONTROL WORD TO OPEN CONTACT
1     CALL SETBT(WORDOUT,COM(1)+5)
      GO TO 10
C  ROUTINE FOR CONTACTS 9-12
C  REQUEST TO OPEN CONTACT?
2     IF(COM(2).EQ.0)GO TO 5
C  SET BIT TO CLOSE CONTACT
      CALL SETBF(WORDOUT,COM(1)-8)
      GO TO 10
C  SET BIT TO OPEN CONTACT
5     CALL SETBT(WORDOUT,COM(1)-8)
10    CONTINUE
C  OUTPUT CONTROL WORD TO I/OIS DEVICE
      INLINE
        IDI,2  UFT
        REX,1
        DFC WORDOUT,2
        BRU OUT
        DFC  SCAN,1
UFT     DFC 0,0DC0,#8400,0,0,0
OUT     NOP
      FINI
      RETURN
C  SET ERROR FLAG
1000  IERR=1
C  PRINT MESSAGE AND RETURN
      WRITE(6,100)
100   FORMAT(' ***ERROR - SUBROUTINE SETBC')
      RETURN
      END
C
C
      SUBROUTINE ACQUIRE(TIM)
C
C  ROUTINE PREPARES DATA STRUCTURES REQUIRED BY DATA ACQUISITION
C  TASK, DAC, MAKES POINTERS TO THESE STRUCTURES AVAILABLE TO
C  DAC, AND THEN ACTIVATES THE DAC TASK.
C
      INTEGER UFT(10,9),BUF(2946,9),IARRAY(30),CNT
      DATA MAXBUF/9/,CNT/2944/
C  INITIALIZE ERROR COUNT FOR PARITY ERRORS DISCOVERED BY
C  ROUTINE GETDATA
1     IERR=0
C  CALL ROUTINE WHICH INPUTS A BLOCK OF DATA FROM THE DATA
C  ACQUISITION SYSTEM VIA THE 4805 DEVICE
      CALL GETDATA(IARRAY,30,10,NV,IERR)
C  CHECK FOR PARITY OR CHECKSUM ERRORS
      IF(IERR.EQ.0)GO TO 2
C  PRINT MESSAGE TO USER AND LOOP UNTIL ERROR-FREE BLOCK IS INPUT
```

```
505          WRITE(6,3)IERR
506    3     FORMAT(' ***4805 BAD',I5,' ERRORS - SUBROUTINE ACQUIRE',
507         1//,' PARITY OR CHECKSUM ERRORS')
508          GO TO 1
509    C   DETERMINE TIME OF ONE BLOCK OF DATA
510    2     TIME=(IARRAY(10)-IARRAY(2))*0.2E-6
511    C   DETERMINE NUMBER OF DISK TRACKS NEEDED TO STORE DATA FOR
512    C   REQUESTED TIME (TIM) AT PRESENT SAMPLING RATE
513    C   NOTE:  MAXTRK = (1 BLOCK/TIME SEC) * (10 WORDS/BLOCK) *
514    C                    (1 TRACK/23 SECTORS USED) * (1 SECTOR/128 WORDS)
515          MAXTRK=(10.*TIM)/(TIME*2944.)+1
516    C
517    C   CREATE UFT'S FOR WRITING TO DISK AND BUFFER AREAS FOR READING
518    C   FROM 4905 USING DATA CHAINING
519    C   NOTE:  DATA INPUT SCHEME USES CIRCULAR FILLING OF 9 BUFFERS
520    C   AND STORAGE OF EACH BUFFER ON A SEPARATE DISK FILE.
521          DO 10 I=1,MAXBUF
522    C   SET WORD COUNT FOR DATA CHAINING
523          BUF(CNT+1,I)=IAND(-CNT,4Z7FFF)
524    C   SET UP UFT'S
525          UFT(1,I)=0
526          UFT(2,I)=ICAN(30)
527          UFT(3,I)=42A400
528          DO 10 J=4,10
529    10    UFT(J,I)=0
530    C   SET POINTERS TO NEXT BUFFER FOR DATA CHAINING
531          M=MAXBUF-1
532          DO 20 I=1,M
533    20    BUF(CNT+2,I)=IADR(BUF(1,I+1))
534    C   SET POINTER IN LAST BUFFER TO POINT BACK TO FIRST BUFFER
535          BUF(CNT+2,MAXBUF)=IADR(BUF(1,1))
536    C
537    C   PUT MAXTRK ON DISK, SO THAT ANALYSIS ROUTINES WILL KNOW EXTENT
538    C   OF STORED DATA
539    C
540    C   SET RANDOM ADDRESS POINTER IN UFT
541          UFT(4,1)=23
542    C   READ FROM DISK TO POSITION PAST CALIBRATION DATA
543    C   STOPED BY ROUTINE CALAD
544          CALL READ(BUF(1,1),128,UFT(1,1),.TRUE.)
545    C   RESET RANDOM ADDRESS POINTER IN UFT
546          UFT(4,1)=23
547          BUF(1,1)=MAXTRK
548          CALL WRITE(BUF(1,1),128,UFT(1,1),.TRUE.)
549    C
550    C   ASSEMBLY LANGUAGE ROUTINE TO LOAD POINTER IN TASK COMMUNICATION
551    C   AREA WITH ADDRESS OF UFT'S AND DATA BUFFERS AND NUMBER OF TRACKS
552    C   TO BE ACQUIRED.
553    C   THESE PARAMETERS WILL BE PICKED UP BY THE DAC TASK WHICH WILL
554    C   BE TAKEN OUT OF A WAIT STATE AT THE END OF THIS ROUTINE.
555    C
556          INLINE
557            LDI,2  @BFR      POINTER NAME IS BFR
558            LDI,3  BUF
559            REX,#4E          STORE ADDRESS OF BUFFER IN POINTER
560            LDI,2  @TRK      FLAG NAME
```

```
561           LDM,3  MAXTPK
562           REX,#4E          STORE NUMBER OF TRACKS
563           LDI,2  @UFT      POINTER NAME
564           LDI,3  UFT
565           REX,#4E        STORE POINTER
566           REX,#15
567           DFC  0
568           DFC  @DAC      RESUME DAC TASK
569         FINI
570         RETURN
571         END
572   C
573   C
574   C  THE FOLLOWING ROUTINES ARE USED TO INPUT FREE-FORMAT REAL
575   C  AND INTEGER PARAMETERS FOR GENERATOR TEST SEQUENCE MAINTENANCE
576   C  COMMANDS.
577   C
578   C
579         SUBROUTINE READI(N,IARRAY)
580   C
581   C  ROUTINE INPUTS ASCII STRING IN FREE FORMAT AND RETURNS N
582   C  EQUIVALENT INTEGER VALUES IN IARRAY
583   C
584         INTEGER IARRAY(1),BUF(74),BUFSIZ
585         DATA BUFSIZ/74/
586   C  INPUT STRING OF ASCII CHARACTERS
587   100   READ(7,1000)BUF
588   1000  FORMAT(74A1)
589   C  CALL ROUTINE TO FORMAT AND CONVERT TO INTEGER
590         CALL INTFORM(N,BUF,BUFSIZE,IARRAY,IERR)
591   C  CHECK FOR ERROR IN PARAMETERS
592         IF(IERR.EQ.0)RETURN
593   C  ECHO INVALID CHARACTER TO USER
594         WRITE(8,2000)BUF(IERR)
595   2000  FORMAT(' INVALID CHARACTER -',A1,5X,'RE-ENTER LINE')
596   C  GET NEW STRING
597         GO TO 100
598         END
599   C
600   C
601         SUBROUTINE INTFORM(NPAR,STRING,LTH,INT,IERR)
602   C
603   C  ROUTINE CONVERTS ASCII STRING (STRING) OF LENGTH, LTH, TO
604   C  NPAR EQUIVALENT INTEGERS STORED IN ARRAY INT.  IERR IS SET
605   C  IF ANY CHARACTER IN STRING IS NOT A VALID DIGIT.
606   C
607         INTEGER INT(1),STRING(1)
608   C  INITIALIZE ERROR FLAG
609         IERR=0
610   C  INITIALIZE POINTER INTO STRING
611         IPNT=1
612   C  CONVERT EACH INTEGER
613         DO 100 I=1,NPAR
614   C  INITIALIZE VALUE
615         INT(I)=0
616   C  CHECK IF POINTER EXCEEDS STRING LENGTH
```

```
617  10    IF(IPNT.GT.LTH)GO TO 100
618  C  CHECK FOR DELIMITER
619  20    IF(STRING(IPNT).NE.' '.AND.STRING(IPNT).NE.',')GO TO 30
620  C  INCREMENT POINTER
621        IPNT=IPNT+1
622  C  GET NEXT CHARACTER
623        GO TO 10
624  C  CHECK FOR VALID DIGIT
625  30    IF(STRING(IPNT).LE.'9'.AND.STRING(IPNT).GE.'0')GO TO 40
626  C  SET ERROR FLAG AND RETURN
627        IERR=IPNT
628        RETURN
629  C  CONVERT TO INTEGER AND ADD TO ACCUMULATION OF VALUE
630  40    INT(I)=INT(I)*10 + (STRING(IPNT)-'0')/256
631  C  INCREMENT POINTER
632        IPNT=IPNT+1
633  C  CHECK FOR END OF STRING OR DELIMITER
634        IF(IPNT.LE.LTH.AND.STRING(IPNT).NE.' '.AND.
635       1STRING(IPNT).NE.',')GO TO 30
636  100   CONTINUE
637        RETURN
638        END
```

APPENDIX C
DSC

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  16:24:56        PAGE   1

 1          PROGRAM DSC
 2  C
 3  C   FOLLOWING ROUTINE PROCESSES DRIVE STAND COMMAND
 4  C   INSTRUCTIONS RECEIVED FROM ROUTINE RUNSYS.
 5  C   RESULTS FROM THIS ROUTINE ARE OUTPUT (VIA DEVICE OUT=A01)
 6  C   TO THE 8080 DRIVE STAND CONTROLLER WHICH PROVIDES
 7  C   ACTUAL CONTROL OF THE DRIVE STANDS.
 8  C   THIS ROUTINE HAS BEEN CATALOGUED IN THE FORM OF AN
 9  C   AUTO-START TASK SO THAT IT EXECUTES CONTINUOUSLY UNLESS
10  C   FORCED INTO A HOLD STATE.
11  C
12  C   *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
13  C   *  *  *  CATALOGUED ON LM  9,26,79  *  *  *  *  *  *  *  *  *  *  *
14  C   *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
15  C
16  C   EXTERNAL SUBROUTINES REQUIRED:
17  C
18  C    NAME          LOCATION
19  C
20  C   1. SAFCHECK       UL
21  C   2. WRITE          LB
22  C   3. READ           LB
23  C   4. WAIT           UL
24  C   5. TERMIN         LB
25  C   6. HOLD           LB
26  C
27  C
28          INTEGER LASTCOM(3),BUF(4),TEMP(2),IUFT(10),OUFT(10)
29          INTEGER ERRCNT
30          REAL DSCALE(3)
31          LOGICAL TESTB
32          DATA IUFT/0,Z3A70,ZE200,0,1,5*0/
33          DATA OUFT/0,Z611C,ZE200,0,8,5*0/
34  C   NOTE:  #3A70=@IN; #611C=@OUT
35  C
36  C   INITIALIZE COMMAND; THIS COMMAND WILL BE PROCESSED UNTIL
37  C   A VALID COMMAND IS RECEIVED FROM RUNSYS.
38  C   NOTE:  LASTCOM(1)=DS#,LASTCOM(2)=SPEED,LASTCOM(3)=RATE
39          LASTCOM(1)=1
40          LASTCOM(2)=0
41          LASTCOM(3)=0
42  C   FOLLOWING IS AN ASSEMBLY LANGUAGE ROUTINE WHICH SETS FLAGS
43  C   IN A TASK COMMUNICATION AREA SO THAT LATER PART OF THIS
44  C   ROUTINE DOES NOT TRY TO PROCESS A NEW COMMAND UNTIL ONE IS
45  C   AVAILABLE.
46  C   TASK COMMUNICATION AREA IS HANDLED BY CUSTOM REX CALLS #4F
47  C   AND #4D.  SEE LISTING OF O/S FOR FULL EXPLANATION OF THESE
48  C   REX CALLS
49  C
50          INLINE
51            LDI,2 @DSC      FLAG IS CALLED DSC
52            ZRP,3
53            REX,#4E      LOAD FLAG WITH ZERO
54  *   MUST ALSO ZERO SBF FLAG, WHICH UNTIL CLEARED BY RUNSYS
55  *   WILL CAUSE SAFETY CHECKING TO BE IGNORED.
56            LDI,2  @SBF
```

```
57          ZRR,3
58          REX,#4E
59        FINI
60  100   CONTINUE
61  C  PERFORM CHECK OF SAFETY PARAMETERS
62        CALL SAFCHECK
63  C  TEST DSC FLAG.  IF ZERO, KEEP PROCESSING CURRENT COMMAND.
64  C  IF NONZERO, DSC FLAG POINTS TO NEW COMMAND.  LOAD THIS
65  C  NEW COMMAND INTO ARRAY, LASTCOM.
66        INLINE
67          LDI,2 @DSC
68          REX,#4D     GET FLAG INTO REG. 3
69          TRRB,3,3 $+4     CHECK FOR ZERO
70          BRU A001     IF ZERO, TAKE THIS BRANCH
71  *  IF FLAG NOT ZERO, JUMP IS MADE TO FOLLOWING INSTRUCTIONS
72          LDX,2,3     REG 2 POINTS TO START OF NEW COMMAND
73          STM,2 LASTCOM     GET FIRST WORD
74          ABR,3,15
75          LDX,2,3
76          STM,2 LASTCOM+1     GET SECOND WORD
77          ABR,3,15
78          LDX,2,3
79          STM,2 LASTCOM+2     GET THIRD WORD
80  *  ZERO DSC FLAG, SO THAT UNTIL RUNSYS ISSUES A NEW COMMAND
81  *  (AND CHANGES FLAG) COMMAND IN LASTCOM WILL BE PROCESSED.
82          LDI,2 @DSC
83          ZRR,3
84          REX,#4E
85  A001    NOP
86        FINI
87  C  CHECK VALIDITY OF DRIVE STAND COMMAND
88        IF(LASTCOM(1).LT.1.OR.LASTCOM(1).GT.3.OR.
89       1   LASTCOM(2).LT.0.OR.LASTCOM(2).GT.10000.OR.
90       2   LASTCOM(3).LT.0.OR.LASTCOM(3).GT.10)GO TO 400
91  C  SPEED AND RATE COMMANDS MUST BE CONVERTED TO FORMAT
92  C  REQUIRED BY 8080 CONTROLLER.
93  C  I.E., 12 BITS OF 1 (4095) = MAX. OUTPUT
94        DO 302 I=1,2
95  302   TEMP(I)=IFIX(FLOAT(LASTCOM(I+1))*0.4095)
96  C  ARRAY BUF WILL CONTAIN COMMAND STRING SENT TO 8080 DS
97  C  CONTROLLER.  THE FOLLOWING INSTRUCTIONS PACK ARRAY BUF.
98  C
99  C  LOAD BUF(1) WITH 0D=CR WHICH SIGNALS BEGINNING OF
100 C  COMMAND AND DS# IN ASCII.
101       BUF(1)=IOR(4Z0D30,LASTCOM(1))
102 C  THE FOLLOWING INSTRUCTIONS USE FUNCTION IASCII TO CONVERT
103 C  THE SPEED AND RATE COMMANDS TO ASCII AND POSITION THEM
104 C  IN THE PROPER BYTE POSITION OF THE WORD.
105 C  I.E., SPEED = MSB(BUF(2)),LSB(BUF(2)),MSB(BUF(3))
106 C        RATE = LSB(BUF(3)),MSB(BUF(4)),LSB(BUF(4))
107       BUF(2)=IOR(IASCII(TEMP(1),3,1),IASCII(TEMP(1),2,0))
108       BUF(3)=IOR(IASCII(TEMP(1),1,1),IASCII(TEMP(2),3,0))
109       BUF(4)=IOR(IASCII(TEMP(2),2,1),IASCII(TEMP(2),1,0))
110 C  OUTPUT COMMAND STRING TO 8080 DS CONTROLLER.  WAIT UNTIL
111 C  OUTPUT OPERATION IS COMPLETE BEFORE CONTINUING.
112       CALL WRITE(BUF,8,CNFT,,TRUE.)
```

```
113  C  INPUT ONE BYTE FROM 8080 (VIA DEVICE IN=AI1).  THIS
114  C  IS THE DS# WHICH THE CONTROLLER ECHOES BACK TO INDICATE
115  C  THAT IT IS OPERATING CORRECTLY.  THIS INPUT OPERATION IS
116  C  DONE IN QUICK-RETURN MODE, SO THAT CONTROL IS RETURNED
117  C  TO THE ROUTINE AFTER THE OPERATION IS INITIATED.
118        CALL READ(BUF,1,IUFT,.FALSE.)
119  C  DELAY 20 TICKS TO ALLOW INPUT TO FINISH.
120        CALL WAIT(20,0,DUMMY)
121  C  CHECK TO SEE IF INPUT HAS FINISHED (BIT 15 OF IUFT = 0)
122  C  NOTE:  TESTB RETURNS TRUE FOR 0.
123        IF(TESTB(IUFT(1),15))GO TO 303
124  C  IF INPUT HAS NOT FINISHED, TERMINATE THE INPUT OPERATION.
125        CALL TERMIN(IUFT)
126  C  INCREMENT ERROR COUNT WHICH KEEPS TRACK OF HOW MANY TIMES
127  C  8080 HAS FAILED TO ECHO
128        ERRCNT=ERRCNT+1
129  C  ALLOW MAX. OF 2 ERRORS BEFORE HOLDING
130        IF(ERRCNT.GT.2)GO TO 500
131        GO TO 400
132  C  IF INPUT OPERATION WAS FINISHED, CHECK TO SEE IF DS#
133  C  ECHOED MATCHES DS# SENT
134  C  NOTE:  SINCE DS# WAS INPUT AS 1 BYTE, IT MUST BE SHIFTED
135  C  RIGHT BEFORE COMPARISON.  ALSO CONVERTED TO ASCII.
136  303   IF(ISHFT(BUF(1),-8)-2Z30.NE.LASTCOM(1))GO TO 500
137  C  IF DS#'S MATCH, RESET ERROR COUNT
138        ERRCNT=0
139  400   CONTINUE
140  C  DELAY NEEDED HERE FOR PROPER TIMING SYNCHRONIZATION WITH
141  C  8080 DS CONTROLLER
142        CALL WAIT(200,0,DUMMY)
143  C  LOOP BACK TO GET NEW COMMAND
144        GO TO 100
145  500   CONTINUE
146  C  IF 8080 DOESN'T ECHO CORRECTLY, PUT TASK IN HOLD STATE
147        CALL HOLD(.TRUE.)
148  C  AFTER TASK IS RESUMED, GO BACK TO LOOP
149        GO TO 400
150        END
151  C
152  C
153        FUNCTION IASCII(ICMD,NHEX,NBYTE)
154  C
155  C  ICMD IS 12 BIT COMMAND TO DRIVE STAND.  THIS ROUTINE
156  C  EXAMINES ONE HEX DIGIT (4 BITS) OF ICMD (POINTED TO BY NHEX)
157  C  ,CONVERTS THIS HEX DIGIT TO ASCII, AND POSITIONS IT IN
158  C  THE UPPER OR LOWER BYTE (DETERMINED BY NBYTE) OF A WORD
159  C  WHICH IS OUTPUT TO THE 8080 DRIVE STAND CONTROLLER.
160  C
161  C  SHIFT HEX DIGIT OF INTEREST TO RIGHT-MOST POSITION IN WORD
162  C  AND MASK OFF OTHER DIGITS.
163        KDIG=IAND(ISHFT(ICMD,(1-NHEX)*4),1ZF)
164  C  IF DIGIT > 9, ASCII CODE IS DIFFERENT
165        IF(KDIG.GT.9)GO TO 1
166  C  CONVERT TO ASCII AND POSITION IN WORD
167        IASCII=ISHFT(KDIG+2Z30,NBYTE*8)
168        RETURN
```

```
 169  1      IASCII=ISHFT(KDIG+2Z37,NBYTE*8)
 170         RETURN
 171         END
```

APPENDIX D
DAC

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  16:25:13        PAGE   1

 1          PROGRAM DAC
 2  C
 3  C  THIS ROUTINE IS CATALOGUED AS A TASK.  IT CAUSES DATA TO BE
 4  C  ACQUIRED FROM THE DATA ACQUISITION SYSTEM AND STORED ON DISK.
 5  C  THIS DATA ACQUISITION IS PERFORMED ACCORDING TO PARAMETERS
 6  C  SPECIFIED IN THE NEWSYS ROUTINE.
 7  C
 8  C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
 9  C  *  *  *  *  *  CATALOGUED ON LM  9,26,79  *  *  *  *  *  *  *  *
10  C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
11  C
12  C
13          LOGICAL TESTB
14          INTEGER BUF1(10),BUF2(14),TEMP2,TEMP1,TEMP3
15          INTEGER FLAG,TRKCNT,ERROR
16          DATA BUF1/' ERROR HAS  OCCURRED'/
17          DATA BUF2/' POSSIBLE ERROR HAS OCCURRED'/
18          DATA MAXBUF/9/
19          DATA TEMP2,TEMP3/0,0/
20          MAXINT=(MAXBUF*16)+428200
21  C  MAXINT REFLECTS MAXIMUM ALLOWABLE NUMBER OF INTERRUPTS
22  C
23      99 FLAG=0
24  C  FLAG STORES INDICATION OF INTERRUPT.
25  C
26  C  FOLLOWING IS ASSEMBLY LANGUAGE ROUTINE THAT FIRST
27  C  PUTS TASK IN WAIT.  TASK IS REMOVED FROM WAIT BY
28  C  SUBROUTINE ACQUIRE WHICH HAS JUST FILLED PARAMETER
29  C  PASSING AREA.  TASK THEN PICKS UP PARAMETERS AND
30  C  READS DATA FROM 4805 INTO BUFFER AREA.
31  C
32          INLINE
33  DEV     EQU          3       DEVICE #3 IS 4805
34  SECAMT  EQU          23              ;23 SECTORS/TRACK USED
35  WDCNT   EQU          SECAMT*128      ;WORDCOUNT PER TRACK
36            LDI,2  CIRLOC
37            STM,2  #80+DEV
38            STM,2  #C0+DEV          SET UP INT. ROUTINES
39            LDI,1  #4500
40            OCA,1,DEV               TERMINATE 4805
41  WAIT      REX,#11             PUT TASK IN WAIT MODE
42            LDI,2  @BFR
43            REX,#4D
44            TRRB,2,2  PEXIST        IF REG=/ 0; REQ. PAR. EXISTS
45            BRU  WAIT               IF REG2 = 0; PAR. DOES NOT EXIST
46  PEXIST TRRB,3,3  PAROK  IF REG3=/0; PAR HAS BEEN LOADED
47            BRU WAIT
48  PAROK     STM,3  BUF          GET ADDR. OF BUFFER
49            LDI,2  @BFR
50            LDI,3  0
51            REX,#4E             LOAD ZERO IN BFR
52            LDI,2  @TRK
53            REX,#4D
54            STM,3  MAXTRK       GET NUMBER OF TRACKS
55            LDI,2  @UFT
56            REX,#4D
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  16:25:13        PAGE   2


  57          STM,3  UFTT         GET ADDR. OF UFT
  58          LDM,3  BUF
  59        LDI,2          INT              ;USE INT ROUTINE "INT"
  60        STM,2          #80+DEV          ;SET UP DI INTERRUPT TRAP
  61        LDI,1          #8200            ;SET UP OBMM,0,0 INSTR
  62        STM,1          INT              ;PUT IN INT
  63        LDI,1          CIRLOC           ;GET ADDR OF DUMMY INT ROUTINE
  64        STM,1          #C0+DEV          ;SET UP SI LOCATION
  65        LDI,1          #4000
  66        OCA,1,DEV                       ;DISCONNECT SI & DI
  67        LDI,1          #00C1
  68        OCA,1,DEV                       ;RESET SI FLAGS AND SET UP COMMAND BFR.
  69        ISA,1,DEV                  ;GET STATUS
  70        TBRB,1,0       STOK             ;CHECK STATUS ERRORS
  71  STEP  NOP                             ;***FUTURE ERROR ROUTINE
  72  STOK  LDI,1     -WDCNTS#7FFF   ;WORD COUNT FOR TRACK WITH DATA CHAINING
  73        STM,1          #60+DEV          ;STORE IN DMP AREA
  74        LDM,2          BUF              ;GET ADDR OF FIRST BUFFER
  75        STM,2          #70+DEV          ;STORE IN DMP AREA
  76        LDI,1          #F300
  77        OCA,1,DEV      ;INITIATE INPUT
  78        BRU            DOMORE           ;GET OUT OF ASSEMBLY CODE
  79  *     INTERRUPT ROUTINE FOR 4805
  80  INT   OBMM,0         FLAG             ;SET BIT IN FLAG TO IND INT ENTERED
  81        ABMM,11        INT              ;SET UP FOR NEXT INTERRUPT TO SET NEXT BIT IN
  82        STM,1          RSAVE            ;SAVE REG1
  83        LDM,1          INT              ;GET OBMM INSTR IN REG1
  84        CRMB,1         MAXINT,$+4,INTOK ;CHECK FOR VALID OBMM INSTR
  85        LDI,1          #8200            ;SET UP OBMM,0,0 IN REG1
  86        STM,1          INT              ;PUT IN INT FOR NEXT TIME
  87  INTOK LDI,1          #7000            ;SET UP COMMAND WORD
  88        OCA,1,DEV                       ;RE-CONNECT INTERRUPTS
  89        LDM,1          RSAVE            ;RESTORE REG1
  90  CIRLOC CIR                            ;RETURN(ALSO DUMMY INT ROUTINE LOC.)
  91  RSAVE RES            1                ;SAVE AREA FOR REG1
  92  UFTT   RES    1
  93  BUF  RES     1
  94  DOMORE               NOP              ;END OF MACRO ASSEMPLER CODE
  95        FINI
  96        TRKCNT=9
  97  C
  98  C  CHECK FOR ERRORS IN READING DATA FROM 4805.
  99  C
 100        IPERR=0
 101  C  SET POSSIBLE ERROR COUNT TO 0
 102        ERROR=0
 103  C  SET ERROR COUNT TO 0
 104        I=0
 105  11    I=I+1
 106        IF(I.GT.MAXBUF)I=1
 107        IF(TESTB(FLAG,I))GO TO 11
 108        J=I
 109        IF(J.GE.MAXBUF)J=0
 110  C
 111  C  ASSEMBLY CODE TO ACCESS UFTT(1,J+1) NEEDED IN NEXT
 112  C  FORTRAN STATEMENT.
```

```
113  C
114        INLINE
115          LDI,3  10
116        MPM,3          J              REG3=J*10
117        ADM,3          UFTT           ;REG3=UFTT+ J*10
118          LDX,3,3
119          STM,3  TEMP1      TEMP1= UFTT(1,J+1)
120        FINI
121        IF(.NOT.TESTB(TEMP1,16))IPERR=IPERR+1
122  C
123  C  ASSEMBLY CODE TO ACCESS UFTT(1,I),BUF(1,I) AND
124  C  TO LOAD UFTT(4,I) WITH TRKCNT*24.
125  C
126        INLINE
127          LDM,3  I
128          SBR,3,15          REG3= I-1
129          LDI,2  10
130          MPR,3,2           REG3= (I-1)*10
131        ADM,3          UFTT           ;REG3=UFTT+(I-1)*10
132          STM,3  TEMP3
133          LDX,4,3
134          STM,4  TEMP1      TEMP1= UFTT(1,I)
135          ABR,3,15
136          ABR,3,14          REG3 POINTS TO UFTT(4,I)
137          LDI,5  24
138        MPM,5          TRKCNT         ;REG5=TRKCNT*24
139          STX,5,3           UFTT(4,I)= TRKCNT*24
140          LDM,2  I
141          SBR,2,15          REG2 = I-1
142          LDI,3  2946
143        MPR,3,2                      ;REG3=(I-1)*2946
144        ADM,3          BUF            ;REG3=BUF+(I-1)*2946
145          STM,3  A001
146          LDX,3,3
147          STM,3  TEMP2      TEMP2= BUF(1,I)
148        FINI
149  15    IF(TESTB(TEMP1,16))GO TO 12
150        ERROR=ERROR+1
151  13    IF(.NOT.TESTB(TEMP1,16))GO TO 13
152  12    CALL SETBT(FLAG,I)
153        INLINE
154          LDI,2  WDCNT*2
155          STM,2  A001+1
156          LDM,2  TEMP3
157          REX,#81
158  A001  DFC  0,0
159          REX,#14
160          DFC  #8000,4      DELAY TO ALLOW OTHER TASKS TO RUN
161        FINI
162        TRKCNT=TRKCNT+1
163        IF(TRKCNT.GE.MAXTRK)GO TO 100
164        GO TO 11
165  100   CONTINUE
166        INLINE
167        LDI,1          #4500
168        OCA,1,DEV                    ;TERMINATE 4805 DEVICE
```

```
169         FINI
170         IF(ERROP.NE.0)CALL MESSAG(MES1,.TRUE.)
171         IF(IPERR.NE.0)CALL MESSAG(MES2,.TRUE.)
172         GO TO 99
173  999    STOP
174         END
```

APPENDIX E

DRIVE STAND CONTROLLER

The drive stand controller is a microprocessor based system which provides control of the test facility drive stands. Actual control of a drive stand is achieved by supplying a DC voltage to the drive stand power supply. A voltage of 0-10 VDC input to the power supply produces a drive stand speed of 0-10,000 rpm. The drive stand controller receives control commands from the minicomputer (via the procedure explained in the main text of this manual), processes these commands, and issues the appropriate DC voltage signals to the drive stand power supply.

The drive stand controller consists of an Intel 8080 microprocessor, associated peripheral components, and a digital-analog conversion (DAC) circuit. Following is a discussion of how these components interact to process drive stand commands.

The microprocessor basically inputs commands from the minicomputer, converts the commands to the format required by the DAC circuit, and outputs the processed commands to the DAC at timed intervals. The DAC converts the digital signal to the appropriate DC voltage.

The microprocessor contains a read-only memory (ROM) programmed with a control routine. This control routine directs the system components to perform the input, process, output sequence. The source language for the control routine is Intel PLM. A copy of this code is attached.

Commands are input from the minicomputer via a peripheral interface device (8255). The 8255 receives a serial command from the minicomputer, converts this command to parallel, and issues an interrupt to the microprocessor CPU signalling that an input command is available.

The CPU accepts the command and converts it to the format required by the DAC to produce the DC voltage which will achieve the requested drive stand speed. To control acceleration rate, this DC voltage will be output in timed steps.

An interval timer (8253) outputs a command from the microprocessor to the DAC every 10 milliseconds. Therefore, for example, to accelerate the drive stand from 0 rpm to 1,000 rpm at 10,000 rpm/sec (maximum acceleration rate), the following series of DC voltage steps would be output from the DAC.

DC voltage	time (msec)
0	0
0.1	10
0.2	20
0.3	30
0.4	40
0.5	50
0.6	60
0.7	70
0.8	80
0.9	90
1.0	100

The minicomputer and the drive stand controller operate in a handshaking mode as follows. A time count is decremented by the control routine on each iteration. Input of a new command from the minicomputer resets this count. Thus to maintain a speed, the minicomputer must periodically output the command which defines that speed. Details of how this is achieved are included in the main text of this manual.

If the controller count reaches zero, which means the minicomputer has failed to issue a new command, the drive stand controller is programmed to force the drive stands to zero speed. In addition, the controller must echo the DS# portion of the drive stand command to the minicomputer. If the minicomputer does not receive this echo, it puts itself in a hold state. This forces the drive stand controller to take the shutdown action described above, since no further commands will be issued by the minicomputer. Thus if communication is broken between the minicomputer and the drive stand controller, the drive stands are brought to zero speed.

Figures E-1 and E-2 are circuit drawings of the drive stand controller. Figure E-1 contains the additional circuitry added to a basic Intel MCS-80 microprocessor kit to achieve the structure needed for the drive stand controller. For further details on the MCS-80 kit refer to Intel manual "MCS-80 System Design Kit User's Guide." Figure E-2 is the digital-to-analog conversion circuit which produces the analog signal output to the drive stand power supply.

```
*D MODCOMP SOURCE EDITOR     DATE  01/03/80  14:44:20        PAGE   1

  1  /*  FOLLOWING IS THE PLM SOURCE CODE FOR THE CONTROL  */
  2  /*  ROUTINE STORED IN THE EPROM OF THE INTEL 8080 MICRO-  */
  3  /*  PROCESSOR OF THE DRIVE STAND CONTROLLER  */
  4       DECLARE FLAG BYTE;
  5       DECLARE (SPEED,RATE)(4)BYTE;
  6       DECLARE (DS,K,TIME)BYTE;
  7       DECLARE (SPD,RT,CONTROL)(4)ADDRESS;
  8       DECLARE BUF(7) BYTE;
  9  READMC$ PROCEDURE INTERRUPT 1;
 10  /*  ROUTINE TO READ COMMAND FROM MODCOMP  */
 11  /*  INVOKED BY TYPDY OF 8251 AT LEVEL 1  */
 12       DECLARE (VALUE,POINT) BYTE;
 13  /*  INPUT COMMAND AND STRIP OFF PARITY  */
 14       VALUE = INPUT(0FAH) AND 7FH;
 15  /*  CHECK POINTER INTO BUFFER  */
 16       IF ((POINT<7) OR (VALUE=0DH))
 17         THEN DO;
 18              IF(VALUE=0DH) /* 0D=CR;BEGINNING OF NEW COMMAND */
 19                THEN POINT=0; /* RESET POINTER */
 20                ELSE DO;
 21  /*  CHANGE VALUE TO ASCII AND STORE IN COMMAND BUFFER  */
 22                     IF(VALUE< 3AH)
 23                       THEN BUF(POINT) = VALUE - 30H;
 24                       ELSE BUF(POINT) = VALUE - 37H;
 25  /*  CHECK FOR VALID HEX DIGIT  */
 26                     IF(BUF(POINT) < 10H)
 27                       THEN DO;
 28                            POINT = POINT + 1; /* INCREMENT POINTER */
 29  /*  IF POINTER EXCEEDS 6, SET FLAG TO INDICATE COMMAND BUFFER IS FULL */
 30                            IF (POINT=7)
 31                              THEN FLAG=0;
 32                            END;
 33                     END;
 34              END;
 35  /*  RESET INTERRUPT CONTROLLER  */
 36              OUTPUT(0F6H) = 27H;
 37  END READMC;
 38  /*                                                    */
 39  /*                                                    */
 40  ZEROSPEEDS$ PROCEDURE;
 41  /*  ROUTINE WHICH FORCES ALL SPEED AND RATE OUTPUT COMMANDS  */
 42  /*  TO ZERO.  ROUTINE IS CALLED AT INITIALIZATION AND WHENEVER */
 43  /*  MODCOMP FAILS TO ISSUE A NEW COMMAND BEFORE TIMER EXPIRES  */
 44      DO K= 1 TO 3;
 45         SPD(K)=0;
 46         RT(K)=0;
 47      END;
 48      RETURN;
 49  END ZEROSPEEDS;
 50  /*                                                    */
 51  /*                                                    */
 52  DRIVESTAND$ PROCEDURE INTERRUPT 2;
 53  /*  ROUTINE TO UPDATE DRIVESTAND SPEEDS  */
 54  /*  INVOKED BY 12 MSEC TIMER AT LEVEL 2  */
 55      DECLARE (HSB,LSB,K) BYTE;
 56  /*  ENABLE OTHER INTERRUPTS, SO THAT THIS ROUTINE CAN BE  */
```

```
/*  INTERRUPTED BY LEVEL 1 (INPUT)  */
    ENABLE;
    TIME=TIME-1;  /*  DECREMENT COMMAND TIMER  */
/*  CHECK TO SEE IF TIMER HAS EXPIRED  */
    IF(TIME=0)
       THEN CALL ZEROSPEEDS; /* IF SO, FORCE DRIVES TO ZERO */
/*  OTHERWISE, CALCULATE NEW OUTPUT COMMAND TO BE ISSUED TO D/A */
       ELSE DO K=1 TO 3;
/*  IF REQUESTED SPEED IS GREATER THAN LAST SPEED COMMAND  */
/*  ISSUED PLUS ONE INCREMENT OF RATE REQUESTED  */
          IF('SPD(K)>(CONTROL(K)+RT(K)) AND(RT(K)<>0))
/*  THEN NEW SPEED COMMAND IS SUM OF LAST COMMAND PLUS ONE  */
/*  INCREMENT OF RATE  */
             THEN CONTROL(K)=CONTROL(K)+RT(K);
/*  ELSE, CHECK IF REQUEST IS FOR DECELERATION  */
             ELSE IF ((SPD(K)<(CONTROL(K)-RT(K)))AND(RT(K)<>0)
               AND (CONTROL(K) > PT(K)))
/*  IF VALID DECELERATION, NEW COMMAND IS LAST COMMAND MINUS  */
/*  RATE  */
                  THEN CONTROL(K)=CONTROL(K)-RT(K);
/*  OTHERWISE, NEW COMMAND IS SPEED REQUESTED  */
                  ELSE CONTROL(K)=SPD(K);
/*  NOTE:  SINCE THIS ROUTINE EXECUTES EVERY 10 MSEC, A RATE OF  */
/*     1 CORRESPONDS TO 100 RPM/SEC  */
/*                                                                */
/*  BITS MUST BE INVERTED BEFORE OUTPUTTING TO D/A CONVERTER  */
       H&B=NOT(HIGH(CONTROL(K)) + (TIME AND 2F2H));
/*  NOTE:  UPPER HEX DIGIT OF TIMER IS OUTPUT TO A 7 SEGMENT  */
/*     LED DISPLAY ON CONTROLLER  */
       L&B=NOT(LOW(CONTROL(K)));
/*  DISABLE INTERRUPTS, SO THAT FOLLOWING OUTPUT ROUTINE IS  */
/*  NOT INTERRUPTED  */
       DISABLE;
/*  OUTPUT COMMAND TO PAIR OF OUTPUT PORTS DETERMINED BY K  */
       DO CASE K-1;
/*  DRIVE STAND #1 OUTPUT  */
          DO;
            OUTPUT(17H)=L&B;
            OUTPUT(37H)=H&B;
          END;
/*  DRIVE STAND #2 OUTPUT  */
          DO;
            OUTPUT(57H)=L&B;
            OUTPUT(77H)=H&B;
          END;
/*  DRIVE STAND #3 OUTPUT  */
          DO;
            OUTPUT(97H)=L&B;
            OUTPUT(0B7H)=H&B;
          END;
       END;
/*  RE-ENABLE INTERRUPTS  */
       ENABLE;
       END;
/*  RESET INTERRUPT CONTROLLER  */
    OUTPUT(0F6H)=20H;
```

```
113       RETURN;
114   END DRIVESTAND;
115   /*                                                    */
116   /*                                                    */
117   ECHO% PROCEDURE(CHAR);
118   /*  ROUTINE TO ECHO CHARACTER TO MOTCOMP  */
119       DECLARE CHAR BYTE;
120   /*  WAIT UNTIL TXRDY IS 1; I.E., TRANSMITTER IS READY  */
121       DO WHILE '(INPUT(0FBH)AND 01)=0;
122       END;
123   /*  CHANGE CHARACTER TO ASCII AND OUTPUT TO MOTCOMP  */
124       OUTPUT(0FAH)=CHAR+30H;
125   /*  SAVE THIS CHARACTER (WHICH IS DS#) IN VARIABLE K BY INPUTTING */
126       K=INPUT(0FAH);
127       RETURN;
128   END ECHO;
129   /*                                                    */
130   /*                                                    */
131   STARTCOUNT% PROCEDURE;
132   /*  ROUTINE TO INITIALIZE DEVICES  */
133   /*  INITIALIZE COUNTER, 8253  */
134       OUTPUT(0D7H)=0B6H;  /* MODE 3, BINARY COUNT  */
135       OUTPUT(0D6H)=30H;  /*  LSB OF COUNT  */
136       OUTPUT(0D6H)=0;  /*  MSB OF COUNT  */
137   /*  NOTE:  0030H GIVES A COUNT OF 10 MSEC  */
138   /*  INITIALIZE INTERRUPT CONTROLLER, 8259  */
139       OUTPUT(0F6H)=12H;  /* ICW1 - USE NORMAL INTERRUPT TRAPS  */
140       OUTPUT(0F7H)=0;  /* ICW2 - COMPLETES ICW1  */
141       OUTPUT(0F7H)=0F9H;  /* OCW1 - MASK ALL INTERRUPTS EXCEPT 1&2 */
142   /*  INITIALIZE RS232 PORT, 8251  */
143       OUTPUT(0FBH)=0CEH;  /* 2 STOP BITS, NO PARITY, 8 BIT CHAR */
144                           /*       64 X BAUD RATE                */
145       OUTPUT(0FBH)=27H;  /* ENABLE DEVICE  */
146        RETURN;
147   END STARTCOUNT;
148   /*                                                    */
149   /*                                                    */
150   /*  MAIN ROUTINE WHICH ACCUMULATES COMMANDS INPUT BY READMC  */
151   /*  AND MAKES THEM AVAILABLE TO DRIVESTAND  */
152       CALL STARTCOUNT;  /*  INITIALIZE DEVICES  */
153       CALL ZEROSPEEDS;  /*  INITIALIZE COMMANDS  */
154       FLAG=1;  /*  INITIALIZE BUFFER-FULL FLAG  */
155   /*  FOLLOWING IS SEQUENCE OF INSTRUCTIONS WHICH ARE CYCLED  */
156   /*  THROUGH CONTINUOUSLY  */
157   BEGIN% DISABLE;  /*  DISABLE INTERRUPTS  */
158   /*  CHECK FLAG TO SEE IF COMMAND BUFFER IS FULL  */
159       IF FLAG=0
160   /*  IF NOT, SKIP FOLLOWING PROCEDURE  */
161         THEN DO;
162               FLAG=1;  /*  RESET FLAG  */
163               TIME= 0FFH;  /*  RESET TIMER  */
164               DS= BUF(0);  /*  GET DS# FROM BUFFER  */
165   /*  CHECK FOR VALID DS#  */
166               IF('DS > 0) AND (DS < 41)
167                 THEN DO;
168   /*  ACCUMULATE SPEED AND RATE COMMANDS  */
```

```
169                     SPD(DS) = SHL(DOUBLE(SHL(BUF(1),4)+BUF(2)),4)
170                               +BUF(3);
171                     PT(DS) = SHL(DOUBLE(SHL(BUF(4),4)+BUF(5)),4)
172                               +BUF(6);
173                   END;
174             CALL ECHO(DS);  /*  ECHO TO MODCOMP  */
175 END;
176 ENABLE;  /*  RE-ENABLE INTERRUPTS  */
177 GO TO BEGIN;  /*  LOOP BACK  */
178 /*                                                          */
179 /*                                                          */
180 /*  NOTE:  IN IMPLEMENTING THIS DESIGN, IT WAS FOUND THAT A  */
181 /*     DELAY WAS NEEDED BEFORE ECHOING TO THE MODCOMP.  THE  */
182 /*     INSTRUCTIONS TO ACHIEVE THIS DELAY WERE ADDED DIRECTLY  */
183 /*     TO THE EPPOM.  THEPFORE, THE ACTUAL PROGRAM STORED IN  */
184 /*     THE EPROM IS SLIGHTLY DIFFERENT FROM THE ONE PRESENTED */
185 /*     HERE  */
186 EOF
```

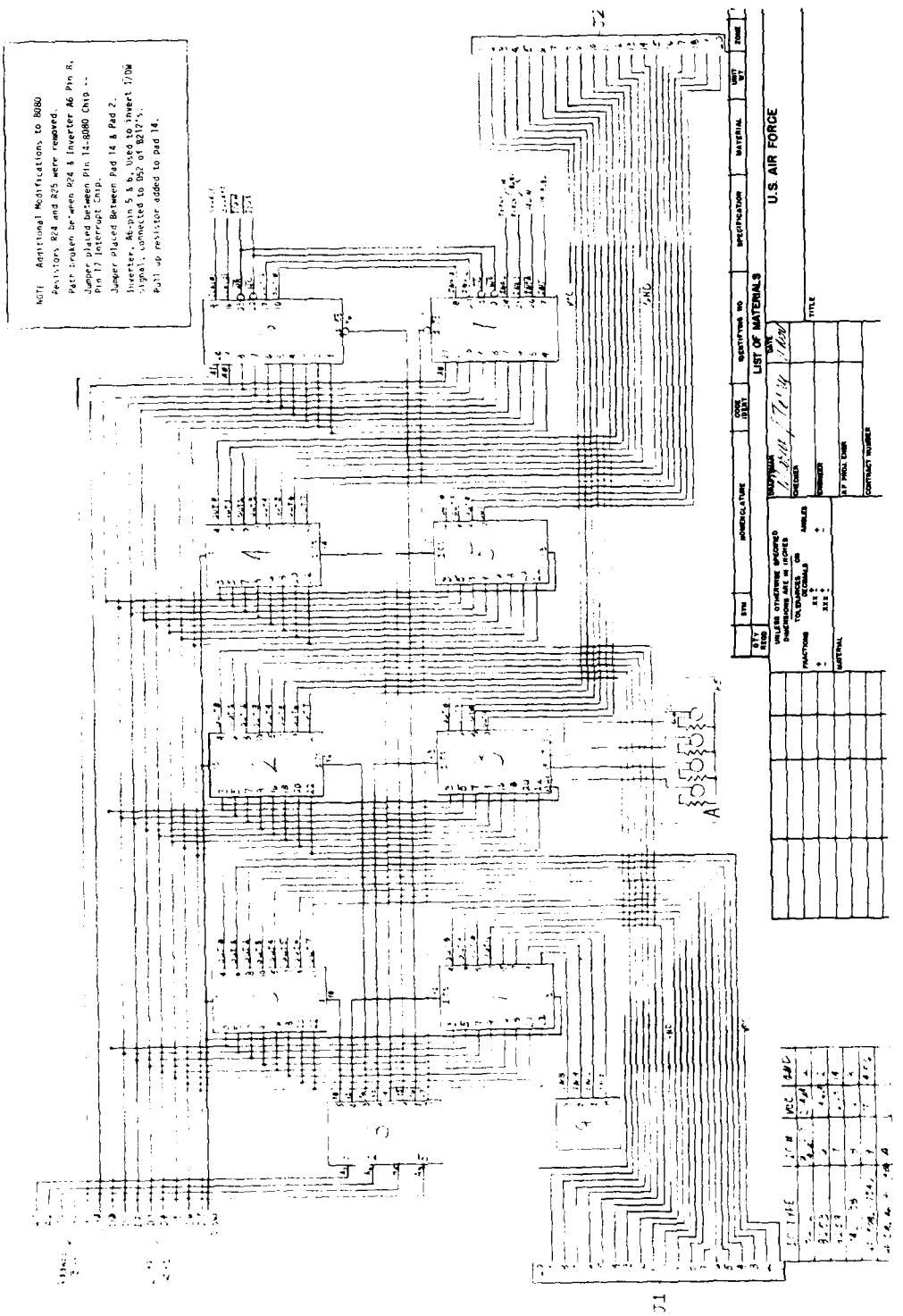


Figure E-1. Drive Stand Controller

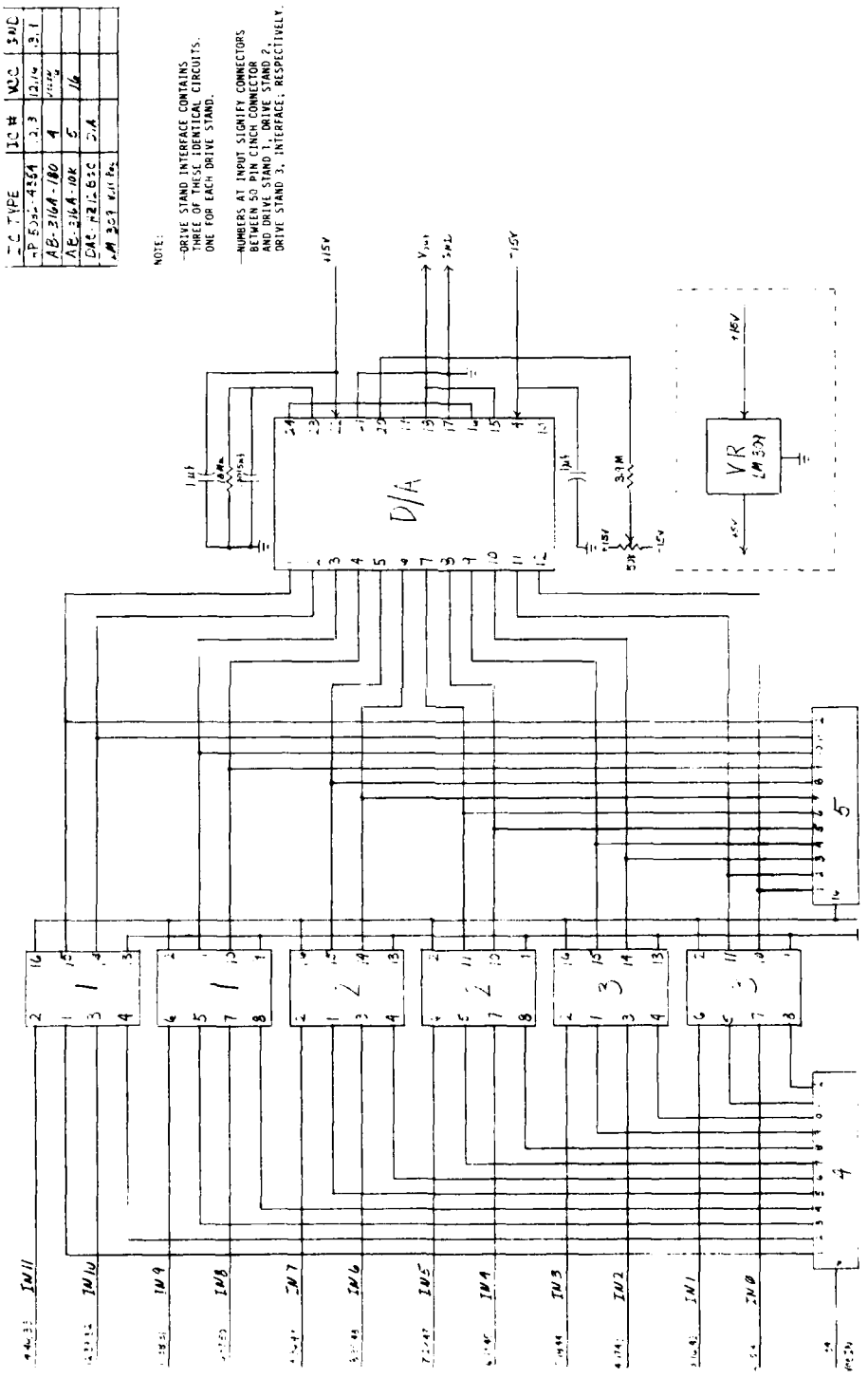


Figure E-2. D/A Converter

APPENDIX F

SAFETY CONDITION MONITORING SYSTEM

One feature of the Generator Test Facility is the automatic monitoring of certain safety conditions during computer-controlled testing. This safety monitoring is implemented using: (1) signals input to the minicomputer via the wide range analog input system, (2) a direct memory processor (DMP) routine, INITSAF, which stores these inputs in a dedicated buffer, and, (3) an overlimit checking routine, SAFCHECK, which executes during test sequence operation. The checking routine has output routines which, in the event of an overlimit safety condition, take actions to shut down the generator test. Following is a detailed description of these features.

In order to monitor a condition for safety purposes, a DC voltage signal proportional to that condition must be developed. This voltage signal should be relatively smooth and can be in the range from a few millivolts to ten volts. However, for better noise rejection and in the interest of standardization, it is recommended that a signal varying from 0-5 VDC be supplied. In addition, filtering and/or amplification of the raw transducer output is sometimes required.

After conditioning, these voltage signals are presented to an input channel of the analog input system of the Modcomp minicomputer. The analog input system accepts voltage inputs of 12 different ranges varying from 5 millivolts to 10.24 volts. The software routine used to input from the device selects the range for reading and optionally a zero suppression value. The zero suppression feature measures the variance of a reading about a preselected value. By specifying the safety limit for each signal as its zero suppression value, a positive input is present only if the signal is over its limit.

The analog input system can address up to 128 different channels. The channel addresses which exist in the system depend upon which card slots in the card file are occupied. Refer to Modcomp technical manual for a full description of the wide-range solid state analog input system.

Following is a description of the software routines used to input, store, and check the safety condition signals. These specific routines implement a safety monitoring scheme used for a 30/40 KVA VSCF generator system. To provide a safety monitoring scheme for other systems, these routines must be modified to reflect the new system's limits and particular safety signals. The signals monitored and limits used for the example safety scheme are given in Figure F-1.

The subroutine INITSAF initiates input from the analog input system. INITSAF is called by the routine RUNSYS immediately before execution of any test sequence is begun. Input is performed by a DMP which leaves the main processor of the minicomputer free to control generator test sequence execution. The DMP input causes a dedicated memory buffer to be filled with indications of any overlimit safety condition. All that is required to complete the safety monitoring scheme is a method to recognize an overlimit signal and the means to shut the test down. These features are included in the drive stand control task, DSC.

The drive stand control task is the software routine which controls the drive stand speed and acceleration rate. This control is provided through an Intel 8080 microprocessor-based conditioning circuit. Details of the control software of the DSC task and the drive stand controller are given in other sections of this manual. What is important to the safety monitoring scheme is that the DSC task is executing almost continuously during the execution of a generator test sequence. Therefore, by using the DSC task to invoke the overlimit checking routine, a constant monitoring of safety conditions is provided.

Upon activation of the DSC task, a key in the task communication area is set to zero. As long as this key (identified as SBF) is zero, no overlimit checking is performed. The RUNSYS routine, upon initiation of execution of a test sequence, sets the SBF key to a nonzero value. In fact, the SBF key is set to the address of the buffer of safety data located in central memory. Setting the SBF key causes the SAFCHECK subroutine of the DSC task to perform an overlimit check on each condition being monitored. Because of the zero suppression scheme employed by the INITSAF routine, any positive value in the safety buffer represents an overlimit signal.

AIS CHANNEL #	SIGNAL	LIMIT	TRANSDUCER OUTPUT
4	DRIVE STAND SPEED	8667 RPM	8.667 V
0	OIL TEMPERATURE IN	75°C	75 mV
1	OIL PRESSURE IN	120 psi*	-1200 mV
5	VIBRATION	8G	400 mV
68	DRIVE STAND BEARING #1 TEMPERATURE	100°F	2.56 V
69	DRIVE STAND BEARING #2 TEMPERATURE	100°F	2.56 V
70	DRIVE STAND BEARING #3 TEMPERATURE	100°F	2.56 V
71	DRIVE STAND BEARING #4 TEMPERATURE	100°F	2.56 V
64	GEARBOX BEARING #1 TEMPERATURE	200°F	4.0 V
65	GEARBOX BEARING #2 TEMPERATURE	200°F	4.0 V
66	GEARBOX BEARING #3 TEMPERATURE	200°F	4.0 V
67	GEARBOX BEARING #4 TEMPERATURE	200°F	4.0 V
7	DRIVE STAND SPEED	3000 RPM*	3 V

* OIL PRESSURE IN must be above 120 psi when the drive stand speed monitored at channel #7 is above 3000 RPM

Figure F-1 - Example of Safety Scheme

In the event a safety condition is found to be overlimit, the SAFCHECK routine takes the particular shutdown action for that condition. In this way, the most critical actions for that condition can be performed. The mechanism for performing shutdown involves basically closing a relay contact in the minicomputer input/output interface subsystem which controls power to the circuit which performs the action. The SAFCHECK routine also issues a message to the test operator on the ISC color display terminal. This message states what condition caused the shutdown and what actions were taken during shutdown. Also during shutdown, the DSC task puts itself in a hold state. As explained in Appendix E, this forces the drive stands to decelerate to zero speed.

To continue testing after an automatic shutdown, the test operator must take several steps. First, he must discover and correct the conditions which caused the shutdown. After correcting the overlimit condition, the operator may have to perform other steps. If the message, "DRIVE POWER REMOVED" is issued on the ISC display, the operator must disable the "lock-out" mechanism of the drive stand controller. This lockout interrupts the DC voltage supplied to the drive stand power supply. To reset the lockout, momentarily depress the red push button on the top of the lockout box. Specific shutdown procedures may require additional restart actions. These will be included in the ISC message. Finally, the DSC task must be resumed. This is done by depressing the CONSOLE INTERRUPT switch on the Modcomp minicomputer control panel and typing/DSC/A on the teletype console. This aborts the DSC task which not only resumes its operation but also causes the safety checking to be inactive until the RUNSYS routine activates it again.

It should be noted that the RUNSYS routine continues executing during and after an automatic shutdown. This execution must be allowed to complete before any restart actions are performed, so that all actions initiated during the test sequence are completed. When RUNSYS has completed a test sequence and is ready for a new test sequence maintenance command, it issues the message "ENTER COMMAND" on the Tektronix CRT screen. The test restart actions can then be performed.

Copies of the code for the INITSAF and SAFCHECK routines are included at the end of this appendix. These routines are written in Modcomp Fortran and assembly language.

Also included is a copy of the routine SAFETY. This routine can be run when operating the generator test facility in the manual mode. SAFETY inputs the safety data which is processed by the INITSAF and SAFCHECK routines during computer-controlled testing. However, instead of checking for overlimit conditions, SAFETY presents a tabular display of the signals on the ISC display screen. Thus during manual operation, the test operator can visually monitor these safety conditions. In addition, the SAFETY routine should be run prior to computer-controlled testing to insure that the safety signal transducers, and the minicomputer analog input system are operating correctly.

To execute the SAFETY routine, the test operator should type the following commands on the Tektronix terminal:

```
[JOB]
[EXE SAF LM]
```

The routine will respond with the message,

```
ENTER MAX SPEED
```

The user then enters the maximum drive stand speed to be allowed during manual operation. The drive stand speed, in addition to being displayed on the ISC terminal, is also checked against this maximum speed. If the drive stand speed exceeds this maximum, the routine SAFETY activates an emergency trip of the drive stand which then coasts to zero speed. This protection is provided in the manual mode since generator overspeed is probably the most critical unsafe condition.

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  22:24:44        PAGE   1

 1          SUBROUTINE INITSAF(ITEMP)
 2  C
 3  C  ROUTINE INITIALIZES A DMP TRANSFER OF SAFETY MONITORING
 4  C  DATA INPUT FROM ANALOG INPUT SYSTEM (DEVICE NAME=AII) TO
 5  C  BUFFER (IBUF).
 6  C  EACH ANALOG INPUT HAS A ZERO SUPPRESSION VALUE EQUAL TO THE
 7  C  UPPER LIMIT OF THE SIGNAL BEING MONITORED, SO THAT AN
 8  C  ENTRY IS MADE INTO THE BUFFER ONLY IF THE SIGNAL EXCEEDS
 9  C  THE UPPER LIMIT.
10  C  THE DSC TASK SCANS THE BUFFER AND TAKES 'SHUTDOWN' ACTION
11  C  IF IT FINDS ANY NON-ZERO ENTRY.
12  C
13  C  NOTE:  ITEMP IS A LOCATION IN ROUTINE RUNSYS IN WHICH THIS
14  C  SUBROUTINE STORES ADDRESS OF ORIGINAL SYSTEM SI INTERRUPT
15  C  HANDLER FOR DEVICE AII.
16  C
17  C
18  C  ROUTINE CURRENTLY HANDLES 13 CHANNELS OF ANALOG INPUT;
19  C  MAXIMUM IS 16.
20  C
21  C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
22  C  *  *  *  CATALOGUED ON UL 10,31,79   *  *  *  *  *  *  *  *
23  C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
24  C
25  C
26          DIMENSION ITAB(34)
27          DATA ITAB/Z5404,Z0D8A,Z5400,Z001E,Z5401,ZFE20
28         1,Z5405,Z0140,Z5444,Z0400,Z5445,Z0400,Z5446,Z0400
29         1,Z5447,Z0400,Z5440,Z0640,Z5441,Z0640,Z5442,Z0640
30         1,Z5443,Z0640,Z5407,Z04B0,8*0/
31  C
32  C  ITAB IS TABLE WHICH CONTROLS INPUT OF ANALOG DATA.
33  C  EACH CHANNEL IS CONTROLLED BY A TWO-WORD ENTRY.  FIRST
34  C  WORD SPECIFIES RANGE AND CHANNEL, SECOND WORD SPECIFIES
35  C  ZERO SUPPERSSION VALUE.
36  C  FOR COMPLETE DESCRIPTION OF ANALOG INPUT SYSTEM, REFER TO
37  C  MODCOMP INPUT/OUTPUT MANUAL.
38  C
39  C  SIGNALS PRESENTLY BEING MONITORED:
40  C
41  C  CHANNEL#      SIGNAL          RANGE          LIMIT
42  C       4       DS SPEED          A            8.667V=8667RPM
43  C       3       OIL TEMP IN       A            75MV=75C
44  C       1      OIL PRESS IN       A
45  C       7      D.S. SPEED         A
46  C       2      VIBRATION          A      400MV=8G
47  C      68      D.S. BEAR. TEMP    A
48  C      69      D.S  BEAR. TEMP    A
49  C      70      D.S  BEAR. TEMP.   A
50  C      71      D.S. BEAR. TEMP.   A
51  C      64      G.B. BEAR. TEMP.   A
52  C      65      G.B. BEAR. TEMP.   A
53  C      66      G.B. BEAR. TEMP.   A
54  C      67      G.B. BEAR. TEMP.   A
55  C
56  C
```

```
 57  C
 58  C  NCH=NUMBER OF CHANNELS BEING MONITORED
 59        NCH=13
 60        LNCH=NCH*2
 61  C  FOLLOWING IS ASSEMBLY LANGUAGE CODE WHICH INITIATES DMP
 62  C  TRANSFER
 63        INLINE
 64          LDM,10  #00D3      STORE SI TRAP ADDRESS IN TEMP. LOCATION
 65          STM*,10  ITEMP
 66          LDI,2   SI         SI INTERRUPT HANDLER
 67          STM,2   #00D3      PUT IN SI TRAP
 68          LDI,2   NCH        GET ADDR OF NCH
 69          LDX,5,2            P5=NCH
 70          TTR,2,5            R2=-NCH,INPUT WORD COUNT
 71          ZBR,2,0            SET DMP LINK BIT
 72          STM,2   #0066      INPUT TC
 73          STM,2,5 IBUF       1ST WORD AFTER DATA BUFFER
 74          LDI,2   LNCH       GET ADDR OF LNCH
 75          LDX,6,2            R6=LNCH
 76          TTR,2,6            R2=-LNCH,OUTPUT WORD COUNT
 77          ZBR,2,0            SET DMP LINK BIT
 78          STM,2   #0065      OUTPUT TC
 79          STM,2,6 ITAB       1ST WORD AFTER SCAN TABLE
 80          LDI,2   IBUF       ADDR. OF DATA BUFFER
 81          STM,2   #0076      INPUT TA
 82          ABR,5,15           R5=NCH+1
 83          STM,2,5 IBUF       2ND WORD AFTER DATA BUFFER
 84          LDI,2   ITAB       ADDR. OF SCAN TABLE
 85          STM,2   #0075      OUTPUT TA
 86          ABR,6,15           R6=LNCH+1
 87          STM,2,6 ITAB       2ND WORD AFTER SCAN TABLE
 88  *  USE CUSTOM REX CALL TO PASS ADDRESS OF DATA BUFFER, IBUF,
 89  *  TO DSC TASK
 90          LDI,2   @SBF       SBF=SAFETY BUFFER
 91          LDI,3   IBUF
 92          REX,#4E
 93  *  ALSO PASS NUMBER OF CHANNELS OF DATA, NCH, TO DSC TASK
 94          LDI,2  @NCH
 95          LDI,3  NCH
 96          LDX,3,3
 97          REX,#4E
 98  *  INITIATE DMP
 99          LDI,3  #4400       SET UP TERMINATE COMMAND
100          OCB,3,2            TERMINATE OUTPUT
101          OCB,3,3            TERMINATE INPUT
102  BSY1    ISB,3,3       GET STATUS OF INPUT
103          TBRB,3,7 BSY1      IF BUSY, LOOP
104  BSY2    ISB,3,2       GET STATUS OF OUTPUT
105          TBRB,3,7 BSY2      IF BUSY, LOOP
106          LDI,3  #C000       SET UP TRANSFER INITIATE COMMAND,IMP,DI=0
107          OCB,3,3            INITIATE INPUT
108          OCB,3,2            INITIATE OUTPUT
109          BRU  OUT           JUMP AROUND DATA
110  SI      CIR              IGNORE SI INTERRUPTS
111  IBUF    BES  18,0
112  OUT     NOP
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  22:24:44          PAGE   7

 113          FINI
 114          RETURN
 115          END
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  22:25:03        PAGE   1

  1         SUBROUTINE SAFCHECK
  2  C
  3  C  ROUTINE WHICH CHECKS SAFETY MONITORING BUFFER FOR
  4  C  ANY ENTRY LARGER THAN LIMIT
  5  C  CURRENT ROUTINE USED FOR F-18 VSCF TEST
  6  C
  7  C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
  8  C  *  *  *  CATALOGUED ON UL  11,9,79  *  *  *  *  *  *  *  *  *  *  *
  9  C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
 10  C
 11  C  EXTERNAL SUBROUTINES REQUIRED:
 12  C
 13  C   NAME           LOCATION
 14  C
 15  C  1. AIFL            LB
 16  C  2. INITISC         UL
 17  C  3. ALNUM           UL
 18  C  4. SETBF           UL
 19  C  5. SETBT           UL
 20  C  6. BLNKOFF       UL
 21  C
 22  C
 23  C  INITIALIZE STORAGE LOCATIONS
 24         ITEMP1=0
 25         ITEMP2=0
 26  C  GET ADDRESS OF SAFETY BUFFER (SBF) FROM TASK DATA AREA
 27         INLINE
 28           LDI,2  @SBF
 29           REX,#4D
 30           STM,3  ITEMP1
 31         FINI
 32  C  IF SBF ADDR. = 0, SAFETY ACQUISITION HAS NOT BEEN INITIATED
 33         IF(ITEMP1.EQ.0)RETURN
 34  C  GET NUMBER OF CHANNELS OF SAFETY DATA BEING USED (NCH)
 35         INLINE
 36           TRR,4,3      SAVE ADDRESS OF SBF IN REG 4
 37           LDI,2  @NCH
 38           REX,#4D
 39           STM,3  ITEMP1
 40         FINI
 41  C  CHECK FOR ANY VALUE OVER LIMIT
 42  C  NOTE:  LOOP EXTENDS ONLY TO NCH-1, SINCE LAST CHANNEL
 43  C  IS A MINIMUM DS SPEED WHICH IS CHECKED ONLY IF OIL PRESS IS LOW
 44         JEND=ITEMP1-1
 45         DO 10 J=1,JEND
 46  C  GET ONE WORD OF SAFETY DATA FROM SBF
 47         INLINE
 48           LDX,2,4      REG 2 = SAFETY WORD
 49           STM,2  ITEMP2     STORE IN FORTRAN VARIABLE
 50         FINI
 51  C  CONVERT ANALOG INPUT VALUE TO DECIMAL
 52         CALL AIFL(1,ITEMP2,DATA)
 53  C  CHECK FOR OVER LIMIT VALUE
 54         IF(DATA.GT.0.)GO TO 50
 55  C  INCREMENT POINTER INTO SBF
 56         INLINE
```

```
AD MODCOMP SOURCE EDITOR      DATE  12/27/79  22:24:44        PAGE   7

113             FINI
114             RETURN
115             END
```

```
57          ABR,4,15
58        FINI
59  10    CONTINUE
60  C  IF NO "OVER LIMIT" VALUES, RETURN
61        RETURN
62  C  ERASE ISC SCREEN WITH BLINK ON
63  C  SELECT PROPER SHUTDOWN ROUTINE
64  50    GO TO(100,300,400,500,700,700,700
65       1,700,800,800,800,800),J
66  C  ROUTINE TO HANDLE DRIVE STAND OVER SPEED
67  100   CALL STOPDRIVE
68        CALL EMERTRIP
69        CALL INITISC(1)
70  C  FOLLOWING STATEMENTS PRINT MESSAGE ON ISC DISPLAY
71        CALL ALNUM(2,10,'DS OVERSPEED',12)
72        CALL ALNUM(2,15,'DS CONTROL POWER REMOVED',24)
73        CALL ALNUM(2,20,'DS EMERGENCY TRIP',18)
74        CALL BLNKOFF
75  C  HALT DSC TASK AND PRINT MESSAGE
76  C     INLINE
77  C       REX,#90
78  C       DFC MES1
79  C       BRU OUT
80  CMES1    DFC " DS OVERSPEED",0
81  COUT     NOP
82  C     FINI
83        CALL BELL
84        RETURN
85  C  ROUTINE TO HANDLE REVERSE DRIVE STAND ROTATION
86  200   CALL STOPDRIVE
87        CALL EMERTRIP
88        CALL INITISC(1)
89  C  ISC MESSAGE
90        CALL ALNUM(2,10,'REVERSE DS ROTATION',20)
91        CALL ALNUM(2,15,'DS CONTROL POWER REMOVED',24)
92        CALL ALNUM(2,20,'DS EMERGENCY TRIP',18)
93        CALL ALNUM(2,25,'DSC TASK HALTED',16)
94        CALL BLNKOFF
95        INLINE
96          REX,#90
97          DFC MES2
98          BRU OUT2
99  MES2     DFC " REVERSE DS ROTATION",0
100 OUT2     NOP
101       FINI
102       RETURN
103 C  OIL TEMPERATURE ROUTINE
104 300   CALL STOPDRIVE
105 C  ISC MESSAGE
106       CALL INITISC(1)
107       CALL ALNUM(2,10,'OIL TEMP IN TOO HIGH',20)
108       CALL ALNUM(2,15,'DS CONTROL POWER REMOVED',24)
109       CALL ALNUM(2,20,'DSC TASK HALTED',16)
110       CALL BLNKOFF
111       INLINE
112         REX,#90
```

```
113           DFC MES4
114           BRU OUT4
115   MES4    DFC " OIL TEMP HI",0
116   OUT4    NOP
117         FINI
118         RETURN
119   C  OIL PRESSURE ROUTINE
120   400   CONTINUE
121   C  CHECK DRIVE STAND SPEED
122         INLINE
123           ABR,4,14
124           ABR,4,12
125           LDX,2,4
126           STM,2 ITEMP2     MIN. DS SPEED
127         FINI
128   C  CHECK IF DS UP TO SPEED
129         CALL AIFL(1,ITEMP2,DATA)
130         IF(DATA.LT.0.)RETURN
131         CALL STOPDRIVE
132         CALL INITISC(1)
133   C  ISC MESSAGE
134         CALL ALNUM(2,10,'LOW OIL PRESSURE',16)
135         CALL ALNUM(2,15,'DS CONTROL POWER REMOVED',24)
136         CALL ALNUM(2,20,'DSC TASK HALTED',24)
137         CALL BLNKOFF
138         INLINE
139           REX,#90
140           DFC MES5
141           BRU OUT5
142   MES5    DFC " LOW OIL PRESSURE",0
143   OUT5    NOP
144         FINI
145         RETURN
146   C  VIBRATION ROUTINE
147   500   CALL STOPDRIVE
148         CALL INITISC(1)
149   C  ISC MESSAGE
150         CALL ALNUM(2,10,'EXCESSIVE VIBRATION',20)
151         CALL ALNUM(2,15,'DS CONTROL POWER REMOVED',24)
152         CALL ALNUM(2,20,'DSC TASK HALTED',16)
153         CALL BLNKOFF
154         INLINE
155           REX,#90
156           DFC  MES6
157           BRU  OUT6
158   MES6    DFC  " EXCESSIVE VIBRATION",0
159   OUT6    NOP
160         FINI
161         RETURN
162   C  SCAVENGE PUMP ROUTINE
163   600   CALL STOPDRIVE
164   C  ISC MESSAGE
165         CALL ALNUM(2,10,'SCAVENGE NOT OPERATING',22)
166         CALL ALNUM(2,15,'DS CONTROL POWER REMOVED',24)
167         CALL ALNUM(2,20,'DSC TASK HALTED',16)
168         CALL BLNKOFF
```

```
      INLINE
        REX,#90
        DFC  MES7
        BRU OUT7
MES7    DFC  " SCAVENGE NOT OPERATING",0
OUT7    NOP
      FINI
      RETURN
C  DRIVE STAND BEARINGS ROUTINE
700   CALL STOPDRIVE
      CALL INITISC(1)
C  ISC MESSAGE
      CALL ALNUM(2,10,'DS BEARINGS OVERHEATED',22)
      CALL ALNUM(2,15,'DS CONTROL POWER REMOVED',24)
      CALL ALNUM(2,20,'DSC TASK HALTED',16)
      CALL BLNKOFF
      INLINE
        REX,#90
        DFC  MES8
        BRU  OUT8
MES8    DFC " DRIVE STAND BEARINGS OVERHEATED",0
OUT8    NOP
      FINI
      RETURN
C  GEAR BOX BEARINGS ROUTINE
800   CALL STOPDRIVE
      CALL INITISC(1)
C  ISC MESSAGE
      CALL ALNUM(2,10,'GEAR BOX BEARINGS TOO HOT',26)
      CALL ALNUM(2,15,'DS CONTROL POWER REMOVED',24)
      CALL ALNUM(2,20,'DSC TASK HALTED',16)
      CALL BLNKOFF
      INLINE
        PEX,#90
        DFC  MES9
        BRU OUT9
MES9    DFC  " GEAR BOX BEARINGS OVERHEATED",0
OUT9    NOP
      FINI
      RETURN
      END
C
C
      SUBROUTINE SETBIT(NBIT)
C
C  ROUTINE TO SET BIT AND OUTPUT IN EXTERNAL RELAY
C  OUTPUT SCHEME
C
      DATA IWORD/0/,NSCAN/Z4019/
C  SET BIT IN CONTROL WORD
      CALL SETBF(IWORD,NBIT)
C  OUTPUT CONTROL WORD
      INLINE
        LDI,2 UFT
        REX,1
        DFC IWORD,2
```

```
225           BRU OUT
226           DFC NSCAN,1
227  UFT      DFC 0,@D00,#8400,2,0,0
228  OUT      NOP
229         FINI
230         RETURN
231         END
232  C
233  C
234         SUBROUTINE CLRBIT(NBIT)
235  C
236  C  ROUTINE TO CLEAR BIT AND OUTPUT IN EXTERNAL RELAY OUTPUT
237  C  SCHEME
238  C
239         DATA IWORD/0/,NSCAN/Z4019/
240  C  CLEAR BIT IN CONTROL WORD
241         CALL SETBT(IWORD,NBIT)
242  C  OUTPUT CONTROL WORD
243         INLINE
244           LDI,2 UFT
245           REX,1
246           DFC IWORD,2
247           BRU OUT
248           DFC NSCAN,1
249  UFT      DFC 0,@D00,#8400,0,0,0
250  OUT      NOP
251         FINI
252         RETURN
253         END
254  C
255  C
256         SUBROUTINE STOPDRIVE
257  C
258  C  FIRST USED IN 150 KVA IDG TEST
259  C  ALSO USED IN F-18 VSCF TEST
260  C  SHOULD BE APPLICABLE FOR ALL TESTS
261  C  PUTS PULSE ON CONTACT #12, WHICH REMOVES DS CONTROL
262  C  VOLTAGE AND ACTIVATES "LOCK-OUT"
263  C  "LOCK-OUT" IS RESET WITH RED BUTTON ON 4080 DS CONTROL
264  C INTERFACE BOX
265  C
266  C  SET BIT
267         CALL SETBIT(4)
268  C  WAIT .5 SEC
269         INLINE
270           REX,#14
271           DFC #8000,100
272         FINI
273  C  CLEAR BIT
274         CALL CLRBIT(4)
275         RETURN
276         END
277  C
278  C
279         SUBROUTINE EMERTRIP
280  C
```

```
281  C  USED FIRST IN 150 KVA IDG TEST
282  C  SHOULD BE APPLICABLE FOR ALL TESTS
283  C  PUTS A PULSE ON CONTACT #7, WHICH ACTIVATES DS EMERGENCY
284  C  TRIP
285  C
286  C  SET BIT
287        CALL SETBIT(12)
288  C  WAIT .5 SEC
289        INLINE
290          REX,#14
291          DFC #8000,100
292        FINI
293  C  CLEAR BIT
294        CALL CLRBIT(12)
295        RETURN
296        END
297  C
298  C
299        SUBROUTINE BLNKOFF
300        INLINE
301          LDI,2  UFT
302          REX,1
303          DFC  BUF,2
304          BRU  OUT
305  UFT     DFC  0,@ISC,#9030,0,0,0
306  BUF     DFC  #0F0E
307  OUT     NOP
308        FINI
309        RETURN
310        END
311  C
312  C
313        SUBROUTINE BELL
314  1     CONTINUE
315        INLINE
316          LDI,2  UFT
317          REX,1
318          DFC  BUF,2
319          BRU OUT
320  UFT     DFC  0,@ISC,#9030,0,0,0
321  BUF     DFC  #0707
322  OUT     NOP
323        FINI
324        IF(1.EQ.2`RETURN
325        GO TO 1
326        END
```

*D MODCOMP SOURCE EDITOR DATE 12/27/79 22:25:27 PAGE 1

```
      PROGRAM SAFETY
C
C  THIS ROUTINE IS RUN WHEN OPERATING A GENERATOR IN THE MANUAL MODE.
C  IT SERVES TWO FUNCTIONS.  FIRST, ALL SAFETY SIGNALS BEING MONITORED
C  ARE DISPLAYED IN TABULAR FORM.  THIS DEMONSTRATES THAT THOSE AIS
C  CHANNELS,TRANSDUCERS,ETC. ARE FUNCTIONING PROPERLY BEFORE COMPUTER
C  CONTROLLED TESTING IS BEGUN.  NOTE THAT FOR EACH NEW GENERATOR
C  SYSTEM, THIS ROUTINE MUST BE CHANGED TO TAKE INTO CONSIDERATION
C  ALL SAFETY SIGNALS USED FOR THAT SYSTEM.  SECONDLY, THE USER ENTERS
C  THE MAXIMUM DRIVE STAND SPEED ALLOWABLE FOR THE CURRENT GENERATOR.
C  IF THIS SPEED IS EXCEEDED, THE EMERGENCY TRIP CIRCUIT IS ACTIVATED,
C  WHICH STOPS THE DRIVE STAND.
C
      DIMENSION INUFT(22),IBUF(12)
      DATA INUFT/0,12,0,0,Z9400,0,0,0,'AI','I',Z0204
     1,Z0200,Z0201,Z0205,Z0244,Z0245,Z0246,Z0247
     1,Z0240,Z0241,Z0242,Z0243/
C  USER ENTERS MAX SPEED
      WRITE(8,100)
100   FORMAT(' ENTER MAX SPEED')
      READ(7,200)SPEED
200   FORMAT(F10.2)
C  ERASE ISC SCREEN
      CALL GRAPHICS(3)
      CALL ERASCREEN
10    CONTINUE
C  LOOP TO INPUT AND DISPLAY SAFETY SIGNALS
      DO 70 KK=1,10000
C  INPUT BUFFER OF SAFETY DATA
      CALL AIRDW(12,INUFT,IBUF,M)
C  CONVERT DRIVE STAND SPEED TO FLOATING POINT
      CALL AIFL(1,IBUF(1),DATA)
C  CHECK FOR OVERSPEED
      IF(DATA.LT.SPEED)GO TO 20
C  ACTIVATE DRIVE STAND EMERGENCY TRIP
      CALL EMERTRIP
C  OUTPUT MESSAGE
      CALL INITISC(1)
      CALL ALNUM(2,10,'DS OVERSPEED',12)
      CALL ALNUM(2,15,'DS EMERGENCY TRIP',18)
      CALL ALNUM(2,20,'SAFETY MONITORING HALTED',24)
C  EXIT THIS ROUTINE
      STOP
C  OUTPUT DRIVE STAND SPEED
20    DATA=DATA*3.
      WRITE(6,300)DATA
300   FORMAT(//' DRIVE STAND SPEED = ',F10.2,' RPM')
C  CONVERT OIL TEMP TO FLT. PT.
      CALL AIFL(1,IBUF(2),DATA)
C  OUTPUT OIL TEMP
      WRITE(6,400)DATA
400   FORMAT(//' OIL TEMP IN = ',F10.2,' DEG C')
C  CONVERT OIL PRESSURE TO FLT. PT.
      CALL AIFL(1,IBUF(3),DATA)
C  CONVERT TO PSI
      DATA=DATA/10.
```

```
 57  C  OUTPUT OIL PRESSURE
 58         WRITE(6,500)DATA
 59  500    FORMAT(//' OIL PRESSURE IN = ',F10.2,' PSI')
 60  C  CONVERT VIBRATION READING TO FLT. PT.
 61         CALL AIFL(1,IBUF(4),DATA)
 62  C  CONVERT TO G'S
 63         DATA=DATA/100.
 64  C  OUTPUT VIBRATION
 65         WRITE(6,600)DATA
 66  600    FORMAT(//' VIBRATION = ',F10.2,' G')
 67  C  CHECK BEARING TEMPERATURES
 68         WRITE(6,650)
 69  650    FORMAT(80X)
 70  C  INITIALIZE FLAG WHICH INDICATES HOW MANY OVER-TEMP BEARINGS
 71         NGOOD=0
 72  C  CHECK DRIVE STAND BEARINGS
 73         DO 30 K=1,4
 74  C  CONVERT TO FLOATING POINT
 75         CALL AIFL(1,IBUF(K+4),DATA)
 76  C  CHECK FOR VALID READING
 77         IF(DATA.GT.500..AND.DATA.LT.2500.)GO TO 30
 78  C  OUTPUT INVALID READING
 79         WRITE(6,700)K,DATA
 80  700    FORMAT(' DRIVE STAND BEARING #',I1,' = ',F10.2,10X)
 81  C  INCREMENT FLAG
 82         NGOOD=NGOOD+1
 83  30     CONTINUE
 84  C  CHECK GEAR BOX BEARINGS
 85         DO 40 K=1,4
 86  C  CONVERT TO FLOATING POINT
 87         CALL AIFL(1,IBUF(K+8),DATA)
 88  C  CHECK FOR VALID READING
 89         IF(DATA.GT.2500..AND.DATA.LT.3500.)GO TO 40
 90  C  OUTPUT INVALID READING
 91         WRITE(6,800)K,DATA
 92  800    FORMAT(' GEAR BOX BEARING #',I1,' = ',F10.2,10X)
 93  C  INCREMENT FLAG
 94         NGOOD=NGOOD+1
 95  40     CONTINUE
 96  C  WERE ANY BEARING TEMPS BAD?
 97         IF(NGOOD.EQ.0)GO TO 60
 98  C  IF THERE WERE ANY BAD READINGS, WIPE OUT "OLD" MESSAGES
 99         LPEND=9-NGOOD
100         DO 50 LP=1,LPEND
101  50     WRITE(6,900)
102  900    FORMAT(80X)
103  C  GO TO END OF LOOP
104         GO TO 70
105  C  OUTPUT MESSAGE INDICATING ALL BEARING TEMPS ARE O.K.
106  60     WRITE(6,1000)
127  1000   FORMAT(80X,/,80X,/
108        1,' ALL DRIVE STAND AND GEAR BOX BEARING'
109        1,/,' TEMPERATURES ARE WITHIN LIMITS',10X,4(/,80X))
110  C  RESET CURSOR TO REPRINT TABLE
111  70     CALL HOME
112  C  ALLOW USER TO RESTART LOOP OR QUIT
```

```
*D MODCOMP SOURCE EDITOR     DATE  12/27/79  22:25:27        PAGE   3

113         WRITE(8,1100)
114  1100   FORMAT(' IS TEST COMPLETE?(Y OR N)')
115         READ(7,1200)IANS
116  1200   FORMAT(A1)
117         IF(IANS.EQ.'N')GO TO 10
118         STOP
119         END
120  C
121  C
122         SUBROUTINE HOME
123  C
124  C  ROUTINE WHICH MOVES CURSOR OF ISC TERMINAL TO HOME POSITION.
125  C
126         INLINE
127           LDI,2  UFT
128           REX,1
129           DFC  BUF,2
130           BRU  OUT
131  UFT       DFC  0,@ISC,#9030,0,0,0
132  BUF       DFC  #0800
133  OUT       NOP
134         FINI
135         RETURN
136         END
```


*U.S.Government Printing Office: 1980 — 657-084/697